CSCi 4907
Introduction to IoT and Edge Computing Applications

Prof. Kartik Bulusu, CS Dept.

Week 3 [02/03/2023]	• Setting up the Edge Lab • Some more programming constructs • Recap on RPi skeleton code • Bash script to run RPi-python code	• Raspberry Pi programming [Blinking LEDs using Thonny] • Setting up a Python virtual environment [venv] - Sandboxing your code • Blinking LEDs on boot

```
git clone git@github.com:gwu-csci3907/Spring2024.git

git clone https://github.com/gwu-csci3907/Spring2024.git
```

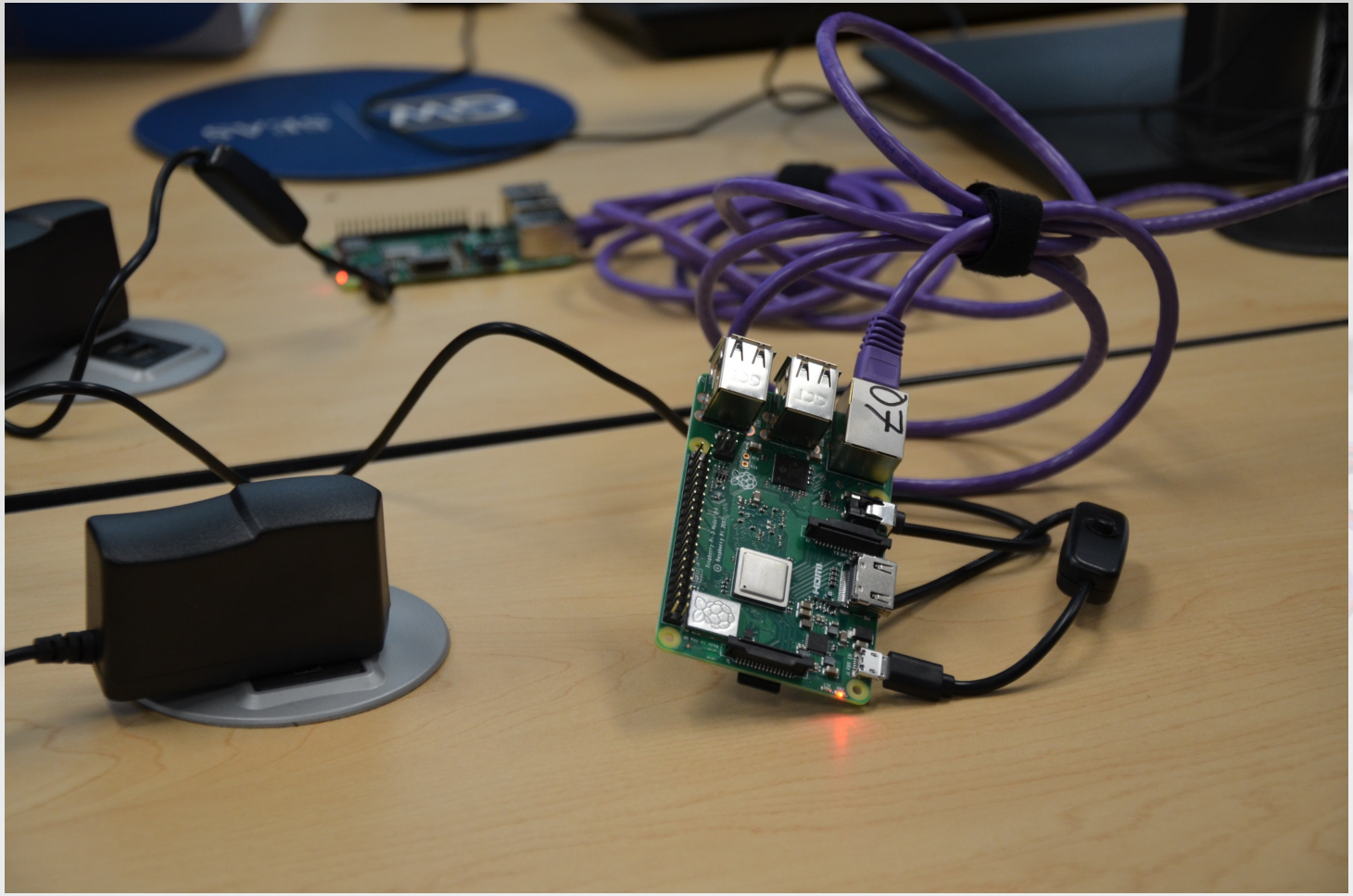
GW School of Engineering & Applied Science
THE GEORGE WASHINGTON UNIVERSITY

Spring 2024

Photo: Kartik Bulusu

Set up lab the Edge-lab

STEP [1]:
Connect the RPis to each desk power outlet as shown



- Make sure there is a microSD card installed in the RPi
- Connect the RPi using the microUSB cable provided
- Connect the purple colored ethernet cables specifically for RPi connections
- LEDs on the RPi will start blinking indicating that it is booting up

STEP [2]:
Access the RPi in the Edge-lab

2.1 Open up remote desktop connection (using the VNC server)

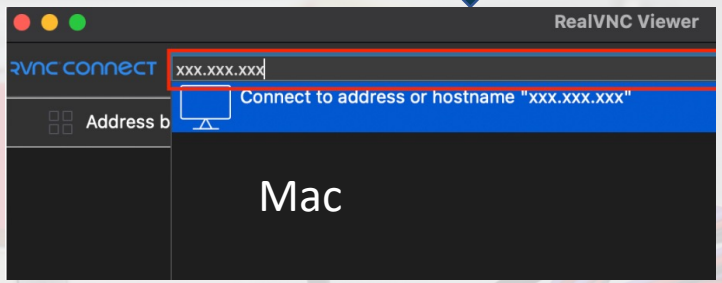
Each RPi has unique alpha-numeric name (e.g., Pi07, Pi152 etc)

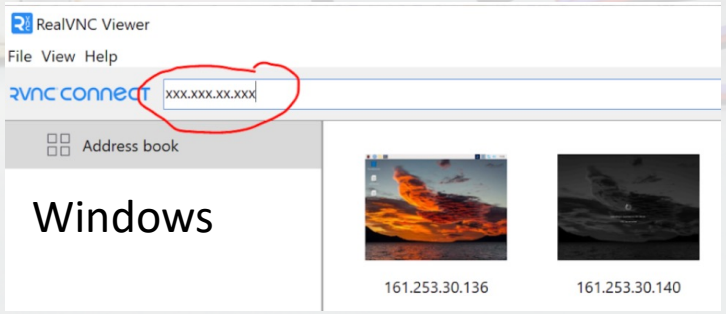
- Locate the Pi-name and the IP address on the <128.164.139.xx>

OR

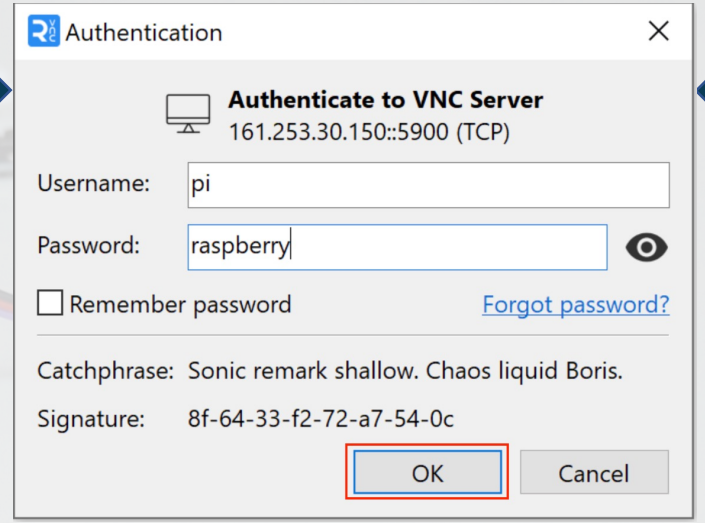
Each RPi connected using an ethernet cable directly to your laptops

- raspberrypi.local





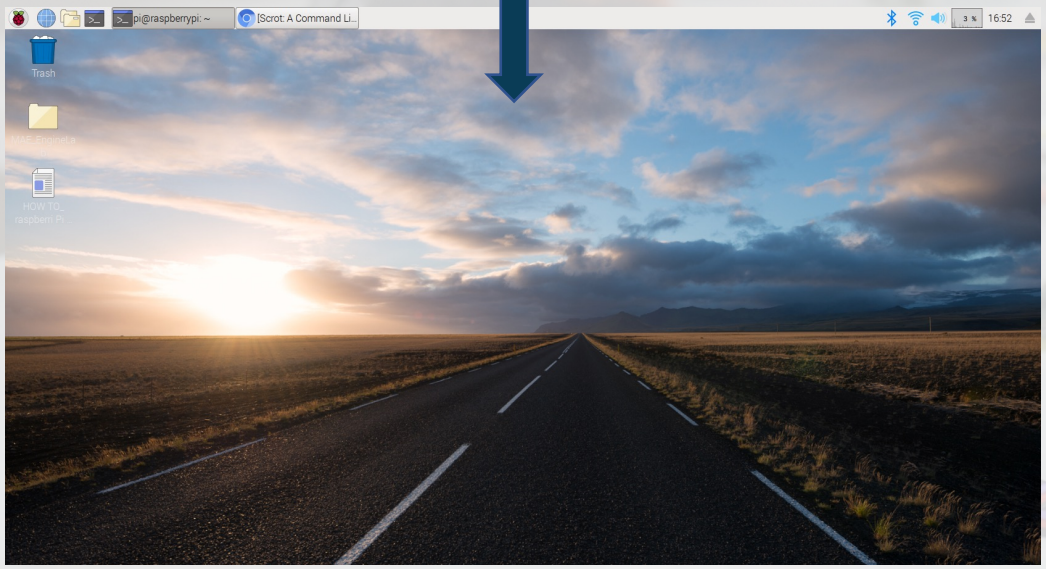
2.2 Once you are connected you will
- See Authentication box below
- Type in the Username and Password



