1. The package scipy and pyplot Course

b) Plotting data read file=open("sample.txt","w") file.write("ARJUN 15 \n") file.write("Bharat 20 \n") file.write("Charan 36 \n") file.write("Durga 42 \n") file.write("Eshwar 25 \n") file.write("Funny 42 ") file.close() file=open("sample.txt","r") data=file.read()

print(data) file.close() OUTPUT:

ARJUN 15

Bharat 20

Charan 36

Durga 42

Eshwar 25

Funny 42

program:

import matplotlib.pyplot as plt names = []

marks = []

f = open('sample.txt','r') for row in f:

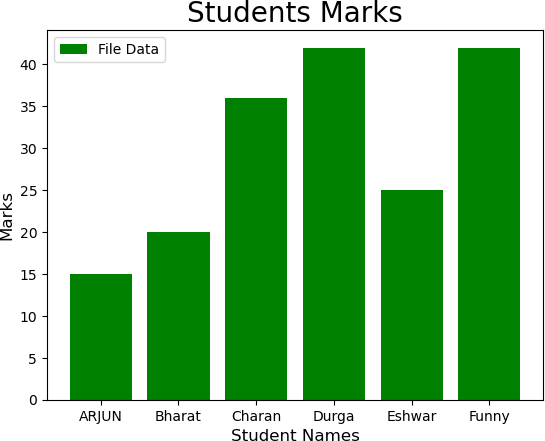
row = row.split(' ') names.append(row[0]) marks.append(int(row[1]))

plt.bar(names, marks, color = 'g', label = 'File Data') plt.xlabel('Student Names', fontsize = 12) plt.ylabel('Marks', fontsize = 12)

plt.title('Students Marks', fontsize = 20) plt.legend()

plt.show()

# OUTPUT:



1. The strings package

b) Read text from a file and return a list of all n letter words beginning with a vowel

test\_list = ["all", "love", "and", "get", "educated", "by",

"gfg","orange","eagle","icecream","umberella"] # printing original list

print("The original list is : " + str(test\_list)) res = [ ]

vow = "aeiou"

for sub in test\_list: flag = False

# checking for begin char for ele in vow:

if sub.startswith(ele): flag = True

break if flag:

res.append(sub) # printing result

print("The extracted words : " + str(res)) OUTPUT:

The original list is : ['all', 'love', 'and', 'get', 'educated', 'by', 'gfg', 'orange', 'eagle', 'icecream', 'umberella']

The extracted words : ['all', 'and', 'educated', 'orange', 'eagle', 'icecream', 'umberella']

C)Finding a secret message hidden in a paragraph of text def cipher(encrypted,key):

message=""

for offset,line in zip(key,encrypted.readlines()): message = message + line[offset]

return message from io import StringIO f = StringIO("""sree chaitanya

institute of technological sciences krimngar """)

print(cipher(f,[0,0,4,0,7])) OUTPUT:

Scits

d) Plot a histogram of words according to their length from text read from a file program:

from collections import Counter import numpy as np

import matplotlib.pyplot as plt

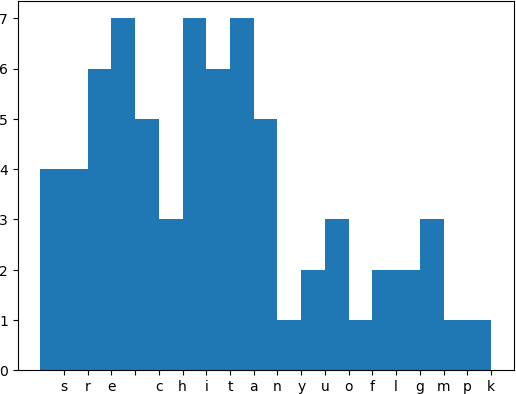
full = "sree chitanya institute of technological sciences thimmapur karimnagar" xxx=list(full)

labels, values = zip(\*Counter(xxx).items( )) indexes = np.arange(len(labels))

width = 1

plt.bar(indexes, values, width) plt.xticks(indexes + width \* 0.5, labels) plt.show( )

OUTPUT:



Cycle-2

1. Installing OS on Raspberry Pi
2. Installation using PiImager
3. Installation using image file Downloading an Image

* Writing the image to an SD card
* using Linux
* using Windows
* Booting up
* Follow the instructions given in the URL <https://www.raspberrypi.com/documentation/computers/getting-started.html>

# Installing OS on Raspberry Pi Introduction to Raspberry Pi 4 Mod B

Raspberry Pi is a small single-board computer developed by the [Raspberry Pi](https://www.raspberrypi.org/)

[Foundation](https://www.raspberrypi.org/) in the United Kingdom. It is a credit card-sized computer that is powerful enough to be used as a desktop computer or to control robots, build

projects, and much more. The Raspberry Pi is powered by an ARM processor, which means it is capable of running a variety of operating systems.

We have chosen Raspberry Pi mod 4B board to install Raspberry Pi OS as it is powered by a Broadcom BCM2711 quad-core Cortex-A72 processor and comes with 1GB, 2GB, 4GB, or 8 RAM of RAM variants. It also has a Gigabit Ethernet port, two USB 3.0 ports, two USB 2.0 ports, and a 4-pole stereo output and composite video port. All this hardware makes it used as a general-purpose computer, and it is ideal for projects such as building a [file server](https://thesecmaster.com/the-ultimate-guide-to-build-a-personal-cross-platform-file-server-on-raspberry-pi/), [media center](https://thesecmaster.com/build-your-own-ott-platforms-like-netflix-using-raspberry-pi-and-plex-media-server/), making a home automation system, or setting up a [home surveillance system](https://thesecmaster.com/step-by-step-procedure-to-build-your-own-surveillance-system-using-raspberry-pi/).

The Raspberry Pi 4 is also ideal for running a variety of operating systems like Ubuntu, Debian, [Mint Linux](https://www.thesecmaster.com/step-by-step-procedure-to-install-linux-mint-linux-on-vmware-workstation/), [Kali Linux](https://www.thesecmaster.com/install-kali-linux-on-vmware-workstation/), Manjaro, and Windows 10, and the list is endless. Since it has enough processing power and memory to run the Linux distribution, it is a great choice for running many DIY projects. It is also relatively inexpensive and easy to set up, making it an attractive option for users who want to explore the world of digital forensics and penetration testing.

# What is Raspberry Pi OS?

Raspberry Pi OS is a Debian-based operating system specifically designed for the Raspberry Pi Single Board Computer, which was developed by

the [Raspberry Pi Foundation](https://www.raspberrypi.org/) in the United Kingdom. It is an open-source software project, meaning that it is free to use, and anyone can modify its source code with the proper knowledge. [Raspberry Pi OS](https://thesecmaster.com/step-by-step-guide-to-install-raspberry-pi-os-on-a-raspberry-pi-single-board-computer/) comes with a wide range of functionalities built in, such as support for GPIO pins, video output, and audio output. It is a Linux-based operating system that offers users access to the extensive library of applications available within its repositories. The operating system can also be configured to run specific tasks such as media streaming, home automation, and robotics via the open-source software libraries provided by [Raspberry Pi OS](https://thesecmaster.com/step-by-step-guide-to-install-raspberry-pi-os-on-a-raspberry-pi-single-board-computer/). With its flexibility and ease of use, Raspberry Pi OS is an ideal choice for those looking to get started with the Raspberry Pi.

# What Are the Prerequisites for Installing Raspberry Pi OS on A Raspberry Pi 4?

Before you install Raspberry Pi OS on your [Raspberry Pi](https://www.thesecmaster.com/how-set-up-a-raspberry-pi-for-the-first-time/) 4, you need to make sure that you have all of the necessary components.

First and foremost, you will need a Raspberry Pi 4, which is available in 1GB, 2GB, 4GB, and 8GB RAM variants.

You will also need a power supply and a microSD card with at least 8GB of storage capacity or USB storage with the same storage requirements.

Additionally, you will need a micro HDMI cable, a USB keyboard and mouse, and a compatible monitor.

## Hardware:

* 1. Raspberry Pi Board: Well, you can use any Raspberry Pi module 4, or

400. Preferably, Raspberry Pi 3 and 4 with 4 GB RAM.

* 1. Micro SD Card with Card Reader: This is to write the Raspberry Pi OS image.
  2. USB Drive (Optional): This is to install Ubuntu. It is optional. This is required only if you boot your Pi from USB storage.
  3. IO Peripheral Devices: Keyboard, Mouse, HDMI Cable, Ethernet Cable, and USB Cable, depending on your requirements.
  4. PC or Mac with an Internet connection: This is required to download the [Raspberry Pi Imager application](https://www.raspberrypi.com/software/) and write the Raspberry Pi OS image to the SD Card or USB Storage Drive using the Imager application.

