

Getting Ready

Before starting building the projects, you need to make some preparation first, which is so crucial that you must not skip.

0.1 Installing Thonny (Important)

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.

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 | https://github.com/thonny/thonny/releases/download/v4.1.1/thonny-4.1.1.exe |
| Mac OS | https://github.com/thonny/thonny/releases/download/v4.1.1/thonny-4.1.1.pkg |
| Linux | <p>The latest version:</p> <p>Binary bundle for PC (Thonny+Python): <code>bash <(wget -O - https://thonny.org/installer-for-linux)</code></p> <p>With pip: <code>pip3 install thonny</code></p> <p>Distro packages (may not be the latest version):</p> <p>Debian, Rasbian, Ubuntu, Mint and others: <code>sudo apt install thonny</code></p> <p>Fedora: <code>sudo dnf install thonny</code></p> |

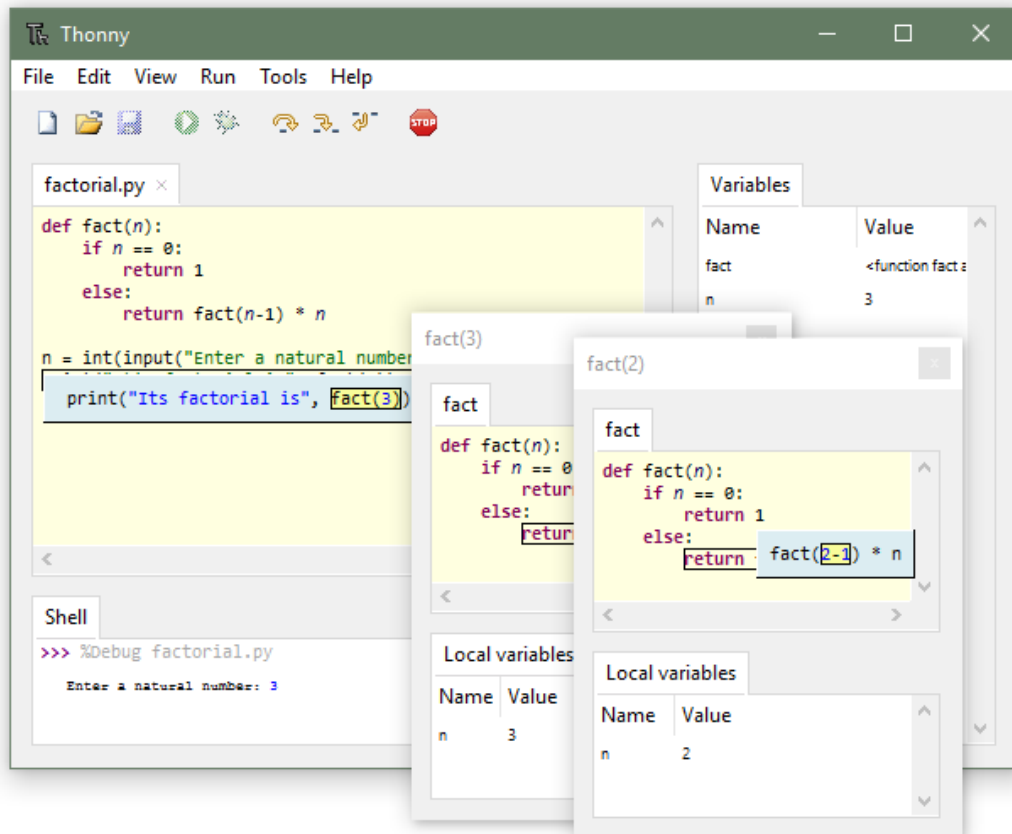
You can also open **'Raspberry_Pi_Pico Kit Tutorial\Lesson2-Python\Python_Software'**, we have prepared it in advance.

Thonny

Python IDE for beginners



Download version [4.1.1](#) for
[Windows](#) • [Mac](#) • [Linux](#)



Installing on Windows

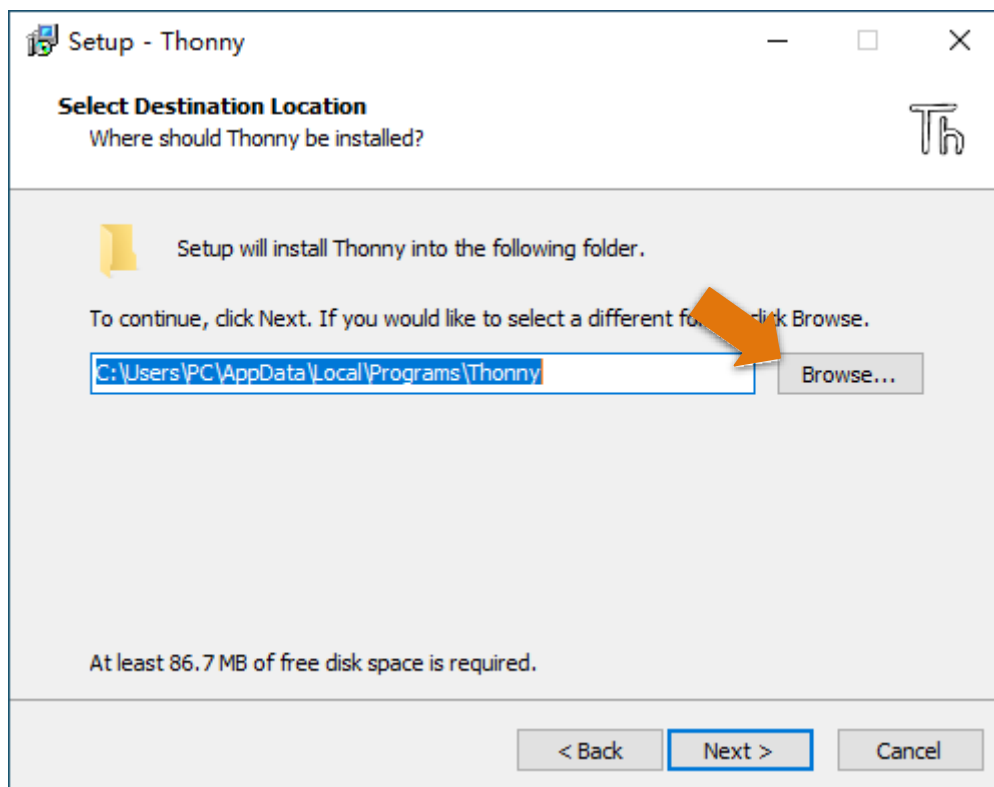
The icon of Thonny after downloading is as below. Double click "thonny-4.1.1.exe".



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

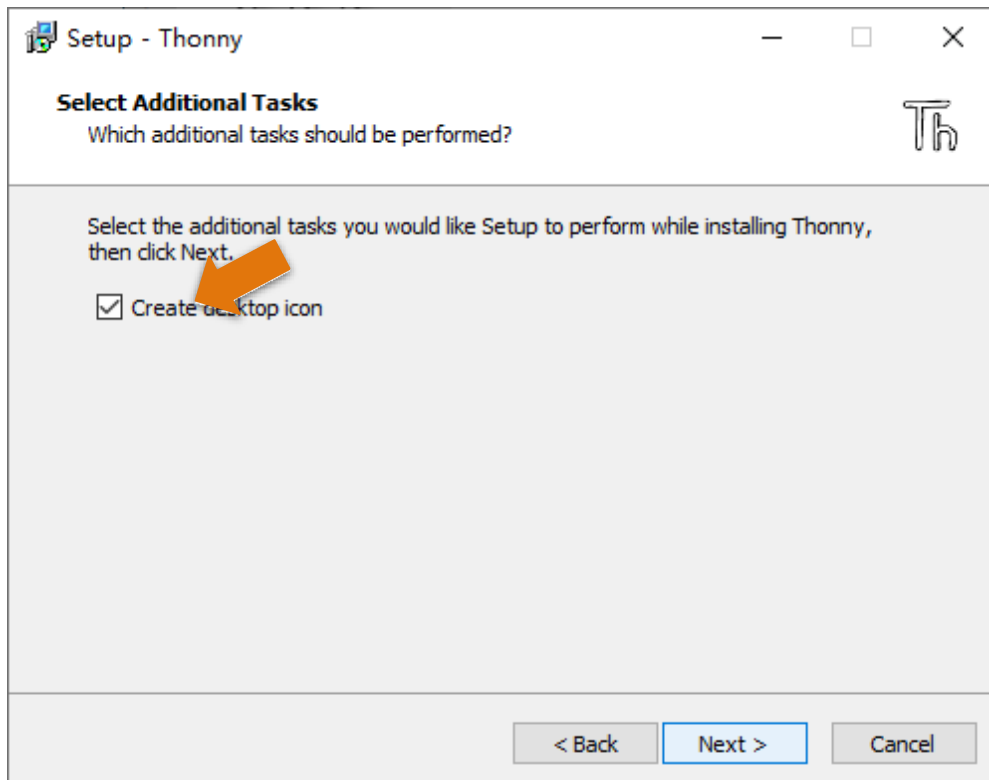


If you want to change Thonny's installation path, you can click "Browse" to modify it. After selecting installation path, click "OK".

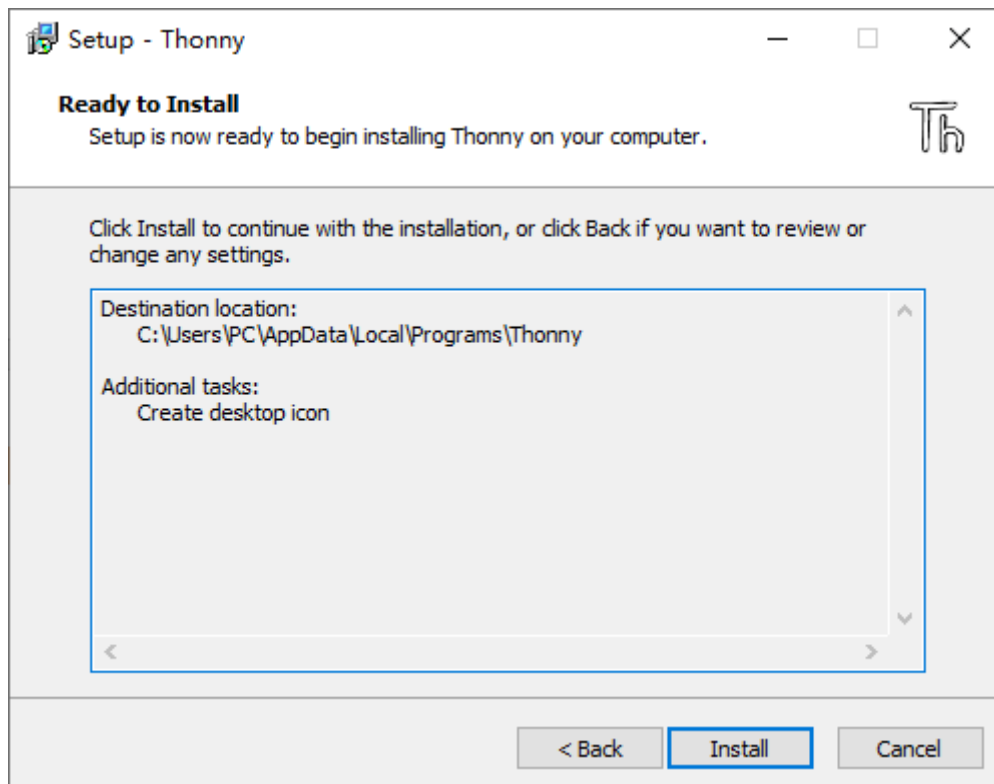


If you do not want to change it, just click "Next".

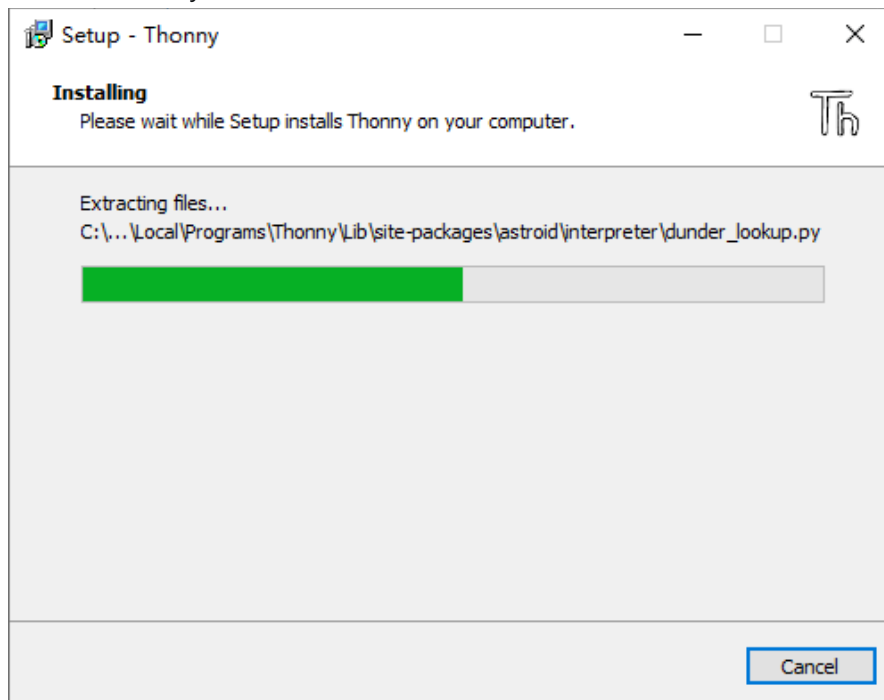
Check “[Create desktop icon](#)” and then it will generate a shortcut on your desktop to facilitate you to open Thonny later.



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.

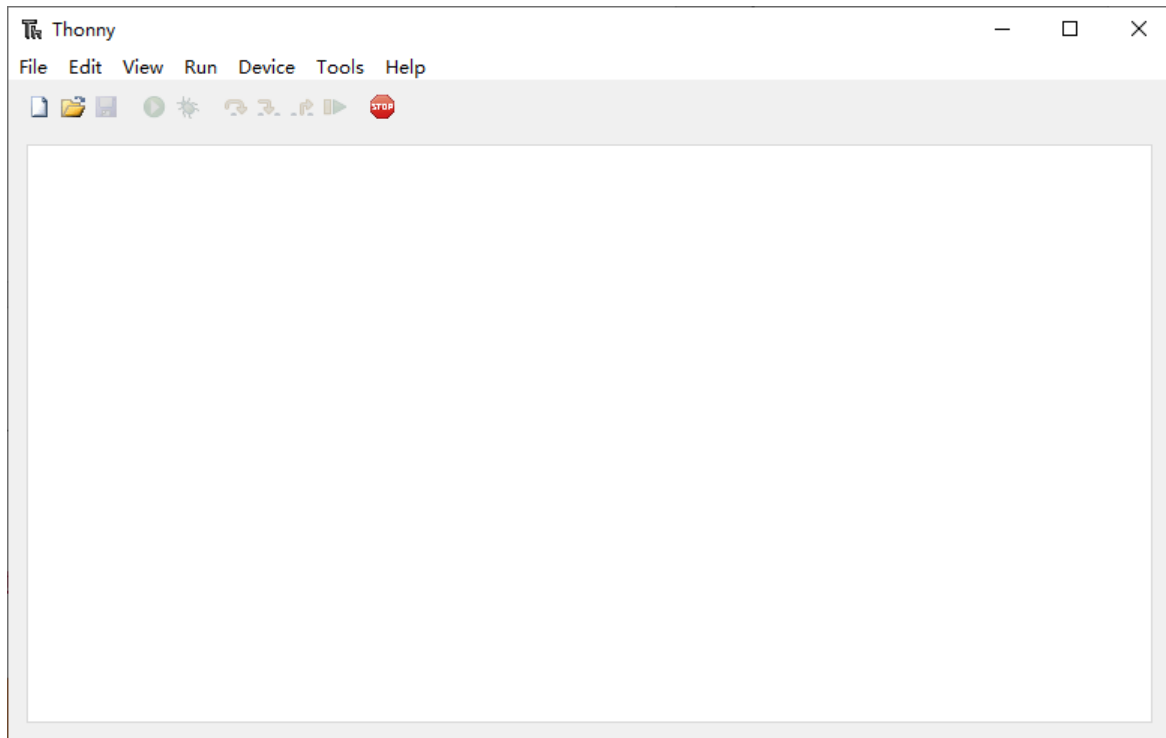


If you've checked "Create desktop icon" during the installation process, you can see the below icon on your desktop.

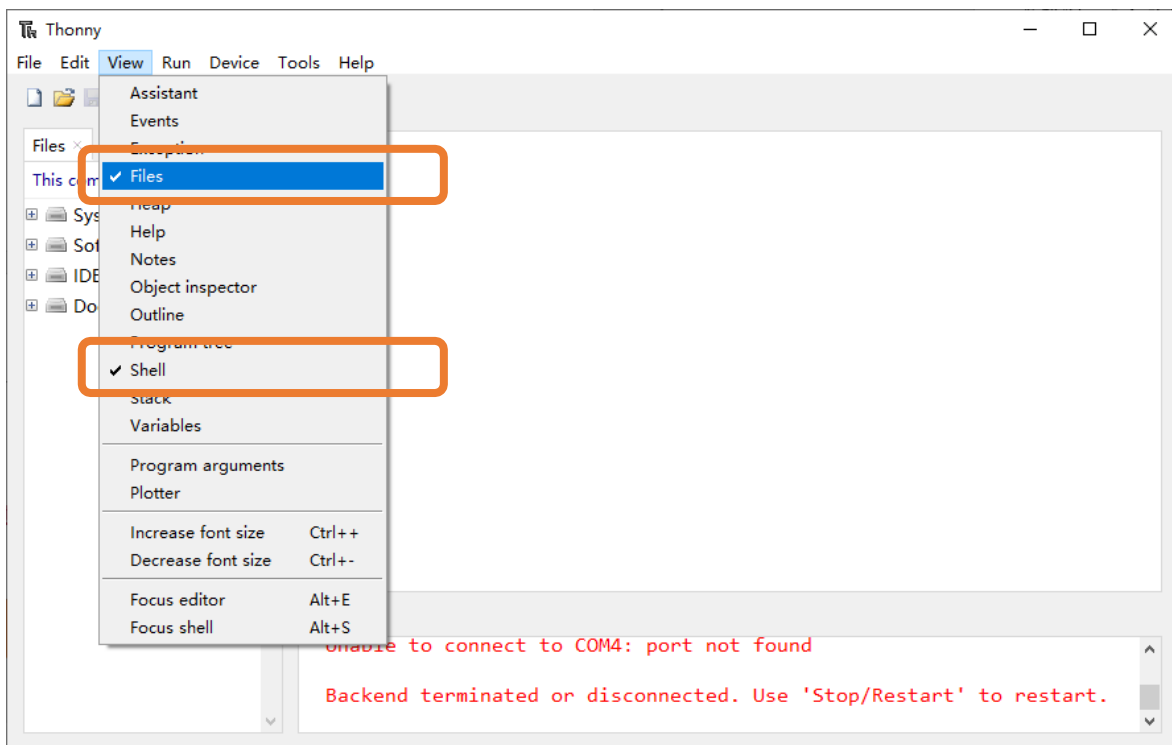


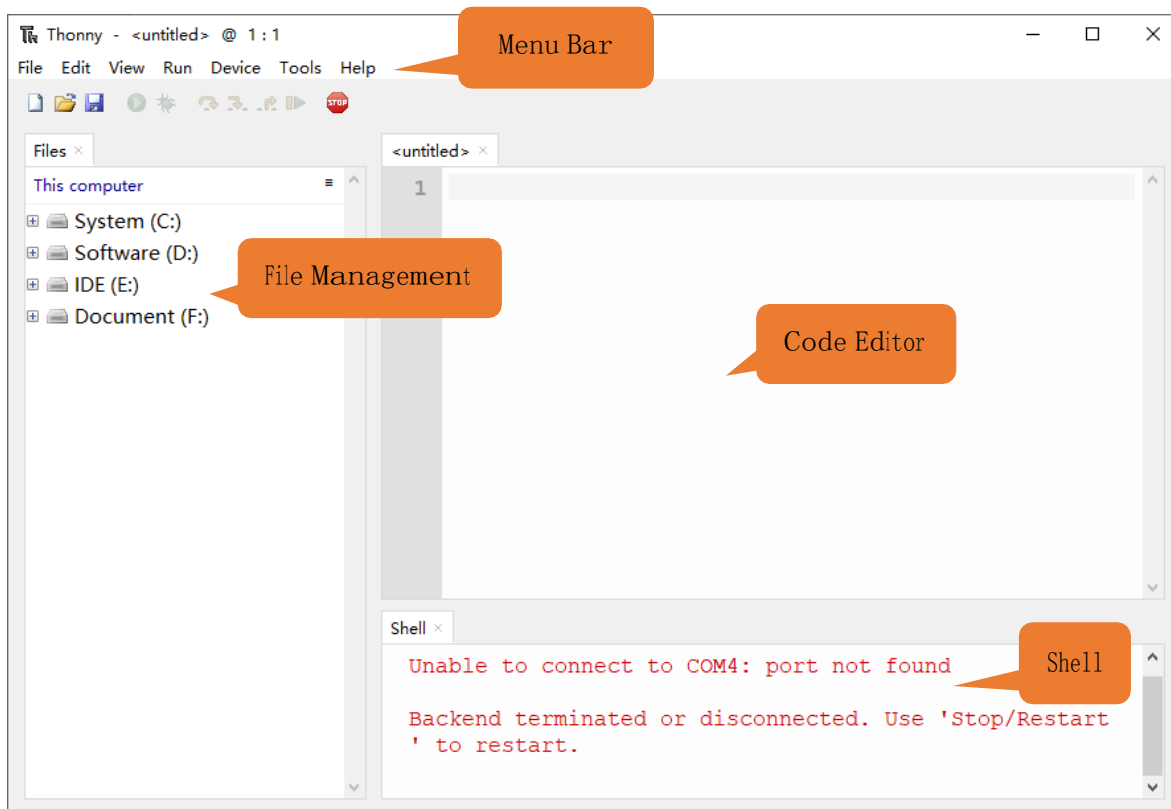
0.2 Basic Configuration of Thonny

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



Select "View"--- "Files" and "Shell".





