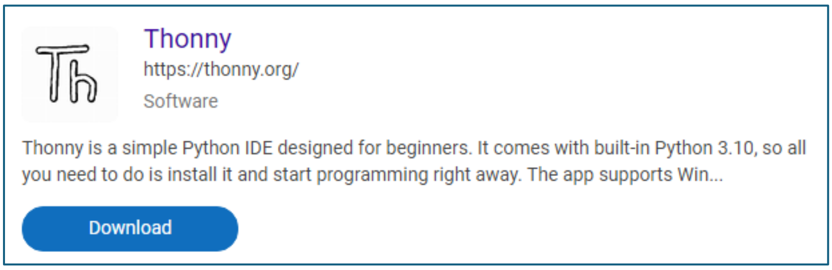
We are going to install Thonny and then test it. Start a browser and go to:



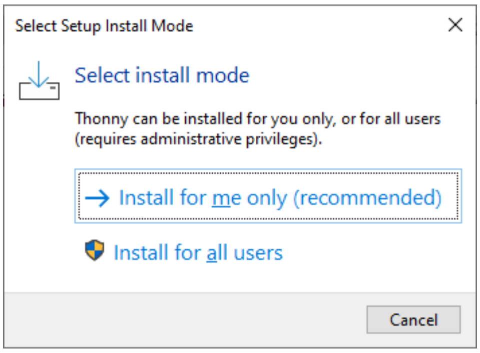
Thonny

https://thonny.org/

Software

Thonny is a simple Python IDE designed for beginners. It comes with built-in Python 3.10, so all you need to do is install it and start programming right away. The app supports Win...

I downloaded and started the Windows version:

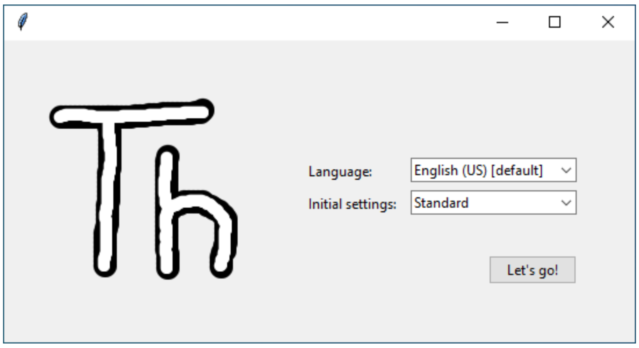


Select Setup Install Mode

Select install mode

Thonny can be installed for you only, or for all users (requires administrative privileges).

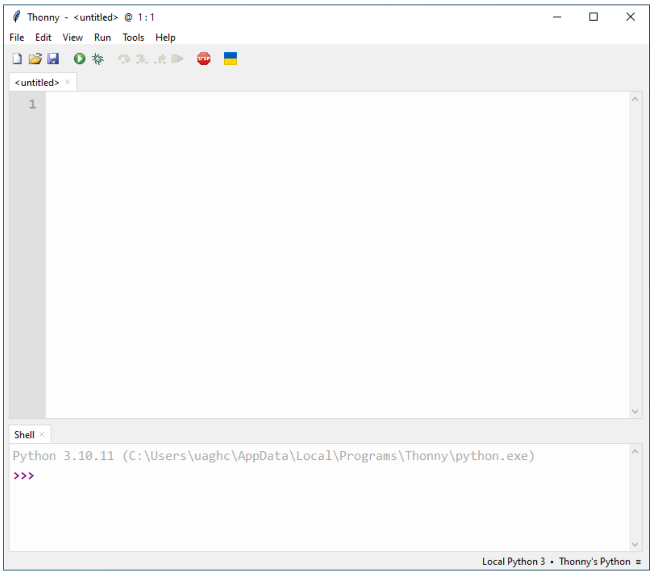
Now you just have to wait. Start Thonny after it is installed, do not change any settings, and click 'Let's go!':



Language: English (US) [default]

Initial settings: Standard

Thonny starts up, there is nothing to see yet. The bottom part shows which Python interpreter is active, at the moment it is the local Python 3.10 interpreter:

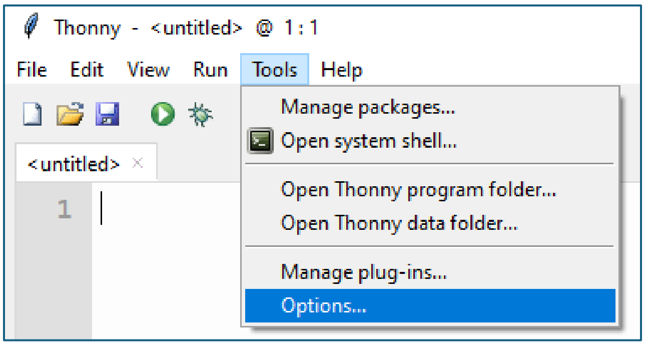


Thonny - <untitled> @ 1 : 1

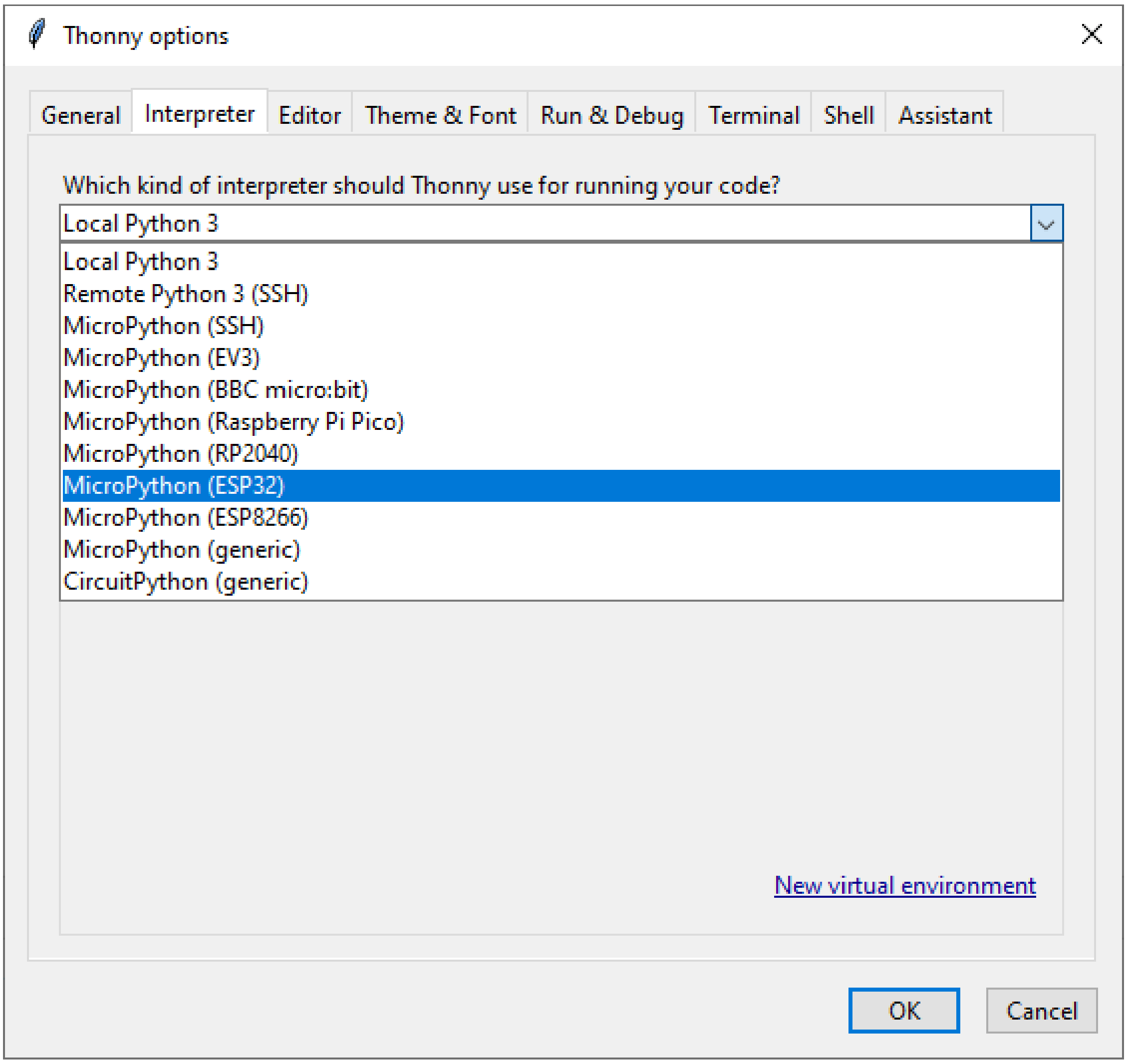
```
Python 3.10.11 (C:\Users\uaghc\AppData\Local\Programs\Thonny\python.exe)
>>>
```

Local Python 3 • Thonny's Python

We will now first set up Thonny for MicroPython. Go to "Tools", "Options":