## Software:

1. Raspberry imager: A free utility released by [Raspberry Pi Imager](https://www.raspberrypi.com/software/) [application](https://www.raspberrypi.com/software/) for flashing content to SD cards or USB drives.
2. Internet: An Internet connection is a must to have to download the Raspberry Pi Imager and write the Raspberry Pi OS to the SD Card or USB Storage Drive.

# How to Install Raspberry Pi OS on A Raspberry Pi 4?

Installing Raspberry Pi OS on a [Raspberry Pi](https://www.thesecmaster.com/how-set-up-a-raspberry-pi-for-the-first-time/) 4 is a relatively straightforward process. Well, you can install Raspberry Pi OS on a Raspberry Pi in two different ways.

1. Write the bootable Raspberry Pi OS image to the micro SD card or the USB drive directly using the Raspberry Pi imager application.
2. Download the bootable ISO image of Raspberry Pi OS from the official Raspberry Pi [website](https://www.raspberrypi.com/software/operating-systems/) and write using the Raspberry Pi imager or [Etcher](https://etcherofficial.com/) application to the micro SD card or the USB drive.

# Method 1: Flash the Ubuntu image to the micro SD or USB drive directly from the Raspberry Pi Imager

Download the [Raspberry Pi](https://www.thesecmaster.com/how-set-up-a-raspberry-pi-for-the-first-time/) Imager application on your Windows PC or mac book. Just follow the on-screen instructions, and the installation process should be completed in a few minutes. Once the installation is complete, you can start exploring the features of the Raspberry Pi Imager application.

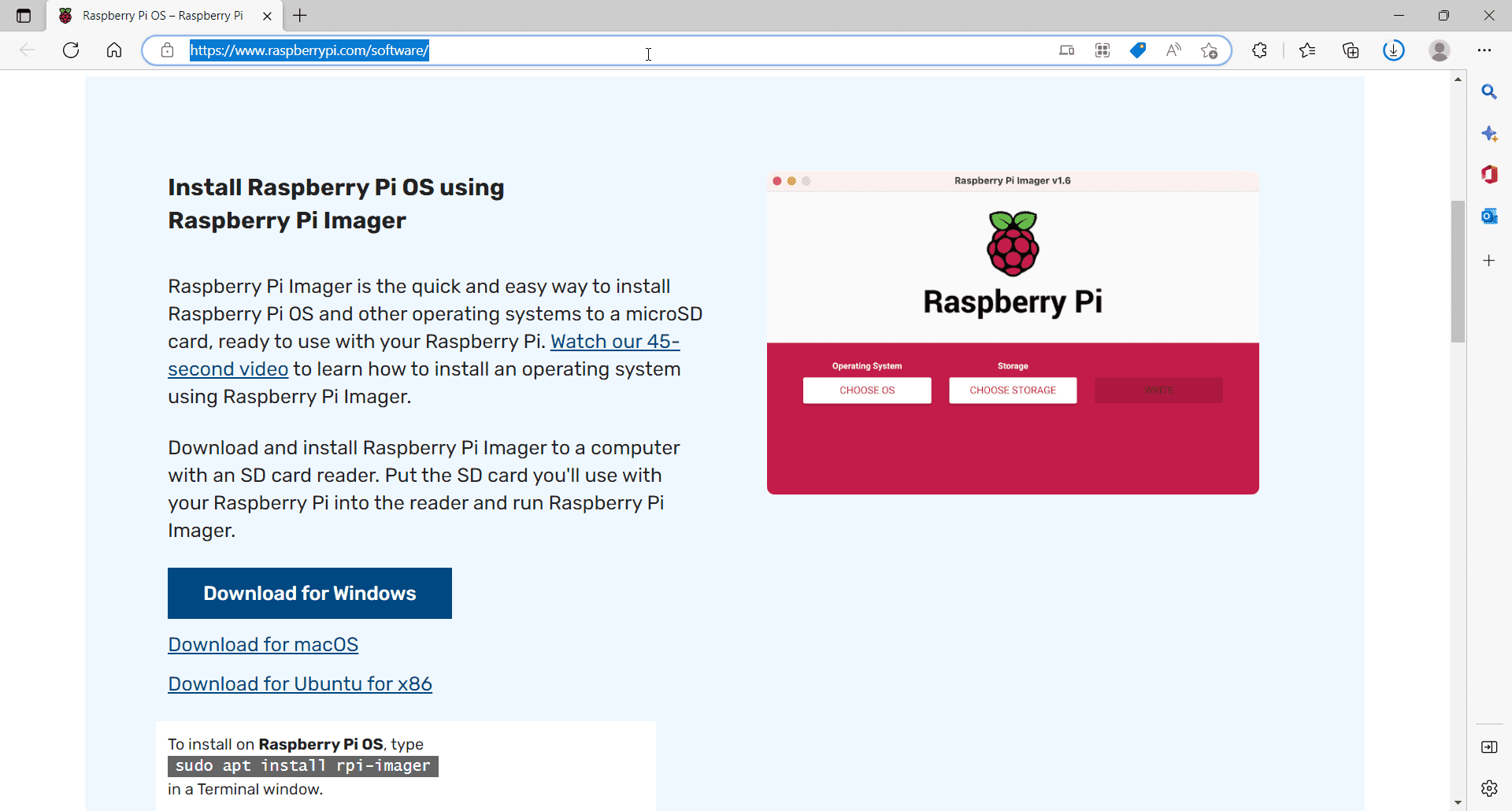
Time needed: 30 minutes

# How to Install Raspberry Pi OS on A Raspberry Pi 4?

1. **Download and Install the Raspberry Pi Imager application** Raspberry Pi Imager is a quick and easy way to install Raspberry Pi OS and other operating systems to a microSD card or USB

drive. Watch this 45-second video to learn how to install an operating

system using Raspberry Pi Imager.<https://www.youtube.com/watch?v=ntaXWS8Lk34Download> the Raspberry Pi Imager application to a computer with an SD card reader. run the installer app and follow a few instructions to Install Raspberry Pi Imager. Put the SD card or USB drive you’ll use to install Raspberry Pi OS.



# Run the Raspberry Pi Imager application

Upon installation, when you run the Imager application, you will be greeted with this window.



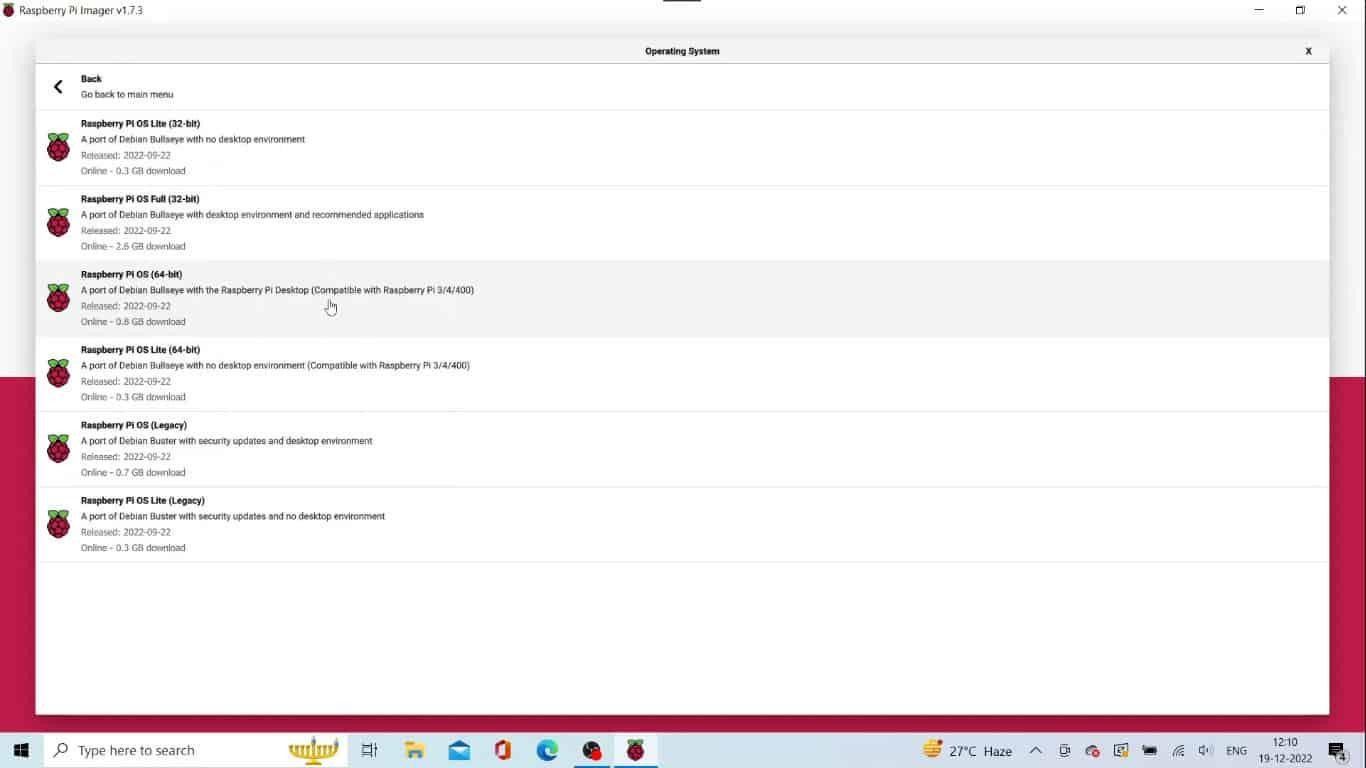
# 1.Choose Raspberry Pi OS Image on the Raspberry Pi

