

# Configuring the Operating Environment For Micro Python

1. Introduce of Micro Python.....	2
2. Installing Thonny.....	2
2.1 Downloading Thonny.....	2
2.2 Installing on Windows.....	4
3. Basic Configuration of Thonny.....	9
4. Burning Micropython Firmware (Important).....	11
4.1 Downloading Micropython Firmware.....	11
4.2 Burning a Micropython Firmware.....	12
5. Connect Pico with Thonny.....	13
6. Testing codes.....	18
6.1 Testing Shell Command.....	18
6.2 Running Online.....	19
6.3 Running Offline.....	22
7. Thonny Common Operation.....	31
7.1 Uploading Code to Raspberry Pi Pico.....	31
7.2 Downloading Code to Computer.....	31
7.3 Deleting Files from Raspberry Pi Pico's Root Directory.....	32
7.4 Deleting Files from your Computer Directory.....	32
7.5 Creating and Saving the code.....	33
8. Make your suggestion and get support.....	36

## 1. Introduce of Micro Python

MicroPython is a complete Python compiler and runtime that runs on hardware for microcontrollers such as raspberry PI Pico. The user is presented with an interactive prompt (REPL) to execute the supported commands immediately. Includes a set of core Python libraries; MicroPython includes modules that allow programmers to access low-level hardware.

The original Kickstarter campaign released MicroPython as a development board "Pyboard" with STM32F4, and MicroPython supports many ARM-based product architectures. The ports supported by the mainline are ARM Cortex-M (many STM32 boards, TI CC3200/WiPy, Teensy boards, Nordic nRF family, SAMD21 and SAMD51), ESP8266, ESP32, 16-bit PIC, Unix, Windows, Zephyr and JavaScript. Second, MicroPython allows for rapid feedback. This is because you can use the REPL to interactively enter commands and get responses. You can even tweak the code and run it immediately instead of going through the code-compile-upload-execute loop.

While Python has the same advantages, for some microcontroller boards like the Raspberry PI Pico, they are small, simple, and have no memory at all to run Python. This is why MicroPython continues to evolve, retaining the main Python features and adding many new ones to work with these microcontroller boards.

Then Thonny is a Python IDE for beginners, Thonny comes with Python 3.7 built in, so just one simple installer is needed and you're ready to learn programming.

## 2. Installing Thonny

Thonny is a free, open-source software platform with compact size, simple interface, simple operation and rich functions, making it a Python IDE for beginners. In this tutorial, we use this IDE to develop Raspberry Pi Pico during the whole process.

Thonny supports various operating system, including Windows、Mac OS、Linux.

### 2.1 Downloading Thonny

Official website of Thonny: <https://thonny.org>

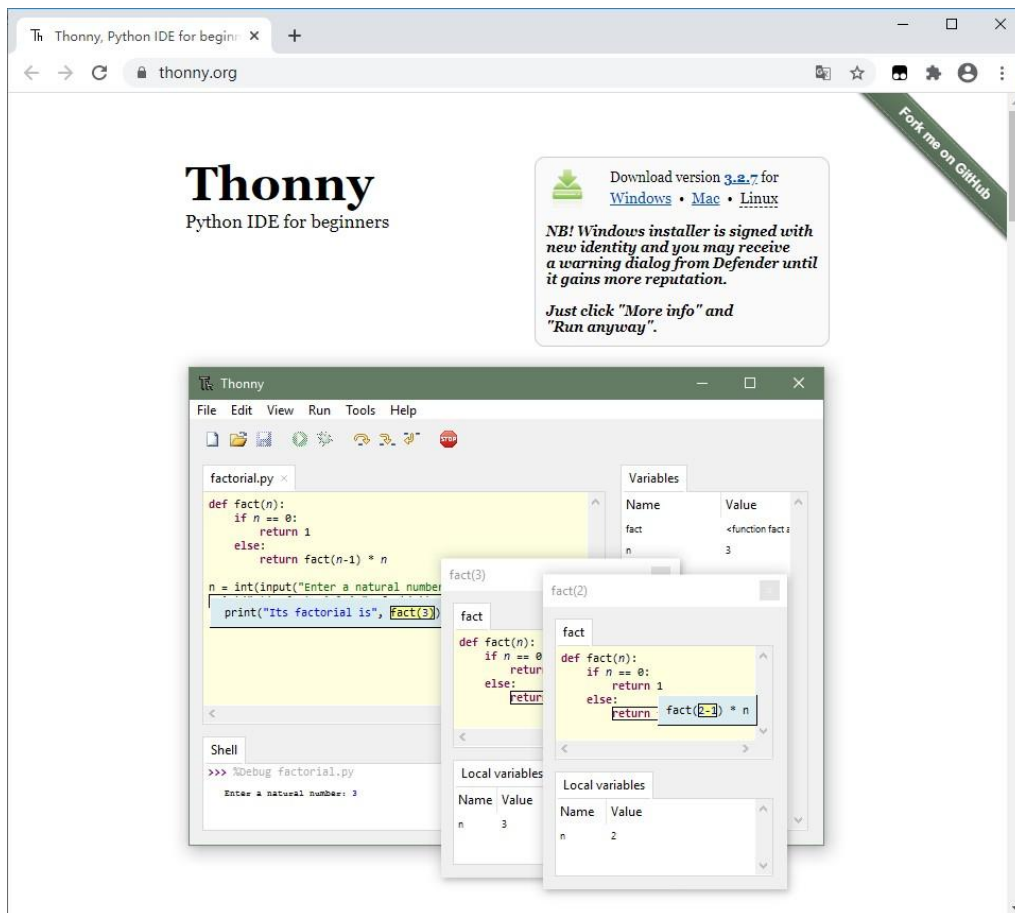
Open-source code repositories of Thonny: <https://github.com/thonny/thonny>

Follow the instruction of official website to install Thonny or click the links below to download and install. (Select the appropriate one based on your operating system.)

Operating System	Download links/methods
Windows	<a href="https://github.com/thonny/thonny/releases/download/v3.2.7/thonny-3.2.7.exe">https://github.com/thonny/thonny/releases/download/v3.2.7/thonny-3.2.7.exe</a>

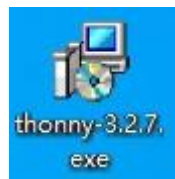
Mac OS	<a href="https://github.com/thonny/thonny/releases/download/v3.2.7/thonny-3.2.7.pkg">https://github.com/thonny/thonny/releases/download/v3.2.7/thonny-3.2.7.pkg</a>
Linux	<p>The latest version:</p> <p>Binary bundle for PC (Thonny+Python):</p> <pre>bash &lt;(wget -O - https://thonny.org/installer-for-linux)</pre> <p>With pip:</p> <pre>pip3 install thonny</pre> <p>Distro packages (may not be the latest version):</p> <p>Debian, Rasbian, Ubuntu, Mint and others:</p> <pre>sudo apt install thonny</pre> <p>Fedora:</p> <pre>sudo dnf install thonny</pre>

You can also open “[CKK0016-Main/Tutorial/Python/Python\\_Software](#)”, we have prepared it in advance.

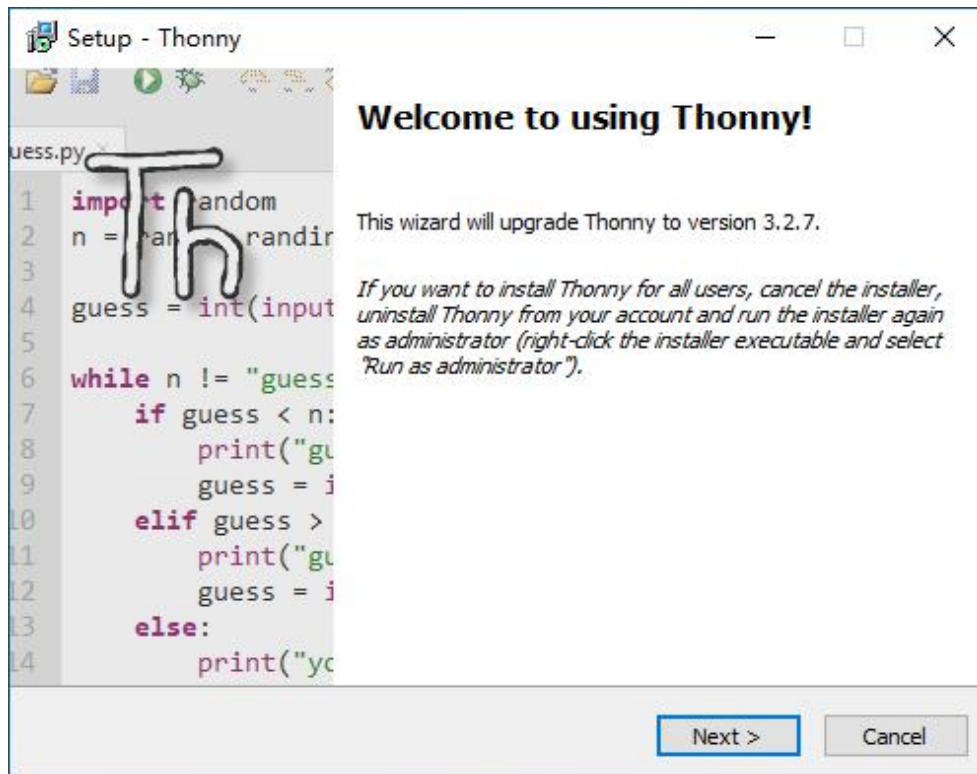


## 2.2 Installing on Windows

The icon of Thonny after downloading is as below. Double click "[thonny-3.2.7.exe](#)".

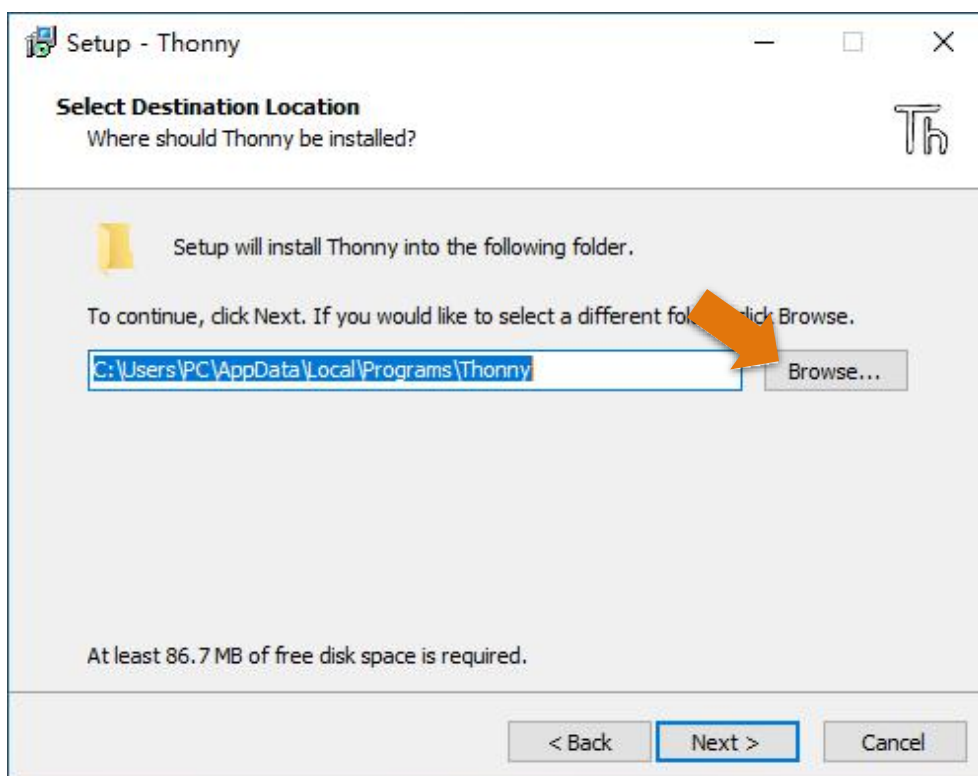


If you're not familiar with computer software installation, you can simply keep clicking “Next” until the installation completes.

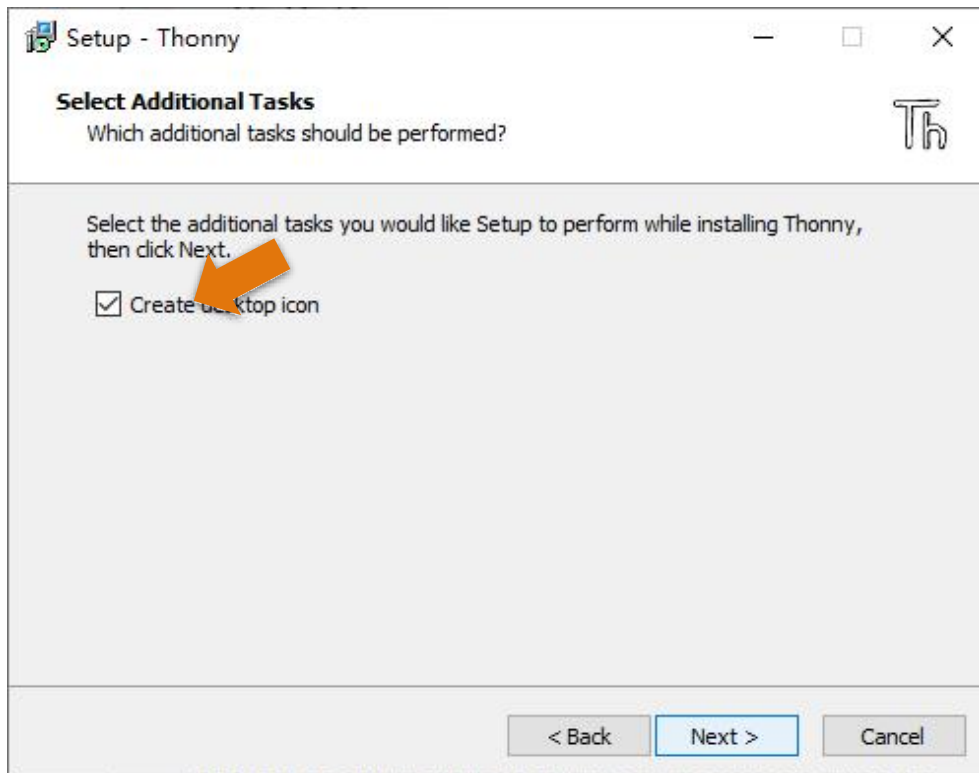


If you want to change Thonny's installation path, you can click “Browse” to modify it. After selecting installation path, click “OK”.

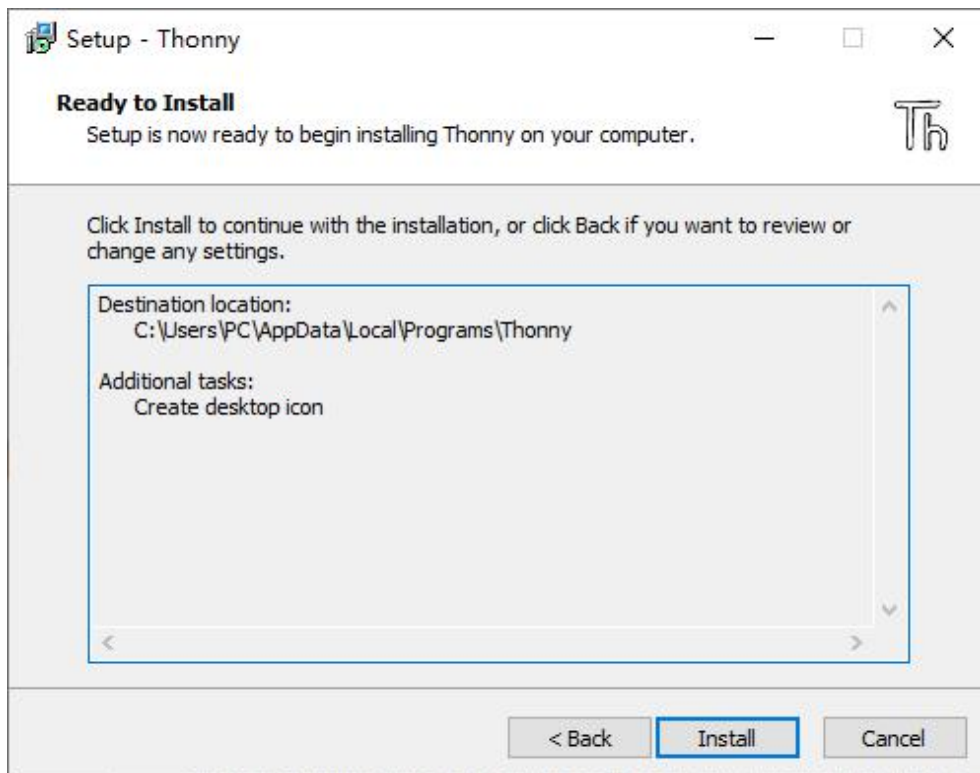
If you do not want to change it, just click “Next”.



