**Week1\_1 Programming Platform Set-up**

**General Notes**

This is the first lab in a series of Python tutorials for the Advanced GIS course. The tutorial series is intended to teach basic Python skills for quantitative spatial analysis under the context of urban planning and urban data science. We will begin with the basic Python syntax and data structure, followed by the instruction of data cleaning and management, geospatial data processing and visualization, as well as OSM APIs and web-scraping technique (optional) for data collection. Various Python packages will be covered, including but not limited to Pandas, Geopandas, Matplotlib, OSMNx, and Selenium (Optional).

No specific Python textbooks are required for these Labs; however, there are numerous resources for learning Python online, and one good book you may have is *Python Data Science Handbook* by Jake VanderPlas, which has been posted on Canvas. It is important to frequently google the answer for error reports and the usage of functions when you get stuck. Each Python package we use has very detailed user guide that documents the code examples and the usage of functions (for example, [Pandas](https://pandas.pydata.org/docs/user_guide/index.html#user-guide) and [Geopandas](https://geopandas.org/)). Browse the user guide as often as possible when debugging. In addition, the [Python Forum](https://stackoverflow.com/questions/tagged/python) (e.g., Stack Overflow) can be particularly useful to solve your specific coding question.

My advice for Python beginners is to experiment with small changes in existing code, read and google errors closely, and get familiar with online help forums. In this course, all the codebooks will be uploaded on Canvas before class. However, as a Python beginner, you should follow the instructor in the class demo and type in the code by yourself. Please DO NOT copy and paste the code directly because it does harm to your understanding of Python.

We will use Jupyter Notebook to help us code in Python. Homework will be submitted using Jupyter Notebook and the resulting .html file. The Jupyter Notebook files (.ipynb) are used to write and integrate text with code (i.e., literate programming), whereas the .html file will present the final analysis. You will be asked to **submit both the .ipynb file and the .html file**. Name your homework **NETID\_lab#.** If you do not follow this naming convention, you will be asked to resubmit.

**The goals for today’s lab are as follows:**

1. Install Python and Jupyter Notebook on your computer

2. Get familiar with the keyboard shortcuts of Jupyter Notebook and successfully run the basic code and narrative text on it.

**1. Installation of Python and Jupyter Notebook**

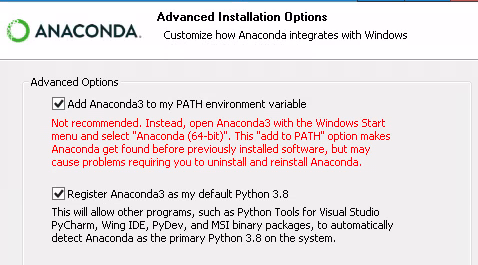
**1.1 Installation**

We will follow instructions of a Cornell Computer Science course (CS1113: Short Course in Python) to install Python and an interactive coding application called Jupyter Notebook using the Anaconda Distribution. The reason for using Anaconda is that it provides a standardized channel of establishing Python environment and contains lots of add-ons (third-party packages), limiting the number of things you need to install later.