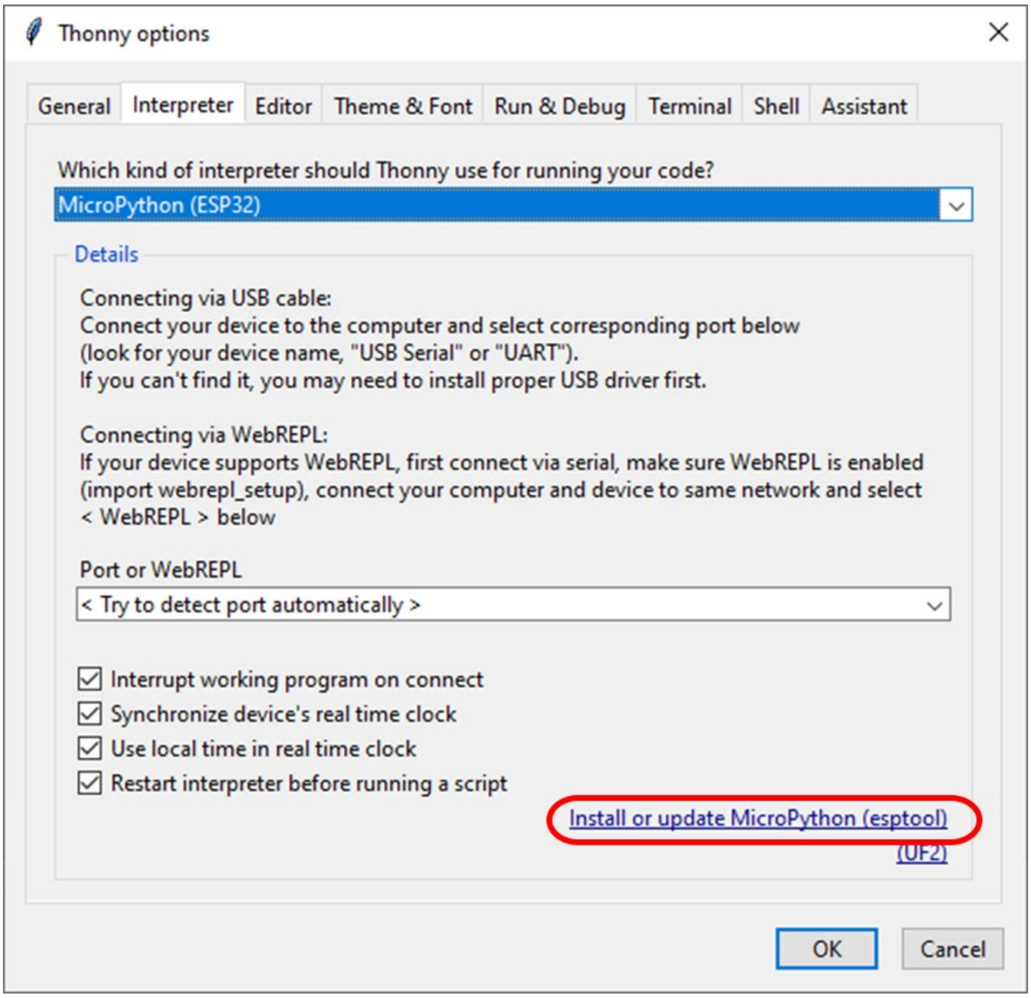
Go to the "Interpreter" tab and choose "MicroPython (ESP32)":



Now you’re on the "Interpreter" tab, what you need to depends on whether or not MicroPython is already installed on the ESP32-C3. For now click “OK”.

If MicroPython is already installed on the ESP32-C3 the Thonny “shell” window will show the details of the active MicroPython interpreter (see picture below the “Thonny options”, “Interpreter” image).

If you have a brand new ESP32-C3, return to "Tools", "Options", “Interpreter” tab: "Install or update Micropython (esptool)":



Which kind of interpreter should Thonny use for running your code?

MicroPython (ESP32)

Details

Connecting via USB cable:
Connect your device to the computer and select corresponding port below
(look for your device name, "USB Serial" or "UART").
If you can't find it, you may need to install proper USB driver first.

Connecting via WebREPL:
If your device supports WebREPL, first connect via serial, make sure WebREPL is enabled
(import webrepl_setup), connect your computer and device to same network and select
< WebREPL > below

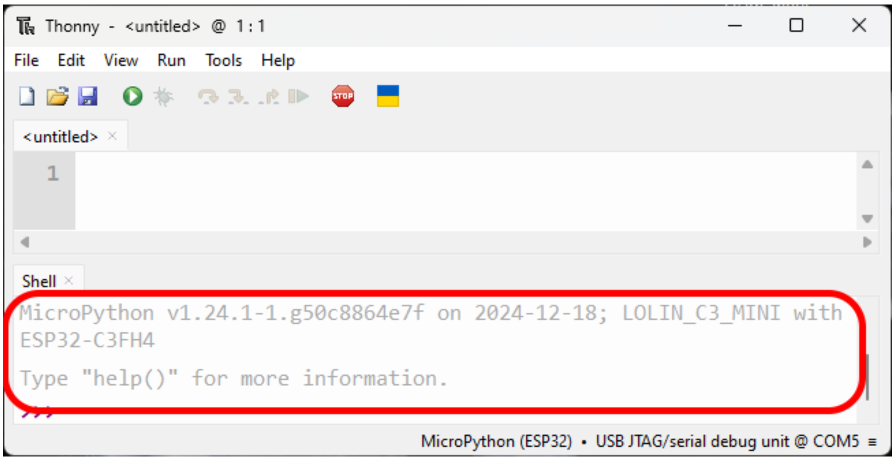
Port or WebREPL

< Try to detect port automatically >

☑ Interrupt working program on connect
☑ Synchronize device's real time clock
☑ Use local time in real time clock
☑ Restart interpreter before running a script