0.3 Burning Micropython Firmware (Important)

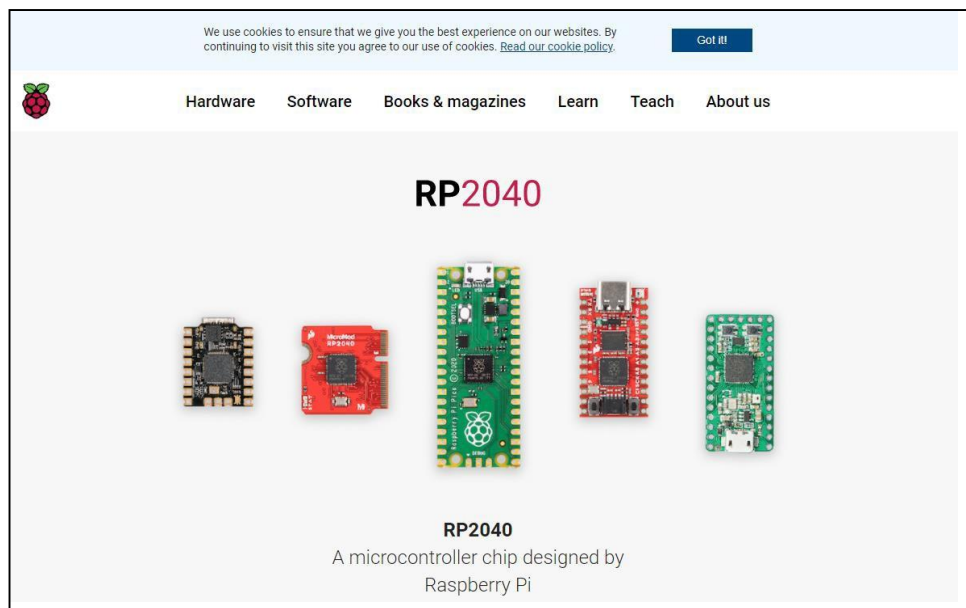
To run Python programs on Raspberry Pi Pico, we need to burn a firmware to Raspberry Pi Pico first.

Downloading Micropython Firmware

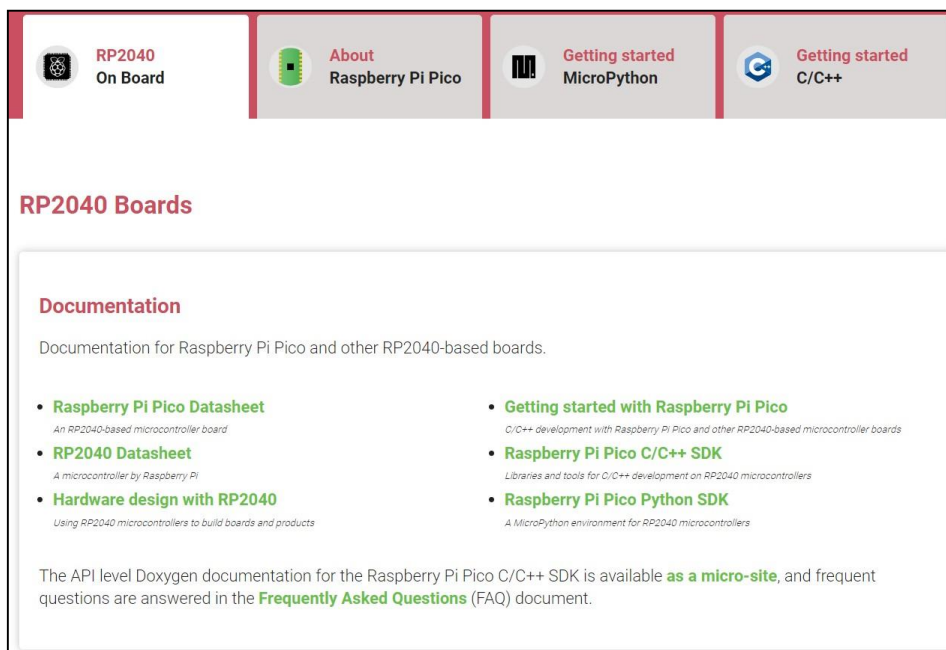
Option 1:

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

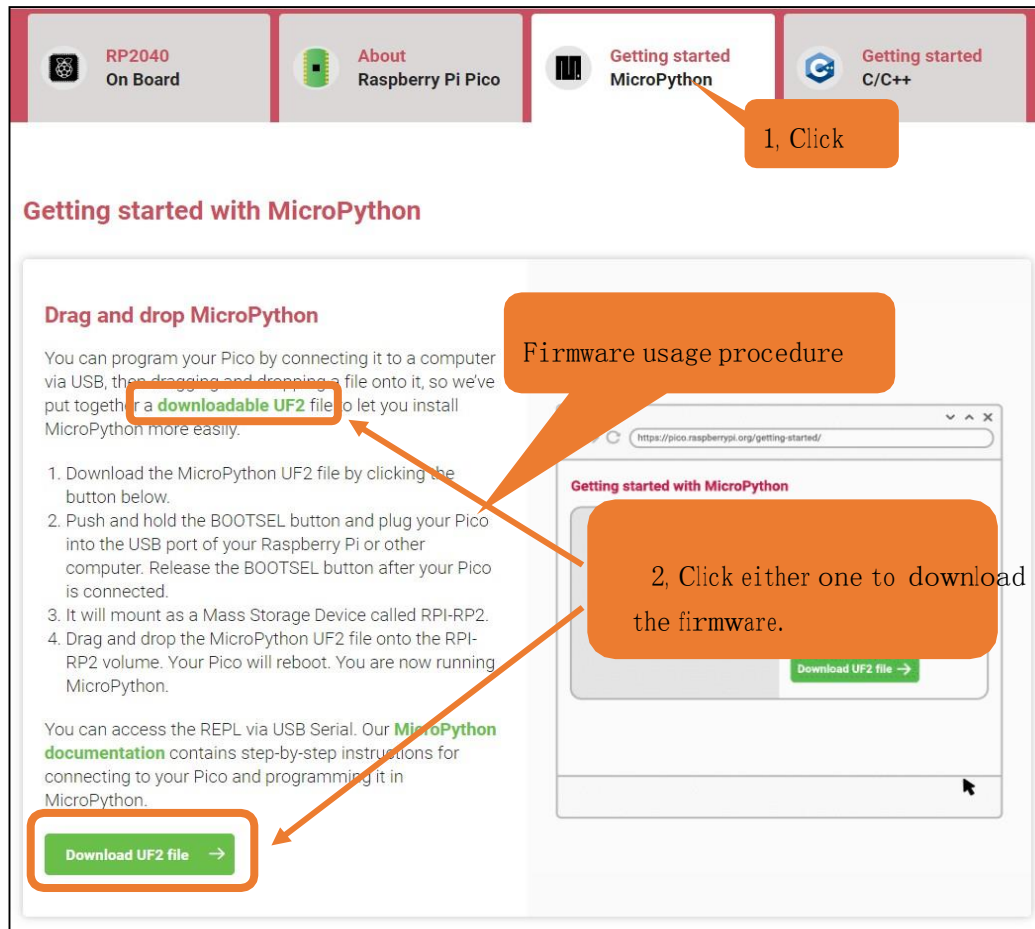
1, Click the link above and you can see the following interface



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



3, Click "Getting started MicroPython" to enter the firmware downloading page.

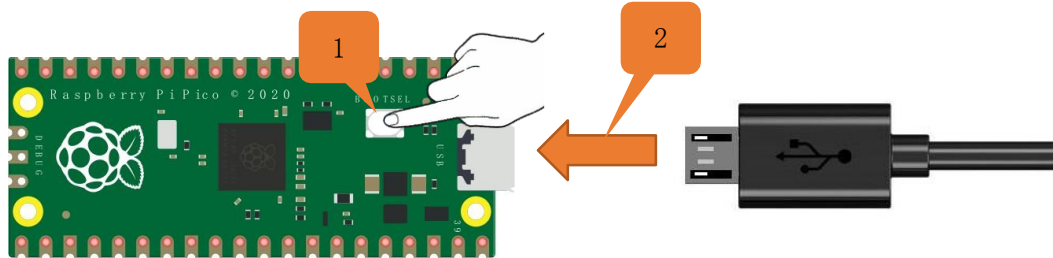


Option 2: If you cannot download it due to network issue or other reasons, you can use the one we have prepared, which locates at the following file path:

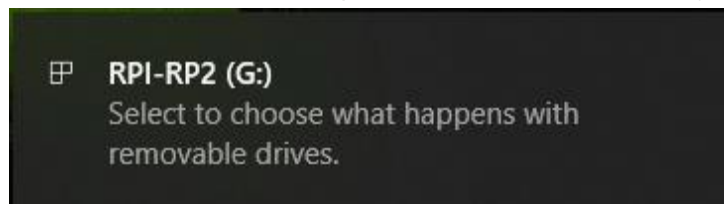
Raspberry_Pi_Pico Kit Tutorial\Lesson2-Python\Python_Firmware

Burning a Micropython Firmware

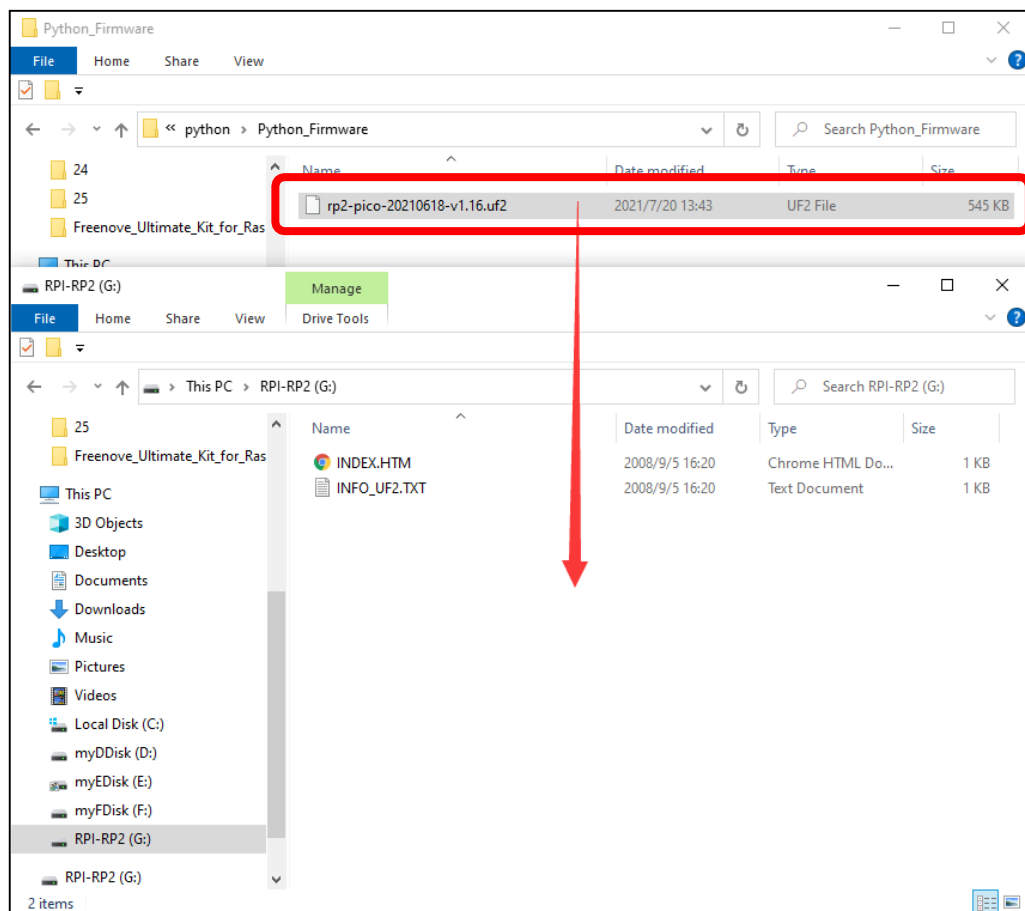
1. Connect a USB cable to your computer.
2. Long press BOOTSEL button on Raspberry Pi Pico and connect it to your computer with the USB cable.



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



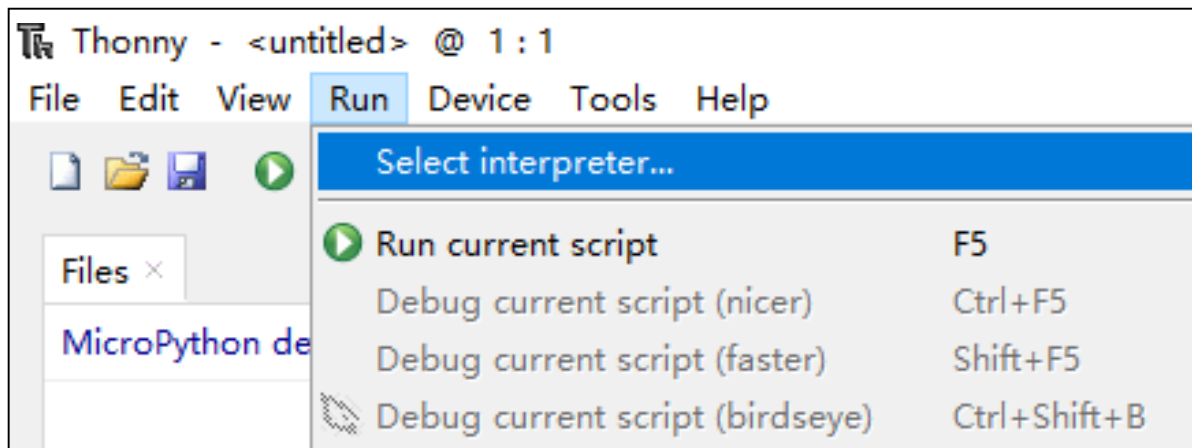
4. Copy the file(rp2-pico-20210618-v1.16.uf2) to RPI-RP2 and wait for it to finish, just like copy file to a U disk.



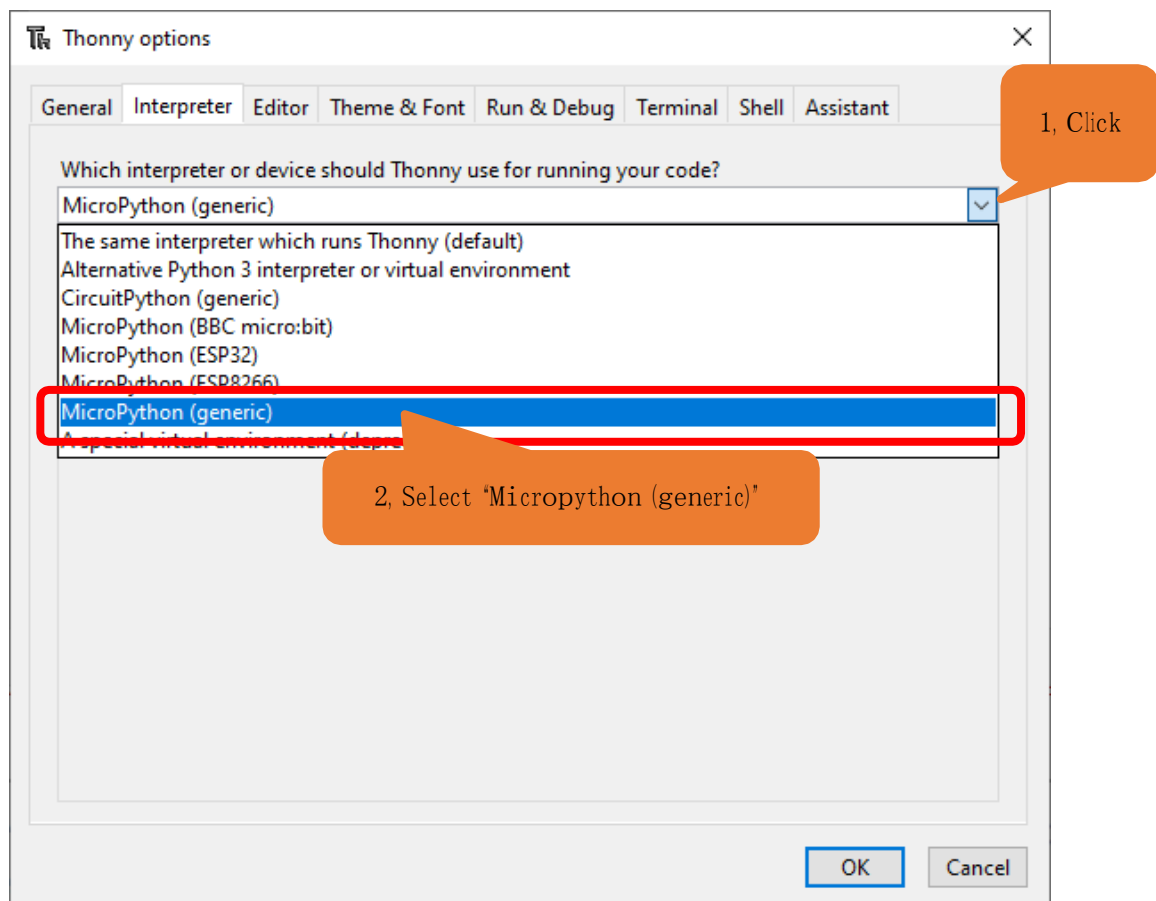
5. When the firmware finishes programming, Raspberry Pi Pico will reboot automatically. After that, you can run Micropython.

0.4 Thonny Connected to Raspberry Pi Pico

1. Open Thonny, click "run" and select "Select interpreter..."



2. Select "Micropython (generic)" or "Micropython (Raspberry Pi Pico)". How to select "Micropython (generic)"? As shown below:

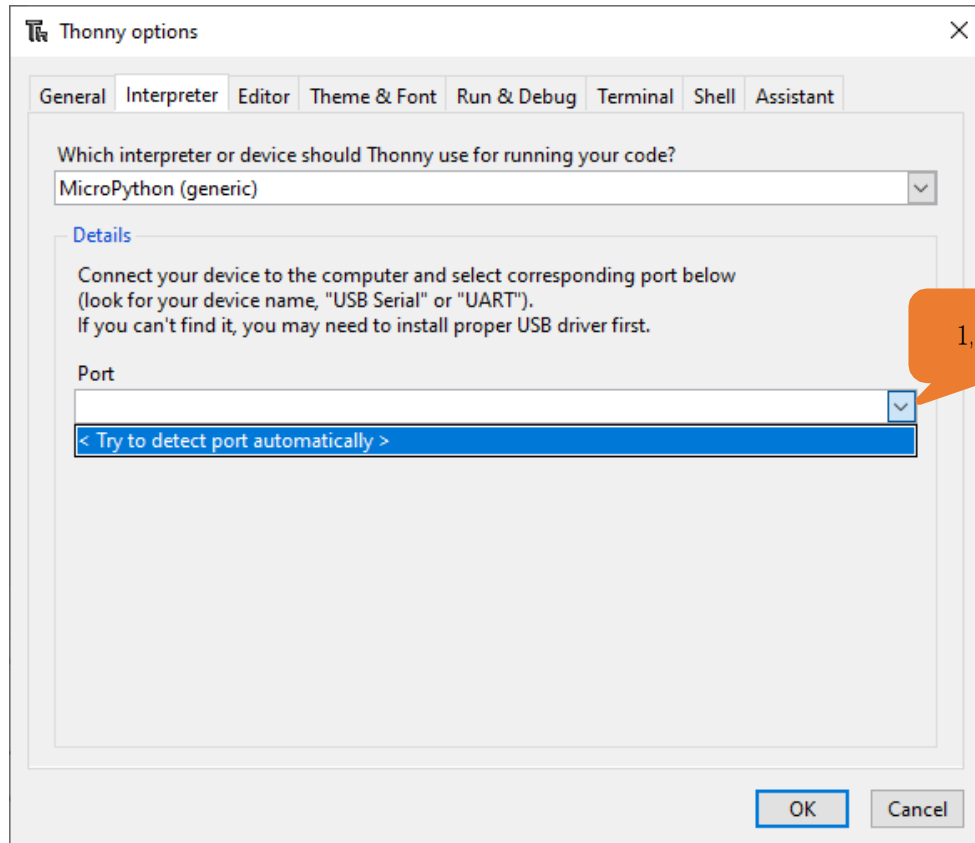


www.cokoino.com

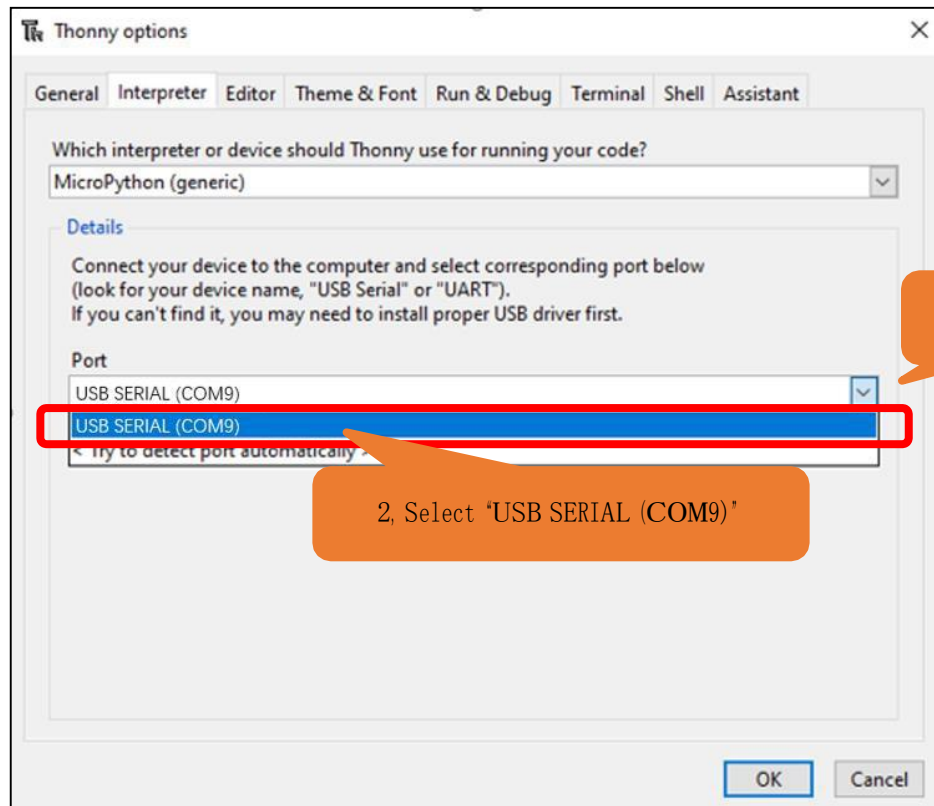
Select "USB-SERIAL (COMx)", The number x of COMx may varies among different computers. You only need to make sure selecting USB-SERIAL (COMx).

