

# Get Started Guide for Python Jupyter Notebooks



The goal of this document is a step-by-step guide to get started using the Python Jupyter Notebooks to perform computational projects in Engineering Math classes. This guide assumes you have some basic knowledge of computer programming but need a few specifics to get started with Python Jupyter Notebooks. If you already have experience running Python on Jupyter Notebooks, you can skip this document.

Before you start, it must be assumed you have a good working laptop with reasonably up-to-date operating system, Mac or Windows. If not, proceed to a U of Utah computer lab computer and open the MATLAB version of the code assignment. It is also assumed that you have downloaded some sample Jupyter notebook “.ipynb” file that was/will be given to you in your Engineering Math class and you will save the file in some file folder where you will store your work. In the following we detail two ways to open Python Jupyter. Then we detail what Jupyter notebooks are, how they work, and code samples to try.

## Two ways to run Python Jupyter notebooks


- (A) Download and install Anaconda Python 3 onto your computer.
- (B) Use Google Drive Colaboratory.

### Anaconda Python 3 step-by-step:

1. Download the version-3 distribution of Python from Anaconda.com to your computer if you don't already have a Python package installed:

<https://www.anaconda.com/download/>

2. Open the Anaconda-Navigator app you just installed. It will open a window with sub appli-

cation options in the Home tab. Choose JupyterLab . Other applications will also work, but the JupyterLab option contains both the Jupyter notebook and a left-side file-system navigation column that displays computer's directory, which is useful for keeping track of multiple projects.

3. Within JupyterLab, in the left file column, navigate to your chosen file folder by clicking on the file icons. Before you start fiddling with the saved code in that folder, also open a new file using the “+” icon in the left-side column and select a new Notebook file which will be titled “Untitled.ipynb”.
4. Within your new Jupyter notebook file “Untitled.ipynb” (rename if you like) you can see a single empty code block. Within that code block, do the first thing everyone does when programming, type “print('hello world')” and then type Control+Return to run the code OR you can click the “>”. To make a new code block below the first block you can either re-run the previous code block by using the alternate command Option+Return, which runs and then opens a new code block field.

## On Google Drive with the Colaboratory App step-by-step:

1. Open a browser and go to your Google Drive web page (assuming you have a Google account and you've logged in!).
2. Click on the "New" button in the upper left, select "More" from the dropdown, and from 2nd dropdown select "Connect more apps". Type in "Colaboratory" in the search field of the resulting pop-up menu, and click on "Connect."
3. Download or open any of the course-supplied ".ipynb" Notebook files, or make an empty new file.
4. Click on the file in Drive, and you may have to select to use the Colaboratory app to open the file. Once open, you can edit and run all the Notebook code snippets right in Drive. Done!

## What is a Jupyter notebook?

Most programming languages including Python (without Jupyter) read the code you write and execute it as a program from the start of your code and end on the last thing you wrote. After execution, all stored data objects are deleted. However, when doing math or figuring out a programming problem, it's useful to be able to write one bit of code that calculates something, store the result, and then use the result later without having to re-run the code from the start again. This storability is critical to how you use a graphing calculator or MATLAB, for example, because you may need to modify/recompute subparts of the program many times before you get what you want. Jupyter notebooks gives Python this functionality. Jupyter is a working environment where you can write and execute code blocks, store the results, and the stored results within subsequent code blocks.

1. Let's do some math in Jupyter: run a code block with the assignment statements " $a = 2$ " then hit return and then type " $b = 3$ " on the next line, then run the block. This stores the data  $a$  and  $b$ . On the next code block run " $a + b$ ", which will compute the result.
2. Let's plot a basic function:  $y = x^2$

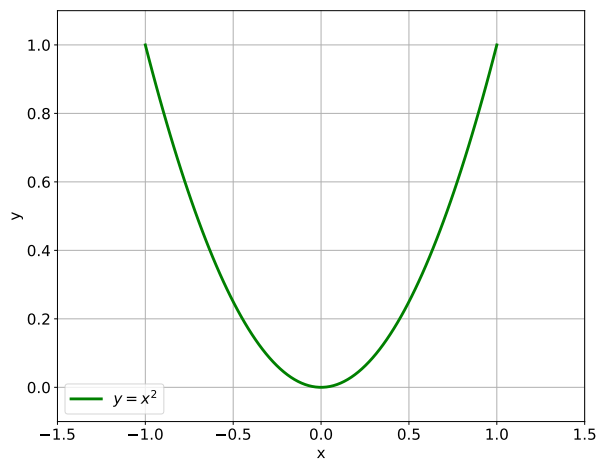
```
# begin code:
import matplotlib.pyplot as plt
import numpy as np

def f(x):
    return x**2

x = np.linspace(-1, 1, 100)
y = f(x)

plt.figure(figsize=(10, 8), dpi= 80, facecolor='w', edgecolor='k')
plt.rcParams.update({'font.size': 16})
plt.plot(x, y, label='$y=x^2$', color='green', linewidth=3)
plt.xlabel('x')
plt.ylabel('y')
plt.xlim([-1.5, 1.5])
plt.ylim([-0.1, 1.1])
plt.legend()
plt.grid()
plt.savefig('Your_First_Python_Graph.pdf')
plt.show()
```

Make sure the code is properly indented. Unless a line is underneath function definition “def” or below a for-loop or if-statement, the code should not be indented. Also, note that copy-pasting from .pdf to other applications can sometimes mis-code certain character types and you may need to re-type them.



3. Finally, now that you’ve done some basics, open up the Jupyter notebook file that was given to you in class and start running it block by block without changing any of the code itself. Then, to complete your assignment, it’s a good idea to make a new notebook file and copy-paste and then modify what you need from the original code. This way if you make a mistake you can always refer back to the original unmodified working code.