Install or update MicroPython (esptool)
(UF2)

An ESP32-C3 with MicroPython installed show this:

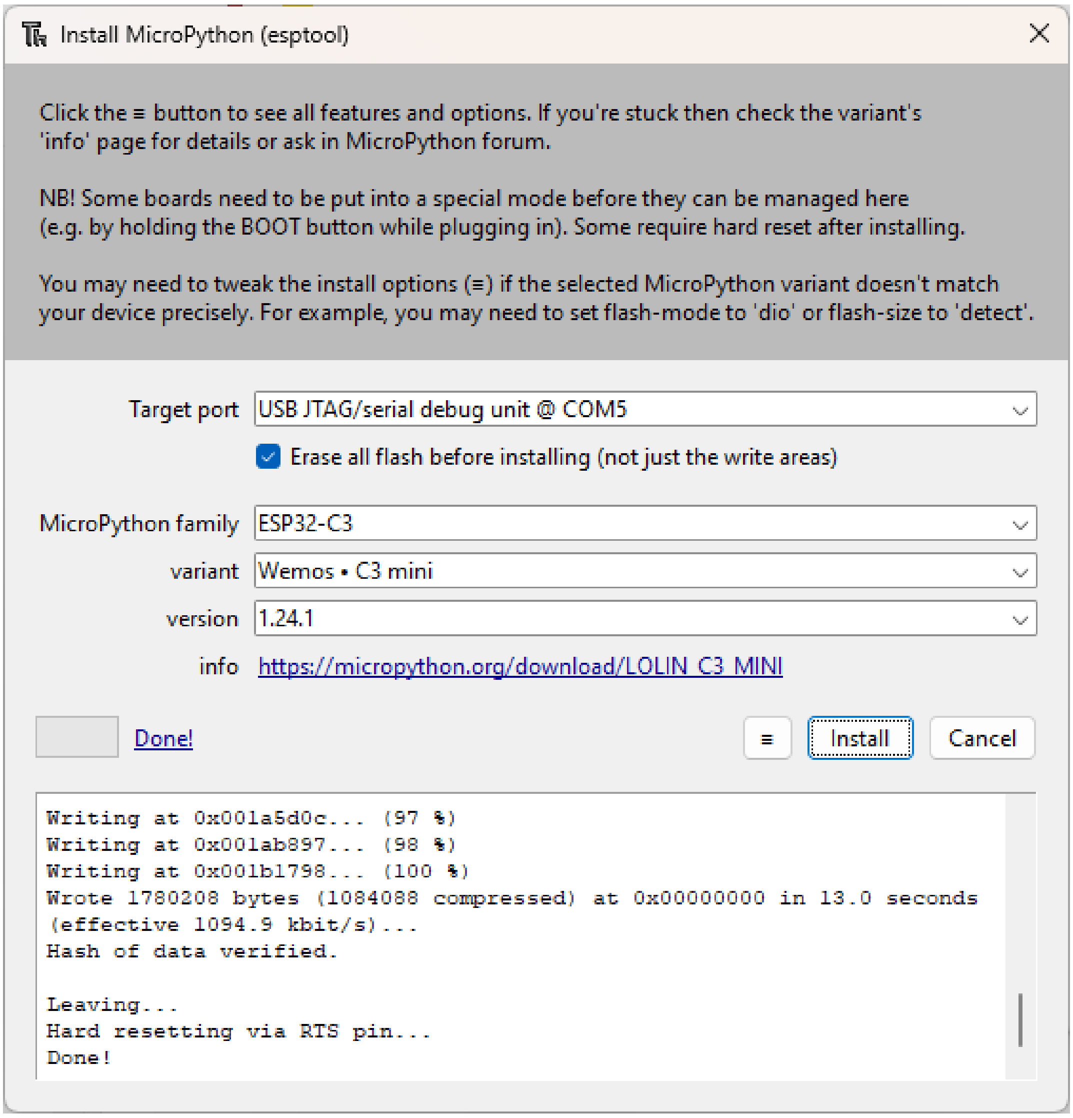


```
MicroPython v1.24.1-1.g50c8864e7f on 2024-12-18; LOLIN_C3_MINI with
ESP32-C3FH4

Type "help()" for more information.
>>>
```

MicroPython (ESP32) • USB JTAG/serial debug unit @ COM5

If you need to install MicroPython, choose the following settings (the version may be newer) and click "Install" (this picture is the result):



Install MicroPython (esptool)

Click the ≡ button to see all features and options. If you're stuck then check the variant's 'info' page for details or ask in MicroPython forum.

NB! Some boards need to be put into a special mode before they can be managed here (e.g. by holding the BOOT button while plugging in). Some require hard reset after installing.

You may need to tweak the install options (≡) if the selected MicroPython variant doesn't match your device precisely. For example, you may need to set flash-mode to 'dio' or flash-size to 'detect'.

