# Objective

To start up your Raspberry Pi (Raspi) and run your first program on Raspi. The objectives of this project tutorial include:

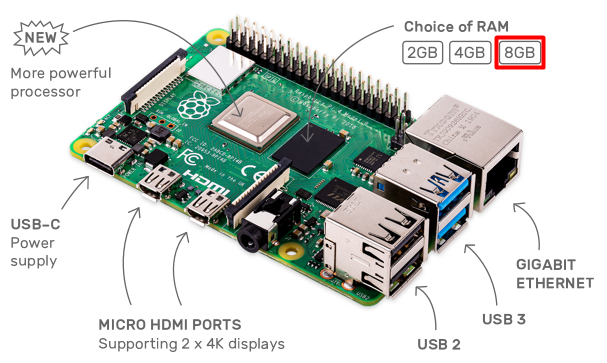
(1) Learn the basic knowledge about Raspi；

(3) Install Raspi OS；

(4) Run a program on Raspi.

# Components

1. Raspi Board and Power；



1. A 16GB SD Card and a SD Card Reader；



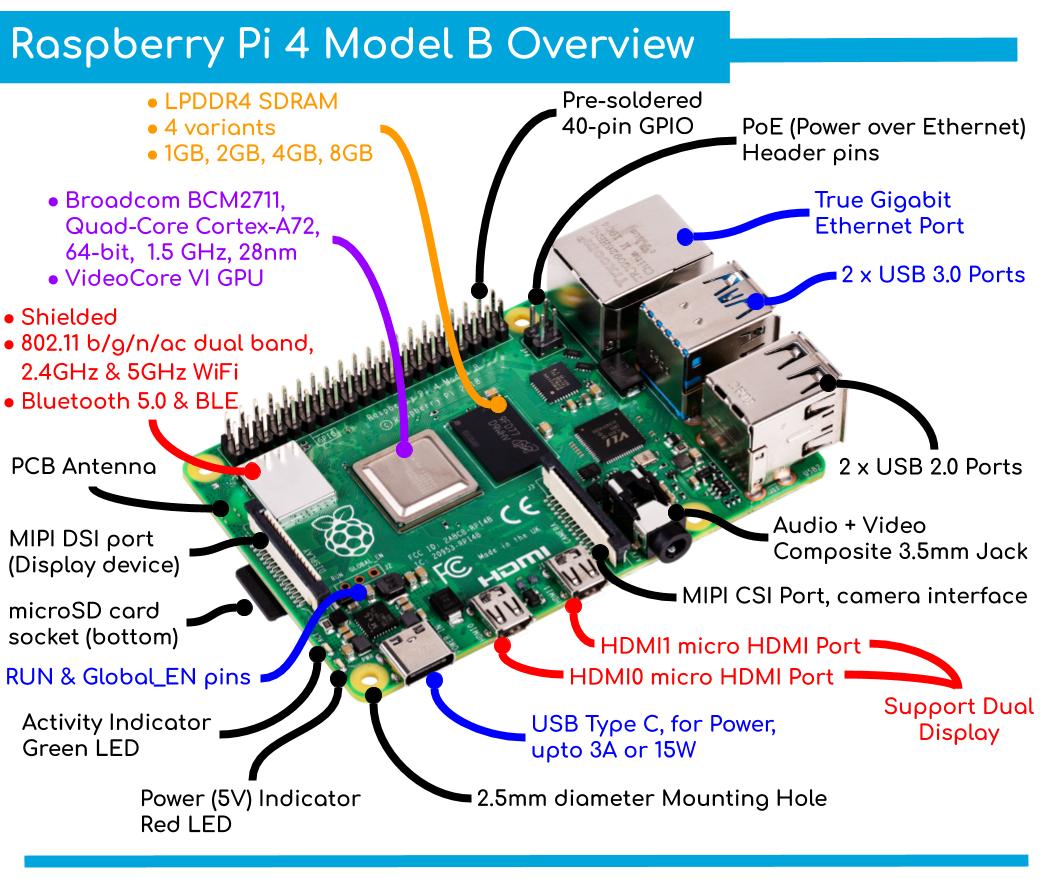
1. HDMI Screen, USB keyboard, USB mouse and Cable





# Principle of The Experiment

## Introduction of Raspi 4B



## BCM2711 SoC of Raspi 4B

BCM2711 is the core system on chip (SoC) of the Raspi 4 Model B. It integrates the powerful quad-core ARM A72 core and has a greatly improved GPU feature set with fast input/output, due to the incorporation of a PCIe link that connects the USB 2 and USB 3 ports, and a natively attached Ethernet controller.

The ARM cores are capable of running at up to 1.5 GHz, making the Pi 4 about 50% faster than the Raspi 3B+. The new VideoCore VI 3D unit now runs at up to 500 MHz. The ARM cores are 64-bit, and while the VideoCore is 32-bit, there is a new Memory Management Unit, which means it can access more memory than previous versions.

A datasheet for the BCM2711 can be found [here](https://datasheets.raspberrypi.org/bcm2711/bcm2711-peripherals.pdf).

* **Processor**: Quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5 GHz. See [Wikipedia page](https://en.wikipedia.org/wiki/ARM_Cortex-A72) on the A72 for more details.
* **Memory**: Accesses up to 8GB LPDDR4-2400 SDRAM (depending on model)
* **Caches**: 32 KB data + 48 KB instruction L1 cache per core. 1MB L2 cache.
* **Multimedia**: H.265 (4Kp60 decode); H.264 (1080p60 decode, 1080p30 encode); OpenGL ES, 3.0 graphics
* **I/O**: PCIe bus, onboard Ethernet port, 2 × DSI ports (only one exposed on Raspi 4B), 2 × CSI ports (only one exposed on Raspi 4B), up to 6 × I2C, up to 6 × UART (muxed with I2C), up to 6 × SPI (only five exposed on Raspi 4B), dual HDMI video output, composite video output.

