

## **Getting Started with Jupyter Notebooks**

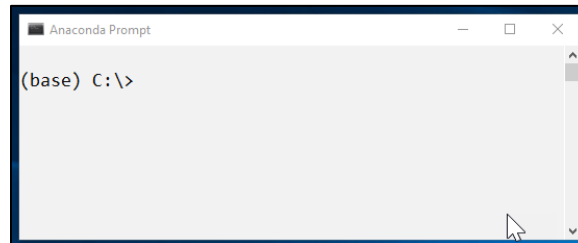
*Please note, this guide assumes you are using the Anaconda distribution of Python, which includes Python and Jupyter. You can download Anaconda from: <https://www.anaconda.com/download/>. This tutorial was created using Python 3.6.5.*

*This tutorial also assumes you have a basic understanding of Python syntax and functionality. The focus of this guide is on the Jupyter notebook interface, not an introduction to Python.*

### **Part 1: How to open Jupyter**

To get started, open the Anaconda prompt. An easy way to do this is to search “Anaconda prompt” in the desktop search box or file explorer.

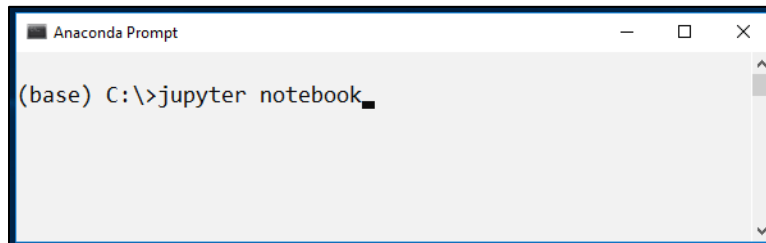
My home directory for the Anaconda prompt is the C: drive, so my open prompt looks like this:



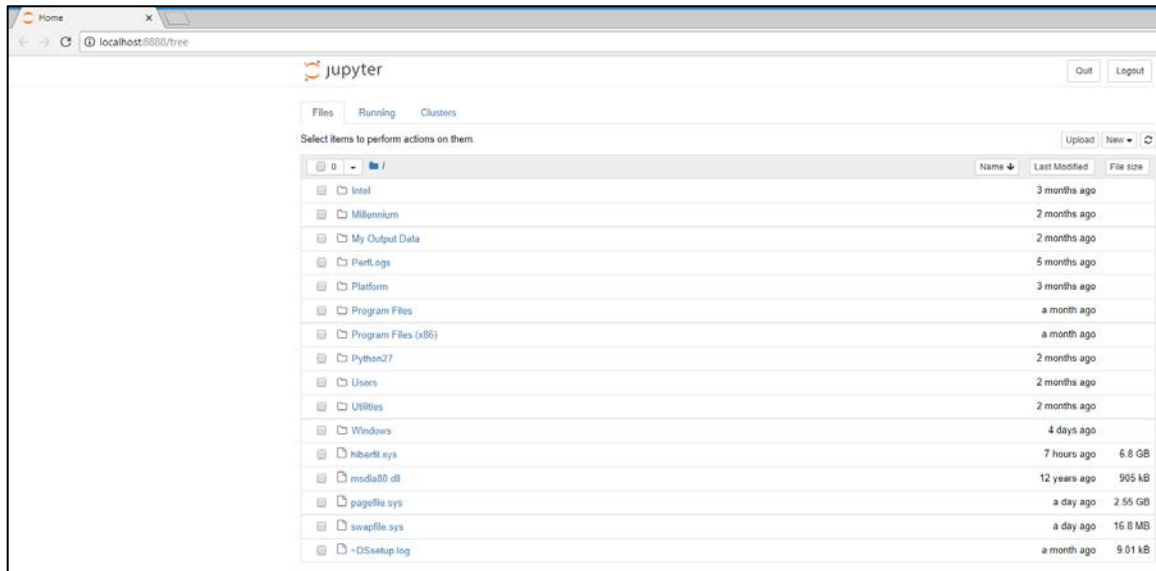
Your location (file path) may vary. For instance, if your home directory is your user folder, your prompt may have a file path along the lines:

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

From wherever your home directory is, type “jupyter notebook” in the prompt, then press enter.