Target port: USB JTAG/serial debug unit @ COM5

☑ Erase all flash before installing (not just the write areas)

MicroPython family: ESP32-C3

variant: Wemos • C3 mini

version: 1.24.1

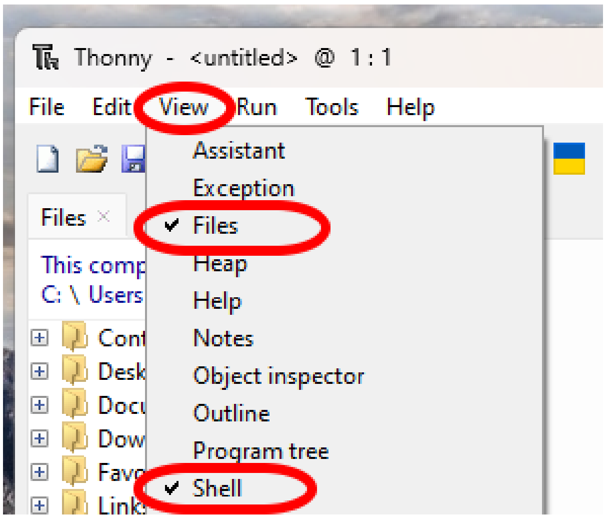
info: https://micropython.org/download/LOLIN_C3_MINI

Done!

```
Writing at 0x001a5d0c... (97 %)
Writing at 0x001ab897... (98 %)
Writing at 0x001b1798... (100 %)
Wrote 1780208 bytes (1084088 compressed) at 0x00000000 in 13.0 seconds
(effective 1094.9 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
Done!
```

Close the window and change the following in the "View" menu:

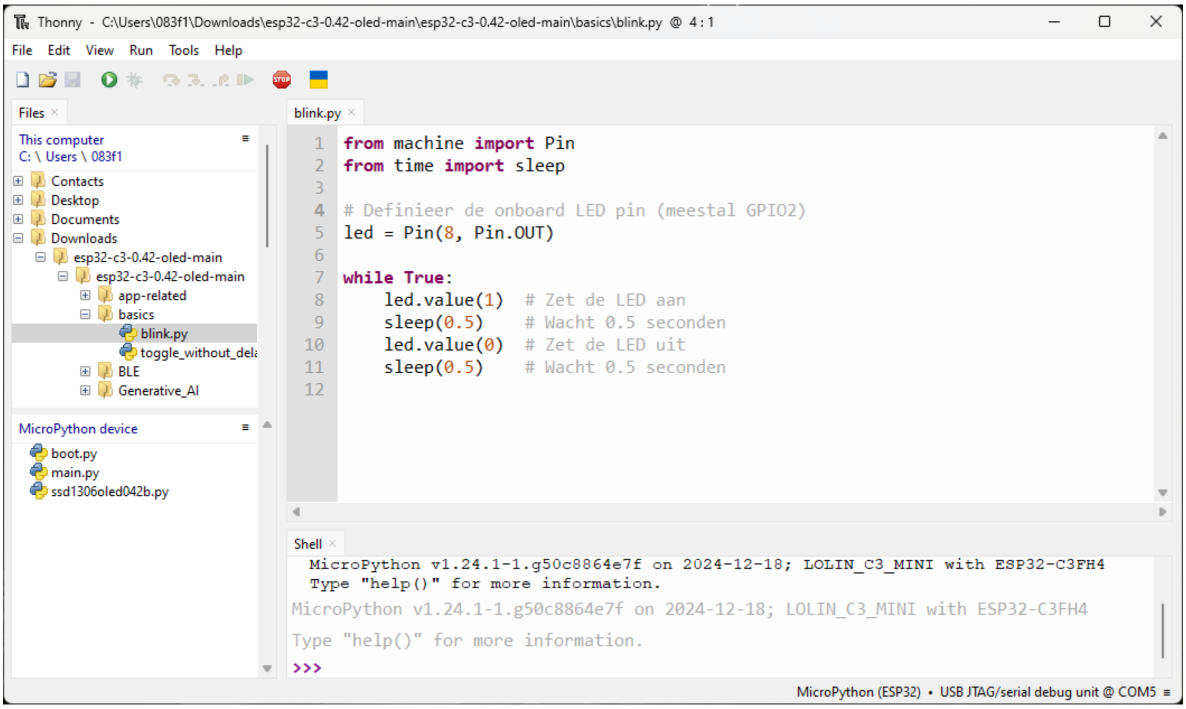


You have now successfully set up Thonny.

How to use Thonny

Please note: MicroPython is a little bit different from standard Python. Look up the differences yourself.