## Raspi OS

Raspi OS is the recommended operating system for normal use on a Raspi. It’s a free operating system based on Debian (Linux), optimised for the Raspi hardware. Raspi OS comes with over 35,000 packages: precompiled software bundled in a nice format for easy installation on your Raspi.

Raspi OS is a community project under active development, with an emphasis on improving the stability and performance of as many Debian packages as possible.

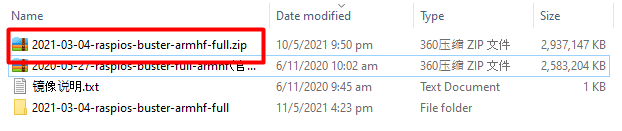
# Experiment Steps

Now, let’s start the experiment step by step!

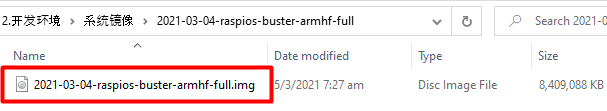
## Installing operating system images

In this part, we will install a Raspi operating system on the given SD card. In this step, we will need a Window/Mac OS computer with an SD card reader.

1. Unzip the system image package “**2021-03-04-raspios-buster-armhf-full.zip**”.

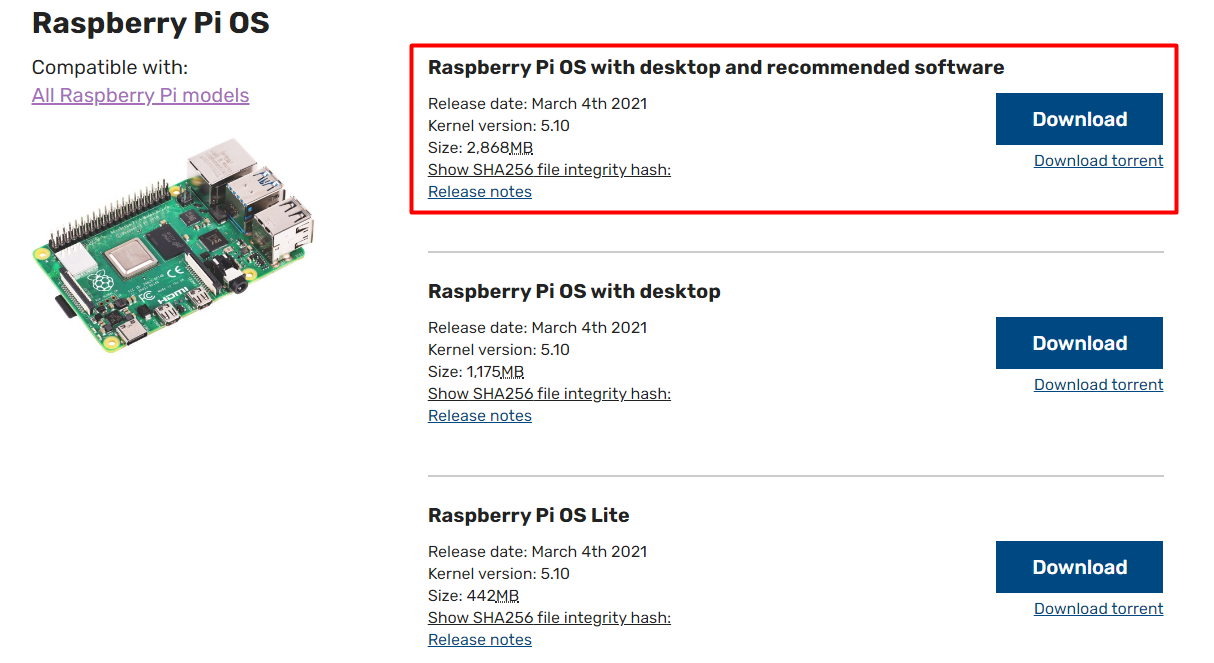


Then, we will get the image file “**2021-03-04-raspios-buster-armhf-full.img**” in the new folder.



If you haven’t the operating system image file (“**2021-03-04-raspios-buster-armhf-full.zip**”), please download it from [*Raspi website*](https://www.raspberrypi.org/software/operating-systems/#raspberry-pi-os-32-bit).

There are three different versions. We select the **Raspi OS with desktop and recommended software** in our experiments which is highlighted as below.

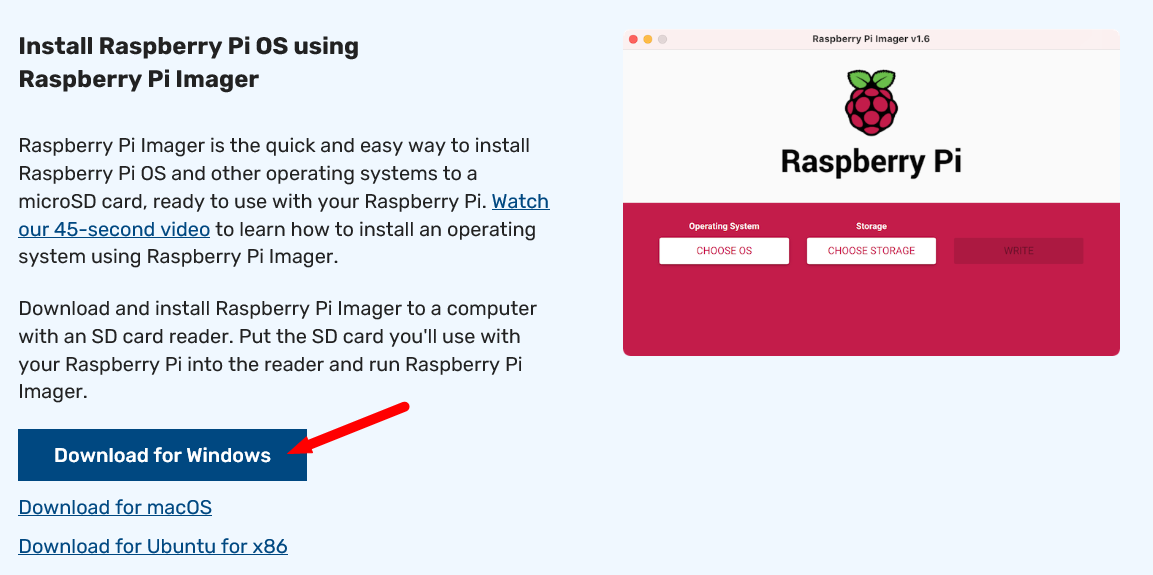
]

