

What is function:

A function is a named block of code that performs a specific task.

Instead of writing the same code again and again, we write it once inside a function and call it whenever needed.

How Functions Work

- When a function is defined, Python remembers the steps inside it.
- When the function is called, Python runs those steps.
- The same function can be called many times from different places in the program.

What Happens to Variables Inside a Function?

- Variables created inside a function exist only while the function is running.
- When the function finishes, those variables are cleared from memory.
- When the function runs again, it starts fresh with new values.
- This is why the same function gives different results when it is called with different values.

Why Functions Are Useful

Functions help us:

- avoid repeating code
- make programs easier to read
- fix or improve behavior by changing code in **one place**
- reuse the same logic in many projects

Function Definition Syntax in Python

General Syntax

def function_name(parameters):

statements

return value

- **def** → keyword used to define a function
- **function_name** → name of the function (should describe what it does). It should always be followed by a colon
- **parameters** → values passed into the function (optional)

- statements → instructions that run when the function is called
- return → sends a value back (optional)

Example:

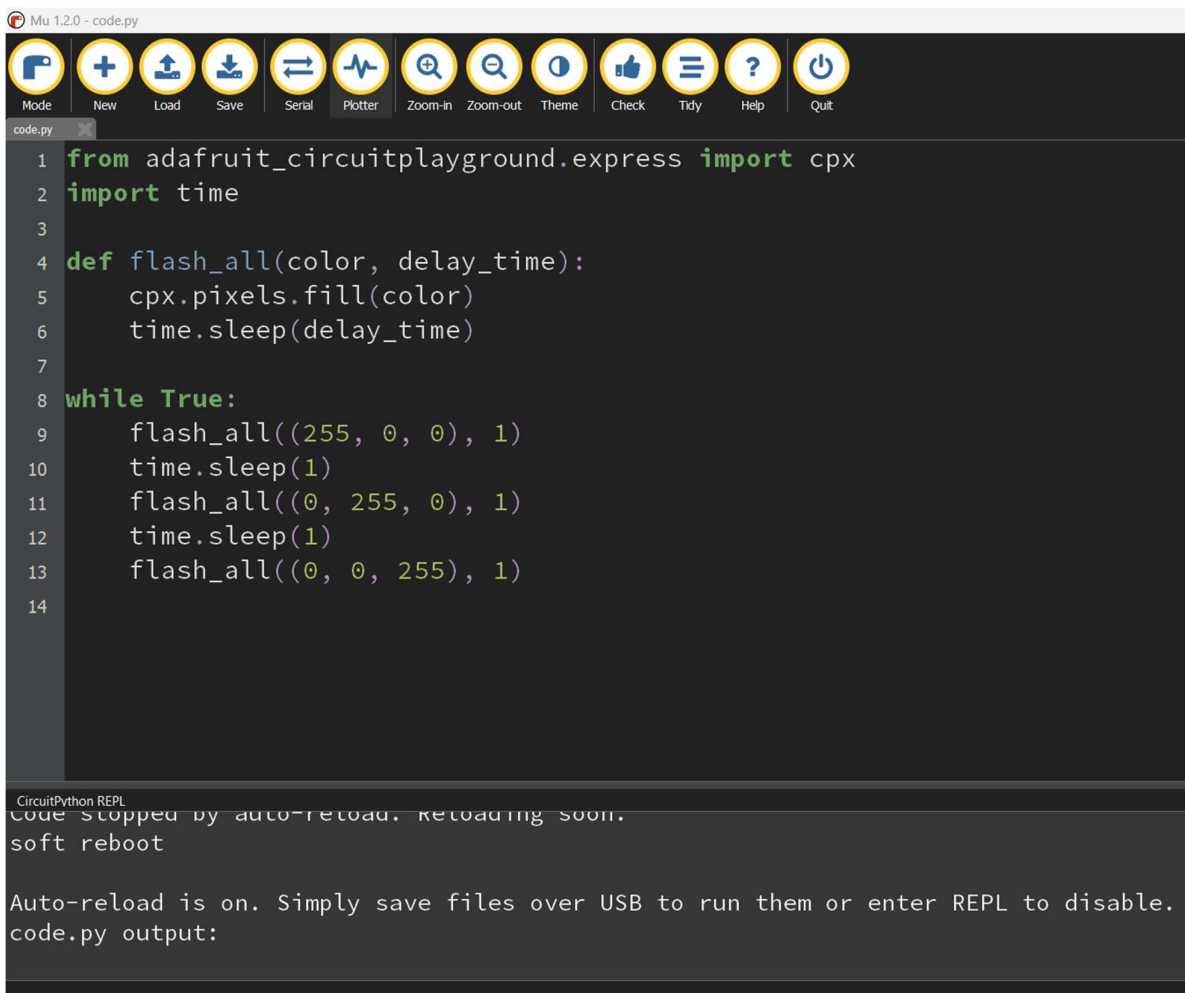
```
def multiply(a, b):
```

```
    result = a * b
```

```
    print("Answer is ", result)
```

```
multiply(5, 4)
```

Now let us write a function to flash Neopixels on the CPX:



The screenshot shows the Mu Python IDE interface. The top toolbar includes icons for Mode, New, Load, Save, Serial, Plotter, Zoom-in, Zoom-out, Theme, Check, Tidy, Help, and Quit. The main code editor displays a Python script named `code.py` with the following content:

```
1 from adafruit_circuitplayground.express import cpx
2 import time
3
4 def flash_all(color, delay_time):
5     cpx.pixels.fill(color)
6     time.sleep(delay_time)
7
8 while True:
9     flash_all((255, 0, 0), 1)
10    time.sleep(1)
11    flash_all((0, 255, 0), 1)
12    time.sleep(1)
13    flash_all((0, 0, 255), 1)
14
```

Below the code editor, the CircuitPython REPL output is visible, showing a message: "Code stopped by auto-reload. Reloading soon. soft reboot". At the bottom, a status message reads: "Auto-reload is on. Simply save files over USB to run them or enter REPL to disable. code.py output:".