STEP [3]:

Now that you accessed the RPi…

You will see a screen like the one shown below

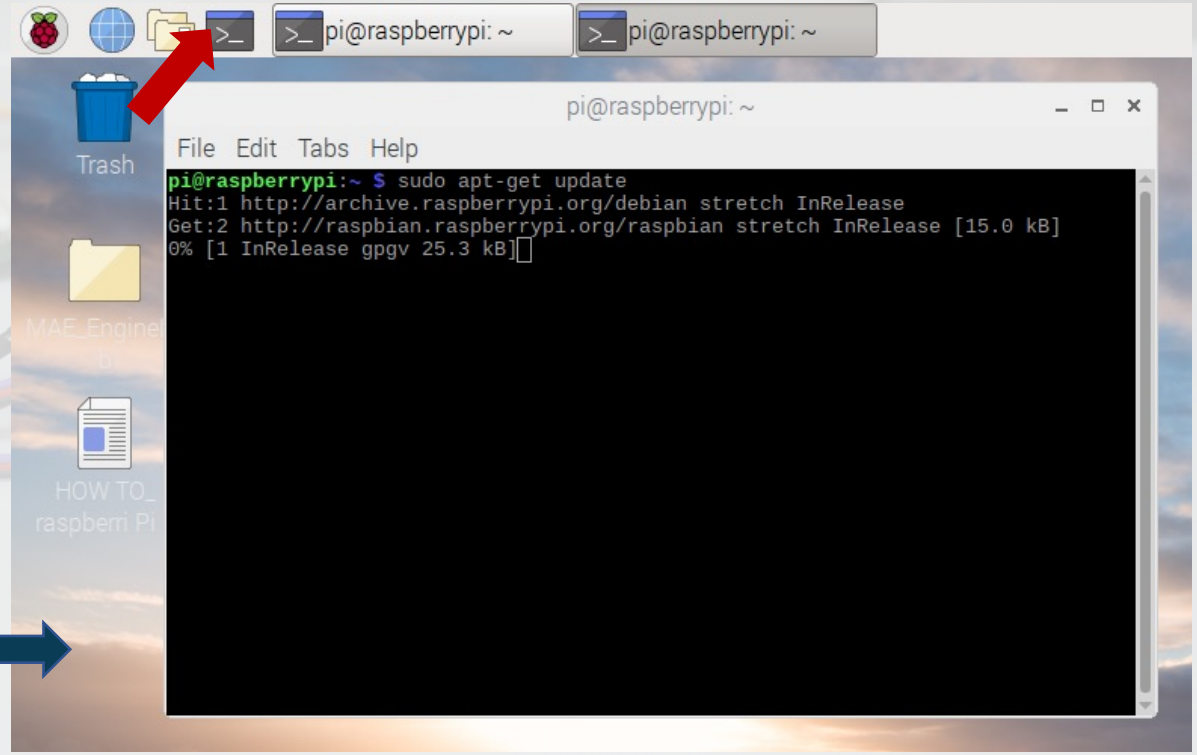


3.2 Testing is complete when you get to this step.

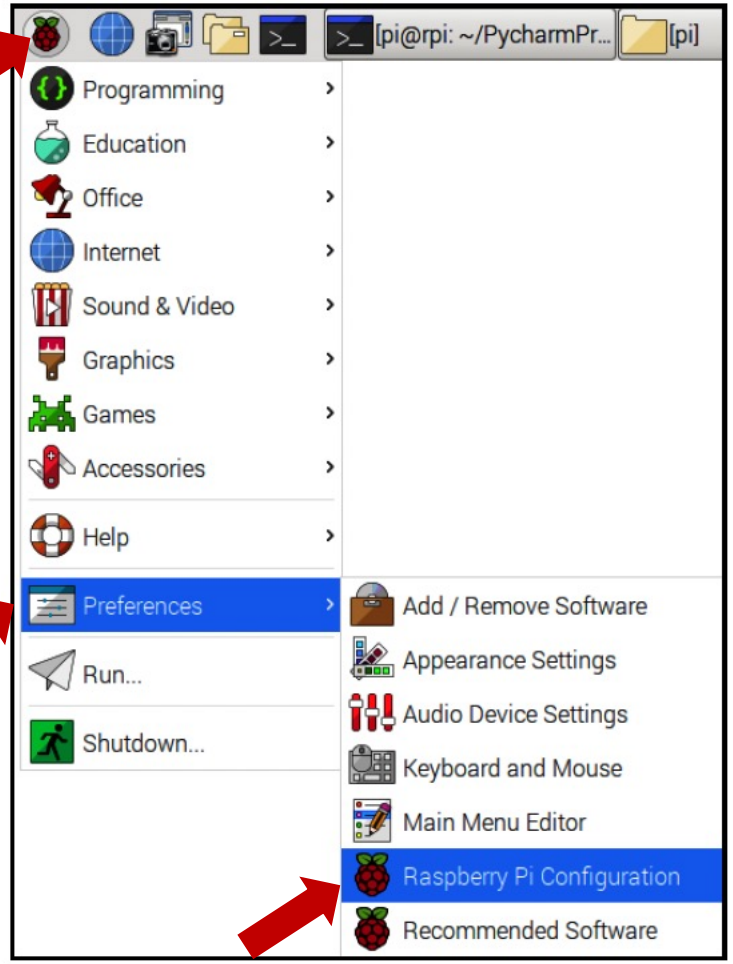
- Students should get the RPis to this step before the laboratory modules begin.

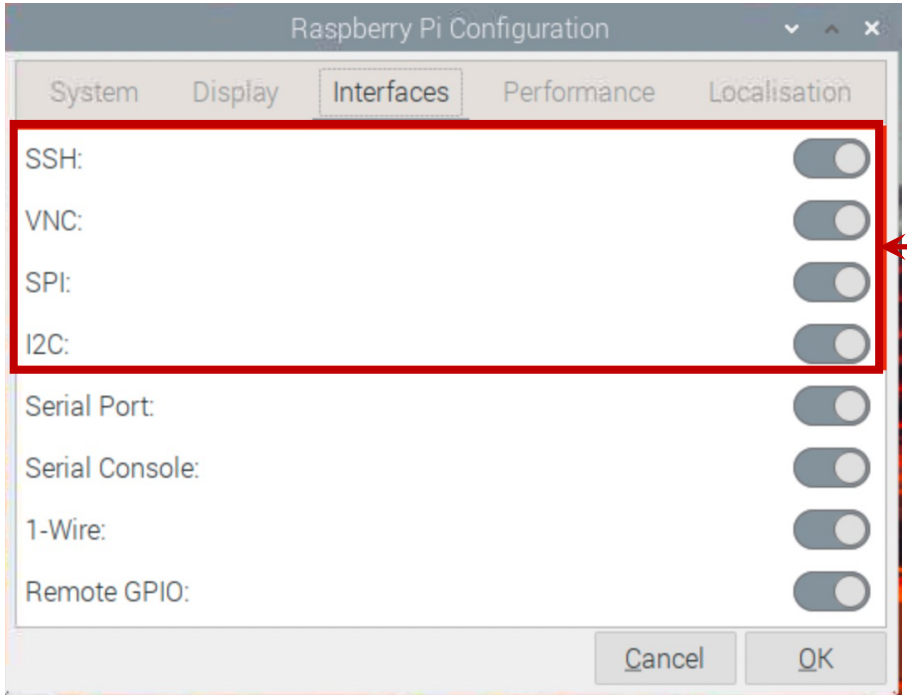
3.1 Click on terminal(shown with a red arrow below)

- At the prompt type: **sudo apt-get update**
- Wait for the updates to complete
- Then type: **sudo apt-get upgrade**
- If you get the following prompt
  - **Do you want to continue [Y/n]**
  - **Type: y**
  - **And hit "Enter" on your keyboard and let the upgrades complete**