1. Insert the given SD card into the SD card reader, and insert them into the computer through the USB interface.

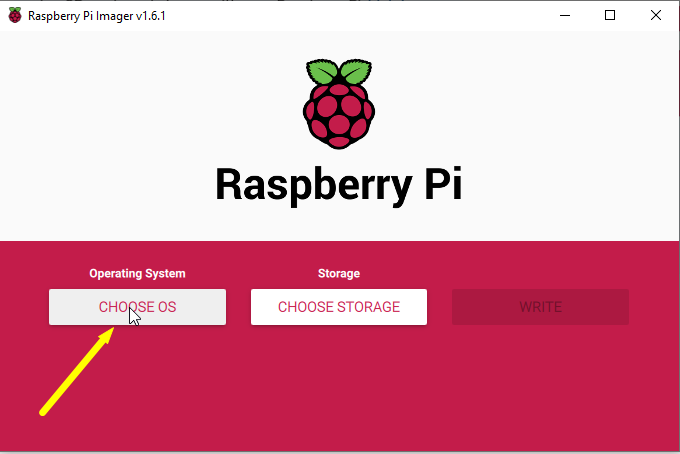
(Note: Make sure the SD card have enough space (>8GB))

1. Click on this link: [Raspberry Pi Imager](https://www.raspberrypi.org/downloads/) , Download and install the latest version of Raspberry Pi Imager.

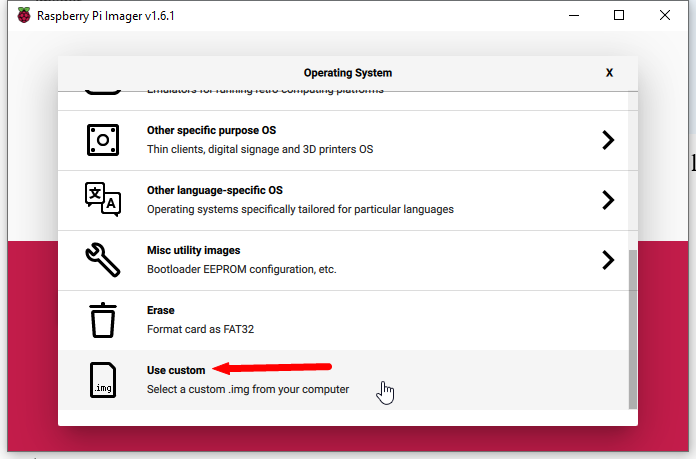
If you are using window computer, selecting Download for Windows as below. Otherwise, choose according to the computer system you actually use.



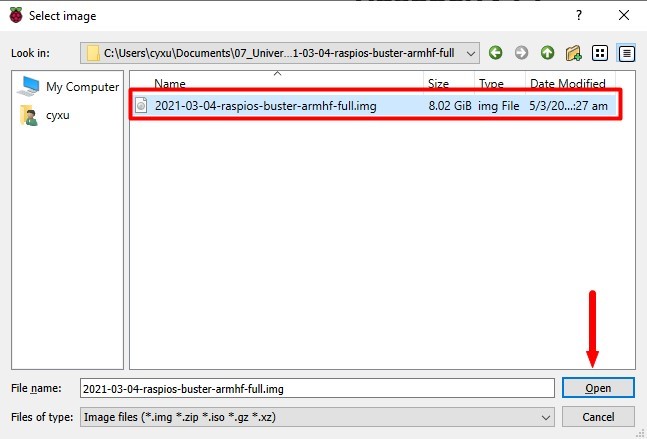
1. After the download, open Raspberry Pi Imager, click CHOOSE OS.



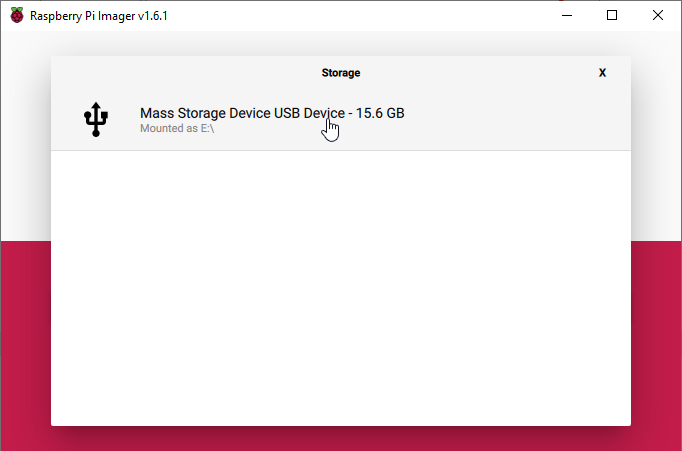
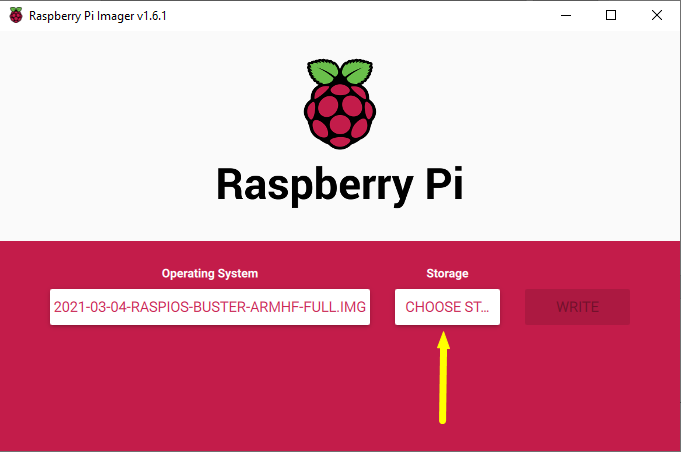
1. Click Use custom button at the bottom of the drop down box.



