Getting Started with iPython/Jupyter Notebooks

Jupyter Notebooks are web-based interactive documents that combine programming and rich text formatting into one - imagine a coding environment mixed with a simplified word document.

There are two main ways that we will access and use Jupyter Notebooks in this class:

1. Use online tools like Google Colaboratory
2. Download software to your own machine

These other options are discussed more in other documents. For now we will discuss the main features that are general to **all** Jupyter Notebooks.

Working in the Notebook

A *quick* note: the images included in this document are from Anaconda's distribution of Jupyter Notebook. They will look slightly different than what you see on Google Colab, but the functionality is all the same.

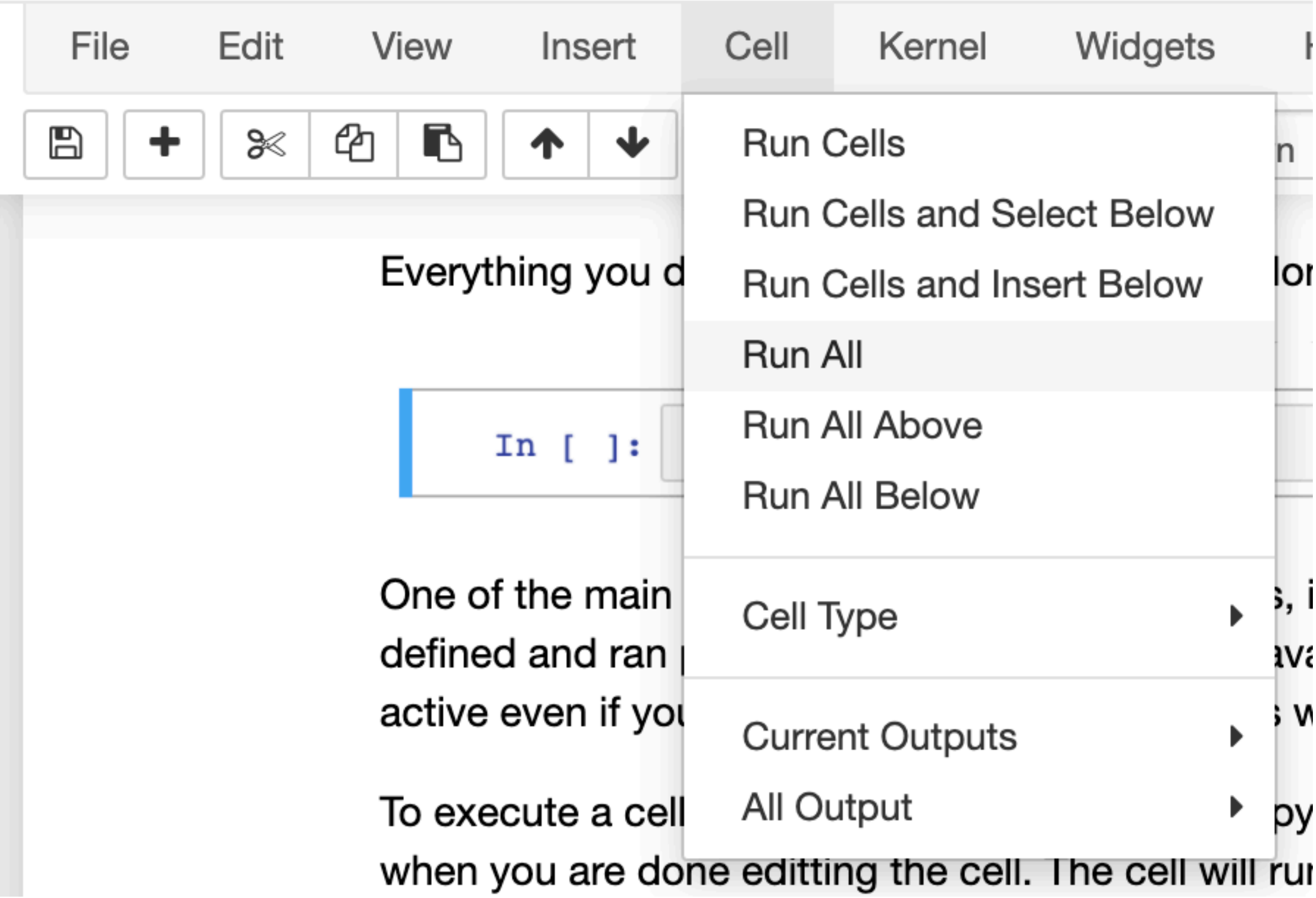
Everything you do in Jupyter Notebooks will be done in what are known as "cells":

```
In [ ]:
```