Couple of checks and balances

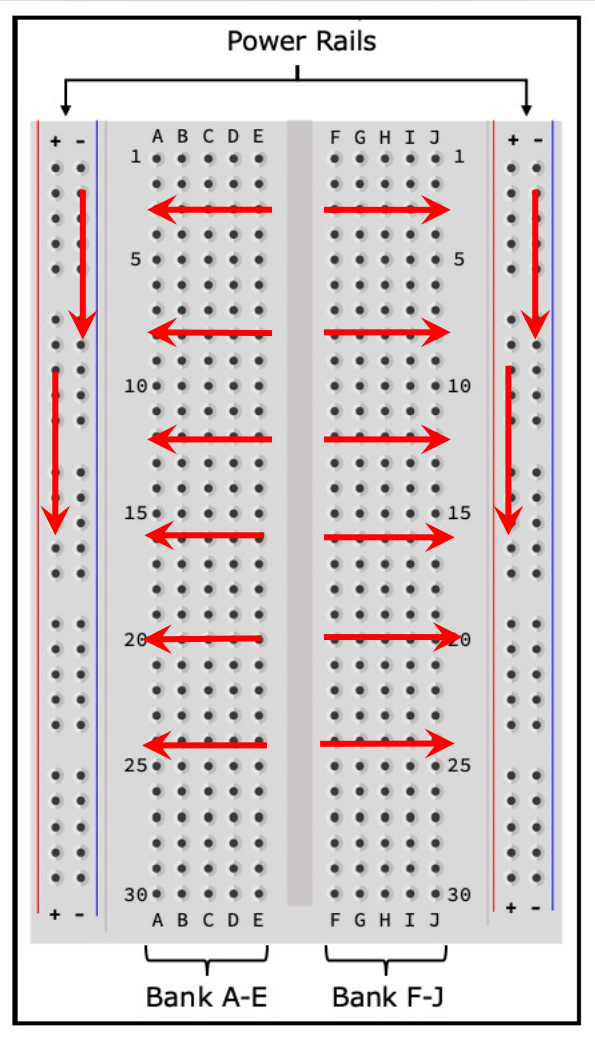


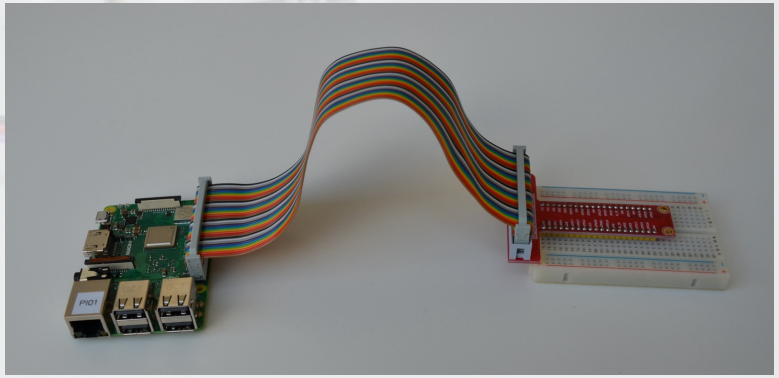


Make sure these are enabled

Connect the Raspberry Pi Model 3 B+ (RPi) to a bread board

Sources of the breadboard image:
Gary Smart, Practical Python Programming for IoT

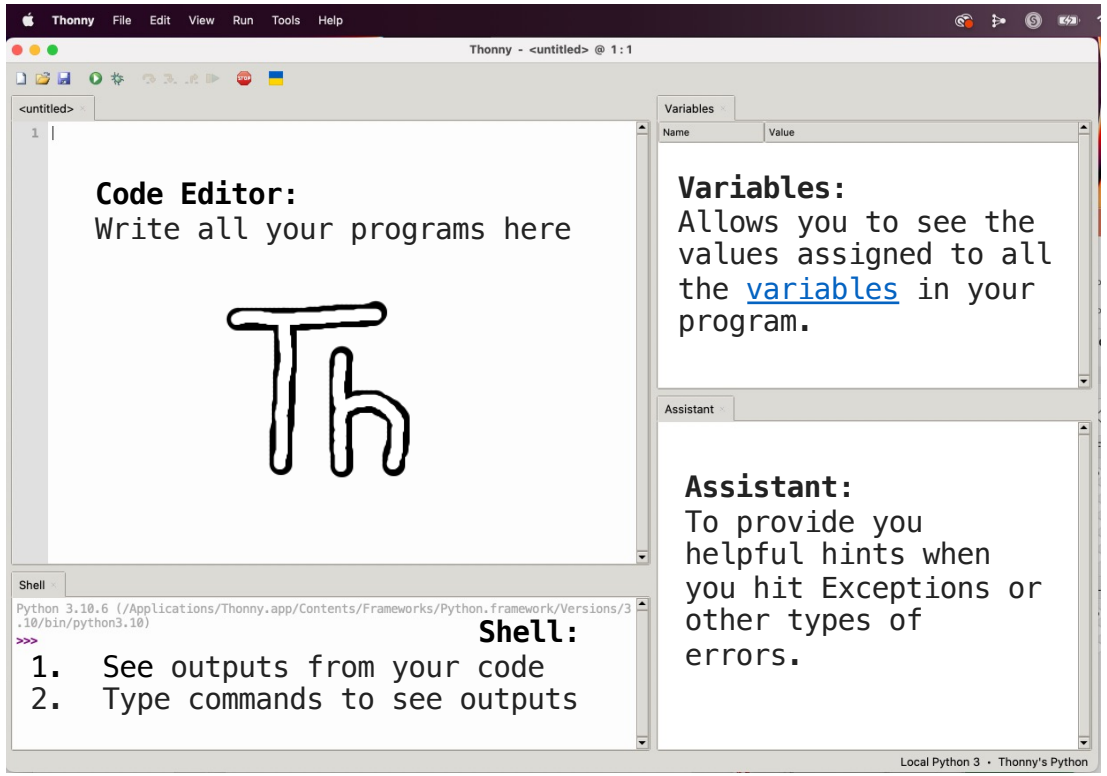




Let’s get to know some programming paradigms

Thonny integrated development environment (IDE)

Sources:
Thonny (IDE): https://en.wikipedia.org/wiki/Thonny
Thonny: The Beginner-Friendly Python Editor: https://realpython.com/python-thonny/



Allows you to run the code, i.e., "Do what I told you do do!".

Allows you to open a file that already exists on your computer

Create a new file

Save your code. Press this early and often.

Allows you to debug your code.
A bug is another name for a problem.

The arrow icons allow you to run your programs step by step.
These icons are used after you press the bug icon

The resume icon allows you to return to play mode from debug mode.

The stop icon allows you to stop running your code

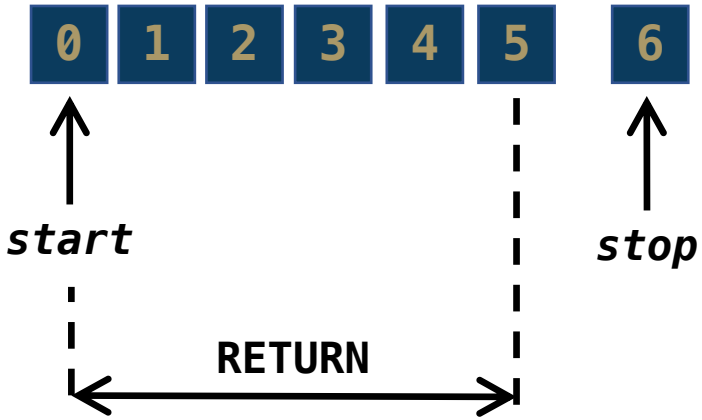
Built-in function range()

Python's range() function
- returns a sequence of numbers
- works only with integers

start:	at the value (default = 0)
step:	up or down at the increment value (default = 1)
stop:	at the value but not including it

range(stop)

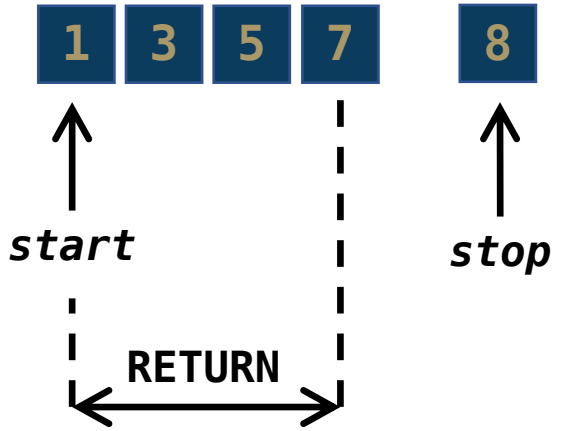
```
>>> range(6)
```



range(start, stop)

range(*start*, *stop*, *step*)

```
>>> range(1,8,2)
```

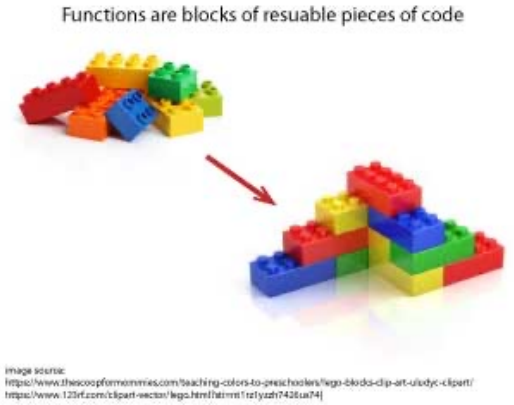


range()
- can be utilized in (for) loops
- to specify a range to iterate or do repetitions

Demos

Syntax and Skeleton of a user-defined `function`

```
def name(parameters):

        statement
        statement
        . . .

        return value
```



Function name: Identifier by which it is called in the program

(Optional) **Arguments:** values passed to the function

Function Declaration: Starts with "def" that is not indented

```
def func_name(parameters):
```

Colon; Don't miss it!

