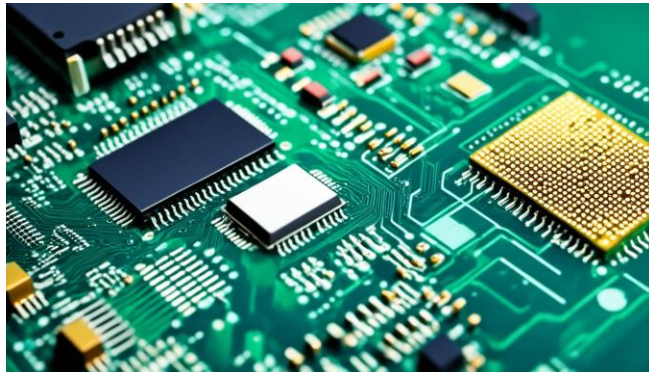
EP1001 FURTHER DIGITAL FABRICATION & PROTOTYPING

Embedded Programming

COMPUTER ARCHITECTURES



- Computers follow 2 basic memory architectures
  - Von Neumann: shared instruction & data space
  - Harvard: separate instruction & data space
- CPU architectures can be
  - CISC: complex, multi-cycle instructions
  - RISC: simple, single cycle instructions
- Specialized CPUs include
  - GPU: graphics processing unit
  - TPU: tensor processing unit (for AI tasks)
- Hardware implementation
  - CPUs: general purpose, slower
  - FPGAs: field (end-user) programmable, custom circuits
  - ASICs: application specific, highly customised at IC manufacturer level

MICROPROCESSORS VS MICROCONTROLLERS VS SOC

Microprocessors

- general purpose processors
- require external hardware (memory, peripherals)

Microcontrollers

- integrate CPU, memory, peripherals within a single chip
- limited memory
- CPU not as fast as microprocessors

System on a Chip (SoC)

- Single IC that combines most or all components of a computer on a single chip
- Includes CPU, GPU, memory, I/O ports, peripherals, wireless connectivity and display units

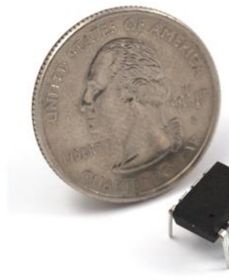
TYPES OF MEMORY

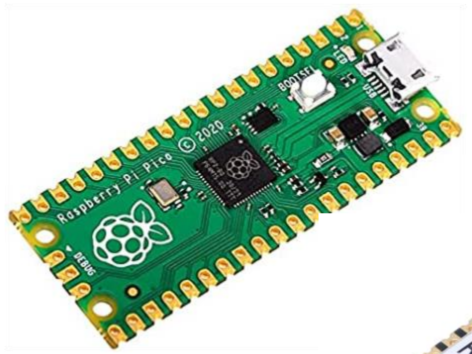
- Registers (within CPU)
- Non-volatile memory
  - PROM
  - EPROM
  - EEPROM
  - Flash memory
  - NVRAM
- Volatile memory
  - SRAM
  - DRAM
- Fuse (for configuration settings)

TYPICAL MICROCONTROLLER PERIPHERALS

- Input/Output ports
- Analog-to-Digital converters
- Analog comparators
- Digital-to-analog converters
- Timer/Counter units
- PWM waveform generators
- UART (Serial Communications)
- USB interface

PROCESSOR FAMILIES

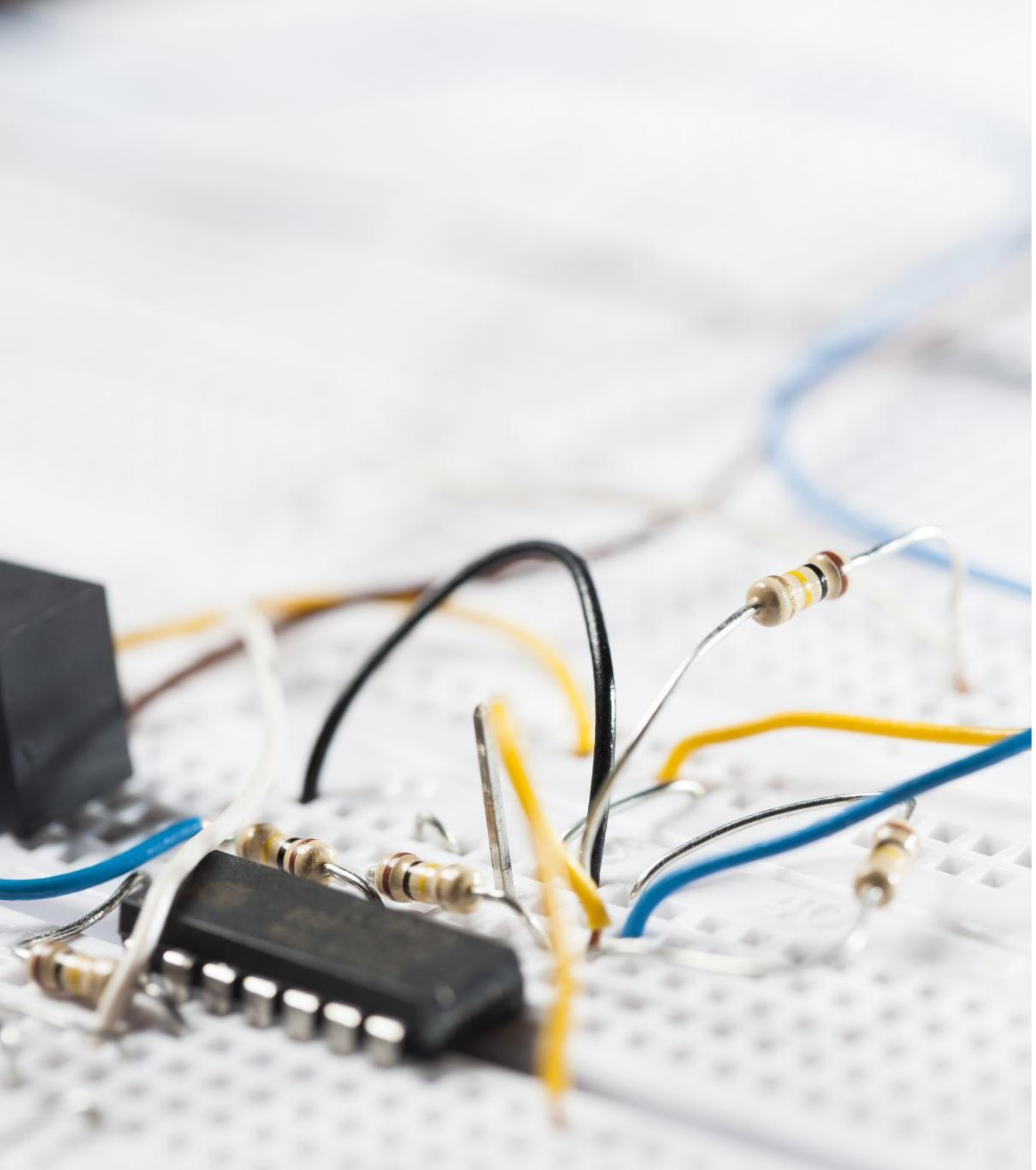




- 8051
- PIC
- AVR
  - ATmega328
  - ATtiny412, ATtiny1614, ATtiny3216
- ARM
  - ATSAMD11, ATSAMD21
  - STM32
- Raspberry Pi
  - RP2040
  - RP2350
- Xtensa
  - ESP8266
  - ESP32
- RISC-V
  - ESP32-C3, ESP32-C6, ESP32-S3

MICROCONTROLLER PROGRAMMING LANGUAGES

- Assembly language
- C/C++
- MicroPython/CircuitPython
- JavaScript
- Basic
- Forth
- Lua

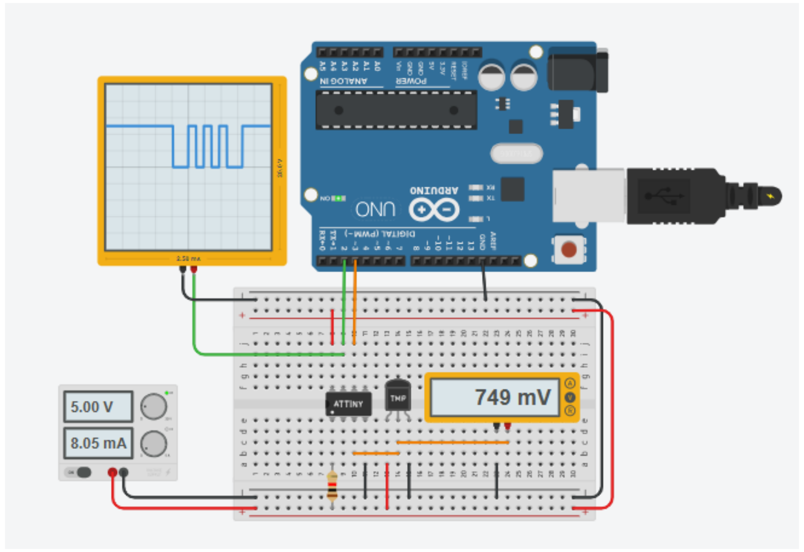


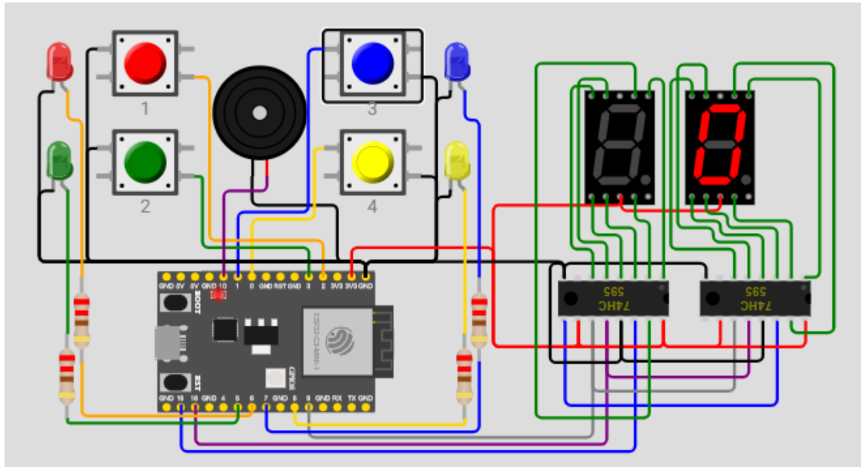
DEVELOPMENT ENVIRONMENTS

- Arduino IDE: C/C++, Arduino sketch
- Visual Studio Code / Platform IO: multiple languages
- Eclipse: multiple languages
- Microchip Studio: C/C++
- GCC: C/C++
- Thonny IDE: MicroPython, CircuitPython

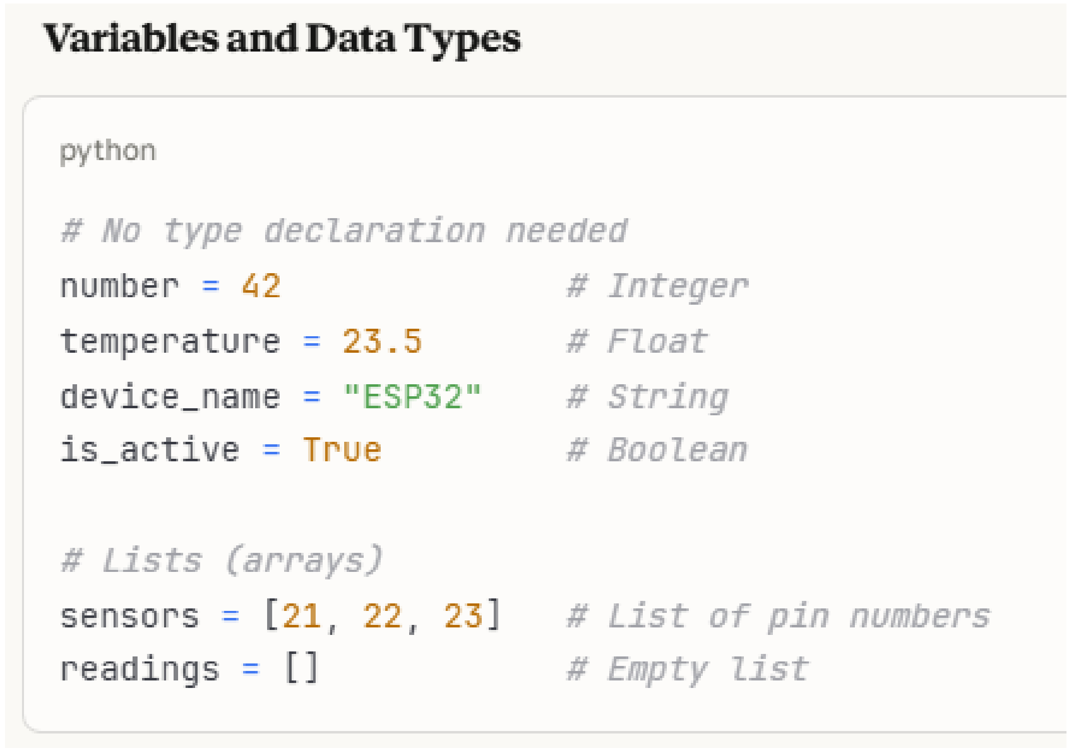
SIMULATORS

- Tinkercad Circuits
    - Basic electronics
    - Arduino Uno
    - ATtiny85
    - Micro:bit
- Wokwi
    - Arduino Uno, Mega
    - ATtiny85
    - ESP32
    - STM32
    - Raspberry Pi Pico