1. Select image file “**2021-03-04-raspios-buster-armhf-full.img**” obtained in step 1), then click Open.



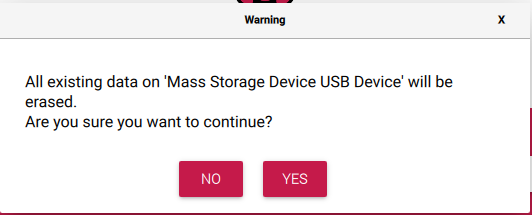
1. Click CHOOSE STORAGE and select Mass Storage Device USB Device



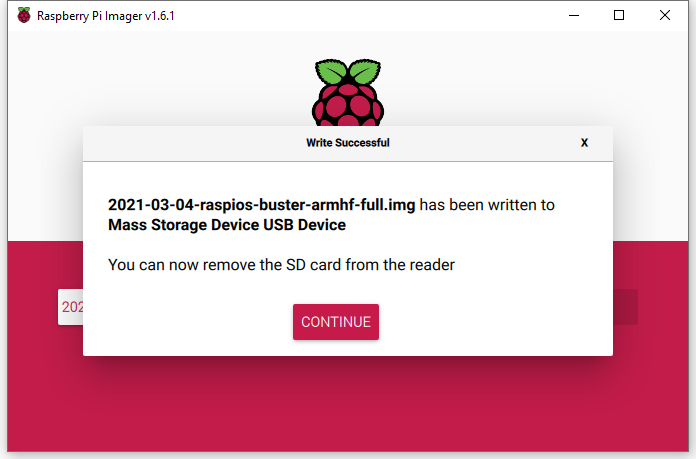
1. Click WRITE and select YES in the pop-up warning window, then wait for writing finish!

The writing procedure would consume 10-30 minutes.





1. Click CONTINUE and close Raspberry Pi Imager, then remove SD card reader from computer.

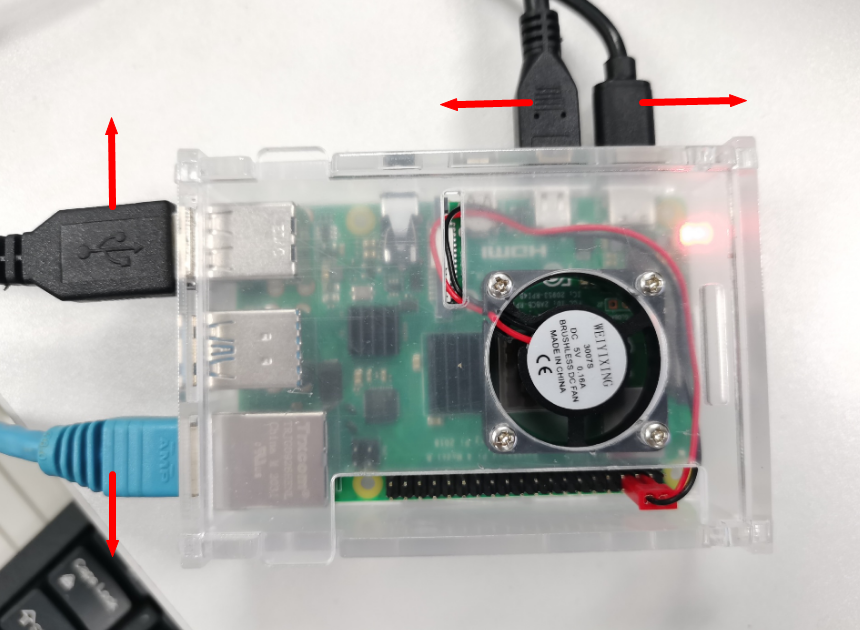


## Connect Hardware

Before connecting hardware, we’d better install the acrylic case for Raspi to prevent static electricity from damaging the Raspi board.

Warning: Never use you hand touch Pins of Raspi directly.