Indentation: Tab or 4 spaces for each statement

```
    statement
    statement
    . . .

    return value
```

Body: Statements executed each time a function is called

(Optional) **return value:** Can end function call and send data back to the main program

Function definition

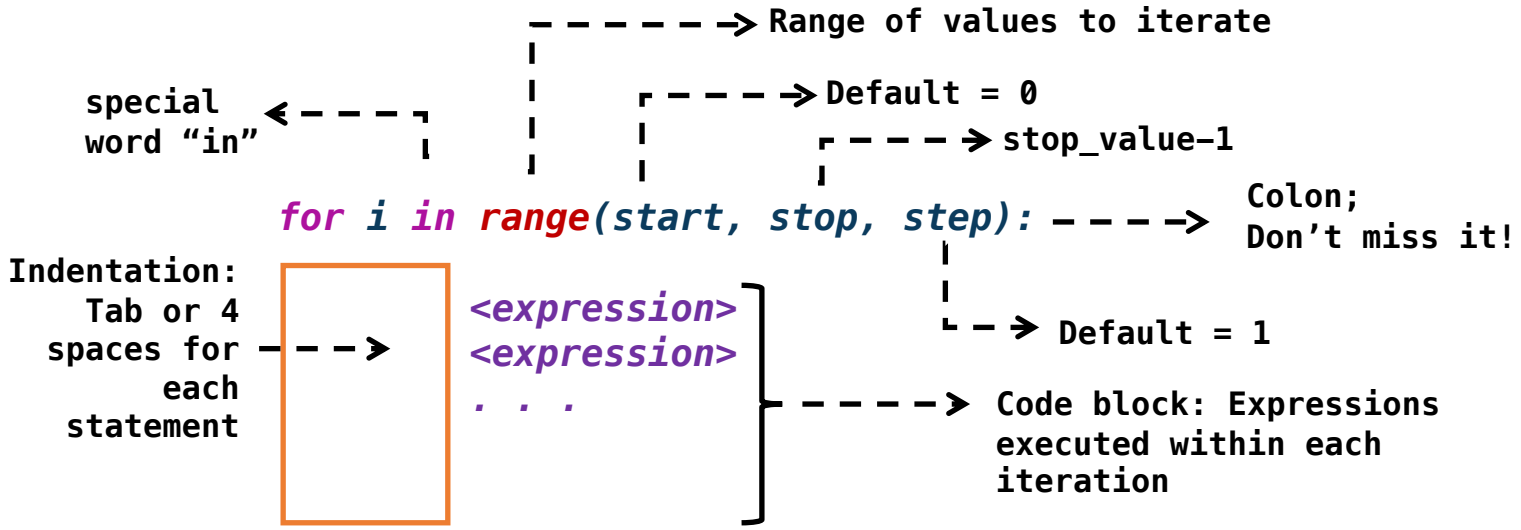
```
func_name()
```

Function call

Skeleton of the for-loop

```
for i in range(start, stop, step):

        <expression>
        <expression>
        . . .
```

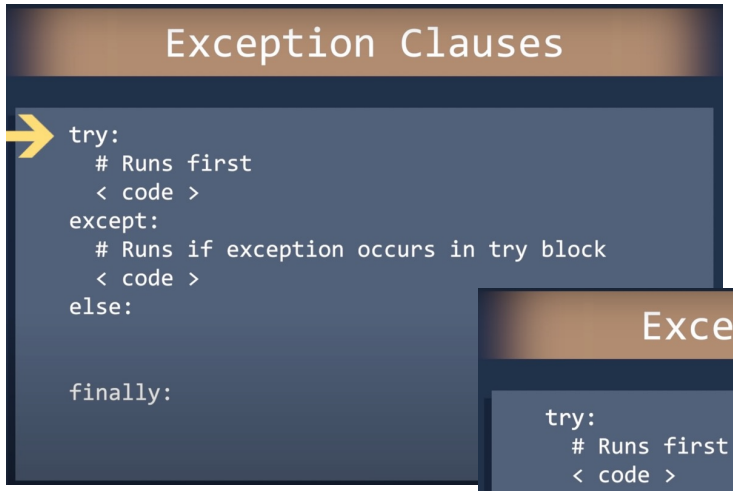
Demos

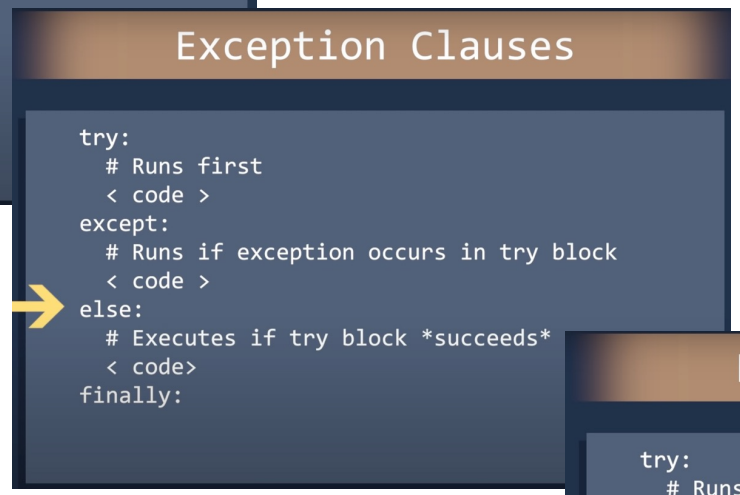


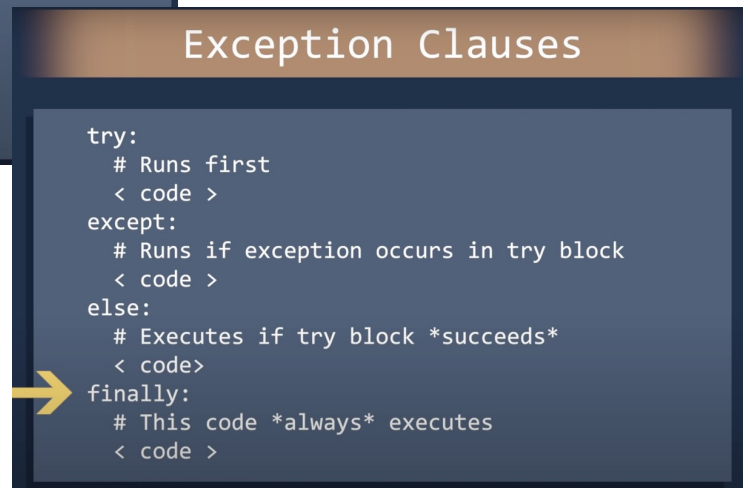
Sources:
Exceptions in Python || Python Tutorial || Learn Python Programming: https://youtu.be/nlCKrKGHSSk
Python Try Except: https://www.w3schools.com/python/python_try_except.asp

Skeleton of
try:
except:
else:
finally:

```
try:
        statement
        function
except KeyboardInterrupt:
        statement
        function
else:
        statement
        function
finally:
        statement
        function
```







The `try` block lets you test a block of code for errors.
The `except` block lets you handle the error.
The `else` block lets you execute code when there is no error.
The `finally` block lets you execute code, regardless of the result of the try- and except blocks.

Skeleton of the updated Python program written for Raspberry Pi

```
import library1 as name1
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)

GPIO.setup(12, GPIO.OUT)

def function_name(arguments):
        statement
        statement
        . . .
        return value

if __name__ == "__main__":
        try:
                function_name1()
        except KeyboardInterrupt:
                function_name2()
```

Import libraries that are relevant for interaction with the Raspberry Pi hardware such as GPIO pins, camera ports etc.

Set up GPIO pins as data outputs or inputs for sensors and actuators

Create user-defined functions to modularize your code and make it easy to work.

Create user-defined functions to release the GPIO pins

Create entry point into the program and pull all functions in

Create a keyboard-interrupt exception clause

Let’s blink some LEDs

Preliminaries:

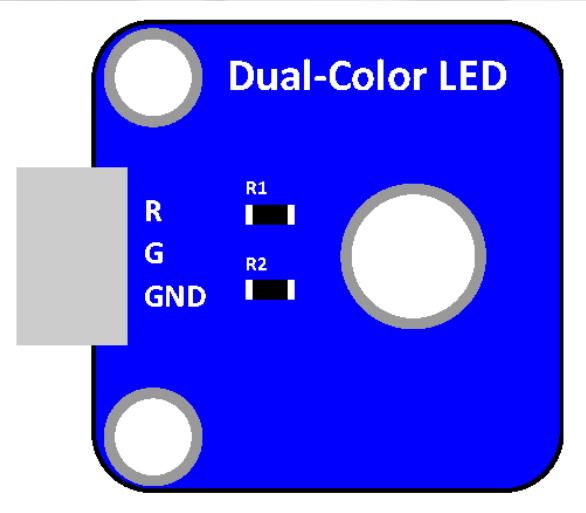
1. git-clone the repo preferably on the desktop

```
git clone git@github.com:gwu-csci3907/Spring2024.git
git clone https://github.com/gwu-csci3907/Spring2024.git
```

2. I have the files on a USB in case there is a slowdown!

Photo: Kartik Bulusu

Know your Light Emitting Diode (LED)



