

Module 1.

MicroPython Basic

safyzan salim
019 622 0575



Th



Theoretically:

- ✓ To run Python code on an ESP32 microcontroller, you need to flash ESP32's MicroPython firmware.
- ✓ MicroPython ports Python 3.x, designed to run on microcontrollers, small embedded systems and is tailored for resource-constrained environments like the ESP32.
- ✓ Link to download ESP32's MicroPython firmware:
https://micropython.org/download/ESP32_GENERIC/
- ✓ Python IDE: Thonny IDE
<https://thonny.org/>



Listed are Arduino families that can be used for MicroPython:



Arduino Nano 33 BLE Sense
Arduino



Arduino Giga
Arduino



Arduino Nano ESP32
Arduino



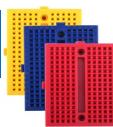
ESP32 / WROOM
Espressif



ESP8266
Espressif

>> Do refer at this link for further details:

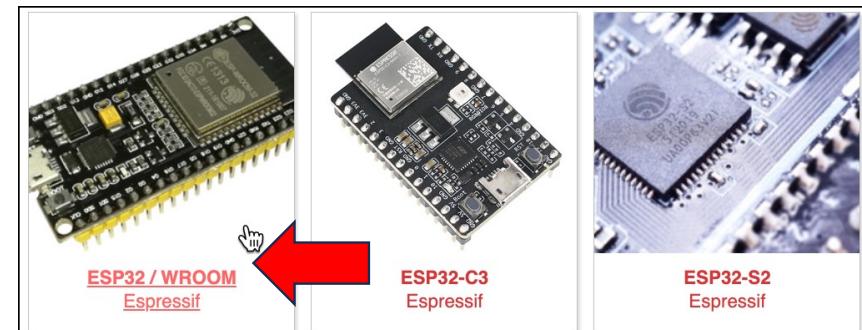
<https://micropython.org/download/>

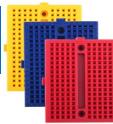


1. Downloading MicroPython Firmware.

The screenshot shows the official MicroPython website at micropython.org. The header includes the MicroPython logo, navigation links for DOWNLOAD, DOCS, DISCORD, DISCUSSIONS, WIKI, and STORE, and a search bar. The main content features a large image of a Pyboard microcontroller board. Text on the page describes MicroPython as a lean implementation of Python 3 for microcontrollers and constrained environments. It highlights the Pyboard, which runs MicroPython on bare metal and is compatible with normal Python. Buttons at the bottom include TEST DRIVE A PYBOARD, BUY A PYBOARD, and USE MICROPYTHON ONLINE.

- Download from <https://micropython.org/download>
- Select the appropriate board.





1. Downloading MicroPython Firmware.

Firmware

Releases

[v1.20.0 \(2023-04-26\) .bin \[.elf\] \[.map\] \[Release notes\] \(latest\)](#)
[v1.19.1 \(2022-06-18\) .bin \[.elf\] \[.map\] \[Release notes\]](#)
[v1.18 \(2022-01-17\) .bin \[.elf\] \[.map\] \[Release notes\]](#)
[v1.17 \(2021-09-02\) .bin \[.elf\] \[.map\] \[Release notes\]](#)
[v1.16 \(2021-06-23\) .bin \[.elf\] \[.map\] \[Release notes\]](#)
[v1.15 \(2021-04-18\) .bin \[.elf\] \[.map\] \[Release notes\]](#)
[v1.14 \(2021-02-02\) .bin \[.elf\] \[.map\] \[Release notes\]](#)
[v1.13 \(2020-09-02\) .bin \[.elf\] \[.map\] \[Release notes\]](#)
[v1.12 \(2019-12-20\) .bin \[.elf\] \[.map\] \[Release notes\]](#)

Nightly builds

[v1.20.0-448-g5e5059373 \(2023-09-05\) .bin \[.app-bin\] \[.elf\] \[.map\]](#)
[v1.20.0-445-g8bd2494c9 \(2023-09-04\) .bin \[.app-bin\] \[.elf\] \[.map\]](#)
[v1.20.0-442-gd00105494 \(2023-09-04\) .bin \[.app-bin\] \[.elf\] \[.map\]](#)
[v1.20.0-441-gbf35eefc6 \(2023-09-04\) .bin \[.app-bin\] \[.elf\] \[.map\]](#)



- Choose any dot bin version **releases** firmware. e.g.: **v1.19.1(2022-06-18).bin**
- Nightly builds are unstable with fresh features and bugs not thoroughly tested.
- Normally, select **one or two** version lower from the recent version.

Firmware (ESP32 Unicore)

Releases

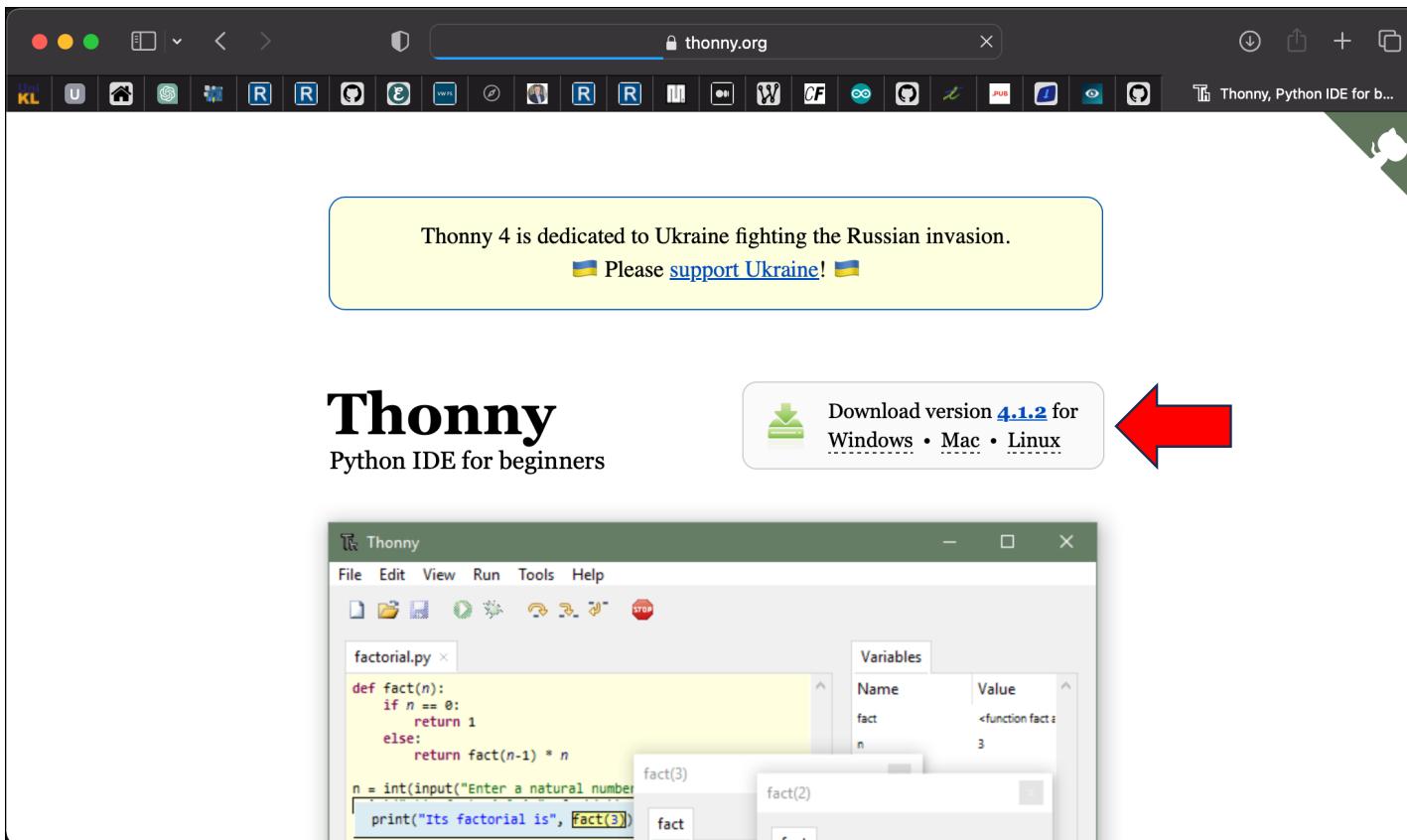
[v1.20.0 \(2023-04-26\) .bin \[.elf\] \[.map\] \[Release notes\] \(latest\)](#)

