

Welcome

Thank you for choosing BLONWINER products!

Getting Started

When reading this, you should have downloaded the ZIP file for this product.

Unzip it and you will get a folder containing tutorials and related files. Please start with this PDF tutorial.

! Unzip the ZIP file instead of opening the file in the ZIP file directly.

! Do not move, delete or rename files in the folder just unzipped.

Get Support

Encounter problems? Don't worry! Contact us.

When there are packaging damage, quality problems, questions encountering in use, etc., just send us an email. We will reply to you within one working day and provide a solution.

blonwiner@outlook.com

Safety and Precautions

Please follow the following safety precautions when using or storing this product:

- Keep this product out of the reach of children under 14 years old.
- This product should be used only when there is adult supervision present as young children lack necessary judgment regarding safety and the consequences of product misuse.
- This product contains small parts and parts, which are sharp. This product contains electrically conductive parts. Use caution with electrically conductive parts near or around power supplies, batteries and powered (live) circuits.
- When the product is turned ON, activated or tested, some parts will move or rotate. To avoid injuries to hands and fingers, keep them away from any moving parts!
- It is possible that an improperly connected or shorted circuit may cause overheating. Should this happen, immediately disconnect the power supply or remove the batteries and do not touch anything until it cools down! When everything is safe and cool, review the product tutorial to identify the cause.
- Only operate the product in accordance with the instructions and guidelines of this tutorial, otherwise parts may be damaged or you could be injured.
- Store the product in a cool dry place and avoid exposing the product to direct sunlight.
- After use, always turn the power OFF and remove or unplug the batteries before storing.

Any concerns? ✉ blonwiner@outlook.com

About BLONWINER

BLONWINER provides open source electronic products and services worldwide.

BLONWINER is committed to assist customers in their education of robotics, programming and electronic circuits so that they may transform their creative ideas into prototypes and new and innovative products. To this end, our services include but are not limited to:

- Educational and Entertaining Project Kits for Robots, Smart Cars and Drones
- Educational Kits to Learn Robotic Software Systems for Arduino, Raspberry Pi, ESO32 and micro: bit
- Electronic Component Assortments, Electronic Modules and Specialized Tools
- Product Development and Customization Services

You can find more about BLONWINER and get our latest news and updates through our website:

<http://www.123mygift.com>

Copyright

All the files, materials and instructional guides provided are released under [Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License](#). A copy of this license can be found in the folder containing the Tutorial and software files associated with this product.



This means you can use these resource in your own derived works, in part or completely, but **NOT for the intent or purpose of commercial use.**

BLONWINER brand and logo are copyright of BLONWINER Creative Technology Co., Ltd. and cannot be used without written permission.

Other registered trademarks and their owners appearing in this document:

Arduino® is a trademark of Arduino LLC (<https://www.arduino.cc/>).

Raspberry Pi® is a trademark of Raspberry Pi Foundation (<https://www.raspberrypi.org/>).

Raspberry Pi Pico® is a trademark of Raspberry Pi Foundation (<https://www.raspberrypi.org/>).

micro:bit® is a trademark of Micro:bit Educational Foundation (<https://www.microbit.org/>).

ESPRESSIF® and ESP32® are trademarks of ESPRESSIF Systems (Shanghai) Co, Ltd (<https://www.espressif.com/>).

Any concerns? ✉ blonwiner@outlook.com

Contents

Welcome	1
Contents.....	1
Preface.....	2
Raspberry Pi Pico.....	3
Raspberry Pi Pico W.....	6
Chapter 0 Getting Ready (Important)	9
0.1 Installing Thonny (Important)	9
0.2 Basic Configuration of Thonny	14
0.3 Burning Micropython Firmware (Important).....	16
0.4 Thonny Connected to Raspberry Pi Pico.....	19
0.5 Testing codes (Important)	23
0.6 Thonny Common Operation.....	34
0.7 Paste the Sticker on the Breadboard.....	39
Chapter 1 LED (Important).....	40
Project 1.1 Blink.....	40
Project 1.2 Blink.....	49
What's Next?.....	54



Preface

Raspberry Pi Pico is a tiny, fast, and versatile board built using RP2040, a brand new microcontroller chip designed by Raspberry Pi in the UK. Getting started is as easy as dragging and dropping a file, it is suitable for beginners and makers to develop, design and research.

Raspberry Pi Pico is programmable in C/C++ and MicroPython. In this tutorial, we use MicroPython to develop, which is a very easy to learn language with lean and simple code, hence it is very suitable for beginners to learn and for secondary development.

If you haven't downloaded the related material for Raspberry Pi Pico tutorial, you can download it from this link:

https://github.com/BLONWINER/BLONWINER_Breakout_Board_for_Raspberry_Pi_Pico/archive/refs/heads/master.zip

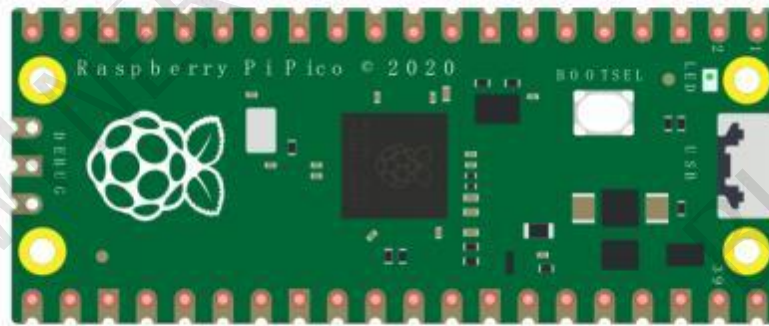
After completing the projects in this tutorial, you can also combine the components in different projects to make your own smart homes, smart car, robot, etc., bringing your imagination and creativity to life with Raspberry Pi Pico.

If you have any problems or difficulties using this product, please contact us for quick and free technical support: blonwiner@outlook.com

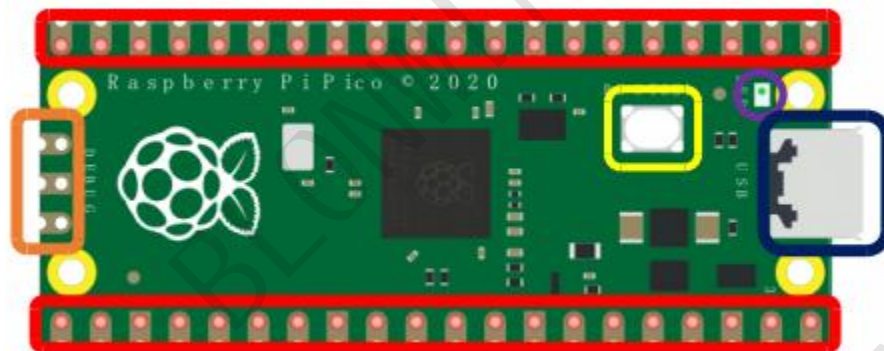
Raspberry Pi Pico

Raspberry Pi Pico applies to all chapters except Wireless in this tutorial.

Before learning Pico, we need to know about it. Below is an imitated diagram of Pico, which looks very similar to the actual Pico.



The hardware interfaces are distributed as follows:



Frame color



Description

Pins

BOOTSEL button

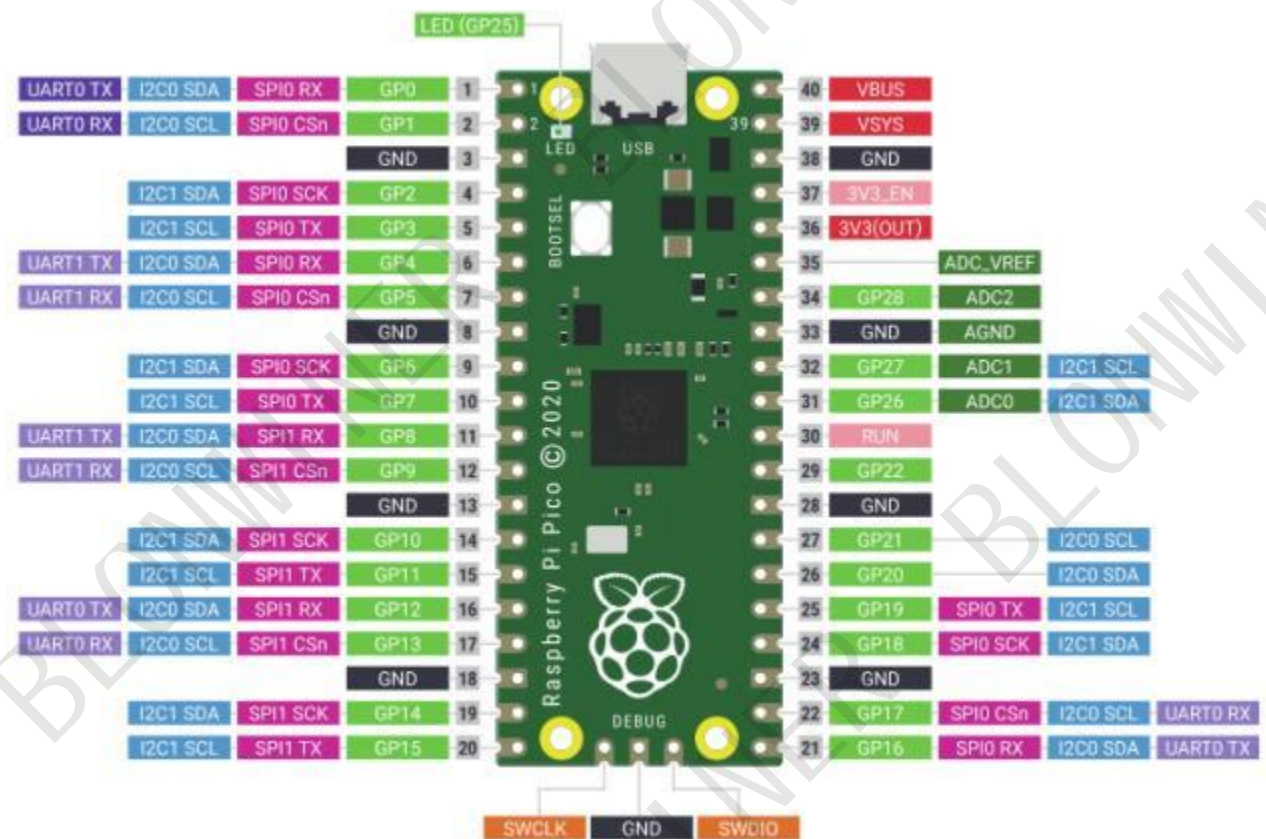
USB port

LED

Debugging



Function definition of pins:



Color



Pins

GND

Color



Pins

Power



GPIO



ADC



UART(default)



UART



SPI



I2C



System Control



Debugging

For details: <https://datasheets.raspberrypi.org/pico/pico-datasheet.pdf>

Any concerns? ✉ blonwiner@outlook.com

UART, I2C, SPI Default Pin

In Arduino IDE, the default pins of serial port are Pin0 and Pin1.

Note: Serial port is virtualized by RP2040. Therefore, when using the serial port, please enable the verification function of DTR. It can work under any baud rate.

UART

Function	Default
UART_BAUDRATE	X
UART_BITS	8
UART_STOP	1
UART_TX	Pin 0
UART_RX	Pin 1

I2C

Function	Default
I2C Frequency	400000
I2C_SDA	Pin 4
I2C_SCL	Pin 5

SPI

Function	Default
SPI_BAUDRATE	1000000
SPI_POLARITY	0
SPI_PHASE	0
SPI_BITS	8
SPI_FIRSTBIT	MSB
SPI_SCK	Pin 18
SPI_MOSI	Pin 19
SPI_MISO	Pin 16
SPI_SS	Pin 17

Raspberry Pi Pico W

Raspberry Pi Pico W applies to all chapters in this tutorial.

Raspberry Pi Pico W adds CYW43439 as the WiFi function on the basis of Raspberry Pi Pico. It is connected to RP2040 chip through SPI interface.



The hardware interfaces are distributed as follows:



Frame color



Description

Pins

BOOTSE button

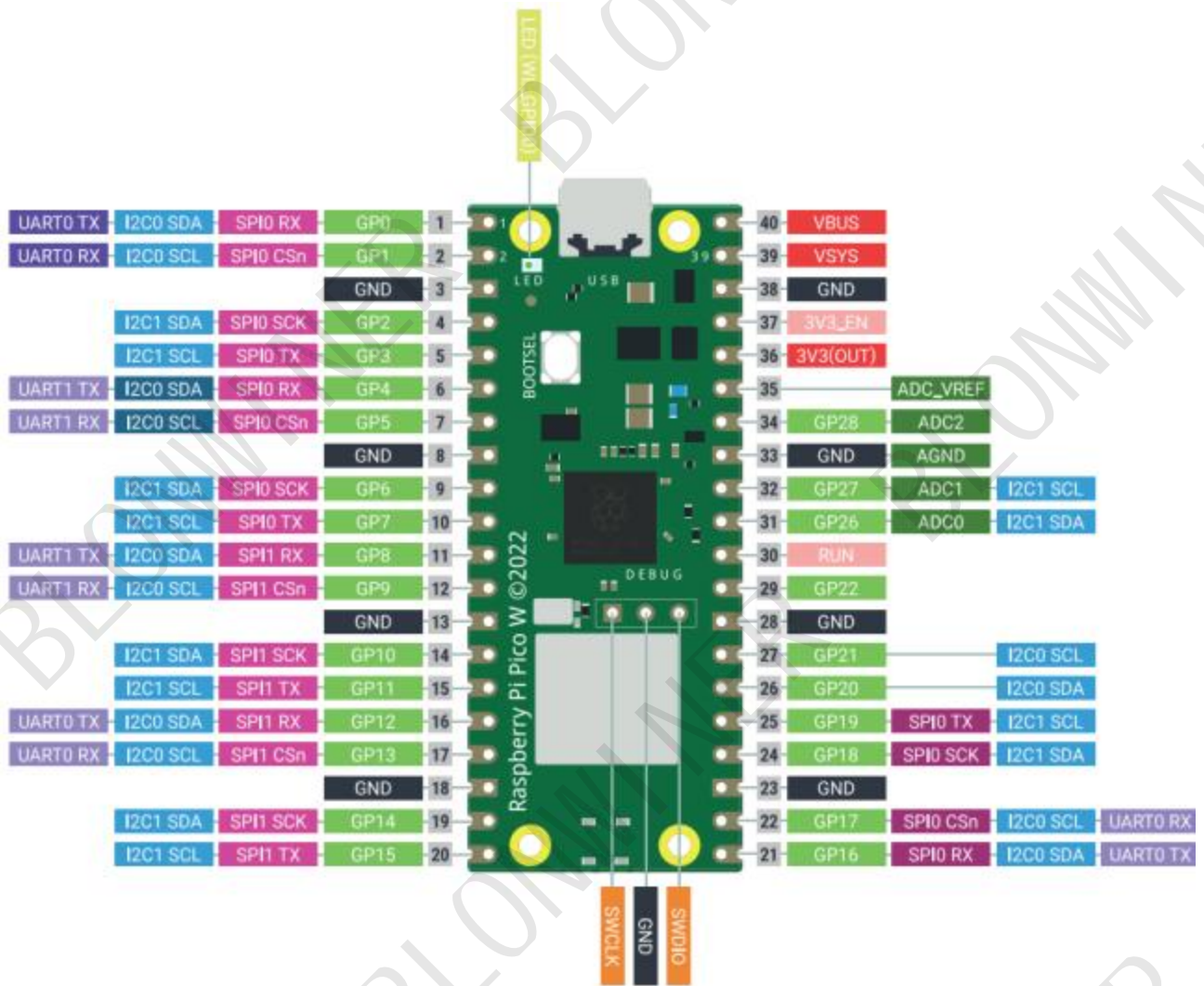
USB port

LED

Debugging

Wireless

Function definition of pins:



Color



Pins

GND

Color



Pins

Power

GPIO



ADC

UART(default)



UART

SPI



I2C

System Control



Debugging

For details: <https://datasheets.raspberrypi.com/picow/pico-w-datasheet.pdf>

Any concerns? [✉ blonwiner@outlook.com](mailto:blonwiner@outlook.com)



UART, I2C, SPI, Wireless Default Pin

In Arduino IDE, the default pins of serial port are Pin0 and Pin1.

Note: Serial port is virtualized by RP2040. Therefore, when using the serial port, please enable the verification function of DTR. It can work under any baud rate.

UART

Function	Default
UART_BAUDRATE	X
UART_BITS	8
UART_STOP	1
UART_TX	Pin 0
UART_RX	Pin 1

I2C

Function	Default
I2C Frequency	400000
I2C_SDA	Pin 4
I2C_SCL	Pin 5

SPI

Function	Default
SPI_BAUDRATE	1000000
SPI_POLARITY	0
SPI_PHASE	0
SPI_BITS	8
SPI_FIRSTBIT	MSB
SPI_SCK	Pin 18
SPI_MOSI	Pin 19
SPI_MISO	Pin 16
SPI_SS	Pin 17

Wireless

Function	Default
WL_ON	GPIO23
WL_D	GPIO24
WL_CLK	GPIO29_ADC
WL_CS	GPIO25

Chapter 0 Getting Ready (Important)

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/v3.2.7/thonny-3.2.7.exe
---------	---

Mac OS	https://github.com/thonny/thonny/releases/download/v3.2.7/thonny-3.2.7.pkg
--------	---

The latest version:

Binary bundle for PC (Thonny+Python):

bash <(wget -O - https://thonny.org/installer-for-linux)

With pip:

pip3 install thonny

Linux

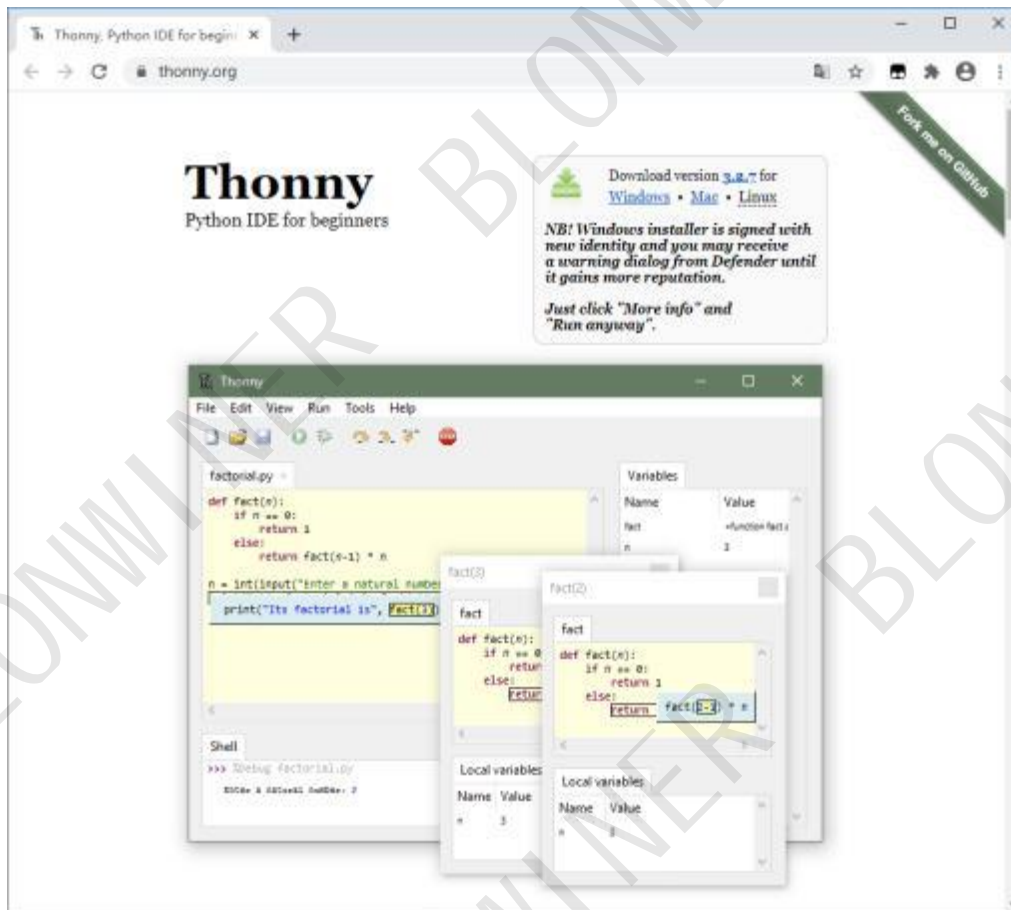
Distro packages (may not be the latest version):

Debian, Rasbian, Ubuntu, Mint and others:

sudo apt install thonny

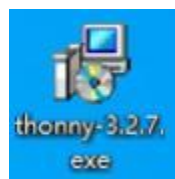
Fedora:

sudo dnf install thonny



Installing on Windows

The icon of Thonny after downloading is as below. Double click "thonny-3.2.7.exe".