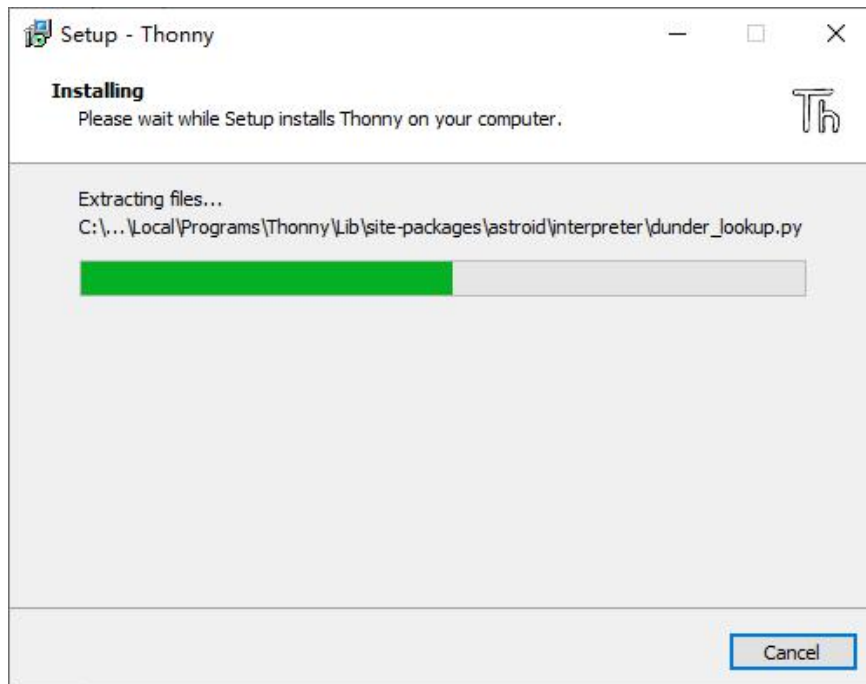
Check “[Create desktop icon](#)” and then it will generate a shortcut on your desktop to facilitate you to open Thonny later.



Click “[install](#)” to install the software.



During the installation process, you only need to wait for the installation to complete, and you must not click "Cancel", otherwise Thonny will fail to be installed.



Once you see the interface as below, Thonny has been installed successfully.



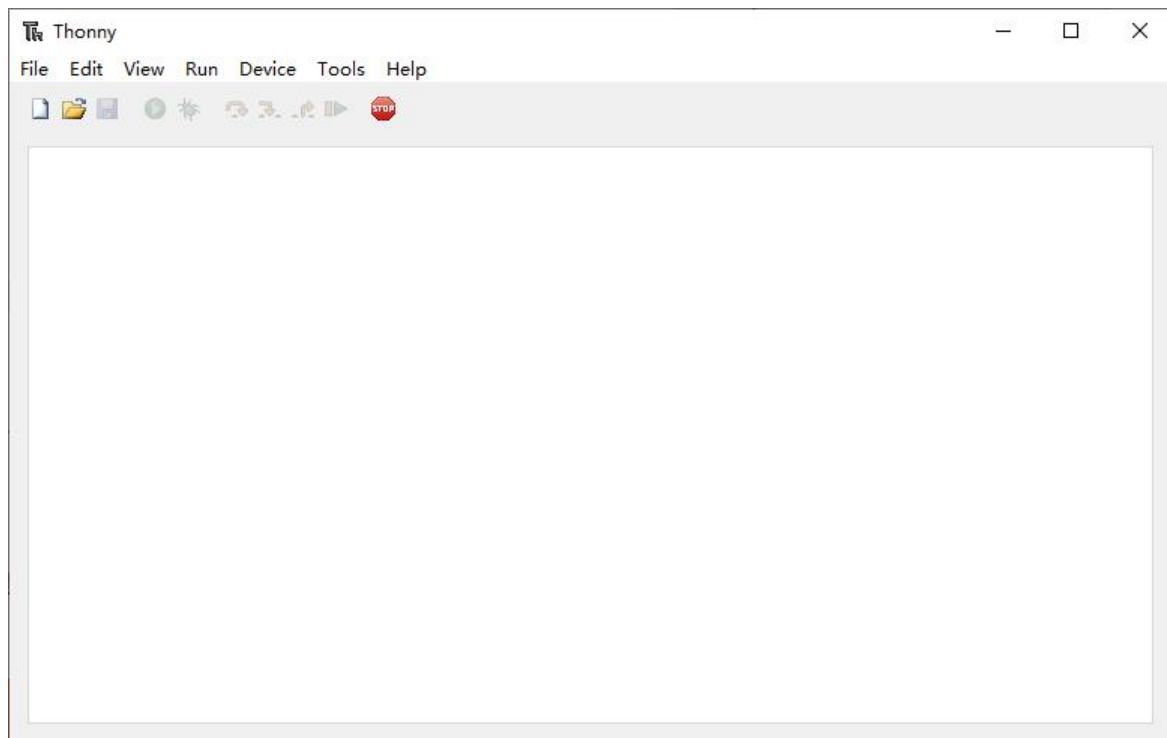
If you've checked "Create desktop icon" during the installation process, you can see the below icon on your desktop.



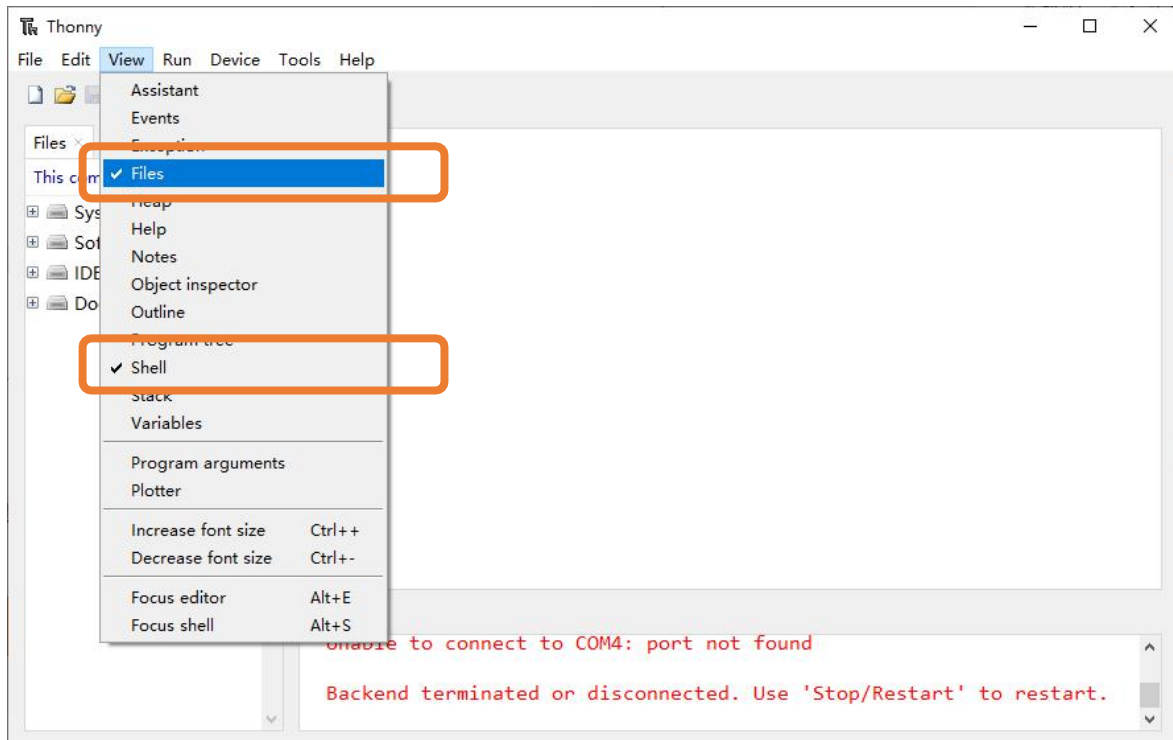


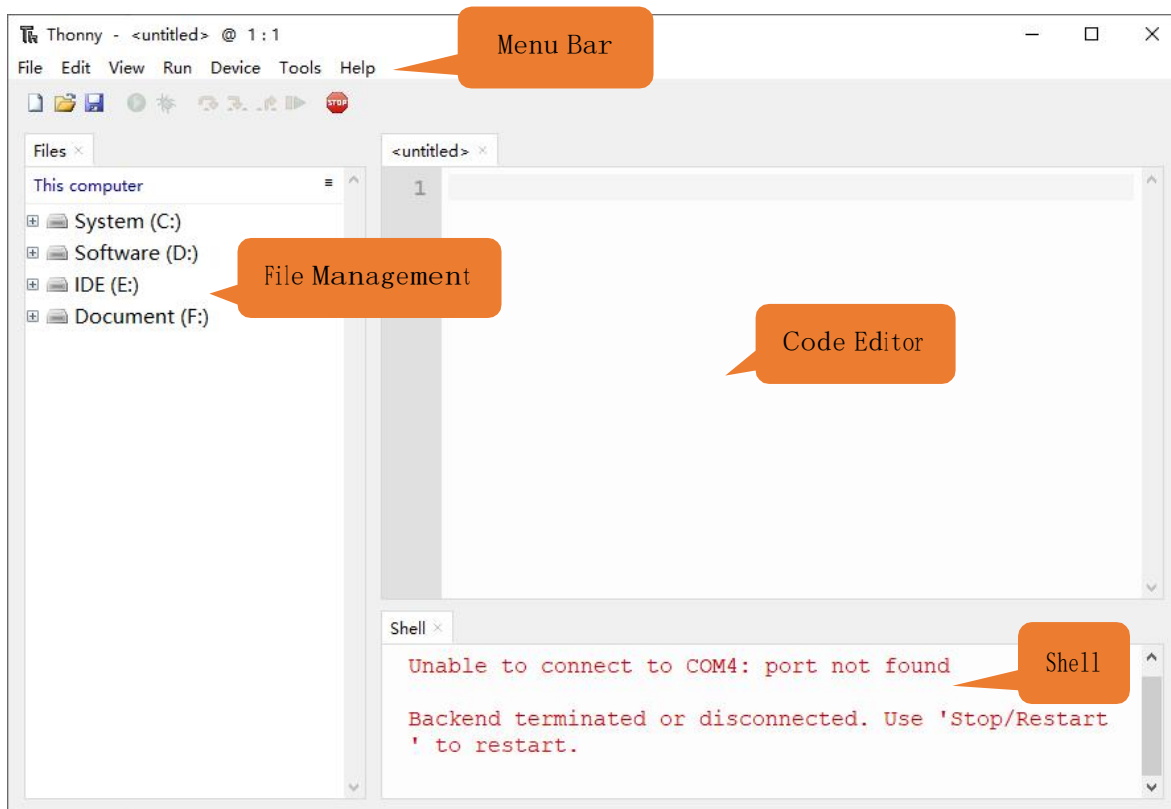
### 3. Basic Configuration of Thonny

Click the desktop icon of Thonny and you can see the interface of it as follows:



Select “View” “Files” and “Shell”.





## 4. Burning Micropython Firmware (Important)

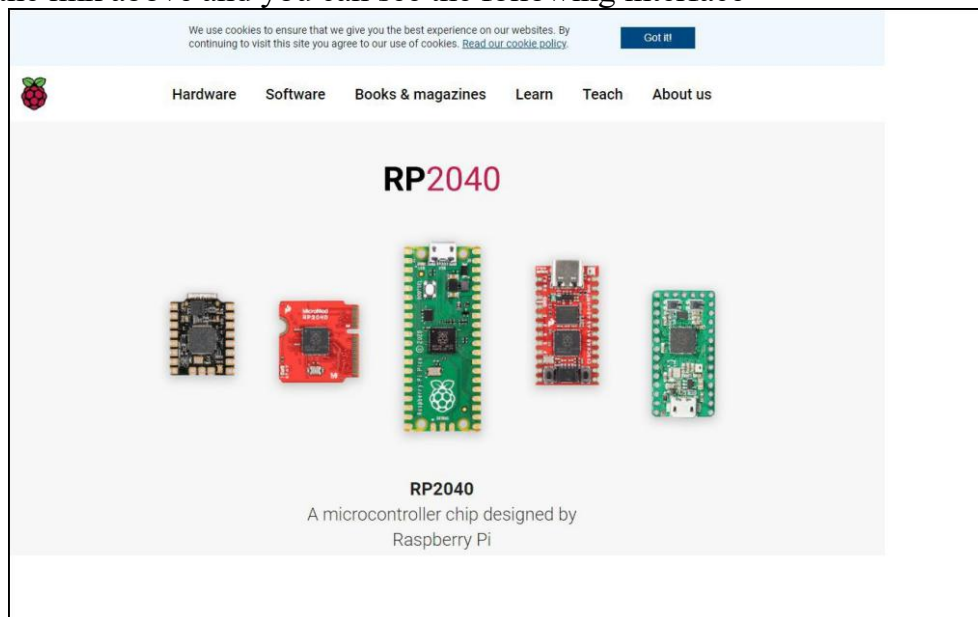
To run Python programs on Raspberry Pi Pico, we need to burn a firmware to Raspberry Pi Pico first.

### 4.1 Downloading Micropython Firmware

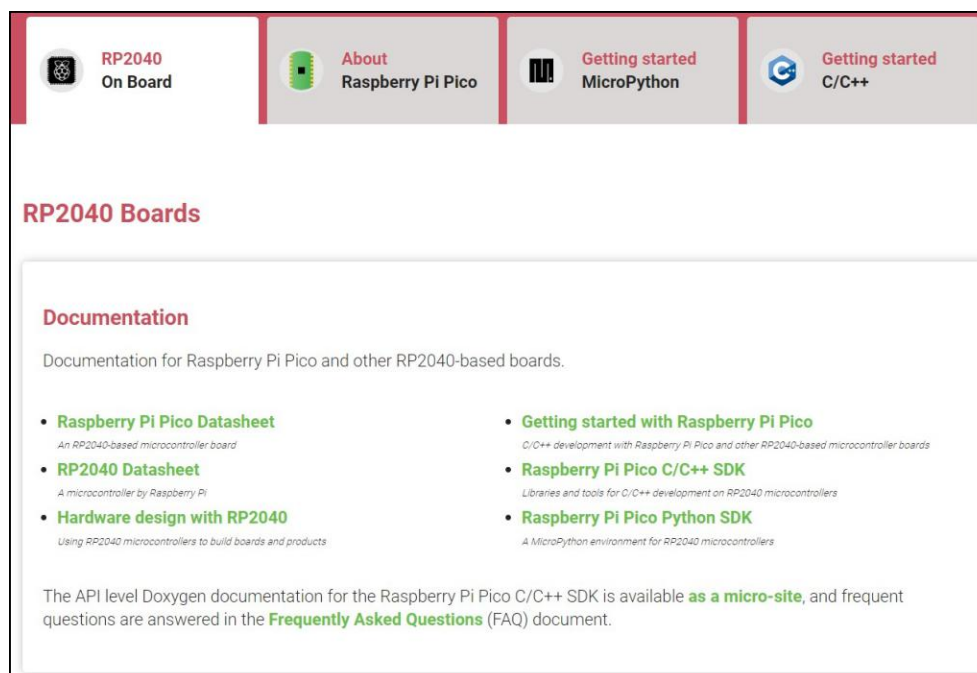
#### Option 1:

Raspberry Pi Pico official website: <https://www.raspberrypi.org/documentation/rp2040/getting-started/>