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

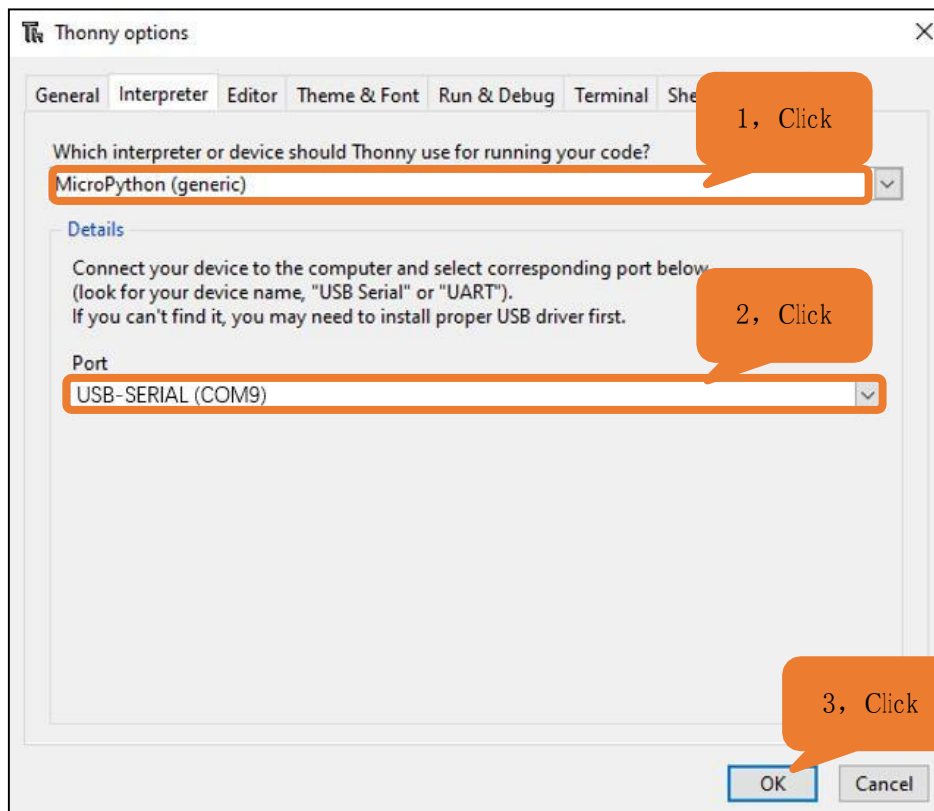
Step 1: When Pico doesn't connect to computer, open Thonny, click "Run", select "Select interpreter" and then a dialog box will pop up, click "Port" and you can check the ports currently connected to your computer, as shown below:



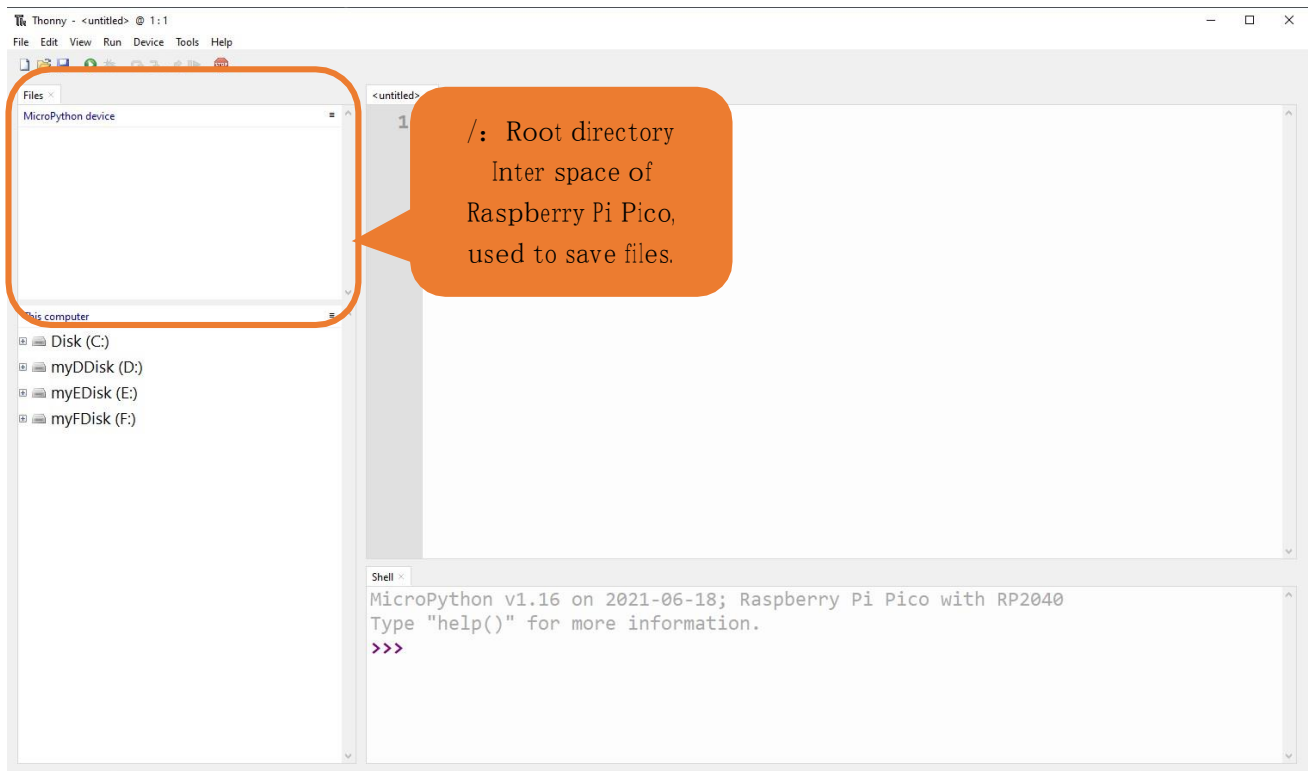
Step 2: Close the dialog box. Connect Pico to your computer, click "Run" again and select "Select interpreter". Click "Port" on the pop-up window and check the current ports. Now there is a newly added port, with which Pico communicates with the computer.



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



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

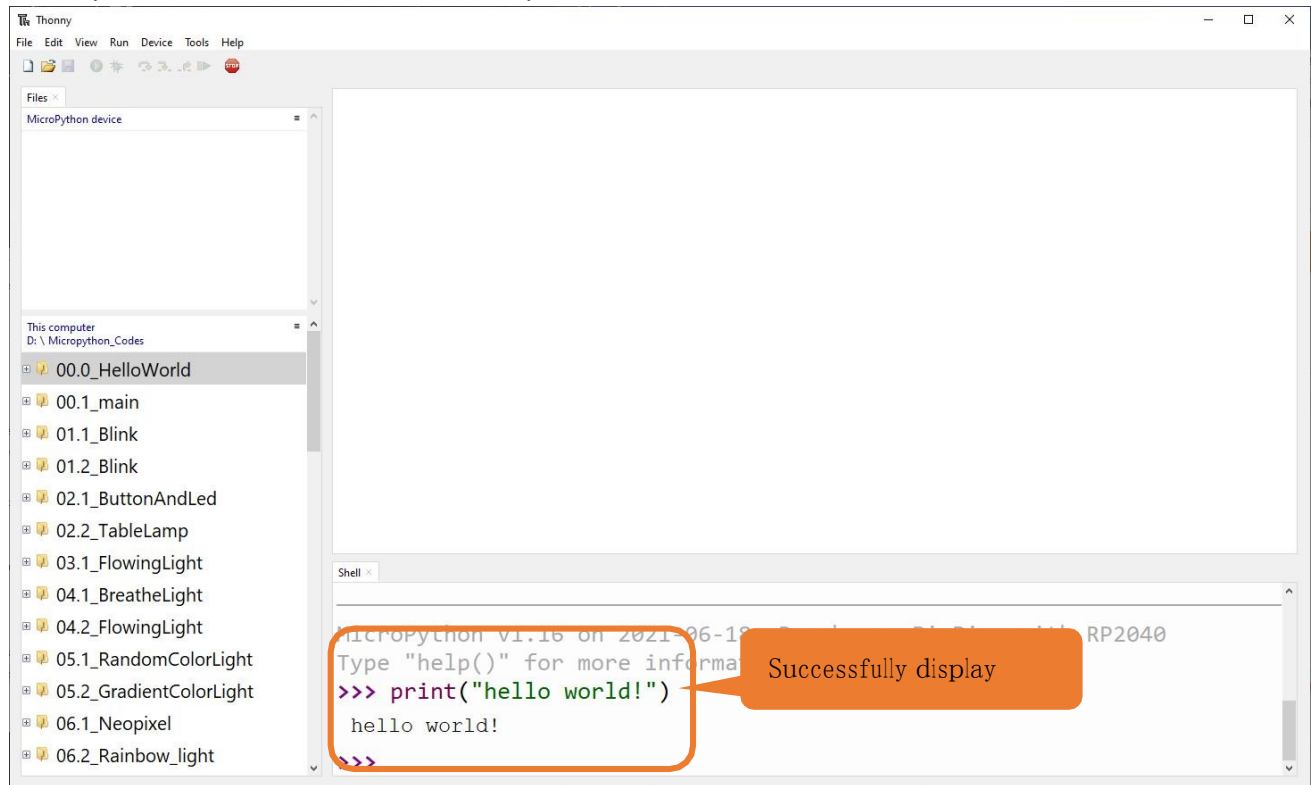


5. So far, all the preparations have been made.

0.5 Testing codes (Important)

Testing Shell Command

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



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:

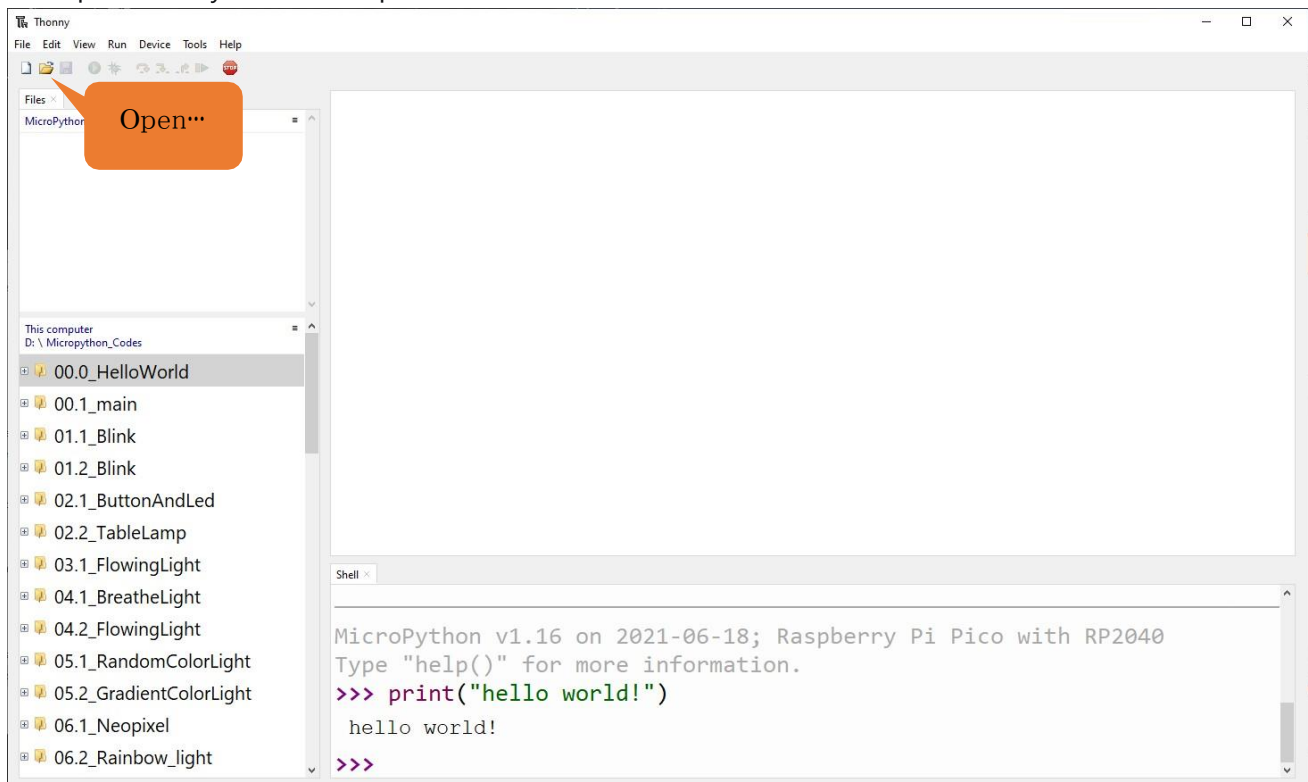
1. Users can use Thonny to compile or debug programs.
2. Through the "Shell" window, users can read the error information and output results generated during the running of the program and query related function information online to help improve the program.

Disvantages:

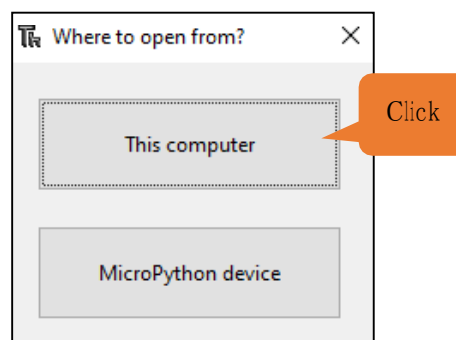
1. To run Raspberry Pi Pico online, you have to be connected to a computer and run with Thonny.
2. If Raspberry Pi Pico disconnects from computer, the program won't run again when they reconnect to each other.

Opertation

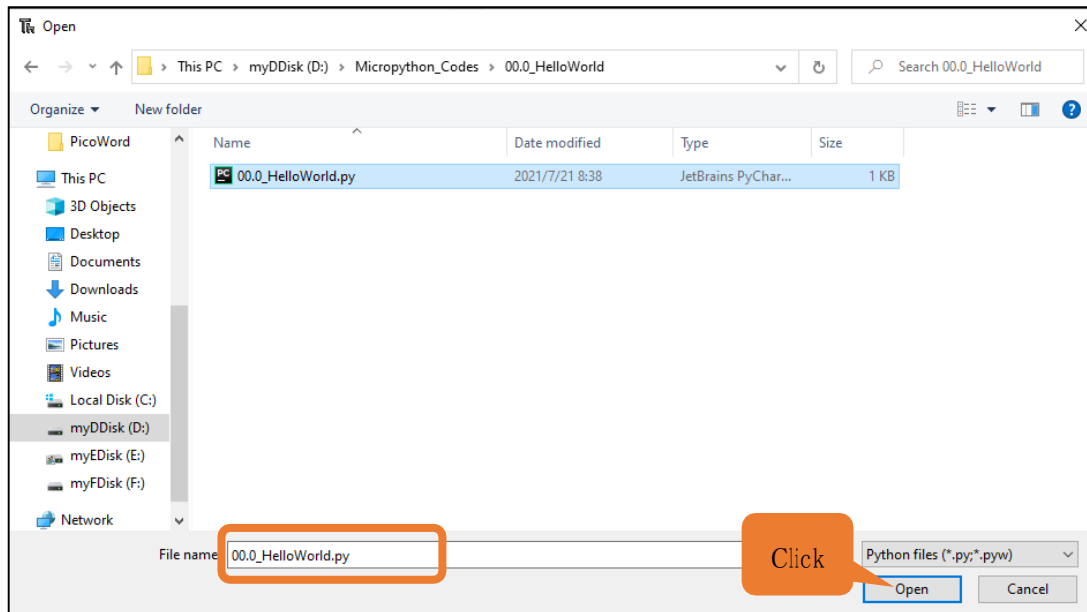
1. Open Thonny and click "Open..."



2. On the newly pop-up window, click "This computer".



In the new dialog box, select '00.0_HelloWorld.py' in "Raspberry_Pi_Pico Kit Tutorial\Lesson2-Python\Python_Codes\00.0_HelloWorld" folder.

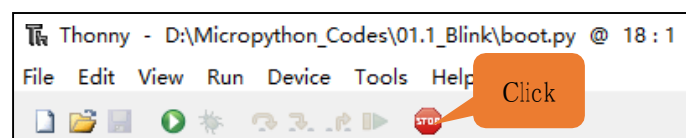


Click "Run current script" to execute the program and "Hello World!", "Welcome Cokoino" will be printed in "Shell".



Exiting Running Online

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



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.

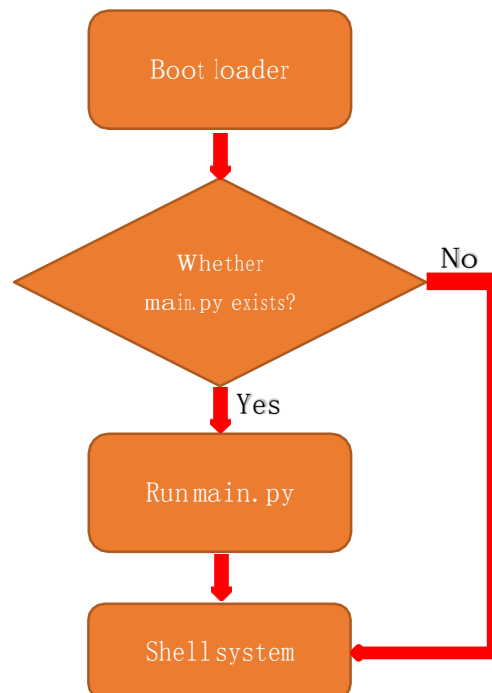
Advantage: It can run programs when powered up without connected to computer and Thonny.

Disadvantage: The program will stop automatically when error occurs or Raspberry Pi Pico is out of power.

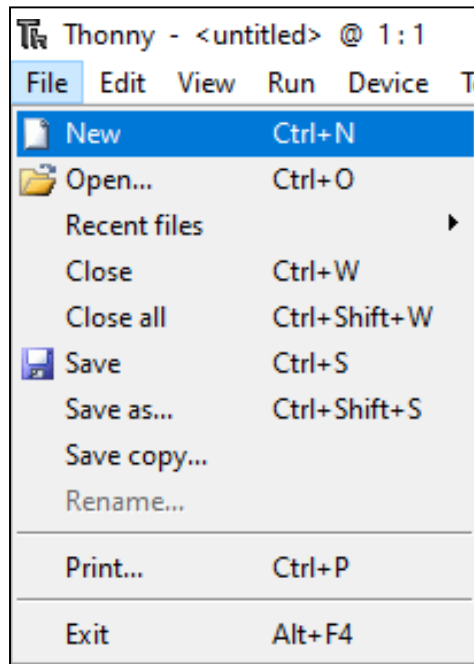
Code cannot be changed easily.

