

Today's Topics:

- Numpy

Lee wilkins

Office : 6C.9

Contact: MIO

INTRODUCTION TO COMPUTER PROGRAMMING IN ENGINEERING AND SCIENCE

Comp Sci assignments (All tests and assignments occur Wednesdays in our lab block)

2

Test 1 week 7 (15%) Wednesday March 5)

Assignment 2 week 8 (10%)

Assignment 3 week 11 (10%)

Test 2 week 13 (15%)

Physics assignments

Assignments (4 x 2%) 8% Date communicated by the Physics teacher

Project 1: Solving differential equations 10% Week 11

Project 2: Applying programming in science 22% Week 15

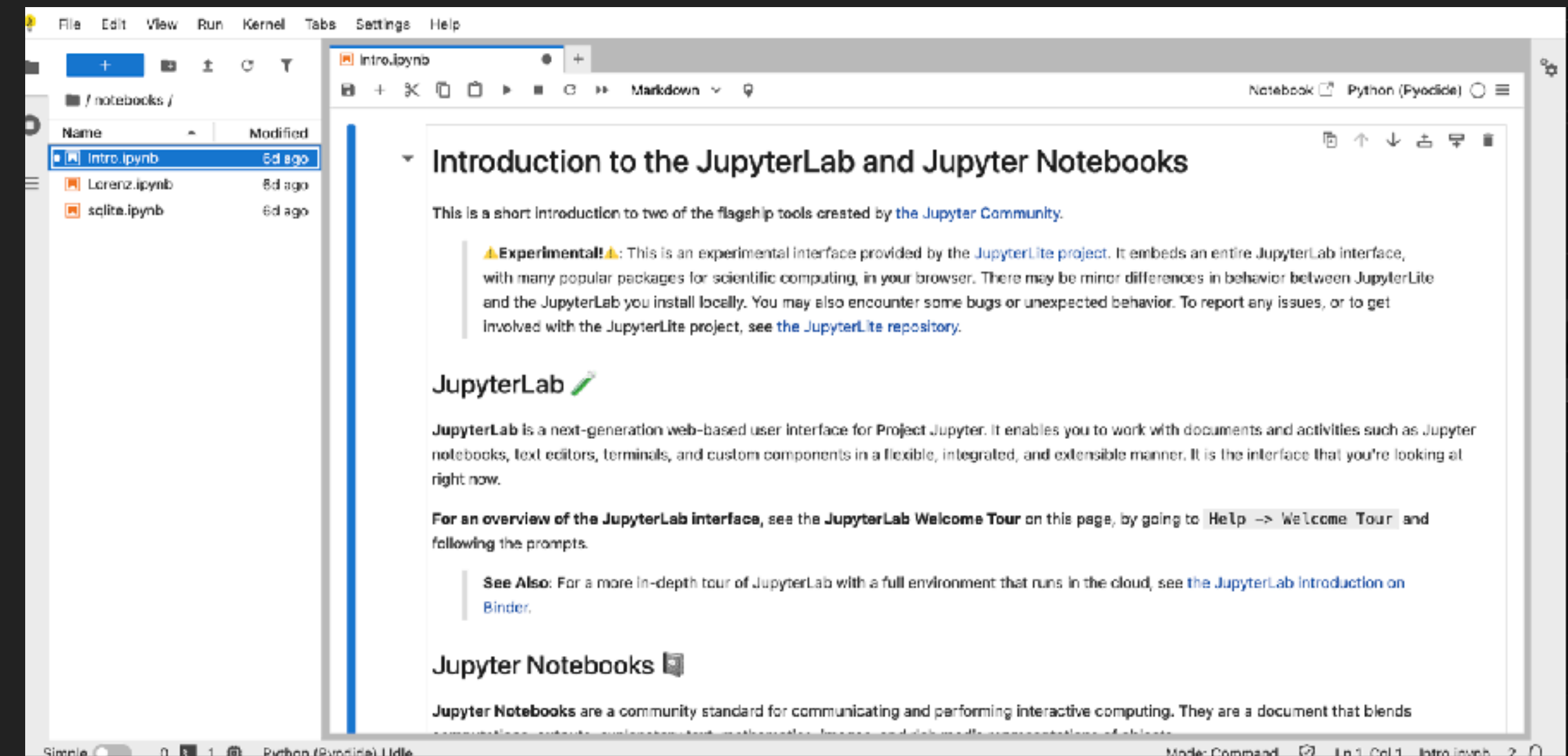
GRADE BREAKDOWN REVIEW

<https://jupyter.org/try-jupyter/lab/>

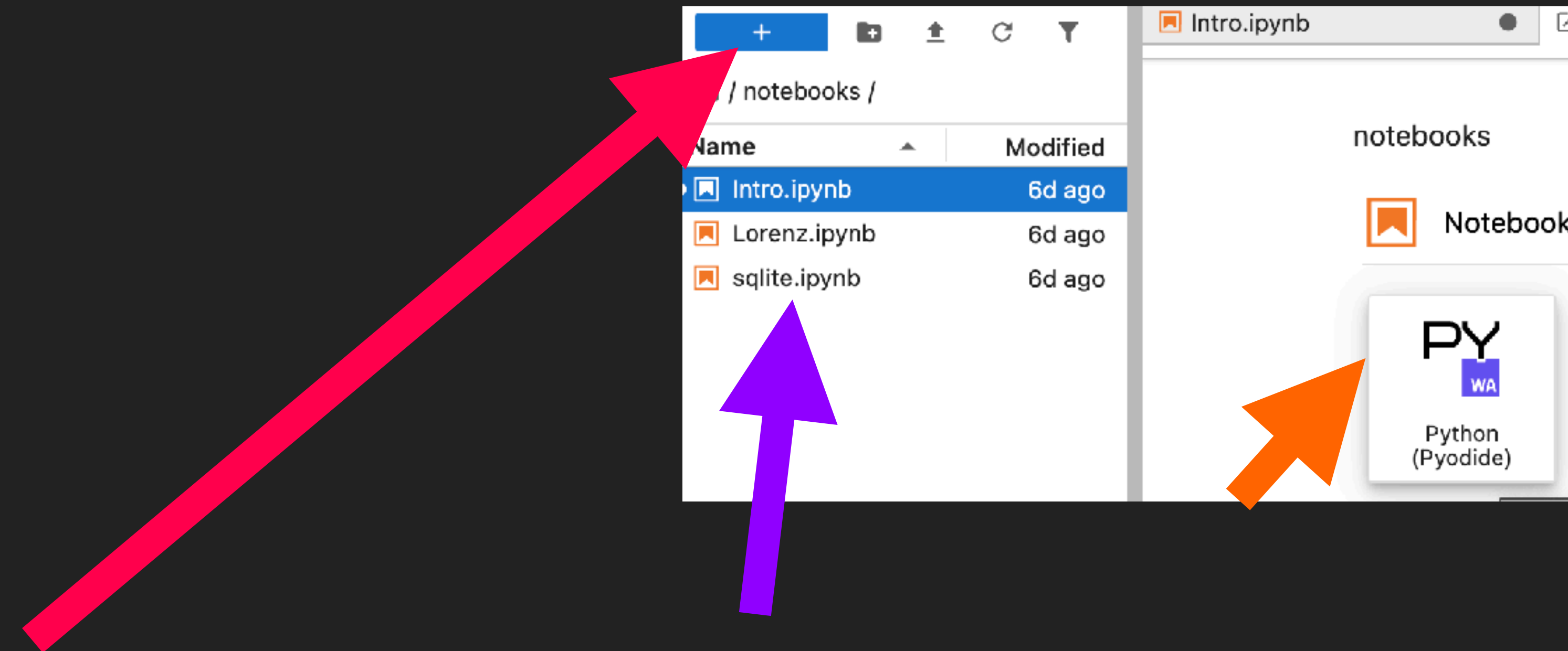
<https://jupyterhub.dawsoncollege.qc.ca>

<https://colab.research.google.com/> if you have a google account

Jupyter notebook is a way of storing code, making visuals, and writing text in the same place on a readable page. Its great for readability when your user can interact with code online.