This will open up a browser interface that looks like this (similar to File Explorer on Windows):



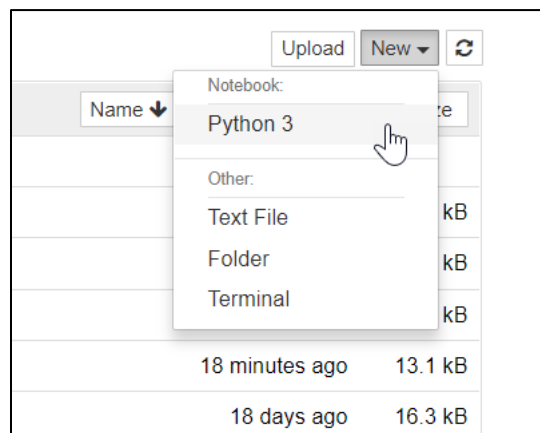
Although this looks like an internet browser, Jupyter is running locally – you do not need to be connected to the internet for this to work. If for any reason a Jupyter interface does not open following the steps above, you can access it by opening a browser, like Chrome, and entering the URL <http://127.0.0.1:8888>.

## **Part 2: Creating and working with notebooks**

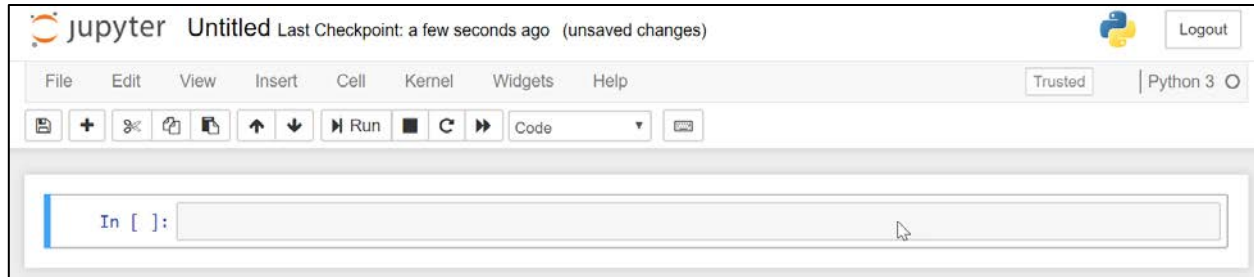
The Jupyter notebook interface will open in your home directory. If this is not the desired location for your notebook, use the Jupyter interface to navigate to wherever you want to create your notebook. For instance, maybe you have a folder for a specific class, or a separate folder for your research.

Alternatively, you can navigate to the desired location from the Anaconda prompt above, then open Jupyter from that location. Both methods have the same end result.

Once you're in the correct location, you're ready to create a notebook. On the upper right of the Jupyter interface, you'll see an option "New" with a drop down menu. Select Python 3. (Note: if you have a different version of Python, like 2.7, the Python version available to select here will be named accordingly.)