This is the Thonny view I prefer (see previous page for how to set this up):



```python
from machine import Pin
from time import sleep

# Definieer de onboard LED pin (meestal GPIO2)
led = Pin(8, Pin.OUT)

while True:
    led.value(1)  # Zet de LED aan
    sleep(0.5)    # Wacht 0.5 seconden
    led.value(0)  # Zet de LED uit
    sleep(0.5)    # Wacht 0.5 seconden

```

```
  MicroPython v1.24.1-1.g50c8864e7f on 2024-12-18; LOLIN_C3_MINI with ESP32-C3FH4
  Type "help()" for more information.
MicroPython v1.24.1-1.g50c8864e7f on 2024-12-18; LOLIN_C3_MINI with ESP32-C3FH4

Type "help()" for more information.
>>>
```

The top left pane shows the file browser. Select a file and double-click it to open it in the top right pane. Now you can run this code by clicking the green “play” button (top left). If the program prints information, you will see it in the “Shell” window. Errors are also shown there.

To stop the program, click anywhere in the “Shell” window and press Ctrl+C.

You can also run MicroPython commands from the “Shell”. You could even copy the entire program above to the “Shell” window. To do this: Select and copy the code, click on the right side of the “>>>” symbols (the prompt), and paste it to run. Again, to stop it, press Ctrl+C.

The bottom left window shows the files on the ESP32-C3 device. A blank device only contains “boot.py”. You can copy files there from the file browser by right-clicking on a file and selecting “Upload to /”.

If you want MicroPython code to run when you power on the ESP32-C3, you need to put your code in “main.py”. If it doesn’t exist, press Ctrl+N. This creates a new file. You now have a new tab named < untitled > in the top right window. Paste your code into the new file and click “File”, “Save as”. “Where to save to?” -> MicroPython device”.

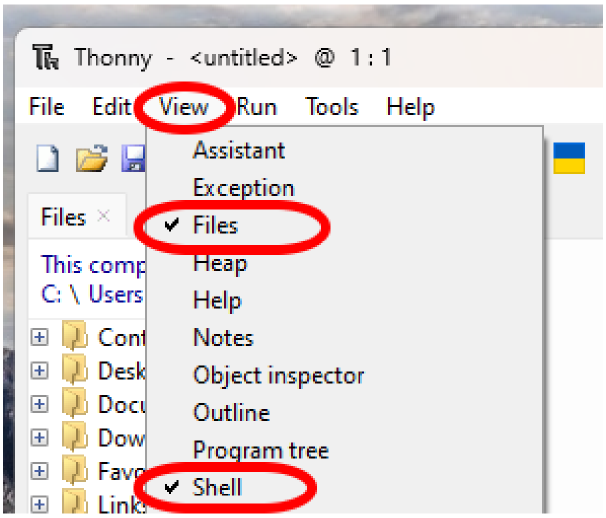
Tip: You can run your “main.py” by clicking in the “Shell” window and pressing Ctrl+D. This will reboot the ESP32-C3.

Troubleshooting: If you don’t get a MicroPython prompt, click the “Stop” icon. If this doesn't work, you might need to (re)install MicroPython on the ESP32-C3.

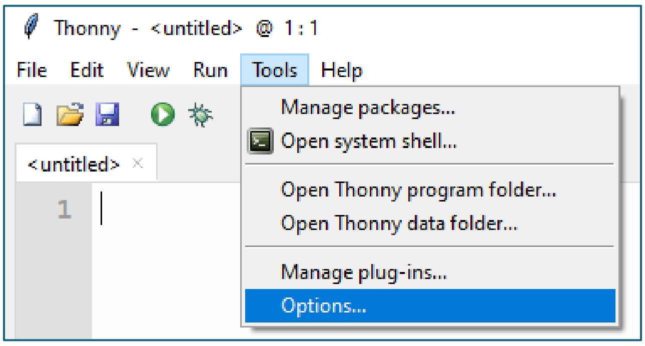
This is as far as I go – have fun playing with the ESP32-C3 and MicroPython!