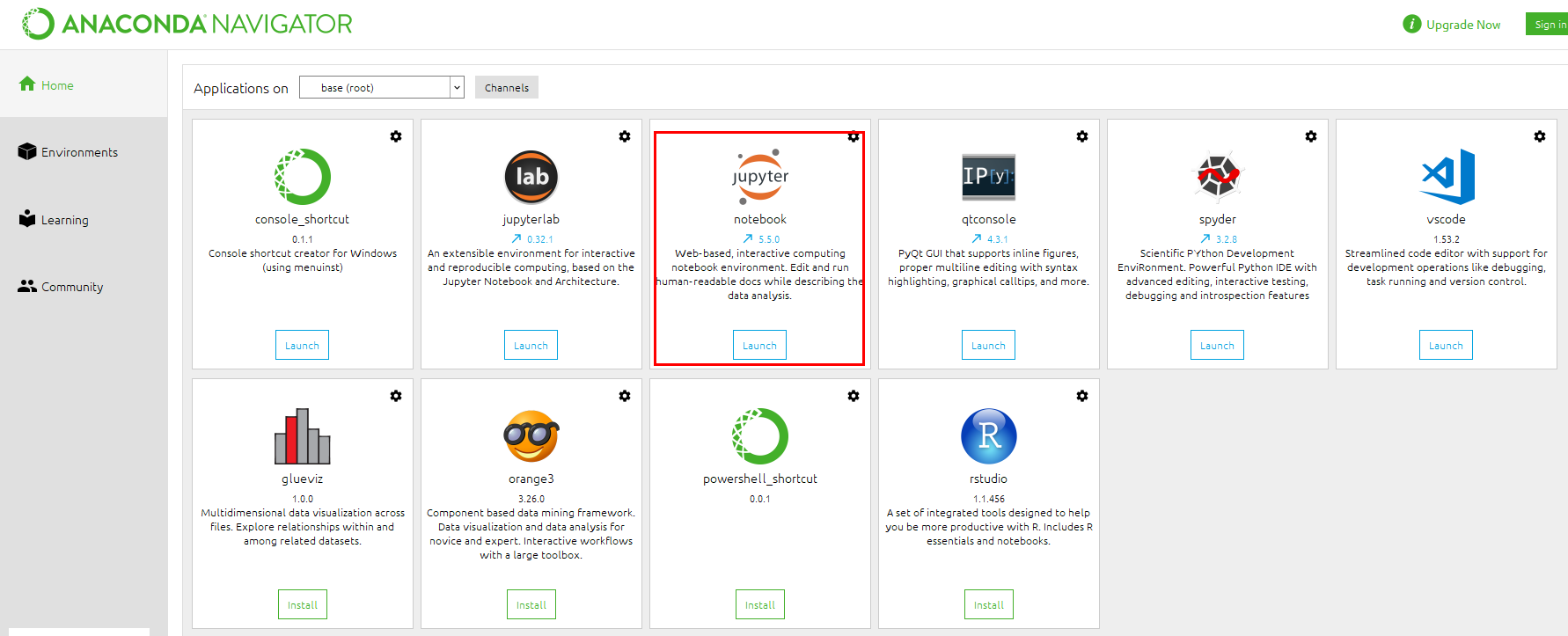
You can find the instruction for installation through this link [https://www.cs.cornell.edu/courses/cs1133/2018fa/materials/python.php](https://www.cs.cornell.edu/courses/cs1133/2018fa/materials/python.phpT). There are step-by-step instructions for installing Anaconda on the Windows and Macintosh (MacOS). Follow the instructions and install Anaconda version of Python on your computer. If the green button in the above link to download Anaconda 3.6 is invalid, right click the green button and choose “Open link in new tab,” then your web browser should begin to download the Anaconda 3.6. You DO NOT need to install the Cornell Python extensions listed in the link because it is customized for that course only.

**Please Note:**

1. You may notice that the mostly recent Anaconda version of Python is 3.8 rather than the 3.6. However, I highly recommend you install a lower version of Python because the Python 3.8 still have incompatibility issues with some packages. The Python 3.6 (Anaconda 3-5.2.0) has been tested to successfully install all the required packages we need for this class.
2. In the installation, you **MUST** check the box “Add Anaconda3 to my PATH environment variable” even it is not recommended.



After installing Anaconda 3, search Anaconda 3 in the Windows search toolbar and open the Anaconda Navigator. Make sure you have the Jupyter Notebook installed.