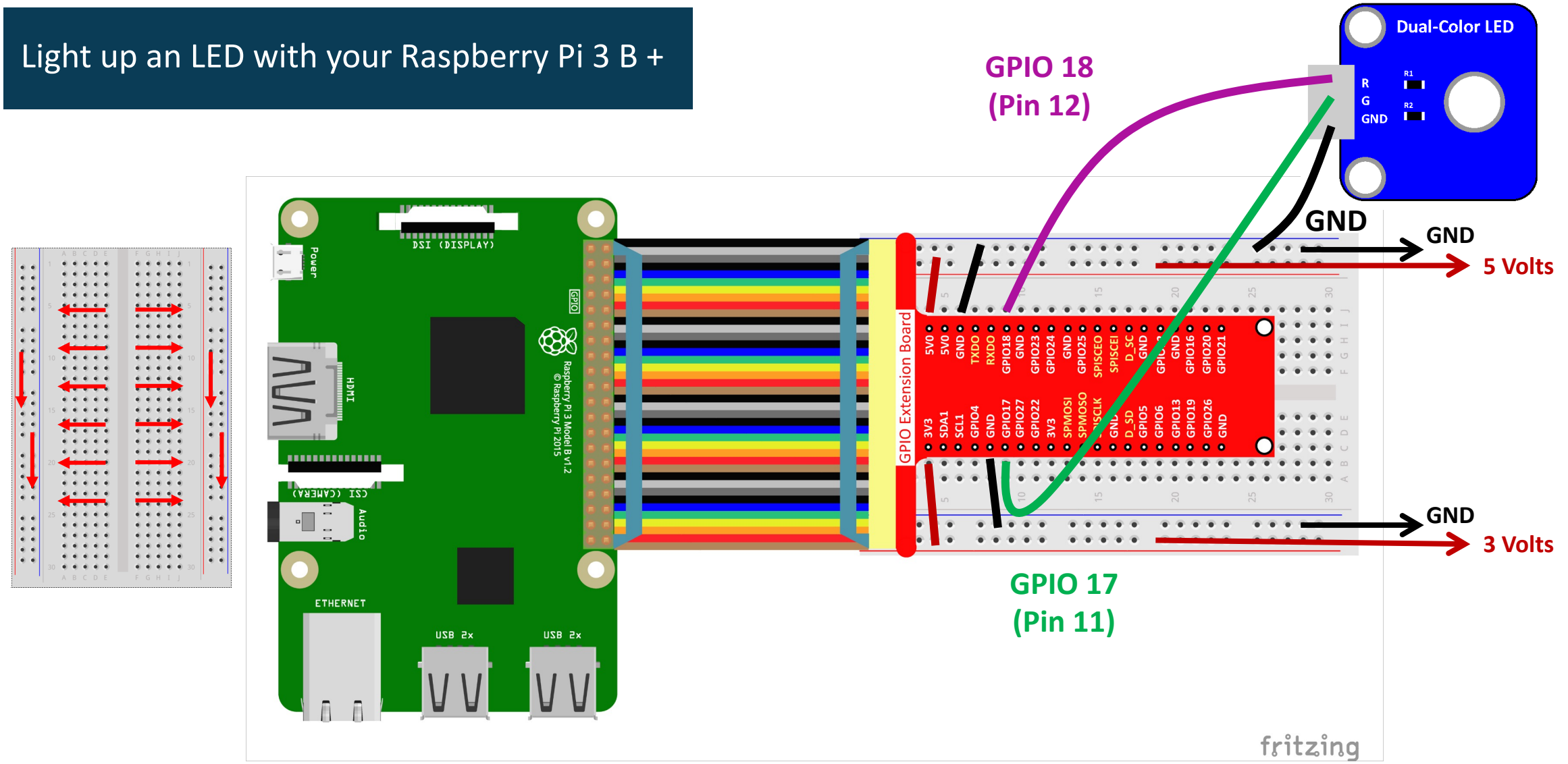
Source:
https://www.sunfounder.com/learn/lesson-1-dual-color-led-sensor-kit-v2-0-for-b.html

A dual-color light emitting diode (LED) is capable of emitting two different colors of light, typically red and green.

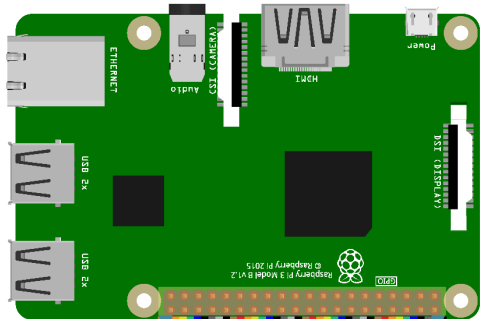
Application:
Variety of devices, such as televisions, digital cameras, and remote controls deploy these type LEDs.

Connector:
3-pin anti-reverse cable

Light up an LED with your Raspberry Pi 3 B +

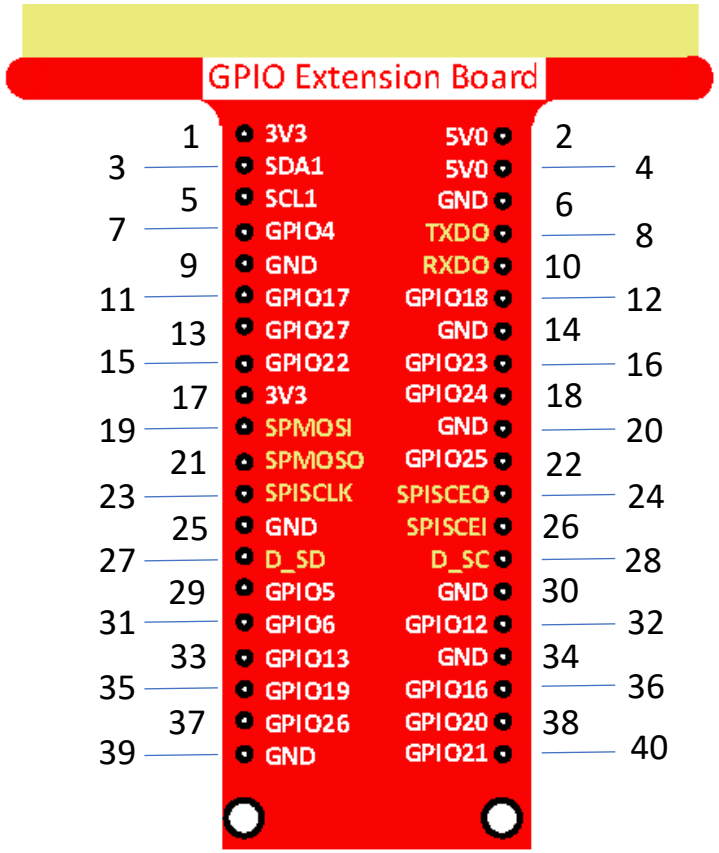


How a python code can light up your LED with Raspberry Pi Model 3 B+ (RPi)

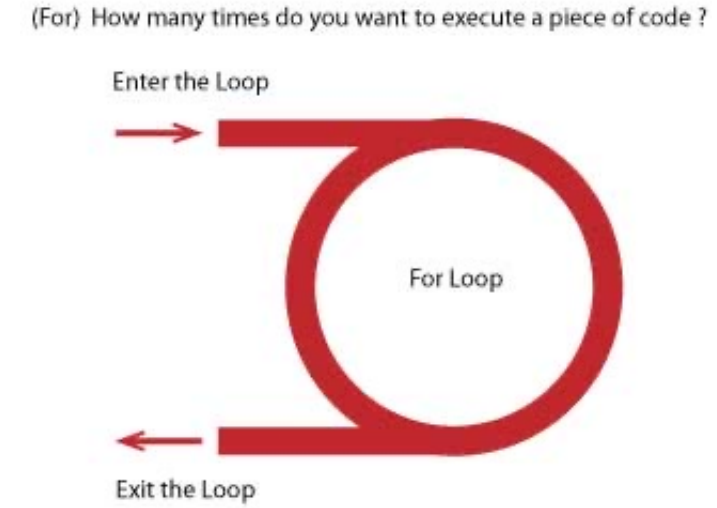


```
import RPi.GPIO as GPIO
import time
```

```
GPIO.setmode(GPIO.BOARD)
```

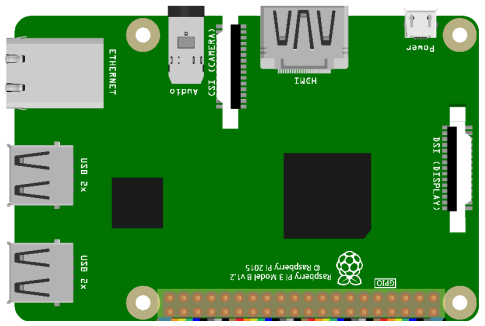



```
GPIO.setup(12, GPIO.OUT)
```



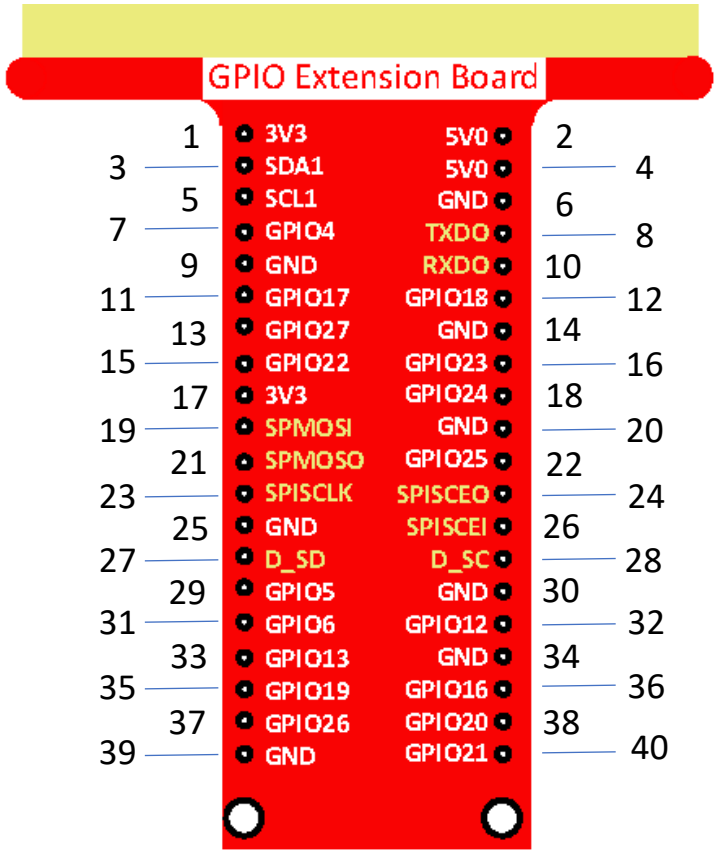

```
for i in range(0,15):

    GPIO.output(12, GPIO.HIGH)
    time.sleep(0.5)
    GPIO.output(12, GPIO.LOW)
    time.sleep(0.5)
    print(i)

GPIO.cleanup()
```