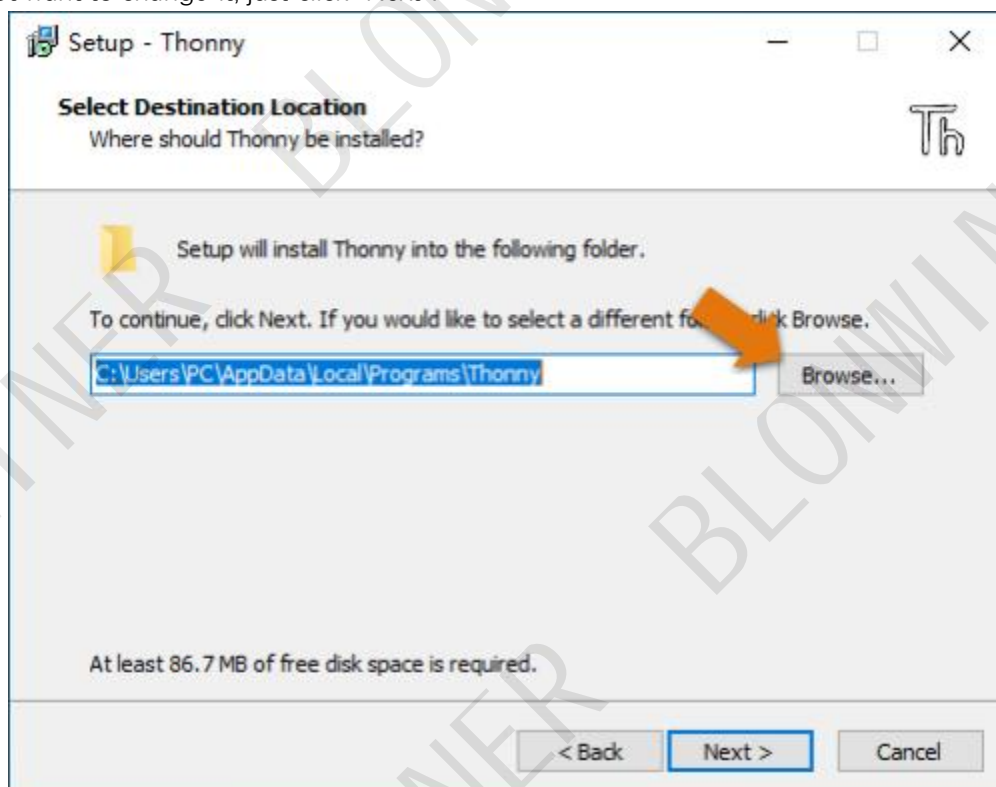
Any concerns? [✉ blonwiner@outlook.com](mailto:blonwiner@outlook.com)

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



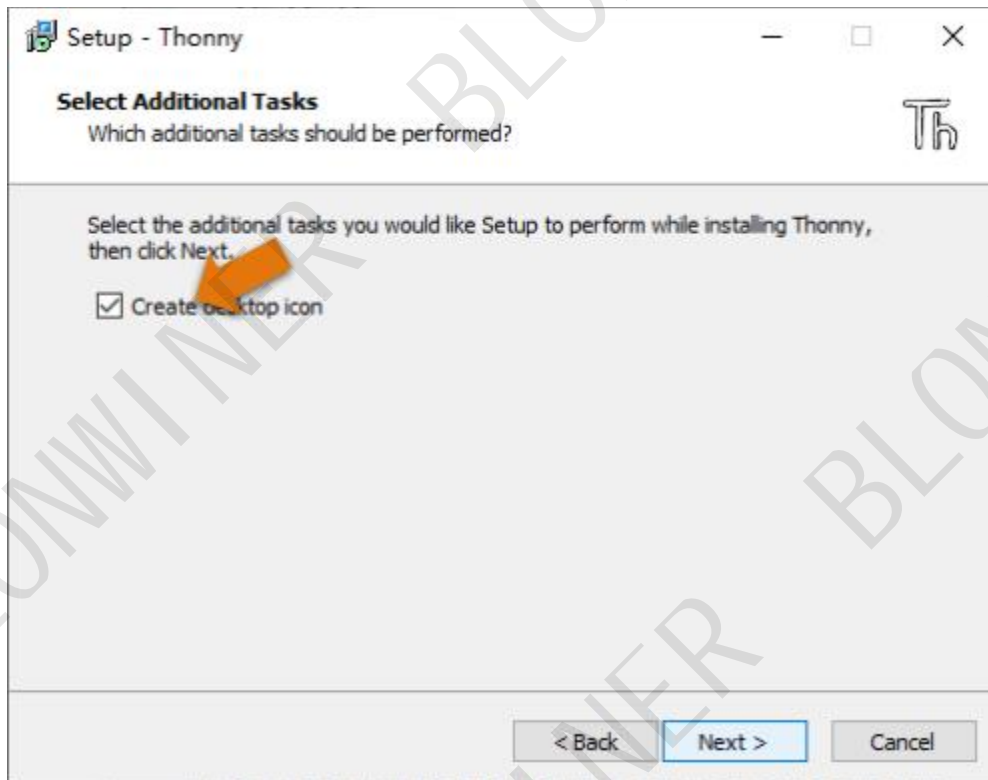
If you want to change Thonny's installation path, you can click "Browse" to modify it. After selecting installation path, click "OK".

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

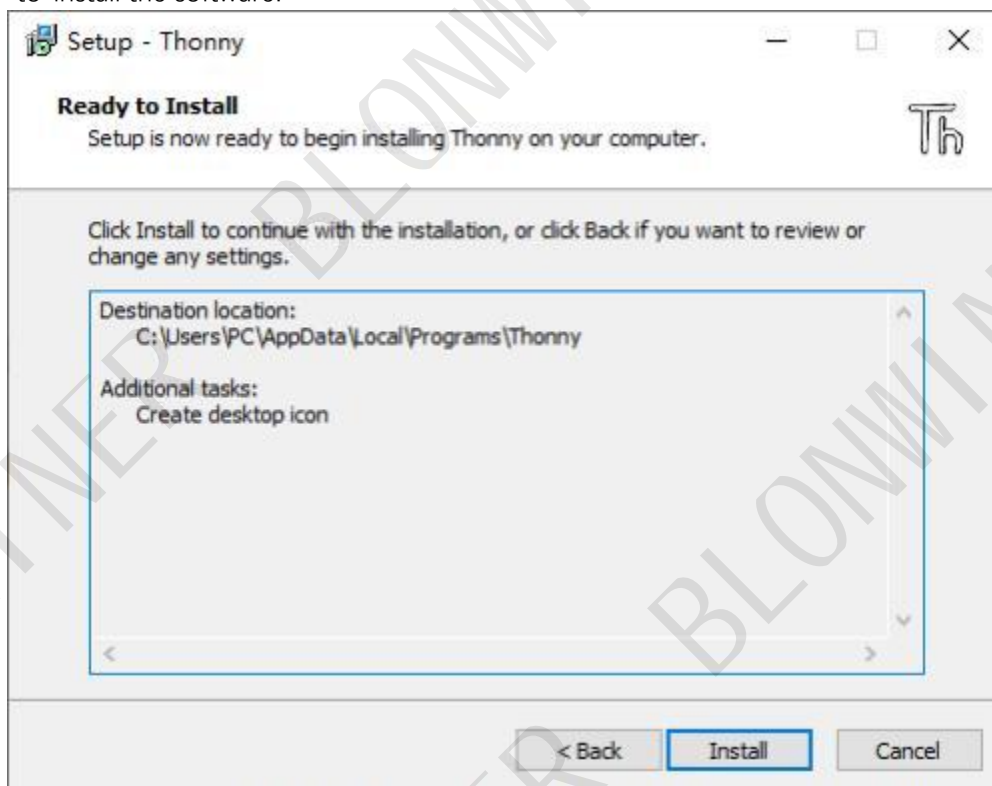




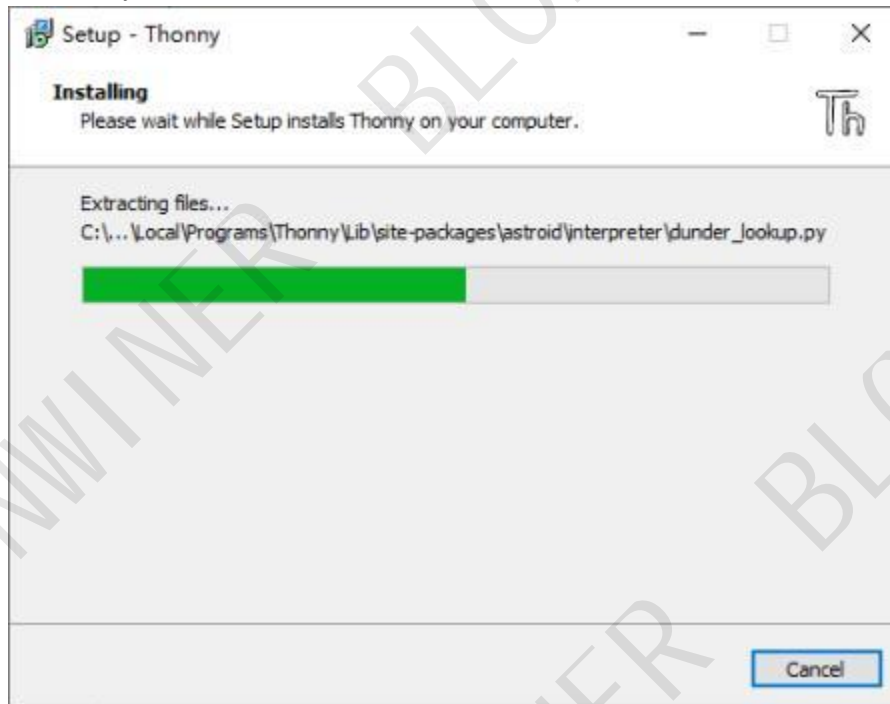
Check “Create desktop icon” and then it will generate a shortcut on your desktop to facilitate you to open Thonny later.



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.

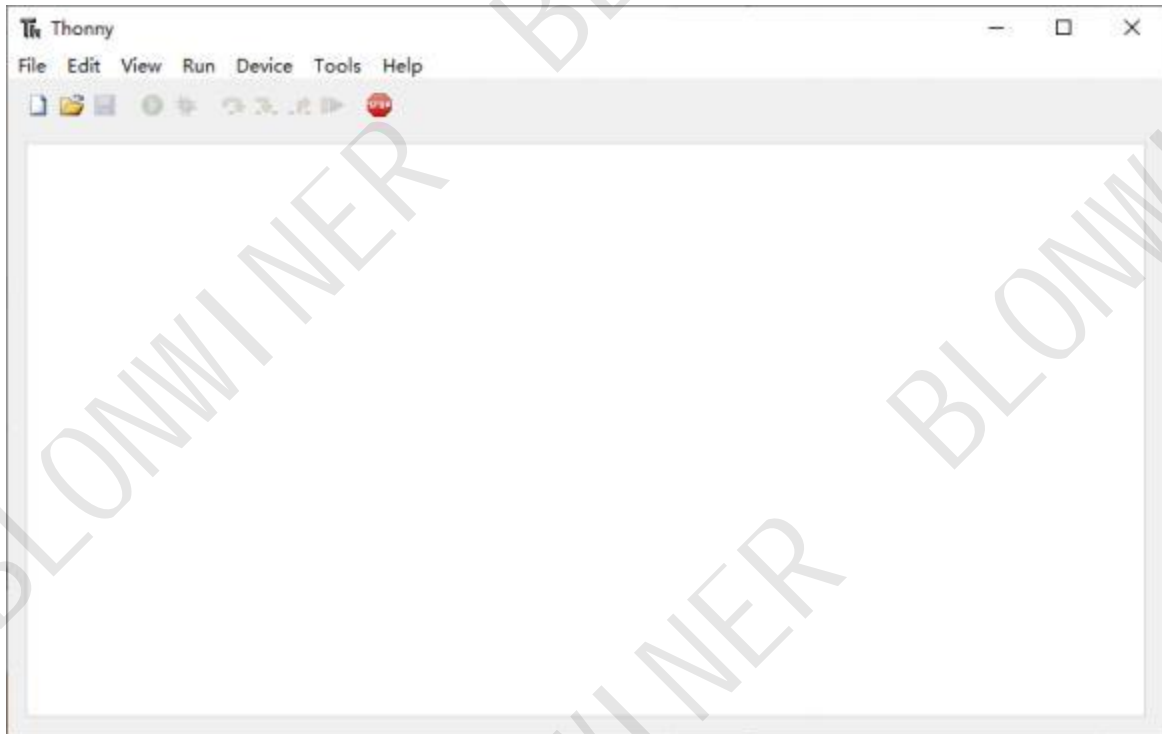


If you've checked "Create desktop icon" during the installation process, you can see the below icon on your desktop.

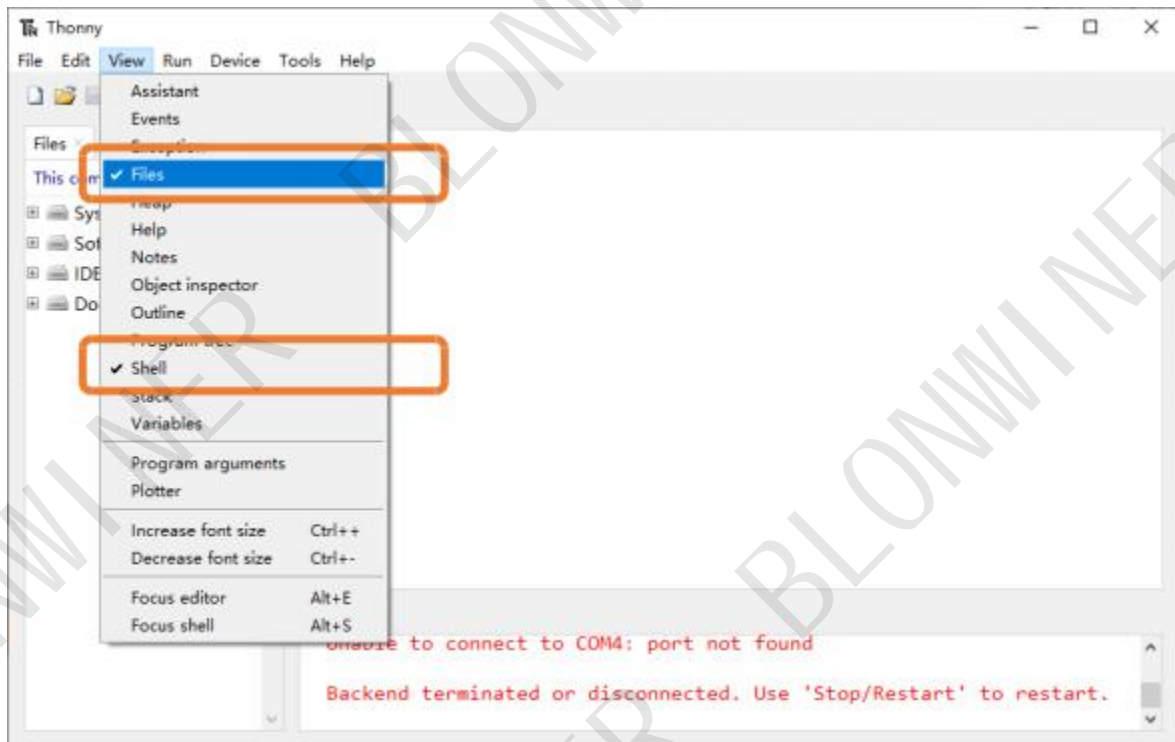


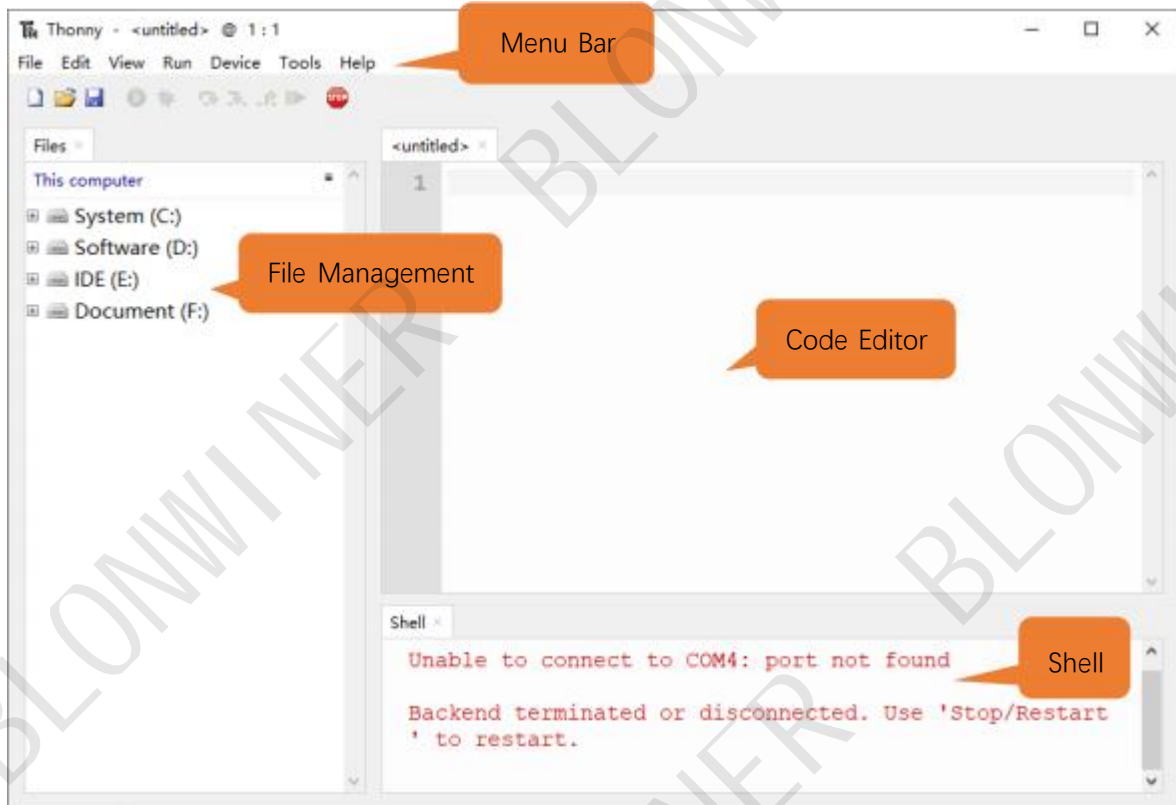
0.2 Basic Configuration of Thonny

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



Select "View" → "Files" and "Shell".