Nightly builds

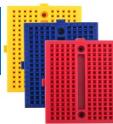
[v1.20.0-448-g5e5059373 \(2023-09-05\) .bin \[.app-bin\] \[.elf\] \[.map\]](#)
[v1.20.0-445-g8bd2494c9 \(2023-09-04\) .bin \[.app-bin\] \[.elf\] \[.map\]](#)
[v1.20.0-442-gd00105494 \(2023-09-04\) .bin \[.app-bin\] \[.elf\] \[.map\]](#)
[v1.20.0-441-gbf35eefc6 \(2023-09-04\) .bin \[.app-bin\] \[.elf\] \[.map\]](#)



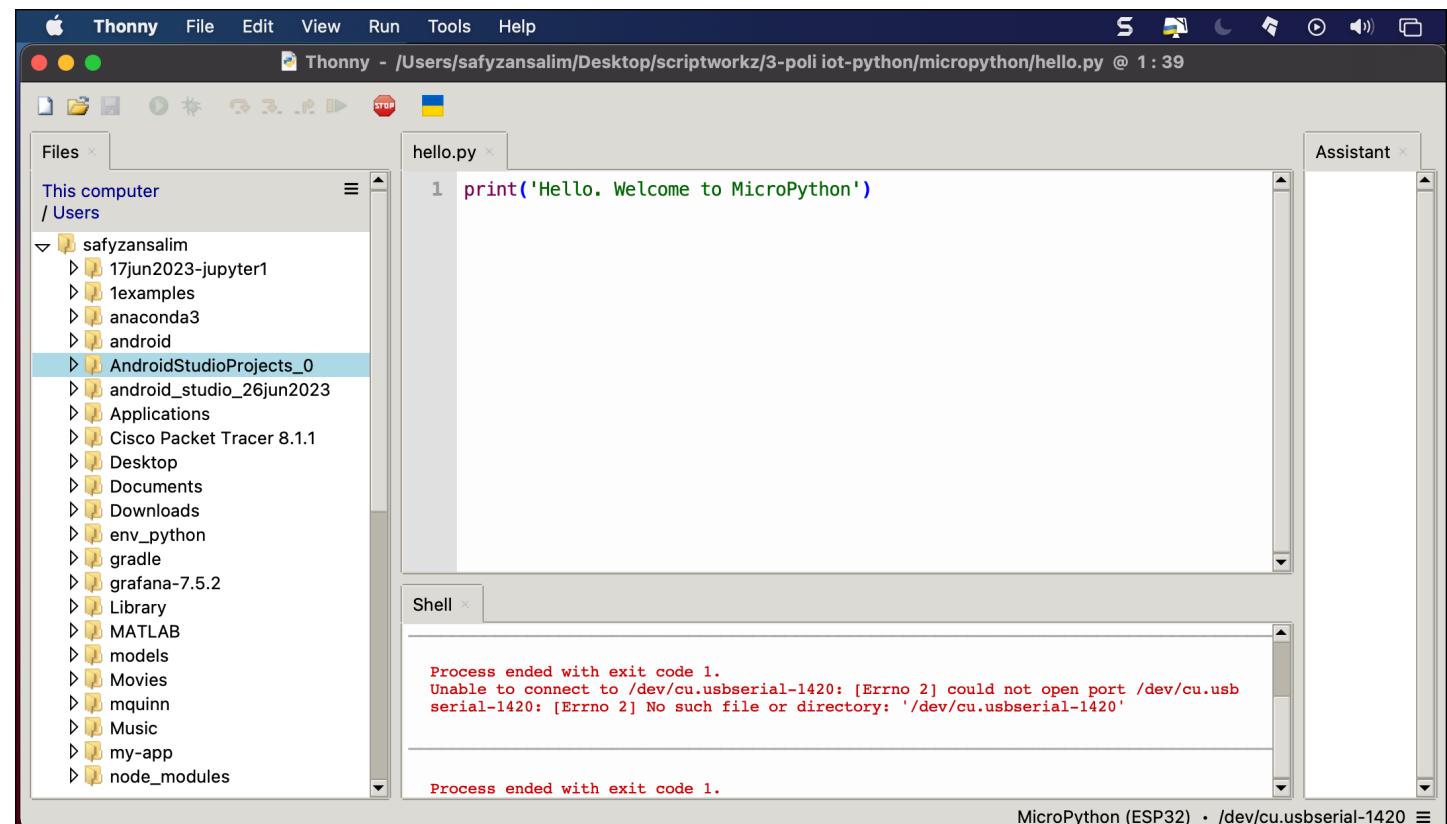
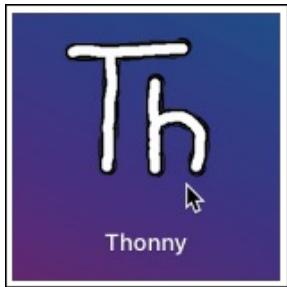
2. Thonny Python IDE: Download & Install



- Download from <https://thonny.org/>
- Thonny is a beginner friendly Python IDE.
- To erase & flush new firmware to ESP32S with the help of ESP tool.
- Act as Python editor.