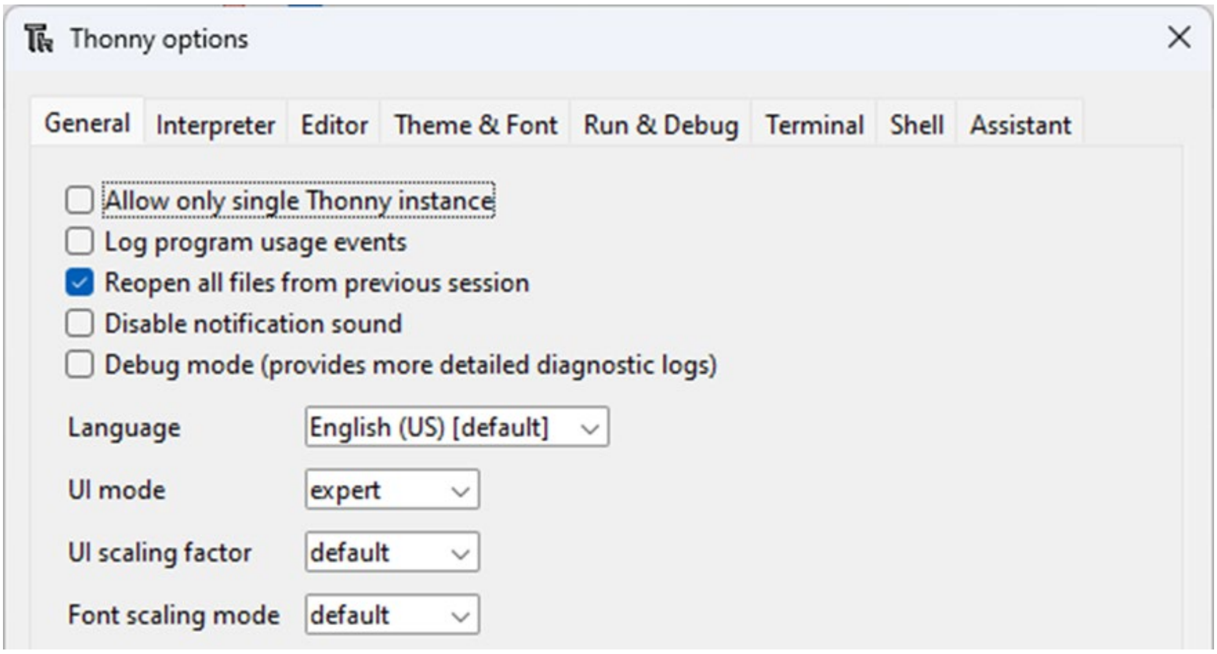
Attachment – Thonny settings

My favorite settings (do with them what you want). First the "View" menu:



The rest of the settings can be found via "Tools", "Options":





☐ Allow only single Thonny instance

☐ Log program usage events

☑ Reopen all files from previous session

☐ Disable notification sound

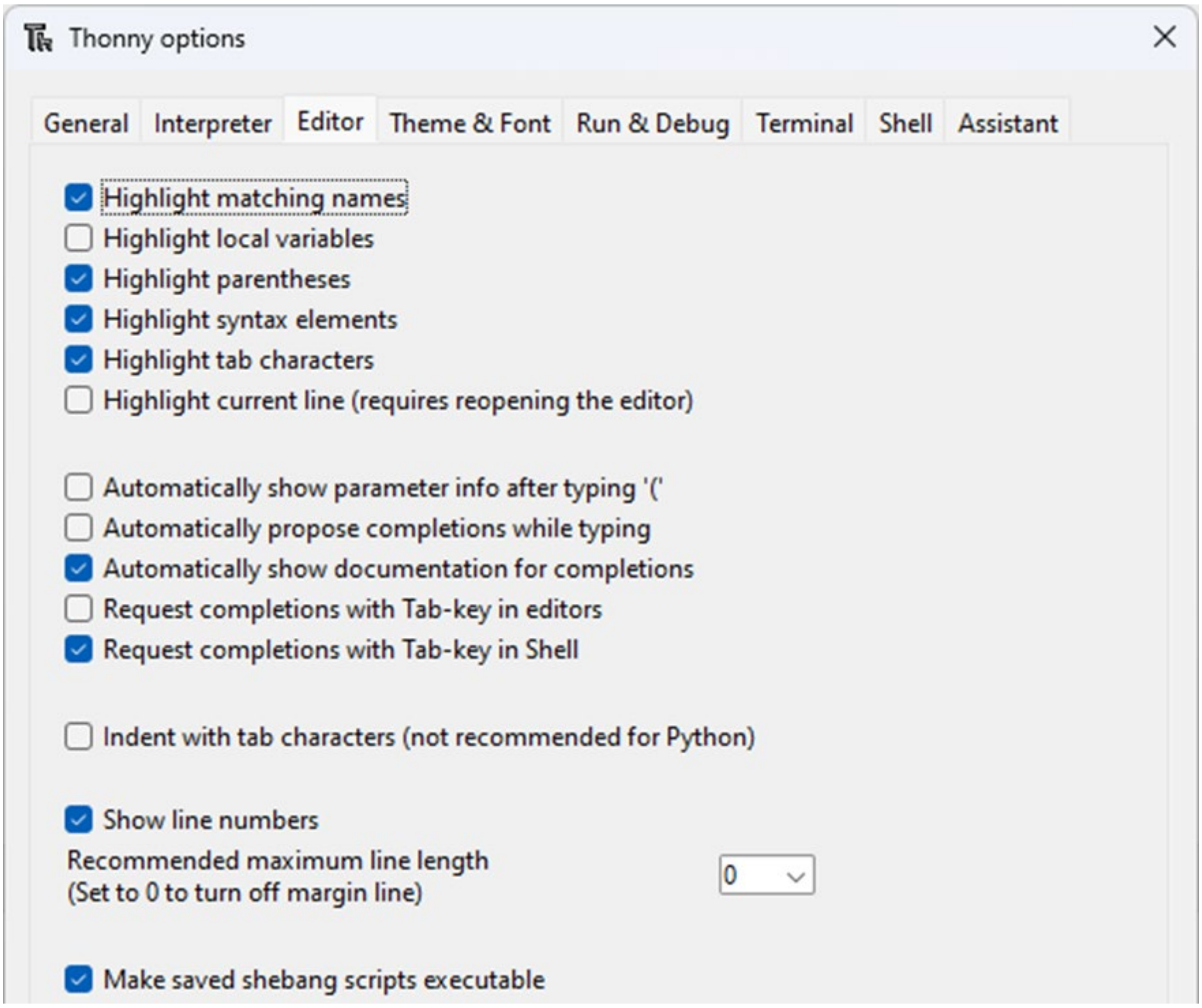
☐ Debug mode (provides more detailed diagnostic logs)