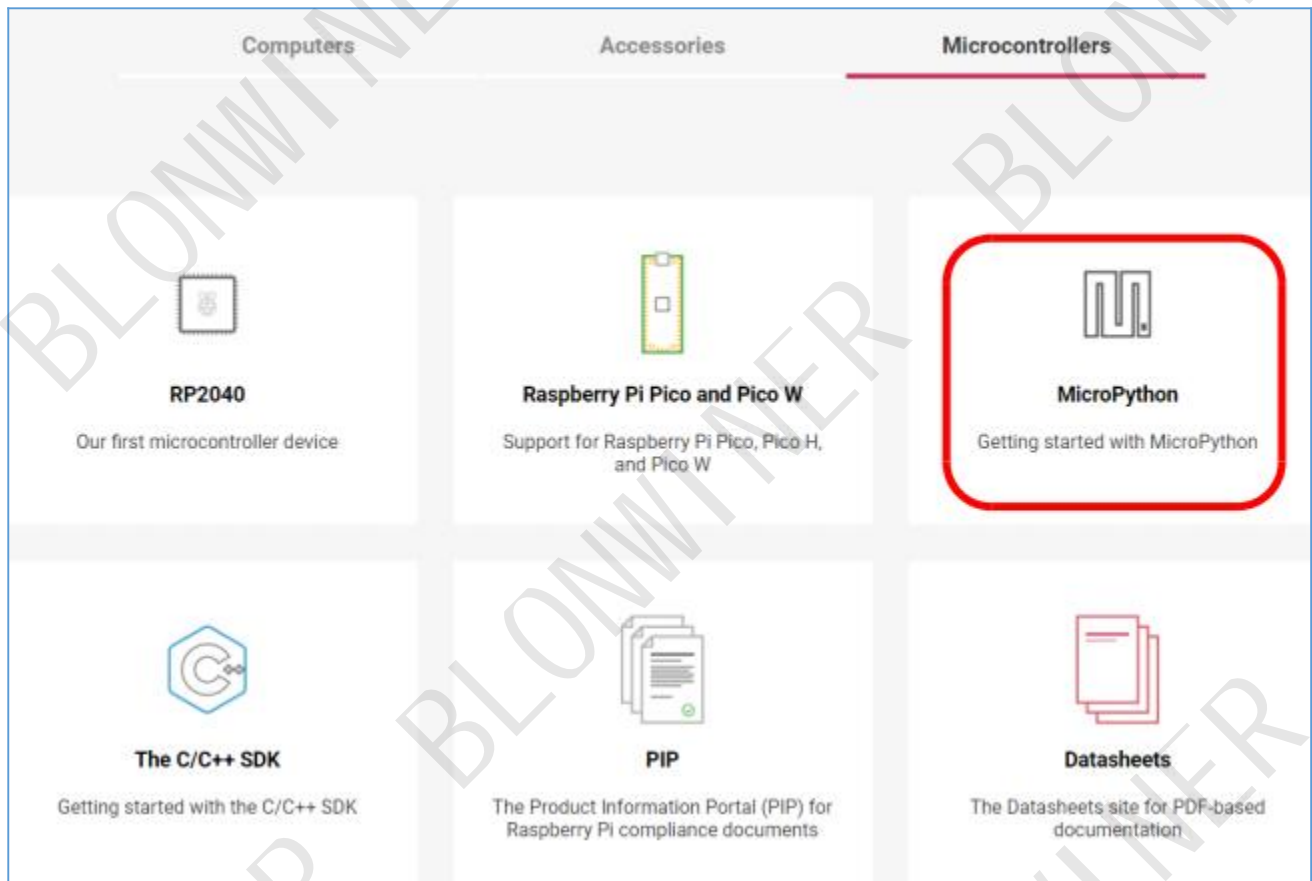
0.3 Burning Micropython Firmware (Important)

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

Downloading Micropython Firmware

Raspberry Pi Pico official website: <https://www.raspberrypi.com/documentation/microcontrollers/>

1, Click Micropython.

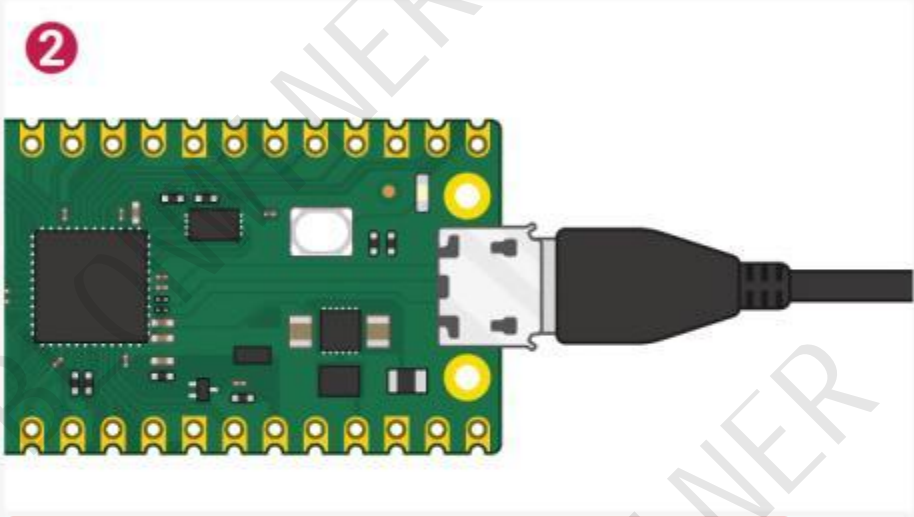


2, In the new interface, find the following content and download the corresponding UF2 according to your Pico version.

Drag-and-Drop MicroPython

[Edit this on GitHub](#)

You can program your Pico by connecting it to a computer via USB, then dragging and dropping a file onto it so we've put together a downloadable UF2 file to let you install MicroPython more easily.



Download the correct MicroPython UF2 file for your board:

- [Raspberry Pi Pico](#)
- [Raspberry Pi Pico W](#) (with `urequests` and `upip` preinstalled)

Then go ahead and:

1. Push and hold the BOOTSEL button and plug your Pico into the USB port of your Raspberry Pi or other computer. Release the BOOTSEL button after your Pico is connected.
2. It will mount as a Mass Storage Device called RPI-RP2.

If you are using a Pico board, please click Raspberry Pi Pico to download it.

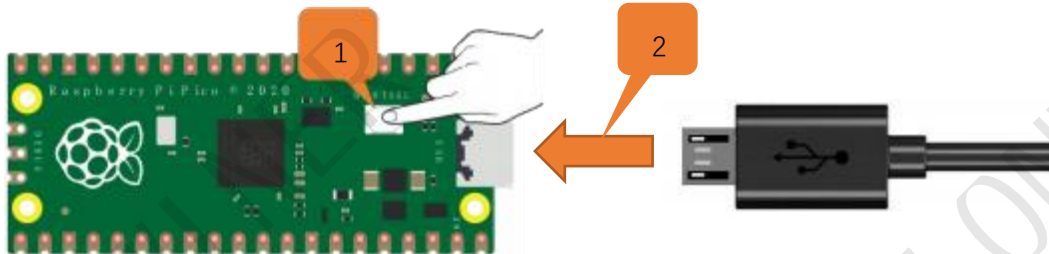
If you are using a Pico W board, please click Raspberry Pi Pico W to download it.



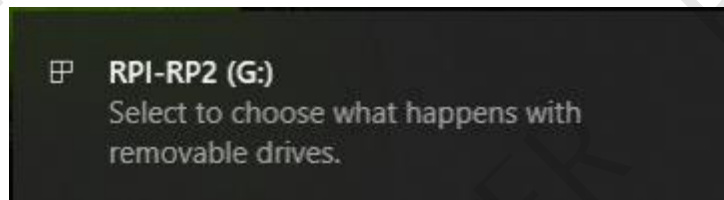
Burning a Micropython Firmware

Note: Pico and Pico W burn firmware in the same way. Pico's map is used here as an introduction.

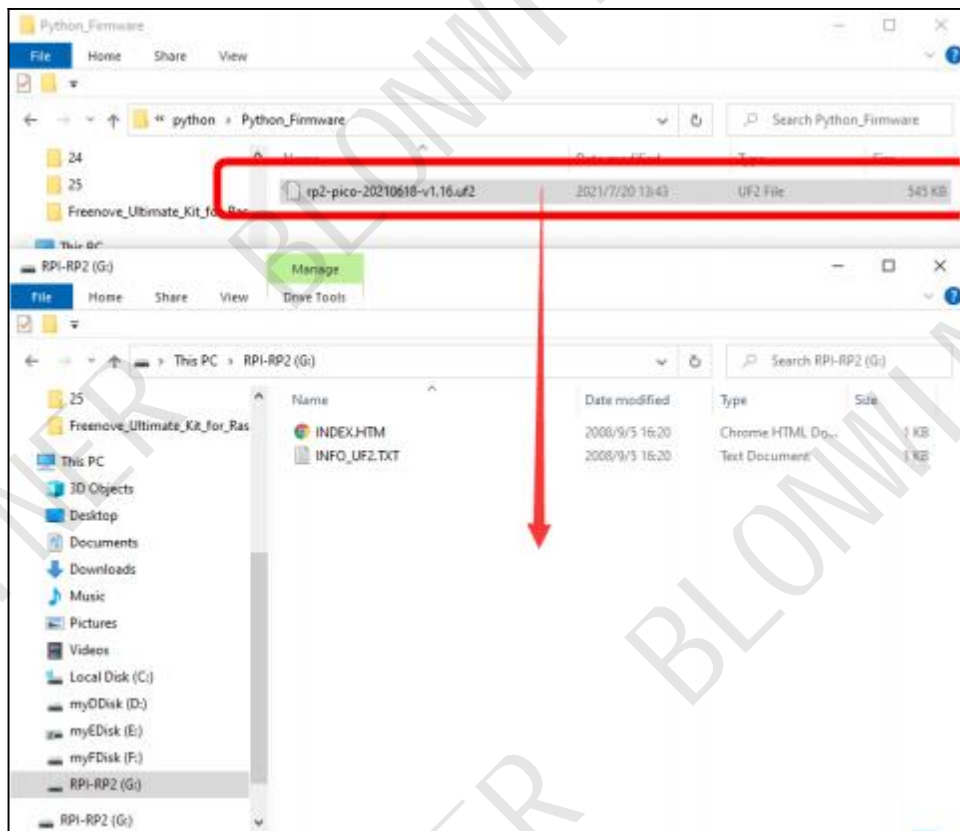
1. Connect a USB cable to your computer.
2. Long press BOOTSEL button on Raspberry Pi Pico and connect it to your computer with the USB cable.



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.

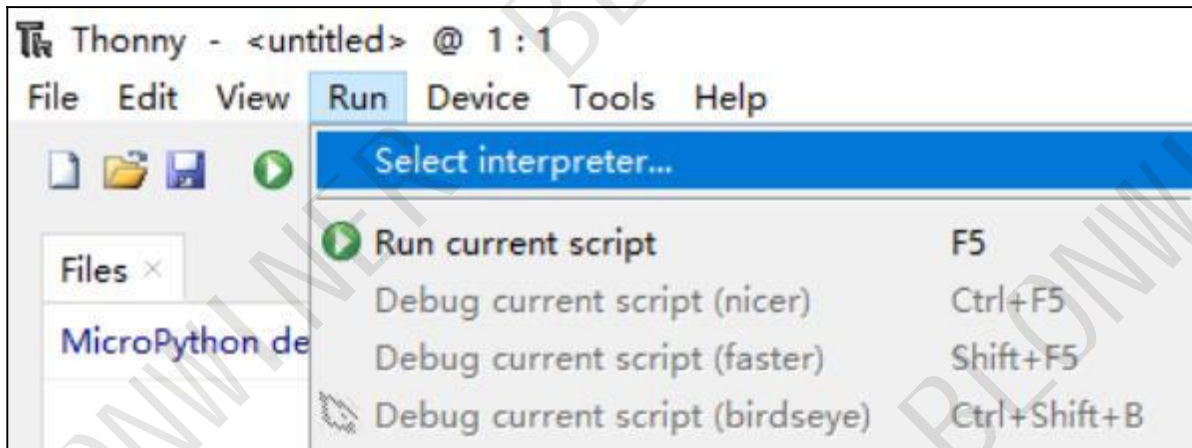


5. When the firmware finishes programming, Raspberry Pi Pico will reboot automatically. After that, you can run Micropython.

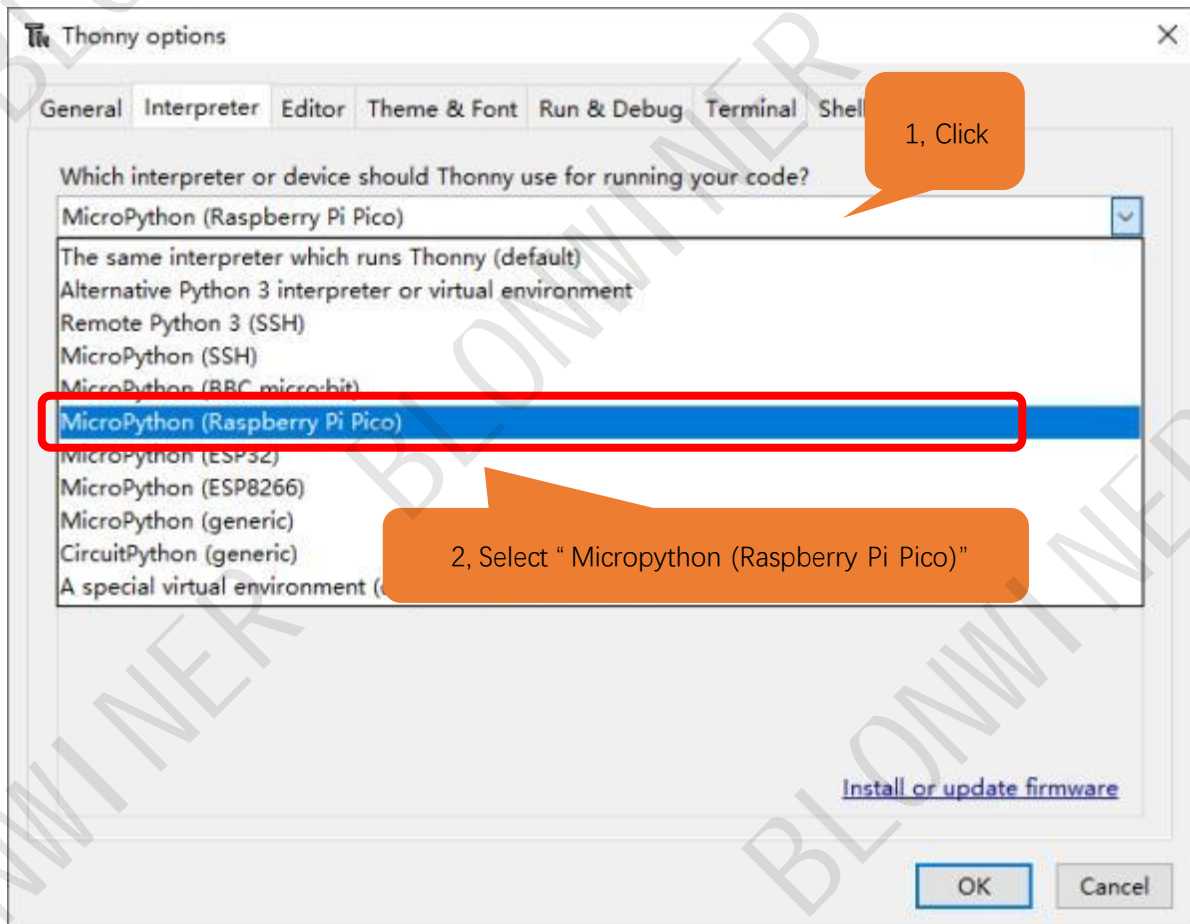
Any concerns? ✉ blonwiner@outlook.com

0.4 Thonny Connected to Raspberry Pi Pico

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



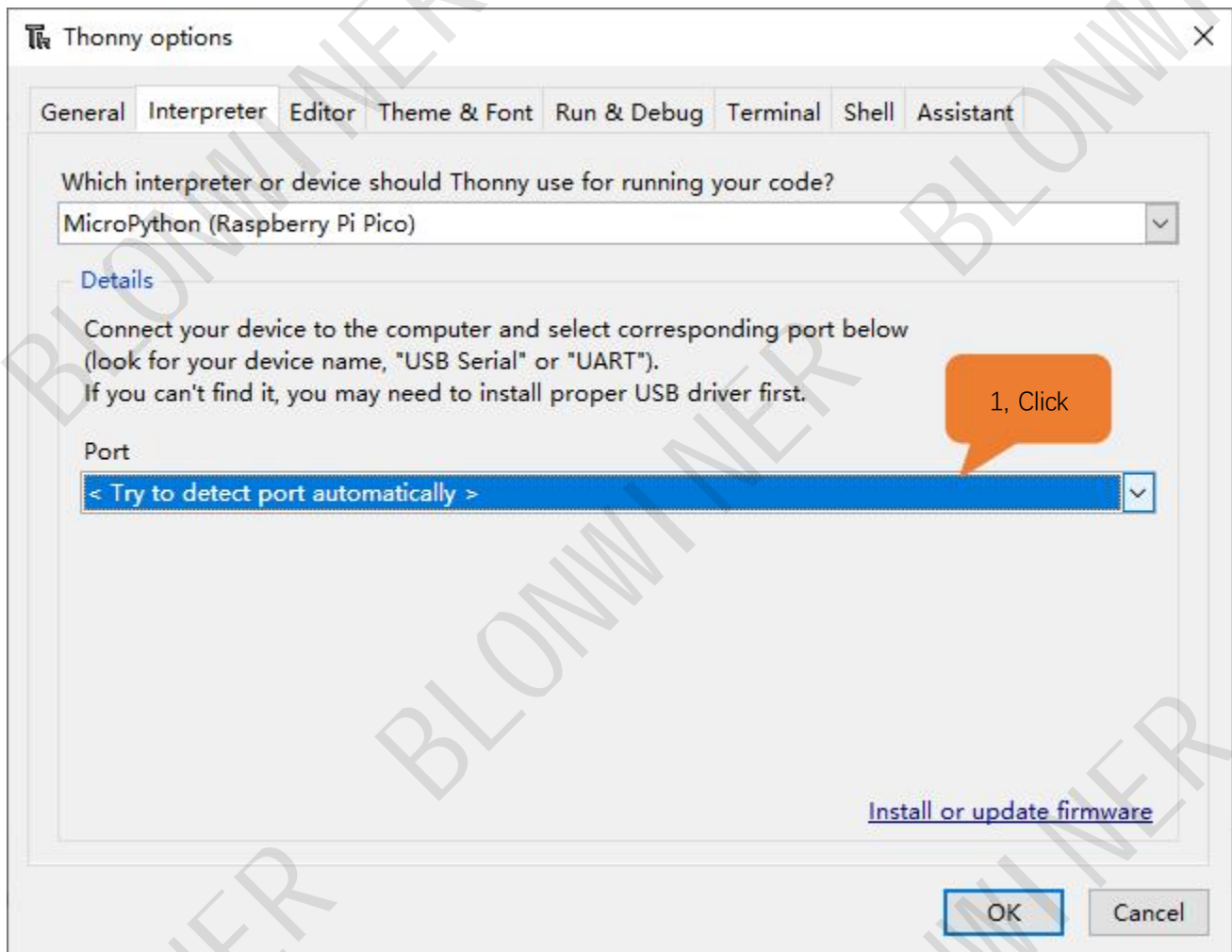
2. Select "Micropython (Raspberry Pi Pico)".



