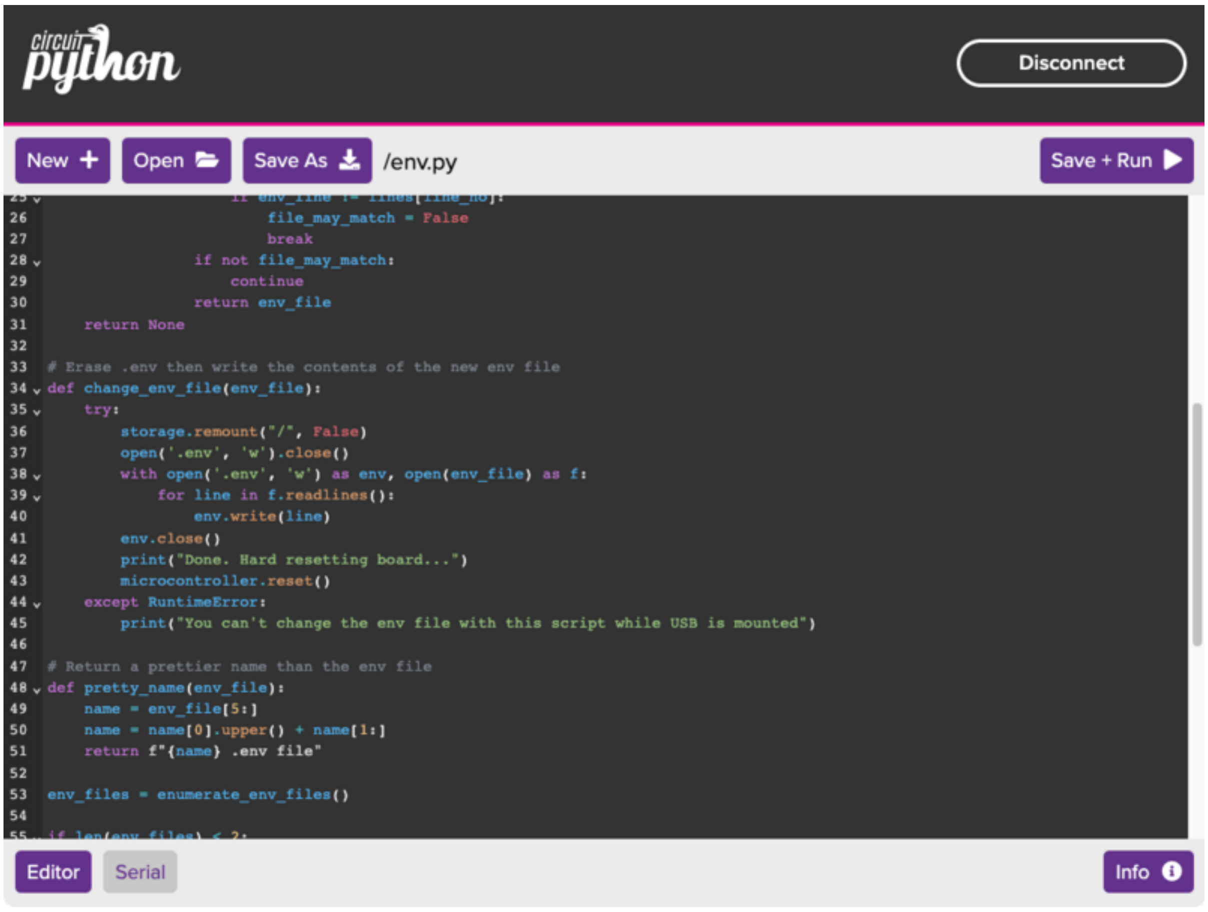
CircuitPython Web Workflow Code Editor Quick Start

Created by Melissa LeBlanc-Williams



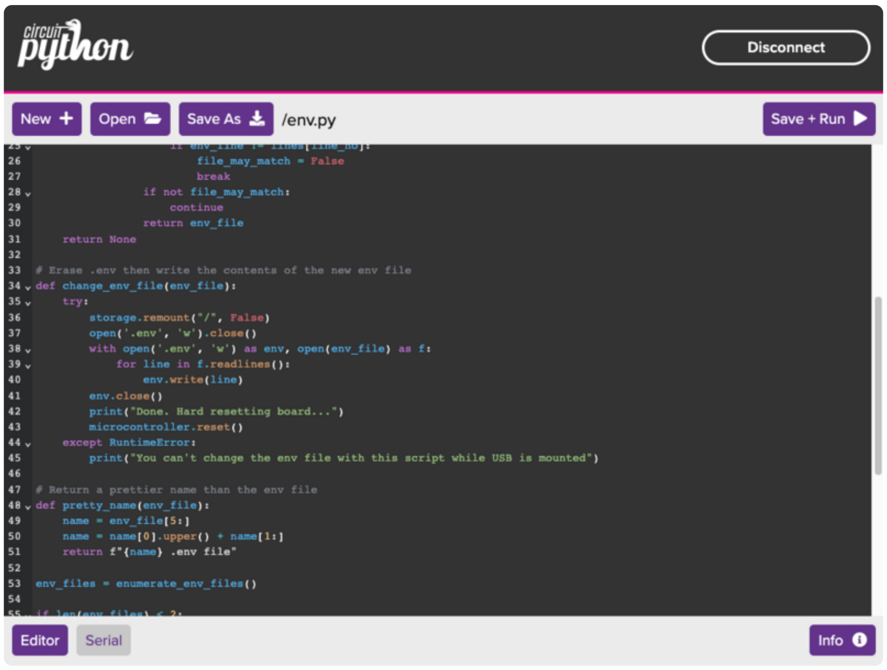
```
                if env_line != lines[line_no]:
                    file_may_match = False
                    break
            if not file_may_match:
                continue
            return env_file
    return None

# Erase .env then write the contents of the new env file
def change_env_file(env_file):
    try:
        storage.remount("/", False)
        open('.env', 'w').close()
        with open('.env', 'w') as env, open(env_file) as f:
            for line in f.readlines():
                env.write(line)
        env.close()
        print("Done. Hard resetting board...")
        microcontroller.reset()
    except RuntimeError:
        print("You can't change the env file with this script while USB is mounted")

# Return a prettier name than the env file
def pretty_name(env_file):
    name = env_file[5:]
    name = name[0].upper() + name[1:]
    return f"{name} .env file"

env_files = enumerate_env_files()

if len(env_files) < 2:
```

https://learn.adafruit.com/getting-started-with-web-workflow-using-the-code-editor

Last updated on 2024-07-03 12:02:26 PM EDT

Table of Contents

Overview



```
                    if env_line != lines[line_no]:
                        file_may_match = False
                        break
                if not file_may_match:
                    continue
                return env_file
    return None

# Erase .env then write the contents of the new env file
def change_env_file(env_file):
    try:
        storage.remount("/", False)
        open('.env', 'w').close()
        with open('.env', 'w') as env, open(env_file) as f:
            for line in f.readlines():
                env.write(line)
        env.close()
        print("Done. Hard resetting board...")
        microcontroller.reset()
    except RuntimeError:
        print("You can't change the env file with this script while USB is mounted")

# Return a prettier name than the env file
def pretty_name(env_file):
    name = env_file[5:]
    name = name[0].upper() + name[1:]
    return f"{name} .env file"

env_files = enumerate_env_files()

if len(env_files) < 2:
```

The CircuitPython Code Editor, available at code.circuitpython.org (https://adafru.it/10QF), provides a fuller and more enriching experience when editing files on your ESP32-based device running the latest version of CircuitPython.

The editor allows you to edit files using web Bluetooth, USB, and Web Workflow over WiFi.

To use Web Workflow with the CircuitPython Code Editor, a connection to the internet is required as well as access to the same Local Area Network that your device is connected to.

While the editor is continuing to be developed and new features are still being added, it is still very usable at this point. No major revisions are expected to the overall usage anytime soon and this guide is intended will help get you started with everything at this point. One of the major changes that was added recently is that the flow now starts off allowing you to work in a disconnected state until you are ready to access files on the device.

The USB Workflow and Bluetooth Workflow will not be covered in this guide.

Device Setup

Make sure you are running version **8.0.0** or later of CircuitPython. You can download it from the downloads page on circuitpython.org (https://adafru.it/Em8).

If you have previously installed it and you're not sure which version of CircuitPython your board is currently running, you can check the **boot_out.txt** file in the root folder of your device. Assuming your mass storage has not been disabled, this should be in the **CIRCUITPY** drive.

Once you have that downloaded, you will want to check out the appropriate learn guide for your specific board for installation instructions. If you are running an ESP32 board, the setup process is a little different and you will want to check out the CircuitPython on ESP32 Quick Start (https://adafru.it/10Ra) guide. If you would like a more general overview of installing CircuitPython (https://adafru.it/Amd), you can check out the Welcome to CircuitPython (https://adafru.it/cpy-welcome) guide.

Web Workflow is not available on AirLift boards such as the Adafruit Metro M4 AirLift Lite or the MatrixPortal M4. These boards have a WiFi co-processor that is controlled completely via CircuitPython code. The boards do not have native WiFi support in the firmware. If you cannot `import wifi` on the board, it will not support the Web Workflow.

Creating a **settings.toml** File

Once you have the correct version of CircuitPython installed and running, the next step is to create an **settings.toml** file in the root of the **CIRCUITPY** drive. Then configure it by adding the following contents and changing the values as appropriate:

For boards with native USB (like ESP32-S2, or ESP32-S3), you can do this by creating a file on the disk drive that appears when plugging in the board. For boards without native USB (like ESP32-C3 or ESP32) you will need to use the REPL. See https://learn.adafruit.com/circuitpython-with-esp32-quick-start/ for more info

```
# To auto-connect to Wi-Fi
CIRCUITPY_WIFI_SSID="yourwifissid"
CIRCUITPY_WIFI_PASSWORD="yourpassword"

# To enable modifying files from the web. Change this too!
# Leave the User field blank when you type the password into the browser.
CIRCUITPY_WEB_API_PASSWORD="passw0rd"
```

```
CIRCUITPY_WEB_API_PORT=80
```

A summary of the values is as follows:

- **CIRCUITPY_WIFI_SSID**: This is the name of the WiFi network that you usually see advertised in a list of available WiFi networks on your computer. It should match exactly.
- **CIRCUITPY_WIFI_PASSWORD**: This is the password for your WiFi network. It should also match exactly.
- **CIRCUITPY_WEB_API_PASSWORD**: This is the password that is used to protect access to the files on your device when connecting with Web Workflow. This is what you will type in when prompted for a password. You will want to change it to something that you'll remember.
- **CIRCUITPY_WEB_API_PORT**: This is the TCP port that the built-in Web Workflow web server on the device will use. At this point in time, the Code Editor hasn't been thoroughly tested with ports other than the default, so leaving it at **port 80 is recommended**.

If you would like a more technical detail of the underlying aspects of Web Workflow, be sure to check out the official CircuitPython documentation (https://adafru.it/10Bp).

After entering in your settings.toml, a "hard" reset is required: power cycle or press the reset button, a soft reset is not enough!

Disabling USB Mass Storage

One of the more challenging aspects of using web workflow is that in order to avoid corrupting files, the USB mass storage device needs to be disabled **on native USB chipsets like ESP32-S2/ESP32-S3**, otherwise the files will only be available in read-only mode over the WiFi link.

The easiest way to do this is to simply eject the **CIRCUITPY** drive in your OS. However, this will only be temporary until the next time you reset the device or plug it back in. If you'd like it to stay disabled, you can create a **boot.py** file in the root folder of the **CIRCUITPY** drive with the following contents:

```
import storage

storage.disable_usb_drive()
```

Once you have saved the file, go ahead and perform a hard reset on the board by pressing the reset button or by temporarily disconnecting it from power. After it starts back up, the **CIRCUITPY** drive should no longer appear. If you'd like to learn more about editing **boot.py** to customize the device, check out the Customizing USB Devices in CircuitPython (https://adafru.it/SC9) guide.

Re-enabling USB Mass Storage

If you find you want to access the mass storage device after you have disabled it, there are a couple of ways you can do it. The first way to temporarily enable it is to boot it into safe mode. To do this, just reset the device and while it is booting, pay attention to the status NeoPixel. When it starts flashing yellow, press the boot button.

Alternatively, if you'd like to permanently re-enable it, you can run the following code from a serial terminal to remove the **boot.py** file:

```
import storage
import os
storage.remount("/", False)
os.remove("/boot.py")
```

After hard resetting your device, the **CIRCUITPY** drive should reappear, though re-enabling it will cause web workflow to be in read-only mode again.

Dealing with Multiple WiFi Networks

If you need to switch between multiple WiFi networks and would like to do so without re-enabling the Mass Storage Device and editing the **settings.toml** file, you can use the following script to automatically switch between different **.toml** files.

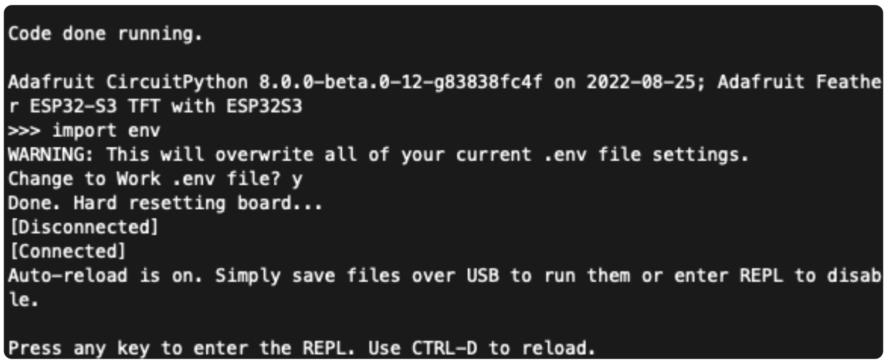
```
# SPDX-FileCopyrightText: 2022 Melissa LeBlanc-Williams for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import os
import storage
import microcontroller

SETTINGS_FOLDER = "/"

# Get all files in the format of xxxxxxxxxx.toml except settings.toml
def enumerate_toml_files():
    found_files = []
    all_files = os.listdir(SETTINGS_FOLDER)
    for current_file in all_files:
        if (
            not current_file.startswith("._")
            and current_file.endswith(".toml")
            and current_file != "settings.toml"
        ):
            found_files.append(SETTINGS_FOLDER + current_file)
    return found_files


# Compare settings.toml to enumerated toml files
def get_current_toml_file(enumerated_files):
    with open("settings.toml") as settings:
        settings_lines = settings.readlines()
        for toml_file in enumerated_files:
            with open(toml_file) as f:
                lines = f.readlines()
                if len(settings_lines) != len(lines):
                    continue
                file_may_match = True
                for line_no, settings_line in enumerate(settings_lines):
                    if settings_line != lines[line_no]:
                        file_may_match = False
                        break
                if not file_may_match:
                    continue
                return toml_file
    return None


# Erase settings.toml then write the contents of the new settings.toml file
def change_toml_file(toml_file):
    try:
        storage.remount("/", readonly=False)
        with open("settings.toml", "w") as settings:
            settings.write("")
        with open("settings.toml", "w") as settings, open(toml_file) as f:
            for line in f.readlines():
                settings.write(line)
        print("Done. Hard resetting board...")
        microcontroller.reset()
    except RuntimeError:
        print("You can't change the env file with this script while USB is mounted")


# Return a prettier name than the toml filename
def pretty_name(toml_file):
    name = toml_file.rsplit("/", 1)[1]
    name = name[:-5]
    name = name[0].upper() + name[1:]
    return f"{name} toml file"


toml_files = enumerate_toml_files()

if len(toml_files) < 2:
    print("You need to have at least 2 .toml files to change")

result = get_current_toml_file(toml_files)
if result:
    toml_files.remove(result)
print("WARNING: This will overwrite all of your current settings.toml file
settings.")
if len(toml_files) == 1:
    answer = input(f"Change to {pretty_name(toml_files[0])}? ")
    answer = answer.lower()
    if answer in ("y", "yes"):
        change_toml_file(toml_files[0])
else:
    valid_selection = False
    while not valid_selection:
        print("Select an option:")
        for index, file in enumerate(toml_files):
            print(f"{index + 1}: {pretty_name(file)}")
        answer = input("Which option would you like? ")
        if answer.isdigit() and 0 < int(answer) <= len(toml_files):
            valid_selection = True
            change_toml_file(toml_files[int(answer) - 1])
        print(f"{answer} was an invalid selection.\n")
```

To use the script, just copy it into the root folder of your **CIRCUITPY** drive. Then you will want to create at least 2 copies of the settings.toml file in the root folder as well. Just name them something like **home.toml**, **school.toml**, or **work.toml**. Make any changes to each file for the particular environment that you will be in.

If you would like to reduce the clutter, you can place the toml files into a separate folder and then update the **SETTINGS_FOLDER** variable to point to the folder making sure the value ends with a "/" character.

To run the script, just connect to the device with a serial terminal, press a key to enter the REPL, and type `import env`. If you only have two files and one of them is identical to the **settings.toml** file, it will just ask you if you want to change to the other file. Just type '**y**' or '**yes**' followed by pressing enter to change it. Anything else will abort. If you have more files or at least 2 of them are different from the current **settings.toml** file, it will prompt you to select the one you want to change to. Just enter the number of next to the file and press enter or you can press **Control+C** to stop the script and abort making changes.

Once you select a file, it will overwrite the contents of the current settings.toml file, so be sure you don't have anything unique in there that you wouldn't want to lose.


```
Code done running.

Adafruit CircuitPython 8.0.0-beta.0-12-g83838fc4f on 2022-08-25; Adafruit Feathe
r ESP32-S3 TFT with ESP32S3
>>> import env
WARNING: This will overwrite all of your current .env file settings.
Change to Work .env file? y
Done. Hard resetting board...
[Disconnected]
[Connected]
Auto-reload is on. Simply save files over USB to run them or enter REPL to disab
le.

Press any key to enter the REPL. Use CTRL-D to reload.
```

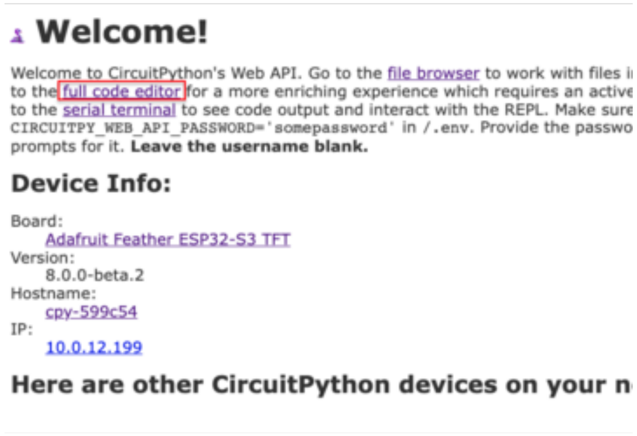
After you select a file, it will automatically hard reset the board and immediately start using the new **settings.toml** file. This makes it very easy to use the board in a variety of settings or even switch between different WiFi networks in the same location.

Connecting

Connecting with Web Workflow

To use the Code Editor, you will need an internet browser such as Google Chrome, Mozilla Firefox, or Microsoft Edge. It's possible that it may work in other browsers as well, but these ones have been more thoroughly tested.

While there are several possible entry points for using the editor, the recommended way is to open up your browser and navigate to http://circuitpython.local/code/ (https://adafru.it/10Rb). Note that it is important that you go to **http** and not **https** and the **trailing slash** must be there as well. This should forward you to the first device that it finds and load the editor.



Welcome!

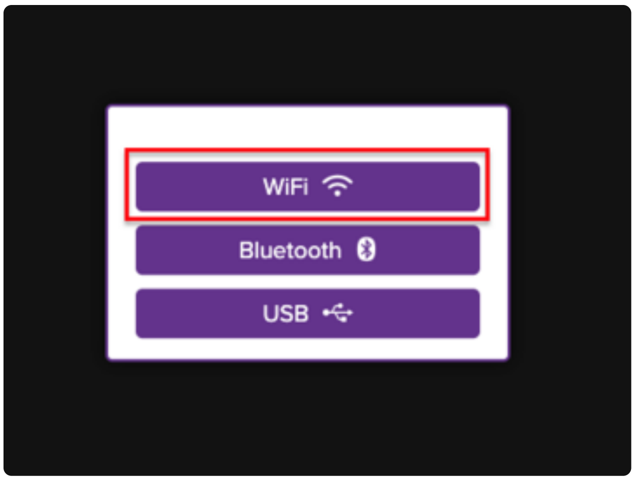
Welcome to CircuitPython's Web API. Go to the file browser to work with files i
to the full code editor for a more enriching experience which requires an active
to the serial terminal to see code output and interact with the REPL. Make sure
CIRCUITPY_WEB_API_PASSWORD='somepassword' in /.env. Provide the passwo
prompts for it. **Leave the username blank.**

Device Info:

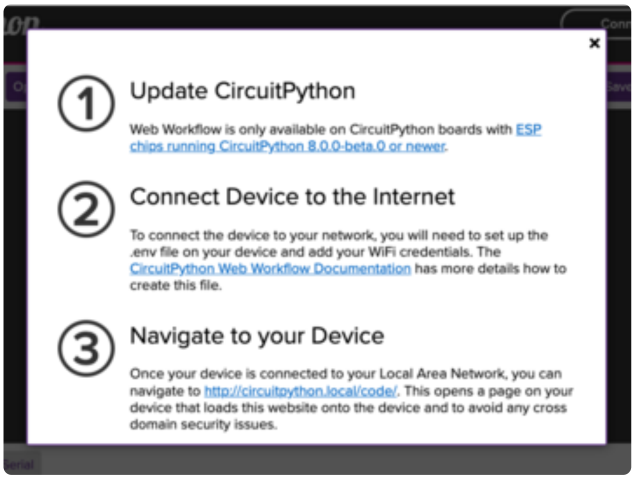
Board:
Adafruit Feather ESP32-S3 TFT
Version:
8.0.0-beta.2
Hostname:
cpy-599c54
IP:
10.0.12.199

Here are other CircuitPython devices on your n

You can also get there by going to http://circuitpython.local/ (https://adafru.it/10Rb) and selecting the **full code editor** link on the Welcome screen.



Another way to get there is by navigating to https://code.circuitpython.org/ (https://adafru.it/10QF) and selecting WiFi on the dialog prompt that comes up.



① Update CircuitPython

Web Workflow is only available on CircuitPython boards with ESP chips running CircuitPython 8.0.0-beta.0 or newer.

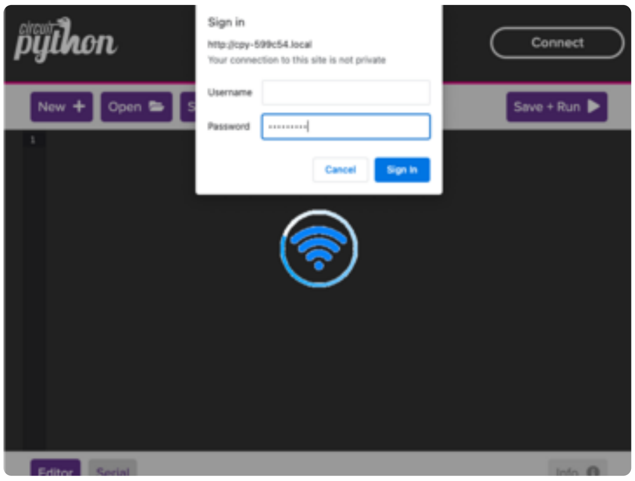
② Connect Device to the Internet

To connect the device to your network, you will need to set up the .env file on your device and add your WiFi credentials. The CircuitPython Web Workflow Documentation has more details how to create this file.

③ Navigate to your Device

Once your device is connected to your Local Area Network, you can navigate to http://circuitpython.local/code/. This opens a page on your device that loads this website onto the device and to avoid any cross domain security issues.

This will display a page of instructions along with a link to http://circuitpython.local/code/ (https://adafru.it/10Rb). Any code you've written in the editor should be automatically transferred by clicking that link.



Sign in

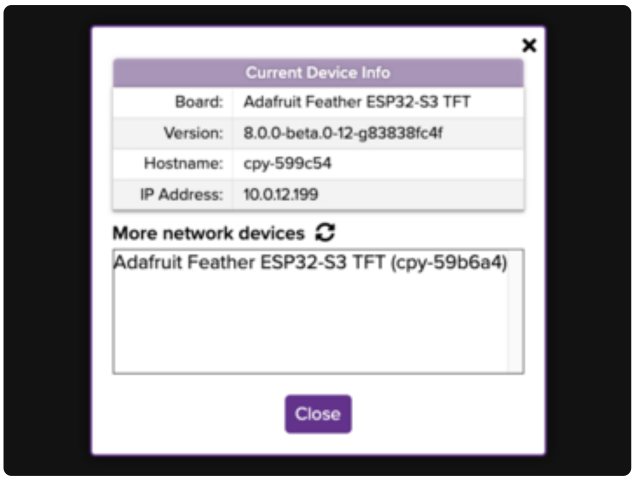
http://cpy-599c54.local

Your connection to this site is not private

Username

Password ••••••••••

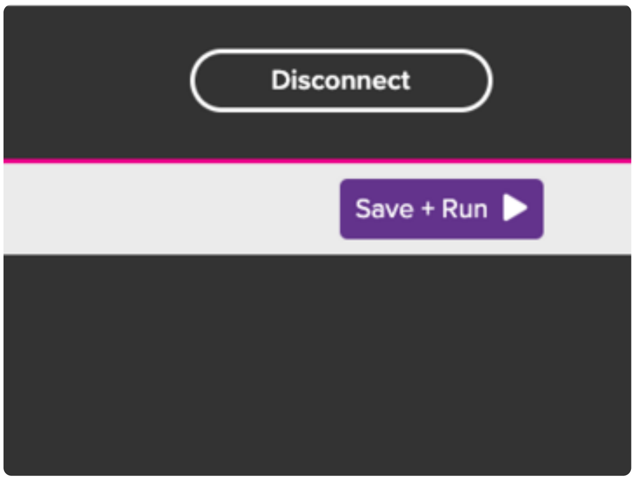
You may be prompted for a password at this point. Leave the **username** blank. For the **password**, type the value that you assigned to the `CIRCUITPY_WEB_API_PASSWORD` setting in you **settings.toml** file.



Current Device Info	
Board:	Adafruit Feather ESP32-S3 TFT
Version:	8.0.0-beta.0-12-g83838fc4f
Hostname:	cpy-599c54
IP Address:	10.0.12.199

More network devices

Adafruit Feather ESP32-S3 TFT (cpy-59b6a4)

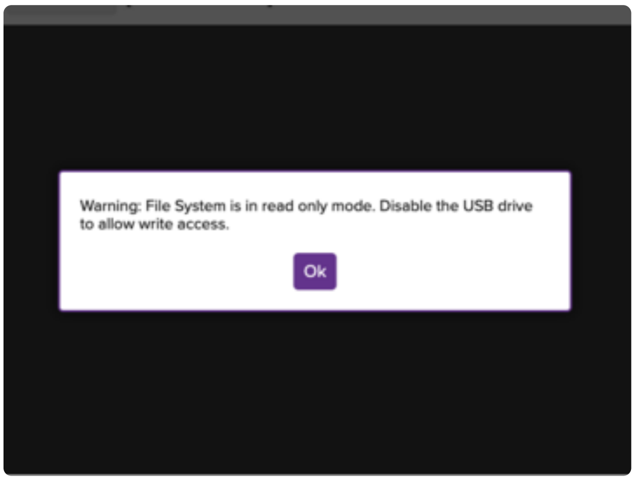


Once you have connected, a current Device Info box should appear which also shows other discovered devices on the network.

The Connect button in the upper Right-hand corner should also change to a Disconnect button.

Read Only Mode

In order to avoid file corruption, CircuitPython requires that there is only one way to write to the device. This means that in order to use Web Workflow to write to the device, the USB Mass Storage must first be disabled. Directions on disabling it are on the Device Setup page.



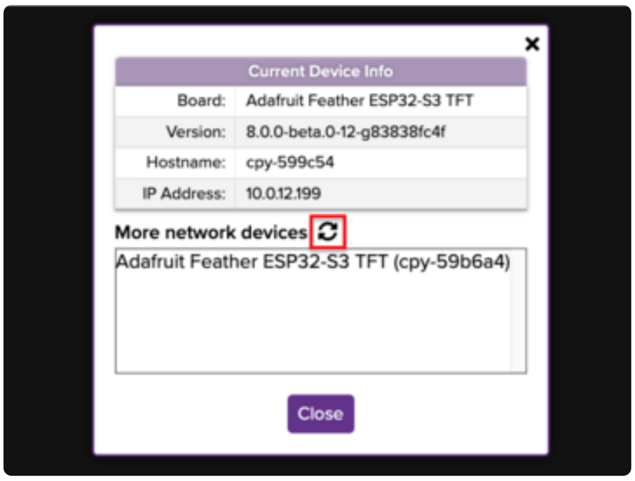
Warning: File System is in read only mode. Disable the USB drive to allow write access.

A notice that you are in Read Only mode will appear if your device's USB Mass Storage is still enabled.

Connecting to Other Devices

If you have multiple devices available on your network, navigating to **circuitpython.local** will use the first device that it finds. If you would like to connect to other devices, you can use the information box that appears when you connect.

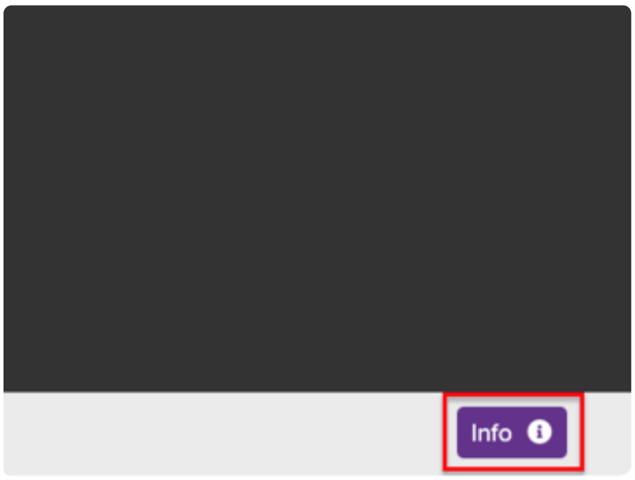
Simply select a device you would like to connect to listed under **More Connected Devices**. If there is a device that you know is online, but is not appearing under this list and there hasn't been any recent network activity, the current device may just believe it is offline. You can reset the device and it should show up in the list.



Current Device Info	
Board:	Adafruit Feather ESP32-S3 TFT
Version:	8.0.0-beta.0-12-g83838fc4f
Hostname:	cpy-599c54
IP Address:	10.0.12.199

More network devices

Adafruit Feather ESP32-S3 TFT (cpy-59b6a4)



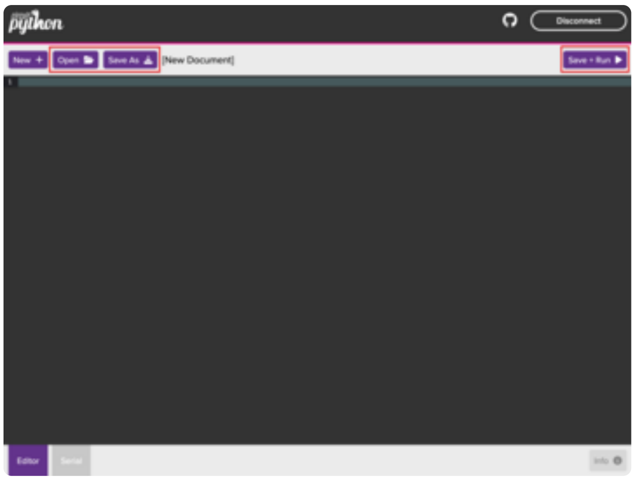
To refresh your devices, you can just click the **Refresh Device List** icon that is above the devices.

If you would like to open the device information and discovery dialog, you can click the **Info** button while you are connected.

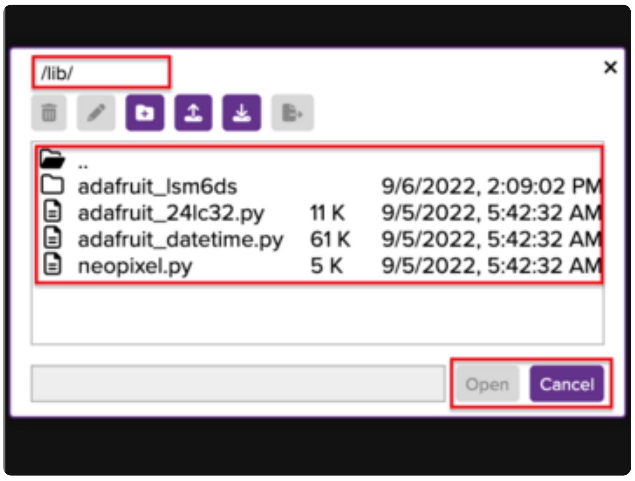
Usage

Opening and Saving Files

Opening and Saving files is designed to be like to most other applications. Just use the buttons along the top of the editor window.



Clicking the **Open** or **Save As** buttons along the top will open the File Dialog. Clicking the **Save + Run** button will save your file and run the code. If your file hasn't been saved yet, this will also bring up the file dialog box.

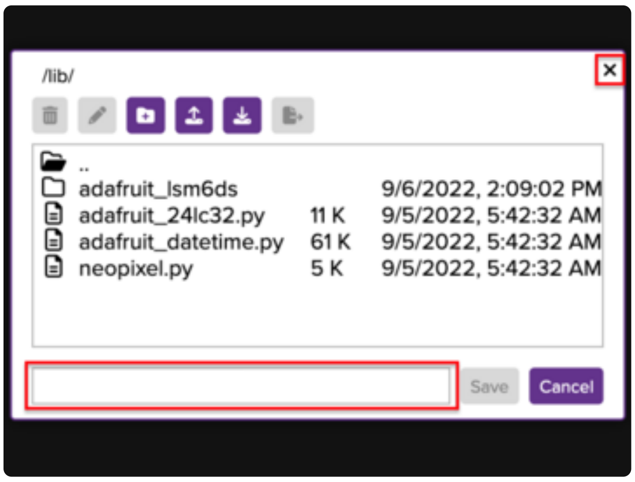


/lib/

..
adafruit_lsm6ds 9/6/2022, 2:09:02 PM
adafruit_24lc32.py 11 K 9/5/2022, 5:42:32 AM
adafruit_datetime.py 61 K 9/5/2022, 5:42:32 AM
neopixel.py 5 K 9/5/2022, 5:42:32 AM

The file dialog that appears is a simplified dialog that displays the current path at the top, allows you to navigate through the file tree to select the file you would like to open, and has buttons on the bottom to open or save the file you would like to use.

Canceling will tell the editor that you do not want to continue with the current operation.



/lib/

..
adafruit_lsm6ds 9/6/2022, 2:09:02 PM
adafruit_24lc32.py 11 K 9/5/2022, 5:42:32 AM
adafruit_datetime.py 61 K 9/5/2022, 5:42:32 AM
neopixel.py 5 K 9/5/2022, 5:42:32 AM

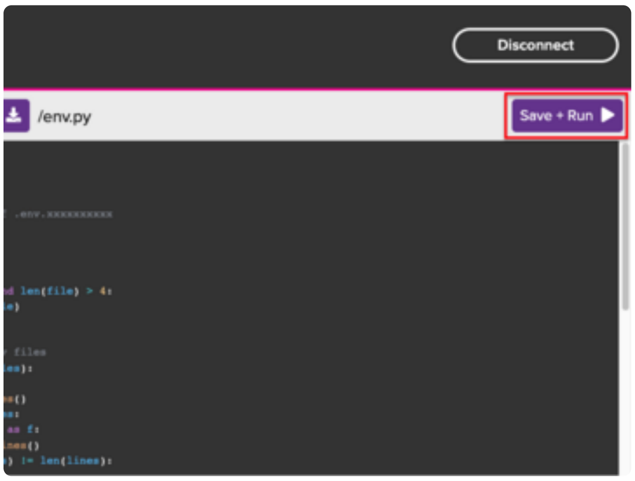
The X at the top performs the same function as the Cancel button as does clicking outside of the dialog.

On the Save As dialog, you can also type in a filename in the field next to the button.

Running Code

As mentioned above, the **Save + Run** button will first save your file, then run the code. The logic to run the code however is currently very simplistic in that it will try a couple of basic strategies to run your code, but doesn't currently do much beyond that.

The way it works is if you are working on **code.py** in the root folder, a soft reset will be performed, which automatically runs **code.py**. If you were working on some code in another file, the editor will attempt to perform an import on this code, which should run it. When you run your code, it will automatically switch over to the serial terminal.



```
f .env.xxxxxxxxxx

d len(file) > 4:
le)

 files
les):

s()
s:
 as f:
ines()
) != len(lines):
```

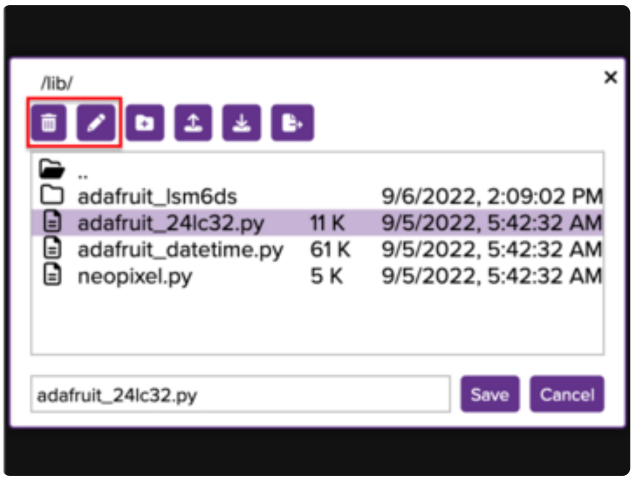
Click the **Save + Run** button to save and run the code current code.

File Dialog Toolbar

The file Dialog toolbar along the top allows you to perform common operations on files and folders regardless of whether you are saving or opening. Clicking the cancel button at the bottom will not undo any operations that were performed with these buttons.

Renaming and Deleting Files and Folders

You can rename or delete both files and folders. An item must be selected first for the buttons to become available.

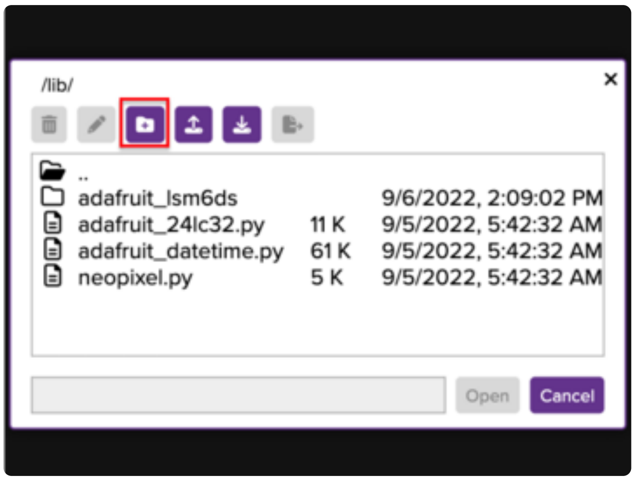


```
..
adafruit_lsm6ds                     9/6/2022, 2:09:02 PM
adafruit_24lc32.py       11 K       9/5/2022, 5:42:32 AM
adafruit_datetime.py     61 K       9/5/2022, 5:42:32 AM
neopixel.py              5 K        9/5/2022, 5:42:32 AM

adafruit_24lc32.py
```

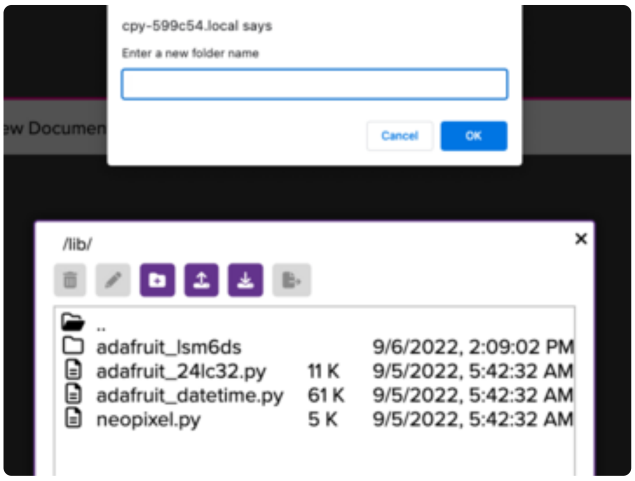
Use the **delete** and **rename** buttons here to perform the corresponding operation on the currently selected file or folder.

Creating New Folders

This feature allows you to create a new folder to store your work inside of.



```
/lib/

..
adafruit_lsm6ds                        9/6/2022, 2:09:02 PM
adafruit_24lc32.py        11 K         9/5/2022, 5:42:32 AM
adafruit_datetime.py      61 K         9/5/2022, 5:42:32 AM
neopixel.py               5 K          9/5/2022, 5:42:32 AM
```



cpy-599c54.local says

Enter a new folder name

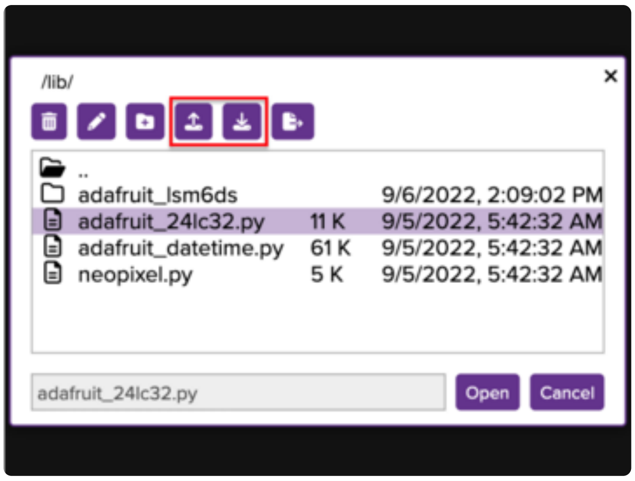
```
/lib/

..
adafruit_lsm6ds                        9/6/2022, 2:09:02 PM
adafruit_24lc32.py        11 K         9/5/2022, 5:42:32 AM
adafruit_datetime.py      61 K         9/5/2022, 5:42:32 AM
neopixel.py               5 K          9/5/2022, 5:42:32 AM
```

Clicking the **new folder** button at the top will prompt you for a folder name. It will inform you of invalid folder names such as the same name as an existing file or folder or a folder that begins with a period.

Uploading and Downloading Files and Folders

This feature allows you to upload or download files as long as they fit in the available space. If you need to add images or sound files for your project, you can use the upload button to add them. If you need to retrieve a file from your device for whatever reason, the download button will give you access to do that.

You can also download folders. When you select a folder and click download, the contents of that folder are automatically zipped into a single file. If nothing is selected when you click the download button, the current folder will be used.



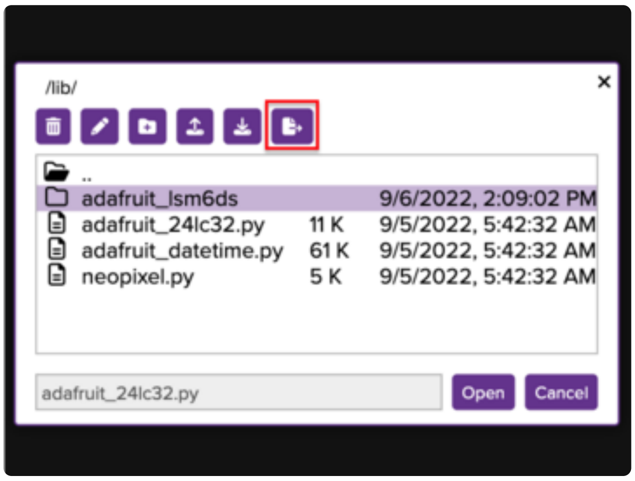
adafruit_lsm6ds		9/6/2022, 2:09:02 PM
adafruit_24lc32.py	11 K	9/5/2022, 5:42:32 AM
adafruit_datetime.py	61 K	9/5/2022, 5:42:32 AM
neopixel.py	5 K	9/5/2022, 5:42:32 AM

adafruit_24lc32.py

Use the **upload** or **download** buttons to easily add files or retrieve them from your board.

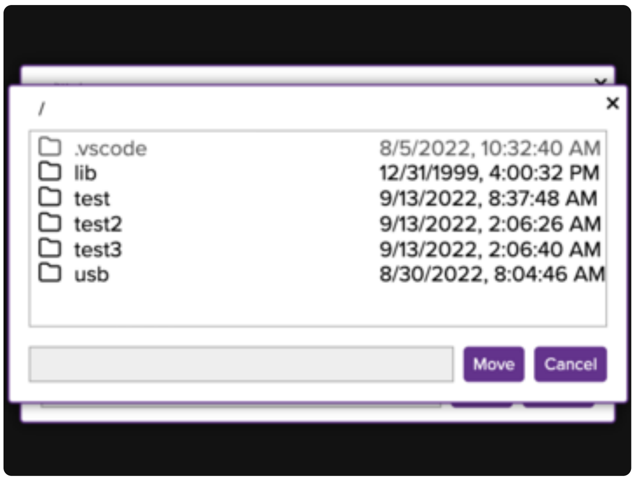
Moving Files and Folders

This feature allows you to move files and folders to a different location on the device. When you click the move button, another prompt will appear on top of the dialog that allows you to navigate to where you would like to move the currently selected item.



adafruit_lsm6ds		9/6/2022, 2:09:02 PM
adafruit_24lc32.py	11 K	9/5/2022, 5:42:32 AM
adafruit_datetime.py	61 K	9/5/2022, 5:42:32 AM
neopixel.py	5 K	9/5/2022, 5:42:32 AM

adafruit_24lc32.py



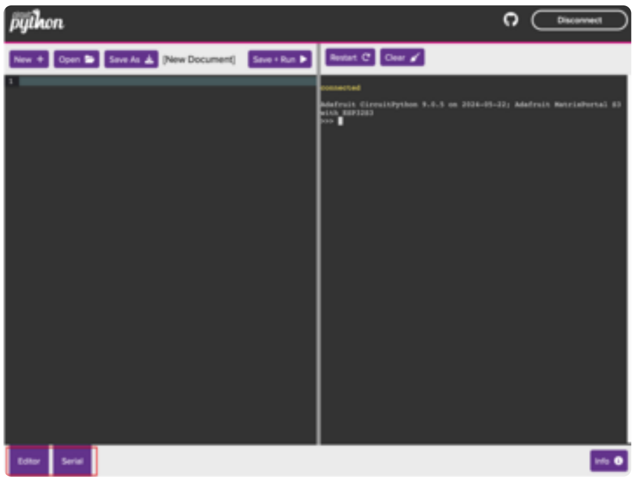
.vscode	8/5/2022, 10:32:40 AM
lib	12/31/1999, 4:00:32 PM
test	9/13/2022, 8:37:48 AM
test2	9/13/2022, 2:06:26 AM
test3	9/13/2022, 2:06:40 AM
usb	8/30/2022, 8:04:46 AM

Use the **move** button to move files or folders to a new location on the device.

The second dialog that appears will show only folders and allow you to navigate to where you would like to move the file.

Using the Serial Terminal

The serial terminal allows you to watch the output of your device as well as type inputs just like you can from a separate application like PuTTY, except there's nothing you need to configure. This allows you to access the REPL or view the output of your currently running code.



```
connected

Adafruit CircuitPython 9.0.5 on 2024-05-22; Adafruit MatrixPortal S3 with ESP32S3
>>> 
```

Use the mode buttons in the bottom left-hand corner to open and close the serial and editor panes.

More Features to Come

The CircuitPython Code Editor is still under development, so expect more features to be added. If you would like to contribute on GitHub (https://adafru.it/10Rc), you can submit any new issues or pull requests for review.