START HERE

Meet the board, then its firmware

01

Your goal: choose a compatible MicroPython image, install it, then prove that the board can run a small Python program. Plan about **25 minutes**, including practice.

Put these three items in front of you



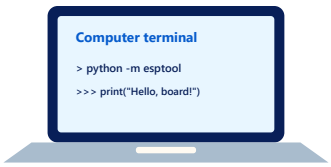
ESP32-S3 board

The small computer that runs your program. Find the module marking and USB sockets.



USB data cable

Connects board and computer. A power-only cable cannot transfer firmware.



Computer

Downloads firmware and runs the flashing tool. Match the cable end to its USB port.

> **Find it:** read the metal module marking. Distinguish **ESP32-S3** from ESP32, ESP32-C3 and ESP32-S2. Similar names do not mean interchangeable firmware.

What you need before the live task

- A working USB data connection; ML-C04 helps with cables and serial drivers.
- Internet for downloads and desktop Python installed on the computer.
- A copy of any existing board files or project you need to keep.

> **Use a bare board.** Disconnect sensors, jumper wires and external power before this USB-only exercise. Unplug USB before changing attached hardware.

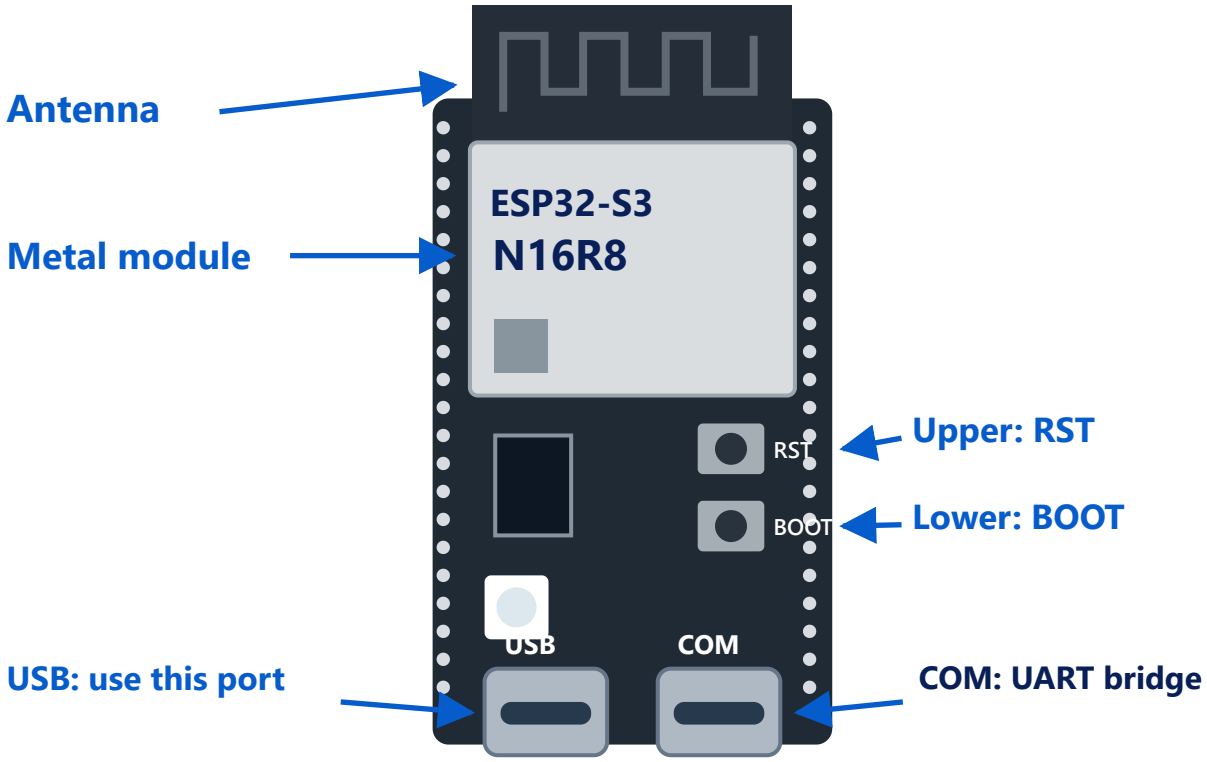
No electronics or Python background is assumed. All later console pictures are examples of what to look for.