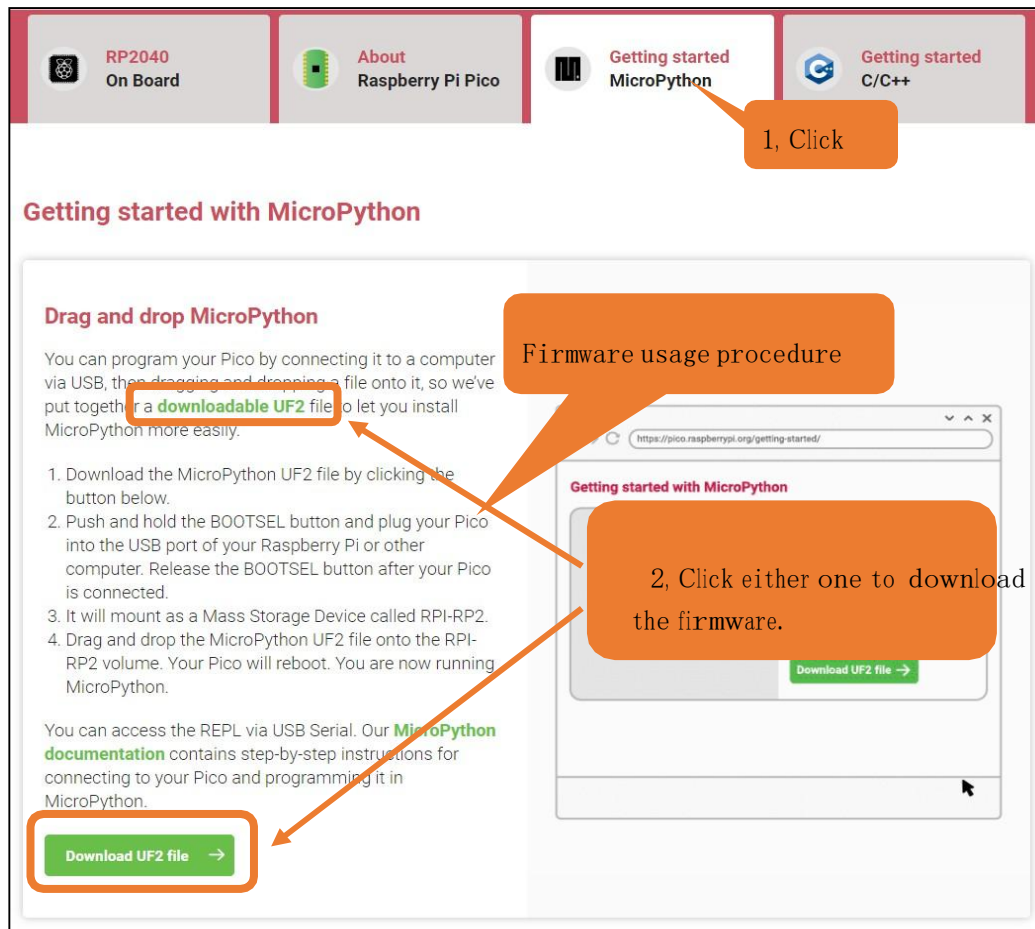
4.1.1 Click the link above and you can see the following interface



4.1.2 Roll the mouse and you can see the links for RP2040 documents.



4.1.3 Click “Getting started MicroPython” to enter the firmware downloading page.



**Option 2:** Directly download via `rp2-pico-20210618-v1.16.uf2`

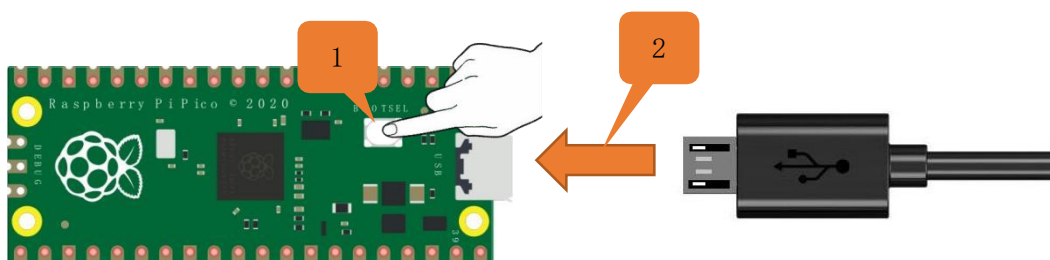
If you cannot download it due to network issue or other reasons, you can use the one we have prepared, which locates at the following file path:

**CKK0016-Main\ Tutorial\Python\Python\_Firmware**

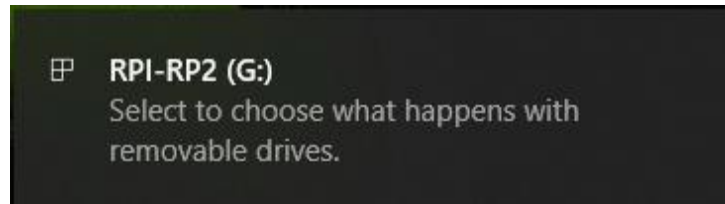
## 4.2 Burning a Micropython Firmware

4.2.1 Connect a USB cable to your computer.

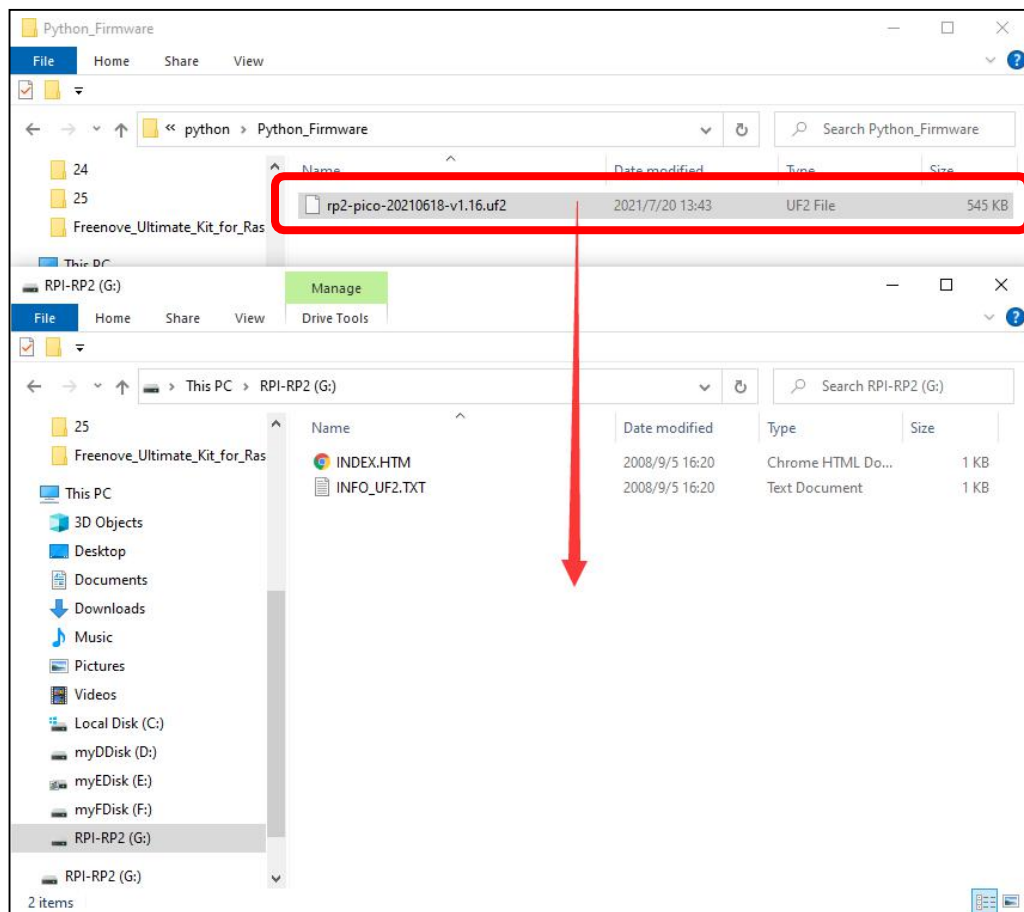
4.2.2 Long press BOOTSEL button on Raspberry Pi Pico and connect it to your computer with the USB cable.



4.2.3 When the connection succeeds, the following information will pop up on your computer.



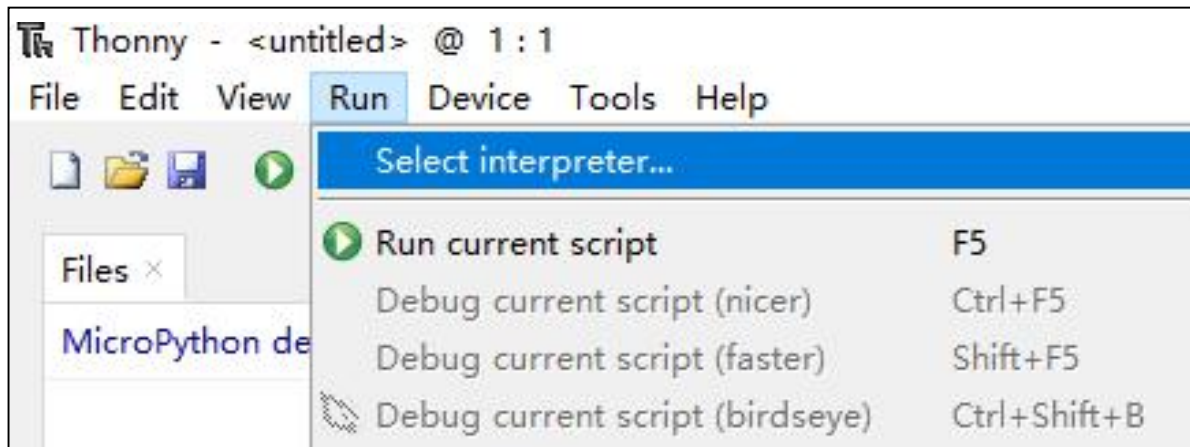
4.2.4 Copy the file(rp2-pico-20210618-v1.16.uf2) to RPI-RP2 and wait for it to finish, just like copy file to a U disk.



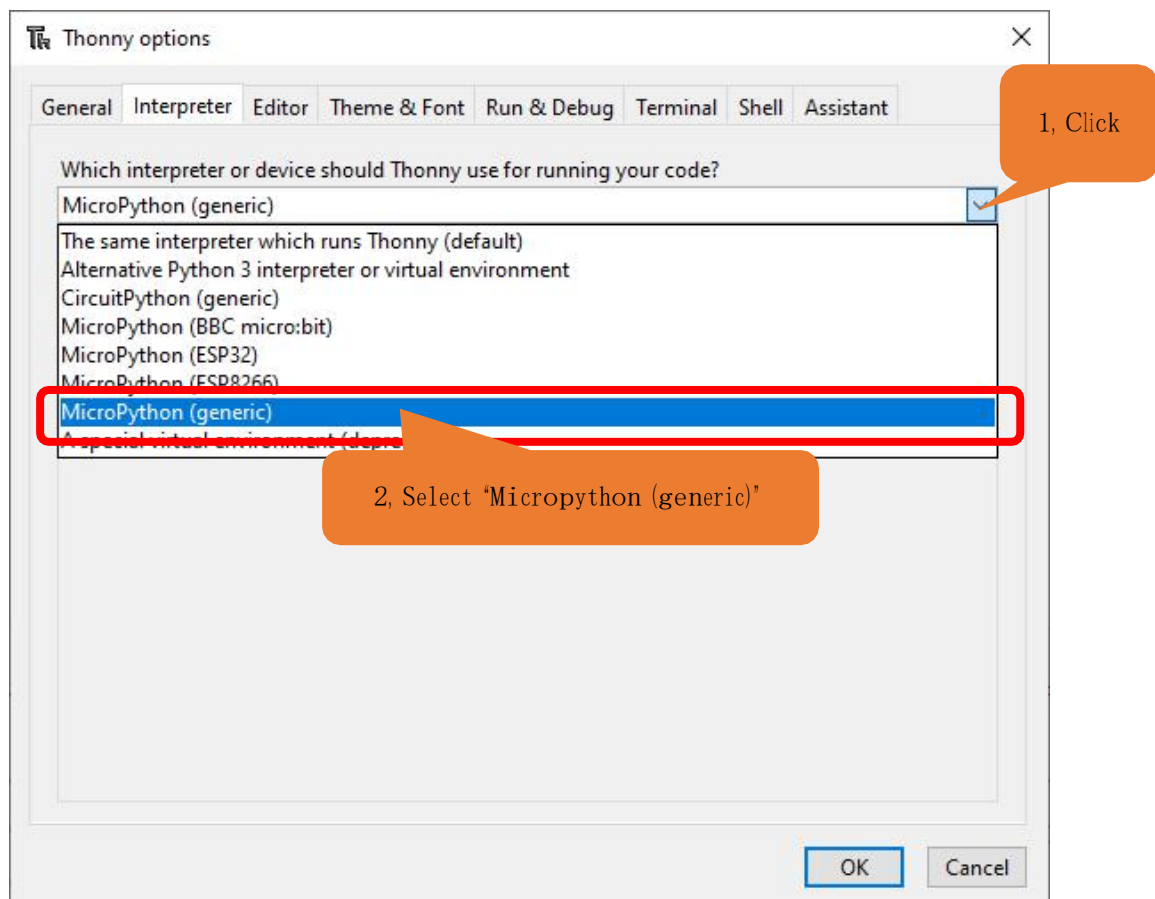
4.2.5 When the firmware finishes programming, Raspberry Pi Pico will reboot automatically. After that, you can run Micropython.

## 5. Connect Pico with Thonny

5.1 Open Thonny, click “run” and select “Select interpreter...”



5.2 Select “Micropython (generic)” or “Micropython (Raspberry Pi Pico)”. How to select “Micropython (generic)”? As shown below:

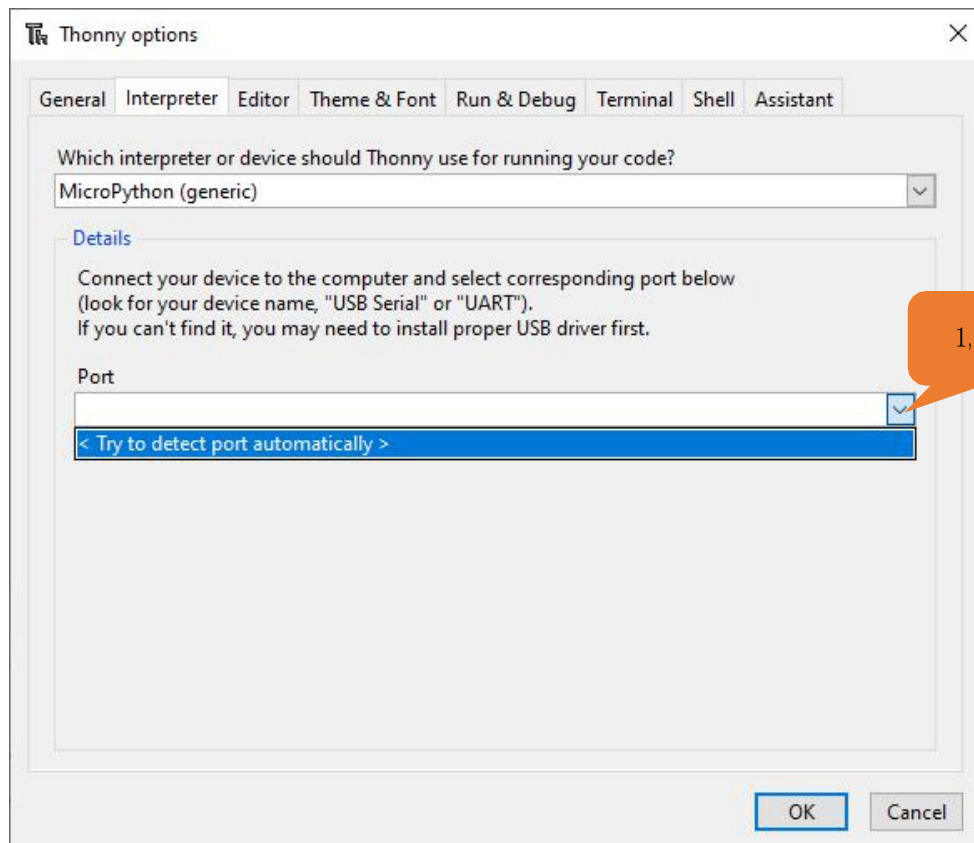


Select “**USB-SERIAL (COMx)**”, The number x of COMx may varies among different computers. You only need to make sure selecting USB-SERIAL (COMx).