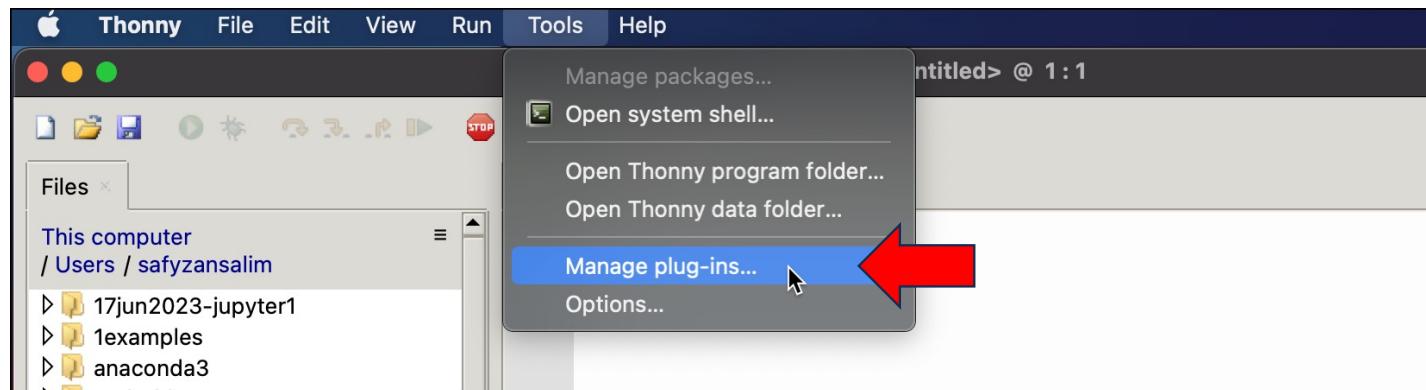


2. Thonny Python IDE: Run Thonny





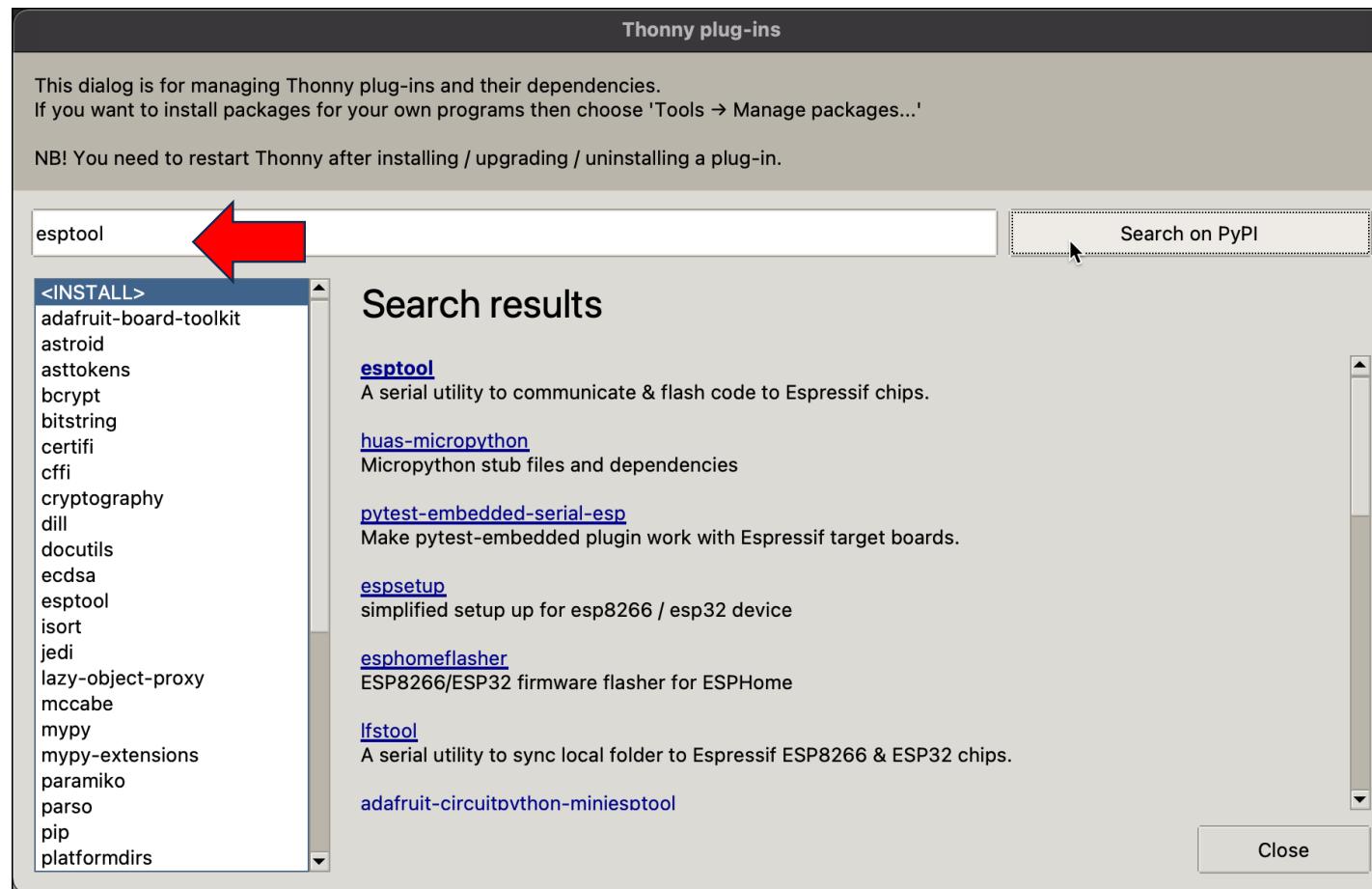
2. Thonny Python IDE: Flushing New ESP32 Firmware.



- Install plug-ins first.
- **Tools > Manage plug-ins.**
- **Connect the ESP32S to your system.**



2. Thonny Python IDE: Flushing New ESP32 Firmware.



- Type **esptool** & click Search on PyPi button.
- Click **esptool** to install.



2. Thonny Python IDE: Flushing New ESP32 Firmware.

Thonny plug-ins

This dialog is for managing Thonny plug-ins and their dependencies.
If you want to install packages for your own programs then choose 'Tools → Manage packages...'.

NB! You need to restart Thonny after installing / upgrading / uninstalling a plug-in.

esptool

Search on PyPI

<INSTALL>
adafruit-board-toolkit
astroid
asttokens
bcrypt
bitstring
certifi
cffi
cryptography
dill
docutils
ecdsa
esptool
isort
jedi
lazy-object-proxy
mccabe
mypy
mypy-extensions
paramiko
parso
pip
platformdirs

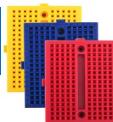
esptool

Installed version: 4.6.2
Installed to:
/Applications/Thonny.app/Contents/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages

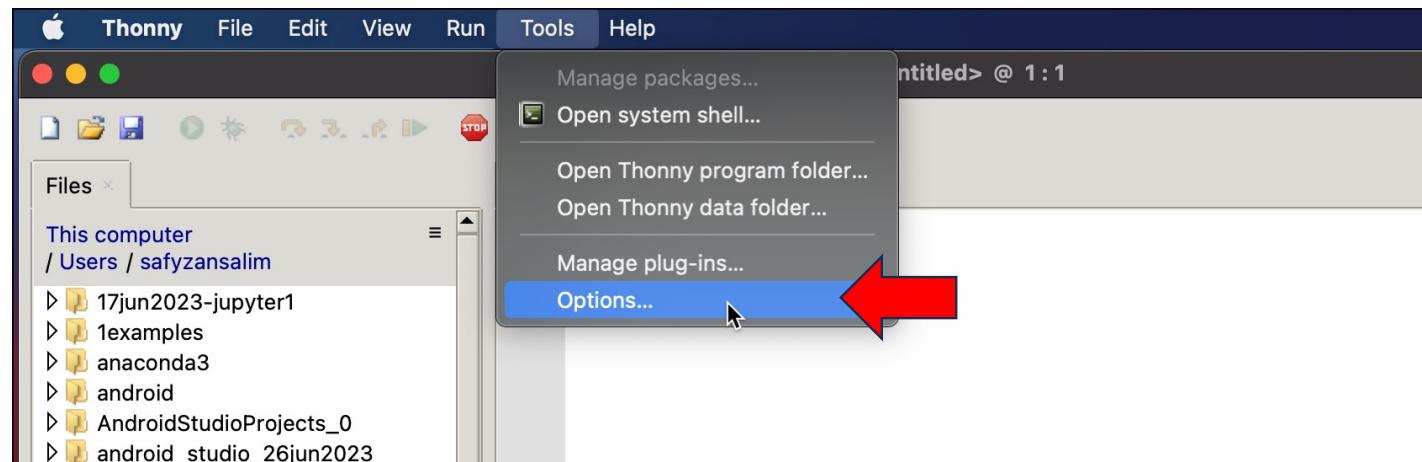
Close



- o **esptool installed.**



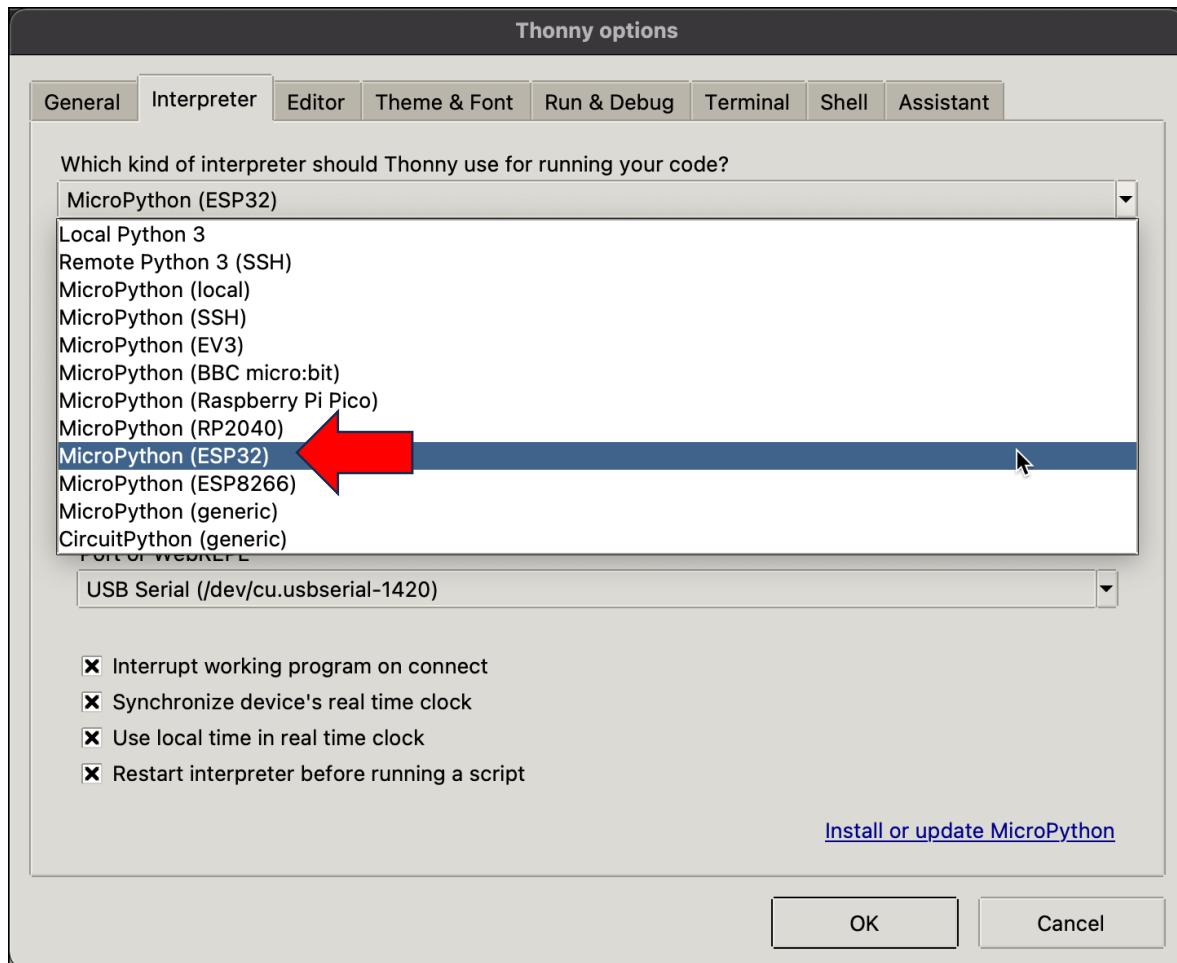
2. Thonny Python IDE: Flushing New ESP32 Firmware.



- To install Firmware.
- **Tools > Options**



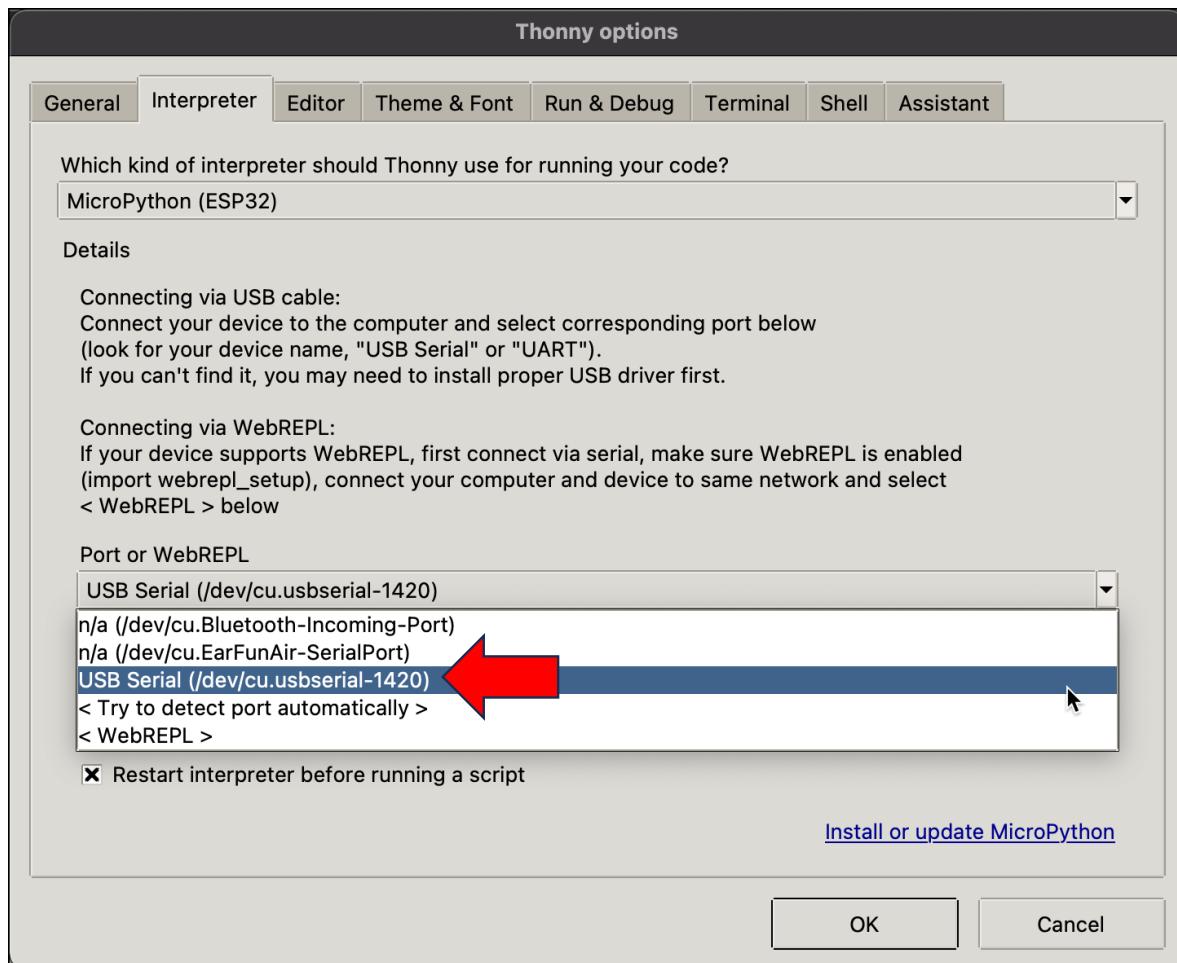
2. Thonny Python IDE: Flushing New ESP32 Firmware.



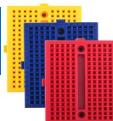
- Go to **Interpreter** tab.
- Select **MicroPython (ESP32)**.