**1.2 Run Python in Anaconda Prompt**

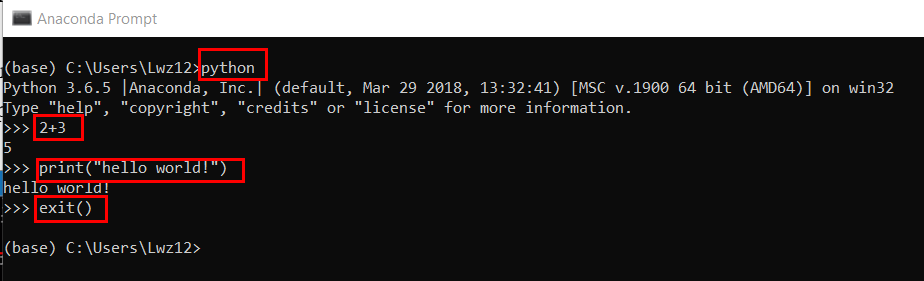
The following steps help you test whether Python has been properly installed in your computer.

1. Open "Anaconda prompt" (search it in Windows/MacOS toolbar).

2. Type in "python", and then press enter. If you see something like the figure below, congratulations, you made it!

3. Now you can talk to Python! Try 2+3 or print ("Hello World!")

4. Quit Python for now by typing in "exit ()" and pressing enter.

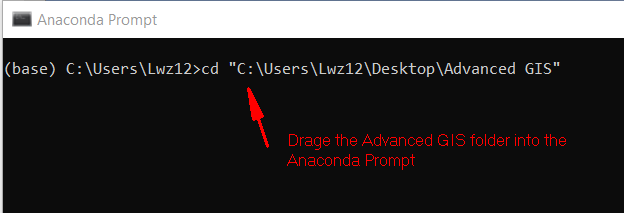


**1.3 Run Python in Jupyter Notebook**

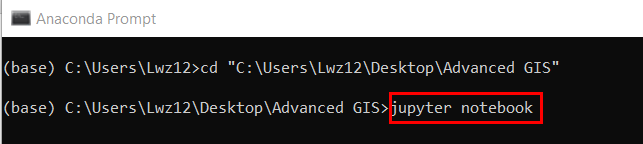
Jupyter Notebook is an interactive coding application for Python. It is a platform where you can run and edit code, display output, add explanation and charts that make your work more understandable and sharable. Let us start the Jupyter Notebook now!

1. Open again the "Anaconda prompt" in Windows/MacOS. The original path folder is something like: C:\Users\[UserName].

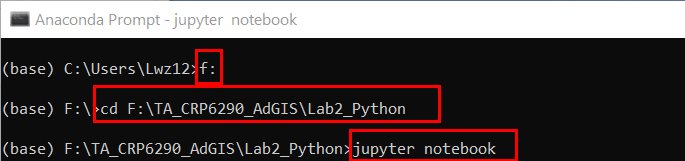
2. You may want to change the path to the directory (file folder) where your data is saved. Let us create and save the new Jupyter Notebook file (.ipynb) in the Advanced GIS course folder. For each time you creating a new Jupyter Notebook, a new .ipynb file will be generated in the corresponding folder. In the Anaconda prompt, type “cd” (means change directory) and **leave a blank**. Then **drag** the Advanced GIS folder into the Anaconda prompt and press enter. The folder path will be generated automatically. MacOS system should follow the same steps.