IDENTIFY YOUR BOARD

Know these sockets and buttons

02

Use the **ESP32-S3-N16R8** board. Hold it with the components facing you, antenna at the top and USB-C sockets at the bottom.



Component side facing you · antenna up · sockets down

Simplified component-side view. The USB / COM socket names are printed on the reverse. Turning the board over reverses left and right.

USB · native connection

Bottom left in this front view. This lesson uses one data cable here for firmware download and the MicroPython console.

COM · serial bridge

Bottom right in this front view. It is a different route through a USB-to-UART bridge. Keep it unplugged for this exercise.

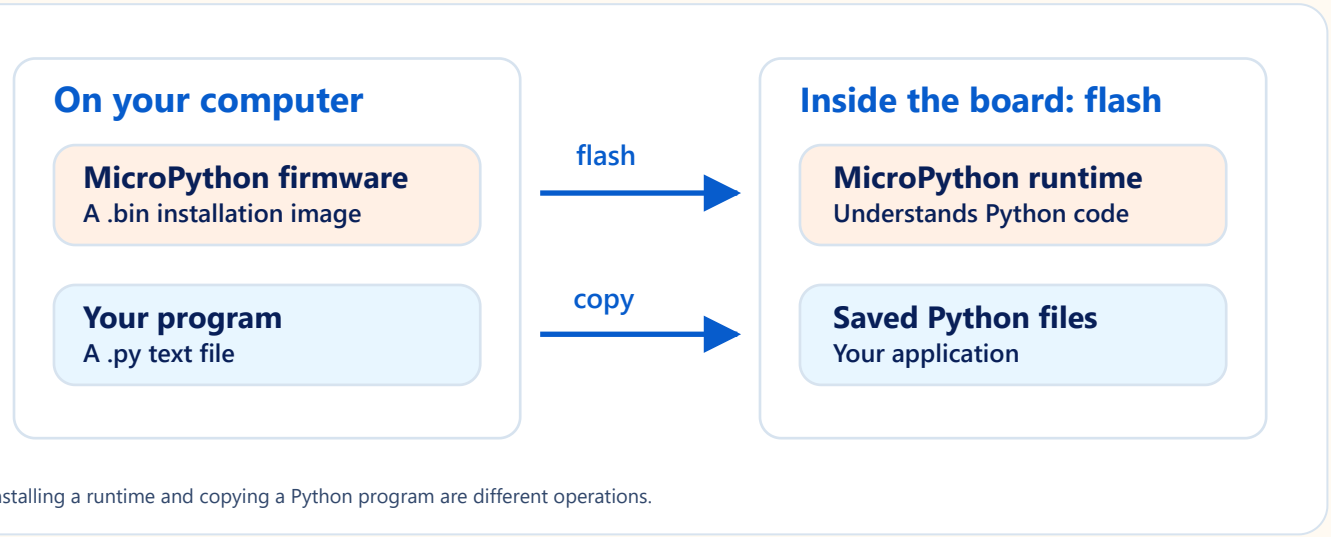
Point before plugging: identify USB, COM, the upper RST button and the lower BOOT button. Read the socket names on the back if needed, then return to the front view.

TWO DIFFERENT JOBS

Firmware and your program

03

Firmware is software installed in the board's flash storage. MicroPython firmware includes a small Python runtime: the software that reads and executes your Python instructions.