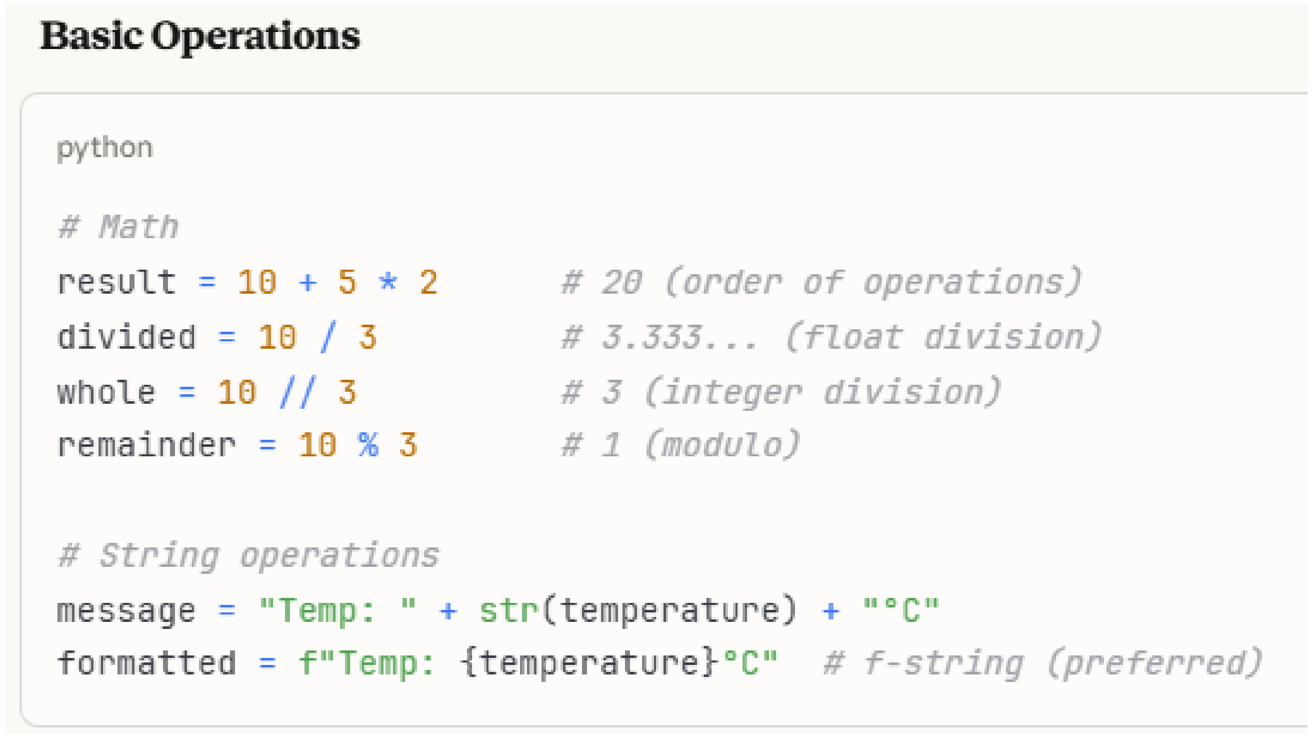
MICROPYTHON BASICS



Variables and Data Types

python

```python
# No type declaration needed
number = 42                # Integer
temperature = 23.5         # Float
device_name = "ESP32"      # String
is_active = True           # Boolean

# Lists (arrays)
sensors = [21, 22, 23]     # List of pin numbers
readings = []              # Empty list
```



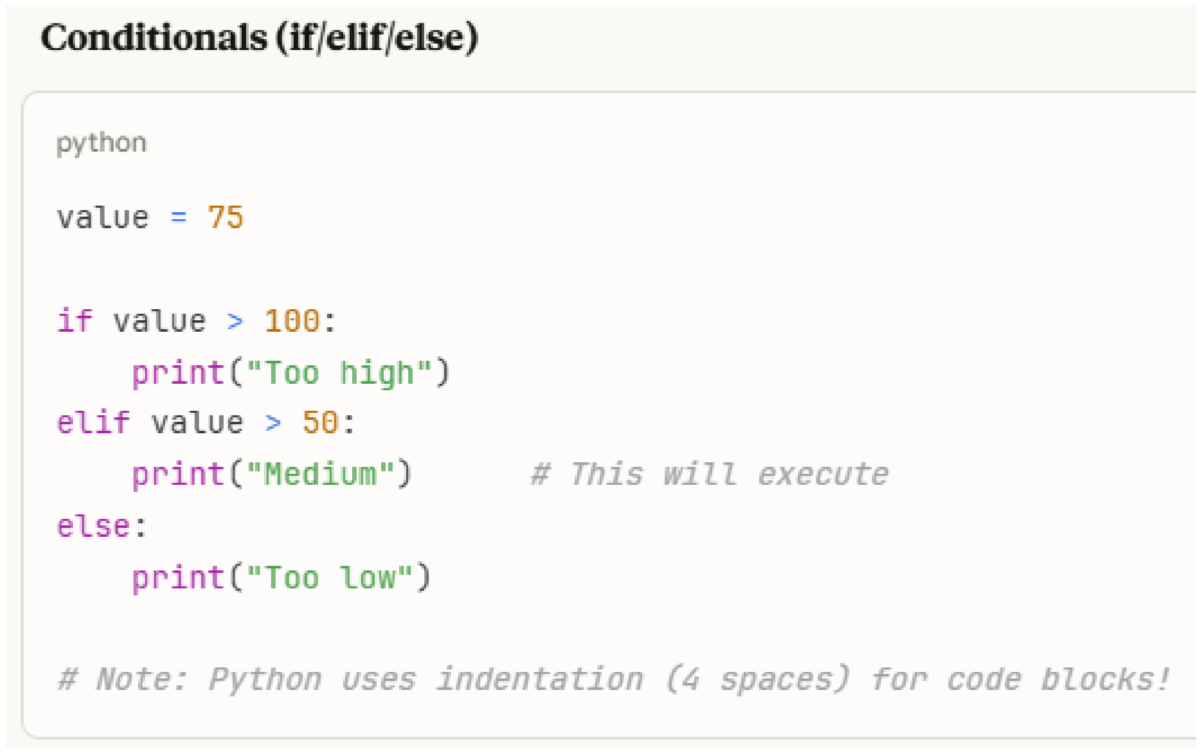
Basic Operations

python

```python
# Math
result = 10 + 5 * 2        # 20 (order of operations)
divided = 10 / 3           # 3.333... (float division)
whole = 10 // 3            # 3 (integer division)
remainder = 10 % 3         # 1 (modulo)

# String operations
message = "Temp: " + str(temperature) + "°C"
formatted = f"Temp: {temperature}°C"  # f-string (preferred)
```

MicroPython Getting Started Guide   ESP32   Raspberry Pi Pico

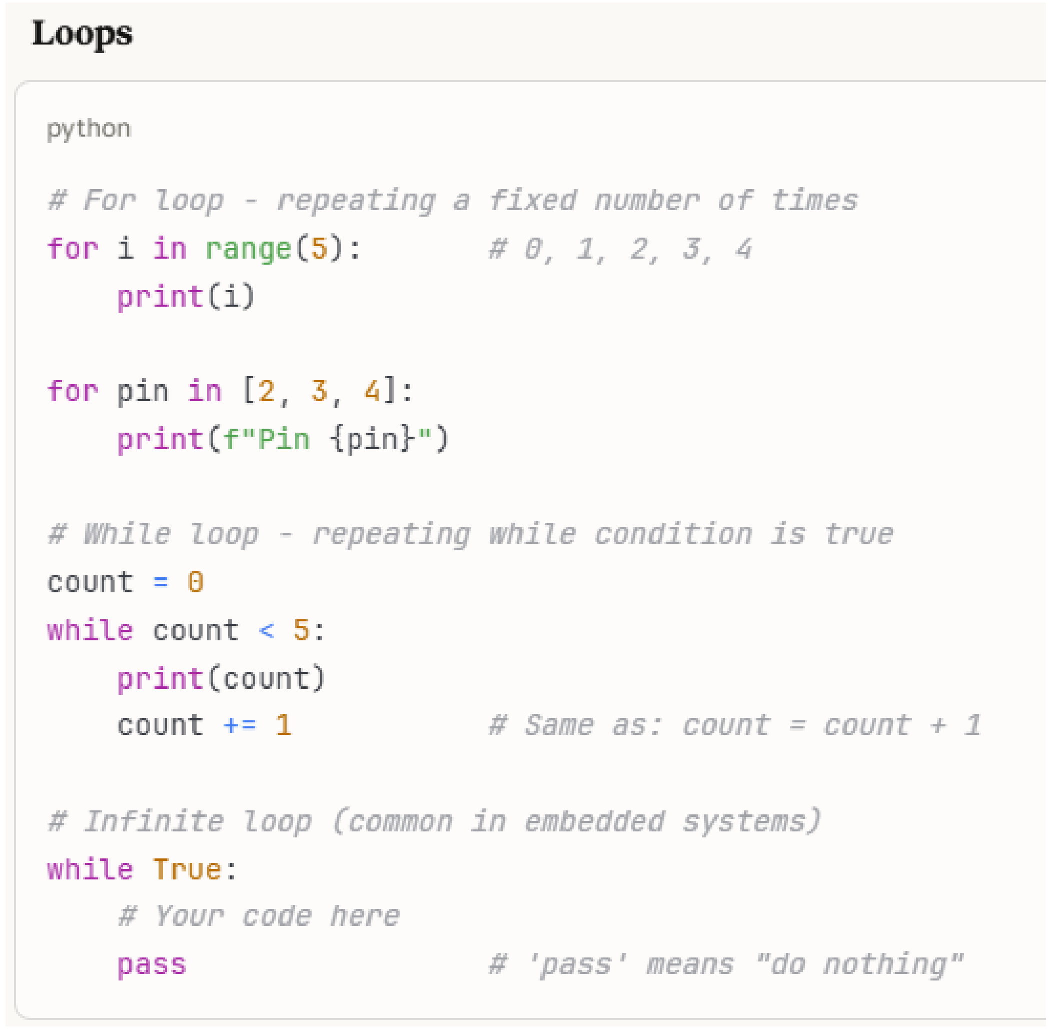
MICROPYTHON BASICS



Conditionals (if/elif/else)

python

```python
value = 75

if value > 100:
    print("Too high")
elif value > 50:
    print("Medium")      # This will execute
else:
    print("Too low")

# Note: Python uses indentation (4 spaces) for code blocks!
```

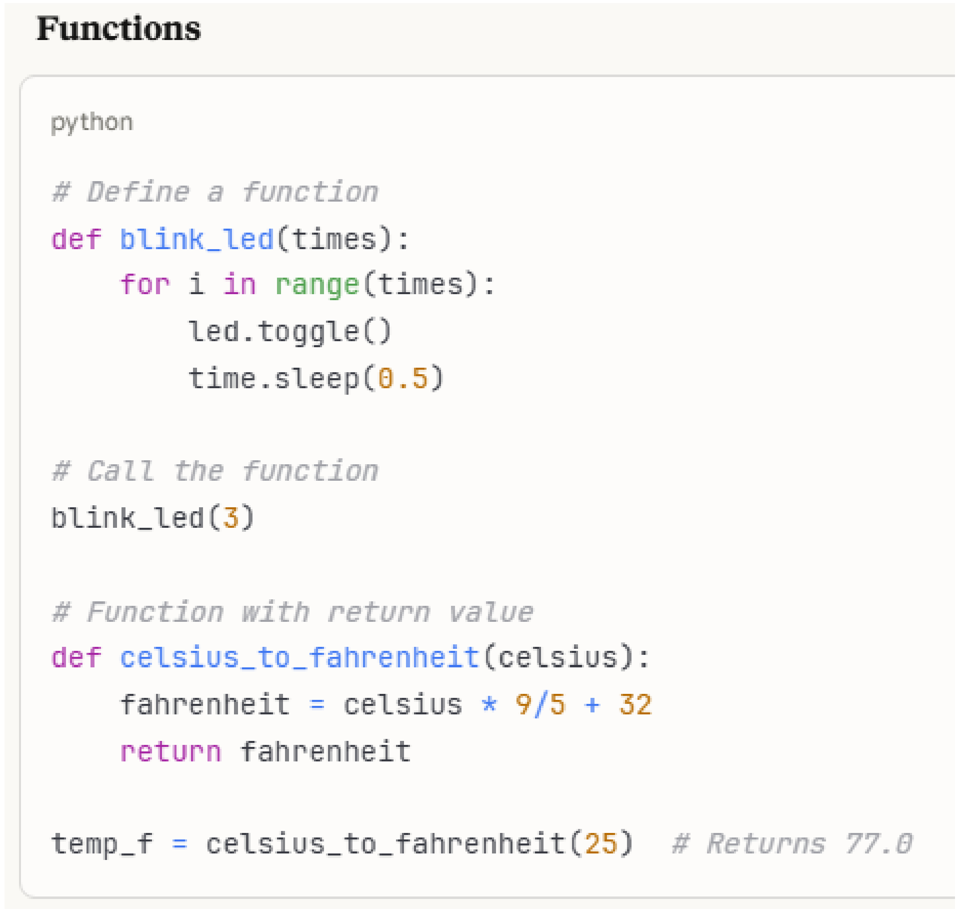


Loops

python

```python
# For loop - repeating a fixed number of times
for i in range(5):          # 0, 1, 2, 3, 4
    print(i)

for pin in [2, 3, 4]:
    print(f"Pin {pin}")

# While loop - repeating while condition is true
count = 0
while count < 5:
    print(count)
    count += 1              # Same as: count = count + 1

# Infinite loop (common in embedded systems)
while True:
    # Your code here
    pass                    # 'pass' means "do nothing"
```