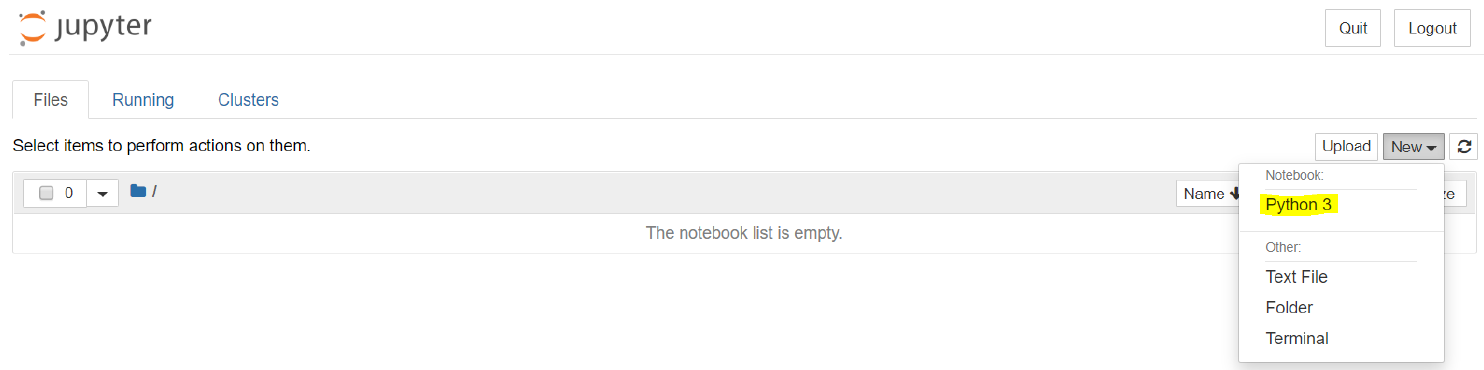
3. In the next line, type in “jupyter notebook” and press enter.



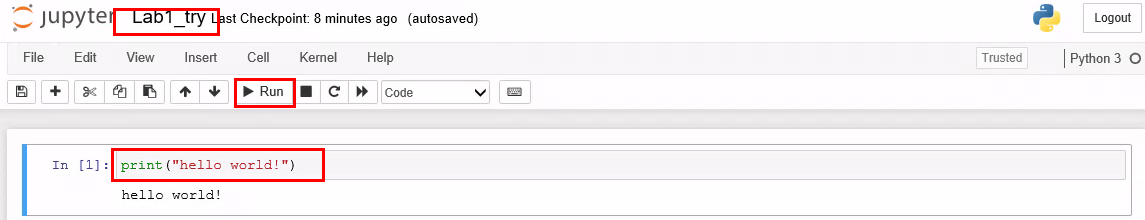
Note that if you create the Advanced GIS folder in other disks rather than C. You need to add an additional step to change the working directory to the disk that save your folder. For example, I create the folder in disk F and I need to firstly type in “F:” and press enter to change the directory.



4. A Jupyter notebook will open in your default browser. This is the Notebook Dashboard, specifically designed for managing your Jupyter Notebooks. Think of it as the launchpad for exploring, editing and creating your notebooks. The dashboard will show nothing if you just create the file folder without anything stored in it. To create a new Notebook, click "New" on the upper-right corner, and choose "Python 3". You will be navigated to a new web page, where we run Python code.



5. First, check out the menus to get a feel for it. Next, pick an appropriate name for your first Python Jupyter Notebook by clicking the “Untitled” on top of the page. In the first code chunk, type in *print ("hello world!")* and click “run”. If there is no error occurs, then congratulations! You have successfully written some code.



6. Click the “Save” icon on the top left, then close the Jupyter Notebook, and the corresponding command prompt. You should find a new Jupyter Notebook file (.ipynb) with the customized name in your Advanced GIS file folder. Next time when you open this notebook, follow the steps above, but in the step 4, select the notebook you want to open in the list instead of creating a new notebook.

Please keep in mind that you should not **close the Anaconda prompt when you are still working on the Jupyter Notebook**, otherwise the notebook kernel will shut down, and you have to open Anaconda Prompt again and launch a new Jupyter Notebook to resume your work.