1. Connect Raspi Board, Power, HDMI Screen, Network, Keyboard and Mouse as the following figure.



**HDMI Cable**

**Power**

**Keyboard and**

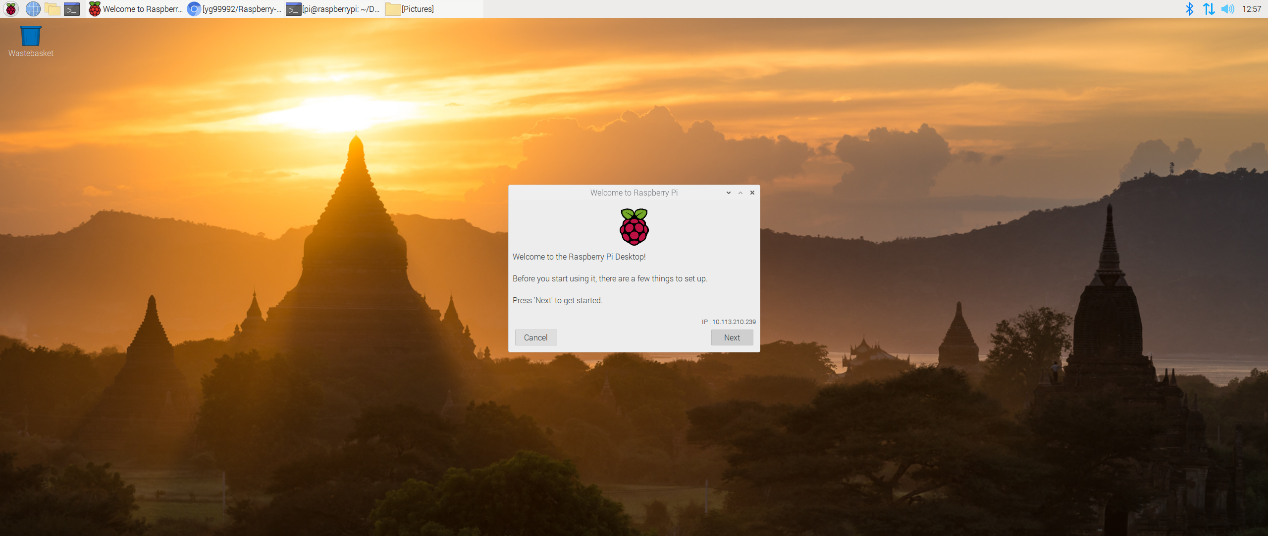
**Mouse**

**Network Wire**

1. Remove the SD card from the card reader and insert it into the card slot on the back of the Raspi and power it up.

If the Raspi desktop appears as below after a few minutes, it means your Raspi start up successfully! Congratulation!!!

You should connect HDMI screen and keep it open before power up Raspi. If not, the HDMI screen may can’t be lighted up and you have to reboot Raspi OS.



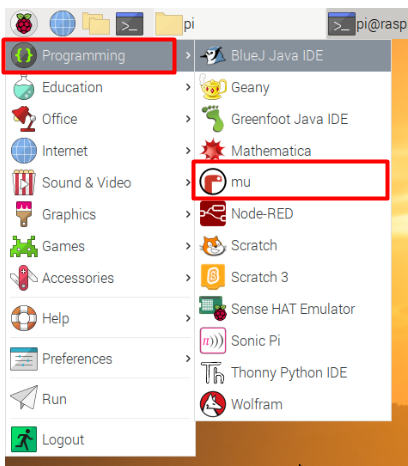
As the system startup first time, it will remind us set the Country, Language, Time zone, Wireless and so on. The system update can be skipped. Then you can restart the system and start the exploration of Raspi.

The default username of Raspi OS is pi, with password raspberry. You can change the default password straight away to ensure the Raspi is secure.

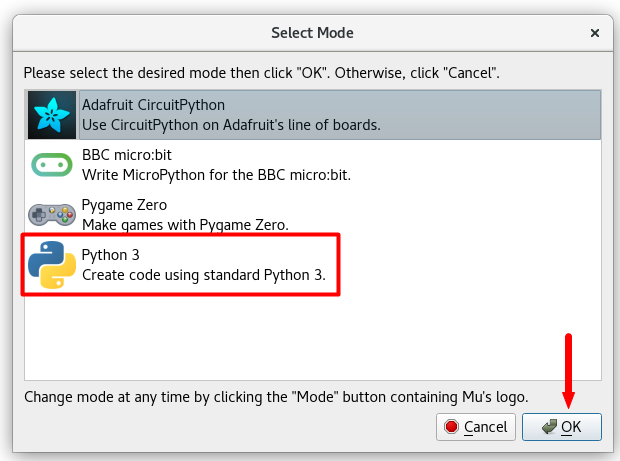
## Write the First Python Code

Mu is a Python code editor for beginner programmers. It’s very easy to create, run and revise your Python programs.

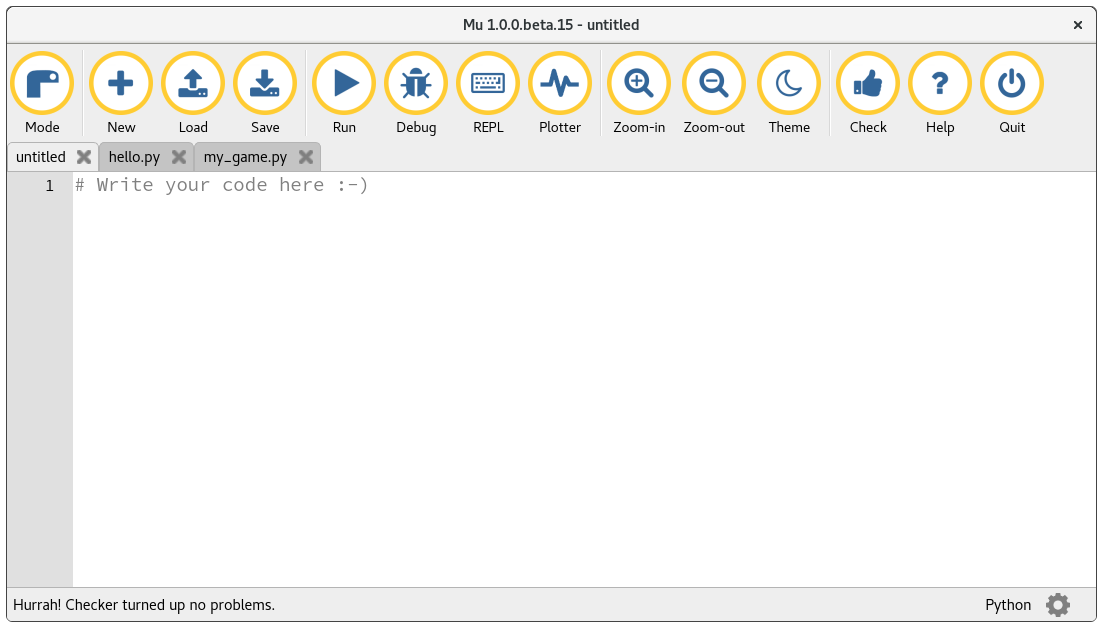
1. Start-up Mu from the Raspi menu: **Menu 🡪 Programming 🡪 mu**, as below



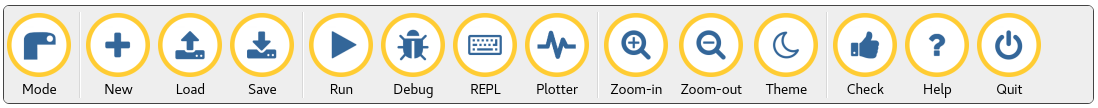
If we start-up Mu first time, it will remind us select mode. We select Python 3 and press OK button.



