SPRING 2024: PHYSICAL COMPUTING CLASS



LibreTexts™

Spring 2024: Physical Computing Class

This text is disseminated via the Open Education Resource (OER) LibreTexts Project (https://LibreTexts.org) and like the thousands of other texts available within this powerful platform, it is freely available for reading, printing, and "consuming."

The LibreTexts mission is to bring together students, faculty, and scholars in a collaborative effort to provide and accessible, and comprehensive platform that empowers our community to develop, curate, adapt, and adopt openly licensed resources and technologies; through these efforts we can reduce the financial burden born from traditional educational resource costs, ensuring education is more accessible for students and communities worldwide.

Most, but not all, pages in the library have licenses that may allow individuals to make changes, save, and print this book. Carefully consult the applicable license(s) before pursuing such effects. Instructors can adopt existing LibreTexts texts or Remix them to quickly build course-specific resources to meet the needs of their students. Unlike traditional textbooks, LibreTexts' web based origins allow powerful integration of advanced features and new technologies to support learning.



LibreTexts is the adaptable, user-friendly non-profit open education resource platform that educators trust for creating, customizing, and sharing accessible, interactive textbooks, adaptive homework, and ancillary materials. We collaborate with individuals and organizations to champion open education initiatives, support institutional publishing programs, drive curriculum development projects, and more.

The LibreTexts libraries are Powered by NICE CXone Expert and was supported by the Department of Education Open Textbook Pilot Project, the California Education Learning Lab, the UC Davis Office of the Provost, the UC Davis Library, the California State University Affordable Learning Solutions Program, and Merlot. This material is based upon work supported by the National Science Foundation under Grant No. 1246120, 1525057, and 1413739.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation nor the US Department of Education.

Have questions or comments? For information about adoptions or adaptions contact info@LibreTexts.org or visit our main website at https://LibreTexts.org.

This text was compiled on 06/08/2025

TABLE OF CONTENTS

Licensing

1: Preface

- 1.1: Introduction To Physical Computing and the Internet of Science Things
- 1.2: Introduction to your LibreTextbook

2: Assignments

3: Introduction and General Resources

- 3.1: Required Resources
- 3.2: Suggested Items
- 3.3: Hypothes.is
- 3.4: Programming and Flowcharts
- 3.5: Pseudocode and Flowchart Conventions
- 3.6: Functions
- 3.7: Computer Interfaces
- 3.8: Conda Package Management System
- 3.9: Thonny
- 3.10: Visual Studio Code
- 3.11: Git and GitHub
- 3.12: GitHub
- 3.13: GitHub Desktop
- 3.14: Online Resources
- 3.15: Git

4: Python Data Types

- 4.1: Introduction to data
- 4.2: Data Types
- 4.3: String (atomic)
- 4.4: Integers (atomic)
- 4.5: Float (atomic)
- 4.6: Boolean (atomic)
- 4.7: String (container)
- 4.8: List (container)
- 4.9: Dictionary (container)
- 4.10: Tuple (container)
- 4.11: 3.11 Set (container)

5: Data Operators

- 5.1: Introduction
- 5.2: Arithmetic Operators and Numbers
- 5.3: Arithmetic Operators and Strings
- 5.4: Assignment Operators
- 5.5: Comparison Operators
- 5.6: Logical Operators
- 5.7: Identity Operators

- 5.8: Membership Operators

6: Introduction to Linux

- 6.1: What is an Operating System?
- 6.2: Introduction to Linux
- 6.3: Git BASH
- 6.4: Linux Commands

7: Starting your Raspberry Pi

- 7.1: SD Card Formatter
- 7.2: Raspberry Pi Imager
- 7.3: Install VNC Viewer
- 7.4: ESD- The Silent Assassin
- 7.5: Installing RPi OS
- 7.6: Starting your Pi the First Time
- 7.7: Updating Your Pi
- 7.8: Registering your Pi with UALR IOT
- 7.9: VNC Viewer - Headless Mode
- 7.10: Adjusting Remote Monitor Settings
- 7.11: Pi Desktop
- 7.12: Pi Terminal
- 7.13: Keyboard Shortcuts
- 7.14: Google email account for class
- 7.15: Sending email without attachment
- 7.16: Sending email with attachment
- 7.17: Find IP Address
- 7.18: email IP address
- 7.19: Cron and Crontab
- 7.20: First Cron Job

8: Python Functions, Packages and Formatting

- 8.1: Functions
- 8.2: Introduction to Functions
- 8.3: Built-In Functions
- 8.4: Standard Library Functions
- 8.5: Third Party Library Functions
- 8.6: User-Defined Functions
- 8.7: User Created Package
- 8.8: Print Formatting Overview
- 8.9: Modulus Method
- 8.10: 3.format() method
- 8.11: f-string formatting
- 8.12: Subscripts and Superscripts

9: Anatomy of a Pi and Camera Activity

- 9.1: History of Pi
- 9.2: Monitoring Your Pi
- 9.3: Pi 4 Overview
- 9.4: SOC (System on Chip)
- 9.5: Random Access Memory (RAM)

- 9.6: Non-Volatile (Primary) Data Storage
- 9.7: LEDs
- 9.8: 40 Pin GPIO
- 9.9: UART (Universal Asnynchronous Receiver-Transmitter)
- 9.10: I2C (Inter-Integrated Circuit)
- 9.11: SPI (Serial Peripheral Interface)
- 9.12: Networking Overview
- 9.13: Radio Chip
- 9.14: Ethernet
- 9.15: USB C- Power Port
- 9.16: USB 2 and 3
- 9.17: AV Jack
- 9.18: Micro HDMI 2.0
- 9.19: CSI (Camera Serial Interface)
- 9.20: DSI (Display Serial Interface)
- 9.21: CSI Camera
- 9.22: Libcamera
- 9.23: 3-Python and Libcamera

10: Logic and Conditional Statements

- 10.1: Introduction
- 10.2: if Statement
- 10.3: if-else Statement
- 10.4: if-elif-else Statement
- 10.5: Compound if/elif Statements
- 10.6: Nested if Statements
- 10.7: Loops - An Introduction
- 10.8: for Loops
- 10.9: While Loops
- 10.10: Nested Loops
- 10.11: Loop Control Structures

11: Introduction to Electronics and First Circuit

- 11.1: Electricity and Units
- 11.2: Introduction to Electric Circuits
- 11.3: Circuit Diagrams
- 11.4: Diagrams.net Electric Circuit Diagrams
- 11.5: Fritzing
- 11.6: Conductors
- 11.7: Circuit Boards
- 11.8: Resistors
- 11.9: Diodes
- 11.10: LED (Light Emitting Diodes)
- 11.11: Multimeter
- 11.12: GPIO Output (LEDs) - First Circuit

12: GPIO Input (Buttons) - Second Circuit

- 12.1: Second Circuit

13: Serial Communication

- 13.1: Serial vs. Parallel
- 13.2: Serial and Morse Code
- 13.3: UART
- 13.4: I2C
- 13.5: SPI

14: APIs

- 14.1: Computer Interfaces
- 14.2: Creating a Google Service Account
- 14.3: Enabling Google Sheets and Obtaining a Key
- 14.4: PEP 668 and Debian Bookworm
- 14.5: Thonny-Virtual Environments
- 14.6: Upload Data to Google Sheet

15: DHT22 Temperature and Humidity Sensor

- 15.1: DHT22 Temperature and Humidity Sensor
- 15.2: DHT 22 - Streaming Temperature and Humidity Data
- 15.3: Writing to a CSV file (DHT22 data)
- 15.4: Reading a csv File

16: ADC and Sensors

- 16.1: SPI
- 16.2: ADC and MQ gas sensors

Index

Glossary

Detailed Licensing

Licensing

*A detailed breakdown of this resource's licensing can be found in **Back Matter/Detailed Licensing**.*

CHAPTER OVERVIEW

1: Preface

Topic hierarchy

1.1: Introduction To Physical Computing and the Internet of Science Things
1.2: Introduction to your LibreTextbook

1: Preface is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

1.1: Introduction To Physical Computing and the Internet of Science Things

Physical Computing

Physical computing is the application of computers to real world systems where computers can respond to the environment around them. There are essentially two components that are utilized in this interaction, sensors and actuators. Sensors collect information on the environment that is processed by interactive systems that consist of microcontrollers or computers/microcomputers, that in turn activate actuators that respond to the environmental stimuli. Figure 1.1.1 outlines this process, and although we may not realize it, physical computing has become ubiquitous in our life and is used to control everything from automobiles to kitchen and household appliances like ovens, washing machines and refrigerators. Most sensor signals are analog in nature, like the height of mercury in a thermometer, and the data needs to be digitized before it can be processed by a computer and acted upon. Central to physical computing are electronic systems that often utilize microcontrollers integrated into Printed Circuit Boards (PCBs). In this class you will create circuits on bread boards that are controlled by Raspberry Pi microcomputers with programs that you code, and you will also solder some simple printed circuit boards that you design, code and teste with a breadboard. No prior programming experience is required for this course.

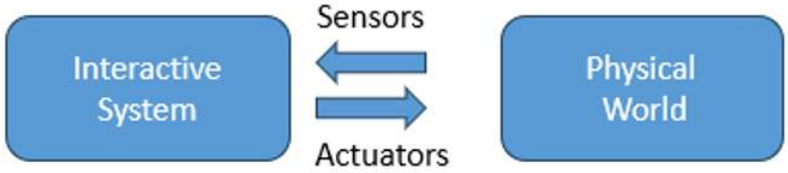


Figure 1.1.1: Outline of processes involved in physical computing. (Belford CC 0.0)

Internet of Science Things

Physical computing enables the Internet of Things (IoT) where different physical devices communicate with each other over the internet, and the Internet of Science Things (IoST) is the application of physical computing and IoT to scientific problems and processes. Figure 1.1.2 outlines some of the fundamental processes and structures of an IoST system. Note that the microcomputer/controllers on the edge of the cloud are doing physical computing and a single edge device can do both of the processes in figure 1.1.2, that is a single edge device can run both sensors and acutators.. The image shows two different devices just to make the processes easier to see, but both processes can be run by the same device.

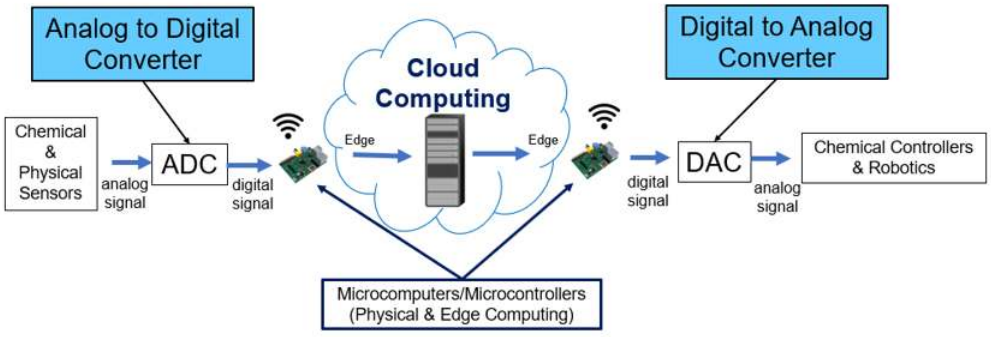


Figure 1.1.2: IoST diagram. Note, is a process diagram and although the images shows two microcomputer/controllers, they can be the same. In a real IoST system there will be many different microcomputer/controllers relaying different types of data from different physical locations (Belford CC-BY)

IoST systems like air quality monitoring will deploy multiple edge devices in different [geographical] locations that can stream data to central data servers that in turn can operate on the data from all of the streaming devices.

Cloud Computing

Cloud computing often involves Machine Learning and Artificial Intelligence data analytics based on data from multiple sensors, although in the broadest sense cloud computing is data analytics (and storage) occurring in remote data centers, in contrast to a

computer a user is actually connected to. The power of cloud computing in IoST is that it can process data from multiple devices in real time and utilize computaional resources located within a remote data center

Edge Computing

Edge computing is when an edge device performs some form of data processing. For example, a microcontroller with an embedded camera monitoring insects might be able to perform AI calculations to identify the type of insect and stream that data, in contrast to the very data intensive image files, and when that would be considered edge computing as the data processing occurs on the edge of the cloud. That is, the device that actually collects the data (physical computing) can also do local data processing (edge computing) and reduce the data streams bandwidth.

Paradigms of Science

Microsoft Research has a free online book, the Fourth Paradigm: Data Intensive Scientific Discovery, that postulates 4 paradigms of science. Before proceeding it is prudent to look at the curriculum content of this class in the context of these paradigms of science.

- **Empirical Science**-1st Paradigm (1,000s of years old): The oldest experimental based science employing qualitative and quantitative analysis of the physical world around us.
- **Theoretical Science-**2nd Paradigm (centuries old): The theoretical causality-based mathematically intensive reasoning that underpins our understanding of experimental data as outlined by the scientific method commonly taught in freshmen level science classes.
- **Computational Science**-3rd Paradigm (decades old): The extension of the first two paradigms to solve complex equations that humans alone could not compute. These may be "ab-initio" ("from the beginning") in that they are entirely based on theoretical relationships, or "semi-empirical", where they also employ empirical data.
- **e-Science**-4th Paradigm (emerging): This is the newest form of science resulting from data exploration enabled through the digitization of data. This can can lead to new discoveries and predictive analytics based on correlative-reasoning without elucidating the underlying causal relationships germane to the first three paradigms.

Physical computing is essentially applying computers to the first paradigm as systems interact with the environment, and as such this course can be an easy way to pick up programming skills. Through edge and cloud computing Physical Computing/IoST system can enable real-time fourth paradigm processes. Thus this course can provide the coding skills to function as a gateway to third and fourth paradigm scientific discovery for empirical science students.

1.1: Introduction To Physical Computing and the Internet of Science Things is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

1.2: Introduction to your LibreTextbook

Textbook as a Map

This is an interdisciplinary course and your textbook is actually a map through various pages of the IoSt Library, that LibreTexts allows through a process called "transcluding". When a page is transcluded all the figure and equation numbers update to the new book, and this allows material to be mixed in different ways. So a section on a topic like what is a string data type could be used in different locations in different books, although the content is the same. There is another reason, and this is that the content of this book deals with new and evolving technologies, and this are going to change. For example, to use your Raspberry Pi at UALR you will need to follow the login protocols for the UALR-IOT network, and these change from time to time. So by having multiple books use the same page for signing into the IOST network, you only have to change one page if the protocols change, and all the other books are updated.

At the beginning of each chapter there will be a table which is a map of the content in that chapter, and it will show where the original information is. This should assist you in navigating the IoST library as your proceed through the course. So you should consider the textbook to be more of a map than a book.

Social Tagging

Part of the goal of this course is to teach you the skills to find the information you need outside of the book, that is, on the world wide web. You may need a sensor that is not covered, and find code in a language we do not cover in this class, and it is important you figure how to handle that. So every week you are going to have an hypothes.is assignment where you go on the web, tag and annotate some open access source that pertains to your assignment. This not only helps you learn how to find material on the web, but also allows students in the class to share the material they have found. That is, web annotations systems are part of the modern semantic web and should be taught in universities.

1.2: Introduction to your LibreTextbook is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

2: Assignments

This page has links to the Google Docs containing your class assignments

Assignment	Due Date
Assignment 1: Flowcharts and Pseudocode	01/28/2024 (midnight)
Assignment 2: Molarity Calculator - First Program	02/04/2024
Assignment 3: Molarity Calculator - First Package	02/14/2024
Assignment 4: Pi Camera and programmatic email	02/19/2024
Assignment 5: LED and First Circuit	02/26/2024
Assignment 6: GPIO Input_LED_Music_Mixer	03/10/2024
Assignment 7: LED Morse Code Transmitter	
Assignment 8: API_Google_Sheets	3/10/2024
Assignment 9: DHT_22_Sensor_GSheet	3/25/2024
Assignment 10: DS18B20 temperature sensor	4/1/2024

2: Assignments is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

CHAPTER OVERVIEW

3: Introduction and General Resources

Topic hierarchy

3.1: Required Resources
3.2: Suggested Items
3.3: Hypothes.is
3.4: Programming and Flowcharts
3.5: Pseudocode and Flowchart Conventions
3.6: Functions
3.7: Computer Interfaces
3.8: Conda Package Management System
3.9: Thonny
3.10: Visual Studio Code
3.11: Git and GitHub
3.12: GitHub
3.13: GitHub Desktop
3.14: Online Resources
3.15: Git

Textbook Map for Chapter 2: Introduction and General Resources

Chapter 2 section	Original Source section
2.1: Required Resources	1.0.1: Required Resources
2.2: Suggested Items	1.0.2: Suggested Resources
2.3: Hypothes.is	1.1.1: Hypothes.is
2.4: Programming and Flowcharts	1.1.3: Programming and Flowcharts
2.5: Pseudocode and Flowchart Conventions	1.8.1: Pseudocode and Flowchart Conventions
2.6: Functions (and flow charts)	1.8.2: Functions
2.7: Computer Interfaces	1.5.1: Computer Interfaces
2.8: Conda Package Management System	1.5.6: Conda Package Management System
2.9: Thonny	1.7.1: Thonny
2.10: Visual Studio Code	1.7.2: Visual Studio Code
2.11: Git and GitHub (Intro)	1.7.5: Git and GitHub
2.12: GitHub	1.7.6: GitHub
2.13: GitHub Desktop	1.7.7: GitHub Desktop
2.14: Online Resources	1.0.3: Resources

3: Introduction and General Resources is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

3.1: Required Resources

This page is essentially a shoping list created for the Spring 2024 Physical Computing Class and the prices will change and there are many more options, so go out and find what you need. You will need a laptop, as you will run your Pi in headless node and use the laptop display and keyboard to operate the Pi. The first time you boot up though you will need a monitor, keyboard and mouse, and will need to run the Pi in Desktop mode. These components will not be considered as required, as we have them in the lab, but you may want them if you have issues at home, and some of the bundles below include these suggested items.

You can buy a kit like the Raspberry Pi 4 Ultimate Kit from Canakit that has most of the material you need, but there are cheaper and more customizable options. You must realize there are hundreds of vendors selling the devices and I am not endorsing one or the other, and that the prices and availability will change.

1. **Raspberry Pi 3b or 4 or 5** (the 5 is not released yet, but will probably be released before class starts) Adafruit currently has Pi 4s available and I would suggest either the 4GB ($55) as it is only $10 more than the 2GB, or the 8 GB ($75). Note, each version of the Pi has different chips and layouts. For example the Pi 3 has one hdmi while the pi 4 and 5 have two micro hdmi, and so once you pick your pi, you will need to get cables that are approriate to that board.

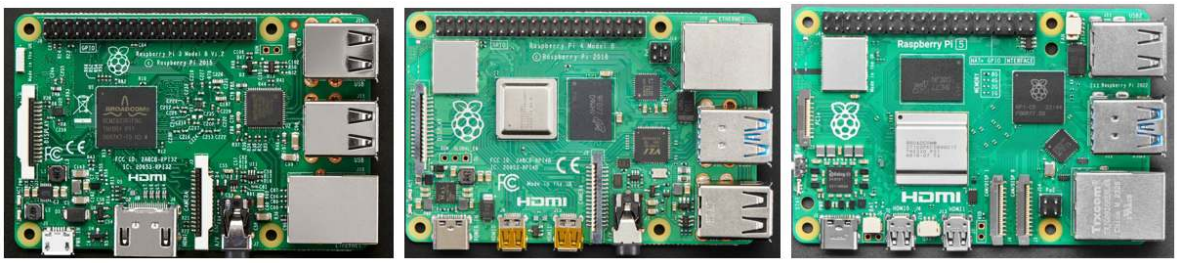


Figure 3.1.1: Raspberry Pi 3B (2015), 4 (2018) and 5 (about to be released). Images from Adafruit's website, the 4GB Pi 4 costs $55 (image credit adafruit.com)

2. **USB C 5V 3 amp power supply**. You may already have one for your cell phone that works, and you can get a power supply by itself like this official power supply from Adafruit. I suggest you get a kit that has other items you will need, like this Power supply with switch, case, microSD card, USB-MicroSD converter, microHDMI to HDMI cable and heat sinks for a Pi 4 that costs $19.89. The only "essential" items here are the power supply and the microSD card, but you will want the others, and they will be further described in the suggested materials page.



Figure 3.1.2: iUnker power supply with case and heat sinks(Amazon) that costs $19.89

3. **Cobbler, Bread Board and Ribbon Cable.** Once again, you can buy from multiple vendors, but this one from Amazon looks like a good deal and also provides additional items like wires, resistors, buttons, all of which I can lend you in class, but you would probably want for your own. I would also suggest you look into buying an extra cobbler and breadboard so you can have multiple circuit boards set up at the same time. But that is not neccessary.

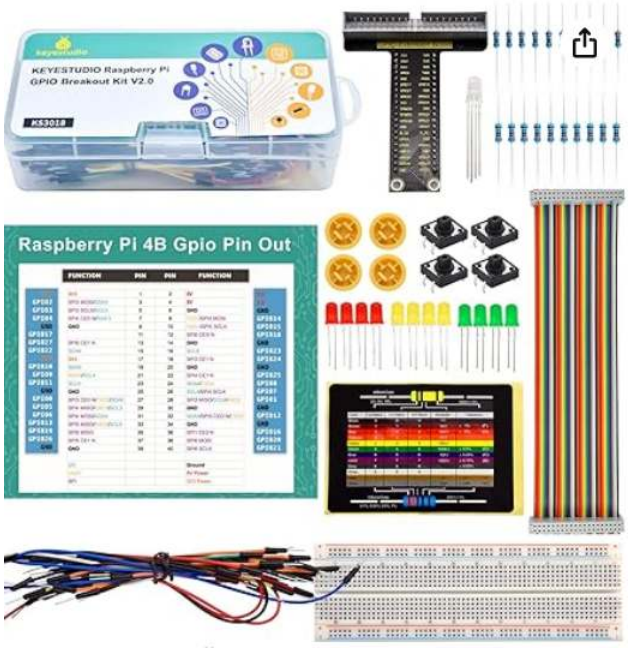


Figure 3.1.3: Breadboard kit from Amazon that costs $16.50. we will use everything in this kit and it even has a little box. (KeyStudio GPIO Breakout Kit)

4. **microSD card** (included in item 4 kit above),. You will want 32 GB or more. Further information on micro SD cards can be found in section 2.2.4 of the IOST library that we will be using in the course. That information will be updated and will not

apply to Pi 5s. Tom's hardware suggests this one. I actually suggest you buy multiple identical microSD cards, as it will then be easy to copy from one to the other. Sometimes we have had issues copying cards between different brands, and learning how to back up your work is very important. The prices are very reasonable on Amazon, with this $8.99 Silicon Power 32GB 3D NAND High Speed MicroSD Card getting high ranks from Ton's Hardware.

5. **CSI camera:** We will use the MIPI CSI (Mobile Industry Processor Interface Camera Serial Interface - CSI for short) port to run a camera, although we could use the USB (Universal Serial Bus), but the CSI is integrated into the chip and so uses less resources on the Pi. Section 2.6.1 of the IoST library covers the camera, which you should take a quick look at to understand what it is you need to buy. I do have a serious word of caution, if this is your first time connecting hardware to software (Physical Computing) you should get a simple camera and learn the basics, and then go and buy a fancy one with all the bells and whistles once you have the coding skills to make them work, so my advice is start out with the simplest one. You do not need to buy the camera before the class starts, but you will need to buy one.



Figure 3.1.4: I am not endorsing any camera, but I advise you to get a camera with a mount, as that will make it easier to deploy. (Image credit Amazon left, middle, right)

6. **DHT22 Temperature and Humidity sensor**. This is a basic sensor that I want each student to own, but once class starts we can buy a multi pack and they get real cheap. Note, there are three pin ones that have a built in circuit board, and you need to be careful with those, as they may be set up for different communication protocols than the one we will use. In our case we will use this to build our own circuit board (you will learn how to solder several circuit simple circuit boards this semester). So I want you to buy a 4 pin DHT22, as that is what our circuit board will be set up to use. Also, as a generic statement, if you are considering buying an electronic component and one option has information stamped on the device, and the other does not, I would suspect the one with the information is of better quality. But that is only a suspicion, sort of a dubious rule-of-thumb, and you really need to see if you can find a data sheet, as there is nothing to stop the manufacturer of a cheap knock-off from adding information to their device.



Figure 3.1.6: DHT22 sensors, you need to search and find what you find(Image source Amazon: left image, right image)

7.

3.1: Required Resources is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

3.2: Suggested Items

The following items are suggested items, and many are included in the "required" item bundles.

1. **Storage box:** I am honestly putting this at the top of the list as you will be buying supplies and will be building circuit boards, that you need to transport, and thought needs to be put into this. Does it need to fit into a back pack? Will a fishing style tackle box work? Or arts box, or small tool box? We will discuss this in class, but you want something you can transport a Pi hooked through a ribbon cable to a breadboard, and transport it in a manner where you do not damage the circuit.
2. **RPi case**: You will want a case to protect your Pi. Do not worry about the fan unless you are doing data intensive calculations, which is not what this class is about. But your case must have a slot for the GPIO ribbon
3. **Heat Sinks**: You will want these, and we can buy them in bulk as a class if needed.
4. **External monitor Cable**: If you want to run the RPi in desktop mode you need a cable to connect your monitor. The type of cable depends on your choice of RPi and your monitor. The RPi 3 uses hdmi and the RPi 4 and 5 use micro HDMI.
5. **USB Keyboard and Mouse:** If you want to run your Pi in desktop mode you need a mouse and keyboard. I suggest wireless. I also suggest full sized keys, but a reduced size board, that way you can carry it around.
6. **USB to MicroSD card adapter:** You will download the operating system to your laptop/desktop and then load it onto a microSD card, and this allows you to do that through a USB port.
7. **Tweezers:** If you get serious you will want a set of tweezers
8. **Magnifying glass:** Well, this depend on your eyesight, but I sure need them.
9.

3.2: Suggested Items is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

3.3: Hypothes.is

Hypothes.is & IoST/Physical Computing

The introduction below was created for General Chemistry and you should go through it if you are not familiar with Hypothes.is. In the value of Hypothes.is is not so much to to navigate the LibreTexts, but to navigate and develop a tag-taxonomy of open access documents across the web. That is, you will come upon new devices you have never seen before and in order to effectively use them you need to figure out how they work and how to code them. This will require you to go through multiple resources and by making annotations and tagging those annotations you can navigate back to the exact text in a resource that you need. If you are taking the 7000 level option of this course you will also need to write up a final report and should combine the functionality of Hypothes.is with the Zotero citation management system, for which you should make a library for the report.

Physical Computing Class Note

In the tutorial below we use hypothes.is to annotate the LibreTexts and we post them to a class group that only the class can see. In this course I want you to make the annotations public and tag them with the following tag **physical-computing.** You should also make a second tag that places context, as we can then search the public annotations for the above tag, and see all the other tags and content associated with it. This contextual link will show you how it works. Note, if you click the tag "physical-computing" within the hypothes.is interface it will provide a list of all pages with that term, and links to the annotations. This is a very important tool that you will be required to use this semester.

Introduction

The Hypothes.is web annotation that is integrated into Libretexts allows you to highlight text, take notes in an collapsable overlay and organize your notes through tags. You can even include notes on pages that are not on LibreTexts. Your instructor may use this resource, but it is also available to you as an individual student to use as you see fit. There are three levels of access your notes can have.

1. Public - anyone on the web can see it and you can go to the Hypothes.is website and search public annotations through their tags.
2. Group - if your instructor assigns a class group you should be provided with instructions that are germane to your class. Annotations to your class group can only be seen by members of your class, and they may be integrated into your grade. You can join more than one class group, and can even make your own groups (which would be useful for team projects). When you view annotations made by a group you must activiate that group in the hypothes.is overlay
3. Private you have access to these annotations.

First you need to go to https://web.hypothes.is/ and create an account. Please check your instructor if there are specific naming conventions. If your class uses a closed discussion group your instructor will provide a link to that group. You can join more than one group, in fact you create your own groups and chooser who enters them.

The following YouTube from a Spring 2020 class gives a quick demonstration on the use of the Hypothes.is Web Annotation service. Your class may or may not use all of the features in this video, but it goes over one way of integrating Hypothes.is into your LibreText (most classes will not use the tag aggregator, and we will discuss tagging at the end of this tutorial).



Making Annotations

To make an annotation you simply highlight the text you want to annotate, choose "Annotate" and write in the overlay. Note, the first time you open a page it will post to "Public" and you need to change that to your class group. Once you have started posting to the class group that will be the default option, but initially the default is public.

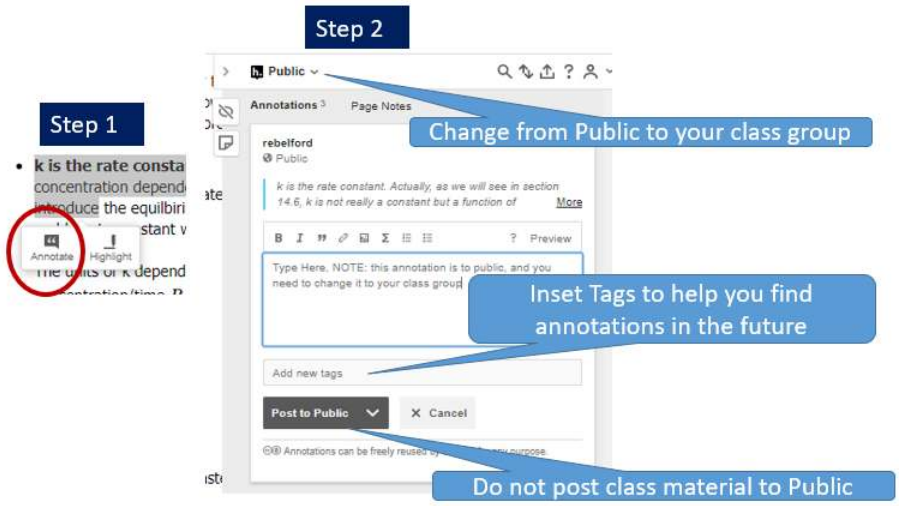


Figure 3.3.1: To annotate you simply highlight text and write in the overlay. (Copyright; Belford, cc.0)

If you tag your annotation you can easily find it by going to your homepage in hypothes.is and then filtering by tags. So you may want to make tags like "exam 1", or "nomenclature". If you then filter with your hypothes.is user name and exam 1, you see those items you tagged. If you just do exam 1, you filter for everything the class tagged exam 1. You can then click the contextual link by the annotation, and hypothes.is will open that page, navigate to the highlighted text and display the annotation. (Note: In LibreText the highlights are hidden by default, and you need to click the "eye" with the slash over it (figure 3.3.2

Using the Hypothes.is Overlay to read discussions

By default the hypothes.is overlay is closed and you do not see the annotations.

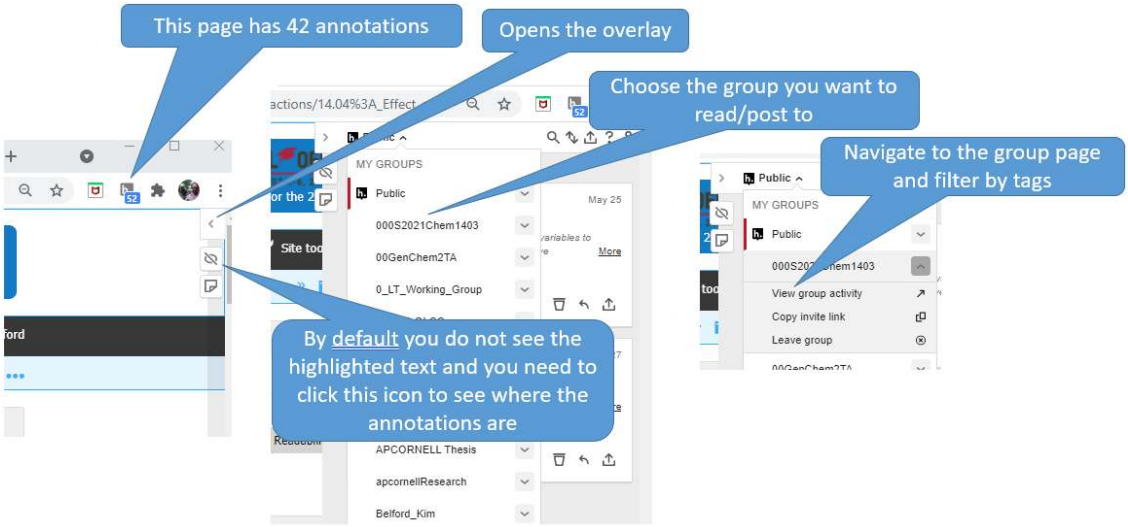


Figure 3.3.2: By default the overlay is closed and annotations are hidden. This image shows how to pick a group and view group activity (Copyright; Belford CC0)

Figure 3.3.2 shows how to pick a group and view group activity.

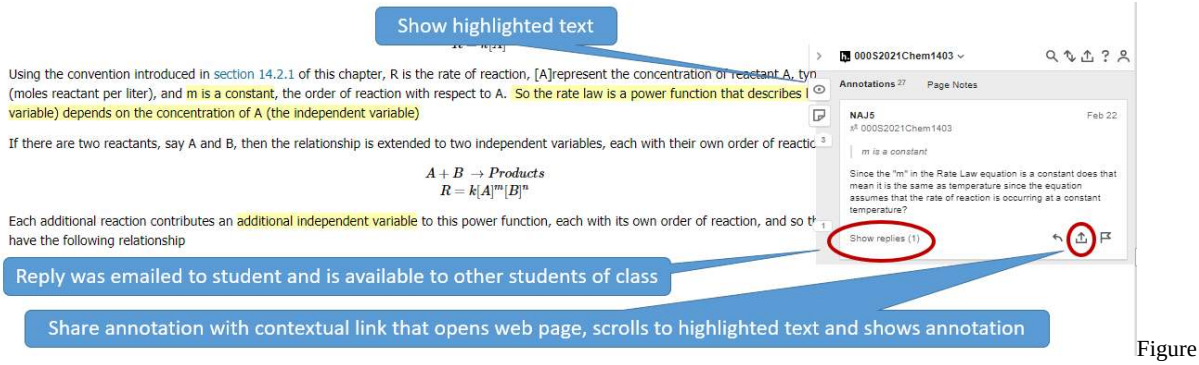


Figure 3.3.3: Annotation and highlights text being displayed. (Copyright; Belford CC0)

Using tags

Tags are a very powerful way of organizing your notes and Hypothes.is has different types of tags. Your class group is a tag, and when you click on your hypothes.is group activity (figure 3.3.2) you only see items posted to your class (note the first tag in the search filter of figure 3.3.4 is a group tag. If you then click your name you see those you have posted to that group. If you remove the group you see all the annotations you have posted anywhere. If you generate a tag based on a topic or exam, you can then easily find that information. Figure 3.3.4 shows a filter using two tags that resulting in four items in a student's "To Do list" for chapter 3 of a cheminformatics class

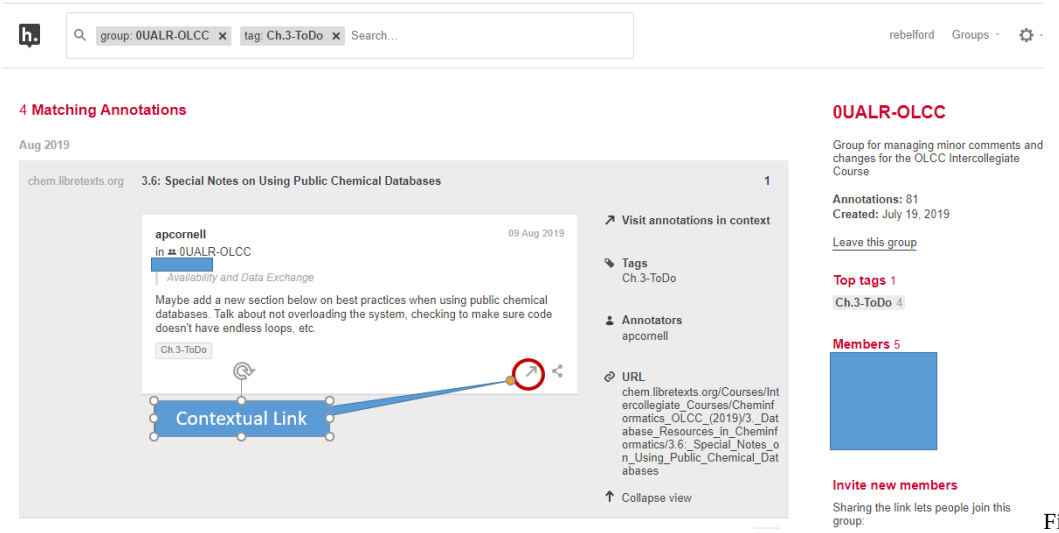


Figure 3.3.4: Hypothes.is search filtered by the group tag 0UALR-OLCC (which was a class group) and a student's tag "Ch.3-ToDo". (Copyright; Belford CC0)

If the student clicks the contextual link hypothes.is will open the webpage that was highlighted, scroll down to the highlighted section and show the annotation in the overlay. This allows students to organize their notes on LibreText and quickly find content. If you get the chrome plugin you can even annotate material outside of LibreText and connect it to your material and notes within LibreText, as long as the material does not require a login to access. It should also be noted that you can not highlight hidden text within LibreText (like the answers to exercises), and if you have a question on the answer to an exercise, you should highlight the question, not the answer.

This page titled 3.3: Hypothes.is is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

3.4: Programming and Flowcharts

Flow Charts

Flow charts are very valuable in writing code. In essence, each node of the flow chart describes a process that you will need to code. If your code gets complicated it can be very difficult to see where you are, but also, in object oriented coding, you can make an object or a function that does a task, and then call it when your flow chart needs it. So you can imagine in procedural coding you have one complicated script that crawls down the entire flowchat, while in object oriented coding you can call objects when the flow chart requires them.

In this class we will use the diagrams.net to make two types of flow charts; those for describing the flow process of a computer script (this section), and later, those involved with electric circuits. You can use shapes of common programs like PowerPoint or a flowchart app. We will install and use the draw.io app at https://app.diagrams.net/ and connect this to your course folder in Google Drive. Once you create a flow chart, you can use it as a template for future flow charts. IOST systems typically involve sensors that provide data about the environment and actuators that do a task. Many of these processes do not need to communicate with the internet and are called edge devices (they are on the edge of the cloud). Your first assignment will be to make a simple flow chart where a sensor provides data that is read, and based on the value, an action is taken.

https://app.diagrams.net/

There are two ways you can install the app. Either right click in your google drive, click on more apps, and search for "diagrams.net". Or you can go to diagrams.net site and follow the following instructions. Figure 3.4.1 is a flow chart that gives the steps for activating drawio and connecting it to your account. Please do not set this to your folder in Google Drive, but the subfolder called current work

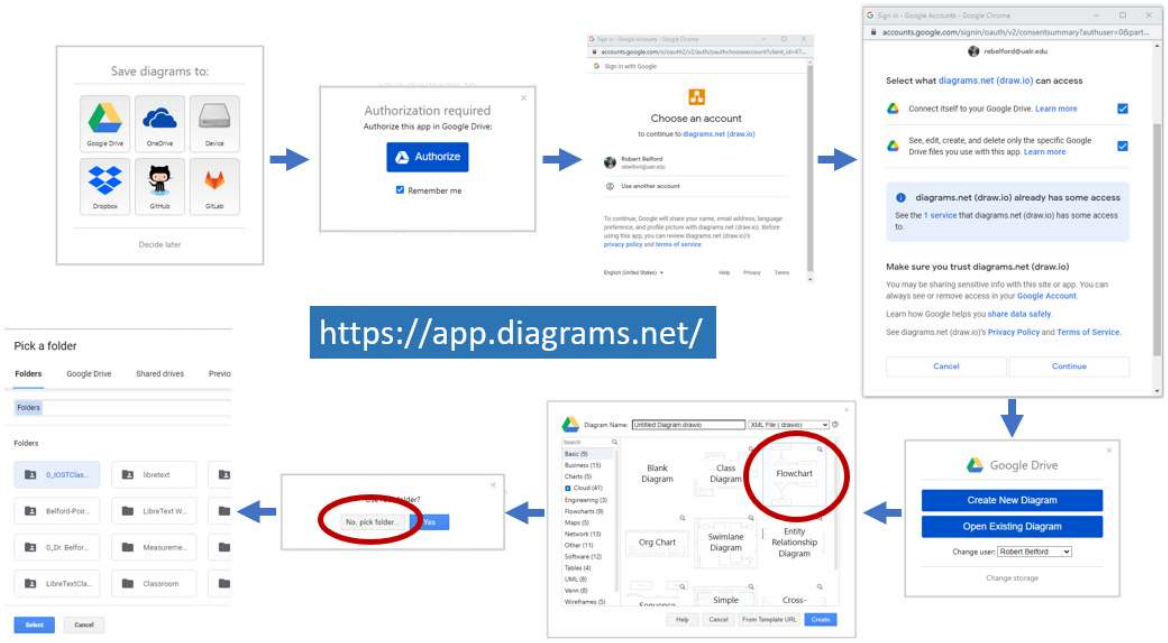


Figure 3.4.1: Steps for activating draw.io in your Google Drive. Be sure to use the **CURRENT WORK** folder of your class directory to keep flow charts you are working on. (Copyright; Bob Belford, CC 0.0)

To activate the draw.io app go to your current work folder in Google drive, right click, and open diagrams.net

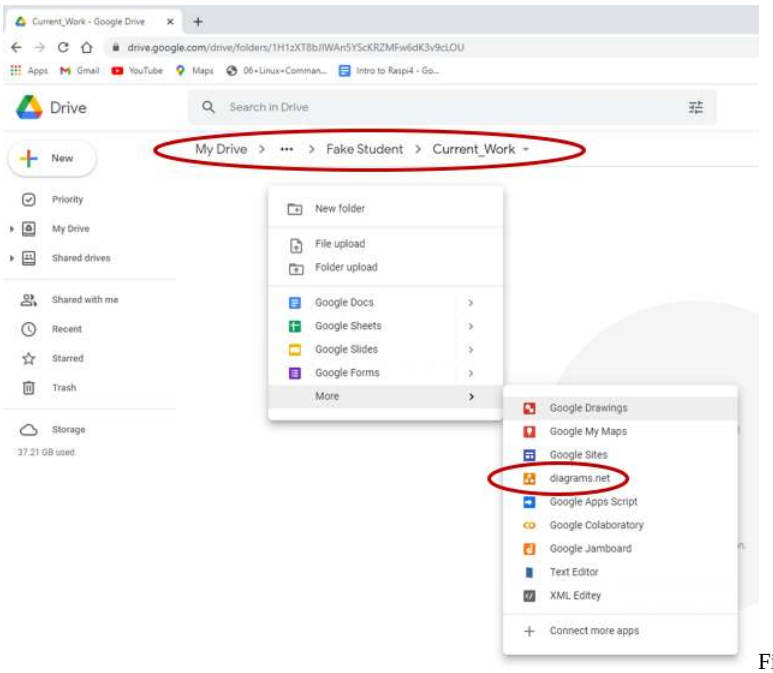


Figure 3.4.2: Right clicking in the "Current Work" folder and choosing more gives the option to open the draw.io app. (Copyright; Bob Belford, CC 0.0)

The following is a fast no-nonsense video that you should watch before proceeding.



Video 3.4.1: 3:22 YouTube, "Create flowcharts with diagram.net" by Bianca Ruiz (https://youtu.be/jbAJk1pBUYk).

This page titled 3.4: Programming and Flowcharts is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

3.5: Pseudocode and Flowchart Conventions

Pseudocode

Pseudocode is an informal way of representing the steps a computer program performs before it is coded into the syntax of any specific programming language. The first step in programming is to break a task into steps a computer can perform and a flow chart is a good way to do this. Now for very complicated programs a flowchart may not be tenable, and you may need to break some steps that are often repeated into a function that can be called to perform the repeating processes. So you may want to break your complex program into smaller ones (functions) that are sequentially called and iteratively develop these one at a time. The order by which a program performs the sequential steps is important and this order can be described with a flowchart.

FlowCharts

Flowcharts are a good way to present pseudocode as they break the task into individual steps and show the order in which they may need to be processed. In a flowchart certain symbols are used to represent the types of individual tasks a computer can perform and we will use the International Standards Organization (ISO) ISO 5807 standards developed by the American National Standards Institute (ANSI) in 1985 (Wikipedia:Flowcharts). Although you can draw these by hand or in a program like Excel or Word, you will be required to develop these using the diagrams.net/draw.io software integrated into your Google Drive as you will also be using that software later this semester when we create electric circuit diagrams, as described in section 1.1.2 Programming and Flowcharts of the web resources section of the general resources of the IOST library.ISO

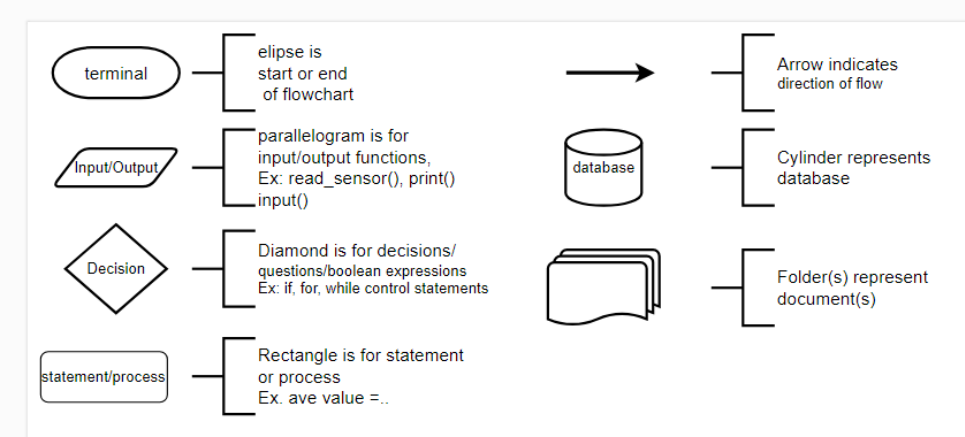


Figure 3.5.1: You can not edit this Read-Only Link but you can go to file/make-a-copy and then create a copy in your own drive and edit the copy, or you can just drag the image from draw.io and make your own flowchart. (Copyright; Belford cc by)

Flowchart with conditional statement

We want to make a small program that checks the input for the molar mass of a molecule and determines if it is valid. Since hydrogen is the smallest molecule, the molar mass can not be less than H_2, or 2 g/mol, and we want to reject any data that is smaller. The left flowchart uses the less than (<) comparison operator with an if-statement to create a condition statement that checks to see if the user input has an invalid molar mass (<than 2 g/mol) and print out a warning if it does. This takes advantage of the Boolean data type that is output from the if statement and has one of two values; true or false. The coding syntax in python for this statement is covered in section 5.9.2: if-statements. Note, a better way to do this would be to switch the comparison operator to greater than or equal (>=) and use an if-else condition, where it prints the molar mass if it is greater than 2 g/mol (True) and requests for a new molar mass that is greater than 2 g/mol if the original input was less than 2/g/mol. This example with python code is covered in the section 5.9.3: else-if statement.

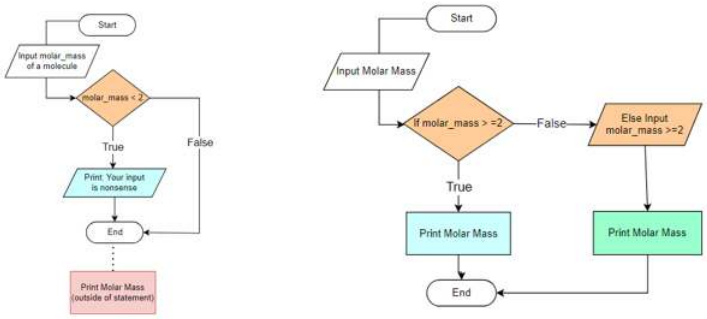


Figure 3.5.2: Flow chart to determine if an input value for molar mass is valid or not. Note, the convention is for the subsequent statement to be performed if the condition is true and skipped if it is FALSE. (Belford CC-BY)

The first assignment is based on the aquarium in figure 3.5.2 that is the home to some Mystery Snails. Your instructions will be posted to the class assignment page and you need to create a flowchart with diagrams.net that maintains the temperature in a safe range during the winter months by turning on a heater if the temperature is too low and turning it off if it is too high. You can identify the safe temperature range in this article on a Care Guide for Mystery Snails by the aquarium coop. In this assignment you will need to read the temperature, and then run an if-statement or two. In an if-statement you use a boolean True/False logic based on the condition you set. You do not need to worry about syntax in this activity as we are not writing code, just creating the flowchart, but you may want to search the web and if you find something you like, annotate it with Hypothes.is and tag it with a tag like [if-statement] (without the brackets). But one of the goals of this course is for you to develop the skills to find code and resources on the web, and you should start now.



Figure 3.5.2: Image of mystery snail in aquarium. (Copyright; Elena Waters, cc-by)

3.5: Pseudocode and Flowchart Conventions is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

3.6: Functions

Functions are blocks of code that perform specific tasks or sets of task. A typical function has an input (arguments or parameters) that it performs some kind of process on and then returns some kind of output. This can help organize code associated with a complex project and make it more readable. Processes that are repeated often are ideal processes to make a function out of. Book 5 section 4 is devoted to functions and we will see that there are standard built-in functions like the print() statement that load everytime you run python, standard library functions that come with python but that you must import to your code to use, third party library functions like those that could be associated with reading values of a specific sensor and you can even make your own user-defined functions.

Anatomy of a Function

- Input
  - Functions operate on data or perform some task that they operate on.
  - In python this information is enclosed in parenthes.is after the function name
  - Parameters are the types of variables a function operates on, it defines the kind of argument a function can accept
  - Arguments are the actual data values submitted to a function and upone which it operates on.
- Function Body
  - Instructional programmatic steps the function performs on the data.
  - This is the program that is run when the function is called.
- Output
  - This is the value or action that is returned when a function is called.

Then all you need to do is call the function and pass it any arguments it needs.

Python Example of User Defined Function

In the following code we create a function called "molar_mass_fct" and gave it the code from the flowchart of the if-else statement associated with the pseudocode of the last activity. All python functions must be followed by parenthesis. Do not worry if you do not understand the syntax as we will get to that later this semester. Right now you need to know what a function is and what it does. But hit run, and try some values, and note that there are only two lines of code outside of the function, the input molar mass line, and the function call line. Also note that in python a comment is indicated with a # sign, so the lines starting with a # are not run as code

```
def molar_mass_fct(molar_mass):
    if molar_mass >= 2.0:
        print(f"The molar mass you entered is: {molar_mass}g/mol")
    else:
        molar_mass=float(input("Enter a molar mass greater than or equal to diatomic I
        print(f"The molar mass you entered is: {molar_mass}g/mol")

#Input molar mass
molar_mass=float(input("Enter the molar mass of a molecule in units of g\mol "))
#call function
molar_mass_fct(molar_mass)
```

run | restart | restart & run all

```
Hello world!
```

The function in the above code was defined by this code block. It is given the name "molar_mass_fct" and operates on the "molar_mass" variable

```
def molar_mass_fct(molar_mass):
    if molar_mass >= 2.0:
        print(f"The molar mass you entered is: {molar_mass}g/mol")
    else:
        molar_mass=float(input("Enter a molar mass greater than or equal to diatomic
        print(f"The molar mass you entered is: {molar_mass}g/mol")
```

In the next line of code we input a value (the argument) that we assign to the variable molar mass. Note we used two built-in functions in this line of code. The first was the input() function, but that returns a string value (letter) and we need to convert it to a floating number so we can sue the >= comparison operator within the function body, and so we placed that function inside of the float function. the result is we have now assigned the variable molar_mass the value of the floating number input by the user

```
molar_mass=float(input("Enter the molar mass of a molecule in units of g\mol "))
```

In the last step we call the function and pass it the argument (value input by the user) and it then either says the value is too small to be a molar mass, or it outputs the value.

```
molar_mass_fct(molar_mass)
```

Flowchart Simplification

In the following diagram we show how making a function to check the molar mass is larger than two simplyfies the flow chart to just an input statement and a function call statement. The material on the top part of the image in red dots is the function, and this shows how a function can simplfy the code and make it more readable.

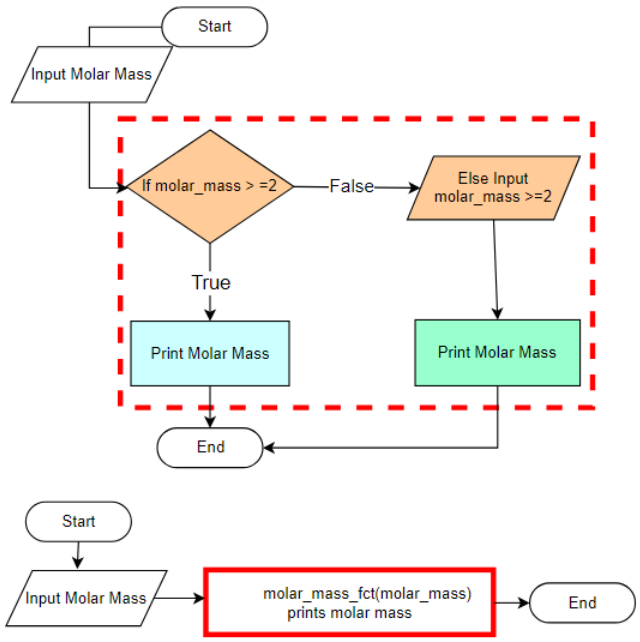


Figure \(\PageIndex{1}\): Two flowcharts doing the same thing. The top one does not use a function, while the bottom one calls a function that does the operations in the dotted box. Once the function has been made, it can be used by multiple programs, and this is why there are libraries (or packages) in python containing functions and other features. (Belford CC-BY)

3.6: Functions is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

3.7: Computer Interfaces

You could say a computer interfaces represent how a computer interacts and communicates with an external entity. That entity may be a human, or some other electronic machine. In this course we are using two types of Human Computer Interfaces (HCI); the Command-Line Interface (CLI) and the Graphical User Interface (GUI). Other human to computer Interfaces include Touchscreen Interface, Voice User Interface (VUI), Virtual Reality Interface (VRI) and Augmented Reality Interface (ARI). An interface between two machines is an Application Program Interface (API), which involves software communicating between two different devices, and these are extensively used in the Internet of Things.

Common HCI

The only Human Computer Interface we are using in this class are the CLI and GUI, as those are the two interfaces you are using in this class to communicate with the Raspberry Pi.

CLI

The terminal where you type Linux commands is the CLI of the Raspberry Pi. This allows you to interact with the operating system and is keyboard driven.

GUI

The desktop of the normal Raspberry Pi OS installation if a GUI that you can navigate by mouse or other pointer device. This allows you to use Interactive Development Environments (IDE) like Thonny, which can assist with programming and troubleshooting. Most people today communicate with computers through GUIs.

API

The Application Program Interface (API) is critical to the internet of things and we will quickly review them as few students will be familiar with APIs. Many of these you use without even thinking about it, but they are fundamental to the processes being run by a computer and it is important you recognize that.

Hardware APIs

These allow software to interact with input and output data and involve ports that you can connect devices to. This includes printers, speakers, microphones, sensors and actuators. These require hardware interfaces or ports such as USB, hdmi, the Raspberry Pi 40 pin GPIO pins or even Bluetooth wireless communication technology to enable communication with the device. The hardware API acts as an access point to allow programmers to use communication protocols to communicate with devices .

Web APIs

Web APIs allow you to communicate between computers across the web. For example, a web page has a URL (Universal resource Locator) which is the unique address the server uses to provide the page you are reading. A URI (Universal Resource Identifier) is like a URL for a specific piece of data accessible over the web.

When we learned how to concatenate strings byu communicating with PubChem to get a molar mass of a molecule we were using the PUG (Power User Gateway) REST (Representational State Transfer) API provided by NIH NCBI to programmatically access chemical data within the PubChem database.

Operating System APIs

Linux is the operating system of a Raspberry Pi and it serves as the interface between you and your computer's hardware. This allows you to manage resources like memory, processor time and input/output operations. This is not really a programming language like Python, although you can use BASH (Bourne-Again SHell) scripts to automate repetitive tasks in Linux systems

Database APIs

These are application program interfaces that allow computers to interact with databases

Library APIs

The import statement in Python is an example of the use of a Library API, where you are taking code in one program and using it in another. That is, two programs are communicating with each other. In Python Libraries are collections of modules that contain

variables, functions and classes that can easily be used by other programs through a library API.

This page titled 3.7: Computer Interfaces is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

3.8: Conda Package Management System

Conda

Conda is a cross platform package management system. It allows you to install, update and manage software packages and dependencies. Conda makes it easy to update Python and ohter packages. There are two ways to run Conda, through the command line, or through the Anaconda-Navigator GUI. There are three advantages to installing Conda and you should install a package management system like Conda if you intend to tackle data science projects..

1. Package Management: Conda allows you to install, update and manage packages for multiple programming languages like Python. Some packages you may be interested in are Scikit learn (Machine Learning package), Matplotlib....
2. Environment Management: Allows you to create isolated environements with specific packages and dependencies (some package may require different versions of Python).
3. Cross-Platform Support:

Resources

- Conda Cheat Sheet

Distributions

There are two distributions of Conda; Anaconda and Miniconda. Anaconda comes with a large number of preinstalled packages and the Anaconda Navigator, which is a GUI that allows you to manage your packages. Miniconda is a bare bones installation of Conda that only includes a few essential packages. You should install Anaconda if you are interested in Data Science and have lots of free space on your computer. We will provide a tutorial on using Miniconda and only install packages we need.

Minconda

Miniconda installs Conda, Python and a few other useful packages like pip, and zlib. To install additional packages you will need to use the conda install command. In this activity we will install miniconda and show you how to make the Python version in miniconda the default interpreter in Thonny. It will then be easy to update Python by using Conda.

Go to the following URL and choose the Miniconda Installer Links for your operating system and download the executable file.

https://docs.conda.io/en/latest/miniconda.html

The following images are screen shots for installing Miniconda that date to 2022, and may have changed since them.

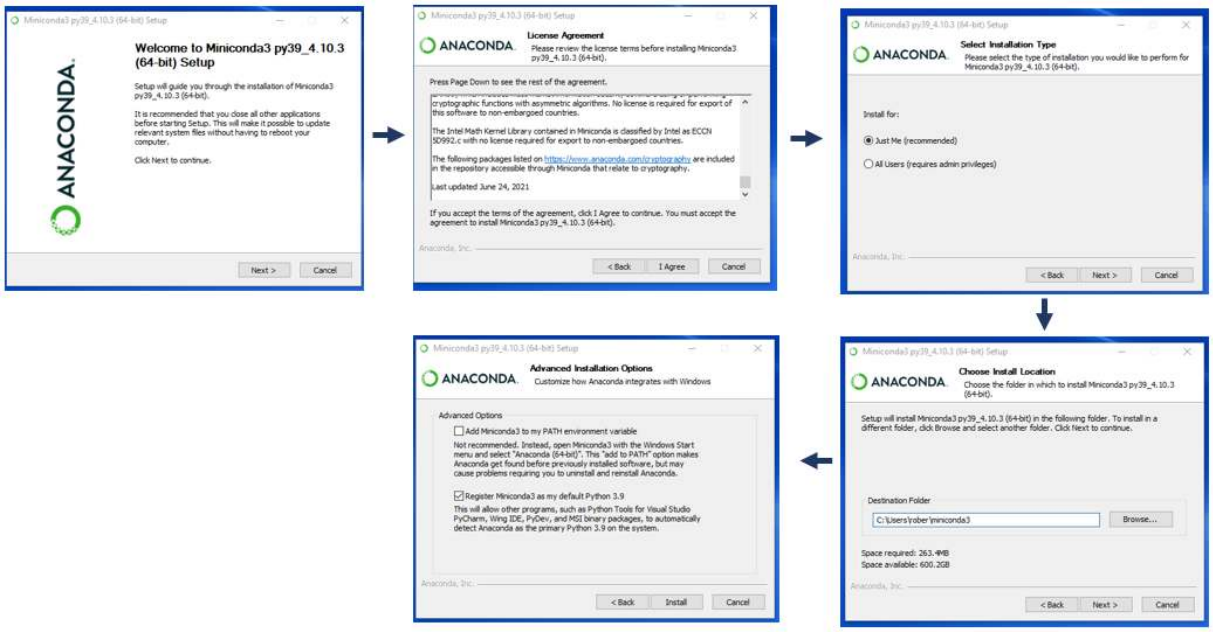


Figure 3.8.8:Screen captures of steps involved with installing miniconda. (Copyright; Bob Belford, CC 0.0)

Once you install Miniconda you want to open the Anaconda prompt (miniconda 3) from your Windows start menu, which can be done by typing in the search: Anaconda

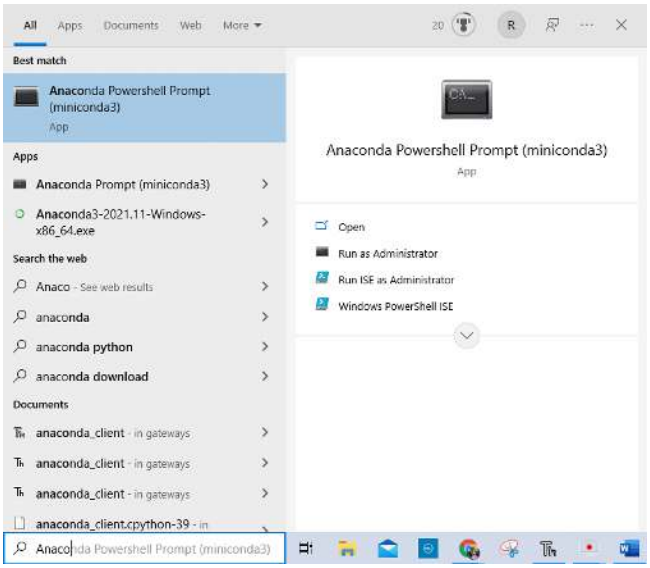


Figure 3.8.9: Opening the Anaconda command prompt for miniconda in Windows 10. (Copyright; Belford CCC 0.0)

If you are using a Windows machine the command prompts are different than Linuxges you want to type the following into the command prompt. You first need to navigate to the correct folder. In the command prompt (which is a terminal shell) type ls to see your folders.

Note <ctrl> C bring back the command prompt if you get stuck

```
ls
#you may need to use if on windows
dir
```

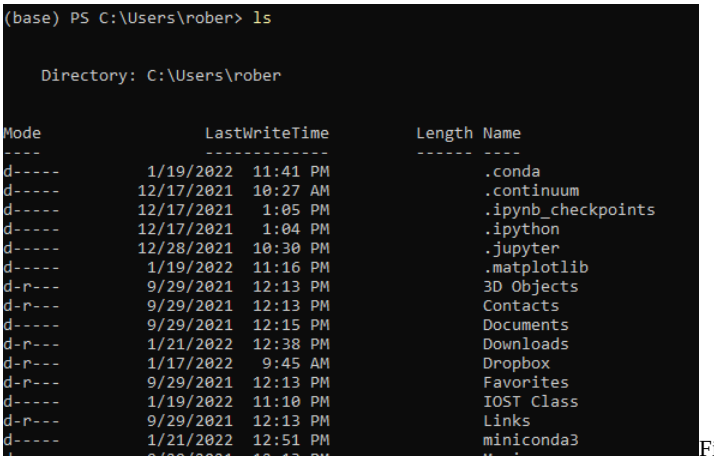


Figure 3.8.10: Viewing directory to locate where conda is installed. (Copyright; author via source)

Now move to that directory (Change Directory)

```
cd .\miniconda3\
```

When you list the files you will see where you installed python

You will also want to update python

```
conda update python
```

```
(base) PS C:\Users\rober\miniconda3> conda update python
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\rober\miniconda3

  added / updated specs:
    - python


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    python-3.9.7               |       h6244533_1        16.5 MB
    ------------------------------------------------------------
                                           Total:        16.5 MB

The following packages will be UPDATED:

  python                                3.9.5-h6244533_3 --> 3.9.7-h6244533_1


Proceed ([y]/n)? y


Downloading and Extracting Packages
python-3.9.7         | 16.5 MB    | ############################################################################ | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
(base) PS C:\Users\rober\miniconda3>
```

Figure 3.8.11: Screen shot showing how to update python with miniconda. (Copyright; Belford CC 0.0)

Exit the anaconda shell

```
exit
```

Changing Default Thonny Python Interpreter

Now we want to link our Thonny to the new install of python. Open Thonny and go to tools/options/interpreter (figure 3.8.12) and choose the "Alternative Python 3 interpreter or virtual environment". Initially the python executable will direct to where you installed Thonny and you need to direct it to the new path, which you found above. Or you can click the three dots tothe right of the current path and navigate to where the python application file is located and open it.

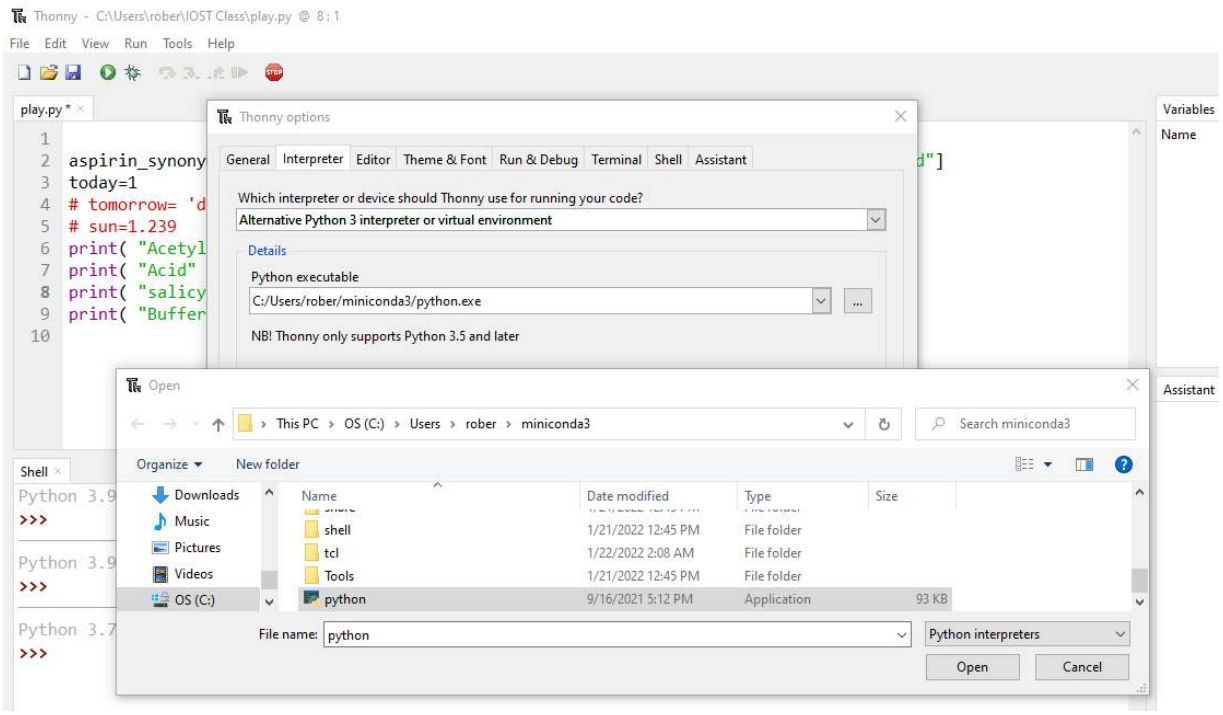


Figure 3.8.12: Copy and Paste Caption here. (Copyright; author via source)

If you now look at the shell in Thonny you can see the new updated Python program you are running



Figure 3.8.13: the version and path of the Python Interpreter Thonny is using. (Copyright; Belford CC 0.0)

Anaconda

Anaconda is a data science platform which allows you to find and install over 7,500 data science and machine learning package and progarams like Python, R, Jupyter, Scikit learn, matplotlib and R, but also includes the conda package manager that helps deal with conflicts across software. There are two installation options, Anaconda and Miniconda. If you are planning to get involved with Data Sciences or have a high quality computer you will probably want to install the full version of Anaconda, but if this class is your only endeavor than you will probably want to install the Miniconda version. Note, the full version comes with the Anaconda Navigator that makes it very easy to launch and maintain packages, where the Miniconda requires you to use to launch programs through the conda prompt. What we are going to do in this class is connect the Thonny IDE to the Python interpreter installed with either Anaconda or Miniconda and use Thonny to launch Python. We are choosing Thonny because it comes installed with the Raspberry Pi. We will also be using MS Visual Studio Code as it integrates with Git and GitHub versioning software

- Anaconda
  - https://www.anaconda.com/
    - Well suited for scientific programmings. Comes with ipython and Spyder, and the Conda package management system.
    - Has 1500 scientific packages and uses 3GB of disk space

Anaconda Navigator

In your user directory type

```
conda install anaconda-navigator
```

3.8: Conda Package Management System is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

3.9: Thonny

The RPi (Raspberry Pi) comes with the Thonny IDE (Interactive Development Environment) installed on it and so it is simplest to install the Thonny IDE on your laptop. As of December 2022 the latest version of Thonny is Thonny 4.0.1, which will also instal Python 3.10.6, and so you do not need to install python. If you are planning on doing machine learning or advanced data science applications you may want to install the Anaconda Package, which helps you update multiple packages without causing conflicts. But unless you are an advanced user, Thonny should be fine.

Download Thonny at https://thonny.org/

Thonny is now the default IDE for *.py and you can open it from either the start menu or by clicking on a *.py file. If you want, you can also pin it to the taskbar.

UI Modes

Thonny has three UI (User Interface) modes, Simple, Regular and Expert. We will typically use the Simple or Regular mode. On your PC you will probably want Regular as it has access to the file menu (fig. 3.9.2), but if you have limited screen space, like are on a Pi with a low resolution monitor, you may wish to use simple mode to maximize the screen's realestate.

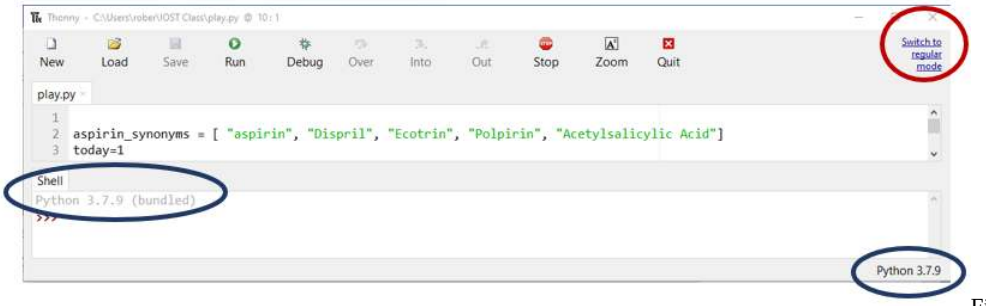


Figure 3.9.1: Thonny UI in SIMPLE MODE, note there is no menu bar (see image below) and this installation is running python 3.7.9. (Copyright; Belford, CC 0.0)

To switch to REGULAR mode you can click the text in the circle of the upper right in figure 3.9.1. To switch from regular mode to simplemode go to tools/options/general/UI mode and pick the option you want (figure 3.9.2). Also, note that the blue oblong circle on the shell of figure 3.9.1 shows that Thonny is running python 3.7.9, which was co-installed with Thonny, but in 3.9.2 we can see that Python 3.9.7 is being run. That is, these images were taken when we were using different versions of Python, and in the next section you will install Python

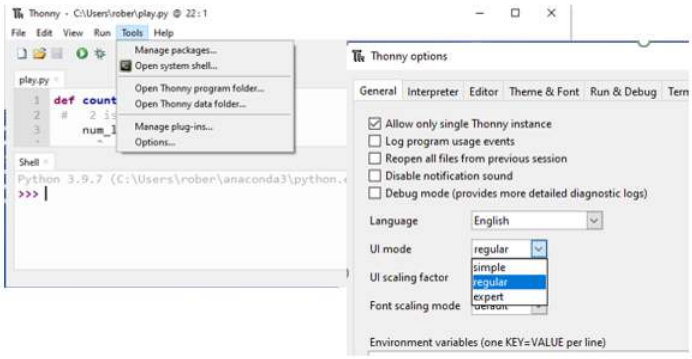


Figure 3.9.2: Thonny UI in regular mode. (Copyright; Belford, CC 0.0)

To switch from regular mode to simple mode choose Tools/General/UI mode/regular (Figure 3.9.2). We will typically use the Regular mode.

Menu Items

When in Regular mode you have access to several menu items and it is prudent to go over these before starting. Many of these like open and save will be familiar to you while others will be unfamiliar. It is always a good idea to check out menu items on any software package before you start using it.

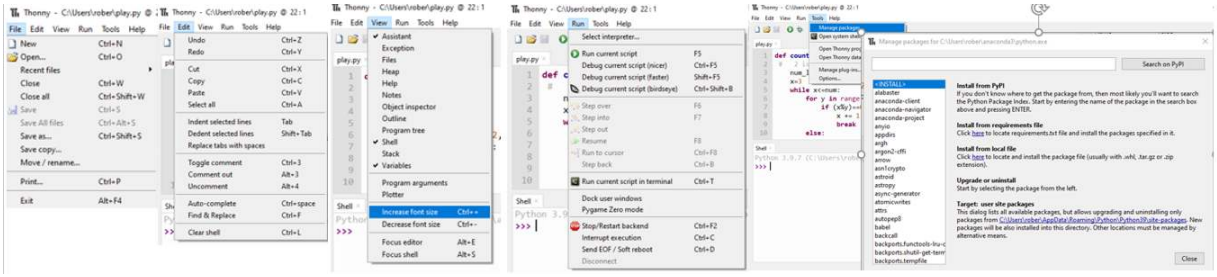


Figure 3.9.3: Different menu options, go ahead and exlore (Copyright; Belford cc0.0)

We will take a quick dive on several of the menu item features that your are probably not familiar with.

Edit

In some IDEs you need to save a file before you run it. In Thonny you do not, and the moment you hit "RUN" it saves the file and then runs it. This means it overwrites the code of the original file. So, a common error is to change some code and run it to see if it works, but you then lose the original code. A good practice is to comment out code with the # symbol. This means that text will not be operated on. In the Edit menu you have the ability to comment or uncomment large blocks.

play.py *

```
1
2  aspirin_synonyms = [ "aspirin", "Dispril", "Ecotrin", "Polpirin", "Acetyls
3  today=1
4  # tomorrow= 'day'
5  # sun=1.239
6  # print( "Acetylsalicylic Acid" in aspirin_synonyms)
7  # print( "Acid" in aspirin_synonyms)
8  # print( "salicylic Acid" in aspirin_synonyms)
9  print( "Bufferin" in aspirin_synonyms)
10
```

Figure 3.9.4: By clicking comment out in the Edit menu we were able to prevent lines 4-8 from being run. We could then try some other script, and if it did not work, we could simple uncomment these in the Edit menu, and not lose the code. It is very important to realize that in Thonny saves code the moment you run it, so if you edit something and run it, you lose the original code. (Copyright; Belford CC 0.0)

Note, the color coding above is a result of the option chosen in the tools/options/theme-fonts tab.

View

The View option allows you to select the interfaces you wish to work with. The code editor is always present and you probably want to run the Shell, which emulates a command line interface. In figure (3.9.3) the Shell, Variables and Assistant options have been opened. We will go over the utility of these features as the course proceeds, but these are probably the most common ones you will wish to work with.

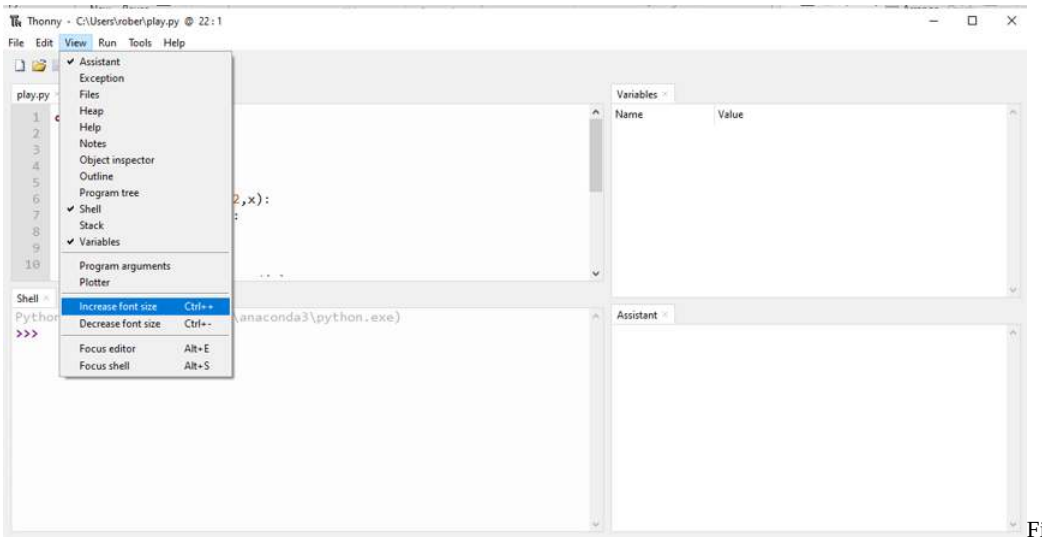


Figure 3.9.3: View menu options for Thonny UI. (Copyright; Belford, CC 0.0)

You will also note that the View menu provides an option to increase or decrease font size with keyboard shortcuts of Ctrl + + and Ctrl + - (click the crtl followed by either plus or minus). This is often very important when you are running a Pi in headless mode using the VNC viewer.

Run

The run tab allows you to execute python code in the editor and interact with it through the shell. You will often want to use the <F5> shortcut key. You can also run the program in "debugger mode" that allows you to walk through the code one step at a time. There are a lot of YouTube videos on the debugger and the following one goes over it (we are starting where the debugger part of the script is, but you may want to watch the entire video.



Tools

Tools is an important menu item, in fact we have already looked at in in figure 3.9.2 when we switched the UI (User Interface) from simple to regular mode. Let's quickly look at some of the other features of the Tools, starting with the manage packages and manage plugins. A Python package is a way to add new features to your program, like say math or time functions, and is essentially a directory of modules that you typically download from a service like the PyPI Python Package Index, that as of 1/22/2022 contains 351,947 projects. A package has a init_.py file sets how the packages are imported, while a directory could be a collection of python modules, but without the init_.py file, they will not be integrated into the interpreter.

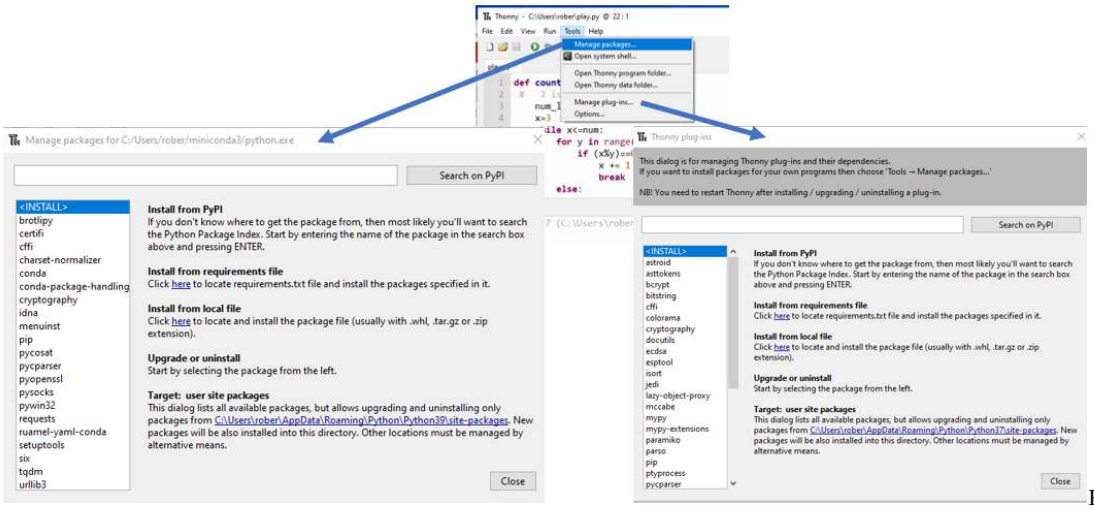


Figure 3.9.6: If you are running the default python interpeter that comes with Thonny you can manage packages from this menu. If you are using a different interpreter, say the one from Anaconda (see below), you would not use this option. The manage packages allows you to install new packages while the manage plug-ins is used for managing existing plug-ins. . (Copyright; Bob Belford, CC 0.0)

We have already looked at the options submenu when we changed the UI (Figure 3.9.2.

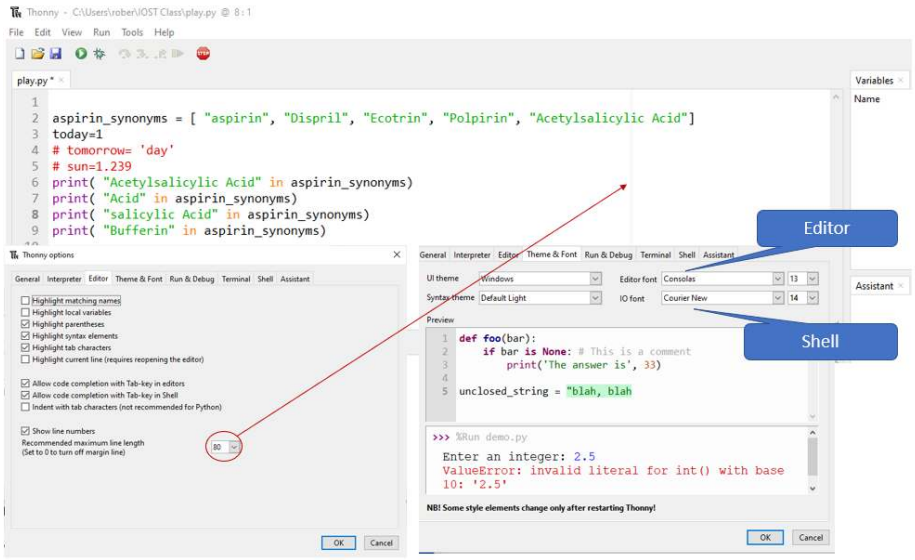


Figure 3.9.7: Editor and "Theme & Font" submenu items of the "options" item of tools overlaid on the IDE. (Copyright; Belford CC 0.0)

PEP 8, the Index of Python Enhancement Proposal style guide suggests a maximum line length of 79 characters and you can set a verticle line in your editor to warn you when your string of code is getting to long (figure 3.9.7).

Editor Options

Under the regular UI you can go to **Tools/Options/Editor** and do things like insert/remove line numbers, set the reccomended line length (standard python libraries limit line length to 79 characters with comments being recommended to be limited to 72)

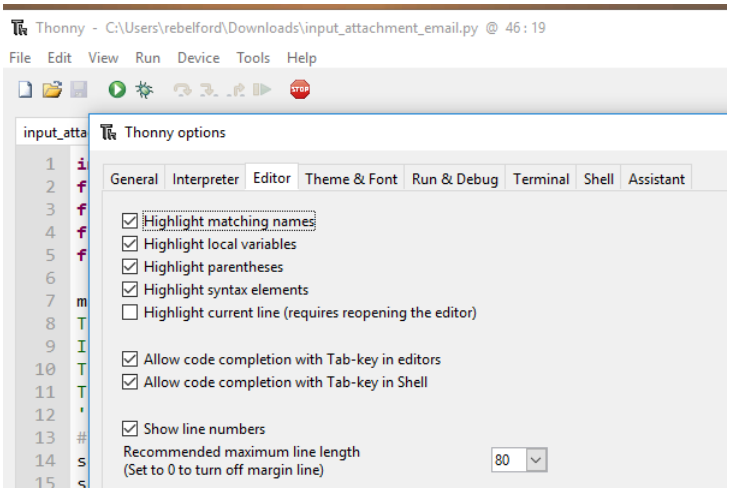


Figure 3.9.3: Editor options under Tools/Options/Editor of normal UI. (Belford CC 0.0)

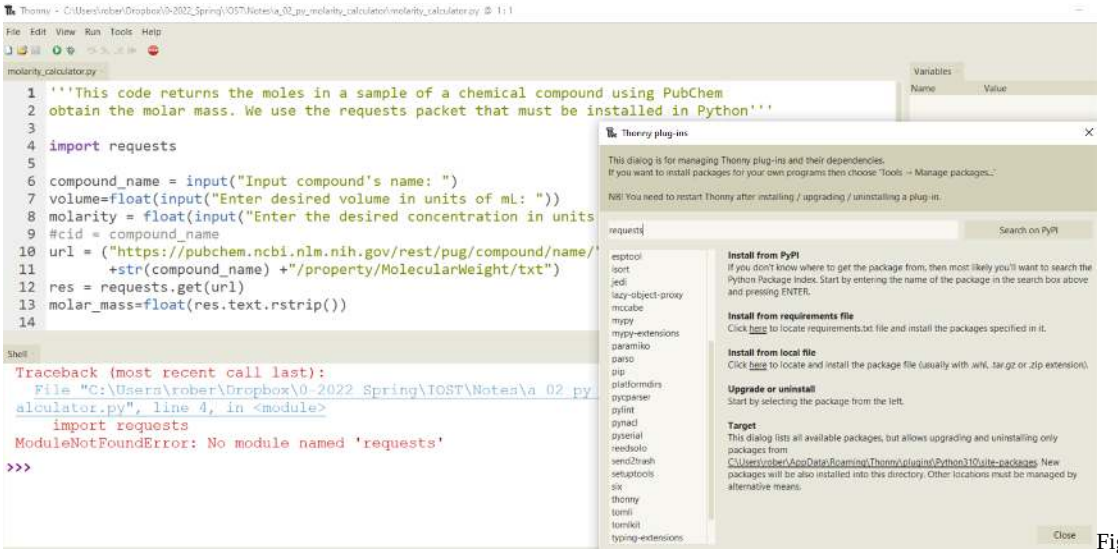


Figure 3.9.1: Installing modules on Thonny. (Belford CC 0.0)

3.9: Thonny is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

3.10: Visual Studio Code

Visual Studio Code is also known as VS Code or VSC is a powerful light weight IDE developed by Microsoft that has Git and GitHub integration. The download link is https://code.visualstudio.com/ and we will be developing this tutorial as the class proceeds. This is also a useful editor for HTML web pages and it even allows you to watch the webpage change as you edit it. Once again, you should surf the web for tutorials and if you find any that you really like you should annotate them and tag them with the tags physical-computing and vs-code. Note, if you want to work with Git you will need to install it (https://git-scm.com/downloads).

Getting Started

https://code.visualstudio.com/docs is the place to get started. Figure 3.10.1 shows the Quick Start Guide for using python. You should familiarize yourself with this resource if you plan to use VSCode. You need to do three things to use VSC with Python:

1. Instal VS Code
2. Instal Python
3. Instal a Python extension from the VS Code Market Place

Note, if you already have Python installed the VS code will connect to the most recent version, and if you try to open a Python file it will direct you to get the extension.