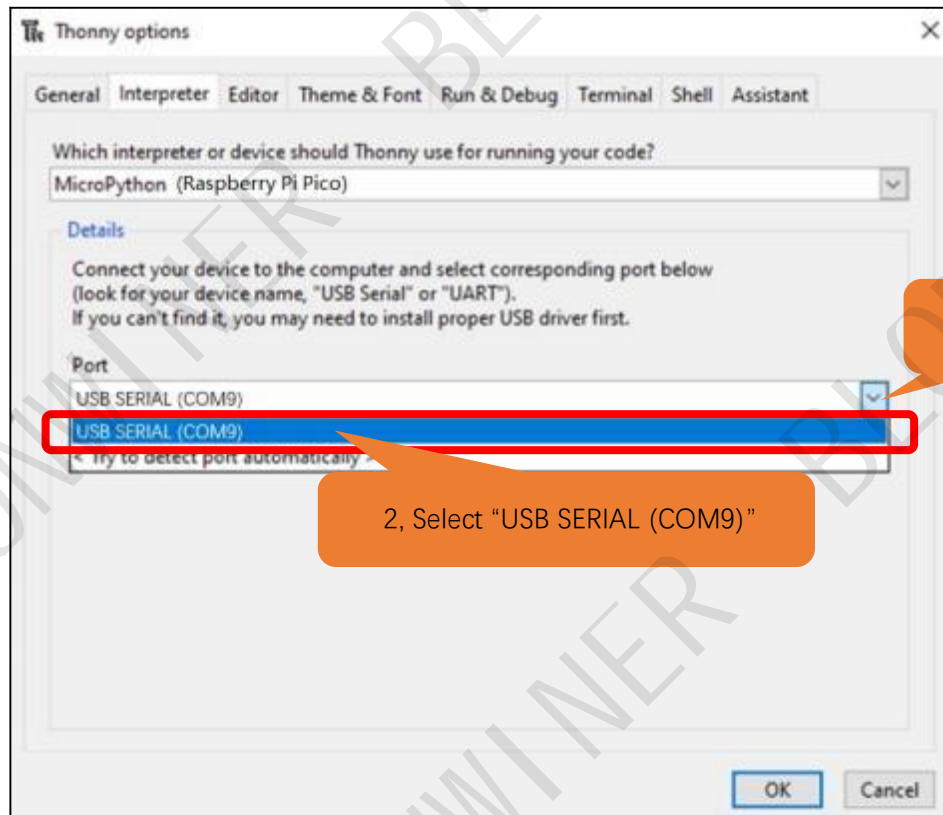
3. Select "USB-SERIAL (COMx)", The number x of COMx may varies among different computers. You only need to make sure selecting USB-SERIAL (COMx).

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

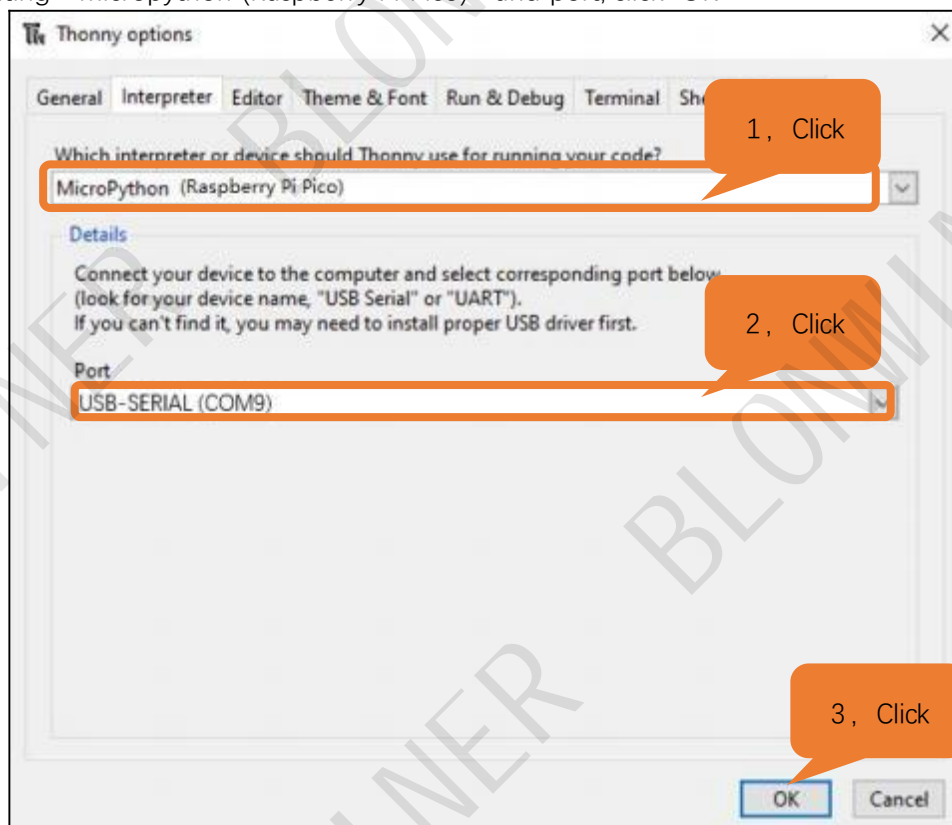
Step 1: When Pico **doesn't** connect to computer, open Thonny, click "Run", select "Select interpreter" and then a dialog box will pop up, click "Port" and you can check the ports currently connected to your computer, as shown below:



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.

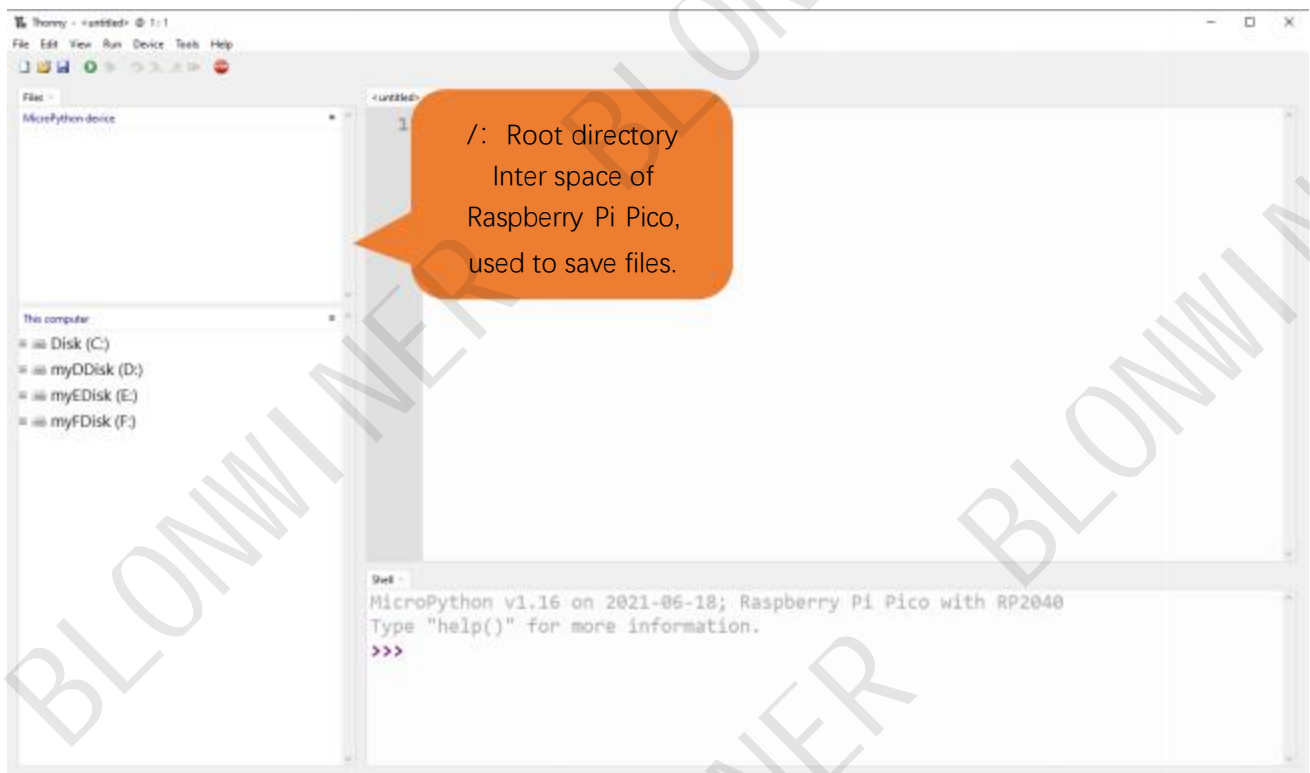


4. After selecting "Micropython (Raspberry Pi Pico)" and port, click "OK"





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



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.

Disadvantages:

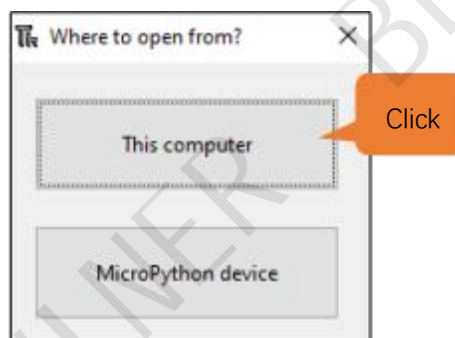
1. To run Raspberry Pi Pico online, you have to be connected to a computer and run with Thonny.
2. If Raspberry Pi Pico disconnects from computer, the program won't run again when they reconnect to each other.

Opertation

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



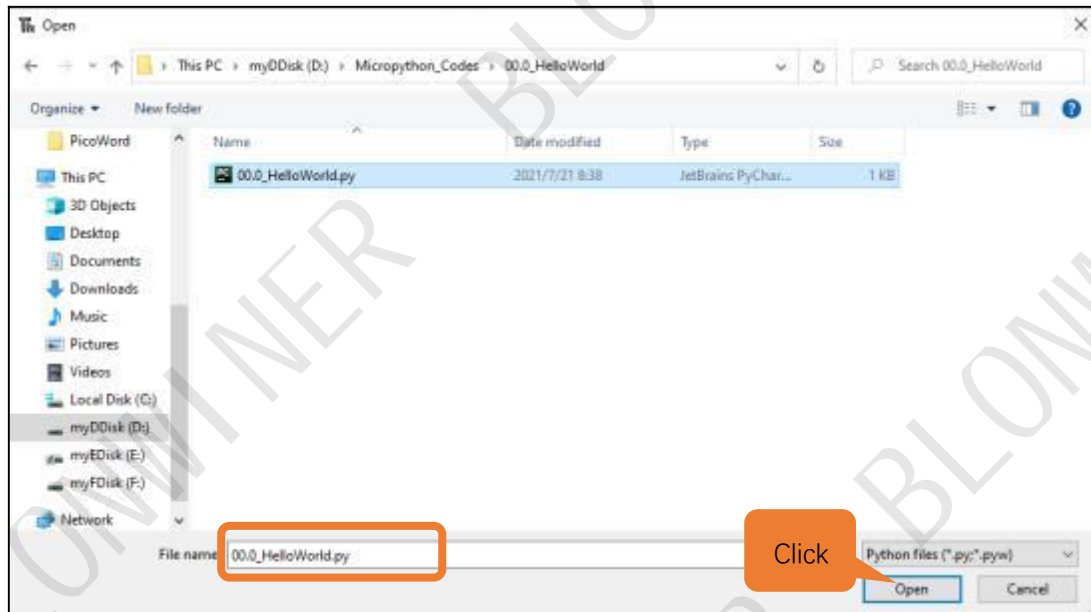
2. On the newly pop-up window, click "This computer".



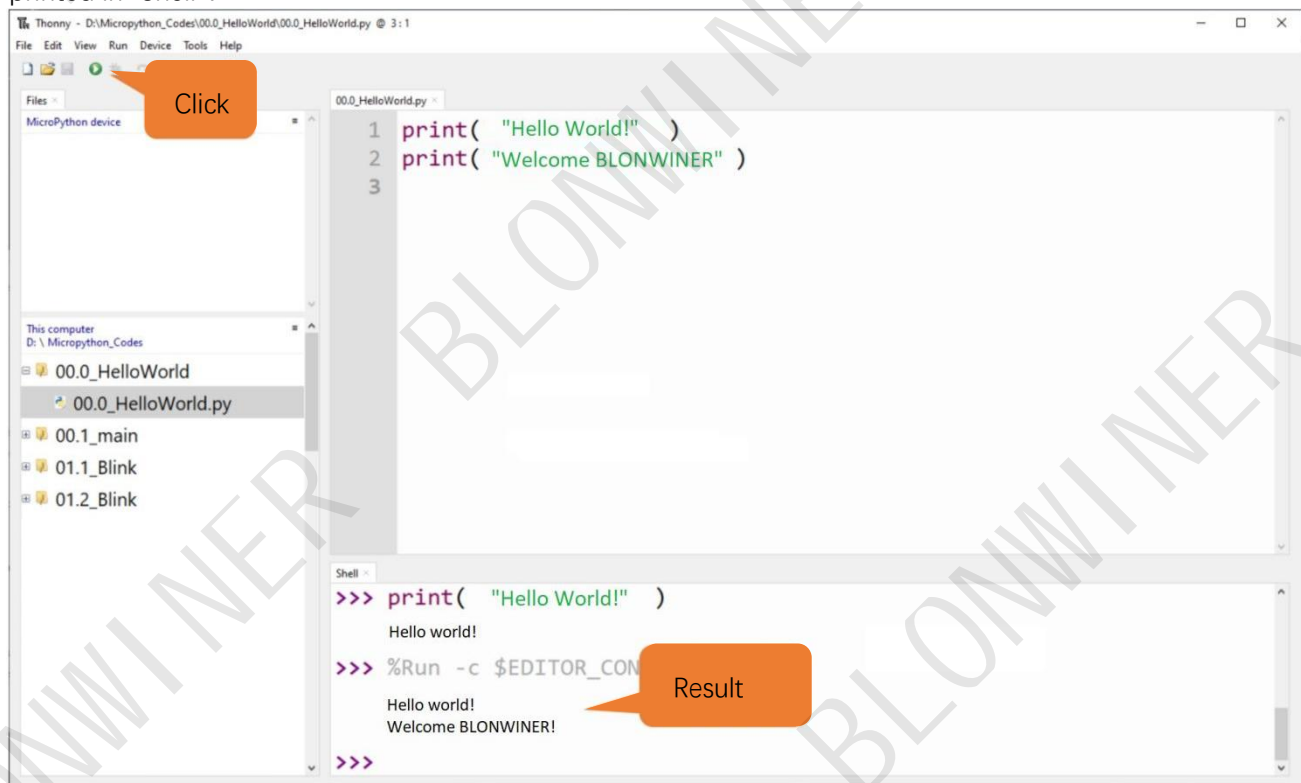
Any concerns? ✉ blonwiner@outlook.com

In the new dialog box, select “00.0_HelloWorld.py” in

“BLONWINER_Breakout_Board_for_Raspberry_Pi_Pico\Python\Python_Codes\00.0_HelloWorld” folder.



Click “Run current script” to execute the program and “Hello World!”, “Welcome BLONWINER” 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.



Any concerns? [✉ blonwiner@outlook.com](mailto:blonwiner@outlook.com)



Running Offline

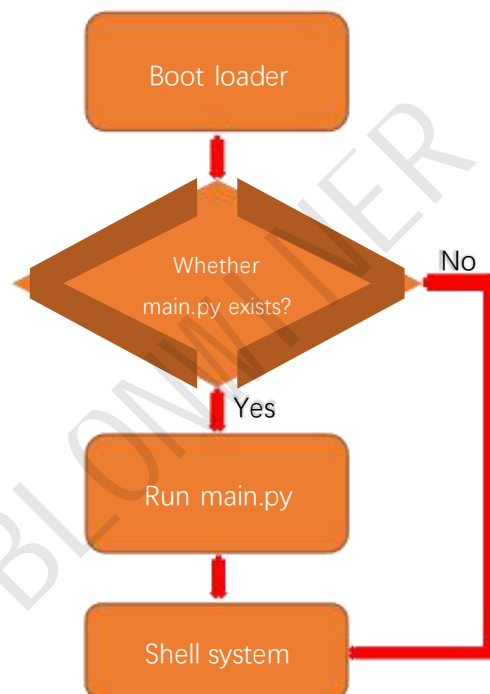
When running offline, Raspberry Pi Pico doesn't need to connect to computer and Thonny. It can run the programs stored in `main.py` on the device once powered up.

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

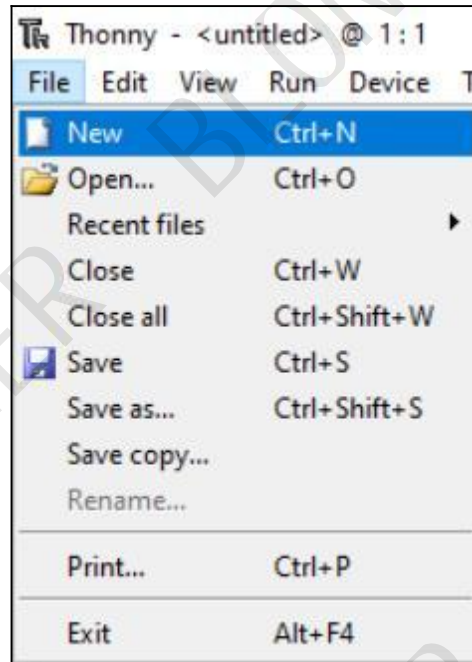
Disvantage: The program will stop automatically when error occurs or Raspberry Pi Pico is out of power. Code cannot be changed easily.