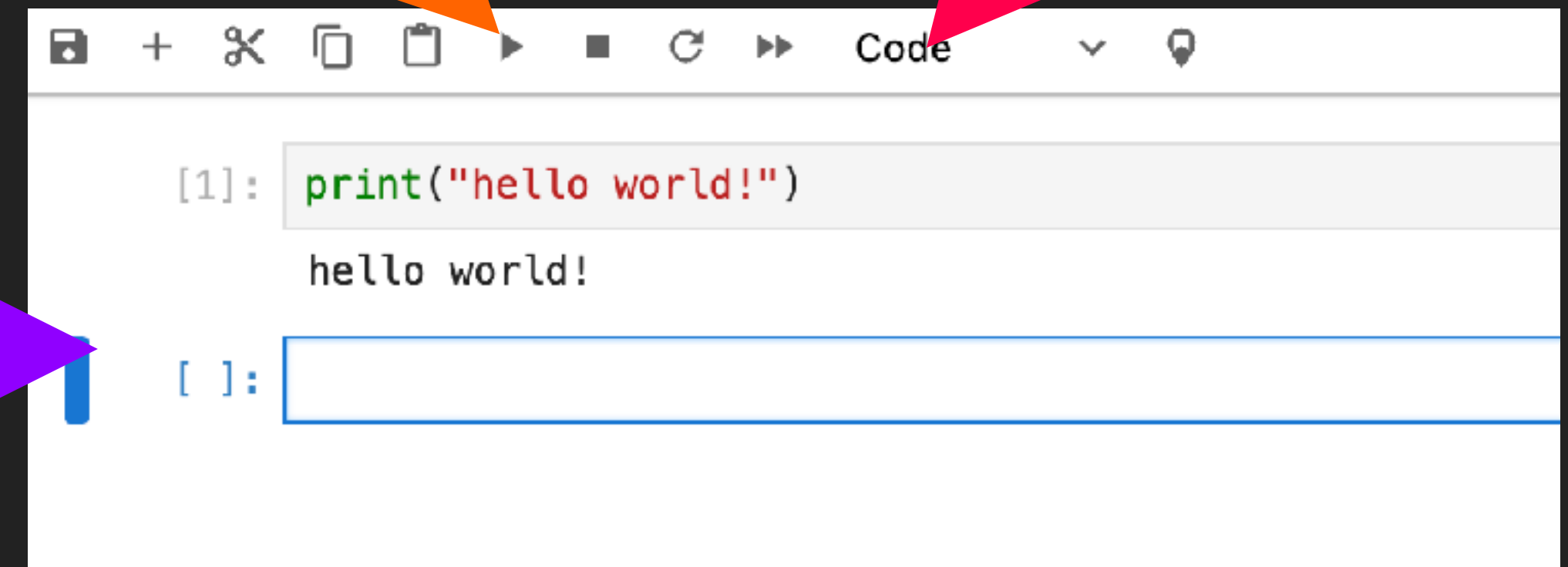
JUPYTER NOTEBOOK



You can **create new** files that are **stored in your account as .ipynb**. These files are specific to Jupyter notebook. Make sure to **create a python pyodide file**.

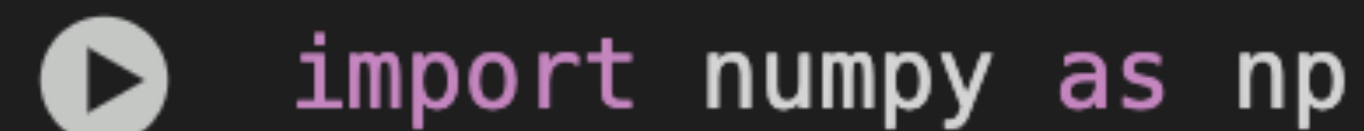
JUPYTER NOTEBOOK

Jupyter notebook uses “cells” to write code, text, or markdown. Each cell can contain something different. Make your cell “code” and then **run** the code to see the output. Remember, you have to run import lines to make sure the module is imported.