Mu is very simple and easy to understand. Here’s what it looks like:



Button description



***Mode:*** *We can click it to change Mu’s current mode. If you want to know more about modes, there is a* [***tutorial about modes***](https://codewith.mu/en/tutorials/1.1/modes) *for you to read.*

***New:*** *create a new blank file.*

***Load:*** *open a file selector to choose a file to load into Mu.*

***Save:*** *save the file to your computer’s hard drive.*

*If the file has no name, you’ll be asked to give one.*

***Run:*** *run the current script.*

*When this happens the textual input and output of the program is displayed in a panel between the text editor and Mu’s footer. When the code is running the “Run” button turns into a “Stop” button. Click “Stop” to force your code to exit in a clean way.*

***Debug:*** *start Mu’s visual debugger.*

*The debugger starts running your code (just like the “Run” button) but does so in a special way that allows you to pause it, look at the state of things in your program and step through your code so you can follow how Python is interpreting your program. This is extraordinarily useful if your code has bugs in it.*

***REPL:*** *open a new panel between the text editor and Mu’s footer.*

*The term “REPL” is an acronym and stands for “Read, Evaluate, Print, Loop”, which succinctly describes what the panel does for you. It reads interactive lines of Python which you type, evaluates, prints out any result it has for you and then loops back to wait for your next Python instruction.*

***Plotter:*** *open the plotter pane between the text editor and Mu’s footer. This is an easy way to visualise numeric data that your program may produce. If you’re running a program that prints numbers in a Python tuple (i.e. the output looks like this: (1, 2, 3)) then the plotter will display these numbers as a graph.*

***Zoom-In:*** *make the text bigger*

***Zoom-Out:*** *make the text smaller*

***Theme:*** *toggle through three different display “themes”:*

*“Day” - a light theme that’s easy on the eyes (used in the example images).*

*“Night” - a dark theme that makes you look like a coder in a Hollywood film.*

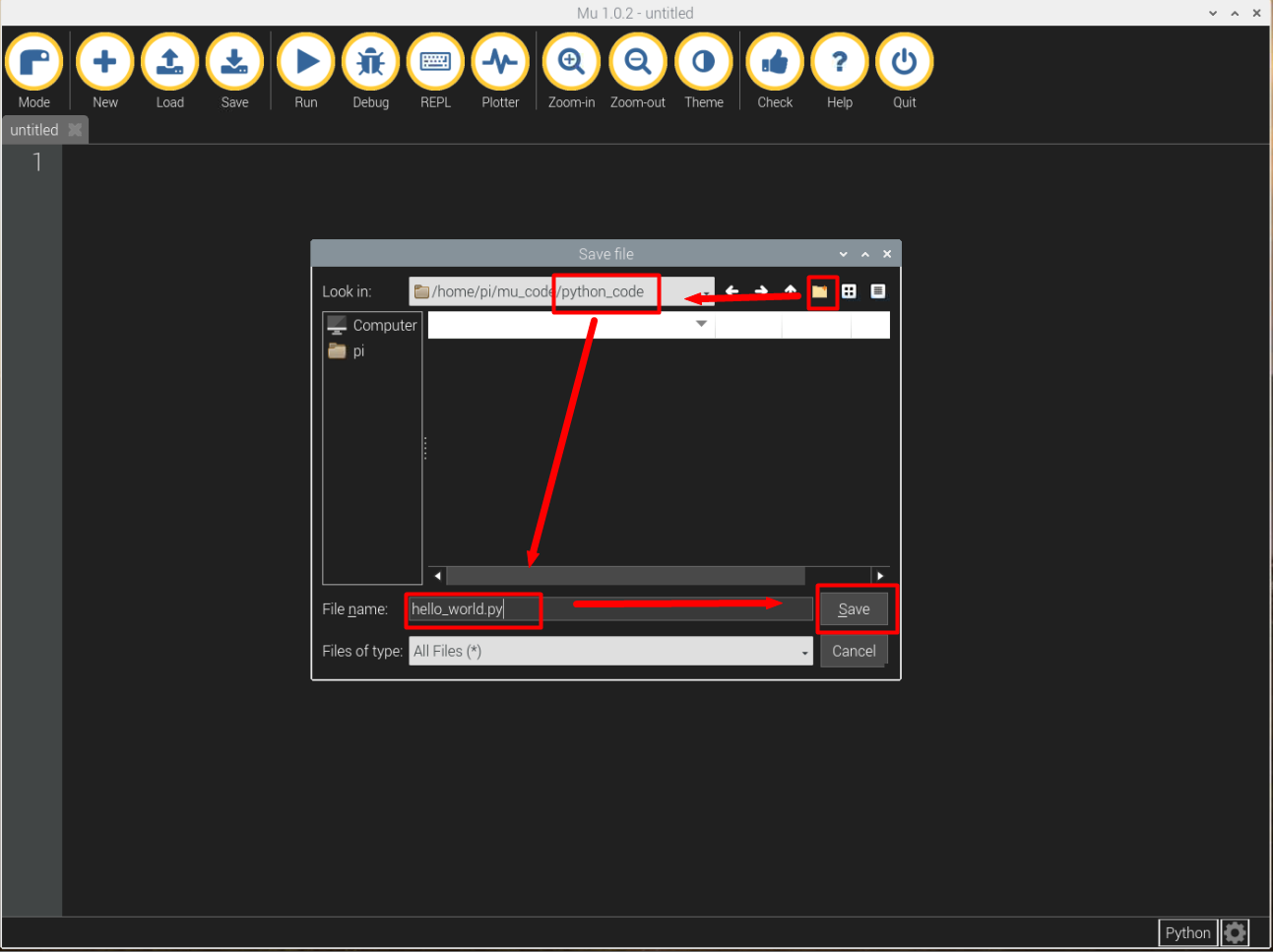
*“High Contrast” - a black and white theme for those who need help seeing Mu’s user interface.*

***Check:*** *analyses your code and suggests ways to improve it.*

***Help:*** *opens your default browser on Mu’s help page.*

***Quit:*** *stops the editor. You may be asked to save your work if you haven’t done so already.*

1. We can select dark theme by clicking the Theme button, which is suitable for our eyes.
2. Click the New button, then before typing anything, click the Save button and create a new folder named “python\_code”, then **double click** the new folder and give your file a name (for example “hello\_world.py”) before save it.

