CircuitPython on ESP32 Quick Start

Created by Carter Nelson

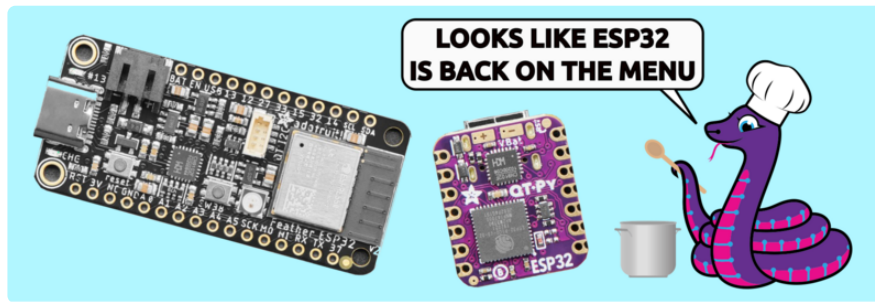https://learn.adafruit.com/circuitpython-with-esp32-quick-start
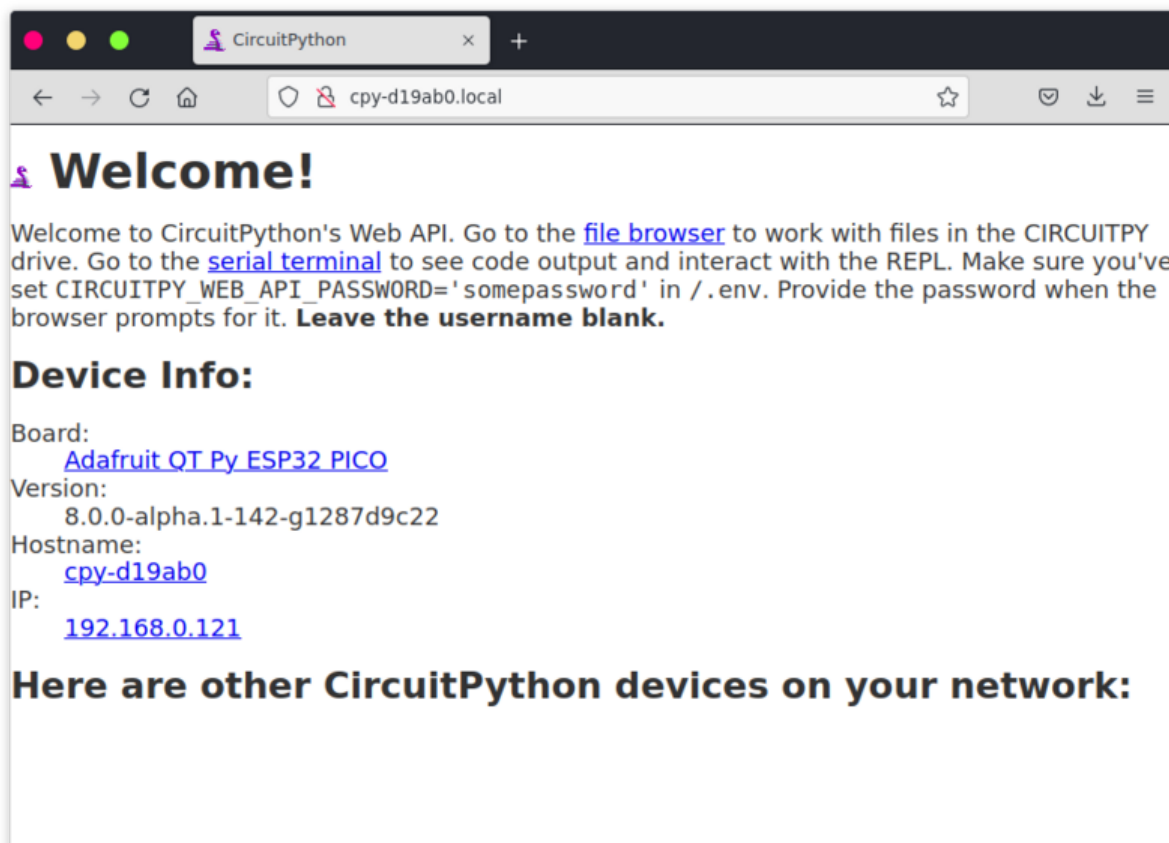
Last updated on 2025-07-01 05:00:21 PM EDT

Table of Contents

Overview

The Espressif ESP32 is a great and very popular processor used on lots of development boards. However, its lack of native USB has kept it from getting a CircuitPython build - for reasons (https://adafru.it/CiG)... Until now!

The new web workflow (https://adafru.it/10Bp) feature being added to CircuitPython 8 has brought the ESP32 back to the scene. For wifi enabled boards, like the ESP32, web workflow allows connecting to a board running CircuitPython over the local network using a web browser. Now it's easy to use the REPL or upload/download files using a browser!

In this guide, we show how to get the CircuitPython web workflow up and running on a supported ESP32 based board.

Remember! There Will Be No CIRCUITPY Folder

It is important to remember that the ESP32 lacks native USB support and therefore **no CIRCUITPY folder will show up**. Loading CircuitPython firmware and getting wifi set up is done over a serial connection (a.k.a a COM port) instead of a folder.

That's really the whole point of this guide. Because of this, the process for getting things set up on the ESP32 is different and more involved than other boards. So we're giving it some special attention. However, once set up, using web workflow is generally the same as other boards - you will upload library files, assets like fonts and images, and .py files for running your project

Variants with Native USB

Web Workflow can be used on wifi capable boards with native USB, like ones based on the EPS32-S2 or ESP32-S3. However, the native USB also allows for direct editing of code stored on the CircuitPython board, which is the generally easier and "standard" approach. This guide mainly focuses on the non-native USB ESP32 since it is more involved to get set up with CircuitPython and Web Workflow. But this does not mean Web Workflow is only available on ESP32.

Installing CircuitPython on ESP32

This is the tricky part. Well, the first tricky part at least. It's really just a simple matter of downloading the proper CircuitPython firmware `.bin` file and loading it onto the board. But since this is done over serial, it's not as easy as dragging a `.uf2` file to a BOOT folder, which is the more typical CircuitPython experience. (Remember, there's no CIRCUITPY or BOOT drive on the original ESP32!)

Download Firmware

First the firmware `.bin` file for the board must be downloaded. These can be downloaded from the main CircuitPython site linked below - just find specific board being used:

<div style="text-align:center">

CircuitPython Firmware

https://adafru.it/Em8

</div>

Installing the Firmware

This must be done over serial. There are two options:

- **Use the web serial browser based tool** (easier, but requires a Chrome webbrowser)
- **Use the command line based esptool.py tool** (harder, unless you happen to have esptool.py installed already and you know how to use it!)

The following pages cover these options in more detail. Choose your adventure!

Web Serial ESPTool

The WebSerial ESPTool was designed to be a web-capable option for programming Espressif ESP family microcontroller boards that have a serial based ROM bootloader. It allows you to erase the contents of the microcontroller and program up to 4 files at different offsets.

For boards that lack native USB, like the ESP32 or ESP32-C3 microcontroller, this is how any firmware like CircuitPython .bin files can be loaded. **There is no drag-and-drop to a folder option for these boards.**

For boards with native USB, like ESP32-S2, -S3, etc. this is how the UF2 bootloader .bin file can be loaded. Once the UF2 bootloader is on, firmware like CircuitPython .uf2 files can be drag-and-dropped to a **BOOT** folder.

This tool is a good alternative for folks who cannot run Python `esptool.py` on their computer or are having difficulty installing or using `esptool.py`

Enabling Web Serial

You will have to use the Chrome or Chromium-based browser for this to work. For example, Edge and Opera are Chromium (https://adafru.it/10BL). Safari and Firefox, etc are not supported - they have not implemented Web Serial (https://adafru.it/10BM)!
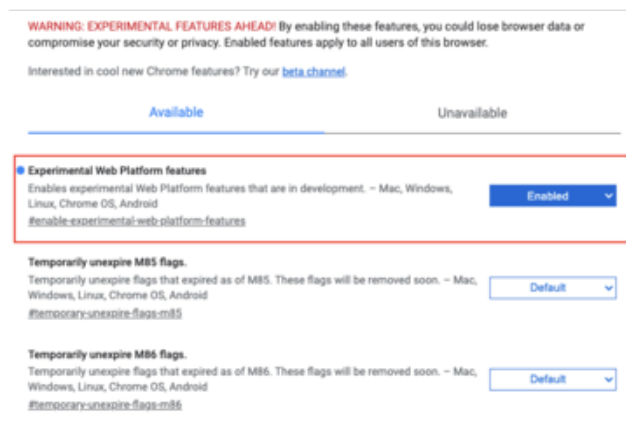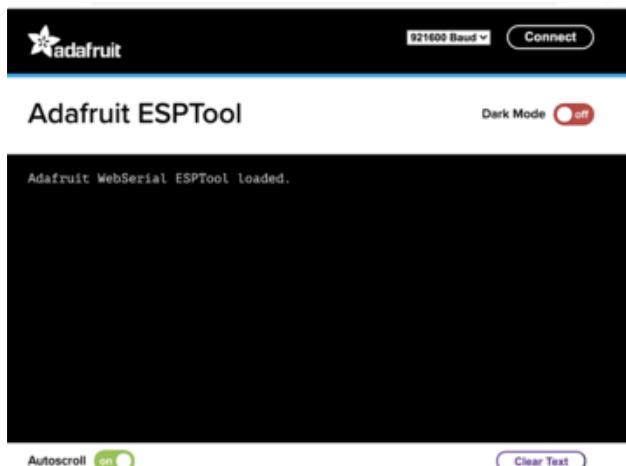

If you have a very old version of Chrome, you'll need to enable the Serial API, which is really easy.

Visit **chrome://flags** from within Chrome. Find and enable the **Experimental Web Platform features**

Restart Chrome

Connecting

Before you can use the tool, you will need to put your board in bootloader mode and connect. Here are the steps:


In the **Chrome browser** visit https://adafruit.github.io/Adafruit_WebSerial_ESPTool/ (https://adafru.it/PMB). You should see something like the image shown.
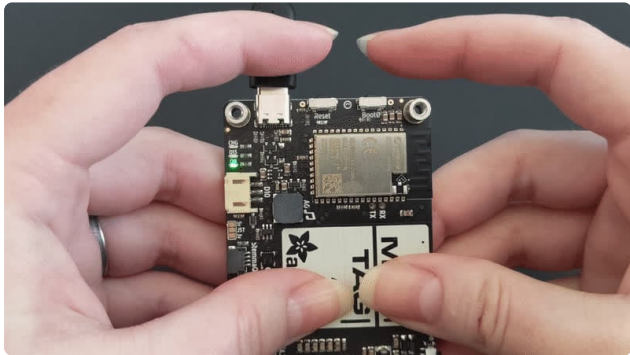
For all ESP32-family boards, you can enter the ROM bootloader by holding down the **BOOT** button while clicking **RST**.

If you have an ESP32 board without BOOT (say because its an original ESP32, not -S2 or -S3, etc) you can skip this step because there's an auto-boot circuit.
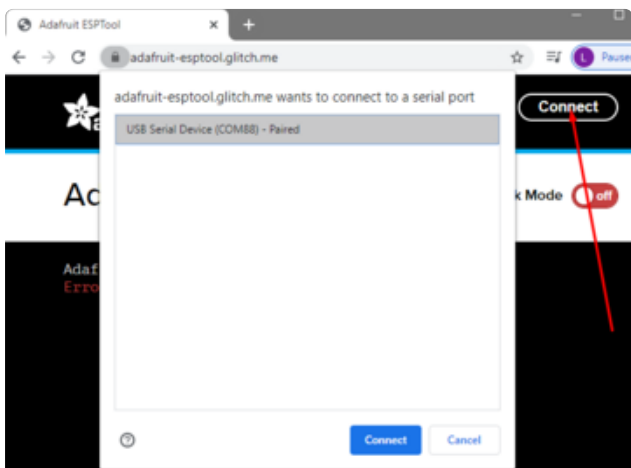
Press the **Connect** button in the top right of the web browser. You will get a pop up asking you to select the COM or Serial port. Look for something with **ESP32**, **JTA G Loader**, **SLAB**, or **FTDI** in the name.

Remember, you should remove all other USB devices so only the target board is attached, that way there's no confusion over multiple ports!

On some systems, such as MacOS, there may be additional system ports that appear in the list.

The Javascript code will now try to connect to the ROM bootloader. It may timeout for a bit until it succeeds. On success, you will see that it is **Connected** and will print out a unique **MAC address** identifying the board along with other information that was detected.
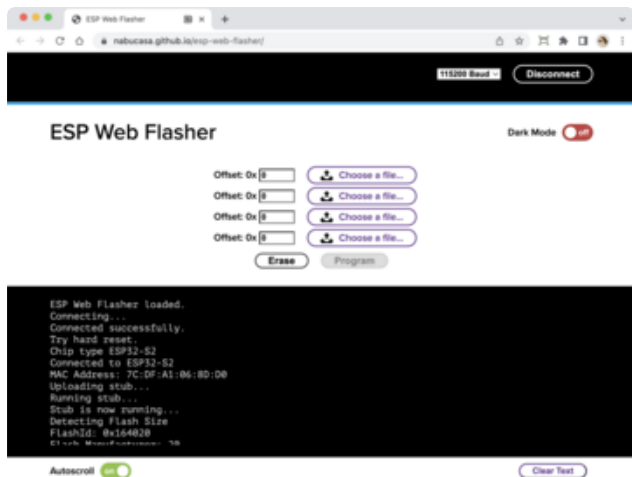
Once you have successfully connected, the command toolbar will appear.

Erasing the Contents

If you would like to erase the entire flash area so that you can start with a clean slate, you can use the erase feature. We recommend doing this if you are having issues.


To erase the contents, click the Erase button. You will be prompted whether you want to continue. Click OK to continue or if you changed your mind, just click cancel.
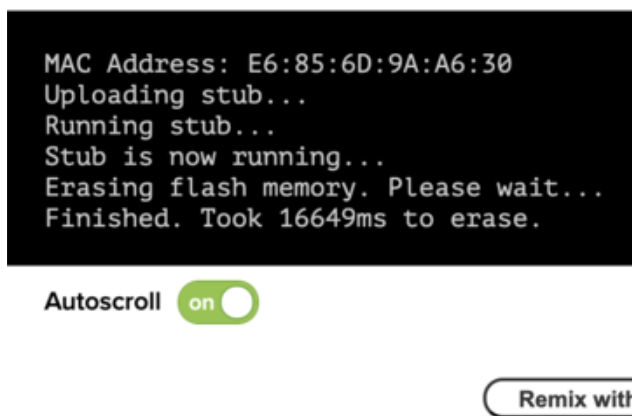
You'll see "Erasing flash memory. Please wait..." This will eventually be followed by "Finished." and the amount of time it took to erase.

Do not disconnect! Immediately continue on to Programming the ESP Microcontroller.

Programming the ESP Microcontroller
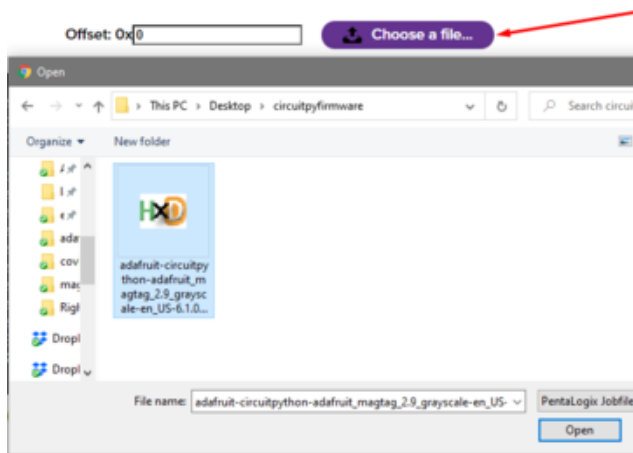
You can click on **Choose a file...** from any of the available buttons. It will only attempt to program buttons with a file and a unique location. Then select the .bin file(s) - **not the UF2 file**!

Verify that the **Offset** box next to the file location you used is 0x0.
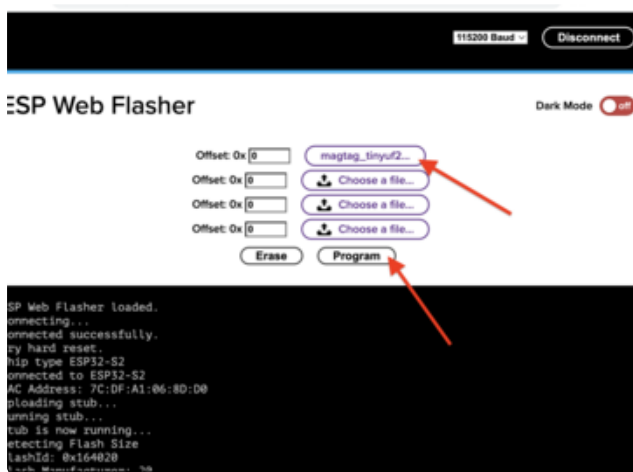
Once you choose a file, the button text will change to match your filename. You can then select the **Program** button to start flashing.

A progress bar will appear and after a minute or two, you will have written the firmware.

After using the tool, press the reset button to get out of bootloader mode and launch the new firmware!

For boards lacking native USB, like the ESP32 and ESP32-C3, **no folder will show up after pressing reset**. If CircuitPython firmware was loaded, the REPL can be accessed over a serial COM port.

For other boards, like ESP32-S2, -S3, etc., a **BOOT** folder should show up. A [CircuitPython UF2](https://adafru.it/Em8) (https://adafru.it/Em8) file can now be copied over to the **BOOT** folder, after which the **CIRCUITPY** folder should then show up.

Command line ESPTool

Before you can load your firmware, you'll need to install `esptool`. Follow the instructions found [here](https://adafru.it/Zel) (https://adafru.it/Zel).

Next up you need to determine the Serial Port the board is connected to.

- For ESP32 boards, just plug it into your computer.
- For ESP32-S2/S3/C3 boards, plug in and then launch 'bootloader' mode by holding down **BOOT** or **B0** button while clicking reset. Then release both buttons.

If you're on something like a Mac or Linux, run ls `/dev/tty.*` to find all serial devices, it will be called something like `/dev/tty.usbserial` or `/dev/ttyUSB0` or `/dev/ttyACM0` or `/dev/tty.SLABtoUSB` or similar!

It will probably not be a shorter name /dev/tty23.

If you're using Windows, the port will be something like `COM2` (check the Device Manager to determine your port)

> If you are having difficulty determining which port is the board, try the "ls /dev/tty.*" or Device Manager listing WITHOUT the board plugged in, then plug in the board and look again - what name is new that wasn't there before?

Once you have `esptool` installed, you will first want to erase the flash on your ESP32 board, it's also a great way to determine that you are able to connect. You can do so by running something similar to the following command. Make sure you update the port to match the serial port to which your board is connected, i.e. change `/dev/tty.usbserial-1144440` to match your connection.

```
esptool.py --chip esp32 --port /dev/tty.usbserial-1144440 erase_flash
```


or - if you're not 100% sure which ESP32 chip you have, you can just specify the port

```
esptool.py --port COM13 erase_flash
```

Once the flash is successfully erased, you can load your firmware `.bin` file by running something similar to the command below. Make sure you update the following:

- The port to match the serial port to which your board is connected, i.e. change /dev/tty.usbserial-1144440 to match your connection.
- The **firmware.bin** file name to match the firmware you downloaded, i.e. change **firmware.bin** to something like **esp32spiram-20220117-v1.18.bin** or **adafruit-circuitpython-esp32-8.0.bin**
- This command assumes you will be loading it to address `0x0`, as we do for circuitpython. **This may not be true for your firmware! check the documentation for the offset if necessary**

Load the firmware using something similar to the following command, with the above changes:

```
esptool.py --port /dev/tty.usbserial-1144440 write_flash -z 0x0 firmware.bin
```

To keep things simple we don't specify the chip or a faster upload speed - you can always add those options if you need to upload to many chips quickly

It's always a good idea to power cycle by unplugging-replugging, or pressing the reset button, to make sure the new firmware is running.

Setting up Web Workflow

Web workflow is enabled when a file named **settings.toml** is added to the root folder of the CircuitPython file system. This file contains local wifi info and other settings. [More info here](https://adafru.it/10Bp) (https://adafru.it/10Bp).

Now, normally creating a file is super easy when the board shows up as a disk drive...but as mentioned before, that's not possible on the original ESP32 because it does not have native USB so it cannot show up as a disk drive. So we have to be a little more creative!

To start out, the minimal contents of this file are:

```
CIRCUITPY_WIFI_SSID = "wifissid"
CIRCUITPY_WIFI_PASSWORD = "wifipassword"
CIRCUITPY_WEB_API_PASSWORD= "webpassword"
```

with each item updated with local specifics.

- `wifissid` - replace with local wifi network name
- `wifipassword` - replace with local wifi network password
- `webpassword` - used when connecting to the board via web browser, set to whatever

There are a couple of options for creating this file.

> Creating the settings.toml file will also enable the wifi hardware, which may be off by default.

Option 1: Create **settings.toml** file via REPL

The **settings.toml** file is simple enough that it can be created with a few simple Python commands directly via the REPL.

What is the REPL?

https://adafru.it/Awz

Here are the basic commands to use. **Note these need to be updated with local wifi specifics. Note also where double quotes and single quotes are used.**

```
f = open('settings.toml', 'w')
f.write('CIRCUITPY_WIFI_SSID = "wifissid"\n')
f.write('CIRCUITPY_WIFI_PASSWORD = "wifipassword"\n')
f.write('CIRCUITPY_WEB_API_PASSWORD = "webpassword"\n')
f.close()
```

- Replace `wifissid` with the name of your local wifi network
- Replace `wifipassword` with your local wifi password
- The other password, `webpassword`, is used when you access the board via a web browser. Set this to whatever you want.
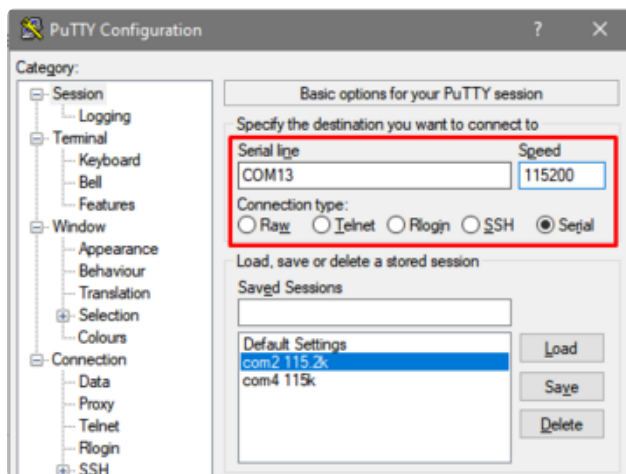
If you have a terminal program like PuTTY, minicom, screen or similar and you know how to use them - connect to the ESP32 board at the correct port name and 115200 baud

Once connected, you can press the **Reset** button to kick the firmware, then hit return a few times to get to the REPL prompt. Now you can copy and paste the lines above (with your correct SSID/password!)

Don't forget, ESP32 does not support 5 GHz networks, so use your 2.4 GHz SSID if you have two.

Now press the **Reset** button again, you will see the ESP32 reboot. This time, it should get an IP address. If you have a fairly smart terminal program, the IP address will appear in the title bar.
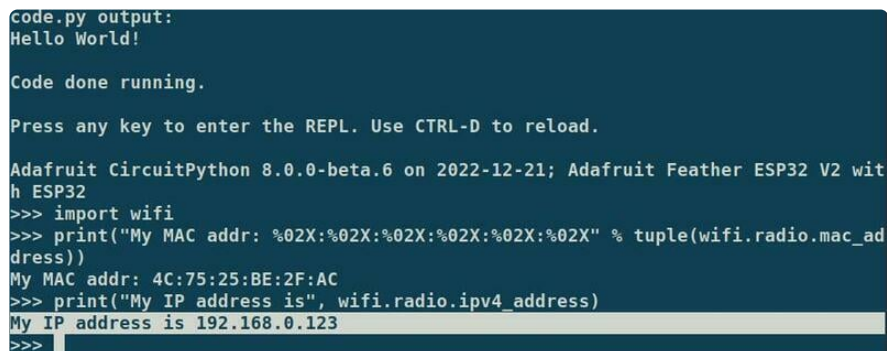
Alternatively, or if you want more deets - you can always query the board over the REPL to ask it the MAC address and IP address:

```
import wifi

print("My MAC addr: %02X:%02X:%02X:%02X:%02X:%02X" % tuple(wifi.radio.mac_address))
print("My IP address is", wifi.radio.ipv4_address)
```


If that's not working either, try asking what SSID's it thinks it's seeing to make sure you have no typos!

```
import wifi

print("Available WiFi networks:")
for n in wifi.radio.start_scanning_networks():
    print("\t%s\t\tRSSI: %d\tChannel: %d" % (str(n.ssid, "utf-8"), n.rssi,
n.channel))
wifi.radio.stop_scanning_networks()
```

Note that since this code has an indented part in it, you shouldn't just paste it in directly into the REPL because it won't handle the tab/spaces correctly. Instead, once you hit **Return** to get to the REPL, type in **Control-E** to enter paste mode . Then paste in the text and type **Control-D** to finish and run

Check that your SSID appears properly - if not, maybe its 5 GHz, maybe its not typed correctly, etc!

Option 2: Create **settings.toml** file using Thonny

This option requires installing additional software - the Thonny Python IDE (https:// adafru.it/Qb6). Thonny provides file access and a text editor which makes creating the **settings.toml** file a little more streamlined than using direct REPL commands. **This technique requires Thonny 4.0.0 or later.**
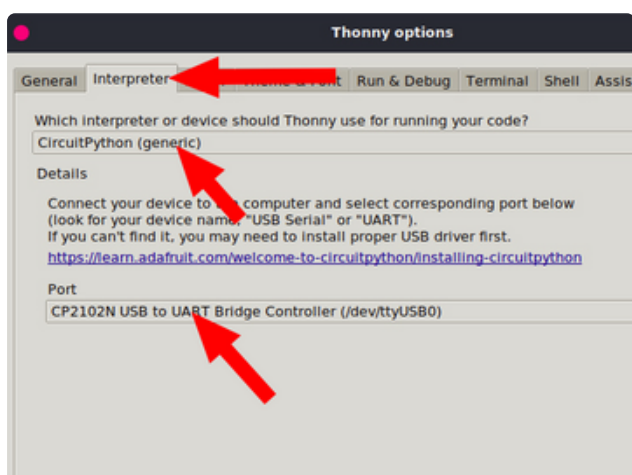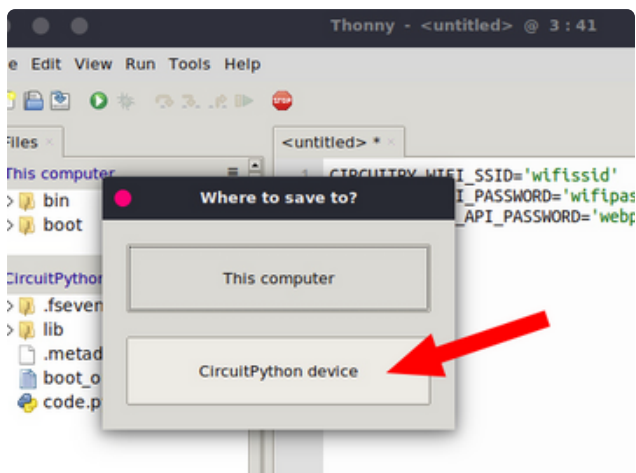

In Thonny, open the **Tools -> Options** dialog and select the **Interpreter** tab.

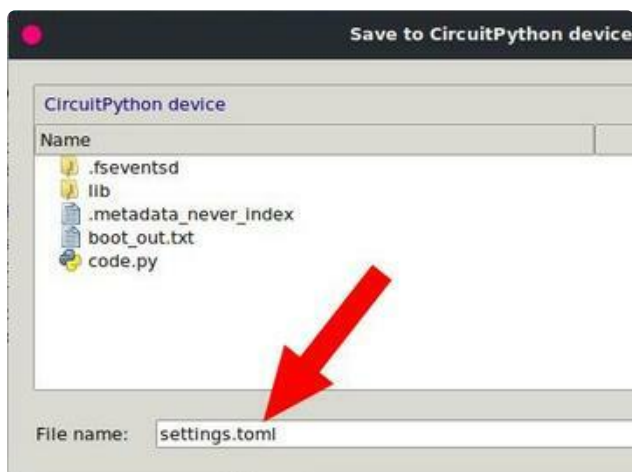Set interpreter to CircuitPython (generic) and the COM port as needed.
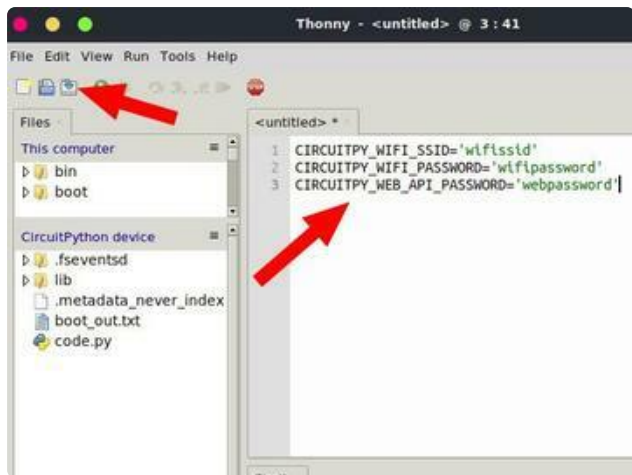
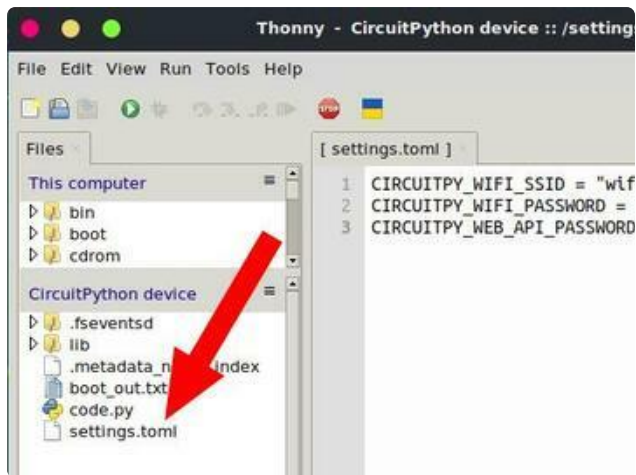Enter the file contents in the editor window. Update `wifissid` etc. as needed.

Click save.

Select **CircuitPython Device**

Save the file as **settings.toml**.

Now the **settings.toml** file shows up on the CircuitPython device.

Connecting via Web Browser

Once the web workflow is enabled and the board connects to the local wifi network with an IP address, the board can be accessed by opening a web browser and entering the board's network address. But what is the address to enter?

Connect using MDNS Address

The CircuitPython web workflow uses [MDNS (https://adafru.it/10BW)](https://adafru.it/10BW) so that connecting to the board can be as simple as using the address `circuitpython.local`. Try using that first and if it works, great. Open a web browser and navigate to:

```
http://circuitpython.local
```

Or just click the button below:

> **circuitpython.local**

https://adafru.it/10BX

Connect using IP Address

If MDNS does not work for some reason, the fallback approach is to use the actual IP address, like `192.168.0.121`, the CircuitPython board was assigned when it connected to the local wifi network.
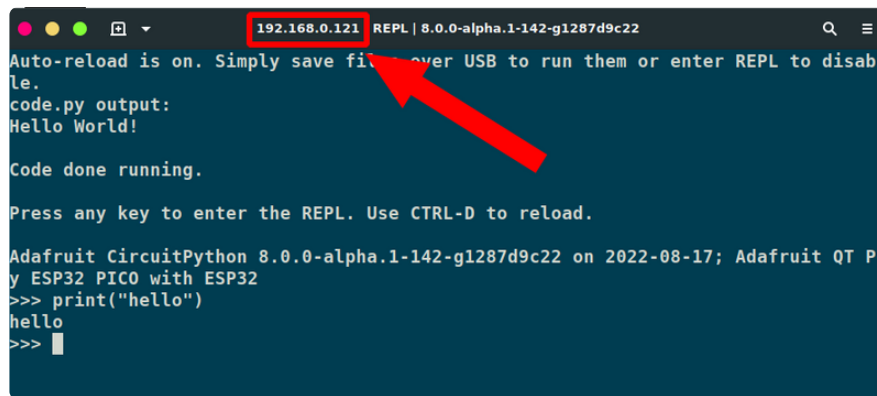
```
http://192.168.0.121
```

There are several options for determining this address.

Terminal Window Title Bar

The web workflow serial output includes some special escape sequences that are intended to update the window title bar of the terminal program being used.

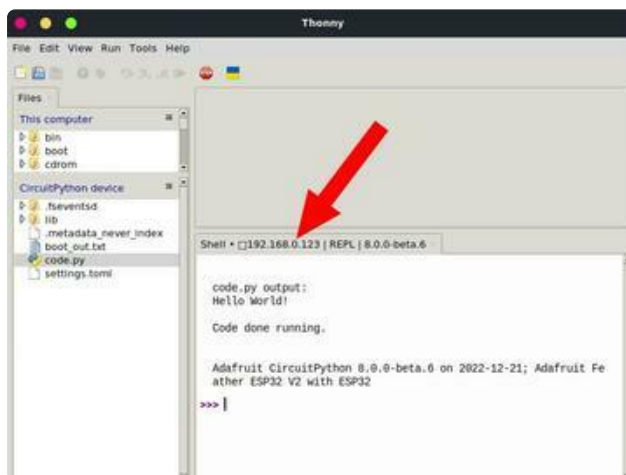For example, here's what a terminal window running screen to connect to the board looks like:

Via REPL

The IP address can be obtained fairly easily via the REPL with a few commands:

```
>>> import wifi
>>> wifi.radio.ipv4_address
192.168.0.121
>>>
```
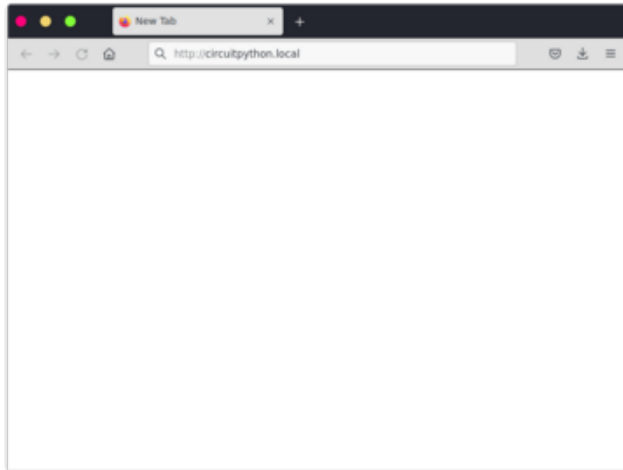
Serial output in Thonny

In Thonny, look for the IP address in the banner text of the serial output in the Shell window.

Note the **settings.toml** file has been added to the CircuitPython device.
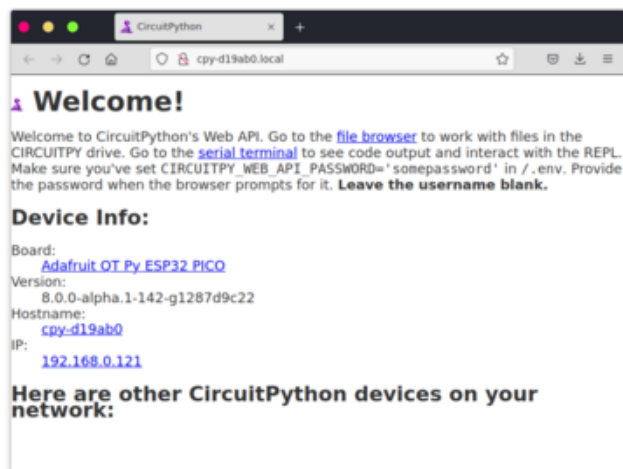
Using Web Workflow

Welcome! Page

This is the initial page seen when first connecting.

Launch a web browser and navigate to `circuitpython.local` **OR** the numeric IP address like `192.168.0.121`

The browser should find the board and show the **Welcome!** screen.

Password Entry

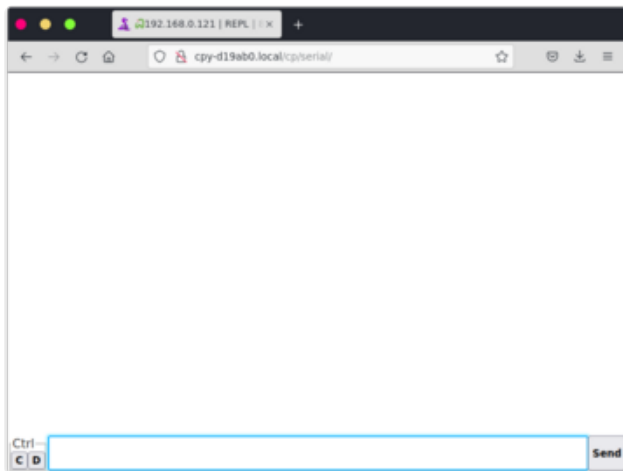When first accessing things from the **Welcome!** page, a password must be entered.

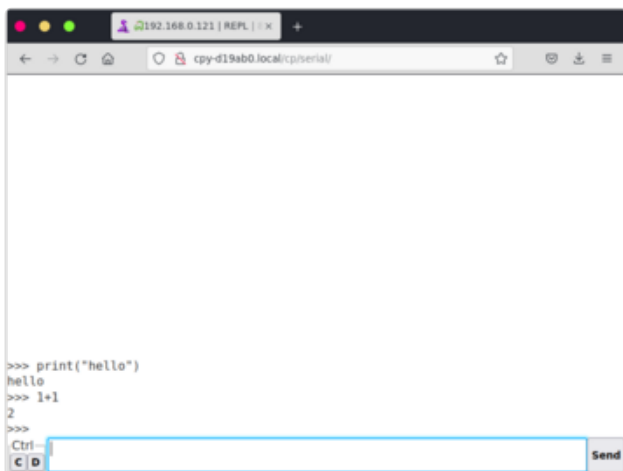Enter the password that was setup for `CIRCUITPY_WEB_API_PASSWORD` in the **settings.toml** file.

Leave Username blank.

Serial Terminal

From the Welcome! page, click the **serial terminal** link to access the serial output as well as REPL for entering commands.
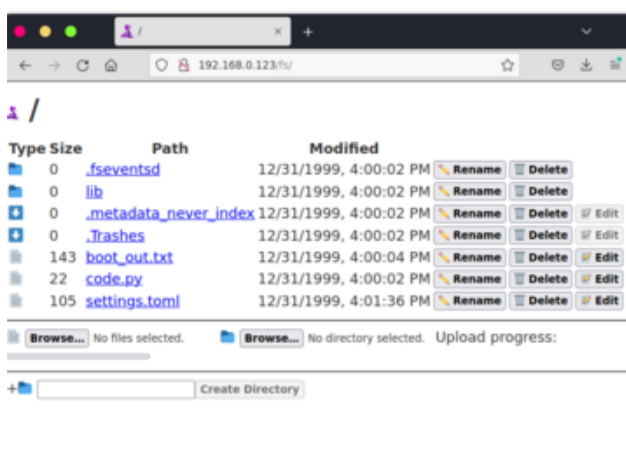
The serial window initially looks blank.

Commands can be entered in the input field at the bottom.

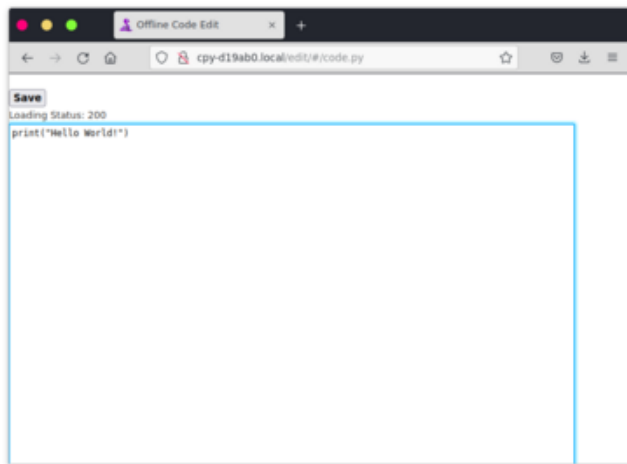The results will be shown above and scroll up.

File Browser

From the Welcome! page, click the **file browser** link on the Welcome! page to access files and folders.
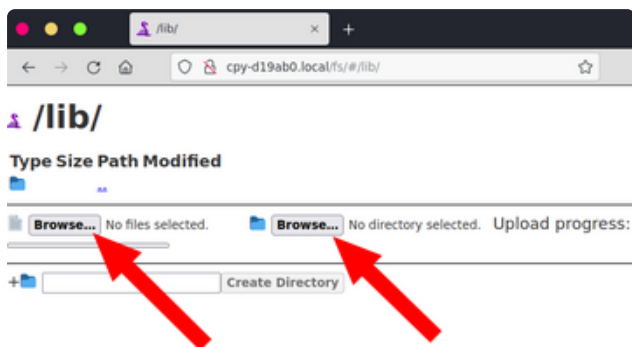

The files and folders should show up.

There are buttons for various actions.

For example, click the **Edit** button for **code.py** bring up a simple editor to allow changing the contents.

Library files and folders from the [Bundle (https://adafru.it/ENC)](https://adafru.it/ENC) can be uploaded to the **/lib** folder using the **Browse...** buttons.

There's a separate button for files and folders (directories).

NeoPixel Example

Here's a simple example showing how to blink the on board NeoPixel of either the Feather ESP32 V2 or the QT PY ESP32 Pico. It covers addings a library file, **neopixel.mpy**, as well as editting **code.py** directly in the browser.

Adding a Library File

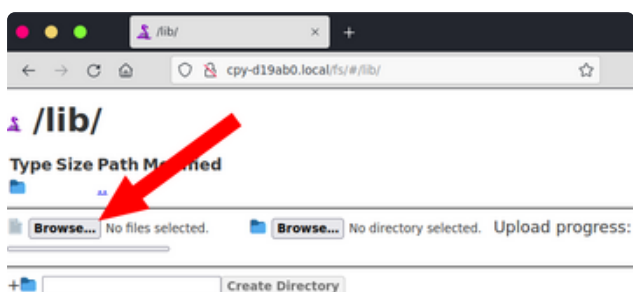Follow these steps to add **neopixel.mpy** to the **/lib** folder.

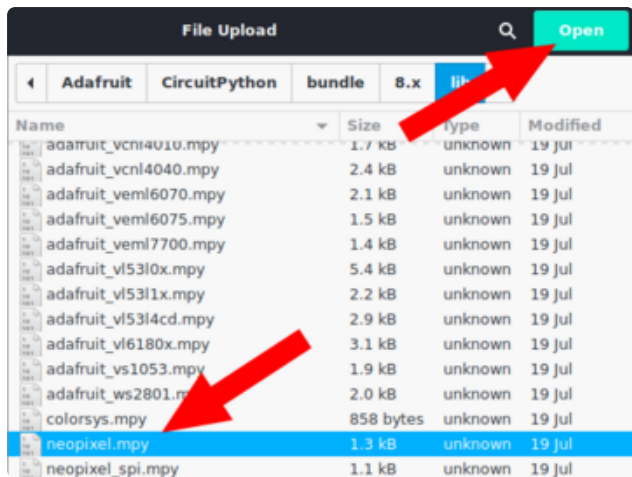From the main Welcome! screen, click on the **file browser** link.
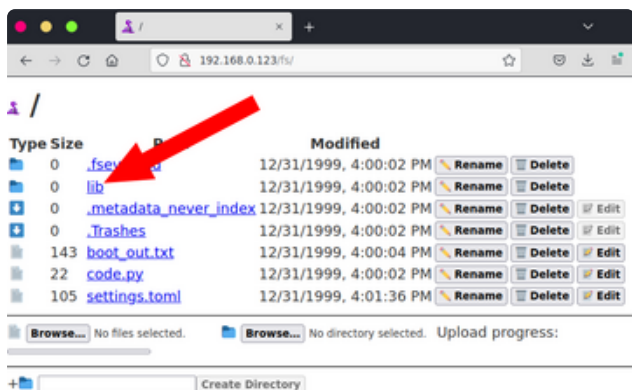
Now click the **lib** folder link to navigate into the /**lib** folder.

We need to navigate first, since uploaded files will go into the current directory location.
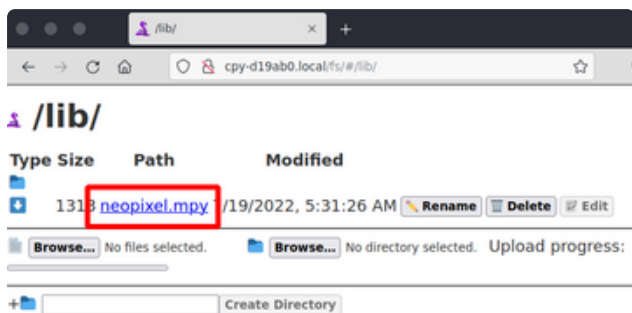

Now click the first **Browse...** button (for files) to bring up a file explorer window.

Navigate to where you've downloaded the [CircuitPython Library Bundle](https://adafru.it/ENC) (https://adafru.it/ENC) and select the **neopixel.mpy** file.
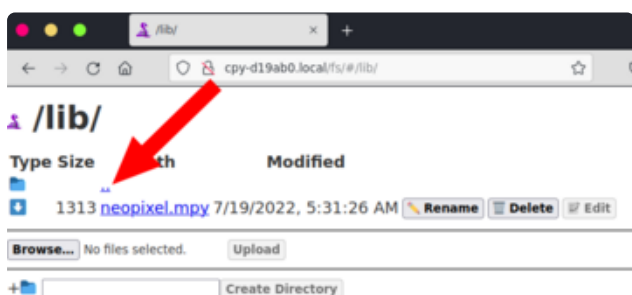
Then click **OK**, or **Open**, or whatever your file window shows.
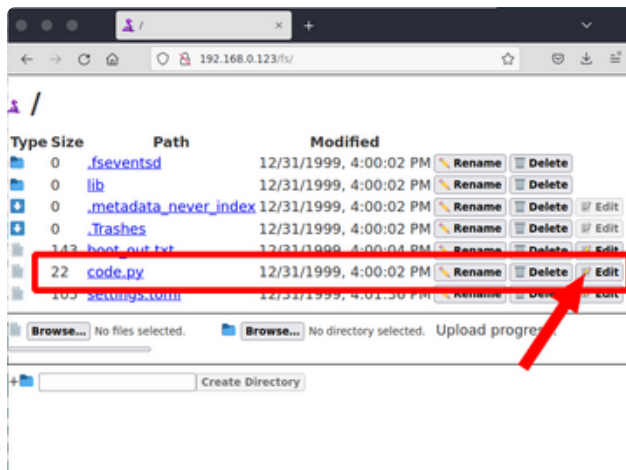

After uploading, the **neopixel.mpy** file will be shown in the **/lib** folder.

Edit code.py

Follow these steps to write a simple NeoPixel blink example and save it as **code.py** so it will run automatically.

From the previous step, click the two little **..** to go back up one level in the folders.
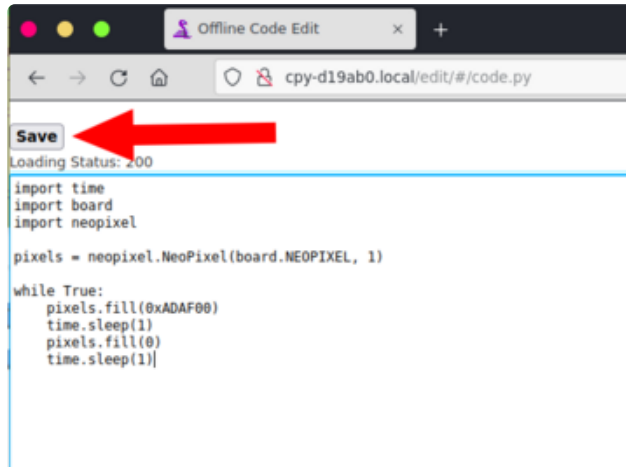
Now click the **Edit** button on the **code.py** line.

Here's the simple example code used:

```python
import time
import board
import neopixel

pixels = neopixel.NeoPixel(board.NEOPIXEL, 1)

while True:
    pixels.fill(0xADAF00)
    time.sleep(1)
    pixels.fill(0)
    time.sleep(1)
```
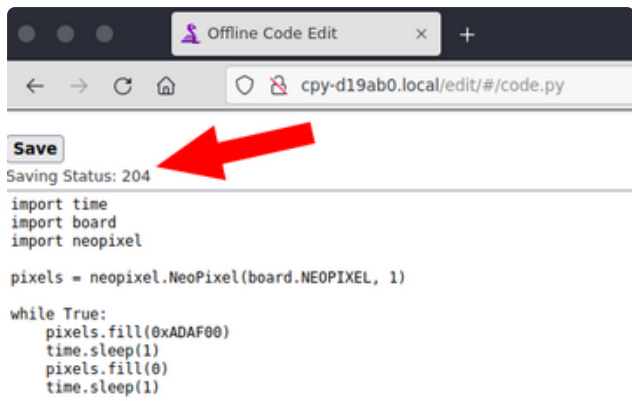

Replace the contents of **code.py** with the code listing shown (can copy from above).

Make sure it matches what is shown - no indent errors, etc. Then click the **Save** button.

```
Save
Saving Status: 204
import time
import board
import neopixel

pixels = neopixel.NeoPixel(board.NEOPIXEL, 1)

while True:
    pixels.fill(0xADAF00)
    time.sleep(1)
    pixels.fill(0)
    time.sleep(1)
```

The page will remain, but the status code should change.

At this point the code should be running and the onboard NeoPixel should be blinking.

There may be a small delay after hitting the Save button before the could actually runs.