Click on the **Choose OS** button on the Imager application, select the **Raspberry Pi OS (Other)** option then click on Raspberry Pi OS variant you want. You will see a list of Raspberry Pi OS versions. Choose the one you want to go with.

**Raspberry Pi Lite:** Raspberry Pi Lite is a version of the Raspberry Pi OS that only includes the essential components and features. It is designed to offer users support for their projects while reducing system resource usage, making it ideal for those who prioritize performance over features.

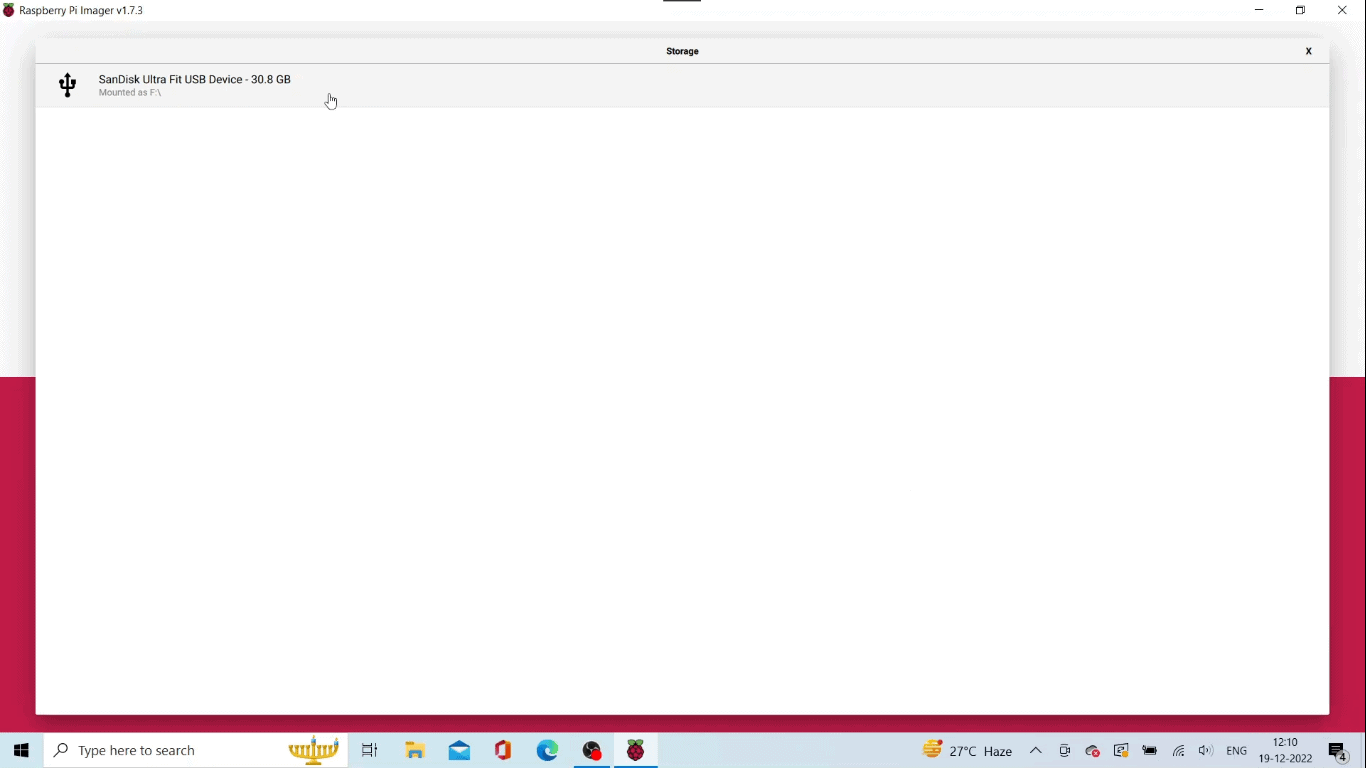
**Raspberry Pi OS Full:** This version of the operating system includes all the features available with Raspberry Pi OS Lite along with the full range of applications and functionalities. It is especially suitable for those who want access to a wide variety of software packages, as it comes with an extensive library of open-source applications.

**Raspberry Pi OS (Legacy):** This version was released alongside the original Raspberry Pi and is an outdated version of the operating system. As such, it is no longer supported by the Raspberry Pi Foundation and should only be used if absolutely necessary.



# 1.Choose the Storage

Click on the Choose Storage button on the Imager application. That will show up on the list of connected micros SD cards or USB storage devices. Select the one you want to install Raspberry Pi OS.



# 3.Write the Raspberry Pi OS image to the storage

Click on the Write button to start the writing process.



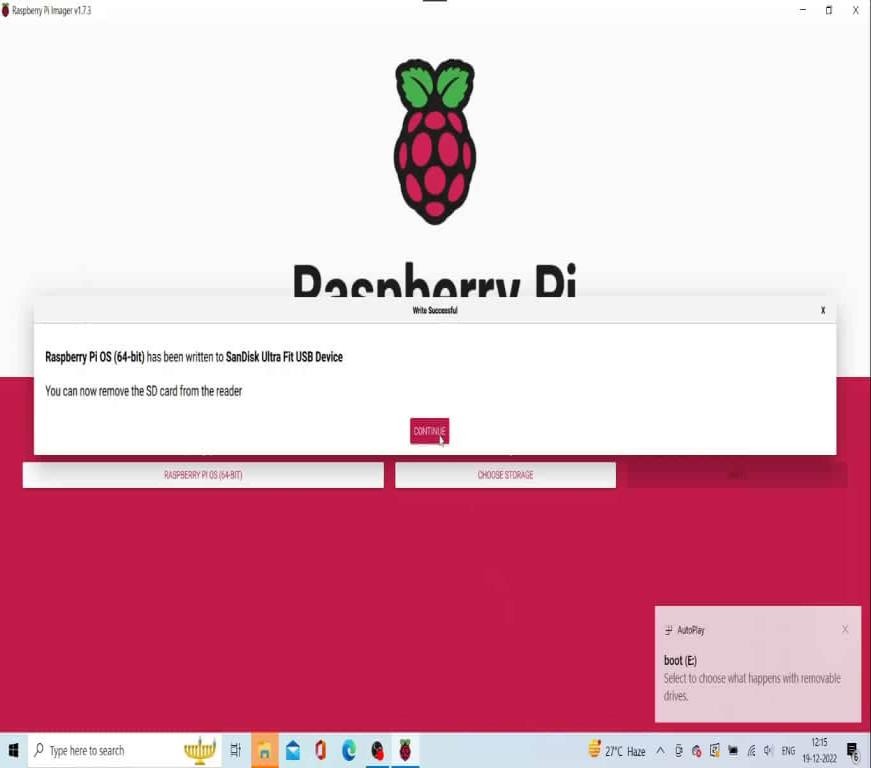
# Raspberry Pi OS image is being written to the USB storage