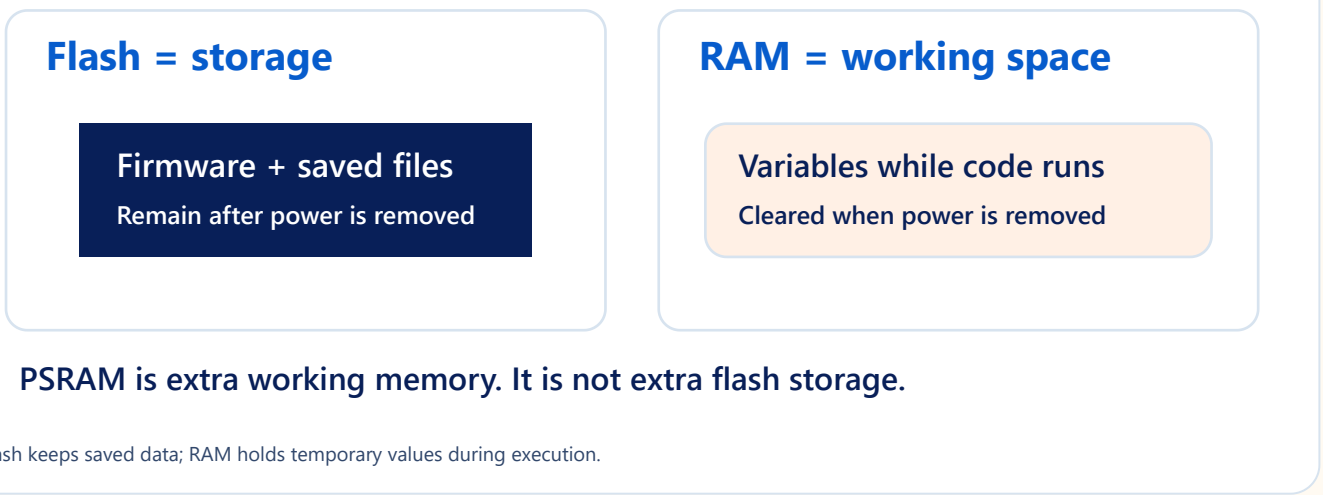
Installing a runtime and copying a Python program are different operations.

Install the runtime

Use a firmware `.bin` image and a flashing tool. Do this for first setup or a deliberate firmware change.

Change the application

Send Python statements or save `.py` files. An ordinary program edit does not require reflashing MicroPython.



PSRAM is extra working memory. It is not extra flash storage.

Flash keeps saved data; RAM holds temporary values during execution.

Your turn: you changed one line in `main.py`. Which job is needed?
Check: copy the changed Python file. Keep the installed MicroPython runtime.

CHOOSE A WORKFLOW

04

Three ways to write software

The board can run firmware made with different software tools. **ESP-IDF, Arduino and MicroPython are programming choices.** They are not three positions of a physical switch.

ESP-IDF	C / C++ project	→ Build firmware	→ Flash + run
Arduino	C++ sketch	→ Build firmware	→ Upload + run
MicroPython	Python program	→ Install runtime once	→ Send / save .py

Build means turn source code into machine firmware. Flash means write that firmware into the board.

ESP-IDF

Espressif's development framework. Write C/C++, configure the project, then build and flash. Useful when you need detailed platform control.

Arduino

Write a C++ sketch using Arduino libraries. The ESP32 core is built on ESP-IDF. The Upload action builds and flashes your sketch.

This course uses MicroPython: install a compatible runtime, then try short Python statements immediately. Save Python files as your application grows.

Switching workflows changes the installed firmware. Uploading an Arduino sketch replaces the MicroPython firmware in the usual single-application setup. Save your Python project on the computer before switching back and forth.