### How to determine the port on which your Raspberry Pi Pico communicates with your computer?

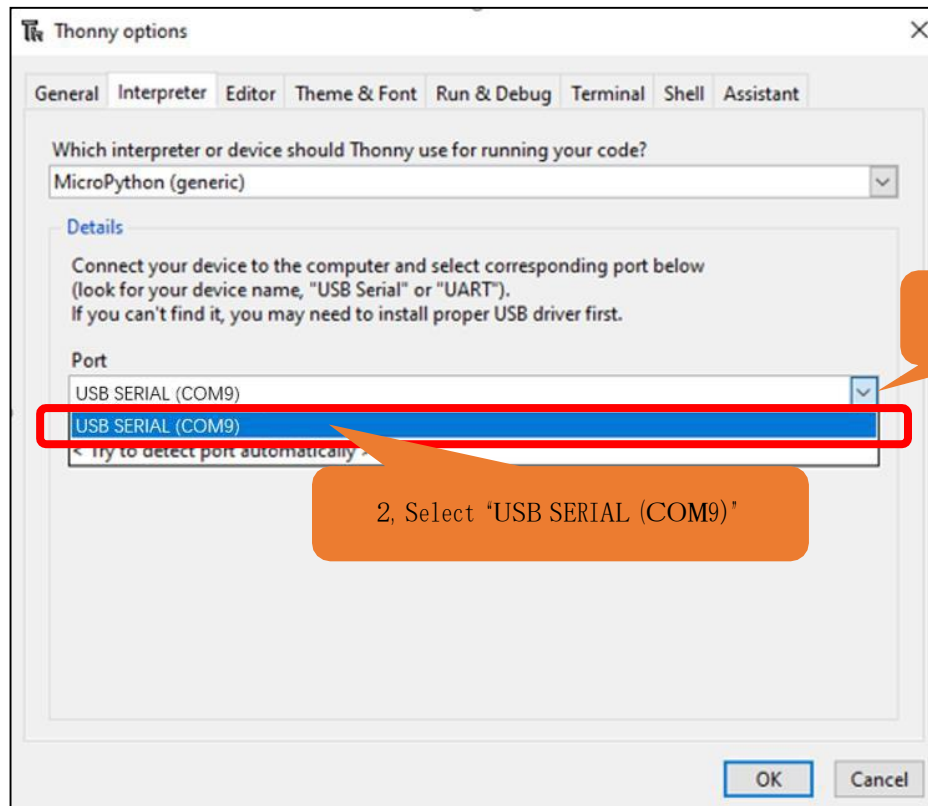
Step 1: When Pico doesn't connect to computer, open Thonny, click “Run”, select “Select interpreter”

and then a dialog box will pop up, click “Port” and you can check the ports currently connected to your computer, as shown below:

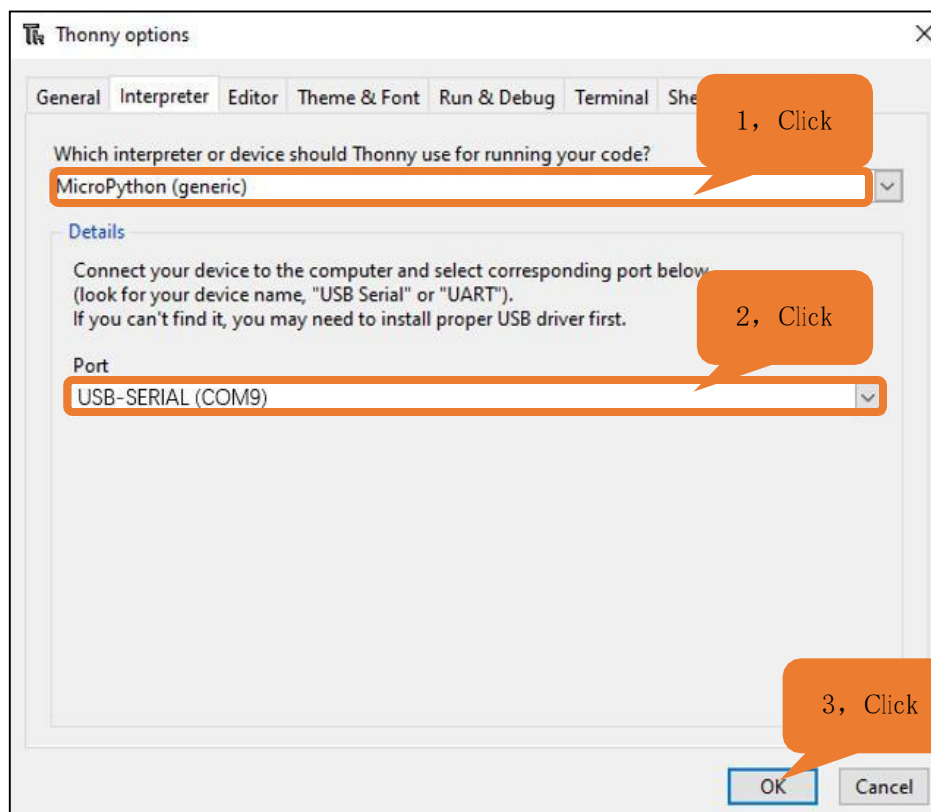




Step 2: Close the dialog box. Connect Pico to your computer, click "Run" again and select "Select interpreter". Click "Port" on the pop-up window and check the current ports. Now there is a newly added port, with which Pico communicates with the computer.

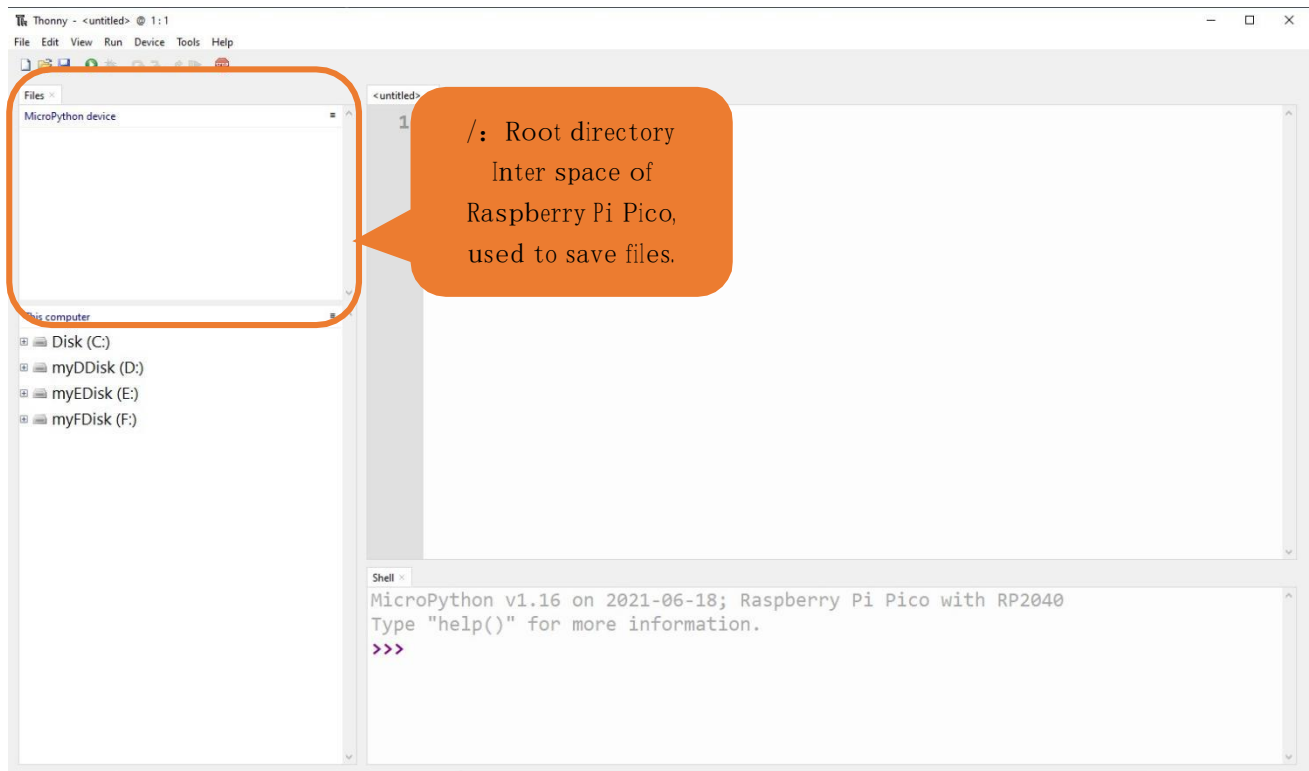


5.3 After selecting "Micropython (generic)" and port, click "OK"





5.4 When the following message displays on Thonny, it indicates Thonny has successfully connected to Pico.

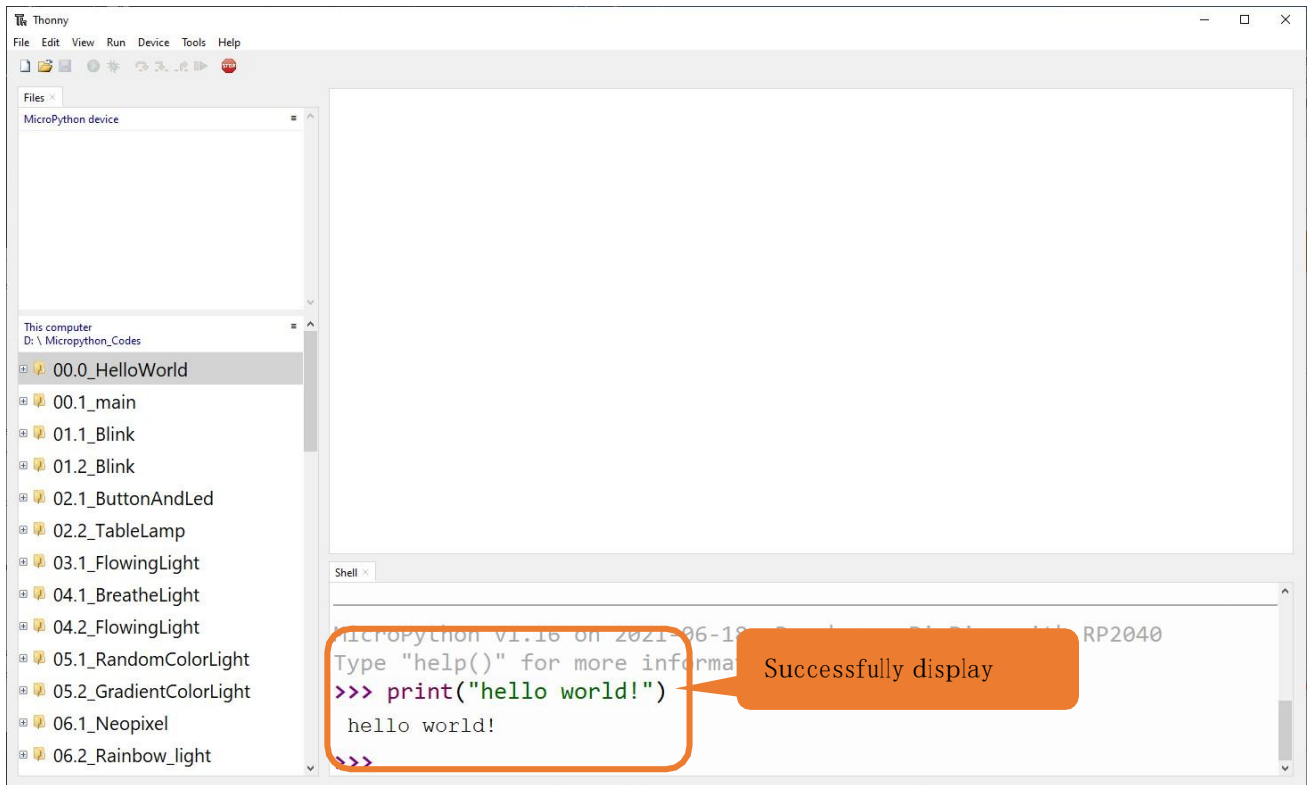


So far, all the preparations have been made.

## 6. Testing codes

### 6.1 Testing Shell Command

Enter “print(“hello world!”)” in “Shell” and press Enter.



## 6.2 Running Online

To run Raspberry Pi Pico online, you need to connect it to computer. Users can use Thonny to compile or debug programs.

**Advantages:**

Users can use Thonny to compile or debug programs.

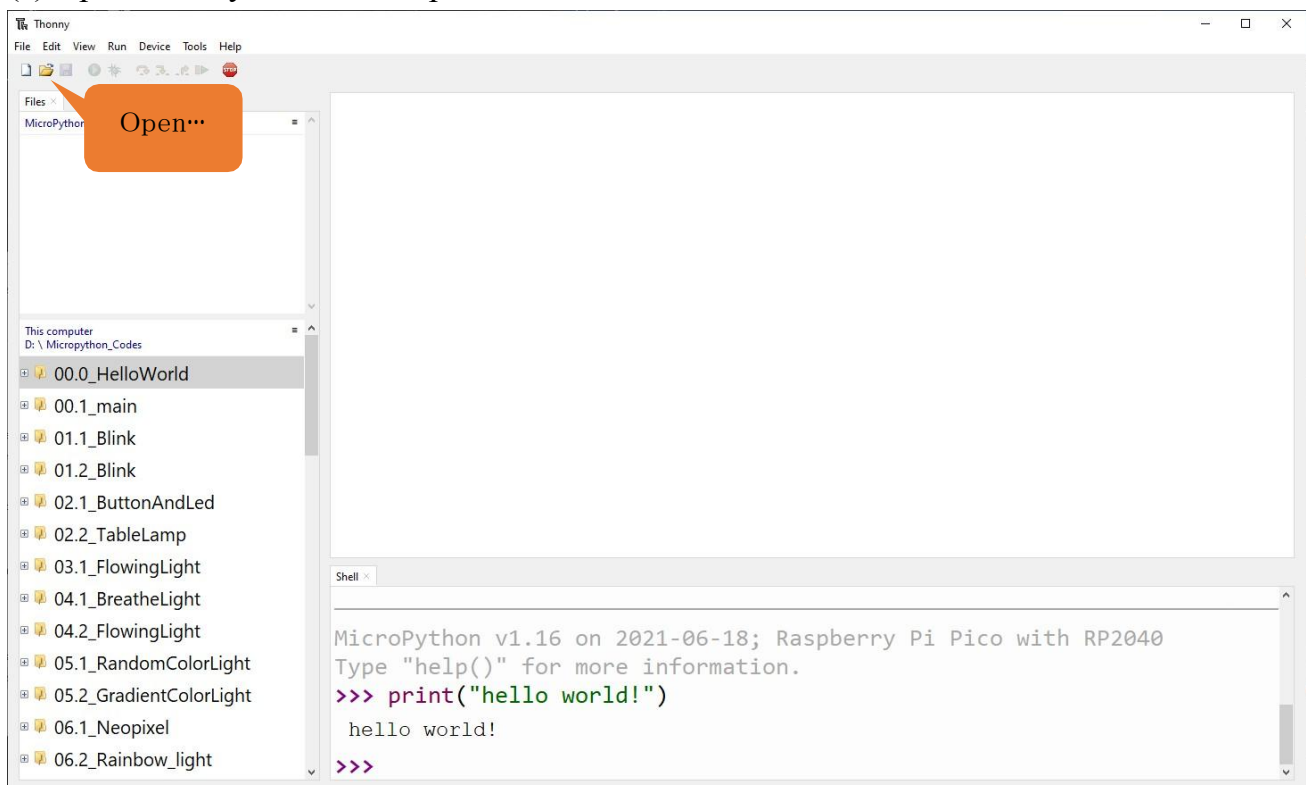
Through the "Shell" window, users can read the error information and output results generated during the running of the program and query related function information online to help improve the program.

**Disvantages:**

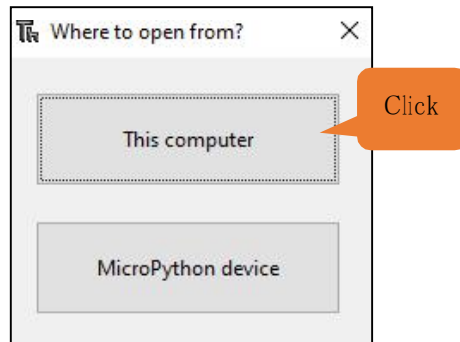
To run Raspberry Pi Pico online, you have to be connected to a computer and run with Thonny. If Raspberry Pi Pico disconnects from computer, the program won't run again when they reconnect to each other.