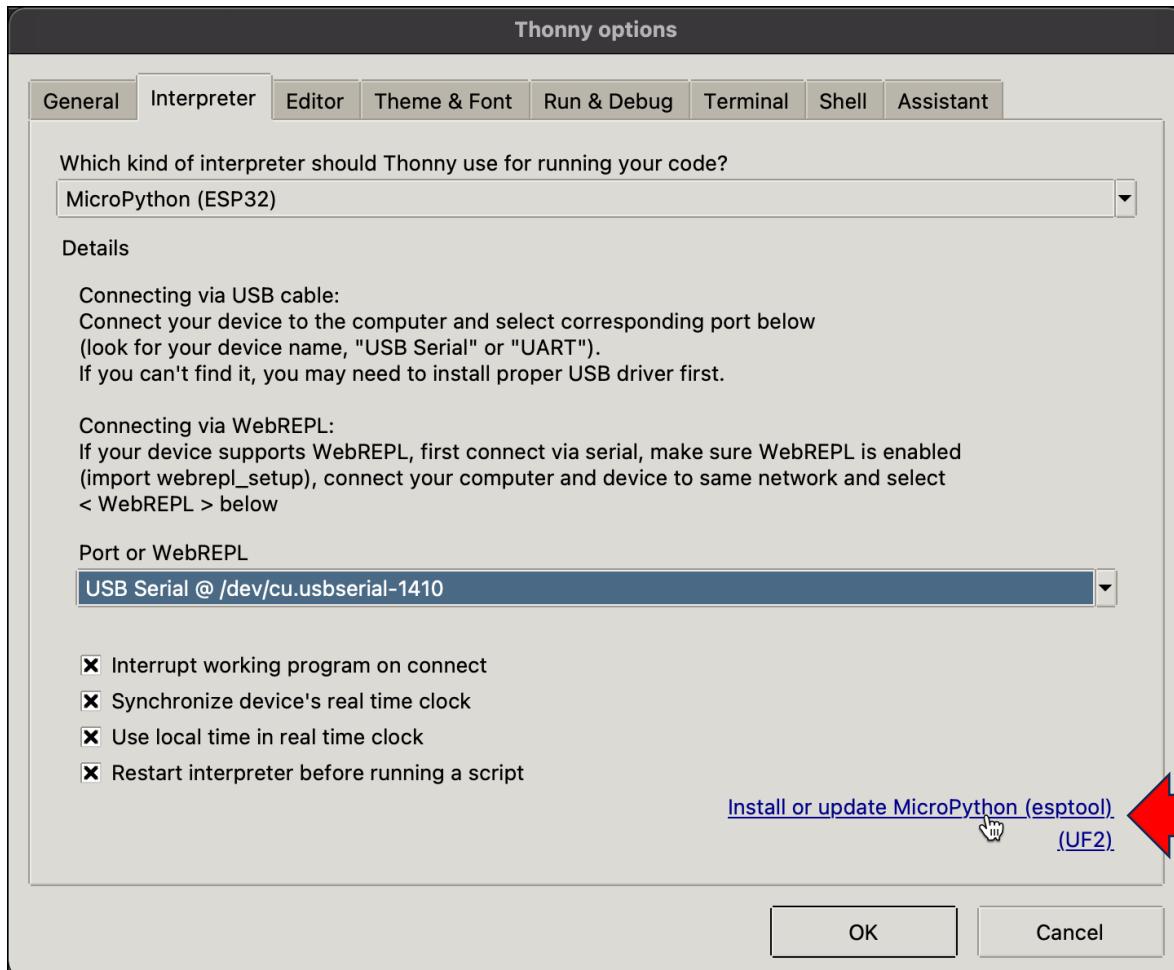
2. Thonny Python IDE: Flushing New ESP32S Firmware.



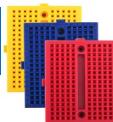
- **Port:** Choose the appropriate comm port of your ESP32S.
- **If port not detected, check your ESP32S connection to your laptop/PC.**



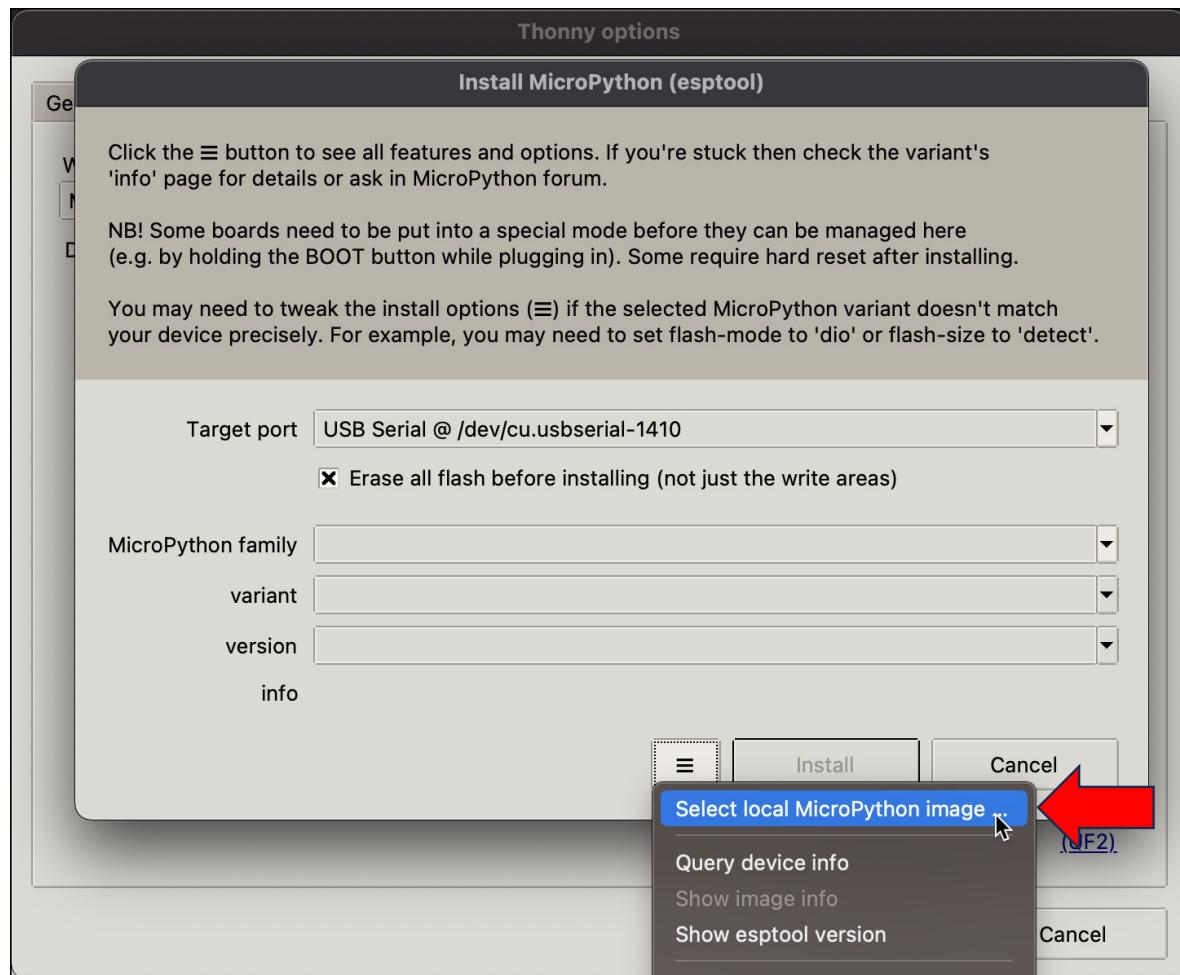
2. Thonny Python IDE: Flushing New ESP32 Firmware.



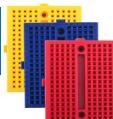
- Click Install or Update Micropython link.