Operation

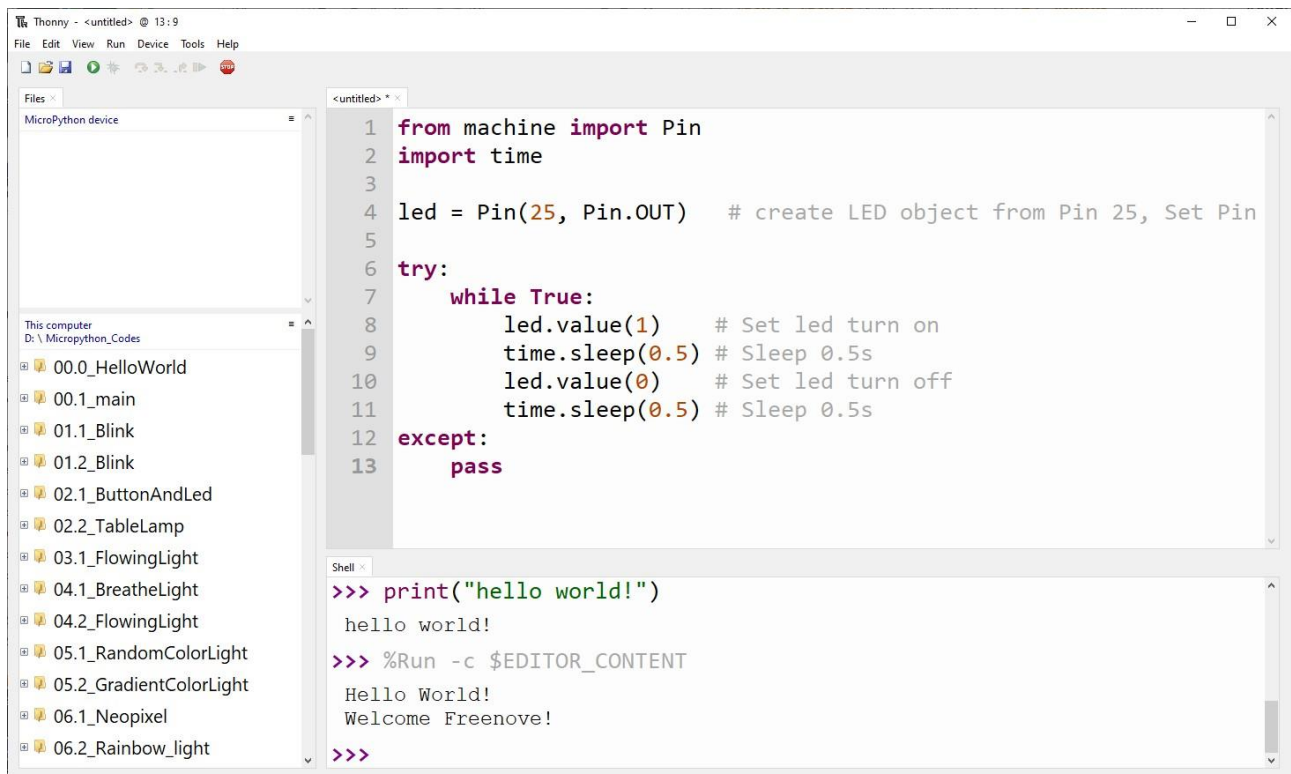
Once powered up, Raspberry Pi Pico will automatically check whether there is main.py existing on the device. If there is, it runs the programs in main.py and then enter shell command system. (If you want the code to run offline, you can save it as main.py); If main.py doesn't exist, it will enter shell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



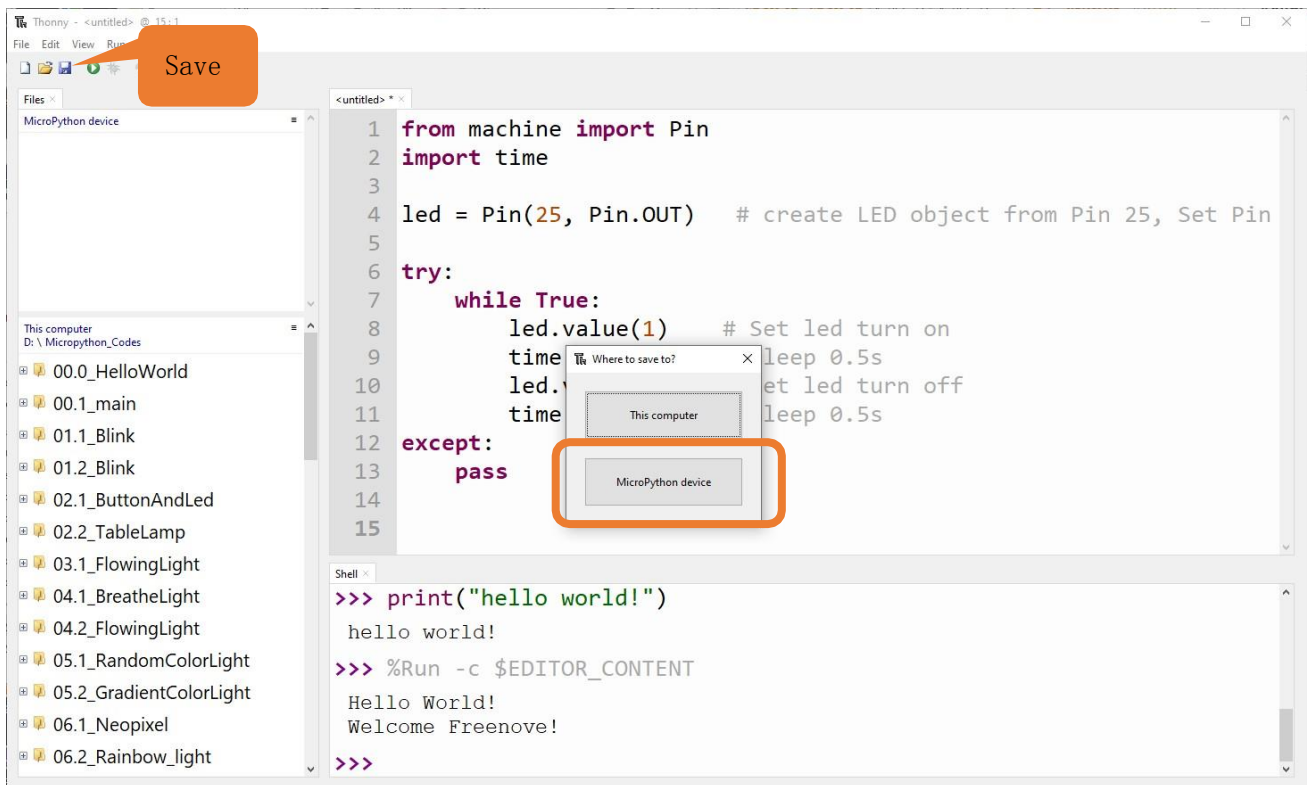
1. Click "File"---"New" to create and write codes.



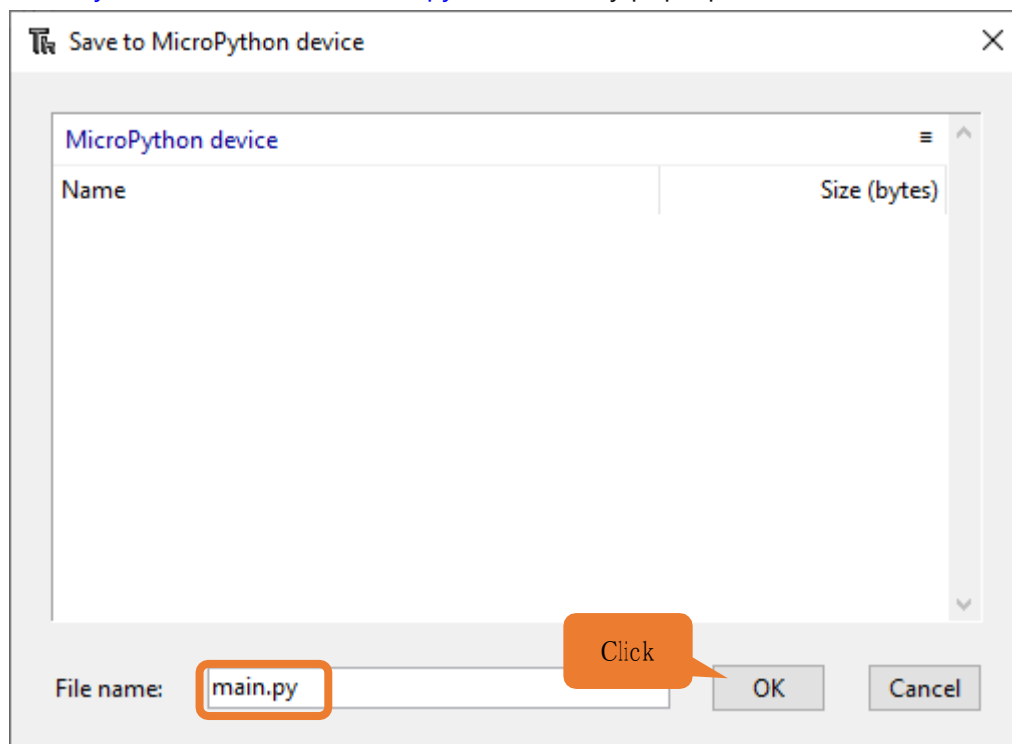
2. Enter codes in the newly opened file. Here we use codes of "01.1_Blink.py" as an example.



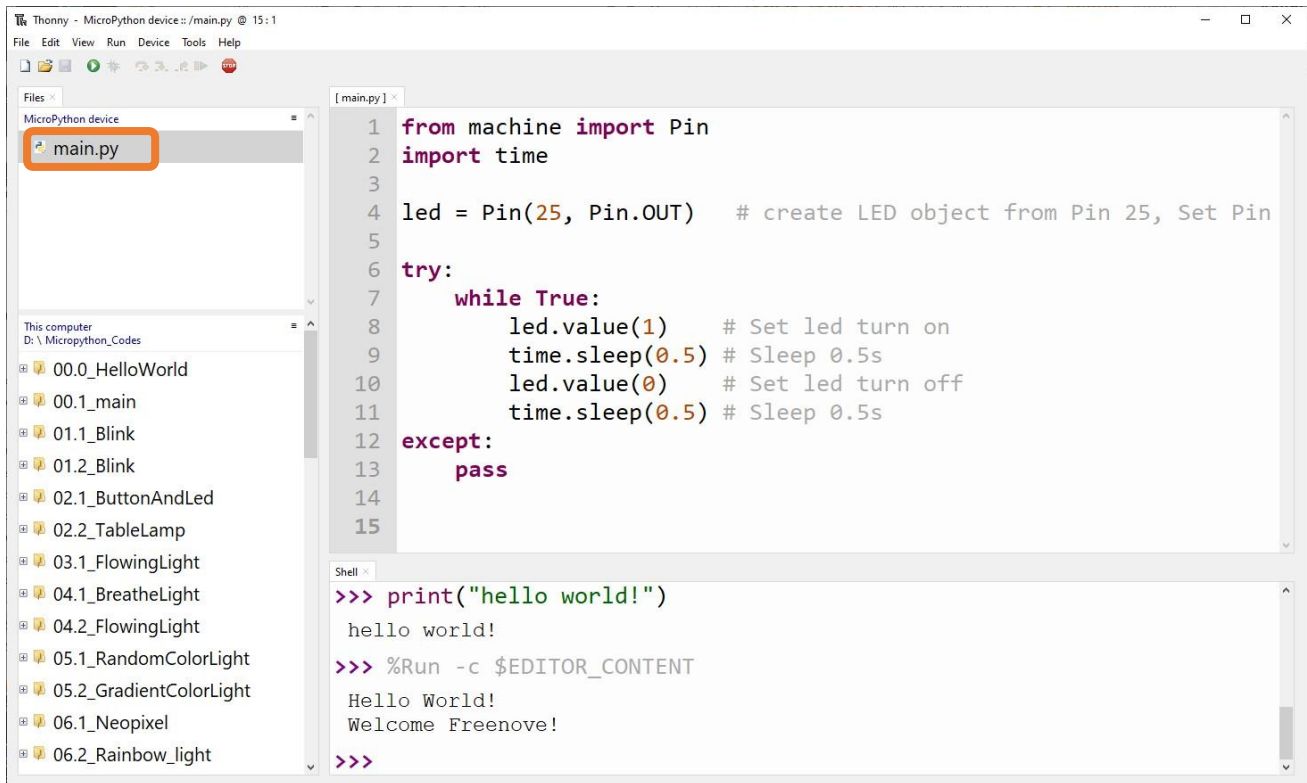
3. Click "Save" on the menu bar. You can save the codes either to your computer or to Raspberry Pi Pico.



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



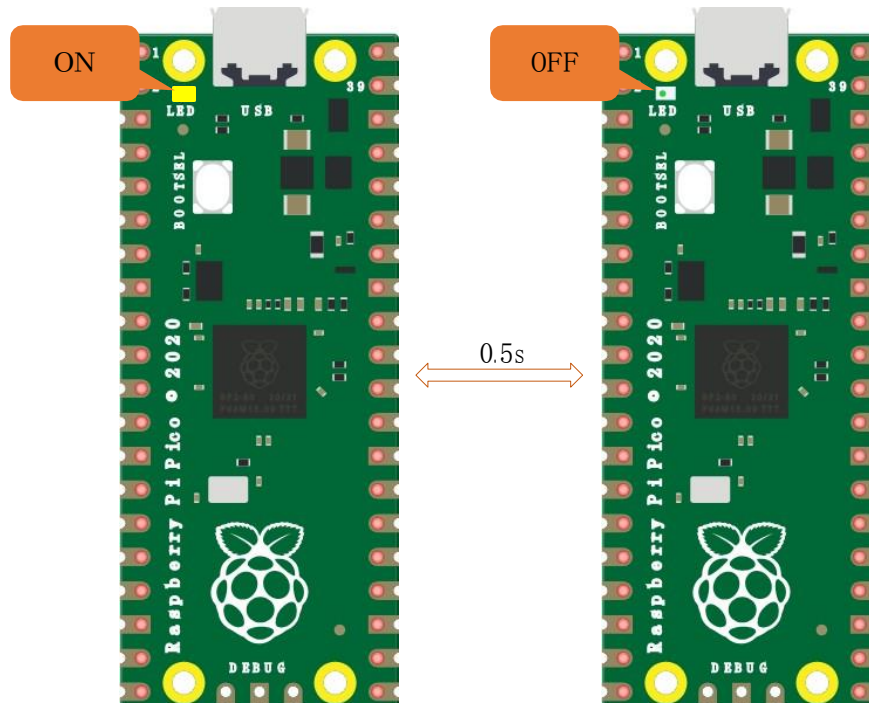
5. You can see that codes have been uploaded to Raspberry Pi Pico.



```
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
```

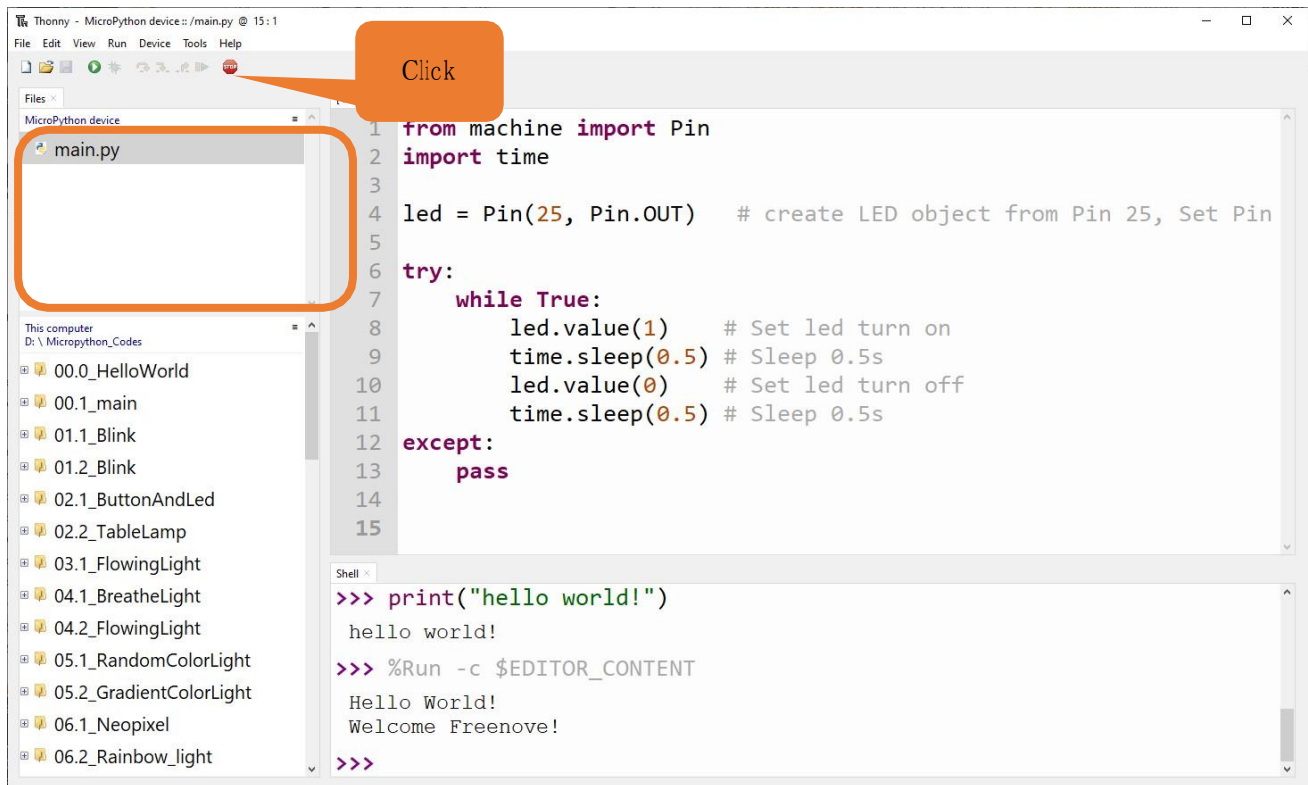
```
>>> print("hello world!")
hello world!
>>> %Run -c $EDITOR_CONTENT
Hello World!
Welcome Freenove!
>>>
```

6. 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 offline.

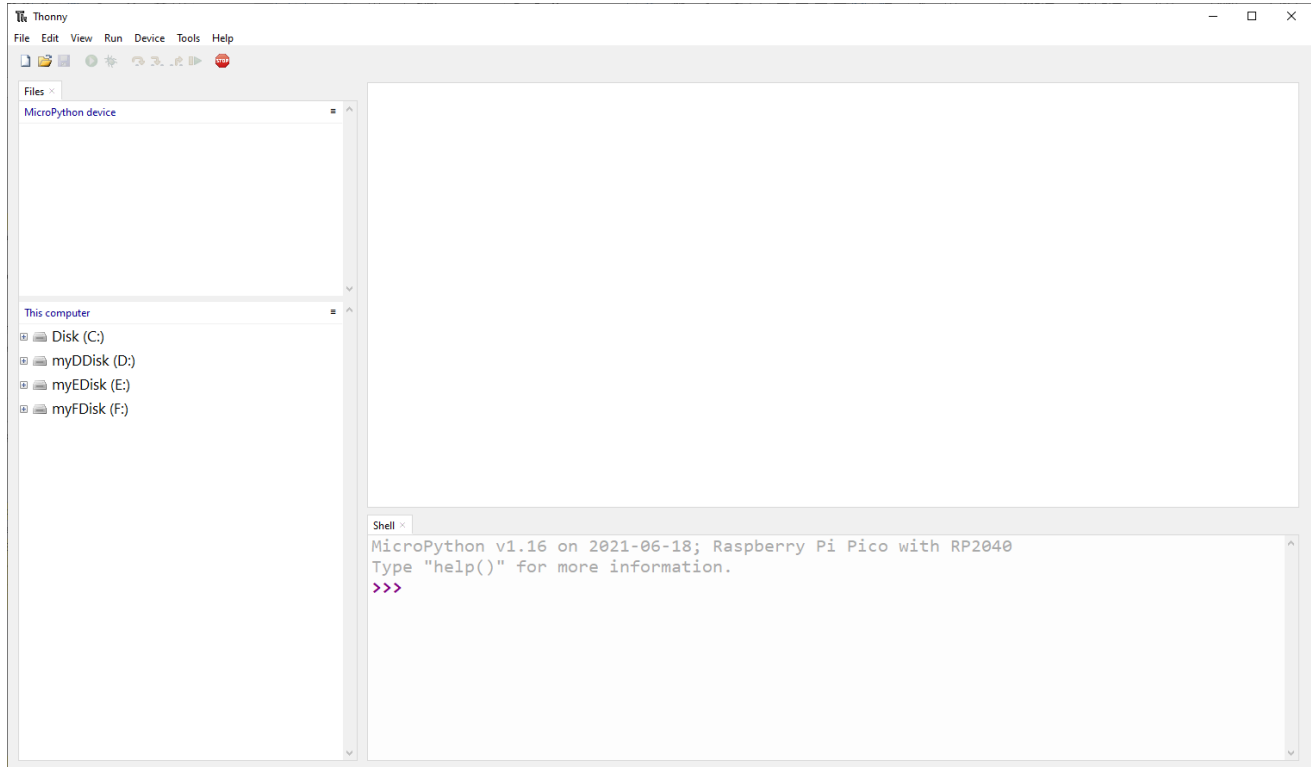


If it doesn't work, please click on "stop/restart backend" for more times or reconnect Pico.

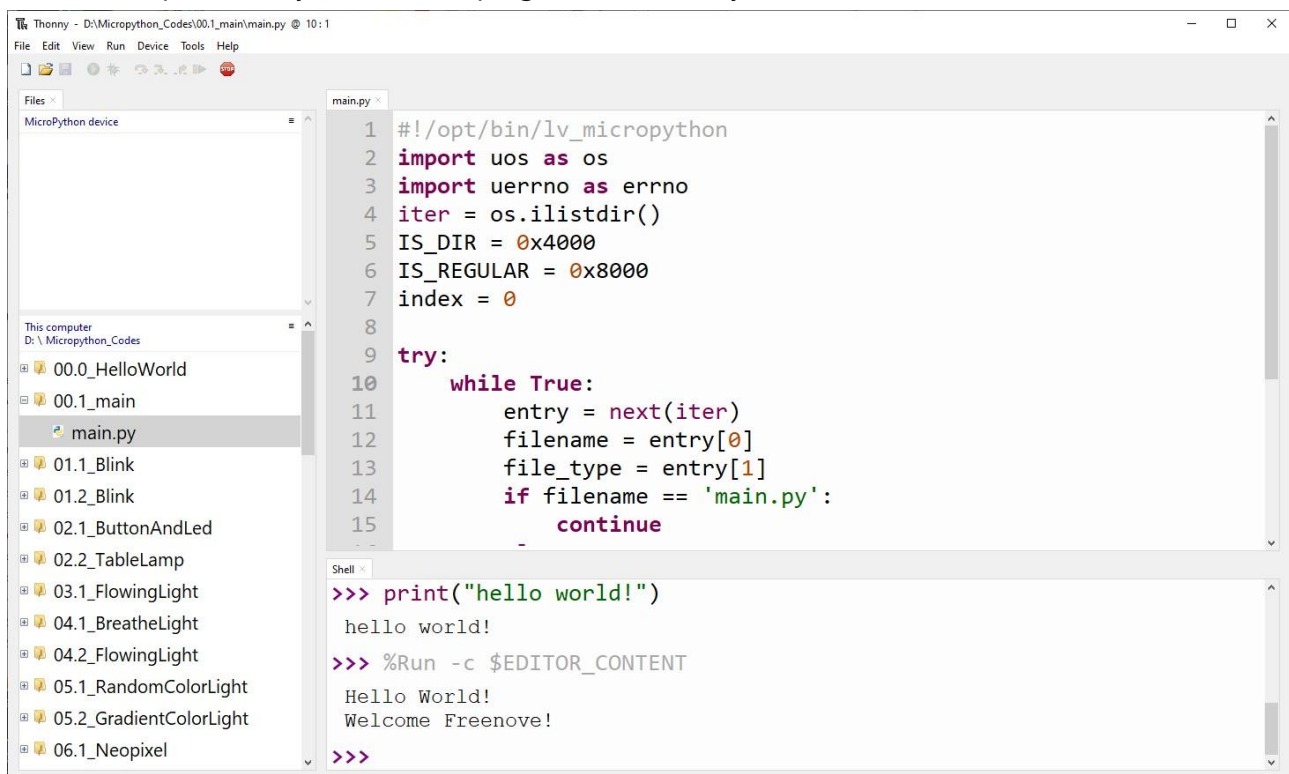


We provide a main.py file for running offline. The code added to main.py is a bootstrap that executes the user's code file. All you need to do is upload the offline project's code file (.py) to the Raspberry Pi Pico device.