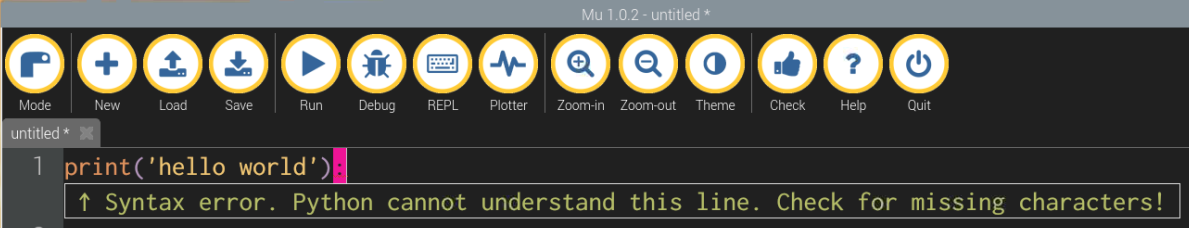


1. Type the following Python code into the text area:

print('Hello World!')

1. Before running the program, we should ensure there is no grammar error by click the Check button.

For example, there has an error char at the end of the sentence, and Mu will remind us the error line as below.



If the prompt information is Syntax error, we must correct it. While if it isn’t error information, we can neglect it if we ensure it has not effect on our program.

1. Click the Run button and the output of your program will be displayed in the “output” area.



1. Click the “Stop” button to return to editing your code.
2. Type the following code manually

a = 'Hi!'

b = a \* 3

print(b)

for char in a:

print(char)

name = "Joe"

if len(name) > 3:

print("Nice name,")

print(name)

else:

print("That's a short name,")

print(name)

n = 0

for i in range(5):

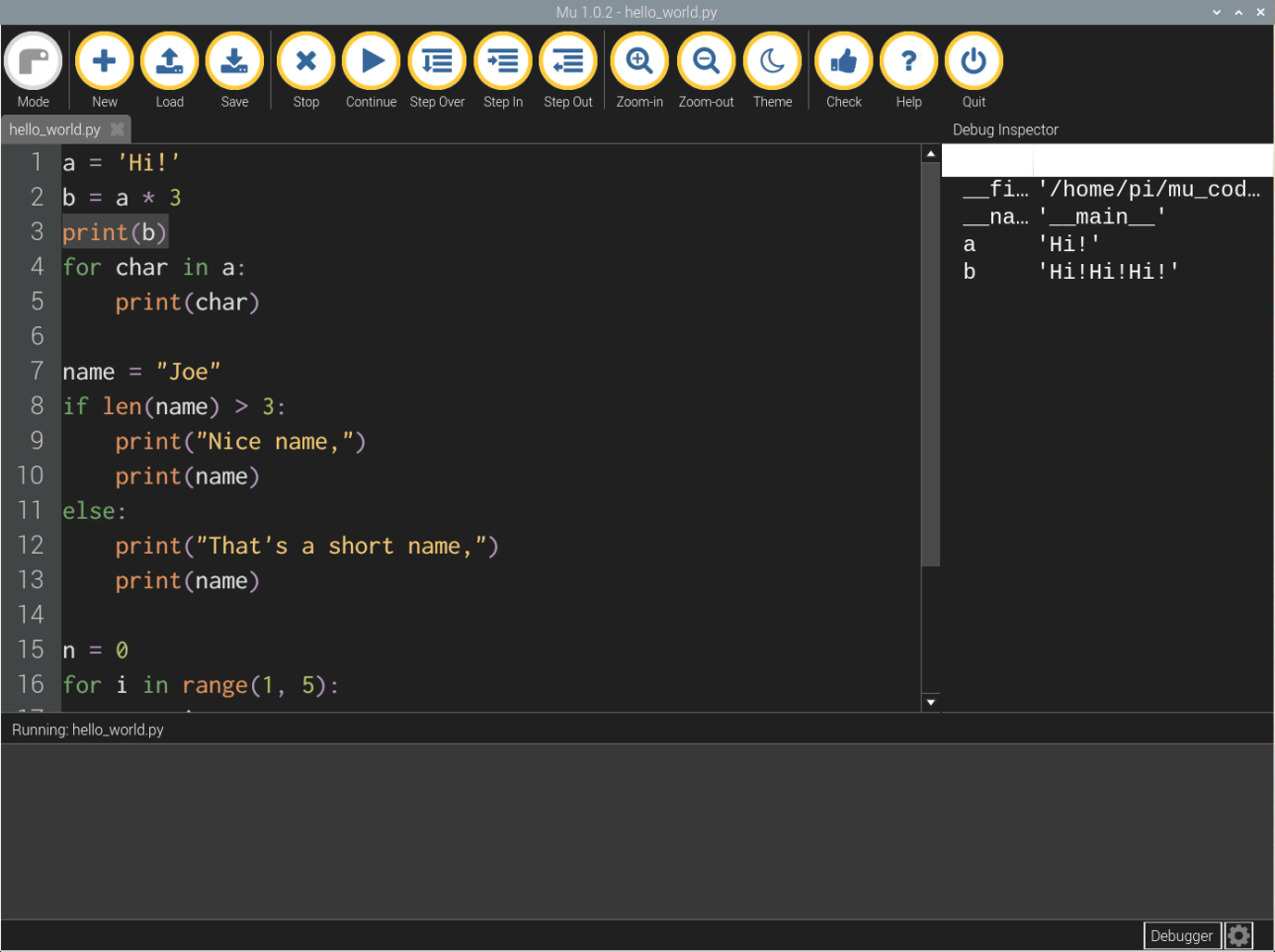
n += i

print("The sum of the numbers 1 to 4 is: %d" % n)

dict\_test = {i: i \*\* 3 for i in range(5)}

print(dict\_test)

1. Click the Debug button to see what will happen.
2. Click Step Over button when the first line of code is highlighted and see the changes of the “debug inspector” area on the right of the window.



1. Continue to click Step Over button until the highlighted bar moves to the last line. Every time you click the Step Over button, pay attention to the “debug inspector” area and the “output” area.

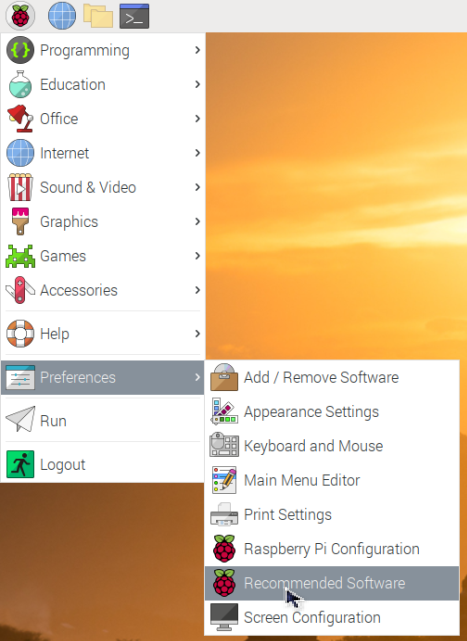
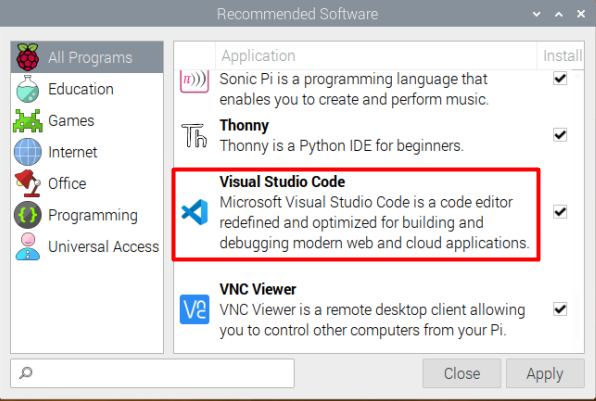
Step Over - runs the next line of code in your program; Step In - if the next line of code is a function, it will ‘step into’ the function and run it; Step Out - if the program is currently running a function, it will ‘step out’ of the function and return to the line of code that called the function.

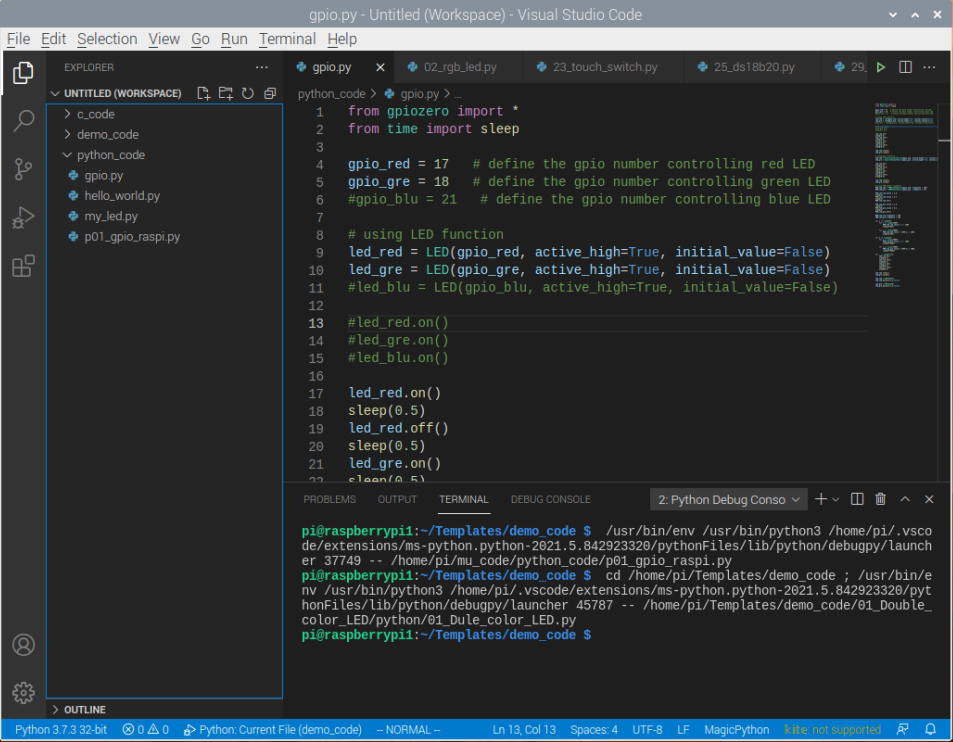
As not all problems with code are syntax errors (which Mu usually recognises). Some errors in your code will be bugs, meaning your program runs fine, but it doesn’t do what you want it to do. So the Debug function is usually used to find out bugs in the program.

1. Click Stop to stop debug.
2. Click Quit to quit Mu

## Extension

1. VScode is a free, powerful and flexible integrated develop environment (IDE) that can be used to replace Mu. It can be used to write and debug python/C/C++/Java code. It supports all kinds of plug-ins, such as Vim, Git, Auto Complete tools and so on. You can install and learn how to use it by yourself.



1. Raspi also support remote access using VNC or SSH. For more information, please click [here](https://www.raspberrypi.org/documentation/remote-access/);
2. For more documents about Raspi, please click [here](https://www.raspberrypi.org/documentation/);
3. For more knowledge about Python language, please click [here](https://docs.python.org/3/index.html);
4. For more tutorial about Mu IDE, please click [here](https://codewith.mu/en/tutorials/1.1/).

**End of Tutorial 1**