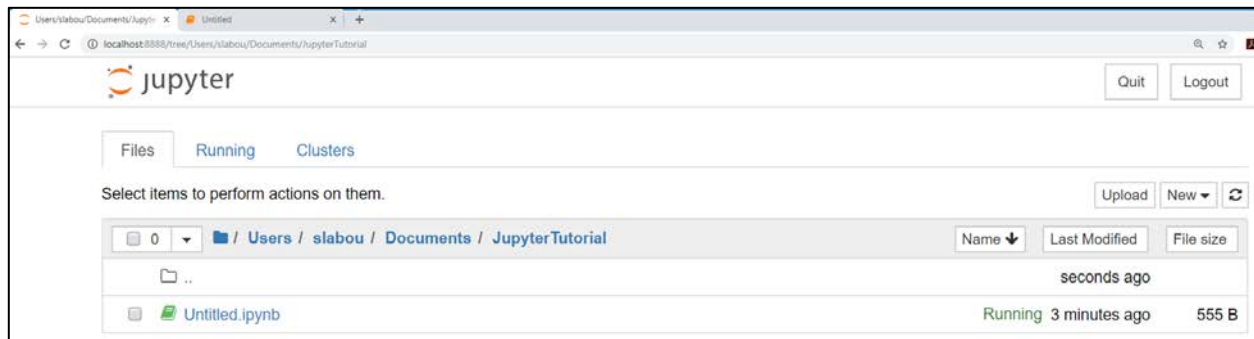


Selecting this will create an empty and untitled Jupyter notebook, which will look similar to:



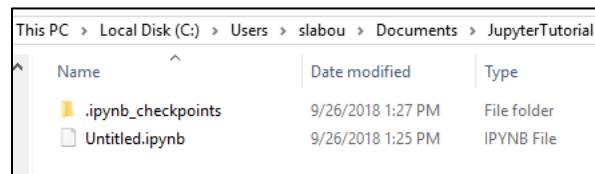
Congratulations! You have made your first Jupyter notebook.

You can see this file (with a .ipynb extension) in whichever folder you created it in. For instance, I created this file in a “JupyterTutorial” folder in my user documents (C:/Users/slabou/Documents/JupyterTutorial), so my Jupyter interface looks like this:



Note that I have one tab for the Jupyter navigational interface and one tab for the notebook itself.

This is what the notebook looks like in the File Explorer interface:

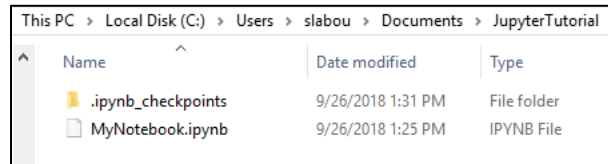
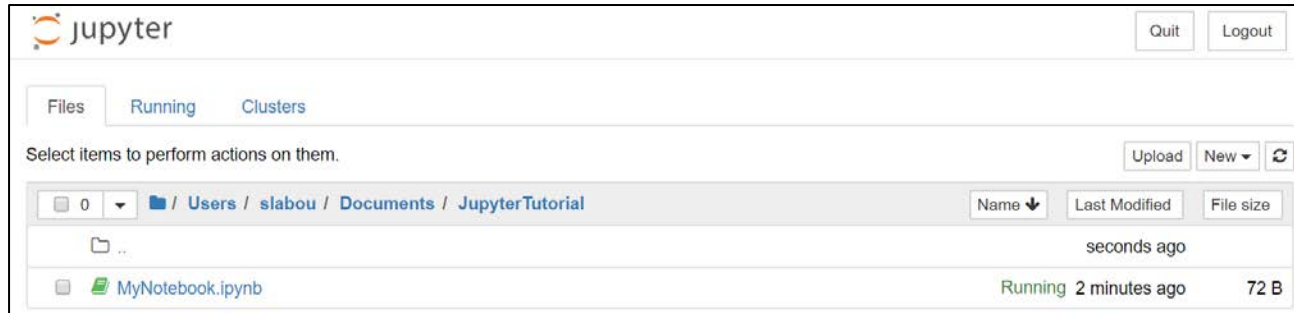


For now, you can ignore the “.ipynb\_checkpoints” folder.

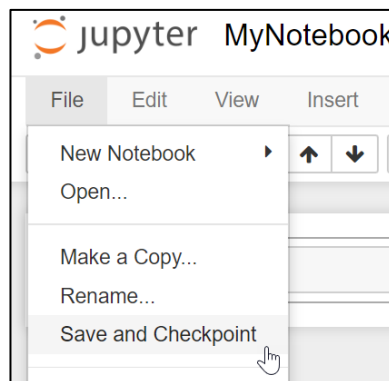
You can rename the notebook by double clicking where the “Untitled” text is and entering a new name.



