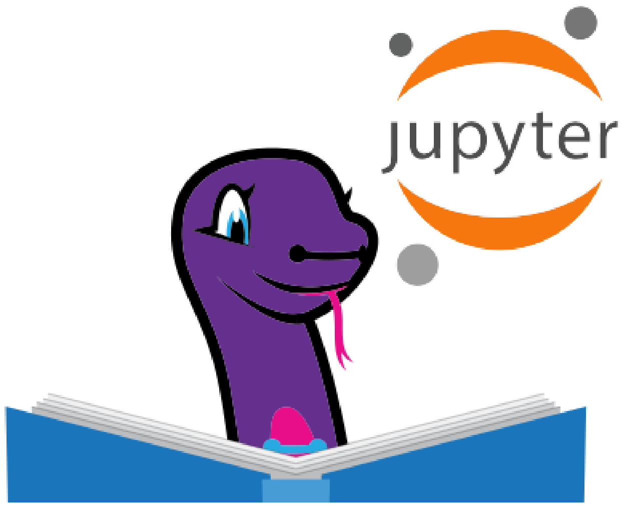
adafruit learning system

CircuitPython with Jupyter Notebooks

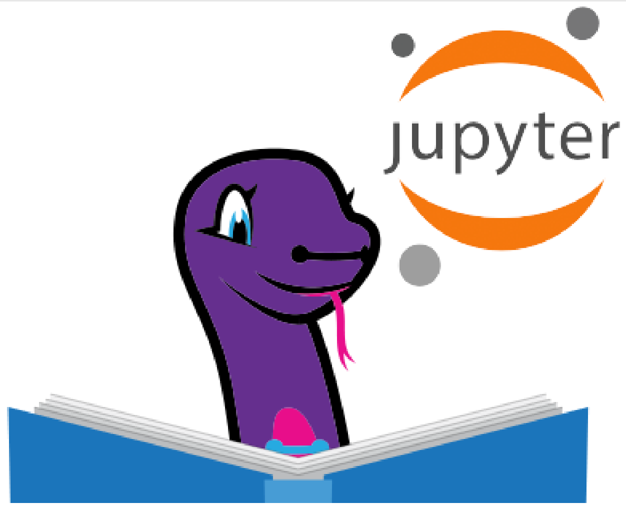
Created by Brent Rubell



Last updated on 2018-08-22 04:08:47 PM UTC

Guide Contents

Overview



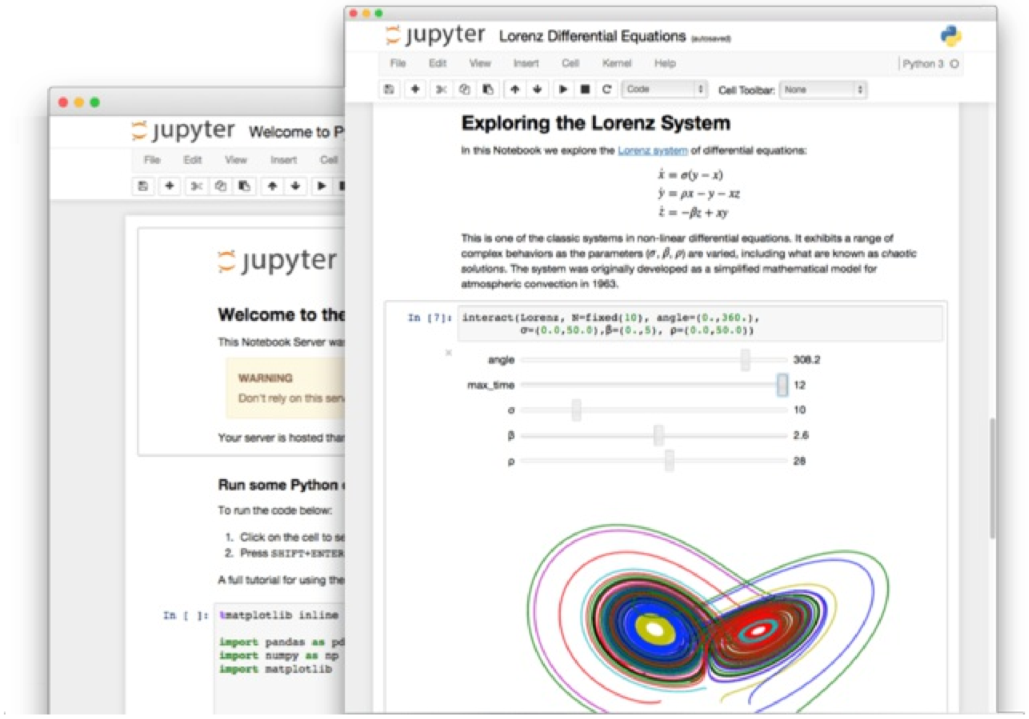
Jupyter lets you create interactive notebooks containing code, text, and rich media that you can share with your friends.

We created a Jupyter Notebook package (called a Kernel) for you to **run CircuitPython code directly from a Jupyter interactive notebook.**

That means your code lives in your web-browser and executes on the CircuitPython hardware through a serial USB link.

We're going to learn how to combine Jupyter with CircuitPython (https://adafru.it/Bid) to create interactive notebooks for your hardware.

What's a Jupyter Notebook?



Exploring the Lorenz System

In this Notebook we explore the Lorenz system of differential equations:

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = \rho x - y - xz$$
$$\dot{z} = -\beta z + xy$$

This is one of the classic systems in non-linear differential equations. It exhibits a range of complex behaviors as the parameters (σ, β, ρ) are varied, including what are known as *chaotic solutions*. The system was originally developed as a simplified mathematical model for atmospheric convection in 1963.

```
In [7]: interact(Lorenz, N=fixed(10), angle=(0.,360.),
                 σ=(0.0,50.0),β=(0.,5), ρ=(0.0,50.0))
```

```
    angle  308.2
 max_time  12
        σ  10
        β  2.6
        ρ  28
```

```
In [ ]: %matplotlib inline

        import pandas as pd
        import numpy as np
        import matplotlib
```

> The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

This is great news for people who are already comfortable with Jupyter, but also for educators, developers and workshop-organizers.

In this learn guide, we're going to walk through installing Jupyter Notebook, installing the Jupyter CircuitPython kernel, and uploading Jupyter Notebook examples.

Installing on Mac and Linux

The installation process for Jupyter and the CircuitPython kernel is *a bit lengthy*, but it's not difficult. If you don't already have Jupyter installed, you'll need to install it before we install the kernel.

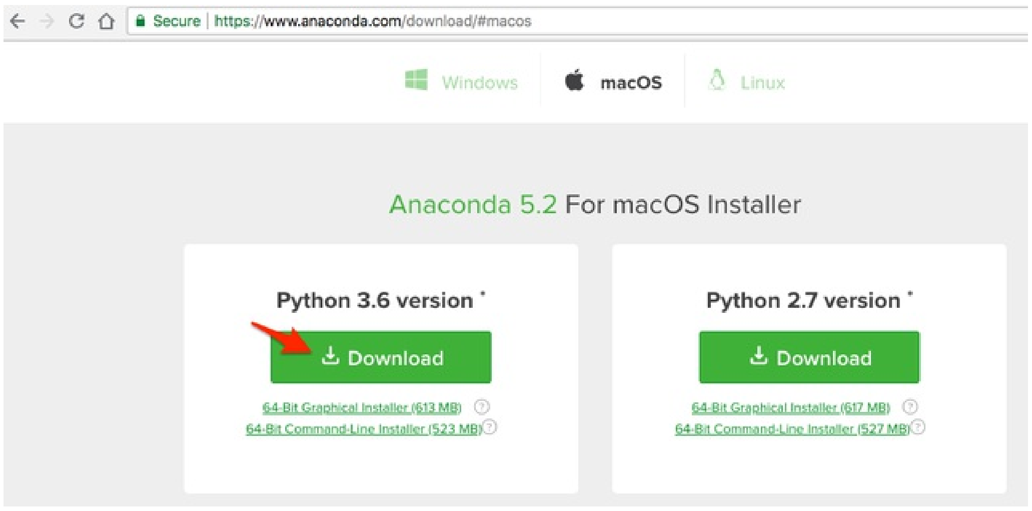
If you have a WIndows machine, please skip to the next page with Windows specific installation instructions.

Installing Jupyter

Install Jupyter with Anaconda

Don't have a Python installation on your computer? If you're new to all this, the Jupyter Project recommends installing Anaconda (https://adafru.it/kse), which installs Python, the Jupyter Notebook, and other commonly used packages for scientific computing and data science.

First, navigate to the Anaconda downloads page (https://adafru.it/kse), select your platform (Windows, Mac, Linux), and **download the installer including Python 3.6+**.


Secure | https://www.anaconda.com/download/#macos
Windows
macOS
Linux
Anaconda 5.2 For macOS Installer
Python 3.6 version *
Download
64-Bit Graphical Installer (613 MB)
64-Bit Command-Line Installer (523 MB)
Python 2.7 version *
Download
64-Bit Graphical Installer (617 MB)
64-Bit Command-Line Installer (527 MB)

Install the version of Anaconda you downloaded by following the instructions in the installation executable.

Install Jupyter with PIP

The CircuitPython Jupyter Kernel requires a Python 3.6+ installation.

If you have a Python installation already on your computer, you may want to use the Python Package Manager (pip (https://adafru.it/Byg)) instead of Anaconda.

If you're not sure which Python version is installed, you can check by running:

```
python3 --version
```

Next, let's ensure we also have the latest version of the Python Package Manager. We can do this by running:

```
pip3 install --upgrade pip
```

Install the Jupyter Notebook:

```
pip3 install jupyter
```

Starting the Notebook Server

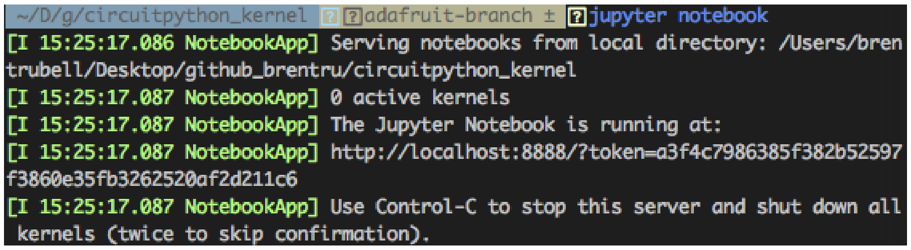
Now that we have Jupyter installed, let's start the notebook server.

On MacOS or Linux, we can use the Terminal.

Let's launch the Jupyter Notebook server by opening either Command Prompt or Terminal and typing:

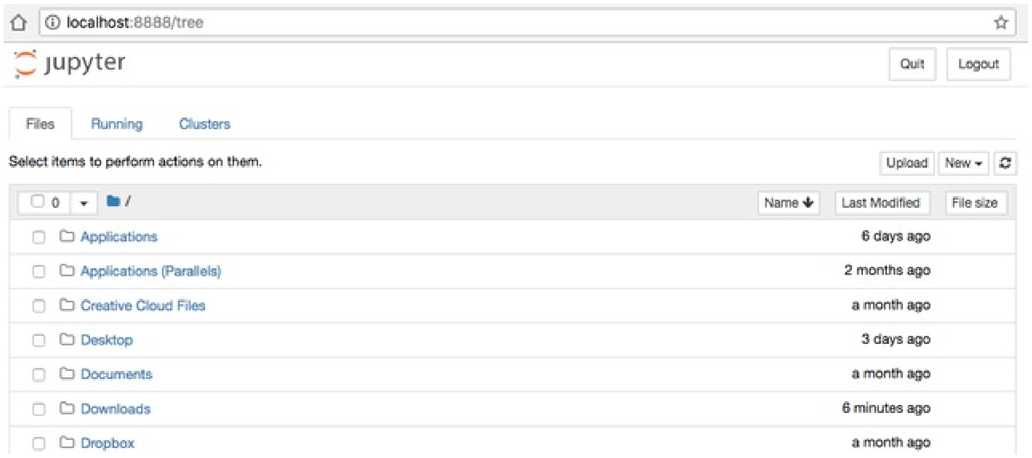
```
jupyter notebook
```

If your installation went well, you'll see information about the notebook server in your command line:



```
 ~/D/g/circuitpython_kernel  ?adafruit-branch ±  jupyter notebook
[I 15:25:17.086 NotebookApp] Serving notebooks from local directory: /Users/bren
trubell/Desktop/github_brentru/circuitpython_kernel
[I 15:25:17.087 NotebookApp] 0 active kernels
[I 15:25:17.087 NotebookApp] The Jupyter Notebook is running at:
[I 15:25:17.087 NotebookApp] http://localhost:8888/?token=a3f4c7986385f382b52597
f3860e35fb3262520af2d211c6
[I 15:25:17.087 NotebookApp] Use Control-C to stop this server and shut down all
 kernels (twice to skip confirmation).
```

Your web browser will automatically open to the **Jupyter Notebook Dashboard.**



Applications	6 days ago
Applications (Parallels)	2 months ago
Creative Cloud Files	a month ago
Desktop	3 days ago
Documents	a month ago
Downloads	6 minutes ago
Dropbox	a month ago

The Notebook Dashboard displays a list of notebooks, files, and folders in the location on your computer from where the notebook server was started.

We're going to spend more time with the Dashboard later in this guide.

Installing the CircuitPython Kernel

Jupyter manages its programming language support through the installation of **kernels**.

By default, the Jupyter Notebook will have the Python programming language installed. We're going to install the CircuitPython Kernel which is a wrapper (https://adafru.it/BLh) which allows CircuitPython's REPL to communicate with Jupyter's code cells.

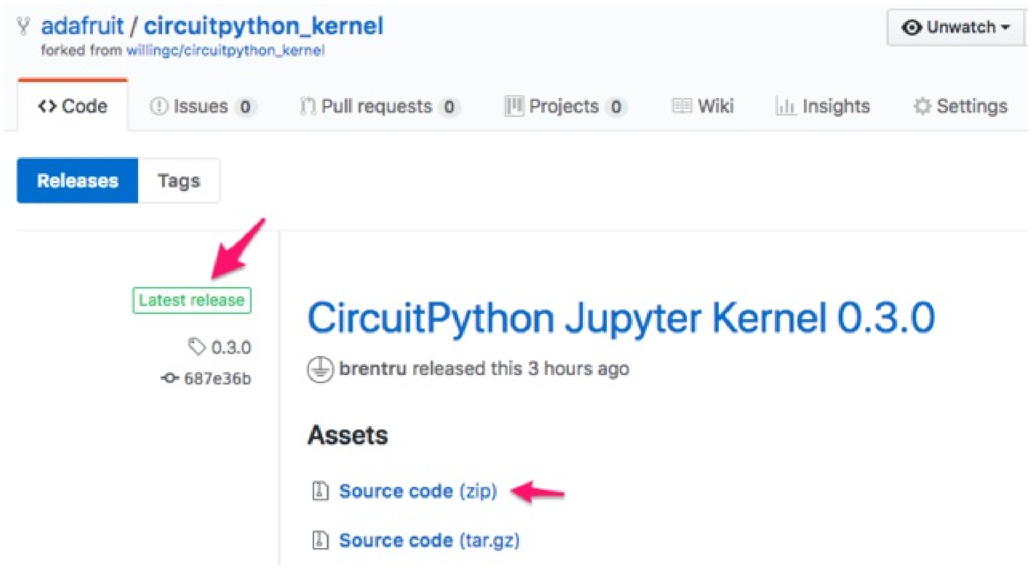
If you have a Git client installed (https://adafru.it/BLi), clone the CircuitPython kernel by running the following in your terminal:

```
git clone https://github.com/adafruit/circuitpython_kernel.git
```

Alternatively, you can download the latest release of the *circuitpython_kernel* from GitHub as a .zip file and unzip it.

https://adafru.it/BLv

https://adafru.it/BLv



In your terminal or command prompt, navigate to the folder directory:

```
cd circuitpython_kernel/
```

Install the Kernel by running:

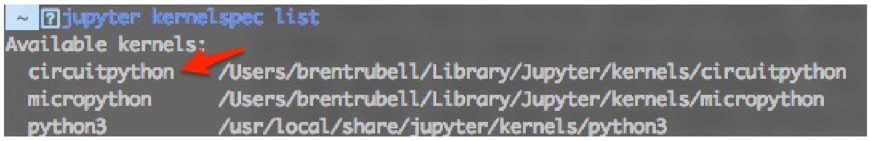
```
python3 setup.py install
```

Finally, let's install the CircuitPython Kernel into Jupyter:

```
python3 -m circuitpython_kernel.install
```

You may need to run this command prefixed by 'sudo' on MacOS or Linux if you see errors during the installation.

The kernel *should* be installed, but let's verify that by running:

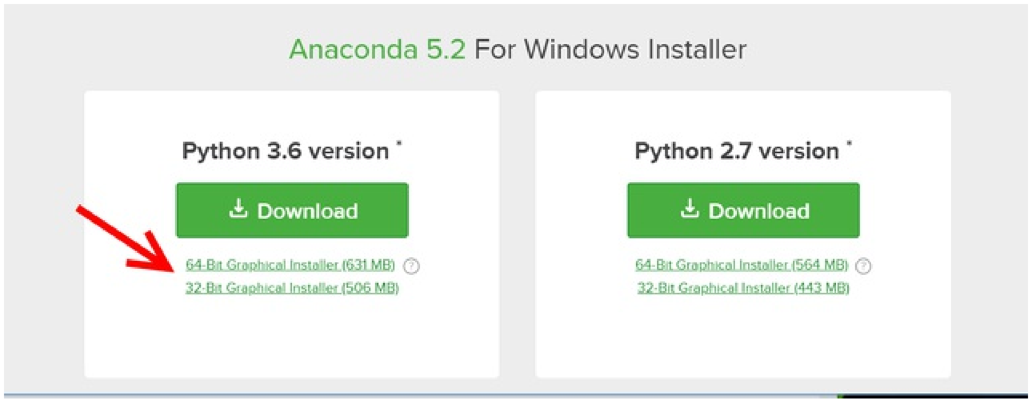


```
 ~ jupyter kernelspec list
Available kernels:
  circuitpython    /Users/brentrubell/Library/Jupyter/kernels/circuitpython
  micropython      /Users/brentrubell/Library/Jupyter/kernels/micropython
  python3          /usr/local/share/jupyter/kernels/python3
```

If **circuitpython** appears as an available kernel, the installation was successful.

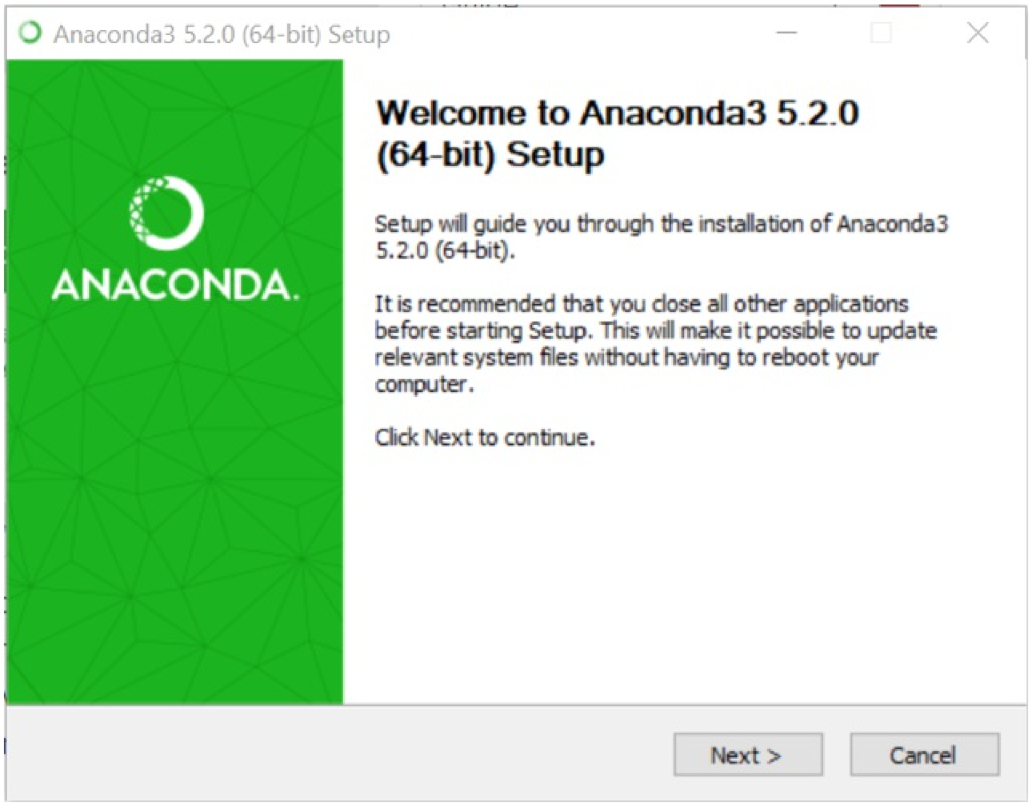
Installing on Windows

Don't have a Python installation on your computer? If you're new to all this, the Jupyter Project recommends installing Anaconda (https://adafru.it/kse), which installs Python, the Jupyter Notebook, and other commonly used packages for scientific computing and data science.


Anaconda 5.2 For Windows Installer
Python 3.6 version
Download
64-Bit Graphical Installer (631 MB)
32-Bit Graphical Installer (506 MB)
Python 2.7 version
Download
64-Bit Graphical Installer (564 MB)
32-Bit Graphical Installer (443 MB)

You will need to know if you are running 32-bit WIndows or 64-bit Windows. To find out if your computer is running a 32-bit or 64-bit version of Windows in Windows 7, Open **System** by clicking the **Start button** , right-clicking **Computer**, and then clicking **Properties**. On Windows 10, type **about** in the search box on your taskbar, and then select **About your PC**. Look under **PC** for **System type** to see if you're running a 32-bit or 64-bit version of Windows.

Select the proper download for your system (32-bit or 64-bit). The file will download, probably into your **Downloads** folder. You can start the installation by clicking the downloaded file icon in the browser or using File Explorer, going to the **Downloads** folder and double clicking the file with a name like **Anaconda3-5.2.0-Windows-x86_64.exe**. You'll see the installation run:



Welcome to Anaconda3 5.2.0 (64-bit) Setup

Setup will guide you through the installation of Anaconda3 5.2.0 (64-bit).

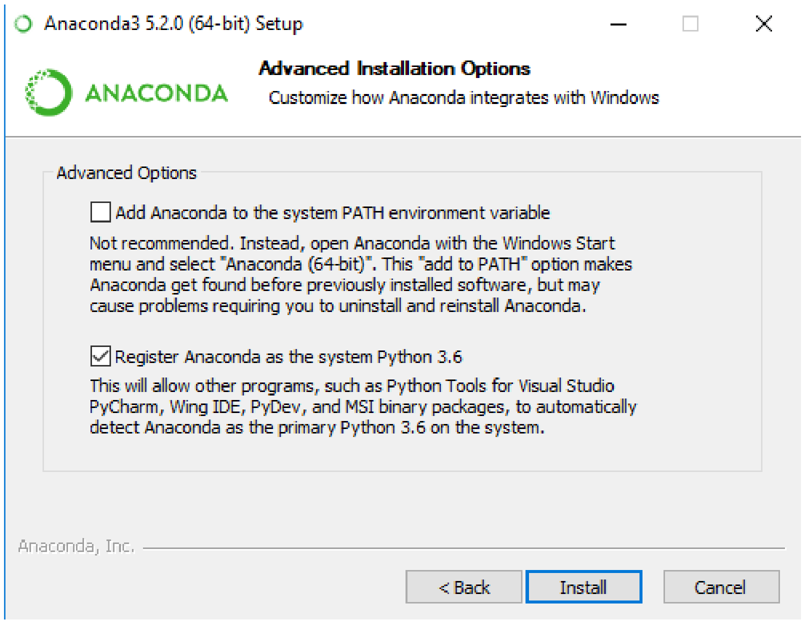
It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

Click **Next** then **All Users** (if you have administrative privileges, otherwise click **Just me**).

The installer will ask where to put the installation. Unless you have a specific place, accept the default which is **C:\ProgramData\Anaconda3** or similar by clicking **Next**.

You'll see the screen below.


Anaconda3 5.2.0 (64-bit) Setup
ANACONDA
Advanced Installation Options
Customize how Anaconda integrates with Windows
Advanced Options
Anaconda, Inc.
< Back
Install
Cancel

☐ Add Anaconda to the system PATH environment variable

Not recommended. Instead, open Anaconda with the Windows Start menu and select "Anaconda (64-bit)". This "add to PATH" option makes Anaconda get found before previously installed software, but may cause problems requiring you to uninstall and reinstall Anaconda.

☑ Register Anaconda as the system Python 3.6

This will allow other programs, such as Python Tools for Visual Studio PyCharm, Wing IDE, PyDev, and MSI binary packages, to automatically detect Anaconda as the primary Python 3.6 on the system.

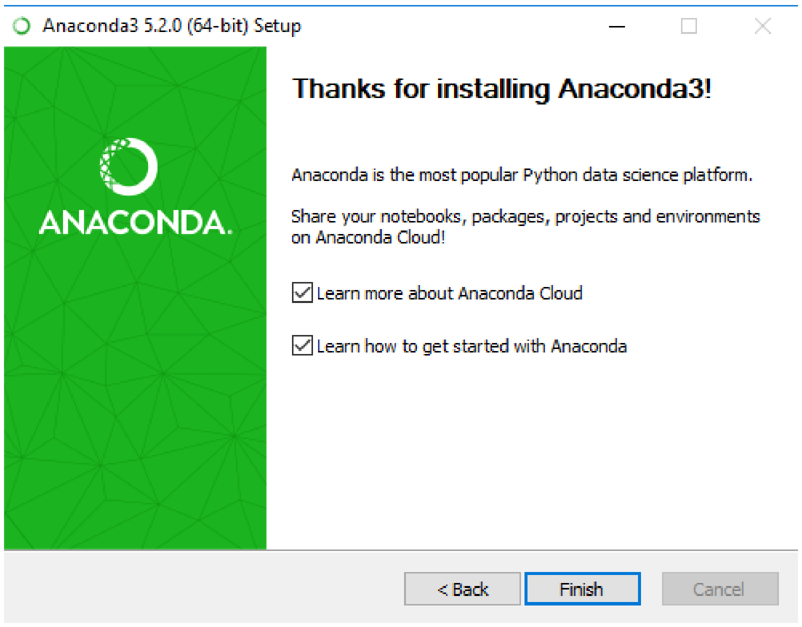
It is recommended you keep the **Advanced Options** the way the installer wants them unless you have been told by an instructor otherwise or have other Python software installed and are knowledgeable about multiple installations. Click **Install**.

Anaconda will now install, it'll take a few minutes depending on your system speed. You may see some windows open and close, that is typical.

When done, the installer will show **Completed**. Click **Next**.

The installer will ask if you want to install the free, open source editor Microsoft VSCode. You will need administrator privileges if you want to install this. Or you can skip. If you are more of a power user, **Install** is good. If you are not the admin of the machine, click **Skip**.

The installer will then show you this screen:


Anaconda3 5.2.0 (64-bit) Setup

Thanks for installing Anaconda3!

Anaconda is the most popular Python data science platform.

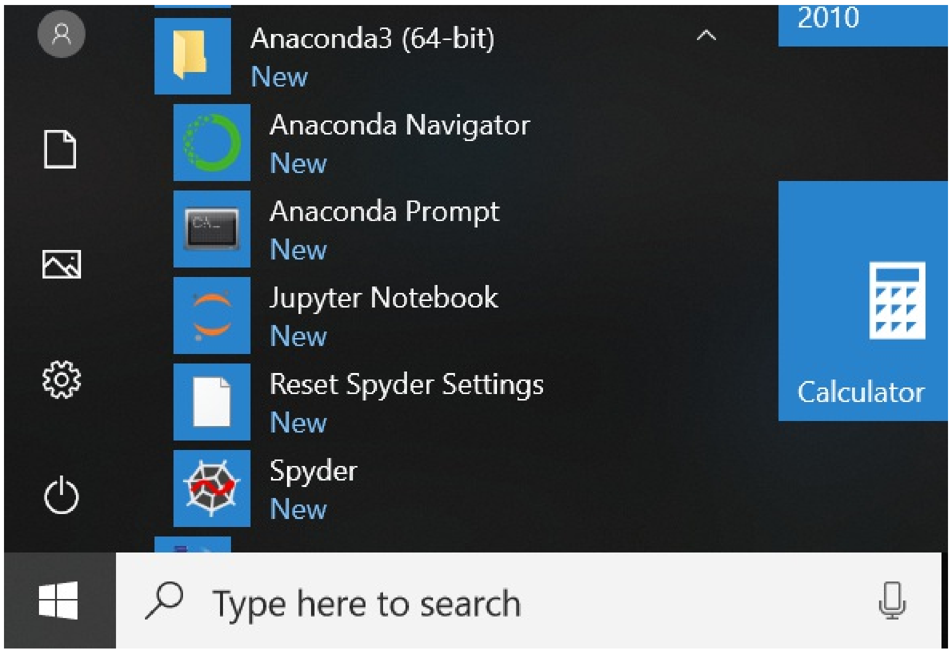
Share your notebooks, packages, projects and environments on Anaconda Cloud!

[✓] Learn more about Anaconda Cloud

[✓] Learn how to get started with Anaconda

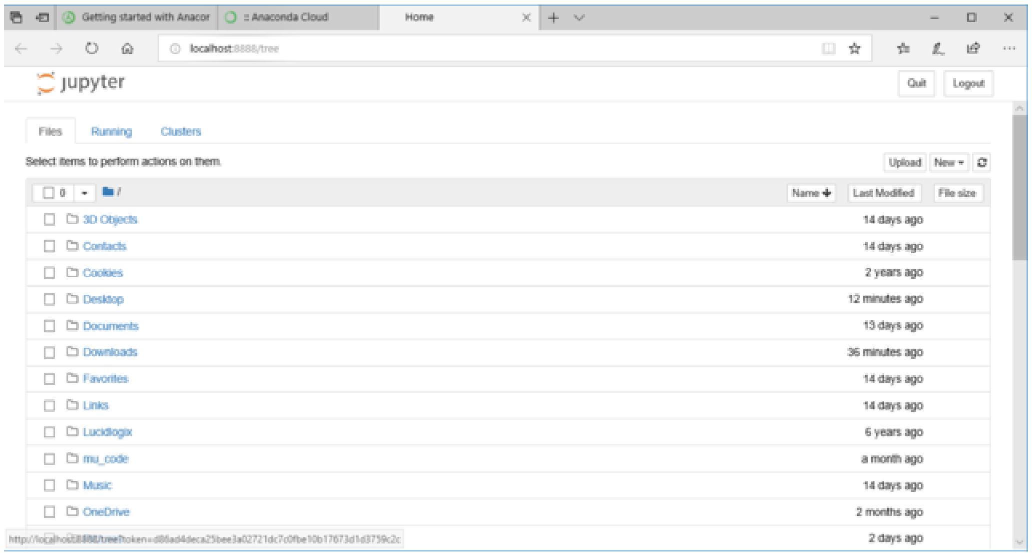
If you want to learn more, keep the check boxes clicked. Click **Finish**.

On Windows 10, you'll have new **Start Menu** items under **Anaconda**. Click the **Start Menu**, then scroll down to **Anaconda** then expand by clicking **Anaconda** and you'll see several items.



Anaconda Prompt works like a Windows command window (an interactive prompt). It uses Windows (DOS-like) type commands, not Unix/Linux commands.

Clicking **Jupyter Notebook** will open a command prompt window and a browser window with Jupyter.



Select items to perform actions on them.

Name	Last Modified
3D Objects	14 days ago
Contacts	14 days ago
Cookies	2 years ago
Desktop	12 minutes ago
Documents	13 days ago
Downloads	36 minutes ago
Favorites	14 days ago
Links	14 days ago
Lucidlogix	6 years ago
mu_code	a month ago
Music	14 days ago
OneDrive	2 months ago
	2 days ago

Installing the CircuitPython Kernel

Kernels are the main logic groups inside Jupyter. The default Windows installation only installs Python 3. We'll need to get a bit more code for the CircuitPython notebook.

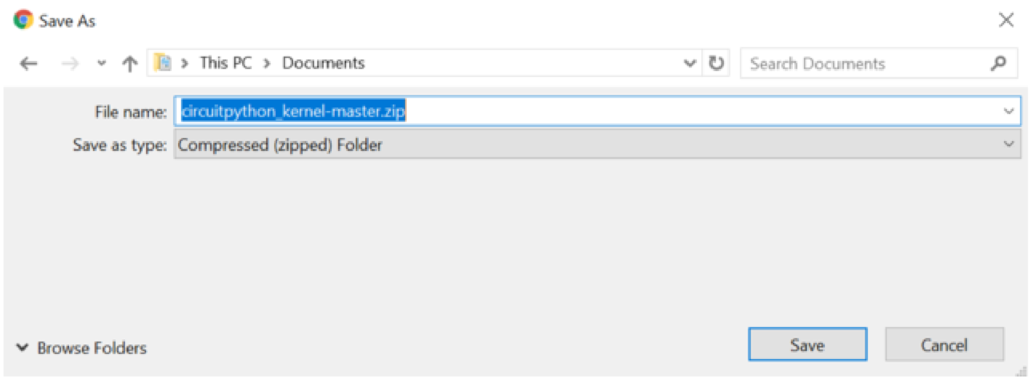
First we need to download the latest kernel file from the Adafruit GitHub website.

Click the button below or in your web browser, go to https://github.com/adafruit/circuitpython_kernel (https://adafru.it/BLj)

https://adafru.it/BLv

https://adafru.it/BLv

Save the file in your Windows **Documents** folder.



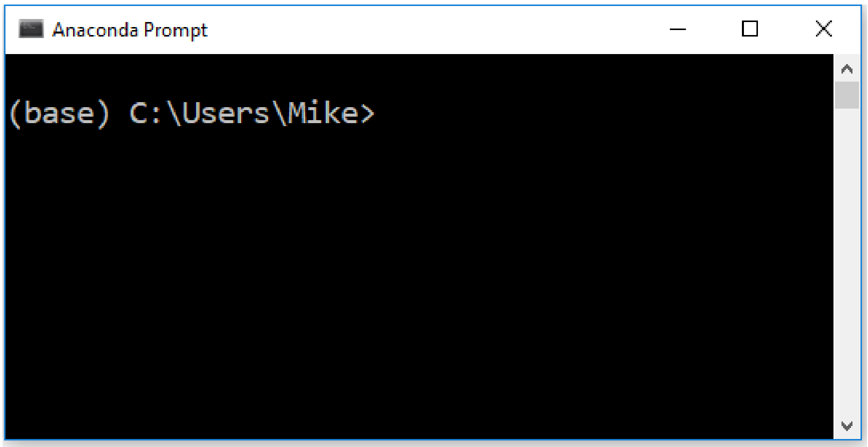
File name: circuitpython_kernel-master.zip
Save as type: Compressed (zipped) Folder

Next use your Windows File Explorer to go to the **Documents** folder.

Double-click the **circuitpython_kernel-master.zip** file. Copy the directory in the ZIP file named **circuitpython_kernel-master** into the **Documents** folder ("unzipping it" so we can use the files).

Now the files for the kernel are in your user's **Documents** folder in the **circuitpython_kernel-master** directory.

Run Anaconda Prompt by going to **Start Button** -> **Anaconda** -> **Anaconda Prompt**. You'll get the terminal like window below:



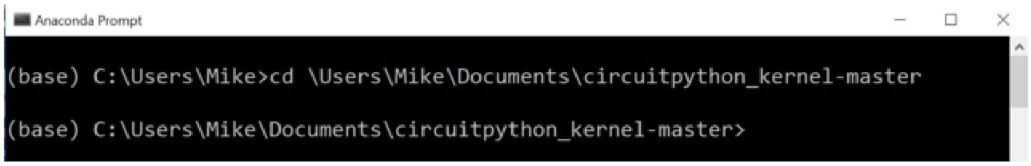
```
(base) C:\Users\Mike>
```

You will probably have another username besides Mike. Note what that username is as you'll need it in the next steps.

Type the following commands in Anaconda Prompt:

`cd \Users\yourusername\Documents\circuitpython_kernel-master`

Your prompt (the part which is before the typing area) should show that you are in the directory. If you do not see that you are in the **circuitpython_kernel-master** directory, double check your typing and try again.

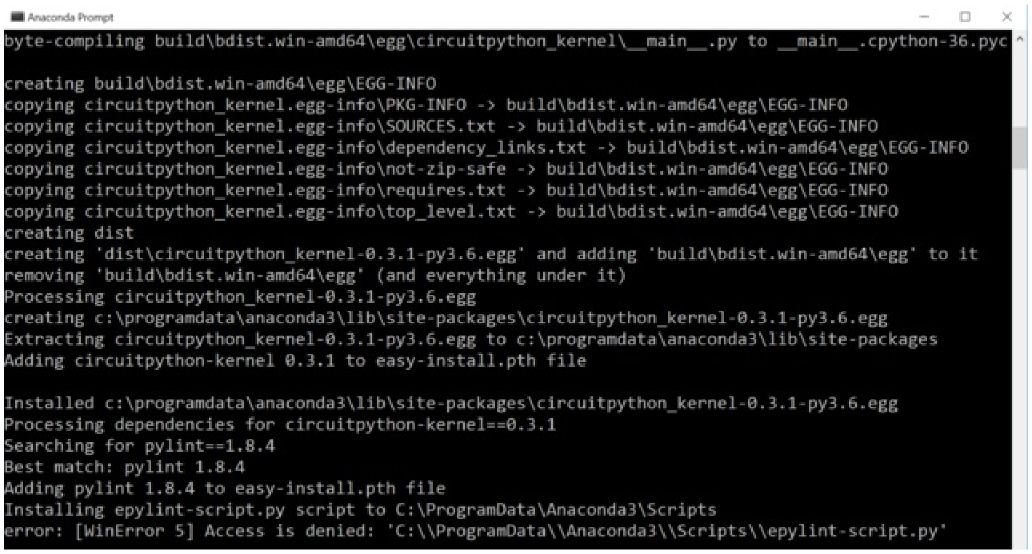


```
(base) C:\Users\Mike>cd \Users\Mike\Documents\circuitpython_kernel-master

(base) C:\Users\Mike\Documents\circuitpython_kernel-master>
```

If all is ok, then we'll start installing!

Type:

`python setup.py install`



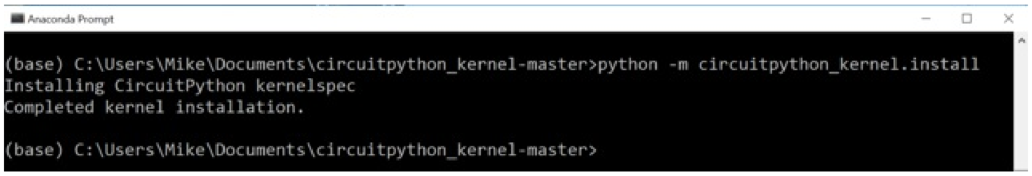
```
byte-compiling build\bdist.win-amd64\egg\circuitpython_kernel\__main__.py to __main__.cpython-36.pyc

creating build\bdist.win-amd64\egg\EGG-INFO
copying circuitpython_kernel.egg-info\PKG-INFO -> build\bdist.win-amd64\egg\EGG-INFO
copying circuitpython_kernel.egg-info\SOURCES.txt -> build\bdist.win-amd64\egg\EGG-INFO
copying circuitpython_kernel.egg-info\dependency_links.txt -> build\bdist.win-amd64\egg\EGG-INFO
copying circuitpython_kernel.egg-info\not-zip-safe -> build\bdist.win-amd64\egg\EGG-INFO
copying circuitpython_kernel.egg-info\requires.txt -> build\bdist.win-amd64\egg\EGG-INFO
copying circuitpython_kernel.egg-info\top_level.txt -> build\bdist.win-amd64\egg\EGG-INFO
creating dist
creating 'dist\circuitpython_kernel-0.3.1-py3.6.egg' and adding 'build\bdist.win-amd64\egg' to it
removing 'build\bdist.win-amd64\egg' (and everything under it)
Processing circuitpython_kernel-0.3.1-py3.6.egg
creating c:\programdata\anaconda3\lib\site-packages\circuitpython_kernel-0.3.1-py3.6.egg
Extracting circuitpython_kernel-0.3.1-py3.6.egg to c:\programdata\anaconda3\lib\site-packages
Adding circuitpython-kernel 0.3.1 to easy-install.pth file

Installed c:\programdata\anaconda3\lib\site-packages\circuitpython_kernel-0.3.1-py3.6.egg
Processing dependencies for circuitpython-kernel==0.3.1
Searching for pylint==1.8.4
Best match: pylint 1.8.4
Adding pylint 1.8.4 to easy-install.pth file
Installing epylint-script.py script to C:\ProgramData\Anaconda3\Scripts
error: [WinError 5] Access is denied: 'C:\\ProgramData\\Anaconda3\\Scripts\\epylint-script.py'
```

You'll get a lot of output and hopefully it should show success. if there is an error about something not found, ensure you are in the directory with all the code from the ZIP file. If there is an error about epylint (like above), that's ok right now, your install should still work ok.

After the **setup.py** installs, type:

`python -m circuitpython_kernel.install`

and you should see the text:

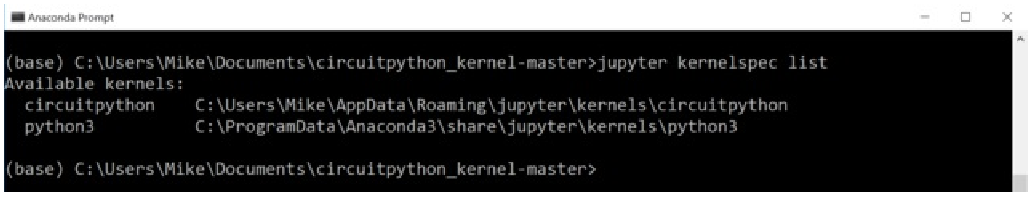


```
(base) C:\Users\Mike\Documents\circuitpython_kernel-master>python -m circuitpython_kernel.install
Installing CircuitPython kernelspec
Completed kernel installation.

(base) C:\Users\Mike\Documents\circuitpython_kernel-master>
```

Time to verify everything is ok! Type in the prompt window:

`jupyter kernelspec list`

which should print:



```
(base) C:\Users\Mike\Documents\circuitpython_kernel-master>jupyter kernelspec list
Available kernels:
  circuitpython    C:\Users\Mike\AppData\Roaming\jupyter\kernels\circuitpython
  python3          C:\ProgramData\Anaconda3\share\jupyter\kernels\python3

(base) C:\Users\Mike\Documents\circuitpython_kernel-master>
```

If you see both **circuitpython** and **python3**, you're set.

Using Jupyter Notebooks

We included some examples to get you started with using Jupyter Notebooks with the Circuit Playground Express (https://adafru.it/wpF) since it has lots of inputs and outputs to play with as is.

Mac or Linux

You can start a Jupyter notebook by running the following in a terminal

```
jupyter notebook
```

Windows

First, ensure there are no Jupyter terminal windows open (use the Ctrl-C key in the window to close one if you have one open. We'll start a new copy of Jupyter:

Start Button -> Anaconda -> Anaconda Prompt

The Anaconda command terminal will open. we need to go where the files were unzipped:

You'll have the **(base) C:\Users\*yourusername*>** prompt.

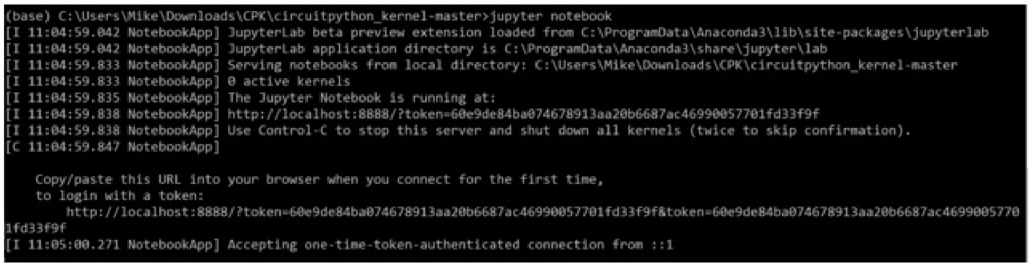
Type the following:

`cd Downloads\CPK\circuitpython_kernel-master`

There type `jupyter notebook`

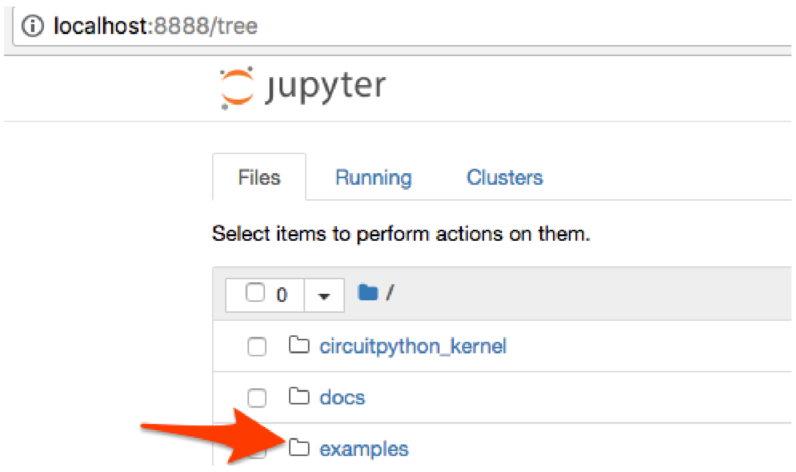
Load up the Example Folder

OK no matter what you're running, you should have a screen like the one below on your Anaconda prompt window and the same Jupyter screen below as the mac/linux users.

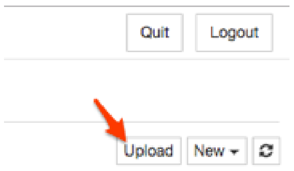


```
(base) C:\Users\Mike\Downloads\CPK\circuitpython_kernel-master>jupyter notebook
[I 11:04:59.042 NotebookApp] JupyterLab beta preview extension loaded from C:\ProgramData\Anaconda3\lib\site-packages\jupyterlab
[I 11:04:59.042 NotebookApp] JupyterLab application directory is C:\ProgramData\Anaconda3\share\jupyter\lab
[I 11:04:59.833 NotebookApp] Serving notebooks from local directory: C:\Users\Mike\Downloads\CPK\circuitpython_kernel-master
[I 11:04:59.833 NotebookApp] 0 active kernels
[I 11:04:59.835 NotebookApp] The Jupyter Notebook is running at:
[I 11:04:59.838 NotebookApp] http://localhost:8888/?token=60e9de84ba074678913aa20b6687ac46990057701fd33f9f
[I 11:04:59.838 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 11:04:59.847 NotebookApp]

    Copy/paste this URL into your browser when you connect for the first time,
    to login with a token:
        http://localhost:8888/?token=60e9de84ba074678913aa20b6687ac46990057701fd33f9f&token=60e9de84ba074678913aa20b6687ac4699005770
1fd33f9f
[I 11:05:00.271 NotebookApp] Accepting one-time-token-authenticated connection from ::1
```

If you started this from the directory we were using earlier (*circuitpython-kernel*), you'll see an *examples/* folder:

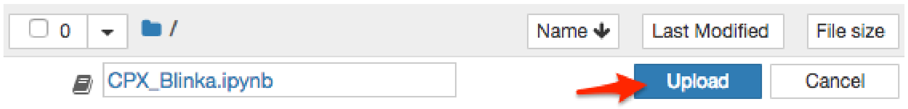


If you don't see this folder, click the upload button on the Notebook Dashboard

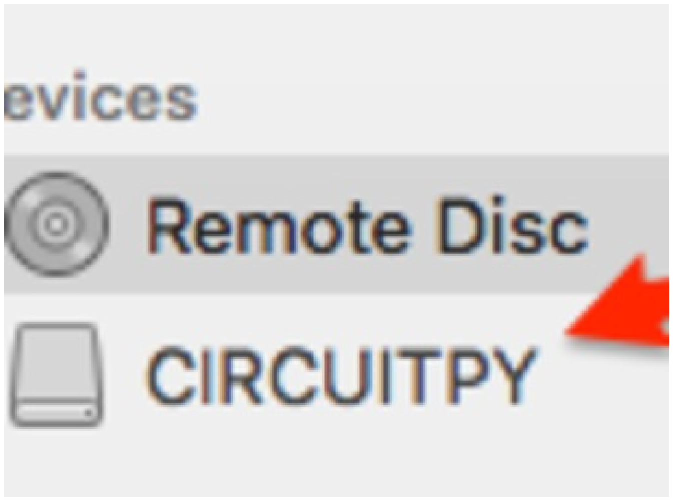


Navigate to the folder for *circuitpython-kernel (*or *circuitpython-kernel-master*) you cloned or downloaded earlier. Navigate to the examples folder. Then, click upload on the imported notebook on your dashboard.