Choose for the task

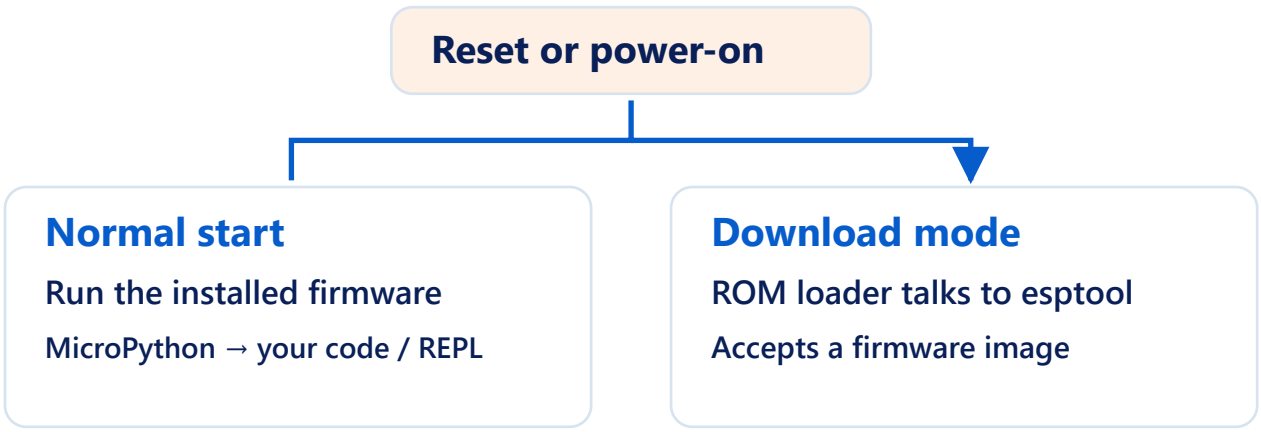
Quick interactive Python experiment → MicroPython.
Existing Arduino library or sketch → Arduino.
Project needing ESP-IDF configuration and APIs → ESP-IDF.
These are starting points. Experienced projects can combine frameworks; that is beyond this first lesson.

WHAT STARTS AFTER RESET?

05

Run mode and download mode

Reset restarts the chip. During a normal start it runs the firmware already in flash. In **download mode**, the chip's built-in ROM loader receives new firmware instead.



A download-mode board is not waiting for Python statements.

The ROM loader is built into the chip; it is separate from your MicroPython program.

Normal MicroPython start

The runtime starts, runs startup files and can show the interactive Python prompt. Your program belongs here.

Firmware download

The flashing tool communicates with the loader. Python expressions such as `2 + 3` are not commands for this loader.

Predict: a board is accepting firmware from esptool. Should it show >>>?
Check: no. Finish flashing, leave download mode and restart the installed firmware before opening Python.

Pressing reset does not erase flash. Erasing flash is a separate tool command and removes the firmware and stored files. Keep copies of files before an erase.

FIRMWARE SELECTION

Choose the matching image

06

For this board, **N16R8** identifies 16 MB of flash and 8 MB of **Octal-SPI PSRAM**. Octal describes the memory interface; it matters when choosing the MicroPython build.

1	2	3	4
Chip family	**Memory type**	**Image format**	**Release**
ESP32-S3, not ESP32 / C3 / S2	Flash size + PSRAM type	Documented .bin + offset	Stable release before preview

A filename is a compatibility choice, not just a download.

Check all four fields before an erase or a write.

Use this download route

1. Open MicroPython → ESP32-S3 downloads.
2. Find **Firmware (Support for Octal-SPIRAM)**. Choose a stable release, not a preview.
3. Download the **merged `.bin`**, not `.app-bin`, `.uf2` or `.elf`.

> **Worked selection:** `ESP32_GENERIC_S3-SPIRAM_OCT-20260824-v1.29.0.bin`
> Chip: ESP32-S3 · PSRAM: Octal · image: merged BIN · write address: **0x0**.

Save the file in your lesson folder. For shorter commands below, make a copy named `firmware.bin`. Keep the original filename in your notes. Make sure your file browser shows extensions so the copy is not named `firmware.bin.bin`.

> **Reject the mismatch:** an ESP32-C3 image and a non-Octal PSRAM build are not the selection for this N16R8 exercise. A `.py` file is your program, not a firmware image.

The firmware's partition layout determines usable file storage. A 16 MB flash chip does not promise a 16 MB Python filesystem.