Another python code to kick start your Raspberry Pi Model 3 B+ (RPi)



```
import RPi.GPIO as GPIO
import time
```

```
GPIO.setmode(GPIO.BOARD)
```




```
GPIO.setup(12, GPIO.OUT)
```

```
def loop():
    while True:
        GPIO.output(12, GPIO.HIGH)
        time.sleep(0.5)
        GPIO.output(12, GPIO.LOW)
        time.sleep(0.5)
```

```
def destroy():
    GPIO.output(12, GPIO.LOW)
      # Turn off all leds
    GPIO.cleanup()
```

```
if __name__ == "__main__":
    try:
        loop()
    except KeyboardInterrupt:
        destroy()
```

Setting up a Python virtual environment - "Sandbox" your Python projects

"Unwearyingly working and making changes to the global Python environment can break Python-based system-level tools, and remedying the situation can become a major headache."

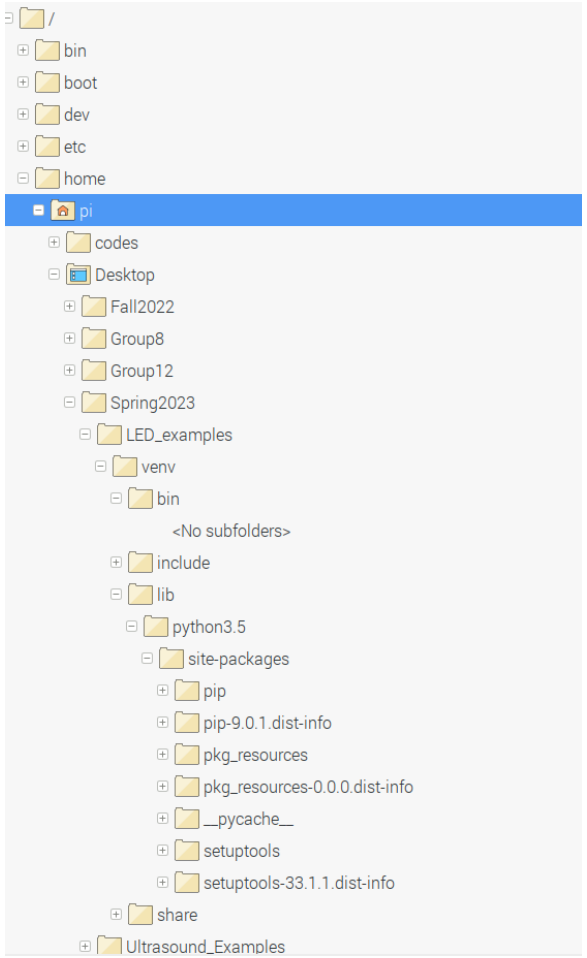
Gary Smart, Practical Python Programming for IoT

Create `python` virtual environments

- To “sandbox” our Python projects
- Avoid adverse interference with
  - system-level Python utilities
  - or other Python projects.

```
pi@raspberrypi: ~/Desktop/Spring2023/LED_examples
File  Edit  Tabs  Help
pi@raspberrypi:~ $ cd Desktop/Spring2023/LED_examples/
pi@raspberrypi:~/Desktop/Spring2023/LED_examples $
pi@raspberrypi:~/Desktop/Spring2023/LED_examples $
pi@raspberrypi:~/Desktop/Spring2023/LED_examples $ python3 -m venv venv
pi@raspberrypi:~/Desktop/Spring2023/LED_examples $
pi@raspberrypi:~/Desktop/Spring2023/LED_examples $
pi@raspberrypi:~/Desktop/Spring2023/LED_examples $ source venv/bin/activate
(venv) pi@raspberrypi:~/Desktop/Spring2023/LED_examples $
(venv) pi@raspberrypi:~/Desktop/Spring2023/LED_examples $
(venv) pi@raspberrypi:~/Desktop/Spring2023/LED_examples $ which python
/home/pi/Desktop/Spring2023/LED_examples/venv/bin/python
(venv) pi@raspberrypi:~/Desktop/Spring2023/LED_examples $
(venv) pi@raspberrypi:~/Desktop/Spring2023/LED_examples $
(venv) pi@raspberrypi:~/Desktop/Spring2023/LED_examples $ python --version
Python 3.5.3
(venv) pi@raspberrypi:~/Desktop/Spring2023/LED_examples $
(venv) pi@raspberrypi:~/Desktop/Spring2023/LED_examples $
(venv) pi@raspberrypi:~/Desktop/Spring2023/LED_examples $
(venv) pi@raspberrypi:~/Desktop/Spring2023/LED_examples $ deactivate
pi@raspberrypi:~/Desktop/Spring2023/LED_examples $
pi@raspberrypi:~/Desktop/Spring2023/LED_examples $
pi@raspberrypi:~/Desktop/Spring2023/LED_examples $
```

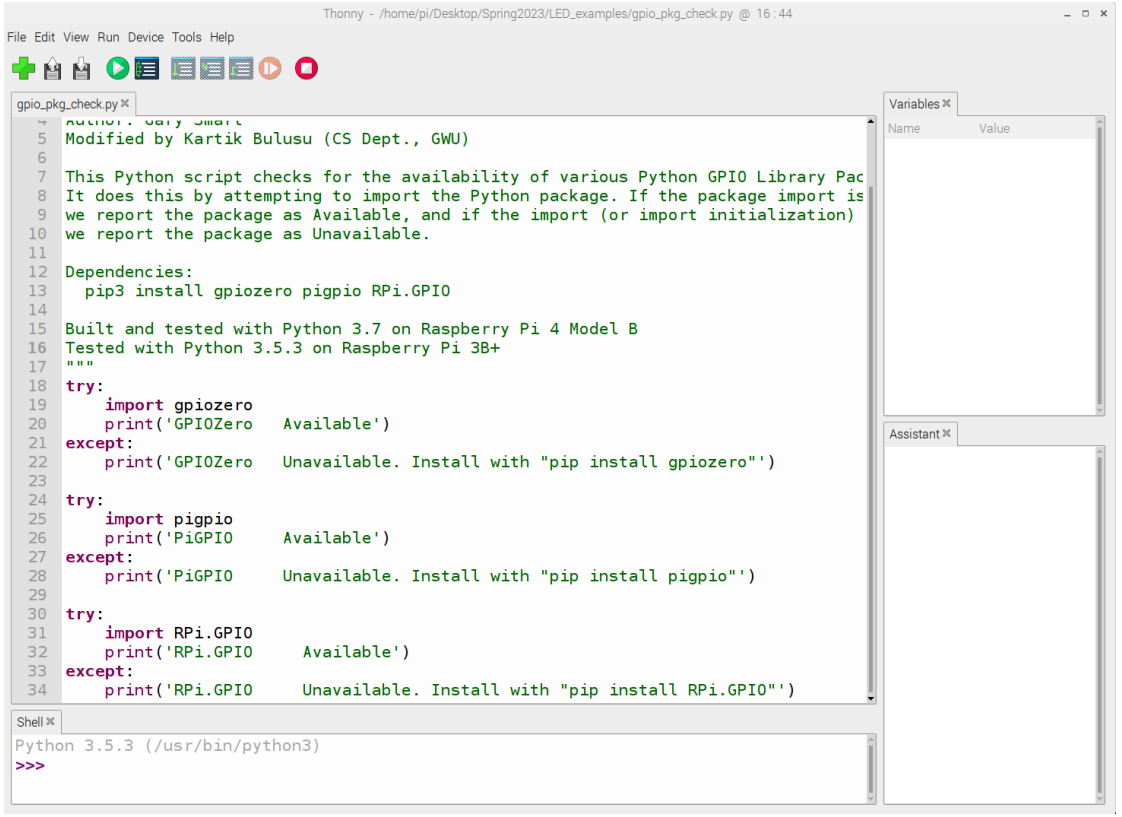
Anatomy of the `python` virtual environments



Activate `python-venv` & Install `pip`

```
pi@raspberrypi: ~/Desktop/Spring2023/LED_examples
File  Edit  Tabs  Help
pi@raspberrypi:~/Desktop/Spring2023/LED_examples $ source venv/bin/activate
(venv) pi@raspberrypi:~/Desktop/Spring2023/LED_examples $ pip3 install --upgrade pip
Cache entry deserialization failed, entry ignored
Cache entry deserialization failed, entry ignored
Collecting pip
  Using cached https://files.pythonhosted.org/packages/27/79/8a850fe3496446ff0d584327ae44e7500daf6764ca1a382d
2d02789accf7/pip-20.3.4-py2.py3-none-any.whl
Installing collected packages: pip
  Found existing installation: pip 9.0.1
    Uninstalling pip-9.0.1:
      Successfully uninstalled pip-9.0.1
Successfully installed pip-20.3.4
(venv) pi@raspberrypi:~/Desktop/Spring2023/LED_examples $
(venv) pi@raspberrypi:~/Desktop/Spring2023/LED_examples $
(venv) pi@raspberrypi:~/Desktop/Spring2023/LED_examples $ pip list
DEPRECATION: Python 3.5 reached the end of its life on September 13th, 2020. Please upgrade your Python as Python 3.
5 is no longer maintained. pip 21.0 will drop support for Python 3.5 in January 2021. pip 21.0 will remove support f
or this functionality.
Package       Version
------------- -------
pip           20.3.4
pkg-resources 0.0.0
setuptools    33.1.1
(venv) pi@raspberrypi:~/Desktop/Spring2023/LED_examples $
```