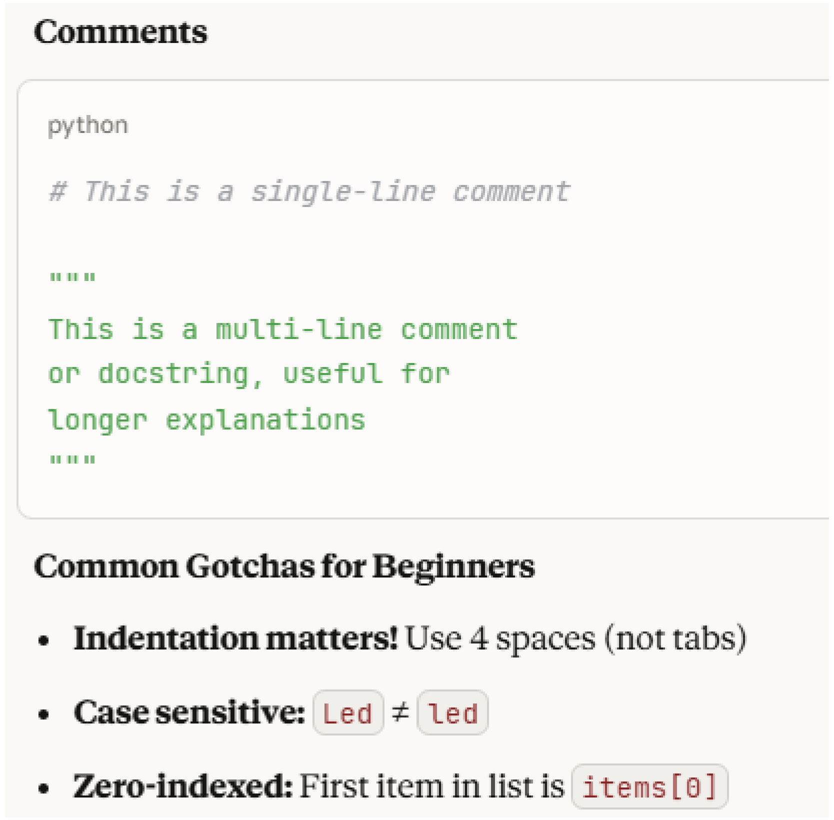
MICROPYTHON BASICS



Functions

python

```python
# Define a function
def blink_led(times):
    for i in range(times):
        led.toggle()
        time.sleep(0.5)

# Call the function
blink_led(3)

# Function with return value
def celsius_to_fahrenheit(celsius):
    fahrenheit = celsius * 9/5 + 32
    return fahrenheit

temp_f = celsius_to_fahrenheit(25)  # Returns 77.0
```



Comments

python

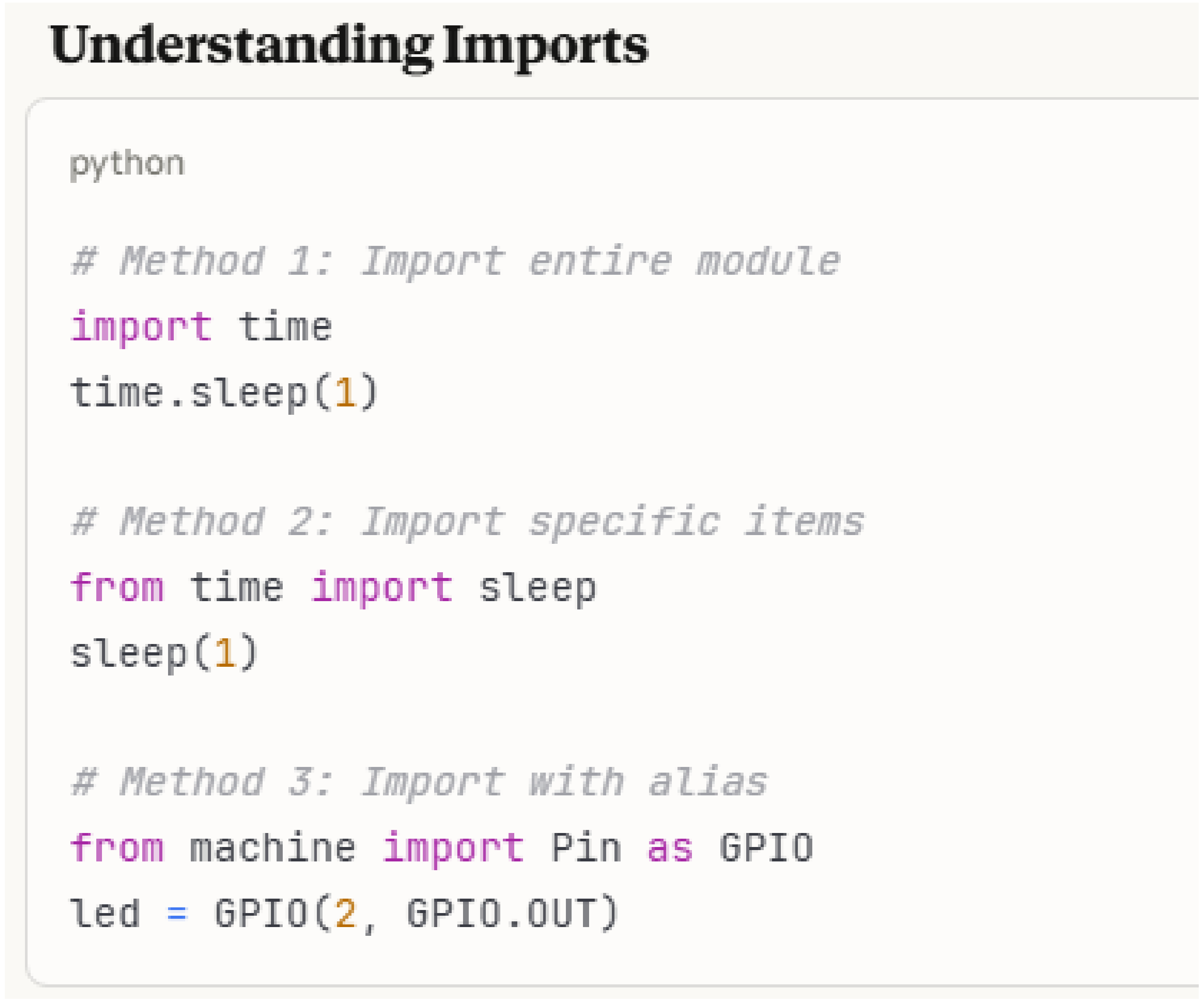
```python
# This is a single-line comment

"""
This is a multi-line comment
or docstring, useful for
longer explanations
"""
```

Common Gotchas for Beginners

- **Indentation matters!** Use 4 spaces (not tabs)
- **Case sensitive:** `Led` ≠ `led`
- **Zero-indexed:** First item in list is `items[0]`

MICROPYTHON BASICS

- Importing libraries
- Import vs from … import

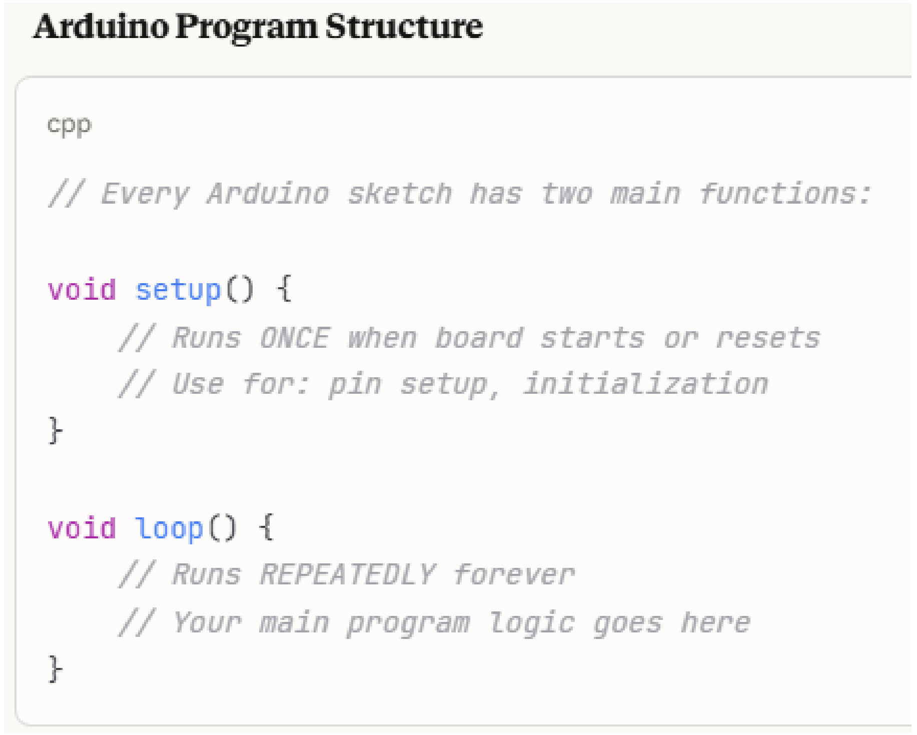


Understanding Imports

python

```python
# Method 1: Import entire module
import time
time.sleep(1)

# Method 2: Import specific items
from time import sleep
sleep(1)

# Method 3: Import with alias
from machine import Pin as GPIO
led = GPIO(2, GPIO.OUT)
```

ARDUINO (C/C++) BASICS

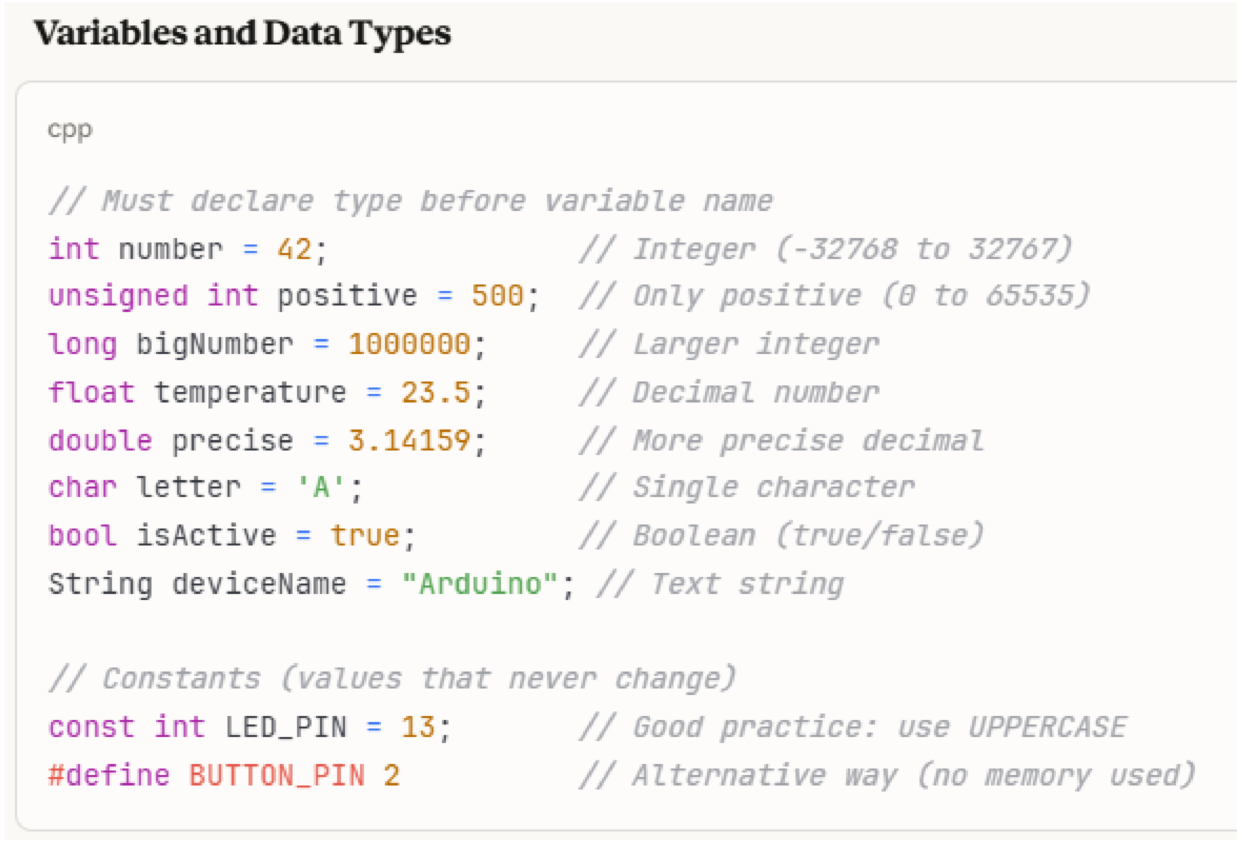


Arduino Program Structure

cpp

```cpp
// Every Arduino sketch has two main functions:

void setup() {
    // Runs ONCE when board starts or resets
    // Use for: pin setup, initialization
}

void loop() {
    // Runs REPEATEDLY forever
    // Your main program logic goes here
}
```

Getting Started with the ESP32C3 Supermini
Arduino-Pico ReadTheDocs

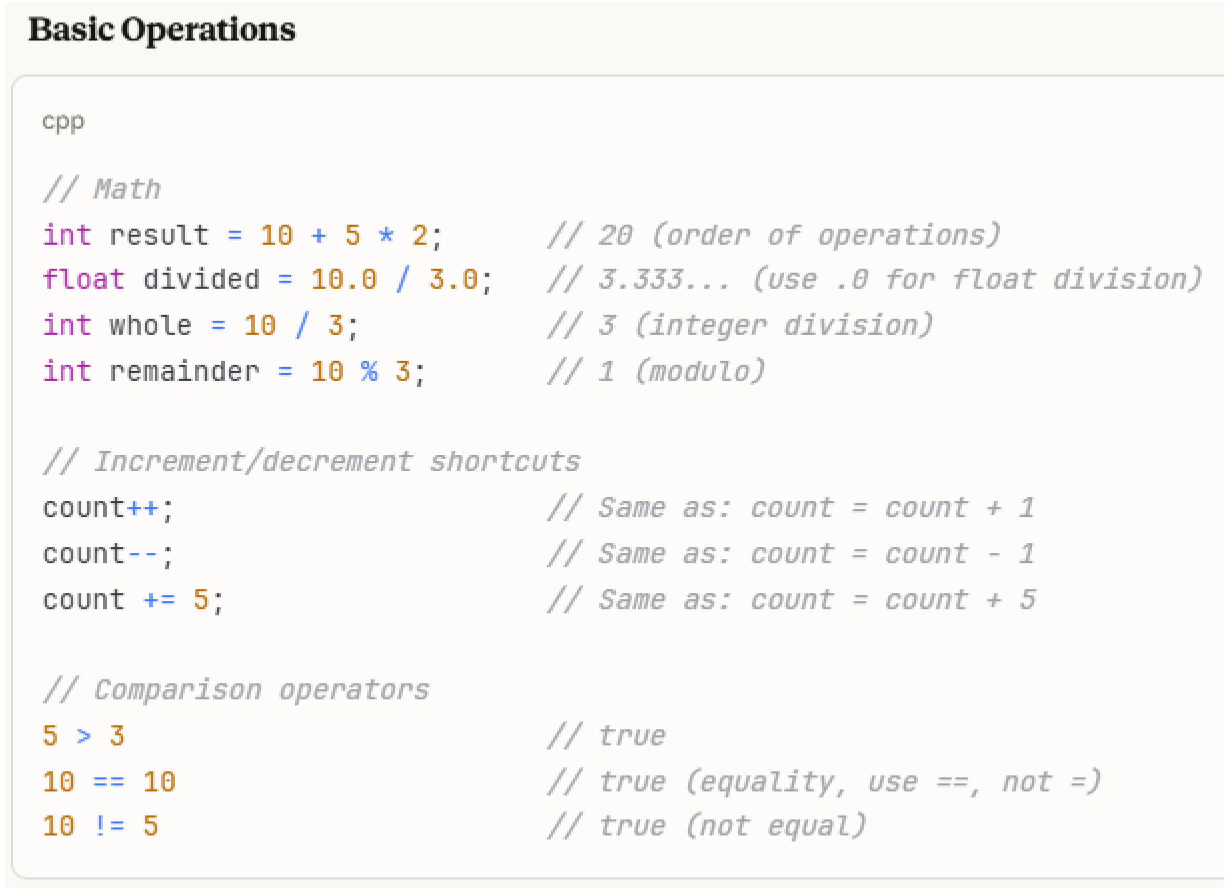


Variables and Data Types

cpp

```cpp
// Must declare type before variable name
int number = 42;                // Integer (-32768 to 32767)
unsigned int positive = 500;    // Only positive (0 to 65535)
long bigNumber = 1000000;       // Larger integer
float temperature = 23.5;       // Decimal number
double precise = 3.14159;       // More precise decimal
char letter = 'A';              // Single character
bool isActive = true;           // Boolean (true/false)
String deviceName = "Arduino"; // Text string

// Constants (values that never change)
const int LED_PIN = 13;         // Good practice: use UPPERCASE
#define BUTTON_PIN 2            // Alternative way (no memory used)
```

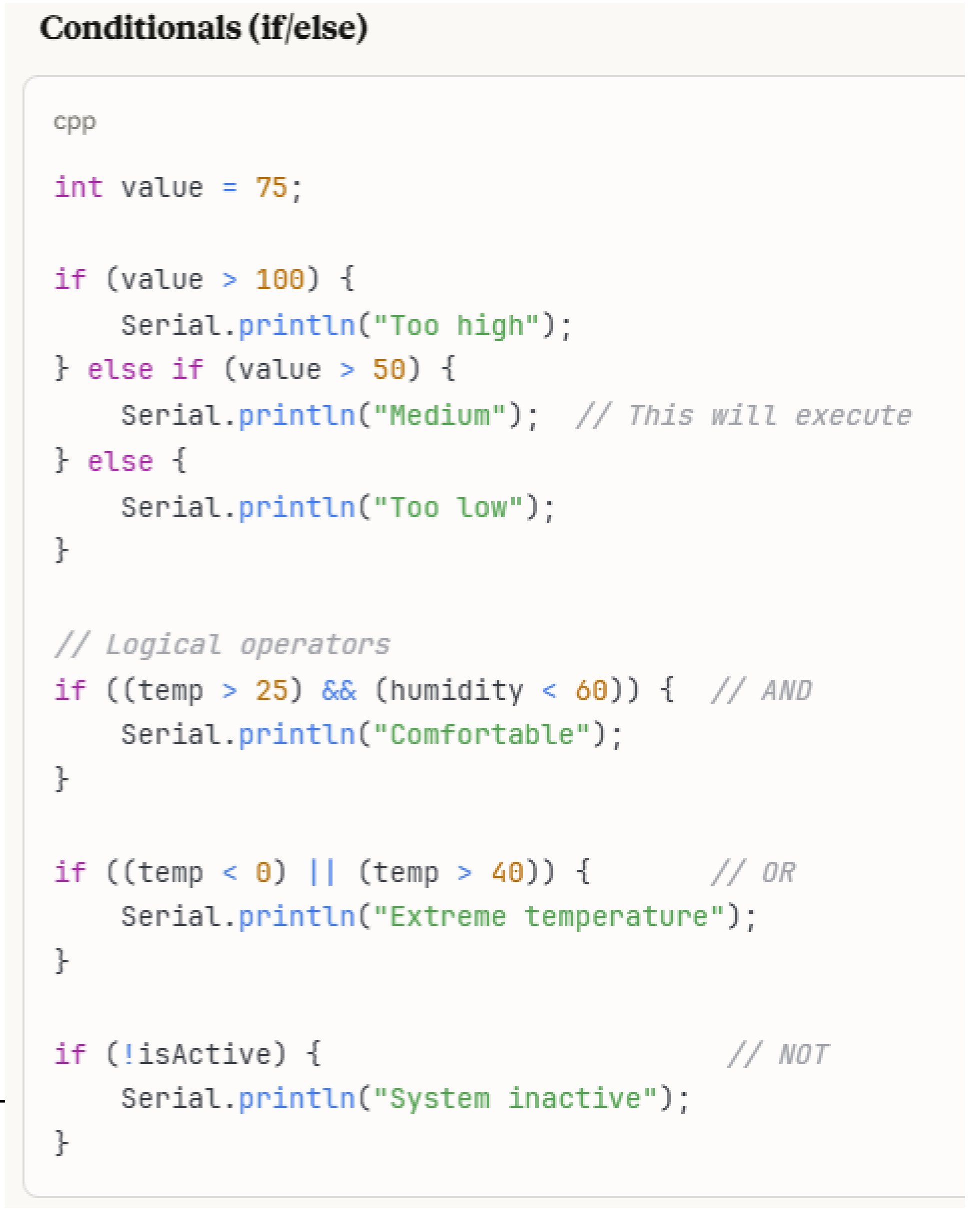
ARDUINO (C/C++) BASICS



Basic Operations

cpp

```cpp
// Math
int result = 10 + 5 * 2;        // 20 (order of operations)
float divided = 10.0 / 3.0;     // 3.333... (use .0 for float division)
int whole = 10 / 3;             // 3 (integer division)
int remainder = 10 % 3;         // 1 (modulo)

// Increment/decrement shortcuts
count++;                        // Same as: count = count + 1
count--;                        // Same as: count = count - 1
count += 5;                     // Same as: count = count + 5

// Comparison operators
5 > 3                           // true
10 == 10                        // true (equality, use ==, not =)
10 != 5                         // true (not equal)
```

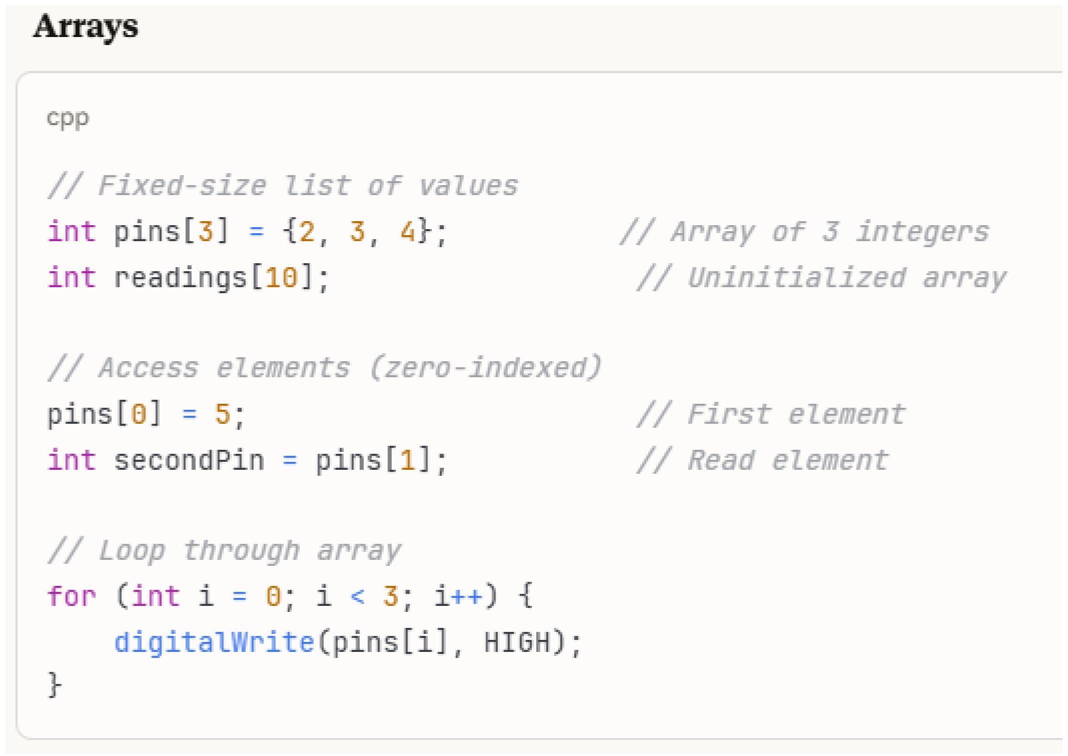


Conditionals (if/else)

cpp

```cpp
int value = 75;

if (value > 100) {
    Serial.println("Too high");
} else if (value > 50) {
    Serial.println("Medium");  // This will execute
} else {
    Serial.println("Too low");
}

// Logical operators
if ((temp > 25) && (humidity < 60)) {  // AND
    Serial.println("Comfortable");
}

if ((temp < 0) || (temp > 40)) {       // OR
    Serial.println("Extreme temperature");
}

if (!isActive) {                        // NOT
    Serial.println("System inactive");
}
```

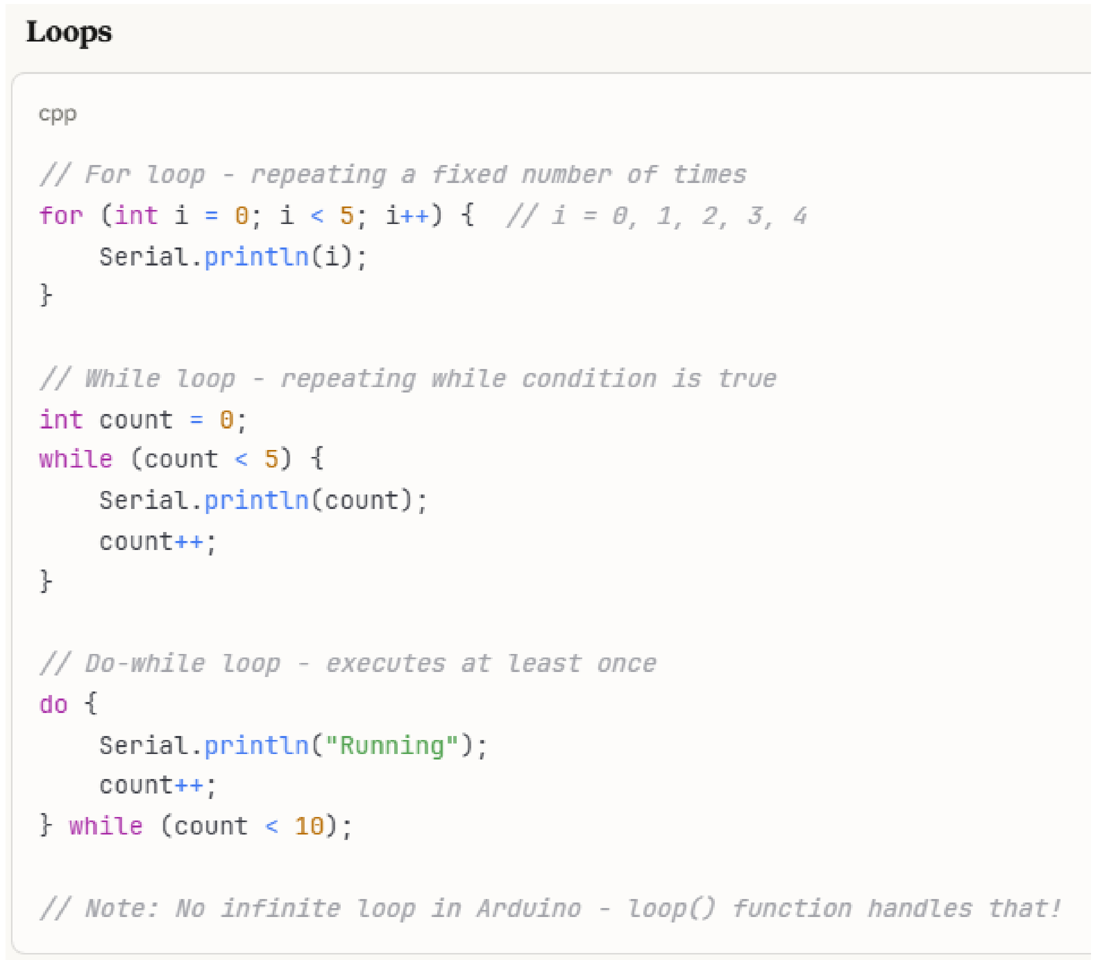
ARDUINO (C/C++) BASICS



Arrays

cpp

```cpp
// Fixed-size list of values
int pins[3] = {2, 3, 4};          // Array of 3 integers
int readings[10];                  // Uninitialized array

// Access elements (zero-indexed)
pins[0] = 5;                       // First element
int secondPin = pins[1];           // Read element

// Loop through array
for (int i = 0; i < 3; i++) {
    digitalWrite(pins[i], HIGH);
}
```

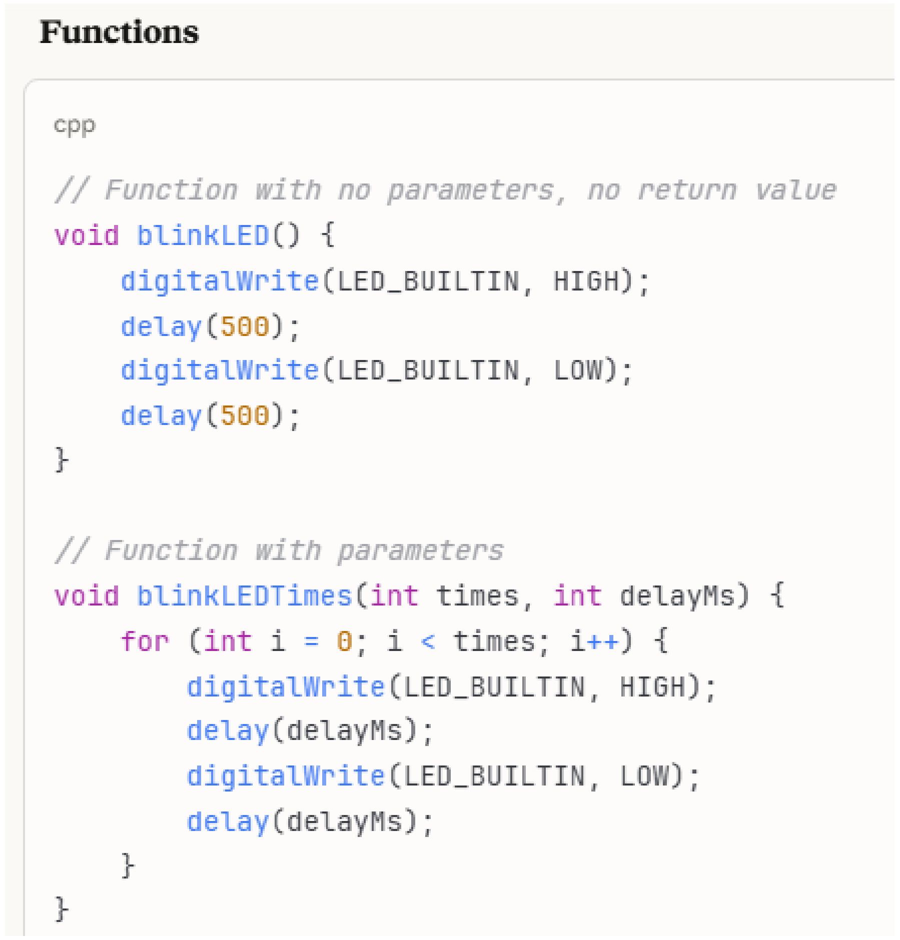


Loops

cpp

```cpp
// For loop - repeating a fixed number of times
for (int i = 0; i < 5; i++) {  // i = 0, 1, 2, 3, 4
    Serial.println(i);
}

// While loop - repeating while condition is true
int count = 0;
while (count < 5) {
    Serial.println(count);
    count++;
}

// Do-while loop - executes at least once
do {
    Serial.println("Running");
    count++;
} while (count < 10);

// Note: No infinite loop in Arduino - loop() function handles that!
```

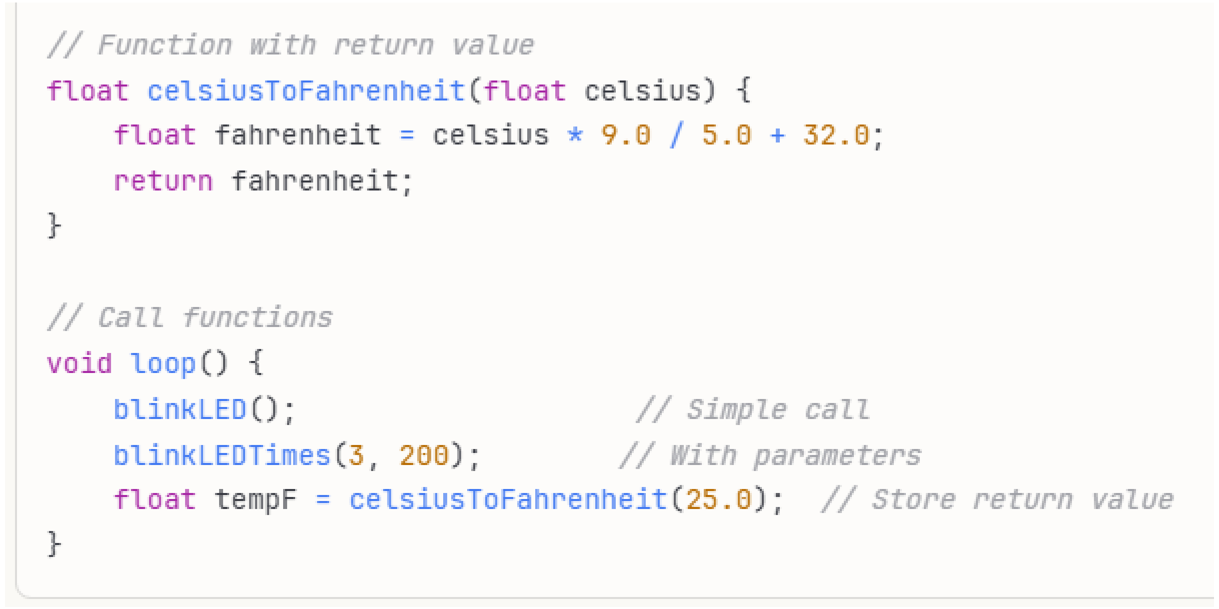
ARDUINO (C/C++) BASICS



Functions

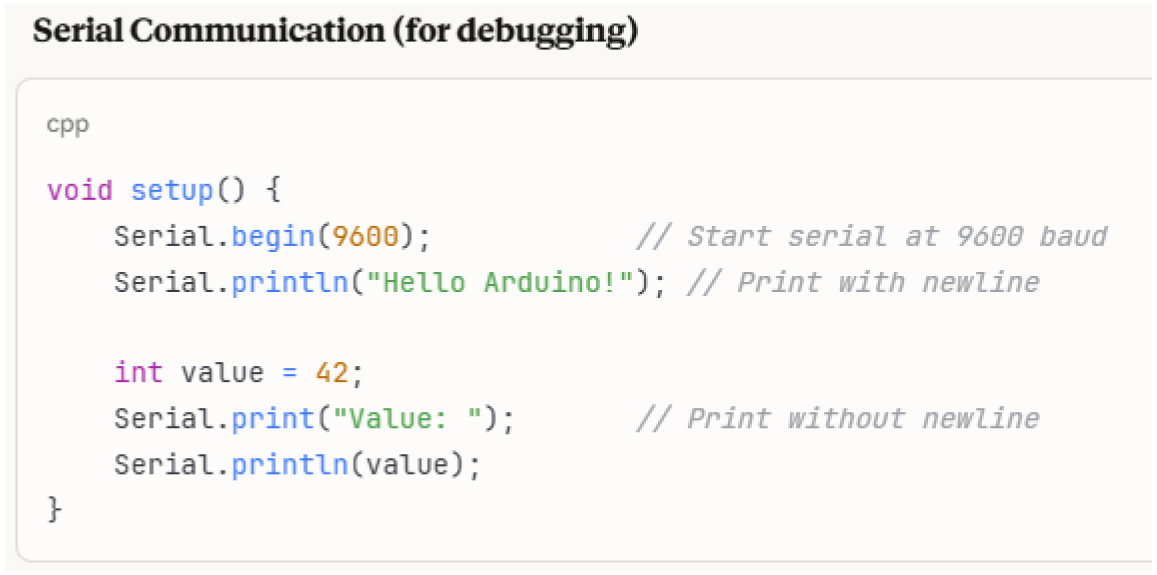
cpp

```cpp
// Function with no parameters, no return value
void blinkLED() {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(500);
    digitalWrite(LED_BUILTIN, LOW);
    delay(500);
}

// Function with parameters
void blinkLEDTimes(int times, int delayMs) {
    for (int i = 0; i < times; i++) {
        digitalWrite(LED_BUILTIN, HIGH);
        delay(delayMs);
        digitalWrite(LED_BUILTIN, LOW);
        delay(delayMs);
    }
}
```



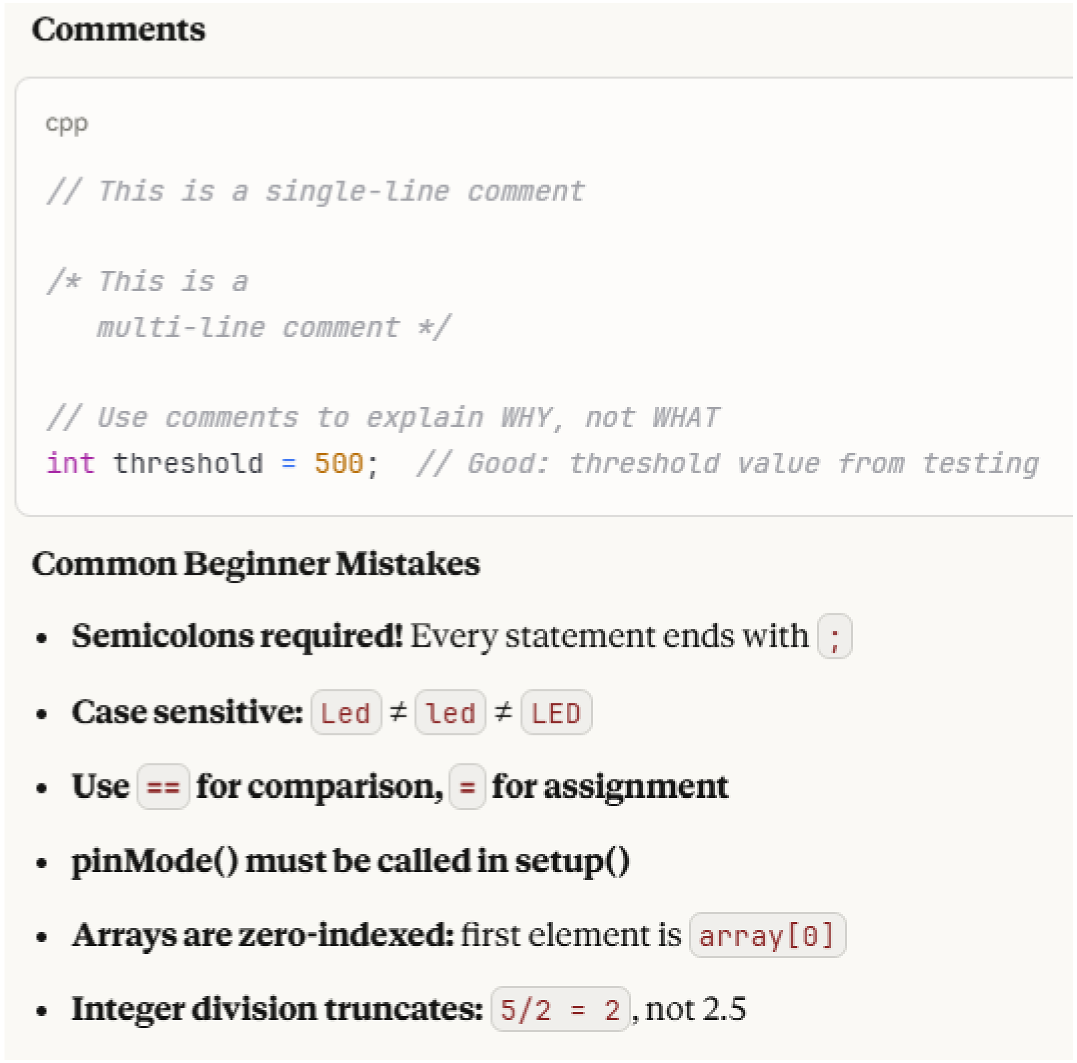
```cpp
// Function with return value
float celsiusToFahrenheit(float celsius) {
    float fahrenheit = celsius * 9.0 / 5.0 + 32.0;
    return fahrenheit;
}

// Call functions
void loop() {
    blinkLED();                        // Simple call
    blinkLEDTimes(3, 200);            // With parameters
    float tempF = celsiusToFahrenheit(25.0);  // Store return value
}
```

ARDUINO (C/C++) BASICS



Serial Communication (for debugging)

```cpp
void setup() {
    Serial.begin(9600);           // Start serial at 9600 baud
    Serial.println("Hello Arduino!"); // Print with newline

    int value = 42;
    Serial.print("Value: ");       // Print without newline
    Serial.println(value);
}
```



Comments

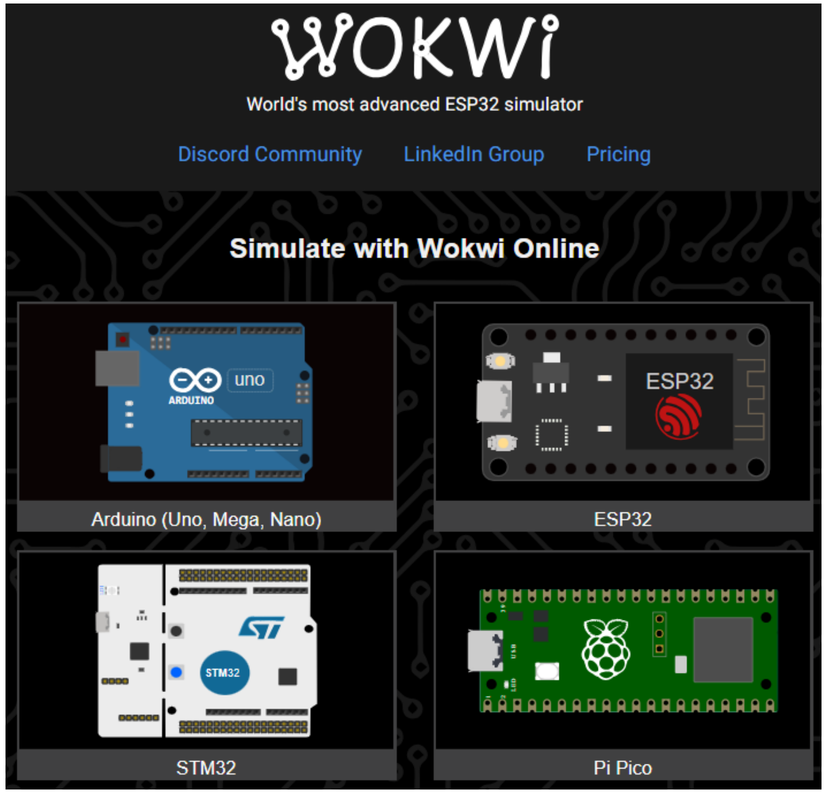
```cpp
// This is a single-line comment

/* This is a
   multi-line comment */

// Use comments to explain WHY, not WHAT
int threshold = 500;  // Good: threshold value from testing
```

Common Beginner Mistakes

- **Semicolons required!** Every statement ends with `;`
- **Case sensitive:** `Led` ≠ `led` ≠ `LED`
- **Use `==` for comparison, `=` for assignment**
- **pinMode() must be called in setup()**
- **Arrays are zero-indexed:** first element is `array[0]`
- **Integer division truncates:** `5/2 = 2`, not 2.5

HANDS-ON EXERCISES

INTRODUCTION TO WOKWI

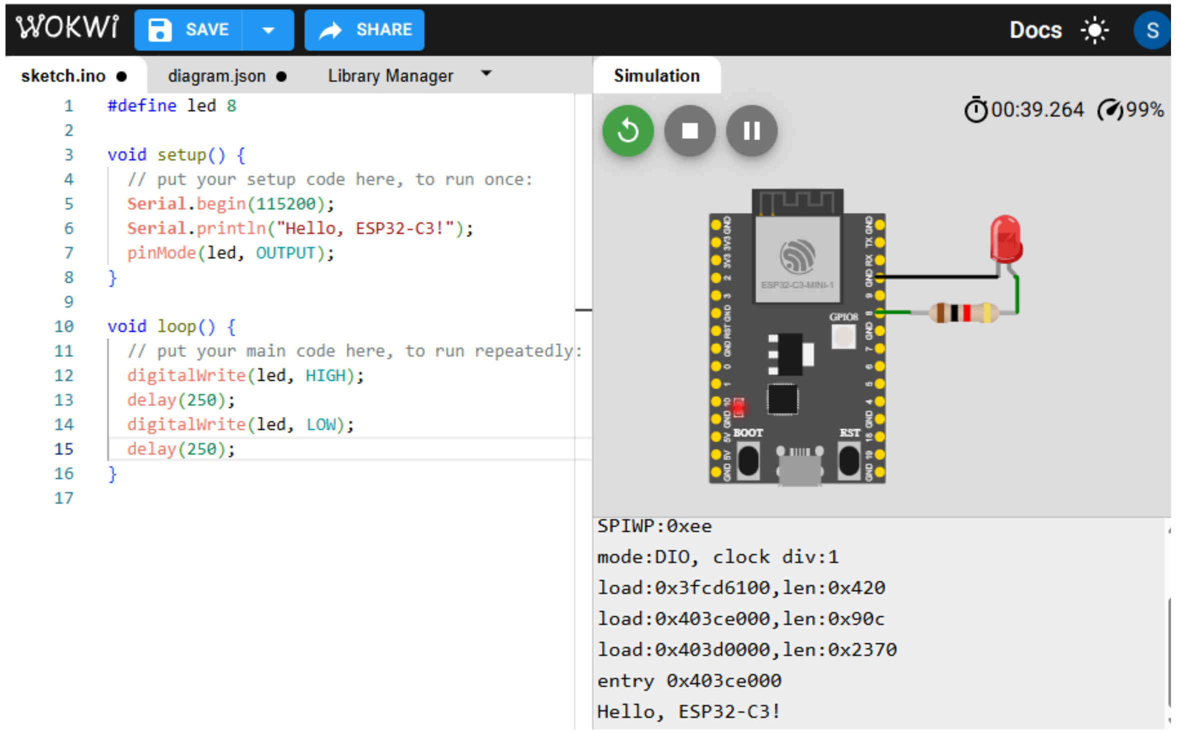

WOKWi
World's most advanced ESP32 simulator
Discord Community
LinkedIn Group
Pricing
Simulate with Wokwi Online
Arduino (Uno, Mega, Nano)
ESP32
STM32
Pi Pico

- online electronics simulator
- Supports multiple microcontroller families:
    - Arduino
    - ESP32
    - Raspberry Pi Pico
    - STM32
- Supports C/C++ and MicroPython, Rust
- Extensive component library
- Advanced features
    - Wifi simulation
    - Virtual logic analyzer
    - VS Code integration

Website: https://wokwi.com/

Makerpro Wokwi Tutorial

EXERCISE 1: ESP32-C3 HELLO WORLD (C/C++)



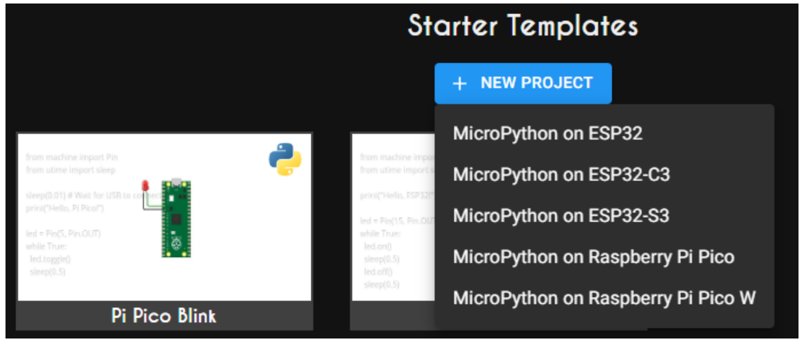
```
#define led 8

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  Serial.println("Hello, ESP32-C3!");
  pinMode(led, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(led, HIGH);
  delay(250);
  digitalWrite(led, LOW);
  delay(250);
}
```

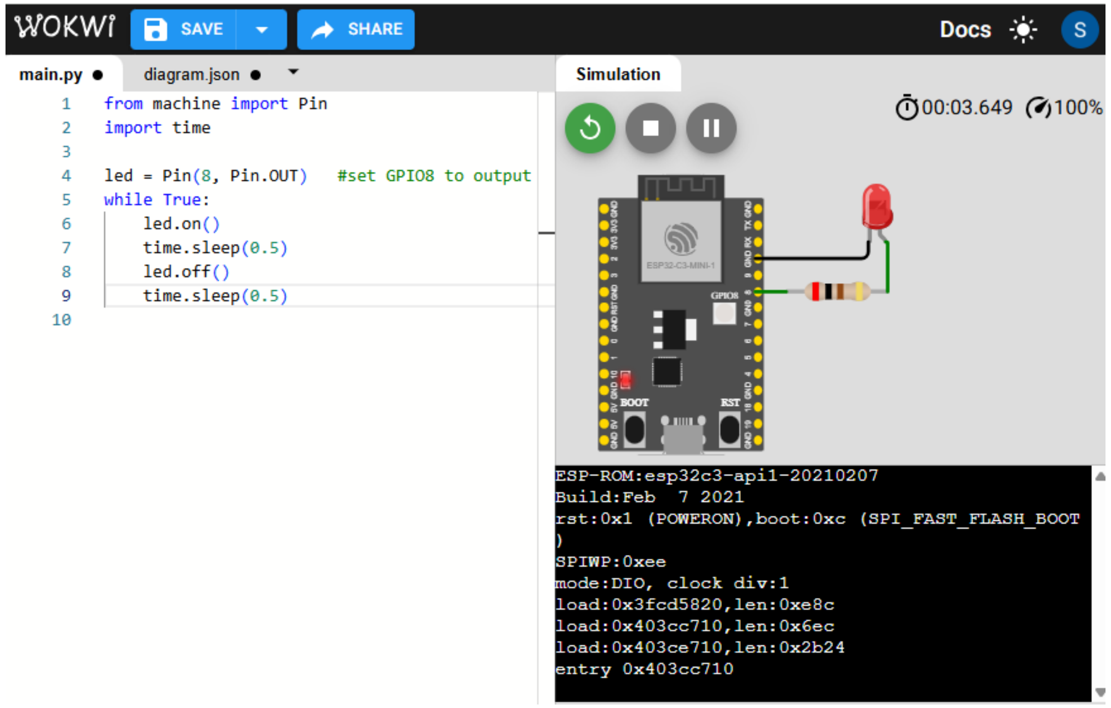
```
SPIWP:0xee
mode:DIO, clock div:1
load:0x3fcd6100,len:0x420
load:0x403ce000,len:0x90c
load:0x403d0000,len:0x2370
entry 0x403ce000
Hello, ESP32-C3!
```

- Go to the Wokwi website: https://wokwi.com
- Select ESP32 simulation, then ESP32-C3 starter template
- Add an LED and a resistor as shown in the diagram
- Wire the components:
  - LED cathode to GND, anode to resistor to GPIO8
  - Set resistor value to 200 ohms
- Enter the code on the left panel.
- Start the simulation
- Try changing the delay interval and run the simulation again

EXERCISE 2: ESP32-C3 HELLO WORLD (MPY)

- Start a new MicroPython project
- Scroll down to Starter Project, select MicroPython on ESP32-C3
- Add an LED and a resistor
- Wire the components as shown
  - LED: cathod to GND, anode to resistor to GPIO8
  - Set resistor value to 200 ohms
- Enter the MicroPython code as shown
- Run the simulation
- Modify the program:
  - Use led.toggle( ) instead of led.on( ) / led.off( )
  - Change delay interval to 250 ms



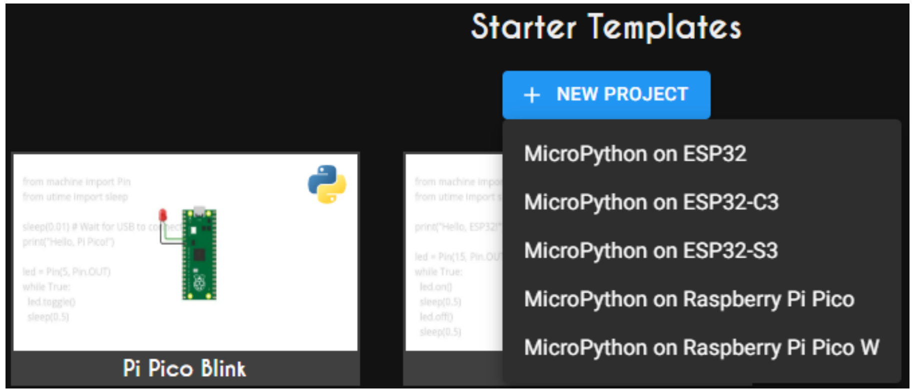


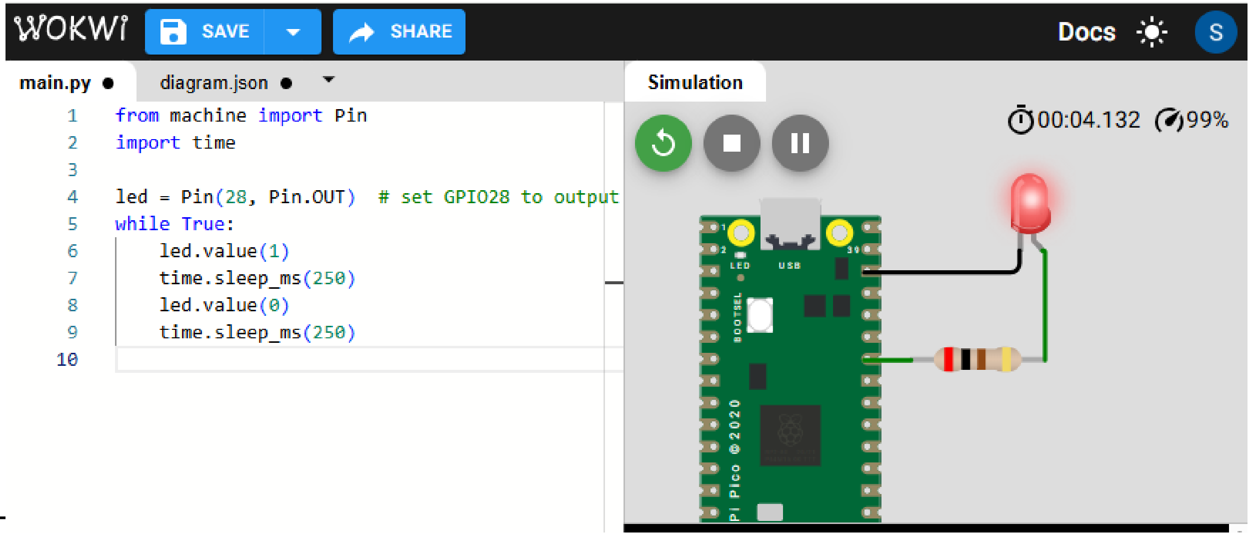
```python
from machine import Pin
import time

led = Pin(8, Pin.OUT)    #set GPIO8 to output
while True:
    led.on()
    time.sleep(0.5)
    led.off()
    time.sleep(0.5)
```

```
ESP-ROM:esp32c3-api1-20210207
Build:Feb  7 2021
rst:0x1 (POWERON),boot:0xc (SPI_FAST_FLASH_BOOT
)
SPIWP:0xee
mode:DIO, clock div:1
load:0x3fcd5820,len:0xe8c
load:0x403cc710,len:0x6ec
load:0x403ce710,len:0x2b24
entry 0x403cc710
```

EXERCISE 3: RP2040 HELLO (MICROPYTHON)

- Start a new MicroPython project
- Scroll down to Starter Project, select MicroPython on Raspberry Pi Pico
- Add an LED and a resistor
- Wire the components as shown
  - LED: cathode to GND, anode to resistor to GPIO8
  - Set resistor value to 200 ohms
- Enter the MicroPython code as shown
- Run the simulation
- Modify the program:
  - Use led.toggle( ) instead of led.value(1) / led.value(0)
  - Change delay interval to 1 s

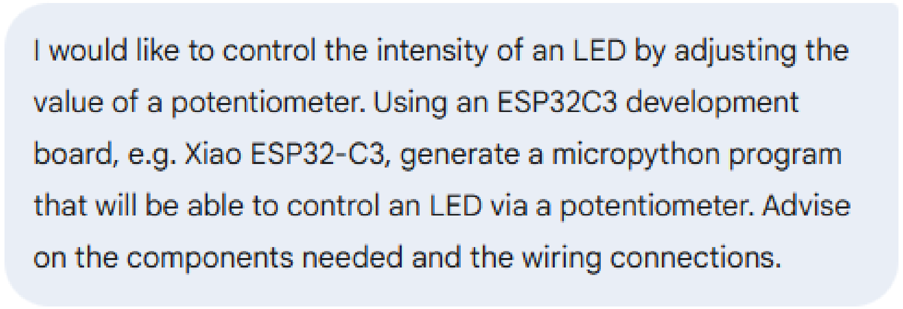


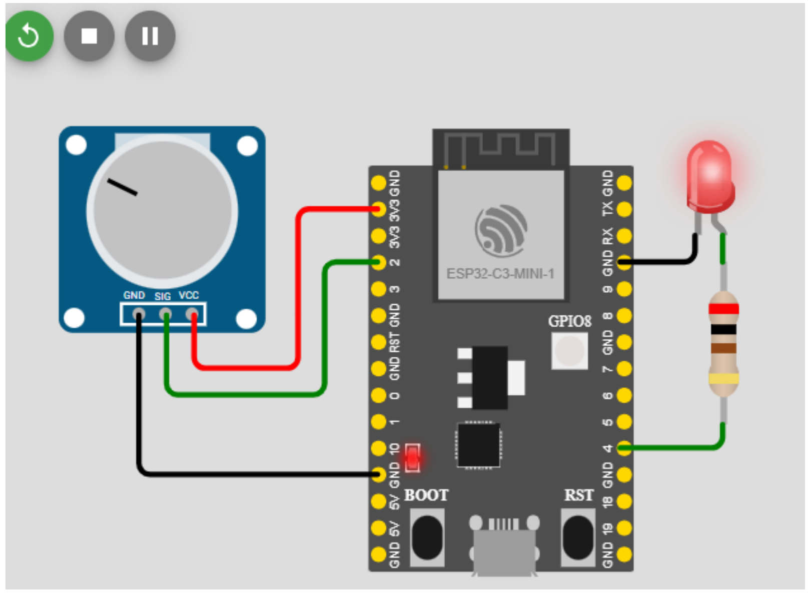


```
from machine import Pin
import time

led = Pin(28, Pin.OUT)  # set GPIO28 to output
while True:
    led.value(1)
    time.sleep_ms(250)
    led.value(0)
    time.sleep_ms(250)
```

EXERCISE 4: ESP32C3 CONTROLLING LED W/PWM

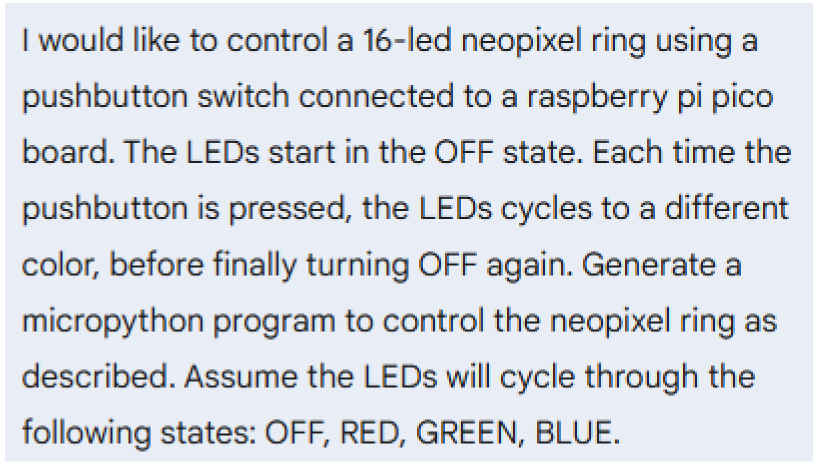
- Start a GenAI chat (deepseek, gemini)
- Write a prompt, asking GenAI for advice on writing a micropython program for the ESP32C3 to read input values from a potentiometer and use it to control the intensity of an LED.
- Start a Wokwi ESP32C3 micropython project. Examine the GenAI code, add components and wire the circuit.
- Copy the code to Wokwi and run the simulation.
- Did the simulation work?
- Determine the problem (if any) and ask GenAI to generate an updated program.
- Repeat the process until you get a working program.


I would like to control the intensity of an LED by adjusting the value of a potentiometer. Using an ESP32C3 development board, e.g. Xiao ESP32-C3, generate a micropython program that will be able to control an LED via a potentiometer. Advise on the components needed and the wiring connections.

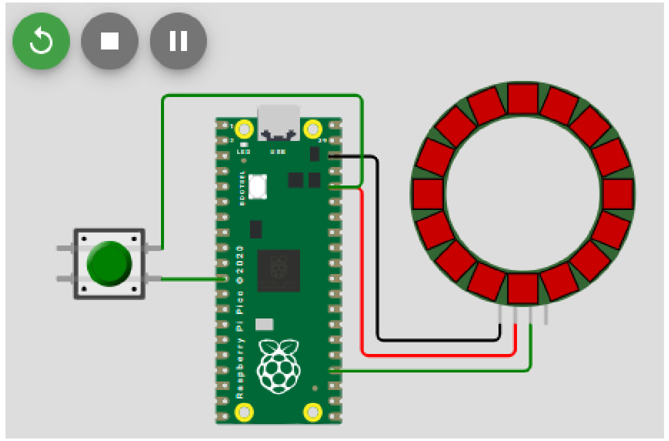


EXERCISE 5: PI PICO NEOPIXEL CONTROL

- Start a GenAI chat (deepseek, gemini)
- Write a prompt, asking GenAI for advice on writing a micropython program to control a neopixel ring display using a pushbutton connected to a Pi Pico board. The LEDs start in the OFF state. Each time the pushbutton is pressed, the LEDs will cycle through different colors, e.g. red, green, blue, then OFF.
- Start a Wokwi Raspberry Pi Pico micropython project. Examine the GenAI code, add components and wire the circuit.
- Copy the code to Wokwi and run the simulation.
- Did the simulation work?
- Determine the problem (if any) and ask GenAI to generate an updated program.
- Repeat the process until you get a working program.

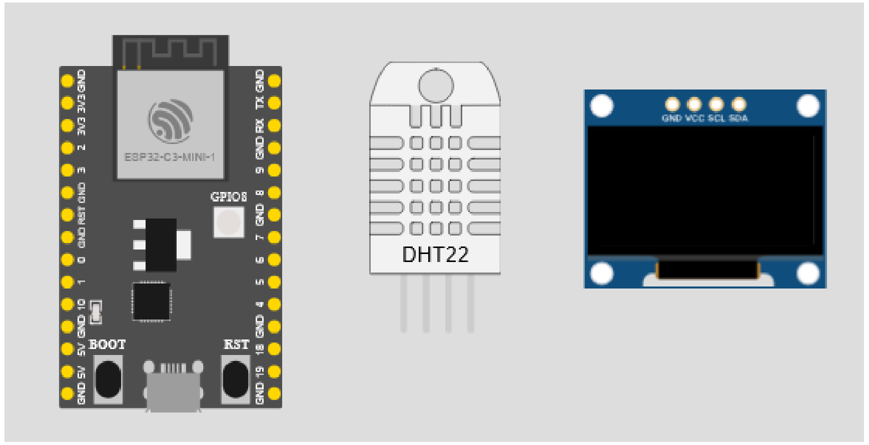


I would like to control a 16-led neopixel ring using a pushbutton switch connected to a raspberry pi pico board. The LEDs start in the OFF state. Each time the pushbutton is pressed, the LEDs cycles to a different color, before finally turning OFF again. Generate a micropython program to control the neopixel ring as described. Assume the LEDs will cycle through the following states: OFF, RED, GREEN, BLUE.