Locate the **CPX_Blinka.ipynb** notebook but **don't** click it yet!



Plug In The CircuitPython Board



At this point, you'll want to plug in your board and verify it shows up as a mountable drive called **CIRCUITPY**

If you don't see the **CIRCUITPY** drive, install CircuitPython on your board (https://adafru.it/Amd).

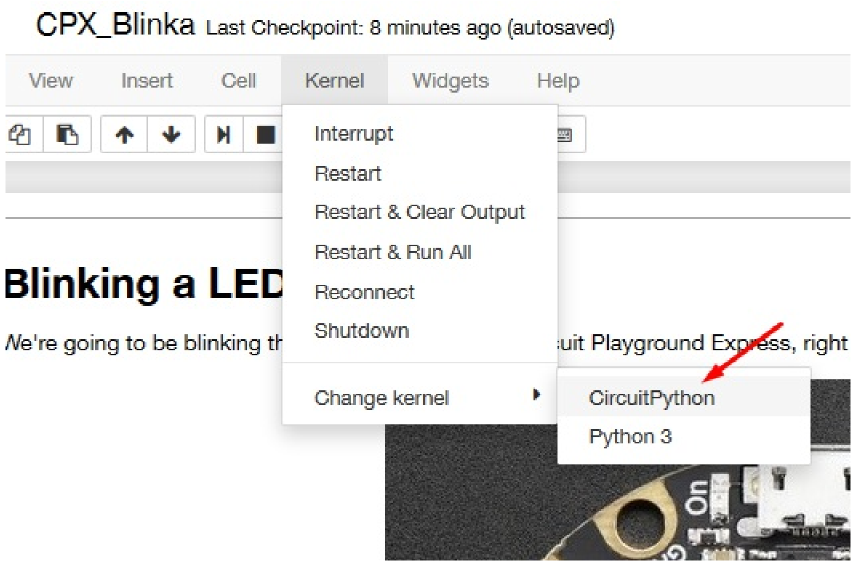
Ensure your board is not connected to a serial terminal somewhere else on your computer, the kernel needs to make a connection to an open port.

Launch the CPX Blinka Example

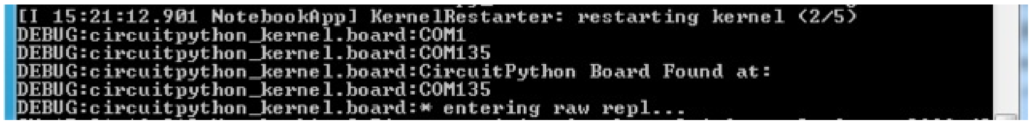
OK now we're ready.

Click on **CPX_Blinka.ipynb** to launch the Jupyter Notebook and the CircuitPython Kernel.

Make sure your notebook is running the **CircuitPython** kernel by clicking *change kernel -> CircuitPython:*



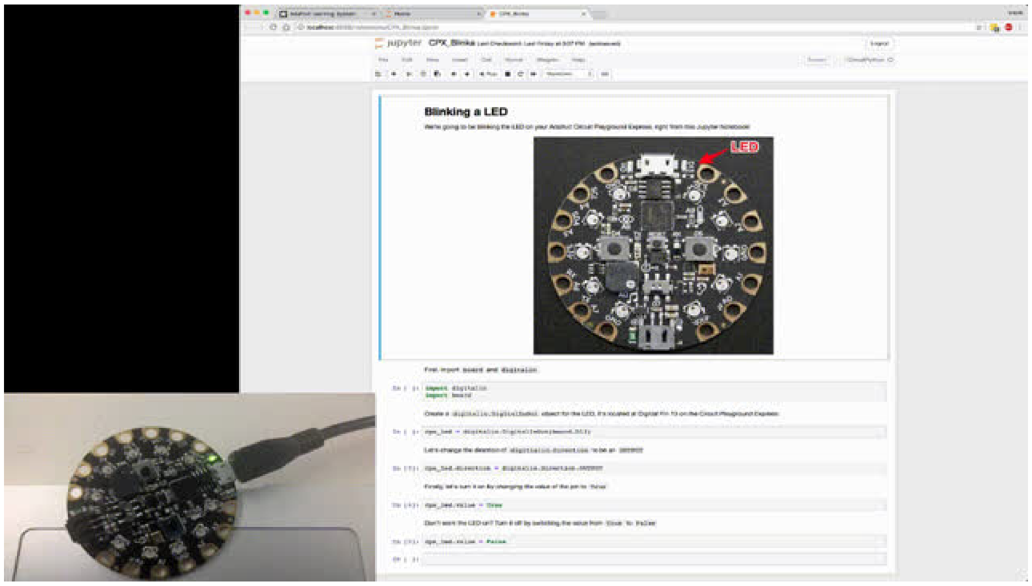
When you launch notebooks using the CircuitPython-Kernel, it'll search the serial ports on your computer and connect to your Adafruit CircuitPython board.



```
[I 15:21:12.901 NotebookApp] KernelRestarter: restarting kernel (2/5)
DEBUG:circuitpython_kernel.board:COM1
DEBUG:circuitpython_kernel.board:COM135
DEBUG:circuitpython_kernel.board:CircuitPython Board Found at:
DEBUG:circuitpython_kernel.board:COM135
DEBUG:circuitpython_kernel.board:* entering raw repl...
```

The **CPX_Blinka** notebook should open. We're going to use the notebook and learn about Jupyter in the process.

The first cell should be highlighted. Each of these cells hold media (like pictures, equations, or rich text) or code, executable by your CircuitPython board. Click **run** in the toolbar or press **shift+enter** to run the cell:



Blinking a LED

We're going to be blinking the LED on your Adafruit Circuit Playground Express, right from this Jupyter Notebook!

First import board and digitalio

```
In [ ]: import digitalio
        import board
```

Create a digitalio.DigitalInOut object for the LED. It's located at Digital Pin 13 on the Circuit Playground Express.

```
In [ ]: cpx_led = digitalio.DigitalInOut(board.D13)
```

Let's change the direction of digitalio.Direction to be an OUTPUT

```
In [7]: cpx_led.direction = digitalio.Direction.OUTPUT
```

Finally, let's turn it on by changing the value of the pin to True

```
In [8]: cpx_led.value = True
```

Don't want the LED on? Turn it off by switching the value from True to False

```
In [7]: cpx_led.value = False
In [ ]:
```

As you go use the **run** command to move the cells, you should see the Circuit Playground's LED turn on when the cell containing `cpx_led.value = True` is executed:

Congrats - you just ran your first Jupyter Notebook!

Want to try another example? Navigate to the Notebook Dashboard and navigate to the *examples/* folder.

Next, we'll learn how to create our own Jupyter Notebook.

Troubleshooting

I get a "The kernel appears to have died" dialog and then it enters a reset-loop

Open your terminal to view the errors from the kernel. If the problem persists, copy the output from the terminal and file an issue on the GitHub Repository for this project.

My terminal is displaying an error: Port must be configured before it can be used.

There's an issue with your serial connection which is preventing the kernel from opening a serial connection with your board. Make sure your drivers are installed correctly and the CircuitPython board is displayed as **CIRCUITPY** drive in file explorer.

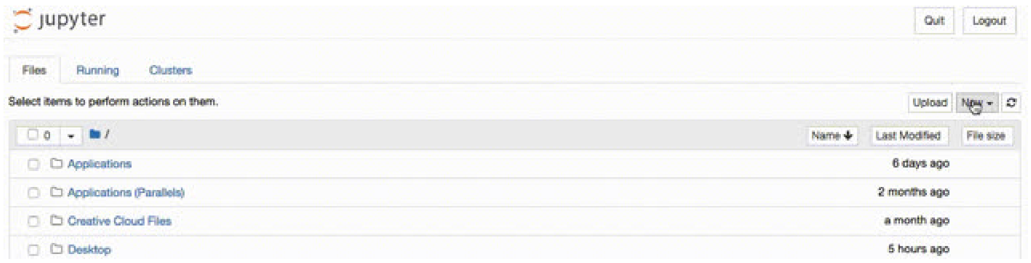
Creating Jupyter Notebooks

Let's walk through creating a new Jupyter Notebook using the CircuitPython Kernel.

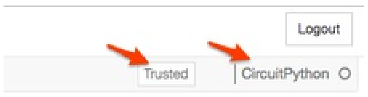
Creating a CircuitPython Jupyter Notebook

Make sure your CircuitPython board is plugged into your computer and displays as a removable drive 'CIRCUITPY' before creating a new notebook

To create a new Jupyter Notebook for use with the circuitpython kernel, navigate to the Notebook Dashboard and click **New -> Notebook -> CircuitPython**

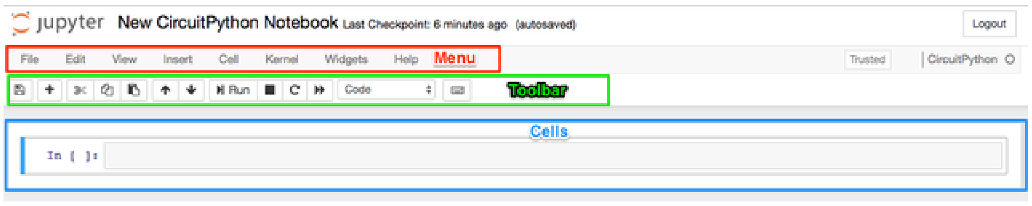


Your notebook should connect to your board using the CircuitPython kernel. You can verify this by checking two things on the toolbar: the *Trusted* and CircuitPython badges should display



Navigating Jupyter Notebooks

All Jupyter Notebooks contain three areas: menu, toolbar, and the code cells



The **Menu** contains actions on the notebook and the kernel. You can add/remove cells, restart the kernel, and perform file operations, all from the menubar.

The **toolbar** has buttons for common actions within the notebook. You can switch cell modes (between code/text), run/restart cells, and save the Notebook state.

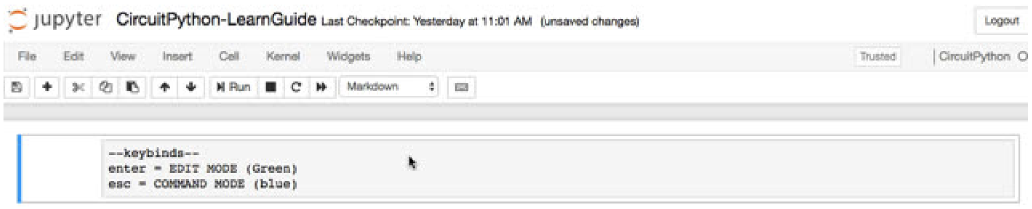
Cells are text-input fields which can be executed (the code gets sent to the kernel) and display output. They're the interactive part of Jupyter, and a bit different from any other software out there. There are three different types of cells, we'll cover into the first two in this guide:

- Code Cells
- Markdown Cells

- Raw Cells

Jupyter Modes

Like the Vim Editor, there are two different keyboard input modes in Jupyter which control whether you're controlling the notebook **(command mode)** or editing the code **(edit mode)** within a cell.


jupyter CircuitPython-LearnGuide Last Checkpoint: Yesterday at 11:01 AM (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
CircuitPython
Run
Markdown

```
--keybinds--
enter = EDIT MODE (Green)
esc = COMMAND MODE (blue)
```

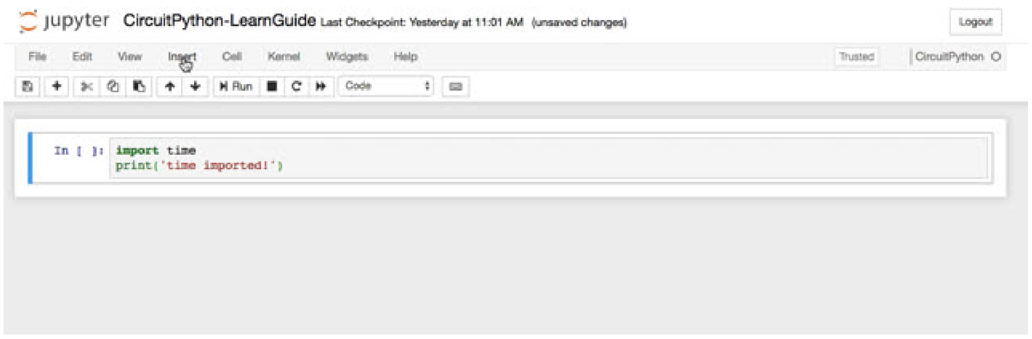
Command Mode allows you to perform notebook actions such as switching cell-type (code to markdown), executing code on the kernel, stopping the kernel and inserting cells. You can get into command mode by pressing the **enter key**, the cell will turn blue.

To enter **edit mode**, press the escape key on your keyboard. The cell's border will turn blue and you're able to type into the text-box of the cell (multi-line code *is* supported).

Using Cells

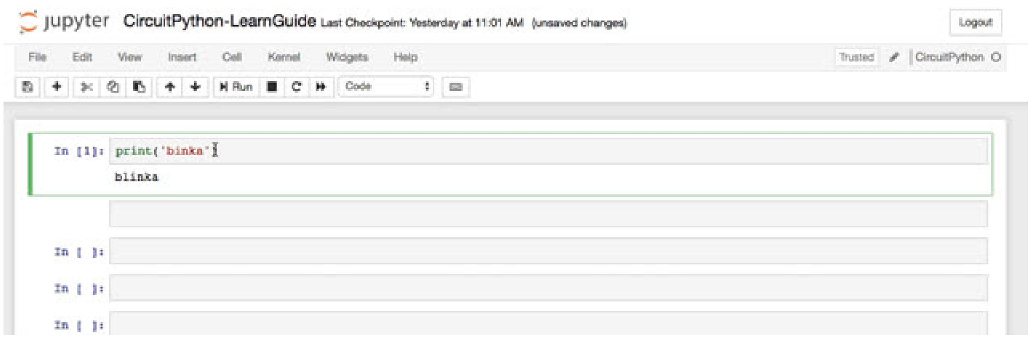
Want to write some CircuitPython code? You'll need a code cell. You'll be able to execute the code cell using the run button on the toolbar or by pressing **shift+enter**

To create new code cell, click **Insert -> Insert Cell Below** (or Above)


jupyter CircuitPython-LearnGuide Last Checkpoint: Yesterday at 11:01 AM (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
CircuitPython
Run
Code

```
In [ ]: import time
        print('time imported!')
```

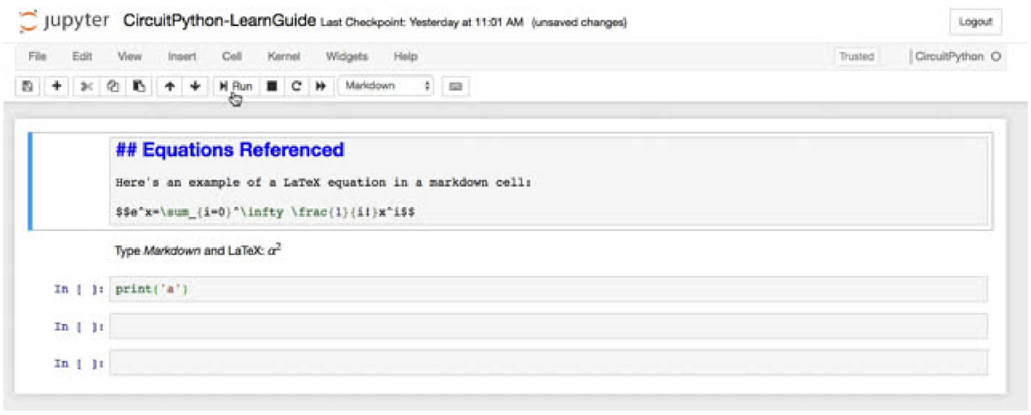
Code Cells are also capable of returning errors from the CircuitPython REPL so you'll know when your code is broken.


jupyter CircuitPython-LearnGuide Last Checkpoint: Yesterday at 11:01 AM (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
CircuitPython
Run
Code

```
In [1]: print('binka')
        blinka

In [ ]:
In [ ]:
In [ ]:
```

Want to document your progress by adding text, images, rich media, or even equations? You'll want to use a Markdown Cell.

You can switch to a markdown cell by clicking the *cell selection* dropdown on the toolbar and clicking *Markdown*



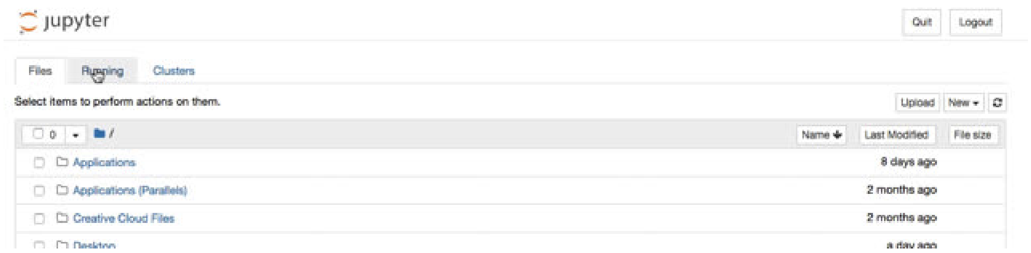
```
## Equations Referenced

Here's an example of a LaTeX equation in a markdown cell:

$$e^x=\sum_{i=0}^\infty \frac{1}{i!}x^i$$
```

Type *Markdown* and LaTeX: α^2

```
In [ ]: print('a')

In [ ]:

In [ ]:
```

Not familiar with the features of Markdown? Check out the *Working with Markdown Cells* example notebook (https://adafru.it/ByP)

Managing Notebooks



Select items to perform actions on them.

Name	Last Modified
Applications	8 days ago
Applications (Parallels)	2 months ago
Creative Cloud Files	2 months ago
Desktop	a day ago

Finished writing in your notebook? Even if you've closed the tab where you're working in, the notebook is still running.

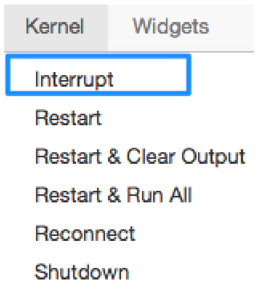
From your dashboard, navigate to the **Running** tab. You'll see your active notebooks, click **Shutdown** to shut the kernel down.

Now that we've learned about creating and editing a Jupyter Notebook, we're going to learn how to share it with others.

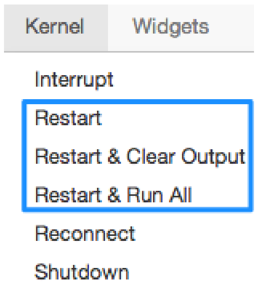
Troubleshooting the Kernel

If your notebook is unresponsive, *don't disconnect your board just yet.* Code sometimes fails or freezes. Luckily, Jupyter Notebooks have controls for kernels built in so you can stop any errors/infinite loops you encounter.

First, we can interrupt the kernel's code execution by clicking **kernel -> interrupt**. This'll retain the program execution state and variables which may have been created earlier.



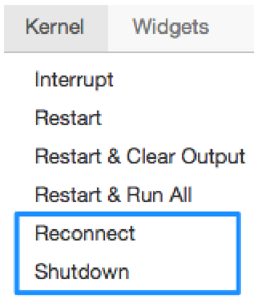
Still frozen? Try restarting your kernel. There are three options for this:



- **Restart:** Restarts the kernel and loses state (variables, etc). The board resets its connection with the kernel. The output from the previous run will be retained along with the cells.
- **Restart & Clear Output:** You can also restart (as explained above) and clear the output of your cells. The code/markdown within the cell will be retained.
- **Restart & Run All:** Restarts the kernel, loses variable states, reconnects the board, and runs all cells.

Still no luck? Having a critical issue?

You can **reconnect** (this will re-establish a connection with the board) or **shutdown** the kernel (however, all variables will be lost if you choose this).

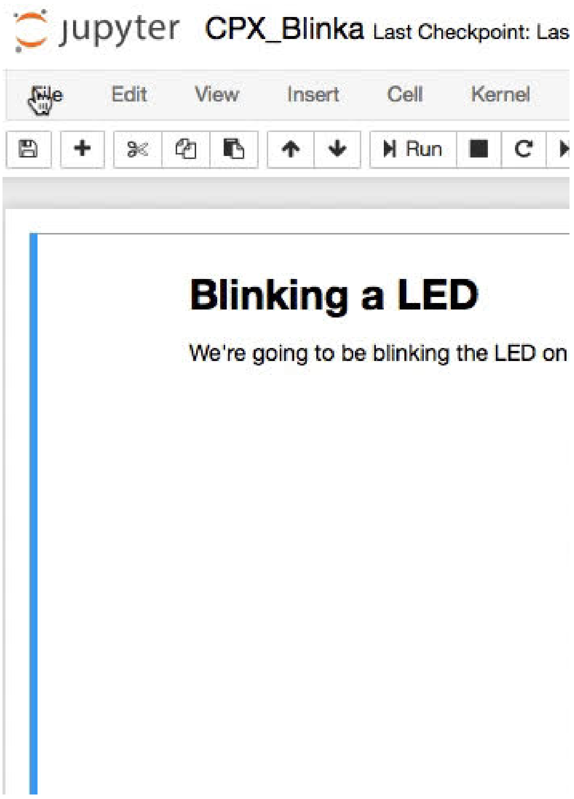


Sharing Jupyter Notebooks

Ready to share your notebook with the world? We're going to highlight a few different options for sharing it with anyone.

Sharing Locally

You can export to a variety of formats from within the notebook by navigating to **File -> Download As**. You'll want to export your notebook as a Jupyter Interactive Notebook (`.ipynb` file format) if you'd like the person you're sharing it with to interact with the notebook.



Blinking a LED

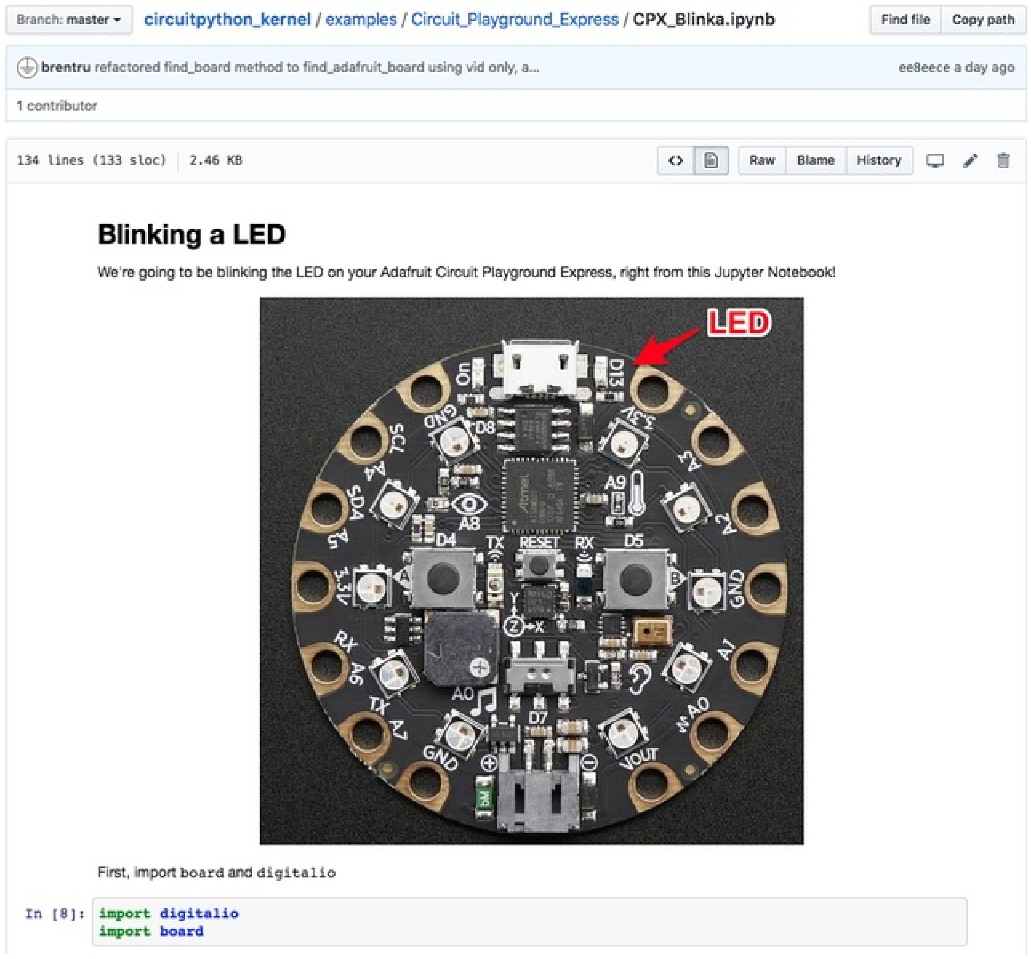
We're going to be blinking the LED on

Sharing Online

Want the other person to preview your notebook before opening it? There are a few options for giving the other person a chance to view it online before downloading the file, we like using GitHub and nbviewer.

GitHub

Have a GitHub account? If you drop a Jupyter notebook (`.ipynb`) file in your repository, it'll be automatically rendered directly in the browser. It works for both public and private repositories.


Branch: master ▾ circuitpython_kernel / examples / Circuit_Playground_Express / CPX_Blinka.ipynb Find file Copy path

brentru refactored find_board method to find_adafruit_board using vid only, a... ee8eece a day ago

1 contributor

134 lines (133 sloc) | 2.46 KB Raw Blame History

Blinking a LED

We're going to be blinking the LED on your Adafruit Circuit Playground Express, right from this Jupyter Notebook!

First, import board and digitalio

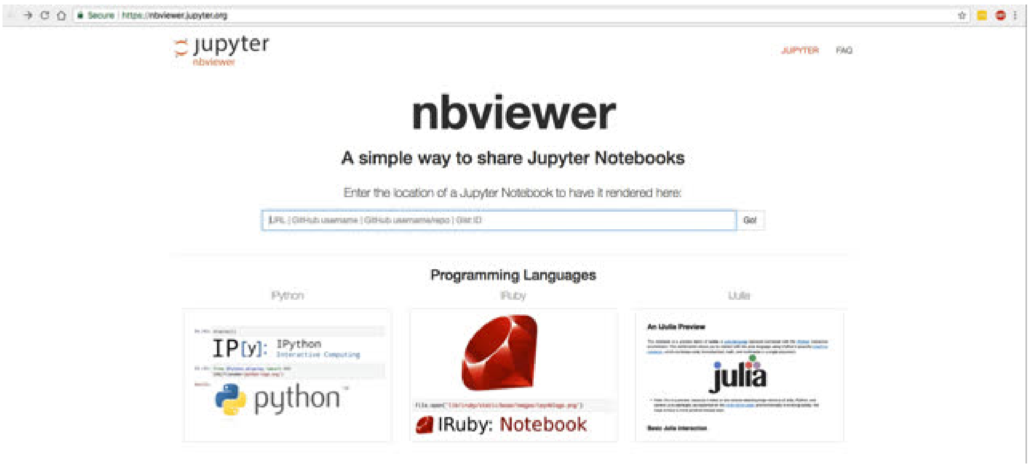
```
In [8]: import digitalio
        import board
```

Here's an example of our *CPX_Blinka.ipynb* rendered on GitHub on the official CircuitPython_Kernel repository (https://adafru.it/BLG)

nbviewer

The nbviewer (https://adafru.it/Bzo) is a Jupyter notebook viewer service hosted by Jupyter. Just give it a URL and it'll render it instantly!

It's the simplest way of having your notebook rendered online.


Secure | https://nbviewer.jupyter.org
jupyter
nbviewer
JUPYTER
FAQ

nbviewer

A simple way to share Jupyter Notebooks

Enter the location of a Jupyter Notebook to have it rendered here:

URL | GitHub username | GitHub username/repo | Gist ID Go!

Programming Languages

Python
IRuby
IJulia
IP[y]: IPython Interactive Computing
python
IRuby: Notebook
julia

While neither of these two methods of sharing notebooks will *run* CircuitPython code, but they can be uploaded to a users' local Jupyter installation.