This change is reflected in the Jupyter navigational tab and the local File Explorer.



Jupyter will autosave your notebook occasionally, but to be safe, be sure to save your notebook frequently. You can use the traditional shortcut of Control + s to save, or you can use File → Save and Checkpoint.



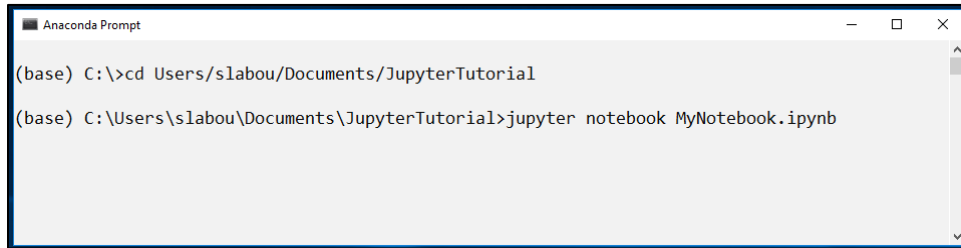
Checkpoints are a more advanced topic than this tutorial is meant to cover; for now, think of them like the last saved version. The important thing to remember is to save your work before you exit the notebook!

To close your notebook, exit the browser, then exit the Anaconda prompt.

To re-open a saved .ipynb file, you'll need to do so through the Anaconda prompt.

You can either open the Jupyter interface the same way as before (i.e., type "jupyter notebook" in the Anaconda prompt then navigate to your notebook location) or you can open the notebook directly.

To open the notebook directly, use the syntax “jupyter notebook filename.ipynb” in the Anaconda prompt. For instance, to open MyNotebook.ipynb, I would navigate to the folder where this file is located, then type “jupyter notebook MyNotebook.ipynb”.



```

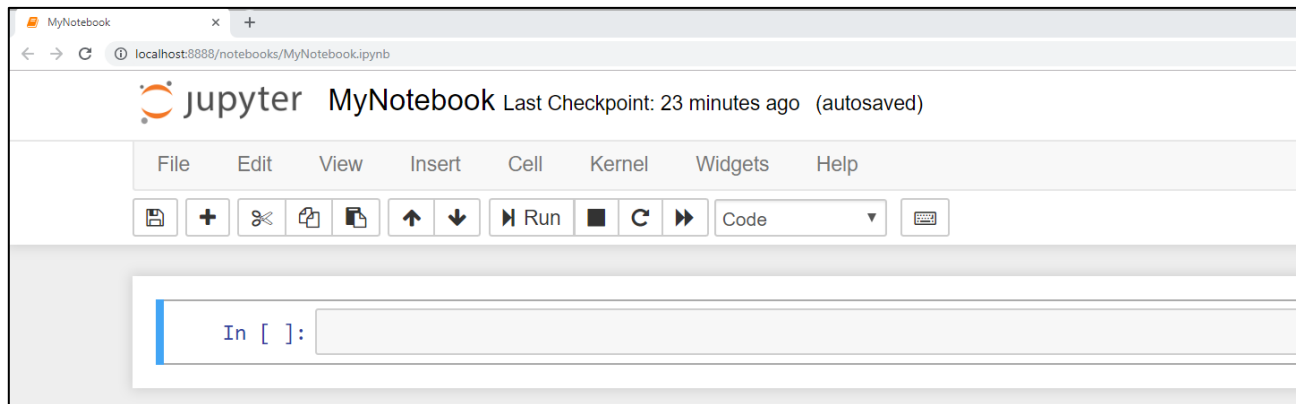
Anaconda Prompt

(base) C:\>cd Users\slabou/Documents/JupyterTutorial

(base) C:\Users\slabou\Documents\JupyterTutorial>jupyter notebook MyNotebook.ipynb