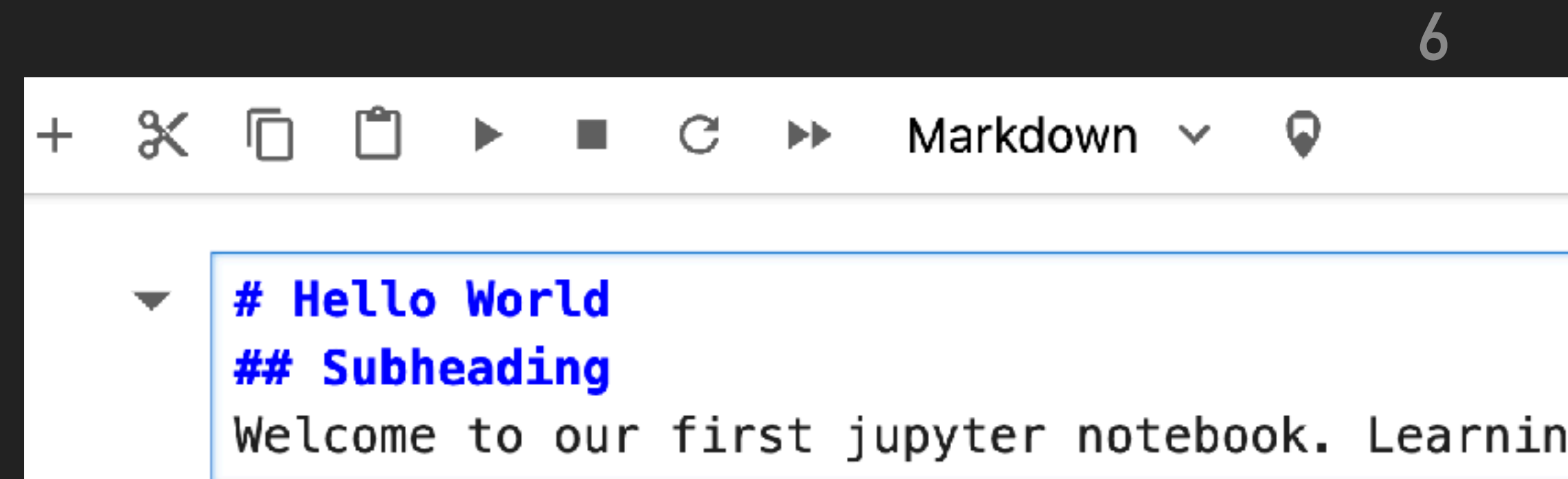
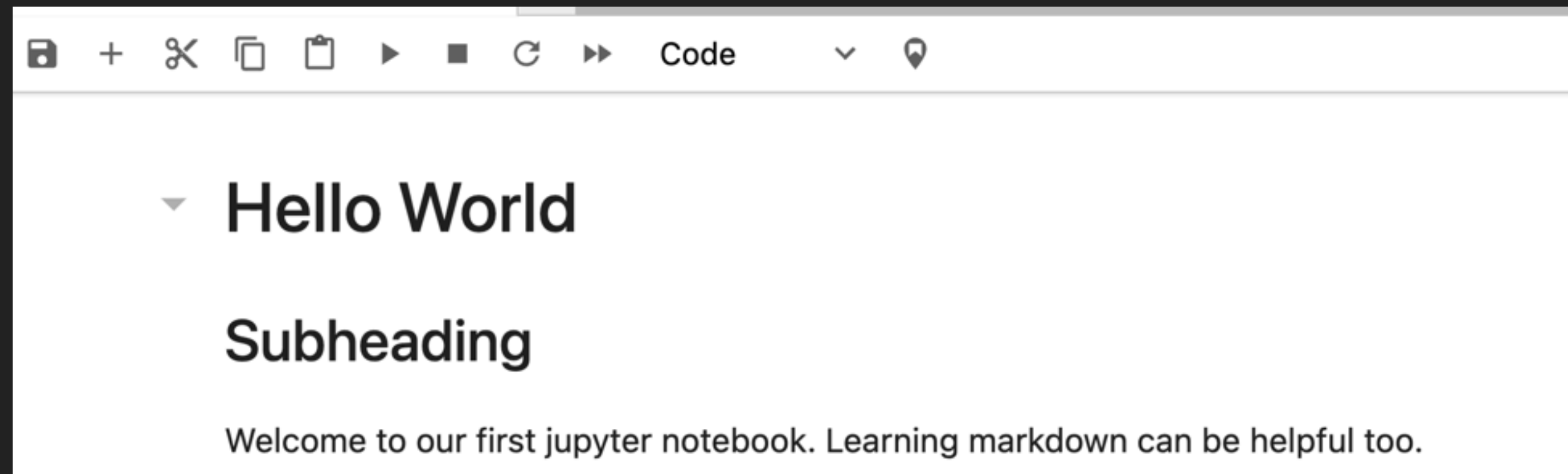
```
[1]: print("hello world!")  
hello world!  
[ ]:
```

Import Numpy. We import it as np so



```
import numpy as np
```

JUPYTER NOTEBOOK



Markdown is a way of writing and formatting text that is common for code documentation. Here is a sheet of markdown notations

<https://jupyter-notebook.readthedocs.io/en/stable/examples/Notebook/Working%20With%20Markdown%20Cells.html>

Markdown is helpful for making your document more readable

MARKDOWN

Matplotlib is a python module used to create plots and graphs.

<https://matplotlib.org/stable/>

Numpy is a module used to work with arrays and lists of data.