Language English (US) [default]

UI mode expert

UI scaling factor default

Font scaling mode default



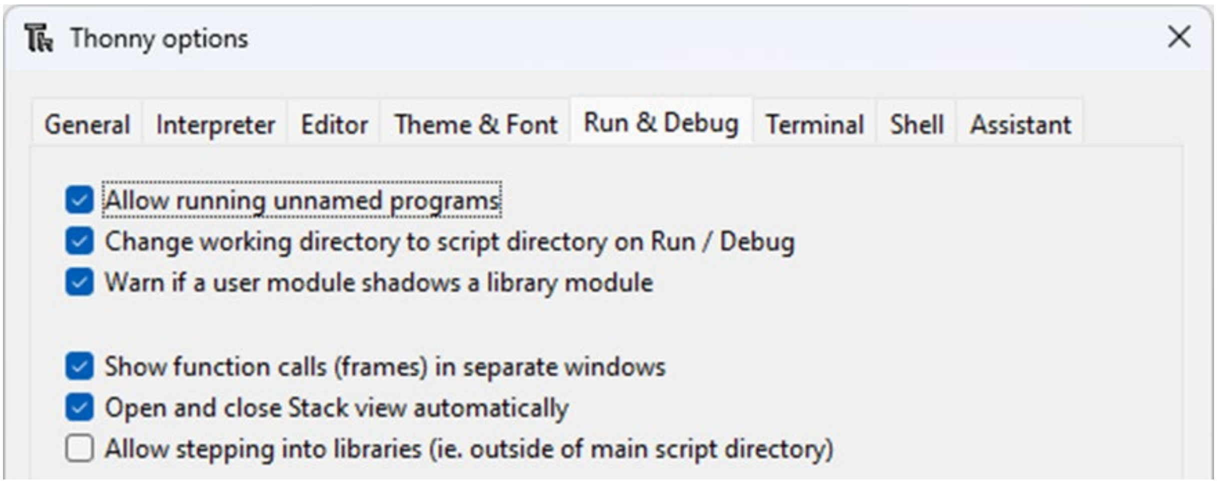
☑ Highlight matching names
□ Highlight local variables
☑ Highlight parentheses
☑ Highlight syntax elements
☑ Highlight tab characters
□ Highlight current line (requires reopening the editor)

□ Automatically show parameter info after typing '('
□ Automatically propose completions while typing
☑ Automatically show documentation for completions
□ Request completions with Tab-key in editors
☑ Request completions with Tab-key in Shell

□ Indent with tab characters (not recommended for Python)

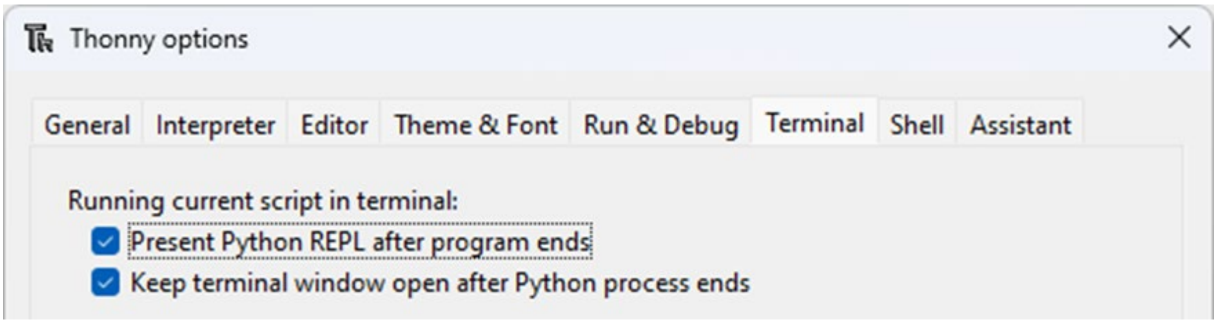
☑ Show line numbers
Recommended maximum line length
(Set to 0 to turn off margin line) 0

☑ Make saved shebang scripts executable



☑ Allow running unnamed programs
☑ Change working directory to script directory on Run / Debug
☑ Warn if a user module shadows a library module

☑ Show function calls (frames) in separate windows
☑ Open and close Stack view automatically
□ Allow stepping into libraries (ie. outside of main script directory)



Running current script in terminal:
☑ Present Python REPL after program ends
☑ Keep terminal window open after Python process ends