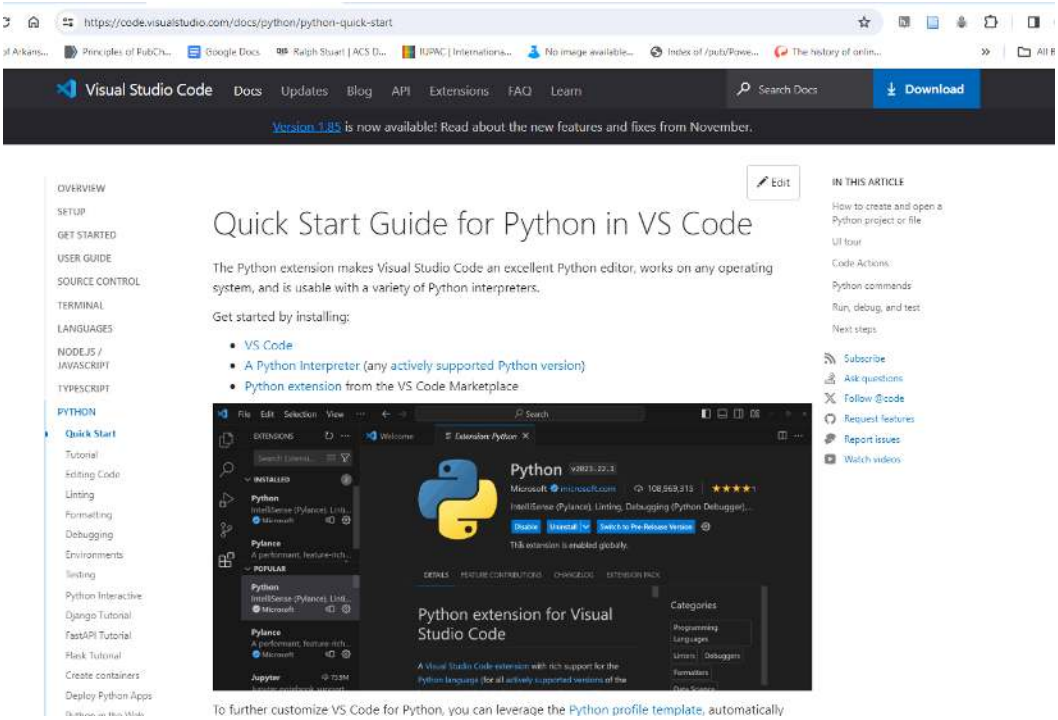


Figure 3.10.1: Screenshot (1/22/2024) of VS Code Docs Quick Start page. (Belford CC 0.0)

Live Preview (html)

go to extensions (bottom icon on left panel) and search for "live preview" and then instal

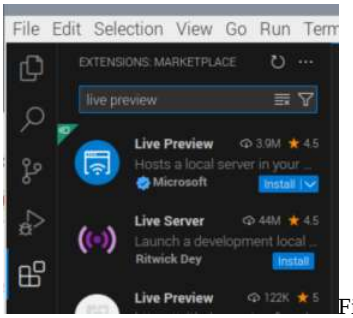


Figure 3.10.1: Copy and Paste Caption here. (Copyright; author via source)

Installing VSC on Raspberry Pi 4

VS Code is recommended software and thus simple to install. Click the Pi icon, choose preference/Recommended Software. After the list is compiled scroll down and click Visual Studio Code and apply.

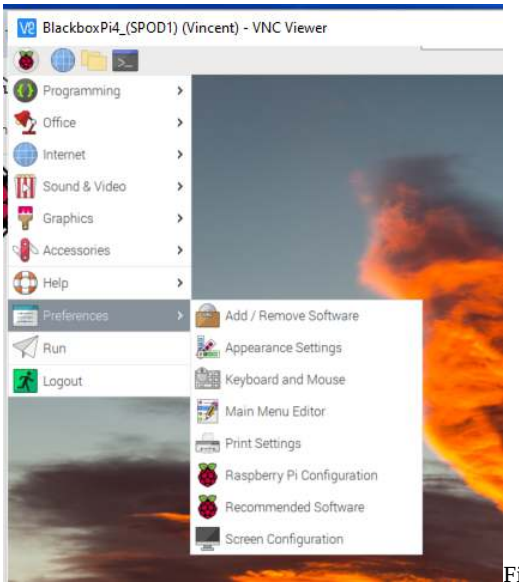


Figure 3.10.1: Screenshot (1/22/2024) of menu options to Recommended software. (Belford CC 0.0)

Now choose Visual Studio Code

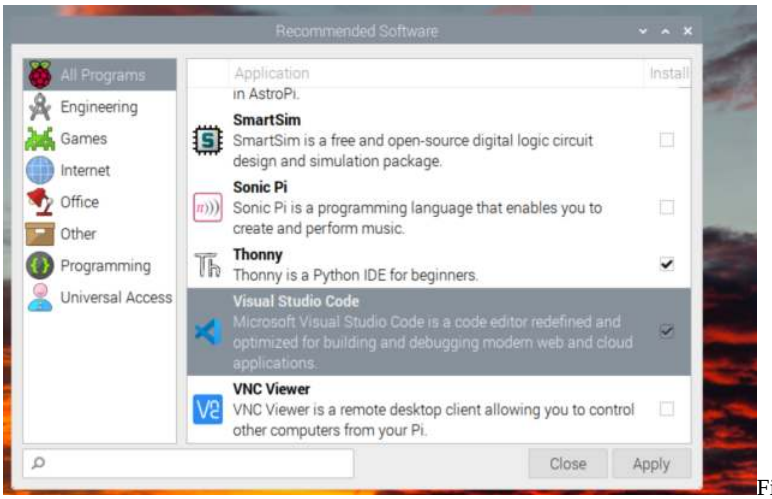


Figure 3.10.2: Screenshot (1/22/2024) of VSC option in recomended software option. (Belford CC 0.0)

and apply.

Note, the first time you open a python file with VS Code it will ask you if you want to

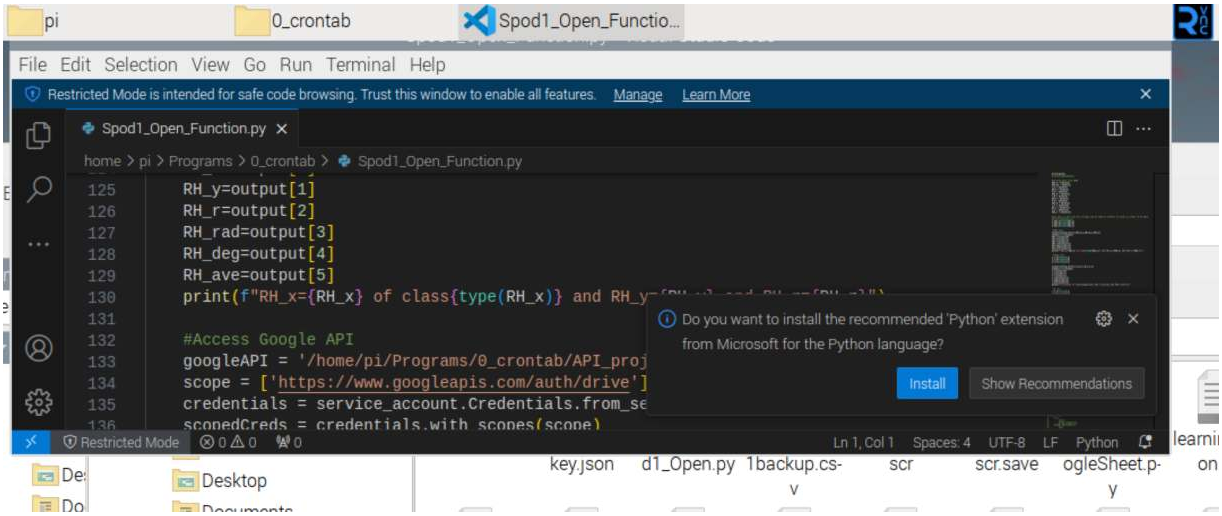


Figure 3.10.3: Screenshot (1/22/2024) of the first time VSC is run on a Pi and it asks if we want to install the recommended Python Extension. (Belford CC 0.0)

It will then instal the extension, and this make take a bit of time

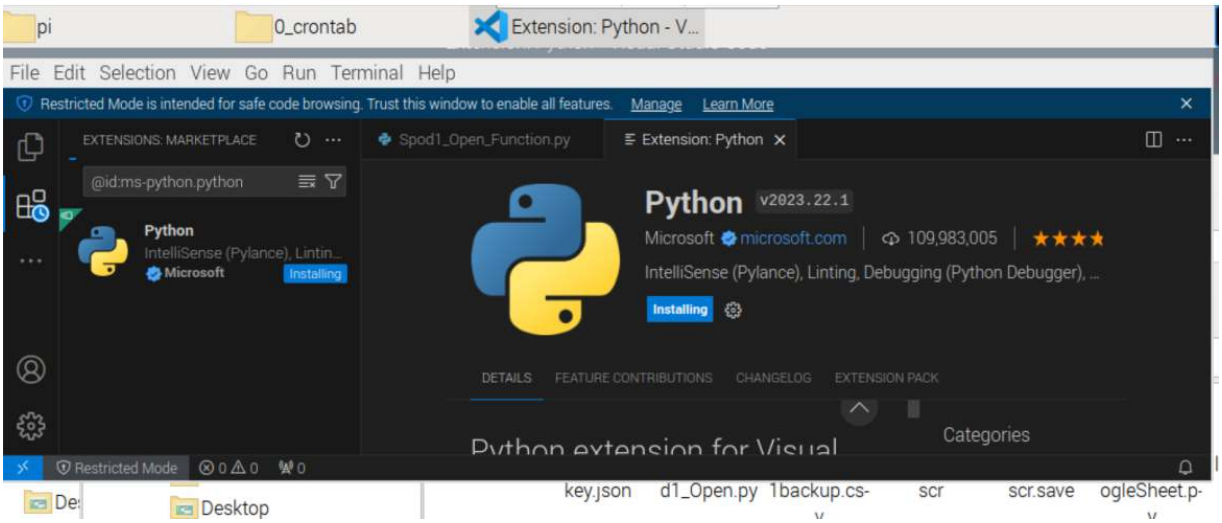


Figure 3.10.4: Screenshot of VSC Python Extension being installed. (Belford CC 0.0)

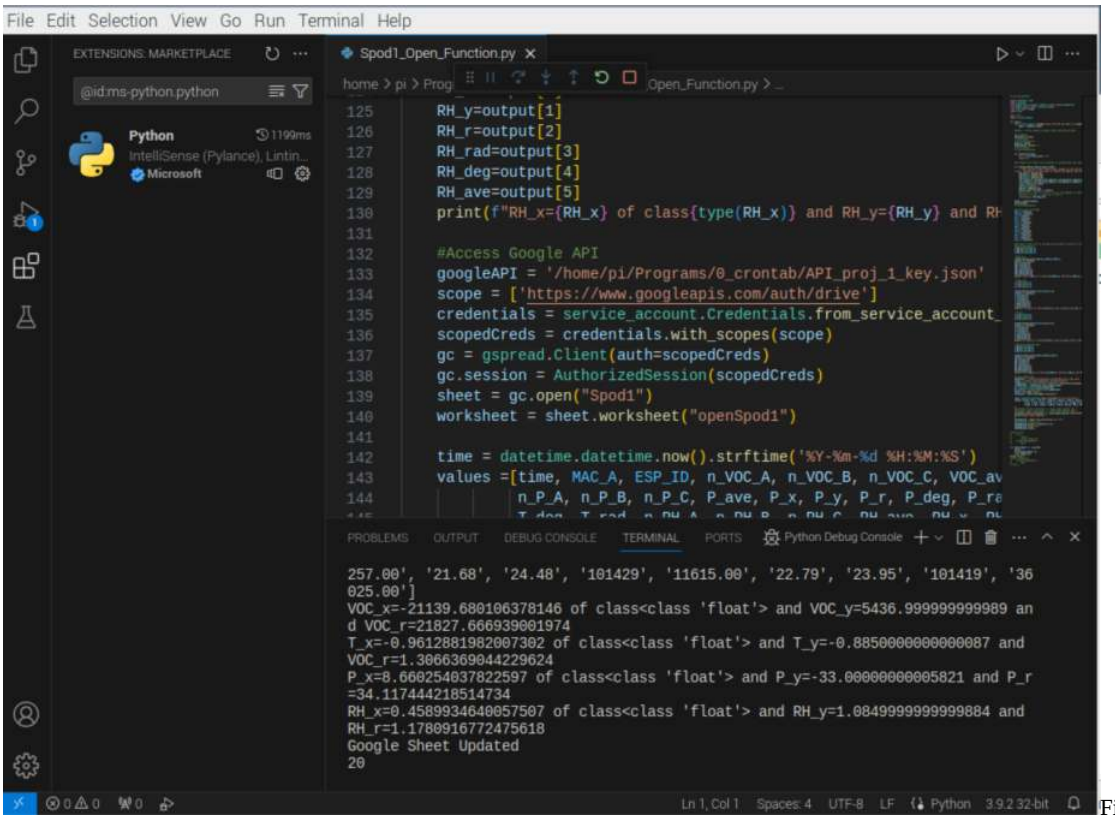


Figure 3.10.5: Screenshot (1/22/2024) of Python Program being run (in the editor) and the data being streamed to the terminal (bottom). (Belford CC 0.0)

This program involves a 40 iteration loop (the 20th was just completed) that prints the data to the terminal and then uploads the data to a Google sheet.

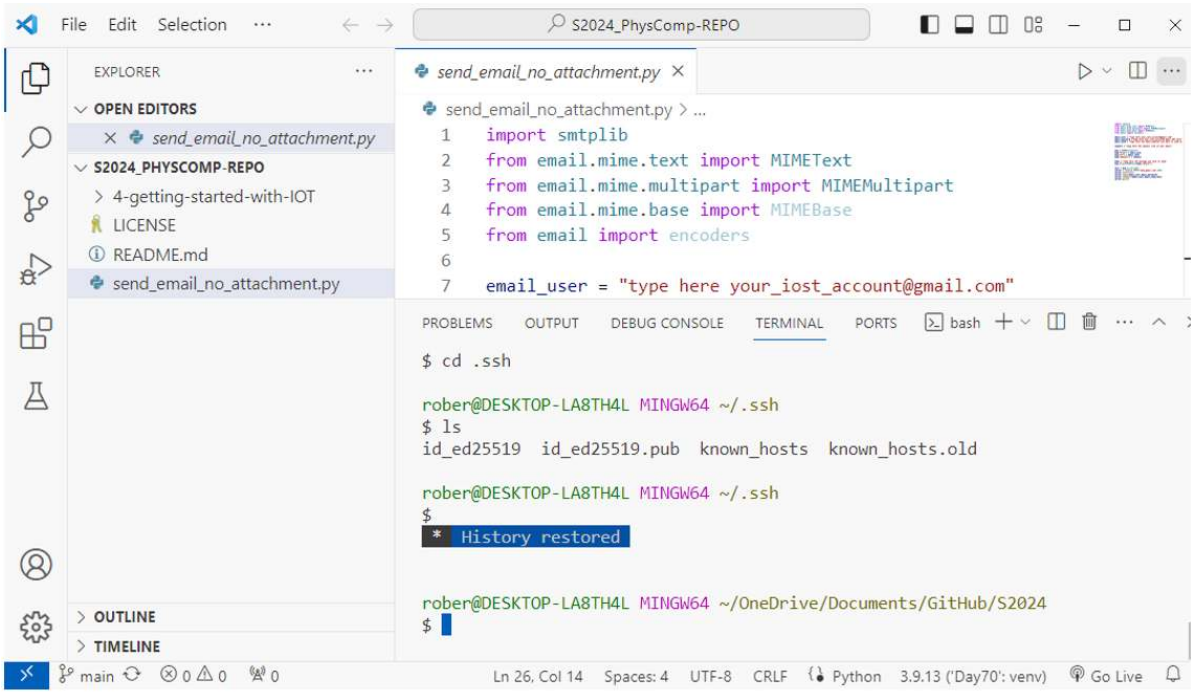


Figure 3.10.1: Copy and Paste Caption here. (Copyright; author via source)

3.10: Visual Studio Code is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

3.11: Git and GitHub

Git is a Versioning Control System you can run on your computer that allows you to back up files, track changes and revert to older versions by creating a Git Repository (Repo). Within Git you can add your files to a staging area and once you are ready you can commit them to the local repository (Repo), which is in a hidden .git folder. This creates a hash string to identify each version and allows you to access them later, and compare versions. You can also push them to a remote repository, of which GitHub is the most common, and the one we will be using in this class. Figure 3.11.1is a shematic showing this interaction.

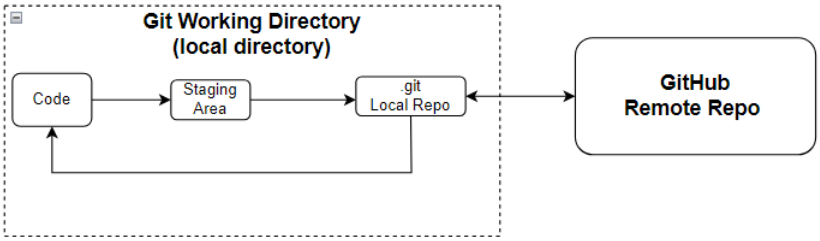


Figure 3.11.1: Overview of Git/GitHub integration. (Belford; CC-BY)

If you wish to use Git you will need to install it on your computer, and if you wish to post material to GitHub or sync your local Git repo with GitHub you will need to create an account on GitHub, but you can access files and download content from a public GitHub without having an account, and we will be posting code at the following GitHub repo, https://github.com/rebelford/S2024_PhysComp-REPO.

Git

To install Git go to https://git-scm.com/ and install the appropriate version for your operating system. As always, you can surf the web for tutorials and YouTube if a useful place to find them, but be aware that Git, like any software, is always being updated and so the interface will evolve over time. There are three common ways to use Git, on the command line, through an integrated IDE (Interactive Development Environment) like Visual Studio or PyCharm, or through GitHub Desktop.

To install Git go to the downloads page (https://git-scm.com/downloads) and install for your operating system. To see what version you have go to the command line and type

```
git --version
```

GitHub Desktop

GitHub Desktop is ideal for students who are not used to the command line and one of the things I like about it is that itallows you to open any local Git Repository to Visual Studio, Windows Explorer and GitHub. The following YouTubes are pretty good; (Deborah Kurata (Clips 10-13) and Cameron McKenzie, but go out and find your own tutorials and if you find you like you can make a hypothes.is annotation using the tags physical-computing and github-desktop, and I can add them to the site later.

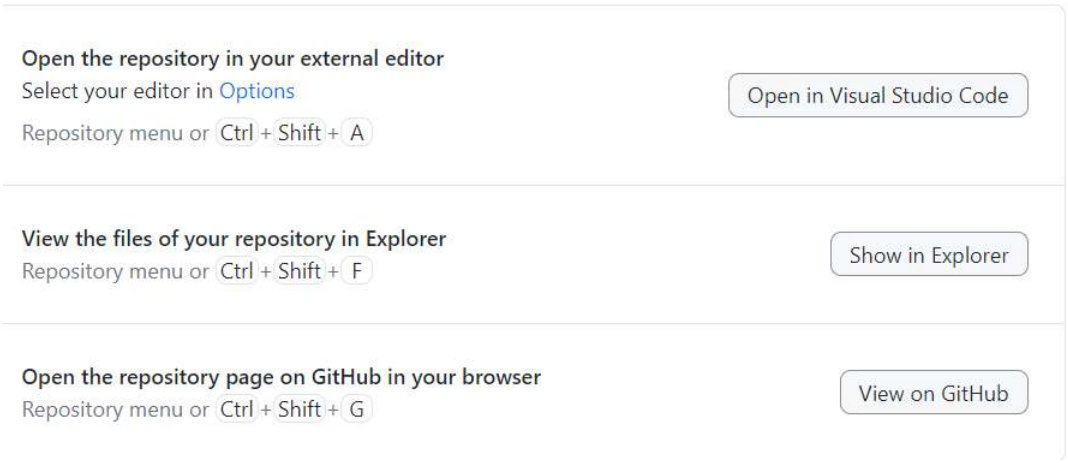


Figure 3.11.2: GitHub desktop allows you to open any Git Repo in VS Code, Explorer or GitHub through an intuitive GUI. (Belford CC 0.0)

Please note, this is just an introduction to GitHub Desktop and further information can be found in the GitHub Desktop section of the IOST library.

Visual Studio

Like GitHub Desktop we have a section of the IOST library devoted to Visual Studio, which is an IDE you should install for multiple reasons. Once again you should explore the web, look for tutorials and post them to hypothes.is using the tags phys-comp and vs-code. Note, on VD-code you can install Git-Bash on the terminal and run command line commands from VS-code. The advantage is that it allows you to use Linux commands on a Windows machine, and so the command line commands you need to run your Raspberry Pi will also work here. See Using Git in VS Code (Official Beginner Tutorial).

Command Line

We are probably not going to be using the command line very much as this is an introductory class, but a quick view of the commands may also be the best way to understand Git and GitHub. I am also including common commands here as a reference point, because as you gain skills you will want to use the command line.

command syntax	function
git init	creates a hidden .git file and transforms current directory into a git working directory, note, do not nest these.
git add <filename>	places the current file into the staging area so you can commit it (this step is skipped in GitHub Desktop)
git add .	the dot allows you to move all files from working directory to staging area

git status	tells status of staging area
git commit -m"message"	commits files from staging area to repo - you need to add a message
git log	see commits
git checkout <filename>	moves file from local Repo to working directory
git remote add <name> <url>	creates remote repo (on github defined by url), typically use <name> as "origin"
git push -u origin main	pushes local repo to remote using -u switch and sets as main branch
touch .gitignore	creates the git ignore file that allows you to specify files not to be synced with remote see github for gitignore
git clone <url>	clones remote to your machine
git branch <branch-name>	creates a branch (you need to do this if you want to clone someone elses stuff)
git branch	shows you branches

GitHub

3.11: Git and GitHub is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

3.12: GitHub

3.12: GitHub is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

3.13: GitHub Desktop

GitHub Desktop is an easy way to connect your local git repository to a remote GitHub repository. You first need to install it from this site here:

https://desktop.github.com/

Note, GitHub Desktop does not install Git, and we have to install it separately if we want to use command line operations. You need to connect the desktop to your github account. One of the nice things this does is allow you to avoid setting up SSH keys for authentication as it is done behind the scenes. If you run into issues, go to the Github Docs on Authenticating GitHub to GitHub Desktop.

Cloning a GitHub Repo

If you go to a GitHub Repo and want to clone it you can click on code, and choose "Open with GitHub Desktop" and then choose "Open GitHubDesktop.exe" and a bunch of things are done under the hood. It creates a folder in the path you specify and creates (initializes) the .git file for the working directory (if you do not see it on Windows click the view menu/hidden items), copies the commit history from the remote to the local, and then it copies all the files from the latest commit to the working folder.

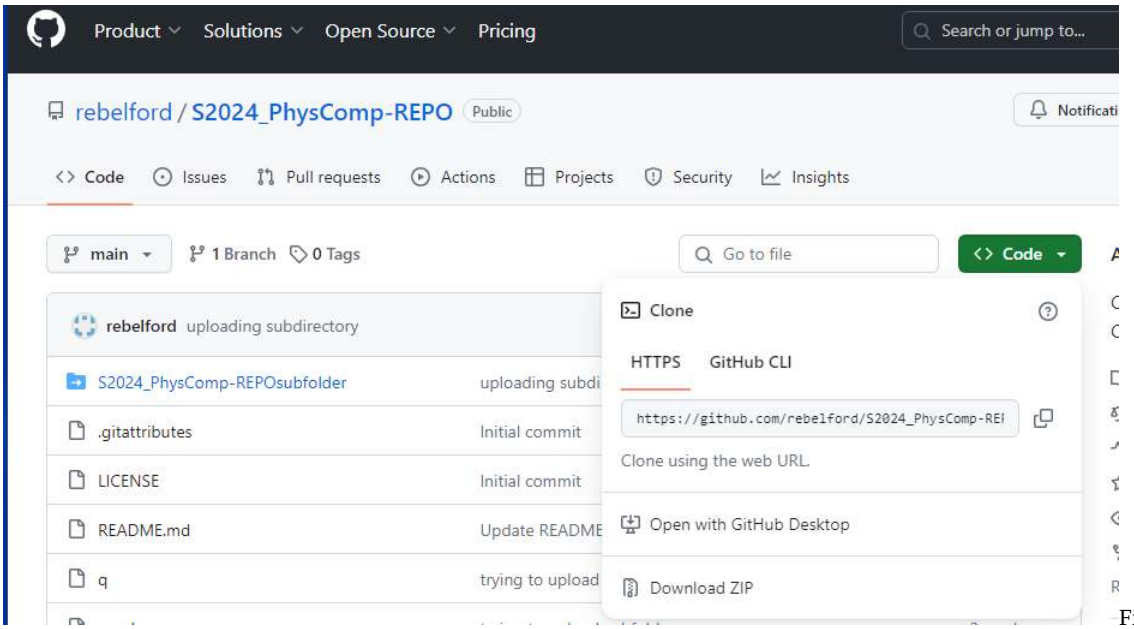


Figure 3.13.1: Copy and Paste Caption here. (Copyright; author via source)

Committing changes to main

In figure 3.13.2 several files were deleted (red squares with negative signs) and others were added or renamed (green squares with plus signs) and these edits were done using windows Explorer. By clicking "Commit to main" the add to staging step is bypassed and the files are made as commits and indexed in the .git file with an associated hash tag. Once complete the GUI looks like figure 3.13.4 and you can push the local changes from your git repository to the remote GitHub repository.

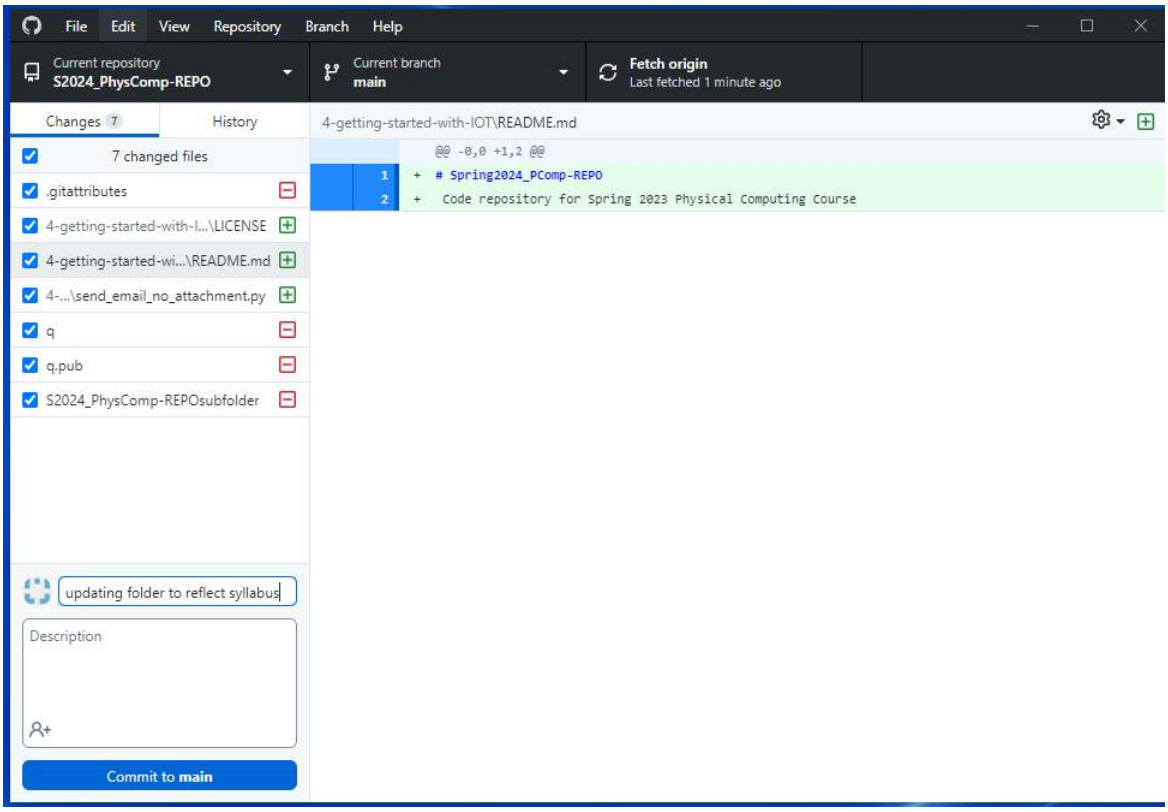


Figure 3.13.2:There are seven changes that will be placed in a commit once you hit "commit to main". (Belford CC 0.0)

If you click the History tab you can see all the commits and by clicking on one of them you can see what was added or deleted in that commit, as in figure 3.13.3

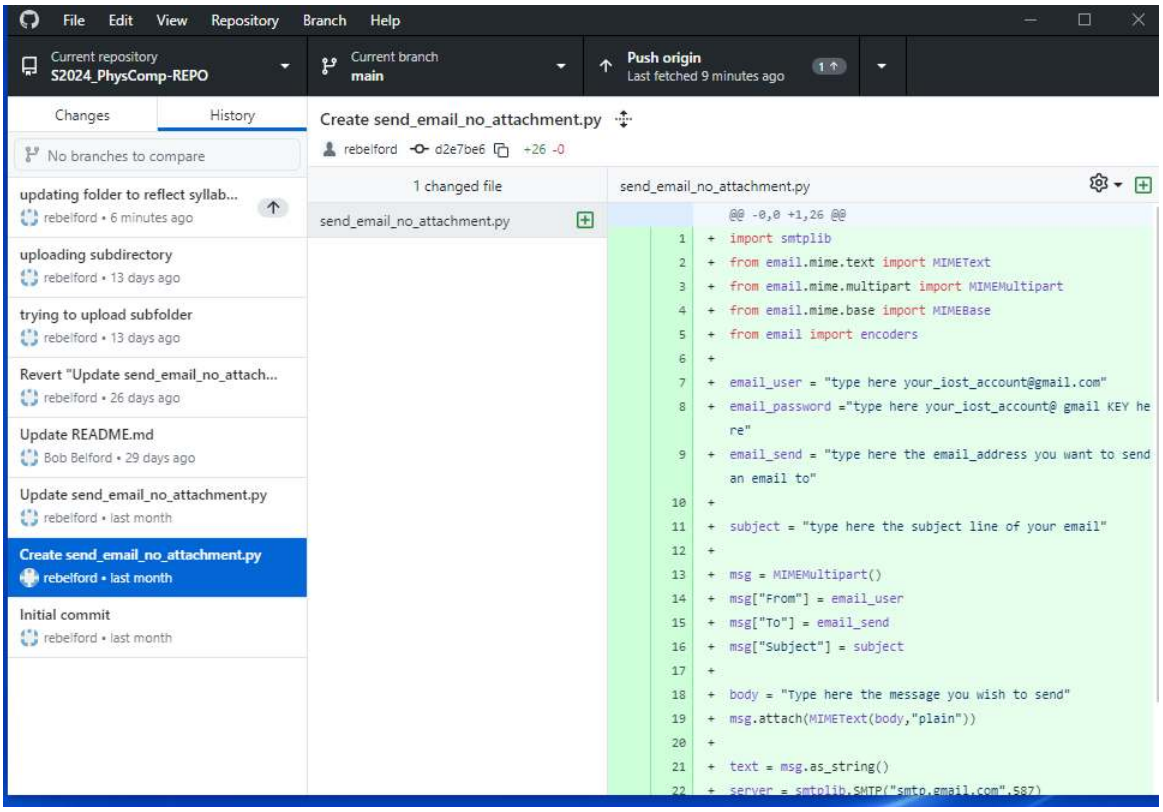


Figure 3.13.3: Here we clicked the History tab and can see the hisroty of commits. If you click any commit it will show the changes, if material was deleted it would show in pink. If you want to push the latest commit to GitHub you click "push origin" (Belford CC 0.0)

If you hover over the d2e7be6 in \(PageIndex{3} you will see the full hash of the commit

Commit Messages

There should only be one message per commit, and it should be in present tense, not past tense. Commit message are very important, especially for team work where there may be a merge conflict.

Push to GitHub

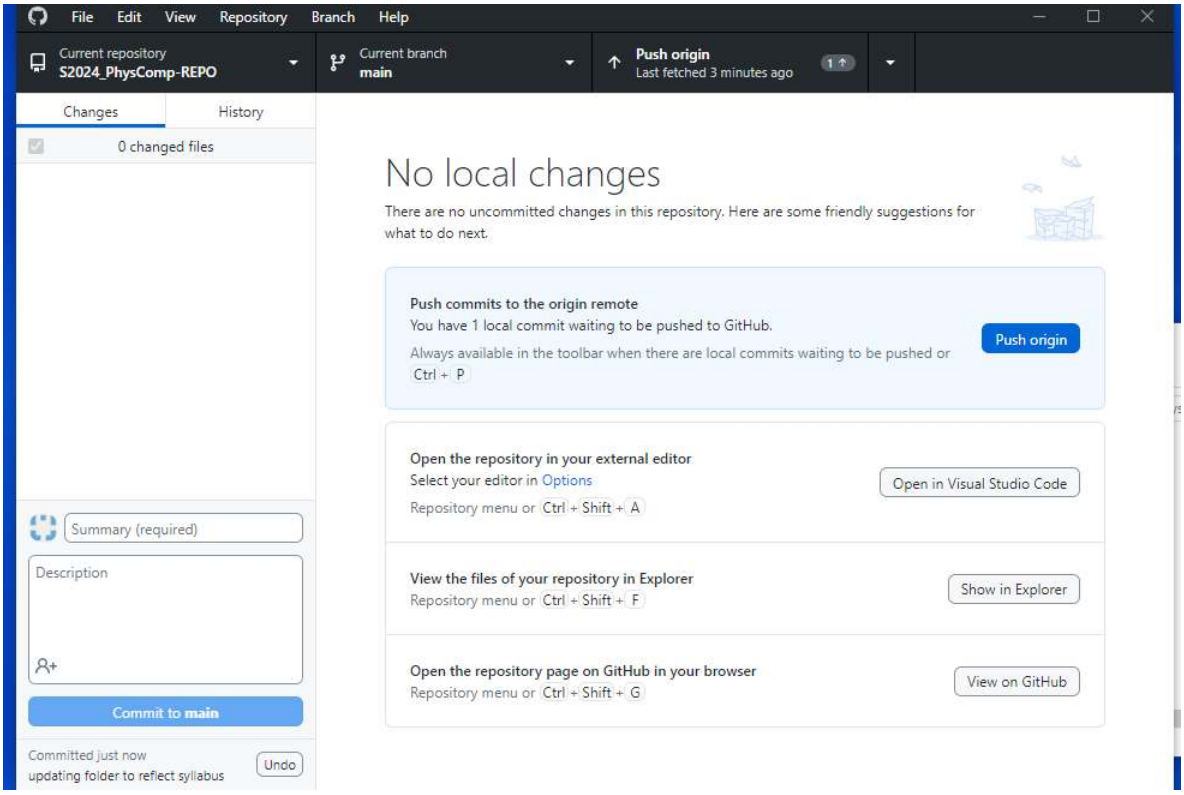


Figure 3.13.4: Copy and Paste Caption here. (Copyright; author via source)

When you click Push origin the local repo syncs with the remote repo in GitHub

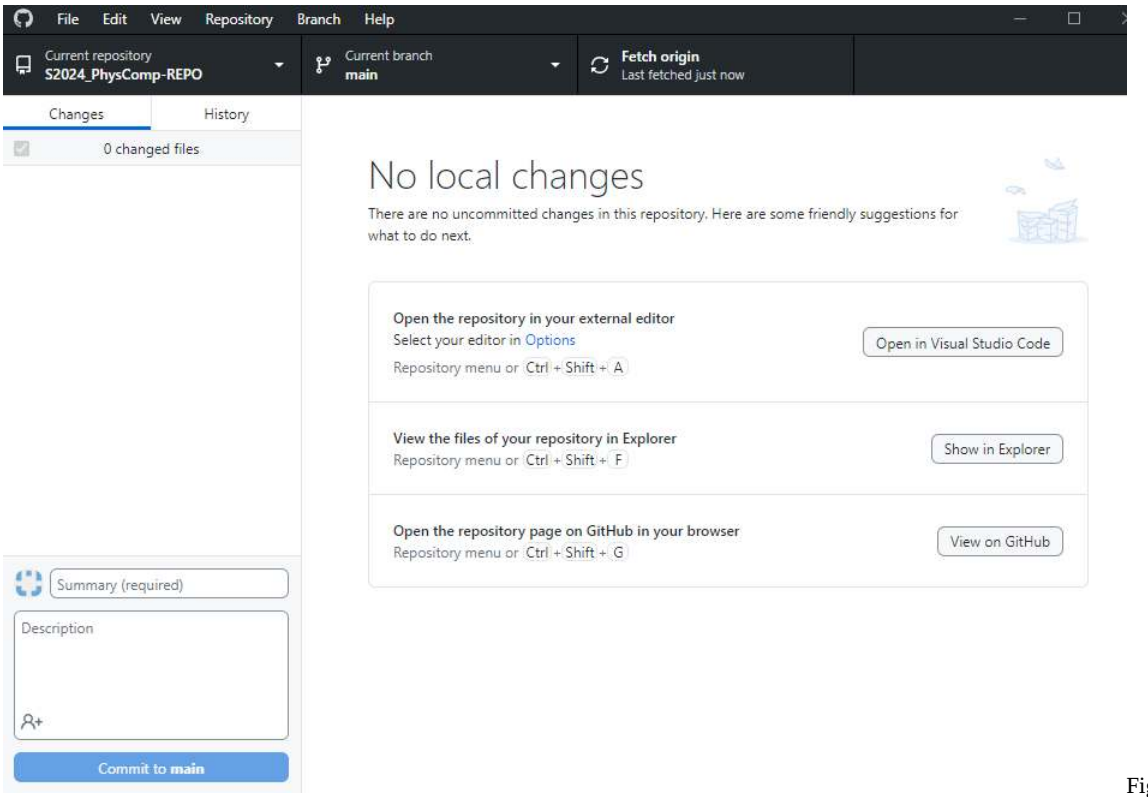


Figure 3.13.5: Copy and Paste Caption here. (Copyright; author via source)

Note -Collaborative Work

If you are on a team you should pull from github to be sure you have the latest version, and then push your commit.

3.13: GitHub Desktop is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

3.14: Online Resources

Course Resources

Computing in General

1. Computational Thinking by Karl Beecher, a high level book with Python examples
2. PBS Crash Course in Computer Science - Excellent 41 YouTube video series moderated by Carrie Anne Philbin

Flow Charts and Circuit Diagrams

Draw.io Tutorials - official tutorials

Python

1. Conversational Python: An Introduction to Computer Science Experienced through Conversational Banter by Jason Shepherd - well worth reading
2. ESCIP (Enhancing Science Courses by Integrating Python) - Community resources in Jupyter notebooks for chemistry, physics and astronomy education. These are not introductory but advanced level applications.

Linux

Raspberry Pi

3.14: Online Resources is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

3.15: Git

3.15: Git is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

CHAPTER OVERVIEW

4: Python Data Types

Topic hierarchy

4.1: Introduction to data
4.2: Data Types
4.3: String (atomic)
4.4: Integers (atomic)
4.5: Float (atomic)
4.6: Boolean (atomic)
4.7: String (container)
4.8: List (container)
4.9: Dictionary (container)
4.10: Tuple (container)
4.11: 3.11 Set (container)

IoST Map for 3: Data Types

Section	Original Source
4.1: Introduction to Data	1.6.1: Introduction to Data
4.2: Data Types	1.8.3: Data
4.3: String (atomic)	5.2.1: String (atomic)
4.4: Integers (atomic)	5.2.2: integers
4.5: Float (atomic)	5.2.3: Float
4.6: Boolean (atomic)	5.2.4: Boolean
4.7: String (container)	5.2.5: String (container)
4.8: List (container)	5.2.6: List
4.9: Dictionary (container)	5.2.7: Dictionary
4.10:Tuple (container)	5.2.8: Tuple
4.11: Set (container)	5.2.9: Set

4: Python Data Types is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

4.1: Introduction to data

What is data?

To the chemist data are the measured or counted values that can be collected or produced to understand relationships of observable or computed phenomena that are germane to the practice of science (both empirical and computational). To the chemist there are different types of data that are defined by how the data is generated, like the mass or temperature of a sample, or the spectra of a compound. This data is often stored on a computer in a file or database, and can be subsequently processed through various software programs.

To the computer scientist or software program data has a different meaning in that there are different data types that represent how the computer stores information. That is, a computer does not store a measured phenomena like the temperature of a sample, but a digital data type, a representation of the temperature that a software agent can interact with. For example, a letter of the alphabet would be a different type of data than a number, because you can not do arithmetic calculations on letters like you do on numbers. These different ways of representing information are data types.

We need to understand both meanings of the concept of data, and in this section we will learn how computers store data, and the different types of data from the perspective of programming and software agents.

What is a database?

Databases are a way computers store information in a manner that can be retrieved. You use databases all the time. Do you realize that as you read this web page you are using a database? Yes, this web page is not a digital file like a MS Word document that saves the information like a sheet of paper, but instead the web browser is displaying information that was pulled from a database as the page is loaded. That is, LibreText is a Wiki that is hosted on the MindTouch knowledge management platform and the information you see is drawn from a database when the page is loaded. Webpages that are pulled from databases are often called dynamic web content, and those that are files are called static web content. Of course, databases can store different types of information, and this class will be using databases that store information related to chemical compounds. But it is important to realize that the use of databases in the twentieth century are pervasive, and you are actually using a database right now, as you read this webpage.

How do databases store information?

Databases store data, which is the representation of information through a binary code that computing machines can read. A bit is the smallest binary value with two possibilities, 0 or 1. This data needs to be stored on a physical medium so the machine can read it. In the old days data was stored on punch cards (figure 1.3.1), which allowed for a binary representation of each position, which could be either punched or not punched (bitten or not bitten). If each location of memory is allowed a certain number of bits, then you can generate different combinations, and give those different combinations different meanings.

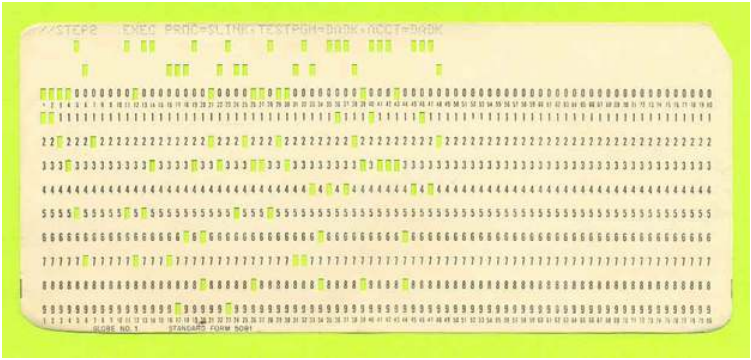


Figure \(\PageIndex{1}\): 5081 data processing card containing a line of DOS JCL code. The code reads: //STEP2 EXEC PROC=SLINK,TESTPGM=DADK,ACCT=DADKThe (CC BY-SA; Dick Kutz at English Wikipedia)

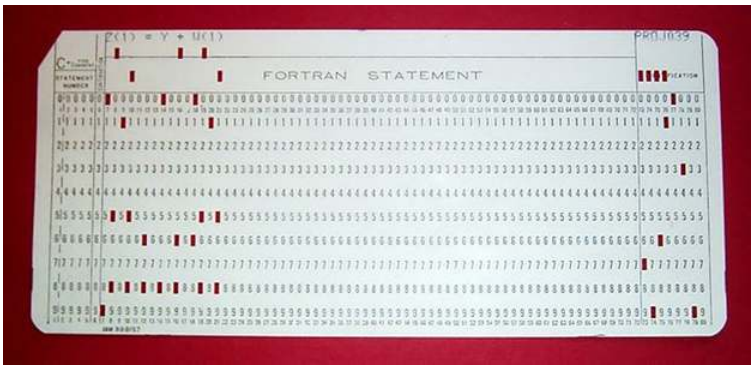


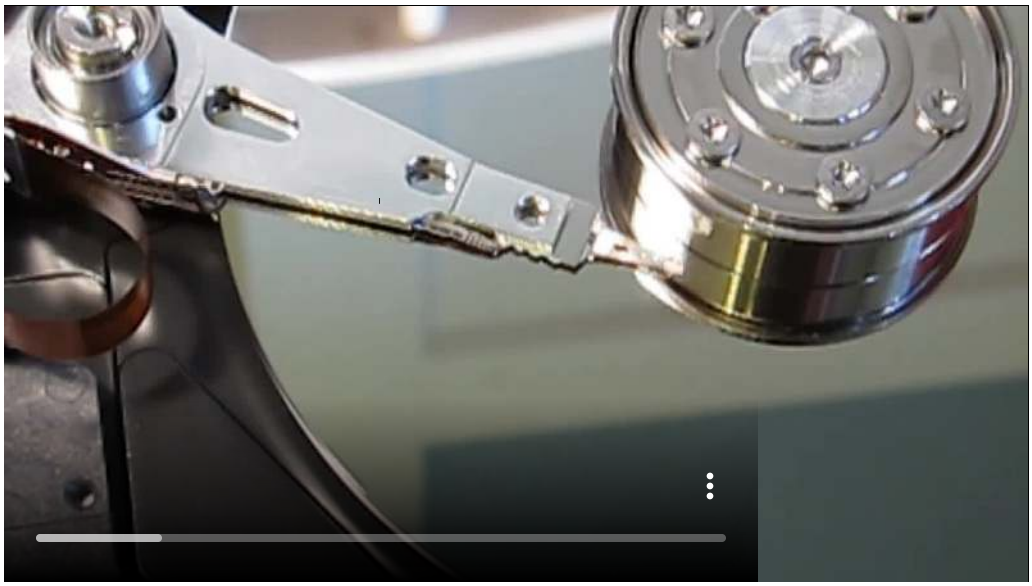
Figure \(\PageIndex{2}\): Old Fortran punch card, one of the earliest computer based means for storing data (CC BY-SA; Arnold Reinhold)

A quick look at these possibilities shows that n bits gives 2^n possible combinations.

- 1 bit has two (2^1)possibilities : 0 or 1, and so can represent two different things
- 2 bits has four (2^2) possibilities: 00, 01, 10, or 11, and so can represent four different things
- 3 bits has 8 (2^3) possibilities: 001, 010, 100, 011, 101, 101, 110, 111, and so can represent 8 different things.
- 8 bits has 8 (2^8) possibilities, which is 128, ranging from 00000000 to 11111111, and we won't write them all down here.
- n bits has (2^n)

A byte of data is defined as 8 bits and so has (2^8) or 128 values (which run for 0 to 127).

How a hard drive works?



The take home message here is that everything is stored on the computer in the form of a binary bit, be it a text document, picture, molecular structure data or a spectral file. Each of these represent a different data type and so when you interact with the database, you need to know what type of data is stored, and then use software that can "read" that type of data. Likewise, if you write some simple script to interact with data, you need to recognize the data type you are interacting with, for example, you can do math with numbers, but not letters, and so a number needs to be a different data type than a letter.

Today we do not use punch cards but still store data as a binary representation on a physical device that can be electronically read, like magnetic tape, hard drives, flash drives, SSD (Solid State Disk) and the like. The way magnetic based storage devices work is through the North-South alignment of the magnetic field, where one of these (N-S) would be given the value of 1, and the other (S-N) would be the 0. If you are interested in learning how a hard drive works there is a real good 6 minute video on Nick Parlante's computer science page from Stanford. Flash drives and SSDs have no moving parts and are not based on magnetism, but represent ones and zeros by the ability of tiny channels (gates) within a transistor to be able to conduct (1) or not conduct (0) electricity. It

should be noted that after 10-20 years flash drives can lose their memory. In fact, surprisingly magnetic tape is the longest lasting digital storage, although it is the slowest to use.

What is a Data Type?

If you store some data on a computer it must have a data type, which allows python to know how to read it. In Python there are two different types of Data Type (no pun intended), those of a single character, and those with multiple characters. These data types are actually a type of object in Python called a class, and each class has different attributes and methods (functions) associated with it. For example the number "1" as an interger, when added to itself, gives 2, but the number "1" as a letter (string), gives 11 when added to itself. In the box below we define two variables for the value of one. If we print the output of the type command, we can see what type of data each value is, and as we shall see, 1 is stored differently on the computer than 1.0.

```
x=1
y=1.0
print(type(x))
print(type(y))
```

run | restart | restart & run all

```
Hello world!
```

Note, python treats "letters" as a single character of the string data type, and I have chosen to put it in both places of the following table, which might not be correct, but makes sense to me.

Single Character (atomic) Data Types

Name	Python type	Description
Integer	Int	Whole Number
Floating point	float	Decimal point numbers
Boolean	bool	True or False logical values
Strings	str	single character (this is really a string of length one)

Container (Multiple Character Data) Types

Name	Python type	Description
Strings	str	ordered immutable string of characters (a sentence is a string)
List	list	unordered mutable sequence of objects [10, "10", 10.0]
Dictionary	dict	unordered Key:Value pair {k1:v1, k2:v2}
Tuple	tup	ordered immutable sequence of objects (10, "10", 10.0)
Set	set	unordered group of unique objects {'a','b'}

note

- ordered means it can not switch its position, like the order of the elements in the periodic table, carbon is always 6.
- unordered means any order can occur
- mutable means a value can change, like the mass of a person can change over time and thus mutable
- immutable means a value can not change, like the atomic mass of an isotope of hydrogen is a constant and immutable.

Tutorials

- SparkFun: Binary

This page titled 4.1: Introduction to data is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

4.2: Data Types

In many ways programming can be called data processing, and we need to look at data and variables before proceeding. Data are the values that are assigned to a variable by the assignment operator (=) and there are different data types

Data Types

and we have a book 5.2 is devoted to Data Types and Structures and we will go into these in more detail as the course proceeds. But let's start with three basic data types, string (these are like letters of the alphabet), integers and floats (have decimal values). We will see that these are stored differently in the computer, and they behave differently.

In the following code we define a=b=c=d=1, and then we add a+b and then c+d, and the answers are different. Do you know what is going on? (Do not worry about the print statement, we will discuss that later).

```
a="1"
b="1"
c=1
d=1
print(f"a+b={a+b}")
print(f"c+d={c+d}")
```

run | restart | restart & run all

```
Hello world!
```

a and b were variables assigned data of the type string, and so they were treated as letters and not numbers, and you effectively concatenated them getting 11, while c and d were numbers. Python has a built-in function called "type" that will give you the type of data a variable is.

```
a="1"
b="1"
c=1
d=1
print(f"a is type: {type(a)}")
print(f"c is type: {type(c)}")
```

run | restart | restart & run all

```
Hello world!
```

Notice how the output of the above code is

```
a is type: <class 'str'>
c is type: <class 'int'>
```

Python is an Object Oriented Programming language and a class is a type of objects that has attributes (features) and methods (functions), and so a data type is actually more than just how you store a data value on a computer, and in fact there are sort of two data types, scalar and container. Strings are actually a bit tricky in python as they are sort of both.

Scalar (atomic) data types

These are related to how the individual values are stored in a computer (see book 5 section 2). These are

- str: string-single character in a sequence of characters of type string
- float: floating point numbers (3.14, -0.005), 2.004...)
- int: integer values (4, -3, 100...)

- bool: Boolean values (True or False)

Container (collection) data types

These are collections of the other data types. Note that string is actually in both forms, as

- string: sequence of characters of type string (these can be treated like a set)
- list: ordered (mutable) collection of items
- tuple: ordered (immutable) collection of items
- set: unordered (mutable) collection of unique items
- dictionary: mutable collection of key-value pairs

Scope and Namespace

When assigning a value to a variable there is a scope for which that value is valid, even within a program. For example, an assignment within a function (local scope) is not valid outside of the function (global scope). Look at the following code. You would think you start with mass_solute =1 and then it changes to 2, but the function does not effect the value of the first assignment. that is, you have two different variables with the same name.

```
mass_solute=1
def add_solute():
    mass_solute=2
    print(f"solute inside function: {mass_solute}")

add_solute()
print(f"solute outside function: {mass_solute}")
```

run | restart | restart & run all

```
   Hello world!
```

If you want the function to change the value of a variable defined outside of the function, you need to add the global command to extend the scope outside of the function

```
mass_solute=1
def add_solute():
    global mass_solute
    mass_solute=2
    print(f"solute inside function: {mass_solute}")

add_solute()
print(f"solute outside function: {mass_solute}")
```

run | restart | restart & run all

```
          hello world
```

Edit the following code for a titration so it works

```
vol_titrant=0
def add_titrant():

    vol_titrant += 1
    print(f"total vol titrant inside function is: {vol_titrant}")
```

```
add_titrant()
print(f"total vol titrant outside function: {vol_titrant}")
```

run | restart | restart & run all

```
Hello world!
```

The problem with the above is that you are making a variable global within a function and this could cause problems if you use the function is some program where you were not aware of that. A better way is to use the return statement and make the assignment based on that.

```
vol_titrant=0
def add_titrant():
    print(f"total vol titrant inside function is: {vol_titrant}")
    return vol_titrant+1

vol_titrant = add_titrant()
print(f"total vol titrant outside function: {vol_titrant}")
```

run | restart | restart & run all

```
Hello world!
```

Where you want to use global variables is for constants, and these should be in upper case. For example, your web address, or a constant like the number pi

```
PI=3.14
URL='https://ioct.tech'
```

Local vs. Remote Data

local

4.2: Data Types is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

4.3: String (atomic)

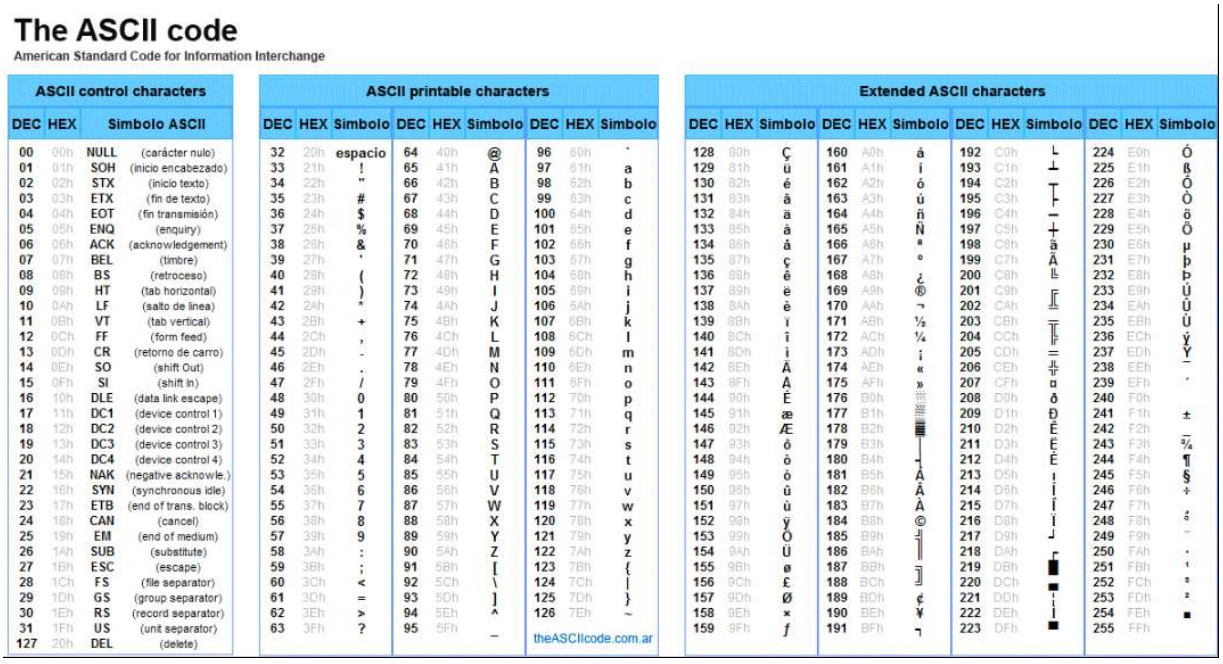
In the pre-digital era data was preserved in the form of books and data tables, with data that has been excerpted from the primary literature often being classified as legacy data. A string character is just a digital representation of a letter or number and can not be used in computational mathematical operations (if you add one and one you get eleven, not two). That is, it is two ones next to each other.




Figure 2.12.1: Royal Typewriter (Public Domain: Wikimedia Commons), Christian Herald Press Room, 1898 (Public Domain, Wikimedia Commons)

ASCII (American Standard Code for Information Interchage)

One of the first things needed to be encoded if people were to move from typewriters to word processors were the characters of the alphabet. In the early days of computers 8 bit chip of memory was common and the American Standard Code for Information Interchange (ASCII) was developed and based on the 8 bit byte, which shows one set of code that allows computers to interact with a keyboard to store information. Note the first 32 ASCII characters are unprintable codes used to control devices, and the remaining 156 characters are used to store symbols like numbers and the letters of the alphabet. We are calling this an atomic code level as each byte (8 bits) is the encoding of a single character of the class string. As we shall see, a word is also a string, but a string of multiple characters.



The ASCII code

American Standard Code for Information Interchange

ASCII control characters

DEC	HEX	Simbolo ASCII	
00	00h	NULL	(carácter nulo)
01	01h	SOH	(inicio encabezado)
02	02h	STX	(inicio texto)
03	03h	ETX	(fin de texto)
04	04h	EOT	(fin transmisión)
05	05h	ENQ	(enquiry)
06	06h	ACK	(acknowledgement)
07	07h	BEL	(timbre)
08	08h	BS	(retroceso)
09	09h	HT	(tab horizontal)
10	0Ah	LF	(salto de linea)
11	0Bh	VT	(tab vertical)
12	0Ch	FF	(form feed)
13	0Dh	CR	(retorno de carro)
14	0Eh	SO	(shift Out)
15	0Fh	SI	(shift In)
16	10h	DLE	(data link escape)
17	11h	DC1	(device control 1)
18	12h	DC2	(device control 2)
19	13h	DC3	(device control 3)
20	14h	DC4	(device control 4)
21	15h	NAK	(negative acknowle.)
22	16h	SYN	(synchronous idle)
23	17h	ETB	(end of trans. block)
24	18h	CAN	(cancel)
25	19h	EM	(end of medium)
26	1Ah	SUB	(substitute)
27	1Bh	ESC	(escape)
28	1Ch	FS	(file separator)
29	1Dh	GS	(group separator)
30	1Eh	RS	(record separator)
31	1Fh	US	(unit separator)
127	20h	DEL	(delete)

ASCII printable characters

DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo
32	20h	espacio	64	40h	@	96	60h	`
33	21h	!	65	41h	A	97	61h	a
34	22h	"	66	42h	B	98	62h	b
35	23h	#	67	43h	C	99	63h	c
36	24h	$	68	44h	D	100	64h	d
37	25h	%	69	45h	E	101	65h	e
38	26h	&	70	46h	F	102	66h	f
39	27h	'	71	47h	G	103	67h	g
40	28h	(	72	48h	H	104	68h	h
41	29h	)	73	49h	I	105	69h	i
42	2Ah	*	74	4Ah	J	106	6Ah	j
43	2Bh	+	75	4Bh	K	107	6Bh	k
44	2Ch	,	76	4Ch	L	108	6Ch	l
45	2Dh	-	77	4Dh	M	109	6Dh	m
46	2Eh	.	78	4Eh	N	110	6Eh	n
47	2Fh	/	79	4Fh	O	111	6Fh	o
48	30h	0	80	50h	P	112	70h	p
49	31h	1	81	51h	Q	113	71h	q
50	32h	2	82	52h	R	114	72h	r
51	33h	3	83	53h	S	115	73h	s
52	34h	4	84	54h	T	116	74h	t
53	35h	5	85	55h	U	117	75h	u
54	36h	6	86	56h	V	118	76h	v
55	37h	7	87	57h	W	119	77h	w
56	38h	8	88	58h	X	120	78h	x
57	39h	9	89	59h	Y	121	79h	y
58	3Ah	:	90	5Ah	Z	122	7Ah	z
59	3Bh	;	91	5Bh	[	123	7Bh	{
60	3Ch	<	92	5Ch	\	124	7Ch	|
61	3Dh	=	93	5Dh	]	125	7Dh	}
62	3Eh	>	94	5Eh	^	126	7Eh	~
63	3Fh	?	95	5Fh	_			

theASCIIcode.com.ar

Extended ASCII characters

DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo
128	80h	Ç	160	A0h	á	192	C0h	└	224	E0h	Ó
129	81h	ü	161	A1h	í	193	C1h	┴	225	E1h	ß
130	82h	é	162	A2h	ó	194	C2h	┬	226	E2h	Ô
131	83h	â	163	A3h	ú	195	C3h	├	227	E3h	Ò
132	84h	ä	164	A4h	ñ	196	C4h	─	228	E4h	õ
133	85h	à	165	A5h	Ñ	197	C5h	┼	229	E5h	Õ
134	86h	å	166	A6h	ª	198	C6h	ã	230	E6h	µ
135	87h	ç	167	A7h	º	199	C7h	Ã	231	E7h	þ
136	88h	ê	168	A8h	¿	200	C8h	╚	232	E8h	Þ
137	89h	ë	169	A9h	®	201	C9h	╔	233	E9h	Ú
138	8Ah	è	170	AAh	¬	202	CAh	╩	234	EAh	Û
139	8Bh	ï	171	ABh	½	203	CBh	╦	235	EBh	Ù
140	8Ch	î	172	ACh	¼	204	CCh	╠	236	ECh	ý
141	8Dh	ì	173	ADh	¡	205	CDh	═	237	EDh	Ý
142	8Eh	Ä	174	AEh	«	206	CEh	╬	238	EEh	¯
143	8Fh	Å	175	AFh	»	207	CFh	¤	239	EFh	´
144	90h	É	176	B0h	░	208	D0h	ð	240	F0h	
145	91h	æ	177	B1h	▒	209	D1h	Ð	241	F1h	±
146	92h	Æ	178	B2h	▓	210	D2h	Ê	242	F2h	‗
147	93h	ô	179	B3h	│	211	D3h	Ë	243	F3h	¾
148	94h	ö	180	B4h	┤	212	D4h	È	244	F4h	¶
149	95h	ò	181	B5h	Á	213	D5h	ı	245	F5h	§
150	96h	û	182	B6h	Â	214	D6h	Í	246	F6h	÷
151	97h	ù	183	B7h	À	215	D7h	Î	247	F7h	¸
152	98h	ÿ	184	B8h	©	216	D8h	Ï	248	F8h	°
153	99h	Ö	185	B9h	╣	217	D9h	┘	249	F9h	¨
154	9Ah	Ü	186	BAh	║	218	DAh	┌	250	FAh	·
155	9Bh	ø	187	BBh	╗	219	DBh	█	251	FBh	¹
156	9Ch	£	188	BCh	╝	220	DCh	▄	252	FCh	³
157	9Dh	Ø	189	BDh	¢	221	DDh	¦	253	FDh	²
158	9Eh	×	190	BEh	¥	222	DEh	Ì	254	FEh	■
159	9Fh	ƒ	191	BFh	┐	223	DFh	▀	255	FFh	

Figure 4.3.3: ASCII Code. (CC BY-SA; Yuriy Arabskyy)

The original asciII code was 7 bit and the extende asci II is 8 bit and so there are 256 characters. The complete table can be seen at www.ascii-code.com and a few representations are shown in the table below.

Dec	BIN	Symbol	Description

0	00000000		Null Char
10	00001010		Line Feed
48	00110000	0	Zero
68	01000100	D	Uppercase D
100	01100100	d	Lowercase d
128	10000000	€	Euro
197	11000101	Å	Angstrom symbol

The original ASCII code was 7 bit and the extended asci II is 8 bit and so there are 256 characters. The complete table can be seen at www.ascii-code.com. One of the shortcoming of ascii is that there are only 256 possible representations and so there is a limit to the number of symbols (or commands like "tab") that can be encoded.

Unicode & UTF-8

In 1988 the Unicode Consortium was formed, which was formed as a public benefit (non-profit) in 1991 and unicode is a variable length encoding schema based on the UTF-8 (**U**niversal coded character set **T**ransformation **F**ormate - **8** bit) format. This is backward compatible with ascii, but the variable length of between 1 and 4 eight bit bytes allows for the encoding of of more characters (two bytes = 256 x 256=65536 possible code points) and this allows for various symbols, sub/superscripts beyond the original ascii.

STIX fonts - the Scientific Technical and Information Exchange (STIX) is a project based on UTF-8 that provides font for scients and is integrated into Google Fonts.

In Python strings are also a class of multiple characters, and what is being described here is the representation of a single character of this class. See the section on string (container) to learn about the methods associated with a string.

References

Information Science for Chemists by Stuart Chalk, 2015 Cheminformatics OLCC

This page titled 4.3: String (atomic) is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

4.4: Integers (atomic)

Integers are whole numbers represented in a binary representation, in contrast to the decimal system where we have values of 0 to 9, we only have values of 0 and 1. So in a decimal system each subsequent digit in a number is a factor of 10 bigger (the 100ths position is 10^2, the thousandths is 10^3), in a binary they increase by 2^n.

2^3	2^2	2^1	2^0	
8's	4's	2's	1's	Decimal
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

Figure 4.4.1: binary and decimal representations for a four bit coding scheme, notice one is used for zero. (Belford CC0.0)

when determining the range of possible values for an integer we also have to reserve a configuration of the sign [+] or [-]

range of 32 bit (4 byte) chip is:

$$-2^{31} \ \ to \ \ 2^{31}-1$$

with the largest value being 2,147,483,648.

For a 64 dbit (8 byte) chip it is:

$$-2^{63} \ \ to \ \ 2^{63}-1$$

with the largest value being 9,223,372,036,854,775,807

This page titled 4.4: Integers (atomic) is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

4.5: Float (atomic)

A floating type represenation using an encoding scheme similiar to scientific notation. The first byte represents the exponent and the remaining represent the prexponential, with one bite reserved for the sign.

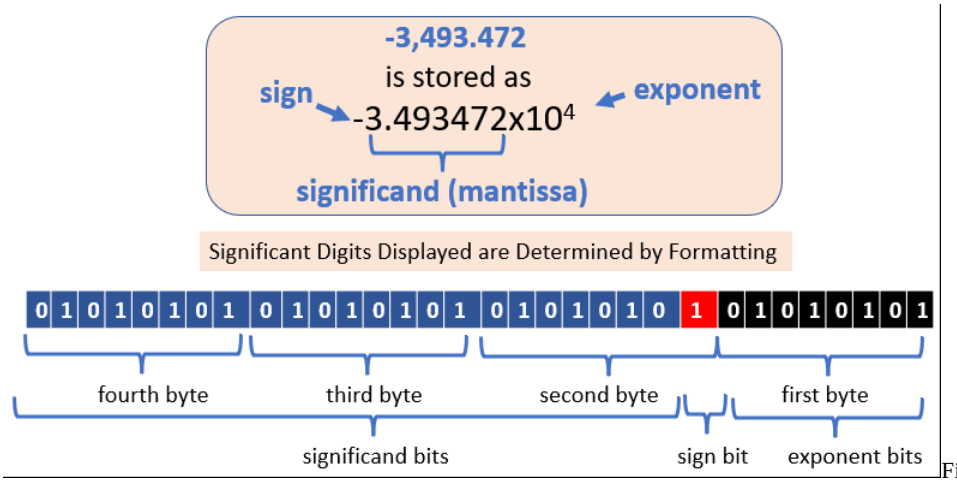


Figure 4.5.1: floating decimal representation for 32 bit binary encoding. (Copyright; belford cc 0.0)

The following table gives the range of values for floating systems in 32 and 64 bit binary encoding.

Format	Total bits	Significand bits	Exponent bits	Smallest Number	Largest Number
Single precision	32	23 + sign	8	ca. 1.2 x10^{-38}	ca. 3.4 x10^{38}
double precision	64	52 + sign	11	ca. 2.2 x10^{-308}	ca. 1.8 x10^{308}

This page titled 4.5: Float (atomic) is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

4.6: Boolean (atomic)

Boolean Data Types

Boolean expressions are expressions that can result in only one of two possible answers, true or false. These results in a special data type, where Boolean values associate a value of zero for a false statement and a value of 1 for a true statement. Boolean algebra had it origins in the 1800's and George Boole's book "The Mathematical Analysis of Logic, Being an Essay Towards a Calculus of Deductive Reasoning" that is available through the Gutenberg Project. Boolean data types are essential to computational reasoning because they can result in logical reasoning based outcomes based on the truth (or falseness) of a statement or condition in terms of binary code. For example, you could have an if statement branch your programming logic based on the conditions resulting from a comparison operator that decides if one number is greater than another.

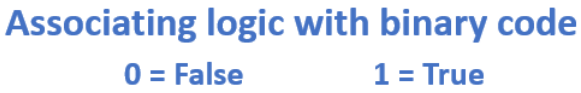


Figure 4.6.1: In binary code a value of zero equates to a false statement and a value of 1 equates to a true statement. (Belford cc 0.0)

There are several types of operators that evaluate to boolean values (true or false) and these include Boolean Operators and Comparison Operators

Boolean Operators

Boolean operators AND, OR and NOT are used in search engines and in python they evaluate to values of the boolean data type. These can be understood by the Truth Table and these can be applied to nonmathematical data types like strings (words).

AND

In an AND condition both statements must be true for the result to evaluate to true. For example, if you have an exam and a calculator you can use the calculator on the exam, but if you have an exam and no calculator, or a calculator and no exam, or neither a calculator or an exam, you can't use a calculator on an exam.

Truth Table for AND operator

Operand 1	Operand 2	Result
TRUE	TRUE	TRUE
TRUE	FALSE	FALSE
FALSE	TRUE	FALSE
FALSE	FALSE	FALSE

Operand 1	Operand 2	Result
1	1	1
1	0	0
0	1	0
0	0	0

Figure 4.6.1: Copy and Paste Caption here. (Copyright; author via source)



Login with LibreOne to run this code cell interactively.

If you have already signed in, please refresh the page.

Login

```
have_calculator=True
have_exam = True
print(f"I have calculator {have_calculator} \n and I have exam {have_exam} \n so can
```

```
have_calculator=False
have_exam = True
#print("Can I use calculator on exam?",have_calculator and have_exam)
print(f"I have calculator {have_calculator} \n and I have exam {have_exam} \n so can


have_calculator=True
have_exam = False
#print("Can I use calculator on exam?",have_calculator and have_exam)
print(f"I have calculator {have_calculator} \n and I have exam {have_exam} \n so can


have_calculator=False
have_exam = False
#print("Can I use calculator on exam?",have_calculator and have_exam)
print(f"I have calculator {have_calculator} \n and I have exam {have_exam} \n so can
```

```
Hello world!
```

OR

For an OR value to be true either one or both conditions are true, and this only evaluates false if both conditions are not met.

Truth Table for OR operator

Operand 1	Operand 2	Result
TRUE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	TRUE	TRUE
FALSE	FALSE	FALSE

Operand 1	Operand 2	Result
1	1	1
1	0	1
0	1	1
0	0	0

Figure 4.6.1: Copy and Paste Caption here. (Copyright; author via source)

Login with LibreOne to run this code cell interactively.

If you have already signed in, please refresh the page.

Login

```
have_pen = True
have_pencil = True
print(f"I have a pen {have_pen} \n and I have a pencil {have_pencil} \n so can I writ
```

```
have_pen = False
have_pencil = True
print(f"I have a pen {have_pen} \n and I have a pencil {have_pencil} \n so can I writ

have_pen = True
have_pencil = False
print(f"I have a pen {have_pen} \n and I have a pencil {have_pencil} \n so can I writ

have_pen = False
have_pencil = False
print(f"I have a pen {have_pen} \n and I have a pencil {have_pencil} \n so can I writ
```

```
Hello world!
```

NOT

The NOT operator negates a boolean value. This is a little confusing, but it takes a true statement and makes it false. For example we can say if it is sunny, then it is not cloudy. This should not be confused with the not equal (!=) comparison operator, that also evaluates to a Boolean value.



Login with LibreOne to run this code cell interactively.

If you have already signed in, please refresh the page.

Login

```
is_sunny = True
is_cloudy = not is_sunny
print("Is it sunny",is_sunny)
print("Is it cloudy", is_cloudy)
```

```
Hello world!
```

Comparison Operators

There are 8 common comparison operators

symbol	meaning	symbol	meaning
<	less than	<=	less than or equal
>	greater than	>=	greater than or equal

==	equal	!=	not equal
is	object identity	is not	negated object identity

Note that in python the equal sign [=] is the assignment operator and assigns a value to a variable.

This page titled 4.6: Boolean (atomic) is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

4.7: String (container)

Overview

- Strings use single, double or triple quotes
  - Triple quote strings span multiple lines
  - Represent characters not numbers ("1 proton" vs 1 "proton"+
  - Strings can be concatenated ("1"+"1" becomes "11")
- Strings are Ordered
  - Strings can be indexed
  - Strings can be reverse indexed
- Strings are immutable
  - Can not change a string (does not support item reassignment)
  - Can be reassigned by slicing and concatenation

Formatting Strings

When you print a string there are a variety of format options. I personally tend to use the f-string, but you need to be prepared to read code that uses any of the options.
Modulo (%) method
.format() method
f-string method

String Methods

String methods are functions that are built into the string object class. The following are some string methods See Python.org string section for a comprehensive list of string methods. Note, these should be quickly gone through and then used as references, and some of these involve classes like lists which are covered later in this section.

- str.capitalize()
  Lets analyze the code below. In the first line we assign the string "chemistry class" to a variable called name. Note the string is in parenthesis, but the name is not. Now we want to print the variable name but capitalize the first letter of the first word, and so we can use the capitalize method, which is a function of the class string, by typing .capitalize() after the string variable name. Note print is a function and so we place the argument (name) in it, and capitalize is a method (function of the class string), and so it also needs parenthesis, but it operates on the variable it is appended to.

```
name = "chemistry class"

print(name.capitalize())
```

run | restart | restart & run all

If you wanted to capitalize the first letter of every word you would use the title method that is demonstrated below.

- str.find()
  note, this method finds the index number of the string, which is an ordered list. In python index numbers start with zero, and so the word "best" starts at the 17th index position of the string.

```
text = 'Chemistry is the best course'

print(text.find('best'))
```

run | restart | restart & run all

- str.join (iterable)
  note the variable text is a list of strings and we cover lists in the next section. Here we are using the space string to join each of the words of the list into a stingle string. After you run the following, remove the blank spot in space, that is, convert " " to "" and then rerun the code.

```
text = ['Chemistry', 'is', 'the', 'best', 'class']
space = " "

print(space.join(text))
```

run restart restart & run all

- str.lstrip ([chars])
  striping characters can be very important, especially if you import a file with a bunch of characters you do not want. Data transmitted over the web is usually a string and you often need to convert it to a number by removing the quotes. That is, 1 is an integer and "1" or '1' are strings.

```
text = ",,,,,Hello world"

x = text.lstrip(",")

print(x)
```

run restart restart & run all

- str.replace ("old", "new")

```
text = 'Ozone Oxygen Butane'

new_text = text.replace('Butane', 'Hydrogen')

print(new_text)
```

run restart restart & run all

- str.rsplit(separator, max split)

Converts string into list of strings from the right, with the max split number indicating how many items in the list are created to the right.

```
text = "Chemistry,Physics,Calculus"

x = text.rsplit(",", 2)

print(x)
```

run restart restart & run all

- str.rstrip([chars])

```
text = "Pythonnnnnn"
print(text.rstrip('n'))
```

run | restart | restart & run all

- str.split(sep=None, maxsplit=-1)

```
text = 'This programming langauge is cool'

x = text.split(None,-1)
print(x)

x = text.split(None,1)
print(x)

x = text.split(None,2)
print(x)
```

run | restart | restart & run all

- str.title()

```
text1 = 'python programming class'
print(text1.title())

text2 = 'pyTHoN proGraMMinG clAsS'.title()
print(text2.title())
```

run | restart | restart & run all

This page titled 4.7: String (container) is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

4.8: List (container)

```
molar_Mass=['hydrogen',1.00784,'helium',4.002602,'lithium', 6.941, 'beryllium', 9.012182]
```

Figure \(\PageIndex{1}\):The variable molar_Mass is a list. This image was taken from an IDE that displayed strings in red and floating numbers in green (Belford CC0.0)

Overview

- Use Brackets
  - Can have multiple types of objects
  - commas separate objects
- Lists are ordered
  - Support Indexing and Slicing
- Lists can be nested
- Lists are mutable (can be modified)
- Lists are an object class with methods

Methods

- append
- count
- extend
- insert
- pop
- remove
- reverse
- sort

```
molar_Mass=['hydrogen',1.00784,'helium',4.002602,'lithium', 6.941, 'beryllium', 9.012
print('List Created')
```

run | restart | restart & run all

```
  Hello world!
```

```
len(molar_Mass)
```

run | restart | restart & run all

```
          hello world
```

```
molar_Mass=['hydrogen',1.00784,'helium',4.002602,'lithium', 6.941, 'beryllium', 9.012
print(f"There are {len(molar_Mass)} items in the list molar_Mass")
```

run | restart | restart & run all

```
Hello world!
```

print every other item starting with first item (index number = 0)

```
molar_Mass[0::2]
```

run | restart | restart & run all

```
hello world
```

print every other item starting with first item (index number = 0) with a range of up to 4 (it does not print item with index =4)

```
molar_Mass[0:4:2]
```

run | restart | restart & run all

```
hello world
```

Print print every other item starting with second item (index number = 1)

```
molar_Mass[1::2]
```

run | restart | restart & run all

```
hello world
```

Create three lists

```
name_halogen=['fluorine', 'chlorine', 'bromine', 'iodine', 'astatine','tennessine']
z_halogen=[9,17,35,53,85,117]
a_halogen=[18.99840316,35.45,79.90,126.9045,209.98715,294.211]
print('List Created')
```

run | restart | restart & run all

```
Hello world!
```

Print two of the lists, note they are printed as two separate lists

```
print(name_halogen,z_halogen)
```

run | restart | restart & run all

```
hello world
```

Combine the two lists into one and print it

```
print(name_halogen+a_halogen)
```

run | restart | restart & run all

```
          hello world
```

create a new list and print it

```
halogen_masses=name_halogen+a_halogen
print(halogen_masses)
```

run | restart | restart & run all

```
          hello world
```

```
print(name_halogen)
```

run | restart | restart & run all

```
  Hello world!
```

```
popped_halogen=name_halogen.pop()
print(name_halogen)
print(popped_halogen)
```

run | restart | restart & run all

```
  Hello world!
```

```
name_halogen.append(popped_halogen)
print(name_halogen)
```

run | restart | restart & run all

```
          hello world
```

```
remove_bromine=name_halogen.pop(2)
print(name_halogen)
print(remove_bromine)
name_halogen.insert(2,remove_bromine)
print(name_halogen)
```

run | restart | restart & run all

```
          hello world
```

```
halogen=[name_halogen, z_halogen,a_halogen]
print(f'{halogen}\n')

print(f'{halogen[0]}\n')
print(f'{halogen[0:][1]}\n')
print(f'{halogen[0:][0]}\n')
print(f'{halogen[0][0:]}\n')
print(f'{halogen[2][0]}\n')
```

run | restart | restart & run all

```
          hello world
```

```
random_list=[22,41,16,444,224]
print(random_list)
random_list.reverse()
print(random_list)
random_list.sort()
print(random_list)
```

run | restart | restart & run all

```
  Hello world!
```

This page titled 4.8: List (container) is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

4.9: Dictionary (container)

```
z_alkali_dict={"Lithium":3,"Sodium":11,"Potassium":19, "Rubidium":37, "Caesium":55,"Francium":87}
```

Figure \(\PageIndex{1}\): Copy and Paste Caption here. (Copyright; author via source)

Overview

Use KEY:VALUE Pair

- Use Curly Brackets
- Mutable
- Unordered
  - use KEY:VALUE pair separated by commas
- Values can be
  - lists
  - other dictionaries
  - strings, numbers
- Are an object class with methods

Simple Dictionary

```
z_alkali_dict={"Lithium":3,"Sodium":11,"Potassium":19, "Rubidium":37, "Caesium":55,"F
#call the dictionary
z_alkali_dict
```

run | restart | restart & run all

```
  Hello world!
```

Dictionary of lists

First, create the lists. Note, nothing happens as you have to print something, add the following to the end of the script print("list created")

```
name_halogen=['fluorine', 'chlorine', 'bromine', 'iodine', 'astatine','tennessine']
z_halogen=[9,17,35,53,85,117]
a_halogen=[18.99840316,35.45,79.90,126.9045,209.98715,294.211]
symbol_halogen=["F","Cl","Br","I","At","Ts"]
```

run | restart | restart & run all

```
  Hello world!
```

Make the dictionary and call it

```
hal_dict={'name':name_halogen,"atomic_number":z_halogen,"atomic_weight":a_halogen}
#call it
hal_dict
```

run | restart | restart & run all

```
        hello world
```

Use Key to Call Values

Here we can use a key to get its value

```
print("the names of the halogens are ", hal_dict['name'])
```

run | restart | restart & run all

```
          hello world
```

After running the above code, modify it to get the atomic weights of the halogens

Use list index numbers

```
print("The heaviest halogen is ", hal_dict['name'][-1])
print("The heaviest halogen is", hal_dict['name'][-1],", it weighs", hal_dict['atomic_
```

run | restart | restart & run all

```
  Hello world!
```

Methods

```
hal_dict['name'][2].upper()
```

run | restart | restart & run all

```
          hello world
```

Print the keys

```
hal_dict.keys()
```

run | restart | restart & run all

```
  Hello world!
```

Print the values

```
hal_dict.values()
```

run | restart | restart & run all

```
          hello world
```

Print the

```
print(hal_dict)
```

```
print("\n 5*' * '\n")
hal_dict.items()
```

run | restart | restart & run all

```
  Hello world!
```

Methods

This page titled 4.9: Dictionary (container) is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

4.10: Tuple (container)

Definition

A tuple is "container" type of data structure, it is an immutable ordered list and is enclosed in parenthesis, and can contain elements of different data types.(I1, I2, I3...In)

Origin of "Tuple"

The word "tuple" is derived from the sequence of Latin-based number prefixes used to describe multiples:

- Single (1)
- Double/Couple (2)
- Triple (3)
- Quadruple (4)
- Quin**tuple** (5)
- Sex**tuple** (6)
- Sep**tuple** (7)
- Oc**tuple** (8)
- etc.

Overview

- Example of tuple embedded in a list

[('fluorine', 9, 18.99840316), ('chlorine', 17, 35.45), ('bromine', 35, 379.9), ('iodine', 53, 126.9045), ('astatine', 85, 209.98715), ('tennessine', 117, 294.211)]

- Use Parenthesis
  - objects separated by commas
- Are Immutable and ordered
  - can not change values or append
    - maintain data integrity
- can be:
  - indexed
  - concatenated
  - nested
  - sliced
- Object Class with Methods

Methods

This page titled 4.10: Tuple (container) is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

4.11: 3.11 Set (container)

Overview

- Use curly brackets like dictionaries
  - objects separated by commas
- Are unordered
- Immutable (items are unchangeable, but can be removed and added)
- Show unique elements

This page titled 4.11: 3.11 Set (container) is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

CHAPTER OVERVIEW

5: Data Operators

5.1: Introduction
5.2: Arithmetic Operators and Numbers
5.3: Arithmetic Operators and Strings
5.4: Assignment Operators
5.5: Comparison Operators
5.6: Logical Operators
5.7: Identity Operators
5.8: Membership Operators

IoST map for 4: Data Operators

Section	IoST Source
4.1: Introduction	5.3.1: Introduction
4.2: Arithmetic Operators and Numbers	5.3.2A: Arithmetic Operators and Numbers
4.3: Arithmetic Operators and Strings	5.3.2B: Arithmetic Operators and Strings
4.4: Assignment Operators	5.3.3: Assignment Operators
4.5: Comparison Operators	5.3.4: Comparison Operators
4.6: Logical Operators	5.3.5: Logical Operators
4.7: Identity Operators	5.3.6: Identity Operators
4.8: Membership Operators	5.3.7: Membership Operators

This page titled 5: Data Operators is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

5.1: Introduction

Operators are symbols in the Python language that perform operations on one or more variable (operand) to produce a result. Common mathematical operations like + for addition are operators, but the result can depend on the type of data they operate on (1+1=2, but "1"+"1"="11", as the former are integers while the latter are string type)

Types of Operators

1. Arithmetic Operators
2. Comparison Operators
3. Assignment Operators
4. Logical Operators
5. Identity Opeators
6. Membership Operators
7. Bitwise Operators

It should be noted that the result of an operator is based on the data type it operates on. There can also be some confusion between a function and an operator, as a function can be built to do an operation. A function has parameters and so a function call must have parenthesis (). The following code does the same thing, with the first line being the arithmetic power operator, and the second the power function, which is a built-in function.



Login with LibreOne to run this code cell interactively.

If you have already signed in, please refresh the page.

Login


```
x=5**3
y=pow(5,3)
print(x)
print(y)
```

```
Hello world!
```


This page titled 5.1: Introduction is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

5.2: Arithmetic Operators and Numbers

There are 7 arithmetic operators in Python that work for integer and float data types. Addition and multiplication also work for strings.

Operator	Name	Syntax
+	Addition	X+Y
-	Subtraction	X_Y
*	Multiplication	X*Y
**	Exponentiation	X**Y
/	Division	X/Y
%	Modulus	X%Y
//	Floor Division	X//Y

Floor Division (//) and Modulus (%)

Floor Division and the modulus at first will often confuse students, so let's refresh your elementary school long division, and note that long division gives the quotient and the modulus gives the remainder.

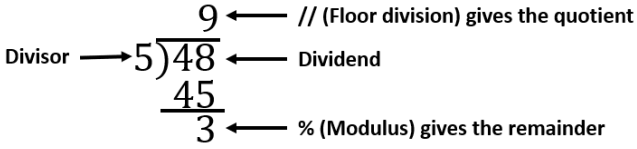


Figure 5.2.3: Floor division gives the quotient and the modulus gives the remainder. (Copyright; Belford CC0.0)

Note, this is commonly written as $dividend \div divisor$ *or* $\frac{dividend}{divisor}$

This page titled 5.2: Arithmetic Operators and Numbers is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

5.3: Arithmetic Operators and Strings

Concatenation

Merriam-Webster's dictionary defines concatenate to mean "linked together. The addition operator concatenates strings, so 1+1=11 if you are adding strings.

Concatenation Subscripts and Superscripts

One way to create subscripts and superscripts in strings is to use unicode and concatenate. The following table gives unicode subscript and superscript values that you may often needc

character	0	1	2	3
superscript	U2070	U00B9	U00B2	U00B3
subscript	U2080	U2081	U2082	U2083

```
#We will now use concatenation to create the formula for sulfuric acid.
sulfuric_acid="H"+"\u2082"+"SO"+'\u2084'
sulfate_ion = "SO"+"\u2084\u207B\u00B2"
print(sulfuric_acid, "and", sulfate_ion)
```

run | restart | restart & run all

```
Hello world!
```

Note, we can make variables for the unicode

```
s2,s3,s4,s5,s6,s7,s8,s9="\u2082","\u2083","\u2084","\u2085","\u2086",\
                        "\u2087","\u2088","\u2089"
ammonium_phosphate="(NH"+s4+")"+s3+"PO"+s4
print(ammonium_phosphate)
```

run | restart | restart & run all

```
Hello world!
```

? Exercise \(\PageIndex{1}\)

You want the following code to display

```
print(n_nonane)
```

run | restart | restart & run all

$CH_3(CH_2)(CH_2)(CH_2)(CH_2)(CH_2)(CH_2)CH_3$

Figure \(\PageIndex{1}\): Alter the code below to produce this output

The following code does not work. comment out the print statement and uncomment the lines above and below it, and then complete the expression for x so that the code works.

```
#the following line assigns subscripts to the varibles using unicode
s2,s3,s4,s5,s6,s7,s8,s9="\u2082","\u2083","\u2084","\u2085","\u2086",\
                        "\u2087","\u2088","\u2089"
#x=
n_nonane="CH"+s3+7*"(CH"+s2+")"+"CH"+s3
#n_nonane="CH"+s3+7*x+"CH"+s3
print(n_nonane)
```

run | restart | restart & run all

```
Hello world!
```

Answer

x = "(CH"+s2+")"

Concatenation and Web Calls

This can be can be very useful if you want to add a variable to a string. For example, the following code takes the variable "compound_name" and adds it to a web address, which gets you the molecular weight of a chemical from the Pubchem online database.