<https://numpy.org/>

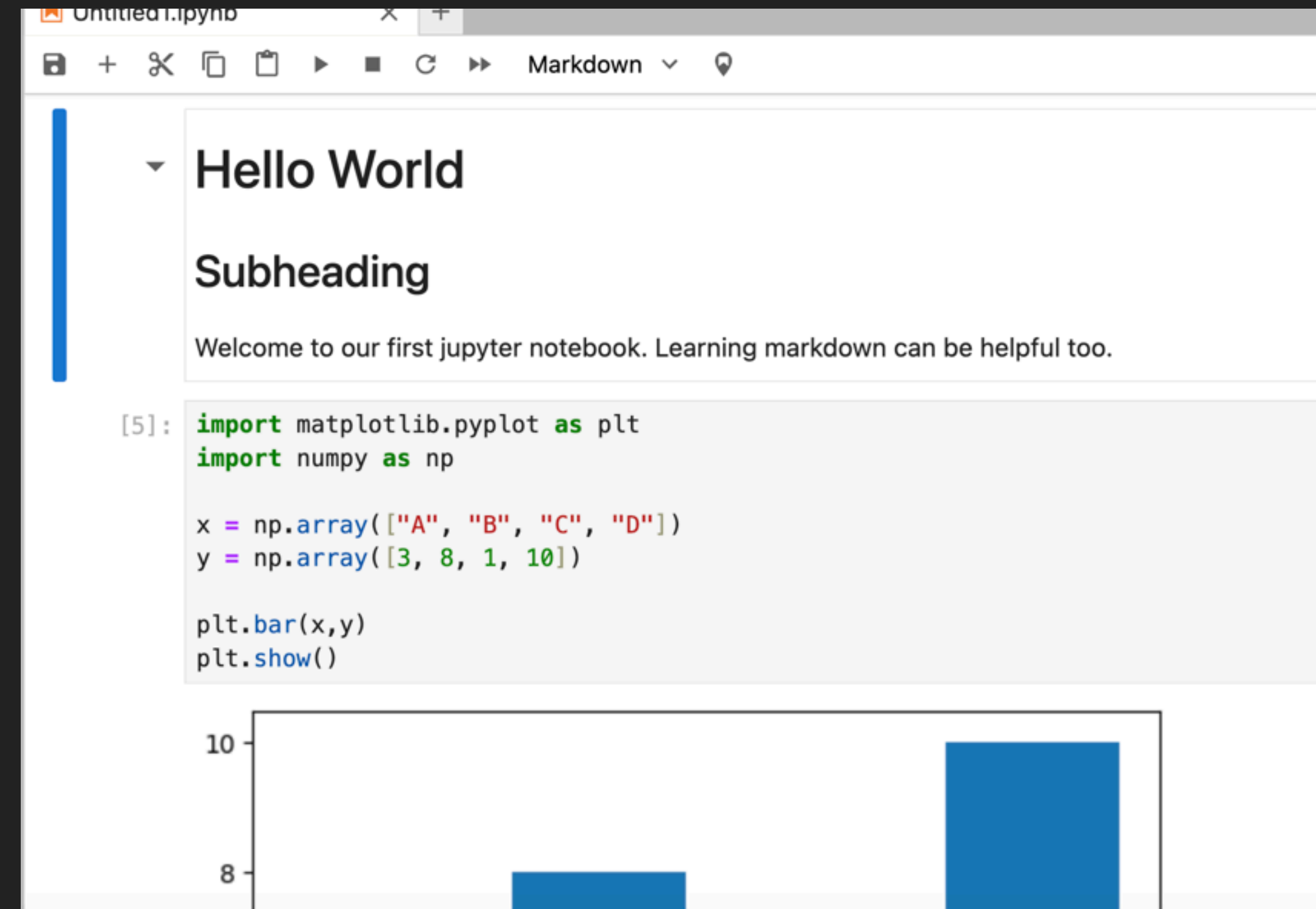
By using them in jupyter notebook, we can create complex visualizations. You can install this modules on your own computer as well, follow instructions on the page. They are already installed on jupyter hub.

MATPLOTLIB AND NUMPY

Numpy makes arrays or lists easier to work with for the purpose of data.

Later, we will incorporate numpy with pandas and matplotlib to make more complex visualizations

NUMPY ARRAYS



1D array

2	7	1	5
---	---	---	---

0 1 2 3

2D array

2	7	1	5
---	---	---	---

0 0 0 1 0 2 0 3

2	7	1	5
---	---	---	---

1 0 1 1 1 2 1 3

```
my_list = [2, 7, 1, 5]  
my_list[0]
```

```
my_list = [[2, 7, 1, 5],  
           [2, 7, 1, 5]]  
my_list[0][1]
```