LIVE TASK · PREPARE

07

Prepare the computer and USB link

The commands below run in the **computer terminal**, not after a board's >>> prompt. The worked commands use Windows PowerShell and a lesson folder containing `firmware.bin`.

esptool writes firmware to the board. **mpremote** opens the Python console and copies Python files.

1 · Create a small tool environment

Install desktop Python 3.10 or newer, then open a terminal in your lesson folder. Confirm `python --version` before continuing.

```
python -m venv .venv
.venv\Scripts\python.exe -m pip install "esptool>=5,<6" mpremote
```

In the following commands, PY means the Python executable inside this environment. Replace PY with `.venv\Scripts\python.exe` on Windows, or `.venv/bin/python` on macOS / Linux.

2 · Identify the board's port

Unplug

Run the list command with the board disconnected.

Connect USB

Use the native USB socket shown on page 2.

Compare

Run the list again. Find the newly appearing port.

```
PY -m mpremote connect list
```

Example names: Windows `COM7` · Linux `/dev/ttyACM0` · macOS `/dev/cu.usbmodem…`.
These are examples. Use your own detected port.

3 · Release the port

Close other serial monitors, editor connections and browser serial sessions before esptool uses the port. Keep only one serial tool connected at a time.

No new port? Try the download-mode sequence on the next page, then list again. If none appears, check a known data cable, a direct computer USB port and ML-C04 before erasing anything.

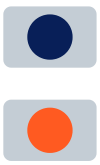
LIVE TASK · DOWNLOAD AND FLASH

Install MicroPython

08

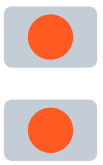
Save any existing project files on the computer before proceeding. **Erase removes the existing firmware, Python files and configuration in flash.** Do not erase a board whose contents you need but have not saved.

1 · Hold BOOT



Keep the lower button down

2 · Tap RST



Press + release the upper key

3 · Release BOOT



The ROM loader can receive

Then identify the serial port again. Its name may have changed.

Use the lower BOOT and upper RST buttons. Repeat this sequence if a later esptool command cannot reconnect.

List ports again. Use the download-mode port now shown. In the commands below, replace `PORT` with that name and replace `PY` as defined on page 7. Keep `firmware.bin` in the current folder.

1 · Check the target before erasing

```
PY -m esptool --chip esp32s3 --port PORT flash-id
```

Proceed only if the tool connects to ESP32-S3 and reports the expected 16 MB flash. Stop and investigate a target or size mismatch. These checks do not identify the PSRAM type.

2 · Erase, then write the merged image

```
PY -m esptool --chip esp32s3 --port PORT erase-flash
PY -m esptool --chip esp32s3 --port PORT write-flash 0x0 firmware.bin
```

Leave the cable connected while writing. Use esptool 5's hyphenated command names. The write address **0x0** belongs to this S3 merged image.

Look for: erase completed, write reached completion and verification succeeded, with no error. Message wording varies; a full progress bar alone is not the final check.

3 · Start the runtime: release BOOT, then tap RST alone. Rediscover the port: normal MicroPython USB may have a different port name from download mode. Continue on page 9.

LIVE TASK · VERIFY

09

Try a line; read the reply

REPL means read–evaluate–print loop. It is an interactive conversation: you enter one Python instruction, the board executes it and returns a result or a new prompt.

In the computer terminal, run `PY -m mpremote connect PORT repl` using the current runtime port. Press Enter; use Ctrl-C if a program is running. Look for the MicroPython banner or prompt.

Example board console

```
>>> 2 + 3
5
>>> print("Hello, board!")
Hello, board!
>>>
```

Type after >>>. Press Enter. Read the reply, then try the next line.

White lines show what you type; green lines show the example reply. Do not type the prompt symbols.

1 · Read

Find >>> in the board console.

2 · Enter

Type `2 + 3` and press Enter.

3 · Check

Look for `5` and another prompt.