Note

The following code uses the requests package which has already been installed on the LibreTexts Jupyter Hub. If you are running your own kernel you need to run the following code:
import requests
If you do not have the requests package installed, you need to install it from the Python Package Index. This can be done in either the command line or in your IDE (Thonny)

OK, lets look at concatenation in action

```
compound_name = "aspirin"
uri = ("https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/name/"
       + compound_name +"/property/MolecularWeight/txt")
print(uri)
```

run | restart | restart & run all

```
Hello world!
```

Run the above code and then click the link it produces. You get the molar mass of aspirin 180.16 (g/mol).
Now, replace the word aspirin with the word caffeine, rerun the code, click the new link, and you get 194.19, the molar mass of caffeine

How did this work?

Concatenation can be very useful for obtaining information off the web. In this example we used concatenation with the Pubchem PUG REST API (Power User Gateway Representational State Transfer Application Program Interface) to obtain the molar mass of a chemical. We are going to do this for the common name, but with more effort we could also do this for synonyms. An API is like a GUI (Graphical User Interface) in that the API allows interaction between two computers while the GUI allows a person to interact with a computer. When you navigate to a webpage the web address in the hyperlink is a URL (Universal Resource Locater). Just as a webpage has a URL, you can also address items within a database by a URI (Universal Resource Identifier). The W3C (World Wide Web Consortium) set forth a RDF (Resource Descriptive Framework) that involves an RDF triple (subject, predicate, object) relationship and we are taking advantage of that to obtain the molar mass by concatenating a string into a URI. OK, it looks complicated, but it is simple. The subject is a molecule (which we will input and assign to a variable), the predicate is a property of a molecule (its molar mass, but we could seek other properties) and the object is the value of the molar mass for the molecule, which is the result of the query
First, place the following link into a new tab of your browser:
pubchem.ncbi.nlm.nih.gov/rest/pug/compound/name/aspirin/property/MolecularWeight/txt
Note, compound words like hydrochloric acid need a dash (hydrochloric-acid).

String Multiplication

You can also multiply strings. Predict the outcome of the following code, and then run it. Did it do what you expected?

```
spacer=20*"*"
print(spacer)
```

run | restart | restart & run all

```
Hello world!
```

Predict the output of the following code

```
print(25%2)
print(25*"2")
print(25//2)
```

run | restart | restart & run all

```
Hello world!
```

d

This page titled 5.3: Arithmetic Operators and Strings is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

5.4: Assignment Operators

The Assignment Operator (=)

Assignment operators assign a value to a variable in python. The equal sign is the simple assignment operator, and it assigns the value on the right hand side to a variable on the left hand side. The value on the right hand side can any data type (integer, float, boolean)

Predict the output of the following code, and then run it. Note, the type function indicates the data type of a variable. In Python you can change variable data types with the assignment operator

```
a="b"
print(a)
print(type(a))
print(3*" *! ")
a=3
print(a)
print(type(a))
```

run | restart | restart & run all

```
  Hello world!
```

Shorthand Assignment Operators

Shorthand assignment operators allow you to assign a variable and do an arithmetic operation at the same time

Plus Equal (+=)

Allows you to add a value to a variable and reassign the variable the new value. This can be very useful if a variable is incrementally increasing in value. Predict the output before running the code

```
a=10
print(a)
a=a+5
print(a)
```

run | restart | restart & run all

```
  Hello world!
```

Now try this

```
a=10
print(a)
a+=5
print(a)
```

run | restart | restart & run all

```
          hello world
```

the += operation also works on strings

```
a="hydrogen"
print(a)
a+=" helium"
print(a)
```

run | restart | restart & run all

```
hello world
```

Minus Equal (-=)

Allows you to remove a value from a number. Note, you can mix integers with floating numbers, but can not do this for strings. When you do an operation with an integer and a float, the value turns to a float

```
a=10
print(a)
a-=5
print(a)
print(type(a))
a-=1.0
print(a)
print(type(a))
a+=1
print(a)
print(type(a))
```

run | restart | restart & run all

```
Hello world!
```

Note, the -= does not work for strings

```
a="10"
print(a)
a-="5"
print(a)
```

run | restart | restart & run all

```
Hello world!
```

Multiple equal (*=)

```
a=2
print(a)
a*=3
print(a)
```

run restart restart & run all

```
hello world
```

Divide Equal (/=)

```
a=2
print(a)
print(type(a))
a/=3
print(a)
print(type(a))
```

run restart restart & run all

```
Hello world!
```

Modulus Equal (%=)

```
a=25
print(a)
a%=2
print(a)
b=24
print(b)
b%=2
print(b)
```

run restart restart & run all

```
Hello world!
```

Floor Equal (//=)

```
a=25
print(a)
a//=2
print(a)
b=24
print(b)
b//=2
print(b)
```

run restart restart & run all

```
hello world
```

This page titled 5.4: Assignment Operators is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

5.5: Comparison Operators

The output of a comparison operator is a Boolean value

Operation	Meaning
<	less than
<=	less than or equal
>	greater than
>=	greater than or equal
==	equal
!=	not equal
is	object identity
is not	negated object identity

Exercise

Predict the output of the following code

```
a=2
b=3
print(a==b)
print(3*" * ")
print('a!=b')
```

run | restart | restart & run all

```
hello world
```

? Exercise \(\PageIndex{1}\)

Why did the third print statement not say True, and change the code so it says True

Answer

The code was a string, not a comparison operation. To make it a comparison operation you need to remove the quotes.

Predict the output of the following code

```
a=2
b=3
c=2
print(c is a)
print(c is not a)
```

run | restart | restart & run all

```
Hello world!
```

This page titled 5.5: Comparison Operators is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

5.6: Logical Operators

Logical operators are also called Boolean operators and there are three types. Now it needs to be realized that Boolean data types are true or false, and other operators (like comparison operators) produce Boolean output.

Operator	Description	Example
and	returns True if both statements are true	x<5 and x>1
or	returns True if either statement is true	x<5 or x>1
not	returns True if result is false	not(x<5 and x>1)

or operator



Truth Table for OR operator

take exam	do homework	need calculator
TRUE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	TRUE	TRUE
FALSE	FALSE	FALSE

take exam	do homework	need calculator
1	1	1
1	0	1
0	1	1
0	0	0

Figure \(\PageIndex{3}\): Truth table for OR statement (Belford cc 0.0)

```
a=25
x=(a>10 or a<5)
print(a>10 or a<5)
print(x)
print(type(x))
a=7
y=(a>10 or a<5)
print(y)
print(type(y))
```

run | restart | restart & run all

```
Hello world!
```

and Operator



Truth Table for OR operator

have juice	have glass	have glass of juice
TRUE	TRUE	TRUE
TRUE	FALSE	FALSE
FALSE	TRUE	FALSE
FALSE	FALSE	FALSE

have juice	have glass	have glass of juice
1	1	1
1	0	0
0	1	0
0	0	0

Caption here. (Belford cc 0.0)

Figure \(\PageIndex{4}\): Copy and Paste

```
a=25
x=(a>5 and a<10)
print(a>5 and a<10)
print(x)
print(type(x))
a=7
y=(a>5 and a<10)
print(y)
print(type(y))
```

run | restart | restart & run all

```
Hello world!
```

not Operator

```
a=25
print(not(a<10 and a>5))
a=7
print(not(a<10 and a>5))
```

run | restart | restart & run all

```
hello world
```

This page titled 5.6: Logical Operators is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

5.7: Identity Operators

These are used to see if two objects are the same, that is, if they have the same memory location

Operator	Description	Example
is	return True if both objects are same	x is y
is not	return True if objects are different	x is not y

```
x=25
y=25
z=y
print(x is y)
print(x is z)
```

run | restart | restart & run all

```
hello world
```

Predict the output for this code

```
x=25
y=25
z=y
y=3
print(x is y)
print(x is z)
```

run | restart | restart & run all

```
hello world
```

This page titled 5.7: Identity Operators is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

5.8: Membership Operators

This allows us to see if something is present in an object

Operator	Description	Example
in	returns True if item is in object	x in y
not in	returns False if item is in object	x not in y

```
halogens = ["Fluorine", "Chlorine", "Bromine", "Iodine", "Astatine"]
y="oxygen" in halogens
print(y)
print(type(y))
x="chlorine"
z="Chlorine"
print(x in halogens)
print(z in halogens)
```

run | restart | restart & run all

```
  Hello world!
```

```
halogens = ["Fluorine", "Chlorine", "Bromine", "Iodine", "Astatine"]
y="oxygen" not in halogens
print(y)
print(type(y))
x="chlorine"
z="Chlorine"
print(x not in halogens)
print(z not in halogens)
```

run | restart | restart & run all

```
          hello world
```

This page titled 5.8: Membership Operators is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

CHAPTER OVERVIEW

6: Introduction to Linux

Topic hierarchy

6.1: What is an Operating System?
6.2: Introduction to Linux
6.3: Git BASH
6.4: Linux Commands

Transclusion Map

Section	Original Source
6.1: What is an Operating System	1.5.3: What is an Operating System
6.2: Introduction to Linux	3.1.1: Introduction to Linux
6.3: Git BASH	1.7.4: Git BASH
6.4: Linux Commands	3.1.4: Linux Commands

6: Introduction to Linux is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

6.1: What is an Operating System?

As the name implies, the operating system (OS) is the software that operates you computer. Common operating systems are Unix, Windows, Mac OS and Linux Distributions. The Raspberry Pi OS is a Linux distribution based on the Debian distribution, where Linux is the kernel, the part of the operating system that interacts with both hardware and software applications. So when you down load and install software or hardware drivers (like those for your printer) you need to choose a package that is appropriate for your OS. 6.1.1 shows the key components of an operating system

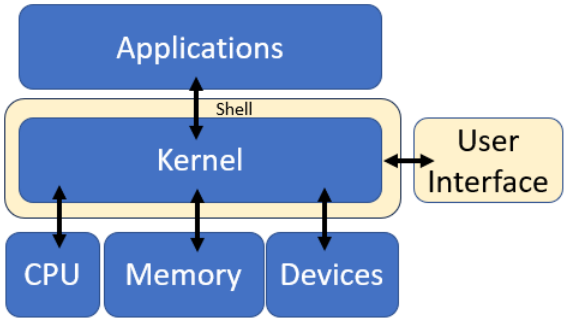


Figure 6.1.1: Key components of the operating system. (Belford CC-BY)

For the Raspberry Pi OS uses a Linux kernel and a bash (Bourne Again Shell) shell, and you can provide bash commands to the kernel through the terminal Command Line (user) Interface. Linux commands are really bash commands, but we will call them Linux commands because bash is so commonly used in Linux OS distributions. Some of the key processes the Linux kernel performs are:

- Memory Management
- Process Management
- File System Management
- Device Management
- Software Application Management
- User Interface
- Security and System Calls
- Networking

The Raspberry Pi OS is specifically designed for the PI SOC (System on a Chip) and has commands that are unique to the RPi hardware

6.1: What is an Operating System? is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

6.2: Introduction to Linux

Introduction

The Linux ecosystem is the largest open source software project in the world with the Git repository of the Linux kernel having 27.8 million lines of code in 2020. It is used in everything from embedded devices, to mobile devices, Raspberry Pis and personal computers to mainframes and supercomputers, in fact in January of 2023 one hundred percent of the top 500 supercomputers run Linux kernels. In 1983 Richard Stallman founded the Free Software Foundation (fsf.org) and created the GNU (GNU Not Unix) Public License. In 1991 Linus Torvalds (at age 21) started developing the Linux system, which is similar to Unix and in 1992 released it under a GNU GPL license.

Linux is the kernel of an operating system, the part of the operating system that connects software applications to the computer hardware. It manages the memory, processors, input/output devices and the file system. Combining the Linux kernel with other GNU software results in a complete open source operating system, and there are many variations of Linux operating systems. The Raspberry Pi is based on Debian Linux

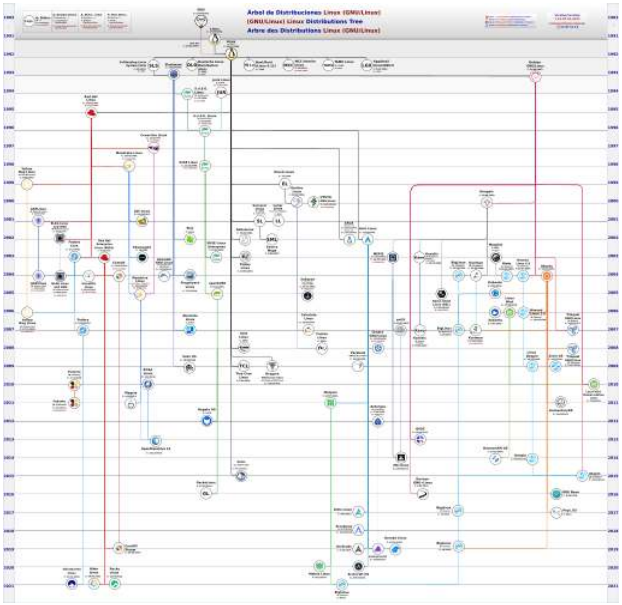


Figure 6.2.1: Linux distribution tree, if you click it twice you will open an interactive version. VARGUX, CC BY-SA 4.0, via Wikimedia

Debian

Debian was one of the oldest Linux distributions and the article "The History of Debian Linux Disribution" is a good place to start.

The three most recent versions for the Raspberry PI OS are:

- Debian 10.0 (Buster) - released July 2019
- Debian 11.0 (Bullseye) - released August 2021
- Debian 12.0 (Bookworm) - released December 2023

Debian is freely available through a GNU/GPL license and there are over 51,000 installable packages.

Raspberry Pi Imager

Although there are many options for installing the Raspberry Pi OS you will probably want to use the imager apt. Instructions can be found here and we will do the next lesson: 6. Getting Started with IOT.

1. Download Imager
2. Install Raspberry Pi OS
3. Start RPi for First time

This page titled 6.2: Introduction to Linux is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

6.3: Git BASH

6.3: Git BASH is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

6.4: Linux Commands

Directory (Folder) Commands

(transclusion link)

cd

cd <path> directs to a specific directory in the tree

note, there must be a space after the cd

cd switches

the following switches work on cd

```
cd /home/pi
cd   #by itself takes you home
cd ~  navigate to home directory
cd /  navigate to root directory
cd .. go up one level
cd - go back one level
```

cp

cp - copy directory

We can copy a directory to a new location by using the command:

- **cp directory_name destination :** Copies directory (or file) to a new location

> Note
>
> To include the subfolders of a directory, we need to add **-r** (recursive)

```
cp /tmp/test/subdir2 /tmp/test/subdir1 -r
cd subdir1
ls
```

Output:

```
pi@raspberrypi:/tmp/test $ cp /tmp/test/subdir2 /tmp/test/subdir1 -r
pi@raspberrypi:/tmp/test $ cd subdir1
pi@raspberrypi:/tmp/test/subdir1 $ ls
subdir2  test_file_1.txt
```

Relative path

The previous examples that we worked on used some ***relative*** paths. Working with a relative path means that the place where you go depends on your current working directory.

The "etc" directory is a file that is located directly inside the root of the file system. If you were to try and ***cd*** to the "etc" folder while you are in the root directory, it will work.

```
cd /
pwd
cd etc
pwd
```

However, if you are not in your root directory, that won't work.

```
cd
pwd
cd etc
pwd
```

You will get an error saying "No such file or directory".

Changing the working directory will affect where you start. The path of the directory you want to access only makes sense while it is **relative** to your working directory.

Absolute path

Absolute commands have the same effect no matter what your current working directory is. From our previous examples, the **cd** command is an **absolute** one. When you run it on its own, you go straight to your home directory. Another previous example is the **cd /** command. It directly switches you to the root directory when you run it on its own.

Note: Any path starting with **/** is an **absolute** path. When you want to switch to a directory and start your path with the /, it is the equivalent of *"going to the root directory and then to the folder that comes after the slash"*.

```
cd
pwd
cd /etc
pwd
```

find

find works on both directories and files.

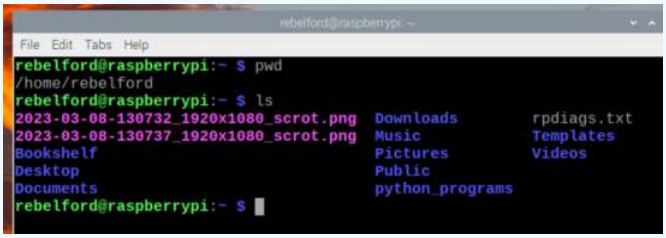
ls

With the last command, we created two new subdirectories. We can look at the list with the command:

- **ls : l**ist

```
ls
```

Output:



ls switches

```
ls
ls /home/pi
ls -1     (one, to make a single column)
ls -l      (long, to list info)
ls -l -a   (long and all, to include hidden files
ls ../   (reaches up one level)
```

mkdir

the following commands show you where you are, make a new directory, show you the new directory and move you to the new directory

```
pwd
mkdir new_directory
ls
cd new_directory
ls
```

rm - Remove Directory

for empty directories

rm switches

```
rm -d directory_name
rm -r recursive and removes subfolders
rm -rf recursive and forced
```

for directories with files in them (-r is recursive, and removes contents of directory)

```
rm -r directory-name
```

The -r is a switch making the command recursive

to do it without asking for confirmation

```
rm -rf directory_name
```

The -f switch "forces" the action

BE VERY CAREFUL WHEN USING rm -r or rm -rf as you delete everything, including subfolders.

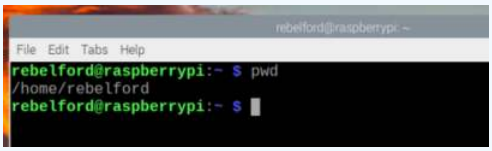
pwd

Identifies Current Directory

rebelford@raspberrypi:~ $ pwd

```
pwd
```

Output:



tree

```
tree
```

This shows the file structure from where you are

File Commands

(transclusion link)

cat

We can view the content of the file on the terminal by using the command:

The following command allows you to see what your operating system is

- **cat file _name**: Displays content of a file (con**cat**enate)

```
cat ls_output.txt
```

cat does not support keyboard scrolling

```
 cat /etc/os-release
```

echo

We can display some text that we type into the command line, not necessarily reading it from a file, by using the command:

- **echo "text"**: printing text on the terminal

```
echo "display new text"
```

The **echo** command prints its **parameters**

echo can create a new file when used with a redirect and place its content in the file

```
echo "insert this text into new file" > newfile.txt
```

find

find a file in a directory

note the "i" in iname makes it case insensitve

```
find /home/path -iname filename.extension
find /home/path -filename *.extension
```

grep

To find a string in a file, we use the command:

- **grep string_content file_name : Gl**obal **R**egular **E**xpression **P**rint

head

To display the beginning of a file, we use the command:

- **head file_name :** Displays the beginning (head) of a file

```
head -n1 ls_output.txt       (displays first 1 line of ls_output.txt)
head -n2 ls_output.txt       (displays first 2 lines of ls_output.txt)
```

less

more

displays content of a file one page at a time

mv

Let's move one of the test files that we created previously to one of our subdirectories. This is done b the command:

- **mv file_name directory_name : mo**ve a file to a directory

rm

- **rm file_name : re**move

```
rm test_file_2.txt
```

sed

sed- Stream editor

sed [intial word/new word] filename

For command testing purposes, let's create a new file with the following text:

```
echo "Course number 1, course number 2, course number 3, course number 4"  > test_fil
```

To find and replace something in a file, we can use the command:

```
sed 's/course/class/' test_file_3.txt
```

If we now check the content of the file using **cat**:

```
cat test_file_3.txt
```

Output:

Every word "course" is replaced by the word "class":

"Class number 1, class number 2, class number 3, class number 4"

screenshot to be added

sort

```
sort file_to_sort.txt
```

sort switches

Sorting for Column Number

Sorting can be done with file content that has more than one column. Let's create a new text file entitled "file_to_sort_columns.txt" in which there are names and ages (2 columns).

Let's organize them in ascending order. For that, we use the keyword **–k** in the command and **–n** for numerical sorting (Since there are two columns, **2** is used with **-n**)

```
sort –k 2n file_to_sort_columns.txt
```

Output:

to be added

Checking the Sorting of the file

If we want to check if the file was sorted, we can display the file content using the **cat** command:

```
cat file_to_sort_columns.txt
```

Output:

to be added

Removing duplicates

If there are repeated words in the file, we can use the **sort** command with **-u** option to remove the duplicate, as follows:

```
sort -u file_to_sort_columns.txt
```

Now, we can see that the data is sorted and the repeated items are removed from the output.

to be added

Sorting in Reverse Order

We can sort the file in reverse order (descending order by default) sort by using **-R** option with the **sort** command.

```
sort -R file_to_sort_columns.txt
```

Output:

to be added

tail

Shows end of file

touch

- used to create any type of file with zero size

```
touch demo.txt
```

The creation of files can be done in different ways, one of them is using redirection.

a

wc

word count

wc switches

`wc -l line count`

User Commands

adduser

```
sudo adduser the-name-of-user-you-want
```

password is optional

the new user will be in the home directory

chmod

this command changes the permissions of types of users with regards to access to files and folders

For the use of this command, three things need to be set, who you are setting the permission for, how you are setting it and to what are you setting it

Who we are setting permissions to:

u : owner of the file
g : the file group
o : everyone who is not an owner
a : everyone

How we are setting the permissions:

+ : turns on a permission
- : turns off a permission
= : ignores current permissions and sets new on

What we are setting them to:

r : read
w : write
x : execute
X : special execute for folders

Now let's grant **execute** permissions to the **user** of a fictitious file.

note, you may need to use sudo (if you are not the owner of the file)

```
chmod u+x some_file.txt
```

passwd

allows you to change your password

a user with sudo permission can change a different user's password

Operating System Commands

date

```
date
date +%m-%d-%y
```

df - free space

Display Partitions (free space)

> To make the output easier to understand (in Kilobytes, Megabytes and Gigabytes), use the option **-h** (human-readable) with the **df** command

To know the storage usage of each folder, use the command:

```
df
df -h
```

Output:

```
pi@raspberrypi:~ $ df
Filesystem     1K-blocks    Used Available Use% Mounted on
/dev/root       61291420 7052296  51693724  13% /
devtmpfs          341060       0    341060   0% /dev
tmpfs             472644   18076    454568   4% /dev/shm
tmpfs             189060     856    188204   1% /run
tmpfs               5120       4      5116   1% /run/lock
/dev/mmcblk0p1    258095   50413    207683  20% /boot
tmpfs              94528      24     94504   1% /run/user/1000
pi@raspberrypi:~ $ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        59G  6.8G   50G  13% /
devtmpfs        334M     0  334M   0% /dev
tmpfs           462M   18M  444M   4% /dev/shm
tmpfs           185M  856K  184M   1% /run
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
/dev/mmcblk0p1  253M   50M  203M  20% /boot
tmpfs            93M   24K   93M   1% /run/user/1000
```

du - disk usage

To know how much storage has been used from the disk, use the command:

- **du** : **d**isk **u**sage
- ```
  du
  du/home/pi
  ```

  for a specific directory (home/pi)

Be careful!

By using the **du** command by itself, be ready to get a big number of logs with all the files that exist on your disk!

We will be discussing the options to get precise outputs using the **du** command in the following part.

To know the storage usage of each folder, use the command:

```
du -h --max-depth=1 ./
```

Output:

```
pi@raspberrypi:~ $ du -h --max-depth=1 ./
4.0K    ./Public
324K    ./py-spidev-master
76K     ./.pki
8.0K    ./__pycache__
364K    ./Adafruit_Python_MCP3008
8.0K    ./.cups
4.0K    ./Templates
24K     ./Python_programs
12M     ./Downloads
7.1M    ./QtXBee
4.1M    ./.npm
124M    ./.config
796K    ./Adafruit_Python_DHT
236M    ./.cache
213M    ./Bookshelf
6.5M    ./Pictures
2.2M    ./aws-iot-device-sdk-python
4.0K    ./Music
4.0K    ./Documents
172K    ./.vnc
40M     ./Desktop
928K    ./Adafruit_Python_GPIO
46M     ./.local
6.8M    ./node_modules
4.0K    ./Videos
702M    ./
```

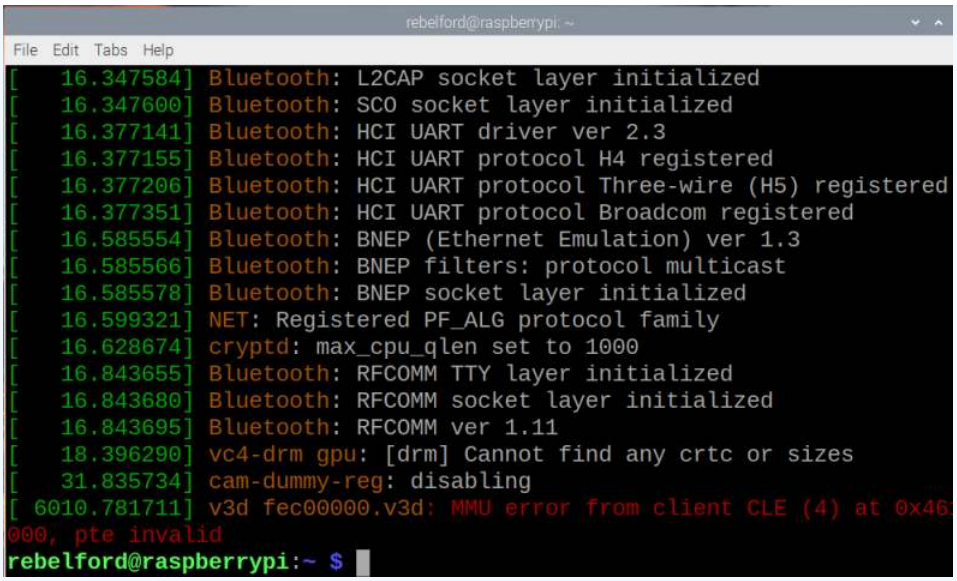
dmesg

displays kernel related messages about hardware, device drivers, initialization and bootup issues

In this activity we will check bootup processes and errors

```
dmesg
```

Output:
```



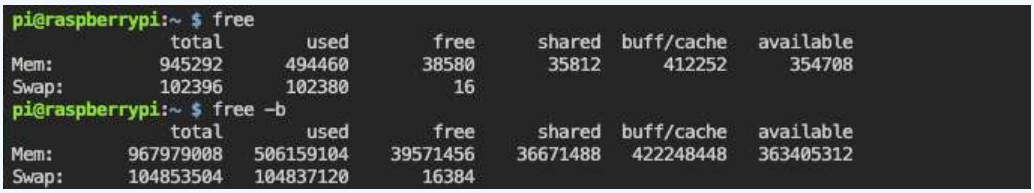
free

To know how much RAM the server has, use the command:

- **free**
- **free -b** : **b** is for bytes
- **free -k** : **k** is for kilo-bytes
- **free -m** : **m** is for mega-bytes
- **free -g** : **g** is for giga_bytes

```
free
free -b
```

Output:



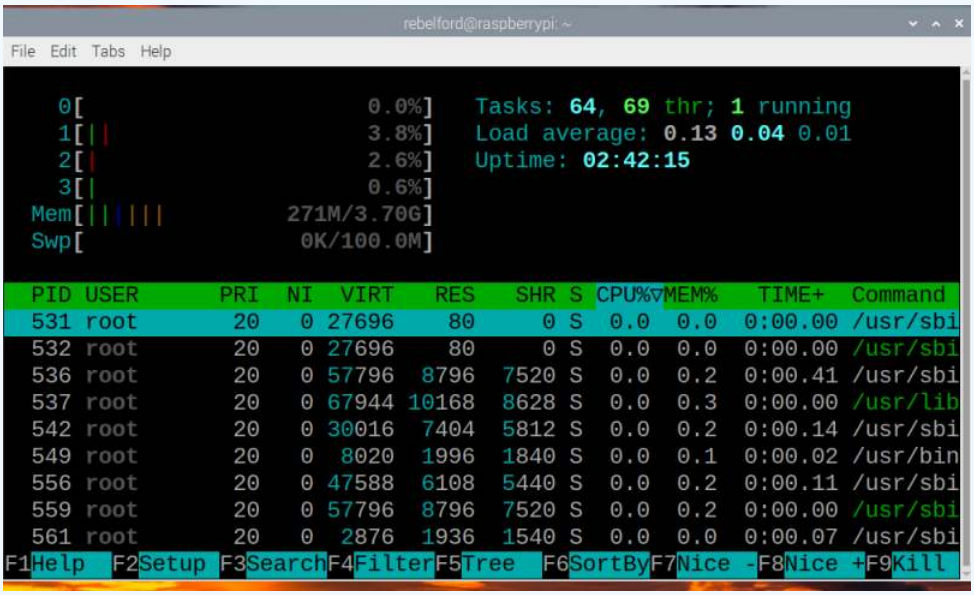
htop

Monitor running processes

In this activity we will check bootup processes and errors (you need to use ctrl c to terminate

```
htop
```

Output:



kill/killall

kills a specific process by its process id, which you get with the ps command

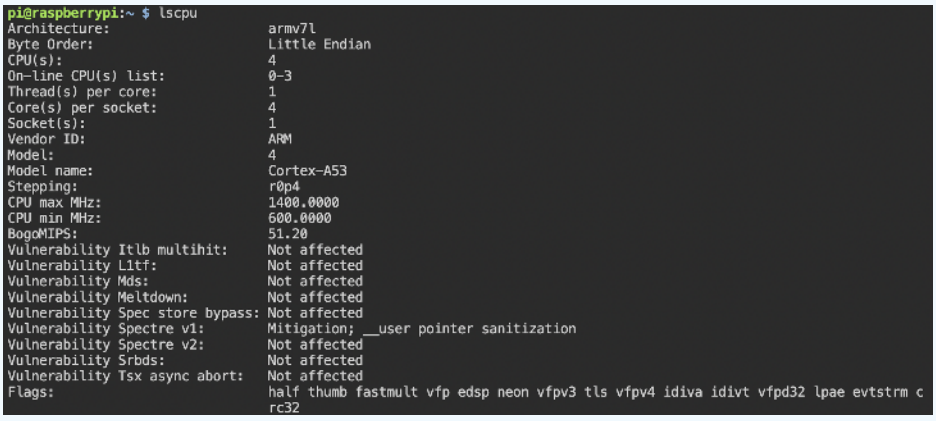
```
ps
kill ###(the process ID of the process you want to kill)
```

lscpu

list number of cpus

```
lscpu
```

Output:



ps

To display the currently running processes, use the command:

- **ps** : **p**rocess **s**tatus: produces a snapshot of the running processes.

```
ps
```

Output:

```
pi@raspberrypi:~ $ ps
  PID TTY          TIME CMD
 4202 pts/1    00:00:00 bash
23751 pts/1    00:00:00 ps
```

The output contains a list of the running processes under 4 columns:

- PID: the process identification number

- TTY: the terminal name

- TIME: the running time

- CMD: the name of the command that launches the process

Options that can be used with the ps command:

- **ps -a** : lists all the running processes of all the users
- **ps -u** : lists additional information (memory usage, CPU usage percentage, process state code, and process owner)

reboot

```
sudo reboot
```

shutdown

```
sudo shutdown -h now
sudo shutdown -h 20:00
```

-h switch halts processes, and the 20:00 specifies the time for shutdown

top

resource-usage of processes

```
top
```

Output:

```
Tasks: 196 total,   1 running, 195 sleeping,   0 stopped,   0 zombie
%Cpu(s):  6.3 us,  1.2 sy,  0.0 ni, 92.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :    923.1 total,     60.9 free,    491.9 used,    370.3 buff/cache
MiB Swap:    100.0 total,      0.0 free,    100.0 used.    334.1 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
19387 pi        20   0   53328  29212   9460 S  13.5   3.1 172:33.08 python3
 2215 pi        20   0  204856  94272   9700 S   7.9  10.0 122:17.62 thonny
 1127 root      10 -10   49908   8384   5200 S   2.6   0.9   3:08.71 monitor_device
29407 pi        15  -5   11352   2868   2432 R   1.3   0.3   0:00.15 top
  679 root       0 -20       0      0      0 I   1.0   0.0   3:35.49 kworker/u9:2-brcmf_wq/mmc1:0001:1
   65 root       0 -20       0      0      0 I   0.3   0.0   4:37.27 kworker/u9:0-brcmf_wq/mmc1:0001:1
  258 root      20   0       0      0      0 S   0.3   0.0   0:36.59 brcmf_wdog/mmc1
  579 root      20   0  377664  68384  28524 S   0.3   7.2   4:46.87 Xorg
 1613 pi        20   0  511420  98156  63384 S   0.3  10.4  12:13.86 chromium-browse
 1691 pi        20   0  309692  48116  42376 S   0.3   5.1   7:43.11 chromium-browse
 2063 pi        20   0  708108 195240  65360 S   0.3  20.7  18:54.59 chromium-browse
24297 root      20   0       0      0      0 I   0.3   0.0   0:00.40 kworker/u8:2-events_unbound
    1 root      20   0   33908   7556   6092 S   0.0   0.8   0:11.22 systemd
    2 root      20   0       0      0      0 S   0.0   0.0   0:00.31 kthreadd
    3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
    4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
    8 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 mm_percpu_wq
    9 root      20   0       0      0      0 S   0.0   0.0   0:00.00 rcu_tasks_rude_
   10 root      20   0       0      0      0 S   0.0   0.0   0:00.00 rcu_tasks_trace
   11 root      20   0       0      0      0 S   0.0   0.0   0:08.86 ksoftirqd/0
   12 root      20   0       0      0      0 I   0.0   0.0   0:37.47 rcu_sched
   13 root      rt   0       0      0      0 S   0.0   0.0   0:00.19 migration/0
   14 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/0
   15 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/1
   16 root      rt   0       0      0      0 S   0.0   0.0   0:00.13 migration/1
   17 root      20   0       0      0      0 S   0.0   0.0   0:02.38 ksoftirqd/1
```

Note

Unlike the **ps** command, the **top** command output updates periodically; You will see real-time updates for running times and CPU usage.

The output of the **top** command is a shell that allows the user to move through processes and interact with them.

Interacting with a process is done by the keys:

- **k** : kills the process
- **M** : sorts the list by memory usage
- **N** : sorts the list by the process identification numbers
- **r** : changes the priority of the process
- **d** : changes the refresh time interval
- **c** : displays the path of the process

These are not normal Bash commands, but commands in the RPi OS that specifically deal with functionality of the RPi Chip and hardware. Note, some of these vary across different version of the RPi OS.

Raspberry Pi OS Commands

libcamera-still

libcamera-vid

raspi-config

```
sudo raspi-config
```

raspi-gpio

vcgencmd measure_temp

measures temperature of Raspberry Pi

General Commands

clear

clears the terminal

history

!history number runs that command in history

man

this is the help manual. the following gets you the manual for the wget command

```
wget man
```

TAR

create archive

```
tar -cvfz archive.tar.gz/home/pi/folder
```

extract files

```
tar -xvfz archive.tar.gz
```

c - to compress
x - to extract
v - verbose
f - specify filename
z - use gZip to compress

zip/unzip

similar to tar but used on Windows systems

create zip

```
zip -r archive.zop /home/pi/folder-to-zip
```

extract zip

```
unzip file.zip
```

Network Commands

ifconfig

interface configuration note ipconfig is a similar windows command

iwconfig

wireless network configuration info

netstat

allows you to monitor network status for all ports

```
netstat
netstat -l
```

ping

checks to see if host is alive

ping ipadress

wget

download a file from the terminal

```
wget https://file_url.extension
```

ssh

connects to another Linux system using SSH

ssh user@IP

scp

copies files using SSH

Package Management

apt update

downloads latest package list from repository

sudo apt update

apt upgrade

downloads and installs packages from repository

sudo apt upgrade

dpkg

lists installed pacakges

dpkg -l

apt install

installs specified package

```
sudo apt instal name-of-package
```

apt search

searches for named package

```
apt search name-of-package
```

apt remove

removes named package

```
sudo apt remove name-of-package
```

6.4: Linux Commands is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

CHAPTER OVERVIEW

7: Starting your Raspberry Pi

Topic hierarchy

7.1: SD Card Formatter
7.2: Raspberry Pi Imager
7.3: Install VNC Viewer
7.4: ESD- The Silent Assassin
7.5: Installing RPi OS
7.6: Starting your Pi the First Time
7.7: Updating Your Pi
7.8: Registering your Pi with UALR IOT
7.9: VNC Viewer - Headless Mode
7.10: Adjusting Remote Monitor Settings
7.11: Pi Desktop
7.12: Pi Terminal
7.13: Keyboard Shortcuts
7.14: Google email account for class
7.15: Sending email without attachment
7.16: Sending email with attachment
7.17: Find IP Address
7.18: email IP address
7.19: Cron and Crontab
7.20: First Cron Job

Transclusion Table

Chapter 7	IOST Library Location
7.1: SD Card Formatter	1.2.1: MicroSD Card Formatter
7.2: Raspberry Pi Imager	1.2.2: Raspberry Pi Imager
7.3: Installing VNC Viewer	1.2.3: Install VNC Viewer
7.4: ESD - The Silent Assassin	2.1.2: ESD - Silent Assassin
7.5: Installing the PI OS	2.1.3: Installing PI OS
7.6: Starting your PI for the First Time	2.1.4: Starting PI for the First Time
7.7: Updating your PI	2.1.5: Updating your PI
7.8: Registering your PI with UALR IOT Network	1.3.1: Registering your PI with UALR IOT
7.9: Operating your PI in Headless Mode	2.1.6: Operating your PI in Headless Mode
7.10: Adjusting Remote Monitor Settings	3.1.2: Adjusting Remote Monitor Settings
7.11: Pi Desktop	2.1.7: PI Desktop
7.12: Pi Terminal	3.1.2: PI Terminal

7.13: Keyboard Shortcuts	3.1.3: Keyboard Shortcuts
7.14: Google email Account for Class	1.4.1: Google email Account for Class
7.15: Sending email without Attachment	5.6.1: Sending email without Attachment
7.16: Sending email with Attachment	5.6.2: Sending email with Attachment
7.17: Finding your IP Address	5.6.3: Find IP Address
7.18: email IP address	5.6.4: email IP Address
7.19: Cron and Crontab	3.4.1: Cron and Crontab
7.20: First Cronjob	3.4.2: First Cron Job

7: Starting your Raspberry Pi is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

7.1: SD Card Formatter

SD Card Formatter

This is required because you need to format a micro SD card and install an image of the Raspberry Pi Operating System (RPi OS) on it. Disk formatting is the process of preparing a data storage device such as a hard disk drive, solid-state drive, floppy disk or USB flash drive for initial use. In some cases, the formatting operation may also create one or more new file systems. **It also erases any previous data on the drive.**

An SD Card Formatter is not included in the Windows operating system, so we need to install it. To do so, go to:

https://www.sdcard.org/downloads/formatter/

Scroll to the bottom of the page and click the Blue Accept button. When you do so, it will automatically start to download the installer software as a Zip file. Right click on the file and extract. Double click on the Setup file go thorough the installation wizard and install the software. When you have done so, the SD Card Formatter App will show in your start menu (you can type it into the search box of the start menu).

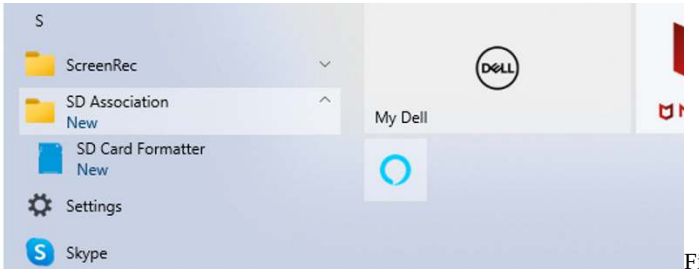


Figure 7.1.1:SD card formatter appears in start menu. You can also type SD Card... in the search option of the start menu and it will show up (CC0.0; Bob Belford)

You will need to use this software in section 1.1 when you install your Raspberry Pi's operating System.

This page titled 7.1: SD Card Formatter is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

7.2: Raspberry Pi Imager

Raspberry Pi Imager

The Raspberry Pi Imager is the easiest way to install the operating system on a micro SD card, and you can download the program from the pi.com software page https://www.raspberrypi.com/software/. Before installing the image you should format your SD card.

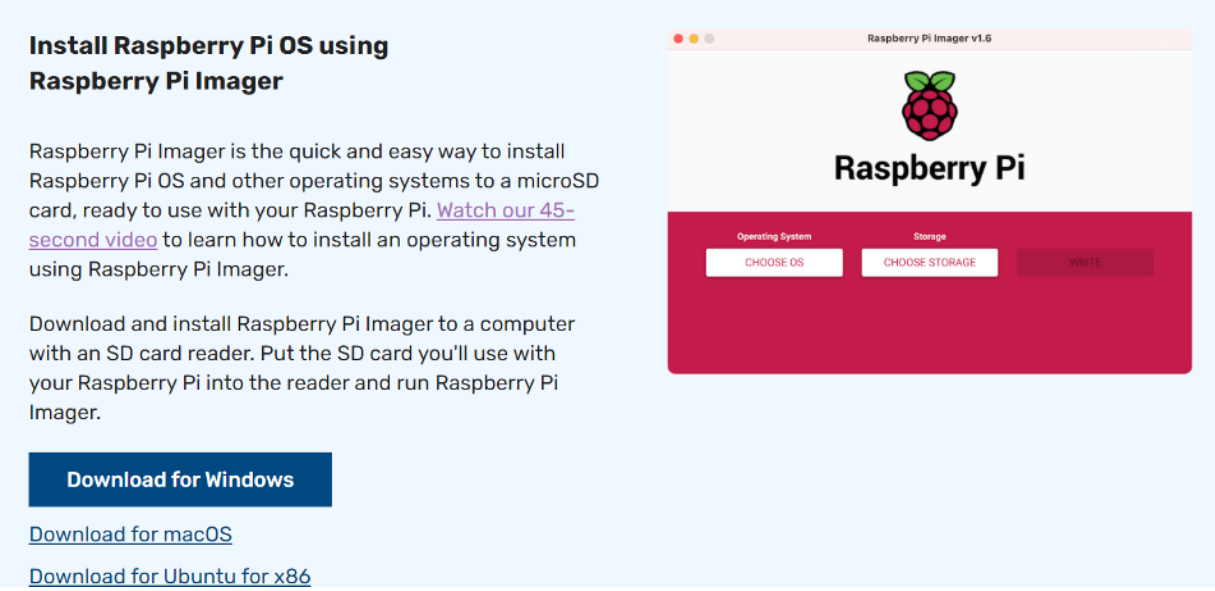


Figure 7.2.1: Copy and Paste Caption here. (Copyright; author via source)

Run the exe file once you have downloaded the imager

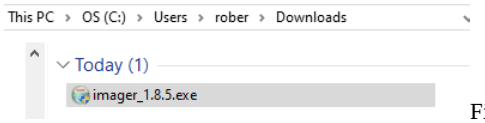


Figure 7.2.1: Copy and Paste Caption here. (Copyright; author via source)

Walk through the setup



Figure 7.2.1: Copy and Paste Caption here. (Copyright; author via source)

Pick the device and make sure you choose the USB drive your micro SD card is in

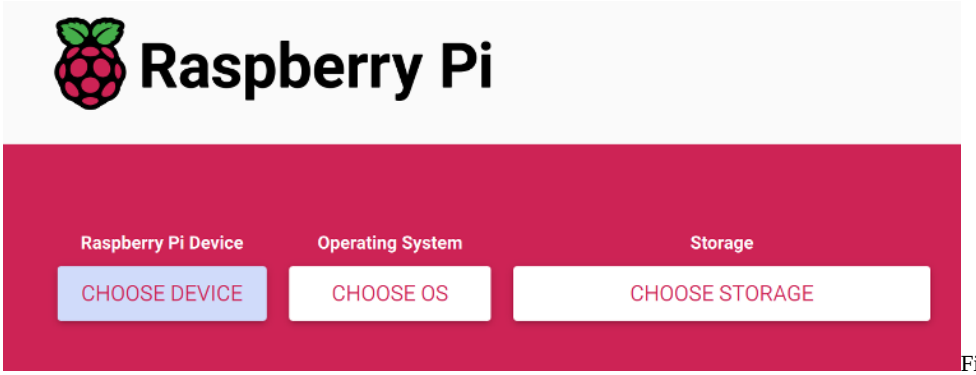


Figure 7.2.1: Copy and Paste Caption here. (Copyright; author via source)

This page titled 7.2: Raspberry Pi Imager is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

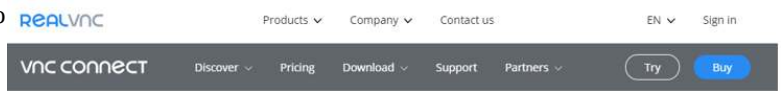
7.3: Install VNC Viewer

VNC Viewer

Install VNC

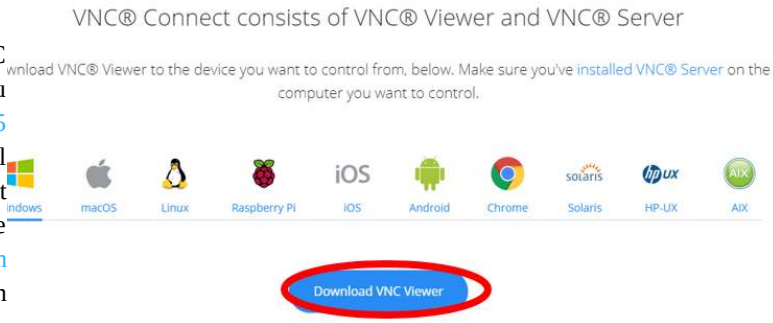
The Raspberry Pi comes with VNC (Virtual Network Computing) and we will install it on our desktop/laptop so that we can connect to the Pi in "headless mode", effectively using our desktop/laptop as the mouse, keyboard and monitor for controlling the Pi. Go to https://www.realvnc.com/en/connect/download/viewer/ and download the viewer (Figure 7.3.16).

In the next section we will use the VNC viewer to connect to the Raspberry Pi over the internet.



Setting up VNC for Headless Access

In this section you will learn how to set up VNC Viewer for headless mode access. To do this you need your Pi's IP address. In module 1.3 section 5 you learned how to have your Pi send you an email with its IP. If you have not done that, are a student at UALR, you can find your Pi's IP address from the IOT WiFi page as described inAppendix 3, section 5.1.4. Otherwise, you can hover over the WiFi icon on the Pi Desktop or go to command mode and type IP as shown in figure of section 1.1.4.



(Figure 7.3.16: Download option for VNC viewer

Start VNC Viewer and then either right-click and choose "new-connection" or choose file/New connection, and you will get the popup window of figure 7.3.1.

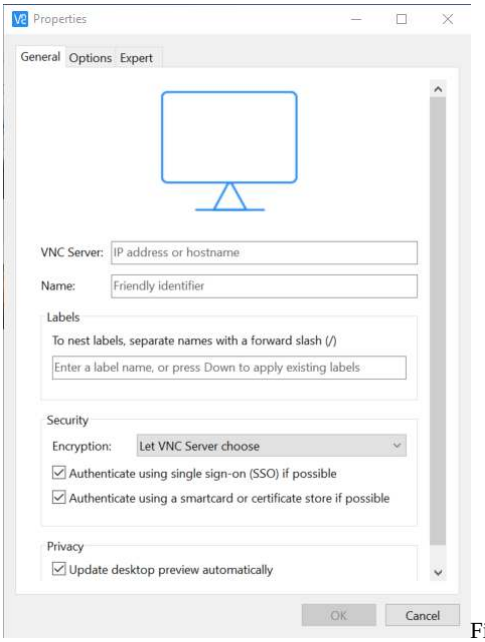


Figure 7.3.1: Paste in your IP into the popup box and give your pi a name you would recognize.

You now have a new connection on your VNC user panel. Double click and use the password and name you set up on your Pi (section 1.2.4), and you are in. Before proceeding further you should adjust the Pi resolution so it matches that of your monitor

VNC Viewer on Pi

This page titled 7.3: Install VNC Viewer is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

7.4: ESD- The Silent Assassin

Electrostatic Discharge (ESD)

Your Raspberry Pi is not only a single board computer that contains many tiny integrated circuit (ic) chips that are susceptible to Electrostatic Discharge (ESD), but so are many of the sensors and devices you will be working on the breadboard with. ESD is a real problem with these small microcircuits where static electricity can build up and discharge when you touch a device, and cause damage to the circuits. Here is an article that can give you a bearing of the problem, Combating ESD: the silent assassin in electronics manufacturing. The following YouTube by the Hoffmann Group describes electrostatic discharge should be watched before proceeding, and especial note should be taken to the difference between direct and latent faults that can be induced through improper handling.



Video 7.4.1: 3:29 min video by Hoffmann Group on electro static discharge (https://youtu.be/bWXNOemu2j4)

ESDS devices (Electrostatic Discharge Sensitive Devices) and are shipped in special antistatic protective packaging. You need to take proper precautions with respect to ESD. Failure to do so can not only result in problems that are almost impossible to troubleshoot, like a latent fault that causes a short circuit that only occurs in rare situations, and intermittently causes your program to crash, but lead to direct faults that can result in the complete destruction of your Raspberry Pi or other devices (like your camera).



Figure 7.4.13: Antistatic wristband (upper left, Wikimedia Commons), ESDS Device logo (lower left, freesvg) and an esp8266 microprocessor in an ESD protective bag that is placed on an antistatic map that needs to be grounded before use. (right, Belford, CC 0.0)

If you start to solder printed circuit boards you should consider investing in an antistatic map with an arm band, but no matter what, you should take prudent action to avoid ESD damage to your devices.

This page titled 7.4: ESD- The Silent Assassin is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

7.5: Installing RPi OS

Installing RPi OS

Preparing your microSD Card

In Appendix A1.1 you installed all the software on your computer that you need for this task. Place a microSD card into a USB adapter sleeve and then place the USB into a USB port on the computer you set up in IOST module 1. Be sure you have no other external storage devices hooked up to your computer.

Format microSD Card

This step will show you how to use the SD Card Formatter to format the microSD card. This will clear all data on the card, and be sure you format the microSD card and not your hard drive!



Figure 7.5.2: USB adaptor sleeve for microSD card

1. From Start Menu, choose SD Card Formatter (you can click through the windows icon and scroll through the Apps to SD Association, or type in the search menu "sd" and the sd card formatter will autoappear

2. Now that you have the SD Card Formatter open you can go and format your microSD card. Now be sure NOT TO FORMAT the C drive. Figure 7.5.4 shows screenshots of this process.

Steps:

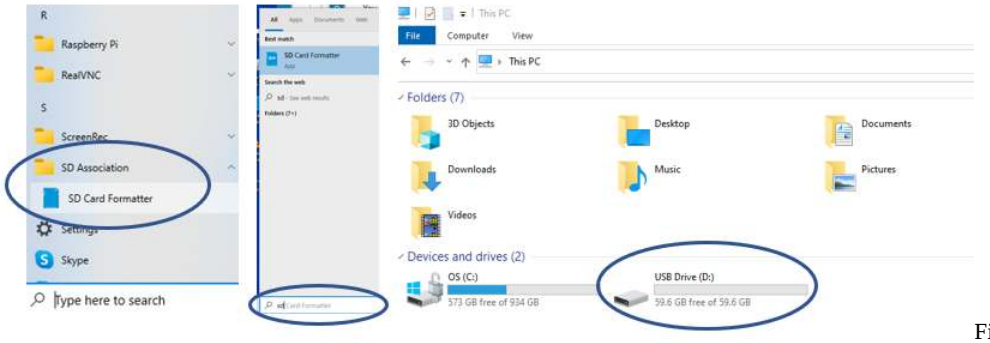
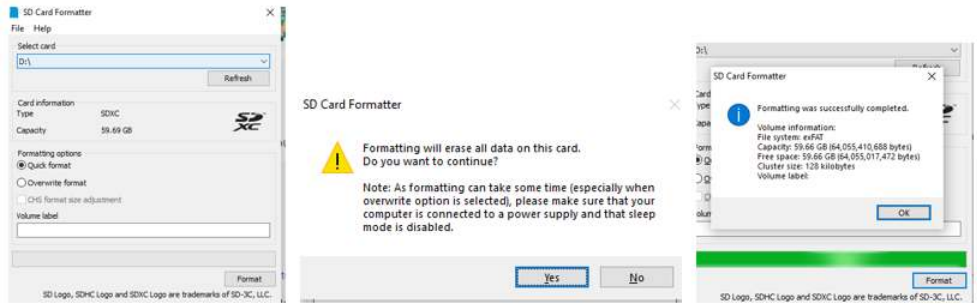


Figure 7.5.3: Two ways to activate SD Card Formatter from Windows Start Menu. Scrolling through the Apps (left image) or using Start menu Search option (middle image). Be sure you are formatting your USB drive with the micro SD card (right image) and not your hard drive!

1. Select Drive SD card is in and choose Quick format option
2. Choose yes if you are in correct drive
3. Choose OK when complete



Step 1 Step 2 Step 3

Figure 7.5.4: Screenshots of formatting the microSD card with SD Card Formatter. (1) Check to be sure which drive your microSD card is in and use quick format and make sure you are formatting the drive identified in step 1 (you do not want to format your computer's hard drive!). (2) The warning statement that formatting a drive will erase all data in the drive. (3) The formatting complete message, you are now ready to place an image of Raspbian on your micro SD card

Install RPi OS

There are multiple ways to install the RPi OS. The easiest is to use the Raspberry Pi Imager at https://www.raspberrypi.com/software/.

If you have not already installed Raspberry Pi Imager, follow these instructions. If you have, jump to step 4 and open the app. At the time of this writing the operating system is Debian 11 (Bullseye), which was released Oct. 30, 2021. As this is a new release there may be some issues with projects that use HATs (Hardware Attached on Top) and you may wish to use the Buster operating system. You can download both from this link, and then make an image on your personal computer that can be transferred to a micro-SD card. If you choose this later route, you will want to follow the instructions from the 2020 IOST class.

Step 1: Insert formatted micro SD card into USB reader and insert into a USB drive of your personal computer.
Step 2: Download Raspberry Pi Imager (figure 7.5.5.
Step 3: Click on the downloaded exe file (imager_1.7.1.exe in figure 7.5.6

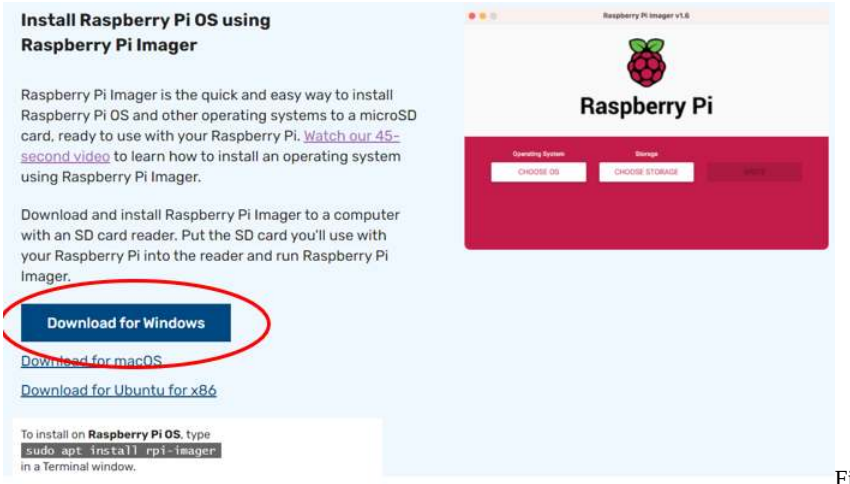


Figure 7.5.5: Download RPi OS Imager, here we are downloading onto a Windows PC. (Copyright; Belford CC 0.0)

After you download the imager you can install it



Figure 7.5.6: Installing Pi Imager. (Belford CC0.0)

Step 4: The Raspberry Pi Imager is now one of your apps and you can choose the operating system and storage device you wish to generate the image on

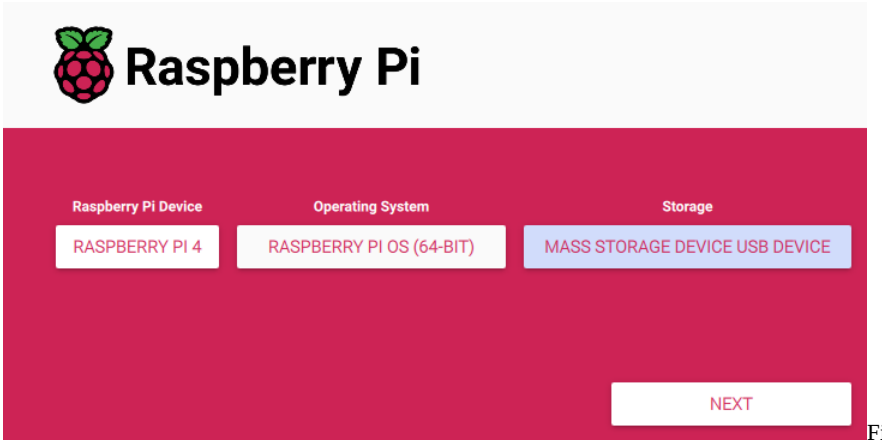


Figure 7.5.7: Installing RPi OS on a a micro SD card with imager (Belford CC 0.0)

The RPi operating system will now be installed

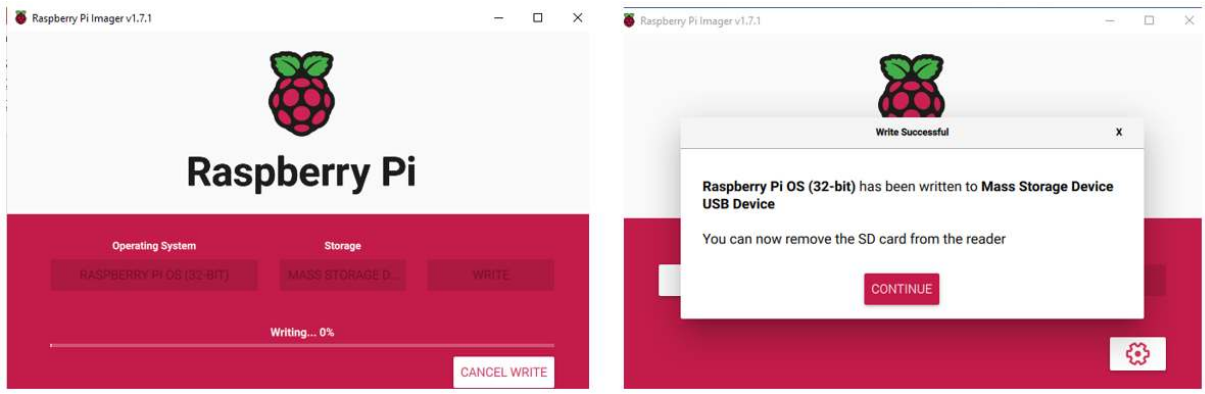


Figure 7.5.8: You are now ready to start your Pi. (Belford CC 0.0)

You now have the image of a Disc operating system that you will need to transfer to the micro SD card.

This page titled 7.5: Installing RPi OS is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

7.6: Starting your Pi the First Time

Starting Your Pi for the First Time

The first time we start the Pi will be in Desktop Mode, where we will set up the preferences and get the system MAC address so that we can remotely access the Pi through VNC. Before we connect the monitor, keyboard and mouse we need to install the microSD card. Figure 7.6.7 shows the correct and incorrect way to install the SD card.

Figure 7.6.8 shows a Pi in desktop mode. Note, the last thing you should connect is the power supply. If you are using a Pi 4 there are two mini HDMI ports (0 and 1). If you are only using one monitor you want to hook up to port 0, which is closest to the HDMI port.



Figure 7.6.1: Pi 4, note micro hdmi port 0 is near usb power port, and this is the one you hook up if you are only running one monitor. (Belford, cc 0.0)



Figure 7.6.7: Insert the micro SD card into the Pi before connecting monitor, keyboard and mouse.

Turn the Pi on and you will be taken through a startup menu. Follow the instructions.

Proceed through the setup menu, when you get to update software you should skip it if you are in the class as we do not have time, but do it if you are logged in and doing this from your home. In the next section we will learn how to update the software from the command line, and you can do that from home.

Now that the Pi has been setup you can go to preferences and double check your configuration.

1. Set Country: Choose United States, American English, Chicago (if you are in Little Rock), and click "Use US Keyboard".

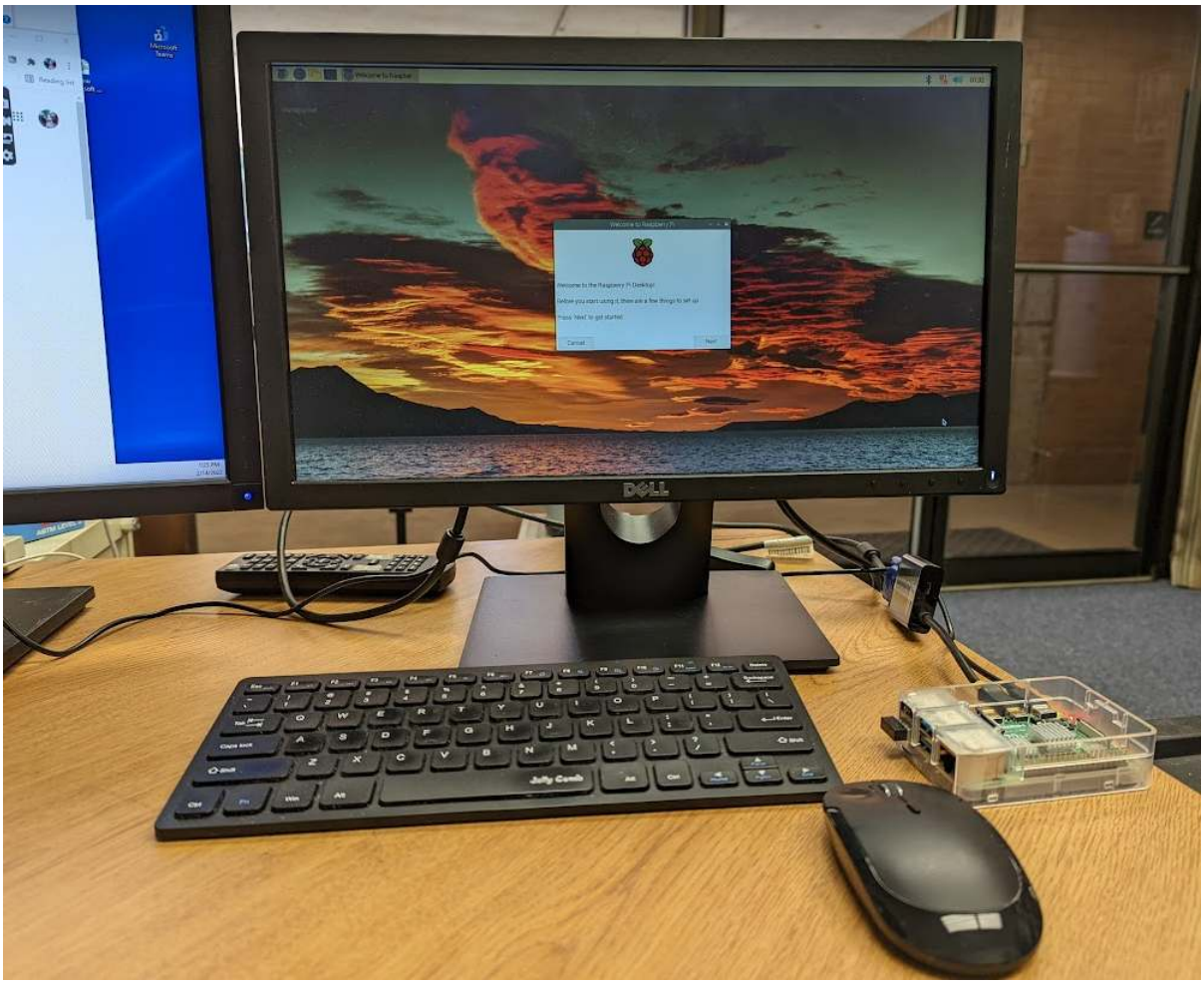


Figure 7.6.8: Raspberry Pi 3 in Desktop Mode, note the wireless mouse and keyboard dongle in the USB 2.0 port



Figure7.6.9: If you live in Little Rock Arkansas choose US, American English and Chicago as your time zone. Also click "Use US Keyboard". Then enter a new password as everyone has the same default password (raspberry). If you are on campus you need to skip Select WiFi Network as you need to register your RPi's MAC address with IT services before you can hook up. If you are at home you can now connect to the WiFi Network.

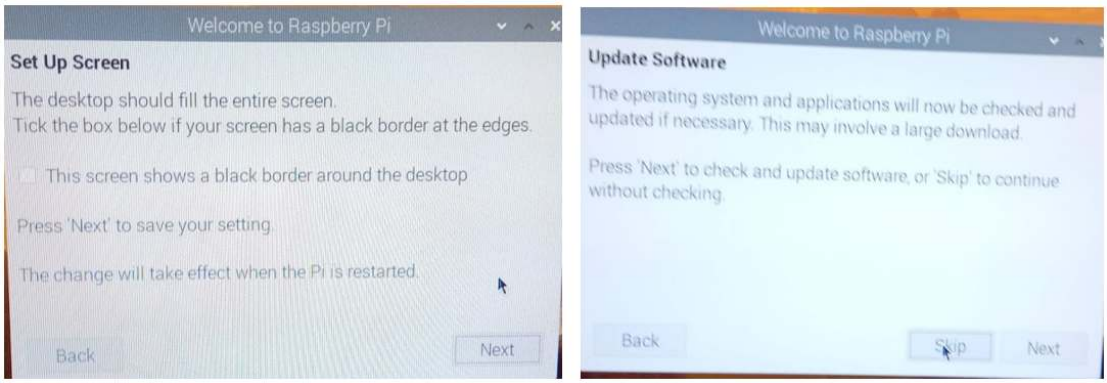


Figure 7.6.10: Do not update the software if you are doing this during class as we do not have enough time.

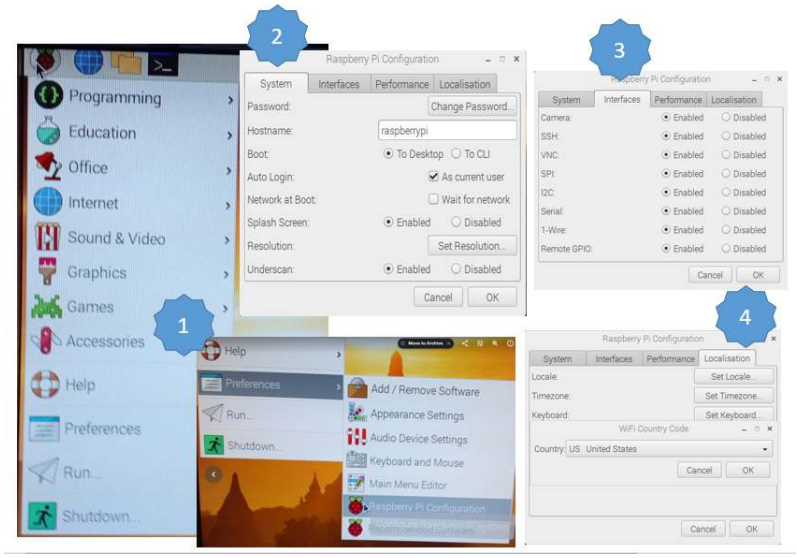


Figure 7.6.11:

This page titled 7.6: Starting your Pi the First Time is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

7.7: Updating Your Pi

Bullseye Taskbar Update

The taskbar of Bullseye has an icon you can use to update your Pi.



Figure 7.7.1: 2/14/2023 screen capture of Bullseye taskbar updat icon. (Belford, cc 0.0)

Updating with APT

APT is Debian's Advanced Packaging Tool and keeps a list of your software in /etc/apt/sources.list. The first thing you want to do is update and upgrade your Pi with the following commands. Open the terminal window and type

```
sudo apt update
```

The above code downloads package information on installed programs from the internet, allowing you to get updates. You can see what is upgradable with the following (optional) command

```
apt list --upgradable
```

Now install the new upgradable packages by running upgrade

```
sudo apt full-upgrade
```

Note, full-upgrade picks up any dependencies that upgrade may miss.

Now reboot your pi

```
sudo reboot
```

To do this in one command

```
sudo apt update && sudo apt upgrade -y
```

the -y is your response when the upgrade command asks you if you want to go ahead and upgrade.

You can also remove old packages after upgrade

```
sudo apt autoremove
```

apt vs apt-get

The apt and apt-get commands essentially do the same things, but the apt gives you more information. I believe the apt-get is the older command, and it is probably best to just use apt. See the article by Baeldung: Differences between apt and apt-get.

This page titled 7.7: Updating Your Pi is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

7.8: Registering your Pi with UALR IOT

Obtain Pi MAC Address

The MAC (Media Access Control) address is the ID that identifies your Pi and is required when you register your device on the UALR IOT WiFi network. Press ctrl-alt-t to bring up the command line terminal and type the following code, which is a shortcut for ifconfig

```
ip a
```

The above command gives all ip address associated with all network interfaces, and you should see two, the ethernet (eth0) and the WiFi (wlan0), and you need the MAC address associated with wlan0

(be sure to get the MAC address for the wlan0 and not eth0



Figure 7.8.3: MAC address for wlan0 (the pi's default wifi

Register MAC address

Either Google "UALR IOT WiFi" or go directly to the UALR Information Technology Services Wireless network page

https://ualr.edu/itservices/applications/v/wireless/

Scroll down the page until you see the image on the left of figure 7.8.4 and choose UALR-IoT (you would use UALR-WiFi to register your laptop or smartphone)

Go to the bottom of the page and click "Add new device"

Obtain IP address

Now that your PI is registered and on the network you can obtain it's IP address by clicking on the device (left part of figure 7.8.5).

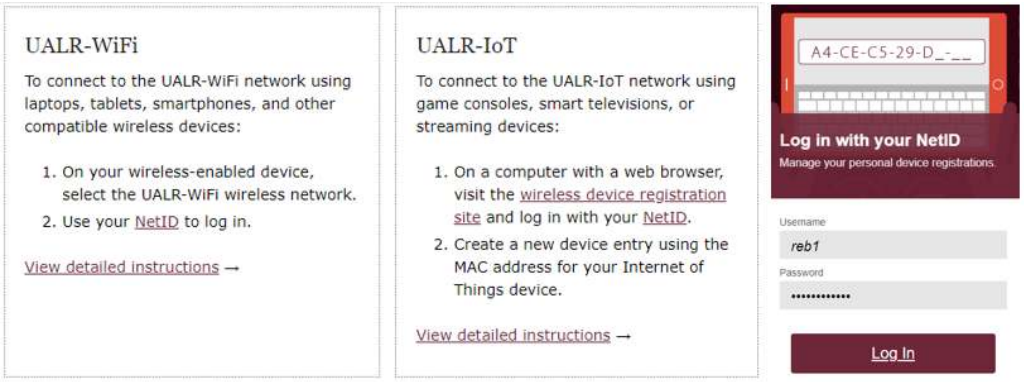


Figure 7.8.4: Screen shots of UALR-IoT sign in process.

NOTE, UALR has dynamic and not static IP addresses. This means if you shut off your Pi and reboot it, the IP may change. If that happens, you have three basic options to obtain your IP address:

1. Run your Pi in desktop mode (with monitor,keyboard and mouse) and type <ip a> in the command line.
2. Login to your UALR IOT network panel and look up the IP address for any device on the IOT network.
3. Run a script when your Pi boots to email its IP address to your email account whenever the Pi boots.

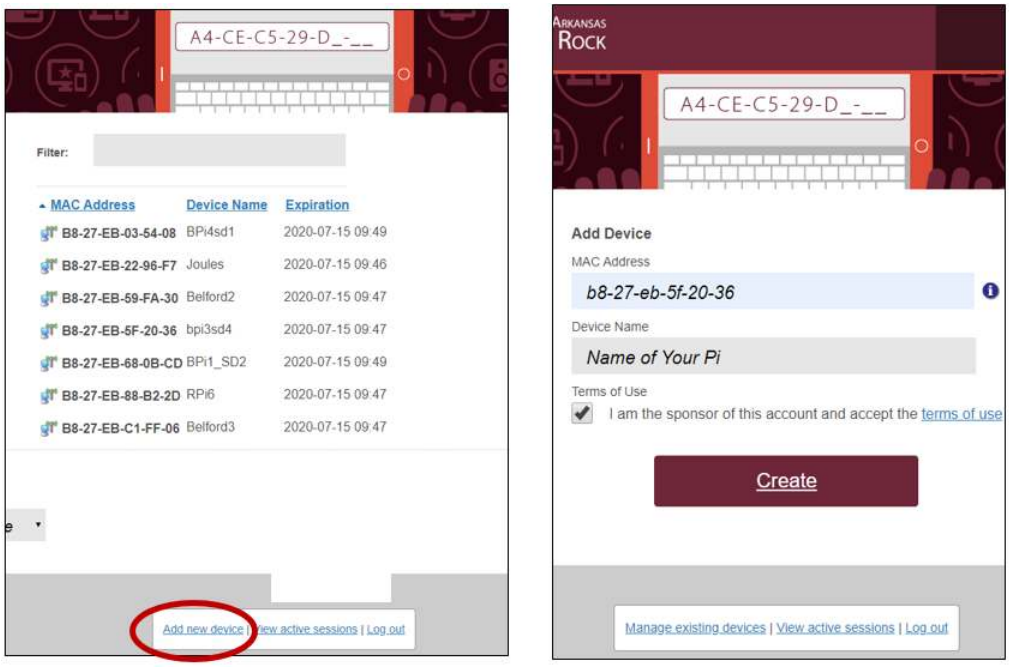


Figure 7.8.5: Screen shots showing prior registered IOT devices and form for registering a new device.

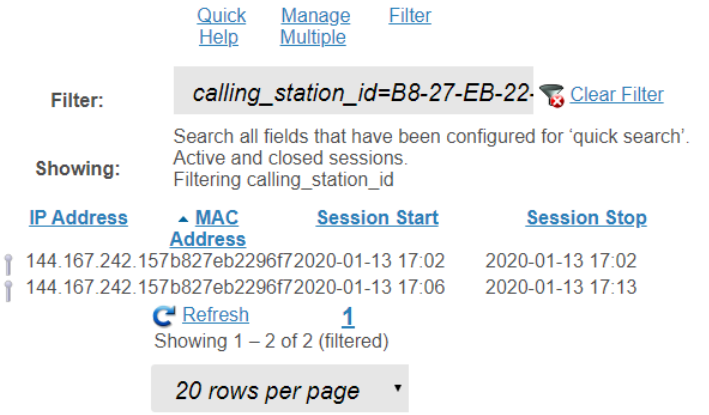


Figure 7.8.6: The IP address for a registered device from UALR's IOT Network access page.

This page titled 7.8: Registering your Pi with UALR IOT is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

7.9: VNC Viewer - Headless Mode

Now that you have your Pi up and running you want to remotely access it (in Headless mode) through VNC viewer, which you installed on your laptop or desktop (see VNC Installation Instructions).

Activate VNC on Raspberry Pi

The easiest way is to click the Raspberry icon on the taskbar, choose preferenes/Raspberry Pi Configuration/Interfaces and toggle VNC to on (figure 7.9.1

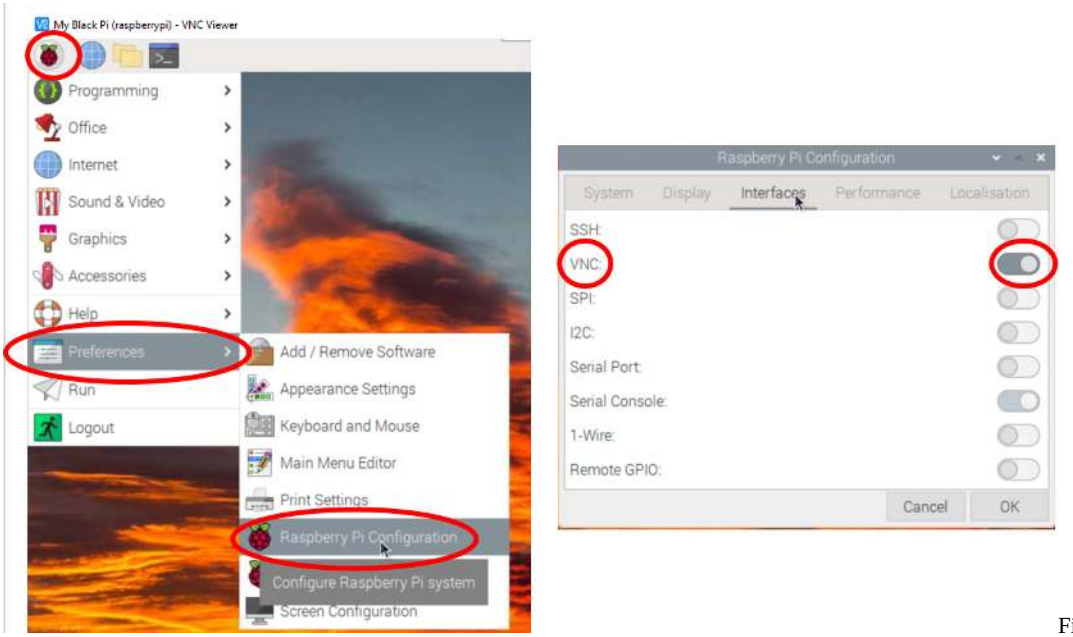


Figure 7.9.1: Screenshoot of procedures for activating VNC Viewer on Raspberry Pi. (Belford cc 0.0)

Obtain Pi IP address

Your desktop VNC viewer connects to the Pi over the internet through the Raspberry Pi's IP (Internet Protocol) address. This is a string of digits containing 4 sets of numbers separated by decimal points xxx.xx.xx.xx, and is like the web address for the Pi. These are assigned by your Internet Service Provider and may be static or dynamic. If it is static, it will never change, and you only have to do this once. If it is dynamic you will have to get the IP address every time it changes. Do deal with this we will set up a program that emails your IP address upon bootup.

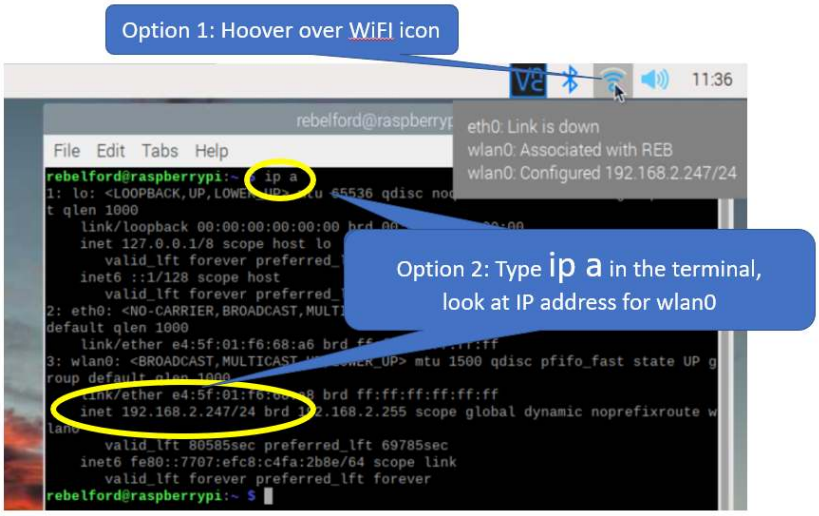


Figure 7.9.2: two ways to get Raspberry Pi IP address. (Copyright; author via source)

Note

If you are at UALR you need to connect your Raspberry Pi to the IOT WiFi network, and this requires you to register it with IT Services (see instructions here). You can get your IP address by going to IT services and looking up your device

Set up Desktop VNC Connection

You can actually run multiple VNC connections at the same time. First you need to set up a connection, which you can do by right-clicking in a blank area of the viewer and choosing "new connection", or choosing file/new connection. This opens the panel if figure 7.9.3, you type in your IP address and give your connection a name

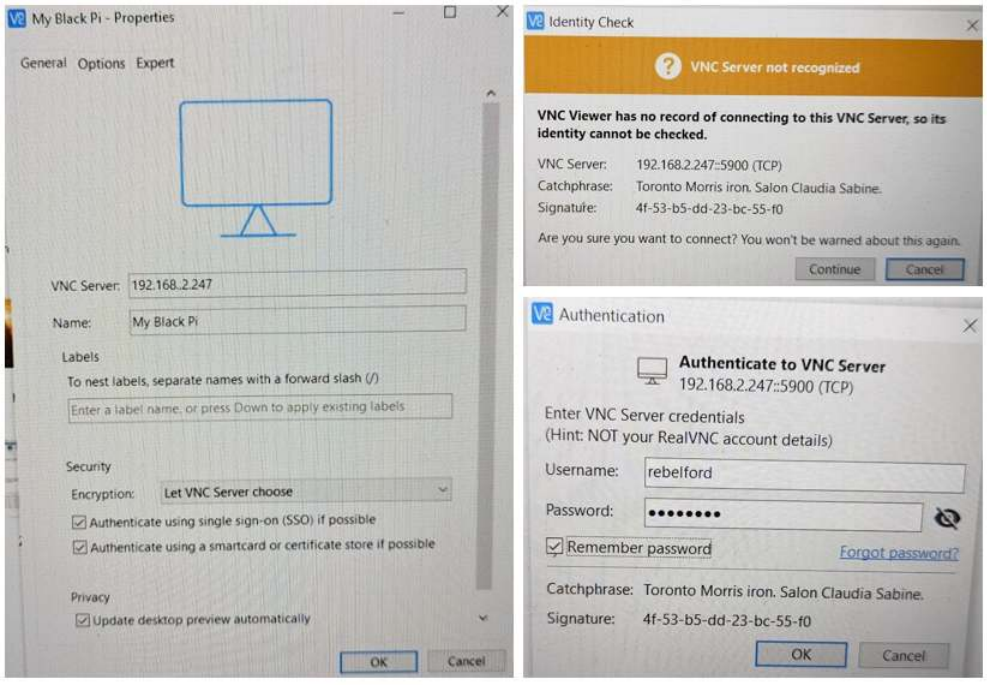


Figure 7.9.3: Right click a blank area of VNC Viewer and create a new connection. After you give it an IP address and click OK, it will try and connect, and the first time will give you the sever not recognized screen. Continure, then put in your username and password to the Raspberry Pi, and you are good to go. If you click "Remember password", it will take you directly to the Pi desktop the next time. (Belford cc 0.0)

> **Note**
>
> If you have an existing VNC connection and it does not work because the Raspberry Pi IP address changed, you can right click on it, and choose properties. That will open the panel in the left image of figure 3, and allow you to update the IP address.

You may wish to change your screen resolution (see section

File Transfer

One of the nice features of VNC is the ease with which you can transfer files between the pi and your PC or laptop. It is very important that you structure your directories (folders) on both your PC and the Pi so that you can keep your work organized. Therefore you should choose the destination folder with some fore thoughtl.

PC2Pi

To transfer from your personal computer (PC) to the the Pi you can use the collapsible menu at the top center of the Raspberry Pi desktop. You will need to navigate to the folder containing the file you want to upload to the Pi, and you can then place it on the Pi's desktop.

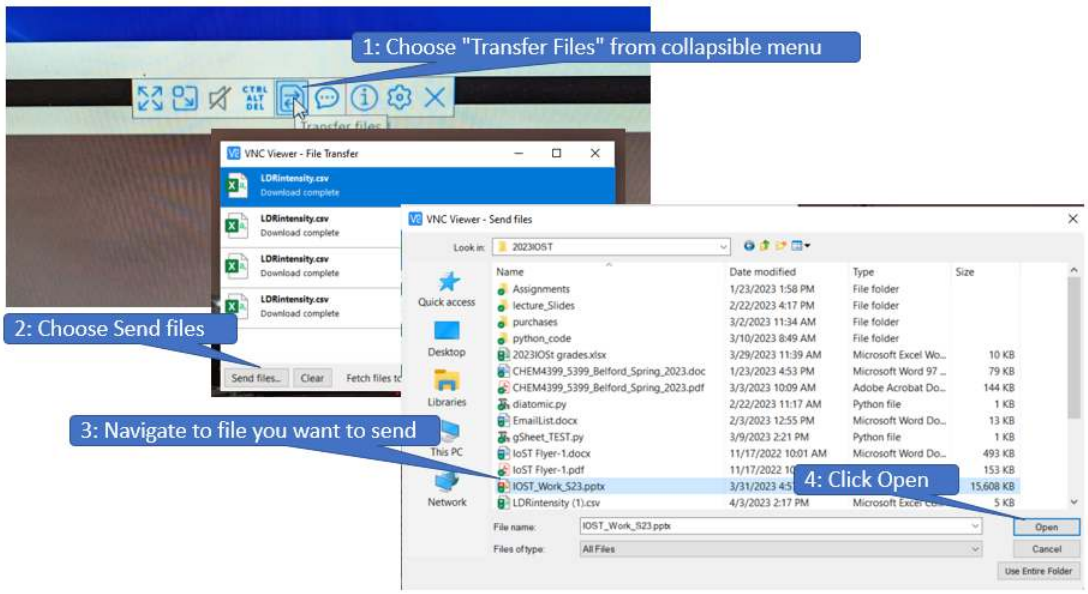


Figure 7.9.4: Copy and Paste Caption here. (Copyright; author via source)

1. Go to the collapsible menu and choose the "Transfer Files" options
2. Choose Send "files.."
3. Navigate to the folder on your PC having the file you want to upload and click on it
4. Choose "Open"

The file will now be uploaded to your Raspberry Pi desktop.

Pi2PC

THe following steps allow you to transfer a file from your Raspberry Pi to your personal compter or laptop. You should set up a destination folder in your PC before you start sending files.

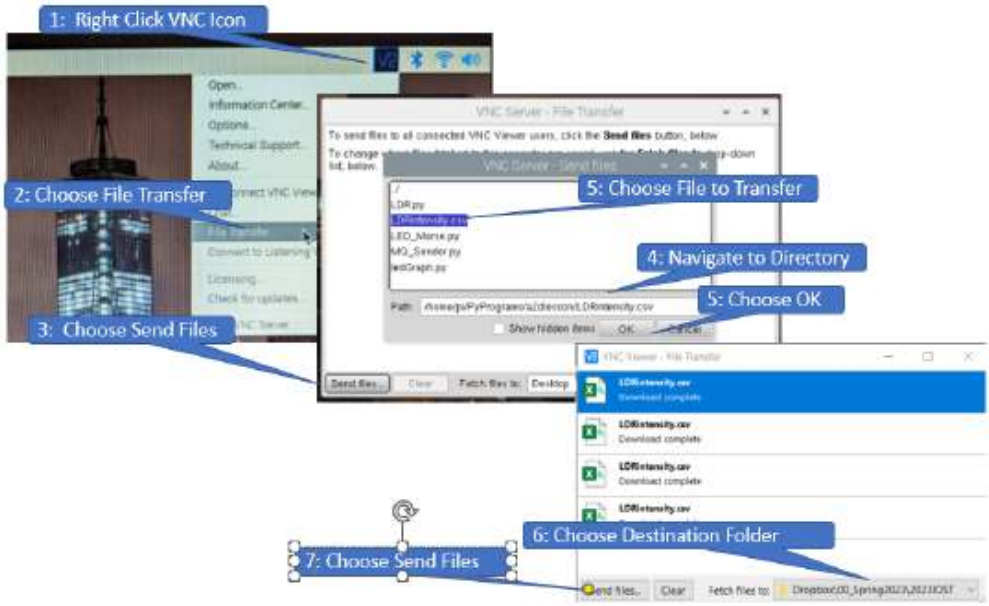


Figure 7.9.1: Copy and Paste Caption here. (Copyright; author via source)

1. Right Click on VNC icon
2. Choose the "File Transfer" option
3. Choose "Send Files"
4. Navigate to the directory of the file you want to send
5. Click OK
6. Choose the destination directory on your PC
7. Choose Send Files.

VNC Issues

Sometimes you lose the VNC Icon and can not transfer files. This is because you are running WayVNC server on the pi and not RealVNC Server. To fix this you must operate on Desktop mode as you will be disconnecting VNC. start by typing:

```
sudo raspi-config
```

this opens the configuration tool and choose:

- 6 Advanced Options
- A6 Wayland
- W1 X11
- finish and reboot

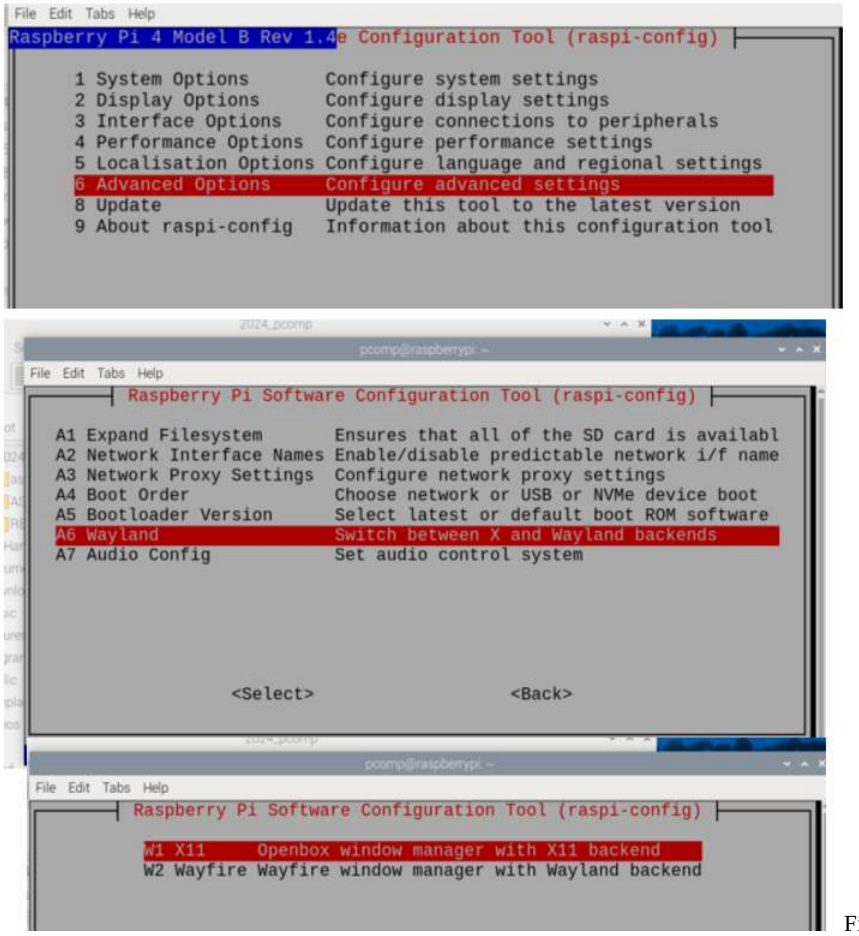


Figure 7.9.1: Copy and Paste Caption here. (Copyright; author via source)

Once you reboot you need to restart VNC Viewer, which you can do from the configuration tool or from preferences. In the case of the configurationt tool choose

- 3. Interface Options
- I2 VNC
- Yes
- finish
-

This page titled 7.9: VNC Viewer - Headless Mode is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

7.10: Adjusting Remote Monitor Settings

In this tutorial you will learn how to set your pi desktop to have the same screen resolution as the monitor or laptop you are using VNC Viewer to access your Pi with. The first step is to determine your monitor (laptop) screen resolution

Laptop Monitor Screen Resolution

In Window's 10 you go to Settings/System/Display (you can type "Change display settings" in the windows search bar) and scroll down to "Resolultion

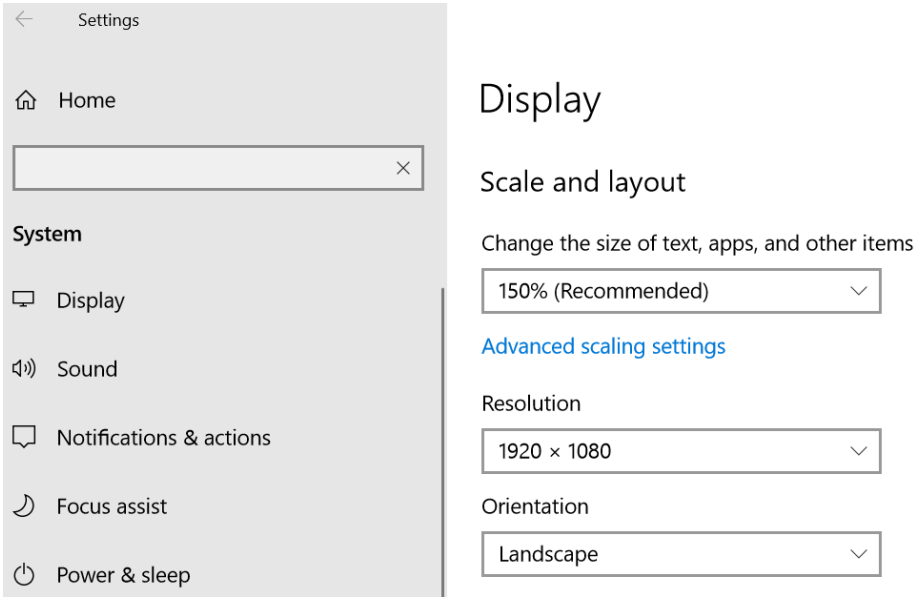


Figure 7.10.1: Screen resolution in Windows 10

Now that you know the resolution of the monitor you are using to remotely access the Pi, you can set the resolution to match that (1920x1080 in this case).

Setting Pi Desktop Resolution

Remotely log into your Pi and click the Raspberry icon/Preferences/Raspberry Pi Configuration (fig. 2)

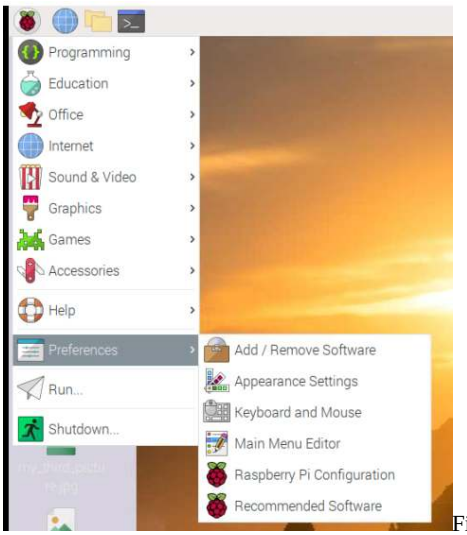


Figure 7.10.2:Choose Raspberry Pi Configuration

Toggle to the Display option and you choose "Set Resolution" (fig. 3)

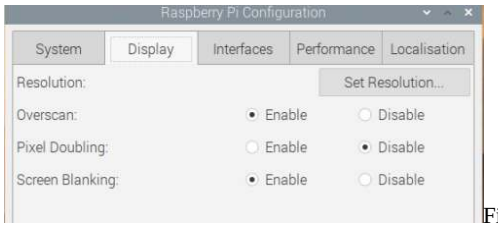


Figure 7.10.3: Choose "Set Resolution" under the Display option

Choose the option of your monitor and click OK (fig. 4)

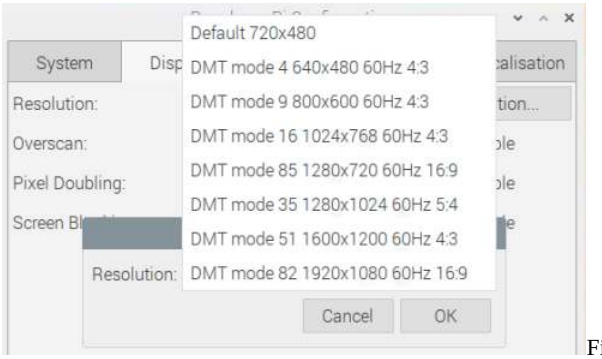


Figure 7.10.4: Copy and Paste Caption here

You will need to reboot the Pi for the settings to take place.

From Terminal

from the terminal type

```
sudo raspi-config
```

This brings up a menu, and you need to use the up/down arrows to navigate and choose option 2

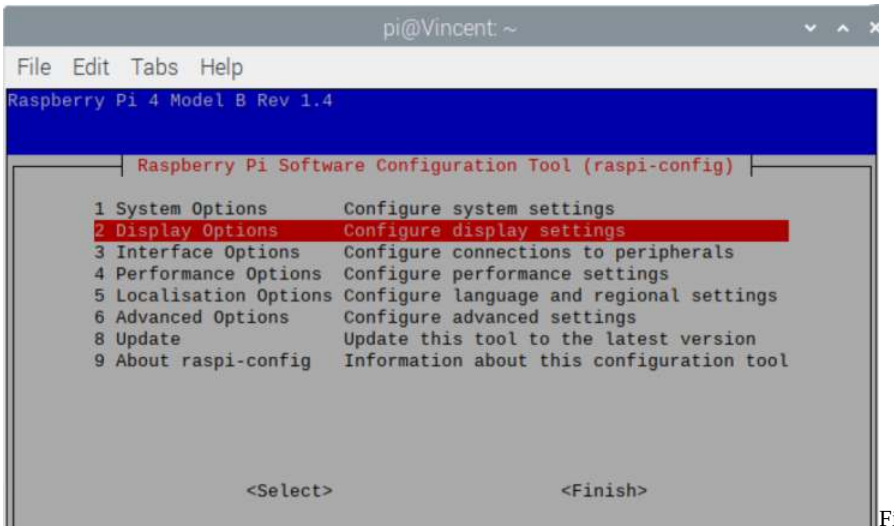


Figure 7.10.1: raspi-config options. (CC 0.0 Belford)

Now choose option D5

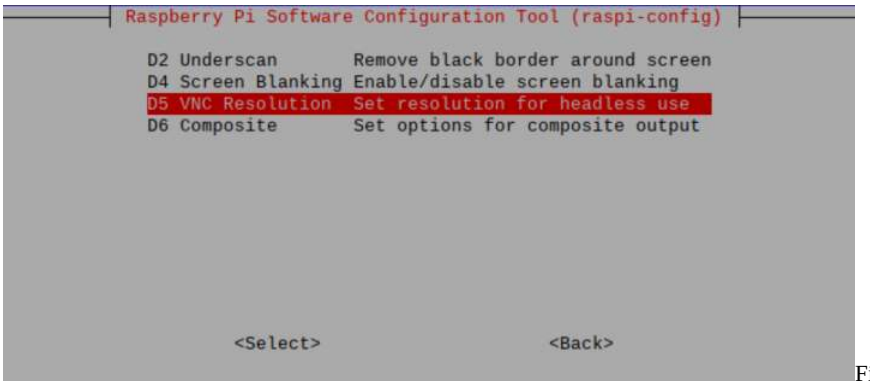


Figure 7.10.2: Display options under Raspi-config. (Belford CC 0.0)

and pick a resolution

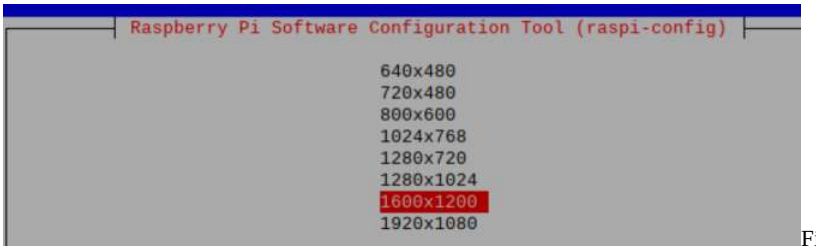


Figure 7.10.3: Available resolutions. (CC 0.0 Belford)

This page titled 7.10: Adjusting Remote Monitor Settings is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

7.11: Pi Desktop

This page is under construction

Install LibreOffice

Libreoffice is available within the Raspbian package repository

```
sudo apt install libreoffice
```

7.11: Pi Desktop is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

7.12: Pi Terminal

Opening Terminal (CLI)

On the Raspberry Pi the Command Line Interface (CLI) is called the Terminal. There are three ways to open the Terminal

1. Terminal Icon on menu bar (most common)
2. Terminal Icon through Applications menu (Pi Icon)/Accessories/terminal
3. Keyboard Shortcut <ctrl>+<alt>+<t>

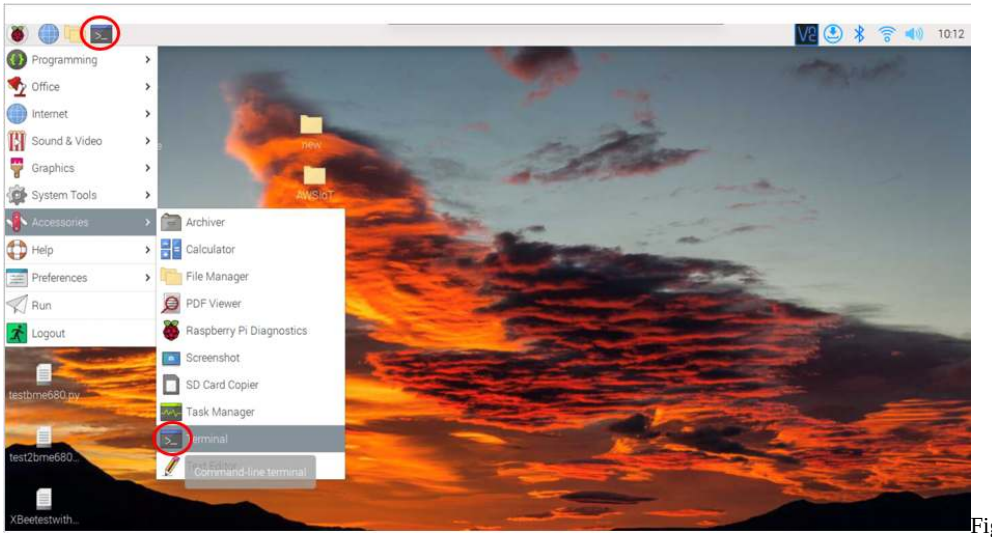


Figure 7.12.1: Terminal icon on the top menu bar, or accessing through Pi Icon. (Belford cc 0.0)

You now have opened up the terminal.

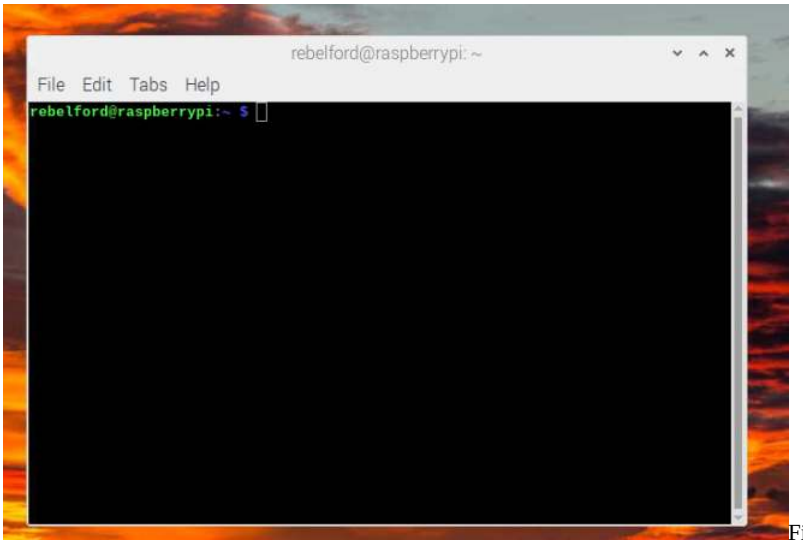


Figure 7.12.2: Terminal CLI of Raspberry Pi. (Belford cc 0.0)

Resizing Terminal

To resize the terminal use the following keyboard shortcuts (you press all three keys at once):

- Larger
  - <ctrl>+<shift>+<+>
- Smaller

- <ctrl>+<shift>+<->

The Prompt

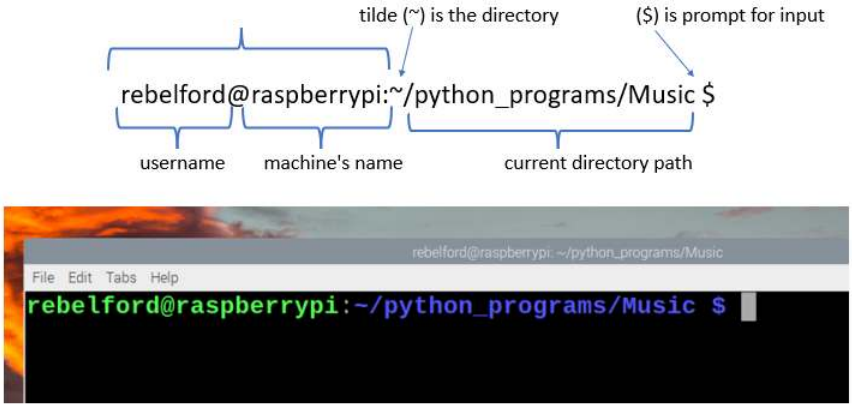


Figure 7.12.3: Pi terminal prompt. (Copyright; author via source)

This page titled 7.12: Pi Terminal is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

7.13: Keyboard Shortcuts

In the following commands you click all keys in < > at once

General Commands

- **Open the Terminal**:
  - <ctrl> <alt> <T>
- **Close the Current Window**:
  - <Ctr> <Q>
- **Kill a running command**
  - <ctrl> <C>
- **Clear screen**
  - <ctrl> <L>
- **Open system menu**
  - <ctrl> <esc>

Keyboarding Commands

- **Copy**
  - <shift>+<ctrl>+<c>
- **Paste**
  - <shift>+<ctrl> +<v>
- **Wipe out from where you are to the beginning**
  - <ctrl>+<u>

- **jump to beginning of a line**
  - <ctrl>+<a>
- **goes to end of line**
- <ctrl>+<e>

This page titled 7.13: Keyboard Shortcuts is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

7.14: Google email account for class

Introduction

In this class you will be hard coding your email and password, which may be discoverable by hackers. You also need to run your account with the "less secure apps access" option as we will not be using high levels of verification. Therefor it is imperative that you do not use your normal account, and this section shows you how to set up an email account in Google and adjust the settings so your Raspberry Pi can talk to it.

Setting Up Google Account

The very first thing that you need to do is to obtain a **new gmail account at** https://www.google.com/gmail. Do not use your UALR account, or a Google Account that you use in normal transactions. You need to set up an account that is used just for this class and your Raspberry Pis, and we suggest you use the gmail address of your last name followed by IOST, **PcompYourFirstName@gmail.com**, as this should be easy to remember. Make sure you don't use the password for this email account for anything else, and make it something easy to remember. If you have problems you can go to their help page, https://support.google.com/mail/answer/56256.

Adjusting your Security Settings

1. Now you need to do some tweaks to the gmail settings. Go to your profile picture (Or a circle with initials if you didn't add a profile picture) in the upper right corner and click "**Manage your Google Account**" (figure 7.14.1).

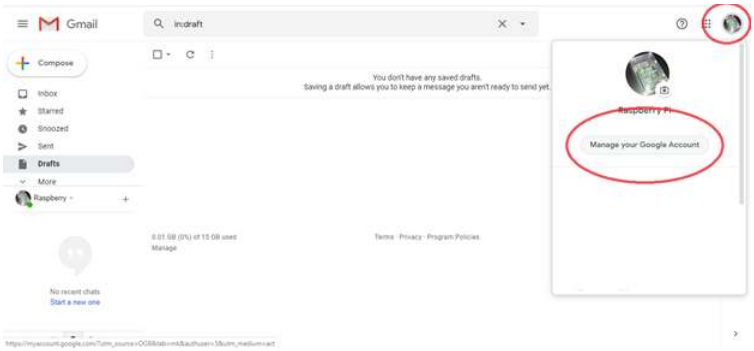


Figure 7.14.1: Screenshot of Google account homepage in February 2020.

2. Once in your account click security on the left (figure 7.14.2).

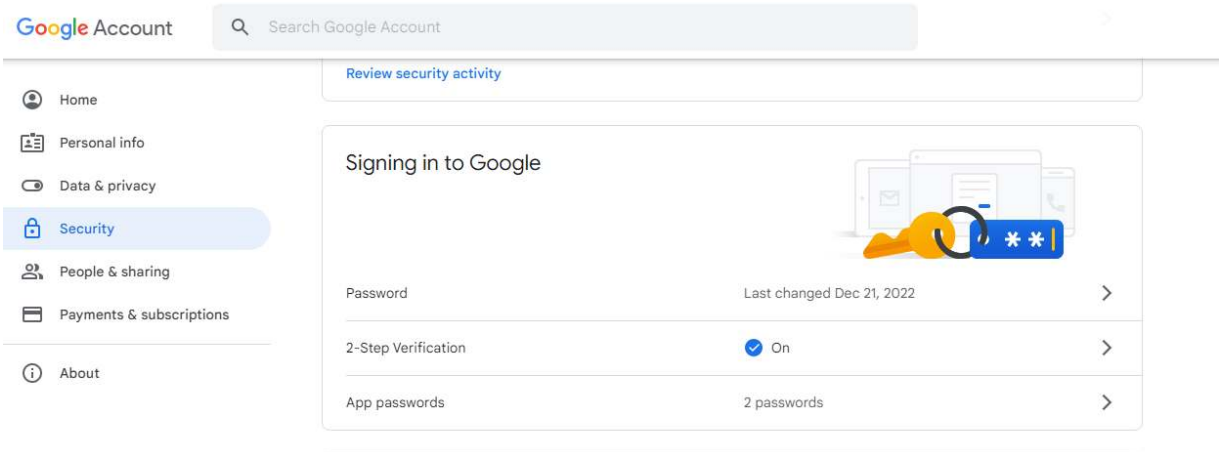


Figure 7.14.2: Click on Security and then scroll down to "Signing in to Google". Now open up "App passwords

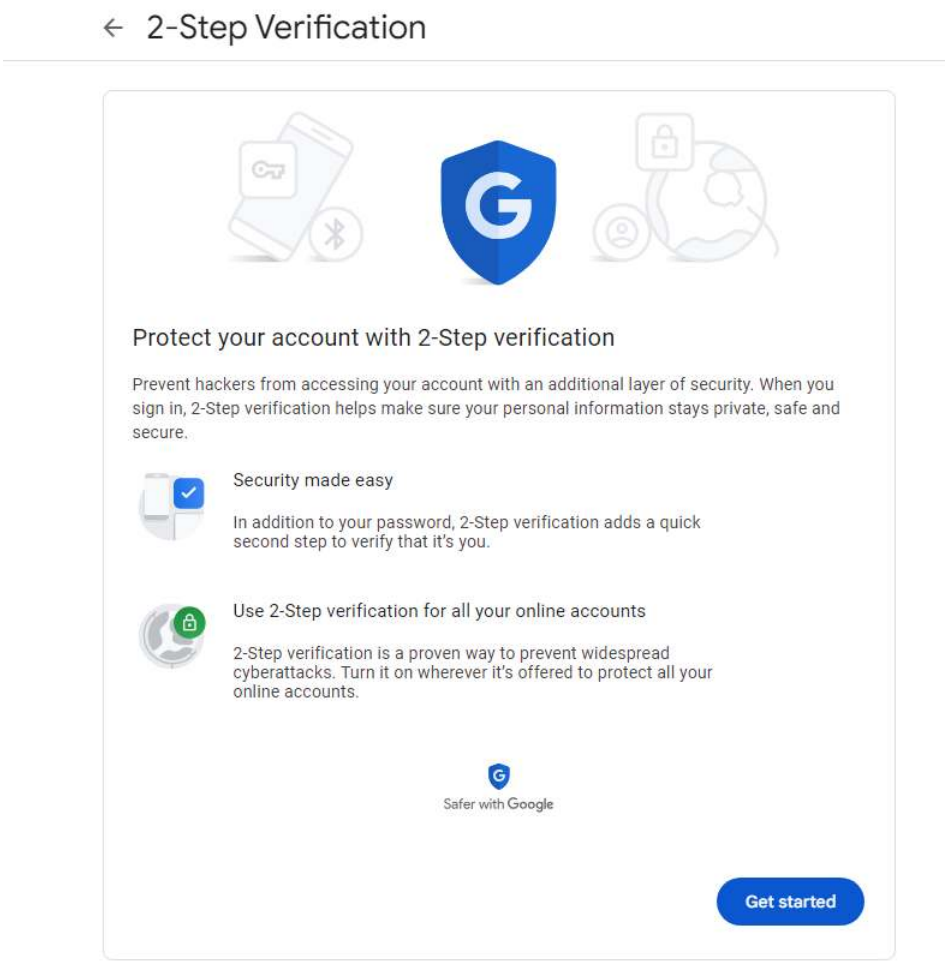


Figure 7.14.1: Copy and Paste Caption here. (Copyright; author via source)

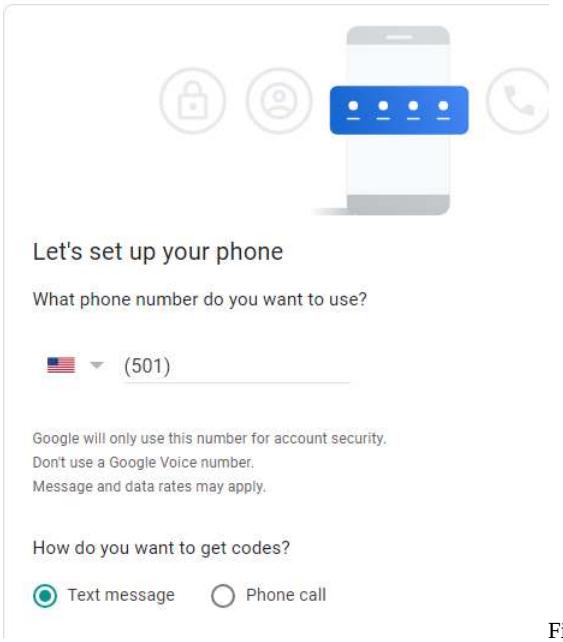


Figure 7.14.1: Copy and Paste Caption here. (Copyright; author via source)

You will get a text message with your code, include the G- when you submit it.

Now go to 2-Step Verification

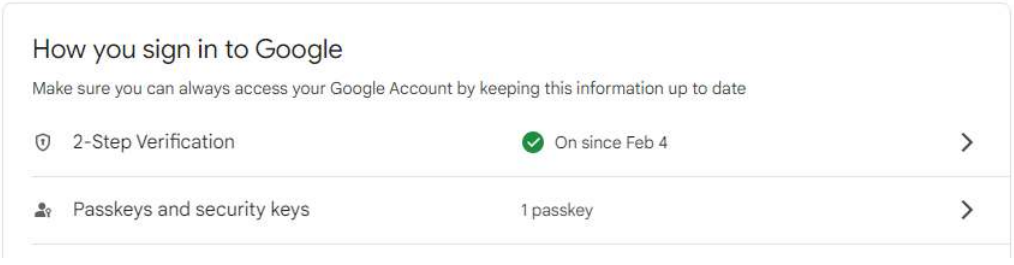


Figure 7.14.1: Copy and Paste Caption here. (Copyright; author via source)

click continue

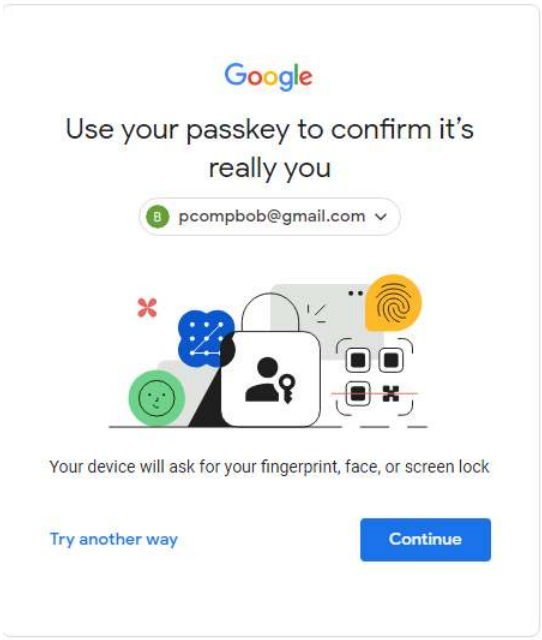


Figure 7.14.1: Copy and Paste Caption here. (Copyright; author via source)

Choose the PIN option and give the login pin for your computer

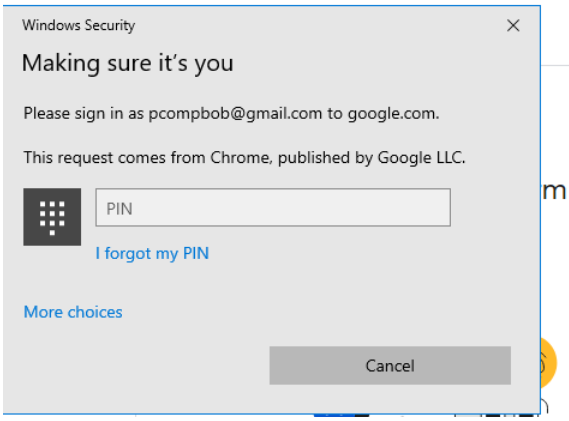


Figure 7.14.1: Copy and Paste Caption here. (Copyright; author via source)

Now you can see "App Passwords" at the bottom of the page, and you can generate a password to your gmail that your Raspberry Pi can use to log in.

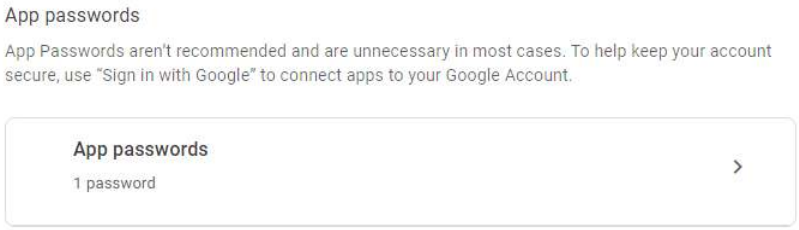


Figure 7.14.1: Copy and Paste Caption here. (Copyright; author via source)

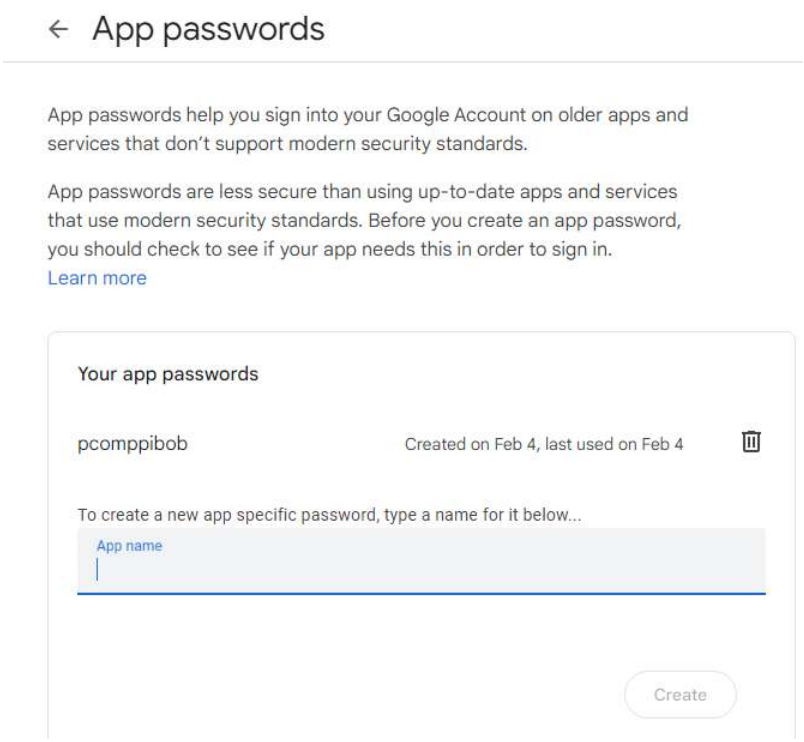


Figure 7.14.3: On select App, choose Mail. then on Select Device, choose a name for your Pi, and then click generate. This will give you a 16 digit key that you can use as a password when you code on your Pi. Note there are currently two Pis up. Realize your password will be put into the code of programs on your Pi, and so anyone who can access your Pi can get your password

Note, Google's interface evolves over time and these screenshots will become out of date, but the functionality is the same.

App Passwords

For obvious reasons Google is always changing how they verify third party apps and your normal gmail loging will not work. Your UALR account will probably not work either, and you want to use a special email account for this class, as your passwords will be in the scripts of multiple programs. You will need to generate a 16 digit key, and you will need to do this for every app that you use. Note, if you change your account password all app passwords are revoked.

Go to

- Passkeys and security keys
- create a passkey
- click more choices pin

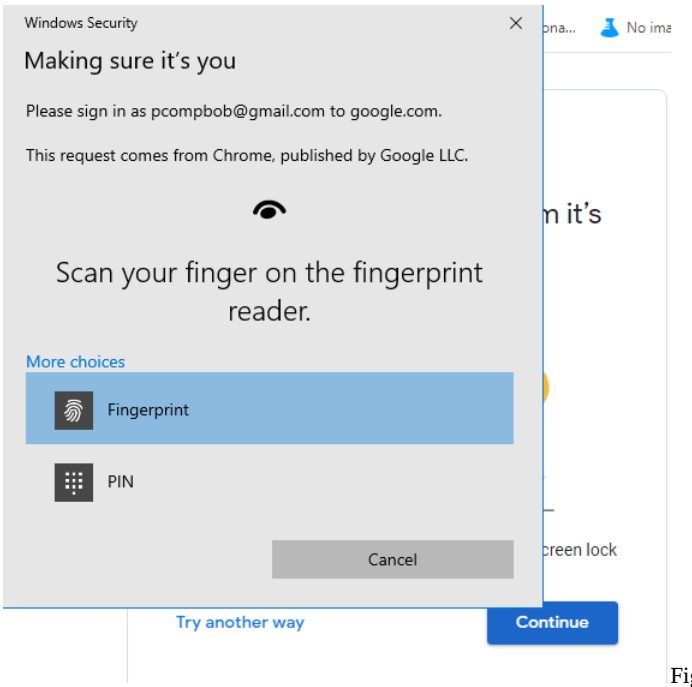


Figure 7.14.1: Copy and Paste Caption here. (Copyright; author via source)Enter the pin for your computer

- Enter your pin
- Go to the bottom and you will find App passwords, there youcan create a key that will let an app sign into your Google account.

As of 2024/0204 the instructions for creating an app password can be found here: Sign in with app passwords , but these only work if you choose the pin option and you must have two party authentication turned on.

This page titled 7.14: Google email account for class is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

7.15: Sending email without attachment

Raspi Directory Structure

This is your first assignment and it is imperative that you organize your files in a functional directory (folder) structure. I suggest the following hierarchy:

1. Desktop
    1. 2024_pcomp (create a folder on your desktop called "2024_pcomp")
        1. Assn_#on (create folders for each assignment within the 2024_pcomp folder, in this case create one for Asn_7)
        2. Cron (create a folder called "cron" in your class folder). We will place in this folder files that use cron jobs, and this assignment will include some of those.

Before proceeding you must set up a Google account for this class. You have two options, open Thonny and paste the following code in the IDE and then save to the Assn_7, or download the file from GitHub. Now the file from Git Hub will be color coded based on your IDE settings, and anything in parenthesis will be green. You only need to change 3 red items in the code below. BUT, internet security is an ongoing battle and the email password that allows third party applications (apps) like this program are of concern to Google, and so they are continually improving their services, which means what worked yesterday may not work tomorrow.

```
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from email.mime.base import MIMEBase
from email import encoders

email_user = "type here your_iost_account@gmail.com"
email_password ="type here your_iost_account@ gmail KEY here"
email_send = "type here the email_address you want to send an email to"

subject = "type here the subject line of your email"

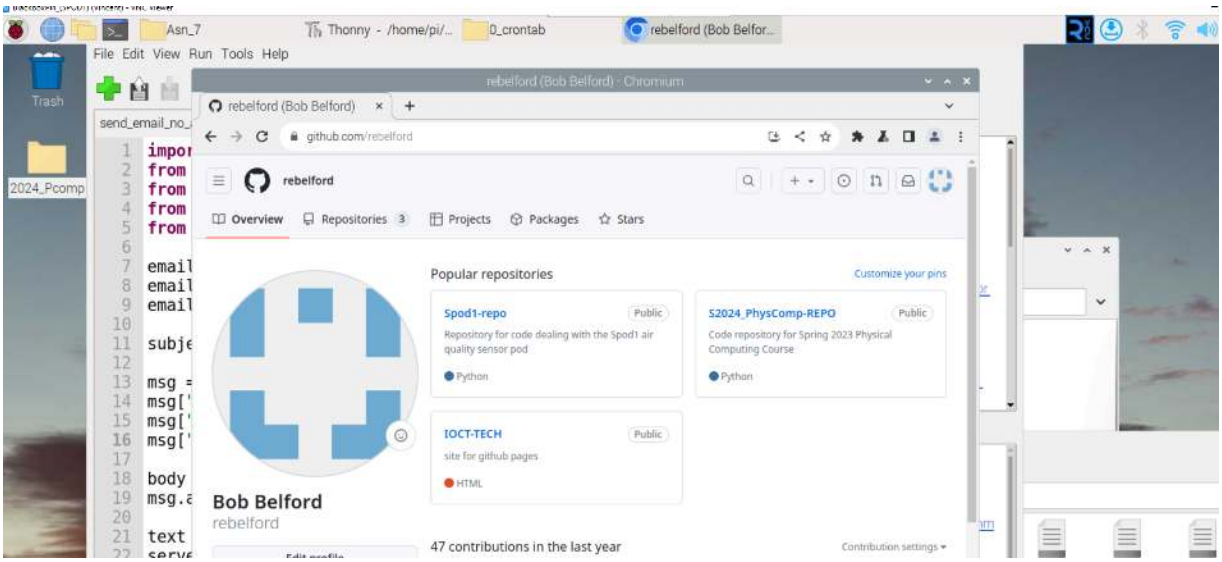
msg = MIMEMultipart()
msg["From"] = email_user
msg["To"] = email_send
msg["Subject"] = subject

body = "Type here the message you wish to send"
msg.attach(MIMEText(body,"plain"))

text = msg.as_string()
server = smtplib.SMTP("smtp.gmail.com",587)
server.starttls()
server.login(email_user,email_password)
server.sendmail(email_user,email_send,text)
server.quit()
```

Using GitHub on raspi

Open Chromium on your raspberry Pi and go to github.com/rebelford



New Figure 7.15.1: Copy and Paste Caption here. (Copyright; author via source)

Open the S2024_PhysComp-REPO, navigate to the 7-Getting-started-with-Raspi folder and open the send_email_no_attachment.py file and then chose the download option

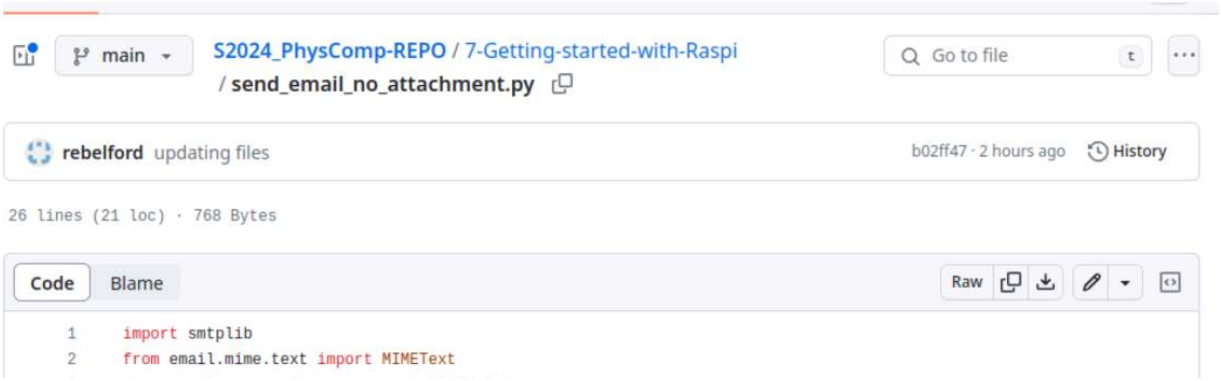


Figure 7.15.1: Copy and Paste Caption here. (Copyright; author via source)

Open the downloads folder and move the file to the /home/username/Desktop.2024_Pcomp/Asn_7

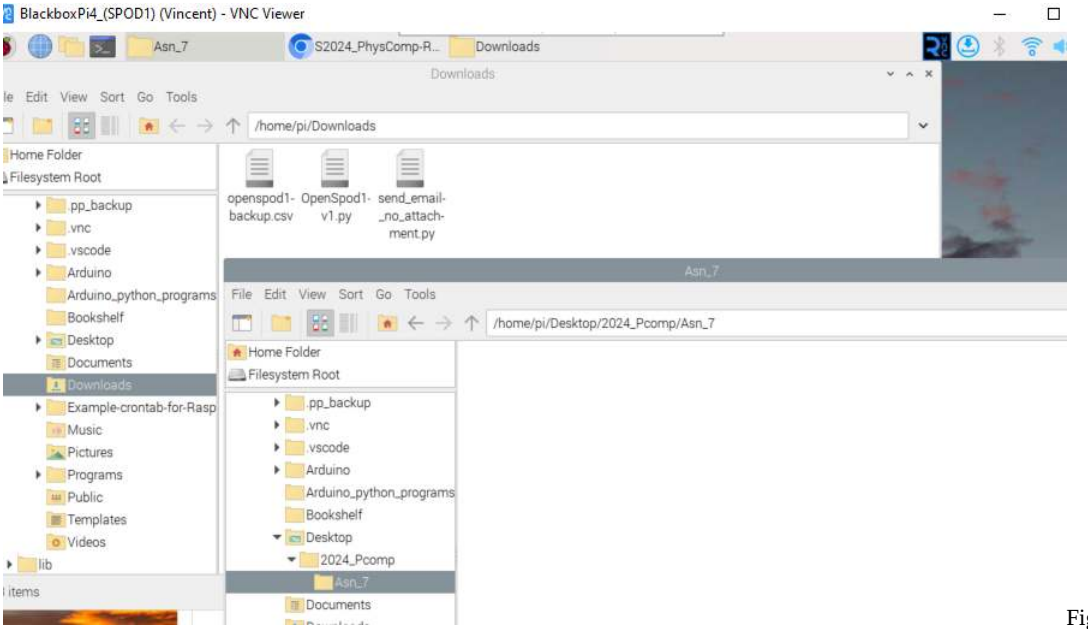


Figure 7.15.1: Copy and Paste Caption here. (Copyright; author via source)

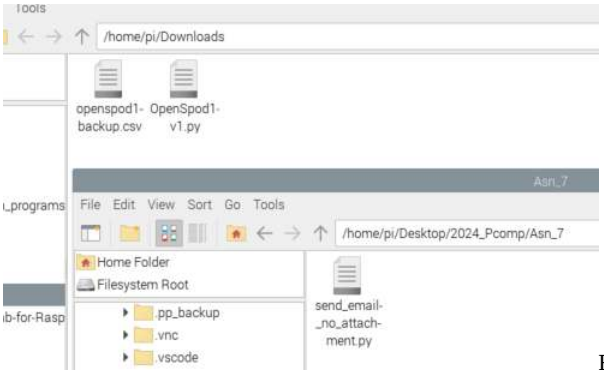


Figure 7.15.1: Copy and Paste Caption here. (Copyright; author via source)

Open the file in Thonny and change the text in lines 7,8,9,11, and 18.

send_email_no_attachment.py * | send_my_ipaddress.py

```
1  import smtplib
2  from email.mime.text import MIMEText
3  from email.mime.multipart import MIMEMultipart
4  from email.mime.base import MIMEBase
5  from email import encoders
6
7  email_user = "pcompxxx@gmail.com"
8  email_password ="xxxxxxxxxxxx"
9  email_send = "xxxxxxxx@ualr.edu"
10
11 subject = "type here the subject line of your email"
12
13 msg = MIMEMultipart()
14 msg["from"] = email_user
15 msg["to"] = email_send
16 msg["subject"] = subject
17
18 body = "Type here the message you wish to send"
19 msg.attach(MIMEText(body,"plain"))
20
```

Figure 7.15.1: Copy and Paste Caption here. (Copyright; author via source)

This page titled 7.15: Sending email without attachment is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

7.16: Sending email with attachment

Open Chromium on your Raspi and go to the github for this assignment and download the file to your assignment folder, then edit the stuff in red on the document below (or you can copy the code below into a new file)

```
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.base import MIMEBase
from email import encoders
mail_content = '''TYPE YOUR
MESSAGE
HERE'''
#The mail addresses and password
#change the next 3 lines
sender_address = 'YOUR_pcomp_CLASS_ACCOUNT@gmail.com'
sender_pass = 'YOUR_pcomp_key'
receiver_address = "EMAIL_YOU_WANT_TO_MAIL_TO"
#Setup the MIME
message = MIMEMultipart()
message['From'] = sender_address
message['To'] = receiver_address
message['Subject'] = 'A test mail sent by Python. It has an attachment.'
#The subject line
#The body and the attachments for the mail
#set the following line to the path of the file
#NOTE: you can right click on a file in the Raspi File Manager and "copy path"
attach_file_name = 'PUT PATH AND FILE NAME YOU WANT TO ATTACH HERE'
message.attach(MIMEText(mail_content, 'plain'))
attach_file = open(attach_file_name, 'rb') # Open the file as binary mode
payload = MIMEBase('application', 'octate-stream')
#payload.set_payload((attach_file).read())
payload.set_payload((attach_file).read())
encoders.encode_base64(payload) #encode the attachment
#add payload header with filename
#payload.add_header('Content-Decomposition', 'attachment', filename=attach_file_name)
payload.add_header('Content-Disposition', 'attachment', filename=attach_file_name)
#message.attach(attach_file_name,maintype='image')
message.attach(payload)
#Create SMTP session for sending the mail
session = smtplib.SMTP('smtp.gmail.com', 587) #use gmail with port
session.starttls() #enable security
session.login(sender_address,sender_pass) #login with mail_id and password
text = message.as_string()
session.sendmail(sender_address, receiver_address, text)
#to send more than one email
#session.sendmail(sender_address, receiver_address.split(","), text)
```

```
session.quit()
print('Mail Sent')
```

This page titled 7.16: Sending email with attachment is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

7.17: Find IP Address

Open Chromium on your Raspi and go to the github for this assignment and download the file to your assignment folder, or create a new python file with the code below, and test it

In this script we are creating a user defined function called get_ip_address() by using the "def" command.import socket

```
import socket
import fcntl
import struct
def get_ip_address(ifname):
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    # uncomment the following line if using python2
    #return socket.inet_ntoa(fcntl.ioctl(s.fileno(),0x8915,struct.pack('256s',ifname[
    # uncomment the following line if using python3
    return socket.inet_ntoa(fcntl.ioctl(s.fileno(),0x8915,struct.pack('256s',bytes(if
address = get_ip_address('wlan0')
#change wlan0 to eth0 if you want the wired IP address
print(address)
```

Note, we are passing wlan0 as the argument. If you are using ethernet you would pass eth0 as the argument



Figure 7.17.1: Copy and Paste Caption here. (Copyright; author via source)

In Thonny go to Tools/Manage Packages and search for fcntl, and install it

This page titled 7.17: Find IP Address is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

7.18: email IP address

Open Chromium on your Raspi and go to the github for this assignment and download the file to your assignment folder, then edit the stuff in red on the document below (or you can copy the code below into a new file)

```
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from email.mime.base import MIMEBase
from email import encoders
#for send IP
import socket
import fcntl
import struct

email_user = "my_sacrifical@gmail.com"
email_password = "my_sf_password"
email_send = "EMAIL_2B_NOTIFIED"


#create function to get ip address
def get_ip_address(ifname):
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    return socket.inet_ntoa(fcntl.ioctl(s.fileno(),0x8915,struct.pack('256s',bytes(if

#assign IP address to variable address
address = get_ip_address('wlan0')

subject=str(address)

msg = MIMEMultipart()
msg["From"] = email_user
msg["To"] = email_send
msg["Subject"] = subject

body="The new IP address is " + str(address)

msg.attach(MIMEText(body,"plain"))
text = msg.as_string()
server = smtplib.SMTP("smtp.gmail.com",587)
server.starttls()
server.login(email_user,email_password)

server.sendmail(email_user,email_send,text)
server.quit()
```

This page titled 7.18: email IP address is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

7.19: Cron and Crontab

Cron

Cron is a standard utility of the Linux operating system that allows for the automation of tasks. It is a daemon, which is a process that typically starts on boot-up and runs in the background. The Cron daemon is a time-based scheduler and gets its name the Greek word for time χρόνος (chronos). A Cron job is a task run by Cron that occurs on boot-up, or during a specified time. Cron is typically used by system administers to perform periodic processes. Examples of processes that could be scheduled with Cron include:

- Backing up important data and files
- Updating software and packages
- Disk cleanup and defragmentation
- Sending automated emails on a schedule
- Sending an email with IP address on boot-up (our first cron job)

Both Cron and Python can be used to schedule the reading of sensor data and subsequent transmission to the web. The fastest Cron can schedule a job is every minute and so it will not work if you need data faster than that. The advantage of using the Cron daemon to read sensors and transmit data to the web is that you do not have to load Python into memory for the entire process, only when you are actively calling the program. It also allows you to keep your Python scripts simpler. But you can not use Cron for conditional scheduling (like sending data when a specific sensor value is acquired) or real-time interactions.

Crontab

Crontab (Cron Table) is a file you can access through the command line and contains the tasks you wish to schedule for Cron. Each user can have their own Crontab and Cron will read each user's crontab and if it has changed, upload the new list to the cron. So the Crontab is how you interact with Cron. The crontab essentially tells cron what files you want to run, and when to run them.

To edit crontab type

```
pi@hostname:~ $  crontab -e
```

The first time you do this, you will be prompted to select your favorite editor. We will select 1 for the nano editor. If a crontab exists you will see its content. At the beginning of this class we will create a crontab that will email your IP address on bootup. We will do this because you will need your IP address to access your Raspberry Pi through VNC viewer, and instructions are here.

At a later point in the semester we will create a functional Crontab file (lesson here). The approach we will use is to download a template called crontab.src that has common tasks commented out. We will then call up various Python scripts by putting the commands into the crontab. It is very important that your python scripts have complete paths for any files that are called because you will be running them from Cron.

Resources

- Cron Job: A Comprehensive Guide for Beginners 2023
    - Very good tutorial and description of basic crontab features
- Cron (Wikipedia)
- Geeks for Geeks

7.19: Cron and Crontab is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

7.20: First Cron Job

Information

The software utility cron is a time-based job scheduler in Unix-like computer operating systems. People who set up and maintain software environments use cron to schedule jobs (commands or shell scripts) to run periodically at fixed times, dates, or intervals. It typically automates system maintenance or administration—though its general-purpose nature makes it useful for things like downloading files from the Internet and downloading email at regular intervals. (from Wikipedia)

We can set up a cron job to run our ip address emailer every time we boot our raspberry pi. That way if we forget to attach the monitor, or want to connect remotely, we have its IP address to connect. To do this at your prompt type the following:

```
pi@hostname:~ $  crontab -e
```

The first time you do this, you will be prompted to select your favorite editor. We will select 1 for the nano editor. At the bottom of the document add the following line. Note, you will need to edit this to the path and name of your file that emails the IP address.

```
@reboot sleep 60 && python3 /home/pi/python_programs/send_my_ipaddress.py
```

Now reboot your Pi and see if you get an email with the IP address. If so, you are done with this part of the assignment.

This page titled 7.20: First Cron Job is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

CHAPTER OVERVIEW

8: Python Functions, Packages and Formatting

Topic hierarchy

8.1: Functions
8.2: Introduction to Functions
8.3: Built-In Functions
8.4: Standard Library Functions
8.5: Third Party Library Functions
8.6: User-Defined Functions
8.7: User Created Package
8.8: Print Formatting Overview
8.9: Modulus Method
8.10: 3.format() method
8.11: f-string formatting
8.12: Subscripts and Superscripts

8: Python Functions, Packages and Formatting is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

8.1: Functions

Functions are blocks of code that perform specific tasks or sets of task. A typical function has an input (arguments or parameters) that it performs some kind of process on and then returns some kind of output. This can help organize code associated with a complex project and make it more readable. Processes that are repeated often are ideal processes to make a function out of. Book 5 section 4 is devoted to functions and we will see that there are standard built-in functions like the print() statement that load everytime you run python, standard library functions that come with python but that you must import to your code to use, third party library functions like those that could be associated with reading values of a specific sensor and you can even make your own user-defined functions.

Anatomy of a Function

- Input
  - Functions operate on data or perform some task that they operate on.
  - In python this information is enclosed in parenthes.is after the function name
  - **Parameters** are the types of variables a function operates on, it defines the kind of argument a function can accept
  - **Arguments** are the actual data values submitted to a function and upone which it operates on.
- Function Body
  - Instructional programmatic steps the function performs on the data.
  - This is the program that is run when the function is called.
- Output
  - This is the value or action that is returned when a function is called.

Then all you need to do is call the function and pass it any arguments it needs.

Python Example of User Defined Function

In the following code we create a function called "molar_mass_fct" and gave it the code from the flowchart of the if-else statement associated with the pseudocode of the last activity. All python functions must be followed by parenthesis. Do not worry if you do not understand the syntax as we will get to that later this semester. Right now you need to know what a function is and what it does. But hit run, and try some values, and note that there are only two lines of code outside of the function, the input molar mass line, and the function call line. Also note that in python a comment is indicated with a # sign, so the lines starting with a # are not run as code



Login with LibreOne to run this code cell interactively.

If you have already signed in, please refresh the page.

Login


```
def molar_mass_fct(molar_mass):
    if molar_mass >= 2.0:
        print(f"The molar mass you entered is: {molar_mass}g/mol")
    else:
        molar_mass=float(input("Enter a molar mass greater than or equal to diatomic
        print(f"The molar mass you entered is: {molar_mass}g/mol")
```

```
#Input molar mass
molar_mass=float(input("Enter the molar mass of a molecule in units of g\mol "))
#call function
molar_mass_fct(molar_mass)
```

```
Hello world!
```

The function in the above code was defined by this code block. It is given the name "molar_mass_fct" and operates on the "molar_mass" variable

```
def molar_mass_fct(molar_mass):
    if molar_mass >= 2.0:
        print(f"The molar mass you entered is: {molar_mass}g/mol")
    else:
        molar_mass=float(input("Enter a molar mass greater than or equal to diatomic
        print(f"The molar mass you entered is: {molar_mass}g/mol")
```

In the next line of code we input a value (the argument) that we assign to the variable molar mass. Note we used two built-in functions in this line of code. The first was the input() function, but that returns a string value (letter) and we need to convert it to a floating number so we can sue the >= comparison operator within the function body, and so we placed that function inside of the float function. the result is we have now assigned the variable molar_mass the value of the floating number input by the user

```
molar_mass=float(input("Enter the molar mass of a molecule in units of g\mol "))
```

In the last step we call the function and pass it the argument (value input by the user) and it then either says the value is too small to be a molar mass, or it outputs the value.

```
 molar_mass_fct(molar_mass)
```

Flowchart Simplification

In the following diagram we show how making a function to check the molar mass is larger than two simplyfies the flow chart to just an input statement and a function call statement. The material on the top part of the image in red dots is the function, and this shows how a function can simplfy the code and make it more readable.

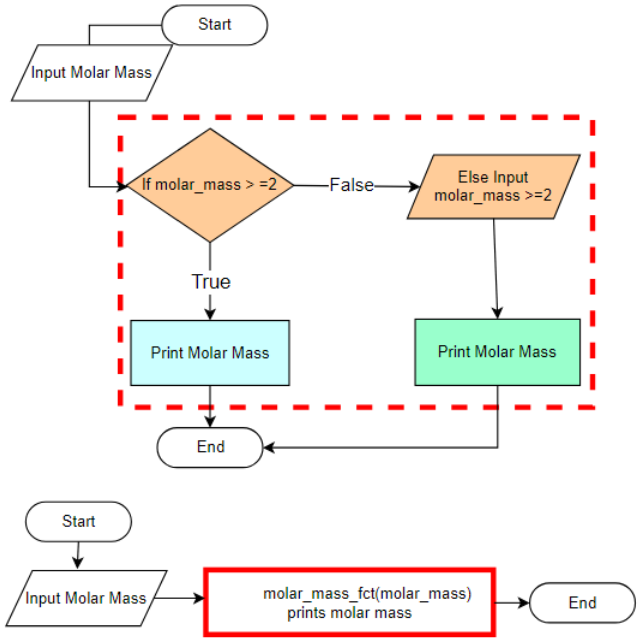


Figure 8.1.1: Two flowcharts doing the same thing. The top one does not use a function, while the bottom one calls a function that does the operations in the dotted box. Once the function has been made, it can be used by multiple programs, and this is why there are libraries (or packages) in python containing functions and other features. (Belford CC-BY)

8.1: Functions is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

8.2: Introduction to Functions

In Python a function is a block of code that performs some task. Functions are objects of the class "function". They can be assigned to a variable, passed as arguments to other functions and returned as values from other functions. In Python 3 a function is always followed by open and closed parenthesis (), within which you pass the arguments of the parameter. You need the parenthesis, even if there are no arguments.

function_name (parameter_1, parameter_2...)

- Parameter - variable name that function uses when it is called (molecule_name, molar_mass)
- Argument - value passed to the function (aspirin, 180.16)

There are several basic types of functions we will be using.

Type of Function	Description
Built-in	These are functions that are available when you load Python
Imported Standard Library Functions*	These are imported from packages or libraries that come with Python but are not loaded into memory when you start it up, and so you need to import them.
Imported Third Party Library Functions*	These are functions from third party packages that you can download from places like PyPi.org and GitHub. Once installed, they behave like standard library functions
User-defined	These are functions you create. You could put them in a package and import them into another program
methods	Many classes of objects have methods, which are functions associated with that class. For example a list has an append function, which adds an item to the list

* A library is a suite of python objects and can contains functions and classes. Classes are a type of entity in python, like strings are a class, and they have attributes and methods associated with them. One of the advantages of Python is you can create classes, or subclasses out of existing classes that are designed for specific applications. If you think about it, a number is a number, but a float, integer and string are all stored in the computer differently, and have different characteristics.

Note

Why does Python have standard library and built in functions, when they both come with the program when you install it?

The reason deals with efficiency and memory. An analogy could be your desk and books. Your desk is your workspace, like the memory is the computer's workspace, and if you want to read a book in the library, you first must get it, which is what the "import" statement does, and now that object is available in your workspace. But, what if there is a book that is not in the library? Then you need to go out and get it, and that is like the third party packages and modules. But once the book is in your library, you have access to it, and so can import it.

This page titled 8.2: Introduction to Functions is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

8.3: Built-In Functions

Built-in functions are loaded into memory when you start Python, and are available to use without having to import them
A list of built-in functions can be found at docs.python.org.
We will look at some common built-in functions.

float()

- convert string or integer to a floating decimal number

```
a="1"
print(a)
print(type(a))
a=float(a)
print(10*"*")
print(a)
print(type(a))
```

run | restart | restart & run all

```
Hello world!
```

input()

- Allows you to input a value
- All inputs are strings

```
a=input("Enter a number")
print(a)
print(type(a))
b=input("write a sentence")
print(b)
```

run | restart | restart & run all

```
Enter a number 3
```

int()

- Converts a string or float to an integer

```
a="1"
print(a)
print(type(a))
a=int(a)
print(10*"*")
print(a)
print(type(a))
```

run | restart | restart & run all

```
Hello world!
```

len()

- returns length of an object

```
print(len('hello world'))
print(len("123456789"))
```

run | restart | restart & run all

```
hello world
```

Math Functions

abs()

- Returns absolute value of a number

```
print(abs(-34.5))
print(abs(34.5))
```

run | restart | restart & run all

```
hello world
```

pow(base,exponent)

- returns base to the power of the exponent

```
print(pow(2,3))
print(pow(10,3))
```

run | restart | restart & run all

```
hello world
```

round()

- Rounds up or down a number

```
print(round(33.545))
print(round(33.545,1))
print(round(33.545,2))
print(round(33.545,-1))
```

run | restart | restart & run all

```
Hello world!
```

max()

- gives largest item of an iterable, or smallest of two or more values

```
print(max([2,8,4]))
print(max(["alpha","gamma","beta"]))
```

run | restart | restart & run all

```
Hello world!
```

min()

- gives smallest item of an iterable, or smallest of two or more values

```
print(min([2,8,4]))
print(min(["alpha","gamma","beta"]))
```

run | restart | restart & run all

```
Hello world!
```

print()

- strings must be in parenthesis
- \n gives new line
- \t is tab
- multiple format options

Note: Print Formatting

Note, there are multiple ways you can format the output of the print statement and because this is very important we have a section devoted purely to print formatting.

str()

- converts an integer or floating decimal to a string

type()

- returns the type of an object

This page titled 8.3: Built-In Functions is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

8.4: Standard Library Functions

There are many modules that do not load into memory when you start Python, but they come with the installation, and these contain functions that you can use, but you must import them to use them. It is best practice to put all of your import statements at the top of your code, so someone who is reading it can instantly see what modules you need to import. These modules are single files that can define functions, classes and variables. To use a function you treat it as a class method, with the following syntax

```
module_name.function_name(arguments)
```

To find what type of modules you have installed you can run the following command on your computer or Raspberry Pi

```
#the following code will list the modules installed on your computer.
help('modules')
```

If you want to see what modules are running on the Jupyter Hub of LibreText you can run the command below. It will take a minute, and probably give you a few warnings.

```
#the following code will list modules installed on LibreText's Jupyter Hub
#Note: you may get multiple warning for modules being deprecated, just be patient, and
help('modules')
```

run | restart | restart & run all

```
  Hello world!
```

To run a module you first need to import it

Import Statement

The import statement loads a module into the program's memory. Please go to the modules section to learn about specific modules. To see this, we will look at the random module. The following code will generate an integer between 1 and 999.

```
random_integer=random.randint(1,999)
print(random_integer)
```

run | restart | restart & run all

```
          hello world
```

The above failed because we did not load the random module into memory

```
import random
random_integer=random.randint(1,999)
print(random_integer)
```

run | restart | restart & run all

```
          hello world
```

Note, if you ran the code above, you loaded the random module into memory, and you do not need to reload it (the code that did not work will now work).

```
random_integer=random.randint(1,999)
print(random_integer)
```

run | restart | restart & run all

```
          hello world
```

Checking the class of the random module

```
import random
print(type(random))
```

run | restart | restart & run all

```
  Hello world!
```

Some Common Standard Modules

Here are some of the standard modules we will use in this class. Any code that uses these functions must first import them.

datetime

- Docs Page
- offers different classes for working with time. Note, epoch time

```
import datetime
print(datetime.datetime.now())
print(datetime.date.today())
print(10*"*")
now=datetime.datetime.now()
print(now.strftime("%H:%M:%S"))
```

run | restart | restart & run all

```
  Hello world!
```

math

- Docs Page
- Provides basic mathematical operations. Note, cmath deals with complex numbers

```
import math
print(math.pi)
print(math.ceil(3.4))
print(math.floor(3.4))
print(math.pow(2,8))
print(math.sqrt(16))
```

```
#to convert 120 degrees to radians
rad=math.radians(120)
#to get the cosine of 120
print(math.cos(rad))
print(math.sin(rad))
```

run restart restart & run all

```
Hello world!
```

random

- Docs Page

```
import random
print(random.randint(1,100))
print(random.randint(1,100))
print(10*"*")
print(random.randrange(0,100,2))
print(random.randrange(0,100,5))
print(random.randrange(0,100,10))
print(random.choice('word'))
even_numbers=[2,4,6,8]
random.shuffle(even_numbers)
print(even_numbers)
```

run restart restart & run all

```
Hello world!
```

Random Seed

Random seed lets you reproduce a random number every time.

```
import random
random.seed(111)
#run this code multiple times and you always get 28,41,64
print(random.randint(1,100))
print(random.randint(1,100))
print(random.randint(1,100))
```

run restart restart & run all

```
Hello world!
```

re Regular Expressions

Socket

- Docs Page
- Real Python Tutorial

The socket module lets you work with network protocols and enable communication between devices over the network. we use it in the "Find IP Address" code that we use to get the IP address of the Raspberry Pi

Statistics

- Docs Page

```
import statistics
print(f"The median is the value in the middle of a data set of 2,2,2,4,6, which is: {s
print(f"The mean is the average value of 2,2,2,4,6, which is: {statistics.mean([2,2,2
print(f"The mode is the most common value of 2,2,2,4,6, which is: {statistics.mode([2
print(f"The standard deviation is the square root of the varians, which for 2.11,2.20
```

run | restart | restart & run all

```
Hello world!
```

Subprocess

- Docs Page
- This allows you to start other processes from your python code, and these can be run in different languages. For example, the following code on a windows machine will read the current directory from the command line. NOTE: if you are on a WINDOWS machine, change the "ls" to "dir", as that is the windows command for showing files in the current directory.

```
import subprocess
result = subprocess.run(["ls"], shell=True, capture_output=True, text=True)
print(result.stdout)
```

So the above code runs the BASH command ls from python

time

- Docs Page
- provides functions to work with time related tasks
- epoch time is the number of seconds since Jan 1, 1970
- the sleep function is very useful in data collection, but you must be careful if you run it before or after a process, as in the later case, the process time is added to the sleep time.

```
import time
print(time.gmtime(0))
print(time.gmtime())
print(f"time since epoch: {time.time()} seconds")
print("print now")
time.sleep(3)
print("it is 3 seconds later")
```

run | restart | restart & run all

```
Hello world!
```

urllib

This page titled 8.4: Standard Library Functions is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

8.5: Third Party Library Functions

Third party functions can be downloaded as packages or libraries, which are collections of modules. These are very similiar to each other, with the main difference being that a package has a directory structure that includes an __init__.py file that initializes the package. So a library can be organized as a package, but it does not have to be. When it is, you can bring up functions within modules as methods [module_name.method_name].

The following modules are at 8: Third Party Library Modules

1. Requests
2. Gpiozero (standard on Raspberry Pi)

PyPi Python Package Index

The python Package Index (https://pypi.org/) is an excellent source of third party python packages and as of 2024-02-07 there are 512,807 projects with 10,234,789 files from a community of 782,675 users.

To install a third party package from PyPi you use the pip install command

```
pip install package-name
```

You bought a 5 pack of DS18B20 temperature sensors and want to find some code.

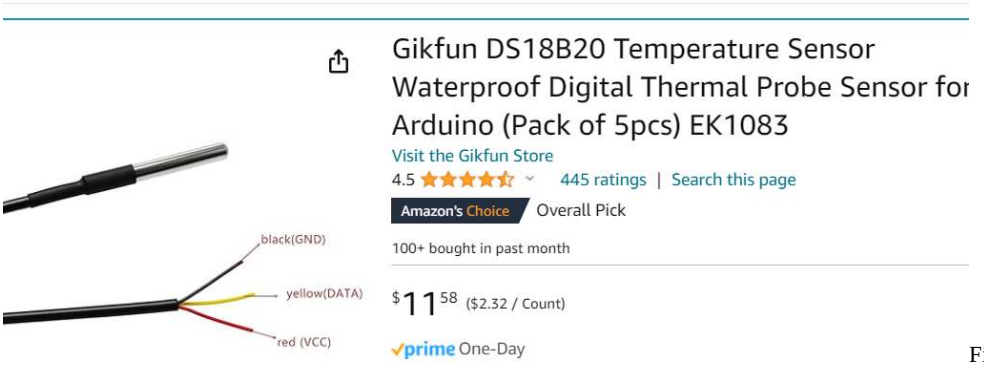


Figure 8.5.1: Copy and Paste Caption here. (Copyright; author via source)

Go to PyPi and do a search for "Raspberry Pi DS18B20 temperature sensor" (figure 8.5.2

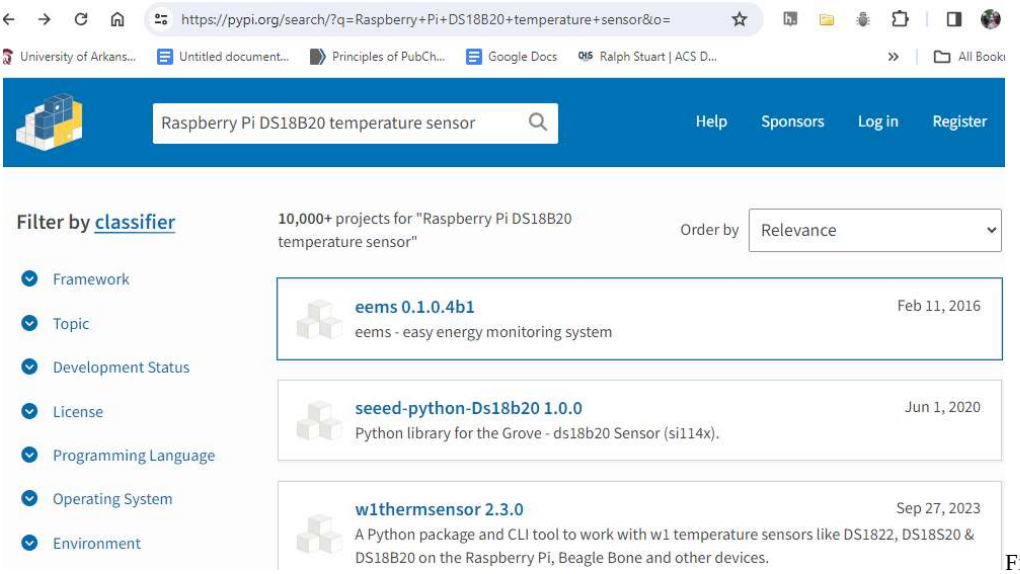


Figure 8.5.2: Search for DS18B20 sensors on 2024-02-07. (Copyright; author via source)

PyPi Search Tip

When searching it is often prudent to open several links in new tabs and quickly skim through them until you find one you like. I typically look at the first 3-5 hits, review them and close them if I do not see what I want, and then if needed, extend my search with a new tab.

The third hit of the above search sends you to a package that was updated within the last 6 months, https://pypi.org/project/w1thermsensor/.

GitHub

GitHub (https://github.com/) is of course one of the best places to look for code. Unlike the PyPi Python Package index (that is devoted to Python Packages) GitHub covers non-Python code and so you may want to alter your search to Raspberry Pi Python DS18B20 temperature sensor, and the first hit is the same hit as hit 3 in PyPi

Instructables

Instructables (https://www.instructables.com/) is a website hosted by AutoDesk that allows people to post all sorts of projects, and these include both hardware, wiring and code.

Vendors

Many "Maker Community" vendors supply code for their products, and this incomplete list will be growing.

AdaFruit

AdaFruit (https://www.adafruit.com/)

This page titled 8.5: Third Party Library Functions is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

8.6: User-Defined Functions

In Python you can make your own functions with the "def" statement.

Note, you must end it with a colon, and all of the lines in the function statement must be indented, as in the following generic code.

```
#Syntax for a Function
def function_name():
    action1
    action2....

Outside of Function
```

Density Function

The following code creates a function called "density_function"

Codebox1:



Login with LibreOne to run this code cell interactively.

If you have already signed in, please refresh the page.

Login

```
def density_function():
  mass = float(input("Enter mass in g"))
  volume = float(input("Enter volume in mL"))
  density=mass/volume
  print(f"{mass}g and {volume}mL give a density of {density}g/mL")
```

```

```

Exercise 8.6.1

Click the run command for the density_function (Codebox 1). You notice nothing happens. Why? Is there an error?

Answer

All you did was create the function. There is no error, you never called the function.

Now call the density function by clicking run in codebox 2.

Codebox 2:

Login with LibreOne to run this code cell interactively.

If you have already signed in, please refresh the page.

Login

```
density_function() #this command calls the function
```

? Exercise 8.6.2

Now refresh the web page so it reloads, and call the density function without running it (run Codebox 2 without running Codebox 1). You get an error statement. Why?.

Answer

You have a name space error because you have not created the density function.

```
NameError                                 Traceback (most recent call last)
<ipython-input-1-22a51af5250c> in <module>
----> 1 density_function() #this command calls the function

NameError: name 'density_function' is not defined
```

Now run Codebox 3 without running Codebox 1:

Codebox 3:

Login with LibreOne to run this code cell interactively.

If you have already signed in, please refresh the page.

Login

```
def density_function():
  mass = float(input("Enter mass in g"))
  volume = float(input("Enter volume in mL"))
  density=mass/volume
```

```
  print(f"{mass}g and {volume}mL give a density of {density}g/mL")
density_function()
```

```
        hello world
```

? Exercise 8.6.3

Now after running codebox 3 go ahead and run codebox 2 (without running codebox 1). Why does it work?

Answer

Because the last line called the function

This page titled 8.6: User-Defined Functions is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

8.7: User Created Package

This activity is correlated to Assignment 3. In your class folder we want you to create a subfolder Sandbox and in it we want you to create a file called my_script.py and subfolder called My_toolbox. and

So the directory hierarchy looks like this, where the folders start with a captial letter and the files all end with .py

- Sandbox
  - my_script.py
  - Tools
    - __init.py__
    - my_tool_box.py

Python files

my_script.py

This file is in the Sandbox and located next to the Tools directory (folder). It will call functions from a module within the Tools package. We will use the follow code for this.

```
#Script 1: my_script.py
from Tools import my_toolbox as mtb
num1=float(input("Enter first number: "))
num2=float(input("Enter second number: "))
value=mtb.addition(num1,num2)
print(f"The sum of {num1} and {num2} is {value}.")
print(f"the type for the variable now is {type(value)}.\n")

now = mtb.current_date()
print(f"The current date is {now}.")
print(f"the type for the variable now is {type(now)}.")
```

__init__.py

Place this file in the Tools directory. The __init__.py (and those are two underscores) is used to mark a directory as a Python Package and when present in a folder, python treats the folder as a package, which means it can be imported into another program as a module. Right now we are going to keep the __init__.py file empty, but for example, if the files in the package had dependencies on other packages, like the time module, they could be set up in the __init__.py file

my_toolbox.py

Place this file in the tools directory. It will serve as a module where you can create functions (tools) that can be called from other programs.

```
#Script 2: my_toolbox.py
'''This package contains functions that can be called by other programs'''

def addition(num1,num2):
    the_sum=num1+num2
    return(the_sum)
```

```
def current_date():
    import datetime
    today=datetime.date.today()
    return(today)
```

Note, there are two functions in the second script; addition and current_time. The addition has two arguments that must be submitted when the first script calls it (num1 and num2 are passed from script 1 to script 2 as arguments) and it returns one value, the sum. The second function (current_time) has no parameters, and so you pass it nothing when you call it, but it returns the variable "today", which you assign to the variable "now" in the calling program

8.7: User Created Package is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

8.8: Print Formatting Overview

We are calling this print formatting, but really we are dealing with different methods for formatting the output of data, and different data types have different issues. For example, only a float has a decimal where you need to be concerned about how many digits after the decimal are formed. So the issues with integers are different than strings which are different than floats.

We will cover three formatting methods, in the order that they were used. I recommend students focus on the f-string method for code they write, as that is the simplest. But you may find code, especially older python 2 code that uses the older methods, and you need to be able to analyze and figure how it works.

Three Print Formatting Methods

1. Modulo (%) method
   - oldest and may be in packages you import
2. .format() method
   - set up like a function
3. f-string method
   - by far the easiest to use

It should be noted that all formatting methods are doing the same thing, they are formatting data output, and so they are all dealing with the same issues. Therefor it is a idea to quickly look over each method to get a bearing on the issues (and opportunities) you can encounter, and then choose the method that is best for your needs. In most cases, you will use the f-string method in original code you write. But if you borrow code that has used other methods, you might as well continue to use them, and so it is good to be familiar with them.

This page titled 8.8: Print Formatting Overview is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

8.9: Modulus Method

This is the older method and you will often find it in packages, especially code developed for older sensors, and so it is important that you are familiar with it, but we will not require you to write code using this method. You should still skim through this material, and be prepared to reference it when needed. Before we get into formatting syntax we need to identify the type, as the way a computer stores data depends on its type.

Types

Remember the type() function lets you know the type of object you are dealing with, and there are four common data types we deal with in formatting

- %s - represents strings
- %i - represents integers
- %d - represents decimal integers
- %f - represents floats

Generic Syntax

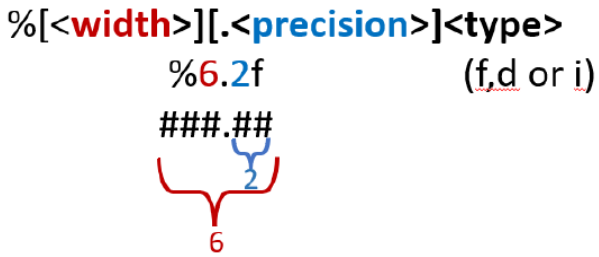


Figure \(\PageIndex{1}\):Generic syntax for formatting non string numbers. (Belford cc 0.0)

The following code depicts this for the float 1.00784

```
print("The molar mass of hydrogen is: %1.2f"%(1.00784))
print("The molar mass of hydrogen is: %10.2f"%(1.00784))
print("The molar mass of hydrogen is: %1.0f"%(1.00784))
print("The molar mass of hydrogen is: %1.5f"%(1.00784))
```

run | restart | restart & run all

```
Hello world!
```

Now go into the fourth line of the above code and change the %1.5f to a %1.5i and look at how the output is changed. Well, you are formatting a floating number as an integer type, which makes no sense.
Now lets format a couple of variables, one being a string and the other a float

```
#here we are assigning two variables in one line, and then printing them in a line
entity, molar_mass="water", 18.01528
print("The molar mass of %s is %.3f g/mol." %(entity,molar_mass))
print("The molar mass of %s is %.5f g/mol." %(entity,molar_mass))
print("The molar mass of %s is %.3d g/mol." %(entity,molar_mass))
```

run | restart | restart & run all

```
Hello world!
```

Exercise \(\PageIndex{1}\)

Can you explain the last line of code? Can you fix it?

Answer

You are formating a float as if it was a decimal integer, and so the precision is not what you want. you should change the %.3d to %.3f in the print statement and rerun the code

This page titled 8.9: Modulus Method is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

8.10: 3.format() method

Generic Syntax

"text {} text {} text {}." .format(var1, var2, var3)

- Can use index numbers
- Place width and precision in curly brackets ({}).

Index Numbers

```
#Demonstration of use of index numbers in .format() method
atom1, atom2, atom3 = "Hydrogen", "Helium", "Lithium"
print("This string has order first, second, third variable {} {} {}."\
      .format(atom1,atom2,atom3))
print("This string has order third variable {2}, second variable {1} "\
      "first variable {0}.".format(atom1, atom2, atom3))
print("This string has order first variable {0}, second variable {1} "\
      "first variable {0}.".format(atom1, atom2, atom3))
```

run | restart | restart & run all

```
Hello world!
```

? Exercise \(\PageIndex{1}\)

What value do index numbers in Python start with (what is the index number of the first variable?

Answer

0, atom one is the first variable and has an index number of zero.

Width and Precision

Analyze the following code, predict the output and run. If it is different than predicted, be sure to understand why.
Note; we are using the pipe symbol (|) to indicate the divider between two columns in our data output. The first column is the names of elements are they are string variables, and the second are their molar masses, and they are floats

```
#Demonstration of use of width and precisioin of .format() method
print("{0:10}|{1:10}\n{2:10}|{3:10}\n1234567890|1234567890"\
      .format("hydrogen",1.00784, "helium", 4.002602))
print(10*" * ")
print("{0:10}|{1:10.3f}\n{2:10}|{3:10.3f}".format("hydrogen",1.00784, "helium", 4.002
```

run | restart | restart & run all

```
Hello world!
```

? Exercise \(\PageIndex{2}\)

Adjust the code above so the print output of the atomic masses below the row of stars show a precision of 4 decimal places

Answer

```
print("{0:10}|{1:10.4f}\n{2:10}|{3:10.4f}".format("hydrogen",1.00784, "helium", 4.002602))
```

Aligning columns

Run and analyze the following code to figure out the syntax to left align, right align and center align data in a columns.

```
#left aligned spaced at 15 digits
print("{0:<15}|{0:<15}".format("left align", "center align", "right align"))
#left then right aligned spaced at 15 digits
print("{0:<15}|{2:>15}".format("left align", "center align", "right align"))
#center aligned spaced at 15 digits
print("{1:^15}|{1:^15}".format("left align", "center align", "right align"))
```

run | restart | restart & run all

```
Hello world!
```

This page titled 8.10: 3.format() method is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

8.11: f-string formatting

f-strings are the most recent and easiest way to format output, and so we will cover them first. You simply place an f in front of the quotes, and use curly brackets to insert variable names.

- Introduced in Python 3.6
- Place f in front of string
- allows you to skip the .format() step

Analyze the code, then run it, and then answer the questions below it.

Login with LibreOne to run this code cell interactively.

If you have already signed in, please refresh the page.

Login

```
molecule ="hydrogen"
molarmass=1.00794
mass = 10
moles =mass/molarmass
print(f'{mass} grams of the molecule {molecule} has {moles} moles.\n \t{mass} \
grams of the molecule {molecule} has {moles} moles. \n \
\t\t{mass} grams of the molecule {molecule} has {moles} moles.\n \
\t\t\t the molar mass of hydrogen is {molarmass}.\n \
This is the last line.')
```

```
Hello world!
```

Exercise 8.11.15.*a*

Answer the following questions concerning the above code

a. Which variable is a string?
b. Which variable is an integer?
c. Which variables are floats?
d. What does \ do?
e. What does \n do?
f. What does \t do?

Answer a

molecule

Answer b

mass

Answer c

molar mass and moles, which is an integer divided by a float

Answer d

Allows you to extend a long line of python script to the next line (this f-string is very long). The best practice is not to have a line of code longer than 80 characters.

Answer e

Next line (like the carriage return on a typewritter)

Answer f

tab

Note that we often want to limit the number of digits displayed to the right of the decimal to represent the precision with which a number is known. In the following example we will try and show how this is done for f-strings. we first print 20 decimal digits, so you can see what position a number is printed, and then we print the variable twice, using different format conventions



Login with LibreOne to run this code cell interactively.

If you have already signed in, please refresh the page.

Login

```
variable=12345.6789
print(f'12345678901234567890')
print(f'{variable:.4f}\n{variable:.2f}')

print(f'12345678901234567890')
print(f'{variable:10.2f}\n{variable:11.2f} \n{variable:11.3f}')
```

```
Hello world!
```

Exercise 8.11.1

consider the expression {variable_name:xx.yyf}

a. What does the colon indiate?
b. What does the xx indicate?
c. What does the yy indicate?
d. What does the f indicate?

Answer a

The colon indicates the following statement describes the formatting of the variable in the print statement

Answer b

This tells you the width of the number. This allows you to right justify all numbers.

Answer c

This tells you the precision of the number, that is, how many values to the right of the decimal are expressed in the print statement

Answer d

this is a floating decimal number

Exercise 8.11.15.*a*

Consider the following code and answer question (a) before running it



Login with LibreOne to run this code cell interactively.

If you have already signed in, please refresh the page.

Login

```
print(f"How long is this number going to be {1234.5678:2.3f}")
```

```
Hello world!
```

a. The :2.3f has a width of 2 and a precision of 3, so how long will the number be printed out?
b. Which variable is an integer?
c. Which variables are floats?
d. What does \ do?
e. What does \n do?
f. What does \t do?

Answer a

The entire number will be printed to the third decimal, so 8 characters including the dot.

Answer b

mass

Answer c

molar mass and moles, which is an integer divided by a float

Answer d

Allows you to extend a long line of python script to the next line (this f-string is very long). The best practice is not to have a line of code longer than 80 characters.

Answer e

Next line (like the carriage return on a typewritter)

Answer f

tab

This page titled 8.11: f-string formatting is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

8.12: Subscripts and Superscripts

One way to create subscripts and superscripts in strings is to use unicode and concatenate. The following table gives unicode subscript and superscript values that you may often needc

character	0	1	2	3
superscript	U2070	U00B9	U00B2	U00B3
subscript	U2080	U2081	U2082	U2083

```
#We will now use concatenation to create the formula for sulfuric acid.
sulfuric_acid="H"+"\u2082"+"SO"+'\u2084'
sulfate_ion = "SO"+"\u2084\u207B\u00B2"
print(sulfuric_acid, "and", sulfate_ion)
```

run | restart | restart & run all

```
Hello world!
```

Note, we can make variables for the unicode

```
s2,s3,s4,s5,s6,s7,s8,s9="\u2082","\u2083","\u2084","\u2085","\u2086",\
                        "\u2087","\u2088","\u2089"
ammonium_phosphate="(NH"+s4+")"+s3+"PO"+s4
print(ammonium_phosphate)
```

run | restart | restart & run all

```
Hello world!
```

? Exercise \(\PageIndex{1}\)

You want the following code to display

```
print(n_nonane)
```

run | restart | restart & run all

$CH_3(CH_2)(CH_2)(CH_2)(CH_2)(CH_2)(CH_2)CH_3$

Figure \(\PageIndex{1}\): Alter the code below to produce this output

The following code does not work. comment out the print statement and uncomment the lines above and below it, and then complete the expression for x so that the code works.