```

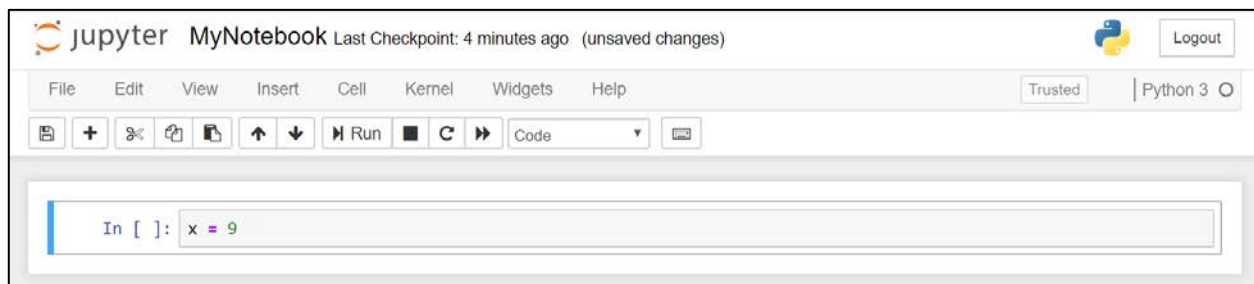
This brings me back to my notebook in the Jupyter interface. Note that this does not open a Jupyter navigational interface tab, only the notebook itself.



### **Part 3: Working with notebook cells**

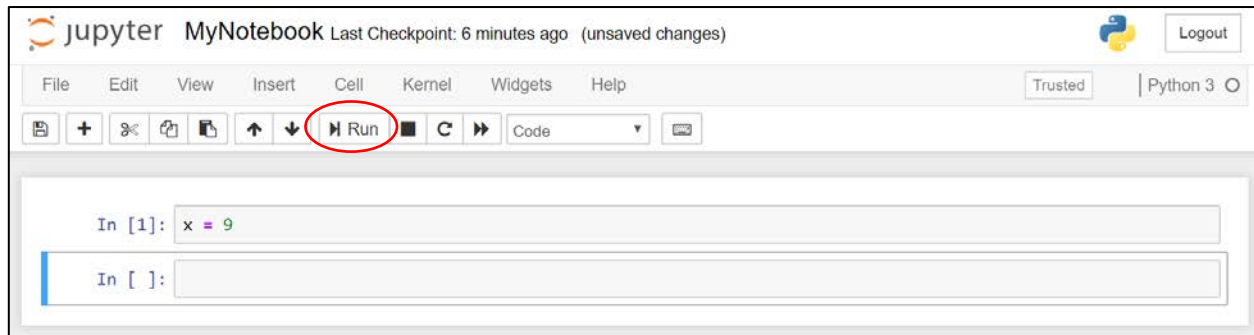
Now you're ready to get started coding!

Type in the cells (the rectangular gray boxes) as you would a regular line of code.