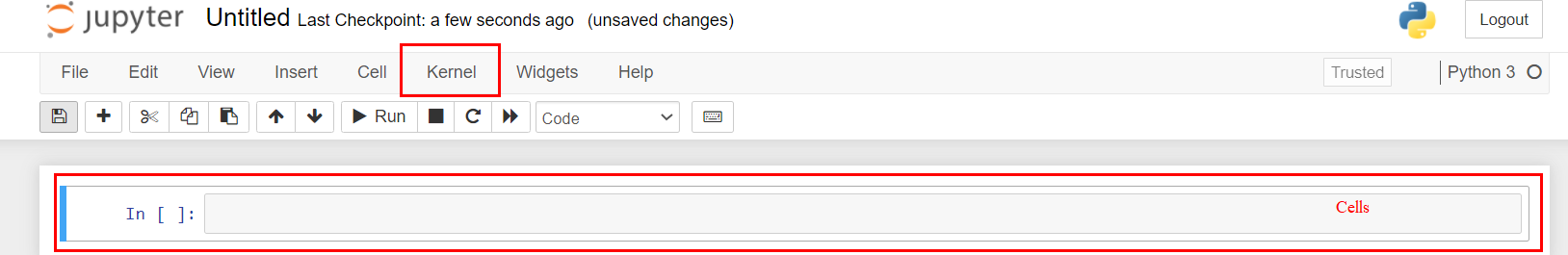


**2. The Basics of Jupyter Notebook**

### 2.1 The Notebook Interface

Once you finished the first session, you are able to create a new Jupyter Notebook and are ready to code. There are two terms that you should notice, which are probably new to you: *cells* and *kernels*.

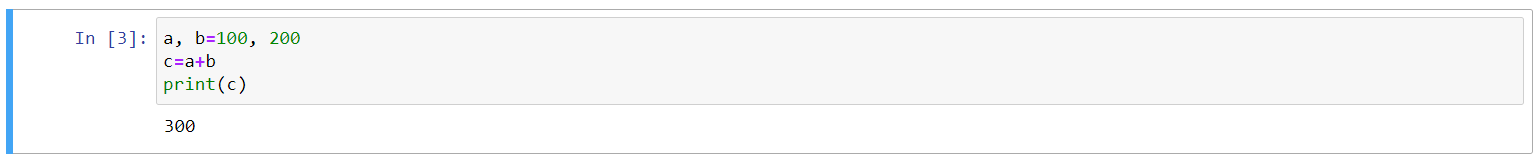
* A kernel is a “computational engine” that executes the code contained in a notebook document. Note that when you close the Anaconda prompt which was used to launch the Jupyter Notebook, the kernel for that Jupyter Notebook will also shut down, meaning that you cannot run, edit or the save the code anymore. Therefore, **do not close the corresponding Anaconda prompt when coding.**
* A cell is a container for text to be displayed in the notebook or code to be executed by the notebook’s kernel.



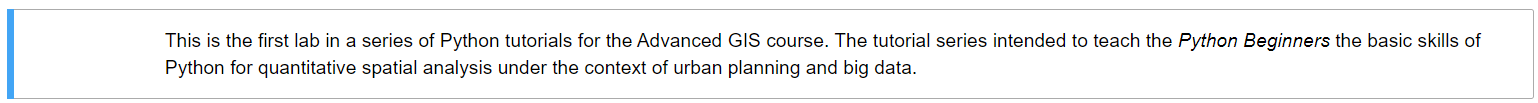
*Figure 2.1: Cell (code chunk) and kernel*

There are two main cell types—Code cell and Markdown cell.

* A **code cell** contains code to be executed in the kernel. When the code is run, the notebook displays the output below the code cell that generated it.
* A **Markdown cell** contains text formatted using [Markdown](https://www.markdownguide.org/basic-syntax/) and displays its output in-place when the Markdown cell is run.



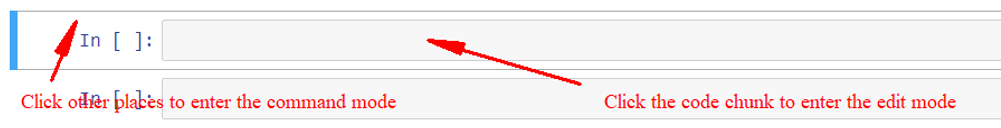
*Figure 2.2: The code cell: for creating and editing the code*



*Figure 2.3: The Markdown cell: for description and narrative text*

Remember that you will always begin with a code cell. Try to type the code shown in Figure 2.2 in your first cell and click  in the toolbar or use the keyboard shortcut “Ctrl + Enter” to run the cell.