```
#the following line assigns subscripts to the varibles using unicode
s2,s3,s4,s5,s6,s7,s8,s9="\u2082","\u2083","\u2084","\u2085","\u2086",\
                        "\u2087","\u2088","\u2089"
```

```
#x=
n_nonane="CH"+s3+7*"(CH"+s2+")"+"CH"+s3
#n_nonane="CH"+s3+7*x+"CH"+s3
print(n_nonane)
```

run | restart | restart & run all

```
Hello world!
```

Answer

x = "(CH"+s2+")"

This page titled 8.12: Subscripts and Superscripts is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

CHAPTER OVERVIEW

9: Anatomy of a Pi and Camera Activity

Topic hierarchy

9.1: History of Pi
9.2: Monitoring Your Pi
9.3: Pi 4 Overview
9.4: SOC (System on Chip)
9.5: Random Access Memory (RAM)
9.6: Non-Volatile (Primary) Data Storage
9.7: LEDs
9.8: 40 Pin GPIO
9.9: UART (Universal Asnynchronous Receiver-Transmitter)
9.10: I2C (Inter-Integrated Circuit)
9.11: SPI (Serial Peripheral Interface)
9.12: Networking Overview
9.13: Radio Chip
9.14: Ethernet
9.15: USB C- Power Port
9.16: USB 2 and 3
9.17: AV Jack
9.18: Micro HDMI 2.0
9.19: CSI (Camera Serial Interface)
9.20: DSI (Display Serial Interface)
9.21: CSI Camera
9.22: Libcamera
9.23: 3-Python and Libcamera

9: Anatomy of a Pi and Camera Activity is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

9.1: History of Pi

The Raspberry Pi is a single board computer that was first released in 2012. The Raspberry Pi Foundation (https://www.raspberrypi.org/) was founded in 2009 and is a UK based Educational non-profit devoted to the development and promotion of Raspberry Pi single board computers, with an initial focus on their use to teach computing in schools and "the democratization of technology". In 2021 rasberrypi.com was released, which is run by Raspberry Pi Ltd, a subsidiary of the Raspberry Pi Foundation and responsible for the design, manufacture and distribution of Raspberry Pi hardware. The Raspberry Pi uses an Open Source ecosystem based on the Linux operating system but the hardware is not open source hardware (OSH), although documentation and the schematics of the hardware is readily available.

Pi Versions

- Pi 1 Model B (2012)
- Pi 1 Model A(2013)
- Pi 1 Model B+ (2014
- Pi 1 Model A+ (2014)
- Pi 2 Model B (2015)
- Pi 2 Zero (2015)
- Pi 3 Model B (2016)
- Pi Zero W (2017)
- Pi 3 Model B+ (2018)
- Pi 3 Model A+ (2019)
- Pi 4 Model B (2019)
- Pi 400 (2021)

We will only use Raspberry Pis in this course, but due to the chip shortage they are typically out of stock and you may be interested in other Single Board Computers that have similar form factors (features). Pallav Aggarwal gives a good 2022 review of Pi alternatives.

This page titled 9.1: History of Pi is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

9.2: Monitoring Your Pi

LED Warning Flash Codes

source: RasPi Documentation

Long flashes	Short flashes	Status
0	3	Generic failure to boot
0	4	start*.elf not found
0	7	Kernel image not found
0	8	SDRAM failure
0	9	Insufficient SDRAM
0	10	In HALT state
2	1	Partition not FAT
2	2	Failed to read from partition
2	3	Extended partition not FAT
2	4	File signature/hash mismatch - Pi 4
3	1	SPI EEPROM error - Pi 4
3	2	SPI EEPROM is write protected - Pi 4
3	3	I2C error - Pi 4
3	4	Secure-boot configuration is not valid
4	4	Unsupported board type
4	5	Fatal firmware error
4	6	Power failure type A
4	7	Power failure type B

Firmware Warning Icons

These show on the desktop

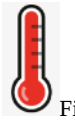
Undervoltage


Figure 9.2.1: Copy and Paste Caption here. (Copyright; author via source)

Over Temperature (80-85 degC)


Figure 9.2.1: Copy and Paste Caption here. (Copyright; author via source)

Over Temperature (>85 deg C


Figure 9.2.1: Copy and Paste Caption here. (Copyright; author via source)

This page titled 9.2: Monitoring Your Pi is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

9.3: Pi 4 Overview

This next section will go over the hardware components of the Raspberry Pi 4. We have created a webpage for each component and the goal is sort of two fold. First, to let you know what that component is, and second, to give you a bearing on how it works. That is, although we all use computers everyday, many of us have no idea how they work, or what the hardware does. Much of this is beyond the scope of the course, and so there will be many external links for you to dive in, as you like.



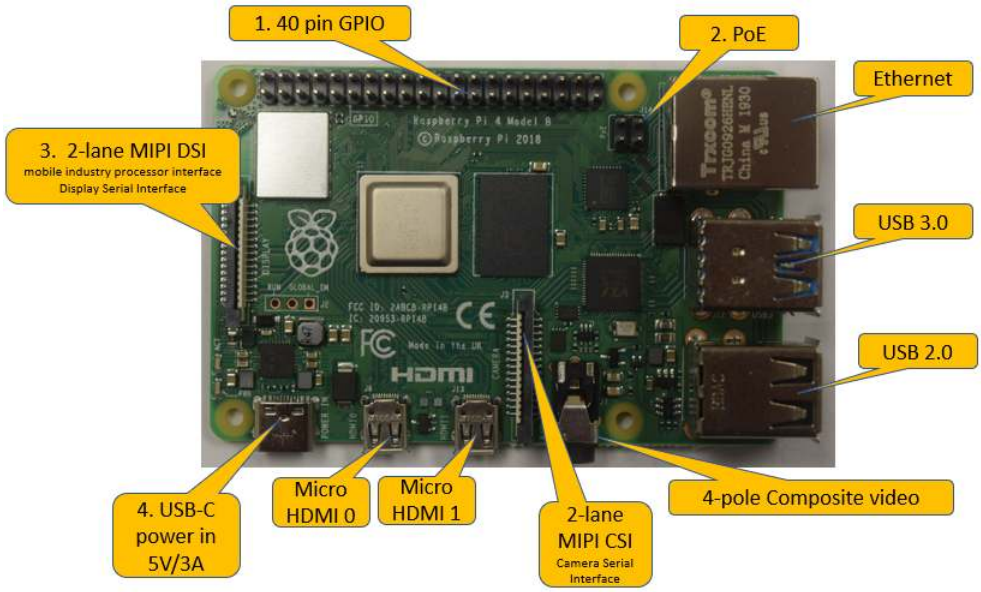


Figure 9.3.1: Various ports on a Pi 4. (Belford cc -0.0)

Pi Hardware

1. SOC (System on Chip)
    - CPU (Central Processing Unit)
    - GPU (Graphics Processing Unit)
    - Cache memory
2. RAM (Random Access Memory)
3. micro SD Card (Non-Volatile Memory)

4. LEDS
5. Networking options
    - Ethernet
    - Radio Chip
        - WiFi
        - BlueTooth
6. GPIO & Data Protocols
    - 40 Pin GPIO
    - UART (Universal Asnychronous Receiver-Transmitter)
    - I2C (Inter-Integrated Circuit)
    - SPI (Serial Peripheral Interface)
7. Ports
    - USB C (Power)
    - USB 2 & 3
    - AV Jack
    - micro HDMI 2.0
    - CSI (Camera Serial Interface)
    - DSI (Display Serial Interface)

This page titled 9.3: Pi 4 Overview is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

9.4: SOC (System on Chip)

The SoC is the CPU (Central Processing Unit) and the GPU (Graphical Processing Unit)

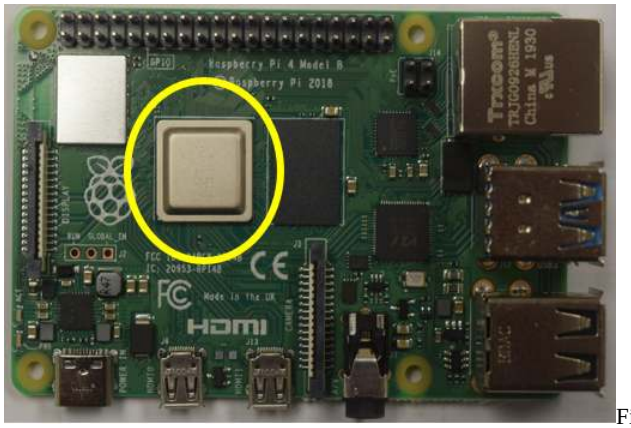


Figure 9.4.1: SoC or Raspberry Pi 4B. (Belford cc 0.0)

You should add a heat sink to your SoC. Pi 4 have Dynamic Voltage and Frequency Scaling (DVFS) which allows SoC components to run at lower voltages and reduces clock speed when not needed. The memory is split between the GPU and CPU and you can identify how much each is allocated with the following command.

```
vcgencmd get_mem arm && vcgencmd get_mem gpu
```

The Top command gives general information on system performance (CPU and memory)

```
top
```

CPU

The CPU (Central Processing Unit) is the "brain" of the computer. It "fetches", "decodes" and "executes" code with each step being performed at a rate determined by the clock. A multicore architecture allows parallel processes to occur. If you want to do a dive into how computers work you should check out the PBS Crash Course on Computer Science with Video 8 being on the CPU

Different versions of the Pi use different CPUs, which can be found on the Raspberry Pi Processors page of the Pi Documentation. the Pi 3 uses a BCM2837B0 and information can be found here.

- 64 bit Quad Core Broadcom BCM2711 (Pi 4)
    - 1.8 GHz (clock speed)
    - BCM2711 ARM Peripherals Documentation
        - section 5.3 has alternative GPIO pin assignments
- SRM Cortex-A72 Cores
    - Architecture handles "out of order" execution
    - prioritizes performance over efficiency.
        - one of the reasons the Pi 4 gets hotter than a pi 3
- Caches
    - 32 KB data and 48 KB Instruction L1 per core, 1 MB L2 shared cache

Some CPU commands

To get Information on your CPU

```
cat /proc/cpuinfo
```

To get CPU temperature

```
vcgencmd measure_temp
```

To find the clock speed your Pi configuration is limited to

```
vcgencmd get_config arm_freq
```

to find current clock speed

```
cat /sys/devices/system//cpu/cpu0/cpufreq/scaling_cur_freq
```

GPU

- 32 bit Broadcom VideoCore 6
    - 500 mHz
        - supports Open GL ES 3
    - Runs 4K video decoder at 60 frames/sec

This page titled 9.4: SOC (System on Chip) is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

9.5: Random Access Memory (RAM)

Random Access Memory (RAM) is volatile quick access memory. You could say there are three types of memory; cache, primary (RAM) and secondary (hard drive). An analogy could be made to studying. The cache are the books on your desk that you are currently using. The RAM is the books on your bookshelf that you have instant access to. The hard drive are like the books in the library, that you have to go and get before you can use. The cache and RAM memory are volatile and you do not preserve the data in them when you shut down the computer (each semester you use a new set of textbooks), while the hard drive is non-volatile and preserves data when you shut down the computer.

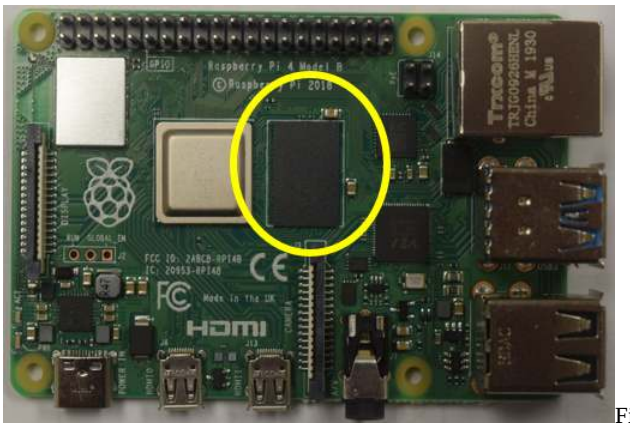


Figure 9.5.1: Random Access Memory chip on a Pi 4 B. (Belford cc 0.0)

- 1,2,4,8 GB options
    - LPDDR4 SDRAM
        - Low-Power Double Data Rate Synchronous Dynamic Random Access Memory
        - Transfers data on rise and fall of clock signal
    - 2400 MHz

Memory Functions

Information on Memory

```
cat /proc/meminfo
```

Free memory

```
free -h
```

This page titled 9.5: Random Access Memory (RAM) is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

9.6: Non-Volatile (Primary) Data Storage

In the Raspberry Pi your "hard drive" is a micro SD card. This card also contains your operating system, which you must load when you boot up the pi. By loading different programs onto different SD cards you can actually change the functionality of a Pi by simply swapping micro SD cards. When you install the micro SD card the contacts should face the board. Instructions for installing the operating system will evolve over time and we will try and keep the Installing RPi OS page updated.



Figure 9.6.1: micro SD card port. (Belford cc 0.0)

Raspberry Pi 4 Specs

- micro SD (Secure Digital) card
    - Hard drive with image of operating system
    - 50 MBps top transfer rate
        - Twice speed of Pi 3

Micro SD Card Specs

Tom's Hardware, Android Central and PCguide have provided reviews of 2023 micro SD cards, and the size really depends on what you want to do, but 32 gigabytes will be plenty for most Applications



SD Card Labels

Speed Label

The problem with the "Speed Label" is this is for sequential reading (looking at an image or video", while when used as a general purpose computer you tend to pull lots of small files, with many read/write statements.

Minimum Sequential Write Speed	Speed Class	UHS Speed Class	Video Speed Class
90MB/sec			V90
60MB/sec			V60
30MB/sec		U with 3 in it	V30
10MB/sec	C with 10 in it	U with 1 in it	V10
6MB/sec	C with 6 in it		V6
4MB/sec	C with 4 in it		
2.5MB/sec	C with 2.5 in it		

Performance Application Class

This gives the minimum number of Input/Output operations per second (IOPS). When using the micro SD card as a hard drive you want this to be as big as possible

Class (App Performance	Minimum Random Read	Minimum Random Write
A1	1500 IOPS	500 IOPS
A2	4000 IOPS	2000 IOPS

There is evidence that the A1 cards are actually slower on the Raspberry Pi, and the verdict is still out (See Jeff Geerling's blog posts July 22, 2019 and July 22, 2019).

Bus Interface Label

These deals with how fast data can be transferred in and out of the card and uses a ROMAN Numeral. UHS means Ultra High Speed

Speed Designation	High Speed	UHS-I	UHS-II	UHS-III
Label	no label	I	II	III
Speed	25 MB/s	104 MB/s	312 MB/s	624 MB/s

Note: High speed and UHS-I micro SD cards have a single row of connectors, while UHS-II and UHS-III have two rows of connectors. Since the fastest transfer rate of a Pi is 50 MB/s there is no reason to get a UHS-II micro SD card.

Formatting SD Cards

FAT stands for File Allocation Table and there are 3 FAT standards that SD cards use

Standard	FAT16	FAT32	exFAT
Capacity	128 MB to 2 GB	4GB to 32GB	64GB to 2 TB

This section needs to be developed as there is a lot of conflicting information

Raspberry Pi Speed test

Instructions at https://www.raspberrypi.com/news/sd-card-speed-test/

in Command Line

```
sudo apt update
sudo apt install agnostics
```

Once installed you will see new menu item under Accessories in main menu

Internet in a Box

It is actually rather amazing what you can put on a microSD card. Internet-in-a-box (iiab) is an organization devoted to making online content available in regions without internet access by capturing images of the web, and these images can be placed into a microSD card, and the Raspberry Pi is then used as an internet hotspot. This is really a fascinating use of the Raspberry Pi. I have an iiab with all of wikipedia, a map of the world with building level precision (think Google Maps), almost 10,000 Khan Academy videos and a ton of other stuff. It is absolutely amazing what you can put onto a micro SD card. The iiabFAQ is a good place to start and links can be found on their download site. Note, on an iiab you are typically only reading, that is, you are using it to read an image off the "web", and so the micro SD card usage is different than a normal Pi, where you are reading and writing all the time.



This page titled 9.6: Non-Volatile (Primary) Data Storage is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

9.7: LEDs

There are two LED status lights on the Raspberry Pi.

- Red LED - Power On
    - Flicker indiates power problem
- Green LED - SD Card Activity
    - Flashes during read/write operations
    - Patterns vary by Pi Model

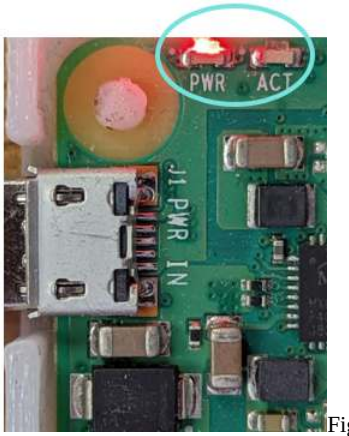


Figure 9.7.1: LED Status Lights on a RasPi 4. (Belford cc 0.0)

LED Warning Flash Codes

source: RasPi Documentation

Long flashes	Short flashes	Status
0	3	Generic failure to boot
0	4	start*.elf not found
0	7	Kernel image not found
0	8	SDRAM failure
0	9	Insufficient SDRAM
0	10	In HALT state
2	1	Partition not FAT
2	2	Failed to read from partition
2	3	Extended partition not FAT
2	4	File signature/hash mismatch - Pi 4
3	1	SPI EEPROM error - Pi 4
3	2	SPI EEPROM is write protected - Pi 4
3	3	I2C error - Pi 4
3	4	Secure-boot configuration is not valid

Long flashes	Short flashes	Status
4	4	Unsupported board type
4	5	Fatal firmware error
4	6	Power failure type A
4	7	Power failure type B

This page titled 9.7: LEDs is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

9.8: 40 Pin GPIO

A GPIO pin has two values, high (3.3V) or low (0V). Any GPIO pin can be designated by software as input or output.

There are two Pin Numbering Systems the Broadcom (BCM) and the physical BOARD numbering. The BCM is based on the broadcom chip and used in most libraries. There are also two common libraries that are used with Raspberry Pis to control the GPIO pins, the RPi.GPIO and the GPIO zero library.

To find out what your pin assignments are you can go to the command line and type

```
pinout
```

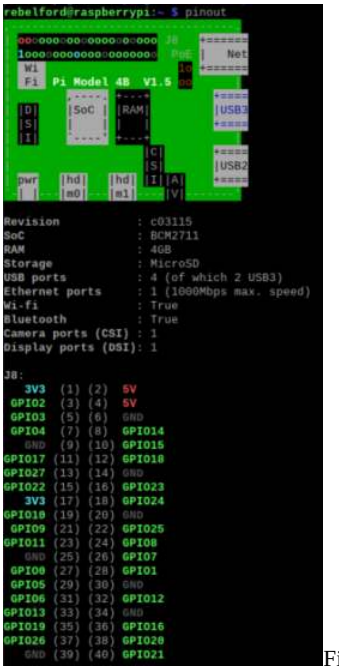


Figure 9.8.1: Output of pinout command in the command line of a Raspberry Pi (Belford CC-0.0)

This will give you the following output, and you can find more information at https://pinout.xyz/

https://www.raspberrypi.com/documentation/computers/raspberry-pi.html#raspberry-pi-4-model-b

Resources

- Raspberry Pi Documentation
- GPIO Zero Python Documentation - gpiozero.readthedocs.io

This page titled 9.8: 40 Pin GPIO is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

9.9: UART (Universal Asnynchronous Receiver-Transmitter)

Universal Asynchronous Receiver-Transmitter (UART) is one of the oldest forms of communication. These are the old serial ports of old computers and has been replaced by USB. The data is sent serial as a "data frame". It is "universal" because it can operate over multiple parameters like speed and data size. It is asynchronous because it does not need a clock to synchronize the transmitting and receiving of the data, although they need to be at common BAUD rates.

Features

- 2 lines and 3 pins
  - TX (Transmit line and pin)
  - RX (Receive line and pin)
  - GND (Ground)
- Asynchronous, bidirectional serial data transmission
  - no clock, it uses a "start bit" and an "end bit"
- Limited to one sensor
- BAUD rate (bps or bits per second) 1200, 2400, 4800, 9600, 19200, 38400, 57600 and 115200
  - 115200 is default BAUD rate
  - both devices must operate at the same BAUD rate
- slow, but easy to implement

UART Fundamentals

Data flows from TX pin of Transmitter to RX pin of receiver. The transmitter can convert parallel data from CPU to serial for communication and then the receiver can convert it back to parallel data

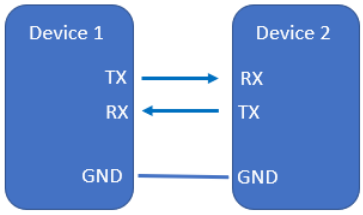


Figure 9.9.1:UART pins and lines for device to device communication (Belford CC0.0)

3 modes of operation

- Simplex (data only goes in one direction)
- Half-Duplex (data transmission in both directions but not simultaneously)
- Full Duplex (simultaneous data transmission in both directions)

UART Data Frame



Figure 9.9.2: UART data frame. (Belford CC0.0)

The data frame has bits that are either high or low. The start bit tells the receiver when a frame is coming and when it is ending. The parity bit is optional and is used to ensure the data is properly transmitted. Since each device has an TX and RX pin, the data can operate in Full Duplex mode where one wire is sending it in one direction and the other in the opposite direction.

Protocol

- Transmitter receives data to be transferred, often from a parallel bus (as CPUs use those because they are faster)
- add start bit, then parity and stop bit
  - parity data checks for data integrity

- Data packet is transmitted until complete
    - Uses a FIFO buffer (First In, First Out) to serialize the stream
- Data in receiving FIFO is cleared as it is transferred

Resources

- seeedstudio blog
- ElectronicWings
- TotalPhase
- electric impDevCenter

This page titled 9.9: UART (Universal Asnynchronous Receiver-Transmitter) is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

9.10: I2C (Inter-Integrated Circuit)

Inter-Integrated circuit

I2C _"Eye Squared C" is a synchronous serial bus that can hold up to 128 devices on a Raspberry Pi. It is used with many common sensors like the BME680 VOC sensor. This is a synchronized protocol using a clock signal that is controlled by the master to align the master and slave. You can have infinite "Masters" and up to 1008 "slaves", although the Raspberry Pi only uses 128 "slaves". Standard mode is 100 kbps, high speed is 3.4 Mbps. A typical I2C device has four pins

- SDA (Serial Data Pin)
- SCL (Serial Clock Pin)
- VIN (Voltage in, 3.3 V on Pi GPIO)
- GND (Ground)

So the I2C uses one line for data transmission and one for the clocksignal, with the other lines running power to a device, and serving as ground.

Circuit Protocols

I2C data is transferred in frames.



Figure 9.10.1: Copy and Paste Caption here. (Copyright; author via source)

- Start Condition:
    - Data line switches voltage (high to low) which starts clock
- Address:
    - 7 to 10 bit address that is unique to each sensor. This identifies sensor the controller is communicating with.
    - Always follows Start
- Read/Write
    - Always follows address
    - Identifies if controller is sending data to sensor or requesting it
    - High voltage means data is requested from sensor
    - Low voltage means data is sent to sensor
- ACK/NACK
    - If address or data frame is successfully received it is acknowledged
    - Always follows an address or data frame
    - ACK bit must be received before next data frame is sent
- Data frame
    - 8 bit data frame

This material is adpated from Circuit Basics and you should read their writeup for a more extensive description of I2C protocols

Enabling I2C

Command Line

In the command line type

```
sudo raspi-config
```

Choose:

1. Interfacing Options
2. I2C
3. Yes

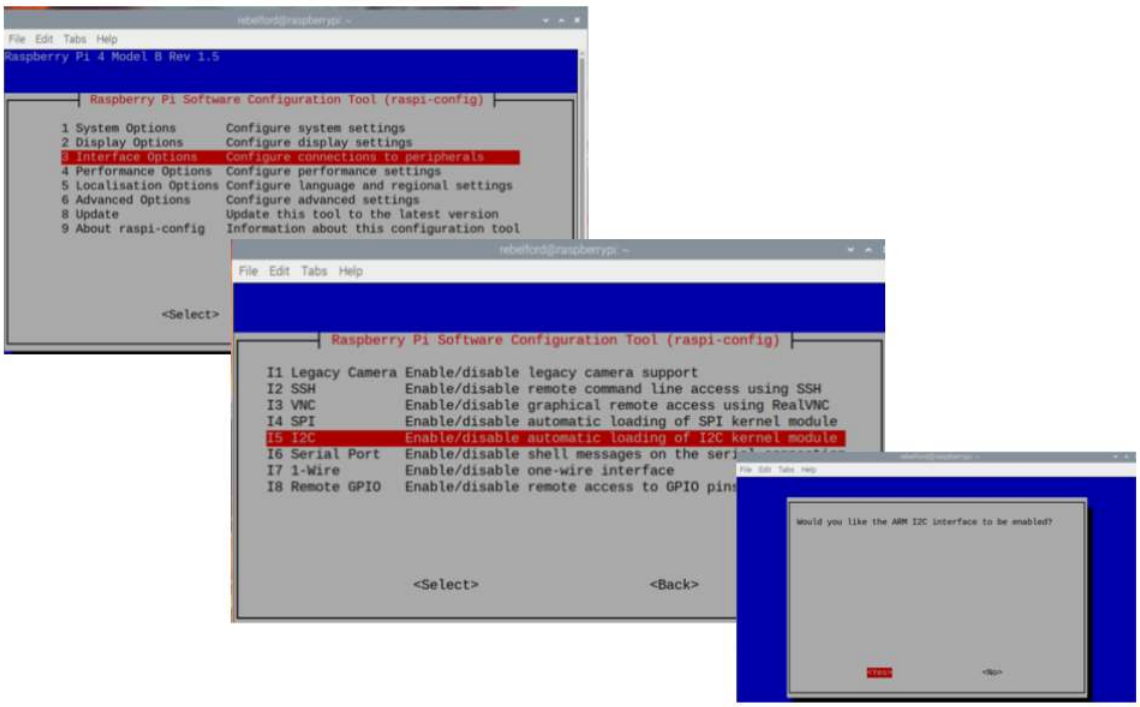


Figure 9.10.1: Copy and Paste Caption here. (Copyright; author via source)

Raspberry Pi Configuration

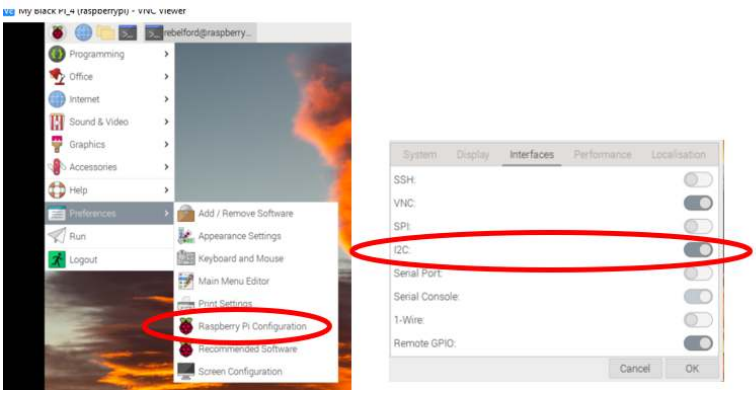


Figure 9.10.1: Copy and Paste Caption here. (Copyright; author via source)

Utility Commands

```
i2cdetect -y 1
```

the -y flag disable interactive mode so you do don't have to wait for confirmation. The 1 indicates you are scanning i2c bus #1.

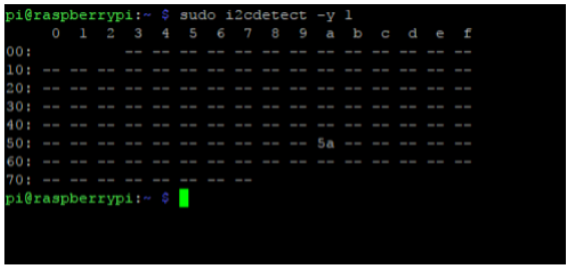


Figure 9.10.1: The addresses are base 16 (hexadecimal), and so 5a is registry 90 in decimal. (Belford, cc0.0)

Resources

- Circuit Basics
  - Good Article on Protocols
- Archive of I2C articles on Raspberry Pi
- SparkFun: Raspberry Pi SPI and I2C Tutorial
- SparkFun: I2C

This page titled 9.10: I2C (Inter-Integrated Circuit) is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

9.11: SPI (Serial Peripheral Interface)

Serial Per

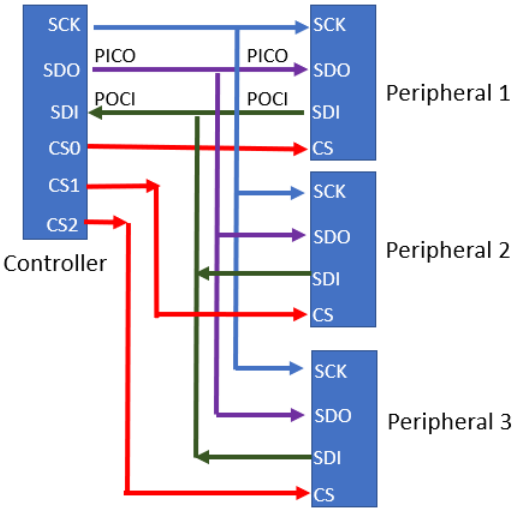


Figure 9.11.1: Data lines for SPI interface with three peripheral devices. Note you still need to provide power and a ground to the perpherials. (Belford CC-BY)

Note

The Open Source Hardware Association has put forth a resolution to redefine the SPI Signal Names. In many devices and data sheets you will see the older names, and you will need to be familiar with both. In essence the old terminology used Master/Slave designation and the new one uses Controller/Peripheral (device) designations.

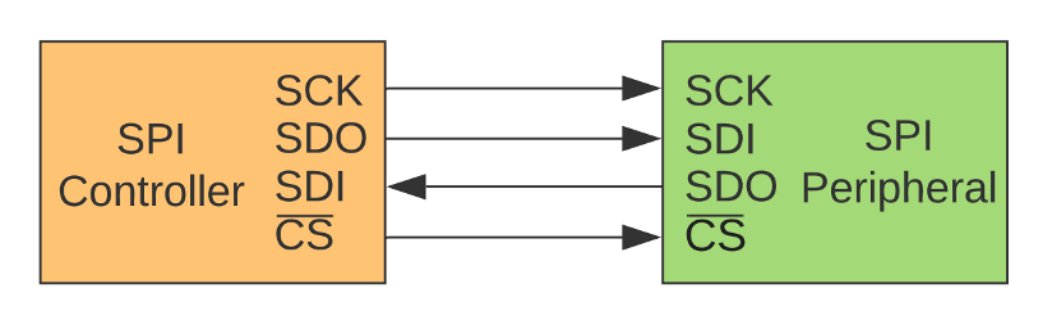


Figure 9.11.2: New signal designations. (SparkFun CC 0.0)

New Signal Names

- SDO - Serial Data Out. An output signal sent to another SPI device
- SDI - Serial Data In. An input signal received from another SPI device.
- CS - Chip Select. Picks which peripheral device is active (old SS).
- PICO - Peripheral In Controller Out. Signal being sent from controller to peripheral (old MOSI).

- POCI - Peripheral Out Controller In. Signal sent from peripheral to controller (old MISO).
- SDIO - Serial Data In/Out. Bidirectional serial signal.

Deprecated Signal Names

- MOSI - Master Out Slave In
- MISO - Master In Slave Out
- SS - Slave Select
- MOMI - Master Out Master In
- SOSI - Slave Out Slave In

Unchanged Signal Names

- SCK - Serial Clock. Controller generated clock signal for the common bus.

solving problems, as long as you get the **proper values a**





- Circuit Basics
- SparkFun: Serial Peripheral Interfaces

Resources

Raspberry Pi Documentation

This page titled 9.11: SPI (Serial Peripheral Interface) is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

9.12: Networking Overview

The Raspberry Pi 4 has several options for communication

- Radio Chip
    - Dual band 2.4/5 Ghz IEEE 802.11 ac WiFi
    - Bluetooth
- Ethernet
    - RJ45 Ethernet port
        - 900+ Mbit bandwidth

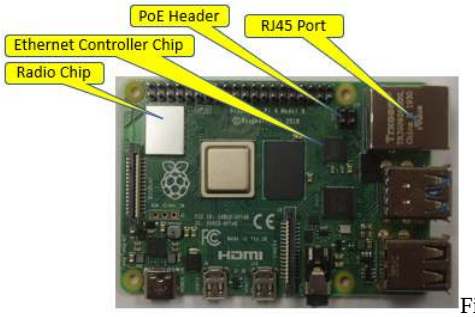


Figure 9.12.1: Communication hardware on Raspberry Pi 4. (Belford cc 0.0)

This page titled 9.12: Networking Overview is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

9.13: Radio Chip

The radio chip contains the Bluetooth and WiFi hardware and has a metal shield to reduce EMF interference

WiFi

- The WiFi is disabled until you set up a host country, which should have been done during your initial setup, but can be changed any time by
  - pi icon/Preferences/Raspberry Pi Configuration/Localisation/Set WiFi Country

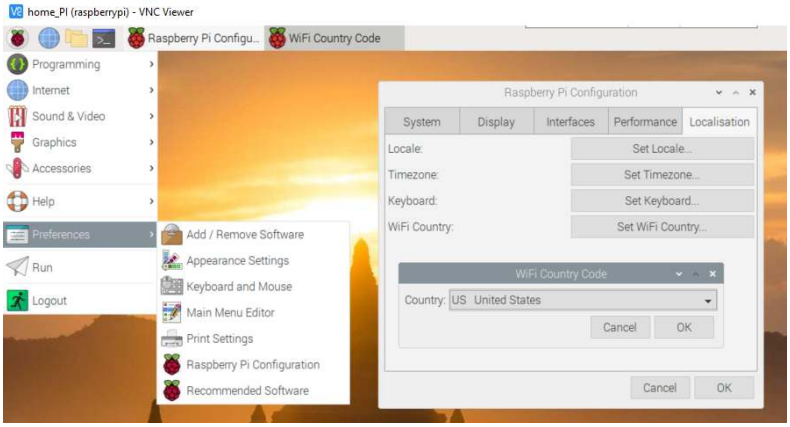


Figure 9.13.1: Setting WiFi country. (Belford cc 0.0)

- Configuring WiFi Network
  - The easiest way to configure the WiFi Network is through the desktop, by clicking the WiFi icon in the taskbar (upper right), choosing network and providing passwords. This can also be done through the command line and instructions are provided with the Raspberry Pi Configuring Networking Documentation.
  - Seeedstudio blog provides step by step instrutions if needed.

WiFi Hotspot

Your Raspberry Pi can also be set up as a Hotspot. This can be useful if you do not have a WiFi router or are using your Pi as an Internet-in-a-box.

- Toms Hardware Tutorial
- ThePi.io Tutorial

Bluetooth

The Raspberry Pi 4 comes with Bluetooth 5.0. The following tutorials provide information on Bluetooth and we will not be using it in this class.

- Using BLE (Bluetooth Low Energy) Devices with Raspberry Pi (Argenox.com)
- Raspberry Pi WIFi and Bluetooth Setup (Digi-Key)
- How to set up Bluetooth on a Raspberry Pi (Howchoo.com)

This page titled 9.13: Radio Chip is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

9.14: Ethernet

There are a lot of advantages to ethernet over WiFi and the ethernet setup should be plug and go, there is a tutorial at AdaFruit if you have issues.

Using Ethernet to connect Pi to Laptop

You can use the ethernet to directly connect a Pi to a laptop



POE Hat

Power Over Ethernet (POE) pins can be used to power a HAT or other device



Figure 9.14.1: POE Hat. (raspberrypi.com, cc 4.0)

This page titled 9.14: Ethernet is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

9.15: USB C- Power Port

USB Power

- 5V 3 amp
- Early issues with "e-marked" cables
  - Hackaday article (2019)
  - Problem fixed (2020 Tom's Hardware)

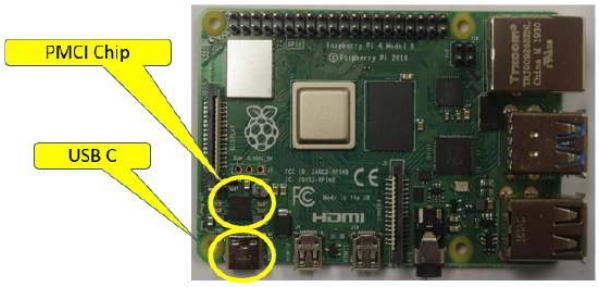


Figure 9.15.1: USB C power port and Power Management Chip. (Belford cc 0.0)

This page titled 9.15: USB C- Power Port is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

9.16: USB 2 and 3

Under Construction

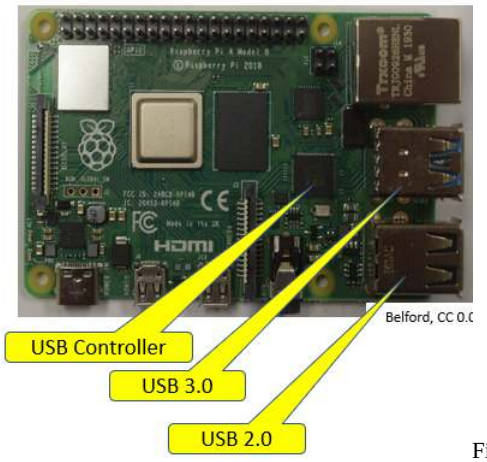


Figure 9.16.1: Copy and Paste Caption here. (Copyright; author via source)

- VL805 USB Controller
    - 2 USB 3.0 ports (340 Mbs)
    - 2 USB 2.0 ports
- Connects to SoC via PCLe Interface
    - (Peripheral Component Interconnect Express)
    - Very Fast (up to 5 Gbps)

This page titled 9.16: USB 2 and 3 is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

9.17: AV Jack

Default output is HDMI



Figure 9.17.1: Copy and Paste Caption here. (Copyright; author via source)

- 3.5 mm TRRS
  - Tip, Ring, Ring, Sleeve
  - 4 pole audio and video
- Headphone Jack





Figure 9.17.2:RCA to 3.5 mm video jack (Wikimedia Commons, cc 4.0)

Playing Audio and Video

Buster and realier use OMXPlayer

Buster uses VLC

This page titled 9.17: AV Jack is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

9.18: Micro HDMI 2.0

Under Construction

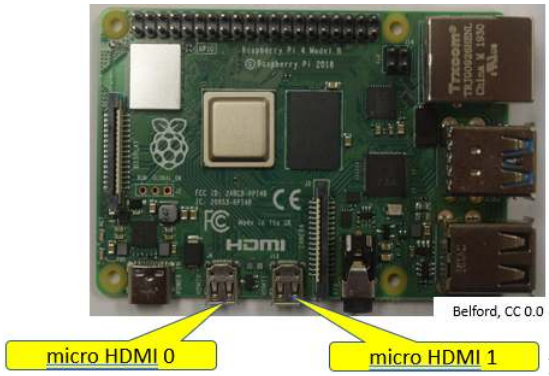


Figure 9.18.1: Copy and Paste Caption here. (Copyright; author via source)

- HDMI (High Definition Multimedia Interface)
    - Drive two 4K displays at 30 fps
    - Drive one 4K display at 60 fps
- 19 pins like full size
- Multiple Adapters Available
- HDMI-0 is near USB-C Conncector
    - Primary (default) port
        - use if only 1 monitor
- HDMI-1 is near CSI port

This page titled 9.18: Micro HDMI 2.0 is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

9.19: CSI (Camera Serial Interface)

Under Construction

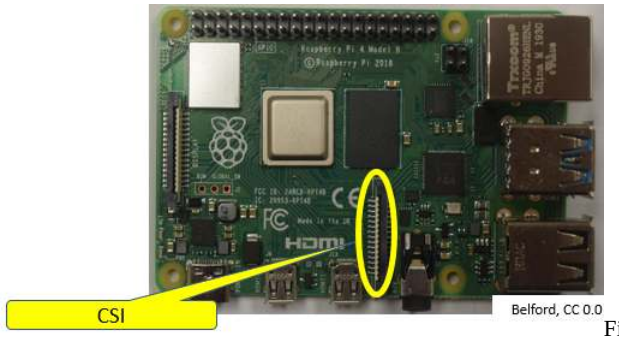


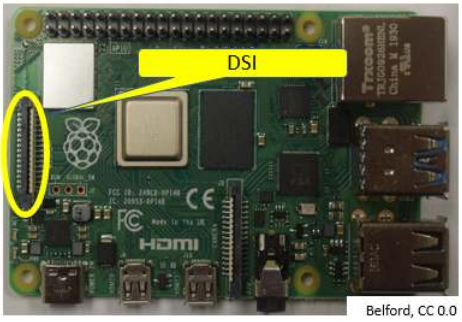
Figure 9.19.1: Copy and Paste Caption here. (Copyright; author via source)

- MIPI-Mobile Industry Processor Interface
    - mobile phone camera
- CSI - Camera Serial Interface
- Provides Bus to SoC
- Fast data transfer
- uses 15 pin ribbon
- Directly interfaces with the GPU
    - Uses fewer resources than USB camera
    - If you use a USB camera, use one with hardware encoding and USB 3.0

This page titled 9.19: CSI (Camera Serial Interface) is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

9.20: DSI (Display Serial Interface)

Under construction



Belford, CC 0.0 Figure 9.20.1: Copy and Paste Caption here. (Copyright; author via source)

- MIPI - Mobile Industry Processor Interface
  - mobile phone displays
- DSI-Display Serial Interface
- Connects Directly to the GPU
  - Reduces CPU load
- Like cell phone display
  - Can be touch screen..

This page titled 9.20: DSI (Display Serial Interface) is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

9.21: CSI Camera

Setting up Camera

You will now set up the Ribbon camera as outlined in Figure 9.21.4, but first, you should watch the following two videos. The camera is very static sensitive, so you want to be careful. It comes in a protective ESD bag, and you should store it in a ESD bag when not in use. It is advisable that you touch a grounded metal surface before working with the camera. You will attach the camera to the pi through the MIPI SCI (Camera Serial Interface) with the 15 pin ribbon, such that the blue side faces towards the USB 2.0 ports (Pi 4), or the Ethernet (Pi 3). The connector has a plastic clip that needs to be raised slightly, you then slip in the ribbon and then lower the clip back into place. I personally like to use tweezers and suggest you get a set of precision tweezers like these, as you will find them very useful when you are building circuits.

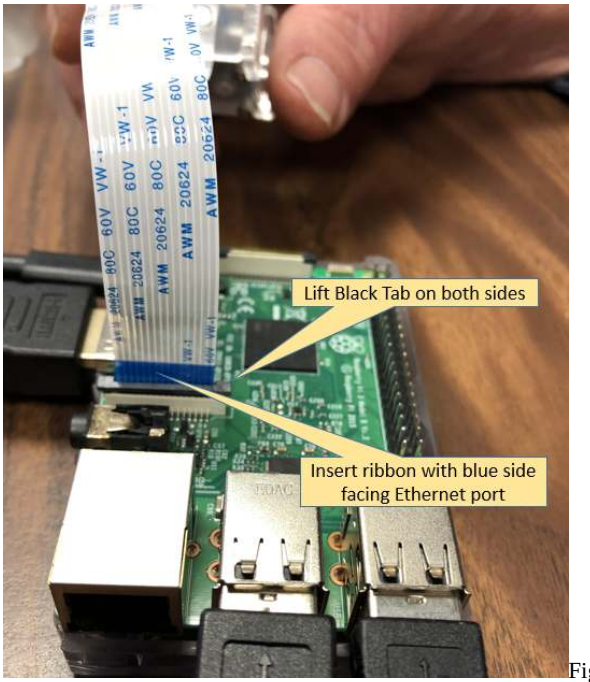


Figure 9.21.4: You lift up the black tabs around the camera port and then insert the ribbon with the blue side facing the ethernet port. Note, this picture is of a Pi 3, and in the Pi 4 the USB 2.0 ports are located where the ethernet port on the Pi 3 is

With the Raspberry Bullseye operating system the old raspicam program was deprecated and the new libcamera is installed. Many programs still use the old system and you can switch between them with the raspi-config command.

Switching camera interfaces

To switch camera interfaces between raspicam and libcamera you go to the Pi Configuration Tool (raspi-config)

- Raspicam using the legacy camera support
- libcamera does not

```
sudo raspi-config
```

Use the down arrow to navigate to option 3 (left image of fig. 9.21.5)

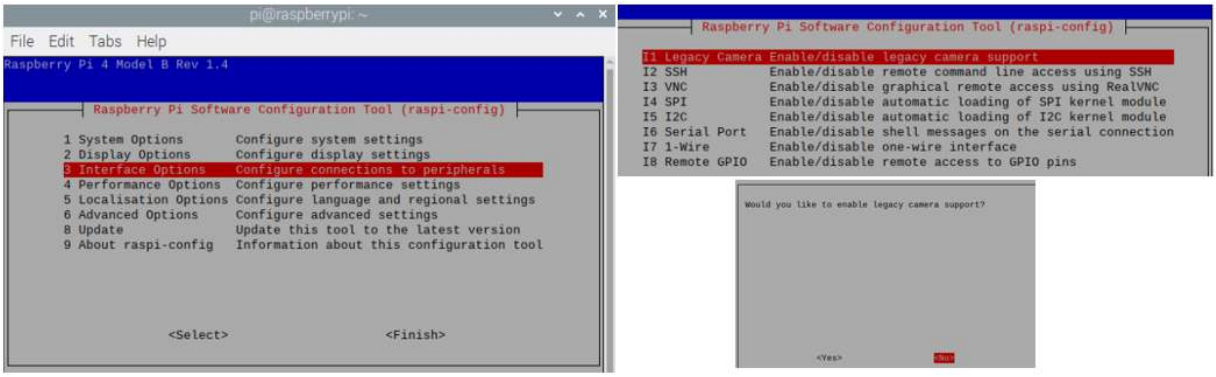


Figure 9.21.5: Raspberry Pi Configuration tool (2022, Belford CC0.0)

The first option is to support the legacy camera support (raspicam), and the default should be for it not to be supported (right two images fig. 9.21.5). If you wish to use Raspicam you need to choose yes. If you wish to use Libcamera, just leave it at No. I suggest you work with Libcamera, but if you have some third party software the issue may be that it is set up for raspicam commands, and so you may need to switch the interface.

If you choose to use Raspicam you can go to the 2020 class section 1.3.7.

Right now we are going to interact with the camera through the command line. Later on we will do this through python scripts.

This page titled 9.21: CSI Camera is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

9.22: Libcamera

Command Line Acces to Camera

With the advent of the Bookworm Pi OS the libcamera application was renamed to rpicam. Right now libcamera commands will still work, but the will be deprecated and so it is best to code forward with rpicam commands. We will predominantly work with two image encoders, jpeg for still images and h.264 for video.

The following material relates to the Bullseye operating system and uses libcamera. If you are using Buster or wish to use the Raspicam legacy interface, go to section 1.3.7 of the 2020 class. The following video and this article from Core Electronics , How to use of Raspberry Pi Cameras with Bulleseye OS Update are a good place to start.



Video

For a complete overview of the librecamera go to Raspberry Pi Camera Software Documentation

Libcamera

Preview

libcamera-hello - gives a 5 second view of what the camera sees

```
libcamera-hello
```

To flip vertically or horizontally use the following switches

```
--vflip
--hflip
```

to find options use the help switch

```
libcamera-jpeg -h
```

To run the camera for a time in milliseconds, use -t followed by the time in milliseconds

```
libcamera-hello -t 8000
libcamera-hello --vflip -t 8000
```

To run the camera indefinitely, until you terminate the command (<ctrl>c) or close the window, set the time to zero.

```
libcamera-hello -t 0
```

Still Picture

When you take a picture you need to define a path for where the picture will be stored. If you do not do that, it will be stored in whatever directory the command line is operating in. You should think about your project and where you want to keep images, and keep them in a folder that is within your user directory (/home/username/....).

The following code will give a 5 second preview and then save a file. NOTE: you need to set the path to a folder on your pi. the -o stands for output (file)

```
libcamera-jpeg -o /home/user/path/test1.jpeg
libcamera-jpeg --vflip -o /home/user/path/test2.jpeg --vflip
```

To change the delay time use -t with time in milliseconds

```
libcamera-jpeg --vflip -o /home/user/path/test3.jpeg -t 100
libcamera-jpeg --vflip -o /home/user/path/test4.jpeg -t 100 --width 640 --height 480
```

Libcamera-still allows you to save in different formats through the equivalency switch (-e)

```
libcamera-still -e png -o /home/user/path//FirstPNGphoto.png
```

Video

Here you are making a 5 second video and storing it in the Videos directory. the .h264 makes an mp4 video.

```
 libcamera-vid -t 10000 -o /home/  /path/video.h264 --vflip
```

to find help options try the -h switch

```
libcamera-vid -h
```

time lapse

```
 libcamera-still --vflip -t 60000 -o test%d.jpg --timelapse 10000
```

```
ffmpeg -framerate 1 -i test%d.jpg -c:v libx264 -r 30 output.mp4
```

Note, you can not view the video over VNC viewer as it is too slow

MotionEye

Using Python 3

https://github.com/motioneye-project/motioneye

Resources

- Raspberry Pi Documentation
-

This page titled 9.22: Libcamera is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

9.23: 3-Python and Libcamera

The easiest way to take pictures from Python is to use the libcamera picamera2library that is a local python library (it is installed with Python but you need to import it in order to use it). You should consult the Picamera2 Library manual for further information.

```
#Script is adapted from page 6 of Picamera2 Library Manual
from picamera2 import Picamera2
picam2 = Picamera2()
file_name=input("enter a file name for your picture: ")
picam2.start_and_capture_file("/home/rebelford/Pictures/"+file_name+".jpg")
```

You will probably want to install FFmpeg. there are two ways you can do this.

1. In Thonny install through Tools/Manage Packages and search for it if it is not there.
2. In command line type

```
sudo apt install -y ffmpeg
```

The following script will take 10 pictures, one every one second.

```
from picamera2 import Picamera2
picam2 = Picamera2()
file_name=input("enter a file name for your picture: ")
picam2.start_and_capture_file("/home/rebelford/Pictures/"+file_name+"{:d}.jpg", initi
```

This page titled 9.23: 3-Python and Libcamera is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

CHAPTER OVERVIEW

10: Logic and Conditional Statements

Topic hierarchy

10.1: Introduction
10.2: if Statement
10.3: if-else Statement
10.4: if-elif-else Statement
10.5: Compound if/elif Statements
10.6: Nested if Statements
10.7: Loops - An Introduction
10.8: for Loops
10.9: While Loops
10.10: Nested Loops
10.11: Loop Control Structures

10: Logic and Conditional Statements is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

10.1: Introduction

Control structures allow program to use logic to execute code. In python the syntax uses blocks of code defined by colons and indentation. There are two basic types, conditionals and loops. The logic is typically defined by an operator. These structures can be nested

Conditionals

Execute one or more statements if a condition is met.

- If statement
- If-Else statement
- If-Elif-Else statement

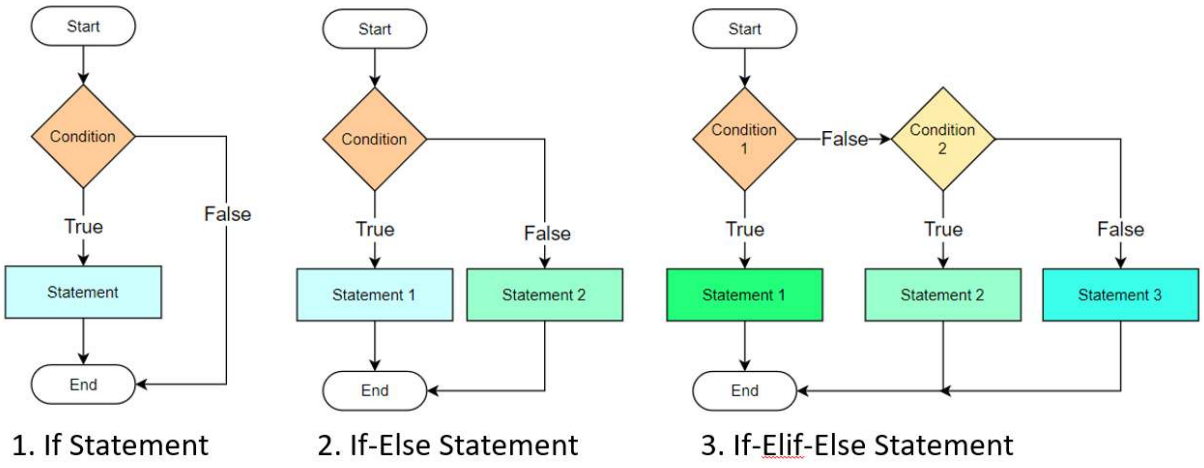


Figure 10.1.1: Conditional statement logic diagrams. (Belford cc 0.0)

Loops

Iterate through a statement if a condition is meet

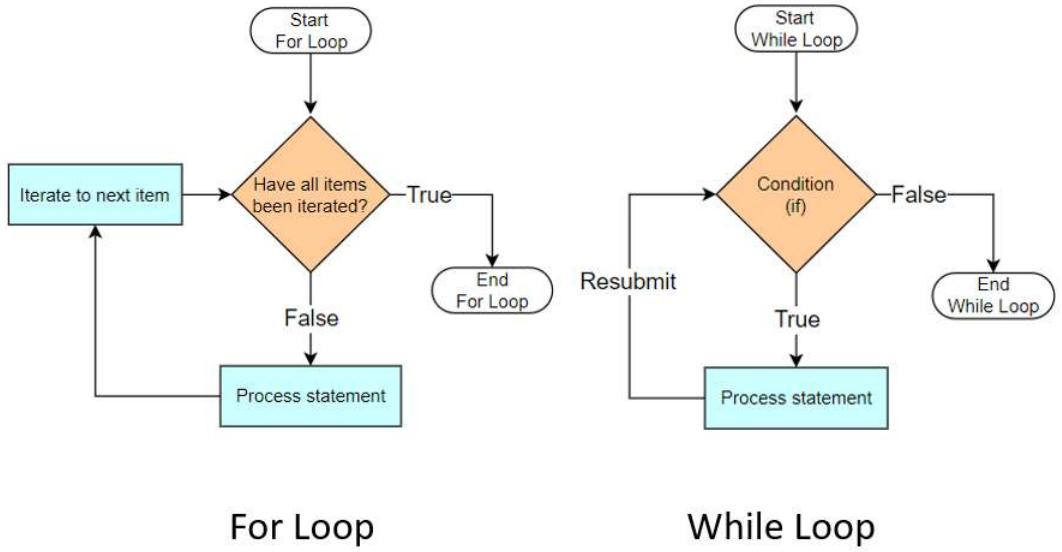


Figure 10.1.2: For and While loop flow diagrams. (Belford cc 0.0)

Boolean vs. Arithmetic Algebra

We are all familiar with arithmetic algebra and for many students this will be their first foray into Boolean Algebra. In arithmetic algebra you can perform operations on variables, and then apply numerical values to those variables and output numerical answers. In Boolean Algebra the output is a binary "truth value" (True or False) that may be based on a numerical relationship, but may also be based on non-numerical relationships such as identity and membership (look at the operators below). It needs to be realized that the output is the "truth value", and the flow charts above can result in binary decision trees where the output is based on the truth values of the individual Boolean operations (nodes). The success of coding often depends on parsing your problem in a manner that is amenable to Boolean algebraic operations in a manner that results in the behavior of the system you want to create.

So in summary, arithmetic algebra can apply data to sequential arithmetic operations and come up with a numerical answer, while Boolean algebra can apply data to sequential Boolean operations that results in a binary decision tree based on the truth value of each operation and be used in decision making.

Operators

There are a variety of operators that are typically used in logic and control structures.

Comparison Operators

See section 2.15: Comparison Operators for examples

- <
- <=
- >
- >=
- ==
- !=

Logical Operators

See section 2.16: Logical Operators for examples

- and
- or
- not

Identity Operators

See section 2.17: Identity Operators for examples

- is
- is not

Membership Operators

See section 2.18: Membership Operators for examples

- in
- not in

This page titled 10.1: Introduction is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

10.2: if Statement

In an if statement a block of code is executed if a condition is TRUE, and you exit if it is FALSE.
if [condition statement]:
    executed action

Note

if statement is defined by a block of code which uses a colon and indentation

Note syntax features:
- colon
- indentation

Lets use an if statement to determine if an input is valid, for example, hydrogen is the smallest atom and so a molecule must have a molar mass greater than hydrogen. The following flow chart describes a program where you input a molar mass and use an if statement to determine if it is acceptable.

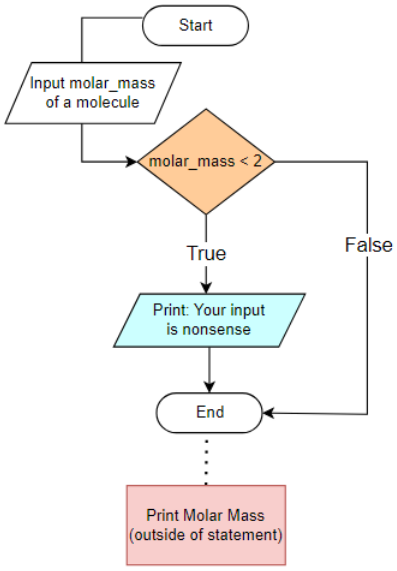


Figure 10.2.1: If statement to check if input is valid or not. (Belford CC 0.0)

The following code performs the action of the above flow chart

```
molar_mass=float(input("Enter the molar mass of a molecule in units of g\mol"))

if molar_mass < 2.0:
    print("Your molar mass makes no sense")

print(f"The molar mass you entered is: {molar_mass}g/mol")
```

run | restart | restart & run all

Note how we put a space between the input step, and the print statement that is outside of the loop, and this if statement has two lines of code.

Exercise 10.2.1

There is a flow in the logic of the above code Do you see it? The smallest molecule is H_2 and so you coud enter a mass of 1.5g/mol, which is smaller than the smallest molecule. Can you fix the code so it does not accept 1.5 as an input?

Answer

Make the value you are doing the less than equal to 2.

This page titled 10.2: if Statement is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

10.3: if-else Statement

In an if-else statement one block of code is executed if a condition is TRUE, and another block of code is executed if it is FALSE.

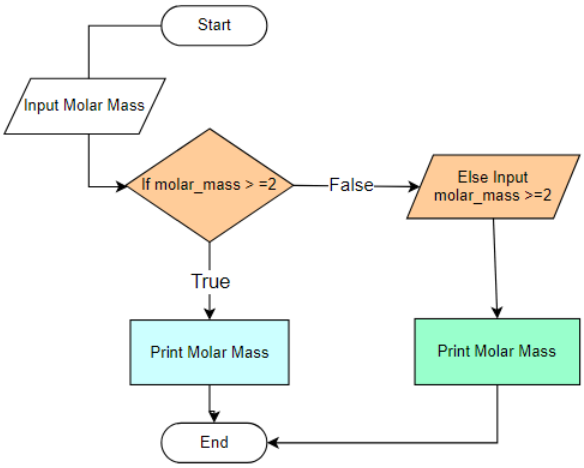


Figure \(\PageIndex{1}\): If-else condtional statement using greater than or equal comparison operator to ensure that molar mass entered is a valid number. (Belford CC-BY)

```
molar_mass=float(input("Enter the molar mass of a molecule in units of g\mol "))

if molar_mass >= 2.0:
    print(f"The molar mass you entered is: {molar_mass}g/mol")
else:
    molar_mass=float(input("Enter a molar mass greater than or equal to diatomic hydr
    print(f"The molar mass you entered is: {molar_mass}g/mol")
```

run | restart | restart & run all

```
Hello world!
```

This page titled 10.3: if-else Statement is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

10.4: if-elif-else Statement

An if-elif-else statement has two parts. The if-elif and the else. For each if or elif condition a block of code is executed if it is true, and the next condition is then checked if it is false. If none of the condition are true, the else block of code is executed. So if and elif execute when a condition is true, while else executes if it is false. Note the conditions are tested sequentially, and sometimes this can effect the logic.

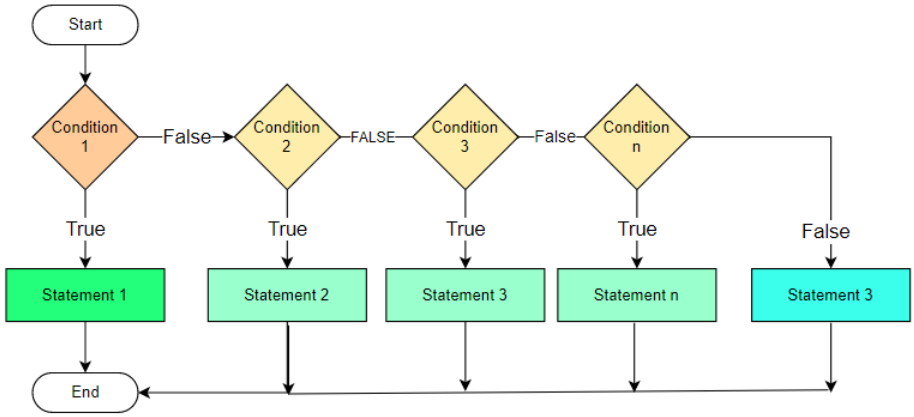


Figure \(\PageIndex{1}\): Copy and Paste Caption here. (Copyright; author via source)

The following code uses the List content type (section 2.7) and the membership operator (section 2.18). These lists are lists of strings (section 2.6) and you are seeing which list the element is in.

Caution

The following code only works if you capitalize the first letter of each element, which is a formatting issue (Boron works, boron does not). Note that element is a string, and we are using the membership to see if it is in a list of strings. Can you find a string method that can fix this problem so that boron will work? There are actually several methods to do this. So, change the code and run it. Here is a hint, you can reassign a variable name to the same variable name with a method operating on it (var1=var1.method), where var1 is the variable name and method is the method you are using.

```
element = input("Enter the element you want to know the family of: ")

if element == "Hydrogen":
    print("Hydrogen belongs to the nonmetal family.")
elif element in ["Lithium", "Sodium", "Potassium", "Rubidium", "Cesium", "Francium"]:
    print(f"{element} belongs to the alkali metal family.")
elif element in ["Beryllium", "Magnesium", "Calcium", "Strontium", "Barium", "Radium"]
    print(f"{element} belongs to the alkaline earth metal family.")
elif element in ["Boron", "Aluminum", "Gallium", "Indium", "Thallium"]:
    print(f"{element} belongs to the metalloid family.")
elif element in ["Carbon", "Silicon", "Germanium", "Tin", "Lead"]:
    print(f"{element} belongs to the nonmetal family.")
elif element in ["Nitrogen", "Phosphorus", "Arsenic", "Antimony", "Bismuth"]:
    print(f"{element} belongs to the pnictogen family.")
elif element in ["Oxygen", "Sulfur", "Selenium", "Tellurium", "Polonium"]:
    print(f"{element} belongs to the chalcogen family.")
elif element in ["Fluorine", "Chlorine", "Bromine", "Iodine", "Astatine"]:
    print(f"{element} belongs to the halogen family.")
elif element in ["Helium", "Neon", "Argon", "Krypton", "Xenon", "Radon"]:
    print(f"{element} belongs to the noble gas family.")
elif elment in ["Cerium", "Praseodymium", "Neodymium", "Promethium", "Samarium",
                "Europium", "Gadolinium", "Terbium", "Dysprosium", "Holmium",
```

```
                "Erbium", "Thulium", "Ytterbium", "Lutetium"]:
    print(f'{element} is a lanthanide')
elif element in ["Thorium", "Protactinium", "Uranium", "Neptunium", "Plutonium",
             "Americium", "Curium", "Berkelium", "Californium", "Einsteinium",
             "Fermium", "Mendelevium", "Nobelium", "Lawrencium", "Rutherfordium",
             "Dubnium", "Seaborgium", "Bohrium", "Hassium", "Meitnerium", "Darmstadti
             "Roentgenium", "Copernicium", "Nihonium", "Flerovium", "Moscovium", "Live
    print(f'{element} is an actinide')
else:
    print(f'element is a transition metal')
```

run | restart | restart & run all

```
Hello world!
```

This page titled 10.4: if-elif-else Statement is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

10.5: Compound if/elif Statements

Compound if/elif statements combine two if statements with a logical operator (and, or, not).

From general chemistry we can determine at which temperature a reaction is spontaneous or not spontaneous based on the enthalpy and entropy values (UALR Gen Chem sec. 18.5.3). That is, for a spontaneous reaction Δ G < 0, and Δ G = ΔH - TΔS. So a negative reaction enthalpy and a positive reaction entropy contribute to spontaneity, as outlined in the following table.

Table10.5.1, Temperature dependence of spontaneity as a function of the enthalpy and entropy changes of a reaction.

Type	ΔH	ΔS	ΔG
1.	[+]	[-]	Always [+], never spontaneous
2.	[-]	[-]	[-] at low T, enthalpy driven process only spontaneous at low T
3.	[+]	[+]	[-] at high T, entropy driven process only spontaneous at high T
4.	[-]	[+]	Always [-], spontaneous at all T, both enthalpy and entropy driven

and conjunction:

The following code uses a compound if/elif statement to provide the results of the first two condigtions of the above table, can you complete this code to cover the third and fourth conditions



Login with LibreOne to run this code cell interactively.

If you have already signed in, please refresh the page.

Login

```
#Temperature and reaction spontaneity
enthalpy=float(input("Enter a reaction enthalpy and include the sign: "))
entropy=float(input("Enter a reaction entropy and include the sign: "))

if enthalpy > 0 and entropy < 0:
    print(f"A reaction with {enthalpy} and {entropy} is never spontaneous")
elif enthalpy > 0 and entropy > 0:
    print(f"A reaction with {enthalpy} and {entropy} is entropy driven and spontaneou
```

```
else:
    print("You need to complete the code to handle this condition")
```

```
Hello world!
```

or conjunction:

The following is an example of using an "or" operator in an "if" statement to determine if an element is an alkaline earth metal:



Login with LibreOne to run this code cell interactively.

If you have already signed in, please refresh the page.

Login

```
#is an element an alkaline earth metal?

atom=input("Enter name of element: ")
#atom=atom.capitalize()
if atom == "Beryllium" or atom == "Magnesium" or atom == "Calcium" or \
   atom == "Barium" or atom == "Strontium" or atom == "Radium":
    print(f"{atom} is an alkaline earth metal")
else:
    print(f"{atom} is not an alkaline earth metal")
```

```
        hello world
```

? Exercise 10.5.1

In the above code type barium in lower case. Then uncomment the second line and try barium in lower case again. Explain the line of code you uncommented

Answer

If you go to the section on string class, you will see the str.capitalize() method. You could also have used the str.title() method.

? Exercise 10.5.2

The above code could get very extensive if you were going to extend it with elif statements to identify what family an element belongs to. Can you think of another approach that could be used

Answer

One additional approach would be to make a list for each family , and then use the membership operator to see if the element is in the list.

This page titled 10.5: Compound if/elif Statements is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

10.6: Nested if Statements

Nested if statements



Login with LibreOne to run this code cell interactively.

If you have already signed in, please refresh the page.

Login

```
#What is the phase of a substance?

melting_point=float(input("What is the melting point of your substance? "))
boiling_point=float(input("What is the boiling point of your substance? "))
temperature=float(input("What is the temperature you wish to know the phase at? "))

if temperature <= melting_point:
    if temperature == melting_point:
        print(f"the substance could be solid and liquid coexisting together")
    else :
        print(f"It is a solid")
elif temperature > melting_point and temperature < boiling_point:
    print(f"Substance is a liquid")
else :
    if temperature == boiling_point:
        print(f"It is at the boiling point")
    else:
        print(f"It is a gas")
```

```
Hello world!
```

This page titled 10.6: Nested if Statements is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

10.7: Loops - An Introduction

A loop structure is a programming construct that allows you to repeat a block of code until a terminating condition is met. There are two basic type of loop structures in Python, the "for" loop and the "while" loop. The basic difference between them is how they are terminated. In the for loop the code is repeated for a certain number of times, while in a while loop it is repeated while a condition is true. We will look at each of these in turn.

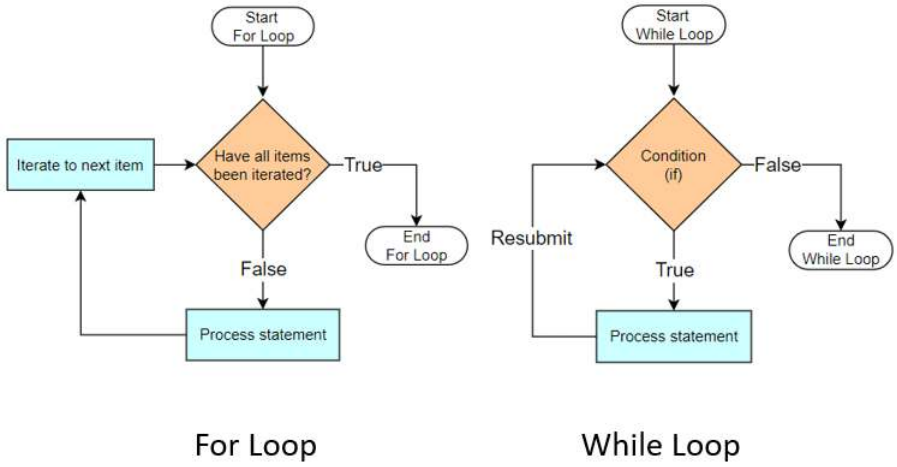


Figure 10.7.1: For and While loop flow charts (Belford cc 0.0)

This page titled 10.7: Loops - An Introduction is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

10.8: for Loops

the following list has the names of the elements in the periodic table, which is a list of strings. If you print the list, what you print is a list.

```
element_list = ['Hydrogen', 'Helium', 'Lithium', 'Beryllium', 'Boron', 'Carbon', 'Nit
                'Neon', 'Sodium', 'Magnesium', 'Aluminium', 'Silicon', 'Phosphorus',
                'Potassium', 'Calcium', 'Scandium', 'Titanium', 'Vanadium', 'Chromium
                'Nickel', 'Copper', 'Zinc', 'Gallium', 'Germanium', 'Arsenic', 'Selen
                'Strontium', 'Yttrium', 'Zirconium', 'Niobium', 'Molybdenum', 'Techne
                'Palladium', 'Silver', 'Cadmium', 'Indium', 'Tin', 'Antimony', 'Tellu
                'Barium', 'Lanthanum', 'Cerium', 'Praseodymium', 'Neodymium', 'Promet
                'Gadolinium', 'Terbium', 'Dysprosium', 'Holmium', 'Erbium', 'Thulium'
                'Tantalum', 'Tungsten', 'Rhenium', 'Osmium', 'Iridium', 'Platinum', '
                'Bismuth', 'Polonium', 'Astatine', 'Radon', 'Francium', 'Radium', 'Ac
                'Uranium', 'Neptunium', 'Plutonium', 'Americium', 'Curium', 'Berkeliu
                'Fermium', 'Mendelevium', 'Nobelium', 'Lawrencium', 'Rutherfordium',
                'Hassium', 'Meitnerium', 'Darmstadtium', 'Roentgenium', 'Copernicium'
                'Livermorium', 'Tennessine', 'Oganesson']
print(element_list)
print(type(element_list))
```

run | restart | restart & run all

```
Hello world!
```

If you want to just print the names of the elements you can run a for loop, and go through and print each item within the list

```
element_list = ['Hydrogen', 'Helium', 'Lithium', 'Beryllium', 'Boron', 'Carbon', 'Nit
for element in element_list:
    print(element)
```

run | restart | restart & run all

```
Hello world!
```

If you want to print them on a single line you can add: , end=", " to your print statement

```
element_list = ['Hydrogen', 'Helium', 'Lithium', 'Beryllium', 'Boron', 'Carbon', 'Nit
for element in element_list:
    print(element, end=', ')
print(f'\n \n The variable element_list is of class {type(element_list)}.\n \n The va
```

run | restart | restart & run all

```
Hello world!
```

Modulus

We can use the modulus to identify even and odd number by dividing by two

```
list1 = [1,2,3,4,5,6,7,8,9,10]
for num in list1:
    if num % 2 == 0:
        print(f'{num} is even')
    else:
        print(f'{num} is odd')
```

run | restart | restart & run all

```
  Hello world!
```

Shorthand assignment operators

If you remember, indexing a string starts with the zero position
The following code uses a for loop to print the letter at each index position

```
index_count = 0
word='abcde'
for letter in word:
    print(word[index_count])
    index_count = index_count + 1
```

run | restart | restart & run all

```
  Hello world!
```

We can reduce the amount of code by using the shorthand += assignment operator

```
index_count = 0
word='abcde'
for letter in word:
    print(word[index_count])
    index_count +=1
```

run | restart | restart & run all

```
          hello world
```

Range Function

The range function allows you to generate a list of numbers over a range. It starts with the zero position and give the number of digits in the number. So a range of 10 prints 0 through 9.

```
for num in range(10):
    print(num, end=' ')
print('\n',10*'*')

for num in range(3,10):
    print(num, end='.')
print('\n',10*'*')

for num in range(3,10,3):
    print(num, end=',')
```

run | restart | restart & run all

```
Hello world!
```

This page titled 10.8: for Loops is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

10.9: While Loops

In the code below enter a number less that 2 three times, and then one greater than 2.

```
fw=float(input("Formula Weight of molecue: "))

while fw <= 2:
    print(f' This is impossible as {fw} is smaller than hydrogen')
    fw=float(input("Input another Formula Weight of the molecule: "))
else:
   print("you may have a real molecule")
print("you are out of loop")
```

run | restart | restart & run all

```
   Hello world!
```

Notice you stayed in the loop as long as you gave an unrealistic molar mass, and so the program prevented you from inputing data that could not be real. The moment you inputed a value greater than 2, the loop terminated, and you could have used that variable in a subsequent calculation.

This page titled 10.9: While Loops is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

10.10: Nested Loops

Nested Loops

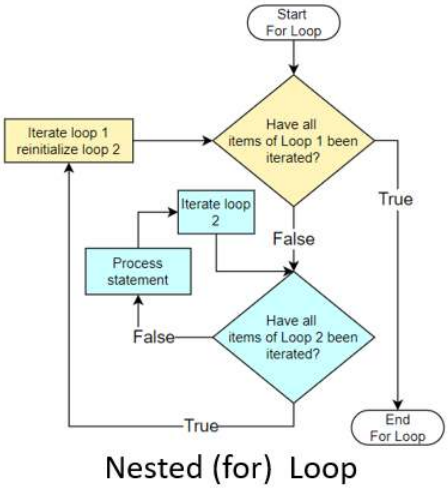


Figure 10.10.1: Nested for loop. (Belford cc 0.0)

This page titled 10.10: Nested Loops is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

10.11: Loop Control Structures

Break Statement

Stops execution of loop based on a statement

Predict the output if you input a value less than one



Login with LibreOne to run this code cell interactively.

If you have already signed in, please refresh the page.

Login

```
fw=float(input("Formula Weight of molecue: "))

while fw < 1:
    break
    print(f' This is impossible as {fw} is smaller than hydrogen')

else:
   print("you may have a real molecule")
print("you are out of loop")
```

```
Hello world!
```

What do you think will happen if you remove the break? **Note: you can click restart and run all to stop an infinite loop (or ctrl C in thonny)**

Continue Statement

Skips current iteration and moves onto next



Login with LibreOne to run this code cell interactively.

If you have already signed in, please refresh the page.

Login

```
number=(input("Input a number larger than 10: "))
print('This for loop will use the "continue" control structure to print all nonzero d
for value in number:
    if value=="0":
        continue
    print(value)
```

```
Hello world!
```

Pass Statement

stops loop from running



Login with LibreOne to run this code cell interactively.

If you have already signed in, please refresh the page.

Login

```
fw=float(input("Formula Weight of molecue: "))
while fw < 1:
    #While writing code you may want to use a pass statement while you figure things
    #without the pass you would go into an infinite loop
    pass
print("Out of loop")
```