One of the main advantages of working with cells, is that you can run snippets of your code by only executing cells you want to. However, the variables you defined and ran previously in other cells are still available in your workspace. In other terms, once you run a cell, everything you defined in that cell will remain active even if you never run that cell again versus when you run a typical script and everything is executed from start to finish.

To execute a cell, you can either press:

- **Run** in Jupyter Notebook

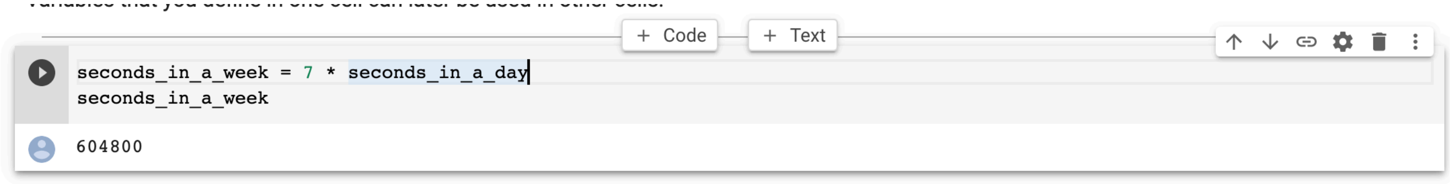


Everything you d

One of the main
defined and ran
active even if you

To execute a cell
when you are done editting the cell. The cell will rur

- hit the **arrow button** in Colab



```
seconds_in_a_week = 7 * seconds_in_a_day
seconds_in_a_week
```

```
604800
```

- even easier you simply press **Shift+Enter** (Mac or PC) in either software when you are done editting the cell. The cell will run (provided you have no errors) and then highlight the next cell.

Cells have two main flavors that must be specified to run them correctly:

1. Code cell
2. Markdown cell

Coding Cell

Coding cells are what they sound like - cells in which you enter Python syntax and run them like you would a normal script. We won't go over Python itself in this document, but you can see a few examples below. Most everything you do in a normal script, you can do here like defining variables, running standard structures like for-loops, importing libraries, and working with classes and functions.

```
x = 1
y = 2
print(x+y)
```

```
3
```

```
for i in range(5):
    if i%2 == 0:
        print(i*i)
    else:
        print(i)
```

```
0
1
4
3
16
```

```
import numpy as np
z = np.zeros(6)
print(z)
```

```
[0. 0. 0. 0. 0. 0.]
```

```
class classy:

    def __init__(self,name):
        self.name = name

    def get_name(self):
        return self.name

    def __str__(self):
        return f'{self.name} is classy'


prof = classy('Zoltan')
print(prof)
```

```
Zoltan is classy
```

You can run multiple cells in succession by choosing the option to **Run all** or by pressing **Shift+Enter** multiple times for different cells. However, cells will run in the order you specify and will not proceed if you encounter an error along the way.

Coding cells will contain a number in brackets that corresponds to when they were run where the highest number refers to the cell most recently run i.e. `[1]` refers to the first cell run.

Markdown Cell

Markdown is a simple markup style language that can be used in cells instead of Python. There a few commands that can be used in a Markdown cell to produce headings, format text, and create lists to name a few. More tips on how to use Markdown in Jupyter Notebooks can be found here.

Markdown cells also support limited use of HTML. If you are comfortable with this markup language, you should be able to seemlessly integrate it into your notebooks for a more polished look.

Further Reading

This document only touches on the basics of Jupyter notebook, but also provides links to other resources to improve your ability to comfortably and efficiently use these types of documents. Below are a few resources to help you with multiple components of Jupyter Notebooks.

- Basic Guide to Jupyter Notebooks
- More Extensive Guide to Jupyter Notebooks
- Markdown Language Tips
- Python Help