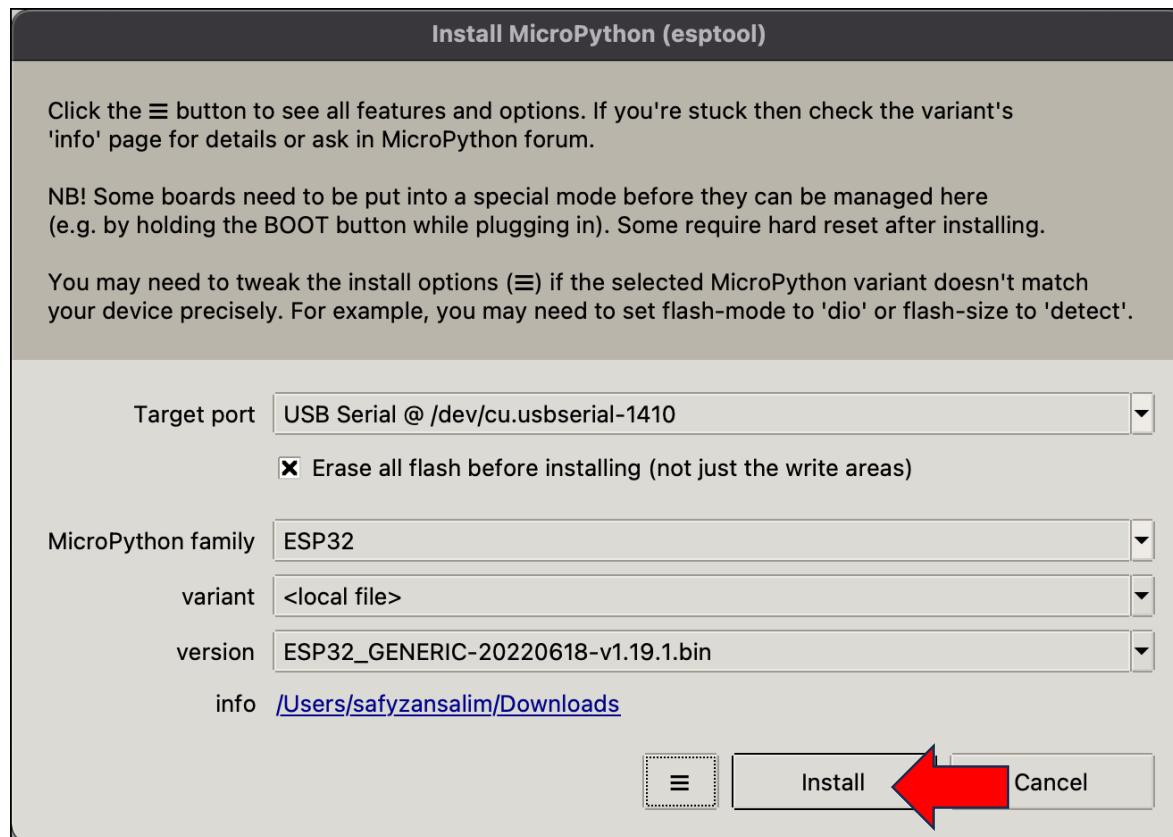
2. Thonny Python IDE: Flushing New ESP32S Firmware.



- **Target Port:** Which port the EPS32S is connected.
-  : browse the downloaded **dot bin** file from micropython.org.
- **Ensure that the Erase flash before installing is checked.**



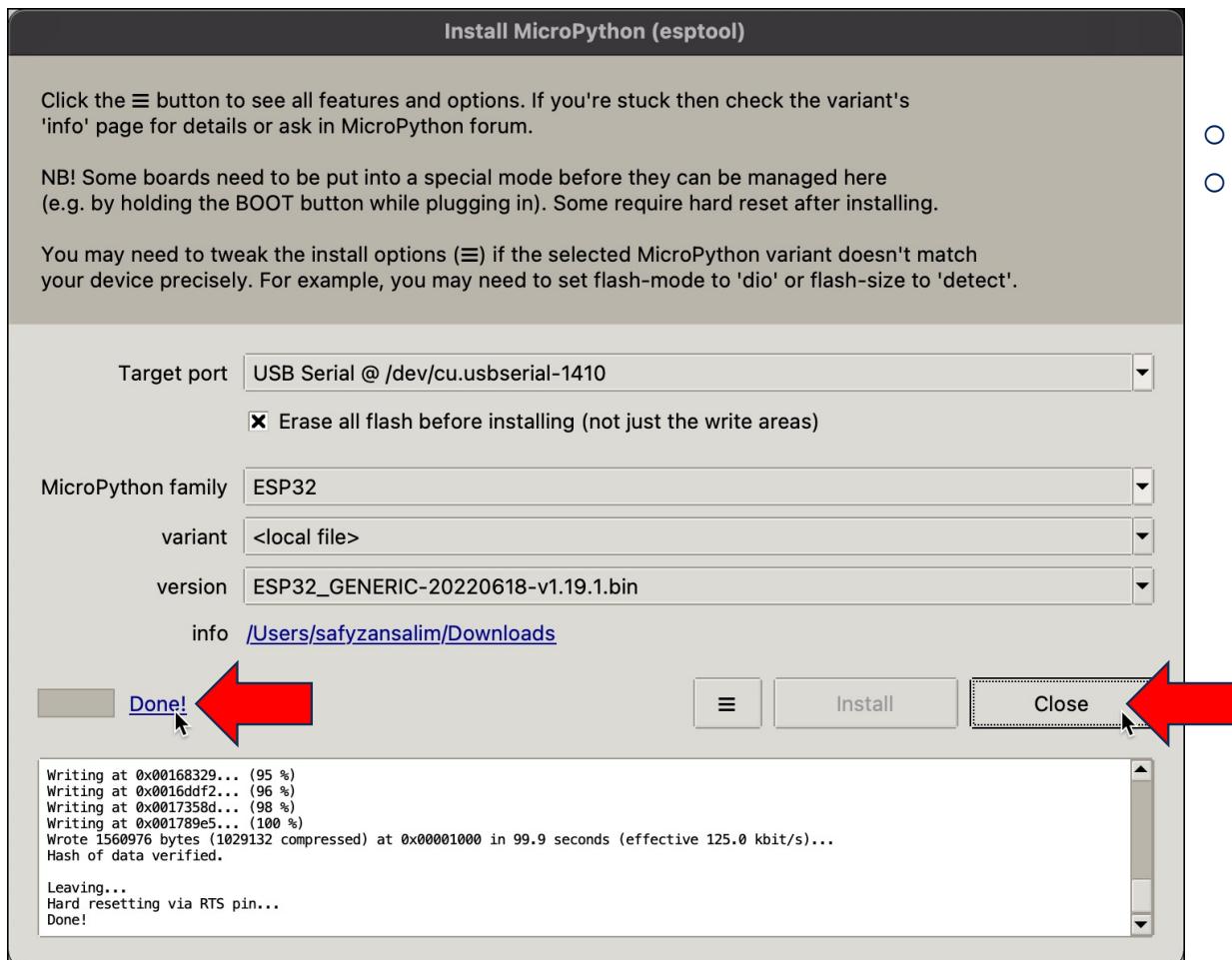
2. Thonny Python IDE: Flushing New ESP32S Firmware.



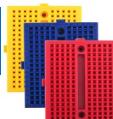
- Proceed with clicking **Install** button.