```
Hello world!
```

This page titled 10.11: Loop Control Structures is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

CHAPTER OVERVIEW

11: Introduction to Electronics and First Circuit

Topic hierarchy

11.1: Electricity and Units
11.2: Introduction to Electric Circuits
11.3: Circuit Diagrams
11.4: Diagrams.net Electric Circuit Diagrams
11.5: Fritzing
11.6: Conductors
11.7: Circuit Boards
11.8: Resistors
11.9: Diodes
11.10: LED (Light Emitting Diodes)
11.11: Multimeter
11.12: GPIO Output (LEDs) - First Circuit

11: Introduction to Electronics and First Circuit is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

11.1: Electricity and Units

Electricity is part of our every day world and yet most of us have very little understanding of what electricity is. This chapter is more of a reference chapter and you should come back to it as needed.

Electricity is typically considered to be the flow of electric charge, although it can also be referred to stored or accumulated charge, usually in the form of electrons, and it can be produced by other sources of energy (coal, natural gas, solar energy, etc..) and harnessed to power electronic devices, provide lighting and do various forms of work. The coulomb (C) is the fundamental unit of charge and the flow of electricity is measured in amperes (C/s). Voltage (V) is the electric potential energy per unit charge (coulomb) and current times voltages is the power that is measured in watts (W). An electric circuit represents the path the current flows as the charge moves from a high to a low potential.

The flow of electric current can cause some confusion to chemistry students who understand that it is the electron that typically flows in a circuit, which is a negative charged particle and thus flows to the positive (high potential), so in engineering circuits the direction of flow is that for which a positive particle would flow.

We will start our discussion of electricity from the perspective of the units of measurement, noting that the way the SI base units that can describe all measurable phenomena were radically changed in 2019 to be based on 7 immutable constants (Chem 1402 Units of Measurement), which are defined (exact) numbers.

SI Units

All measurable phenomena can be measured with the 7 SI (*Système international d'unités*) base units as defined by the *Bureau international des poids et mesures* (BIPM). These base units are:

1. meter (length)
2. kilogram (mass)
3. second (time)
4. Kelvin (absolute temperature)
5. mole (quantity)
6. ampere (electric current)
7. candela (luminous intensity)

In 2019 the way these units were defined was changed to be based on seven immutable constants of nature (figure 1).

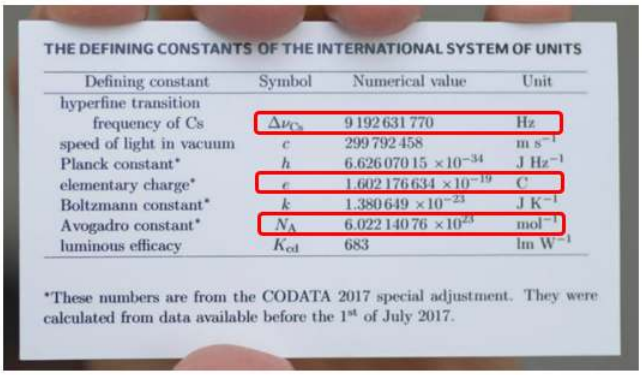


Figure 11.1.1: Seven Immutable Constants that SI base units are defined by. (Stoughton/NIST)

Coulomb (C): Unit of Charge (q)

The Coulomb (C) is defined by the charge of an elementary particle (electron, or positron) where the charge is defined by:

$$\text{Charge of an electron } (e^-) \equiv -1.602176634x10^{-19}C \tag{11.1.1}$$
$$\text{Charge of a positron } (e^+) \equiv 1.602176634x10^{-19}C$$

Therefore:

$$1 \text{ Coulomb (C)} = \frac{1}{1.602176634x10^{-19}C} = \text{the charge of } 6.241509074x10^{18} \text{ electrons or positrons} \quad (11.1.2)$$

In electronics the symbol q is used to denote charge of an individual particle and C is the unit. As electrons and not positrons are what commonly flow in electric circuits we will implicitly refer to an electron as something with a negative charge, without explicitly stating so.

Faraday (F): Charge of a mole of electrons (Q)

Multiplying the SI constant of the charge of an electron (e) with that of quantity (the mole, N_A , Avagaordo's number) gives us the charge of a mole of electrons, which is Faraday's constant:

$$1F \equiv 1(N_A)(e) = (\frac{6.02214076x10^{23} electrons}{mole\ electrons})\frac{1.602176634x10^{-19}C}{electron} = 96,485.3321\frac{C}{mol \cdot electrons} \quad (11.1.3)$$

or more commonly

$$1F = 96,500\frac{C}{molelectrons}$$

In electronics Q represents the total charge passing through a conductor over a period of time. A capacitor is a common electrical device that stores charge in the units of Faradays.

Ampere (A): Unit of Current (I)

We are often interested in the work done as charge flows, that is as charge flows over time, so if we divide the Si constant for charge by time we get the ampere.

$$1A \equiv \frac{1C}{1\ sec} \quad (11.1.4)$$

96,500 amps = flow of one mole of electrons

In electronics the symbol I is used to denote current and amp is the unit

Volt (V): Unit of Electric Potential (V or ΔE)

$$1\text{Volt} \equiv \frac{1 \text{ Joule}}{\text{Coulomb}}, \text{ in terms of the base SI units: } 1V = \frac{kg \cdot m^2 \cdot s^{-2}}{A \cdot s} \quad (11.1.5)$$

Where the Joule is the unit of energy, which is the capacity to do work or transfer heat. This can be understood by comparing translational work to electronic work

- Translation Work: Energy (J) = W (work) = F · Δx
    - Work done when a force displaces an object in a straight line
- Electronic Work: Energy (J) = W (work) = QV (or in chemistry we often write this as W = q ΔE)
    - Work done when charge is moved across an electric potential
    - Current flows from high to low potential
    - Note, here W means work, but in electronics W is also the unit of electric power, the Watt.

so

$$1V = \frac{1J}{1C} \ \ and \ \ C = \frac{1J}{1V} \quad (11.1.6)$$

so we can relate work to Faraday's constant

$$1F \ = \ \frac{96,500C}{mole\ e^-} \ = \ \frac{96,500J}{V \cdot mole\ e^-} \quad (11.1.7)$$

noting ΔG is Gibbs free energy and n = moles electrons, we get the expression

$$\Delta G = -nFE^o_{cell} = -QV \quad (11.1.8)$$

Where nF is the total charge of electrons (Q) and V is the electric potential drop. For a spontaneous reactions the change in Gibbs free Energy is negative ($\Delta G < 0$) and this equates to a positive cell potential. This equation relates the sign of spontaneous flow from the perspective of free energy to that of electric potential as used in circuit diagrams, and a spontaneous flow will occur when V is positive (and (ΔG is thus negative).

In electronics the Symbol V is used to denote electric potential, and it also the symbol of the unit for electric potential. That is, in V=IR we are saying the voltage equals the current times the resistance, but in 10 V, we are using it as a unit of measurement.

Watt (W): Unit of Electric Power (P)

Power is the rate at which work can be performed and has the unit of Watt

$$1W \equiv \frac{1J}{sec} = \frac{C \cdot V}{sec} = A \cdot V \tag{11.1.9}$$

In electronics the symbol P is used to denote power and Watt is the unit. Note, in the above equation V is being used as a unit, while in the next equation it is being used as a symbol for voltage

$$P = IV \tag{11.1.10}$$

IV Curves

A plot of the current as a function of the potential for an electric device is can provide information on. its power efficiency. This are often plotted to determine the best operating potential for a device. If the device is an energy source, like a solar cell, than you want to operate at the highest point on the IV curve that has enough potential to drive the process you want to drive, as that produces the maximum amount of work ($P_{mp}=I_{mp}V_{mp}$).

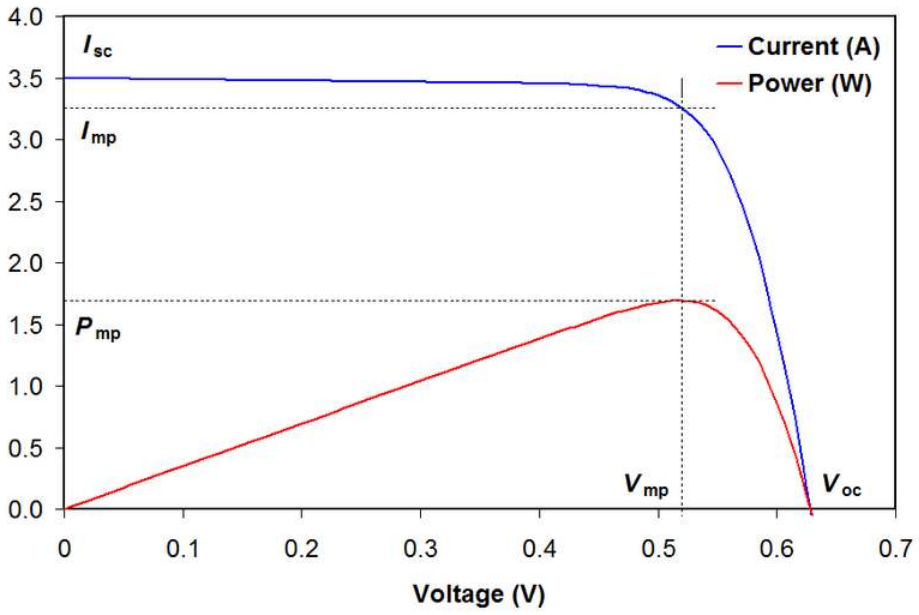


Figure 11.1.1: Generic solar cell IB curve. (CC 3.0 Squirmymcphee, Wikimedia commons)

The above plot is a measure of the current as a function of the voltage as you apply a load and the maximum power the device can produce is when you operate at V_{mp}. Note, in this plot the current is the dependent variable and the voltage is the independent variable and the power is the product of these two (P=IV). As you increase the voltage by adding a load the current drops. V_{oc} is the open circuit voltage when no current flows (infinite load), and I_{sc} is the short circuit current when no load is applied

Watt Hour (Wh): Unit of Electric Energy (E)

$$1Wh \equiv 1W \cdot h = \frac{1J}{sec} \cdot 1h = 3600 Joules \tag{11.1.11}$$

Your electric meter reads in kWh or kiloWatt hours

Ohm (Ω): Unit of Resistance(R) and Impedance(Z) to Electric Current Flow

Resistance (R)

$$1\Omega \equiv \frac{1V}{1A} \tag{11.1.12}$$

Resistance is a measure of the resistance to the flow of electric current and has units of Ω. It is a property of the material with a conductor having a low resistance. For a wire, the smaller the diameter the greater the resistance. For a given voltage the greater the resistance the lower the current. The relationship between resistance, voltage and current is given by Ohm's Law

$$\text{Ohm's Law} \quad V=IR \tag{11.1.13}$$

Impedance (Z)

In alternating circuits there are two other types resistance known as reactance, which occur due to the change in voltage (in contrast to the voltage itself) and the impedance (Z) is a function of both resistance (R) and the total Reactance (X_T)

$$Z = \sqrt{R^2 + X_T^2} \tag{11.1.14}$$

Reactance (X)

Reactance is the resistance to flow due to a change in the voltage, not the voltage itself and always occurs in AC circuits. there are two types of reactance, inductive and capacitive.

Two Types

- X_L = Inductive Reactance - due to induced magnetic currents
- X_C = Capacitive Reactance - due to capacitor charge/discharge

Total Reactance (X_T)

- X_T = Total reactance
- $X_T = X_L + X_C$

Note

In this class we will be working predominantly with DC (Direct Current) circuits and will probably not have to deal with reactance and impedance. It needs to be understood though that the Ω is really the unit of Impedance, which has two components, the resistance and reactance. For DC circuits the reactance is often zero and so the impedance is the resistance, and this systems follow Ohm's Law.

V=IR

SI prefixes

Si prefixes are base 10 multipliers that allow on to quickly scale the magnitude of a number across a wide range of values. The following table gives the SI prefixes.

Table 11.1.1: SI prefixes.

Multiplier	Name	Abbreviation	Name	Multiplier
10^{+30}	quetta*	Q	q	quecto*	10^{-30}
10^{+27}	ronna*	R	r	ronto*	10^{-27}
10^{+24}	yotta	Y	y	yocto	10^{-24}

Multiplier	Name	Abbreviation		Name	Multiplier
10^{+21}	zetta	Z	z	zepto	10^{-21}
10^{+18}	exa	E	a	atto	10^{-18}
10^{+15}	peta	P	f	femto	10^{-15}
10^{+12}	tera	T	p	pico	10^{-12}
10^{+9}	giga	G	n	nano	10^{-9}
10^{+6}	mega	M	m	micro	10^{-6}
10^{+3}	kilo	k	m	milli	10^{-3}
10^{+2}	hecto	h	c	centi	10^{-2}
10^{+1}	deca	da	d	deci	10^{-1}

*quetta, ronna, ronto & quecto were added in 2022 (https://www.bipm.org/en/cgpm-2022/resolution-3).

Use of SI prefixes

For real large or small numbers, the convention is to place a number with between 1 and 999 in front of the SI prefix, and use the appropriate prefix to show the value. So a memory stick with 1,200,000 bytes would be written as 1.2 Mbyte, not as 1,200 kbytes.

$$\begin{aligned} 1,330,000,000m &= 1.33 \times 10^9 m = 1.33\ gigameters \\ 13,300,000,000m &= 13.3 \times 10^9 m = 13.3\ gigameters \\ 133,000,000,000m &= 133 \times 10^9 m = \ 133\ gigameters \end{aligned} \tag{11.1.15}$$

SI prefixes and Calculators

What is the ENG key on scientific calculators?



Figure 11.1.8: Three display modes on TI 30 calculator.

In Scientific Notation there is on digit to the left of the decimal, in Engineering notation there are 1, 2 or 3 digits to the left of the decimal, and you then use one of the Si prefixes that is of the magnitude 10^{3n}.

The ENG key takes large or small numbers and expresses them as integer multiples of 10^3 or 10^{-3}, effectively converting them to values that can be expressed with SI prefixes. If you type 1,200,000 into your calculator and press ENG, it gives 1.2 x 10^6, so 1,200,000 bytes of memory on a memory stick would be 1.2 MBytes (you do not say 1,200 kBytes). If you try 0.12 and press ENG, it gives 120 x 10^{-3}, so 0.12g is 120 mg. So the ENG function quickly converts numbers to values that can easily be expressed with SI prefixes.

Note

Students are expected to be fluent in common SI prefixes as they are ubiqutous in electrical engineering. You will be working with calibrated electrical components like MΩ and KΩ resistors and pF (Farad) capacitors, and need to know what the prefixes mean.

Tutorials

- SparkFun: What is Electricity
- SparkFun: Voltage, Current and Resistance

- SparkFun: Electric Power
- SparkFun: Polarity

This page titled 11.1: Electricity and Units is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

11.2: Introduction to Electric Circuits

Voltage and Direction of Current Flow

A voltage drop (V) across two points creates an electric field (E) that impacts a force on a charged particle (F=qE) and causes it to move. Note, the voltage drop (V) is often called the "Electromotive Force" and this can cause some confusion as it is not a force, but a measure of the electric potential difference. See 5.5: Open Stax Physics and 10.2: Electromotive Force for a discussion of electric fields and Electromotive force. As the particle moves the potential energy across the field is converted to kinetic energy and the field strength decreases. In electronics we consider the current to flow from the region of high potential to the region of low, as if the particle had a positive charge, where in reality, the particle is a negative electron and flows towards the higher potential. This causes a lot of confusion for chemistry students, who like to visualize the current at the electron moving towards the positive voltage, which it does, but the current is defined as moving in the other direction, as in animated gif of figure 11.2.1.

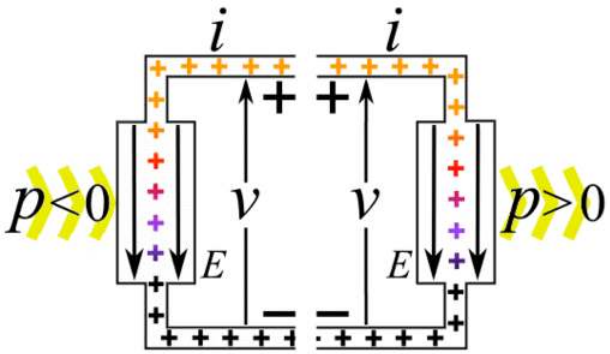


Figure 11.2.1: Copy and Paste Caption here. (Qniemiec ; cc 4.0, Wikimedia commons)

There are several things to note from figure 11.2.1

- P represents the power from the perspective of the system doing work on the surroundings, which is the opposite perspective taken in thermodynamics. That is, in thermodynamics positive energy is when energy is added to the system and negative is when the system does energy on the surroundings. There are two facets to the energy flow, on the left you have a power source, and on the right a load.
    - **Power Source:** On the left (p<0) is the power source that creates the positive voltage. This is a non-spontaneous endothermic process in that energy is being added to the system to pump the charge up the gradient. This could be a device like a generator connected to a hydroelectric turbine that uses the energy of water flowing down hill to move coils in an electric field, of a solar cell that uses the energy of a photon from the sun to excite electrons into higher energy states. A battery is a power source that is driven by a spontaneous chemical redox reaction.
    - **Load:** On the right is the load (p>0), and this represents spontaneous flow of charge across the electric field that could be converted to mechanical work like running a motor, or the generation of heat or light, just to describe a few processes that can be driven by the flow of current.
- i represents the flow of current, here being depicted by the moving positive charged particles
- v represents the voltage, the potential drop across the load that is created by the power source. The current flows from high to low potential
- E is the electric field that the charged particle interacts with, it is a result of the potential drop (voltage)
- The current must form a closed loop to maintain flow or the accumulation of charge would decrease the electric field that interacts with the charged particles to cause the current to flow.

Electric Circuits

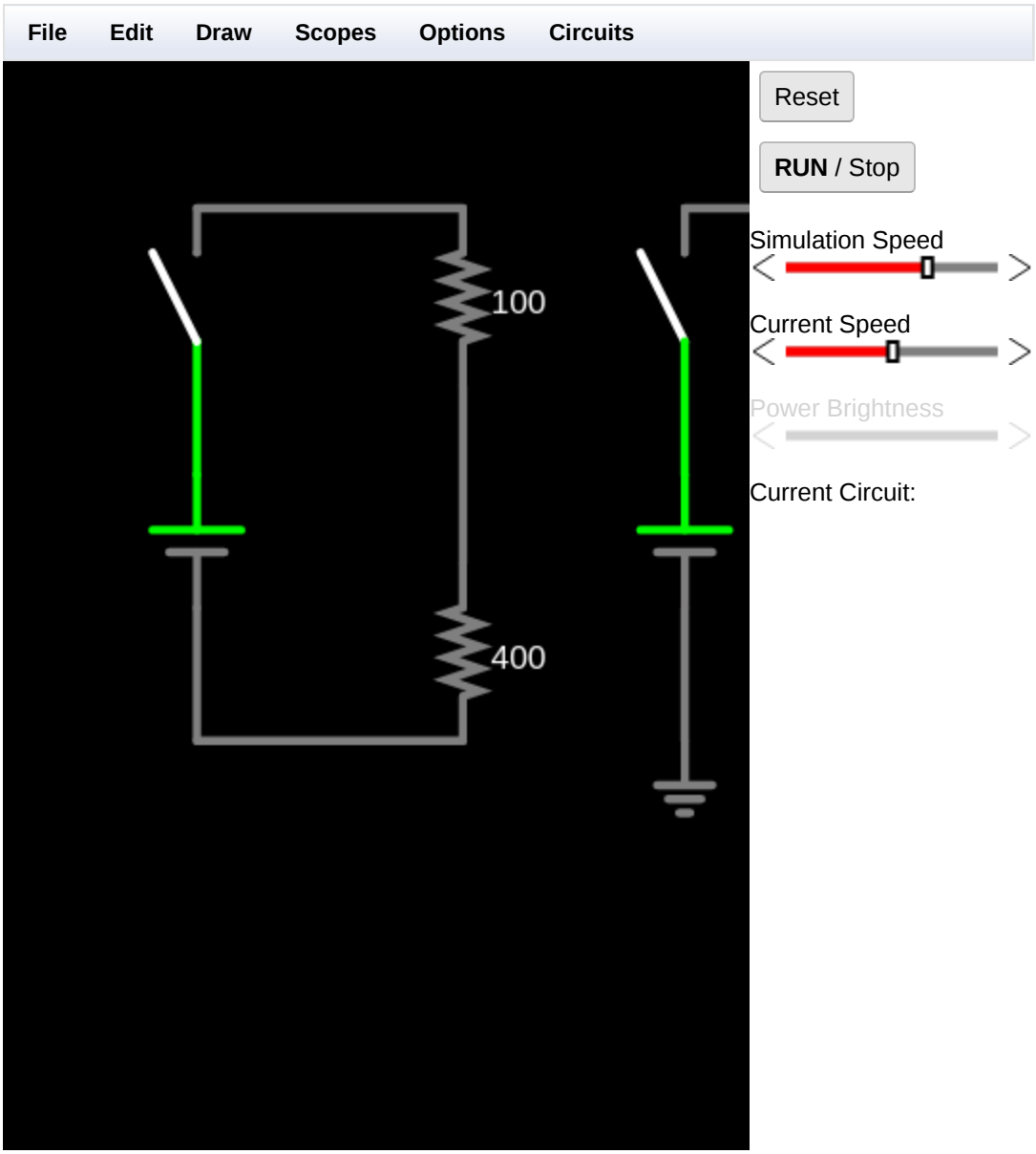
- Current flows through wires and electrical components in a circuit
- Flow is driven by the voltage drop and goes from high to low potential.
    - In a battery the high potential is the [+] electrode and the low potential is the [-] electrode.

- Circuits must form a closed loop or charge will accumulate and create an opposing electric field that stops the flow
    - Ground wires can complete the circuit

The following Falstad Circuit Simulator has two representations of identical circuits, the first shows wires completing the loop, while the second uses two ground wires to complete the loops. These simple circuits have 4 types of components; power source, load, switch and wires. If you hover your mouse over them the applet will provide information on them. The components are:

- Power source (two parallel plates) representing the electodes of an electrochemical cell. The larger (green) one is the high (positive) voltage and this device uses some form of process to create the voltage difference.
- Resistors (series of zig-zag lines) are the load, and here you have 100 and 400 Ω resistors connected in series.
- Switch (white line) can open or close the circuit. By default it is open and you need to click on the switch to close it.
- Wires (lines)
    - Ground wires (three triangle lines on right circuit) show places in the diagram that are connected.

Be sure to play with this diagram and open and close the switch of the two different representations of identical circits.



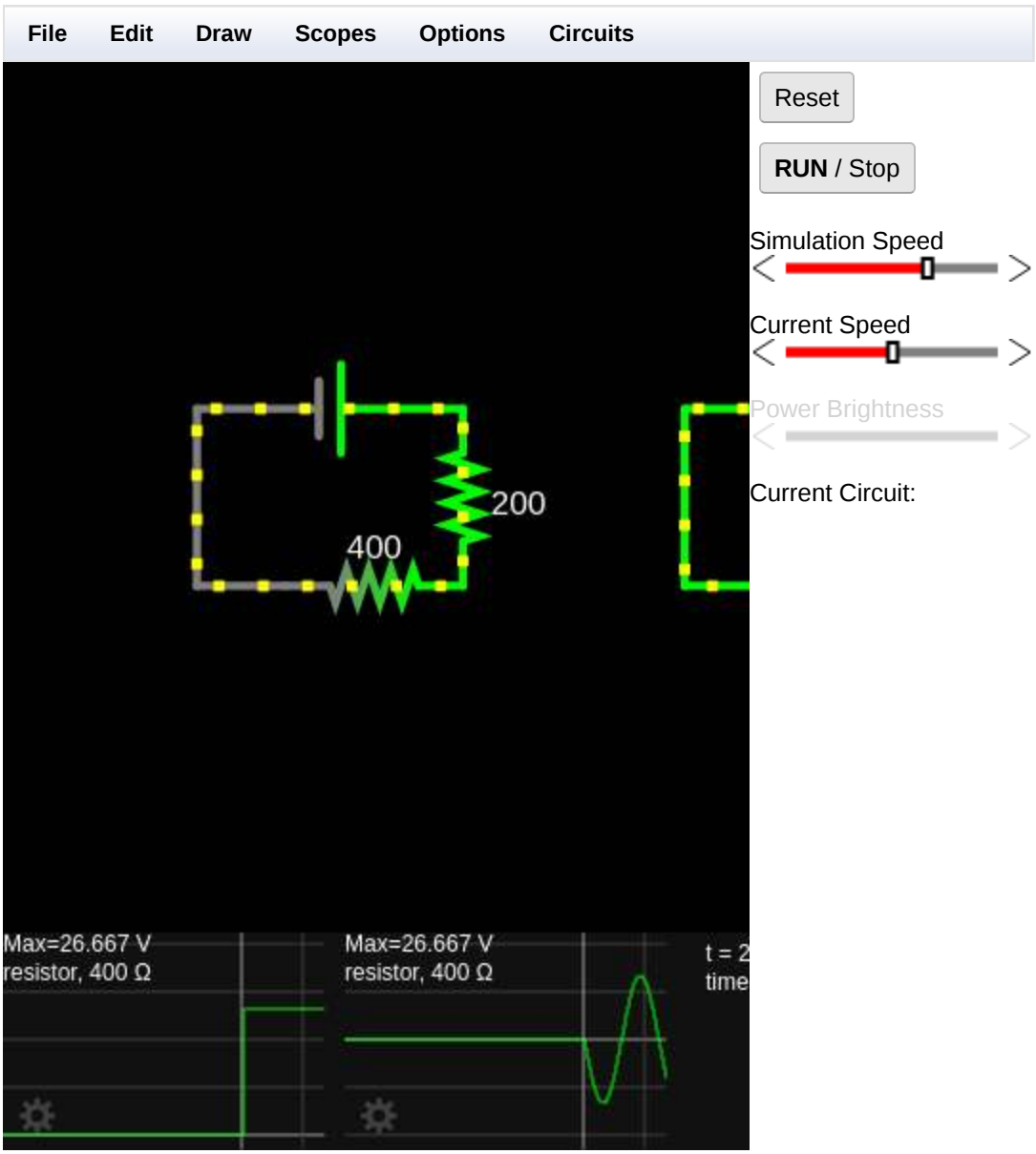
Falstad Circuit.JS 11.2.1: Two identical simple circuits, one showing a closed loop and the other using ground terminals. Hoover over one of the grounds, right click and choose Delete. The circuit stops because you no longer have a closed circuit. Now hit <ctrl> Z (undo) and the ground returns.

AC vs DC Circuits

There are two basic types of electric circuits; direct current (DC) and alternating current (AC). Most of the electronic devices we will work with use DC, but the power grid and outlets in your house are AC.

- AC- Alternating Current
    - Oscillating change in diretion of flow
    - Easier to increase voltage which results in less power loss over long distance transmission, which is why power lines use AC
- DC - Direct Current
    - Typically used in consumer projects
    - We will use DC for our projects

The Falstad simulation 11.2.2 below shows 40 V AC and DC power supplies hooked up to 200 and 400 Ω resistors in series. At the bottom is an oscciloscope reader across the 400 Ω resistor showing the voltage being a constant 26.67 V for the DC power source, and sinusoidally oscillating between 26.67 abd -26.67 V for the AC power source. Note, the scales of the two scopes are different



Falstad Circuit Simulation 11.2.2: 40V DC and AC power supplies.

Tutorials

- SparkFun: What is a Circuit?
- SparkFun: Series and Parallel Circuits
- SparkFun: Alternating Current vs. Direct Current

This page titled 11.2: Introduction to Electric Circuits is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

11.3: Circuit Diagrams

Circuit diagrams are commonly used to represent the compoenents and connections with an electrical circuits and there are several conventions

Generic Circuit Diagram

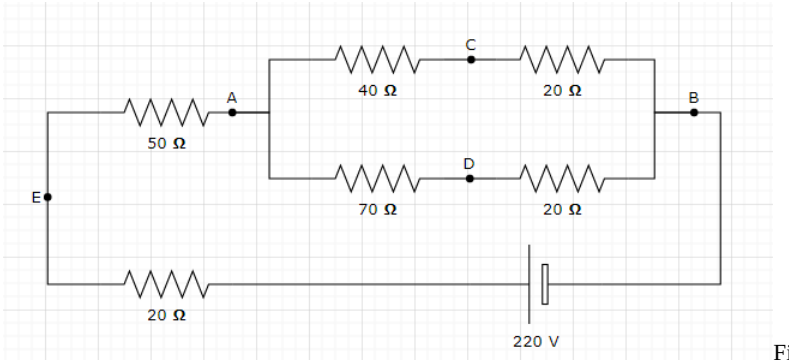


Figure 11.3.1: Copy and Paste Caption here. (Copyright; author via source)

Common Symbols

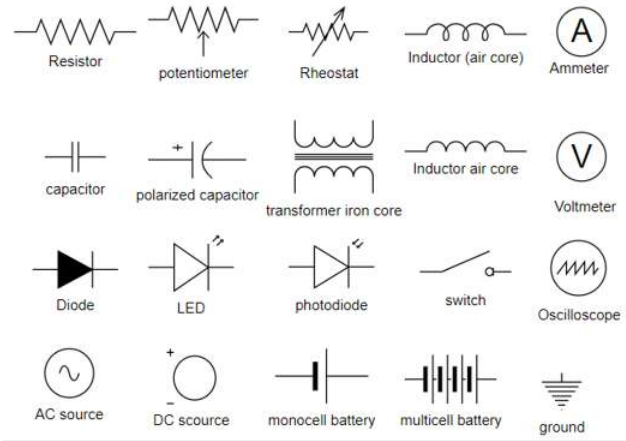


Figure 11.3.1: Copy and Paste Caption here. (Copyright; author via source)

Tutorials

- Sparkfun: How to Read a Schematic
-

This page titled 11.3: Circuit Diagrams is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

11.4: Diagrams.net Electric Circuit Diagrams

The Diagrams.net software you connected to your Google Drive to do flow charts has many other templates, and in the Engineering Option there are two electrical diagram templates as seen in figure 11.4.1

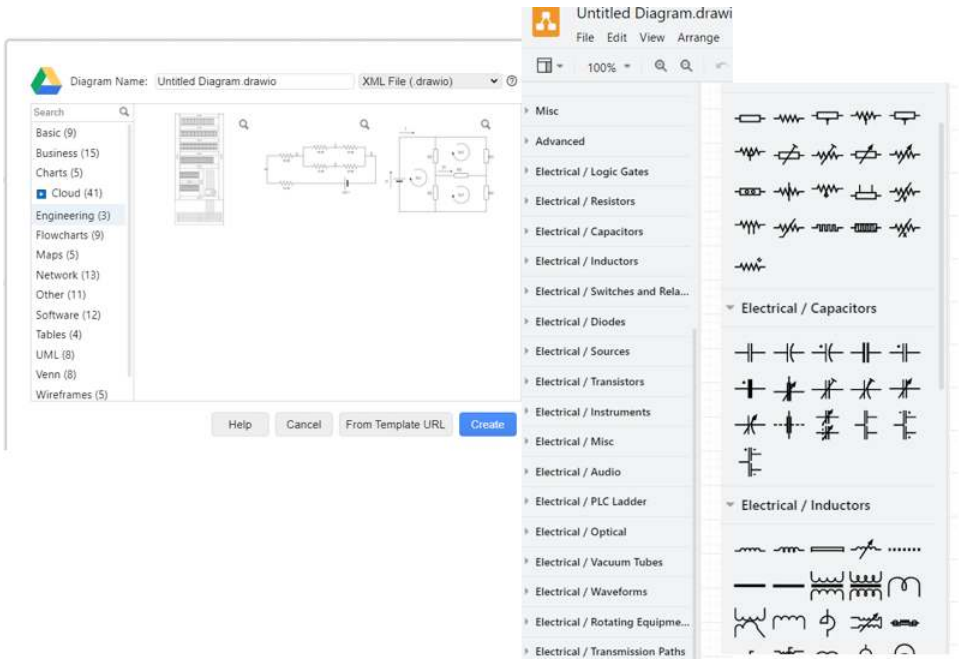


Figure 11.4.1: Drawio/Diagrams.net template for electric circuits. (Screen capture by Belford, 2023, cc 0.0)

The following YouTube shows you how to add the electrical circuits options to Draw.io/Diagrams.net



Arduino Scratch Pad

If you are using Arduino you may want to add the template as described in this YouTube:



This page titled 11.4: Diagrams.net Electric Circuit Diagrams is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

11.5: Fritzing

Fritzing is an open-source CAD (Computer Aided Design) software with downloadable libraries developed by the Instructional Design Lab at Potsdam University in Germany. It is great for the type of projects we are building and for creating breadboards with circuits like the one in figure 11.5.1. The libraries have the common components used in Raspberry Pi and Arduino projects and you can even convert your designs to PCBs (Printed Circuit Boards). Furthermore, many vendors and GitHub repositories provide Fritzing files for their components and projects, which you can find this through a quick Google search.

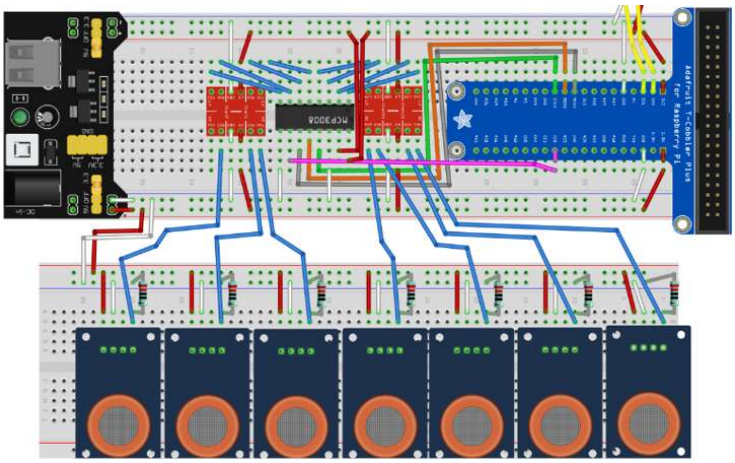


Figure 11.5.1: Fritzing showing connections of eight Metal Oxide Sensors to a bread board. (Hunter Tiner, CC-BY)

Create an account and download Fritzing

1. Open https://fritzing.org/
2. Go to “Sign up” in the top right corner and create a new user account
3. Check your email and follow the link to login

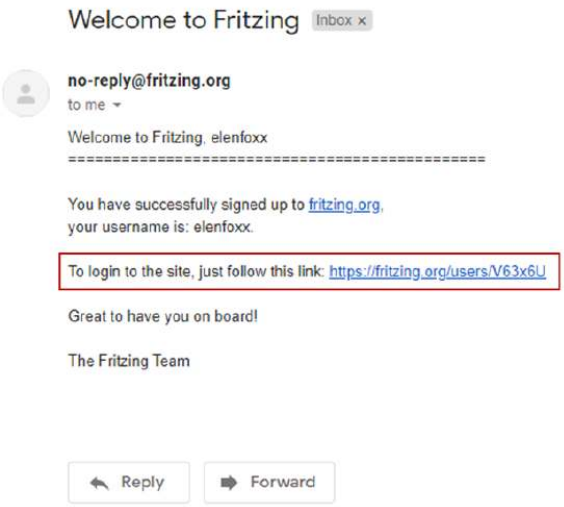


Figure 11.5.2: Email confirmation

4. Look for a link to download the latest version

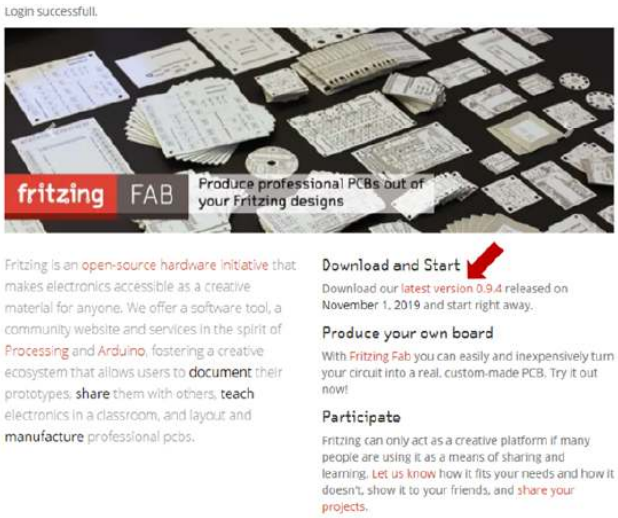


Figure 11.5.3: Click on the link indicated in the screenshot

5. Choose one of the options. You don't have to pay, Fritzing is an open source platform, but feel free to support the project.

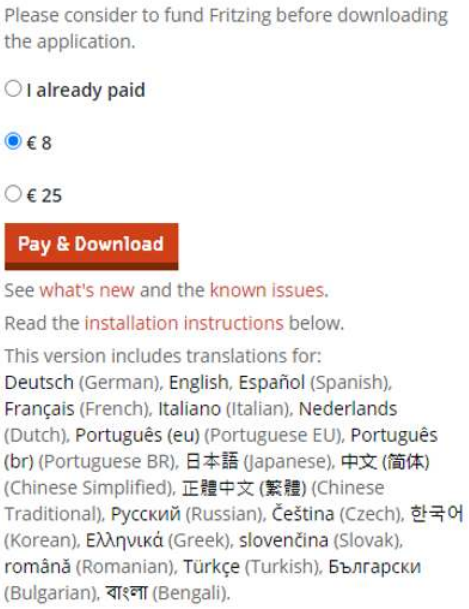


Figure 11.5.4: You can download Fritzing for free or choose to donate to the project

6. Select the platform and download Fritzing to your device



Figure 11.5.5: Select your platform from the list

7. Go to your Downloads folder and extract the files

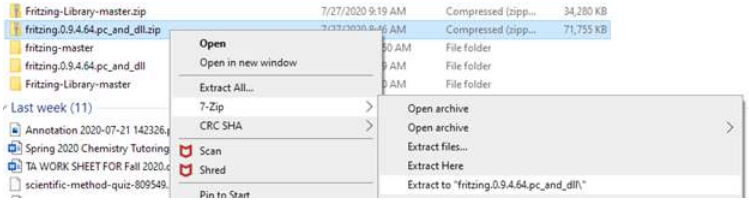


Figure 11.5.6: Extract the files

8. You can leave the folder in Downloads (which is NOT recommended since you might accidentally delete it) or move it to any other location on your computer.

9. Go to fritzing.0.9.4.64.pc_and_dll folder, then fritzing.0.9.4.64.pc and find Fritzing.exe. Right-click on it and create a shortcut.

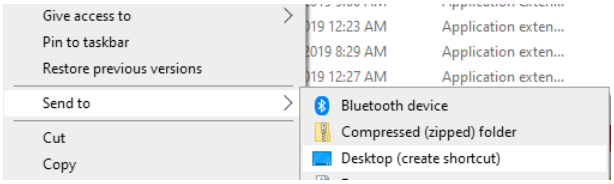


Figure 11.5.7: Create a shortcut for Fritzing

10. Start Fritzing

Downloading Fritzing Libraries

1. Open https://github.com/adafruit/Fritzing-Library/

2. Download the library (Click on Code à Download)

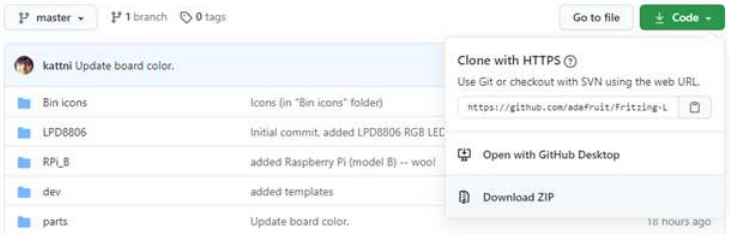


Figure 11.5.8: Download the library

4. Extract the files in your Downloads folder.

3. Go to the Fritzing. Click on File --> Open --> AdaFruit.fzbz

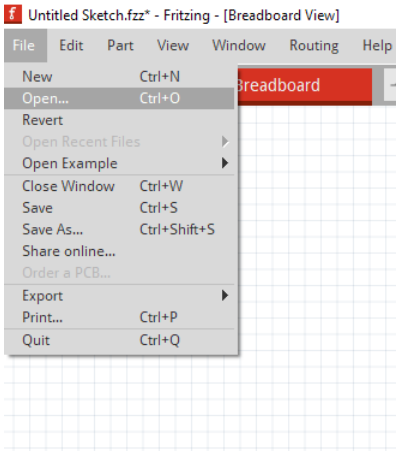


Figure 11.5.9: Import AdaFruit library

5. Wait for the library to install

6. You will see the ADA Fruit library in the “Parts” section on the right

7. Now you need to save this library (bin) so you can use it later. Click on the menu icon next to the library name à Save bin

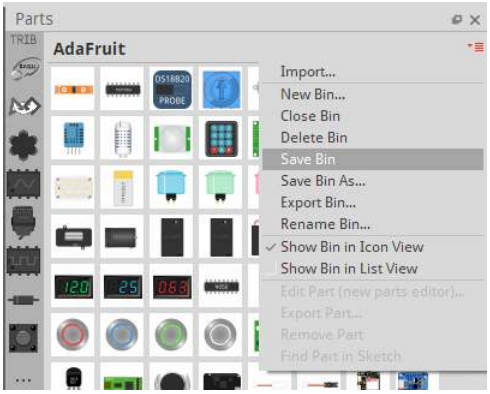


Figure 11.5.10: Don't forget to save the library every time you make changes

8. Repeat steps 3-7 for Adafruit Raspberry Pi.fzbz library

Finding new parts for Fritzing

No one is perfect and you might come across some parts in Fritzing that just don’t fit the breadboard. For example, depending on the library version, your Adafruit T-Cobbler Plus could be the wrong size

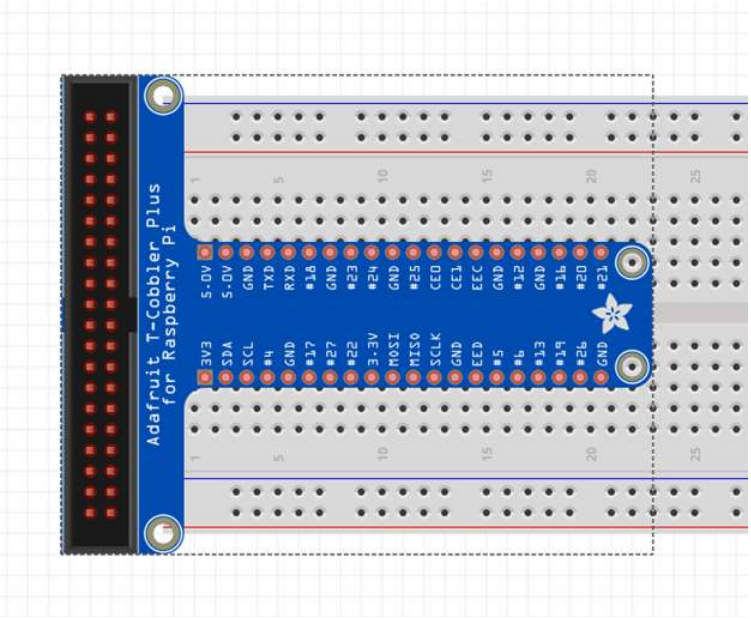


Figure 11.5.11: As you can see, the connections are red (they are supposed to be green) and they are located in between the breadboard's holes

1. Google "Adafruit T-Cobbler Plus Fritzing" and click on the first GitHub link

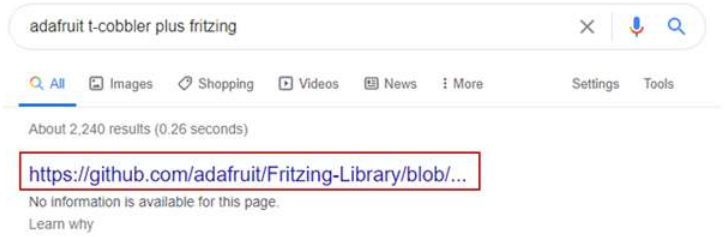


Figure 11.5.12: Copy and Paste Caption here. (Copyright; author via source)

2. Download the file



Figure 11.5.13: Download the fzbz file

3. Since the new cobbler that you just downloaded has the same part number you need to delete the old cobbler from Fritzing before adding the new one. Right-click on the part you want to delete and select Remove Part



Figure 11.5.14: Remove the old part from the bin

4. Save bin and close Fritzing
5. Open Fritzing, navigate to the Raspberry Pi library.
6. Click the menu Icon next to the library name and select Import.

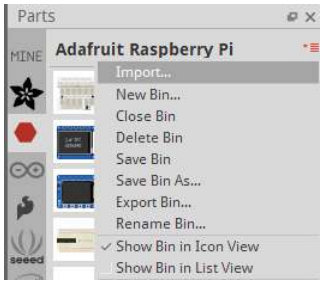


Figure 11.5.15: Import the new part

7. Select the new cobbler file
8. Save the bin

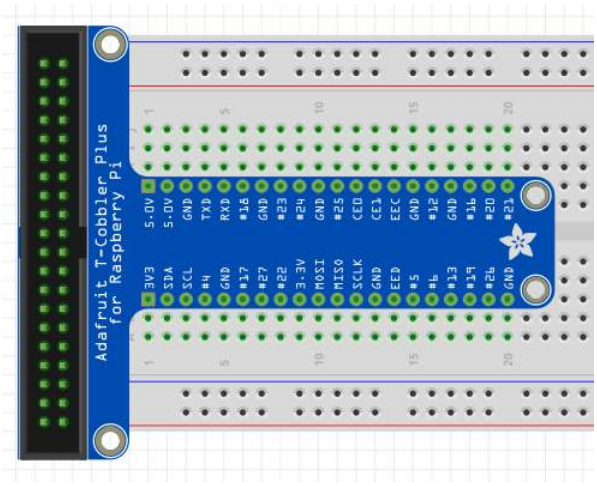


Figure 11.5.16: The new cobbler file fits perfectly!

First circuit with Fritzing

Follow the instructions from Task 3: Building your First Circuit and build your first circuit. You can find all necessary parts in Fritzing.

Resistors found in Fritzing are 220ohm by default. To change that:

1. Add a new resistor to your breadboard

2. In the lower right part of your window you can find the Inspector section that contains information about the selected item
3. Select the resistor that you added to your breadboard
4. In the Inspector section use the drop down menu next to the resistance setting to change it to 330ohm

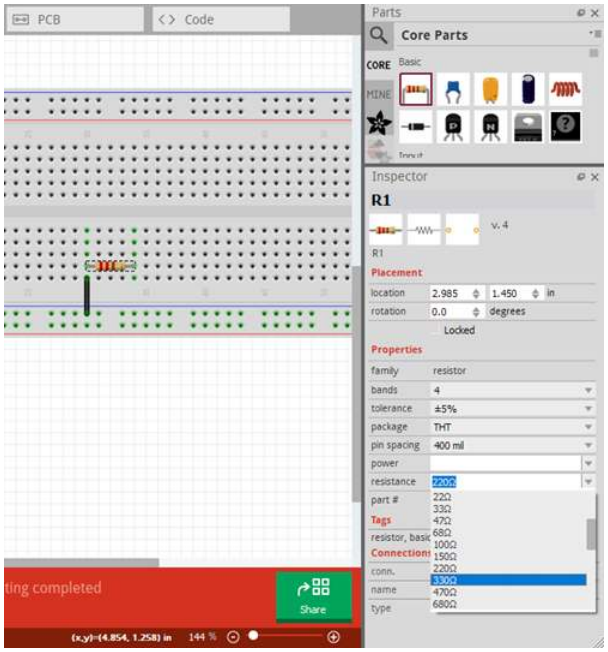


Figure 11.5.17: Changing the resistance

5. The color of the bands on the resistor will change accordingly.

Contributors and Attributions

- Elena Lisitsyna

This page titled 11.5: Fritzing is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

11.6: Conductors

Conductors are substances that conduct charge, which is typically electrons, although ions can also function as conductors within a circuit.

- Have mobile charge particles, typically electrons
- May be ions
- Metals make good conductors
  - weakly held valence electrons
  - copper is a common metallic conductor
  - form wires
    - stranded
    - solid core
- Printed Circuit Boards (PCBs)
  - can result in very small circuits

Wire Conductors

- Resistance goes up as length increases
- Resistance goes up as diameter decreases
- Resistance depends on Materials
- Connections often increase resistance
  - dissimilar metals can react
  - nonconducting ionic compounds (typically oxides) can form
  - loose connections can increase resistance

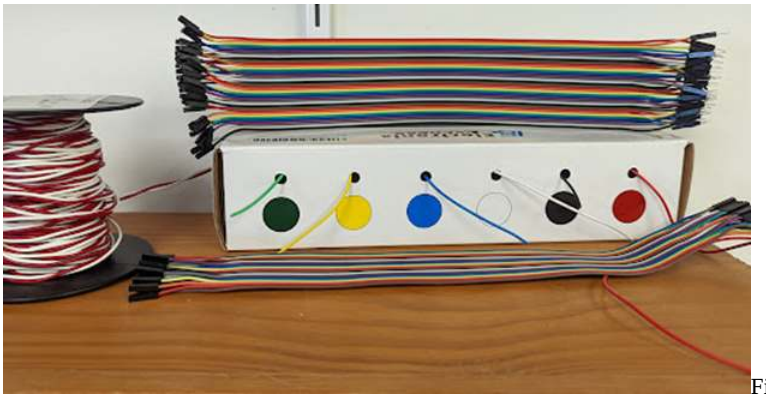


Figure 11.6.1: Color coded solid core and standed wires. (Belford cc 0.0)

AWG Gauge

American Standard Wire Gauge - The higher the number the thinner the wire

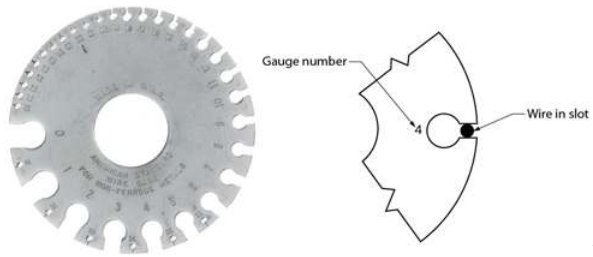


Figure 11.6.2: AWG for non-ferrous metals. (BC Industry Training Authority, CC-BY-NC-SA)

Table 11.6.1 : AWG wire data for copper conductors.

AWG	Diameter (in)	Diameter (mm)	Resistance/length (mΩ/m)	Resistance/length (mΩ/ft)
0000 (4/0)	0.4600	11.684	0.1608	0.04901
00 (2/0)	0.3648	9.266	0.2557	0.07793
0 (1/0)	0.3249	8.251	0.3224	0.09827
2	0.2576	6.544	0.5127	0.1563
4	0.2043	5.189	0.8152	0.2485
6	0.1620	4.115	1.296	0.3951
8	0.1285	3.264	2.061	0.6282
10	0.1019	2.588	3.277	0.9989
12	0.0808	2.053	5.211	1.588
14	0.0641	1.628	8.286	2.525
16	0.0508	1.291	13.17	4.016
18	0.0403	1.024	20.95	6.385
20	0.0320	0.812	33.31	10.15
22	0.0253	0.644 5	52.96	16.14
24	0.0201	0.511	84.22	25.67
26	0.0159	0.405	133.9	40.81
32	0.00795	0.202	538.3	164.1
40	0.00314	0.0799	3441	1049

Stranded vs. Solid Wires

- Solid wires
  - not as flexible
  - preferred if flexibility is not needed
- Stranded wires
  - flexible
  - need connectors to plug into breadboards
  - connectors often become loose and cause problems

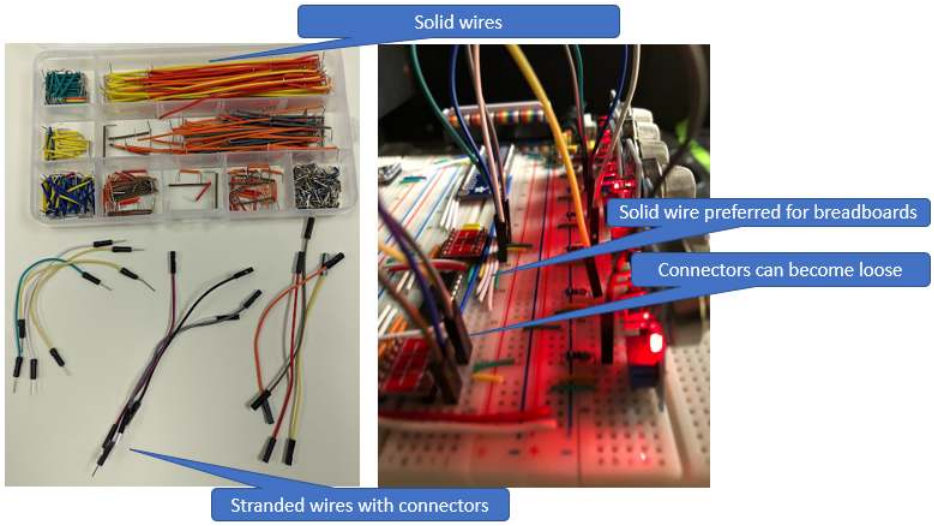


Figure 11.6.3: Examples of common solid and stranded wires used in bread boarding. (Belford CC-BY)

Tutorials

- SparkFun: Working with Wire

This page titled 11.6: Conductors is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

11.7: Circuit Boards

Circuit boards allow you to connect a variety of electronic components to a circuit.

Breadboards

- Good for prototyping
    - 5V or lower voltage
- Bus bars are common connectors that are used to connect devices to a circuit

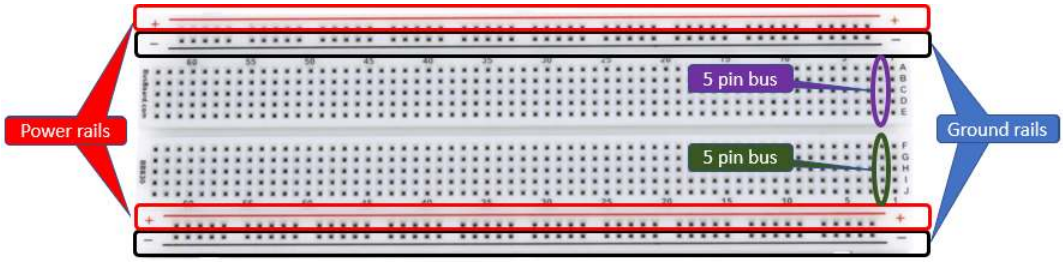


Figure 11.7.1: Common prototyping breadboard. (Belford CC-BY)

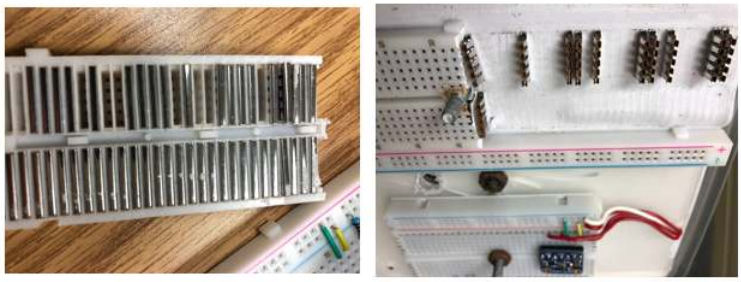


Figure 11.7.2: Damaged rails on a bread board. (Belford CC-BY)



Video 11.7.1: Science Buddies video on use of Breadboards (https://youtu.be/6WReFkfrUIk)

Note, the power and ground bus on some breadboards stop halfway across the board. This allows different circuits with different voltages to be run on the same board. If the red or black lines stop half way across the board, you will need to run jumper wires to connect them if you want to have a common ground or power supply.

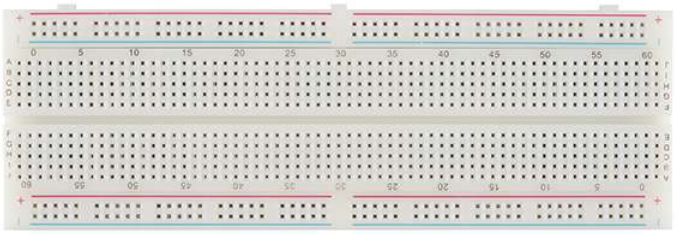


Figure 11.7.1: Split rail breadboard. (Sparkfun CC BY-SA 4.0)

Breadboard Tutorials

The following are excellent tutorials and this list will increase over time

- AdaFruit Tutorial
- SparkFun Tutorial



Collin's Lab and the Real Breadboard (https://youtu.be/HrG98HJ3Z6w?si=ULLUylKepCdw3HS1)

Printed Circuit Boards (PCB)

Printed Circuit Boards are ubiquitous in electronic devices.

- Good for mass production
- Good for miniaturization
- Beware of ESD damage (Electro-Static Discharge)
- Minimize issues of lose wires

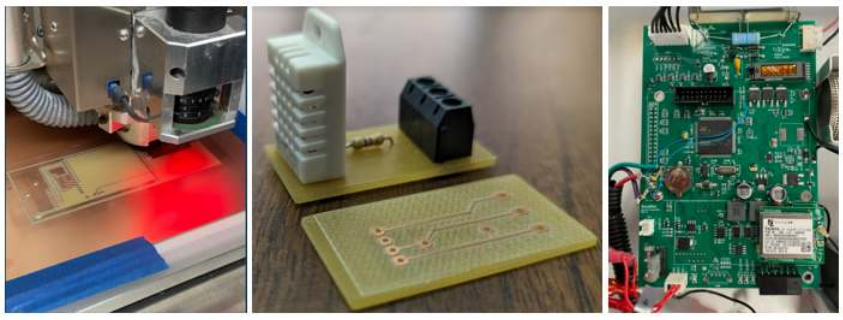


Figure 11.7.3: LPKF Laser & Electronics ProtoMat S62 circuit board machine (left), simple DHT22 circuit board (center) & Solarid smart insect zapper circuit board (right) (Belford & Bouzid, CC-BY)

In this class we will design several circuits using breadboards and then create simple circuit boards to solder the components to. Breadboards are great for prototyping, but their connections are not reliable enough to be deployed in real world applications.

Tutorials

- SparkFun: PCB Basics

This page titled 11.7: Circuit Boards is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

11.8: Resistors

Introduction

- Function as a load
    - convert electric energy to heat
    - P = IV = rate of energy consumption
    - Used in resistance heaters
    - Used to control current
    - Do Not Overload!

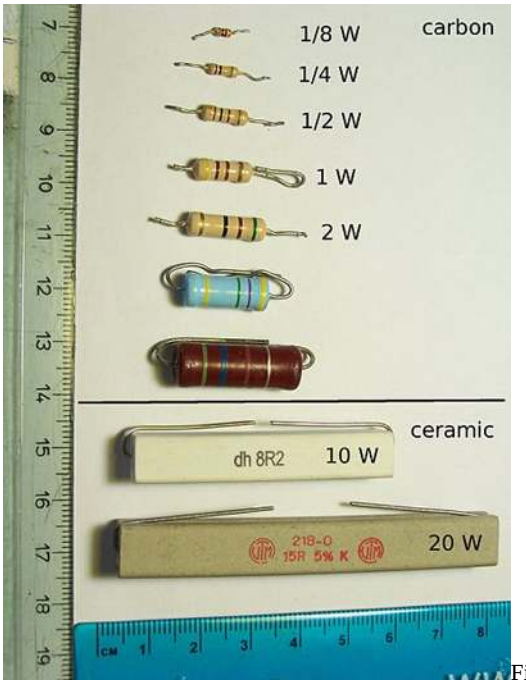


Figure 11.8.1: Common resistors. (David Ludovino, CC 3.0, Wikimedia commons)

What happens when to much current passes through a resistor?



Video 11.8.1: 2:48 min YouTube by Tariq Lab showing what happens if you send too much current through a resistor by hooking it up to too high a voltage (https://youtu.be/M_3QyLOW9Zs?si=-BaxqiSfD2sqqH6u)

Resistor Values

Use bands to describe resistors. There are four types of bands

1. Digits
2. Multiplier
3. Tolerance
4. Temperature Coefficient

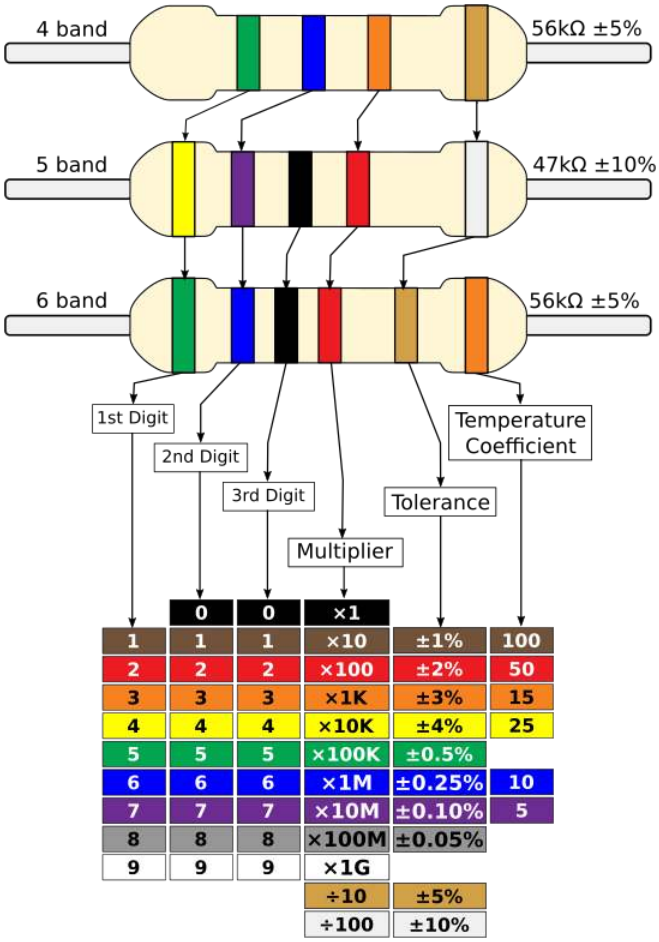


Figure 11.8.2: Resistor Color Chart (4,5 & 6 band). (Koudstall; Open clipart, CC 1.0)

Variable Resistors

There are two basic types of variable resistors, Rheostats and Potentiometers. Rheostats control current and potentiometers can control voltage or current.

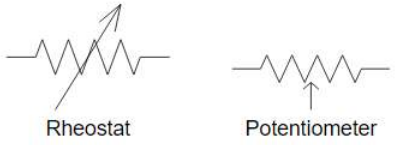


Figure 11.8.3: Circuit diagram symbols for Rheostat and Potentiometer. (Belford, CC 0.0)



Video 11.8.2: 1:18 min YouTube by Calco Tech Tips on the Difference Between Potentiometers and Rheostats (https://youtu.be/qCvnN2F_bRg?si=XnMty1XqjMr2cIxL)

Rheostat

A Rheostat has two terminals and is used to regulate current and can be used for high current applications.



Video 11.8.3: 4:35 min YouTube by DERF Electronics on How a Rheostat works - Step by Step & its different applications (https://www.youtube.com/watch?v=isA5I9JFjA4), This video comes from their tutorial on An Overview on Rheostats.

Potentiometer

A potentiometer has three terminals can be used to regulate either current or voltage, but is not designed for high current flows. Potentiometers are often integrated into circuit boards. They are effectively adjustable voltage dividers

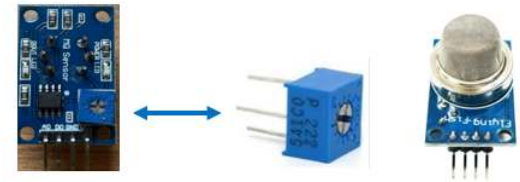
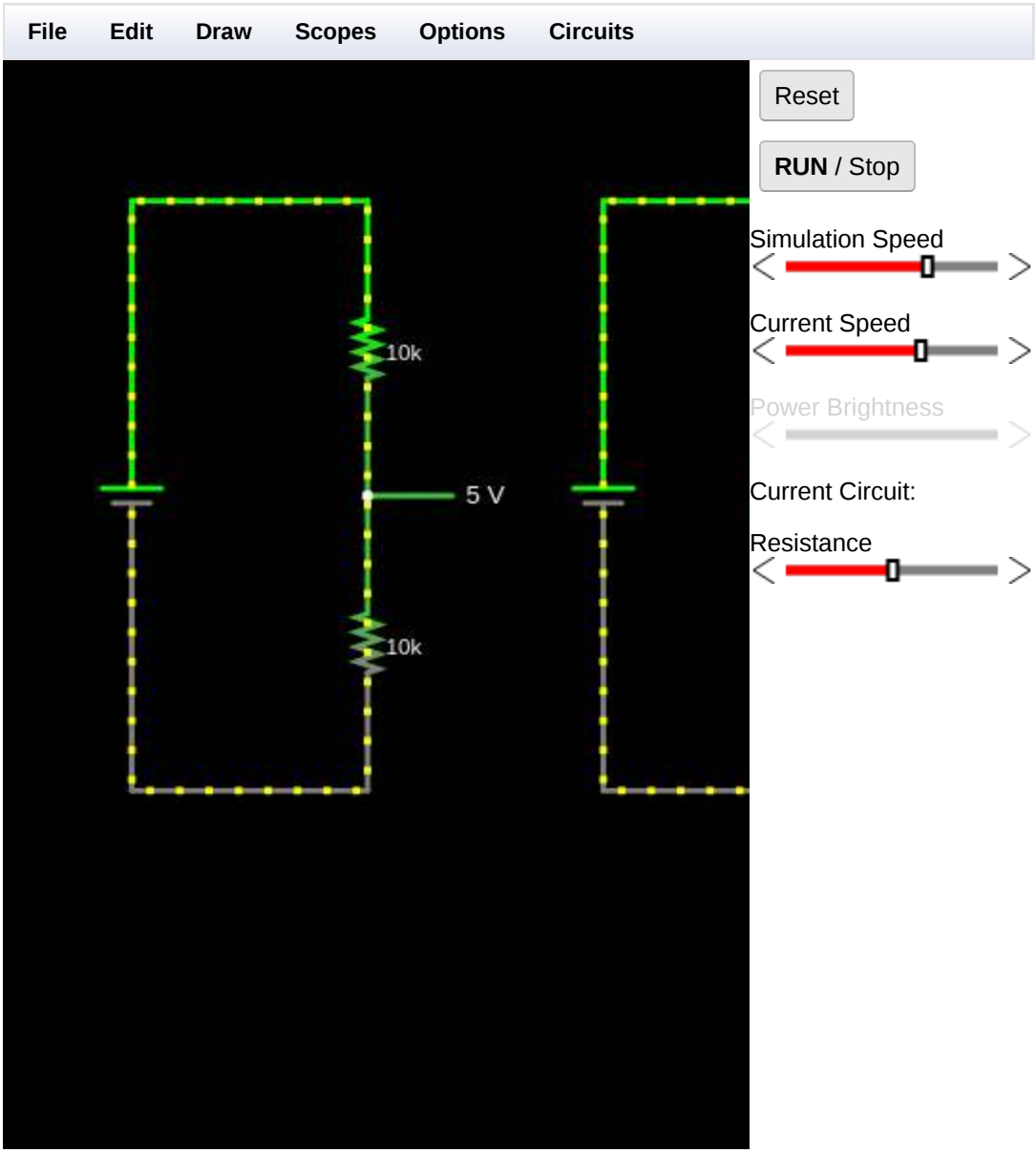


Figure 11.8.4: Copy and Paste Caption here. (Copyright; author via source)

Here the potentiostat can be used to trigger an alarm if a gas concentration reaches a set concentration by adjusting the voltage at which a digital signal is sent.



Video 11.8.4 1:54 min YouTube on Potentiometers by Mouser Electronics (https://youtu.be/oZbSvG7rUAA?si=-t8esOuFKY9okIbv).



The circuit on the left is a voltage divider and on the right is a potentiostat (variable voltage divider). If you hoover over the power supply you will see it is 10 V, and when the simulation loads both circuits have 10K on either side of the divider. Now if you move the resistance on the left side with the slider in the panel, you can see how the potentiometer can alter the current at the divider

https://tinyurl.com/29yp5ffa

Tutorials

- SparkFun: Voltage Dividers

This page titled 11.8: Resistors is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

11.9: Diodes

Diode Structure

Diodes involves the junction of two semiconductor type materials, with one side having extra electrons (N) and the other extra "holes" (P). These material can be made by doping elements like Silicon and Germanium (that typically have 4 valence with electrons with similar sized elements that have 3 or 5 valence electrons, with the former creating the P layer and the later the N layer.

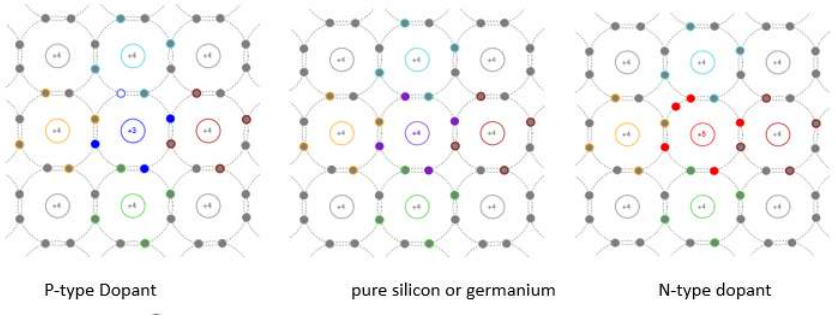


Figure 11.9.1: P and N type dopants. (James Fiore, cc 4.0 LibreTexts)

Diode Bias

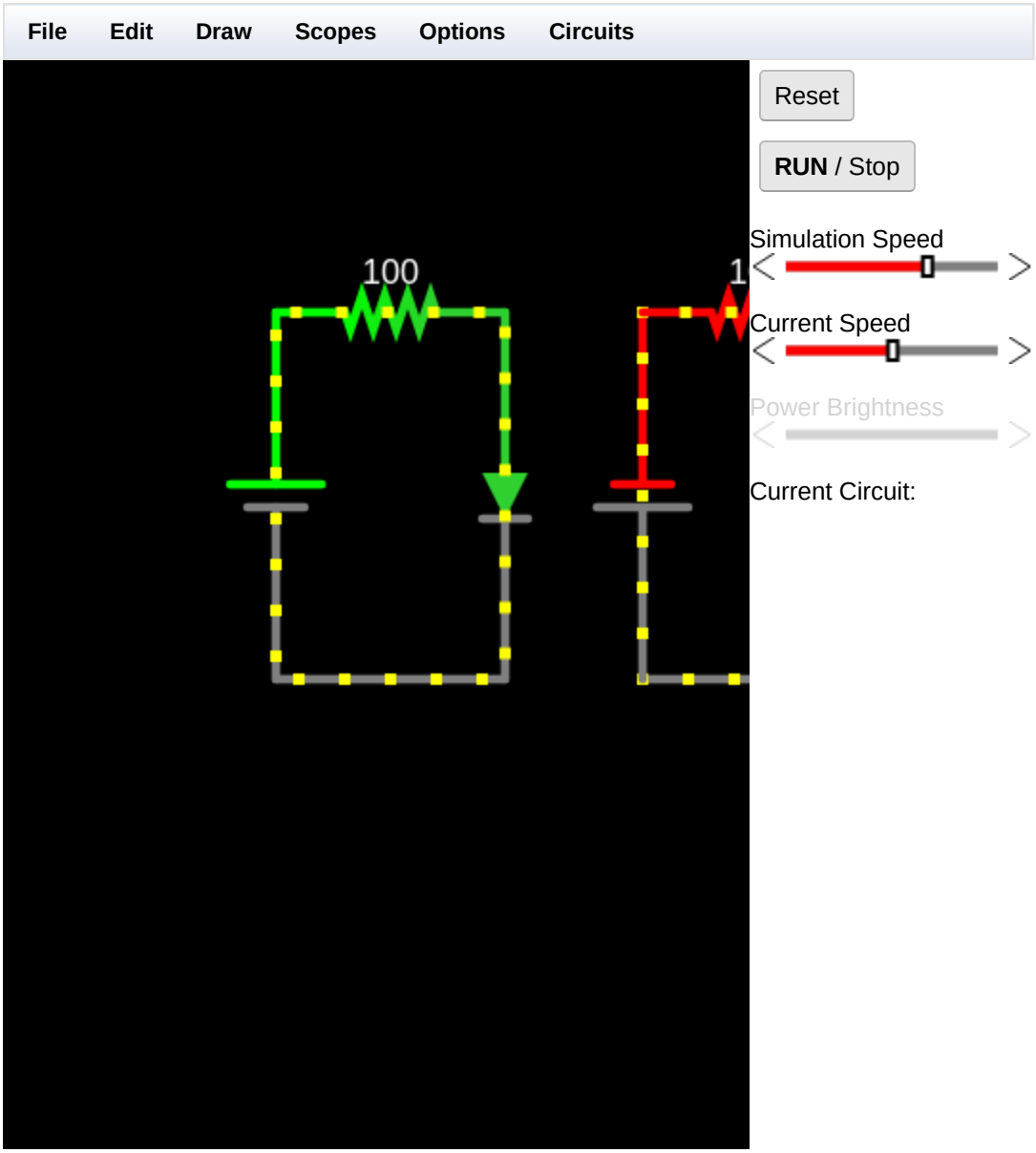
This results in the diode having two polarized terminals with the Anode being the positive terminal and the cathode being the negative terminal.



Figure 11.9.2: Diode symbol showing positive anode and negative cathode. (Belford; CC 0.0)

This means that under normal conditions the diode only allows current to flow in one direction, which is from the anode to the cathode.

- Forward Bias: occurs when a positive voltage source is applied to the anode
- Negative Bias: occurs when a negative voltage source is applied to the anode
- Breakdown Voltage: At some voltage the diode will conduct current in the reverse direction and this results in a breakdown of the diodes normal operating behavior.



Falstad Cicuit JS simulation 11.9.1: Forward (left) and reverse (right) biased diodes. Note the diode stops the reverse flow and so is unidirectional.

This page titled 11.9: Diodes is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

11.10: LED (Light Emitting Diodes)

Most forward biased diodes give off heat when electrons "fall" into the lower energy hole of the PN junction as a result of the conservation of energy. In the case of an LED light is emitted, and this light is of a specific energy (frequency) that is related to the bandgap energy of the PN junction. So LEDs can be of specific wavelengths

Single Color LEDs

Since LEDs are diodes they are polar and must be hooked up correctly. Figure \(PageIndex1 shows a typical LED and you can note that one lead is longer than the other. The longer lead is the Anode, which you hook to the positive terminal, and the shorter lead is the cathode, which you hook to the negative terminal

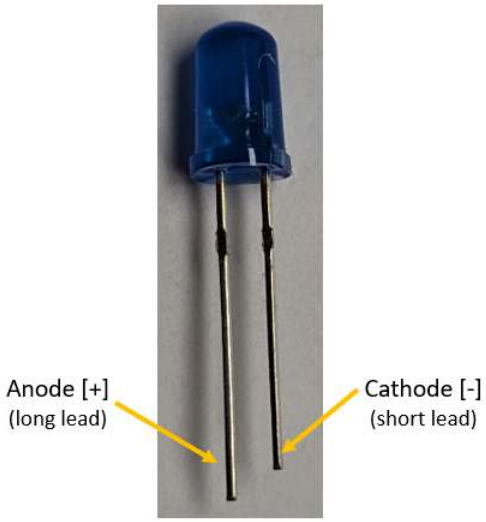


Figure 11.10.1: Copy and Paste Caption here. (Copyright; author via source)

the electronic symbol of an LED is that of a diode with two arrows leaving, to indicate that light is being emitted

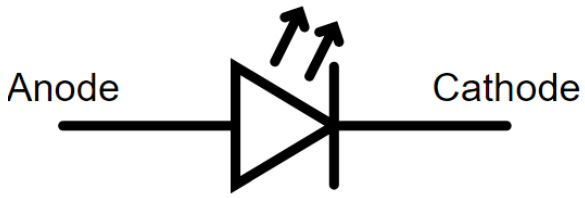


Figure 11.10.1: Copy and Paste Caption here. (Copyright; author via source)

The following shows a Fritzing of a circuit set up on a breadboard for and LED diode. Note, the long lead is indicated by a bend, so they can line up with the breadboard pin holes. In this circuit it does not matter where the resister is (either before or after the diode).

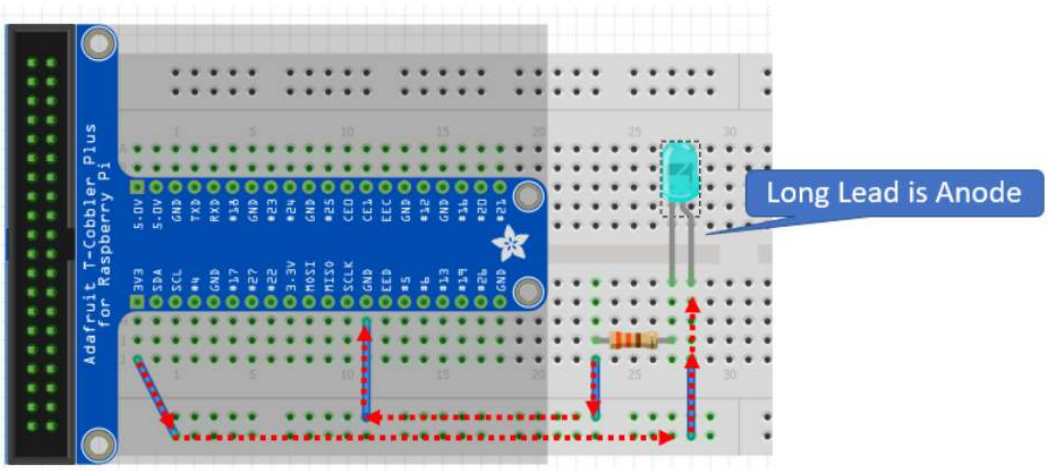
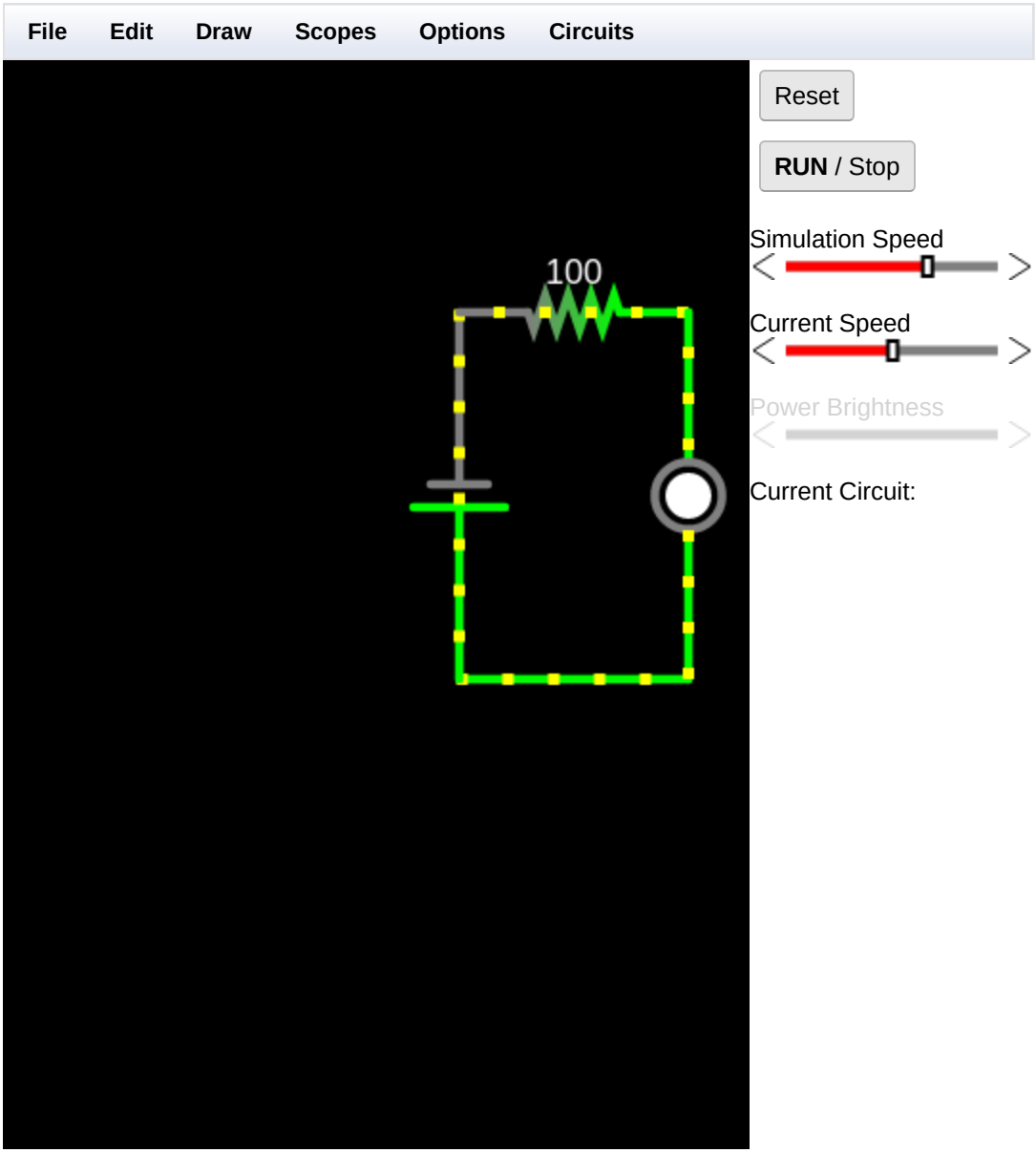


Figure 11.10.1: Copy and Paste Caption here. (Copyright; author via source)

RGB LED

Red Green and Blue can be combined to form different colors and you should go to this RGB calculator at W3 Schools. to see the different colors that can be created by mixing various quantities of Red, Green and Blue. In a RGB LED you have three close LEDS (RG&B) and by varying the amount of current through each you can change the output colore. The following Fritzing has a GRB LED that is set to 1 by default (values vary from 0 to 1), and you can adjust each color by double clicking on it and choosting edit component. Try changing the green to 0.5 and the blue to 0.2, or make any two of them zero and the third 1. I coding 8 bits are often used to define the values, and so you can have values from 0 to 255.



Common Cathode and Andoe RGB LEDs

An RGB diode has 4 leads, and there are two types.

- Common anode you hook the long lead to the hot,
- Common cathode you hook the long lead to the ground.

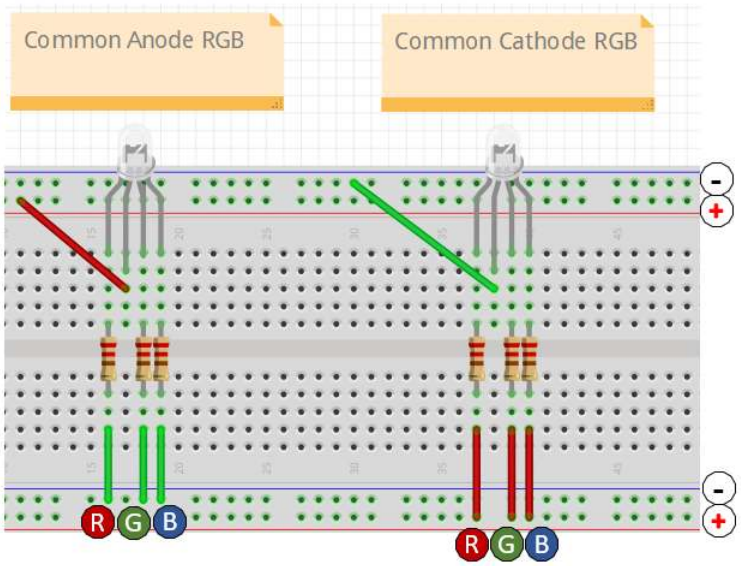


Figure 11.10.1: Copy and Paste Caption here. (Copyright; author via source)



This page titled 11.10: LED (Light Emitting Diodes) is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

11.11: Multimeter

The multimeter is an essential tool for debugging electric circuits.

Parts of a Multimeter

Measuring Voltage

The voltmeter is placed in parallel with the circuit component you wish to measure the voltage drop

Plug the ground probe into the common port and the hot probe into the voltage port



SparkFun YouTube Video at 2:40 minutes showing how to use multimeter to test for voltage.

Measuring Resistance

To measure resistance you disconnect the device you are measuring from everything else (make sure to disconnect the power), and then connect the probes across the device. The multimeter will then send a small current through the circuit and measure the resistance. Note the different ranges have different SI prefixes.

- Reading of 1 indicates the resistance is beyond the range of current setting and you need to increase the range.
- Reading 0f 0.01 indicates the resistance is too small and you need to decrease the range.
- Note, if you are measuring across a polar device like a diode or LED you must send the current in the correct direction (switch the direction of the probes).



SparkFun YouTube Video at 4:59 minutes showing how to use multimeter to test for resistance

Measuring Current

To measure current you need to set the multimeter in series with the circuit you wish to measure. Most multimeters have a seperate port for measuring current. Note, you may need an oscilloscope to see fluctuating current due to capacitors, inductors and other electrical components.



SparkFun YouTube Video at 6.32 minutes showing how to use multimeter to test for current

Applications of Multimeters

Testing a Battery



0:59 min YouTube by AdaFruit Industries: Test Batteries with a Multimeter - Collin's Lab Notes

Testing a Fuse



0:57 min YouTube by AdaFruit Industries: "Blown a Fuse?-Collin's Lab Notes"

Resources

- Sceince Buddies: How to Use a Multimeter
- SparkFun: How to Use a Multimeter

This page titled 11.11: Multimeter is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

11.12: GPIO Output (LEDs) - First Circuit

Learning Objectives

Students will be able to:

Content:

- Access the GPIO on the Raspberry Pi
- Build simple circuits
- turn an LED on and off with GPIOzero

Process:

- write code that turns on LED lights

Prior Knowledge

- Python and IoT concepts from previous activities

Further Reading

- https://gpiozero.readthedocs.io/en/stable/

GPIO Pins

General Purpose Input/Output on the Raspberry Pi:

Caution

Know what you are doing before you connect any leads to a pin. If you directly connect a hot (3.3V or 5V) to a ground you could destroy your board. If you have doubt, look it up on the web. **Always turn off the Pi when working on circuit boards**

Your raspberry pi has a set of bi-directional input/output pins that allow you to interact with the physical world. There are two ways to address the numbers: Broadcom (BCM) pin numbering and board numbering. Board numbering goes from 1-40 corresponding to the pins on the device. Any pin marked GPIO can be used for input and output. Some pins are reserved as power outputs, electrical grounds, or for other purposes.

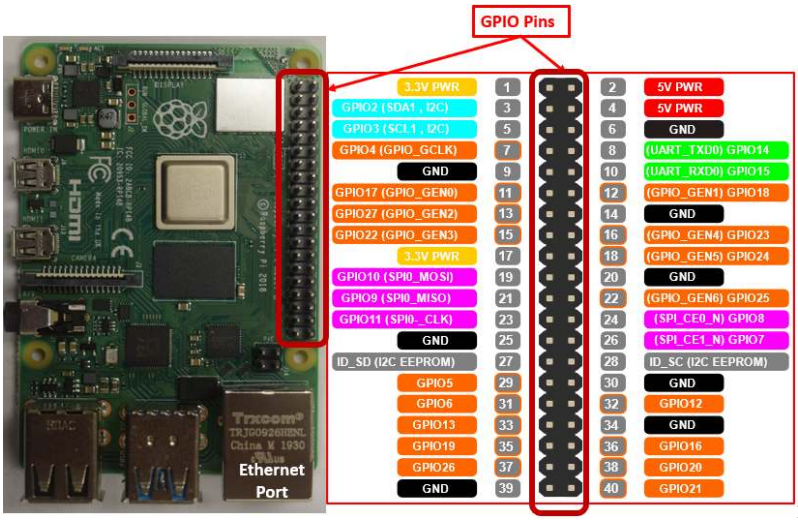


Figure 11.12.1: Raspberry Pi 4 with pin assignments. (Pi Belford, GPIO Diagram FreeSVG CC 0.0)

Pins

The board pins are in grey with the one position farthest away from the ethernet/USB 2.0/3.0 ports. We will be using the GPIO zero module and it is easiest to assign a pin by its GPIO number, which is not its board position number

- 8 Ground Pins (GND pins)
  - If you have multiple circuits make the ground common.
  - If you add new device like a sensor, make its ground common
- 4 Power Pins
  - 2 3.3V pins
  - 2 5V pins

- Used to power small devices like sensors
- You may need to add an external power supply.
- **Never connect power pins to ground pin**

- Reserved Pins
  - 27,28 - used for HATs (Hardware attached on top)
- 12 GPIO Pins
  - General Purpose Input Output
  - All use 3.3 V
    - **Never connect a 5V source to a GPIO pin**
  - Have two states
    - High (3.3V)
    - Low (0V)
- Alternate Functions (Communication Protocols)
  - UART (Universal Asynchronous Receiver-Transmitter)
  - I2C (Inter-Integrated Circuit)
  - SPI (Serial Peripheral Interface)

Critical Thinking Questions:

1. What are the board numbers for GPIO pins that are closest to the USB ports on your Raspberry Pi?
2. Provide the board pin number(s) for each of the following:
   a. Which board pins (numbers) provide 5V of electrical power potential?
   b. Which board pins provide 3V of electrical power potential?
   c. Which board pins are electrical ground?
   d. How many GPIO pins are available on your Raspberry Pi?
3. What colors are used to represent
   a. Ground
   b. 5V
   c. 3V
   d. Input/output
4. You have connected a device to board number 11. What GPIO number will you refer to for programming purposes?

> **Note**
> Whenever connecting/disconnecting items from your Raspberry Pi, you should always power off first to protect the Pi computer

Task 1: Setting up Pi and Cobbler

Setting up your Raspberry Pi to build circuits requires us to use prototyping board called a breadboard. Connect your cobbler to the 40 pin ribbon connector. Make sure your pins are not bent, and that the cobbler starts at line 1 of the breadboard.

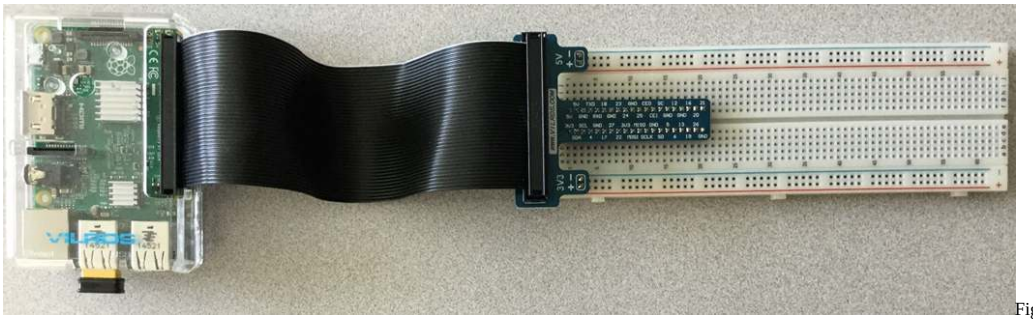


Figure 11.12.2: Breadboard connected to GPOI pins via ribbon and cobbler. Be sure to POWER OFF before connecting devices to your Pi!

The cobbler is a pcb board that you connect the ribbon two and algins the gpio pins down the length of the breadboard.

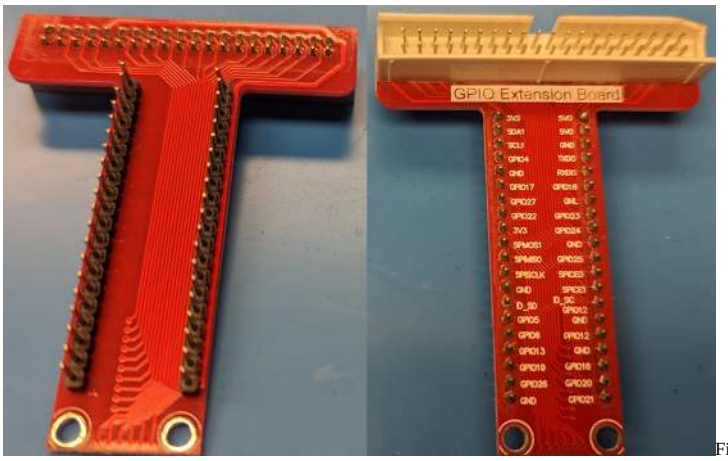


Figure 11.12.3: Close-up image of both sides of Cobbler. Left sides shows pins that are placed into breadboard and the right where the ribbon cable is attached. (Belford CC-BY)

Note the grove on the top of the cobbler on the right side of picture 11.12.3 The ribbon has a tab that fits into this groove and so the ribbon can only be attached one way.

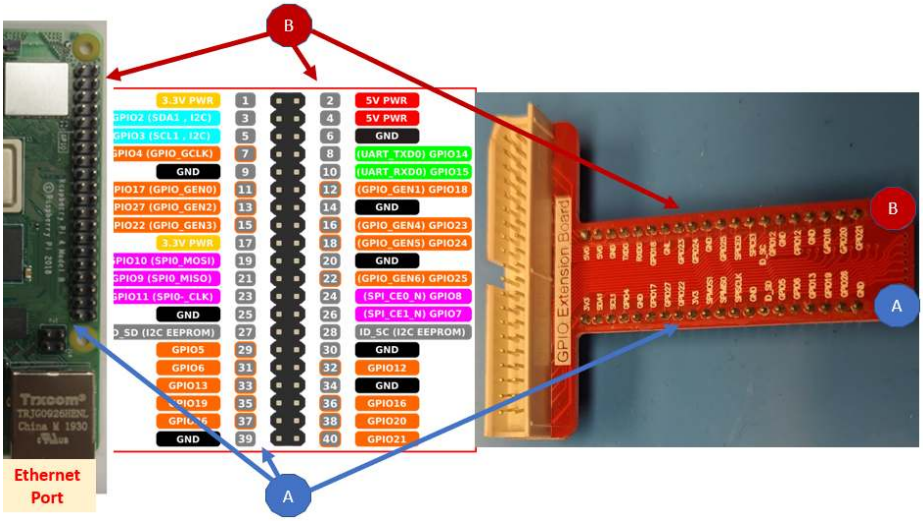
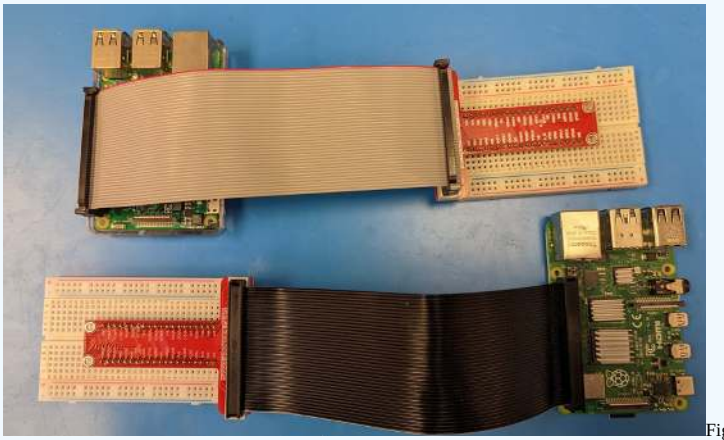


Figure 11.12.4: This image shows how the cobbler takes the left row og GPIO pins labled A and places them on the bottom bus bars of the breadboard, while taking the right tow and placing then on the top row of bus bars. (Copyright; author via source)

? Exercise 11.12.1

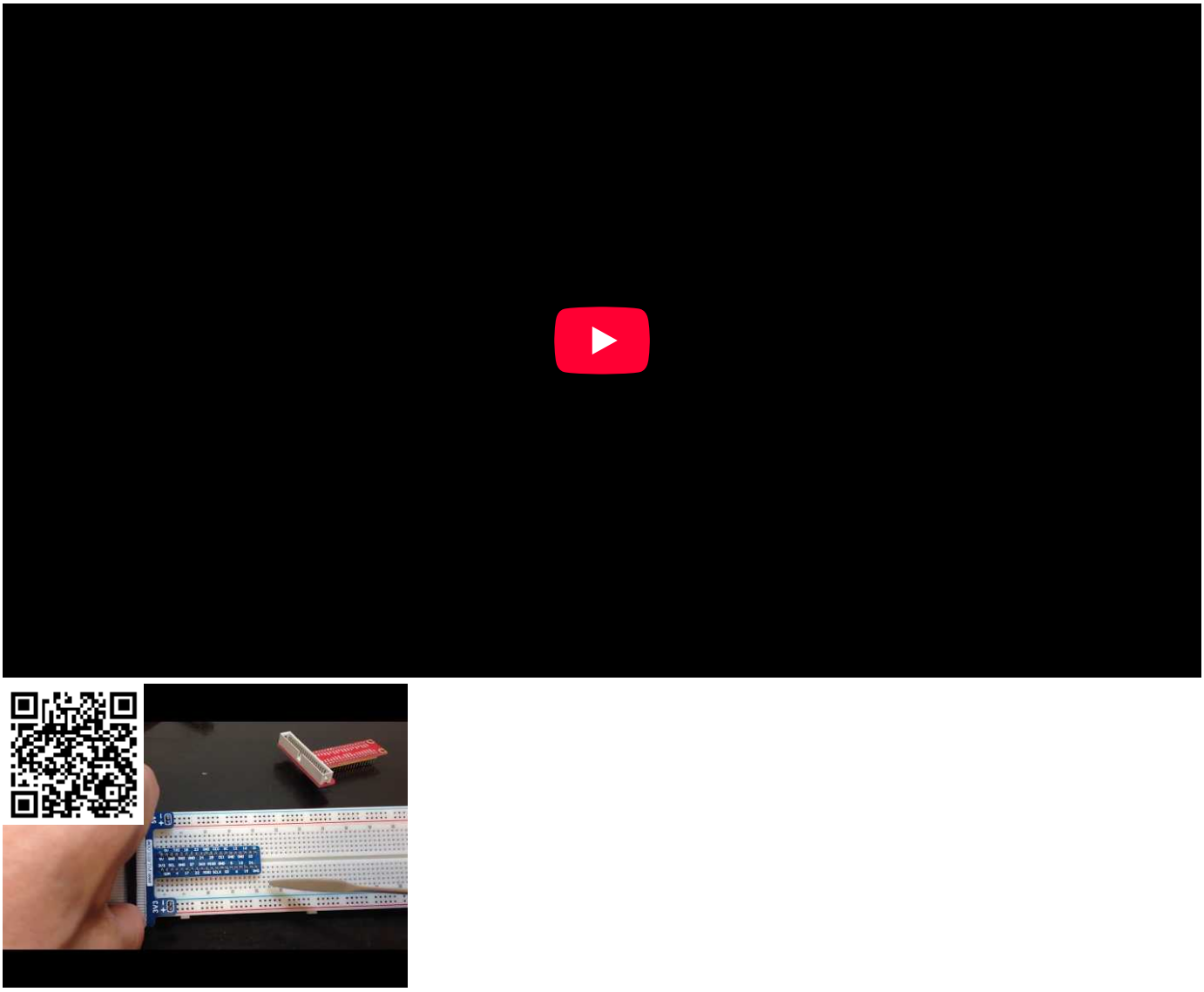
One of these two Pis is set up wrong and the pins are inverted. Which one is correct?


Figure 11.12.5: Correct and incorrect ribbon attachments. (Belford CC-BY)

Answer

The bottom image is correct, and the top one has the rows switched.

If this is the first time setting up a breadboard you should watch the following video.



Companion video available here: https://youtu.be/Ljd3dpfa1JQ

The cobbler lets us connect the breadboard easily to the raspberry Pi. At this point, boot up your Raspberry Pi.

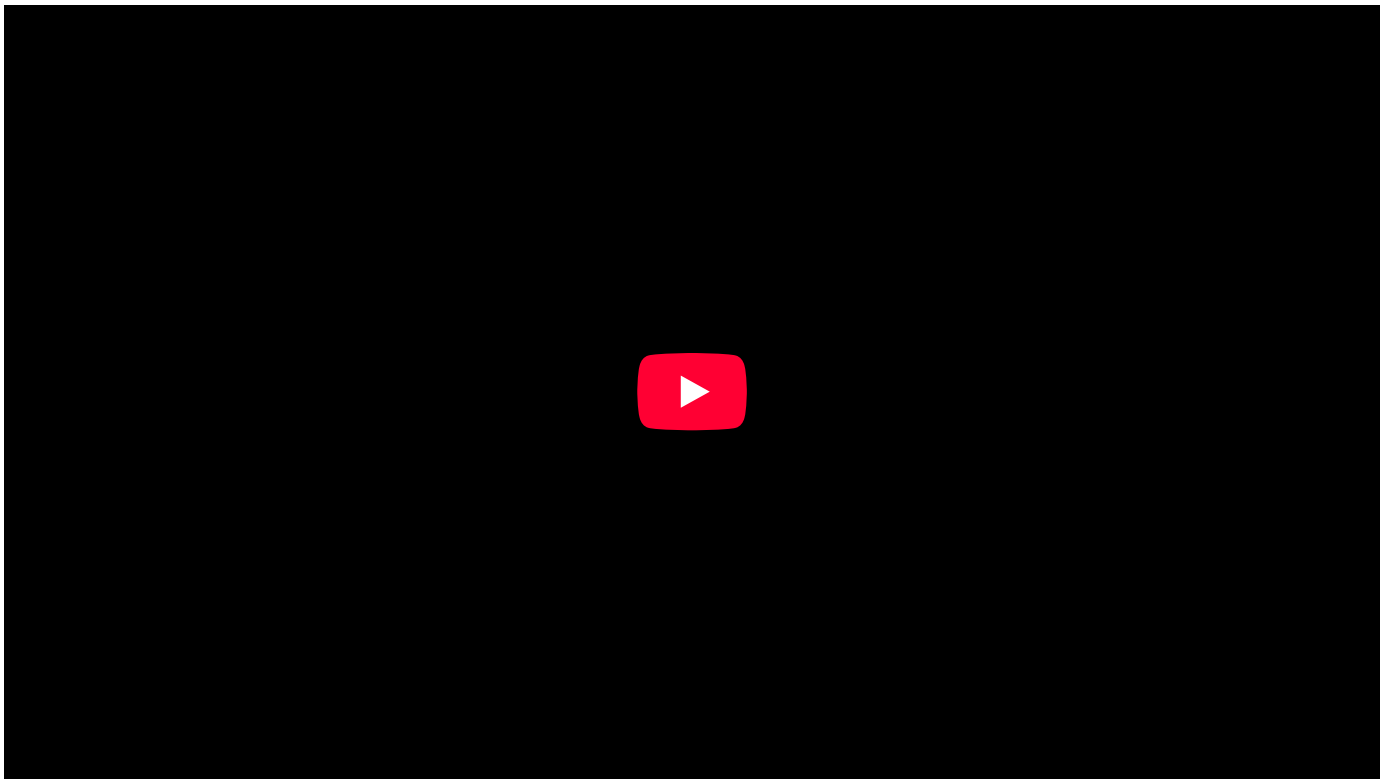
Note

There are two common types of cobblers, those with power pins (like the Vilros Cobbler) and those without power pins (like the AdaFruit Cobbler)

- **Vilros Type cobbler** that has 3.3V and 5V pins for the outer power rails
- **Adafruit Type cobbler**, which does not have these pins built into it, and so you need to run a line to the power rails

Task 2: Building your First Circuit

In this activity you will connect a red LED to a circuit on the bread board with either a 220 or 330 ohm resistor. The resistor is needed to reduce the current so you do not burn out the LED (V=IR). Before building your first circuit watch the following YouTube. Then follow the procedures below for a Vilros Cobbler (part A) or an AdaFruit cobbler (part B), depending on which type of cobbler you have. Never connect a lead across an LED without a resistor. It does not matter what side of the LED the resistor is on, as long as it is in series with the LED.



Companion video available here: https://youtu.be/R-u0DTWCcXk

Your diagram should look like that in figure 11.12.2

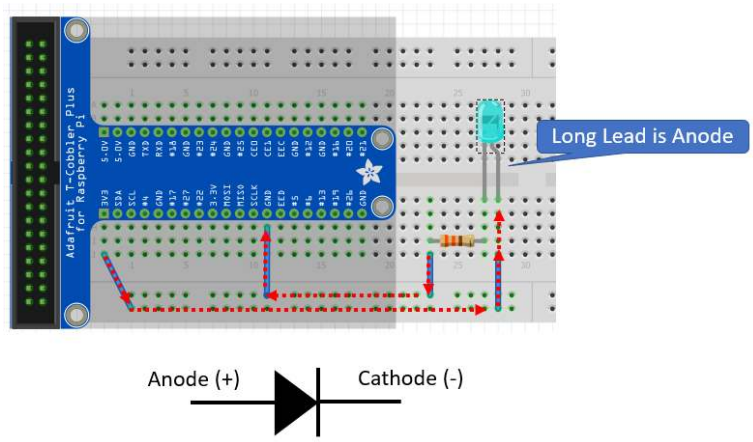


Figure 11.12.6: Basic closed loop circuit for LED. Red lines show the path of the current (Belford CC-BY)

A. Vilros Type Cobler

1. Start by placing one end of a black male-to-male jumper cable in the negative (ground) line connected to 3v3 of your vilros cobbler. Connect other end of the jumpber to slot 30a.
2. Slightly bend a 330 ohm resistor to make a U shape. Put one end in the 30b slot, and the other end in the 34b slot.
3. Take a red LED and connect the short end (cathode) to slot 34c. Connect the long end (anode) to slot 36c.
4. Place one end of a red male-to-male jumper cable in slot 36a.
5. Place the other end of the red jumper cable in the 3v3 + line.

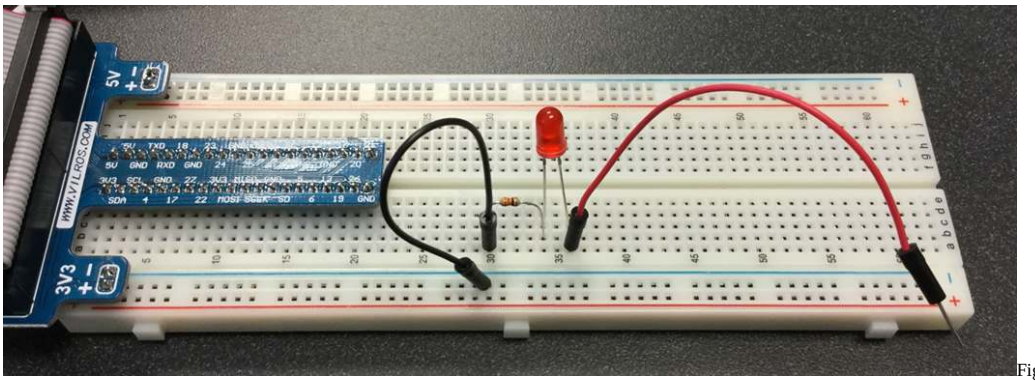


Figure 11.12.7: Vilros type cobbler with power pins

B. AdaFruit Type Cobbler

1. Start by placing one end of black male-to-male jumper cable in the negative (ground) line and connect it to a GND on the Adafruit Cobbler.
2. Place one end of a red male-to-male jumper cable to the positive (+) line and connect it to 3V3 (Pin1) on the cobbler.
3. Place one end of a black male-to-male jumper cable in the negative (ground) line connected to 3v3 of your cobbler. Connect other end of the jumper to slot 30a.
4. Slightly bend a 330 ohm resistor to make a U shape. Put one end in the 30b slot, and the other end in the 34b slot.
5. Take a red LED and connect the short end (cathod to slot 34c. Connect the long end (anode) to slot 36c.
6. Place one end of a red male-to-male jumper cable in slot 36a.
7. Place the other end of the red jumper cable in the 3v3+ line.

Figure 11.12.8: Adafruit type cobbler where you have to connect the power rails, here we are connecting the 3.3V rails.

> **Note**
>
> For all subsequent builds with the Adafruit type cobbler you will need to hook up the power lines as in step 1 & 2. For the rest of the activities in this lesson you need to keep the cables to the 3.3 V pin connected, although in some subsequent activities you will be running power through the 5 V pin.

Important! If you don't have a resistor in this circuit, you will send too much electricity to the LED. It will light up once briefly, and then not work again (much like a blown fuse or light bulb)!

Once you have built your circuit answer the following Critical Thinking Questions.

Task 3: GPIOZero Programs

> **Note**
>
> Included in the Raspbian OS is a library for interfacing to many simple input and output components called GPIOzero. It also has functions for more complex items such as sensors.
>
> **Remember: To stop any program from running you can use <Ctrl +C> to stop it.**

We are now going to run the Raspberry Pi from a GPIO port. The code for this activity comes from the GPIO zero documentation (https://gpiozero.readthedocs.io/en/stable/). As GPIO stands for General Purpose Input Output, a GPIO pin on an electric circuit can function as in output (hot) or input (ground) pin. Use should never hook a GPIO pin to a hot source without a load and you should only run 3.3 V circuits through GPIO pins. In the following Fritzing we are using the GPIO as an output pin, and running the current from the pin through

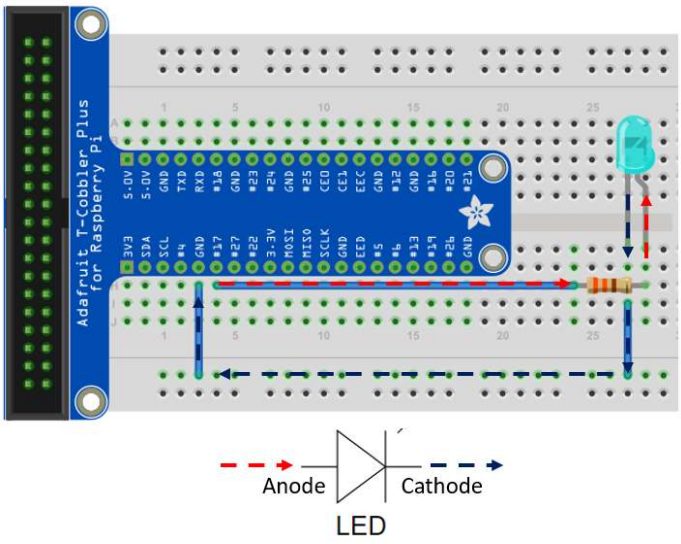


Figure 11.12.9: CGPIO pin 17 being run as an output. Always run a resister in series with an LED (Belford CC-BY)

After removing the hot line from the 3.3V pin and attaching it to the gpio17 pin, you whould run the following code, which comes from the gpiozero readthedocs.

gpiozero.led

IoCT Program 3.1