Computer command prompt

Commands such as `python -m esptool` run on the computer, in its terminal.

Board Python prompt

>>> is where Python statements go. A computer terminal command pasted here causes a Python error.

Practice: change the greeting inside the quotation marks. Keep the quotation marks and parentheses. The reply should contain your new words.

Useful keys: Ctrl-C interrupts a running Python program. Ctrl-D at the normal prompt performs a soft reset. A soft reset restarts Python; it does not install different firmware.

Verify the runtime too: at >>>, type `import sys`, then `print(sys.implementation.name)`. Expect `micropython`. Use `print(sys.version)` to inspect the version.

In mpremote, Ctrl-] exits to the computer terminal. A separate UART terminal, if used, needs 115200 baud; keep this exercise on the native USB route.

LIVE TASK · SAVE AND PROVE

10

Make your program survive restart

A typed statement runs now. A saved file stays in flash. MicroPython runs `main.py` during a normal startup, after `boot.py` if that file exists.

Try without saving

mpremote run main.py

Executes from RAM
Does not install main.py

Save for startup

mpremote fs cp main.py :main.py

Writes a file in flash
Runs at normal startup

Normal startup: runtime → boot.py (if present) → main.py → REPL

The shorter commands in the diagram show the operations; use the explicit Python and port commands below.

1 · Save a file on the computer

Create `main.py` in your lesson folder as plain text. Make sure it is not `main.py.txt`. Replace any existing board copy only after saving a backup.

```
print("ML-C03 ready")
```

2 · Copy it to the board

Exit the REPL connection with Ctrl-] first. In the computer terminal, use the **normal runtime port**:

```
PY -m mpremote connect PORT fs cp main.py :main.py
PY -m mpremote connect PORT repl
```

The colon means “on the board.” With the console open, press Ctrl-C for a prompt, then Ctrl-D at >>> to restart Python and see the startup line.

Success: `ML-C03 ready` appears after the soft restart. For a second check, tap RST with BOOT released; reconnect if USB reappears under a new name. If you missed the startup line, use Ctrl-D at the prompt to see it again.

Change one thing: edit the message, copy `main.py` again and repeat the check. You do not need to erase or reflash firmware for this edit.

PRACTICE AND RECOVERY

11

Use the first failing checkpoint

Work from the first missing observation. Change one thing, then repeat that checkpoint.

1 · No serial port

Check USB socket, data cable and computer USB port. Try BOOT + RST, then list again. Power light alone does not prove data works. See ML-C04 for drivers.

2 · Port busy / denied

Close other monitors and browser sessions. On Linux, check serial-access permissions. Reconnect using the current port name.

3 · Stuck at Connecting

Repeat hold BOOT → tap RST → release BOOT. List ports again and update PORT before retrying. Keep other hardware detached.

4 · Write fails midway

Use a direct USB connection and known data cable. This lesson omits the optional fast baud setting. If you added one, return to the default speed.

5 · Written, but no Python

Release BOOT and tap RST. Find the runtime port; press Enter or Ctrl-C. Recheck S3 + SPIRAM_OCT + merged BIN + offset 0x0 if startup keeps failing.

6 · Python error

Read the first error line. Check quotes and parentheses. A running program may need Ctrl-C. A bad .py file usually needs a file fix, not a firmware erase.

Explain these before you finish

1. You changed a greeting in `main.py`. Reflash firmware or copy a file?
2. The USB port changes after reset. Keep using the old name or list again?
3. The board shows a Python prompt. Download mode or running MicroPython?

> **Answers:** copy the file; list again; running MicroPython.

Your completion check

- ☐ Point to native USB, COM, RST and BOOT in the front view.
- ☐ Explain why this N16R8 needs the Octal-SPIRAM image.
- ☐ Get 5 from `2 + 3` and see your greeting.
- ☐ Save a changed `main.py` and see its message after restart.

Keep the firmware filename, port and your program in your project notes. For a deeper recovery exercise, continue with ML-C10.