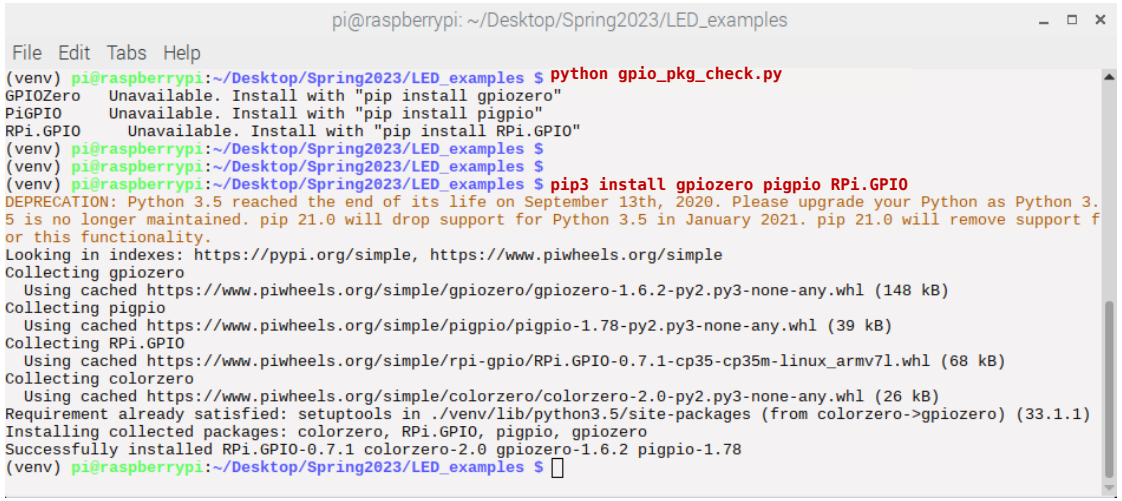
Review of the code provided to you

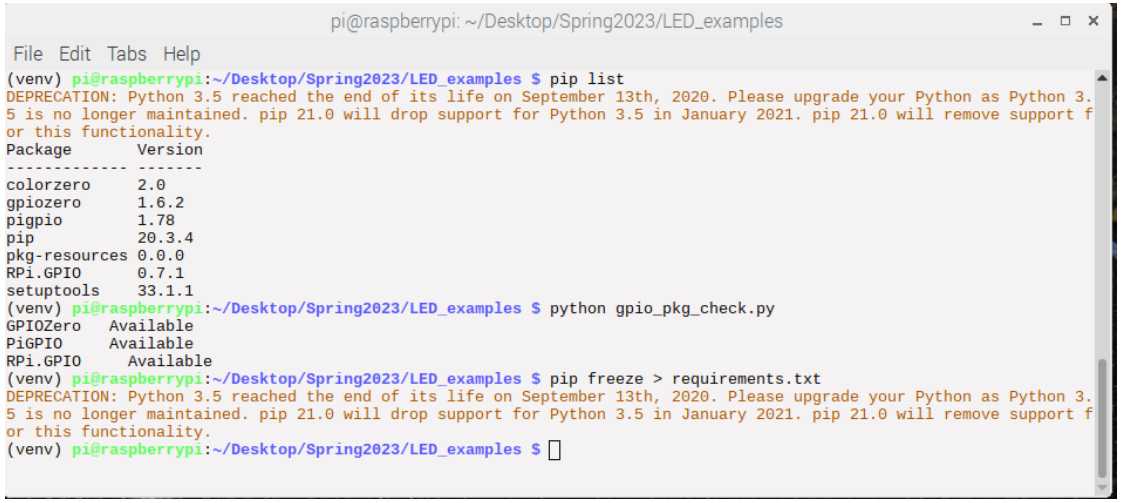



```
Modified by Kartik Bulusu (CS Dept., GWU)

This Python script checks for the availability of various Python GPIO Library Pac
It does this by attempting to import the Python package. If the package import is
we report the package as Available, and if the import (or import initialization)
we report the package as Unavailable.

Dependencies:
  pip3 install gpiozero pigpio RPi.GPIO

Built and tested with Python 3.7 on Raspberry Pi 4 Model B
Tested with Python 3.5.3 on Raspberry Pi 3B+
"""
try:
    import gpiozero
    print('GPIOZero   Available')
except:
    print('GPIOZero   Unavailable. Install with "pip install gpiozero"')

try:
    import pigpio
    print('PiGPIO     Available')
except:
    print('PiGPIO     Unavailable. Install with "pip install pigpio"')

try:
    import RPi.GPIO
    print('RPi.GPIO     Available')
except:
    print('RPi.GPIO     Unavailable. Install with "pip install RPi.GPIO"')
```

Install `GPIO` packages using the code provided.
Note: You are still in `python-venv`

Checks and balances:
`pip list`
`pip freeze`

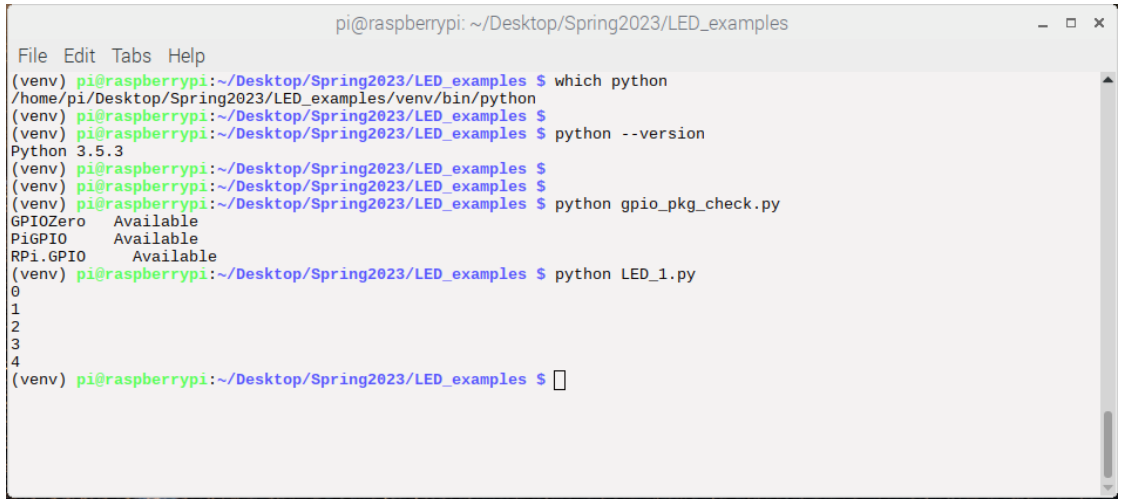




`pip` stands for “Python installs packages”.
`pip` queries the Python Package Index, or simply PyPi
Web source for PyPi: https://pypi.org

Run the python codes
Blink your LEDs
Note: You are still in `python-venv`

`deactivate` your venv



Running a `python` script at boot `cron`, the UNIX scheduler

Awesome IoT project will run automatically every time you start your Raspberry Pi