```
from gpiozero import LED
from time import sleep

red = LED(17)
try:
    while True:
        red.on()
        sleep(1)
        red.off()
        sleep(1)
except KeyboardInterrupt:
    red.off()
    quit()
```

Critical Thinking Questions:

- Change the value of each sleep line to 3. Rerun the program.
- Next to each line of code above, indicate what the statement does.
- Change your code to make a program that cycles your red light on and off 5 times.

Task 4: Add second LED Circuit

Connect a new yellow LED and resistor to create a new circuit connected to GPIO18.

Critical Thinking Questions:

- Alter your program to indicate a yellow LED is now on GPIO18, and it should blink in unison with the red light.
- Alter your program to have the yellow and red alternatively blink. That is when the red light is on, the yellow light is off.

gpiozero.pwmled

PWM stands for Pulse Width Modulation and the default frequency for this method is 100 Hz (see gpiozero.readthedocs.io) and the duty cycle is the fraction of time the signal is high. Because the frequency is faster than the eye can see (30 to 60 Hz, what we see is a reduction of intensity of the light proportional to the time it is off. The duty cycle is the fraction of time the signal is high.

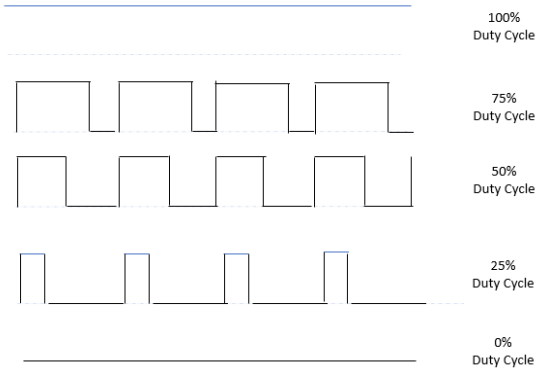


Figure 11.12.10: Various duty cycles for PWM. The default gpiozero.pwmled frequency is 100 Hz. The human eye can see 30 to 60 frames per second, so we do not see the square wavefunction, but a decrease in the light intensity. (Belford CC 0.0)

Enter the following program and run it.

IoCT Program 3.2

```
from gpiozero import PWMLED
from time import sleep

led = PWMLED(17)
try:
    while True:
        led.value = 0
        sleep(1)
        led.value = 0.5
        sleep(1)
        led.value = 1
        sleep(1)
except KeyboardInterrupt:
    led.value = 0
    quit()
```

Critical Thinking Questions:

16. Change the led.value to other numbers between 0 and 1 and rerun the code.
17. What does the PWMLED function do?

Got to READTHEDOCS and play:

https://gpiozero.readthedocs.io/en/stable/api_output.html#

Task 6: Common Anode RGB Circuit

RGB Colors

RGB stands for Red, Green and Blue, and various colors can be derived by mixing various intensities of red, green and blue. There are a variety of ways of describing these and for 8 bit coding there are values from 0 to 255 for each color, although some programs do these as fractions of 1, and so they are represented in decimals ranging from 0 to 1. The following is an RGB calculator from W3schools screen capture with values of rgb(255,255,0) that results in yellow.

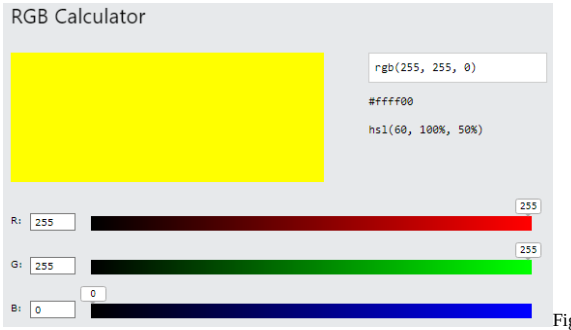


Figure 11.12.11:Screenshot of w3schools RGB Calculator. Note, yellow is a mixture of equal values for red and green. (Belford, cc 0.0))

Note, there are two types of RGB LEDs, common cathode and common Anode. the common is the long lead and in the common anode you hook the common to the high potential and in the common cathode you hook the common to the ground. If you look at the LED from the side where the long lead is second from the left, they run Red, Long Lead, Green and Blue. Be sure to hook up resisters and you probably want to start with the same resister value for each LED and see if the light is white. If it is not, play with the RGB picker such that you get the color you see, and then adjust the resistances in a manner that converts the color to white. Now that you have calibrate the RGB LED, you can pick a color by using the appropriate RGB values in your code.

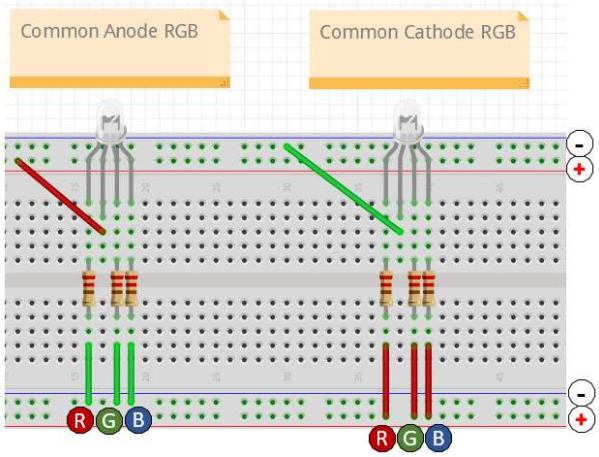


Figure 11.12.12: Common anode RGB LED connected with either a 220 or 330 ohm resistor.

? Exercise 11.12.1

Is the following RGB LED a common anode or cathode RGB LED? How could its wiring be improved?

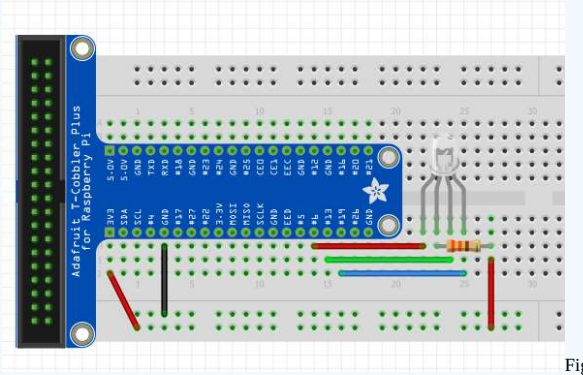


Figure 11.12.13: (Belford cc 0.0)

Answer

Common anode as the hot is wired to the common. There is one resistor to the anode, which will work, but if you set up a single resistor to each R,G & B lead, you could tweak the resistances so the color would be better controlled. That is, the internal efficiency of each LED could vary, and as set up, they are all getting the same voltage drop.

Note, the Fritzing in exercise 11.12.1has a resistor on the common and the intensity of light may be different for each color so that it will not be white when you set all RGB values in your code equal (some of the LEDs may be more efficient than others). This can be corrected by setting it up as in figure 11.12.11such where you choose resistors for each line that result in white light when they all have the same RGB values in the code.

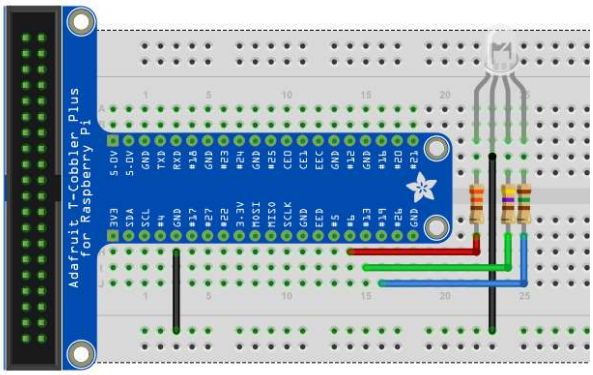


Figure 11.12.14: By placing different resisters on each of the RGB lines we can set it up so the light intensity for each is the same when the values are the same. (Belford CC-BY)

Build the Following Circuit, Remember, whenever connecting/disconnecting items from your Raspberry Pi, you should always power off first, to protect the Raspberry Pi.

Note

RGB (Red Green Blue) is an additive color model based on 8 bits, see, https://rgbcolorcode.com/. (R,G,B) values of (0,0,0) corresponds to black and (255,255,255) corresponds to white. It is not important that you use slots 44-47, but the 4 pins on the common anode RGB LED are short/long/short/short, and in that order represent R/Anode/G/B. You then need to connect the Anode to hot, and the R,G & B to the gpio pins as defined in your code. The first line of code in IOCT Program 3.3 states: led=REBLED(red=6, green=13, blue=19), and so we are connecting the red to gpio6, green to gpio 13 and blue to gpio19. You can use any GPIO pin you want, as long as you properly assign it in your code.

Enter the following program and run it.

IOCT Program 3.3

```
from gpiozero import RGBLED
from time import sleep

led= RGBLED(red= 6, green = 13, blue = 19, active_high = False)
led.red = 0
led.green = 0
led.blue = 0
led.red = 1
sleep(1)
led.red = 0
led.green = 1
sleep(1)
led.green = 0
led.blue = 1
sleep(1)
led.blue = 0
led.red = 0.1
sleep(1)
led.red = 0
led.green = 0.1
sleep(1)
led.green = 0
led.blue = 0.1
sleep(1)
led.red = 0
led.green = 0
led.blue = 0
sleep(1)
```

Critical Thinking Questions

- Next to each line of code above, indicate what that does.
- How does changing the values affect the program?

Enter the following program and run it, realize you need to use the GPIO pins you are using on your breadboard.

IoCT Program 3.4

```
from gpiozero import RGBLED
from time import sleep

led= RGBLED(red= 6, green = 13, blue = 19, active_high = False)
led.color= (0,0,0)
led.color= (1,0,0)
sleep(1)
led.color= (0,1,0)
sleep(1)
led.color= (0,0,1)
sleep(1)
led.color = (0.5,0,1)
sleep(1)
led.color= (0.5,1,0)
sleep(1)
led.color = (0, 0.7, 0.7)
sleep(1)
led.color = (0,0,0)
```

Critical Thinking Questions:

- Next to each led.color line in the code above, write the color the LED makes.
- Color mixed light in equal amounts of red and green should give a yellow light. Color mixed in equal amounts of red and blue should give a magenta/purple light. Alter the code above to have values in line 12 to have equal amounts of red and blue, and line 14 be equal amounts of red and green. Why did the original code not use equal values?
- Color mixed light in equal amounts of red, green, and blue should give a white light. Write a new program that will make the LED produce white light. Report your values to the class.

Now Go to Readthedocs.io and play with RGBLED class of GPIO zero

https://gpiozero.readthedocs.io/en/stable/api_output.html#rgbled

Acknowledgements

The material in this lesson plan was originally developed by Ehren Bucholtz of the University of Health Sciences and Pharmacy in Saint Louis.

This page titled 11.12: GPIO Output (LEDs) - First Circuit is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

CHAPTER OVERVIEW

12: GPIO Input (Buttons) - Second Circuit

12.1: Second Circuit

This page titled 12: GPIO Input (Buttons) - Second Circuit is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

12.1: Second Circuit

Learning Objectives

Students will be able to:

Content:

- Access the GPIO using buttons on a breadboard
- Build simple circuits
- turn an LED on and off with buttons

Process

- write code that turns on LED lights with buttons

Prior Knowledge

- Python concepts from Python Activities 1-7, and IOT Activities 1-4

Further Reading

- https://gpiozero.readthedocs.io/en/stable/

Task 1: Building a Circuit with Buttons.

Build the following circuit containing a button and a red LED light. Connect the button to GPIO22, and the red LED to GPIO17. Don't forget the resistor for the LED or you will burn it out. Note the top image has the Vilros cobbler and the fritzing diagram has the other type. You don't need to add jumpers between the 3v3 and ground from Broadcom pins 1 and 9 if you are using the Vilros cobbler.

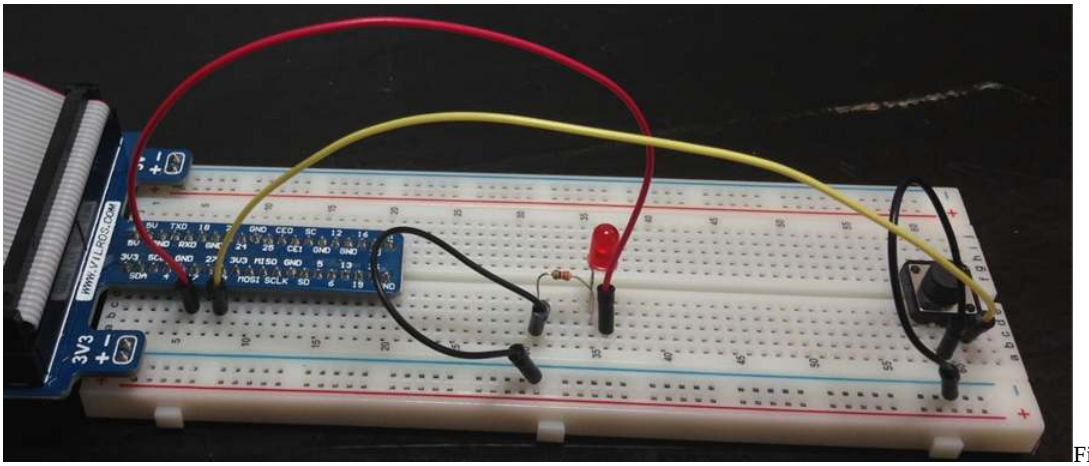


Figure 12.1.1: Image of first circuit with button

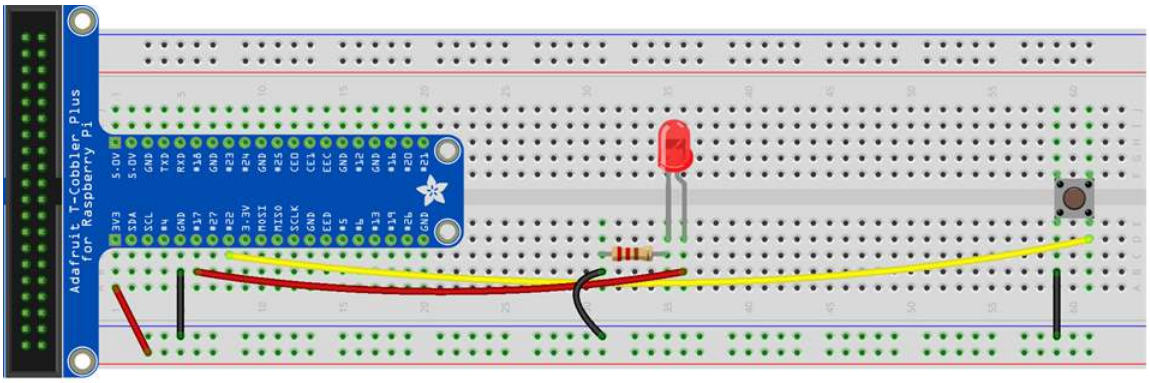


Figure 12.1.2: Fritzing diagram of first circuit with button

> **Note**
>
> GPIO 4 can be used for the button if you have 1-wire disabled in the interface of the RasPi configuration.

Task 1 Python Programs

Python Program 1

```
from gpiozero import Button, LED

red = LED(17)
button = Button(22)

while True:
    if button.is_pressed:
        print("Button is pressed")
        red.on()
    else:
        print("Button is not pressed")
        red.off()
```

Python Program 2

```
from gpiozero import Button, LED
from signal import pause

red = LED(17)
button = Button(22)

def buttonAction1():
    print("Button is pressed")
    red.on()

def buttonAction2():
    print("Button is released")
    red.off()

button.when_pressed = buttonAction1
button.when_released = buttonAction2

pause()
```

Critical Thinking Questions

1. Next to each line of code above, indicate what the statement does.
2. Which of the devices above are GPIO input devices and output devices?
3. Can an input device:
    a. trigger another GPIO event?
    b. trigger screen output?
    c. trigger a function?
4. Which three built in functions of the gpiozero.Button class are being used in the programs above?
5. How is the pause() function from signal different from the sleep() function from time?

Task 2: Three Button RGB LED Board

Add two more buttons and an RGB **common anode** LED to your breadboard. **Common anode** is the longest pin and needs to be connected to **3v3** with a 220 W resistor. Use GPIO pins 22, 19 and 26 for your buttons. Use GPIO pins 16, 20 and 21 for your RGB LED. Your board may not look exactly like these, but should have the same wiring connectivity regardless of holes on the breadboard. (see images next page

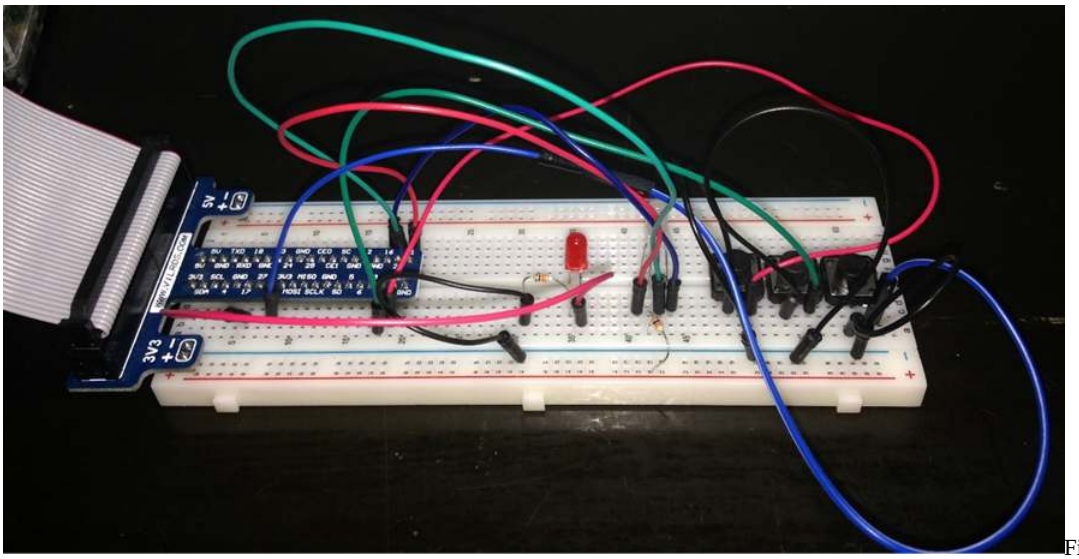


Figure 12.1.3: Three button board for RGB LED

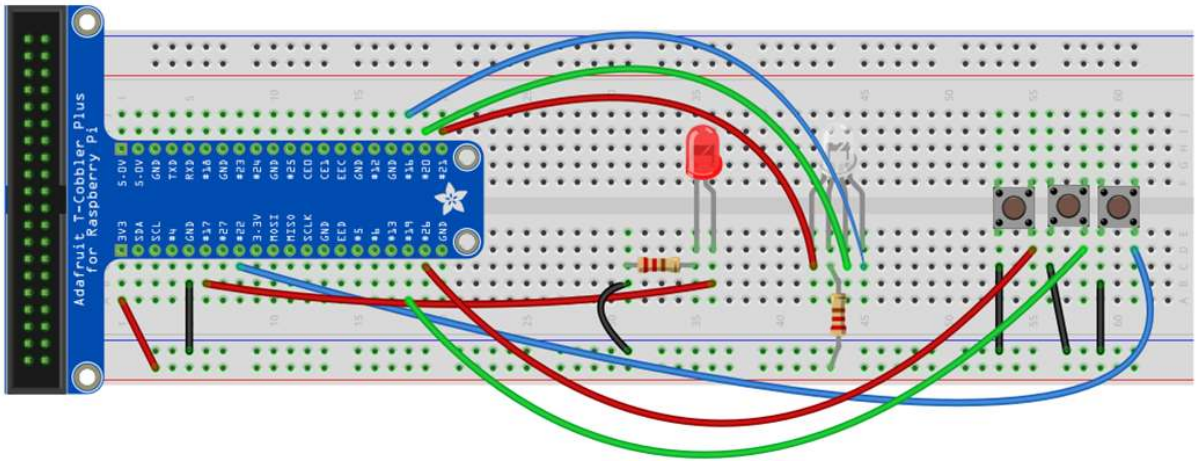


Figure 12.1.4: Fritzing diagram of three button board for RGB LED

Task 2 Python Programs

Python Program 3

```
from gpiozero import Button,RGBLED
from signal import pause

button_red = Button(26)
button_green = Button(19)
button_blue = Button(22)

led = RGBLED(red = 21, green=20, blue=16, active_high = False)

def buttonAction1():
    print("Red button is pressed")
    led.red = 0.3

def buttonAction2():
    print("Red button is released")
    led.red = 0

def buttonAction3():
    print("Green button is pressed")
    led.green = 0.7

def buttonAction4():
    print("Green Button is released")
    led.green = 0

def buttonAction5():
    print("Blue button is pressed")
    led.blue = 1

def buttonAction6():
    print("Blue button is released")
    led.blue = 0

led.color =(0,0,0)
button_red.when_pressed = buttonAction1
button_red.when_released = buttonAction2
button_green.when_pressed = buttonAction3
button_green.when_released = buttonAction4
button_blue.when_pressed = buttonAction5
button_blue.when_released = buttonAction6
pause()
```

Critical Thinking Questions (cont.)

6. What happens when you press and release a single button?
7. What happens when you click two or more buttons simultaneously?

Task 3: Making music

Create a folder called "Music" in the assignment folder, go to the class GitHub and download the file "soundfiles.zip". Once you have downloaded it you can open the file manager and drag into the Music folder as shown in figure 12.1.5

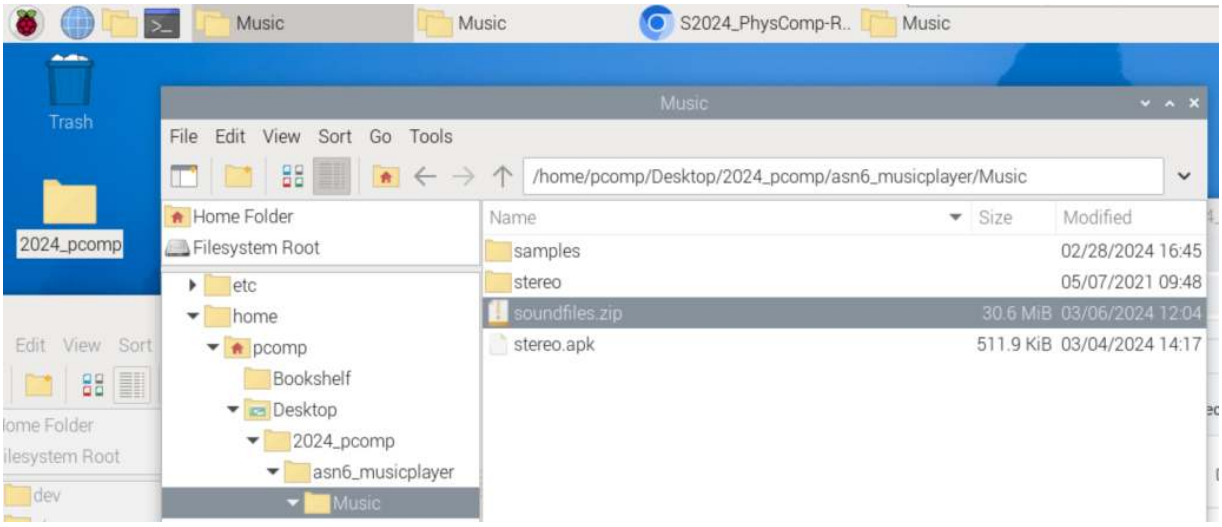


Figure 12.1.5: Copy and Paste Caption here. (Copyright; author via source)

Alternatively, you can go to the command line and use move command (section 3.5.2.3.2)

```
pcomp@raspberrypi:~ $ ls
Bookshelf  Documents  Music     Programs  Templates
Desktop    Downloads  Pictures  Public    Videos
pcomp@raspberrypi:~ $ cd Downloads/
pcomp@raspberrypi:~/Downloads $ ls
arduino-ide_2.3.1_Linux_64bit      sonic-pi_4.5.0_1_bookworm.arm64.deb
arduino-ide_2.3.1_Linux_64bit.zip  soundfiles.zip
flac.zip
pcomp@raspberrypi:~/Downloads $ mv soundfiles.zip /home/pcomp/Desktop/2024_pco
mp/asn6_musicplayer/Music
```

Figure 12.1.6: We first used cd to move to the download folder and ls to check the soundfiles.zip was downloaded. Then we used the mv command to move it tothe Music folder, and specified the full path in the process. (Belford cc 0.0)

to unzip from the command line you should move to the folder the zip file is in and use the following command.

```
unzip soundfiles.zip
```

or alternatively you can rightclick on the zip file within the filemanager and choose extract here

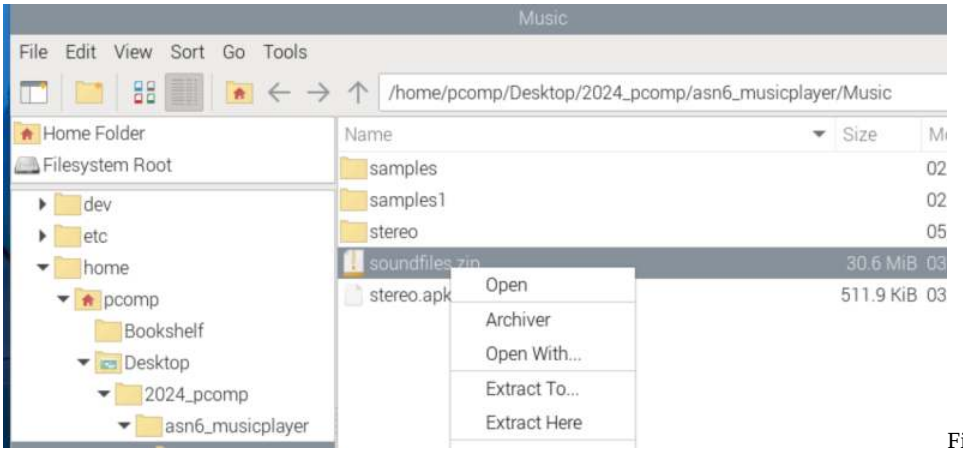


Figure 12.1.7: Copy and Paste Caption here. (Copyright; author via source)

Set up Audio Jacks

Now you need to set up your speakers. There are several options and we will use the audiojack. You can Right Click on the speaker icon in the upper right corner of the Pi desktop and choose Device Profiles and set the AV Jack option to "Analog Stereo Output" if you are using the audio jack.

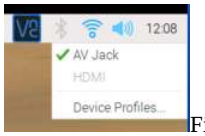


Figure 12.1.8: Copy and Paste Caption here. (Copyright; author via source)

If the icon is not present you can go to the command line type

```
sudo raspi-config
```

Choose

- option 1 (System Options)
- S2 Audio
- 0 Headphones

Connect an external set of speakers or your headphones to the Raspberry Pi, and you should be good to proceed to the next task. Now open up Thonny and proceed to the following python code. Save it in the Music folder (same folder that we copied the sound files to).

Note: the Raspi gives you many options and in figure 12.1.9 the Rasperry Pi is connect to a JBL Flip 2 speaker through BlueTooth (and not the AV jack).

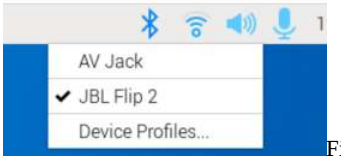


Figure 12.1.9: Copy and Paste Caption here. (Copyright; author via source)

Python code for playing sound files

Task 3c: Python code for playing sound files

```
import pygame.mixer
from pygame.mixer import Sound
from time import sleep

pygame.mixer.init()

sound_1 = Sound("complete/path-to-sound/file.flac")

while True:
    sound_1.play()
    print("Can you hear the sound?")
    sleep(3)
```

Remember, you can always find the complete path to a file in the file manager by right clicking the file and choosing "Copy Path(s)"

Acknowledgements

Ehren Bucholtz (created original content)

12.1: Second Circuit is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

CHAPTER OVERVIEW

13: Serial Communication

Topic hierarchy

13.1: Serial vs. Parallel
13.2: Serial and Morse Code
13.3: UART
13.4: I2C
13.5: SPI

13: Serial Communication is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

13.1: Serial vs. Parallel

Series vs. Parallel

There are two basic ways of setting up wired communication, in series or in parallel. Series communication lines transmit data one bit at a time, while parallel lines can simultaneously transmit multiple bits of data. You can look at the voltage as being on (high) or off (low) and by creating a "start" and "end" bit you can create a data frame that represents information similar to how computers store data in memory chips. Although parallel lines are faster than serial, they are more expensive and the close proximity of multiple lines to each other cause signal distortion because as the voltage rises and falls in one line it induces a magnetic field that induces a current in adjacent wires in the opposite direction (see Faraday's Law of Induction-Lenz's Law). Parallel lines are typically used where speed is important, like when the CPU is communicating with memory modules. Most wired IOT devices use series communication protocols as they are cheaper, and you can run them for greater distances without inducing signal lose.

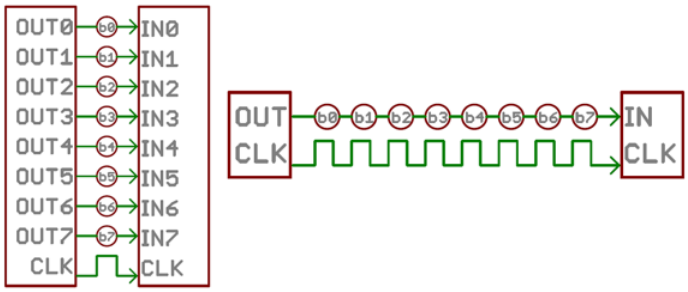


Figure 13.1.1: In parallel communication data is transferred multiple bits at a time, while in serial they are transffered one bit at a time. (cc 4.0 Sparkfun)

Serial vs. parallel communication

	Serial	Parallel
Speed	Slower, one bit is transferred at each clock pulse	faster, multiple bits (8,16,32..) are transferred at each clock pulse
Distance	Good for Long Distance	Good for Short Distance, May Suffer Signal Degradation Over Long Distance
Interference	Less Prone to Crosstalk	Prone to Crosstalk, when the signals of different wires induce flow in adjacent wires
Bandwidth	Limited	High Bandwidth Potential
Cost	Tends to be Cheaper	Tends to be More Expensive
Applications	sensors, peripherals (mice, keyboards)	Inside CPU and RAM
Issues	simpler circuits, speed	Needs More Wire, Crosstalk, Complicated, Clock Skew (bits may not transfer simultaneously)

Tutorials

- SparkFun: Serial Communication

13.1: Serial vs. Parallel is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

13.2: Serial and Morse Code

The earliest forms of serial communication were probably drumbeats and smoke signals. An early alphabetical system would be the Semaphore flag signaling system or optical telegraph that allowed ships to communicate. In the 1830's Samuel Morse and others invented the telegraph, which transmitted electrical signals over a wire, coding information in the forms of dots and dashes (short and long signals). Today information is coded in bits and we have already looked at multiple binary digital data representations like strings, integers and float. Although electronics have evolved since the days of the Morse Code, there is still a common binary component to digital information representation and communication across devices. In this activity we will develop a Morse Code based system as an introduction to digital communication.

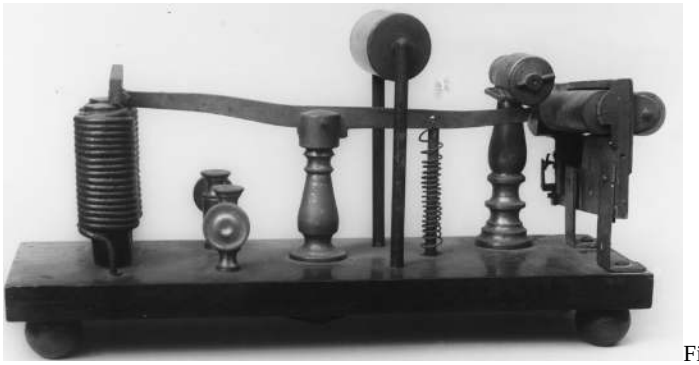


Figure \(\PageIndex{1}\): Copper Telegraph Machine 1837-1843. (Public Domain, Metropolitan Museum of Art, NY)

Series

Most of the devices we will work in this class transmit data through a variety of series communication protocols, and this class will cover several of this. But first, there are a few terms you need to understand with respect to the description of data transfer across devices. Devices hooked in series have either one wire, or two. When there are two wires each is unidirectional, with one sending data from device A to device B, while the other goes in the opposite direction and sends data from device B to device A.

Modes of Operation

There are three basic modes of operation for data transmission along a wire

- Simplex
    - data only goes in one direction
    - Would be use for something like controlling an array of LEDs, they do not send data, only receive it
- Half-Duplex
    - data transmission goes in both directions, but is not simultaneous.
- Full-Duplex
    - simultaneous data transmission in both directions.

Baud (Bd) Rate

Baud rate is a measure of how quick data can be transferred along a line and is named after Émile Baudot for the Baudot code in early telegraphy. The baud rate is the number of symbols (or pulses) per second and the gross baud rate is the number of bits per second. If there are exactly two symbols (0 and 1) then the baud rate equals the bit rate. What is important is that the transmitting device is operating at the same baud rate as the receiving and you may need to check this in your code. Standard baud rates range from 110Bd to 256kBd.

Communication Protocols

It is through protocols that electronic devices can communicate data. The data is transmitted as bits. There are two ways bit can be identified, either as high or low values, or the time a value is high (or low).
Some common serial communications protocols that are germane to IOT are:

- UART (Universal Asnycgronous Receiver-Transmitter)
- I2C (Inter-Integrated Circuit)
- SPI (Serial Peripheral Interface)

- USB (Universal Serial Bus [Wikipedia])

Morse Code

Morse code represents the alphabet by using two characters, dashes and dots, as given in table \(\PageIndex{1}\). Note, there is actually a third entity in the code, which is the space between the dashes and the dots. In Morse code (unlike many digital systems) the number of characters varies from letter to letter, with commonly used letters having few characters and infrequently used letters (or symbols) having more characters ("E" is a dot, "T" is a dash, but a hyphen is dash-dot-dot-dot-dot-dash. Because this is being detected and transmitted by a human the amount of time to send information is very long, but in principle, this is a form of serial communication.

Table \(\PageIndex{1}\): Morse Code

A	•–	K	–•–	U	••–	0	–––––
B	–•••	L	•–••	V	•••–	1	•––––
C	–•–•	M	––	W	•––	2	••–––
D	–••	N	–•	X	–••–	3	•••––
E	•	O	–––	Y	–•––	4	••••–
F	••–•	P	•––•	Z	––••	5	•••••
G	––•	Q	––•–	Period	•–•–•–	6	–••••
H	••••	R	•–•	Comma	––••––	7	––•••
I	••	S	•••	Hyphen	–••••–	8	–––••
J	•–––	T	–	Colon	–––•••	9	––––•

In order to construct a sentence and transmit that information you need to not only identify each character in a letter, but when you are separating letters from letters in the same word, and between different words. For example six dots (\(\cdot \cdot \cdot \cdot \cdot \cdot \)) could be HI or EEEEEE or 5E or E5 or SS or EESE or ESSE or.... you get the problem. So you need to develop a set of protocols to indicate the difference between the dashes, the dots and the time delay between them. Here is a protocol adapted from Bob LeSeyer and FeAtHEr-Cm

Morse Code Translator

The following code comes from GeeksforGeeks. In a future activity you will make an LED Morse code transmitter and some of this code will be useful.

```
# Python program to implement Morse Code Translator

'''
VARIABLE KEY
'cipher' -> 'stores the morse translated form of the english string'
'decipher' -> 'stores the english translated form of the morse string'
'citext' -> 'stores morse code of a single character'
'i' -> 'keeps count of the spaces between morse characters'
'message' -> 'stores the string to be encoded or decoded'
'''

# Dictionary representing the morse code chart
MORSE_CODE_DICT = { 'A':'.-', 'B':'-...',
                    'C':'-.-.', 'D':'-..', 'E':'.',
                    'F':'..-.', 'G':'--.', 'H':'....',
                    'I':'..', 'J':'.---', 'K':'-.-',
```

```
                    'L':'.-..', 'M':'--', 'N':'-.',
                    'O':'---', 'P':'.--.', 'Q':'--.-',
                    'R':'.-.', 'S':'...', 'T':'-',
                    'U':'..-', 'V':'...-', 'W':'.--',
                    'X':'-..-', 'Y':'-.--', 'Z':'--..',
                    '1':'.----', '2':'..---', '3':'...--',
                    '4':'....-', '5':'.....', '6':'-....',
                    '7':'--...', '8':'---..', '9':'----.',
                    '0':'-----', ', ':'--..--', '.':'.-.-.-',
                    '?':'..--..', '/':'-..-.', '-':'-....-',
                    '(':'-.--.', ')':'-.--.-'}

# Function to encrypt the string
# according to the morse code chart
def encrypt(message):
    cipher = ''
    for letter in message:
        if letter != ' ':

            # Looks up the dictionary and adds the
            # corresponding morse code
            # along with a space to separate
            # morse codes for different characters
            cipher += MORSE_CODE_DICT[letter] + ' '
        else:
            # 1 space indicates different characters
            # and 2 indicates different words
            cipher += ' '

    return cipher

# Function to decrypt the string
# from morse to english
def decrypt(message):

    # extra space added at the end to access the
    # last morse code
    message += ' '

    decipher = ''
    citext = ''
    for letter in message:

        # checks for space
        if (letter != ' '):

            # counter to keep track of space
            i = 0

            # storing morse code of a single character
            citext += letter
```

```
        # in case of space
        else:
            # if i = 1 that indicates a new character
            i += 1

            # if i = 2 that indicates a new word
            if i == 2 :

                 # adding space to separate words
                decipher += ' '
            else:

                # accessing the keys using their values (reverse of encryption)
                decipher += list(MORSE_CODE_DICT.keys())[list(MORSE_CODE_DICT
                .values()).index(citext)]
                citext = ''

    return decipher

# Hard-coded driver function to run the program
def English2Morse():
    message = input("Enter English message you want transmitted to morse code. ")
    result = encrypt(message.upper())
    print (result)

def Morse2English():
    message = input("Enter Morse code message you want transmitted to English code. "
    result = decrypt(message)
    print (result)

action=input("Enter e2m if you want to convert English to Morse and m2e if you want t
# Executes the main function
if action == 'e2m':
    English2Morse()
elif action == 'm2e':
    Morse2English()
else:
    print("you did not make a valid request")
```

run | restart | restart & run all

```
Hello world!
```

Resources and References

- Morse Code and the Telegraph (History.com)

This page titled 13.2: Serial and Morse Code is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

13.3: UART

Universal Asynchronous Receiver-Transmitter (UART) is one of the oldest forms of communication. These are the old serial ports of old computers and has been replaced by USB. The data is sent serial as a "data frame". It is "universal" because it can operate over multiple parameters like speed and data size. It is asynchronous because it does not need a clock to synchronize the transmitting and receiving of the data, although they need to be at common BAUD rates.

Features

- 2 lines and 3 pins
  - TX (Transmit line and pin)
  - RX (Receive line and pin)
  - GND (Ground)
- Asynchronous, bidirectional serial data transmission
  - no clock, it uses a "start bit" and an "end bit"
- Limited to one sensor
- BAUD rate (bps or bits per second) 1200, 2400, 4800, 9600, 19200, 38400, 57600 and 115200
  - 115200 is default BAUD rate
  - both devices must operate at the same BAUD rate
- slow, but easy to implement

UART Fundamentals

Data flows from TX pin of Transmitter to RX pin of receiver. The transmitter can convert parallel data from CPU to serial for communication and then the receiver can convert it back to parallel data

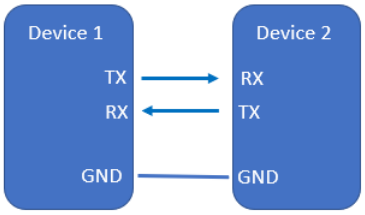


Figure 13.3.1:UART pins and lines for device to device communication (Belford CC0.0)

3 modes of operation

- Simplex (data only goes in one direction)
- Half-Duplex (data transmission in both directions but not simultaneously)
- Full Duplex (simultaneous data transmission in both directions)

UART Data Frame



Figure 13.3.2: UART data frame. (Belford CC0.0)

The data frame has bits that are either high or low. The start bit tells the receiver when a frame is coming and when it is ending. The parity bit is optional and is used to ensure the data is properly transmitted. Since each device has an TX and RX pin, the data can operate in Full Duplex mode where one wire is sending it in one direction and the other in the opposite direction.

Protocol

- Transmitter receives data to be transferred, often from a parallel bus (as CPUs use those because they are faster)
- add start bit, then parity and stop bit
  - parity data checks for data integrity

- Data packet is transmitted until complete
    - Uses a FIFO buffer (First In, First Out) to serialize the stream
- Data in receiving FIFO is cleared as it is transferred

Resources

- seeedstudio blog
- ElectronicWings
- TotalPhase
- electric impDevCenter

This page titled 13.3: UART is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

13.4: I2C

Inter-Integrated circuit

I2C _"Eye Squared C" is a synchronous serial bus that can hold up to 128 devices on a Raspberry Pi. It is used with many common sensors like the BME680 VOC sensor. This is a synchronized protocol using a clock signal that is controlled by the master to align the master and slave. You can have infinite "Masters" and up to 1008 "slaves", although the Raspberry Pi only uses 128 "slaves". Standard mode is 100 kbps, high speed is 3.4 Mbps. A typical I2C device has four pins

- SDA (Serial Data Pin)
- SCL (Serial Clock Pin)
- VIN (Voltage in, 3.3 V on Pi GPIO)
- GND (Ground)

So the I2C uses one line for data transmission and one for the clocksignal, with the other lines running power to a device, and serving as ground.

Circuit Protocols

I2C data is transferred in frames.



Figure 13.4.1: Copy and Paste Caption here. (Copyright; author via source)

- Start Condition:
    - Data line switches voltage (high to low) which starts clock
- Address:
    - 7 to 10 bit address that is unique to each sensor. This identifies sensor the controller is communicating with.
    - Always follows Start
- Read/Write
    - Always follows address
    - Identifies if controller is sending data to sensor or requesting it
    - High voltage means data is requested from sensor
    - Low voltage means data is sent to sensor
- ACK/NACK
    - If address or data frame is successfully received it is acknowledged
    - Always follows an address or data frame
    - ACK bit must be received before next data frame is sent
- Data frame
    - 8 bit data frame

This material is adpated from Circuit Basics and you should read their writeup for a more extensive description of I2C protocols

Enabling I2C

Command Line

In the command line type

```
sudo raspi-config
```

Choose:

1. Interfacing Options
2. I2C
3. Yes

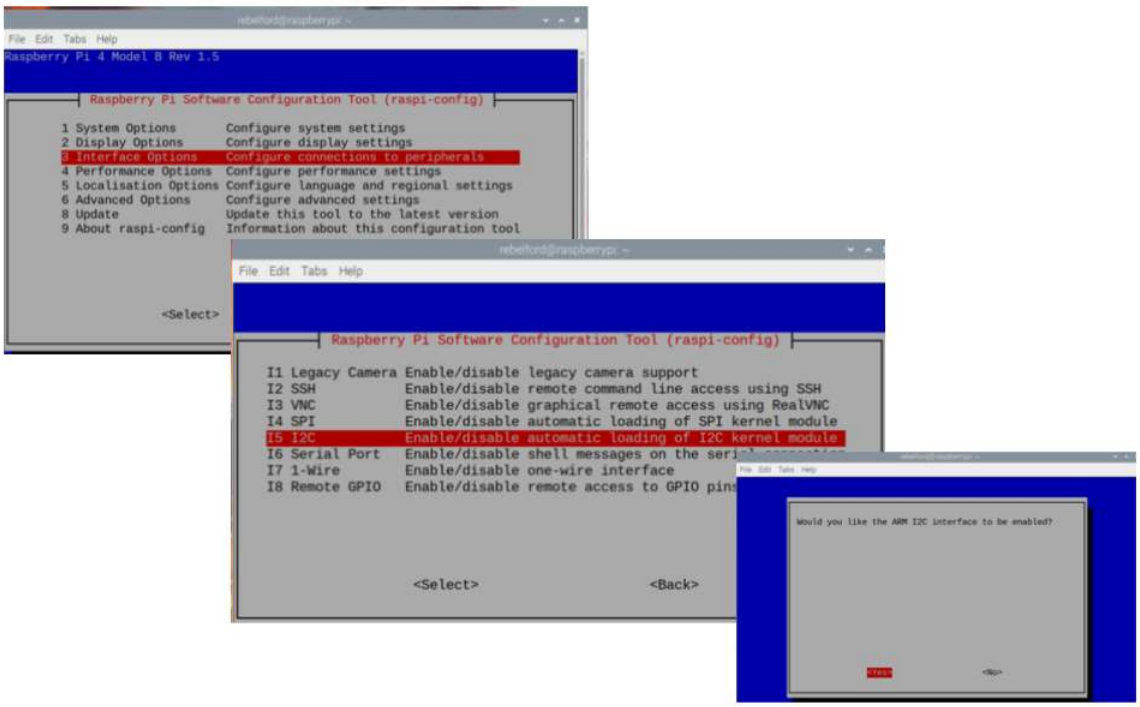


Figure 13.4.1: Copy and Paste Caption here. (Copyright; author via source)

Raspberry Pi Configuration

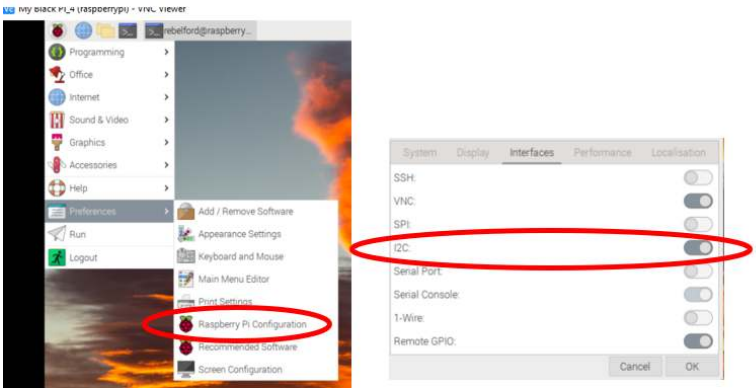


Figure 13.4.1: Copy and Paste Caption here. (Copyright; author via source)

Utility Commands

```
i2cdetect -y 1
```

the -y flag disable interactive mode so you do don't have to wait for confirmation. The 1 indicates you are scanning i2c bus #1.

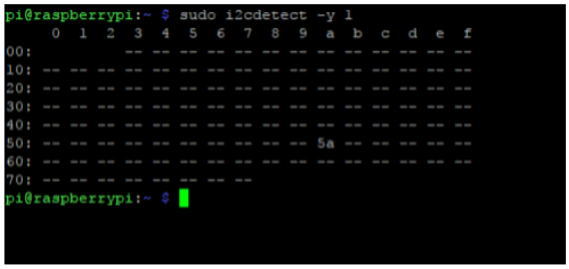


Figure 13.4.1: The addresses are base 16 (hexadecimal), and so 5a is registry 90 in decimal. (Belford, cc0.0)

Resources

- Circuit Basics
  - Good Article on Protocols
- Archive of I2C articles on Raspberry Pi
- SparkFun: Raspberry Pi SPI and I2C Tutorial
- SparkFun: I2C

This page titled 13.4: I2C is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

13.5: SPI

Serial Per

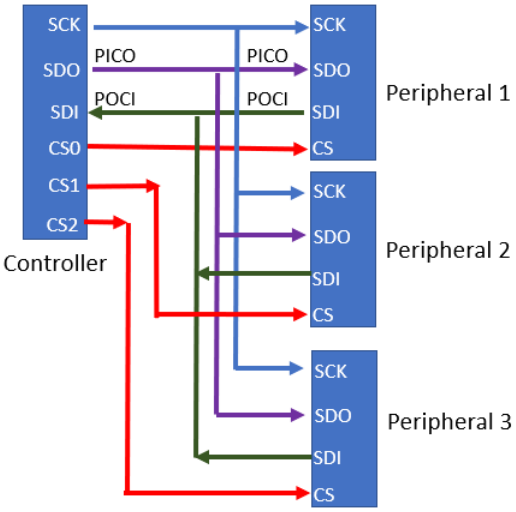


Figure 13.5.1: Data lines for SPI interface with three peripheral devices. Note you still need to provide power and a ground to the perpherials. (Belford CC-BY)

Note

The Open Source Hardware Association has put forth a resolution to redefine the SPI Signal Names. In many devices and data sheets you will see the older names, and you will need to be familiar with both. In essence the old terminology used Master/Slave designation and the new one uses Controller/Peripheral (device) designations.

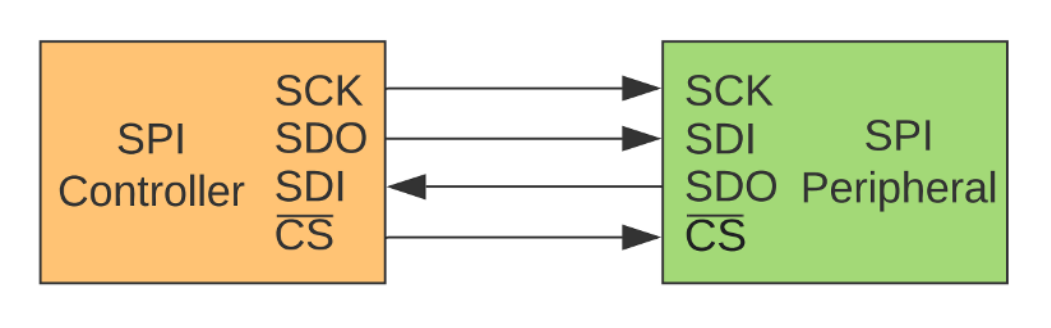


Figure 13.5.2: New signal designations. (SparkFun CC 0.0)

New Signal Names

- SDO - Serial Data Out. An output signal sent to another SPI device
- SDI - Serial Data In. An input signal received from another SPI device.
- CS - Chip Select. Picks which peripheral device is active (old SS).
- PICO - Peripheral In Controller Out. Signal being sent from controller to peripheral (old MOSI).

- POCI - Peripheral Out Controller In. Signal sent from peripheral to controller (old MISO).
- SDIO - Serial Data In/Out. Bidirectional serial signal.

Deprecated Signal Names

- MOSI - Master Out Slave In
- MISO - Master In Slave Out
- SS - Slave Select
- MOMI - Master Out Master In
- SOSI - Slave Out Slave In

Unchanged Signal Names

- SCK - Serial Clock. Controller generated clock signal for the common bus.

solving problems, as long as you get the **proper values a**





- Circuit Basics
- SparkFun: Serial Peripheral Interfaces

This page titled 13.5: SPI is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

CHAPTER OVERVIEW

14: APIs

Topic hierarchy

14.1: Computer Interfaces
14.2: Creating a Google Service Account
14.3: Enabling Google Sheets and Obtaining a Key
14.4: PEP 668 and Debian Bookworm
14.5: Thonny-Virtual Environments
14.6: Upload Data to Google Sheet

14: APIs is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

14.1: Computer Interfaces

You could say a computer interfaces represent how a computer interacts and communicates with an external entity. That entity may be a human, or some other electronic machine. In this course we are using two types of Human Computer Interfaces (HCI); the Command-Line Interface (CLI) and the Graphical User Interface (GUI). Other human to computer Interfaces include Touchscreen Interface, Voice User Interface (VUI), Virtual Reality Interface (VRI) and Augmented Reality Interface (ARI). An interface between two machines is an Application Program Interface (API), which involves software communicating between two different devices, and these are extensively used in the Internet of Things.

Common HCI

The only Human Computer Interface we are using in this class are the CLI and GUI, as those are the two interfaces you are using in this class to communicate with the Raspberry Pi.

CLI

The terminal where you type Linux commands is the CLI of the Raspberry Pi. This allows you to interact with the operating system and is keyboard driven.

GUI

The desktop of the normal Raspberry Pi OS installation if a GUI that you can navigate by mouse or other pointer device. This allows you to use Interactive Development Environments (IDE) like Thonny, which can assist with programming and troubleshooting. Most people today communicate with computers through GUIs.

API

The Application Program Interface (API) is critical to the internet of things and we will quickly review them as few students will be familiar with APIs. Many of these you use without even thinking about it, but they are fundamental to the processes being run by a computer and it is important you recognize that.

Hardware APIs

These allow software to interact with input and output data and involve ports that you can connect devices to. This includes printers, speakers, microphones, sensors and actuators. These require hardware interfaces or ports such as USB, hdmi, the Raspberry Pi 40 pin GPIO pins or even Bluetooth wireless communication technology to enable communication with the device. The hardware API acts as an access point to allow programmers to use communication protocols to communicate with devices .

Web APIs

Web APIs allow you to communicate between computers across the web. For example, a web page has a URL (Universal resource Locator) which is the unique address the server uses to provide the page you are reading. A URI (Universal Resource Identifier) is like a URL for a specific piece of data accessible over the web.

When we learned how to concatenate strings byu communicating with PubChem to get a molar mass of a molecule we were using the PUG (Power User Gateway) REST (Representational State Transfer) API provided by NIH NCBI to programmatically access chemical data within the PubChem database.

Operating System APIs

Linux is the operating system of a Raspberry Pi and it serves as the interface between you and your computer's hardware. This allows you to manage resources like memory, processor time and input/output operations. This is not really a programming language like Python, although you can use BASH (Bourne-Again SHell) scripts to automate repetitive tasks in Linux systems

Database APIs

These are application program interfaces that allow computers to interact with databases

Library APIs

The import statement in Python is an example of the use of a Library API, where you are taking code in one program and using it in another. That is, two programs are communicating with each other. In Python Libraries are collections of modules that contain

variables, functions and classes that can easily be used by other programs through a library API.

This page titled 14.1: Computer Interfaces is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

14.2: Creating a Google Service Account

In order to programmatically upload data to a Google Cloud service your program needs to "log in", but your program is not a human being, and so you need to create a service account that allows you to run an authentication process. As of March 2023 Google uses the OAuth 2.0 authentication framework

1. Log into Google using the account you set up for this class and go to the cloud console (console.cloud.google.com/)
2. Click on the Project dropdown box and choose new project

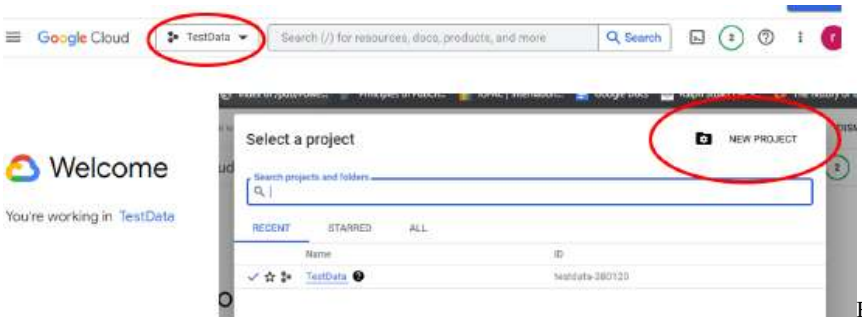


Figure 14.2.1:Screen capture for creating a new project. (Copyright; Belford cc 0.0)

4. Give your project a name and create it.

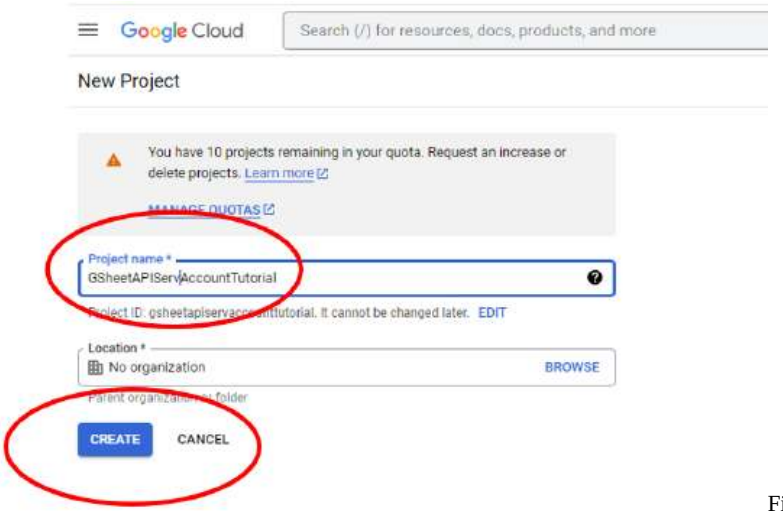


Figure \PageIndex{2}:Creating a new project. (Copyright; Belford cc 0.0)

5. Now select your project (mine is PCompBobData



Figure 14.2.1: Copy and Paste Caption here. (Copyright; author via source)

Go to Navigation menu, choose APIs & Services/Credentials

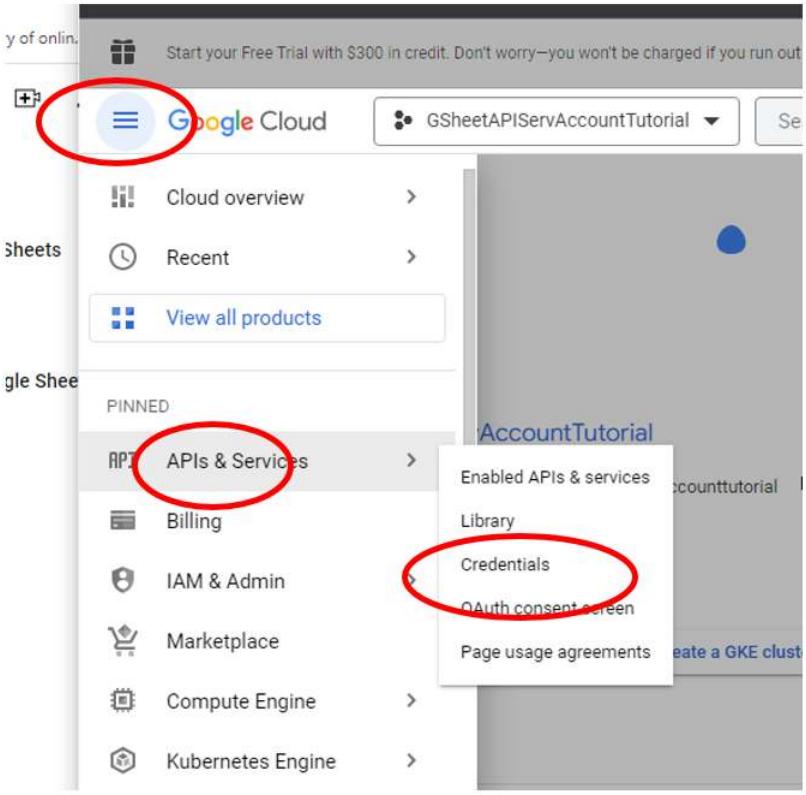


Figure 14.2.3:Screen capture for creating a new project. (Copyright; Belford cc 0.0)

6. Now choose CREATE CREDENTIALS and choose Service Account

ect

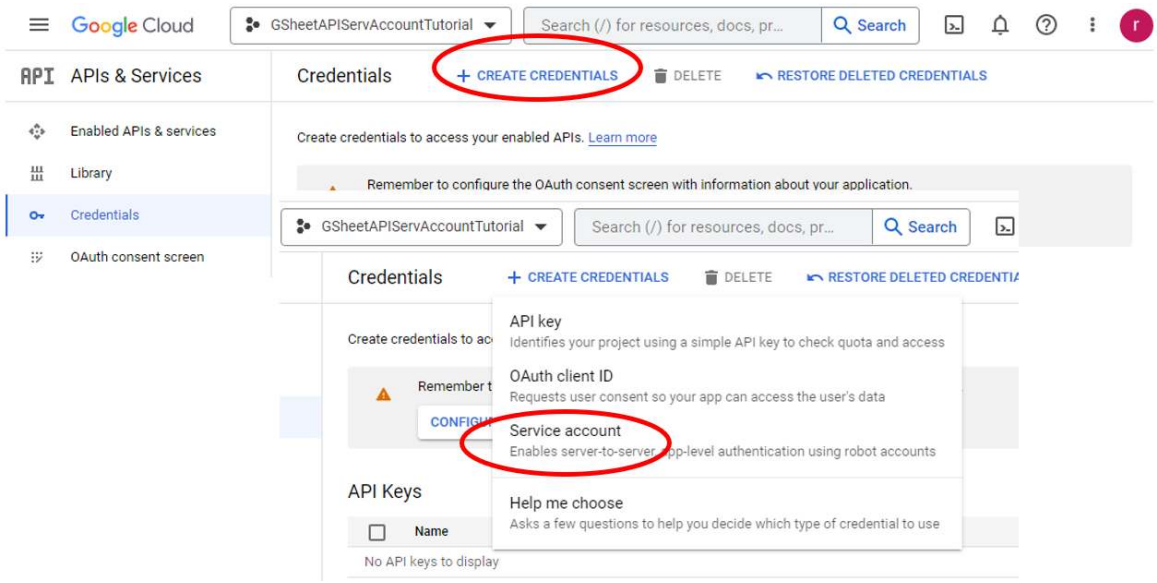


Figure 14.2.4:Screen capture for creating a new project. (Copyright; Belford cc 0.0)

7. Give your account a name and create it.

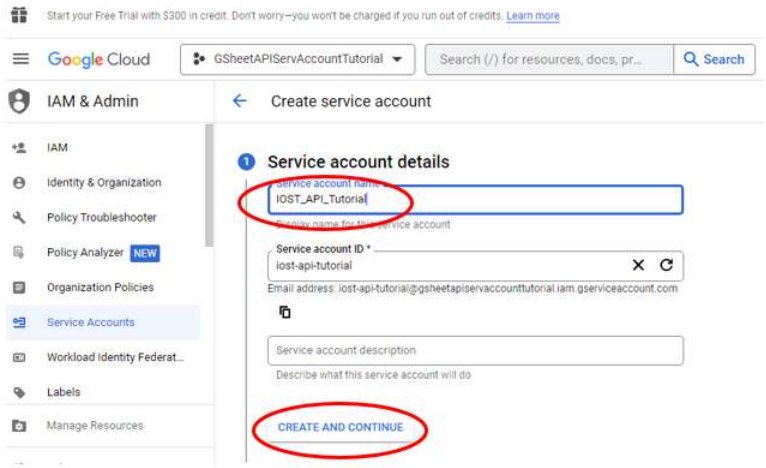


Figure 14.2.5:Screen capture for creating a new project. (Copyright; Belford cc 0.0)

8. Set yourself up as the owner

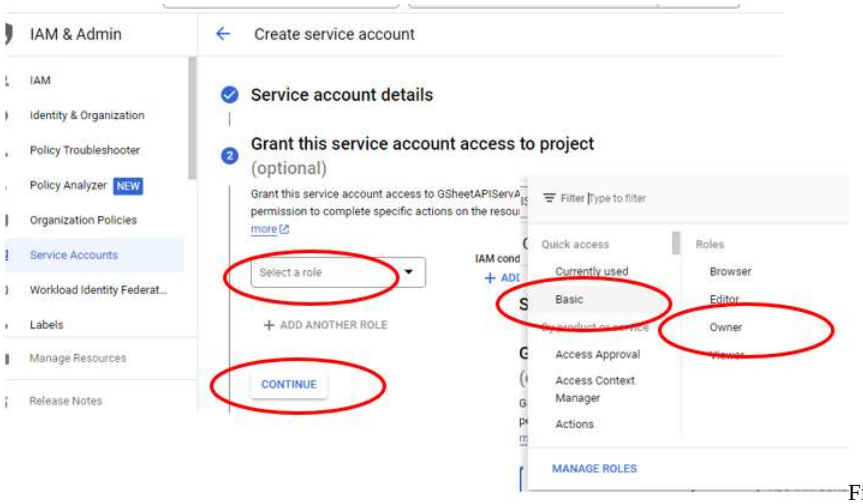


Figure 14.2.6:Screen capture for creating a new project. (Copyright; Belford cc 0.0)

9.

Complete the task

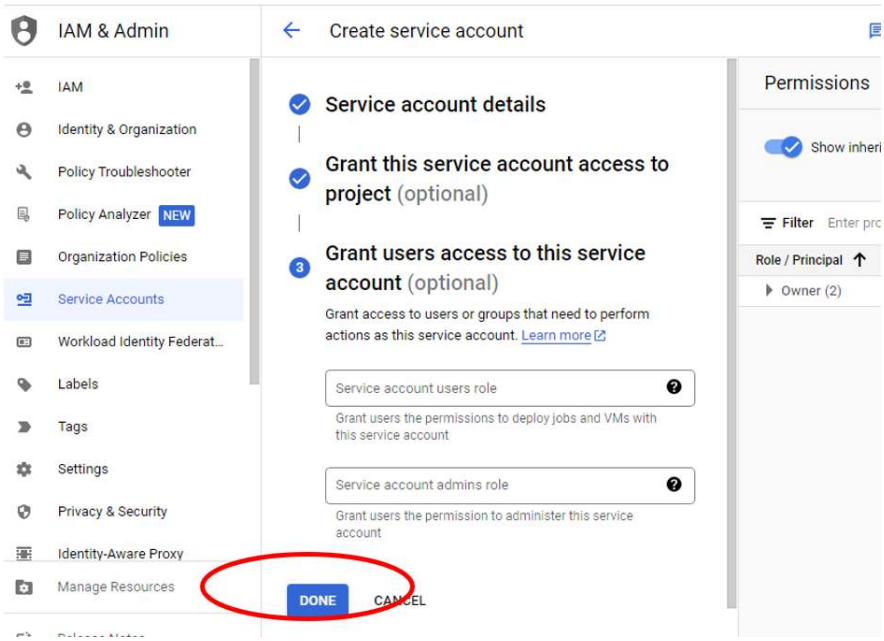


Figure 14.2.7:Screen capture for creating a new project. (Copyright; Belford cc 0.0)

This page titled 14.2: Creating a Google Service Account is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

14.3: Enabling Google Sheets and Obtaining a Key

In order to connect a sensor to a Google sheet you need to enable a sheet in your Drive and then get a key that your code can use to access the sheet. The key will be in the form of a Json file.

Enable Google APIs

1. Go to the Navigation menu/APIs & Services and choose Enabled APIs & services

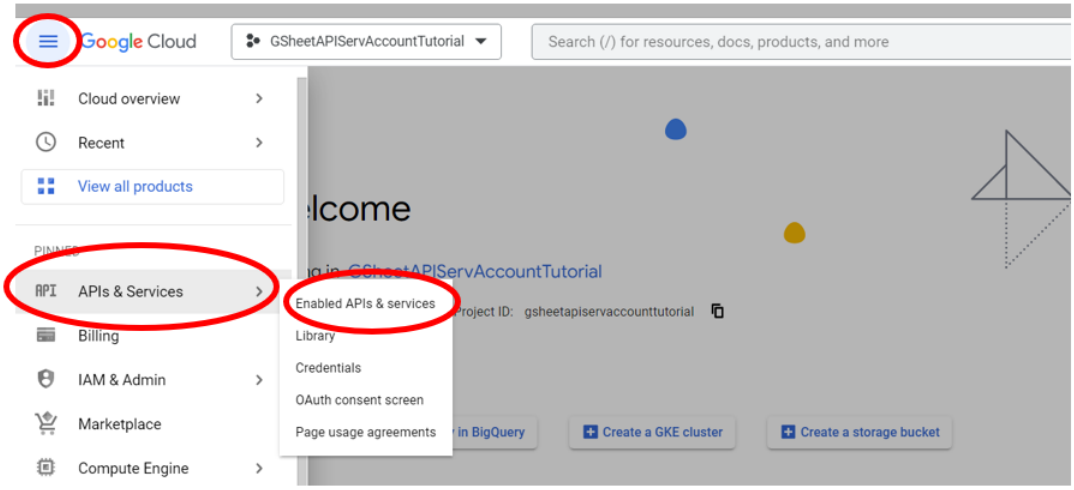


Figure 14.3.1: Enabling Google APIs, March 2023. (Belford cc 0.0)

2. Click +ENABLE APIS AND SERVICES. Note the large selection of APIs and services Google offers

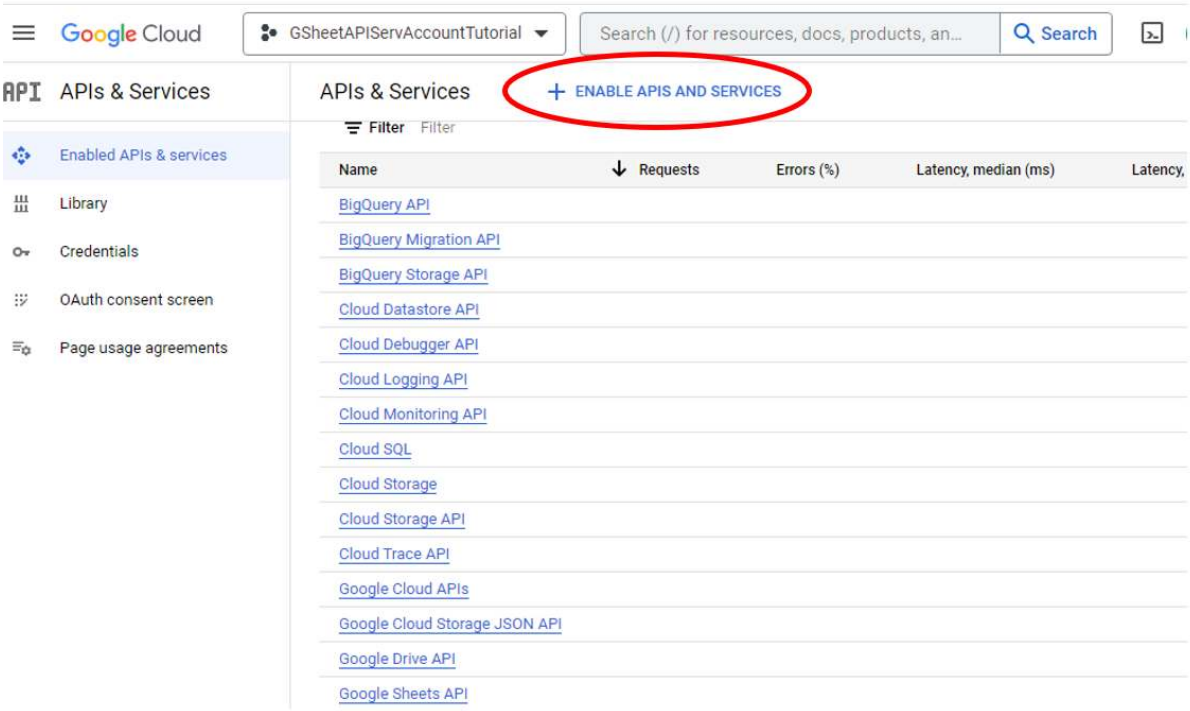


Figure 14.3.2: Enabling Google APIs, March 2023. (Belford cc 0.0)

3. You will need to enable both Google Drive and Google Sheets APIs, and enable both. These are in the Google Workspace and you may need to click view all (31)

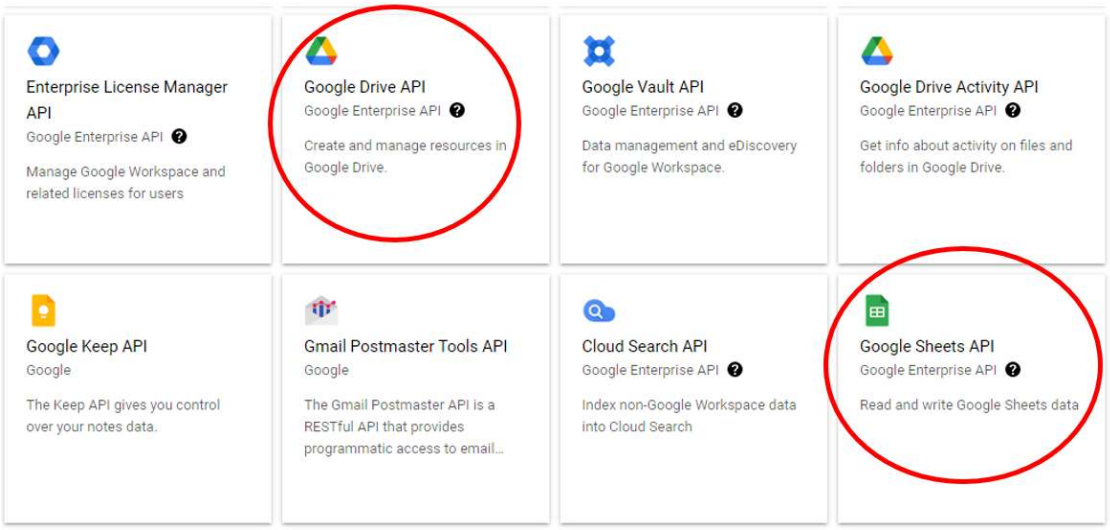


Figure 14.3.3: Enabling Google APIs, March 2023. (Belford cc 0.0)

Obtain Key

1. Go to the Navigation menu/APIs & Services and choose Credentials

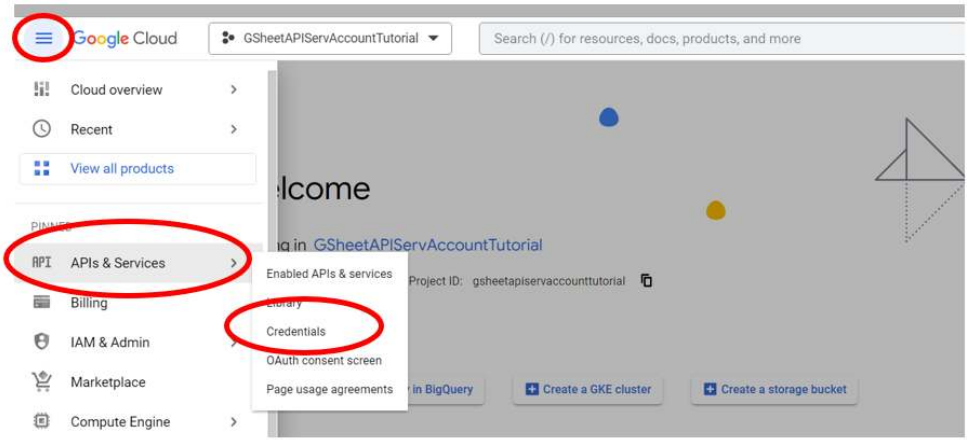


Figure 14.3.4: Creating a key, March 2023. (Belford cc 0.0)

2. Make a new key in the form of a JSON file. To create a JSON file navigate to Credentials. You should see your newly created service account. Click on the edit icon (pencil). Then go to the Keys tab, then click "Add Key" --> "Create new key"

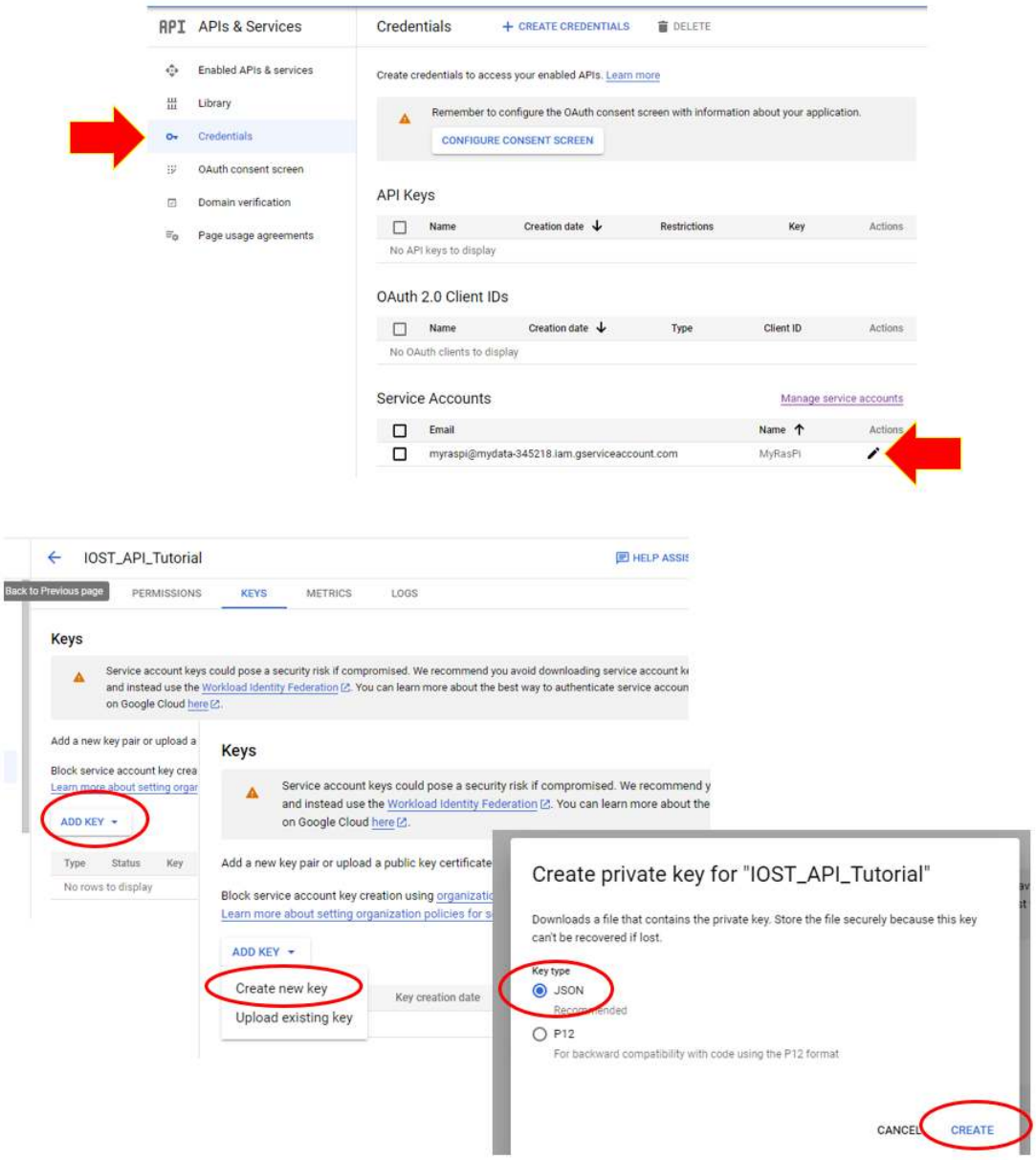


Figure 14.3.5: Creating a key, March 2023. (Belford cc 0.0)

3. The key will be downloaded to your computer.

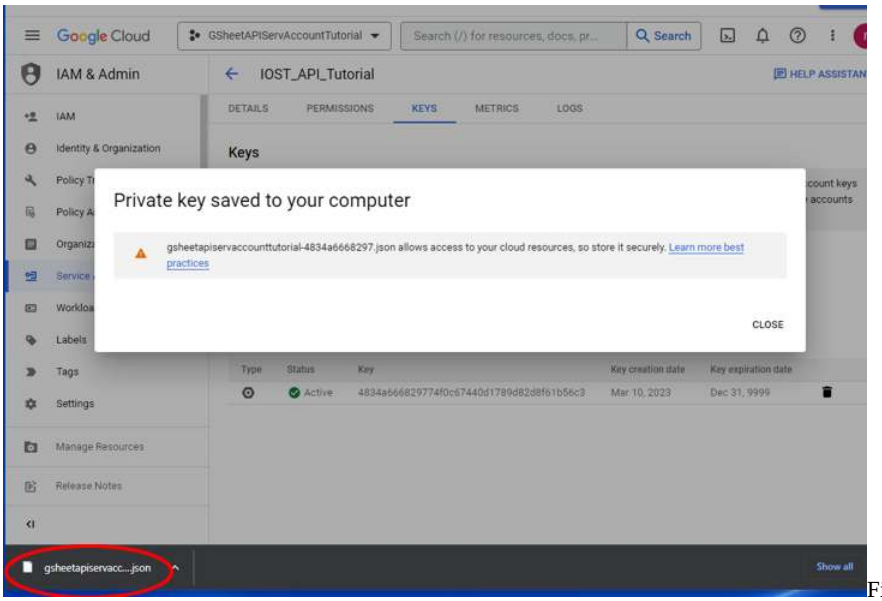


Figure 14.3.6: Creating a key, March 2023. (Belford cc 0.0)

4. You will want to rename the key and move it to the folder with your Python program that will be using it.

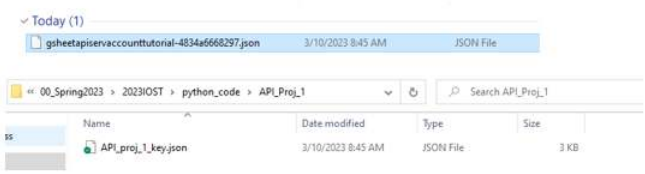


Figure 14.3.7: Creating a key, March 2023. (Belford cc 0.0)

Take a look at the json file. Note that it has information related to your project, your account and the authentication service.

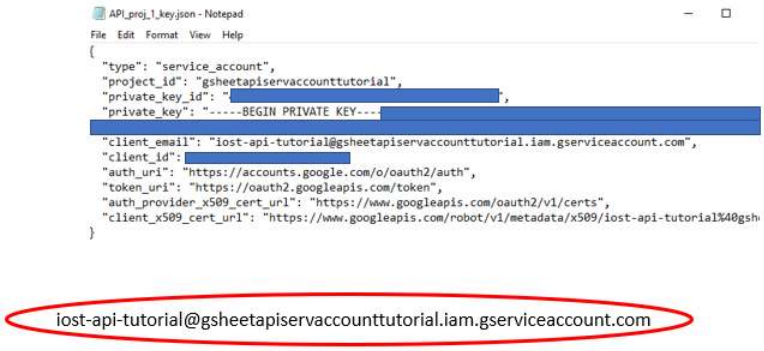


Figure 14.3.8: Creating a key, March 2023. (Belford cc 0.0)

Note, if you go to your credentials you will see that your Service Account for this project has the same email as is in the json file. In the next step you will create a Google Sheet and share it with this email, so that your Python code that uses the key can upload data through the Google sheet API.

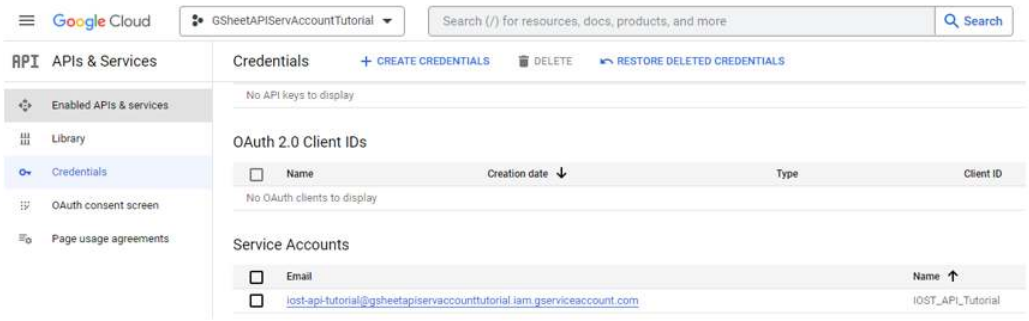


Figure 14.3.9: Creating a key, March 2023. (Belford cc 0.0)

Create a Google Sheet

I suggest you create a folder in your Google Drive for your class projects, that is, your school Google Drive and not the one you used to create this account with. Right click in that folder and create a new Google Sheet, in this tutorial we will call this "classTest". But you should name it as appropriate to your assignment. Once you have created the sheet you need to share it with the email associated with your service account. Be sure to give it editor access.

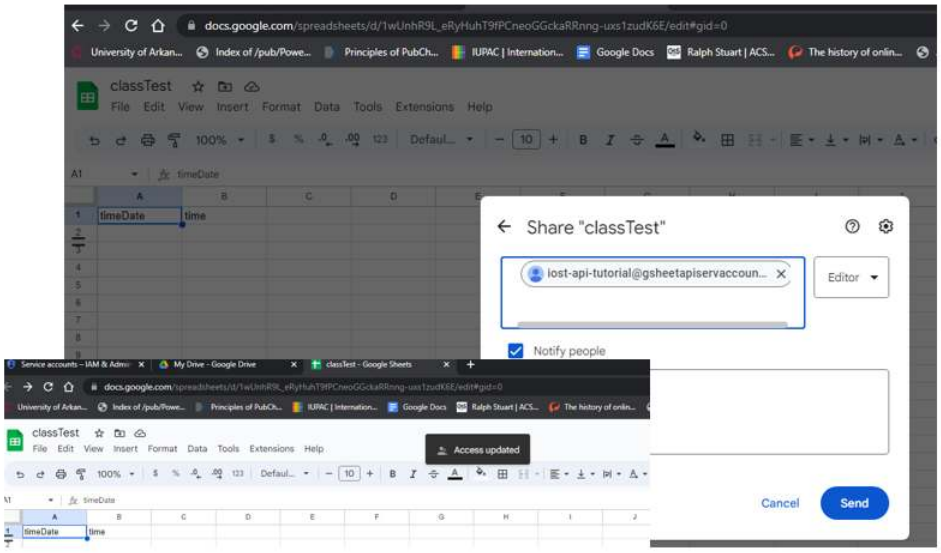


Figure 14.3.10: Creating a key, March 2023. (Belford cc 0.0)

You should now be ready to write some code that will allow you to upload data to the Google Sheet in real time.

This page titled 14.3: Enabling Google Sheets and Obtaining a Key is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

14.4: PEP 668 and Debian Bookworm

Externally Managed Python Base Environments

14.4: PEP 668 and Debian Bookworm is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

14.5: Thonny-Virtual Environments

Thonny makes it very easy to create virtual environments. Once you have created a virtual envrionment you can install packages from resources like the python package index. In this module we will show you how to create a virtual environment and install the google spreadsheet Python API gspread package. First, go to pypi.org and search for gspread, and you see that gspread 6.0.2 was last updated Feb 14, 2024.



Figure 14.5.1: Copy and Paste Caption here. (Copyright; author via source)

when you click on it you see you can install it by running the following pip command on the terminal