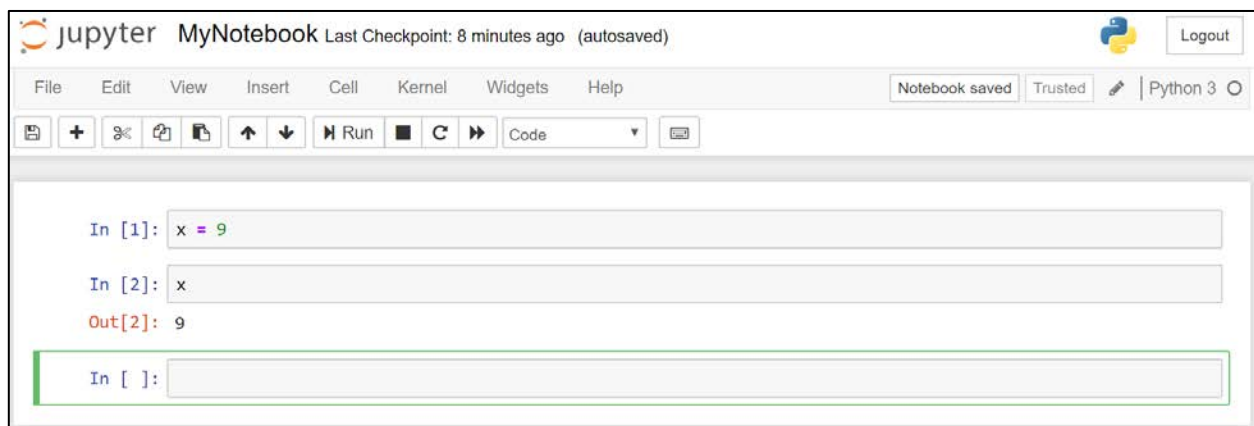


Pressing enter will add another line within the current cell. There is not a limit to how much code can be entered into a single cell.

To run the cell (i.e., run your code), you can click “Run” in the top pane or use a shortcut of shift + enter (on Windows). This will run the current cell and automatically add a new cell below.



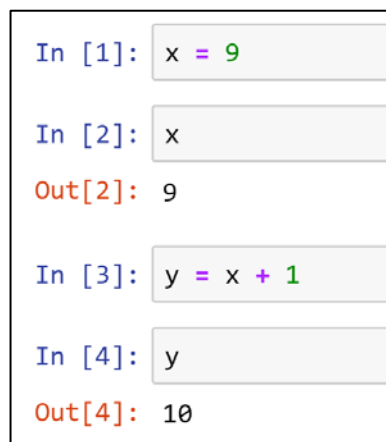
After you run a cell, you can access any variables created in that cell.



Note that the first cell is now labeled “1” and the second is labeled “2”. You’ll want to keep track of these numbers, which represent the sequence of cells run.

Let’s take a closer look at cell numbering.

Here’s an example of some basic code. Let’s say I run my first two cells; my variable x is set to 9. I do some calculation with my variable x. Everything looks good so far!



I decide I want to set x to be equal to a different value, so I change x to be 11.

```
In [1]: x = 11
In [2]: x
Out[2]: 9
In [3]: y = x + 1
In [4]: y
Out[4]: 10
```

I run the first cell, setting x=11. Note that the first cell is now labeled 5, because it is the 5<sup>th</sup> cell run in this session. Remember, I ran the first 4 cells (set x, return x, set y, return y), *then* I went back and re-ran the first cell to set x=11, making that the 5<sup>th</sup> “cell run” I’ve done.

```
In [5]: x = 11
In [2]: x
Out[2]: 9
In [3]: y = x + 1
In [4]: y
Out[4]: 10
```

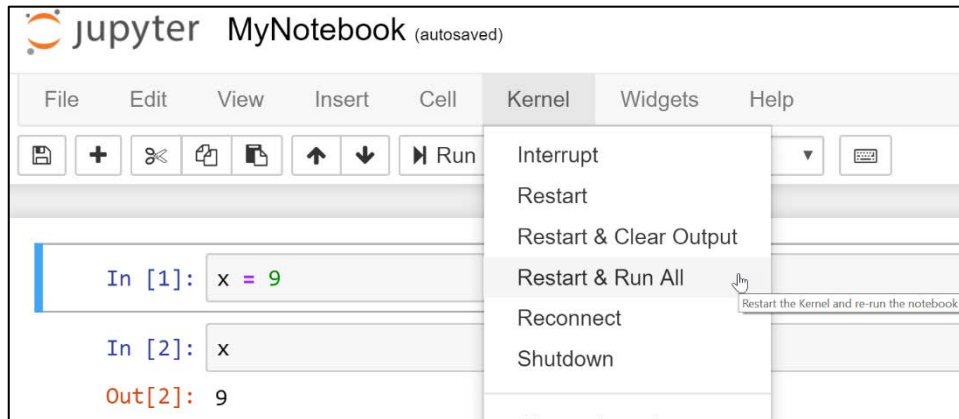
Now my cells are numbered out of order, which is a warning to me that some changes are not carried through to downstream cells. For instance, see that y is still equal to 10, when I would want it to be equal to 12, since I have updated my x variable.