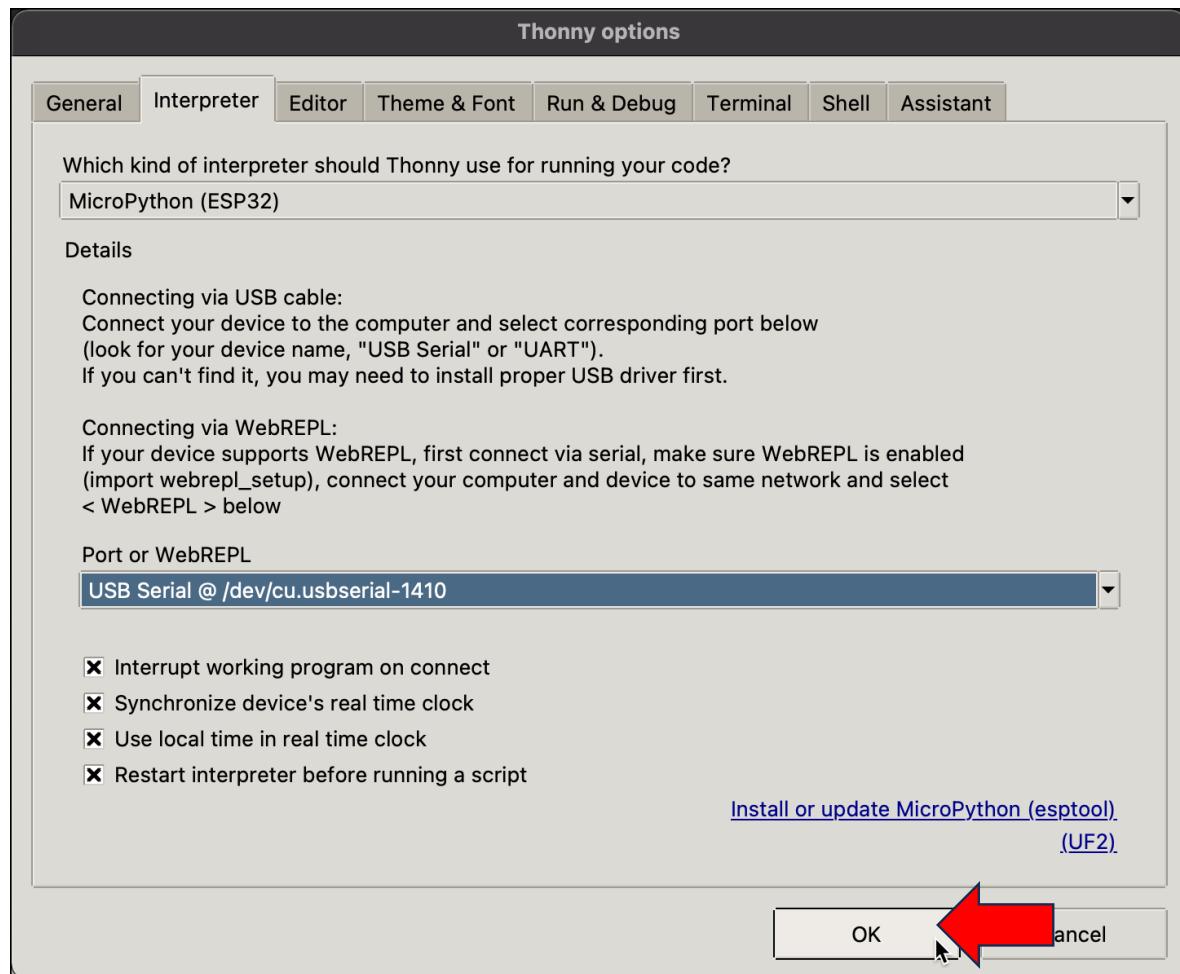
2. Thonny Python IDE: Flushing New ESP32S Firmware.



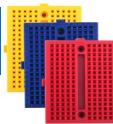
- Click **Done!** to view detail installation.
- Click **Close** button to go to previous page.



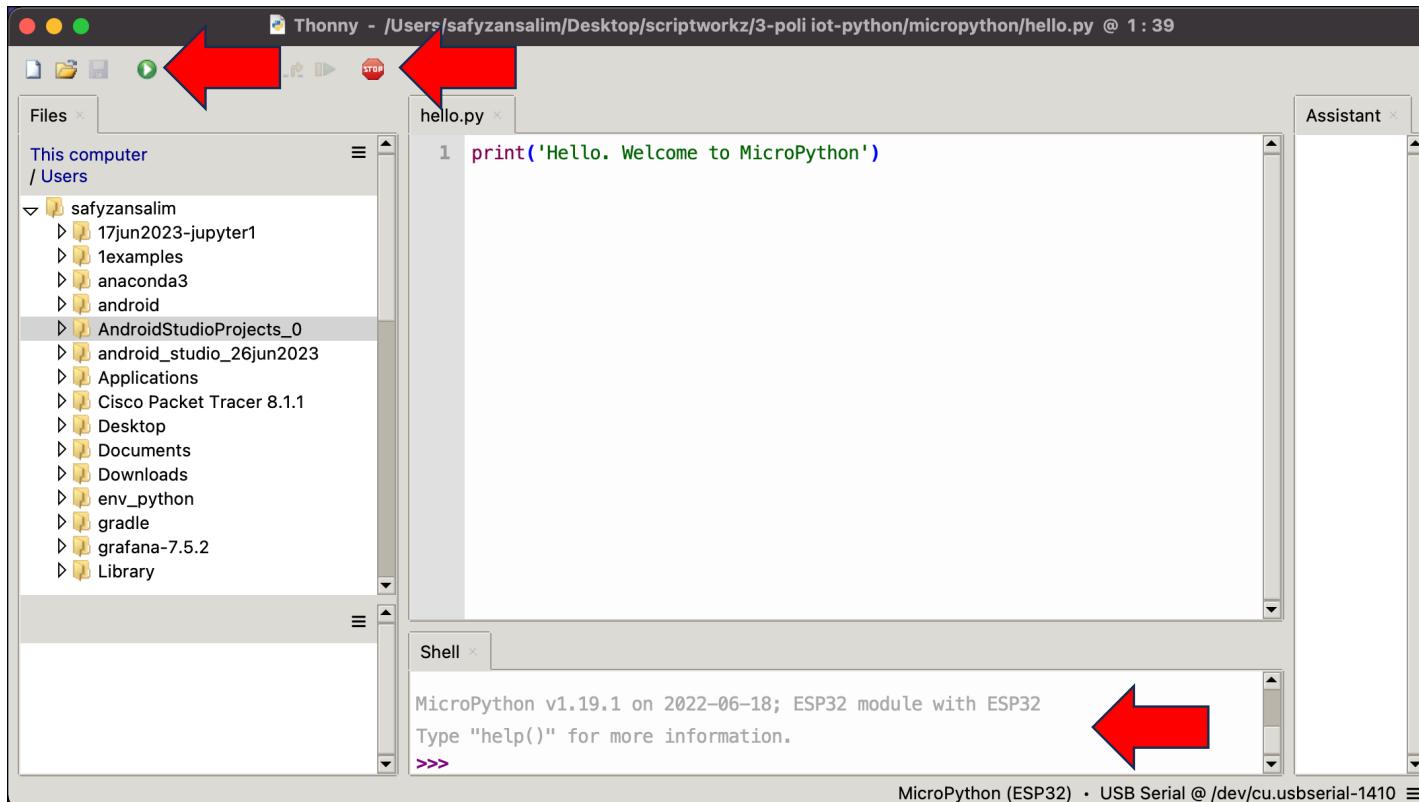
2. Thonny Python IDE: Flushing New ESP32S Firmware.



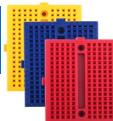
- Click **OK** button upon complete installation.



2. Thonny Python IDE: Flushing New ESP32S Firmware.



- Scripts that type at Shell, will respond immediately at ESP32.
- To stop click **Stop** button at the top.
- Click Run button to resume operation.



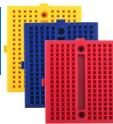
2. Thonny Python IDE: Point to **Shell** tab

```
import machine  
led = machine.Pin(2, machine.Pin.OUT)  
led.on()  
led.off()  
led.value(1)  
led.value(0)  
led.value(true)  
led.value(false)
```

- Type the Python script line by line & press enter.
- Please refer to ESP32S & observe the output .

```
>>> import machine  
>>> led = machine.Pin(2, machine.Pin.OUT)  
>>> led.on()  
>>> led.off()  
>>> led.value(1)  
>>> led.value(0)  
>>> led.value(true)
```

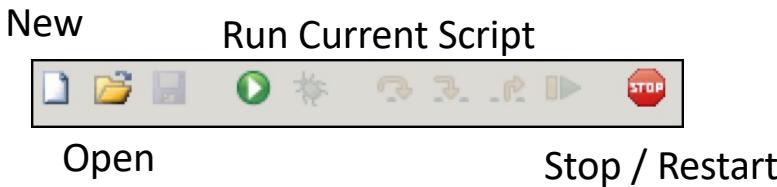
MicroPython (ESP32) • USB Serial @ /dev/cu.usbserial-1410



2. Thonny Python IDE: Point to Shell Tab or A Tab

```
import machine
led = machine.Pin(2, machine.Pin.OUT)
import time
while True:
    led.on()
    time.sleep(0.5)
    led.off()
    time.sleep(0.5)
```

- Type at **A** if you wish to save the file



The screenshot shows the Thonny Python IDE interface. On the left is the file browser ('Files') showing a directory structure under 'This computer / Users'. In the center is the script editor ('hello.py') containing the following code:

```
1 print('Hello. Welcome to MicroPython')
```