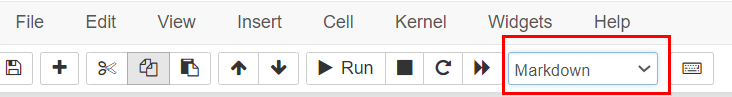
When you click the code chunk and type in the code, the highlighted cell enters the edit mode and displays **green cell border**, indicating you enter the edit mode and are ready for coding. The **edit mode** allows you to type code or text into a cell. The cell will return to **command mode** that displays blue border after running code. You can easily toggle between the edit and command mode by clicking the code chunk (for edit mode) and an empty place (for command mode). Different modes also correspond to different keyboard shortcuts. We will discuss it in the next session.



*Figure 2.4: Toggle between the edit and command mode*

Now keep your first cell in **border blue** and press “B” in the keyboard, you will get a new code cell just below the current cell. Try to press “B” multiple times to add more cells.

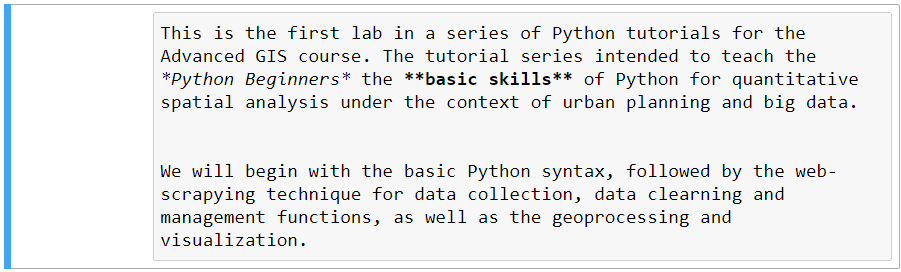
Change one Code Cell to Markdown Cell by selecting the “Markdown” option in the dropdown menu of the toolbar (Figure 2.5). Keep your cell in the edit mode and write down whatever you want. Click  or “Ctrl + Enter”, and you will get the cell displaying the rendered narrative text, just like what is shown in Figure 2.3.



*Figure 2.5: transform the Code cell to Markdown cell*

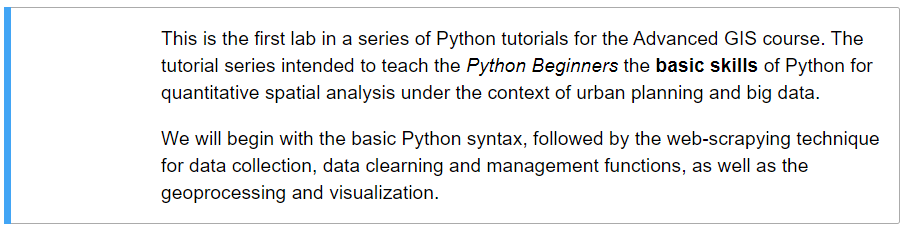
Here are some examples of the Markdown text before and after rendering (after running the cell). You can find the Markdown text after rendering presents much better looking, so please use the Markdown text properly to edit your narrative part when submitting your homework next time.

Original text: These are the original text that you type in the cell.



*Figure 2.6: the narrative text in the Markdown cell*

The rendered markdown shows up as below. Pay attention to the usage of \*\* to bold words and \* to italicize words.



*Figure 2.7: the rendered narrative text in the Markdown cell after running the cell*

Original text:



*Figure 2.8: the narrative text in the Markdown cell*

The rendered markdown shows up as below. Note that the -, \* and + all denote the bullet sign after rendered. The <br> breaks the current line and resumes the remaining contents in a new line.