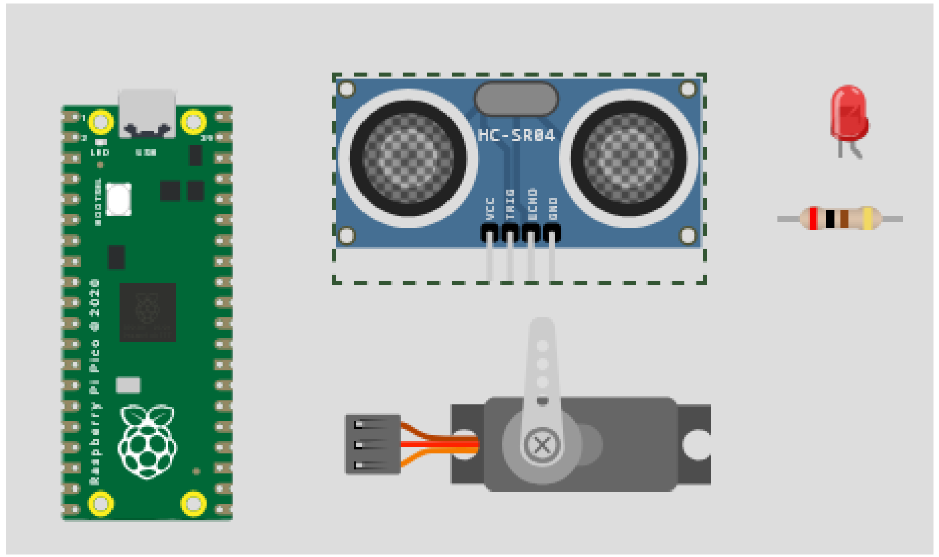
CHALLENGE EXERCISE 1

- You have been tasked to develop a portable environmental monitoring device, comprising of an ESP32-C3 board with 128x64 OLED screen which will sample and display the ambient temperature and humidity
- With the aid of GenAI:
  - Document the components and wiring diagram for the system
  - Write a micropython program to perform the required actions
  - Document the GenAI prompts and outputs, including the debugging/troubleshooting steps used to develop your solution



CHALLENGE EXERCISE 2

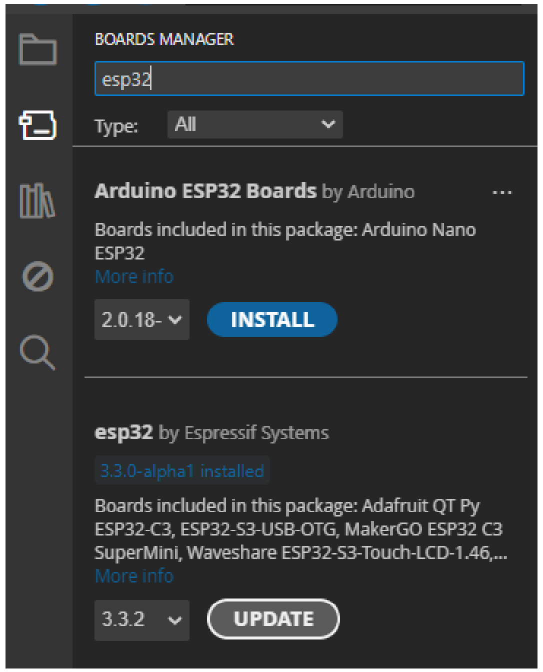
- To help maintain hygiene in hospital wards, you have been tasked to develop an automated sanitizer dispensing unit which comprises of a Pi Pico board, an ultrasonic sensor, a servo motor and an LED indicator.
- When the user places his hand(s) at a distance of < 3 cm from the ultrasonic sensor, the servo motor will be activated to dispense the sanitizer lotion. The LED will light up at the same time to indicate that sanitizer lotion is being dispensed.
- With the aid of GenAI:
  - Document the components and wiring diagram for the system
  - Write a micropython program to perform the required actions
  - Document the GenAI prompts and outputs, including the debugging/troubleshooting steps used to develop your solution



INSTALLING ESP32 SUPPORT FOR ARDUINO IDE

- Ref: Xiao ESP32-C3 Wiko: Getting Started
- Download and install the latest version of the Arduino IDE
- Launch the Arduino application
- Click the BOARDS MANAGER icon and type "esp32" in the search bar
- Select "esp32 by Espressif Systems" and click install

- Follow the same procedure to install support for the RP2040 MCU


BOARDS MANAGER

esp32

Type: All

Arduino ESP32 Boards by Arduino

Boards included in this package: Arduino Nano ESP32

More info

2.0.18- INSTALL

esp32 by Espressif Systems

3.3.0-alpha1 installed

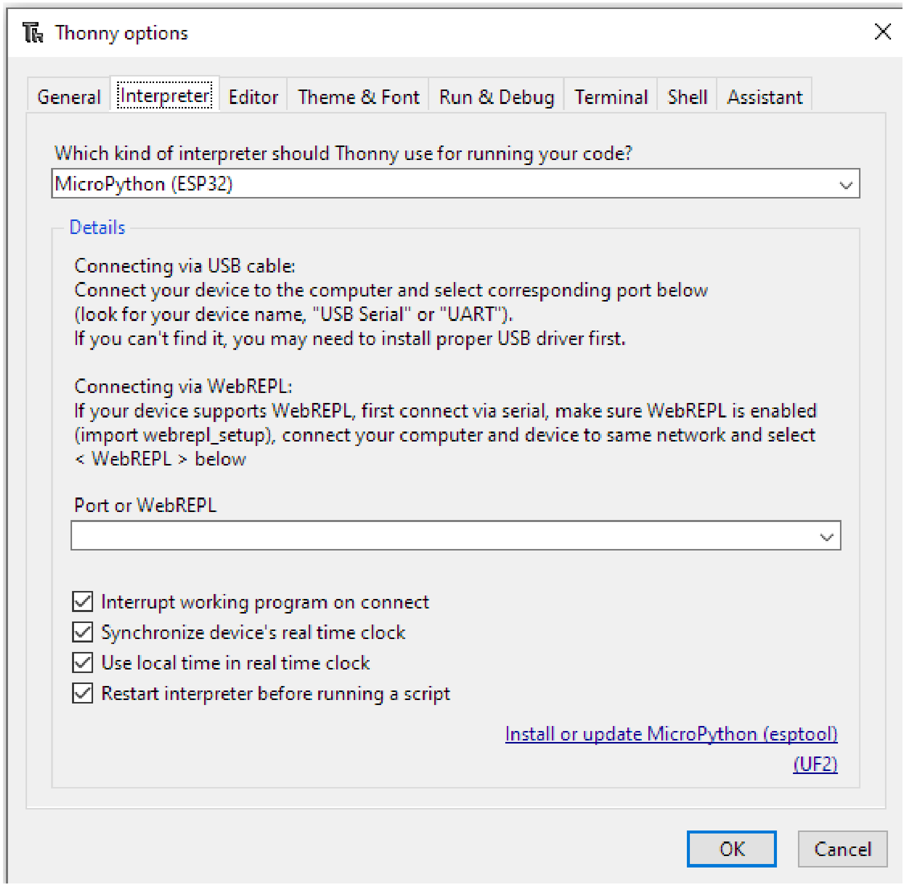
Boards included in this package: Adafruit QT Py ESP32-C3, ESP32-S3-USB-OTG, MakerGO ESP32 C3 SuperMini, Waveshare ESP32-S3-Touch-LCD-1.46,...

More info

3.3.2 UPDATE

INSTALLING THONNY IDE FOR MICROPYTHON

- Ref: Getting Started with Thonny MicroPython
- Go to https://thonny.org, download Thonny IDE
- Run the installer
- After Thonny IDE is installed, to program the microcontroller board, you need to flash the MicroPython firmware to the MCU board
- Connect your ESP32-C3 board, then press "Boot" + "Reset", release "Reset", followed by "Boot" to place the board in the bootloader mode
- Run Thonny IDE. Click Tools > Options > Interpreter



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

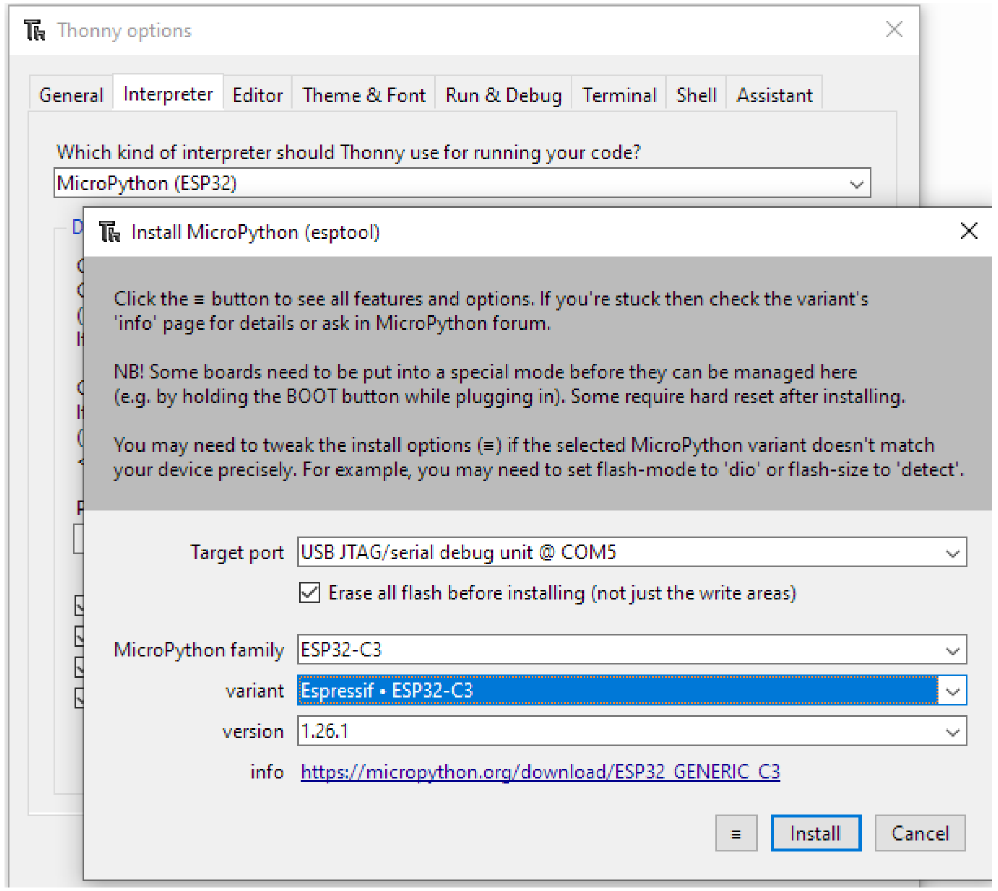
Port or WebREPL

☑ 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)

FLASHING MICROPYTHON FIRMWARE

- Click “Install or update MicroPython”
- Select the target port “USB JTAG/serial debug unit @COMx”
- Select MicroPython Family, variant, version
- Click “Install” to flash the firmware to the ESP32-C3 board
- Once the firmware has been uploaded, click the “Close” button, followed by OK to return to Thonny IDE



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

MicroPython (ESP32)

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

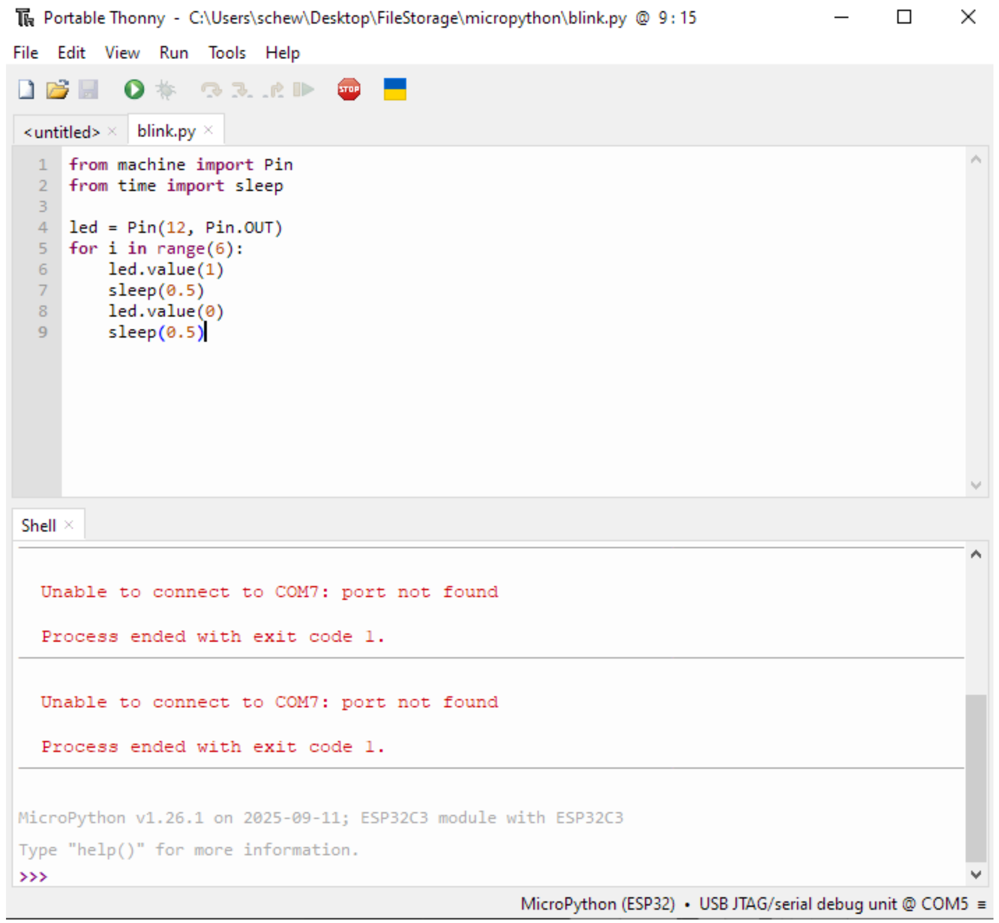
variant: Espressif • ESP32-C3

version: 1.26.1

info: https://micropython.org/download/ESP32_GENERIC_C3

VERIFYING MICROPYTHON FIRMWARE

- Once you return to Thonny IDE, verify that MicroPython has been correctly installed on the ESP32-C3 board
- You should see the MicroPython REPL prompt (>>>)
- If you do not see the REPL prompt, you may need to press "Reset" on the ESP320C3 board, "STOP" icon on Thonny IDE or check Tools > Options > Interpreter to verify that the correct COMport has been selected


Portable Thonny - C:\Users\schew\Desktop\FileStorage\micropython\blink.py @ 9 : 15
File Edit View Run Tools Help
<untitled>
blink.py
Shell
MicroPython (ESP32) • USB JTAG/serial debug unit @ COM5

```
from machine import Pin
from time import sleep

led = Pin(12, Pin.OUT)
for i in range(6):
    led.value(1)
    sleep(0.5)
    led.value(0)
    sleep(0.5)
```

```
Unable to connect to COM7: port not found

Process ended with exit code 1.

Unable to connect to COM7: port not found

Process ended with exit code 1.

MicroPython v1.26.1 on 2025-09-11; ESP32C3 module with ESP32C3
Type "help()" for more information.
>>>
```