Below the script editor is the 'Shell' tab, which contains the same code as the script:>>> import machine

led = machine.Pin(2, machine.Pin.OUT)

import time

while True:

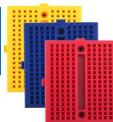
 led.on()

 time.sleep(0.5)

 led.off()

 time.sleep(0.5)

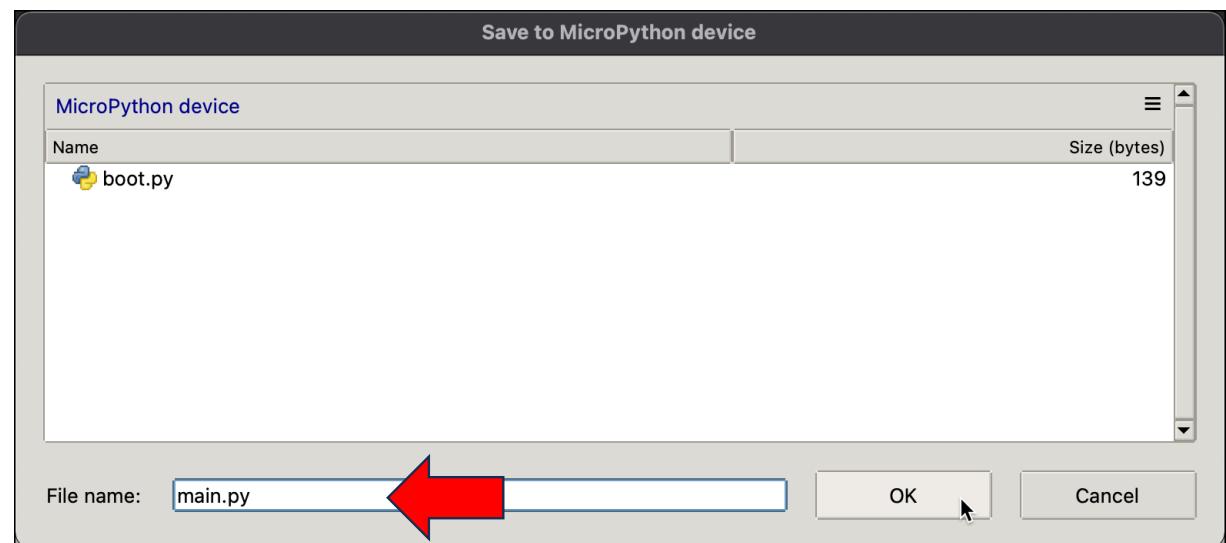
The status bar at the bottom indicates 'MicroPython (ESP32) • USB Serial @ /dev/cu.usbserial-1410'.



2. Thonny Python IDE: Saving Your Works



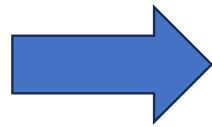
- You have two options to save your file: This computer or MicroPython device.
- Name your file as **main.py** if you wish to save into MicroPython board & any name if you wish to save into your drive.



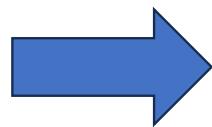


2. Thonny Python IDE: Saving Your Works

Your laptop
/ PC



Your
microcontroller



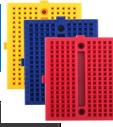
The screenshot shows the Thonny Python IDE interface. On the left, there's a 'Files' sidebar showing local files like 'AndroidStudioProjects_0' and a 'MicroPython device' sidebar showing files 'boot.py' and 'main.py'. The main area has tabs for 'main.py' (containing code to toggle an LED every 0.5 seconds) and 'Shell'. The 'Shell' tab displays an error message about connecting to a port and a traceback indicating a 'KeyboardInterrupt' on the MicroPython device. The bottom status bar says 'MicroPython (ESP32) • USB Serial @ /dev/cu.usbserial-1410'.

```
import machine
led = machine.Pin(2, machine.Pin.OUT)
import time
while True:
    led.on()
    time.sleep(0.5)
    led.off()
    time.sleep(0.5)
```

Unable to connect to /dev/cu.usbserial-1410: [Errno 2] could not open port /dev/cu.usbserial-1410: [Errno 2] No such file or directory: '/dev/cu.usbserial-1410'
Process ended with exit code 1.

Traceback (most recent call last):
 file "main.py", line 8, in <module>
KeyboardInterrupt:
MicroPython v1.19.1 on 2022-06-18; ESP32 module with ESP32
Type "help()" for more information.
MPY: soft reboot
MicroPython v1.19.1 on 2022-06-18; ESP32 module with ESP32
Type "help()" for more information.
>>>

MicroPython (ESP32) • USB Serial @ /dev/cu.usbserial-1410



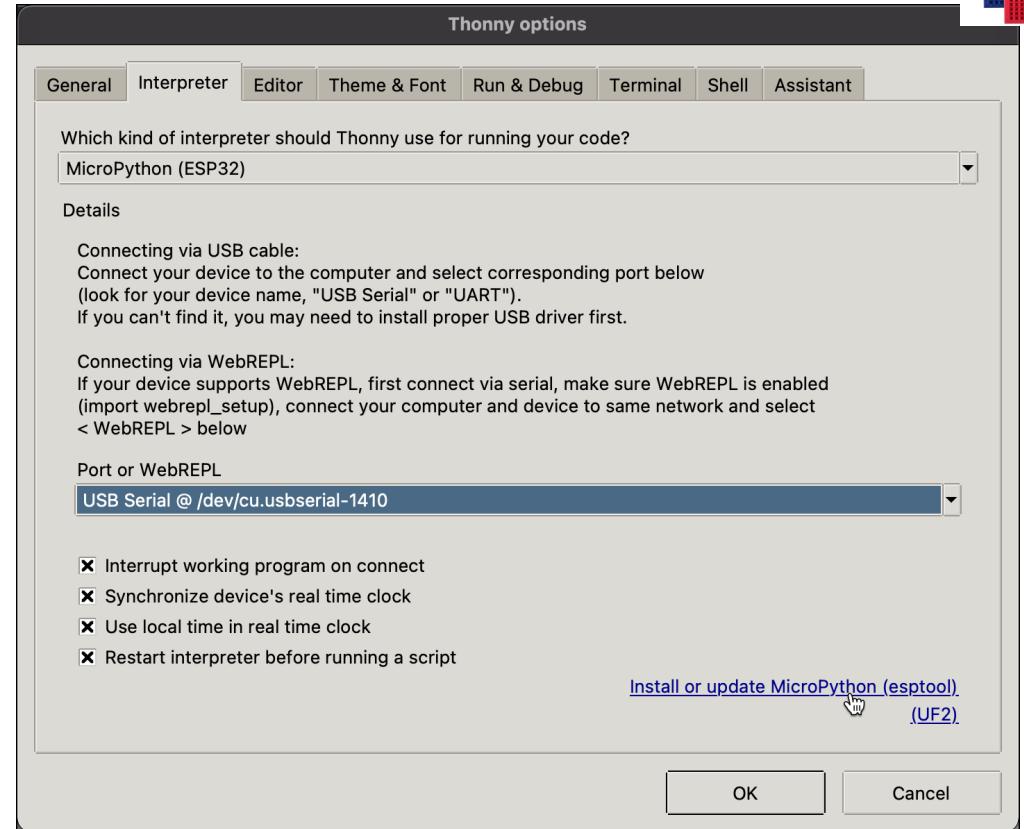
Remember:

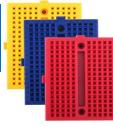
1. Always ensure that you have selected the right Comm Port to communicate with ESP32.

Tools>Option>Interpreter Tab

2. Only upload MicroPython firmware once, unless you have uploaded Arduino ESP32 firmware, then you have to click the Install or update MicroPython link.

3. Open Arduino IDE, select ESP32 board, its port & upload Arduino sketch as usual.





END