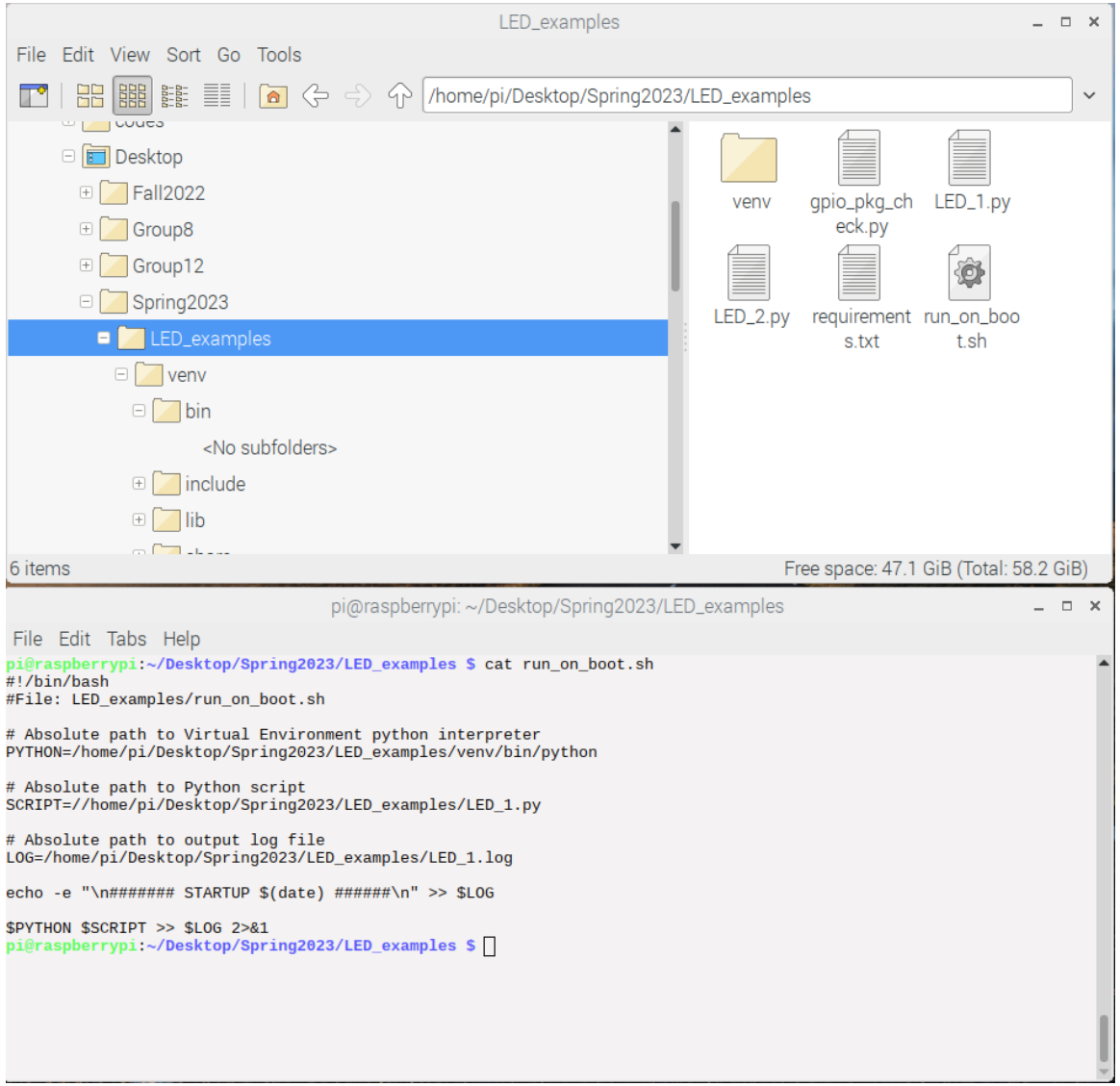
Review `run_on_boot.sh`
Note: You are no longer in `python-venv`

Let's make sure we have a file-folder structure that looks some what like this

Let's open the *run_on_boot.sh on a terminal* to review the script

Absolute paths to
1. Python interpreter in venv
2. Python script
3. Locate the LOG-file

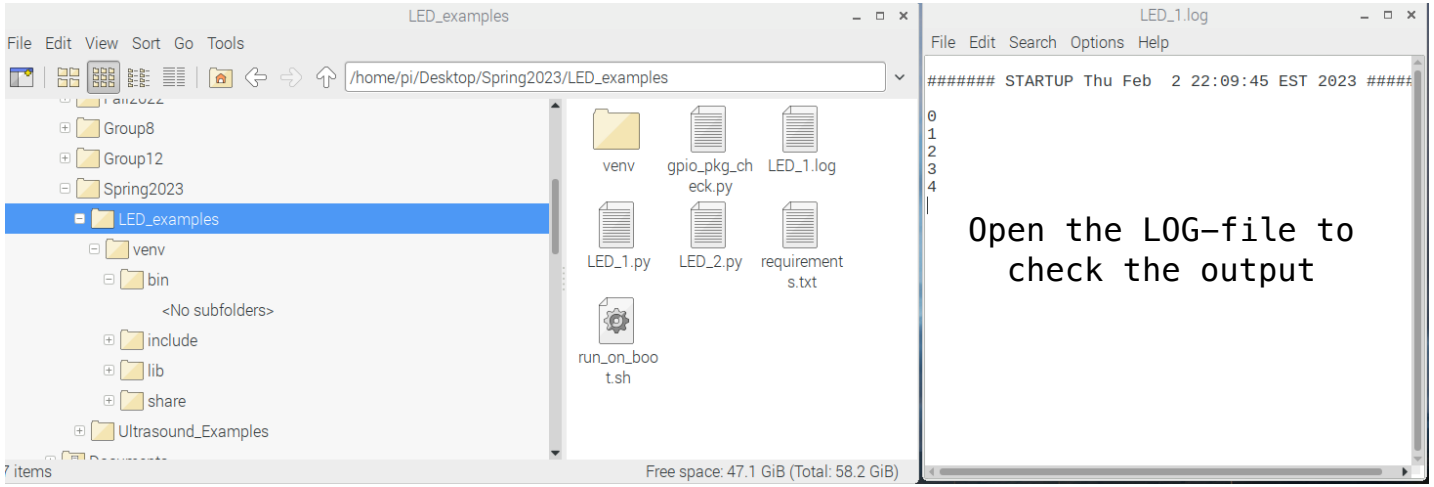
The last two lines tie everything together
1. echo [updates LOG-file with string in quotes]
2. Next line runs py-file and logs output on boot
3. 2>&1 [logs any additional errors during run]



`chmod u+x run_on_boot.sh`

*current **U**ser, make the file e**X**ecutable*

```
pi@raspberrypi: ~/Desktop/Spring2023/LED_examples
File  Edit  Tabs  Help
pi@raspberrypi:~/Desktop/Spring2023/LED_examples $ cat run_on_boot.sh
#!/bin/bash
#File: LED_examples/run_on_boot.sh

# Absolute path to Virtual Environment python interpreter
PYTHON=/home/pi/Desktop/Spring2023/LED_examples/venv/bin/python

# Absolute path to Python script
SCRIPT=//home/pi/Desktop/Spring2023/LED_examples/LED_1.py

# Absolute path to output log file
LOG=/home/pi/Desktop/Spring2023/LED_examples/LED_1.log

echo -e "\n####### STARTUP $(date) ######\n" >> $LOG

$PYTHON $SCRIPT >> $LOG 2>&1
pi@raspberrypi:~/Desktop/Spring2023/LED_examples $
pi@raspberrypi:~/Desktop/Spring2023/LED_examples $ chmod u+x run_on_boot.sh
pi@raspberrypi:~/Desktop/Spring2023/LED_examples $
```



`./run_on_boot.sh`

```
pi@raspberrypi: ~/Desktop/Spring2023/LED_examples
File  Edit  Tabs  Help
pi@raspberrypi:~ $ cd Desktop/Spring2023/LED_examples/
pi@raspberrypi:~/Desktop/Spring2023/LED_examples $ ./run_on_boot.sh
pi@raspberrypi:~/Desktop/Spring2023/LED_examples $
```

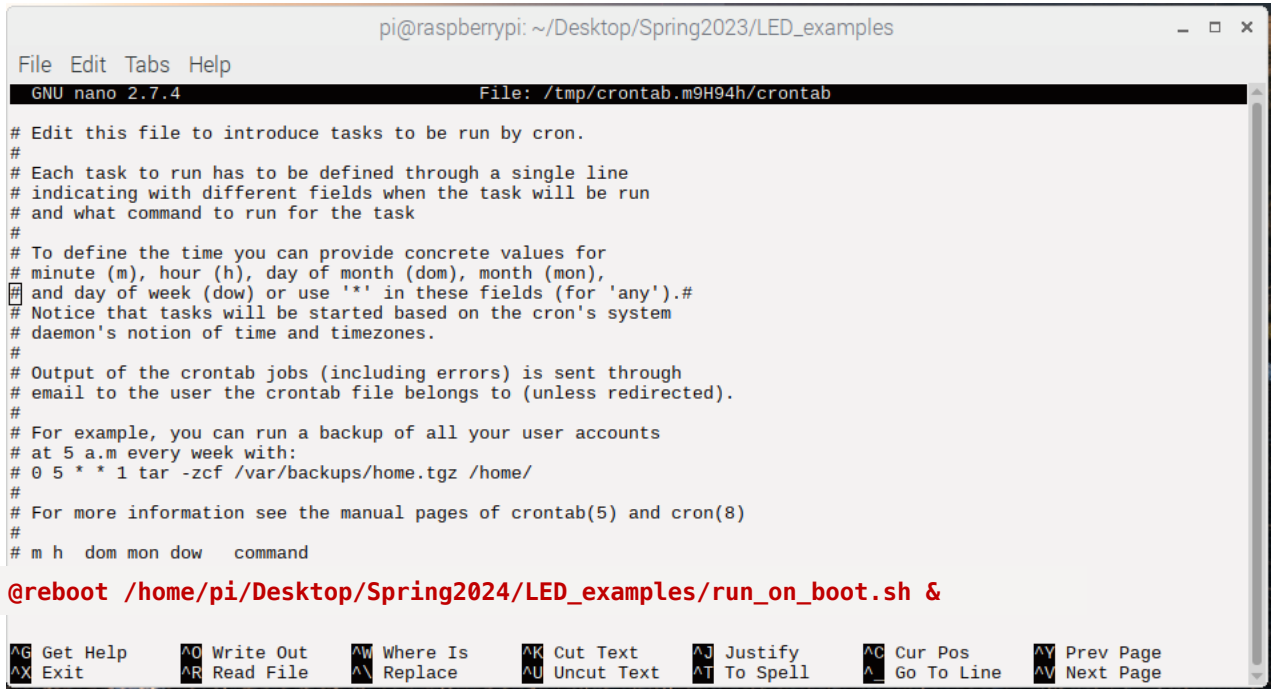
crontab -e

$ crontab -e

@reboot /home/pi/Desktop/Spring2024/LED_examples/run_on_boot.sh &

"Do not forget the & character at the end of the line. This makes sure the script runs in the background."

Gary Smart, Practical Python Programming for IoT



Try manually `./run_on_boot.sh` then `sudo reboot`