Operation

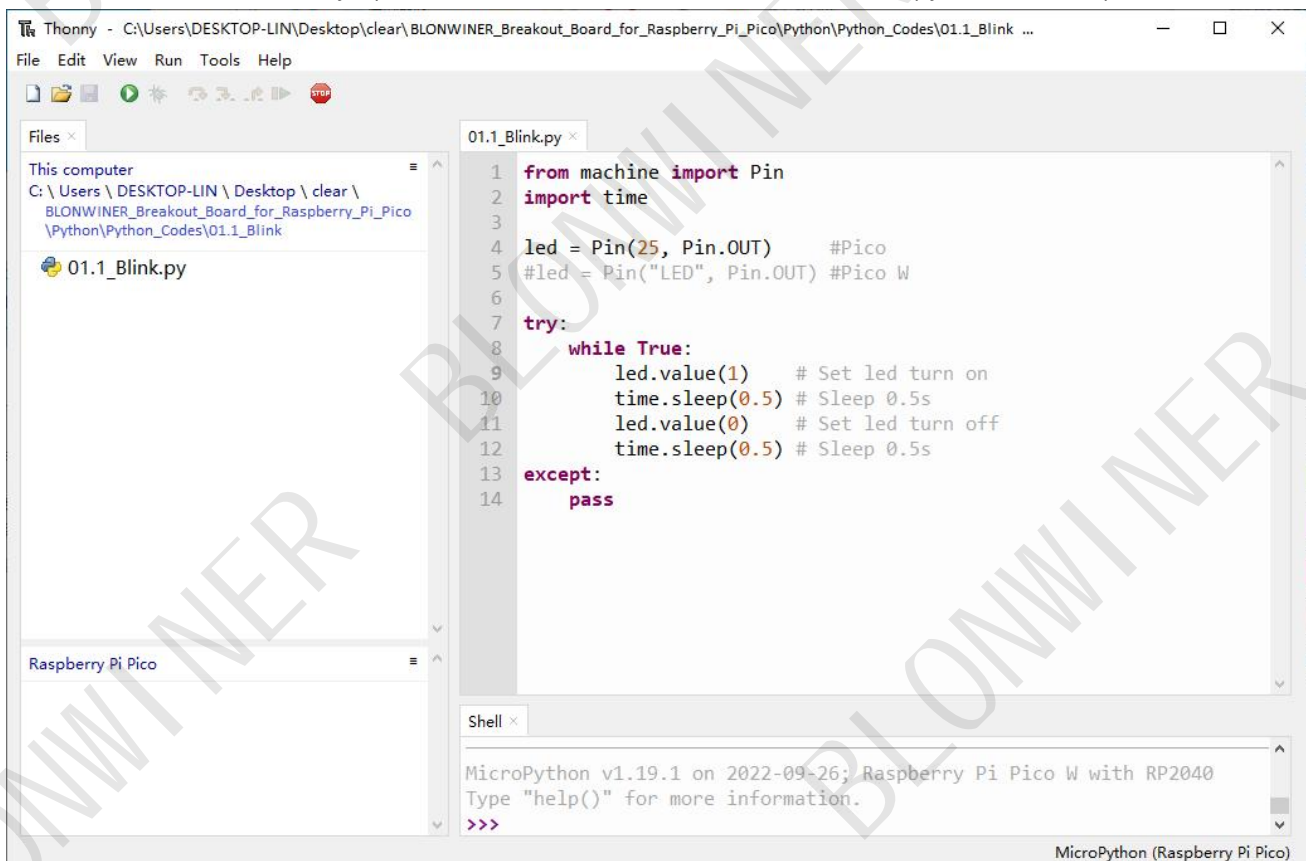
Once powered up, Raspberry Pi Pico will automatically check whether there is `main.py` existing on the device. If there is, it runs the programs in `main.py` and then enter shell command system. (If you want the code to run offline, you can save it as `main.py`); If `main.py` doesn't exist, it will enter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



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

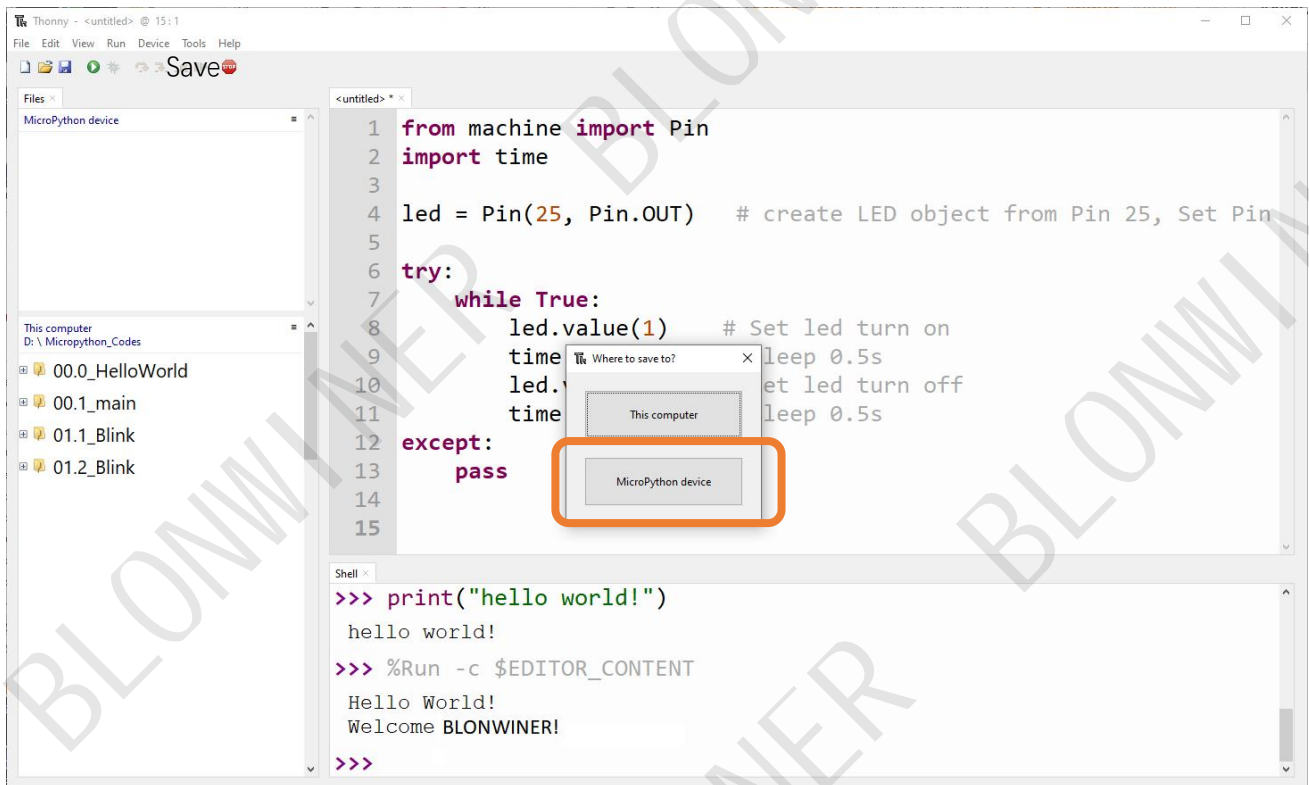


2. Enter codes in the newly opened file. Here we use codes of "01.1_Blink.py" as an example.

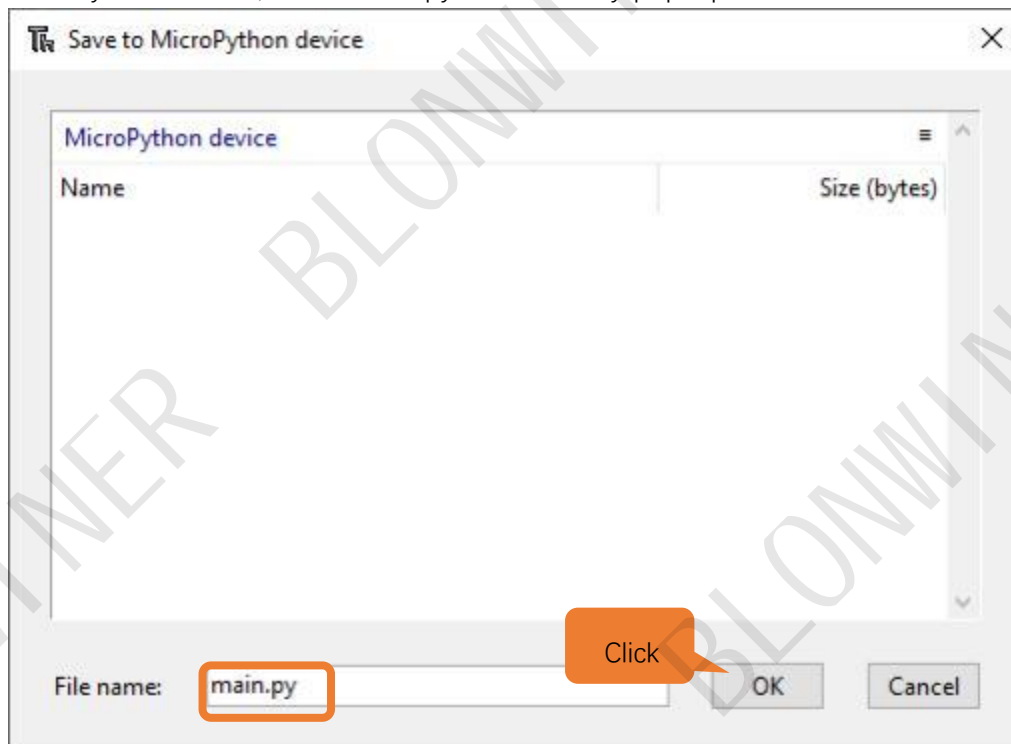


Note: If your board is Pico, please use 25 to drive the on-board LED. If your board is Pico W, use "LED" to drive the on-board LED. Pico is used as an example to explain.

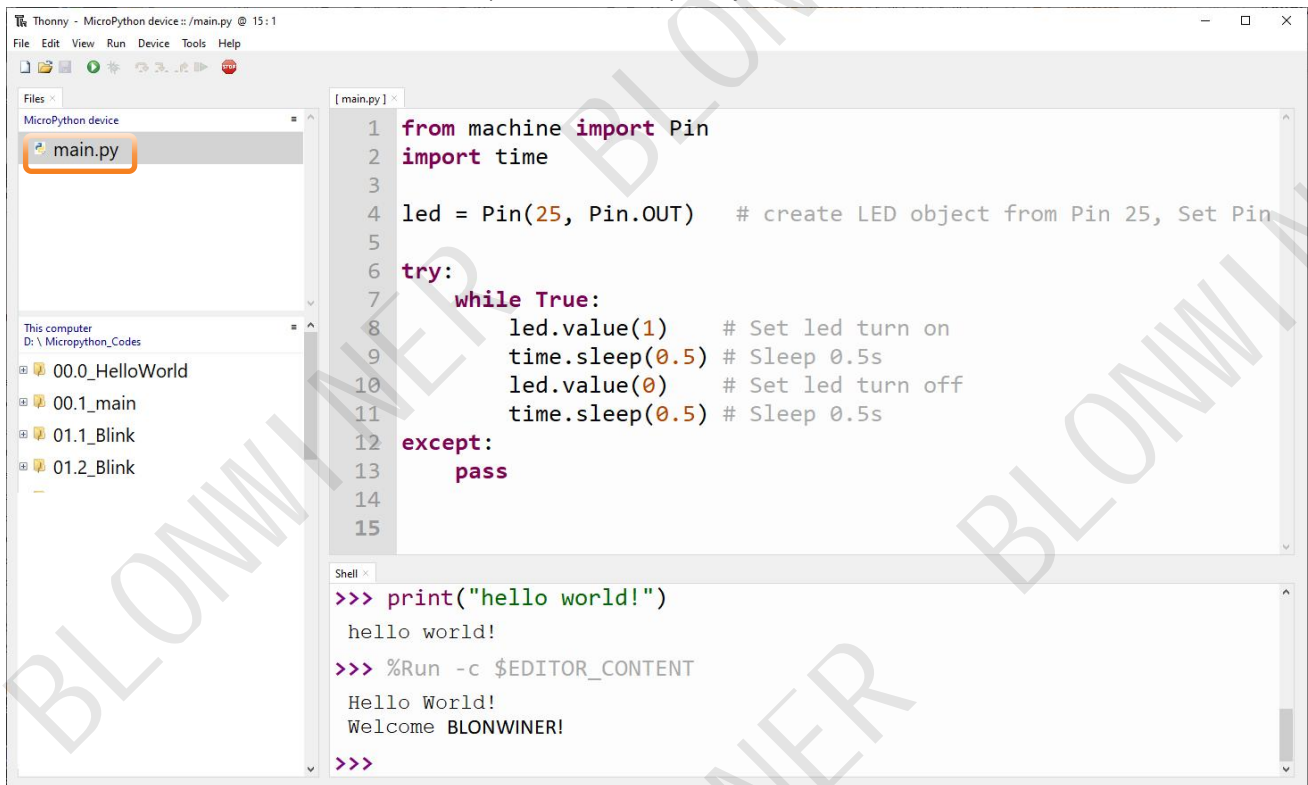
3. Click “Save” on the menu bar. You can save the codes either to your computer or to Raspberry Pi Pico.



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



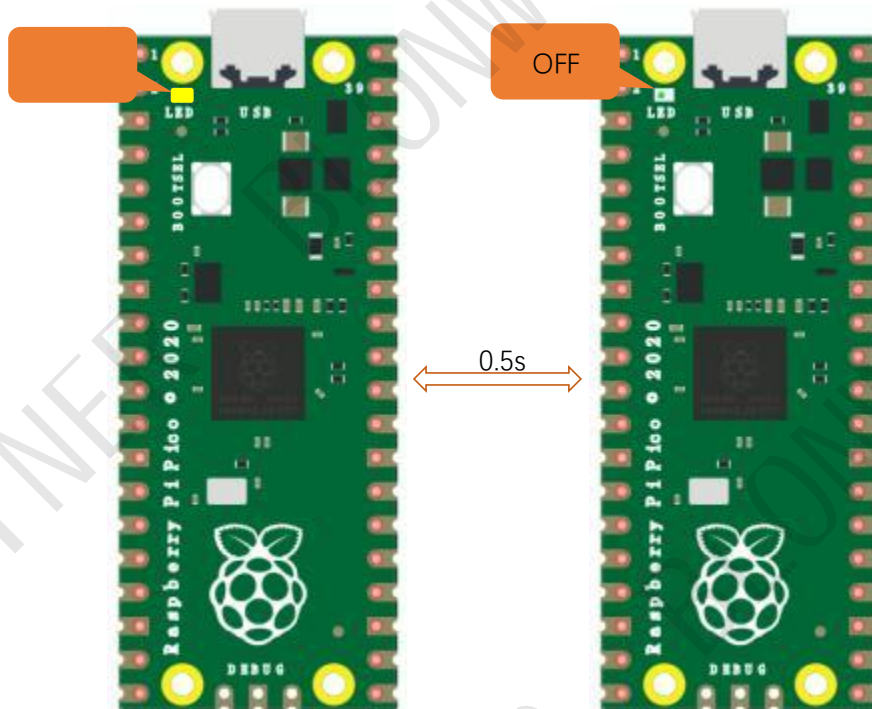
5. You can see that codes have been uploaded to Raspberry Pi Pico.



```
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 BLONWINER!
>>>
```

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.

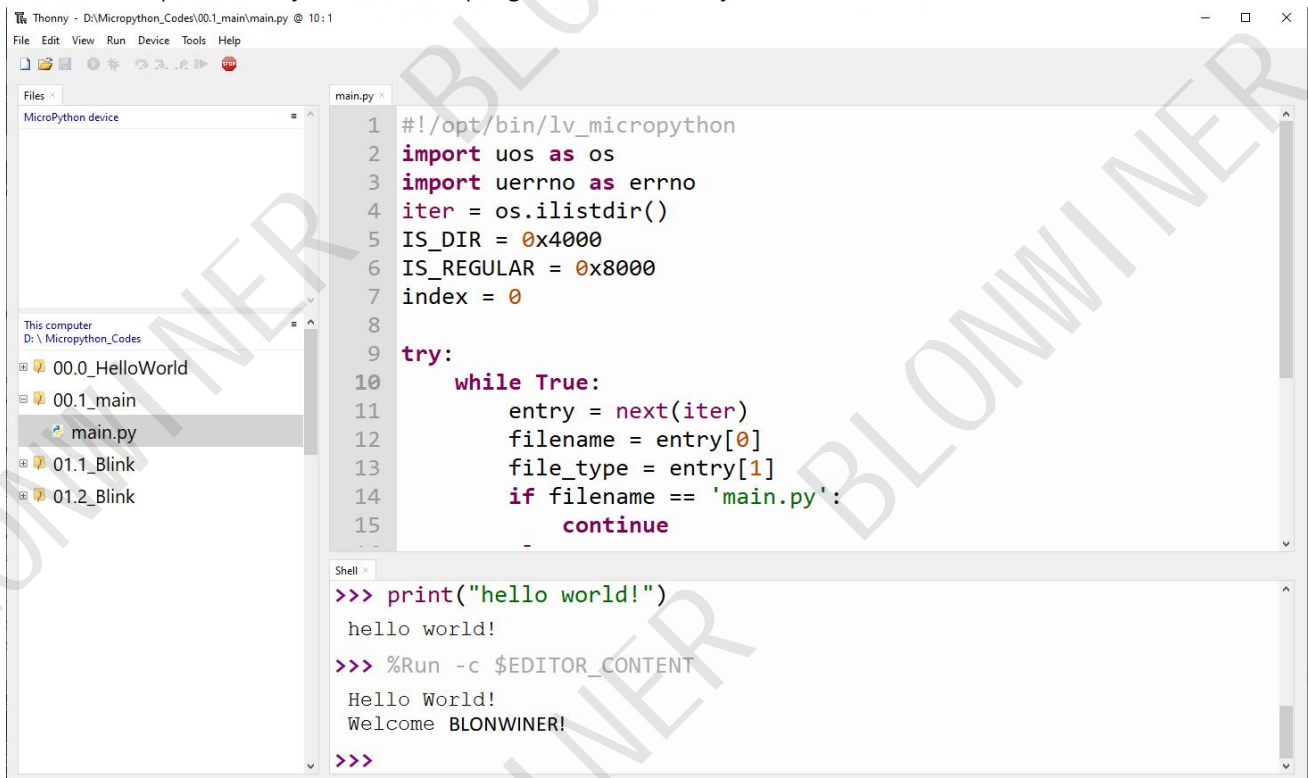
We provide a main.py file for running offline. The code added to main.py is a bootstrap that executes the user's code file. All you need to do is upload the offline project's code file (.py) to the Raspberry Pi Pico device.

1. Move the program folder

"**BLONWINER Breakout Board for Raspberry Pi Pico\Python\Python_Codes**" to disk(D) in advance with the path of "**D:/Micropython_Codes**". Open "Thonny".

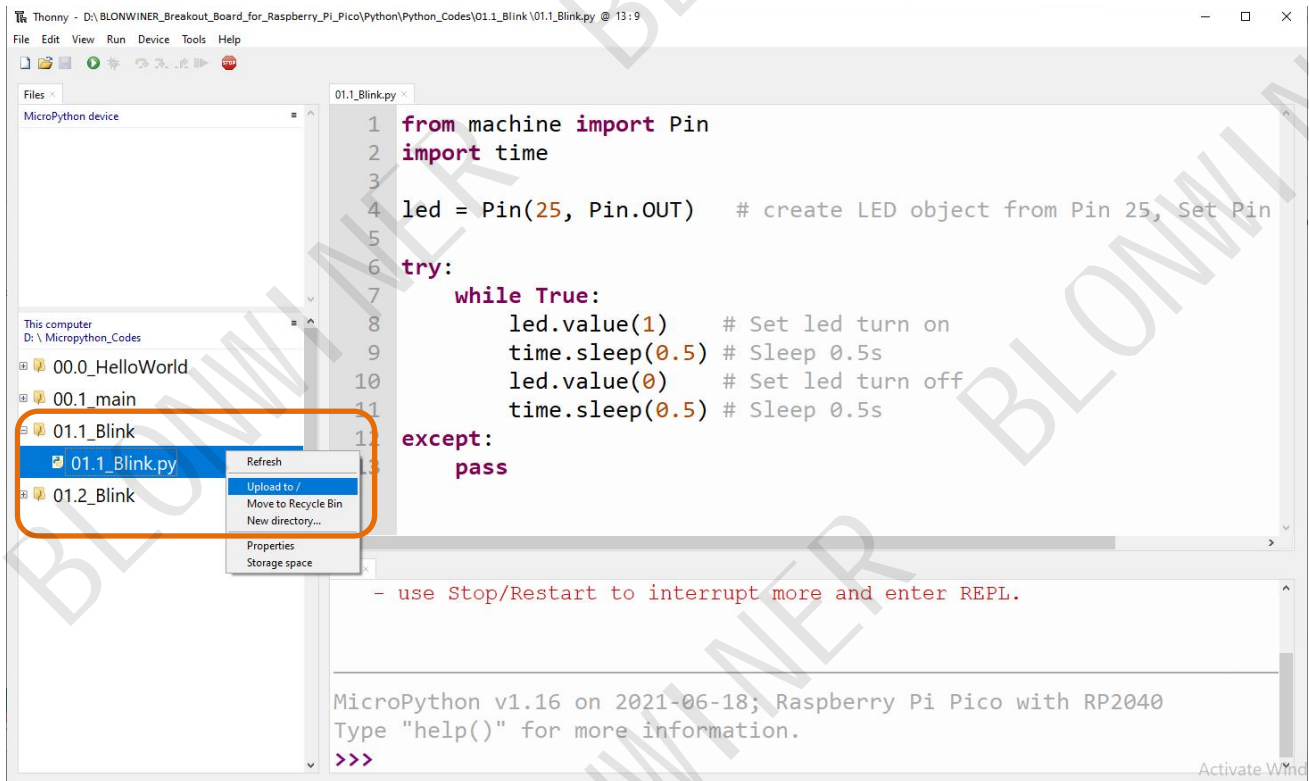


2. Expand "00.1_main" in the "Micropython_Codes" in the directory of disk(D), and double-click main.py, which is provided by us to enable programs in "MicroPython device" to run offline.