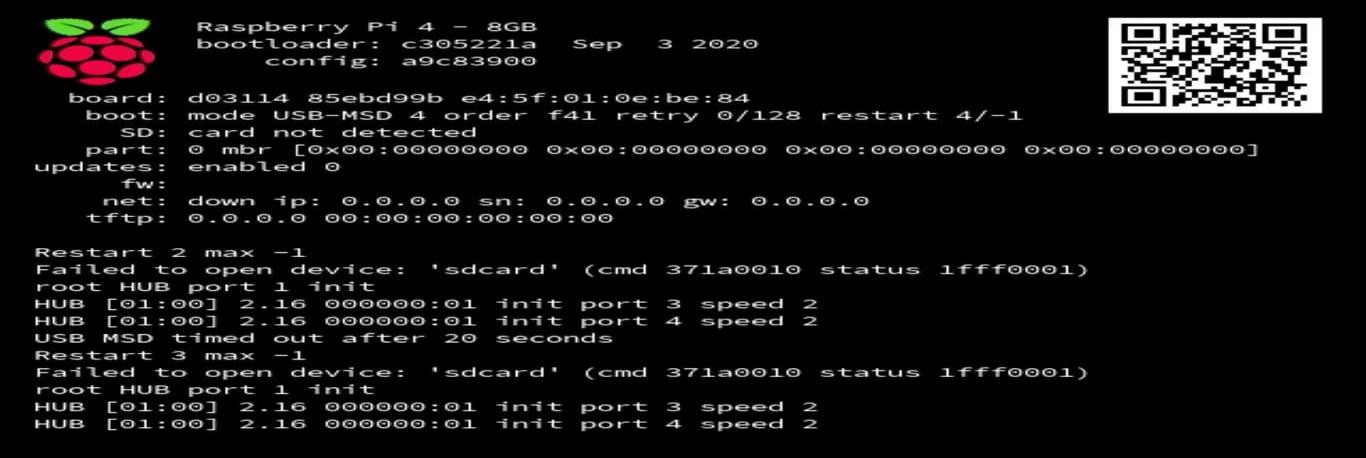
**Raspberry Pi OS image is being verified**



# Raspberry Pi OS image is written to the USB storage drive



2.**Boot the Raspberry Pi inserting the USB drive**

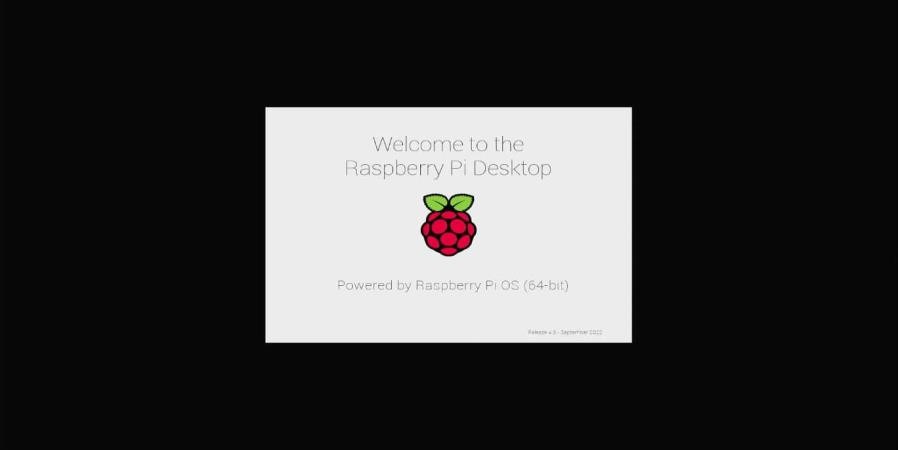


# The colored window of Raspberry Pi

You will see a colored window like shown here. This indicates that your Raspberry Pi passed the POST test and is ready to load the operating system.



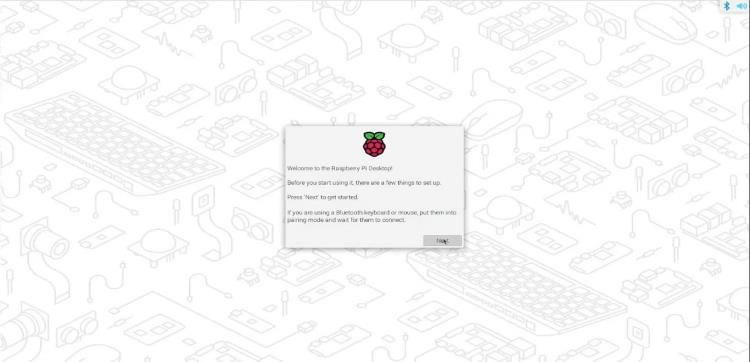
# Raspberry Pi OS is in the boot process



* 1. **System Configuration Wizard**

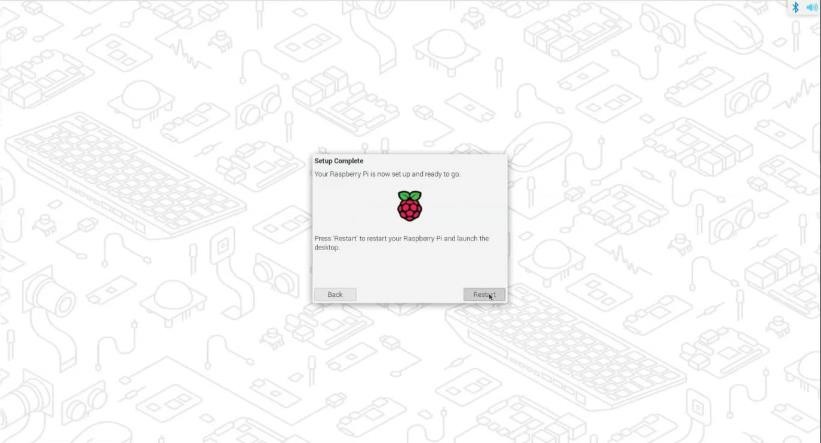
Upon the completion of the boot process. **Raspberry Pi OS** will throw a system configuration wizard. **Raspberry Pi OS** will ask for several configurations to be set up. You should need to Select the preferred settings to configure.

The configuration wizard starts from the Language selection all the way it goes to the keyboard, network, time zone, and login user account settings.



# Reboot the Raspberry Pi

Upon the completion of the set up process, system will ask to reboot to complete the process. Click on the Reboot button to continue Reboot.



# Raspberry Pi OS is running on the Raspberry Pi

That’s it. Immaterially, upon login, you will be greeted with the desktop screen

. 

# Accessing GPIO pins using Python

1. **Installing GPIO Zero library. First, update your repositories list: sudo apt update Then install the package for Python 3: sudo apt install python3- gpiozero**

**Installing GPIO Zero**

GPIO Zero is installed by default in the [Raspberry Pi OS](https://www.raspberrypi.org/software/operating-systems/) desktop

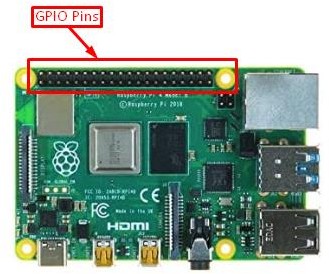
image, [Raspberry Pi OS](https://www.raspberrypi.org/software/operating-systems/) Lite image, and the [Raspberry Pi Desktop](https://www.raspberrypi.org/software/raspberry-pi-desktop/) image for PC/Mac, all available from [raspberrypi.org](https://www.raspberrypi.org/software/). Follow these guides to installing on other operating systems, including for PCs using the [**re**mote GPIO](https://gpiozero.readthedocs.io/en/latest/remote_gpio.html) featu**re**.