DIMENSIONAL ARRAYS / LISTS

3D array

0	2	7	1	5
	0 0 0	0 0 1	0 0 2	0 0 3
1	2	7	1	5
	0 1 0	0 1 1	0 1 2	0 1 3

0	2	7	1	5
	1 0 0	1 0 1	1 0 2	1 0 3
1	2	7	1	5
	1 1 0	1 1 1	1 1 2	1 1 3

```
my_list = 0[[[2, 7, 1, 5], [2, 7, 1, 5]],  
            [[2, 7, 1, 5], [2, 7, 1, 5]]]
```

```
my_list[0][1][1]
```

1

DIMENSIONAL ARRAYS

```
import numpy as np
```

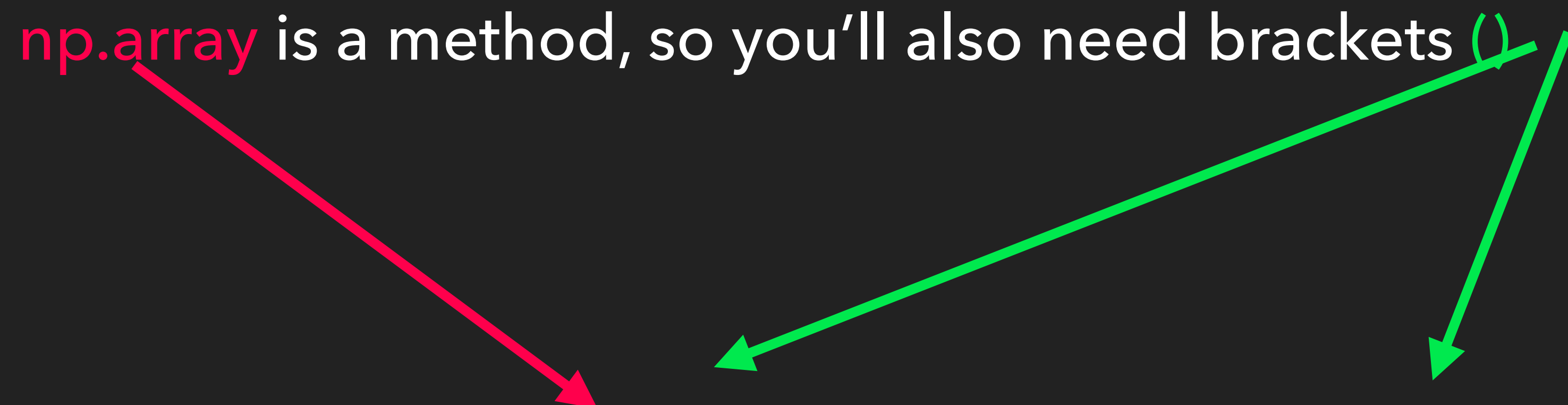
Add the numpy module. "As np" means we will refer to all numpy methods as np.

IMPORT

https://colab.research.google.com/drive/11RwnArf_74zF1gFh_bU7GIvpGmvnaUU#scrollTo=t6ma8eclJYrA In class

To get access to numpy, you need to create a numpy array.

`np.array` is a method, so you'll also need brackets `()`



```
numpy_array = np.array([2, 1, 5, 3, 7, 4, 6, 8])  
python_list = [2, 1, 5, 3, 7, 4, 6, 8]
```

NUMPY ARRAYS

Numpy arrays can perform operations like this, instead of using a for loop

```
arr = np.array([2, 1, 5, 3, 7, 4, 6, 8])
arr = arr*2
print(arr) # [ 4  2 10  6 14  8 12 16] Multiply the array by 2
arr = arr+10
print(arr) # [14 12 20 16 24 18 22 26] Add 10 to the array
brr = np.array([100, 123, 545, 310, 7, 43, 6, 8])
arr = arr+brr
print(arr)# [114 135 565 326 31  61  28  34] Add another array to the array
```

NUMPY ARRAYS

It is possible to generate a list using arange, similar to range. 0, 101 is the range (non inclusive) and 2 is the increment.

```
even_array = np.arange(0, 101, 2)
print(even_array)
odd_array = np.arange(1, 101, 2)
print(odd_array)
```

NUMPY ARANGE

Numpy arrays can be reshaped into any configuration using reshape. This can make a 1d array into an nD array depending on your usage

```
arr = np.array([2, 1, 5, 3, 7, 4, 6, 8])
```

```
arr = arr.reshape(2, 4)
```

```
print(arr)
```

```
[[1 2 3 4]
 [5 6 7 8]]
```

```
arr = arr.reshape(4, 2)
```

```
print(arr)
```

```
[[1 2]
 [3 4]
 [5 6]
 [7 8]]
```

```
arr = arr.reshape(1, 8)
```

```
print(arr)
```

```
[1 2 3 4 5 6 7 8]
```