**IMPORTANT:** if you re-run cells, you’ll also need to re-run all downstream cells to update the output. If you don’t, you run the risk of having incorrect results, since some cells will be run out of order.

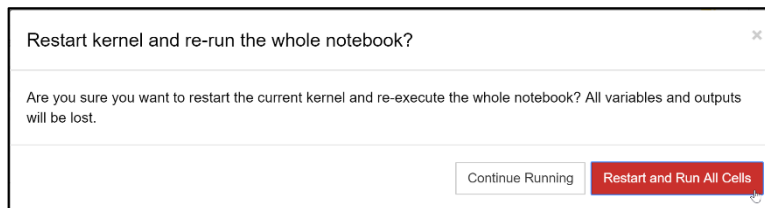
When I re-run all cells, now my cells are numbered correctly and the output correctly reflects my updated value for x *and* y.

```
In [1]: x = 11
In [2]: x
Out[2]: 11
In [3]: y = x + 1
In [4]: y
Out[4]: 12
```

The safest way to deal with this issue is to always re-run cells using the “Restart Kernel & Run All” option. This will ensure that all your cells are run in sequential order.



You will get a warning that looks like this:

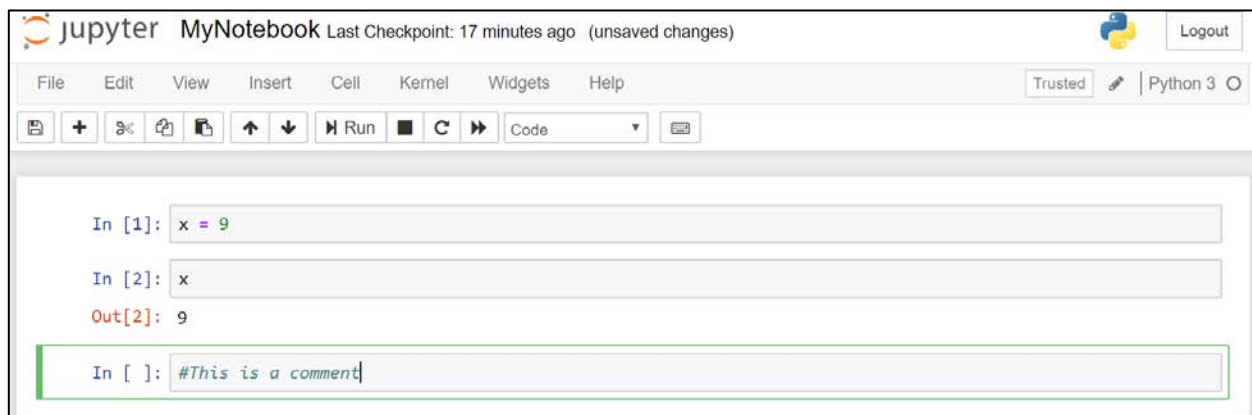


Don't panic! This is fine. You do want to restart the current kernel and re-execute the whole notebook. Press the red button to restart and run all cells.