1. Move the program folder '[Raspberry_Pi_Pico Kit Tutorial\Lesson2-Python\Python_Codes](#)' to disk(D) in advance with the path of '[D:/Micropython_Codes](#)'. Open "Thonny".

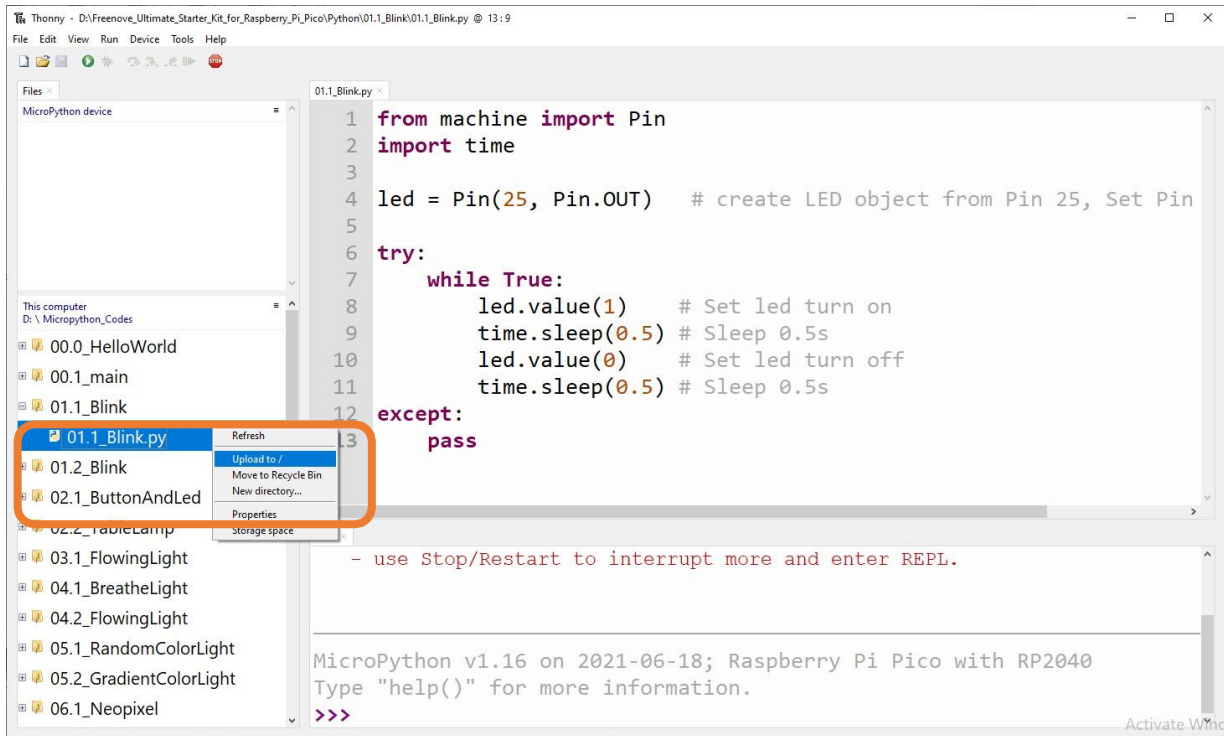


2. Expand "[00.1_main](#)" in the "[Micropython_Codes](#)" in the directory of disk(D), and double-click main.py, which is provided by us to enable programs in "MicroPython device" to run offline.

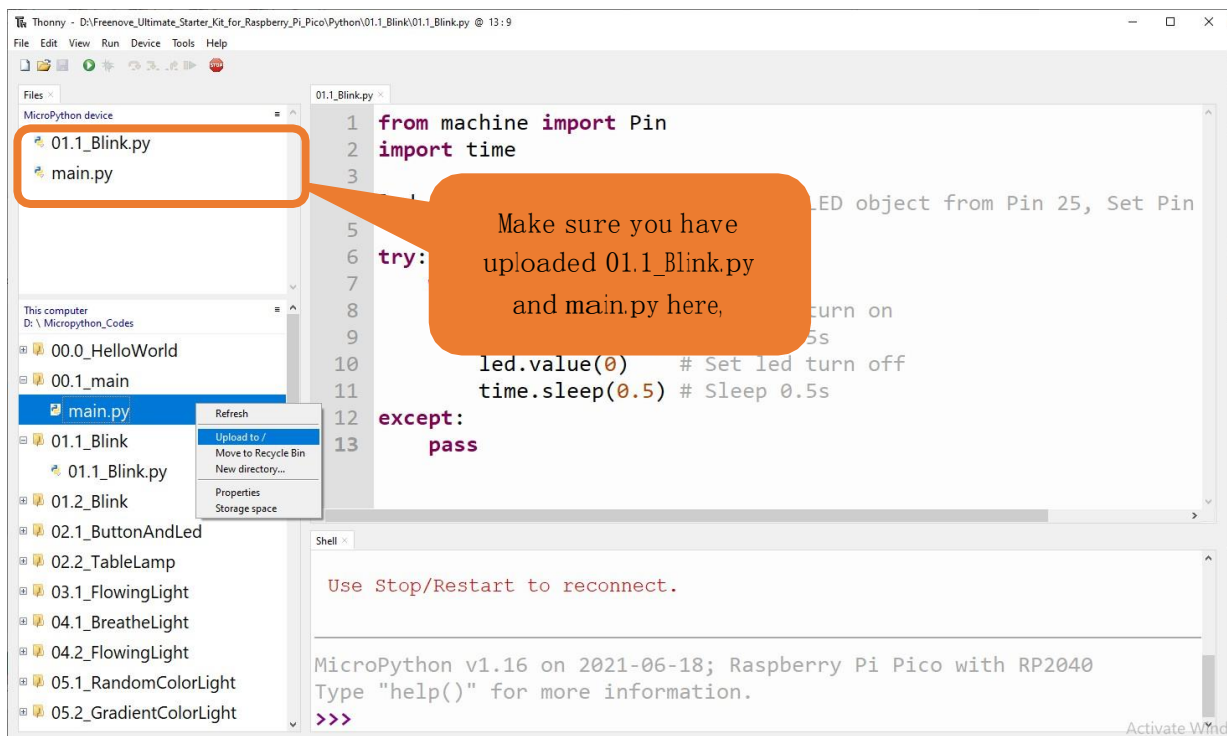


Here we use 00.1 and 01.1 cases as demonstration. The LED on Raspberry Pi Pico is used to show the result, which uses GP25 pin. If you have modified 01.1_Blink.py file, you need to change it accordingly.

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.



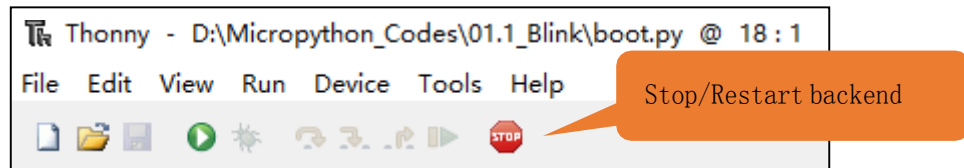
Upload main.py in the same way.



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

Note:

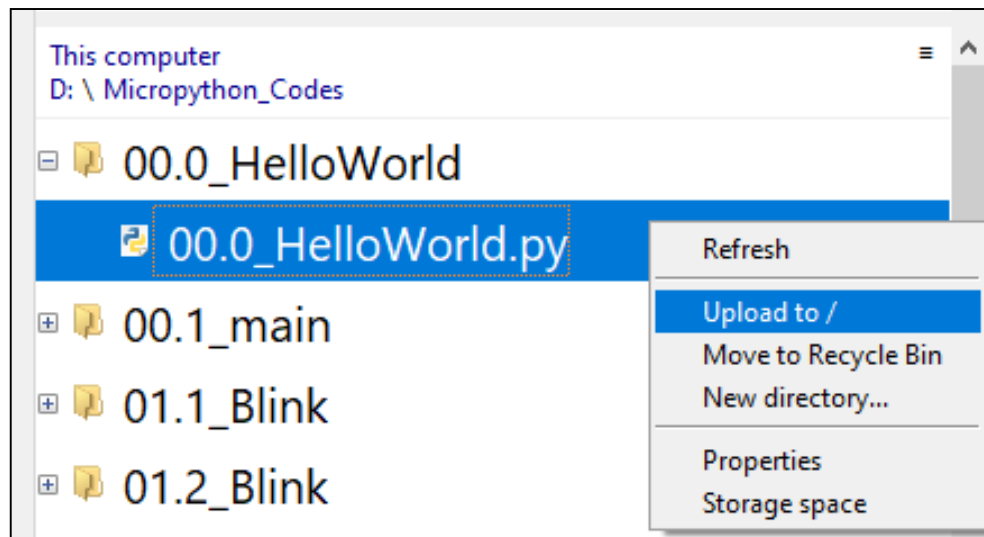
Codes here are run offline. If you want to stop running offline and enter Shell, just click "Stop" in Thonny.



0.6 Thonny Common Operation

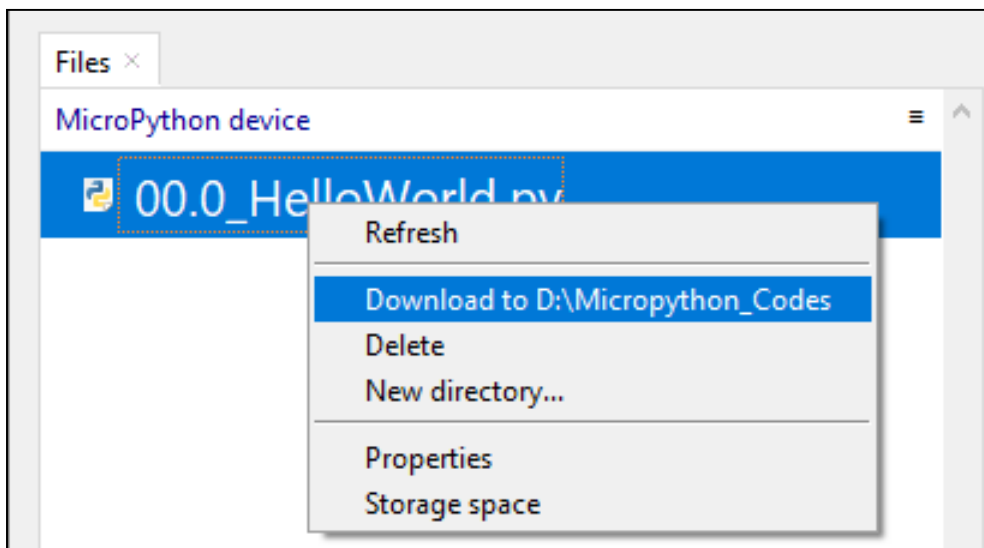
Uploading Code to Raspberry Pi Pico

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.



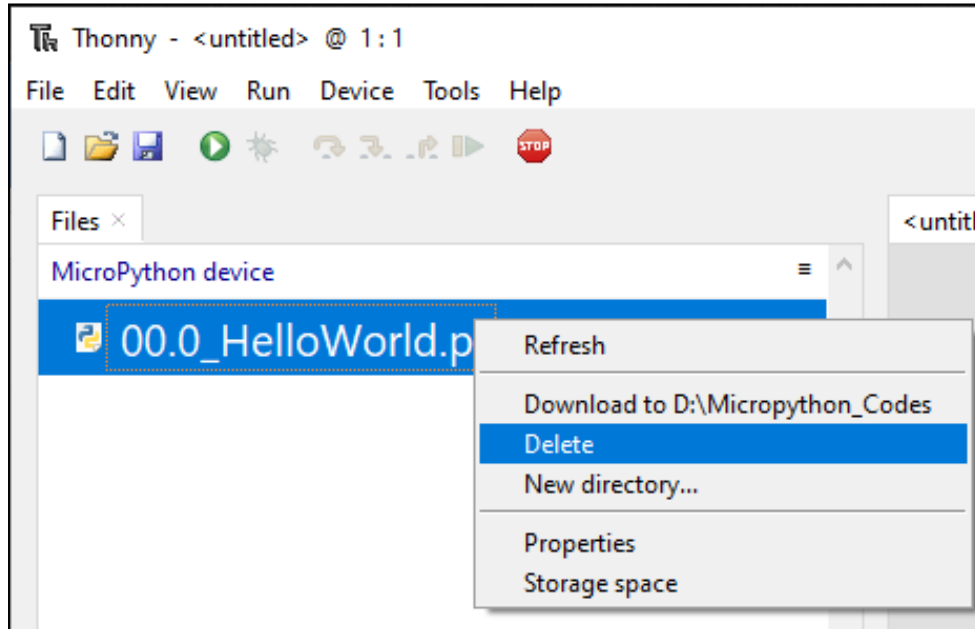
Downloading Code to Computer

Select "[00.0_HelloWorld.py](#)" in "MicroPython device", right-click to select "[Download to ...](#)" to download the code to your computer.



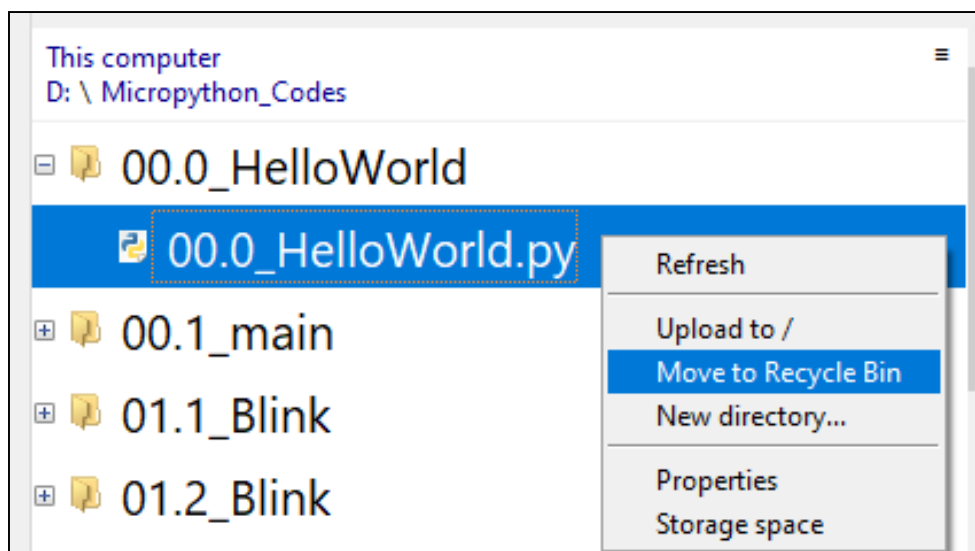
Deleting Files from Raspberry Pi Pico's Root Directory

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.



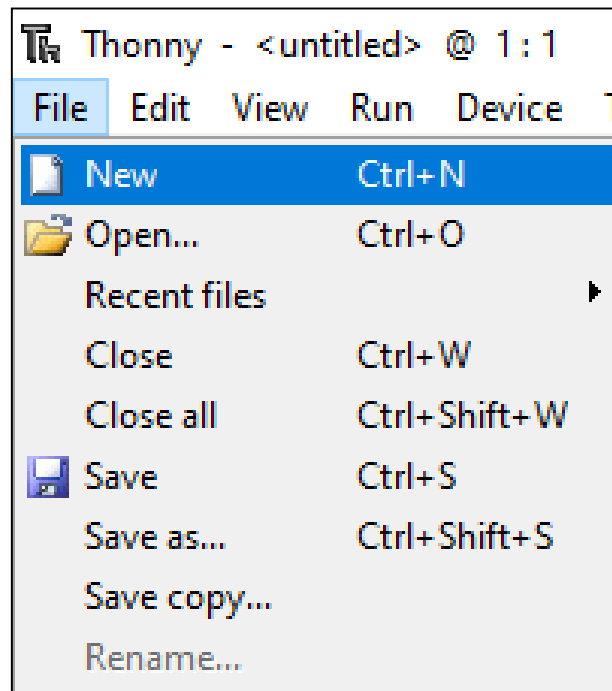
Deleting Files from your Computer Directory

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".



Creating and Saving the code

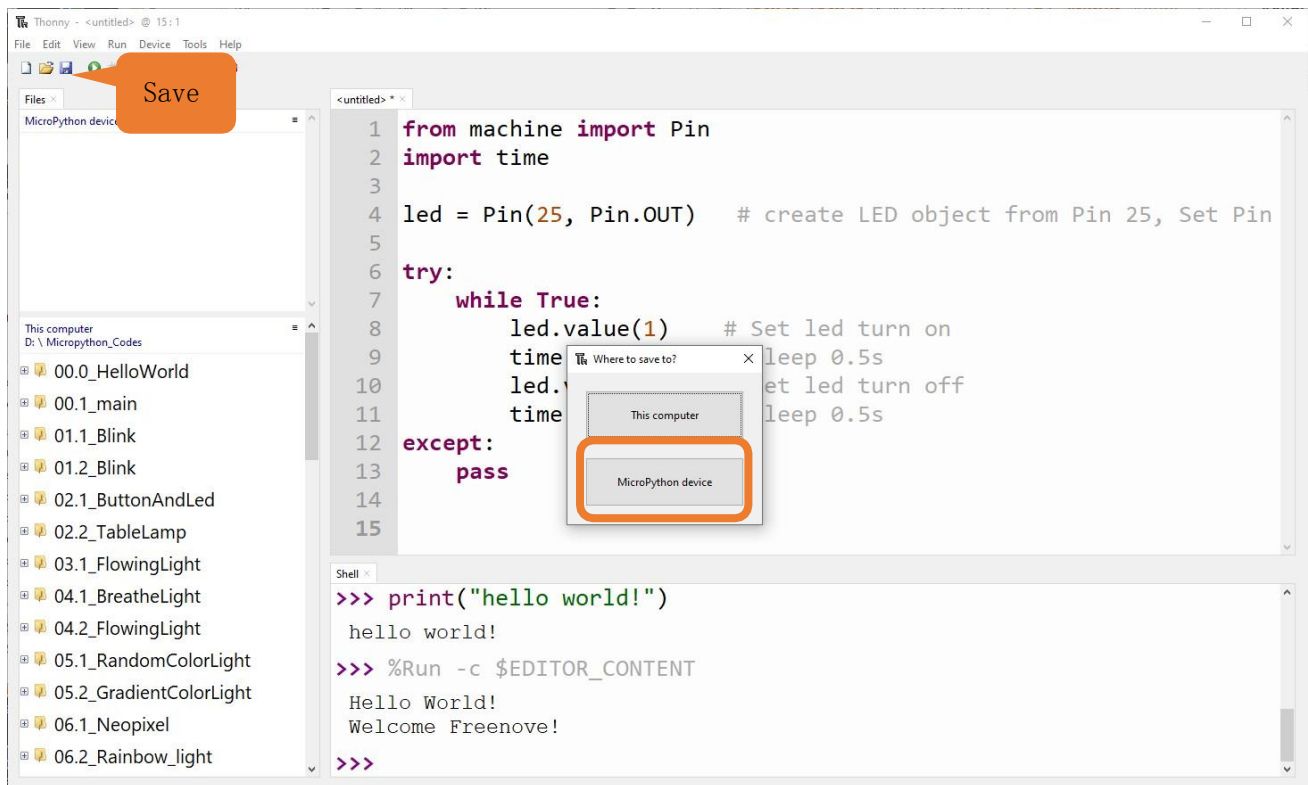
Click "File"---"New" to create and write codes.



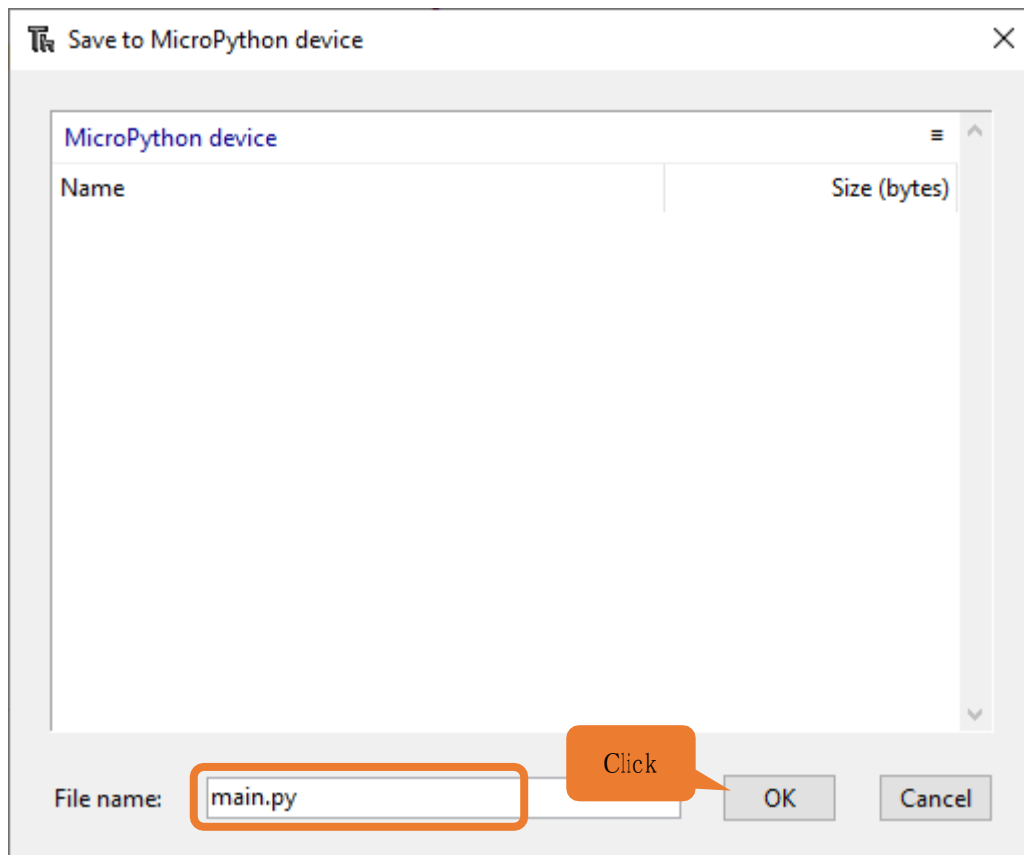
Enter codes in the newly opened file. Here we use codes of "01.1_Blink.py" as an example.



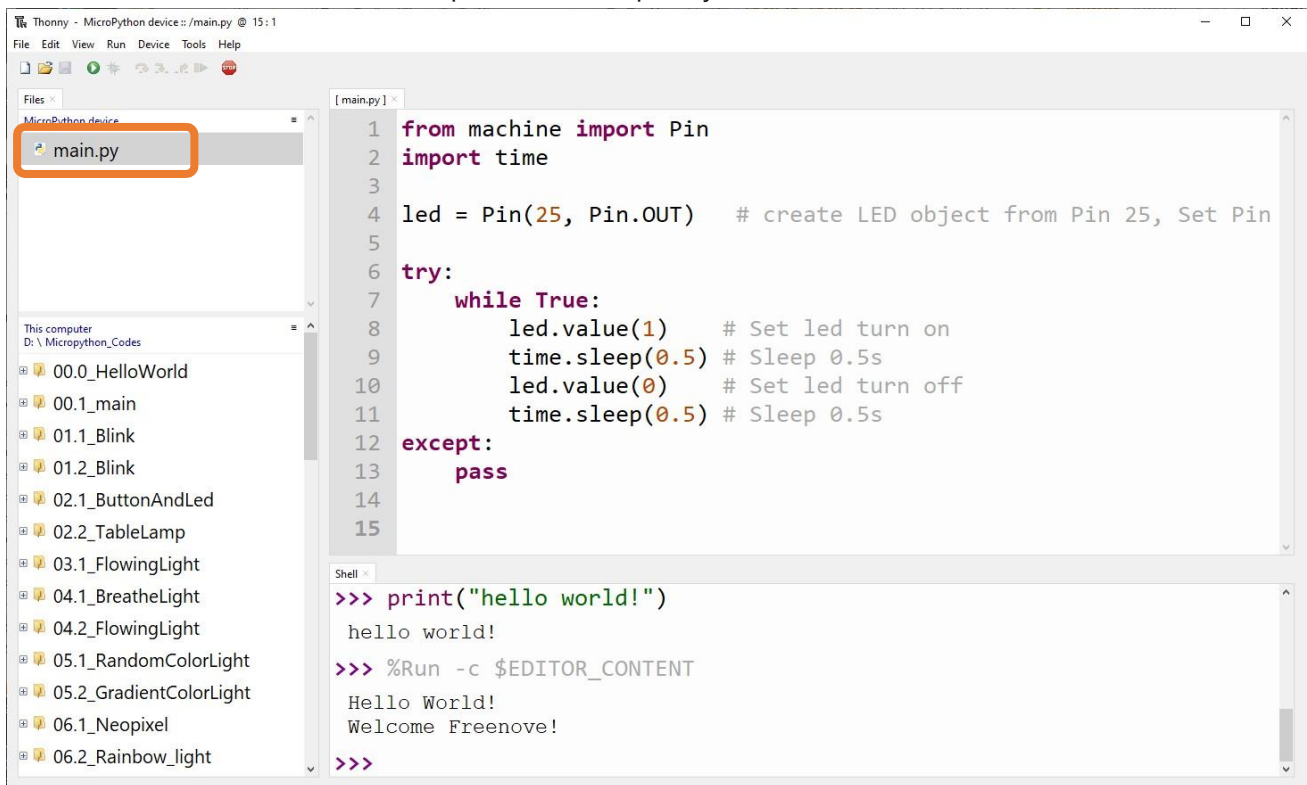
Click "Save" on the menu bar. You can save the codes either to your computer or to Raspberry Pi Pico.



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



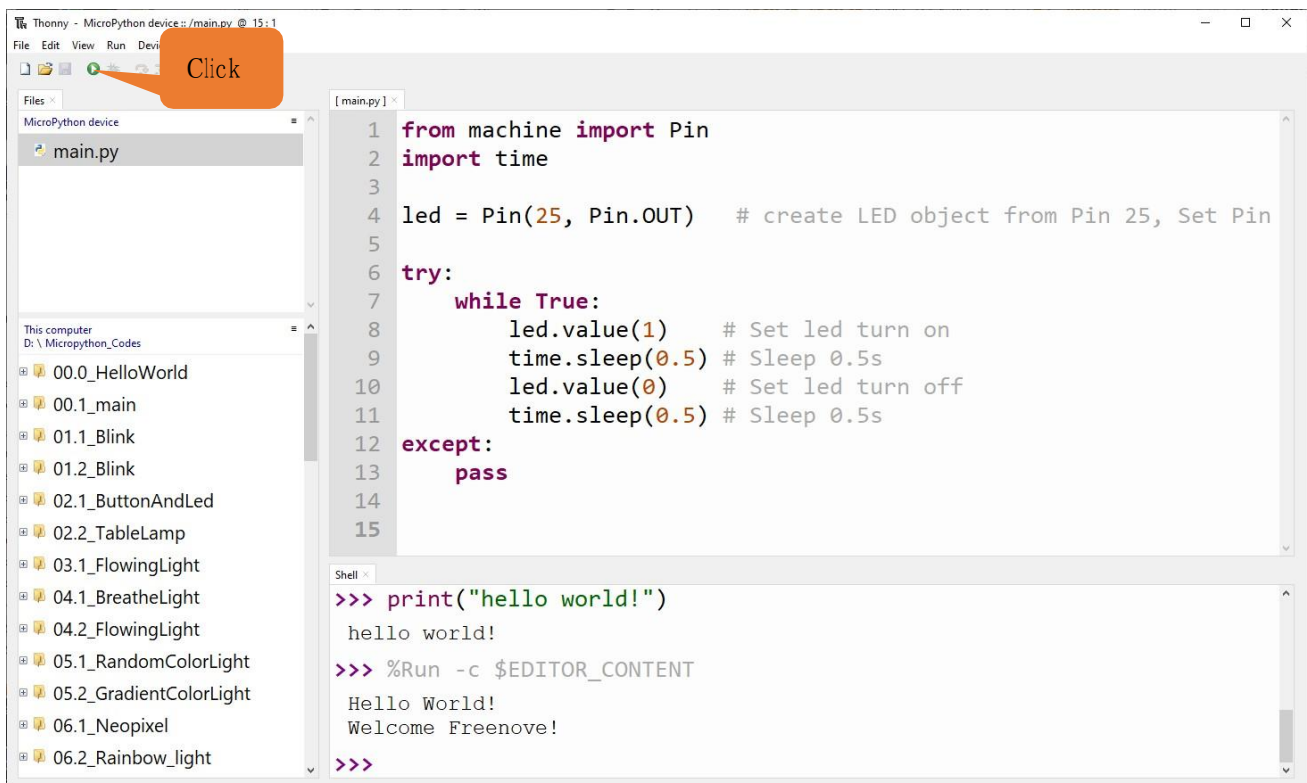
You can see that codes have been uploaded to Raspberry Pi Pico.



```
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
```

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

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
```

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

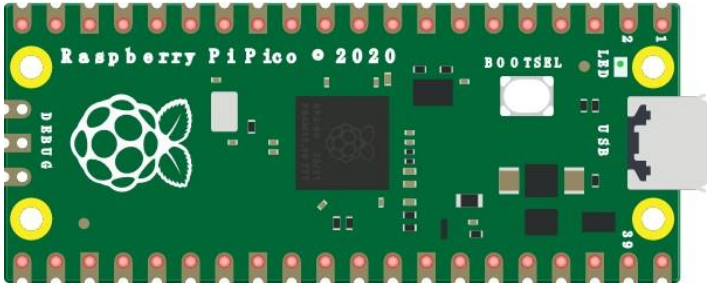
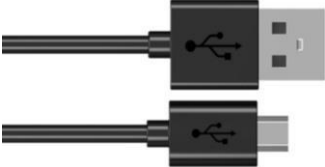
Example

This chapter is the Start Point in the journey to build and explore Raspberry Pi Pico electronic projects. We will start with simple “Blink” project.

Project 1.1 Blink

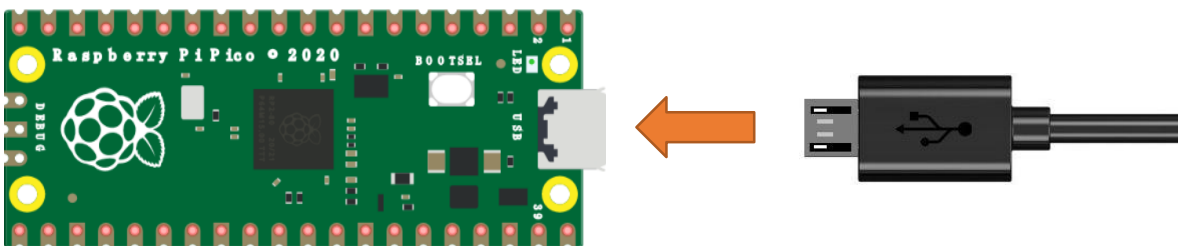
In this project, we will use Raspberry Pi Pico to control blinking a common LED.

Component List

| | |
|--|--|
| Raspberry Pi Pico x1 | USB cable x1 |
|  |  |

Power

In this tutorial, we connect Raspberry Pi Pico and computer with a USB cable.

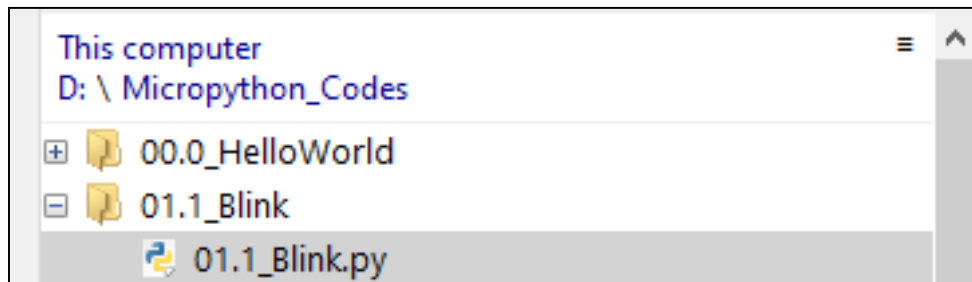


Code

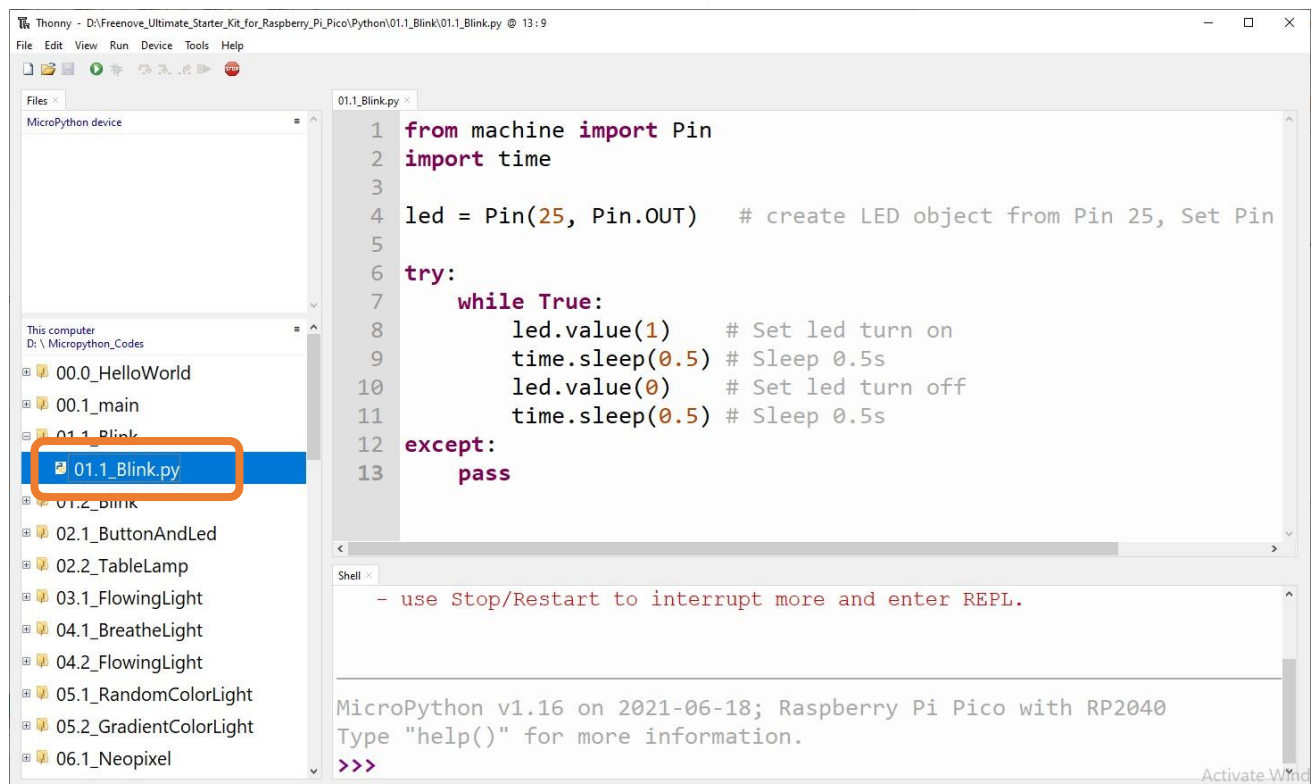
Codes used in this tutorial are saved in "[Cokoino_Ultimate_Starter_Kit_for_Raspberry_Pi_Pico/Python_Codes](#)". You can move the codes to any location. For example, we save the codes in Disk(D) with the path of "[D:/Micropython_Codes](#)".

01.1_Blink

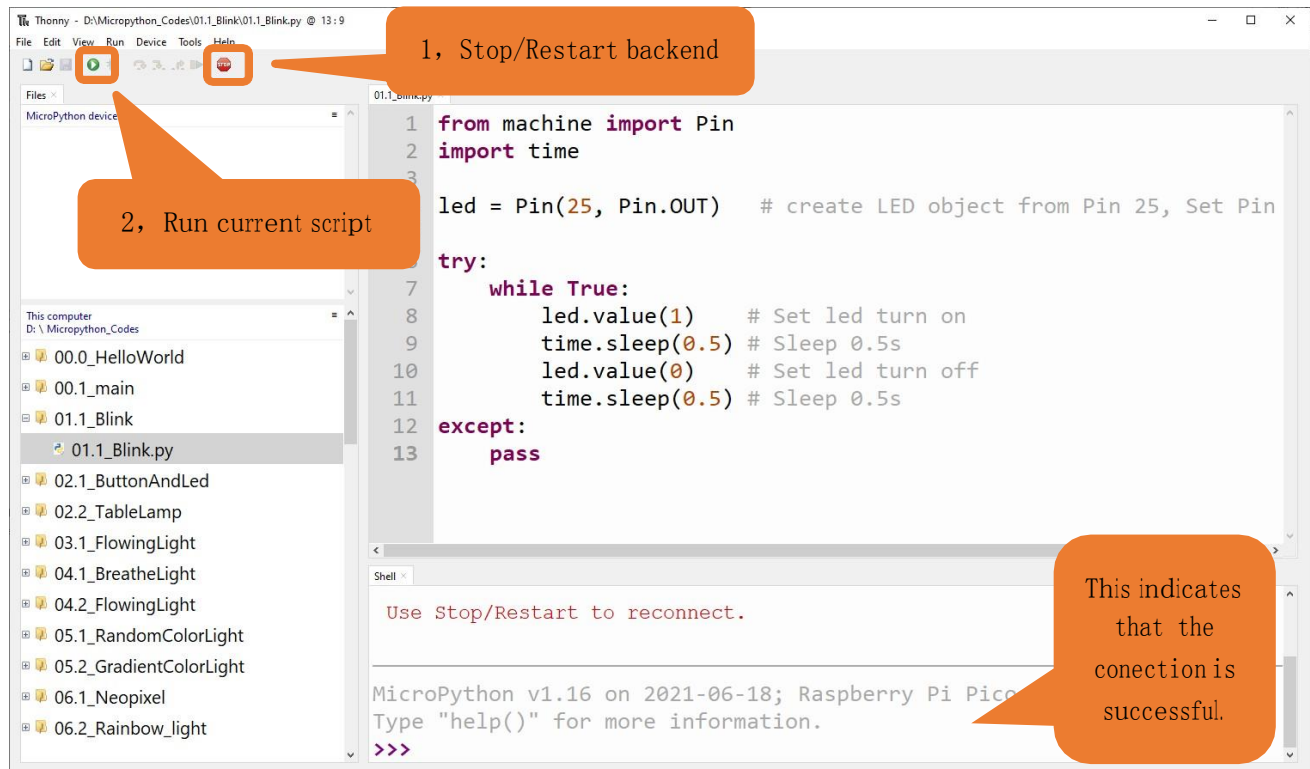
Open "Thonny", click "This computer" "D:" "Micropython_Codes".



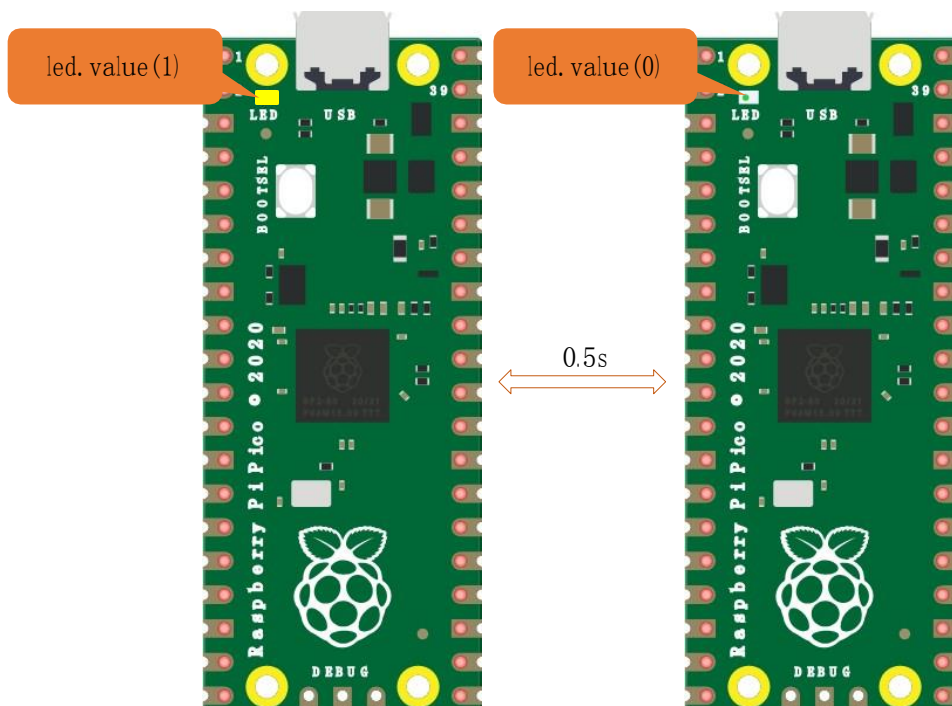
Expand folder "01.1_Blink" and double click "01.1_Blink.py" to open it. As shown in the illustration below.



Make sure Raspberry Pi Pico has been connected with the computer. Click "Stop/Restart backend", and then wait to see what interface will show up.

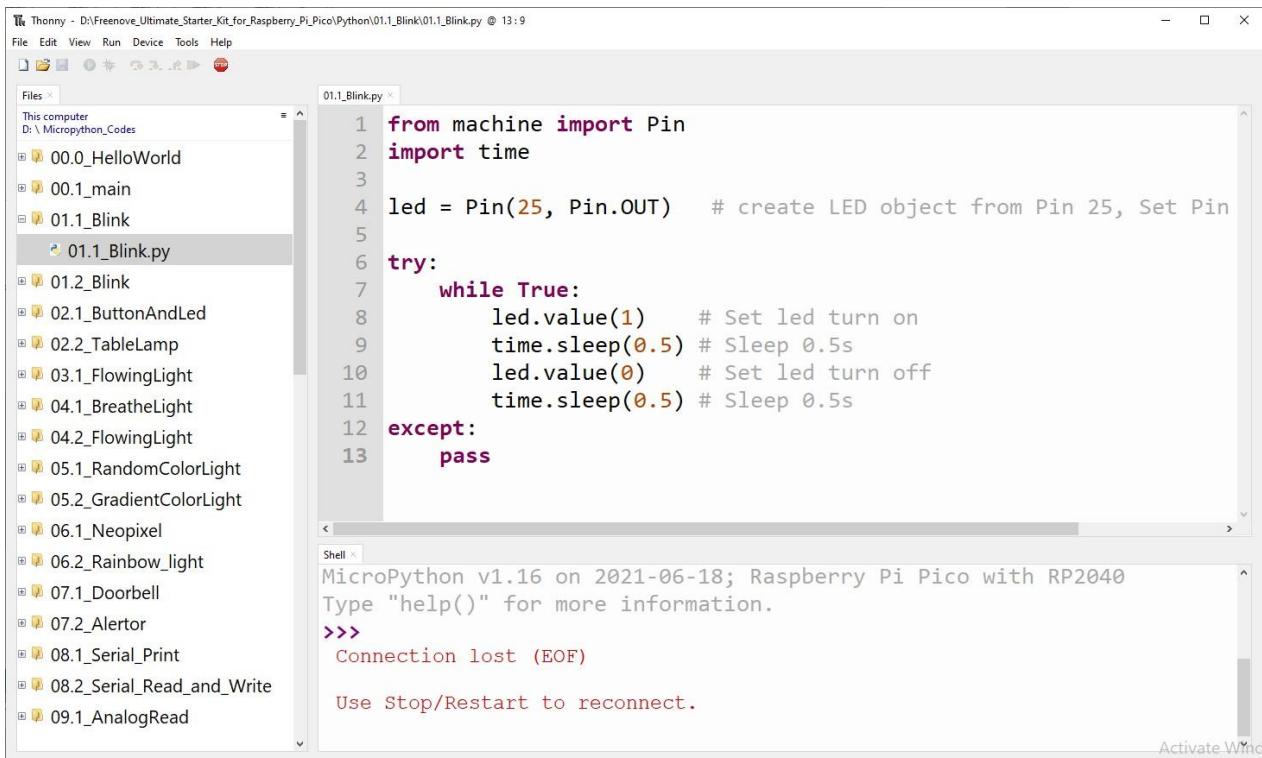


Click "Run current script" shown in the box above, the code starts to be executed and the LED in the circuit starts to blink. Press Ctrl+C or click "Stop/Restart backend" to exit the program.



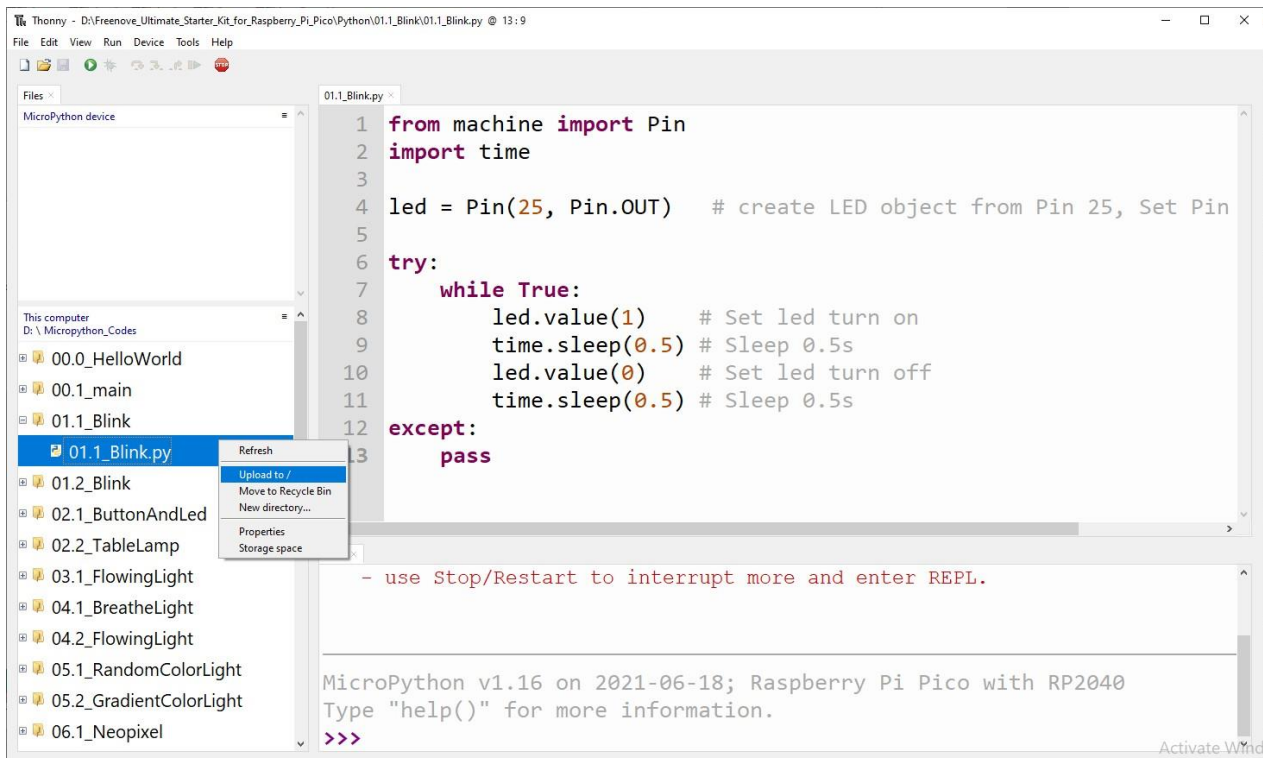
Note:

This is the code running online. If you disconnect USB cable and repower Raspberry Pi Pico, LED stops blinking and the following messages will display in Thonny.

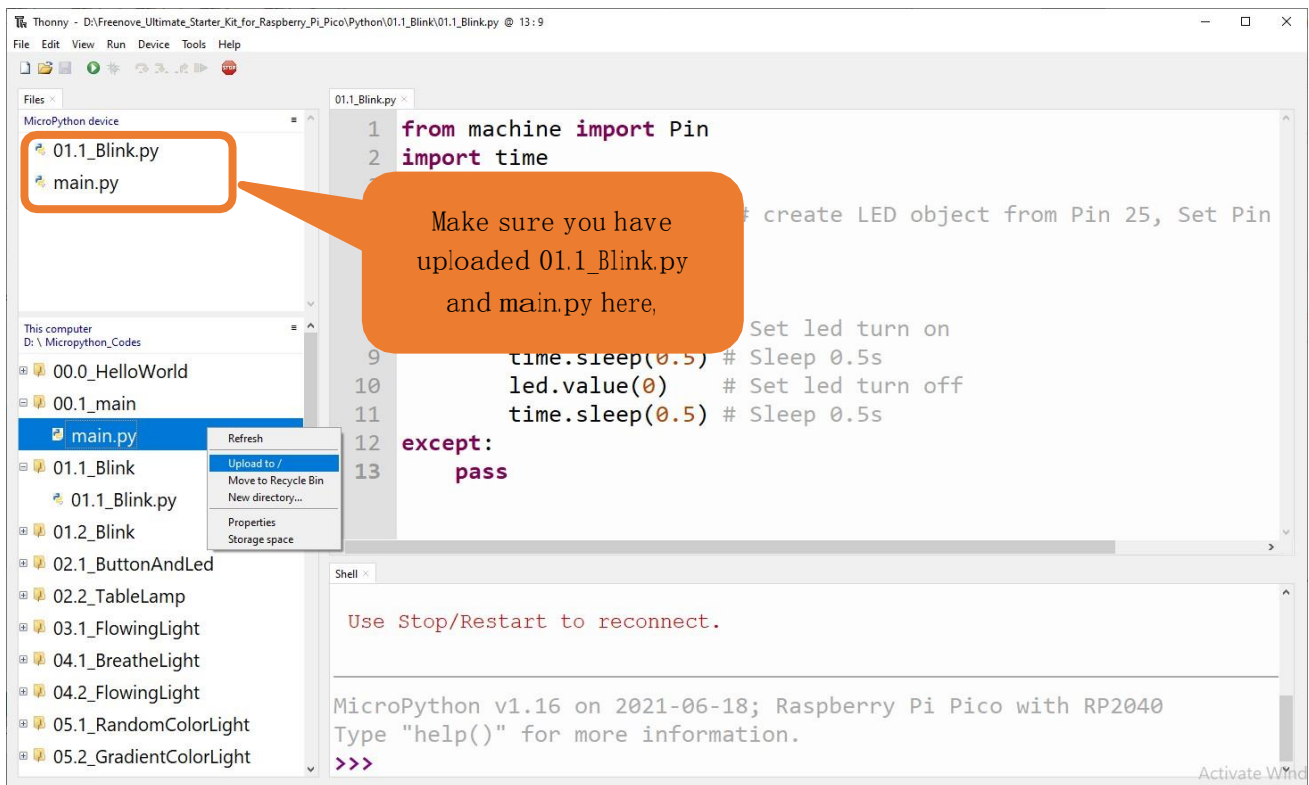


Uploading code to Raspberry Pi Pico

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.



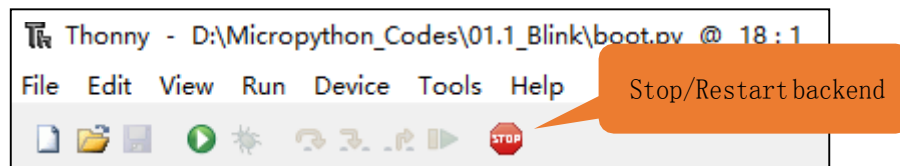
Upload main.py in the same way.



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

Note:

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



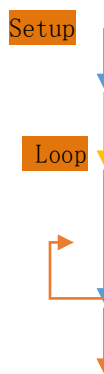
The following is the program code:

```

1  import time
2  from machine import Pin
3
4  led=Pin(25, Pin.OUT) #create LED object from Pin 25, Set Pin 25 to output
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

```

Each time a new file is opened, the program will be executed from top to bottom. When encountering a loop construction, it will execute the loop statement according to the loop condition.



```

1  import time
2  from machine import Pin
3
4  led=Pin(25, Pin.OUT) #create LED object from Pin 25, Set Pin 25 to output
5
6  try:
7      while True:
8          ...
9          ...
10         ...
12 except:
13     pass

```

Print() function is used to print data to Terminal. It can be executed in Terminal directly or be written in a Python file and executed by running the file.

```
print("Hello world!")
```

Each time when using the functions of Raspberry Pi Pico, you need to import modules corresponding to those functions: Import time module and Pin module of machine module.

```

1  import time
2  from machine import Pin

```

Configure GP25 of Raspberry Pi Pico to output mode and assign it to an object named "led".

```
4  led=Pin(25, Pin.OUT) #create LED object from Pin 25, Set Pin 25 to output
```

It means that from now on, LED representing GP25 is in output mode.

Set the value of LED to 1 and GP25 will output high level.

```
8  led.value(1) #Set led turn on
```

Set the value of LED to 0 and GP25 will output low level.

```
10 led.value(0) #Set led turn on
```

Execute codes in a while loop.

```

7      while True:
...          ...

```

Put statements that may cause an error in "try" block and the executing statements when an error occurs in "except" block. In general, when the program executes statements, it will execute those in "try" block. However, when an error occurs to Raspberry Pi Pico due to some interference or other reasons, it will execute statements in "except" block.

"Pass" is an empty statement. When it is executed, nothing happens. It is useful as a placeholder to make the structure of a program look better.

```
6  try:
...      ...
12  except:
13      pass
```

The single-line comment of MicroPython starts with a "#" and continues to the end of the line. Comments help us to understand code. When programs are running, Thonny will skip comments.

```
8  #Set led turn on
```

MicroPython uses indentations to distinguish different blocks of code instead of braces. The number of indentations is changeable, but it must be consistent throughout one block. If the indentation of the same code block is inconsistent, it will cause errors when the program runs.

```
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
```

How to import python files

Whether to import the built-in python module or to import that written by users, the command "import" is needed. If you import the module directly you should indicate the module to which the function or attribute

```
import random

num = random.randint(1, 100)
print(num)
```

belongs when using the function or attribute (constant, variable) in the module. The format should be: <module name>.<function or attribute>, otherwise an error will occur.

If you only want to import a certain function or attribute in the module, use the from...import statement. The format is as follows

```
from random import randint
num = randint(1, 100)
print(num)
```

When using "from...import" statement to import function, to avoid conflicts and for easy understanding, you can use "as" statement to rename the imported function, as follows