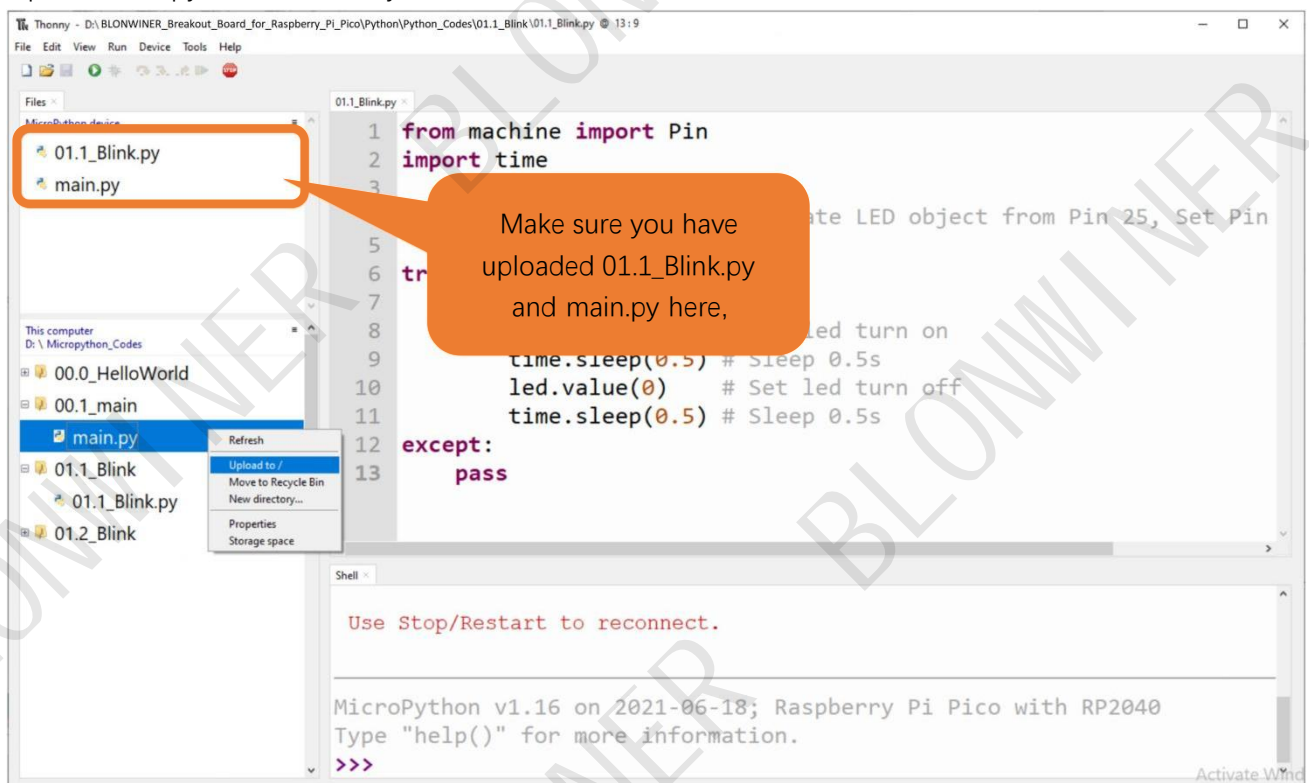


Here we use 00.1 and 01.1 cases as demonstration. The LED on Raspberry Pi Pico is used to show the result, which uses GP25 pin. If you have modified 01.1_Blink.py file, you need to change it accordingly.

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.

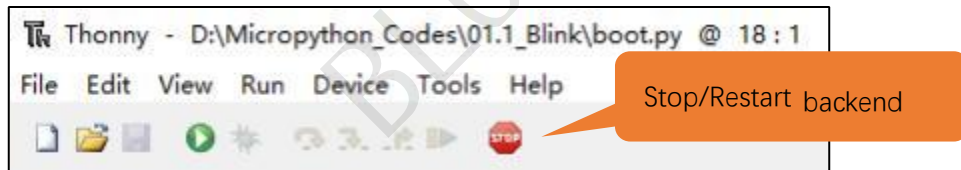


Disconnect Raspberry Pi Pico USB cable and reconnect it, the LED on pico will blink repeatedly.

Any concerns? ✉ blonwiner@outlook.com

Note:

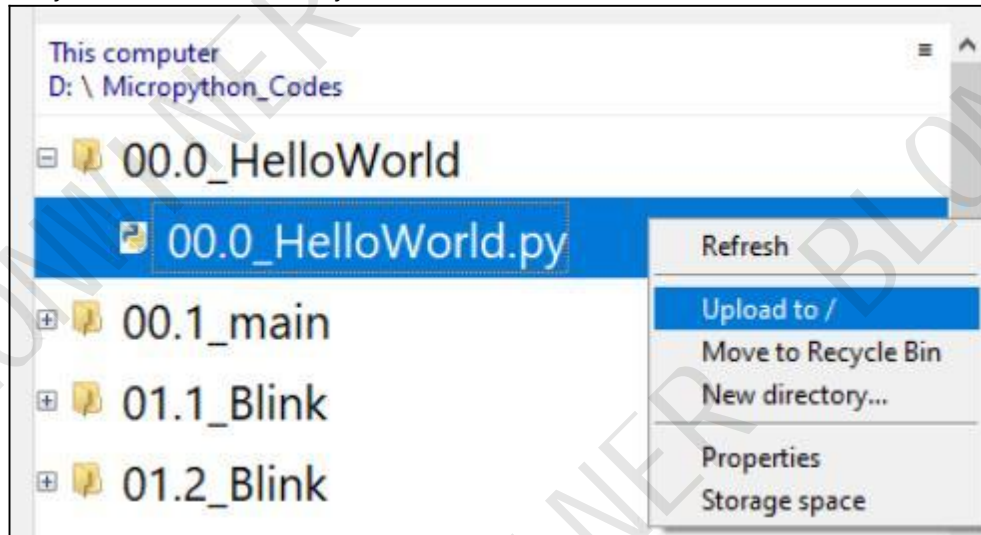
Codes here are run offline. If you want to stop running offline and enter Shell, just click "Stop" in Thonny.



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.



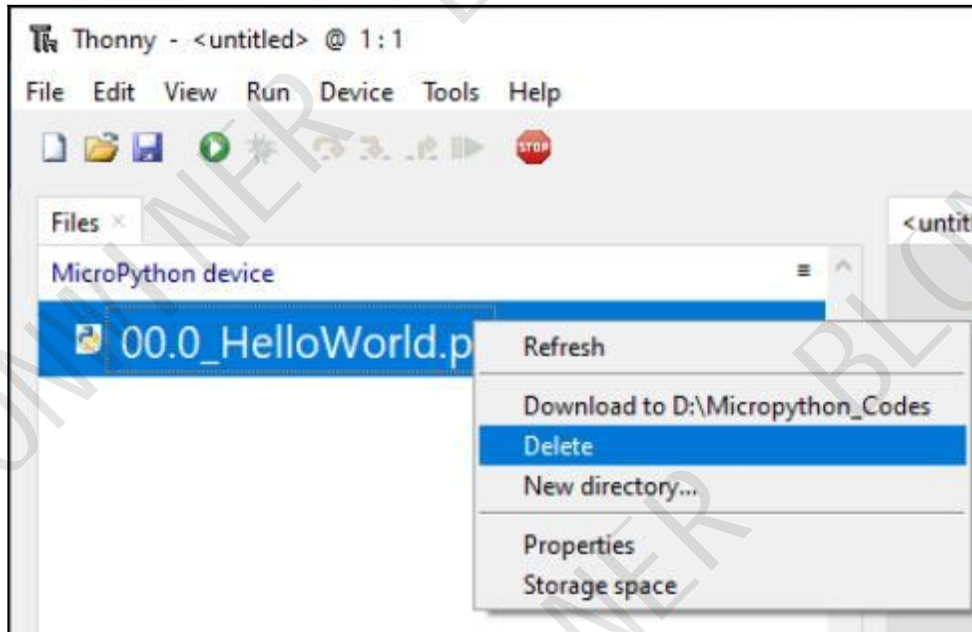
Downloading Code to Computer

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



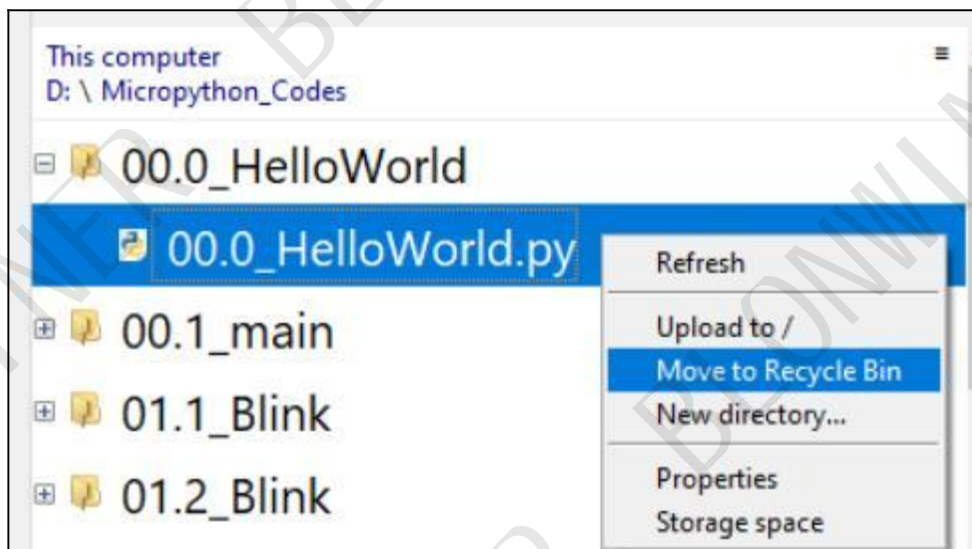
Deleting Files from Raspberry Pi Pico's Root Directory

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".

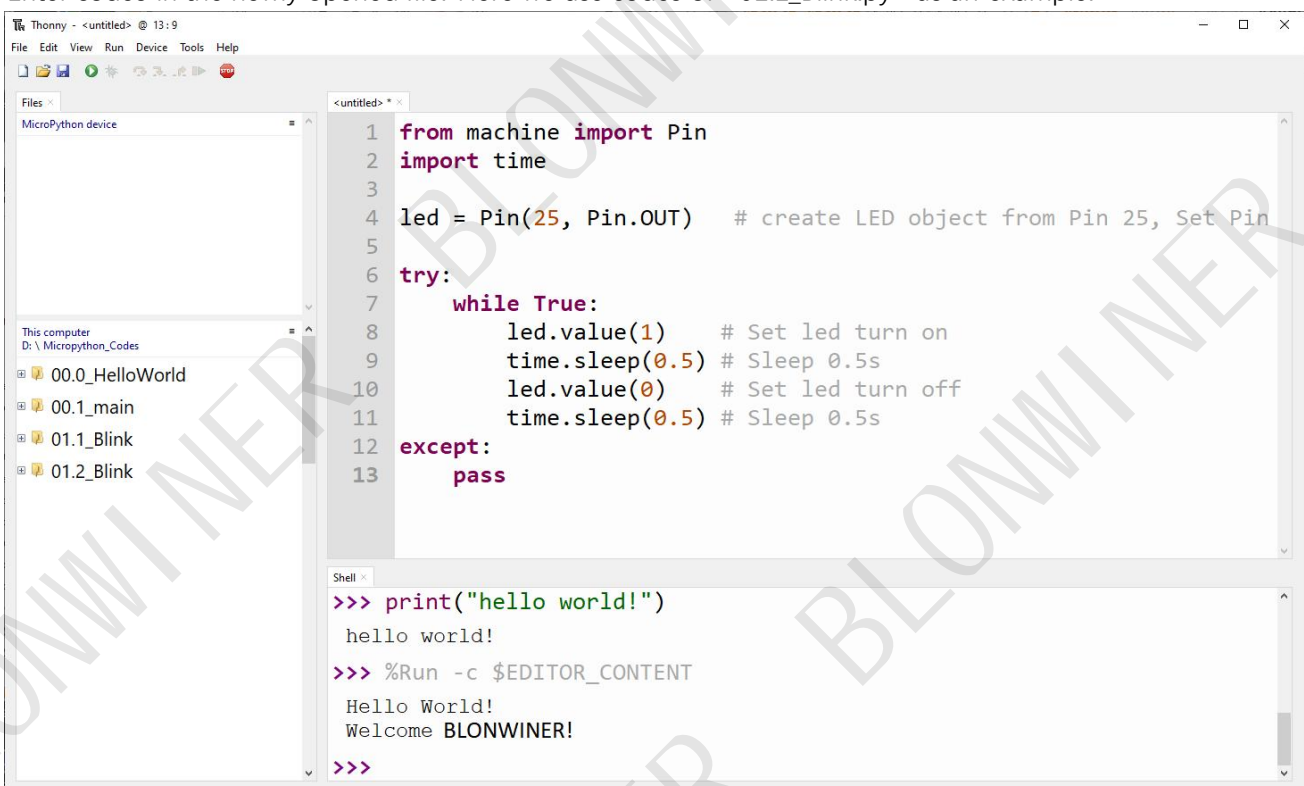


Creating and Saving the code

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



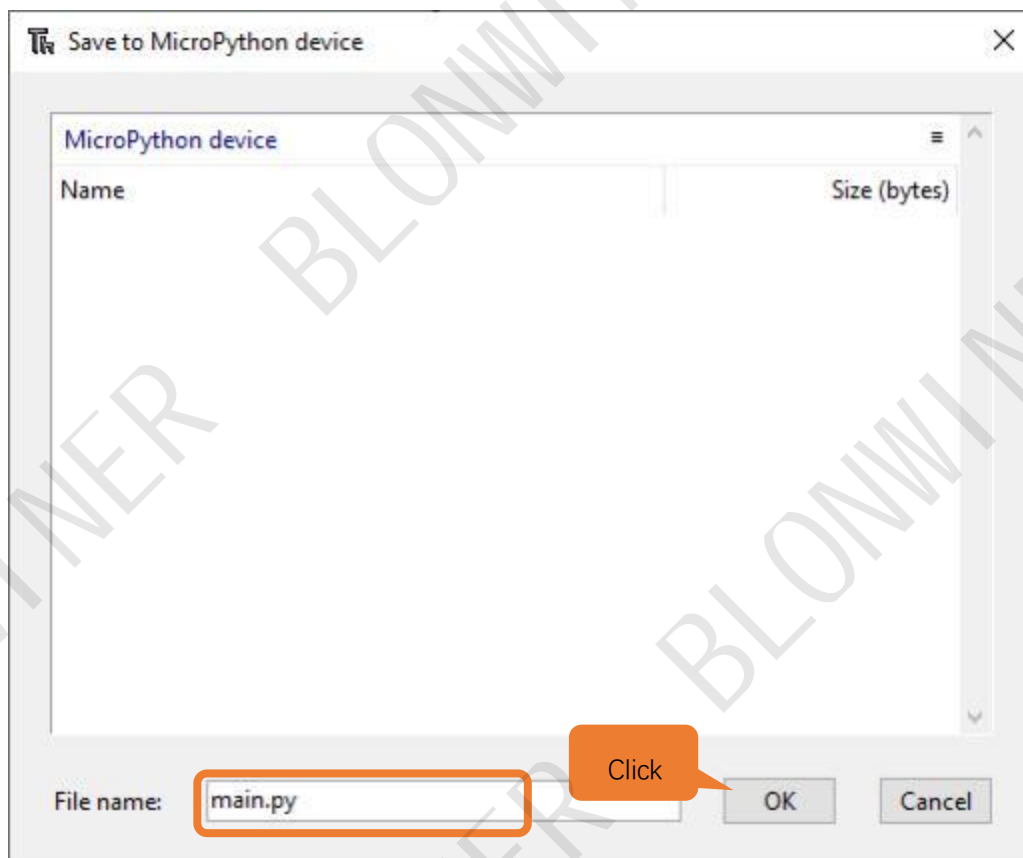
Enter codes in the newly opened file. Here we use codes of “01.1_Blink.py” as an example.



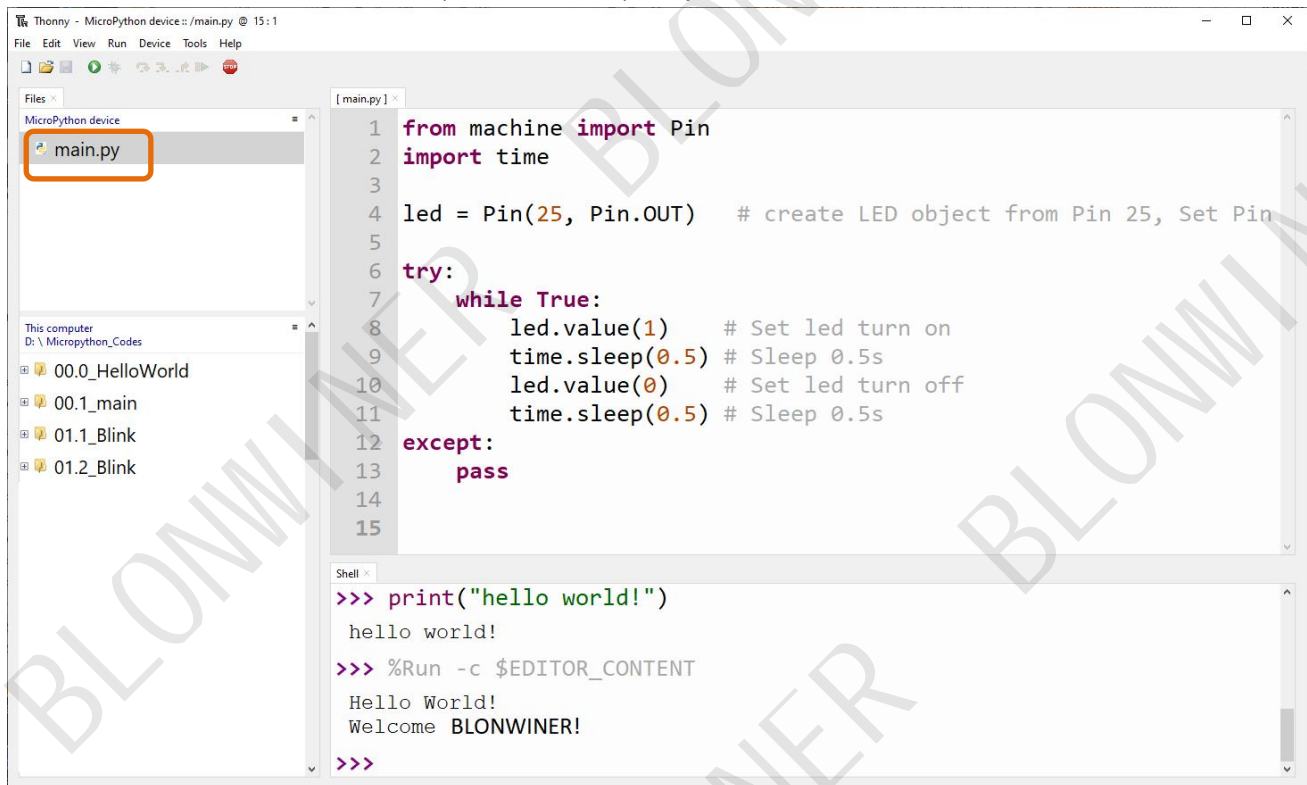
Click “Save” on the menu bar. You can save the codes either to your computer or to Raspberry Pi Pico.