Raspberry Pi

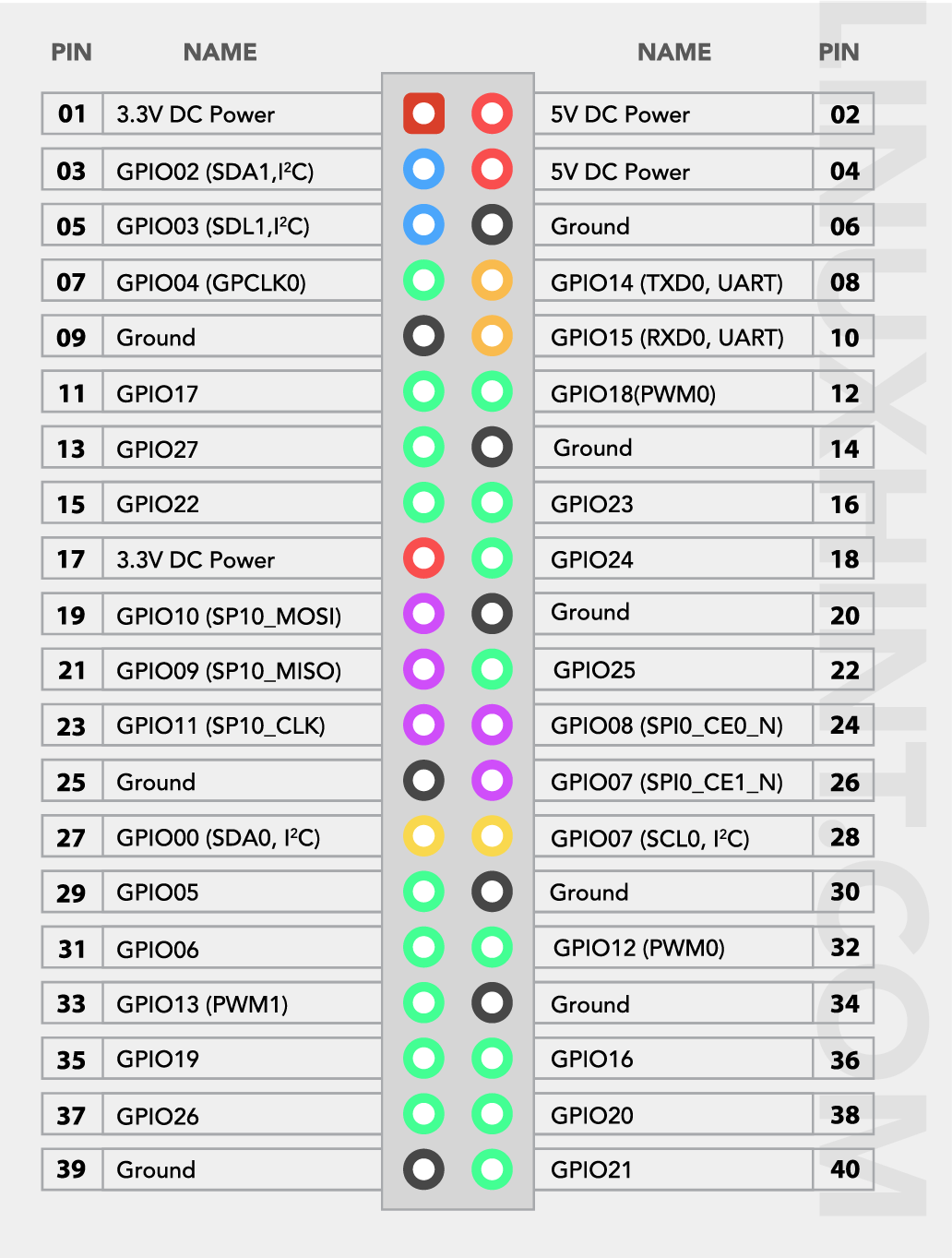
|  |
| --- |
| GPIO Zero is packaged in the apt **re**positories of Raspberry Pi OS, [Debian](https://packages.debian.org/buster/python3-gpiozero) and [Ubuntu](https://packages.ubuntu.com/hirsute/python3-gpiozero). It is also available on [PyPI](https://pypi.org/project/gpiozero/).  **apt**  First, update your **re**positories list: |
| **pi@raspberrypi:~$** sudo apt update |
| Then install the package for Python 3: |
| **pi@raspberrypi:~$** sudo apt install python3-gpiozero |
| or Python 2: |
| **pi@raspberrypi:~$** sudo apt install python-gpiozero |
| **pip**  If you’**re** using another operating system on your Raspberry Pi, you may need to use pip to install GPIO Zero instead. Install pip using [get-pip](https://pip.pypa.io/en/stable/installing/) and then type: |
| **pi@raspberrypi:~$** sudo pip3 install gpiozero |
| or for Python 2: |
| **pi@raspberrypi:~$** sudo pip install gpiozero |
| To install GPIO Zero in a virtual environment, see the [Development](https://gpiozero.readthedocs.io/en/latest/development.html) page. |

# Blinking an LED connected to one of the GPIO pin GPIO Pins on Raspberry Pi

**GPIO or General-Purpose Input/Output Pins** are the key component of the Raspberry Pi board since through these pins you can control any circuit right from your system. In Raspberry Pi 4, they are 40 GPIO pins, which are highlighted in the image below:



The pin labels are shown in the image below, and only those pins that start with the name GPIO are programmable:



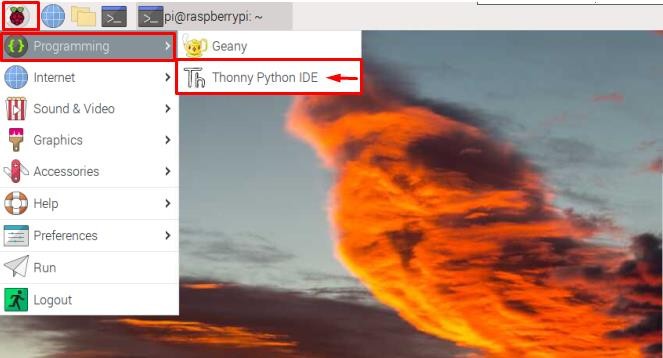
For details about these pins’ headers, follow [this](https://linuxhint.com/gpio-pinout-raspberry-pi/).

# How to Use Raspberry Pi GPIO Pins

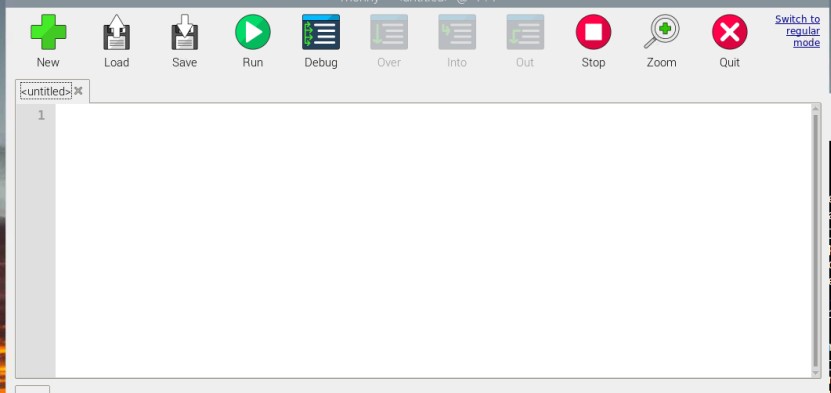
Raspberry PI OS comes with a pre-installed **Python** editor called **Thonny Python IDE** that allows users to code the **GPIO pins** in Python. The steps to write a python code using the **Thonny Python** editor are mentioned below with an example:

# Step 1: Open Python Editor

To use the Python editor, go to the **Application Menu**, select the “**Programming**” option to open the **Thonny Python IDE** on Raspberry Pi desktop.



The **Thonny Python** interface will appear on the screen as shown below:



# Step 2: Importing GPIO Module

To start using the **GPIO pins**, you have to import the **GPIO library** using the following code.

import RPi.GPIO as GPIO

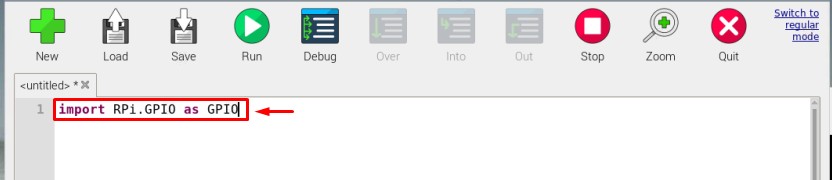
The **GPIO library** is used before writing the code since it allows you to control

the **GPIO pins**. This library is already installed by default on the Raspberry Pi system.

By using this command, we are just importing this RPi.GPIO module and calling it as GPIO so that we can just simply use the **GPIO** instead of writing the whole name again and again in the code.

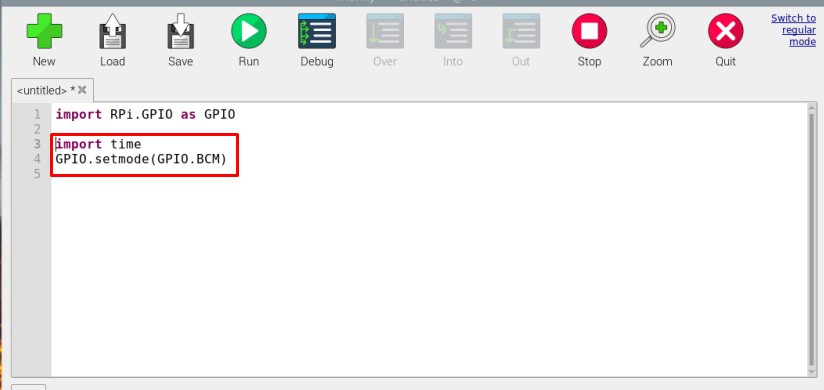
**Physical Board Pin Number**

**GPIO Number**



# Step 3: Import Time and Configure GPIO

Now, for our example, you have to import the **time module** and set the GPIO pins using the following code as this will help you later in the code to use time constraints and utilize the GPIO pin later in the code.



import time GPIO.setmode(GPIO.BCM)

**Note:** The BCM with GPIO in the command represents the Broadcom Channel numbers of pins:

|  |  |
| --- | --- |
| Pin 11 | 17 |
| Pin 12 | 18 |
| Pin 13 | 27 |
| Pin 15 | 22 |

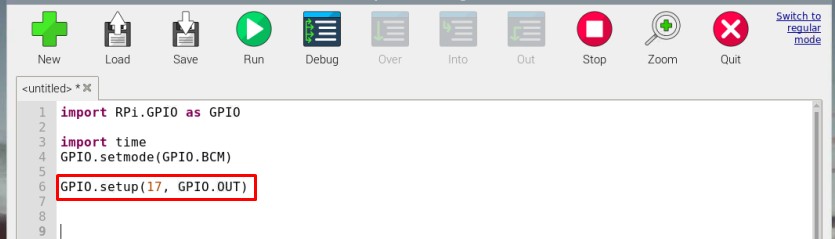
The Broadcom channel number is fixed for instance some GPIO numbers are shared below:

See the above **GPIO table** for further guidance.