```
from random import randint as rand
num = rand(1, 100)
print(num)
```

Reference

Class machine

Before each use of the **machine** module, please add the statement **import machine** to the top of python file.

machine.freq(freq_val): When "freq_val" is not specified, it is to return to the current CPU frequency; Otherwise, it is to set the current CPU frequency.

freq_val: 125000000Hz(125MHz).

machine.reset(): A reset function. When it is called, the program will be reset.

machine.unique_id(): Obtains MAC address of the device.

machine.idle(): Turns off any temporarily unused functions on the chip and its clock, which is useful to reduce power consumption at any time during short or long periods.

machine.disable_irq(): Disables interrupt requests and return the previous IRQ state. The **disable_irq ()** function and **enable_irq ()** function need to be used together; Otherwise the machine will crash and restart.

machine.enable_irq(state): To re-enable interrupt requests. The parameter state should be the value that was returned from the most recent call to the **disable_irq()** function.

machine.time_pulse_us(pin, pulse_level, timeout_us=1000000):

Tests the duration of the external pulse level on the given pin and returns the duration of the external pulse level in microseconds. When pulse level = 1, it tests the high level duration; When pulse level = 0, it tests the low level duration.

If the setting level is not consistent with the current pulse level, it will wait until they are consistent, and then start timing. If the set level is consistent with the current pulse level, it will start timing immediately.

When the pin level is opposite to the set level, it will wait for timeout and return "-2". When the pin level and the set level is the same, it will also wait timeout but return "-1". **timeout_us** is the duration of timeout.

For more information about class and function, please refer to:

<https://docs.micropython.org/en/latest/rp2/quickref.html>

Class time

Before each use of the **time** module, please add the statement **"import time"** to the top of python file

time.sleep(sec): Sleeps for the given number of seconds.

sec: This argument should be either an int or a float.

time.sleep_ms(ms): Sleeps for the given number of milliseconds, ms should be an int.

time.sleep_us(us): Sleeps for the given number of microseconds, us should be an int.

time.time(): Obtains the timestamp of CPU, with second as its unit.

time.ticks_ms(): Returns the incrementing millisecond counter value, which recounts after some values.

time.ticks_us(): Returns microsecond.

time.ticks_cpu(): Similar to ticks_ms() and ticks_us(), but it is more accurate(return clock of CPU).

time.ticks_add(ticks, delta): Gets the timestamp after the offset.

ticks: ticks_ms()、 ticks_us()、 ticks_cpu().

delta: Delta can be an arbitrary integer number or numeric expression.

time.ticks_diff(old_t, new_t): Calculates the interval between two timestamps, such as ticks_ms(), ticks_us() or ticks_cpu().

old_t: Starting time.

new_t: Ending time.

Class Pin(id, mode, pull, value)

Before each use of the Pin module, please add the statement "from machine import Pin" to the top of python file.

id: Arbitrary pin number.

mode: Mode of pins.

Pin.IN: Input Mode.

Pin.OUT: Output Mode.

Pin.OPEN_DRAIN: Open-drain Mode.

Pull: Whether to enable the internal pull up and down mode.

None: No pull up or pull down resistors.

Pin.PULL_UP: Pull-up Mode, outputting high level by default.

Pin.PULL_DOWN: Pull-down Mode, outputting low level by default.

Value: State of the pin level, 0/1.

Pin.init(mode, pull): Initialize pins.

Pin.value([value]): Obtain or set state of the pin level, return 0 or 1 according to the logic level of pins.

Without parameter, it reads input level. With parameter given, it is to set output level.

value: It can be either True/False or 1/0.

Pin.irq(trigger, handler): Configures an interrupt handler to be called when the pin level meets a condition.

trigger:

Pin.IRQ_FALLING: interrupt on falling edge.

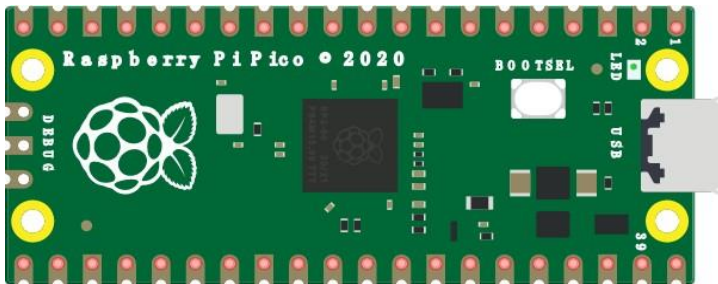

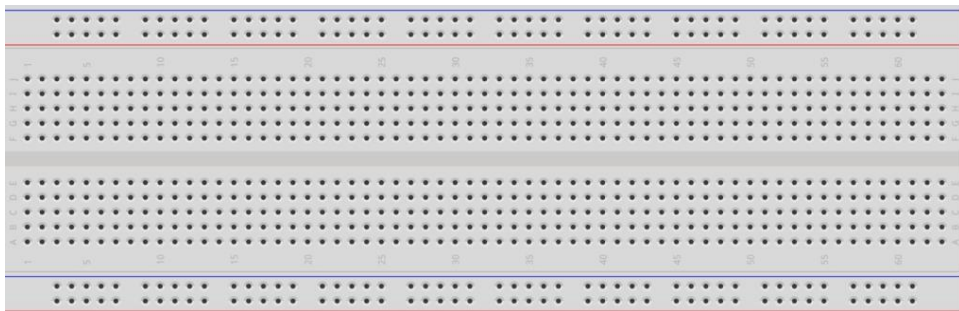



Pin.IRQ_RISING: interrupt on rising edge.

Handler: callback function.

Project 1.2 Blink

In this project, we will use Raspberry Pi Pico to control blinking a common LED.

Component List

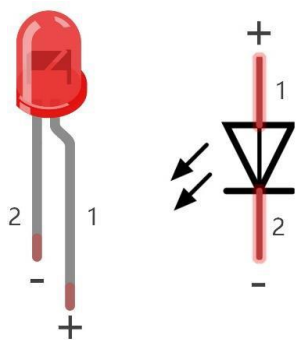
| | | | |
|---|---|--|--|
| Raspberry Pi Pico x1 | | USB Cable x1 | |
|  | |  | |
| Breadboard x1 | | | |
|  | | | |
| LED x1 | Resistor 220Ω x1 | Jumper | |
|  |  |  | |

Component knowledge

LED

An LED is a type of diode. All diodes only work if current is flowing in the correct direction and have two Poles. An LED will only work (light up) if the longer pin (+) of LED is connected to the positive output from a power source and the shorter pin is connected to the negative (-). Negative output is also referred to as Ground (GND). This type of component is known as "Polar" (think One-Way Street).

All common 2 lead diodes are the same in this respect. Diodes work only if the voltage of its positive electrode is higher than its negative electrode and there is a narrow range of operating voltage for most all common diodes of 1.9 and 3.4V. If you use much more than 3.3V the LED will be damaged and burn out.



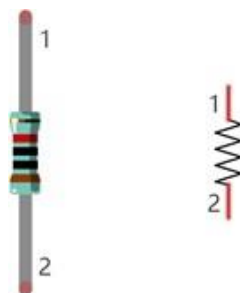
| LED | Voltage | Maximum current | Recommended current |
|--|----------|-----------------|---------------------|
| Red | 1.9-2.2V | 20mA | 10mA |
| Green | 2.9-3.4V | 10mA | 5mA |
| Blue | 2.9-3.4V | 10mA | 5mA |
| Volt ampere characteristics conform to diode | | | |

Note: LEDs cannot be directly connected to a power supply, which usually ends in a damaged component. A resistor with a specified resistance value must be connected in series to the LED you plan to use.

Resistor

Resistors use Ohms (Ω) as the unit of measurement of their resistance (R). $1\text{M}\Omega=1000\text{k}\Omega$, $1\text{k}\Omega=1000\Omega$.

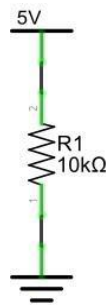
A resistor is a passive electrical component that limits or regulates the flow of current in an electronic circuit. On the left, we see a physical representation of a resistor, and the right is the symbol used to represent the presence of a resistor in a circuit diagram or schematic.



The bands of color on a resistor is a shorthand code used to identify its resistance value. For more details of resistor color codes, please refer to the appendix of this tutorial.

With a fixed voltage, there will be less current output with greater resistance added to the circuit. The relationship between Current, Voltage and Resistance can be expressed by this formula: $I=V/R$ known as Ohm's Law where I = Current, V = Voltage and R = Resistance. Knowing the values of any two of these allows you to solve the value of the third.

In the following diagram, the current through R1 is: $I = U/R = 5V/10k\Omega = 0.0005A = 0.5mA$.

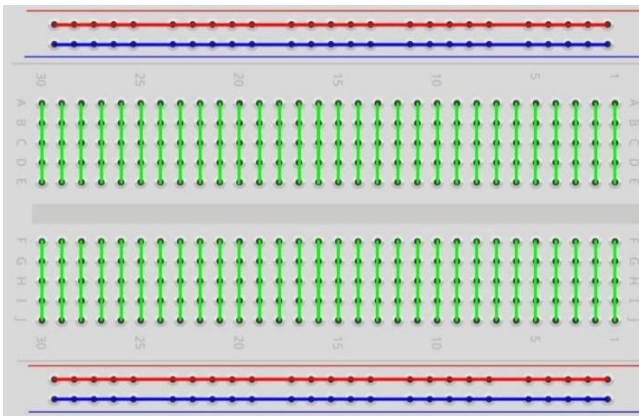


WARNING: Never connect the two poles of a power supply with anything of low resistance value (i.e. a metal object or bare wire) this is a Short and results in high current that may damage the power supply and electronic components.

Note: Unlike LEDs and Diodes, Resistors have no poles and are non-polar (it does not matter which direction you insert them into a circuit, it will work the same)

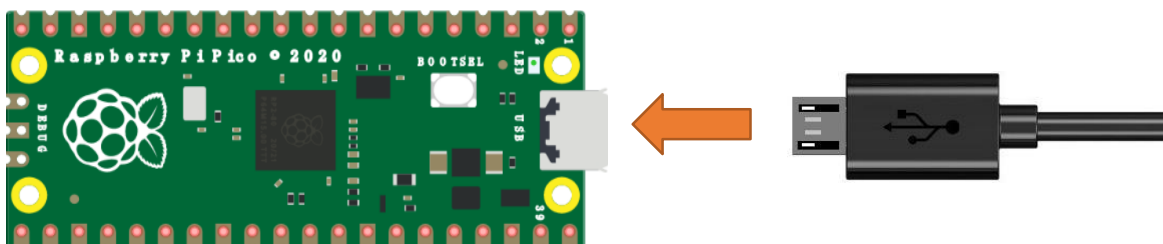
Breadboard

Here we have a small breadboard as an example of how the rows of holes (sockets) are electrically attached. The left picture shows the way to connect pins. The right picture shows the practical internal structure.



Power

In this tutorial, we connect Raspberry Pi Pico and computer with a USB cable.

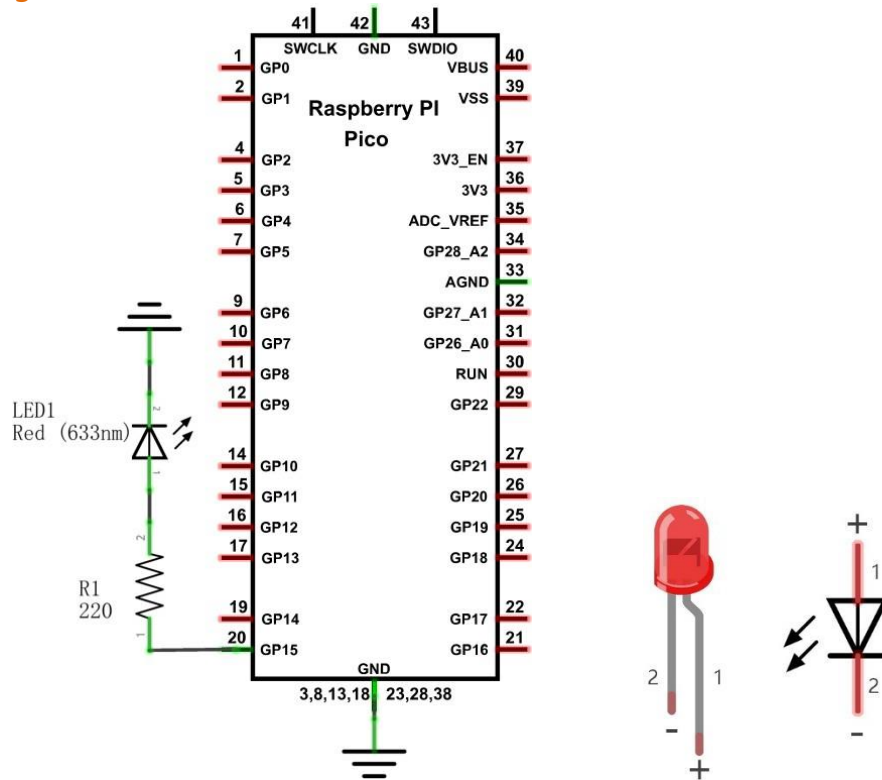


Circuit

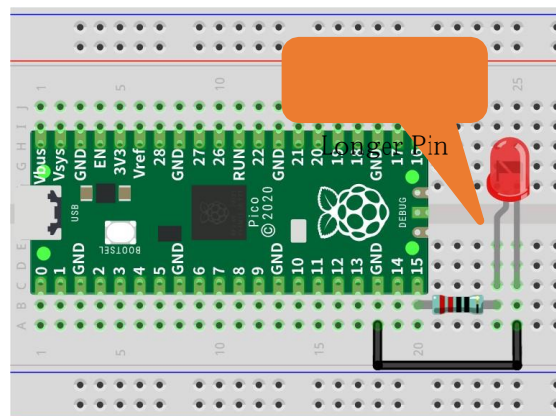
First, disconnect all power from the Raspberry Pi Pico. Then build the circuit according to the circuit and hardware diagrams. After the circuit is built and verified correct, connect the PC to Raspberry Pi Pico.

CAUTION: Avoid any possible short circuits (especially connecting 3.3V and GND)! **WARNING:** A short circuit can cause high current in your circuit, create excessive component heat and cause permanent damage to your hardware!

Schematic diagram



Hardware connection

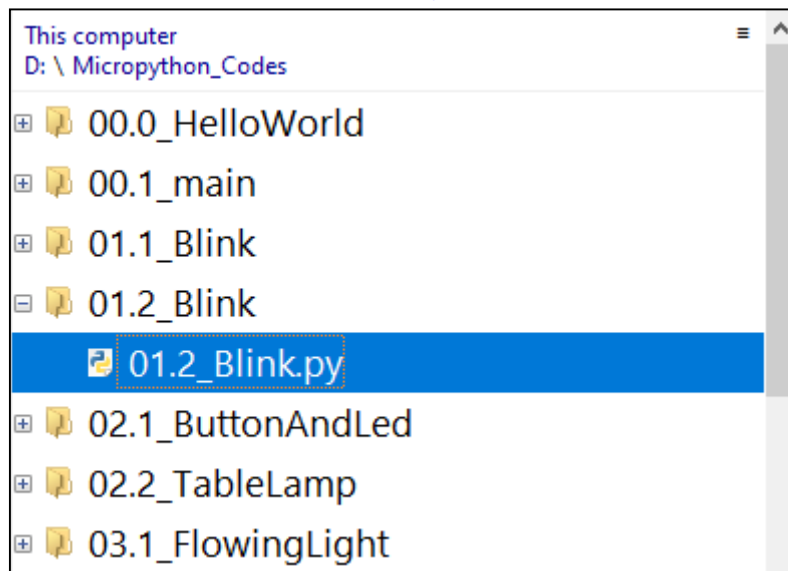


Code

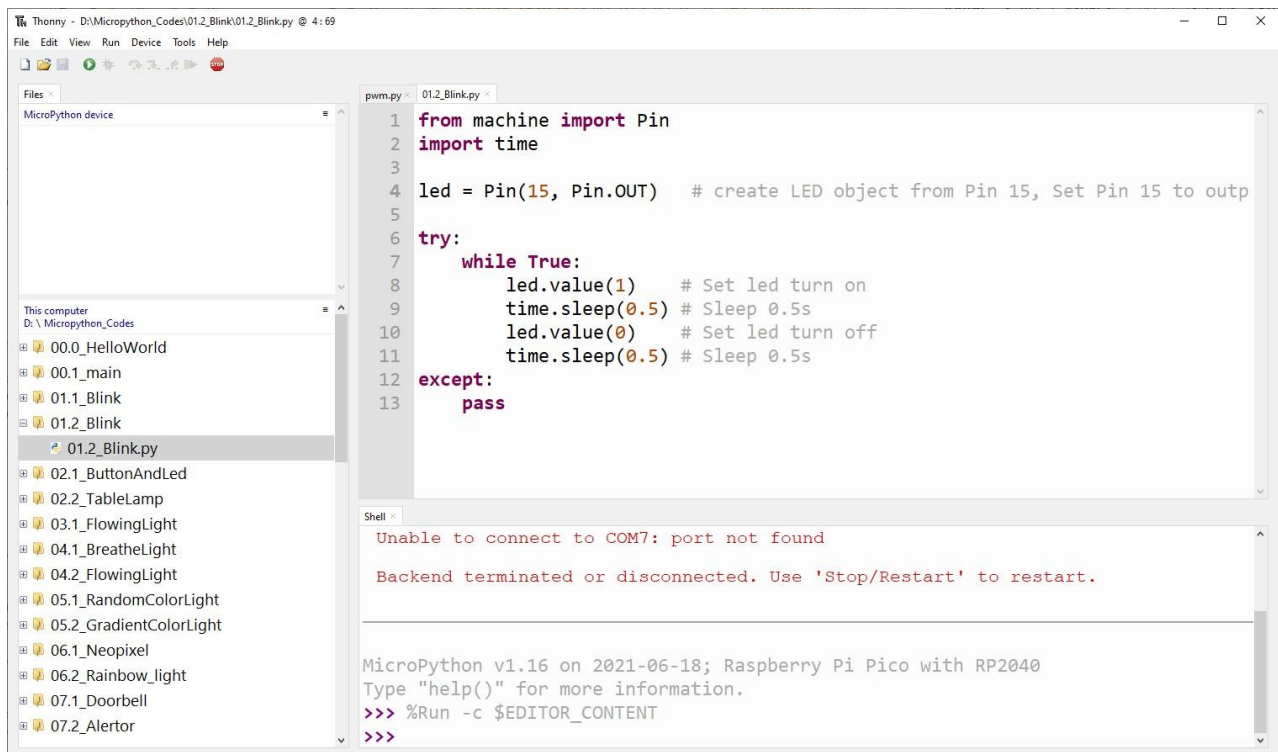
Codes used in this tutorial are saved in [‘Raspberry_Pi_Pico Kit Tutorial\Lesson2-Python\Python_Codes’](#). You can move the codes to any location. For example, we save the codes in Disk(D) with the path of [‘D:/Micropython_Codes’](#).

01.2_Blink

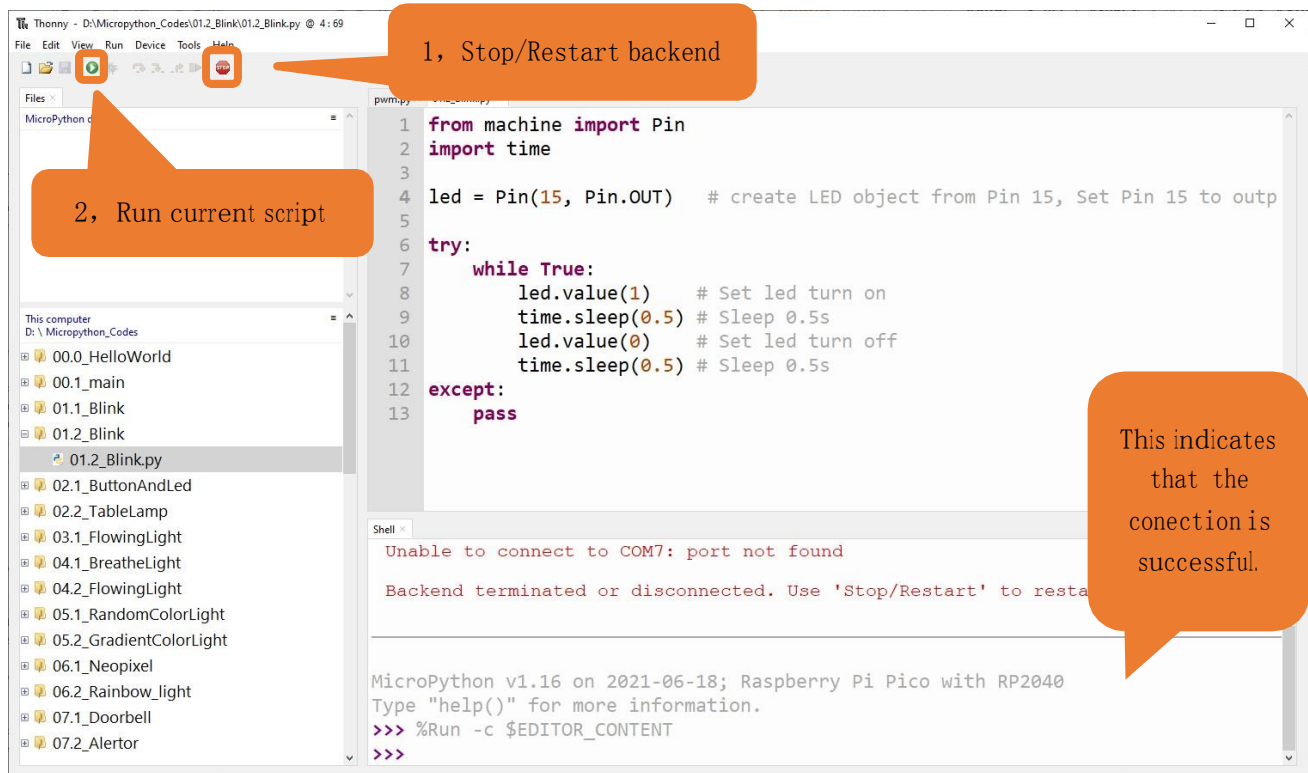
Open [‘Thonny’](#), click [‘This computer’](#) → [‘D:’](#) → [‘Micropython_Codes’](#).



Expand folder [‘01.2_Blink’](#) and double click [‘01.2_Blink.py’](#) to open it. As shown in the illustration below.



Make sure Raspberry Pi Pico has been connected with the computer. Click "Stop/Restart backend", and then wait to see what interface will show up.

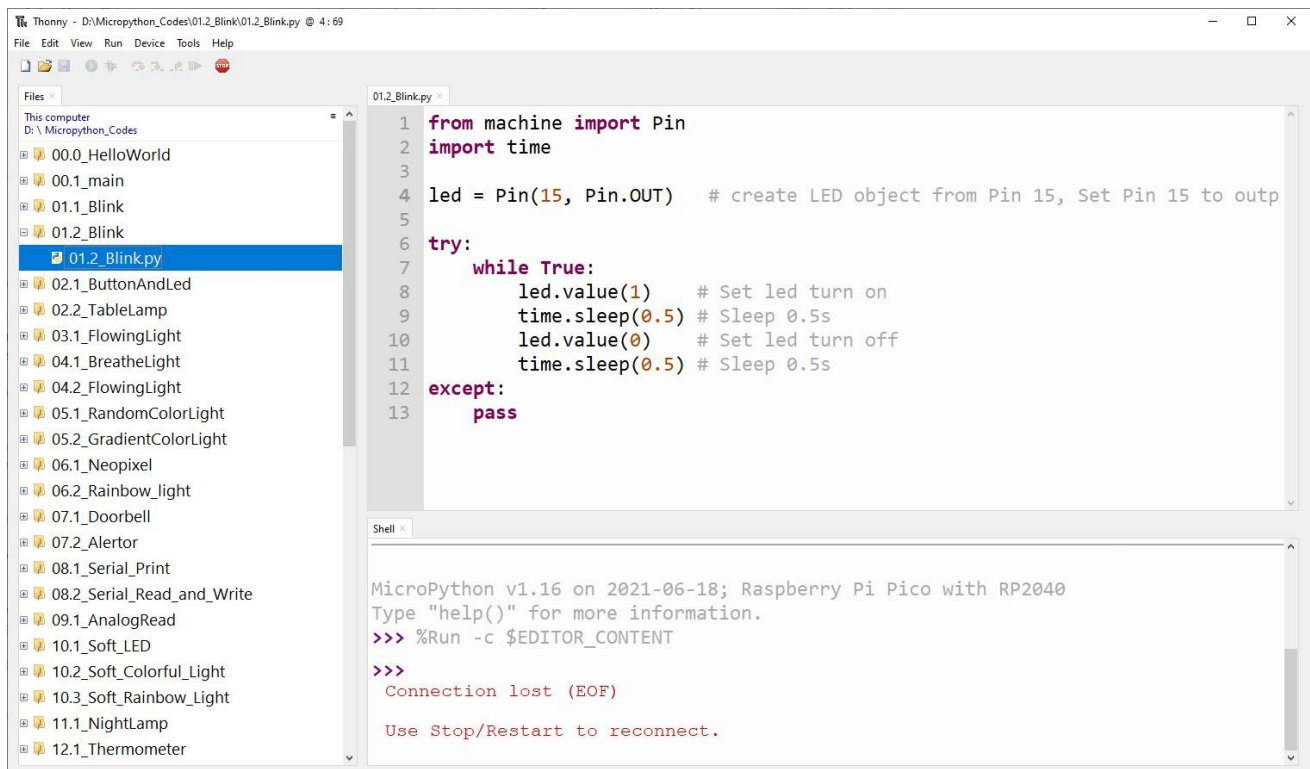


Click "Run current script" shown in the box above, the code starts to be executed and the LED in the circuit starts to blink. Press Ctrl+C or click "Stop/Restart backend" to exit the program.



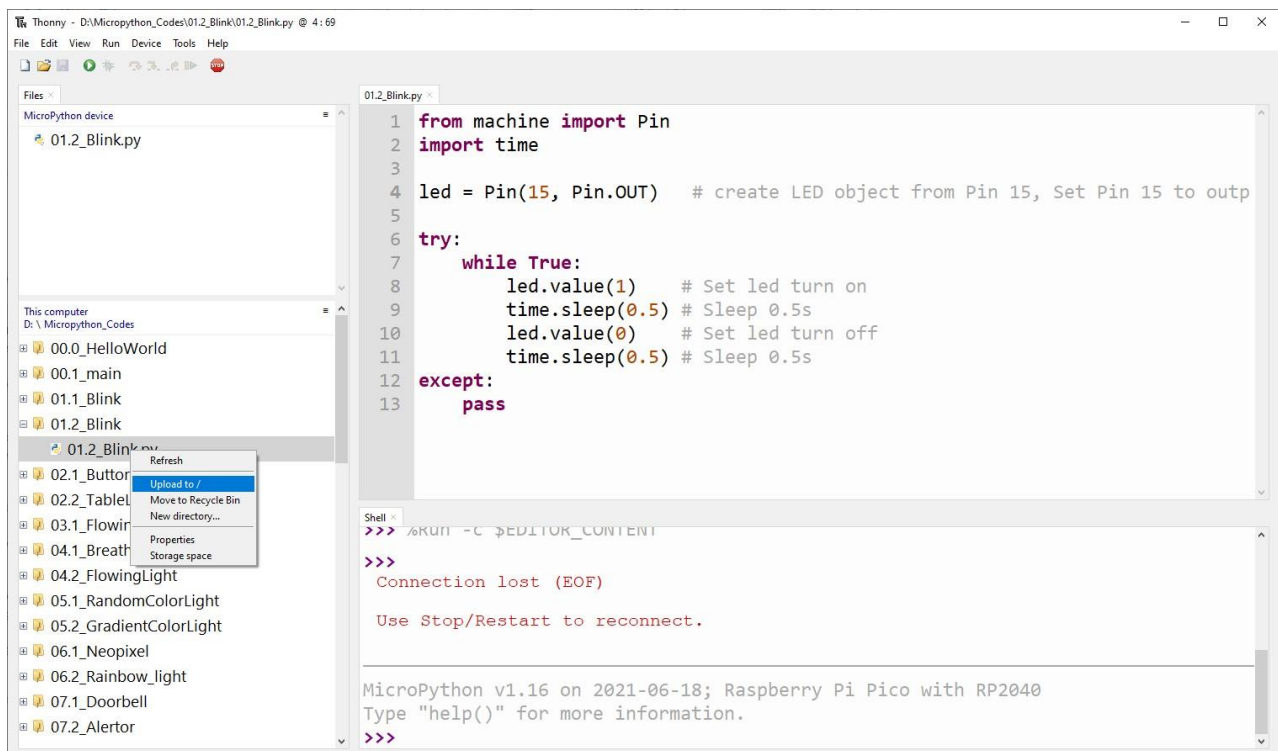
Note:

This is the code running online. If you disconnect USB cable and repower Raspberry Pi Pico, LED stops blinking and the following messages will be displayed in Thonny.

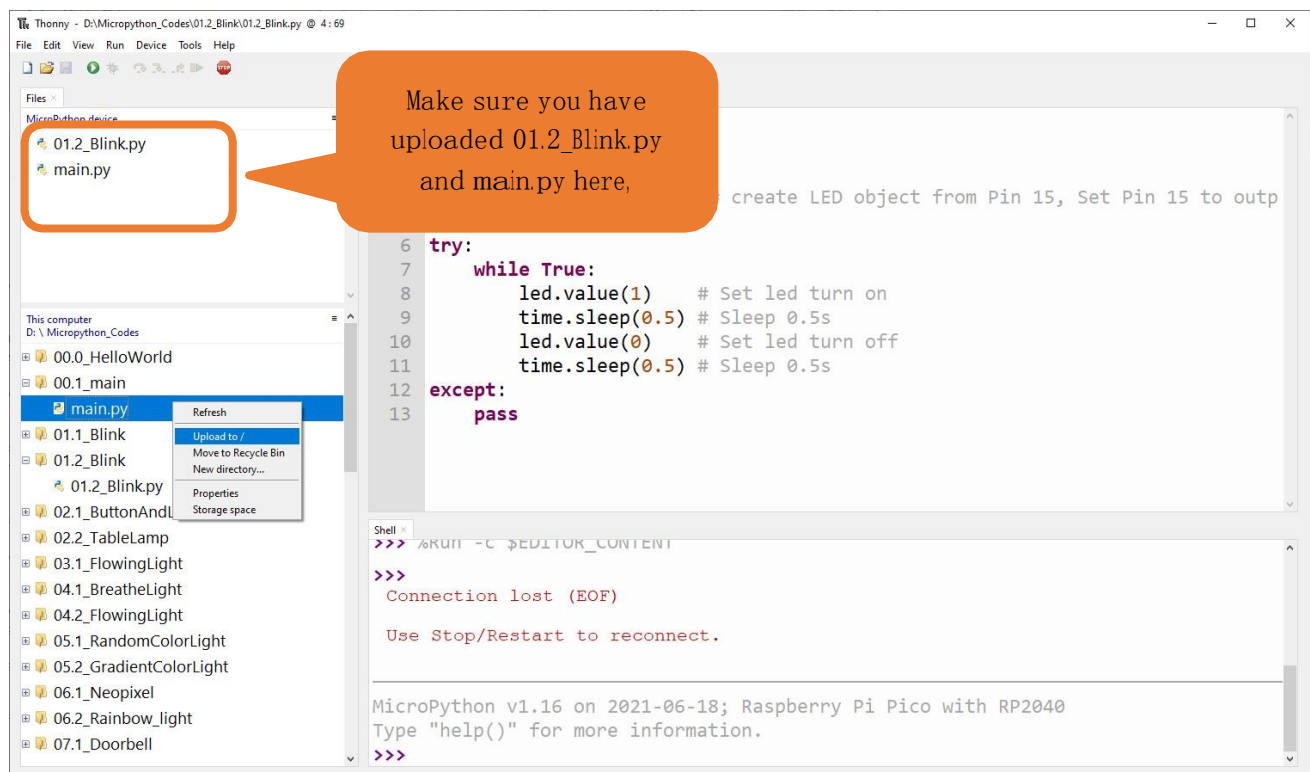


Uploading code to Raspberry Pi Pico

As shown in the following illustration, right-click the file "01.2_Blink.py" and select "Upload to /" to upload code to Raspberry Pi Pico.



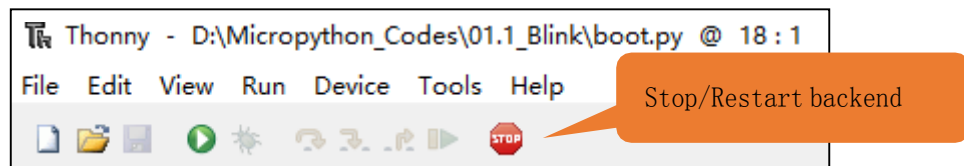
Upload "main.py" in the same way.



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

Note:

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



What's Next?

THANK YOU for participating in this learning experience.

We have reached the end of this Tutorial. If you find errors, omissions or you have suggestions and/or questions about the Tutorial or component contents of this Kit, please feel free to contact us: 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 website: <http://www.cokoino.com/>

We will continue to launch fun, cost-effective, innovative and exciting products.

Thank you again for choosing Cokoino products.