*Figure 2.9: the rendered narrative text in the Markdown cell after running the cell*

For more functions regarding the Jupyter Notebook Markdown, please refer to: <https://www.youtube.com/watch?v=uVLzL5E-YBM>

**2.2 Keyboard Shortcuts**

Keyboard shortcuts are very popular in the Jupyter Notebook because they make your coding process easier and more convenient. Remember that each cell can be in the edit mode when you are typing code in it, or else in the command model.

**Green cell outline**— cell is in "edit mode"

**Blue cell outline** — cell is in "command mode"

The corresponding keyboard shortcuts are also different when the cell is in different modes. Here is the summary of the shortcuts that are often used:

* Toggle between edit and command mode with **Esc** and **Enter**, respectively.
* Run a cell: "**Shift + Enter**" (a new cell will be created below and becomes active); "**Ctrl + Enter**" (no new cell will be created)
* Once in **edit mode**:

Ctrl + c: copy the selected code

Ctrl + v: paste the selected code

Ctrl + z: undo the change.

* Once in the **command mode**:

B: add a cell below the highlighted cell

A: add a cell above the highlighted cell

double click "D": Delete the cell.

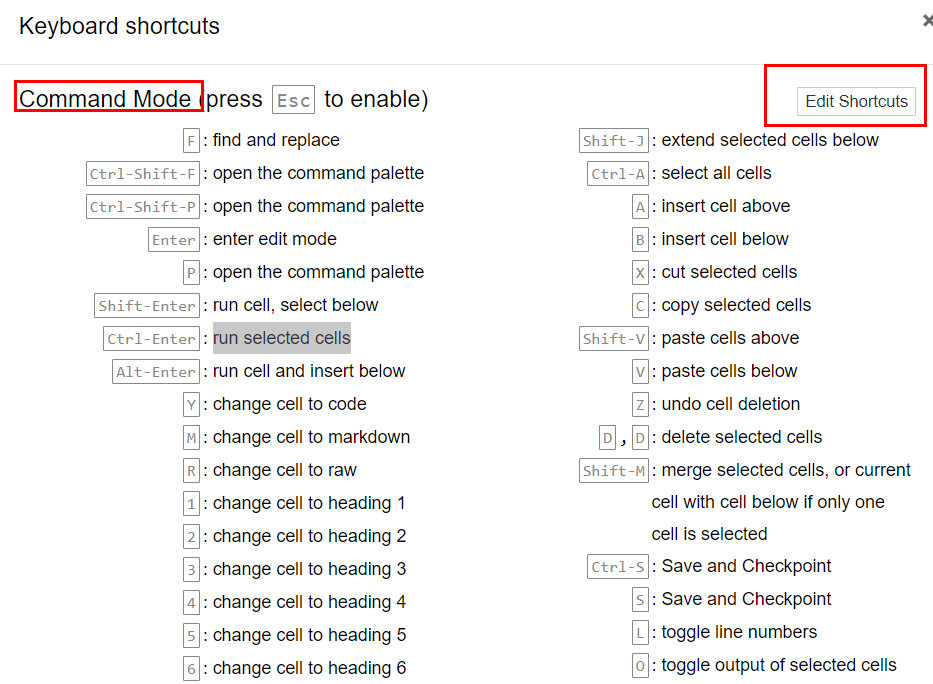
Y: change to code cell

M: change to markdown cell

holding "Shift": Select multiple cells

"Shift + m": Merge multiple cells after selecting all cells.

If you are not satisfied with the shortcuts listed above and want to explore more, click “help” in the main menu, and click “Keyboard Shortcuts”, a keyboard shortcut tab will show you all the default shortcuts in both the command and edit modes. You can also customize your own shortcuts by clicking “Edit Shortcuts”.



*Figure 2.10: Shortcut panel in the Jupyter Notebook*