# Step 4: Pin Configuration

Now, finally it’s time to think about what you are interested in using the **GPIO pins**. If you have to display the output using GPIO pins then you have to configure the GPIO as an output pin and if you are using some sensor or a device that needs to be attached as an input device, configure the pin as the input pin such as **GPIO.setup (22, GPIO.IN)**.

In the example below, I am using **GPIO 17** (which is pin number 11 on the board) as an output because I will use this pin to light up the LED.



GPIO.setup(17, GPIO.OUT)

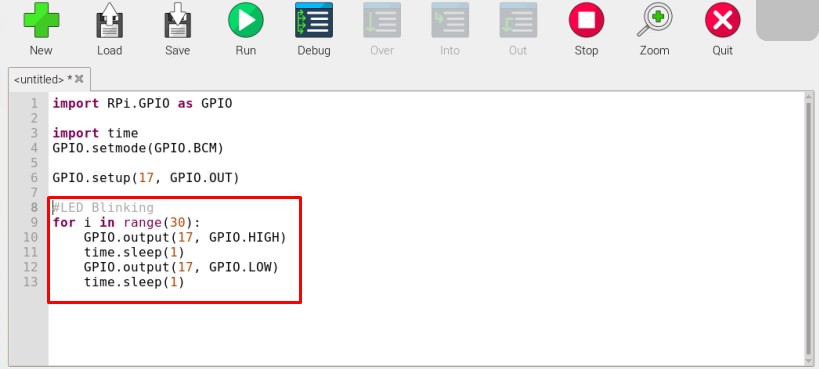
# Step 5: Write Code

The below code can be utilized to toggle the LED on Raspberry Pi. You can use the same code or use a different one because the code is provided for your guidance.

Since I am toggling or blinking the LED for 30 times, so “**for**” loop is used. Further, the **GPIO.HIGH** is used to turn on the LED. The **time.sleep** is used to

hold the state for **1** second before turning off the LED using the **GPIO.Low** code:

**Note:** You can change the Pin number and time for LED blinking according to your choice.



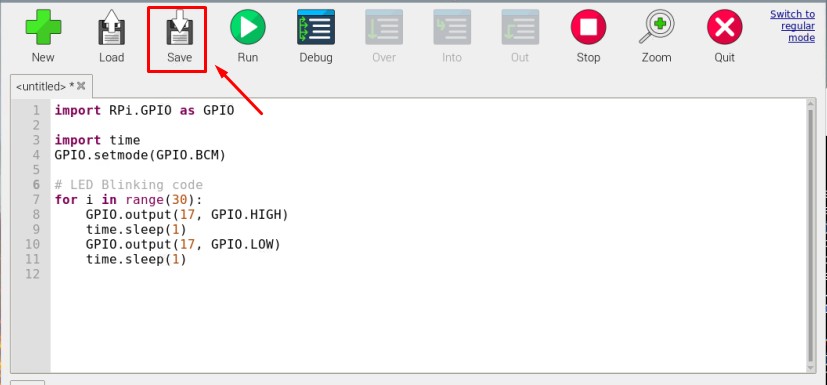
for i in range(30): GPIO.output(17, GPIO.HIGH)

time.sleep(1) GPIO.output(17, GPIO.LOW)

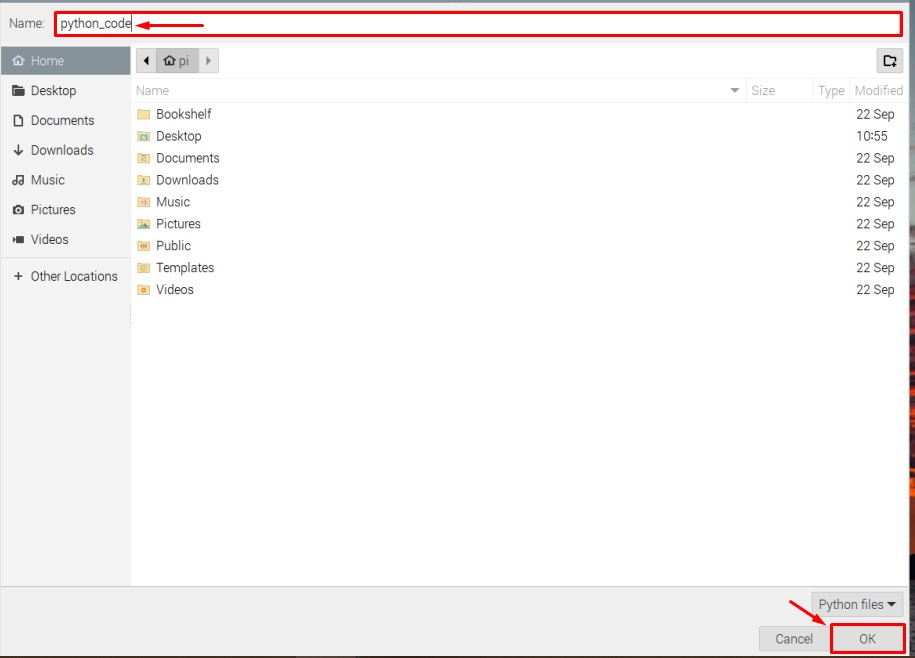
time.sleep(1)

# Step 6: Save the File

After completing the code, save the file using the “**Save**” button from the menu bar.



Choose an appropriate name for your file. In my case, it’s “**python\_code**”.



# Step 7: Build the Circuit

Now the coding part is completed, it’s now time to test the code. However, before that, you must create a circuit using the code you just created in the above steps.

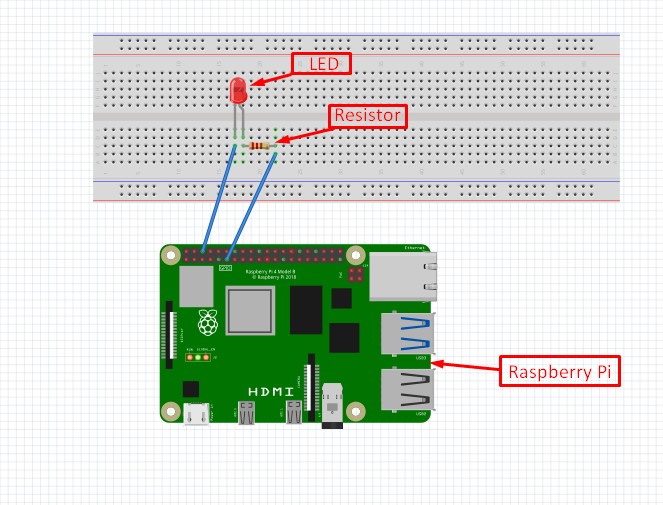
To create the circuit for LED blinking, follow the guidelines given below:

* The **positive terminal** of a LED is connected to **GPIO 17** (pin 11 on board) and the **negative terminal** of the LED is connected to

the **Ground** (pin 6 on board).

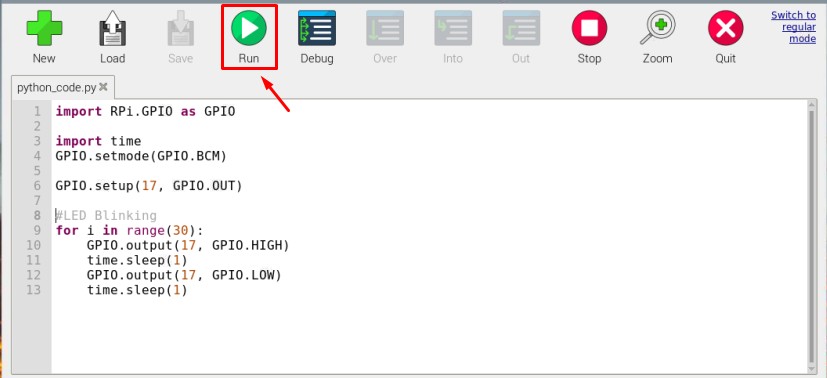
* A resistor is connected to the positive terminal of the LED so that the LED won’t burn due to excessive voltage. If you are using LED with an in-built resistor then you can skip the resistor.

Follow the below-given circuit for a better picture.



# Step 8: Run the Code

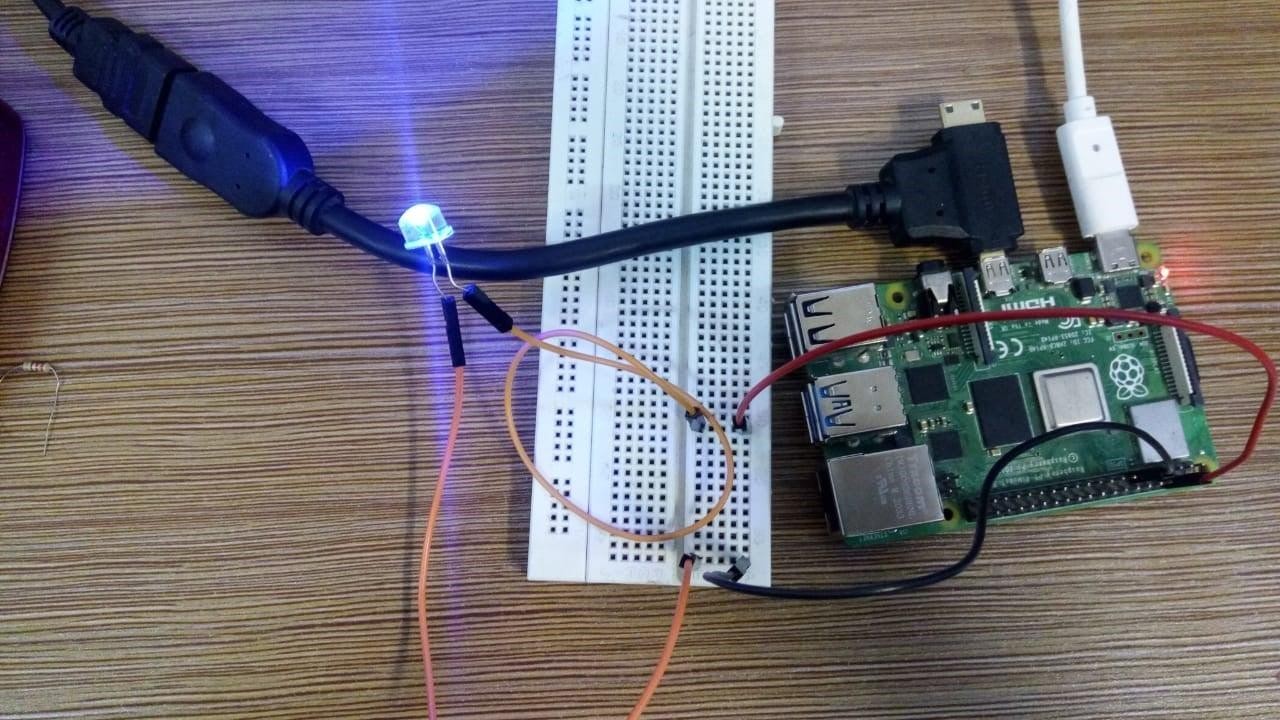
Once the circuit is completed, you can run the code using the “**Run**” button on the Thonny IDE to see if the LED starts blinking.



# Output:

The output of my code can be seen in the below image, the LED has blinked 30 times with a one-second delay between each **Off** and **On** state.

**Note:** In the below circuit I have used an LED with a built-in resistor so no separate resistor is attached.



# Adjusting the brightness of an LED

1. **Adjust the brightness of an LED (0 to 100, where 100 means maximum brightness) using the in-built PWM wavelength.**

# Code for controlling the brightness of LED using Raspberry pi:

import RPI.GPIO as GPIO import time GPIO.setmode(GPIO.Board) GPIO.setup(11, GPIO.OUT)

p = GPIO.PWM(11, 100) //11 is pin number and 100 is max range of [PWM.](https://hackatronic.com/pwm-controller-for-dc-motor-using-555-timer-ic/) p.start(0) //Starting point of PWM signal you can select any value between 0 to 100.

while True:

for x in range (0, 100, 1): //Increasing brightness of LED from 0 to 100 p.ChangeDutyCycle(x)

time.sleep(0.1)

for x in range (100, 0, -1): //fading brightness of LED from 100 to 0 p.ChangeDutyCycle(x)

time.sleep(0.1)

# Collecting Sensor Data

**a) DHT Sensor interface**

# Connect the terminals of DHT GPIO pins of Raspberry Pi.

* **Import the DHT library using import Adafruit\_DHT**

# Read sensor data and display it on screen.

**Raspberry Pi DHT11 Humidity and Temperature Sensor Interface**

* + February 21, 2018
  + [By Administrator](https://www.electronicshub.org/author/elktros/)

In this project, we will learn about DHT11 Humidity and Temperature Sensor and how the Raspberry Pi DHT11 Humidity Sensor interface works. By

Interfacing DHT11 Temperature and Humidity Sensor with Raspberry Pi, you can implement a basic IoT Project like a simple Weather Station.





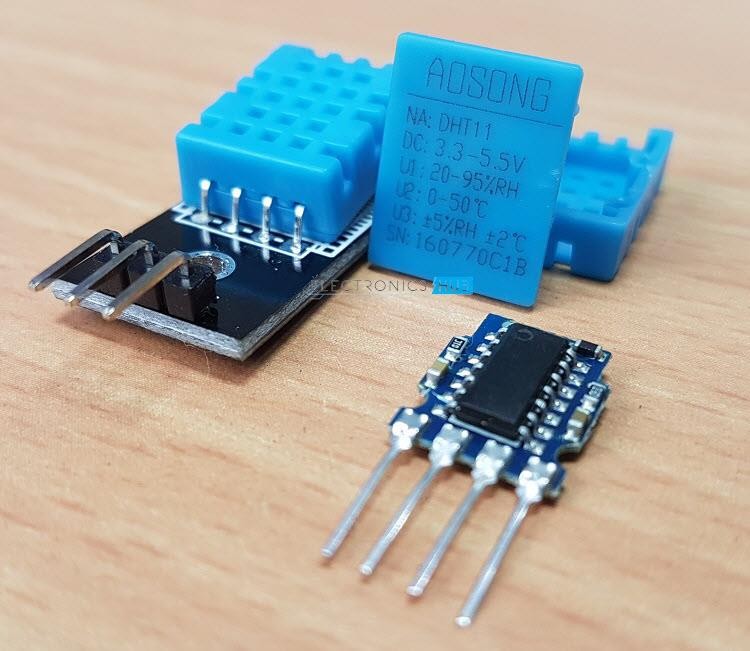
# Overview

DHT11 is a Digital Sensor consisting of two different sensors in a single package. The sensor contains an NTC (Negative Temperature Coefficient)

Temperature Sensor, a Resistive-type Humidity Sensor and an 8-bit Microcontroller to convert the [analog signals from these sensors and produce a](https://www.electronicshub.org/analog-circuits-and-digital-circuits/) [Digital Output.](https://www.electronicshub.org/analog-circuits-and-digital-circuits/)

I have already worked with the DHT11 Sensor in my [**DHT11 Humidity Sensor**](https://www.electronicshub.org/dht11-humidity-sensor-arduino/)[**on Arduino**](https://www.electronicshub.org/dht11-humidity-sensor-arduino/) Project. In that project, I have mentioned the Pin Configuration of the DHT11 Sensor, how to interface it with a Microcontroller and how the digital Output from the DHT11 Sensor can be decoded.

So, I suggest you to refer to that project once for more information on DHT11 Humidity and Temperature Sensor. I’ll explain a few thing which I have missed in the Arduino Project.



We know that the output from the DHT11 Sensor is Digital. But how exactly we can read this digital data?

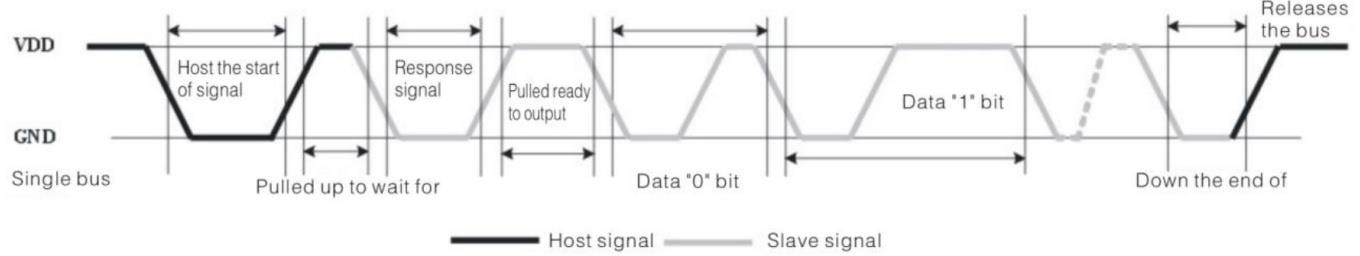
Reading Digital Output from DHT11

DHT11 uses a Single bus data format for communication. Only a single data line between an MCU like Arduino or Raspberry Pi and the DHT11 Sensor is sufficient for exchange of information.

In this setup, the Microcontroller acts as a Master and the DHT11 Sensor acts as a Slave. The Data OUT of the DHT11 Sensor is in open-drain configuration and hence it must always be pulled HIGH with the help of a 5.1KΩ Resistor.

This pull-up will ensure that the status of the Data is HIGH when the Master doesn’t request the data (DHT11 will not send the data unless requested by the Master).

Now, we will the how the data is transmitted and the data format of the DHT11 Sensor. Whenever the Microcontroller wants to acquire information from DHT11 Sensor, the pin of the



Microcontroller is configured as OUTPUT and it will make the Data Line low for a minimum time of 18ms and releases the line. After this, the Microcontroller pin is made as INPUT.

The data pin of the DHT11 Sensor, which is an INPUT pin, reads the LOW made by the Microcontroller and acts as an OUTPUT pin and sends a response of LOW signal on the data line for about 80µs and then pulls-up the line for another 80µs.

After this, the DHT11 Sensor sends a 40 bit data with Logic ‘0’ being a combination of 50µs of LOW and 26 to 28µs of HIGH and Logic ‘1’ being 50µs of LOW and 70 to 80µs of HIGH.

After transmitting 40 bits of data, the DHT11 Data Pin stays LOW for another 50µs and finally changes its state to input to accept the request from the Microcontroller.

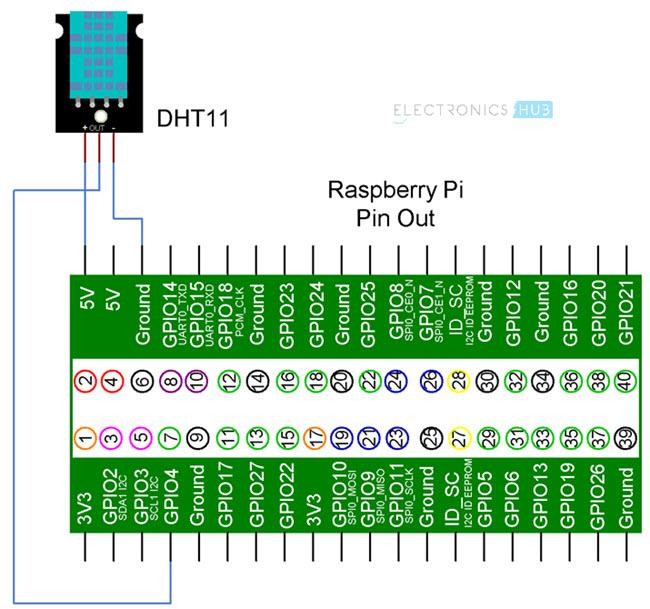
**NOTE:** We have implemented this logic while programming the Arduino. But for Raspberry Pi, we used a library that takes care of all these things.

# Raspberry Pi DTH11 Humidity and Temperature Sensor Interface

By interfacing the DHT11 Sensor with Raspberry Pi, you can build your own IoT Weather Station. All you need to implement such IoT Weather is a Raspberry Pi, a DHT11 Humidity and Temperature Sensor and a Computer with Internet Connectivity.

# Circuit Diagram

The following is the circuit diagram of the DHT11 and Raspberry Pi Interface.



# Components Required

* + Raspberry Pi 3 Model B
  + DHT11 Temperature and Humidity Sensor
  + Connecting Wires
  + Power Supply
  + Computer

# Circuit Design

If you observe the circuit diagram, there is not a lot of stuff going on with respect to the connections. All you need to do is to connect the VCC and GND pins of the DHT11 Sensor to +5V and GND of Raspberry Pi and then connect the Data OUT of the Sensor to the GPIO4 i.e. Physical Pin 7 of the Raspberry Pi.

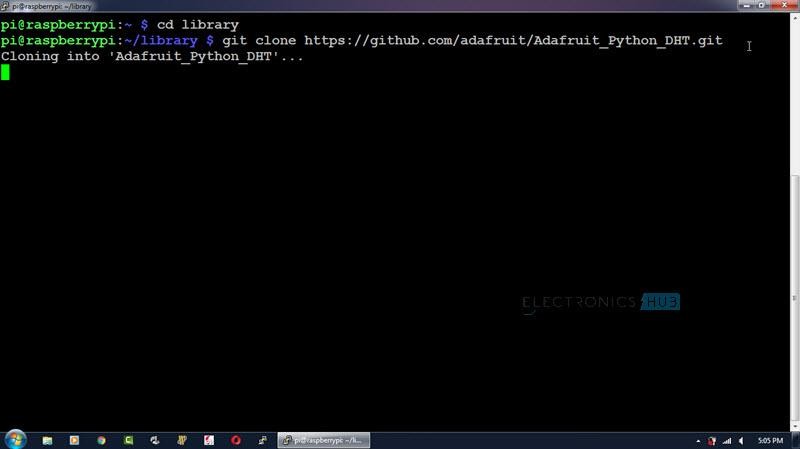
# Installing DTH11 Library

Since we are using a library called Adafruit\_DHT provided by Adafruit for this project, we need to first install this library into Raspberry Pi.

First step is to download the library from GitHub. But before this, I have created a folder called ‘library’ on the desktop of the Raspberry Pi to place the downloaded files. You don’t have to do that.

Now, enter the following command to download the files related to the Adafruit\_DHT library.

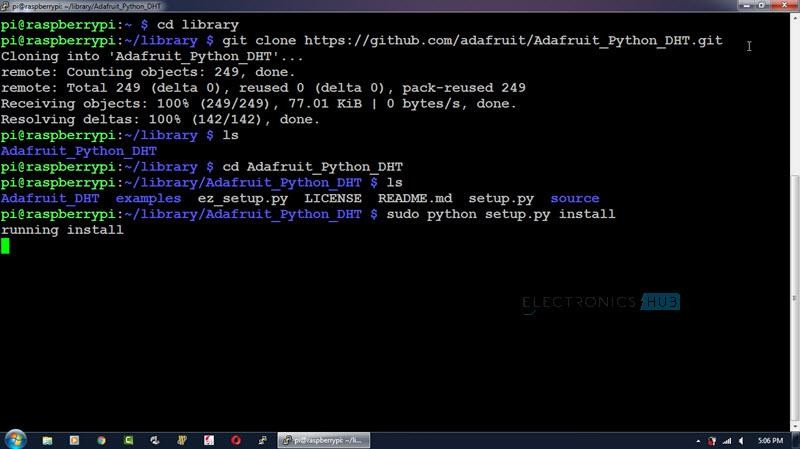
git clone https://github.com/adafruit/Adafruit\_Python\_DHT.git



All the contents will be downloaded to a folder called ‘Adafruit\_Python\_DHT’. Open this directory using cd Adafruit\_Python\_DHT. To see the contents of this folder, use ‘ls’ command.

In that folder, there is file called ‘setup.py’. We need to install this file using the following command.

sudo python setup.py install



# Code

As we are using the library Adafruit\_DHT for this project, there is nothing much to do in the Python Programming part. All you need to do is to invoke the library with the Sensor and GPIO Pin and print the values of Temperature and Humidity.

|  |
| --- |
| import sys |
| import Adafruit\_DHT |
| import time |
| while True: |
| humidity, temperature = Adafruit\_DHT.read\_retry(11, 4) |
| print 'Temp: {0:0.1f} C Humidity: {1:0.1f} %'.format(temperature, humidity) |
| time.sleep(1) |

[view rawRaspberry\_Pi\_DHT\_11.py](https://gist.github.com/elktros/f9f1c9e8475956d43eba95e8681676c7/raw/f0e0e2c64d15aca4ec24069e58a11d76e7c753ae/Raspberry_Pi_DHT_11.py) hosted with ❤ by [GitHub](https://github.com/)

# Working

Make the connections as per the circuit diagram and install the library. Use the above python program to see the results.