## Part 4: Adding comments and text

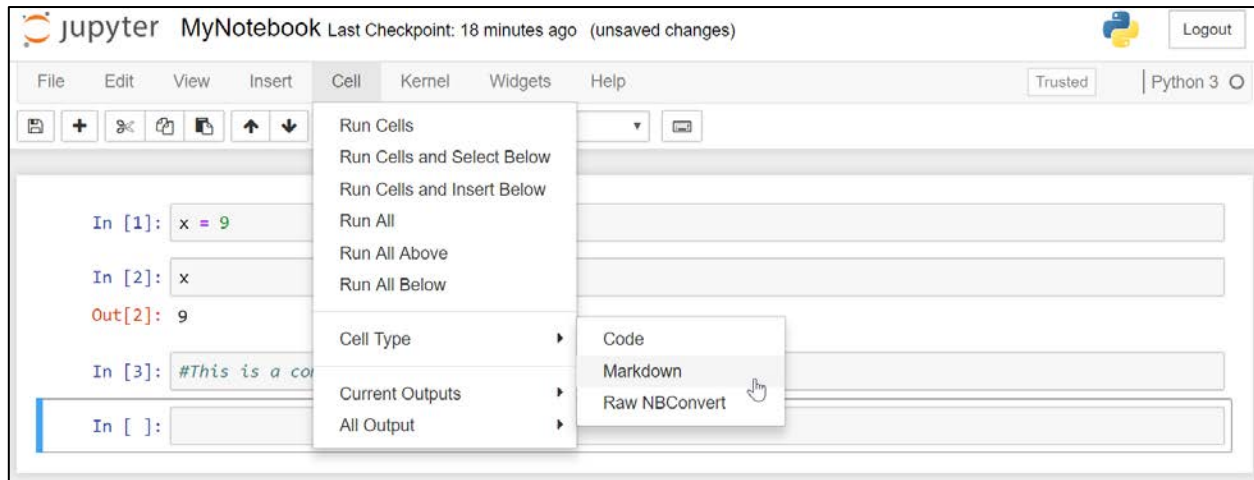
Commenting your code is important, so Jupyter supports two ways to write text within notebooks.

In the first way, you can use a hashtag (#) to add comments, just as you would in any Python script.

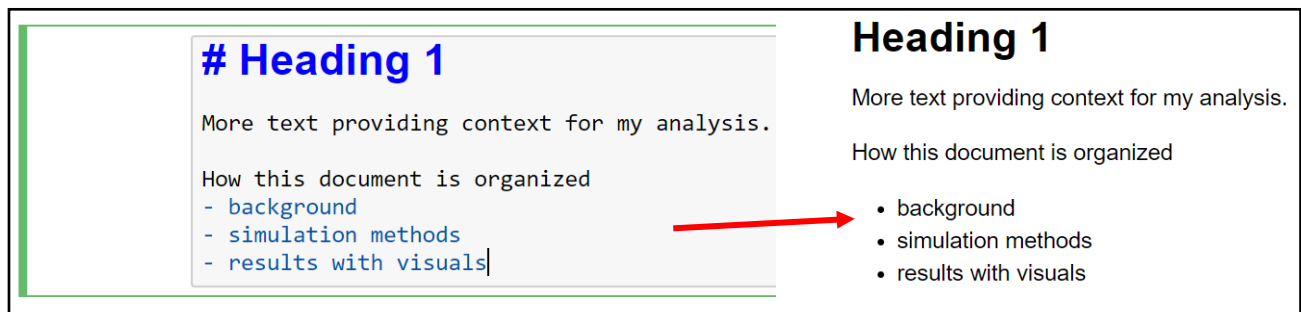




Alternatively, you can change the cell type to the Markdown language for additional flexibility in text additions.



Markdown is a way to add headings, lists, bold and italic text, and more. Here is an example of how a Markdown cell looks in “raw” form and how it looks once the cell is run.



Supporting Markdown cells is a strength of Jupyter notebooks – Markdown allows you to more fully annotate and explain not only your code, but also your overarching project, which provides valuable context for your code.

An extensive overview of Markdown is beyond the scope of this tutorial, but you can check out these sites for more information about how to use Markdown:

<https://github.com/adam-p/markdown-here/wiki/Markdown-Here-Cheatsheet>

<https://medium.com/ibm-data-science-experience/markdown-for-jupyter-notebooks-cheatsheet-386c05aeebed>