```
pip install gspread
```

But when you run it you get the following error message, which essentially says that you must set up a virtual environment (venv) if you want to install an externally managed package.

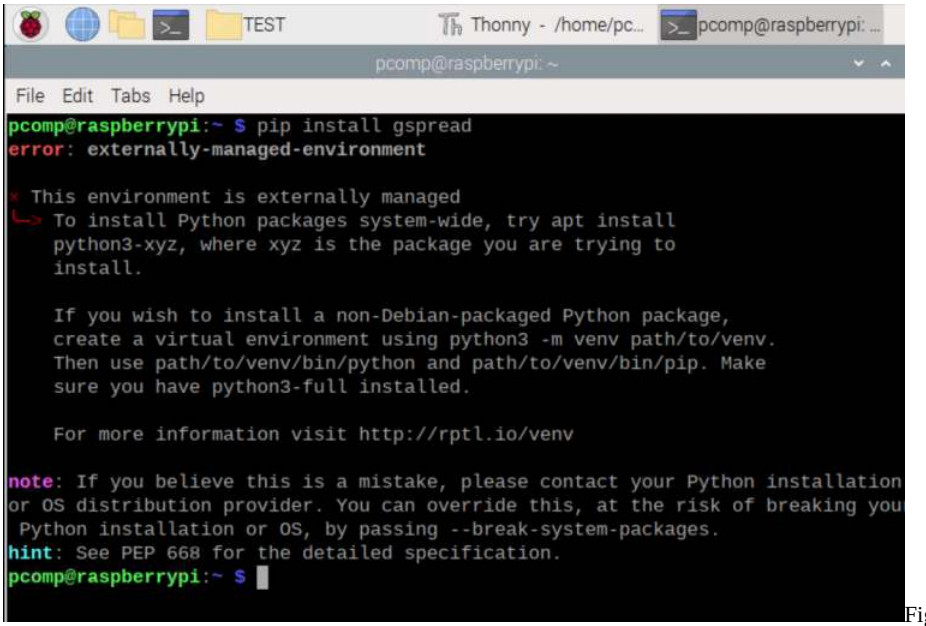


Figure 14.5.1: Copy and Paste Caption here. (Copyright; author via source)

Creating a Virtual Environment

Create an empty folder - this will be the folder we use for assignment 8 the Google API: ASN8_API

Open Thonny and choose tools/options

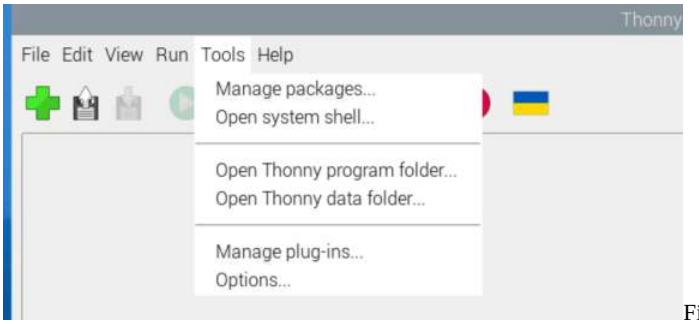


Figure 14.5.1: Copy and Paste Caption here. (Copyright; author via source)

Go to Interpreter and choose New virtual environment

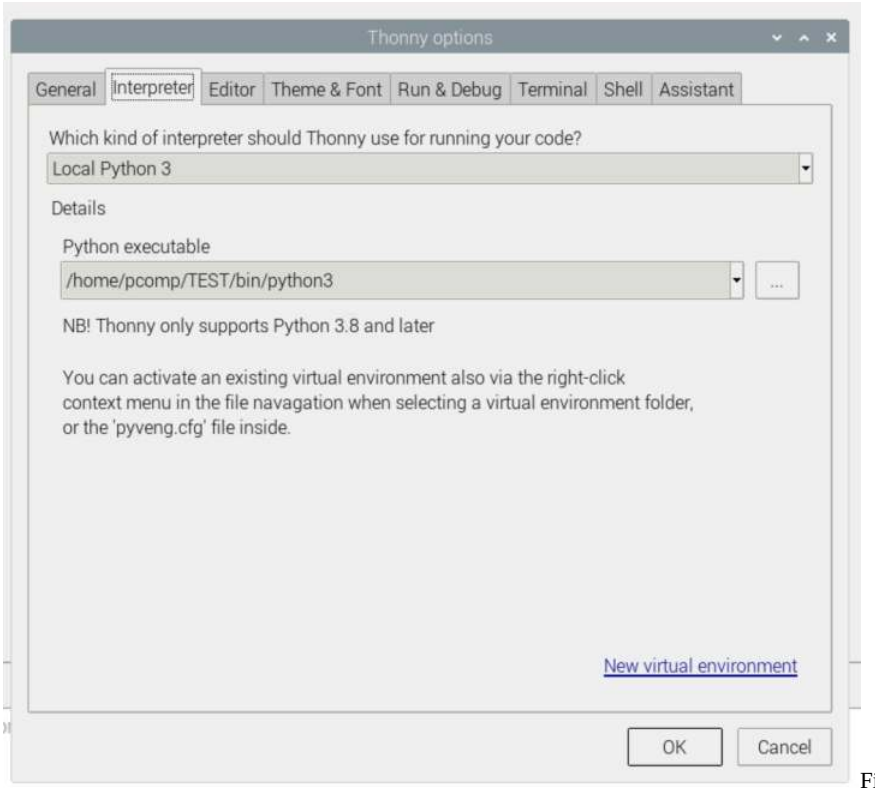


Figure 14.5.1: Copy and Paste Caption here. (Copyright; author via source)

After clicking OK you need to navigate to your empty folder



Figure 14.5.1: Copy and Paste Caption here. (Copyright; author via source)

You will probablty need to use the Other locations, although there is a Desktop option if that is where you are keeping your assignments

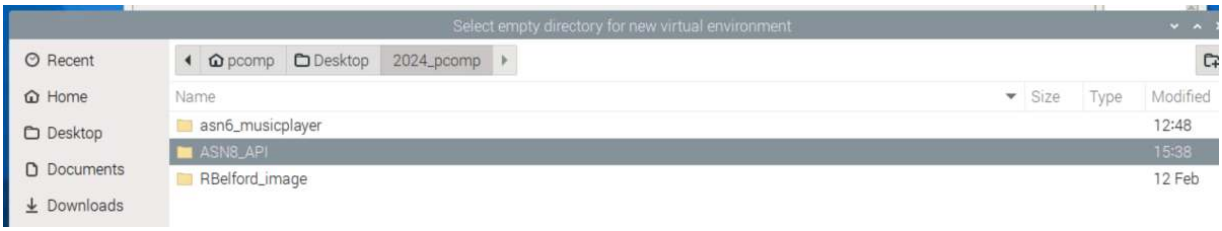
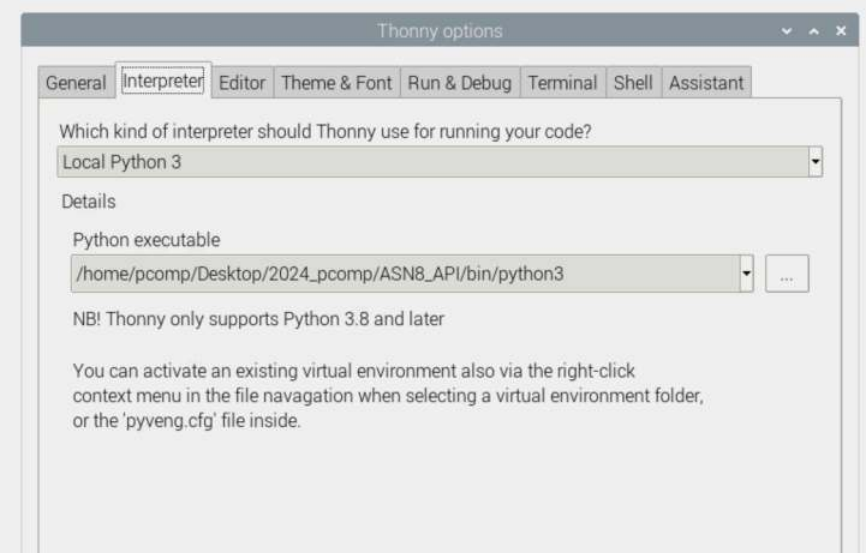


Figure 14.5.1: Copy and Paste Caption here. (Copyright; author via source)

click OK, and it will create a virtual environment in that folder



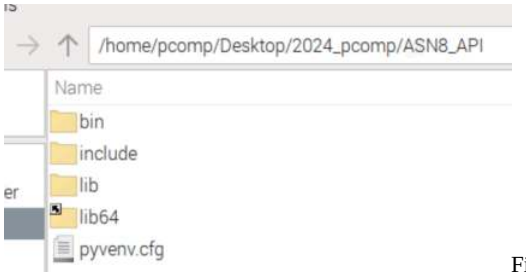


Figure 14.5.1: Copy and Paste Caption here. (Copyright; author via source)

now by simply clicking the pyenv.cfg file you will open Thonny with the virtual enterpreter.

Go to Tools Manage Packages and you can search the PyPi package index from Thonny

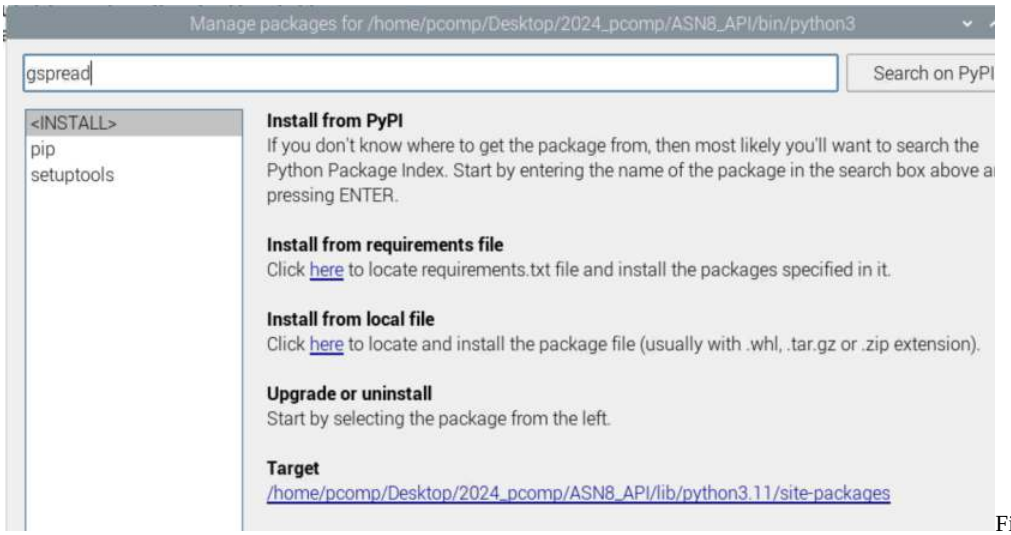


Figure 14.5.1: Copy and Paste Caption here. (Copyright; author via source)

Chose the package you want and click instal

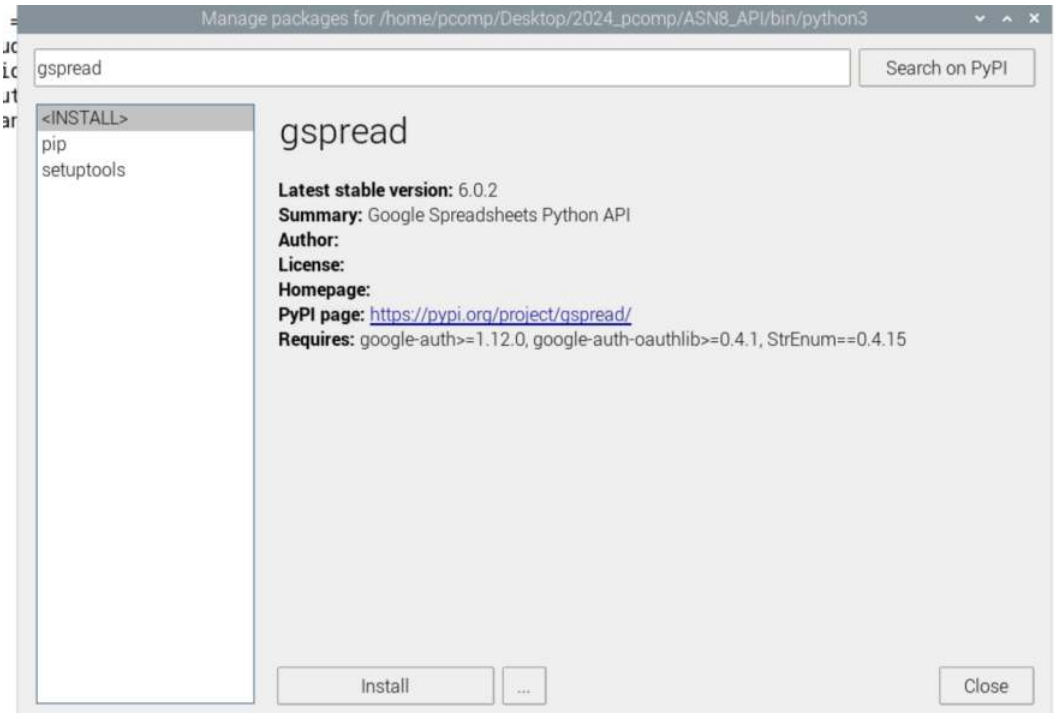


Figure 14.5.1: Copy and Paste Caption here. (Copyright; author via source)

You now have a virtual environment with the gspread suite and all of its dependencies.

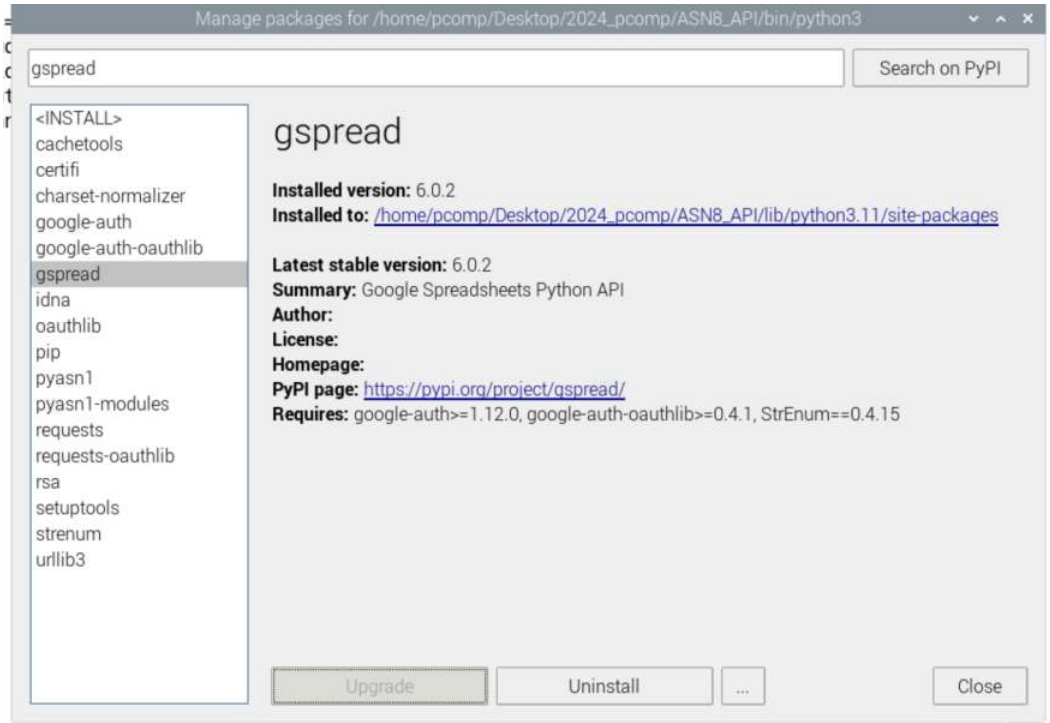


Figure 14.5.1: Copy and Paste Caption here. (Copyright; author via source)

14.5: Thonny-Virtual Environments is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

14.6: Upload Data to Google Sheet

These activity assumes you have (1) created a Google Service Account, (2) Created a Key, (3) Enabled Google Drive and Sheets and (4) Created a Google sheet that is shared with the email associated with the key of your Google Service Account.

Copy JSON Key to Raspberry Pi

The Python script will be referencing the JSON key that we downloaded from the Google Cloud Platform. This means that you need to copy the file from your device to the Raspberry Pi. We can use VNC Viewer for this.

1. Open VNC Viewer and connect to your Raspberry Pi. Move your cursor to the top boarder of the VNC window, you will see the control panel slide down. Select the Icon with two arrows



Figure 14.6.1: Transfer files via VNC. (Copyright; E. Lisitsyna CC0.0)

2. Click "Send Files" and select the JSON file from the Downloads folder on your device (unless you saved it to a different folder). Click Open
3. The file will appear on your Raspberry Pi desktop. Close the file transfer window.
4. Move your JSON file to the assignment folder where you will create your Python scripts and rename it to mydata-key.json

Static Data Upload

1. Open terminal and install gspread module:

```
pip3 install gspread
```

2. Open Thonny, copy and paste the following script. Make sure to save your Python script to the same location where you moved your JSON file to

```
#Static Data Upload Code

import datetime, time
import gspread
from google.auth.transport.requests import AuthorizedSession
from google.oauth2 import service_account

#Access Google API
googleAPI = 'mydata-key.json'
scope = ['https://www.googleapis.com/auth/drive']
credentials = service_account.Credentials.from_service_account_file(googleAPI)
scopedCreds = credentials.with_scopes(scope)
gc = gspread.Client(auth=scopedCreds)
gc.session = AuthorizedSession(scopedCreds)
```

```
sheet = gc.open("RasPi Data")
worksheet = sheet.worksheet("Sheet1")

#Upload data to Google Sheets
time = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
data = 0
values =[time, data]
worksheet.insert_row(values, 2, value_input_option='RAW')
print("Google Sheet Updated")
```

3. Check your Google Sheet. You should see date and time in the first column and "0" in the second column.

Using PubChem API to Upload Data Through Google Sheet API

When we introduced concatenation we showed how you could combine strings to create a uri (universal resource identifier) that allows you to extract data from a chemical compound database API like that in PubChem. In this activity you will make a new tab on your Google Sheet and then merge the script from two scripts to create a program that will only upload data to your Google Sheet when you ask it to.

You want to create a Python program that inputs the name and mass of a chemical and outputs to a spreadsheet the request date stamp, chemical name, molar mass, mass and moles in a sample, as indicated in figure 14.6.2

	A	B	C	D	E
1	Date	Chemical Name	Molar Mass (g/mol)	mass (g)	moles
2	2023-03-13 10:3	sodium chloride	58.44	29.22	0.5

Figure 14.6.2: Copy and Paste Caption here. (Copyright; author via source)

1. Create a new tab in your Google Sheet labeled PubChem
2. Edit the column headers in Google Sheet 2. Put "Compound Name" in cell B1, "Mass, g" in C1, "Moles" in D1 and leave "DateTime" in cell A1
3. Copy and paste the following script to your Thonny IDE, but don't run it yet:

```
#!/usr/bin/python3

import datetime
from time import sleep
import gspread
from google.auth.transport.requests import AuthorizedSession
from google.oauth2 import service_account
import requests

#Access Google API
googleAPI = 'YOUR_JASON_KEY.json'
scope = ['https://www.googleapis.com/auth/drive']
credentials = service_account.Credentials.from_service_account_file(googleAPI)
scopedCreds = credentials.with_scopes(scope)
gc = gspread.Client(auth=scopedCreds)
gc.session = AuthorizedSession(scopedCreds)
sheet = gc.open("YourSpreadSheet") #Change this to match your spreadsheet's name
```

```
worksheet = sheet.worksheet("PubChem")

#Send request to Pubchem
def getMoles():
    uri = ("https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/name/"
           +str(compound_name) +"/property/MolecularWeight/txt")
    res = requests.get(uri)
    molar_mass=float(res.text.rstrip())
    moles = mass/molar_mass
    return moles

# def getMolarMass():
#     uri = ("https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/name/"
#            +str(compound_name) +"/property/MolecularWeight/txt")
#     res = requests.get(uri)
#     molar_mass=float(res.text.rstrip())
#     return molar_mass


#Main loop
another_chemical = True
while another_chemical is True:
    compound_name = input("Input compound's name: ")
    mass = float(input("input mass of sample in units of g: "))
    moles = getMoles()
    #molar_mass=getMolarMass()
    time = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    values =[time, compound_name, mass, moles]
    worksheet.insert_row(values, 2, value_input_option='RAW')
    print("Google Sheet Updated \n Would you like to test another chemical")
    sleep(1)
```

4. You need to alter the above script by inputting your jason key and the name of your Google Sheet. When you run it the output will not be what you want, but what is in figure 14.6.3. You should look at the parameters of the worksheet.insert_row method, see what happens when you change the "2", and think about the python list "values".

	A	B	C	D	E
1	Date	Chemical Name	Molar Mass (g/mol)	mass (g)	moles
2	2023-03-13 10:4	sodium chloride	29.22	0.5	

Figure 14.6.3: Clearly the output is not what we desire. (Belford cc 0.0)

Hint: Analyze the commented out code, and remember the Thonnyt IDE shorcuts, <alt>+3 to comment out code and <alt>+4 to uncomment code.

Contributors

Elena Lisitsyna created this module using prior material developed with assistance from Dr. Sunhwan Kim and Bob Belford.

This page titled 14.6: Upload Data to Google Sheet is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

CHAPTER OVERVIEW

15: DHT22 Temperature and Humidity Sensor

Topic hierarchy

15.1: DHT22 Temperature and Humidity Sensor
15.2: DHT 22 - Streaming Temperature and Humidity Data
15.3: Writing to a CSV file (DHT22 data)
15.4: Reading a csv File

15: DHT22 Temperature and Humidity Sensor is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

15.1: DHT22 Temperature and Humidity Sensor

The DHT22 sensor consists of a capacitive humidity sensor and a thermistor with a built in Analog to Digital (A2D) converter that uses a proprietary 1-wire communication protocol to transmit the signal to a microcontroller or microcomputer. The sampling frequency is rather slow (0.5 Hz or once every 2 seconds) but it is a good first sensor.

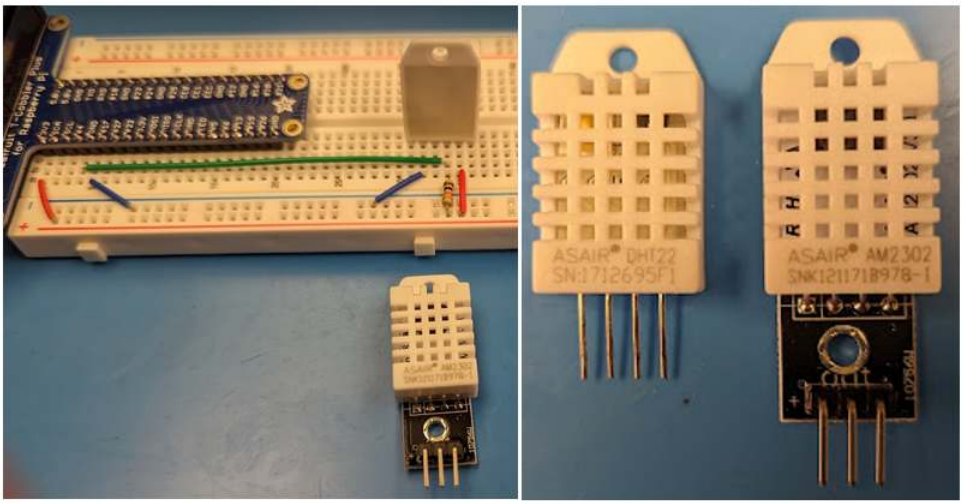


Figure 15.1.1: The DHT22 is sold as either a sensor (4 pin) or as a sensor module (3 pin) with a built in circuit board. If you buy the 4 pin module you need to place a 10 kΩ resistor between the VCC and data pin(Belford CC-BY)

Specs

- 3 to 5 V power
- 2.5 mA max current
- 0-100% humidity range with 2-5% accuracy
- -40 to 80 deg C temp range with ± deg C accuracy
- 0.5 Hz sampling frequency

Wiring

Any time you need to connect a sensor to your Raspberry Pi, you should start with searching the web for the pinout of that sensor. If you search for "DHT22 pinout" you will find many images similar to the one below.

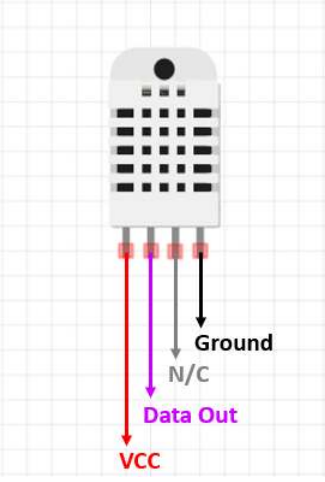


Figure 15.1.2: Fritzing diagram of DHT22 (CC0; E. Waters)

- Pin#1 VCC (Voltage Common Collector) - power input of 3.3 or 5 V (use red wire)
- Pin#2 Data Out pin - connecting to a GPIO pin and a pull-up resistor (use white or yellow wire)
- Pin#3 N/C - not connected
- Pin#4 Ground pin (use black wire)

Communication Protocol

the GPIO pin has two values, high (3.3 V) and low (0V)In a one wire protocol the length of the high voltage determines if a value is one or zero (figure 15.1.3 These bits combine to form bytes (8 bits) and need to transfer both humidity and temperature data. A typical output value can be, 8bit humidity integer data + 8 bit humidity decimal data + 8 bit temperature integer data + 8 bit fractional temperature data + 8 bit parity bit (components 101).

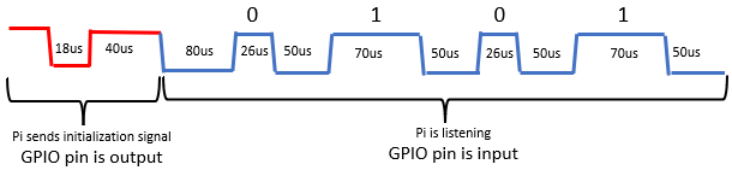


Figure 15.1.3: The duration of the high voltage defines if the bit value is one or zero. these values come from Qengineering's Github, but the principle is a short high voltage is zero and a long one is 1 (Belford cc 0.0)

So initially the GPIO pin is functioning as an output, to start the process, and then it functions as an input, and reads the data stream. At the end the voltage returns to a high value, which is the role of the pull-up resister attached to the data pin.

References and Resources

- AdaFruit DHT11, DHT22 and AM2302 pdf
- Adafruit CircuitPython DHT Library (docs)
- Adafruit CircuitPython_DHT library
- Adafruit Github DHT-sensor-library
- Calibration Notes (Kimberly & Robert's)
- Components 101
- Aosong Electronics Datasheet
- Qengineering/DHT22-Raspberry-Pi Github
- PiMyLifeUp Tutorial

This page titled 15.1: DHT22 Temperature and Humidity Sensor is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

15.2: DHT 22 - Streaming Temperature and Humidity Data

In this lesson, we will connect the DHT22 temperature and humidity sensor and stream readings to a Google Sheet. DHT22 is one of the most common IOT temperature and humidity sensors, it can measure temperature between -40°C and 80°C (±1°C) and humidity from 0% to 100% (±1%).

To start:

1. You can use the same Google Sheet that you've created last time (Upload Data to Google Sheet)
2. Add a new tab to your Google Sheet.
3. Name columns "DateTime", "Temperature, C" and "Humidity, %"
4. Rename the worksheet (the tab at the bottom) to "DHT22" by right-clicking on it and selecting "Rename". This way you can always find your data quickly.

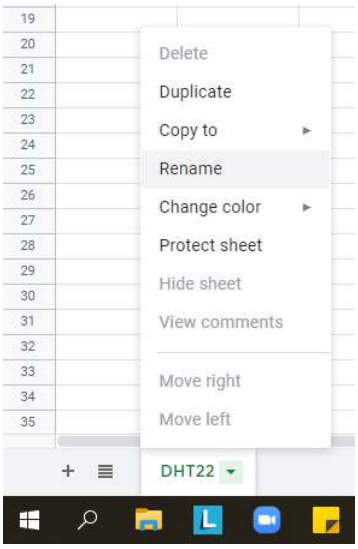


Figure 15.2.1: Rename the worksheet (CC0; E. Lisitsyna)

Fritzing Diagram

Any time you need to connect a sensor to your Raspberry Pi, you should start with searching the web for the pinout of that sensor. If you search for "DHT22 pinout" you will find many images similar to the one below.

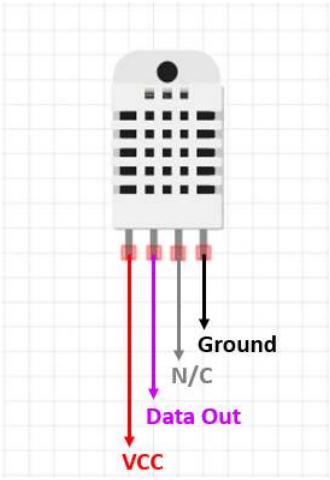


Figure 15.2.2: DHT22 Pinout (CC0; E.Lisitsyna)

As you can see, DHT22 has 4 pins: VCC, Data Out, N/C and Ground. N/C stands for “Not connected” and is not used for wiring. You can connect the DHT22 sensor directly to your Raspberri Pi's GPIO pins using a male-to-female jumper wires, or you can use a T-cobbler and a breadboard. We will be using a breadboard for this lesson.

- Pin#1 is connected to hot rail
- Pin#2 is connected to GPIO4 and a "pull-up" resistor
- Pin#4 is connected to ground rail

Remember, that we are not using pin#3. If your T-cobbler doesn't have built-in connections to hot and ground rails, you will need to run wires from the cobbler's 3.3 V to hot (red) rail and GND to ground (blue) rail. We will also add a 10 kΩ resistor between the data out pin and hot. Remember the GPIO pin has two values, on (3.3 V) or off, and this pulls it to high as the default position. Note, GPIO pins 1 through 8 of the Broadcom BCM2711B0 are pulled up by default (see page 77, the alternative GPIO assignments table of the datasheet), but we will run a pull-up resistor to ensure this. Here is the Fritzing connection diagram:

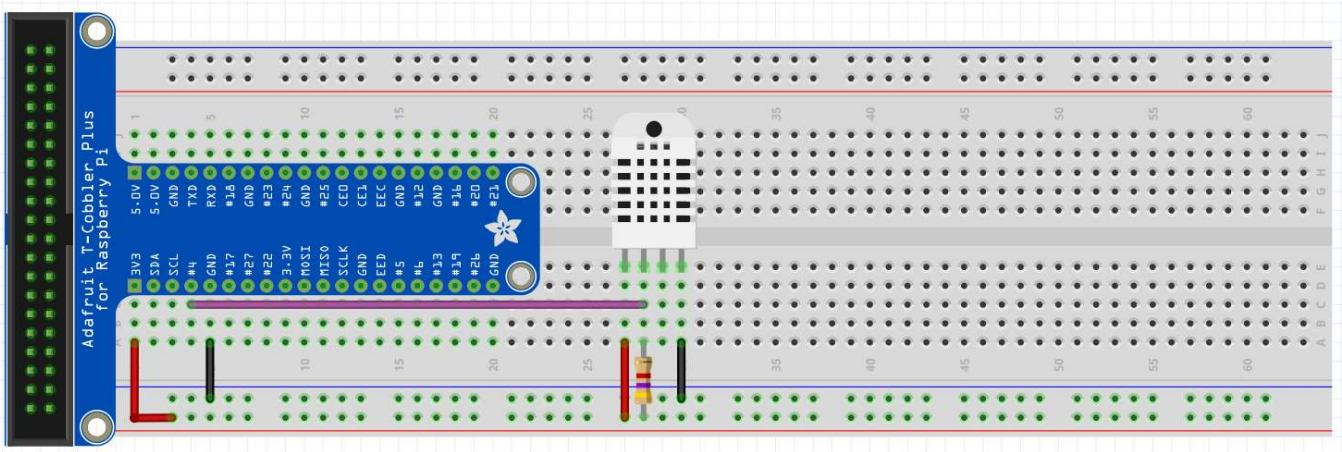


Figure 15.2.3: DHT22 connection diagram. You are using a 10 kΩ pull-up resistor (CC0; E.Lisitsyna)

Python Code

Let's test the sensor. We will need to install a few things first. Open the terminal and execute the following commands: Note, if you are using the Debian Bookworm or newer operating system you will need to install these packages in a virtual environment, and you should go to section 5.12.2 Thonny-Virtual Environments for instructions on how to set up the virtual environment on your Raspberry Pi.

```
pip3 install adafruit-circuitpython-dht
```

If the above code does not work you need to do this in a venv (virtual environment), where you define the path of pip3 in the virtual environment

```
/home/pcomp/Desktop/2024_pcomp/ASN9_DHT22/bin/pip3 install adafruit-circuitpython-dht
```

```
sudo apt-get install libgpiod2
```

We install the gspread module during one of the previous lessons, but if you are starting with a new Raspberry Pi you will need to install it again with the following command:

```
pip3 install gspread
```

remember you may need to give the path to pip3 in your venv

Now run the following program in Thonny:

```
#!/home/pcomp/Desktop/2024_pcomp/ASN9_DHT22/bin/python3
#note the above shbang goes to python in venv
```

```
#Import the modules
import time
import board
import adafruit_dht

#Declare the sensor and the GPIO pin it is connected to
dhtDevice = adafruit_dht.DHT22(board.D4, use_pulseio=False)

while True:
    try:
        # Print the values to the serial port
        tempC = dhtDevice.temperature
        humidity = dhtDevice.humidity
        print(f"Temp: {tempC} C    Humidity: {humidity}% ")
        time.sleep(10.0)  #pause for 10 seconds
    except RuntimeError:
        pass
```

? Exercise 15.2.1

Examine the code. What do you think is the purpose of the “try – except” block?

Answer

DHT22 can fail unexpectedly and return a Runtime Error, that looks like this:

```
>>> %Run DHT22test.py
  Traceback (most recent call last):
    File "/home/pi/1_PythonScripts/DHT22test.py", line 12, in <module>
      temperature = dhtDevice.temperature
    File "/usr/local/lib/python3.7/dist-packages/adafruit_dht.py", line 274, in temperature
      self.measure()
    File "/usr/local/lib/python3.7/dist-packages/adafruit_dht.py", line 230, in measure
      raise RuntimeError("A full buffer was not returned. Try again.")
  RuntimeError: A full buffer was not returned. Try again.
```

This would stop your code, and you would have to restart it manually. To prevent this from happening, we add the “try-except” block that basically tells our program to try to run this code and if the specified error occurs to pass (or ignore) it. This way the loop continues.

Be careful using the "try-except" block. Errors are usually signaling that something is wrong with your code or wiring, and you can miss that message if you are ignoring the errors you shouldn't be ignoring. Runtime Error is a known issue with the DHT22 sensors, so it is safe to use for this lesson.

You can learn more about Runtime Errors and how they are different from Syntax Errors here.

If you want, you can add a print statement to alert you that an error had occurred, so you can see how often it happens.

```
    except RuntimeError:
        print ("Runtime Error. Continue...")
        pass
```

Stream DHT22 Data to Google Sheets

```
#!/usr/bin/python3

import datetime
import time
import board
import adafruit_dht
import gspread
from google.auth.transport.requests import AuthorizedSession
from google.oauth2 import service_account
import requests

dhtDevice = adafruit_dht.DHT22(board.D4, use_pulseio=False)

#Access Google API
googleAPI = '/home/pi/1_PythonScripts/mydata-key.json' #NOTE: replace this with the p
scope = ['https://www.googleapis.com/auth/drive']
credentials = service_account.Credentials.from_service_account_file(googleAPI)
scopedCreds = credentials.with_scopes(scope)
gc = gspread.Client(auth=scopedCreds)
gc.session = AuthorizedSession(scopedCreds)
sheet = gc.open("DHT22_Data")
worksheet = sheet.worksheet("DHT22")

def appendSheet(tempC, humidity):
    dateTime = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    values =[dateTime, tempC, humidity]
    worksheet.insert_row(values, 2, value_input_option='RAW')
    print("Google Sheet Updated")

def getData():
    try:
        tempC = dhtDevice.temperature
        humidity = dhtDevice.humidity
        print(f"Temp: {tempC} C    Humidity: {humidity}% ")
        appendSheet(tempC, humidity)
    except RuntimeError:
        print ("Runtime Error. Continue...")
        pass

while True:
    getData()
    time.sleep(5.0)
```

Notice that in this code, we used the full path to the JSON key file. This is useful if you want to keep your JSON keys separate from the code files or schedule your script in cron.

Check your Google Sheet. You should see the DHT22 readings.

Google Sheet Dashboard

It is easy to set up a Dashboard in Google Sheets, which would be the landing page of the sheet (first tab) that has graphs of data that is being streamed to other tabs.

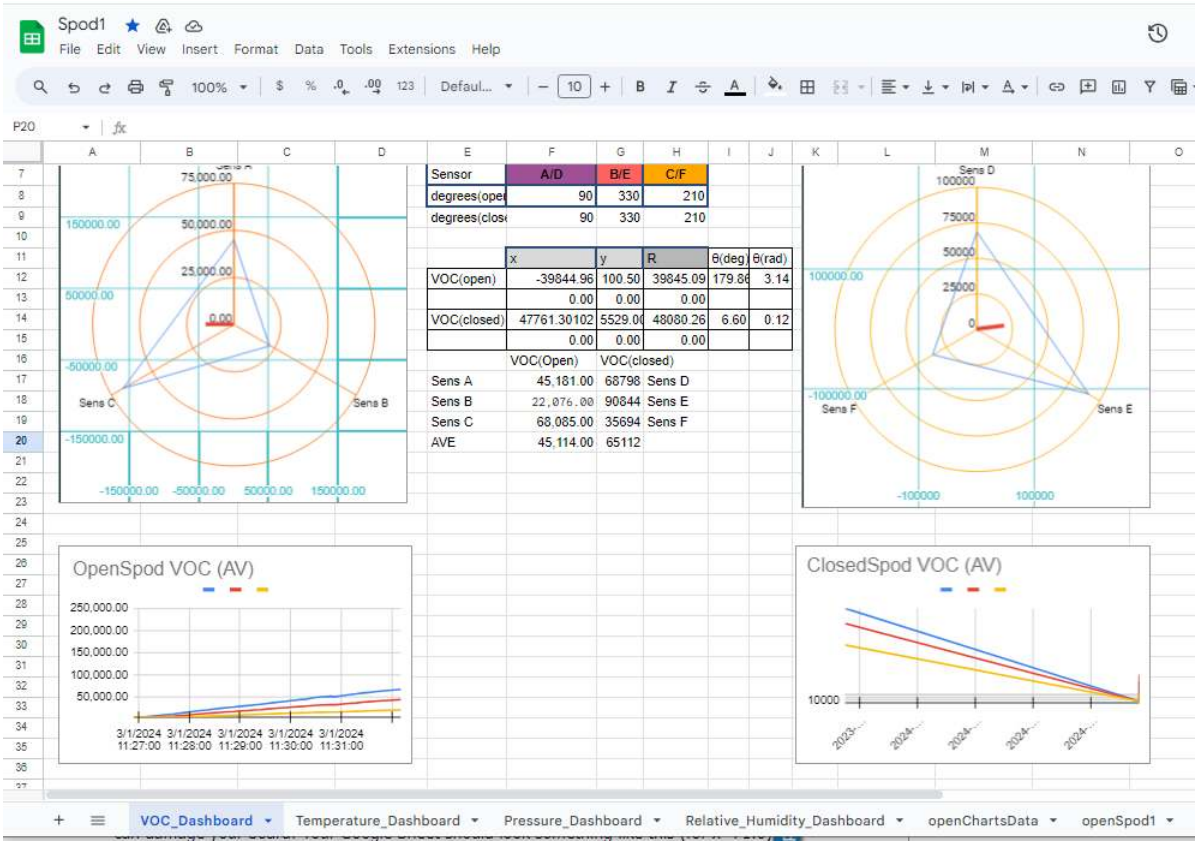


Figure 15.2.1: Copy and Paste Caption here. (Copyright; author via source)

Homework Assignment

1. Edit the code to stream the temperature in degrees Fahrenheit instead of Celsius. You will need to add a new variable (tempF) and convert from degrees Celsius to degrees Fahrenheit. Make sure that you change the heading in your Google Sheet from "Temperature, C" to "Temperature, F". Run your script and make sure it works and your Google Sheet is updated with the temperature in Fahrenheit.
2. Add a red LED to your circuit. You can use the RGB PCB board by connecting it to your breadboard.
3. Create an IF statement that will turn the LED on when the temperature is below x degrees Fahrenheit and turn it off if the temperature is equal or above x degrees Fahrenheit. Set x to about 1ºF above the temperature read from step 1. For example, if in step 1 your temperature was 70.6, make x = 71.6.
4. Add a new variable "ledStatus" that can be either "ON" or "OFF". Add a new column to your Google Sheet, call it "LED Status". Make your program stream all 4 variables (dateTime, tempF, humidity, ledStatus) to your Google Sheet.
5. You can wait for the temperature to change naturally, or you can place a hot or a cold object next to the sensor. Use common sense and don't put anything that can damage your board! Your Google Sheet should look something like this (for x=71.6):
6. Set up a Google Dashboard as the front page of your sheet (see Google Dashboard Tutorial). You want to stream the data to the sheet called DHT22 and show two graphs on the front page (Dashboard). The first graph will be Temp as a fct of time and the second humidity as a fct. of time.

	A	B	C	D
1	DateTime	Temperature, F	Humidity, %	LED Status
2	2022-04-04 16:49:09	70.0	51.8	ON
3	2022-04-04 16:49:03	70.4	50.9	ON
4	2022-04-04 16:48:57	70.7	51.8	ON
5	2022-04-04 16:48:51	70.9	51.8	ON
6	2022-04-04 16:48:46	71.1	51.0	ON
7	2022-04-04 16:48:40	71.4	52.0	ON
8	2022-04-04 16:48:34	71.6	51.1	OFF
9	2022-04-04 16:48:23	71.8	50.9	OFF
10	2022-04-04 16:48:17	71.9	50.2	OFF
11	2022-04-04 16:48:11	72.2	50.8	OFF
12				

Figure 15.2.4: Homework example. (CC0; E. Lisitsyna)

This page titled 15.2: DHT 22 - Streaming Temperature and Humidity Data is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

15.3: Writing to a CSV file (DHT22 data)

In this activity you will read temperature and humidity data from a DHT22 sensor and post them to a csv file along with the time of the reading. If you are not familiar with the DHT22 you will need to check the following two links, where we go over how it works, and how to set it up to collect data.

- DHT22: Temperature and Humidity Sensor - description on how DHT22 works
- DHT22: Streaming Temperature and Humidity Data - Instructions on how to set up the DHT 22 sensor

Checking DHT22 Setup

Run the following code in Thonny

```
#!/usr/bin/python3

#Import the modules
import time
import board
import adafruit_dht

#Declare the sensor and the GPIO pin it is connected to
dhtDevice = adafruit_dht.DHT22(board.D4, use_pulseio=False)

while True:
    try:
        # Print the values to the serial port
        tempC = dhtDevice.temperature
        humidity = dhtDevice.humidity
        print(f"Temp: {tempC} C    Humidity: {humidity}% ")
        time.sleep(10.0)  #pause for 10 seconds
    except RuntimeError:
        pass
```

or alternatively, type the following code into the command line (you will need to use the path to the file on your computer, and be sure the program is executable and has a shebang that identifies the interpreter

```
/home/pi/Programs//DHT22/DHT22_1.py
```

What this code is doing is printing the temperature and time every 2 seconds

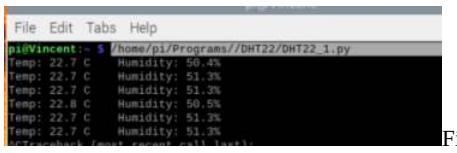


Figure 15.3.1: Copy and Paste Caption here. (Copyright; author via source)

Now that we know the sensor is reading data correctly we need to put the data into a file so that it can be used.

Writing to CSV File

CSV (comma separated variable) are one of the most common types of data files, and are the kinds of files Excel and Google Docs work with. These are not files that are read by python in the sense that they are a data type that python works with. But you can generate these (write) with the python inFile.write command, and in a latter exercise we will see how to read a CSV file by converting its content into a datatype that Python can work with, like a list.

Before running the following code save it in your folder for this exercise and then look at the contents of that folder (you can do this with the "ls" command, or using the desktop file manager). Then run this program and look again at the content of the folder. You should see a new file appear called "DHTtemperatures.csv". Right click on this and choose "text editor" (note, if you installed LibreOffice and click on the file it will open in LibreCalc, which is an open source program like Excel.

```
import datetime
from time import sleep
import board
import adafruit_dht

def getValues():
    dhtDevice = adafruit_dht.DHT22(board.D4, use_pulseio=False)
    for i in range(0,5):
        try:
            humidity = dhtDevice.humidity
            temperature = dhtDevice.temperature
        except RuntimeError as error:
            print(error.args[0])
            sleep(2.0)
            continue
        except Exception as error:
            dhtDevice.exit()
            raise error
    if humidity is not None and temperature is not None:
        return (humidity, temperature)

inFile= open("DHTtemperatures.csv","w")
inFile.write("time,humidity,temperature\n")

for x in range(0,10):
    humidity,temperature = getValues()
    now = datetime.datetime.now().strftime("%H:%M:%S")
    inFile.write('%s,%s,%s\n' %(now,humidity,temperature))
    sleep(2)
    #note the DHT22 does not read faster than this
    print(x)

inFile.close()
```

Questions

? Exercise 15.3.1

What is the role of the for loop in the function getValues()?

Answer

This is assigning humidity and temperature values to the variables humidity and temperature. If the value returned is not valid it retries 5 times. This is important in working with sensor readings as you need checks to be sure the data you are transferring is real data.

Exercise 15.3.2

What does the following line of code do?

inFile=open(DHTtemperatures.csv","w") do? If the file already exists, what happens to the data?

Answer

This creates the file DHTtemperatures.csv and clears out all data if there was data in the file.

Exercise 15.3.3

What does the following line of code do?

inFile.write("time,humidity,temperature\n")

Answer

It inserts into the CSV file the headings of time, humidity and temperature. The \n generates a new line.

Exercise 15.3.4

Where in the code are the variables assigned values, what is the variable for time and why is its assignment executed after the variables humidity and temperature are assigned values?

Answer

They are assigned in the for loop. The variables humidity and temperature are assigned values by calling the function getValues(). The variable for time is now and it is called after the function getValues() has returned a value. If you look at the function getValues() you see a loop with up to 5 iterations, depending on if a valid value is returned. So the time delay could be 0,2,4,6,8 seconds. By getting the time after the value is returned we are certain the time is correct, and so our plot will be correct, although the data values may not be evenly spaced. If 5 bad values are returned you have a serious problem and may need to check your wiring,

Exercise 15.3.5

Change the range of the loop from 10 to 5 iterations and run the code. Before looking at the file, how many values to you expect to be generated?

Answer

You will see 5 data values plus the title. You are writing a new file ('w'), not overwriting the original data

Exercise 15.3.6

Change the "w" of the infile= open( ) statement to an "a" and change the 5 in the range of the loop back to 10. Run the code. What happens, how many lines of data do you see, and what does "a" stand for?

Answer

"a" stands for append, and you are now adding data to the file, and will have 15 rows of data. Note, since you ran the program twice you generated the title twice.

temporary stuff

```
    Spod1Backup = open("openspod1backup.csv","a")
    #Spod1Backup.write("\n")
    Spod1Backup.write(str(values)[1:-1])
    Spod1Backup.write("\n")
    Spod1Backup.close()
```

15.3: Writing to a CSV file (DHT22 data) is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

15.4: Reading a csv File

convert a csv to a list

```
#make a list from csv
import csv
file=open("names_elements1.csv")
element_list=list(csv.reader(file, delimiter=","))
file.close()
#print(element_list)
flat_list=[item for sublist in element_list for item in sublist]
print(flat_list)
```

15.4: Reading a csv File is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

CHAPTER OVERVIEW

16: ADC and Sensors

Topic hierarchy

16.1: SPI
16.2: ADC and MQ gas sensors

16: ADC and Sensors is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

16.1: SPI

Serial Per

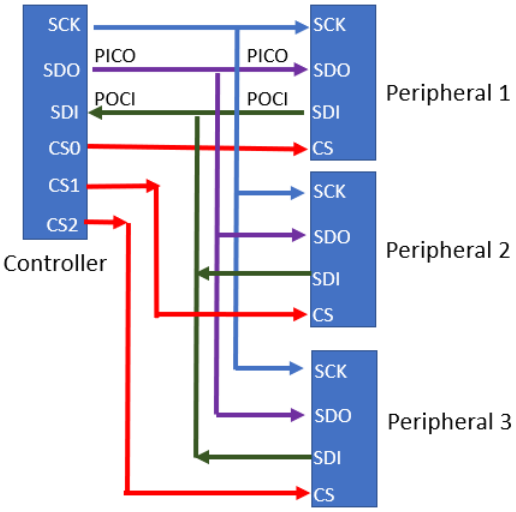


Figure 16.1.1: Data lines for SPI interface with three peripheral devices. Note you still need to provide power and a ground to the perpherials. (Belford CC-BY)

Note

The Open Source Hardware Association has put forth a resolution to redefine the SPI Signal Names. In many devices and data sheets you will see the older names, and you will need to be familiar with both. In essence the old terminology used Master/Slave designation and the new one uses Controller/Peripheral (device) designations.

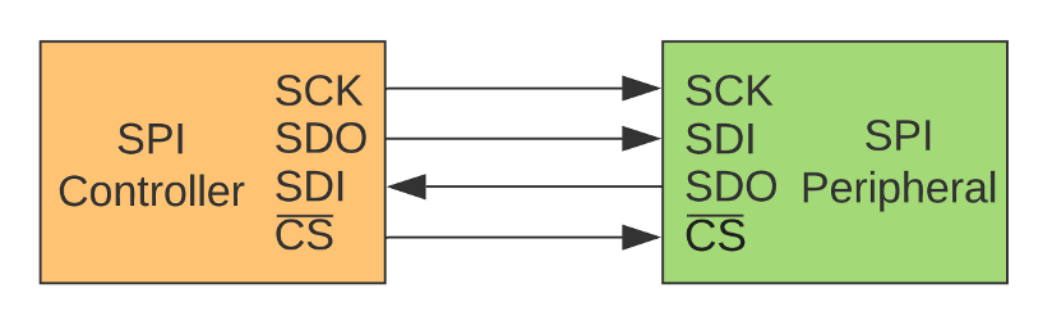


Figure 16.1.2: New signal designations. (SparkFun CC 0.0)

New Signal Names

- SDO - Serial Data Out. An output signal sent to another SPI device
- SDI - Serial Data In. An input signal received from another SPI device.
- CS - Chip Select. Picks which peripheral device is active (old SS).
- PICO - Peripheral In Controller Out. Signal being sent from controller to peripheral (old MOSI).

- POCI - Peripheral Out Controller In. Signal sent from peripheral to controller (old MISO).
- SDIO - Serial Data In/Out. Bidirectional serial signal.

Deprecated Signal Names

- MOSI - Master Out Slave In
- MISO - Master In Slave Out
- SS - Slave Select
- MOMI - Master Out Master In
- SOSI - Slave Out Slave In

Unchanged Signal Names

- SCK - Serial Clock. Controller generated clock signal for the common bus.

solving problems, as long as you get the **proper values a**





- Circuit Basics
- SparkFun: Serial Peripheral Interfaces

This page titled 16.1: SPI is shared under a not declared license and was authored, remixed, and/or curated by Robert Belford.

16.2: ADC and MQ gas sensors

Task 1: View the following YouTube videos on Analog to Digital Converters

The explanation of the analog vs digital signals



MCP3008 Integrated Circuit Chip

The Raspberry Pi computer does not have the ability to read analog signals, therefore we need to change them to digital signals. This is accomplished using an MCP3008 integrated circuit chip. This is a 10 bit chip (1024 steps) from 0 to 1 in the range of voltages that we are using. It can monitor eight analog inputs using the SPI communication protocol we used in the last activity.

This chip has 16 pins, and are numbered according to the following diagram. **Pin one is indicated on the chip with a little "dimple" on the top** of the chip (indicated with the arrow in Figure 16.2.2). The eight analog inputs are found on the side with the dimple, and all other connections are done on the other side.

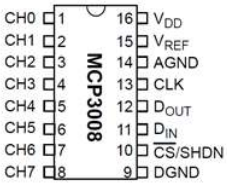


Figure 16.2.1: MCP3008 integrated circuit chip

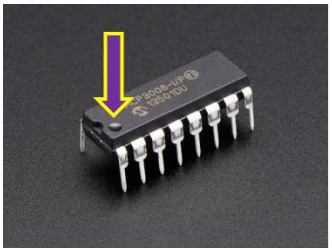


Figure 16.2.2: Pin one is indicated with a little "dimple"

V_{DD} connected to 3.3V and provides the power for the chip. D_{GND} (digital ground) is the ground for the chip itself.

The four SPI data pins are:

- D_{OUT} (Data out from MCP3008) and connects to POCI (Peripheral Out Controller In - old MISO)
- D_{IN} (Data in from Raspberry Pi) and connects to PICO (Peripheral In Controller Out - old MOSI)
- CLK (Clock Pin) and connects to the serial clock (CLK)
- CS (Chip Select) and connect to CE0

Finally, the last two pins are the V_{REF} (Reference Voltage) and the AGND (Analog Ground), which are used to change the scale. We are using 3.3V for our scale.

Light Dependent Resistors

Light dependent resistors, also known as photoresistors, photocells or photo-conductive cell, work by reducing resistance when the light hits the sensitive surface. In the dark, the resistance is at the highest point (up to several megaohms), while in full light the resistance goes down to just a few hundred ohms.



Figure 16.2.3: Light dependent resistor (CC BY-SA 3.0, © Nevit Dilmen via Wikipedia)

Task 2A: Start to build a circuit with an MCP3008 ADC chip

Using your breadboard with a T-cobbler, place the MCP3008 chip on the board. Connect the pins using the male to male leads in your kit. In the connection diagram shown in Figure 16.2.3, the MCP3008 is upside-down for easier connection. Pin 1 of the MCP3008 is marked in the diagram. You can identify it on your MCP3008 by a small circular indentation.

MCP3008:

- **Pin 16 (V_{DD})** connects to your **positive rail**
- **Pin 15 (V_{REF})** connects to your **positive rail**
- **Pin 14 (AGND)** connects to your **ground rail**
- **Pin 13 (CLK)** connects to **SCLK** on your T-cobbler (Board pin 23)
- **Pin 12(D_{OUT})** connects to **POCI** (MISO) on your T-cobbler (Board pin 21)
- **Pin 11(D_{IN})** connects to **PICO** (MOSI) on your T-cobbler (Board pin 19)
- **Pin 10 (CS)** connects to **CE0** on your T-cobbler (Board pin 24)
- **Pin 9(D_{GND})** connects to your **ground rail**

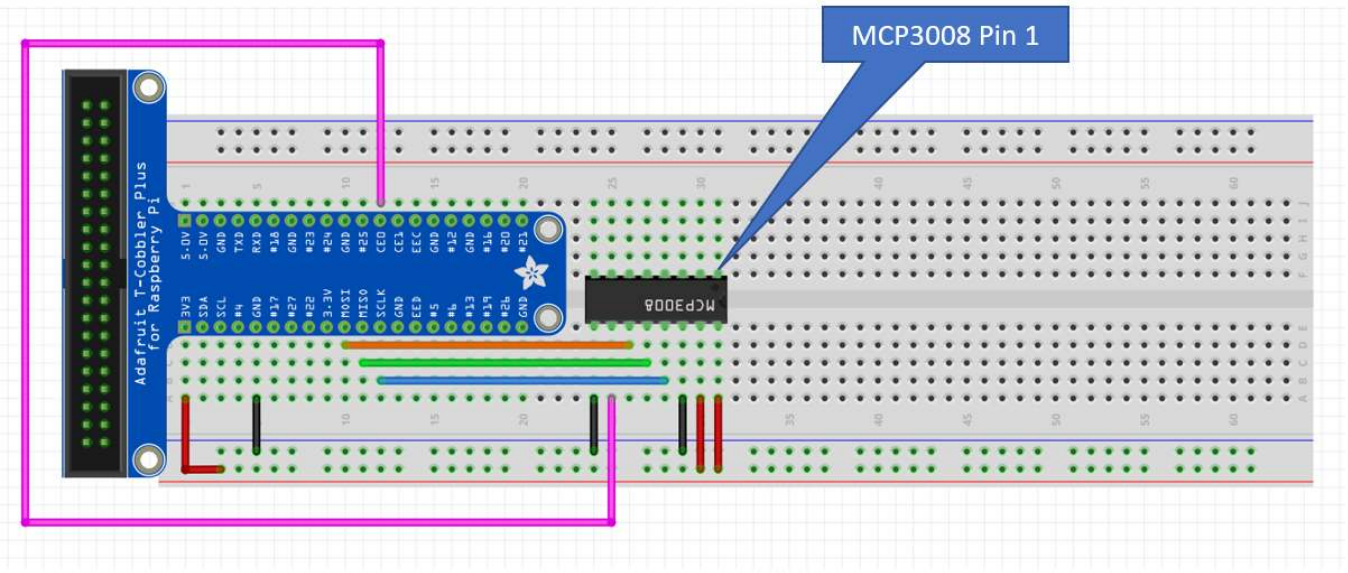


Figure 16.2.4: MCP3008 connection diagram (CC0, E. Lisitsyna)

Task 2B: Connect a light sensor to your MCP3008 ADC chip

Connect one leg of the LDR to the hot rail and the other leg (signal) to pin 1 of the MCP3008 (remember that pin 1 is marked with a round indentation). Now you need to add a 10kΩ resistor between signal and ground rail.

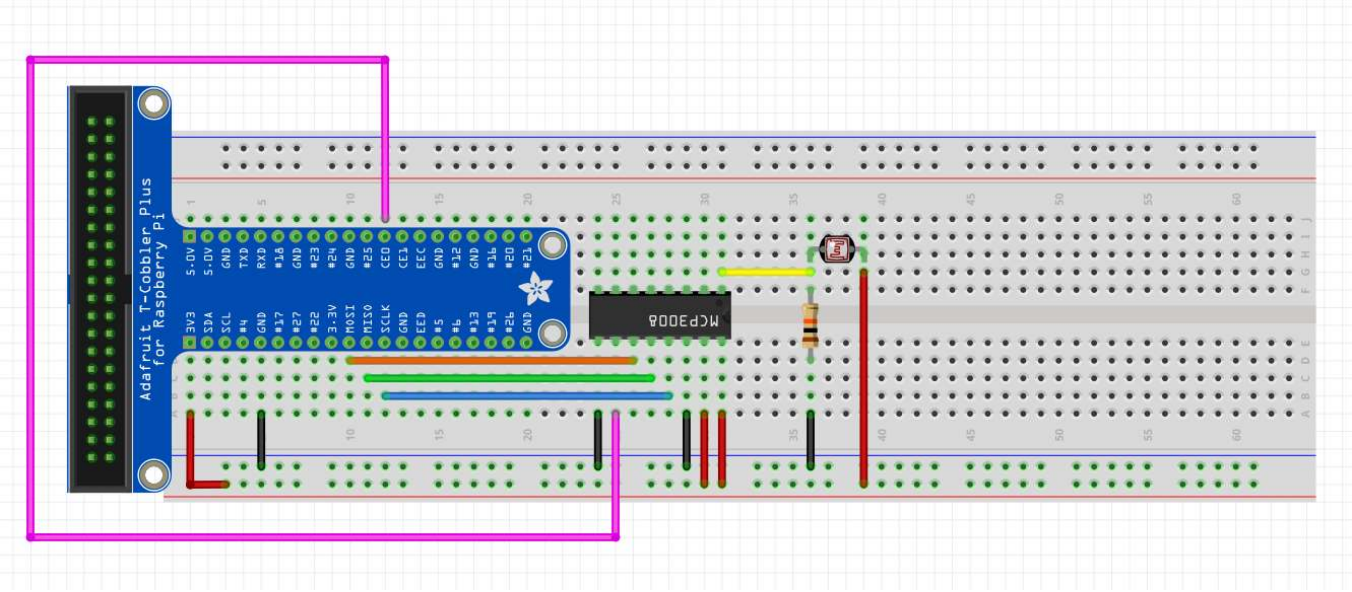


Figure 16.2.5: LDR connection diagram (CC0, E. Lisitsyna)

Task 3: Measure light values

We are using the gpiozero package and you should go to readthedocs - source code for gpiozero.spi_devices for further information

Caution

4/1/2024 (This is not an April Fools day message). There are issues with installing libraries from PyPi on the latest Debian operating systems. If you get an error with the spiDev module, try installing raspi-gpio-0.1.9 in your venv

https://pypi.org/project/raspi-gpio/

If you get an lgpio error, try thi

wget https://github.com/Gadgetoid/PY_LGPI...0.2.2.0.tar.gz

pip install lgpio-0.2.2.0.tar.gz

For this task, enter and execute the following program with your Thonny IDE on your Raspberry Pi. The value argument for MCP3008 returns a read value from the device scaled to a value between 0 and 1.

Python Program 1

```
from gpiozero import MCP3008
from time import sleep

adc0 = MCP3008(channel=0)

while True:
    print(adc0.value)
    sleep(1)
```

Try covering the LDR or use a flashlight. If the value has changed as the result, then your circuit is working correctly, and you can proceed to the next task. If the value stays the same, or you are getting an error, go back and check the wiring. Remember to pay attention to the location of pin one.

Task 4: Introduction to Metal Oxide Sensors

The following short video illustrates how Metal oxide sensors can detect gases.



MQ-2 Gas Sensor

After you made sure that MCP3008 is connected correctly, you can start working on the next circuit. For this project, we will use the MQ-2 sensor, that can measure the concentration and/or presence of combustible gases in the air, such as hydrogen, LPG (liquid petroleum gas), methane, carbon monoxide, alcohol, smoke, and propane. This sensor is traditionally used to help detect gas leaks in many family and industrial practices.

MQ2 sensor (and other metal oxide sensors) has a built-in heating element that requires an initial burn-in, which takes 24 - 48 hours. You can start getting readings from the sensor right away, but they won't be accurate.

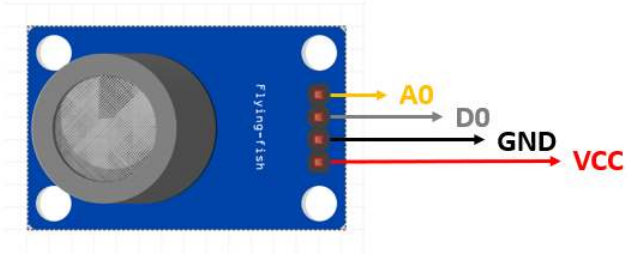


Figure 16.2.5: MQ-2 Pinout (CC0; E. Lisitsyna)

Logic Level Converter

The heating element needs at least 5V to operate properly, but Raspberry Pi can only read a signal at 3.3V. We are introducing a bi-directional logic level converter to this circuit to solve this problem.

The logic level converter has 6 pins on each side. Four of them are used to step down from 5V to 3.3V and to step up from 3.3V to 5V. Both sides need to be connected to hot and ground rails. The side marked HV corresponds to higher voltage (5V) and the side labeled LV corresponds to the lower voltage (3.3V).

Fritzing desn't have the exact model we are using, but it can be downloaded here and added to Fritzing. Read this tutorial to learn how to add parts and bins to Fritzing.

Task 5A: Get Readings from MQ-2 Sensor

You can continue working with the same circuit. Since we already know that MCP3008 is connected right, we can replace the LDR with the gas sensor and add a logic converter.

Supplies

1. MQ Sensor
2. msp 3008 ADC
3. 3.3 to 5 V logic leveler
4. breadboard, Pi, Cobler and jumper wires
5. 10 K ohm resistor

Before you begin, make sure you connected cobbler's 5.0V to the hot rail on the opposite side from 3.3V rail. Connect cobblers GND pin to the ground rail (same side as 5.0V). Refer to the connection diagram below.

MQ-2 Sensor:

- **VCC (voltage)** pin connects to the **5.0V hot rail**
- **GND (ground)** pin connects to the **ground rail**
- **D0 (digital out) is not connected**
- **A0 (analog out)** connects to **channel 1 of the level converter**

Logic Level Converter:

- **HV** connects to the **5.0V hot rail**
- **GND** (on the HV side) connects to the **ground rail** (on the 5.0V side)
- **LV** connects to the **3.3V hot rail**
- **GND** (on the LV side) connects to the **ground rail** (on the 3.3V side)

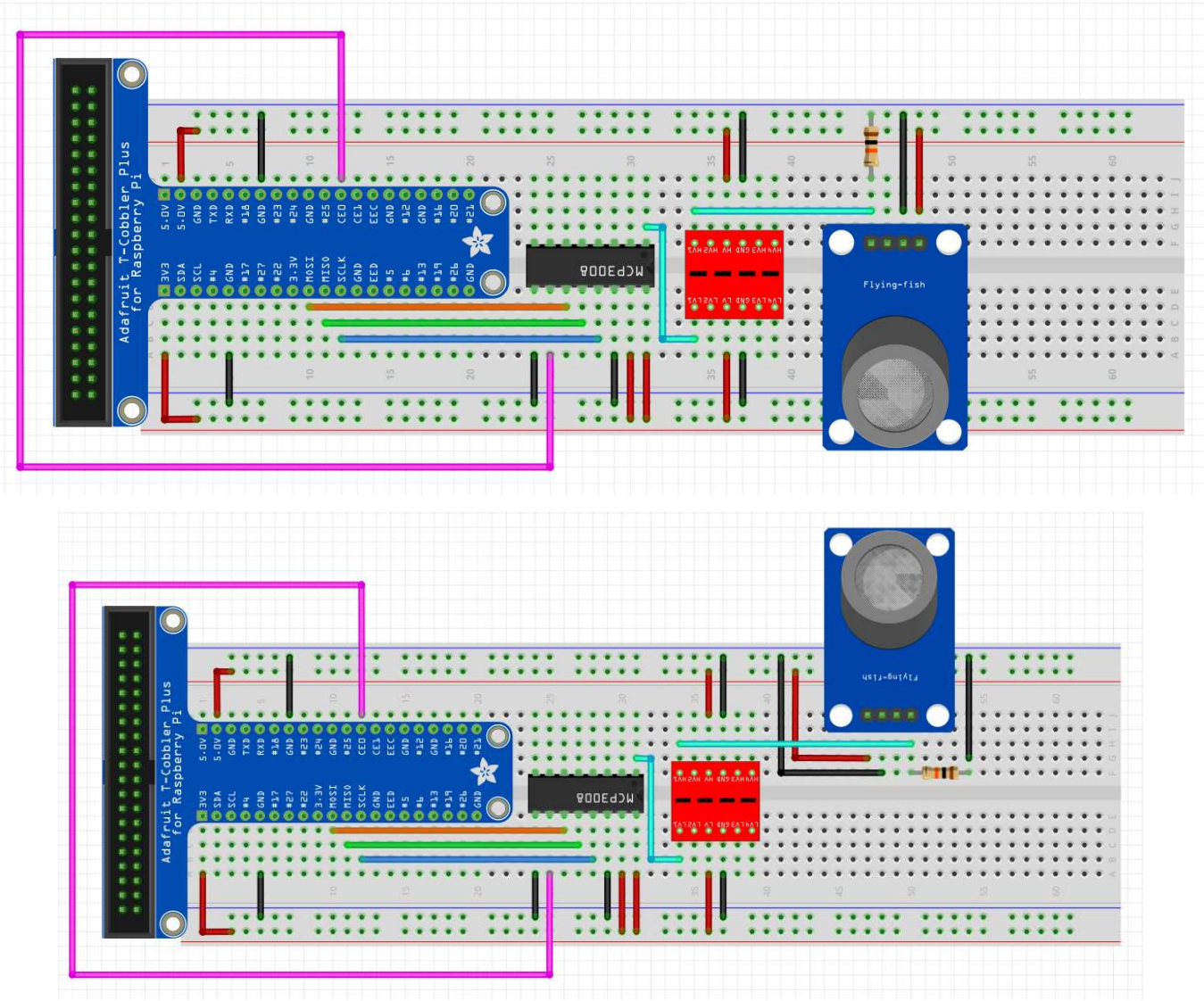


Figure 16.2.6: MQ-2 connection diagram (CC0; E. Lisitsyna)

Task 5B: Run the code

Note: in the code below, you will see two hard-coded values (LPGm and LPGb). Those values were calculated using the graph in the datasheet and Ohm's Law. If you are interested in the calculations, you can read this tutorial for the MQ-4 sensor. The whole process is described in great detail and can be applied to any of the gasses measured with the MQ sensors.

Run the code in Thonny.

Python Program 2

Python Program 2

```
#!/usr/bin/python3

import time
import math
import RPi.GPIO as GPIO
from gpiozero import MCP3008

def getR0():
    #set loop values to 0:
    value=0
    x=0

    for x in range (0,500): #get 500 consecutive readings
        value = value + adc0.raw_value #add all readings together
        x=x+1 #increase loop counter by 1

    sensorValue = value/500 #calculate the adc average value from 500 readings
    sensorVolt = ((sensorValue / 1023.0) * 3.3) #convert raw adc values to voltage

    #calculate R0 (sensor resistance in clean air)
    RSair = ((3.3*10)/sensorVolt)-10 #10 (load resistance or RL) value is taken from the
    R0=RSair/9.7 #from the datasheet Rs/R0 for clean air is constant, see graph (~9.7ppm
    print ("Calculated R0: {:.5}".format (R0))
    return R0

adc0 = MCP3008(channel=0)
R0 = getR0()

while True:
    #LPM and LPGb values are calculated from the datasheet using Ohm's Law
    LPGm = -0.47 #slope (m)
    LPGb = 1.31 #y-intercept (b)

    sensorValue = adc0.raw_value
    sensorVolt=((sensorValue / 1023.0) * 3.3)
    RSgas = ((3.3*10)/sensorVolt)-10
    ratio = (RSgas/R0)

    LPGLog = (math.log10(ratio)-LPGb)/LPGm
    LPGppm = math.pow(10, LPGLog)
    LPGperc=LPGppm/10000

    print ("LPG concentration: {:.3}ppm".format(LPGppm))
    print ("LPG percent: {:.3}%".format (LPGperc))
    time.sleep(5)
```

Critical Thinking Questions

1. The code above will print out the concentration of liquid petroleum gas in ppm and percentage. Do you think those values are accurate? Explain.
2. In your opinion, can the values provided by the MQ-2 sensor be used in science?
3. Imagine that you were asked to set up a calibration procedure for the MQ-2 sensor. Describe the steps.

Homework Assignment

Using the code from Task 5B set up a data stream to a Google Sheet and add a scatter plot to your dashboard. Leave the Pi running for 24 hours and take a reading every 30 minutes (use cron for scheduling this task). As a result, you should see a nice graph showing the change in readings during the burn-in process.

16.2: ADC and MQ gas sensors is shared under a not declared license and was authored, remixed, and/or curated by LibreTexts.

Index

Glossary

Sample Word 1 | Sample Definition 1

Detailed Licensing

Overview

Title: Spring 2024: Physical Computing Class

Webpages: 162

All licenses found:

- Undeclared: 100% (162 pages)

By Page

- Spring 2024: Physical Computing Class - *Undeclared*
  - Front Matter - *Undeclared*
    - TitlePage - *Undeclared*
    - InfoPage - *Undeclared*
    - Table of Contents - *Undeclared*
    - Licensing - *Undeclared*
  - 1: Preface - *Undeclared*
    - 1.1: Introduction To Physical Computing and the Internet of Science Things - *Undeclared*
    - 1.2: Introduction to your LibreTextbook - *Undeclared*
  - 2: Assignments - *Undeclared*
  - 3: Introduction and General Resources - *Undeclared*
    - 3.1: Required Resources - *Undeclared*
    - 3.2: Suggested Items - *Undeclared*
    - 3.3: Hypothes.is - *Undeclared*
    - 3.4: Programming and Flowcharts - *Undeclared*
    - 3.5: Pseudocode and Flowchart Conventions - *Undeclared*
    - 3.6: Functions - *Undeclared*
    - 3.7: Computer Interfaces - *Undeclared*
    - 3.8: Conda Package Management System - *Undeclared*
    - 3.9: Thonny - *Undeclared*
    - 3.10: Visual Studio Code - *Undeclared*
    - 3.11: Git and GitHub - *Undeclared*
    - 3.12: GitHub - *Undeclared*
    - 3.13: GitHub Desktop - *Undeclared*
    - 3.14: Online Resources - *Undeclared*
    - 3.15: Git - *Undeclared*
  - 4: Python Data Types - *Undeclared*
    - 4.1: Introduction to data - *Undeclared*
    - 4.2: Data Types - *Undeclared*
    - 4.3: String (atomic) - *Undeclared*
    - 4.4: Integers (atomic) - *Undeclared*
    - 4.5: Float (atomic) - *Undeclared*
    - 4.6: Boolean (atomic) - *Undeclared*
    - 4.7: String (container) - *Undeclared*
    - 4.8: List (container) - *Undeclared*
    - 4.9: Dictionary (container) - *Undeclared*
    - 4.10: Tuple (container) - *Undeclared*
    - 4.11: 3.11 Set (container) - *Undeclared*
  - 5: Data Operators - *Undeclared*
    - 5.1: Introduction - *Undeclared*
    - 5.2: Arithmetic Operators and Numbers - *Undeclared*
    - 5.3: Arithmetic Operators and Strings - *Undeclared*
    - 5.4: Assignment Operators - *Undeclared*
    - 5.5: Comparison Operators - *Undeclared*
    - 5.6: Logical Operators - *Undeclared*
    - 5.7: Identity Operators - *Undeclared*
    - 5.8: Membership Operators - *Undeclared*
  - 6: Introduction to Linux - *Undeclared*
    - 6.1: What is an Operating System? - *Undeclared*
    - 6.2: Introduction to Linux - *Undeclared*
    - 6.3: Git BASH - *Undeclared*
    - 6.4: Linux Commands - *Undeclared*
  - 7: Starting your Raspberry Pi - *Undeclared*
    - 7.1: SD Card Formatter - *Undeclared*
    - 7.2: Raspberry Pi Imager - *Undeclared*
    - 7.3: Install VNC Viewer - *Undeclared*
    - 7.4: ESD- The Silent Assassin - *Undeclared*
    - 7.5: Installing RPi OS - *Undeclared*
    - 7.6: Starting your Pi the First Time - *Undeclared*
    - 7.7: Updating Your Pi - *Undeclared*
    - 7.8: Registering your Pi with UALR IOT - *Undeclared*
    - 7.9: VNC Viewer - Headless Mode - *Undeclared*
    - 7.10: Adjusting Remote Monitor Settings - *Undeclared*
    - 7.11: Pi Desktop - *Undeclared*
    - 7.12: Pi Terminal - *Undeclared*
    - 7.13: Keyboard Shortcuts - *Undeclared*
    - 7.14: Google email account for class - *Undeclared*
    - 7.15: Sending email without attachment - *Undeclared*
    - 7.16: Sending email with attachment - *Undeclared*
    - 7.17: Find IP Address - *Undeclared*
    - 7.18: email IP address - *Undeclared*
    - 7.19: Cron and Crontab - *Undeclared*
    - 7.20: First Cron Job - *Undeclared*

- 8: Python Functions, Packages and Formatting - *Undeclared*
  - 8.1: Functions - *Undeclared*
  - 8.2: Introduction to Functions - *Undeclared*
  - 8.3: Built-In Functions - *Undeclared*
  - 8.4: Standard Library Functions - *Undeclared*
  - 8.5: Third Party Library Functions - *Undeclared*
  - 8.6: User-Defined Functions - *Undeclared*
  - 8.7: User Created Package - *Undeclared*
  - 8.8: Print Formatting Overview - *Undeclared*
  - 8.9: Modulus Method - *Undeclared*
  - 8.10: 3.format() method - *Undeclared*
  - 8.11: f-string formatting - *Undeclared*
  - 8.12: Subscripts and Superscripts - *Undeclared*
- 9: Anatomy of a Pi and Camera Activity - *Undeclared*
  - 9.1: History of Pi - *Undeclared*
  - 9.2: Monitoring Your Pi - *Undeclared*
  - 9.3: Pi 4 Overview - *Undeclared*
  - 9.4: SOC (System on Chip) - *Undeclared*
  - 9.5: Random Access Memory (RAM) - *Undeclared*
  - 9.6: Non-Volatile (Primary) Data Storage - *Undeclared*
  - 9.7: LEDs - *Undeclared*
  - 9.8: 40 Pin GPIO - *Undeclared*
  - 9.9: UART (Universal Asnynchronous Receiver-Transmitter) - *Undeclared*
  - 9.10: I2C (Inter-Integrated Circuit) - *Undeclared*
  - 9.11: SPI (Serial Peripheral Interface) - *Undeclared*
  - 9.12: Networking Overview - *Undeclared*
  - 9.13: Radio Chip - *Undeclared*
  - 9.14: Ethernet - *Undeclared*
  - 9.15: USB C- Power Port - *Undeclared*
  - 9.16: USB 2 and 3 - *Undeclared*
  - 9.17: AV Jack - *Undeclared*
  - 9.18: Micro HDMI 2.0 - *Undeclared*
  - 9.19: CSI (Camera Serial Interface) - *Undeclared*
  - 9.20: DSI (Display Serial Interface) - *Undeclared*
  - 9.21: CSI Camera - *Undeclared*
  - 9.22: Libcamera - *Undeclared*
  - 9.23: 3-Python and Libcamera - *Undeclared*
- 10: Logic and Conditional Statements - *Undeclared*
  - 10.1: Introduction - *Undeclared*
  - 10.2: if Statement - *Undeclared*
  - 10.3: if-else Statement - *Undeclared*
  - 10.4: if-elif-else Statement - *Undeclared*
  - 10.5: Compound if/elif Statements - *Undeclared*
  - 10.6: Nested if Statements - *Undeclared*
  - 10.7: Loops - An Introduction - *Undeclared*
  - 10.8: for Loops - *Undeclared*
  - 10.9: While Loops - *Undeclared*
  - 10.10: Nested Loops - *Undeclared*
  - 10.11: Loop Control Structures - *Undeclared*
- 11: Introduction to Electronics and First Circuit - *Undeclared*
  - 11.1: Electricity and Units - *Undeclared*
  - 11.2: Introduction to Electric Circuits - *Undeclared*
  - 11.3: Circuit Diagrams - *Undeclared*
  - 11.4: Diagrams.net Electric Circuit Diagrams - *Undeclared*
  - 11.5: Fritzing - *Undeclared*
  - 11.6: Conductors - *Undeclared*
  - 11.7: Circuit Boards - *Undeclared*
  - 11.8: Resistors - *Undeclared*
  - 11.9: Diodes - *Undeclared*
  - 11.10: LED (Light Emitting Diodes) - *Undeclared*
  - 11.11: Multimeter - *Undeclared*
  - 11.12: GPIO Output (LEDs) - First Circuit - *Undeclared*
- 12: GPIO Input (Buttons) - Second Circuit - *Undeclared*
  - 12.1: Second Circuit - *Undeclared*
- 13: Serial Communication - *Undeclared*
  - 13.1: Serial vs. Parallel - *Undeclared*
  - 13.2: Serial and Morse Code - *Undeclared*
  - 13.3: UART - *Undeclared*
  - 13.4: I2C - *Undeclared*
  - 13.5: SPI - *Undeclared*
- 14: APIs - *Undeclared*
  - 14.1: Computer Interfaces - *Undeclared*
  - 14.2: Creating a Google Service Account - *Undeclared*
  - 14.3: Enabling Google Sheets and Obtaining a Key - *Undeclared*
  - 14.4: PEP 668 and Debian Bookworm - *Undeclared*
  - 14.5: Thonny-Virtual Environments - *Undeclared*
  - 14.6: Upload Data to Google Sheet - *Undeclared*
- 15: DHT22 Temperature and Humidity Sensor - *Undeclared*
  - 15.1: DHT22 Temperature and Humidity Sensor - *Undeclared*
  - 15.2: DHT 22 - Streaming Temperature and Humidity Data - *Undeclared*
  - 15.3: Writing to a CSV file (DHT22 data) - *Undeclared*
  - 15.4: Reading a csv File - *Undeclared*
- 16: ADC and Sensors - *Undeclared*
  - 16.1: SPI - *Undeclared*
  - 16.2: ADC and MQ gas sensors - *Undeclared*
- Back Matter - *Undeclared*
  - Index - *Undeclared*
  - Glossary - *Undeclared*
  - Detailed Licensing - *Undeclared*