NUMPY ARRAYS


```
-----  
ValueError                                Traceback (most recent call last)  
<ipython-input-42-82ae0f722f48> in <cell line: 0>()  
      6 arr = arr.reshape(2,2,2)  
      7 print(arr)  
----> 8 arr = arr.reshape(7, 2)  
      9 print(arr)  
     10  
ValueError: cannot reshape array of size 8 into shape (7,2)
```

Numpy arrays can be reshaped into any configuration using reshape, but the reshape must contain the right number of elements

NUMPY ARRAYS

You can use `.shape` to get the attribute of the shape of any array

```
arr = np.array([2, 1, 5, 3, 7, 4, 6, 8])  
print(arr.shape) # (8,)  
arr = arr.reshape(2, 2, 2)  
print(arr.shape) # (2, 2, 2)  
arr = arr.reshape(4, 2)  
print(arr.shape) # (4, 2)  
arr = arr.reshape(1, 8)  
print(arr.shape) # (1, 8)
```

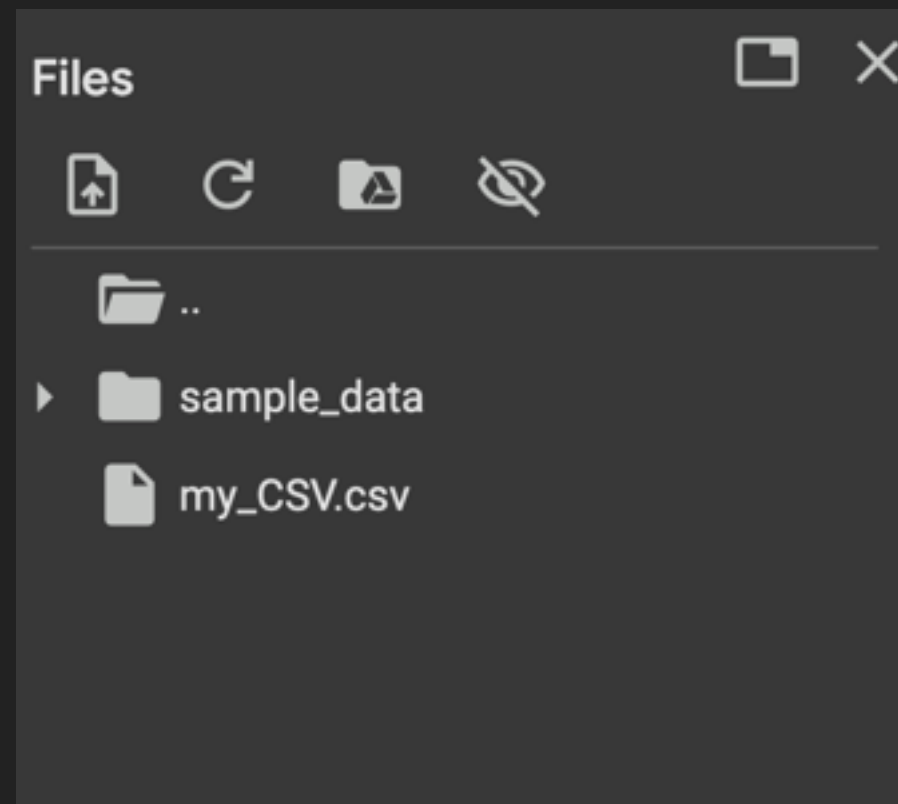
NUMPY ARRAYS

Numpy arrays are often used in data science. We can manipulate the shape of an array to make a visualization. Lets say we have a large data set we want to visualize, but the data isn't formatted.

Lets create a numpy array and save it into a .csv file. Start by creating two arrays. Numpy can create CSV files based on the shape of your array. In this example, we have a 2d array with 3 elements: Ones, Tens, and Hundreds. We can think of each element in the array as a set of data. Vertically we can think of them as columns, and horizontally as rows.

```
data_set_1 = np.array([[1, 10, 100],  
                       [2, 20, 200],  
                       [3, 30, 300]])
```

LOADING AND SAVING FILES



1.000000000000000000000000e+00	1.0000
2.000000000000000000000000e+00	2.0000
3.000000000000000000000000e+00	3.0000

Show 10 rows per page

`np.savetxt` lets us save a file. At minimum, this function needs the file to save as, the data set to save, and a delimiter. This is a common example.