**Opertation:**

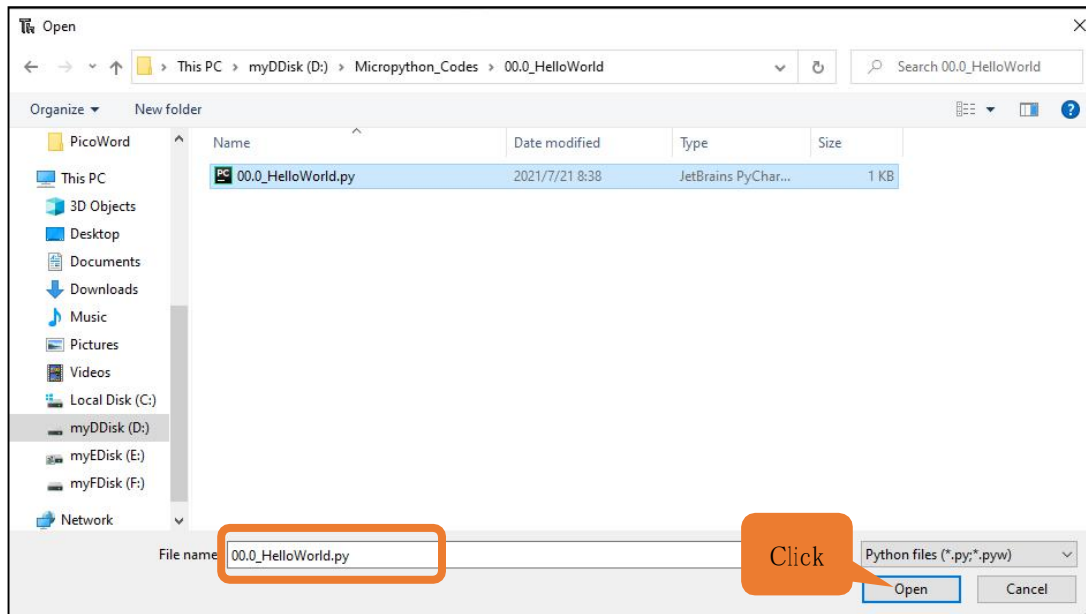
(a),Open Thonny and click "Open...".



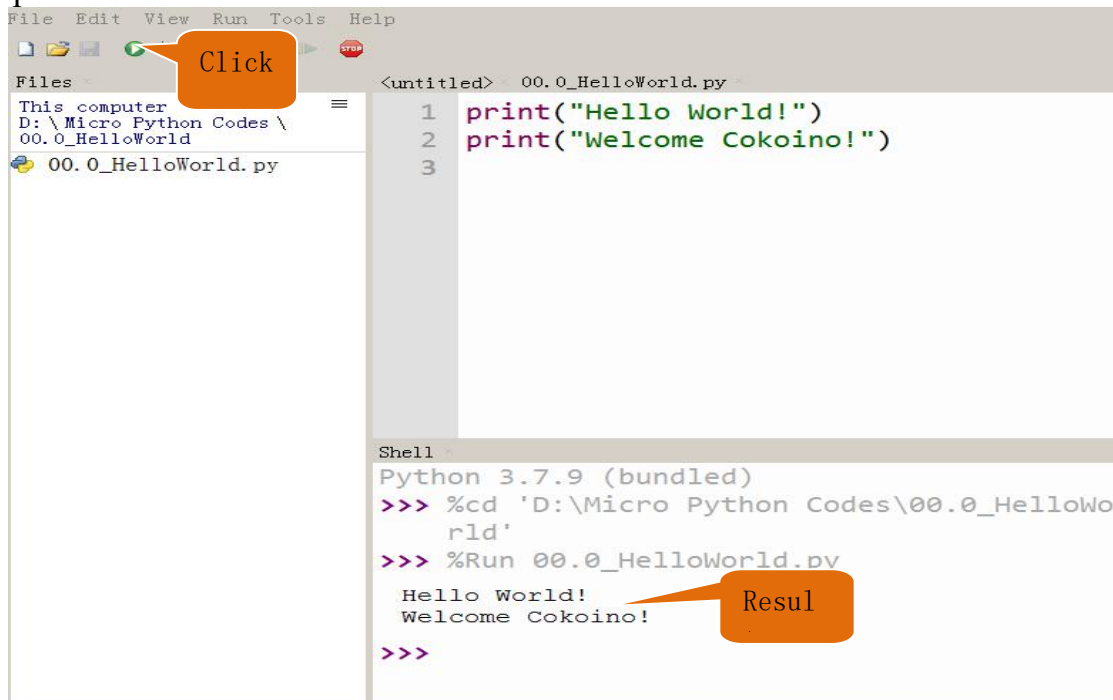
(b),On the newly pop-up window, click "This computer".



(c), In the new dialog box, select “00.0\_HelloWorld.py” in “CKK0016-Main\Tutorial\Python\Python\_Codes\00.0\_HelloWorld” folder.



Click “Run current script” to execute the program and “Hello World!”, “Welcome Cokoino” will be printed in “Shell”.



Exiting Running Online



When running online, click “Stop /Restart backend” on Thonny or press Ctrl+C to exit the program.

## 6.3 Running Offline

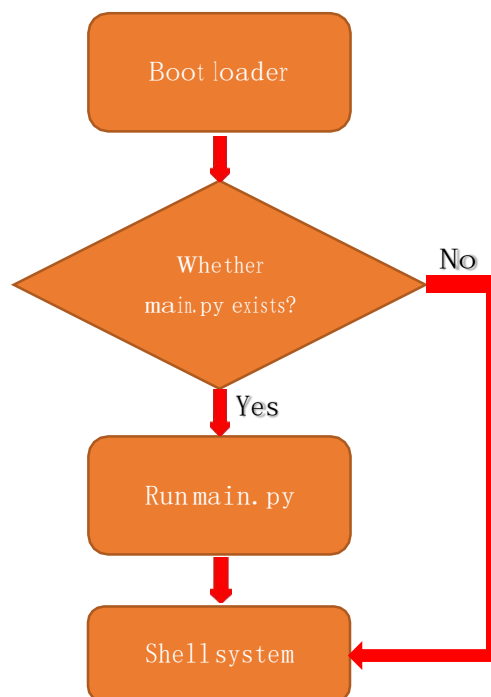
When running offline, Raspberry Pi Pico doesn't need to connect to computer and Thonny. It can run the programs stored in main.py on the device once powered up.

Advantage: It can run programs when powered up without connected to computer and Thonny.

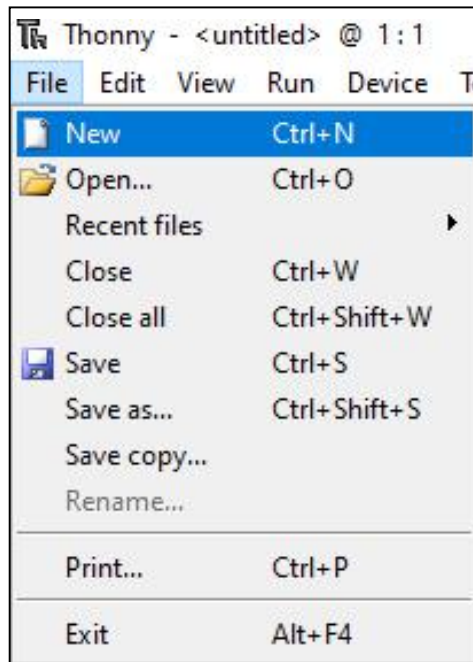
Disadvantage: The program will stop automatically when error occurs or Raspberry Pi Pico is out of power. Code cannot be changed easily.

### Operation

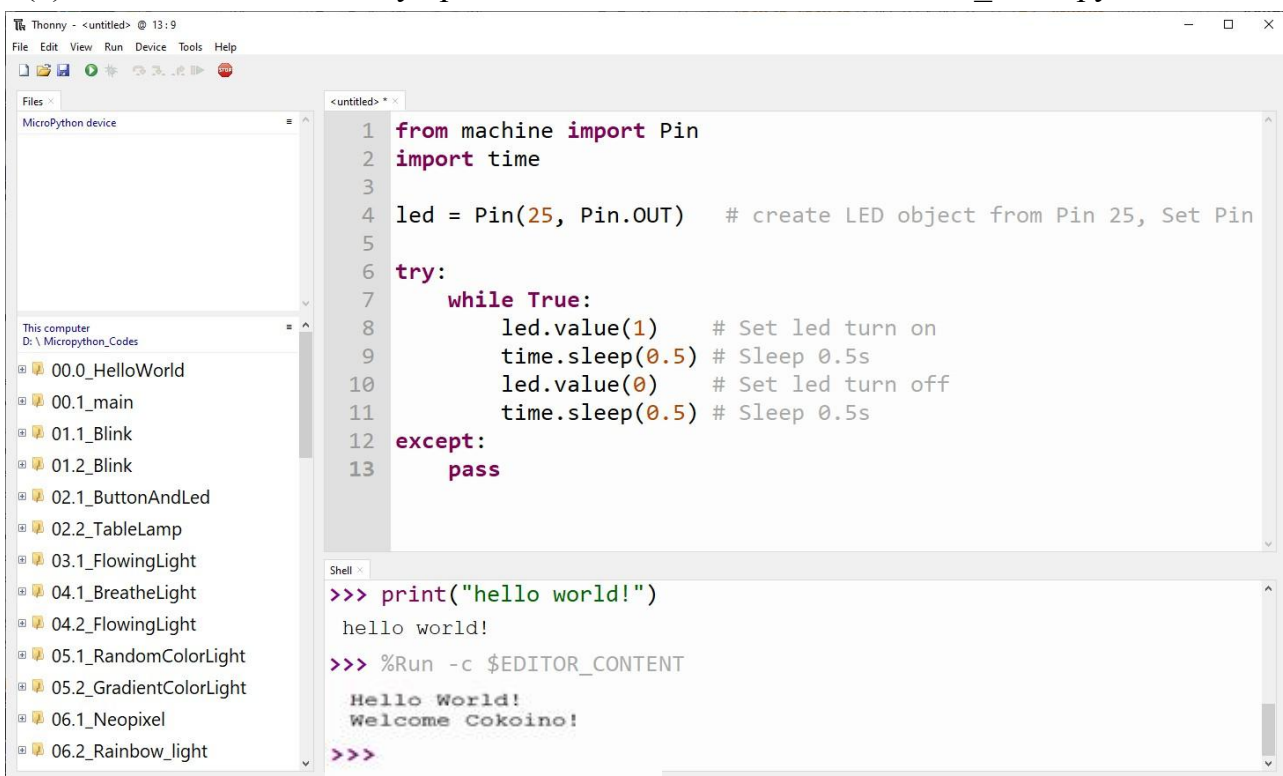
Once powered up, Raspberry Pi Pico will automatically check whether there is main.py existing on the device. If there is, it runs the programs in main.py and then enter shell command system. (If you want the code to run offline, you can save it as main.py); If main.py doesn't exist, it will enter shell command system directly.



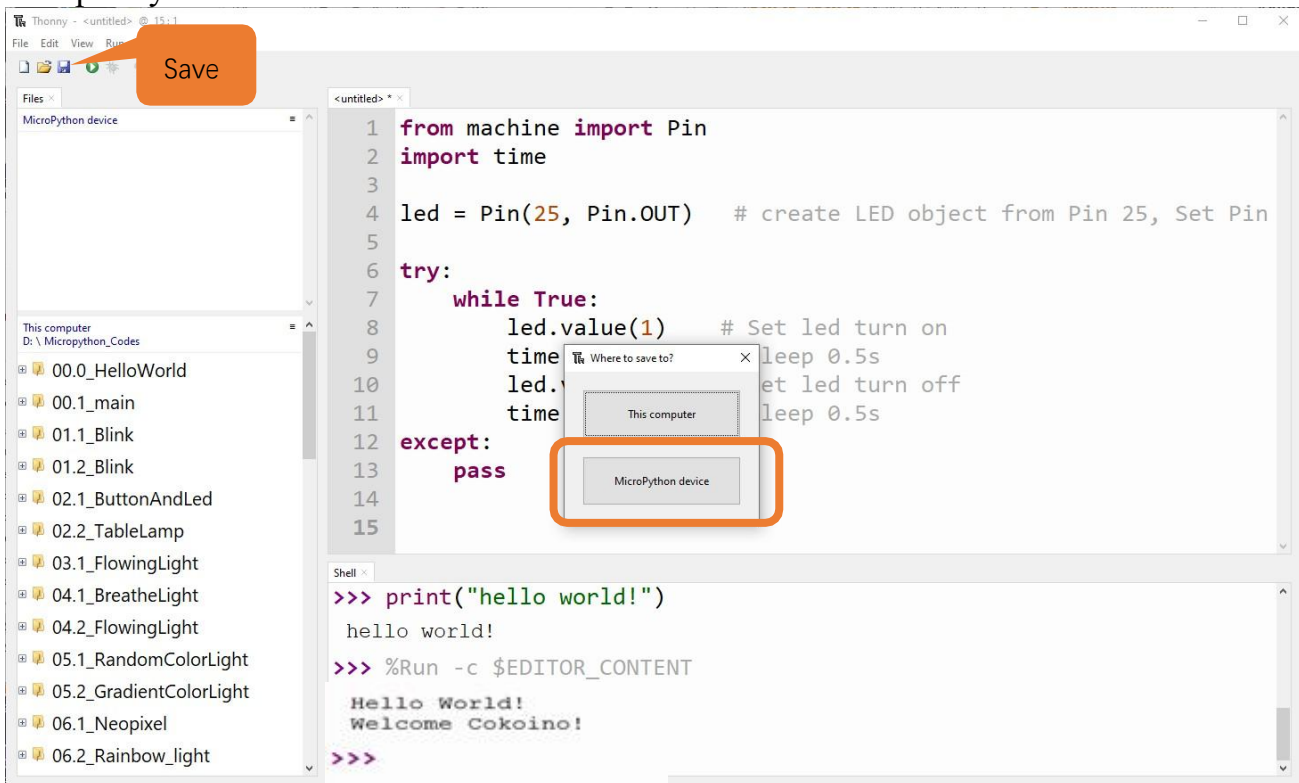
(a), Click “File” “New” to create and write codes.



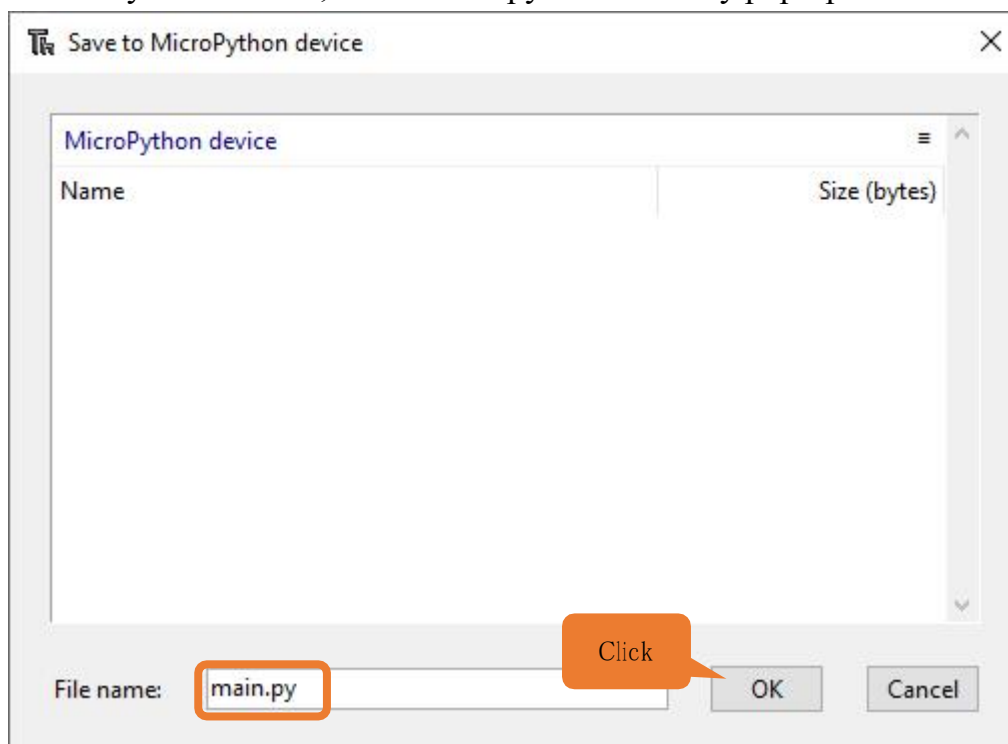
(b), Enter codes in the newly opened file. Here we use codes of “01.1\_Blink.py” as an example.



(c), Click “Save” on the menu bar. You can save the codes either to your computer or to Raspberry Pi Pico.

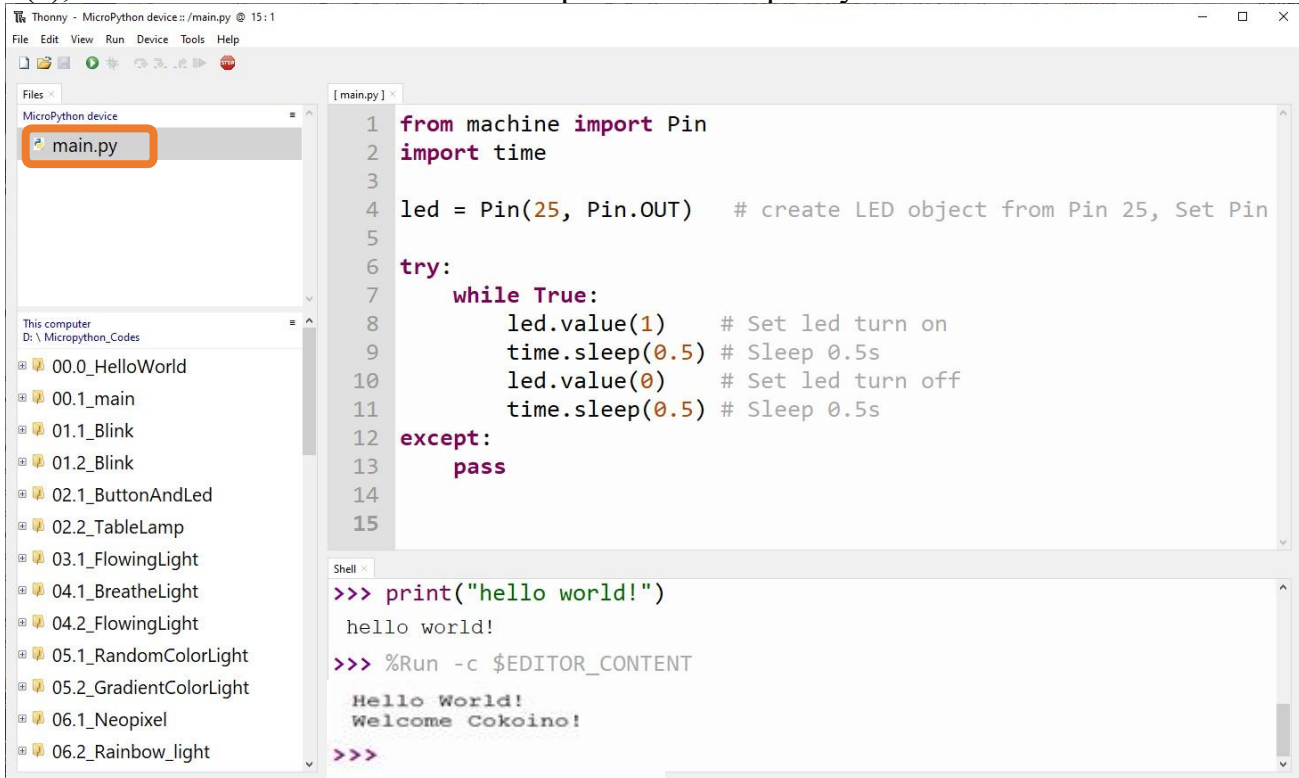


(d), Select “MicroPython device”, enter “main.py” in the newly pop-up window and click “OK”.





(e), You can see that codes have been uploaded to Raspberry Pi Pico.



```

1 from machine import Pin
2 import time
3
4 led = Pin(25, Pin.OUT) # create LED object from Pin 25, Set Pin
5
6 try:
7     while True:
8         led.value(1) # Set led turn on
9         time.sleep(0.5) # Sleep 0.5s
10        led.value(0) # Set led turn off
11        time.sleep(0.5) # Sleep 0.5s
12 except:
13     pass
14
15

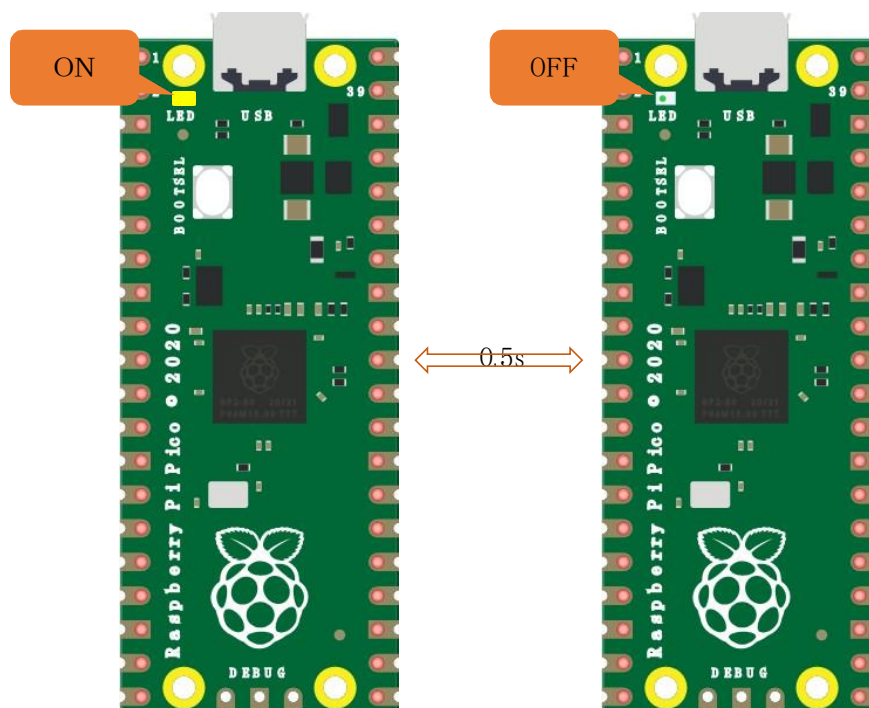
```

```

>>> print("hello world!")
hello world!
>>> %Run -c $EDITOR_CONTENT
Hello World!
Welcome Cokoino!
>>>

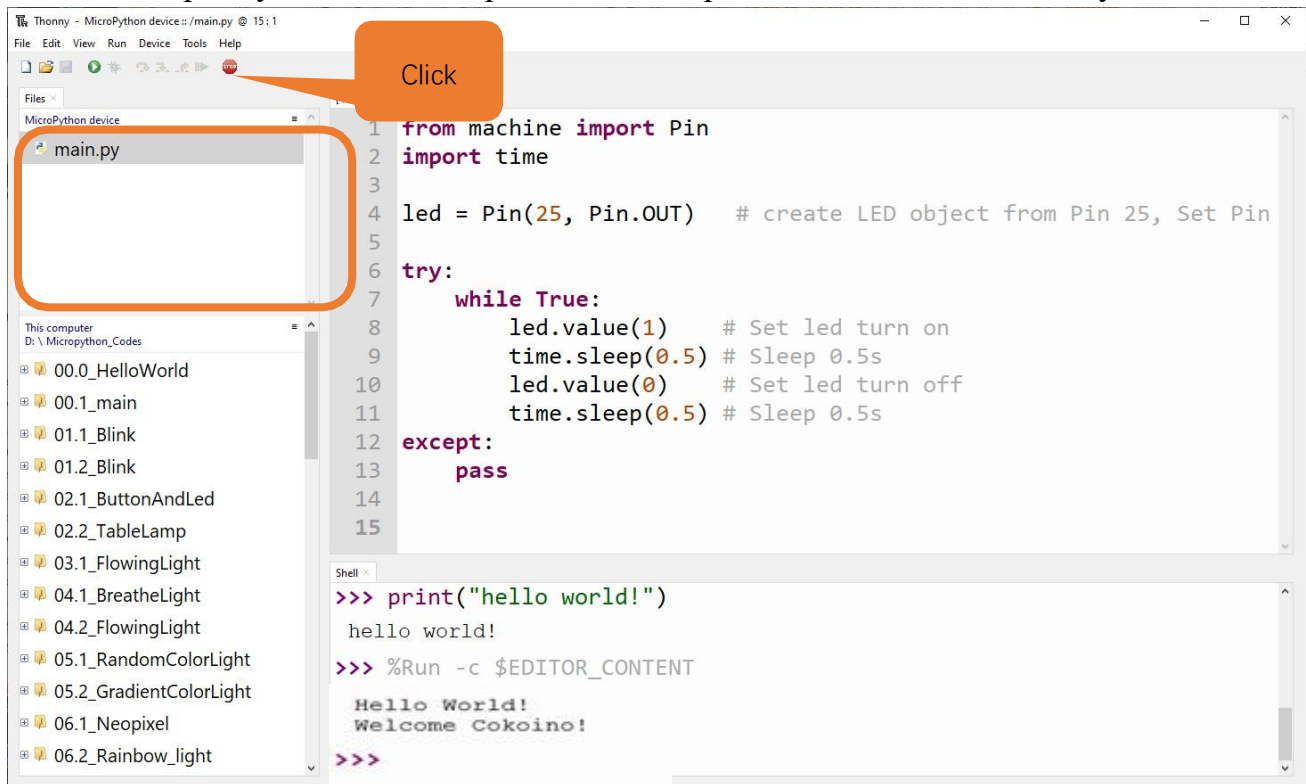
```

(f), Disconnect Raspberry Pi Pico USB cable and then reconnect it, the LED on Raspberry Pi Pico will blink repeatedly.



## Exiting Offline Running

Connect Raspberry Pi Pico to computer, click “stop/restart backend” on Thonny to end running

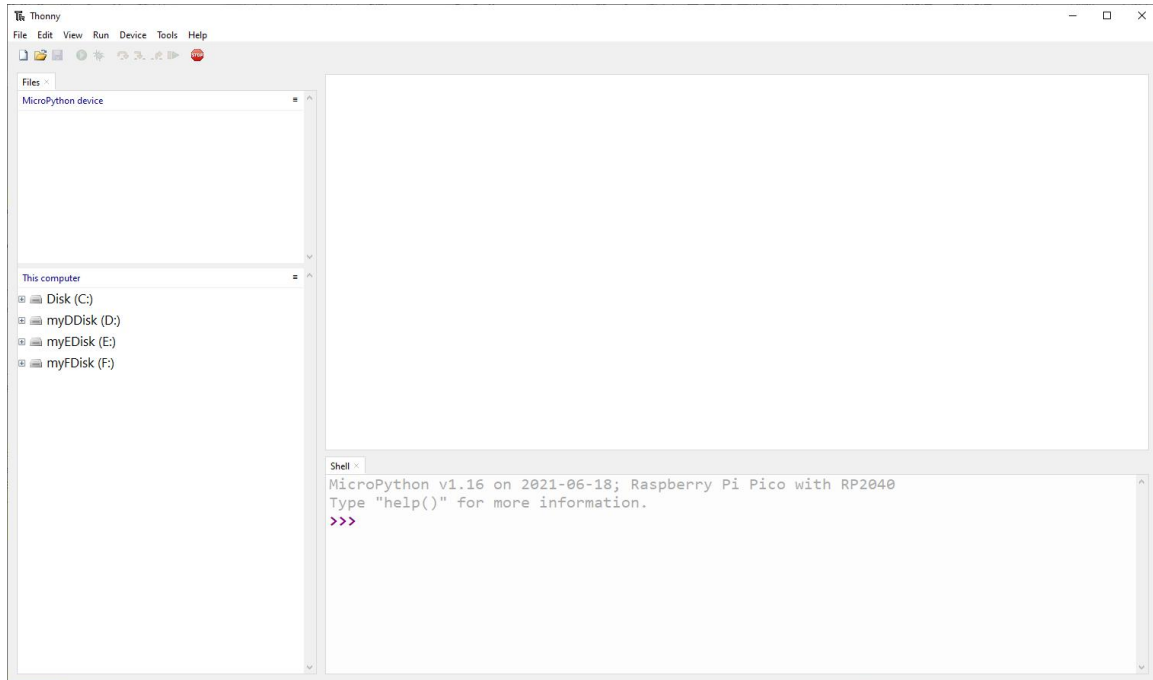


If it doesn't work, please click on “stop/restart backend” for more times or reconnect Pico.

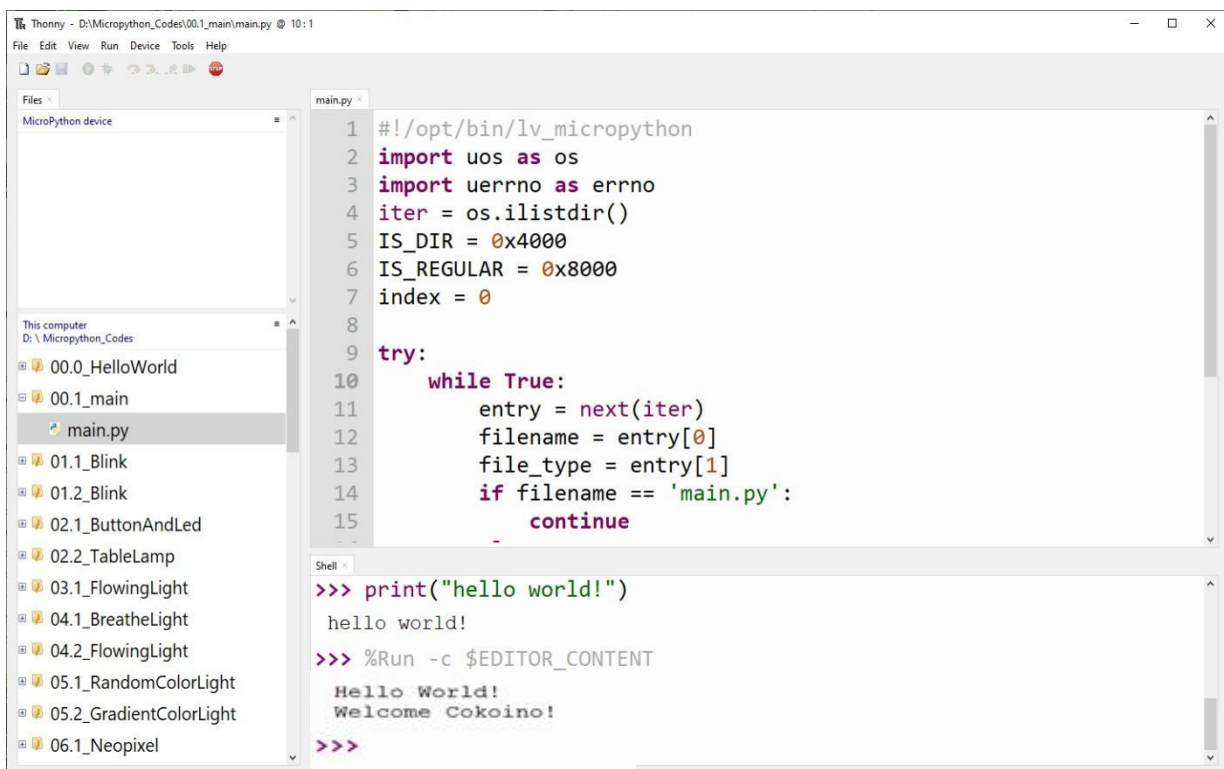


We provide a main.py file for running offline. The code added to main.py is a bootstrap that executes the user's code file. All you need to do is upload the offline project's code file (.py) to the Raspberry Pi Pico device.

Move the program folder “[CKK0016-Main\ Tutorial\Python\Python\\_Codes](#)” to disk(D) in advance with the path of “[D:/Micropython\\_Codes](#)”. Open “[Thonny](#)”.

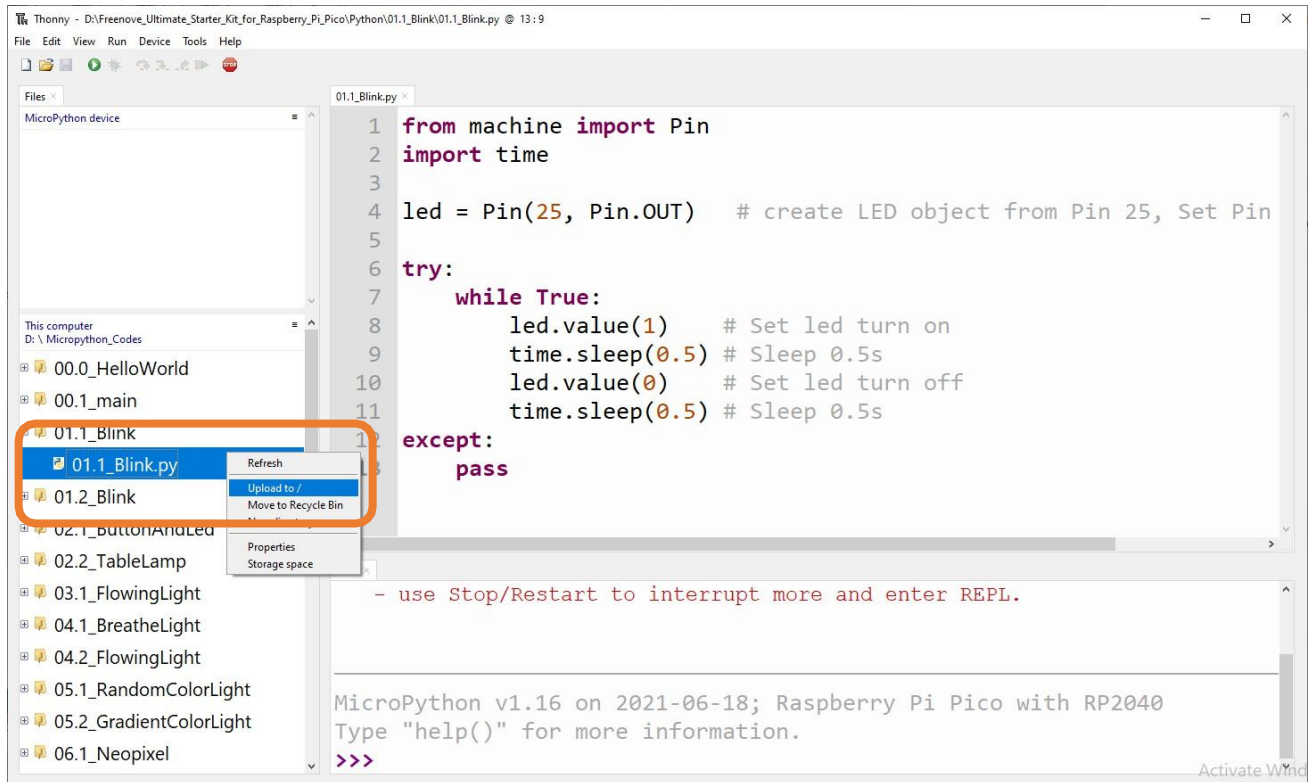


Expand “00.1\_main” in the “[Micropython\\_Codes](#)” in the directory of disk(D), and double-click main.py, which is provided by us to enable programs in “MicroPython device” to run offline.

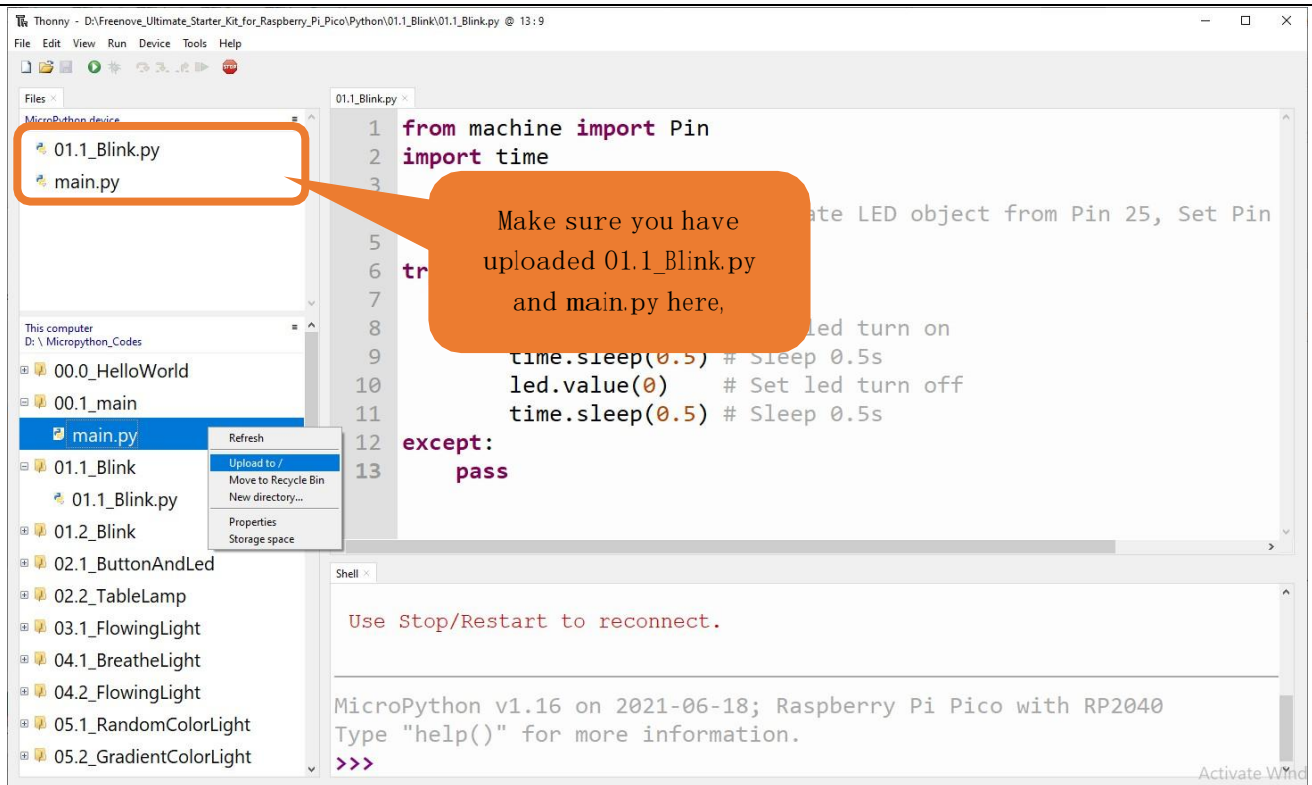


Here we use 00.1 and 01.1 cases as demonstration. The LED on Raspberry Pi Pico is used to show the result, which uses GP25 pin. If you have modified 01.1\_Blink.py file, you need to change it accordingly.

As shown in the following illustration, right-click the file 01.1\_Blink.py and select “Upload to /” to upload code to Raspberry Pi Pico.



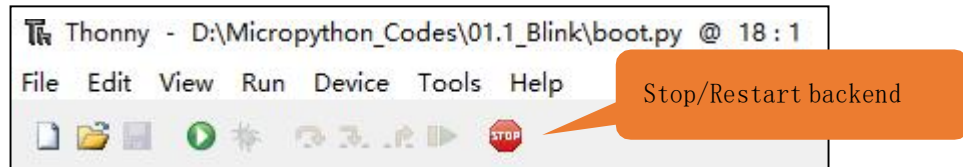
Upload main.py in the same way.



Disconnect Raspberry Pi Pico USB cable and reconnect it, the LED on pico will blink repeatedly.

Note:

Codes here are run offline. If you want to stop running offline and enter Shell, just click “Stop” in Thonny.

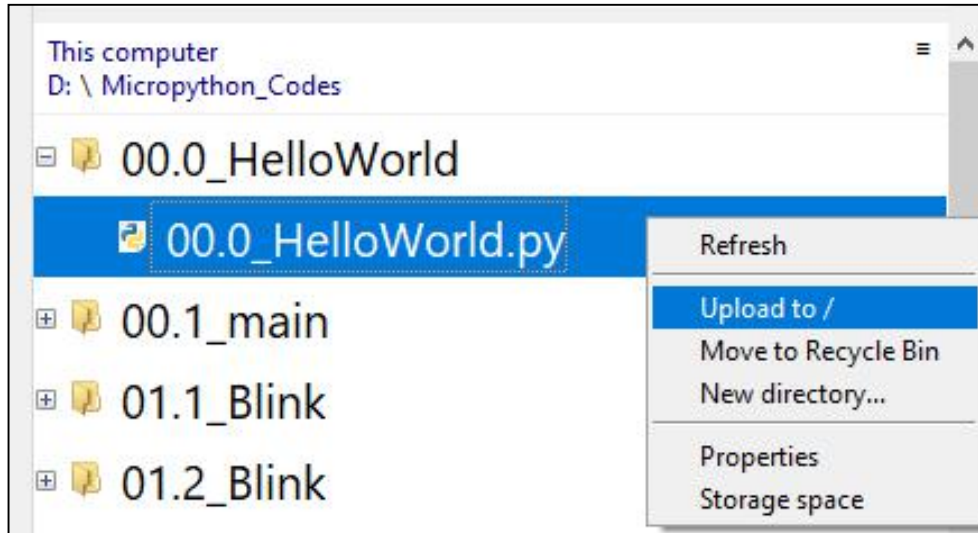




## 7. Thonny Common Operation

### 7.1 Uploading Code to Raspberry Pi Pico

Select “00.0\_HelloWorld.py” in “00.0\_HelloWorld”, right-click your mouse and select “Upload to /” to upload code to Raspberry Pi Pico’s root directory.



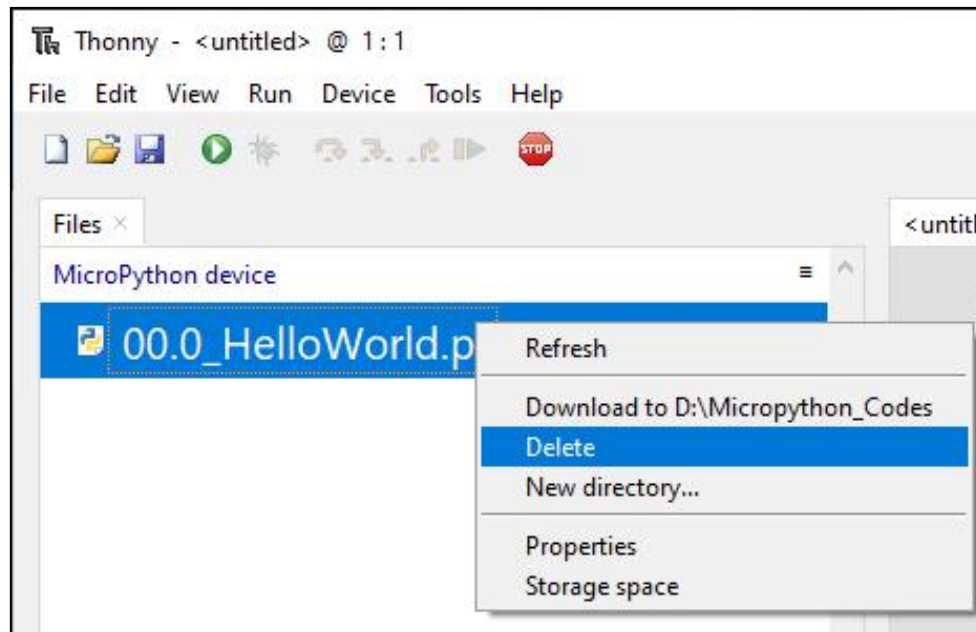
### 7.2 Downloading Code to Computer

Select “00.0\_HelloWorld.py” in “MicroPython device”, right-click to select “Download to ...” to download the code to your computer.



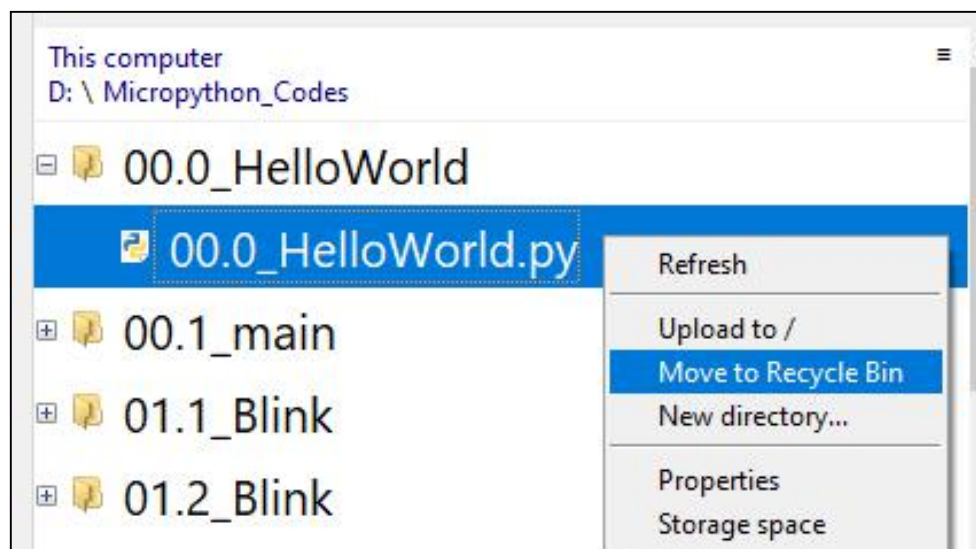
### 7.3 Deleting Files from Raspberry Pi Pico's Root Directory

Select “00.0\_HelloWorld.py” in “MicroPython device”, right-click it and select “Delete” to delete “00.0\_HelloWorld.py” from Raspberry Pi Pico’s root directory.



### 7.4 Deleting Files from your Computer Directory

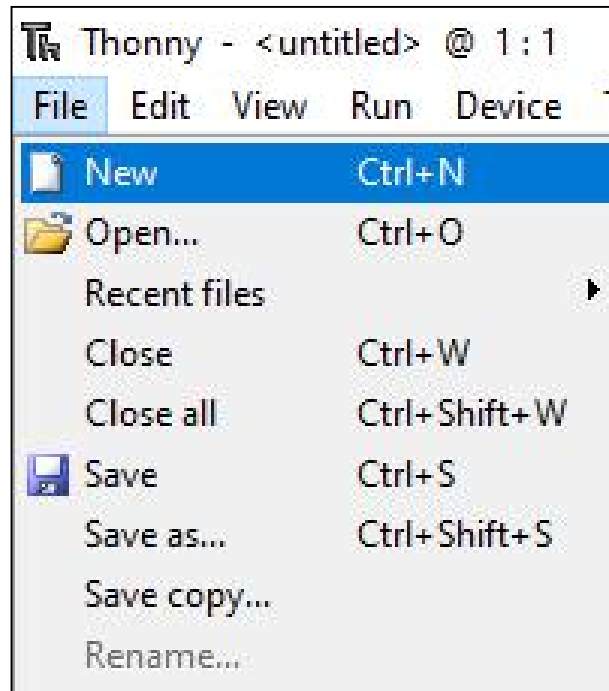
Select “00.0\_HelloWorld.py” in “00.0\_HelloWorld”, right-click it and select “Move to Recycle Bin” to delete it from “00.0\_HelloWorld”.



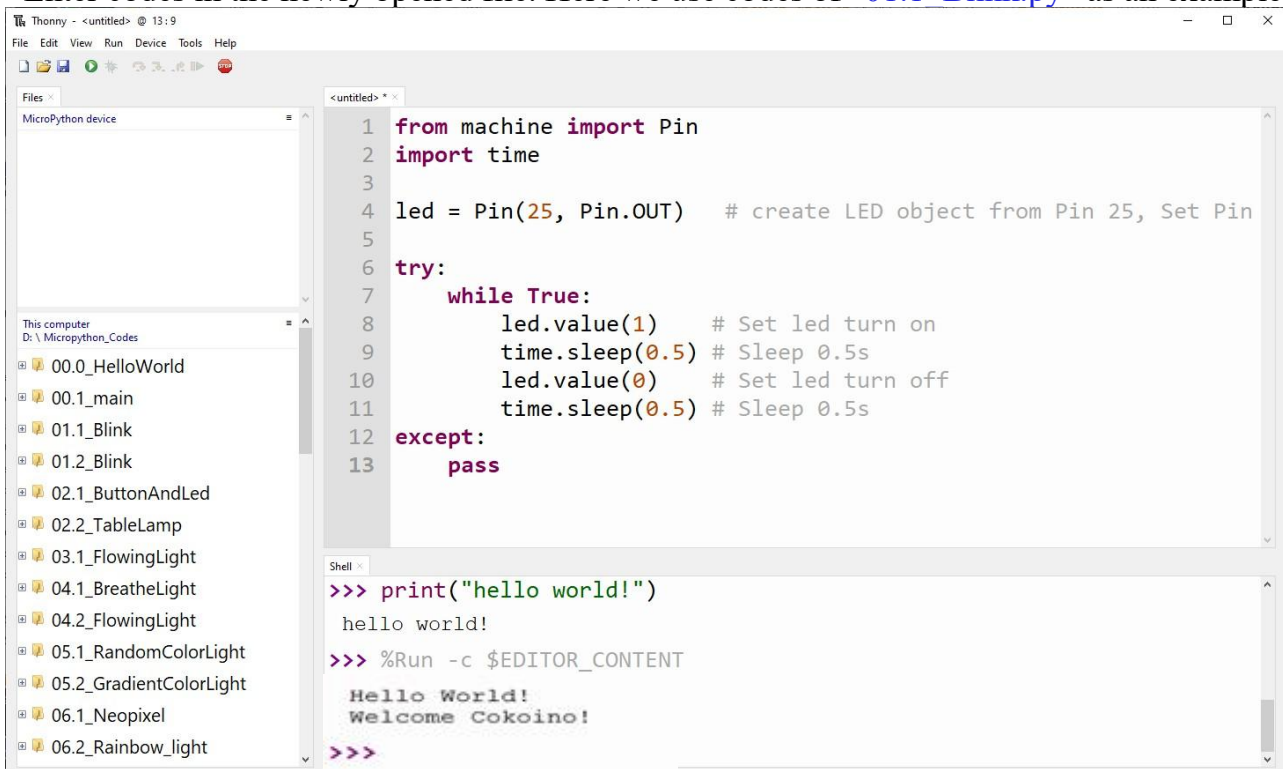


## 7.5 Creating and Saving the code

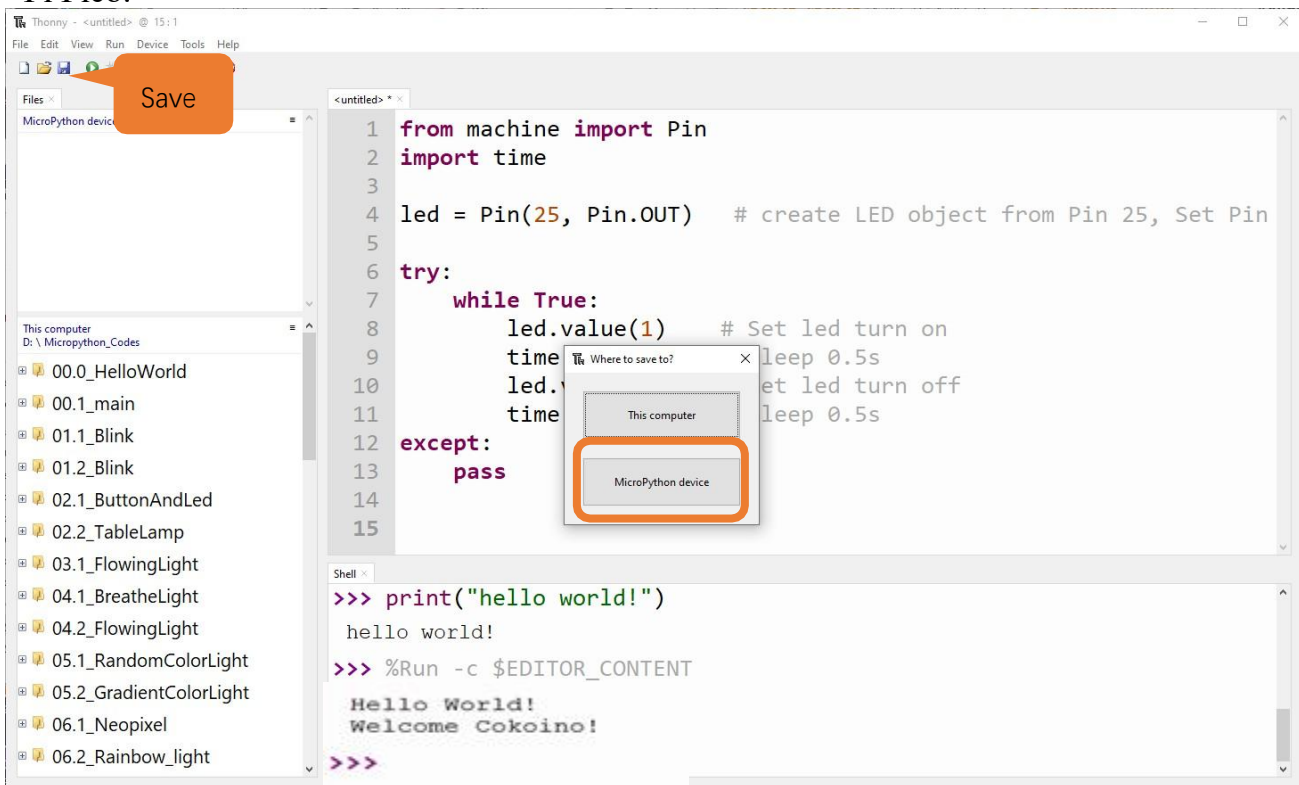
Click “File”--“New” to create and write codes.



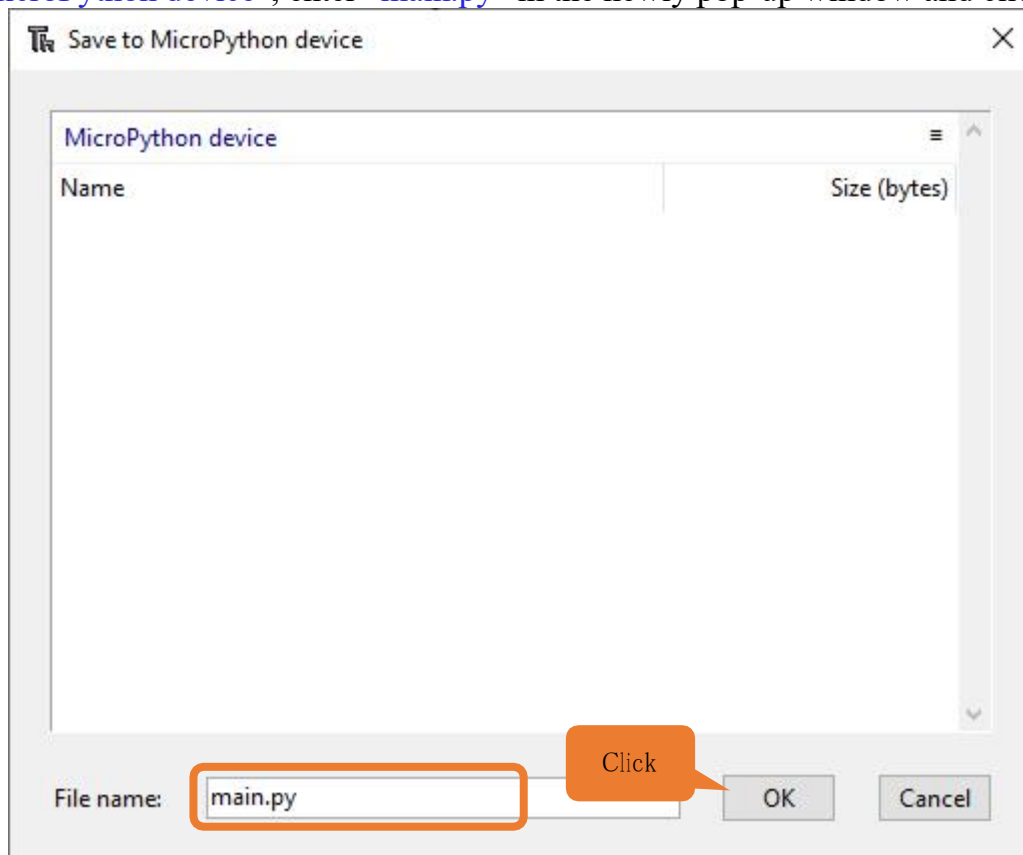
Enter codes in the newly opened file. Here we use codes of “01.1 Blink.py” as an example.



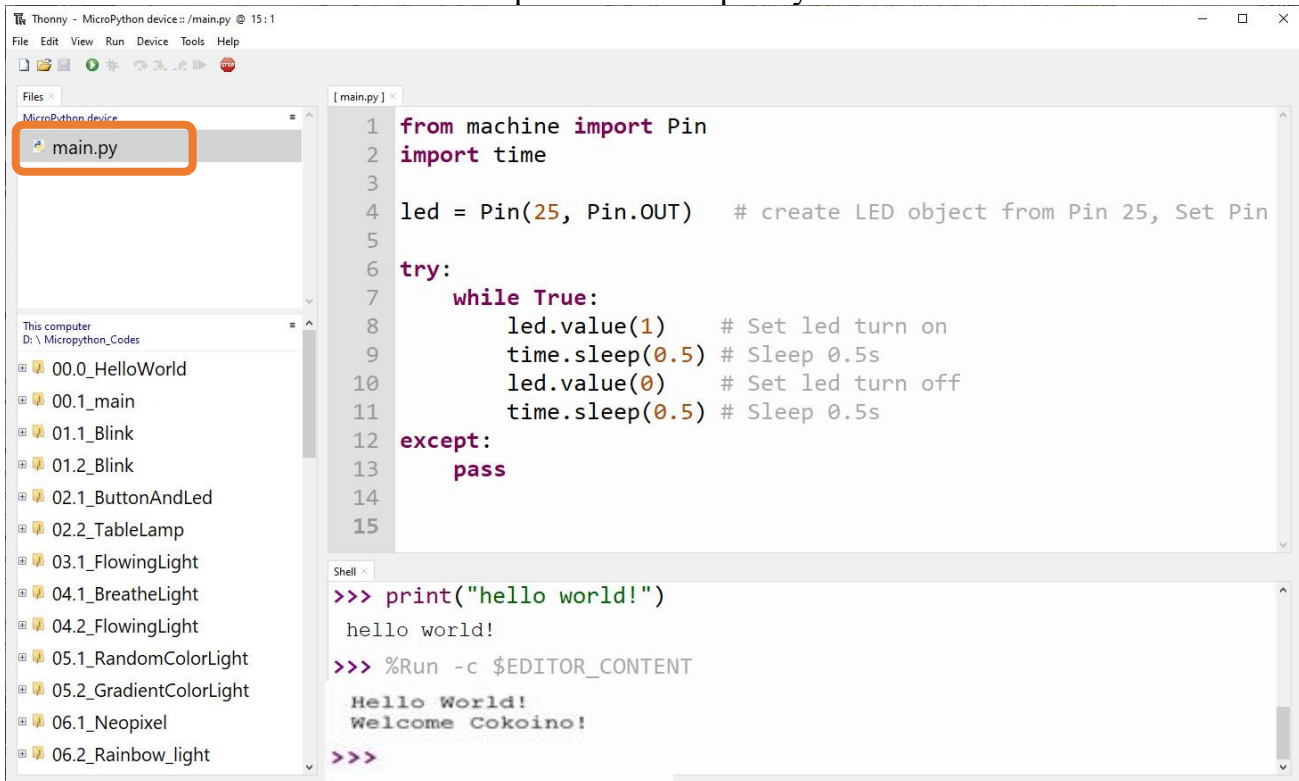
Click “Save” on the menu bar. You can save the codes either to your computer or to Raspberry Pi Pico.



Select “MicroPython device”, enter “main.py” in the newly pop-up window and click “OK”.



You can see that codes have been uploaded to Raspberry Pi Pico.

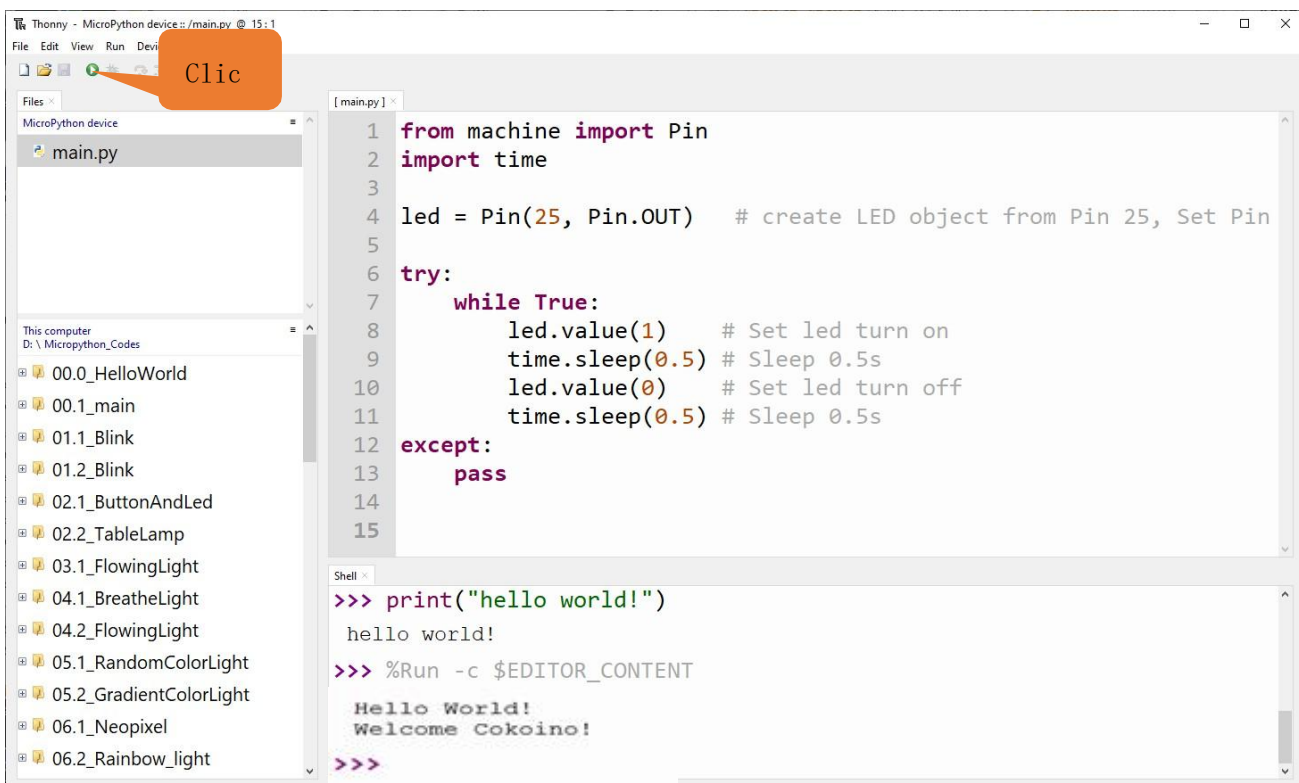


```

1 from machine import Pin
2 import time
3
4 led = Pin(25, Pin.OUT) # create LED object from Pin 25, Set Pin
5
6 try:
7     while True:
8         led.value(1) # Set led turn on
9         time.sleep(0.5) # Sleep 0.5s
10        led.value(0) # Set led turn off
11        time.sleep(0.5) # Sleep 0.5s
12 except:
13     pass
14
15
Shell
>>> print("hello world!")
hello world!
>>> %Run -c $EDITOR_CONTENT
Hello World!
Welcome Cokoino!
>>>

```

Click “Run” and the LED on Raspberry Pi Pico will blink periodically.



```

1 from machine import Pin
2 import time
3
4 led = Pin(25, Pin.OUT) # create LED object from Pin 25, Set Pin
5
6 try:
7     while True:
8         led.value(1) # Set led turn on
9         time.sleep(0.5) # Sleep 0.5s
10        led.value(0) # Set led turn off
11        time.sleep(0.5) # Sleep 0.5s
12 except:
13     pass
14
15
Shell
>>> print("hello world!")
hello world!
>>> %Run -c $EDITOR_CONTENT
Hello World!
Welcome Cokoino!
>>>

```

## 8. Make your suggestion and get support

THANK YOU for reading this document!

If you find errors, omissions or you have suggestions and/or questions about this document, please feel free to contact us: **[cokoino@outlook.com](mailto:cokoino@outlook.com)**

We will make every effort to make changes and correct errors as soon as feasibly possible and publish a revised version.

If you want to learn more about Arduino, Raspberry Pi, Smart Cars, Robotics and other interesting products in science and technology, please continue to visit our Amazon Store by search for "**LK COKOINO**" on Amazon. We will continue to launch fun, cost-effective, innovative and exciting products.

Thank you again for choosing Cokoino products.