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.



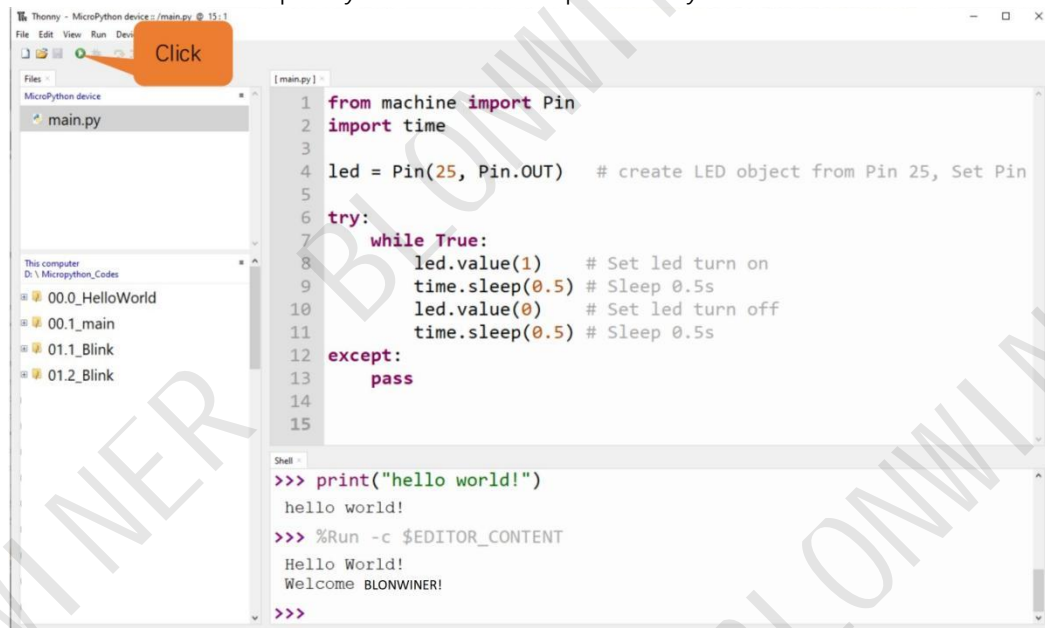
The screenshot shows the Thonny IDE interface. The top menu bar includes File, Edit, View, Run, Device, Tools, and Help. The left sidebar has a 'Files' pane showing 'main.py' selected under 'MicroPython device'. Below it, a 'This computer' pane shows a list of files: 00.0_HelloWorld, 00.1_main, 01.1_Blink, and 01.2_Blink. The main editor window displays the code in main.py:

```
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
```

The bottom pane is the 'Shell' window, which shows the following output:

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

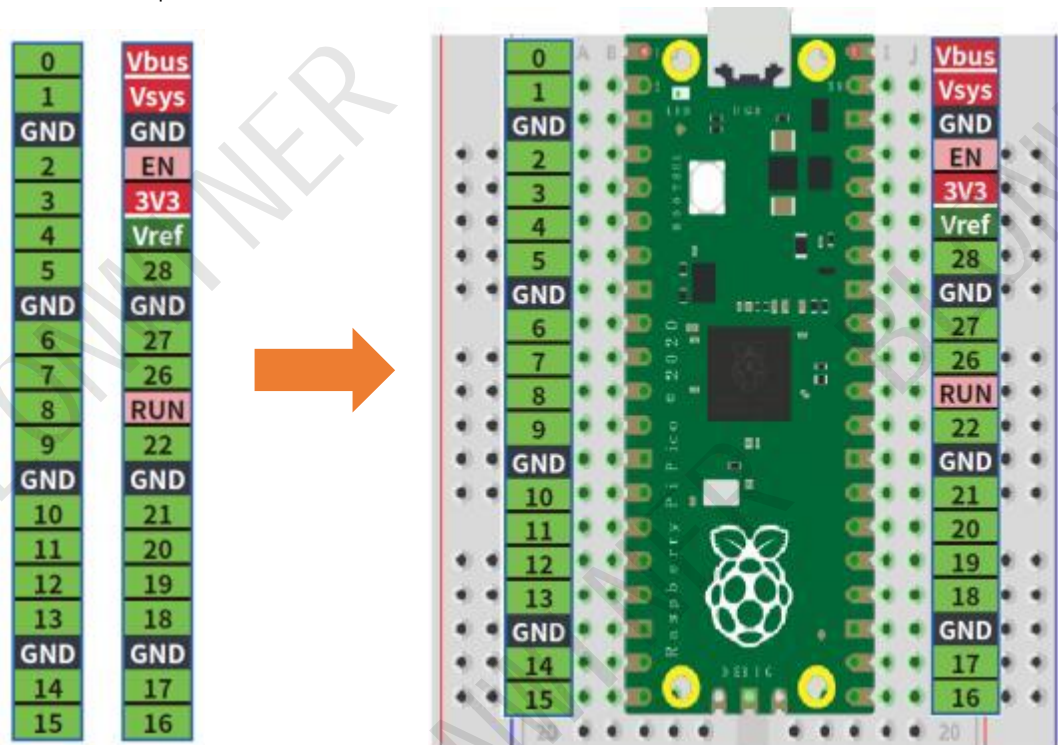
Click "Run" and the LED on Raspberry Pi Pico will blink periodically.



This screenshot is similar to the previous one, but with an orange callout bubble containing the word 'Click' pointing to the 'Run' button in the top toolbar. The code in the main editor is the same as in the first screenshot. The shell window shows the same output as before.

07. Paste the Sticker on the Breadboard

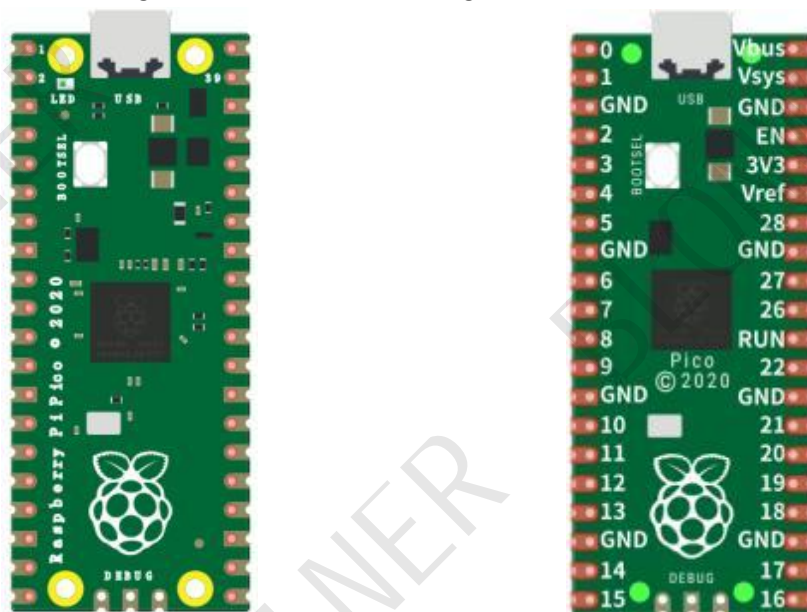
It is not difficult to use the Pico. However, officially, the pin functions are printed on the back of the board, which makes it inconvenient to use. To help users finish each project in the tutorial faster and easier, we provide stickers of the pin functions as follows:



You can paste the sticker on the blank area of the breadboard as above.

Note: The functional pin sequence of Pico and Pico W is the same. Therefore, even if your kit is Pico W, it is also applicable to the sticker above.

To make the tutorial more intuitive, we've made some changes to the simulation diagram as below. The left one is the actual Pico and the right one is its simulation diagram. Please note that to avoid misunderstanding.



Any concerns? ✉ blonwiner@outlook.com

Chapter 1 LED (Important)

Note: Raspberry Pi Pico and Raspberry Pi Pico W only differ by one wireless function, and are almost identical in other aspects. In this tutorial, except for the wireless function, other parts use Raspberry Pi Pico's map for tutorial demonstration.

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.

If you have not yet installed Thonny, click [here](#).

If you have not yet downloaded Micropython Firmware, click [here](#).

If you have not yet loaded Micropython Firmware, click [here](#).

Component List

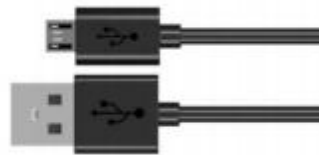
Raspberry Pi Pico (or Pico W) x1



or



USB cable x1



Power

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



Any concerns? ✉ blonwiner@outlook.com

Code

Codes used in this tutorial are saved in “BLONWINER_Breakout_Board_for_Raspberry_Pi_Pico\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”.

01.1_Blink

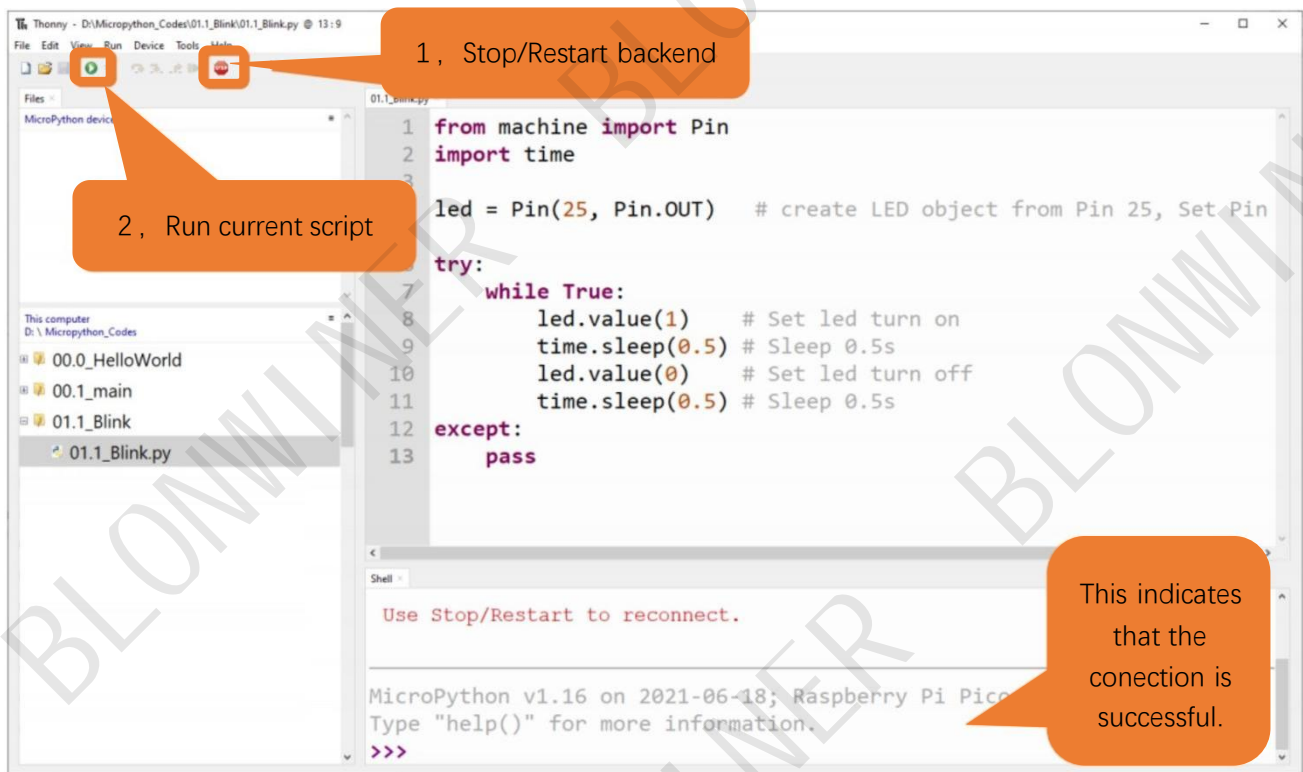
Open “Thonny”, click “This computer” → “D:” → “Micropython_Codes”.



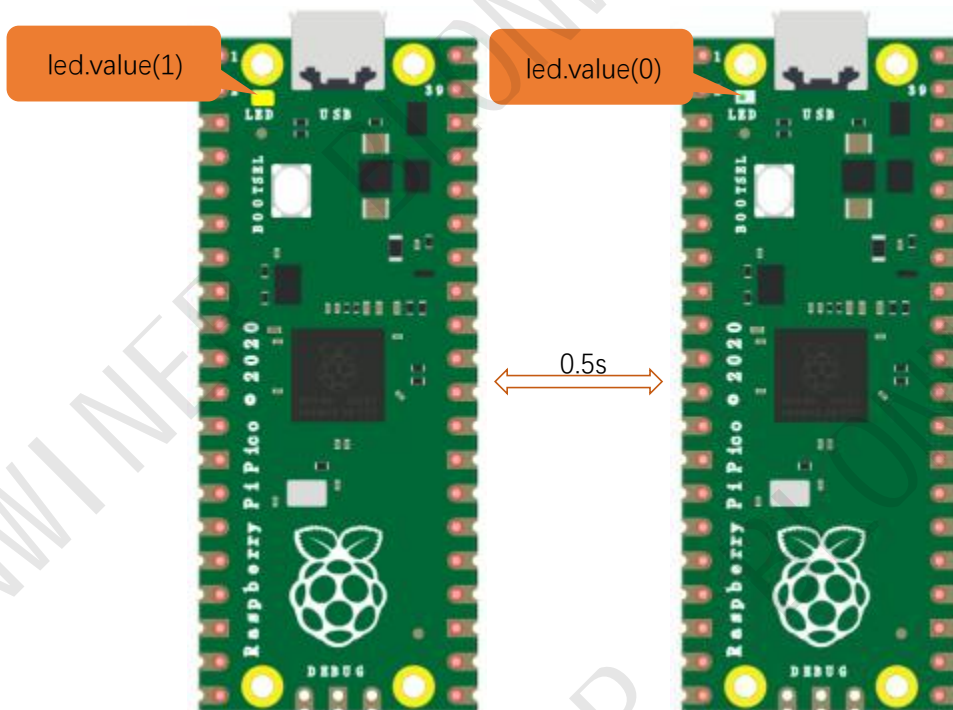
Expand folder “01.1_Blink” and double click “01.1_Blink.py” to open it. As shown in the illustration below.



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.

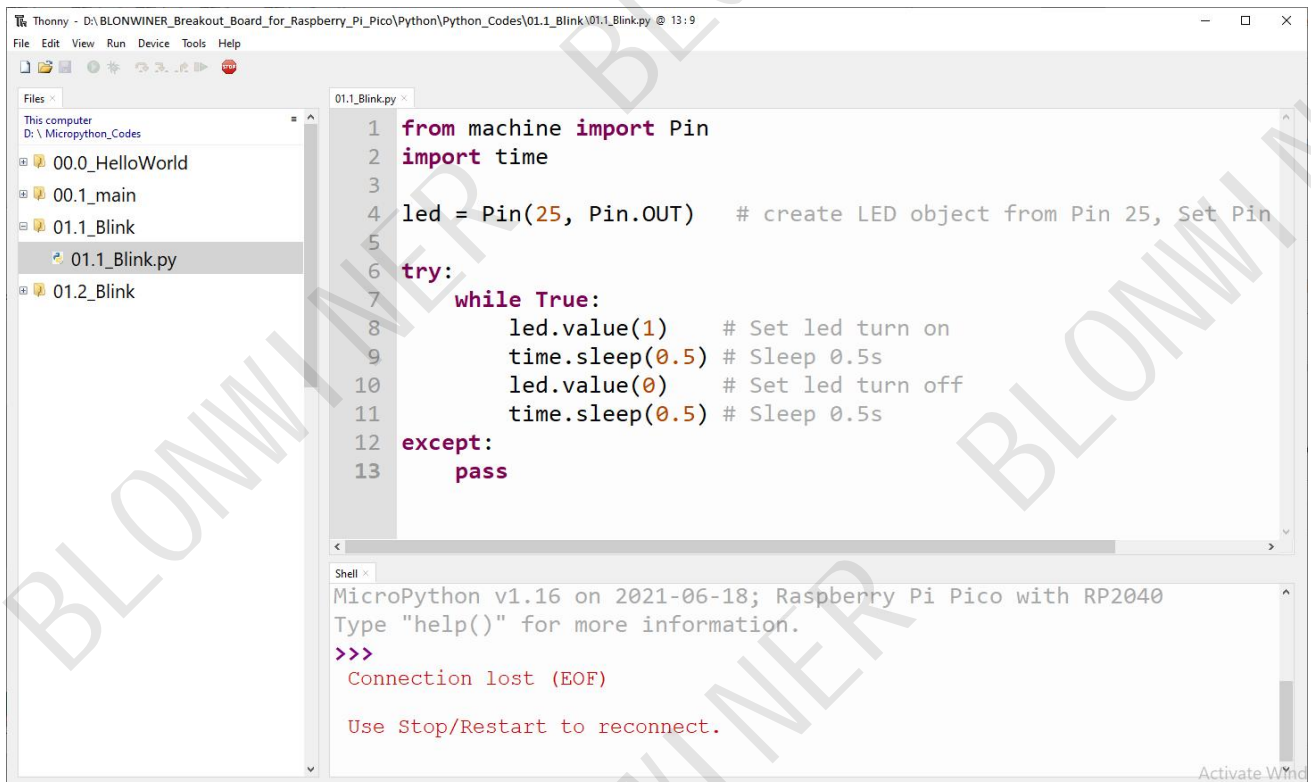


Click "Run current script" shown in the box above, the code starts to be executed and the LED in the circuit starts to blink. Press Ctrl+C or click "Stop/Restart backend" to exit the program.



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.



```

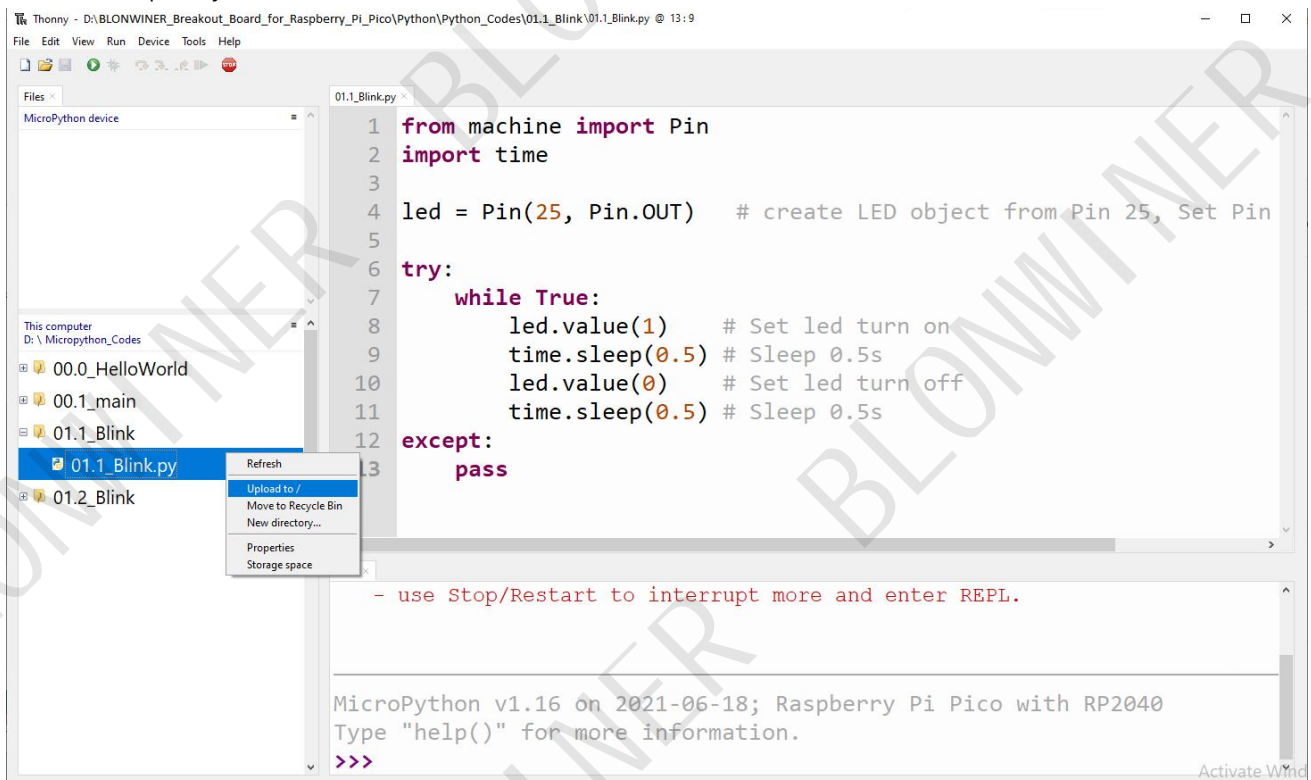
1 from machine import Pin
2 import time
3
4 led = Pin(25, Pin.OUT) # create LED object from Pin 25, Set Pin
5
6 try:
7     while True:
8         led.value(1) # Set led turn on
9         time.sleep(0.5) # Sleep 0.5s
10        led.value(0) # Set led turn off
11        time.sleep(0.5) # Sleep 0.5s
12 except: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

```

MicroPython v1.16 on 2021-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>>
Connection lost (EOF)
Use Stop/Restart to reconnect.

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.



```

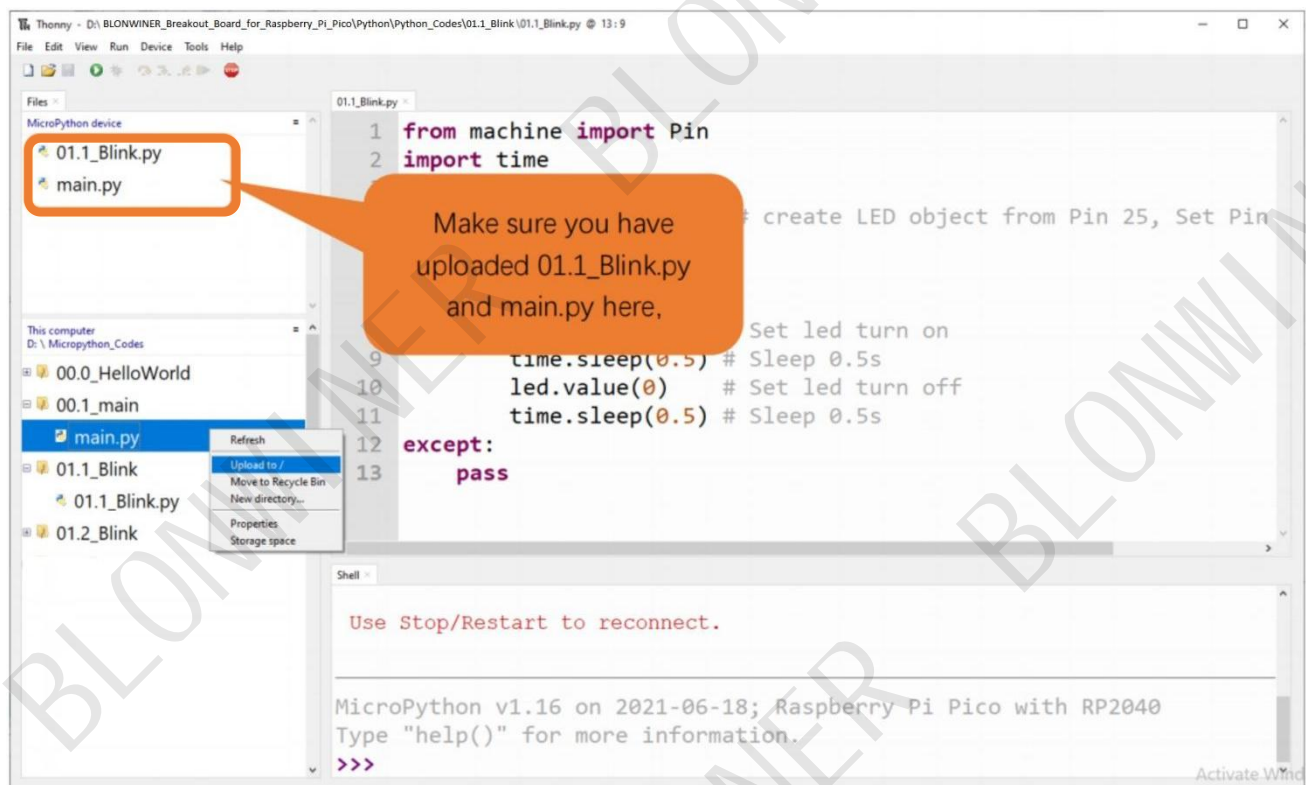
1 from machine import Pin
2 import time
3
4 led = Pin(25, Pin.OUT) # create LED object from Pin 25, Set Pin
5
6 try:
7     while True:
8         led.value(1) # Set led turn on
9         time.sleep(0.5) # Sleep 0.5s
10        led.value(0) # Set led turn off
11        time.sleep(0.5) # Sleep 0.5s
12 except: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

```

- use Stop/Restart to interrupt more and enter REPL.

MicroPython v1.16 on 2021-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>>

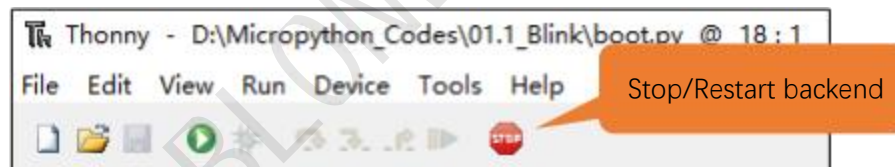
Upload main.py in the same way.



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.



If you have any concerns, please contact us via: blonwiner@outlook.com

Any concerns? ✉ blonwiner@outlook.com

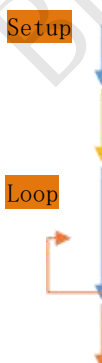
The following is the program code:

```

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

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

```

4  led = Pin(25, Pin.OUT)    #Pico
5  #led = Pin("LED", Pin.OUT) #Pico W

```

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 led pin 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

If you import the module directly you should indicate the module to which the function or attribute 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.

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

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:

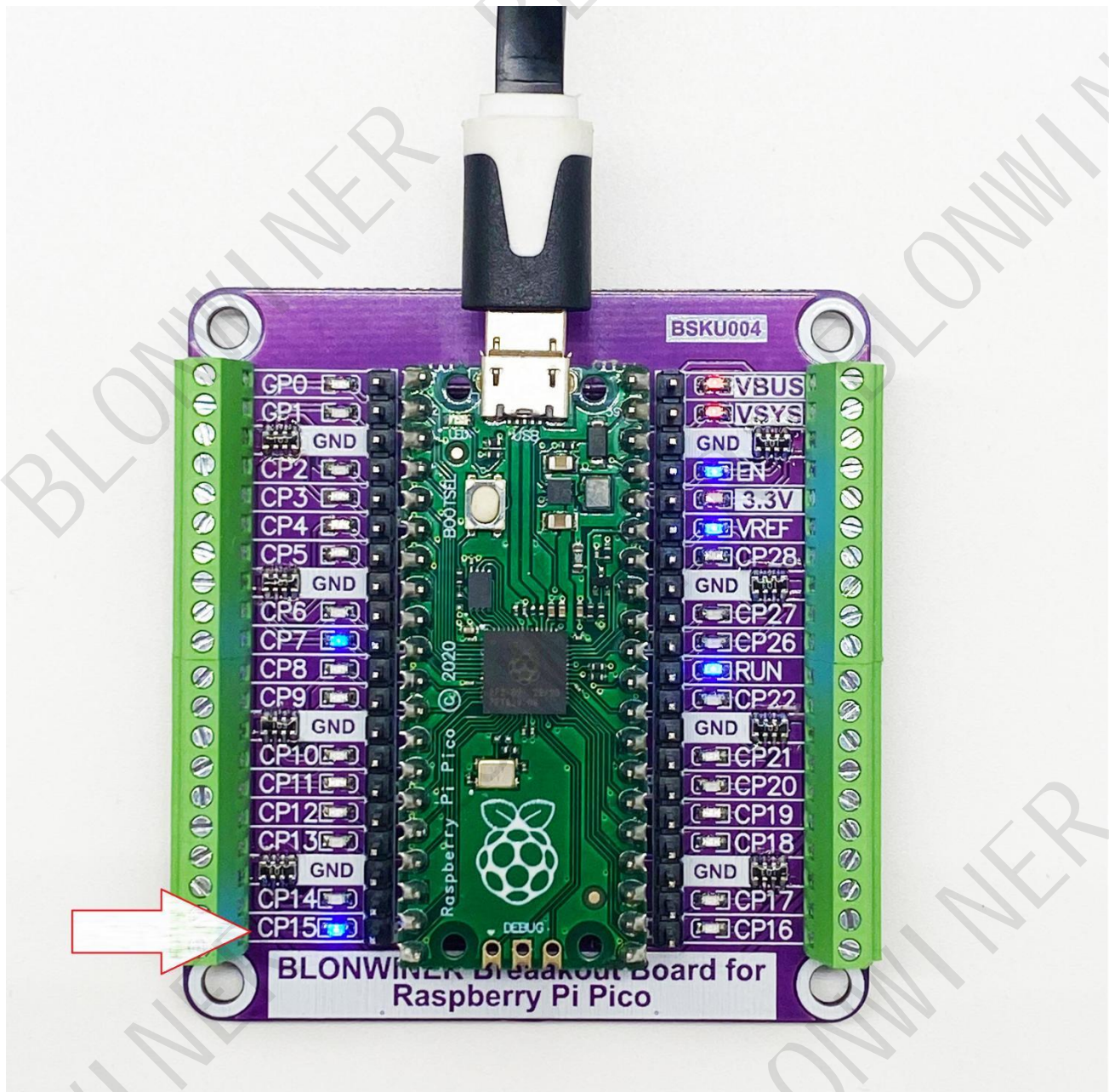
Pin.IRQ_FALLING: interrupt on falling edge.

Pin.IRQ_RISING: interrupt on rising edge.

Handler: callback function.

Project 1.2 Blink

In this project, we will make LED for GPIO15 blink..



Code

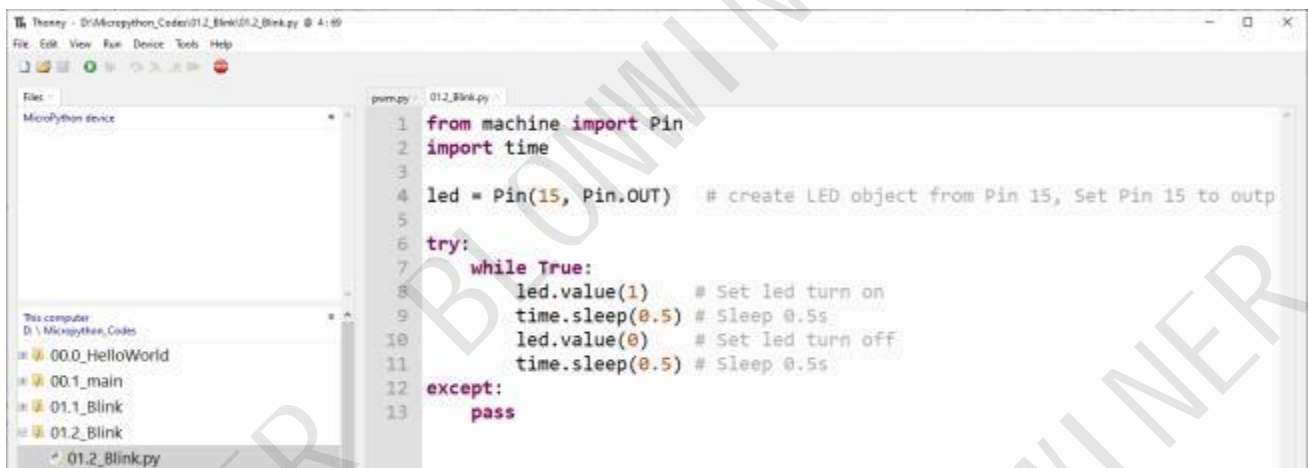
Codes used in this tutorial are saved in “**BLONWINER_Breakout_Board_for_Raspberry_Pi_Pico\Python\Python_Codes**”. You can move the codes to any location. For example, we save the codes in Disk(D) with the path of “**D:/Micropython_Codes**”.

01.2_Blink

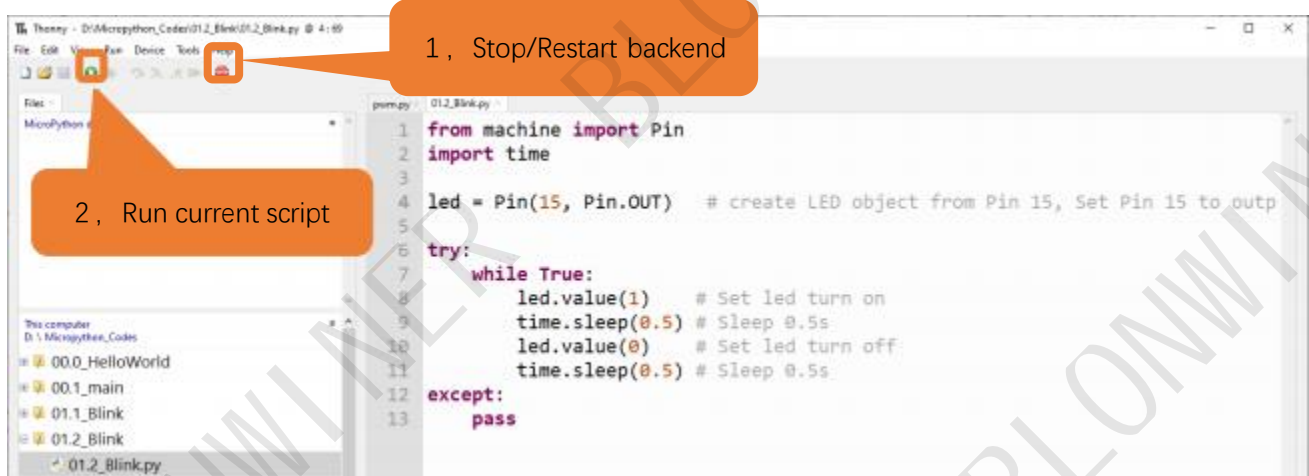
Open “Thonny”, click “This computer” → “D:” → “Micropython_Codes”.



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 then the LED for GPIO15 starts to blink. Press Ctrl+C or click “Stop/Restart backend” to exit the program.

Note:

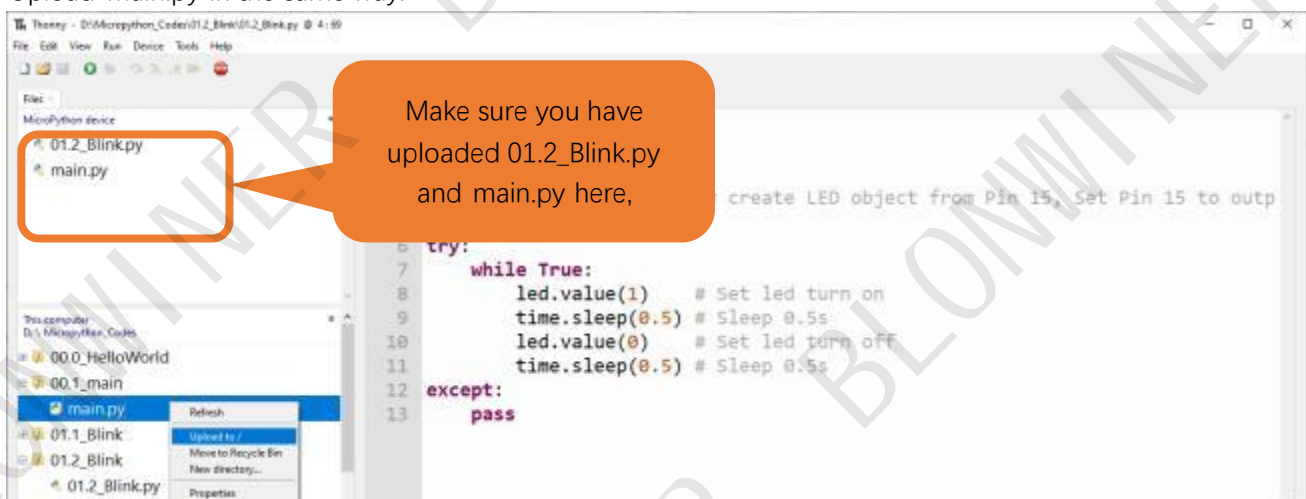
This is the code [running online](#). If you disconnect USB cable and repower Raspberry Pi Pico, LED stops blinking and the following messages will be displayed in Thonny.

**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.

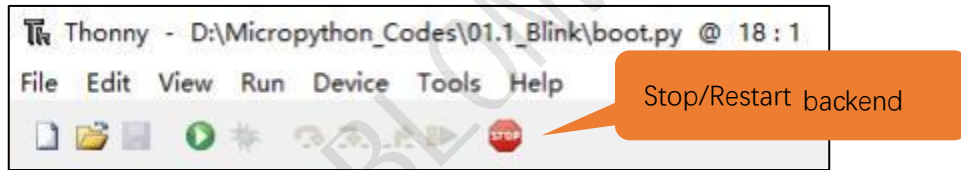


Disconnect Raspberry Pi Pico USB cable and reconnect it, LED on Pico will blink repeatedly.

Note:

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

Any concerns? [✉ blonwiner@outlook.com](mailto:blonwiner@outlook.com)



If you have any concerns, please contact us via: blonwiner@outlook.com

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:

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

www.123mygift.com

<https://www.amazon.com/s?srs=39109692011>

Thank you again for choosing BLONWINER products.

Any concerns? ✉ blonwiner@outlook.com