`np.savetxt('my_CSV.csv', data_set_1, delimiter=',')`

<https://numpy.org/doc/2.1/reference/generated/numpy.savetxt.html>

LOADING AND SAVING FILES

1.000000000000000000e+00	1.0000
2.000000000000000000e+00	2.0000
3.000000000000000000e+00	3.0000



# Ones	Tens	Hundreds
1	10	100
2	20	200
3	30	300

You'll notice that the file is in scientific notation and has no headers

`fmt=` is important because it dictates the format. Without this, your file will appear in scientific notation. `fmt='%4d'` will limit numbers to 4 digits long.

Header allows us to place headers at the beginning of your file, separated by commas.

<https://numpy.org/doc/2.1/reference/generated/numpy.savetxt.html>

```
np.savetxt('my_CSV.csv', data_set_1, fmt='%4d', delimiter=',', header='0nes,
Tens, Hundreds')
```

LOADING AND SAVING FILES

```
np.savetxt('my_CSV.csv', data_set_1, fmt='%4d', delimiter=',', header='Ones, Tens, Hundreds')
```

```
data_set_1 = np.array([[1, 10, 100],  
                        [2, 20, 200],  
                        [3, 30, 300]])
```

# Ones	Tens	Hundreds
1	10	100
2	20	200
3	30	300

LOADING AND SAVING FILES

You can create CSVs out of arrays by arranging them properly using numpy. Here, we can convert a 1d array into the proper 2d array we need to create a CSV as described previously. **Reshape** changes the array into 3x3, and then **.t** (or **transpose**) reverses the rows and columns.

```
data_set_b = np.array([1,2,3,10,20,30,100,200,300])
```

```
data_set_b = data_set_b.reshape((3, 3))
```

```
print(data_set_b)
```

```
# [[ 1  2  3]
```

```
# [10 20 30]
```

```
# [100 200 300]]
```

```
data_set_b = data_set_b.T
```

```
print(data_set_b)
```

```
# [[ 1 10 100]
```

```
# [ 2 20 200]
```

```
# [ 3 30 300]]
```

NUMPY ARRAYS

It is possible to stack arrays vertically or horizontally using **hstack** and **vstack**.

```
stacking_array1 = np.array([1, 10, 100])
stacking_array2 = np.array([2, 20, 200])
vertical_stack = np.vstack((stacking_array1, stacking_array2))
print(vertical_stack)
# [[ 1  10 100]
#   [ 2  20 200]]
horizontal_stack = np.hstack((stacking_array1, stacking_array2))
print(horizontal_stack)
# [ 1  10 100  2  20 200]
```

NUMPY ARRAYS

Another example of reshaping columns and rows. You can also use `vstack` to stack arrays **vertically**, or `hstack` to stack them **horizontally**.

```
even_array = np.arange(2, 101, 2)
print(even_array)
# [ 2  4  6  8 10 12 14 16 18 20 22 24 26 28 30 32 34 36
#  38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72
#  74 76 78 80 82 84 86 88 90 92 94 96 98 100]
odd_array = np.arange(1, 101, 2)
print(odd_array)
# [ 1  3  5  7  9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47
#  49 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95
#  97 99]
number_array = np.vstack((odd_array, even_array))
print(number_array)
# [[ 1  3  5  7  9 11 13 15 17 19 21 23 25 27 29 31 33 35
#   37 39 41 43 45 47 49 51 53 55 57 59 61 63 65 67 69 71
#   73 75 77 79 81 83 85 87 89 91 93 95 97 99]
#  [ 2  4  6  8 10 12 14 16 18 20 22 24 26 28 30 32 34 36
#   38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72
#   74 76 78 80 82 84 86 88 90 92 94 96 98 100]]
number_array = number_array.T
print(number_array)
# [[ 1  2]
#  [ 3  4]
#  [ 5  6]
#  [ 7  8]
#  [ 9 10]
#  ...
#  [99 100]]
```

NUMPY ARRAYS

You can convert a regular python list to a numpy array

```
my_list = [1, 2, 3, 4, 5]
my_array = np.array(my_list)
print(type(my_array)) # <class 'numpy.ndarray'>
print(type(my_list)) # <class 'list'>
```

NUMPY ARRAYS

Questions: <https://colab.research.google.com/drive/1thKnIIDs6Q0RbdpWEldcuT2e6yd1wWc4#scrollTo=krNASWb7LI8o>

Answers: <https://colab.research.google.com/drive/1VM5tzvFlteGNkSul0LZm5JFGM3EzNfPj>

Tutorials: <https://numpy.org/numpy-tutorials/>

LAB TIME

Coming up: Review for test

NEXT CLASS: