Introduction to Data Libraries

Coding Bootcamp

Learning Objectives

- A basic understanding of NumPy
- Understand the difference between arrays in NumPy
- Understand axis and shape properties for arrays
- An understanding of Matplotlib, and how to plot

Learning Requirements

- We will build upon our knowledge of data, and python
- Python syntax.
- Libraries and third party python modules.
- Basic maths understanding.

Today's Task!

Today we will be looking at the following libraries; NumPy and Matplotlib. We will have a look at dimensional arrays in NumPy and use Matplotlib to be able to see them in action.

These libraries are used for the exploration, interpretation, and management of data.

We will explore these libraries and have a go at interpreting some data!

What is NumPy

- NumPy is a fundamental package for scientific computing in Python:

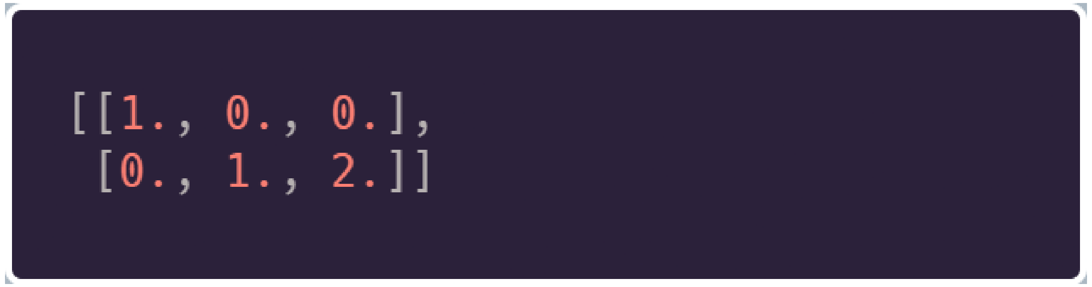
- It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

Why use NumPy

- NumPy is really helpful for us to manipulate data into helpful, readable, or important objects that can be interpreted easily when it comes to either plotting (we will use matplotlib), or Machine Learning.

NumPy basics

- NumPy’s main object is the multidimensional array.
- It is a table of elements (usually numbers), which are all the same type.
- It is indexed by a tuple of non-negative integers. In NumPy dimensions are sometimes called axes.

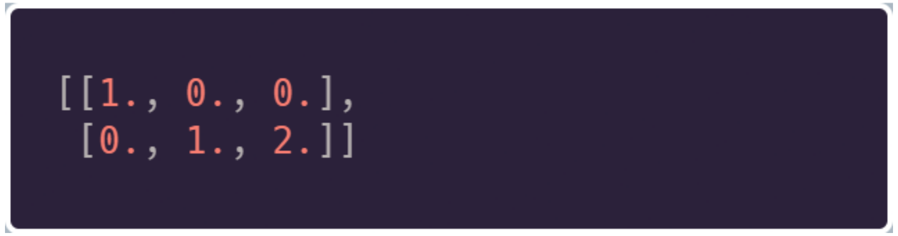


```
[[1., 0., 0.],
 [0., 1., 2.]]
```

NumPy basics

- For example, the array for the coordinates of a point in 3D space, [1, 2, 1], has one axis.

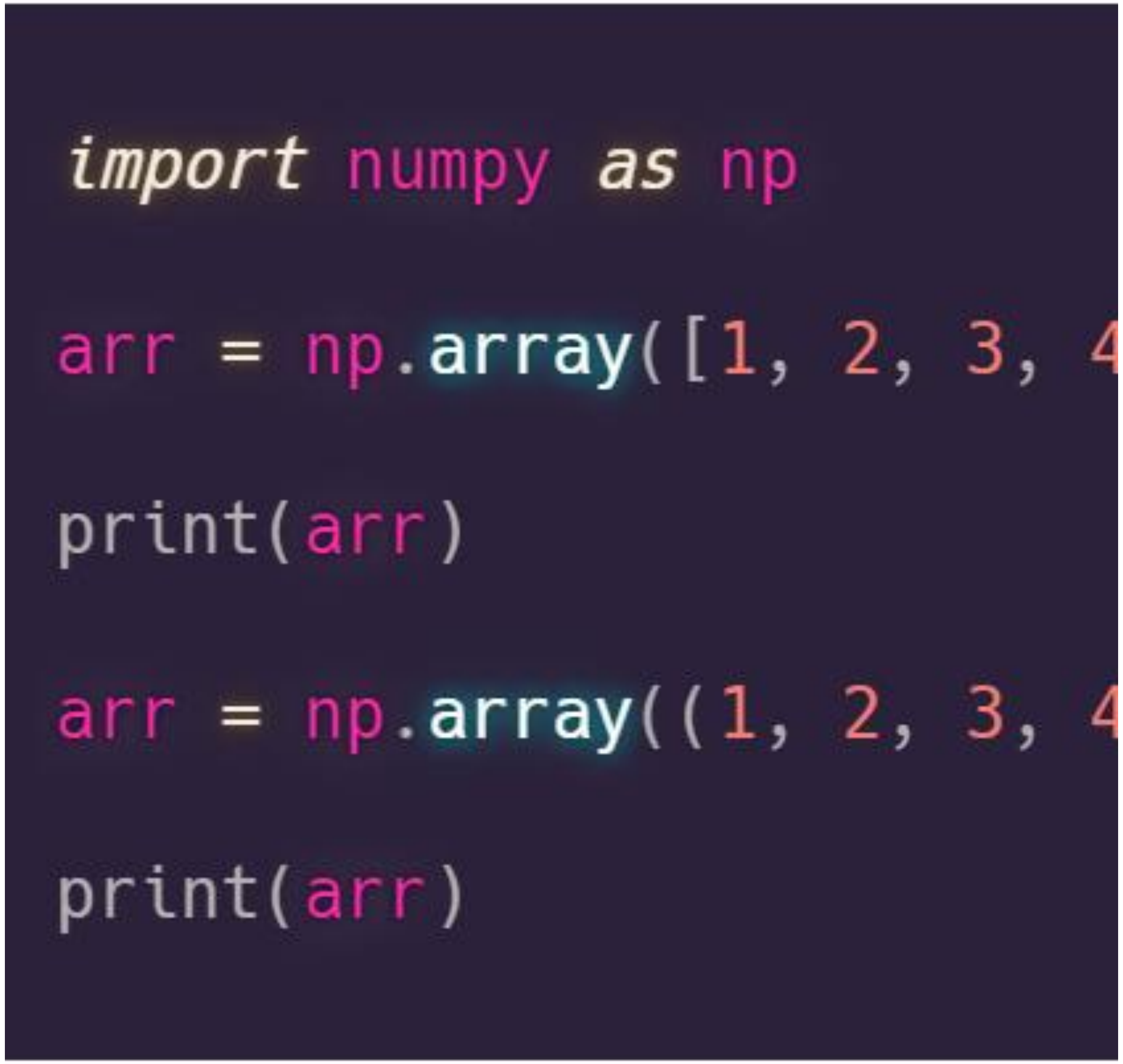
- That axis has 3 elements in it, so we say it has a length of 3. In the example pictured below, the array has 2 axes. The first axis has a length of 2, the second axis has a length of 3.



```
[[1., 0., 0.],
 [0., 1., 2.]]
```

NumPy basics

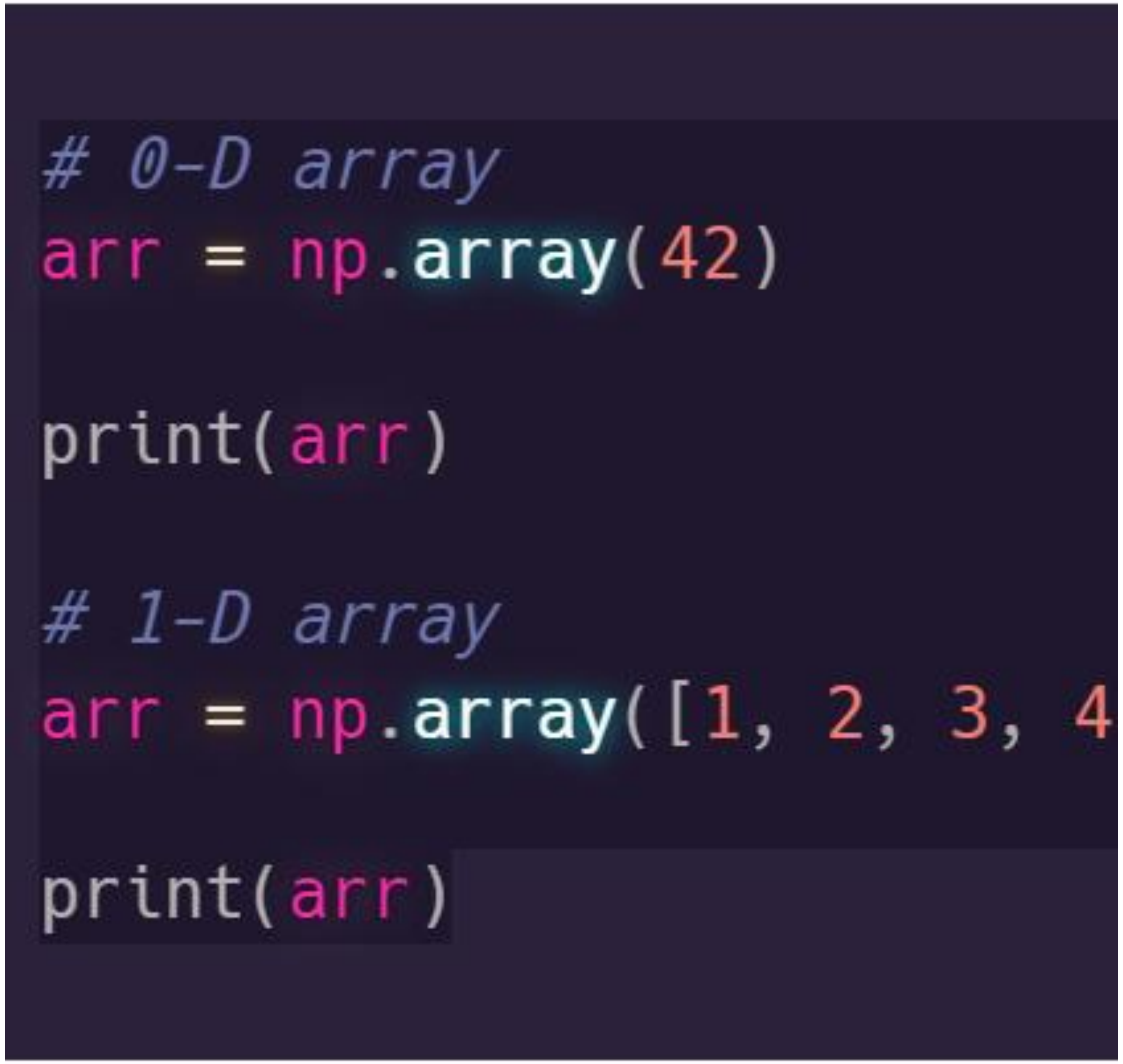
- We import NumPy as np.
- The array() method creates a NumPy array from our list.
- You can also make a NumPy array from a tuple!



```
import numpy as np

arr = np.array([1, 2, 3, 4

print(arr)

arr = np.array((1, 2, 3, 4

print(arr)
```

NumPy basics

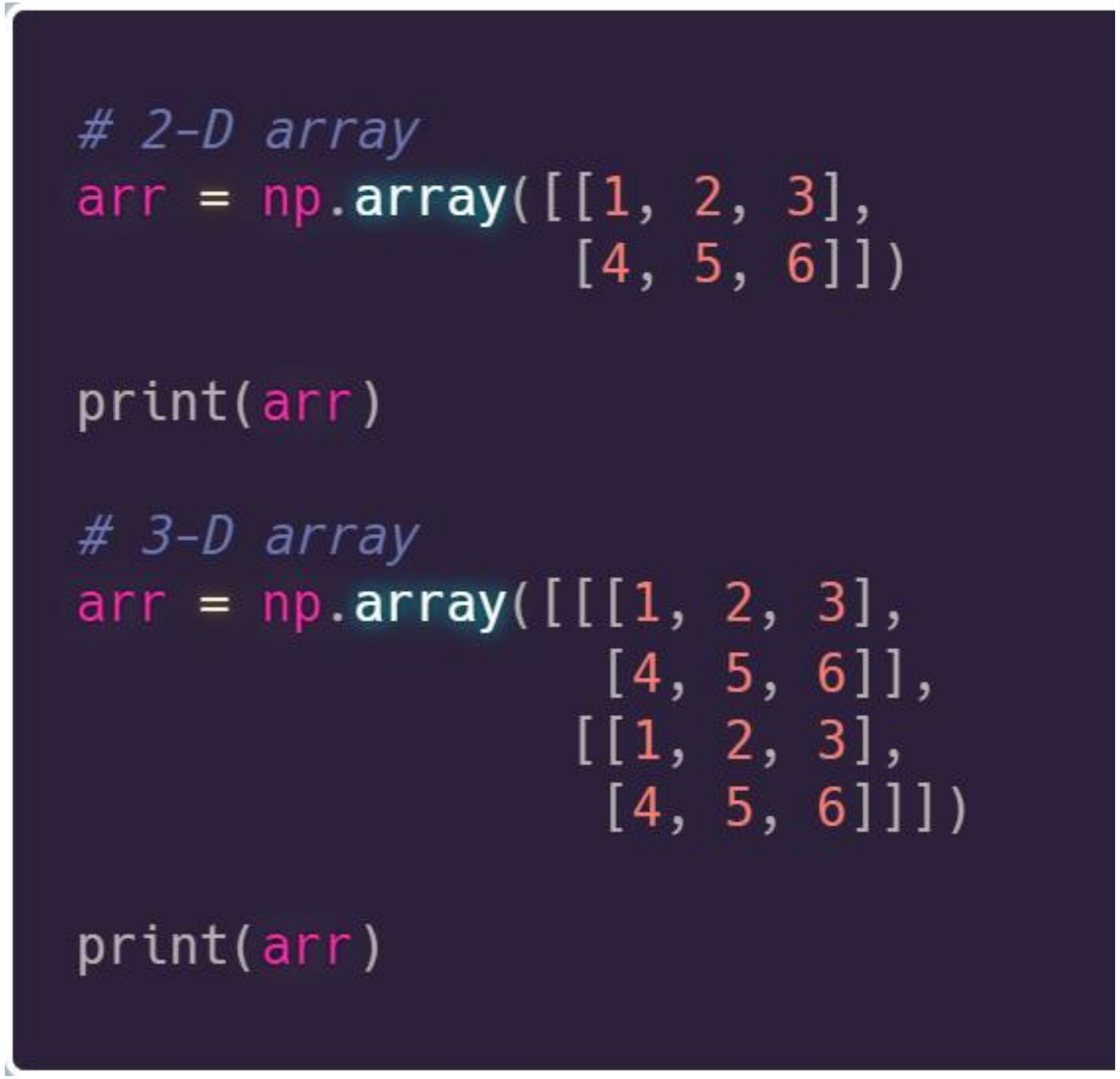
- We can create 0-D (dimensional) arrays.
- These are sometimes called Scalars.
- We can create 1-D arrays, like a list that contain one axis of scalars (numbers; floats or integers)



```
# 0-D array
arr = np.array(42)

print(arr)

# 1-D array
arr = np.array([1, 2, 3, 4

print(arr)
```

NumPy basics

- We can create 2-D (dimensional) arrays.
- These are sometimes called a Matrix, or Matrices.
- We can create 3-D arrays, by having two 2-D arrays, like the example opposite.
- They both contain two 1-D arrays.



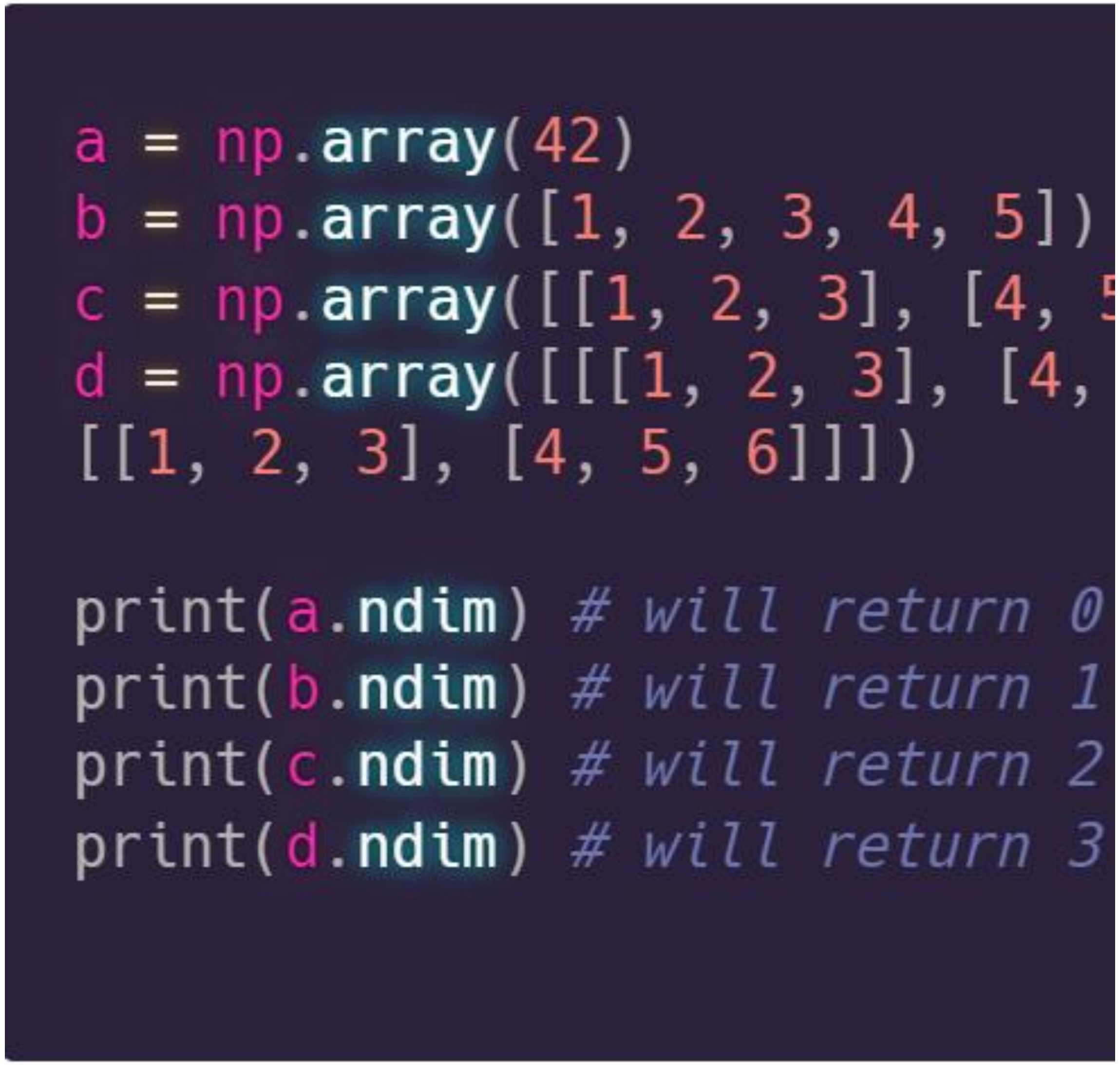
```
# 2-D array
arr = np.array([[1, 2, 3],
                [4, 5, 6]])

print(arr)

# 3-D array
arr = np.array([[[1, 2, 3],
                 [4, 5, 6]],
                [[1, 2, 3],
                 [4, 5, 6]]])

print(arr)
```

NumPy basics

- We can check how many dimensions an array has by calling the .ndim() method!

- This will be useful when it comes to manipulating our data.

- An array can have any number of dimensions.

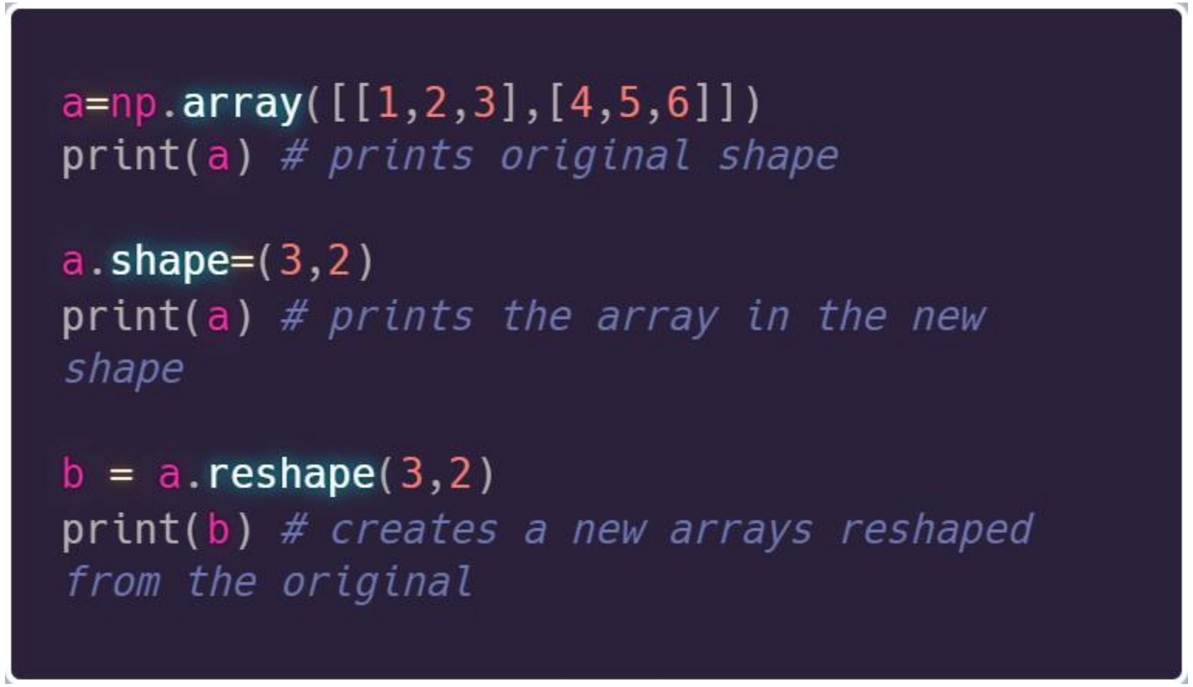


```
a = np.array(42)
b = np.array([1, 2, 3, 4, 5])
c = np.array([[1, 2, 3], [4, 5
d = np.array([[[1, 2, 3], [4,
[[1, 2, 3], [4, 5, 6]]])

print(a.ndim) # will return 0
print(b.ndim) # will return 1
print(c.ndim) # will return 2
print(d.ndim) # will return 3
```

NumPy basics

- We can shape and reshape our arrays to look differently.

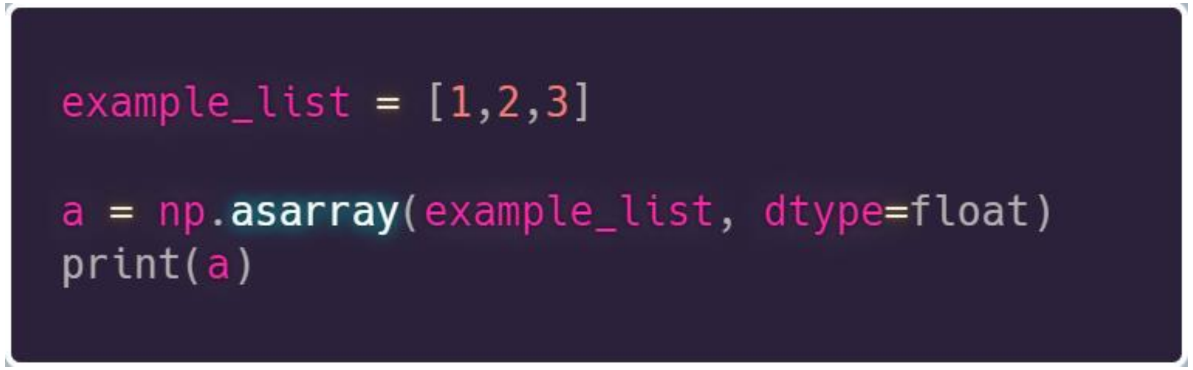
- Have a go at shaping and
- reshaping the arrays.



```
a=np.array([[1,2,3],[4,5,6]])
print(a) # prints original shape

a.shape=(3,2)
print(a) # prints the array in the new
shape

b = a.reshape(3,2)
print(b) # creates a new arrays reshaped
from the original
```

NumPy basics

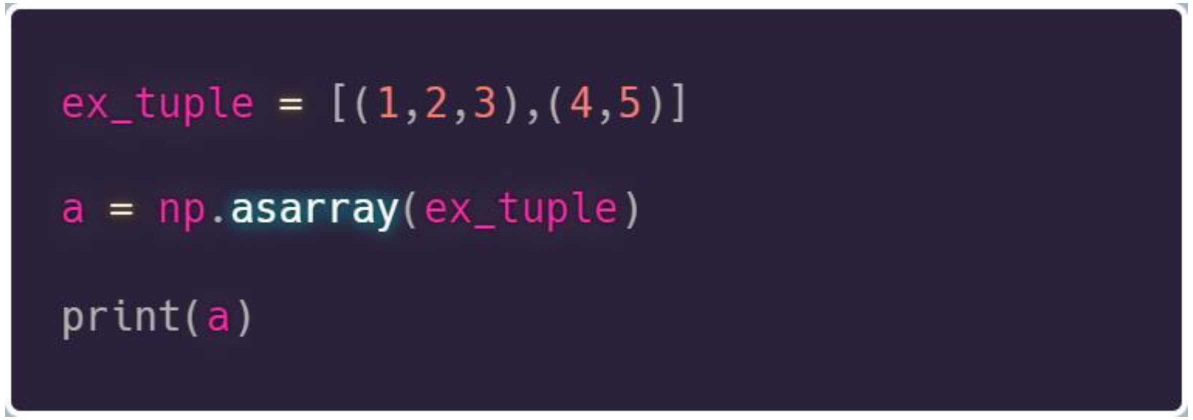
- We can take pre-existing data, like a list, and convert them into a NumPy array.



```
example_list = [1,2,3]

a = np.asarray(example_list, dtype=float)
print(a)
```

NumPy basics

- We can also take pre-existing data, like a tuple, and convert them into a NumPy array.

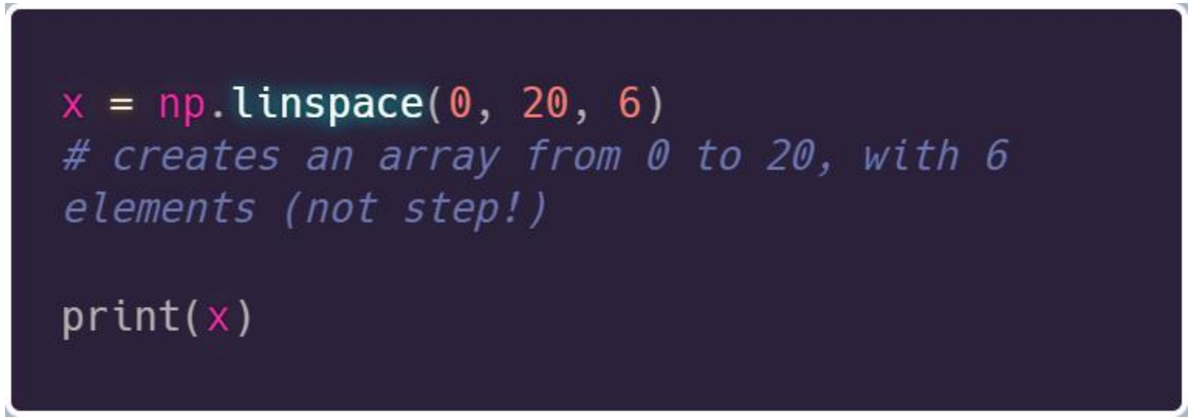


```
ex_tuple = [(1,2,3),(4,5)]

a = np.asarray(ex_tuple)

print(a)
```

NumPy basics

- We can make an array with the linspace() method.

- This will create an array with an equal space between array elements. Handy for making axis for graphs.



```
x = np.linspace(0, 20, 6)
# creates an array from 0 to 20, with 6
elements (not step!)

print(x)
```

Matplotlib basics

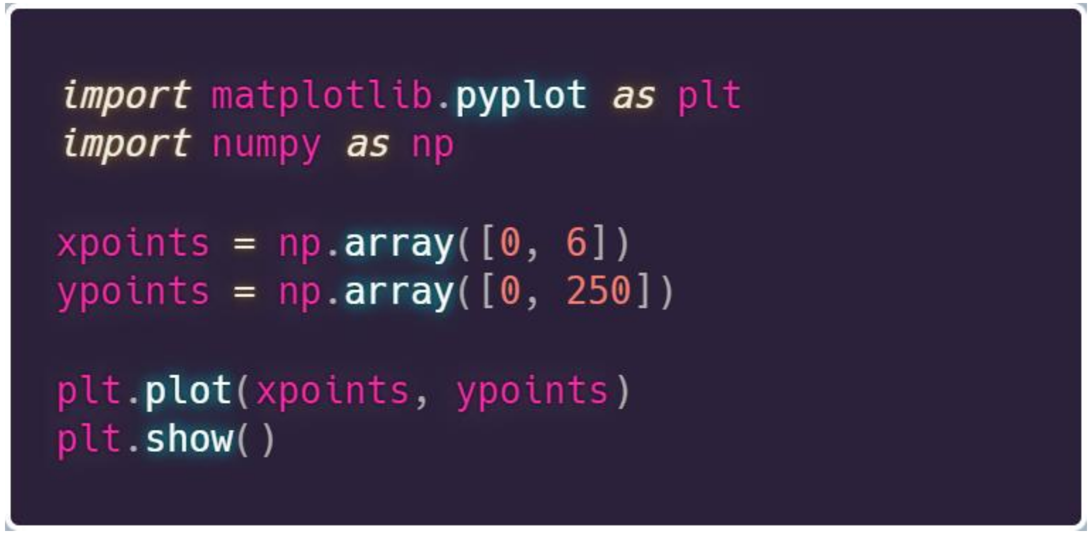
- Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy.

- It offers a viable open source alternative to MATLAB.

Matplotlib basics

- Next, we will take the NumPy arrays we can make, and use them to plot some graphs with matplotlib!

- Import matplotlib as plt (this is the most common representation).

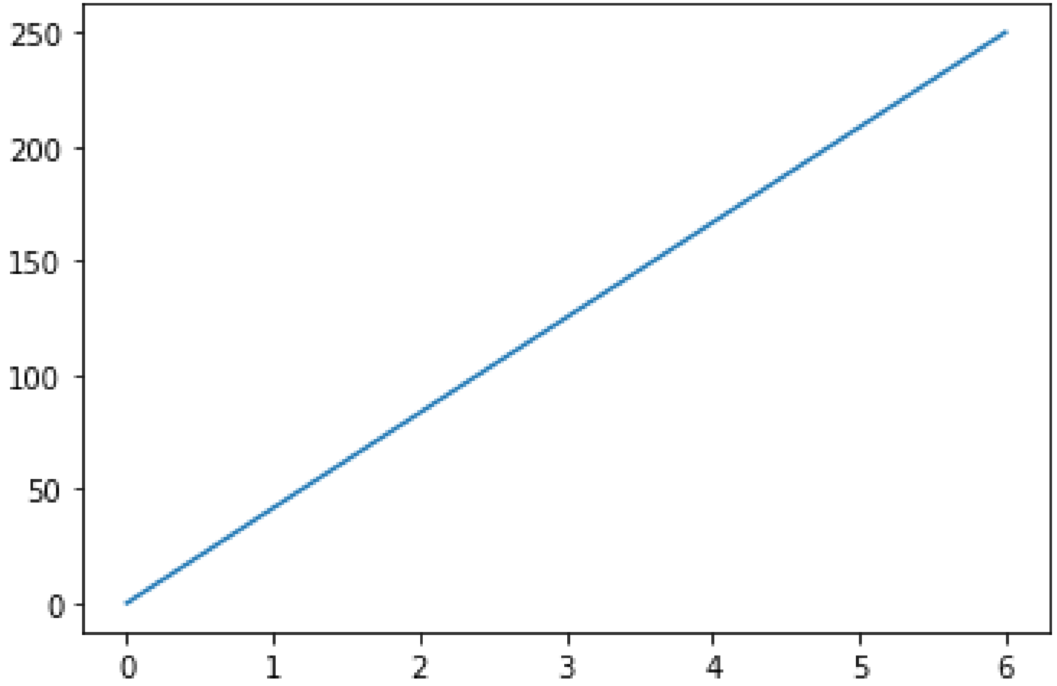


```
import matplotlib.pyplot as plt
import numpy as np

xpoints = np.array([0, 6])
ypoints = np.array([0, 250])

plt.plot(xpoints, ypoints)
plt.show()
```

Matplotlib basics

- We should get a nice straight line like this.

- Our x axis goes from 0 – 6, our y axis from 0 - 250



Matplotlib basics

- Let’s take a vote!

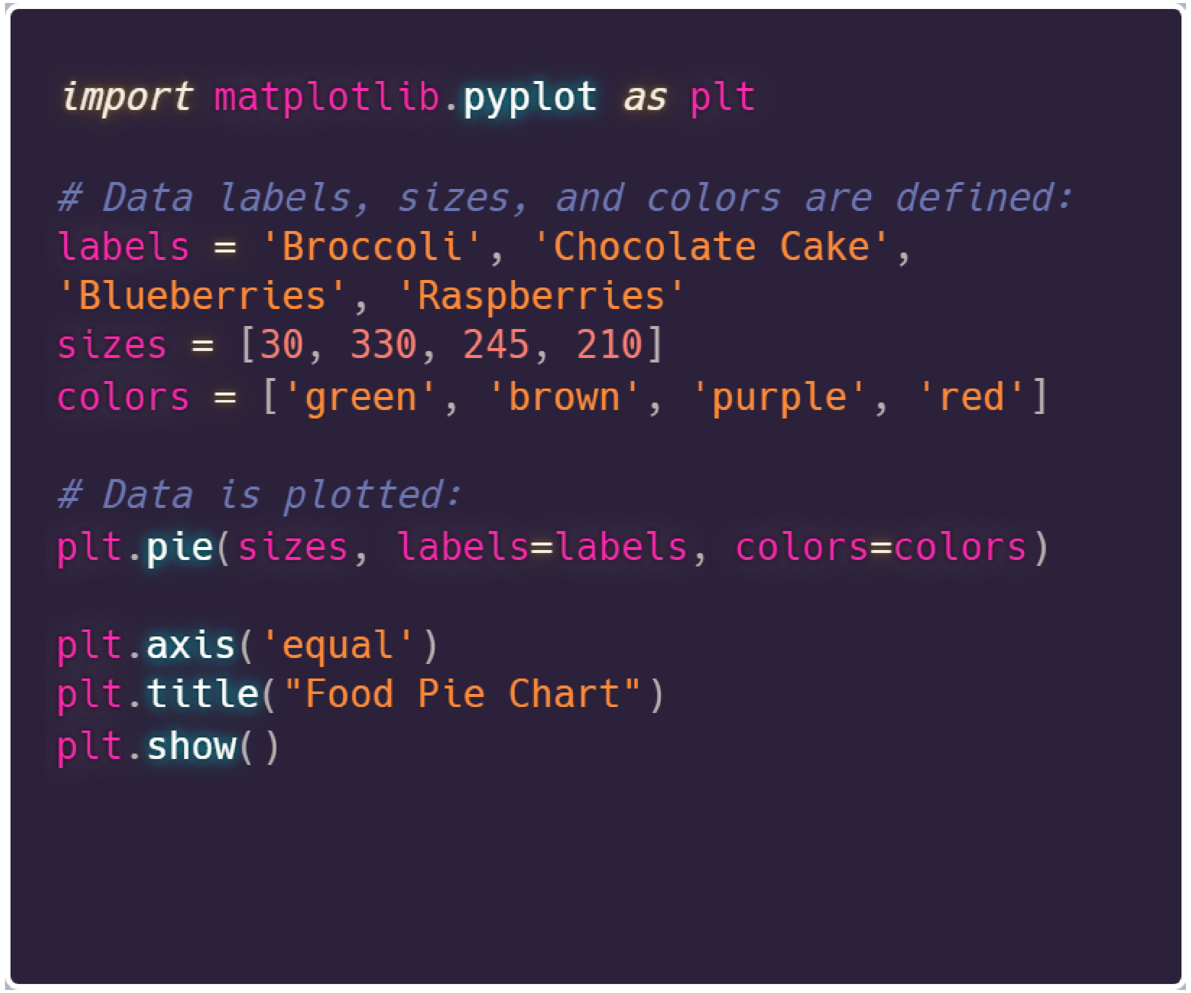
- Out of the following options, which would you prefer:

    - Broccoli
    - Chocolate Cake
    - Strawberries
    - Porridge

Matplotlib basics

- We can make a nice pie chart, by using some similar syntax, we simply use plt.pie() instead of plot().

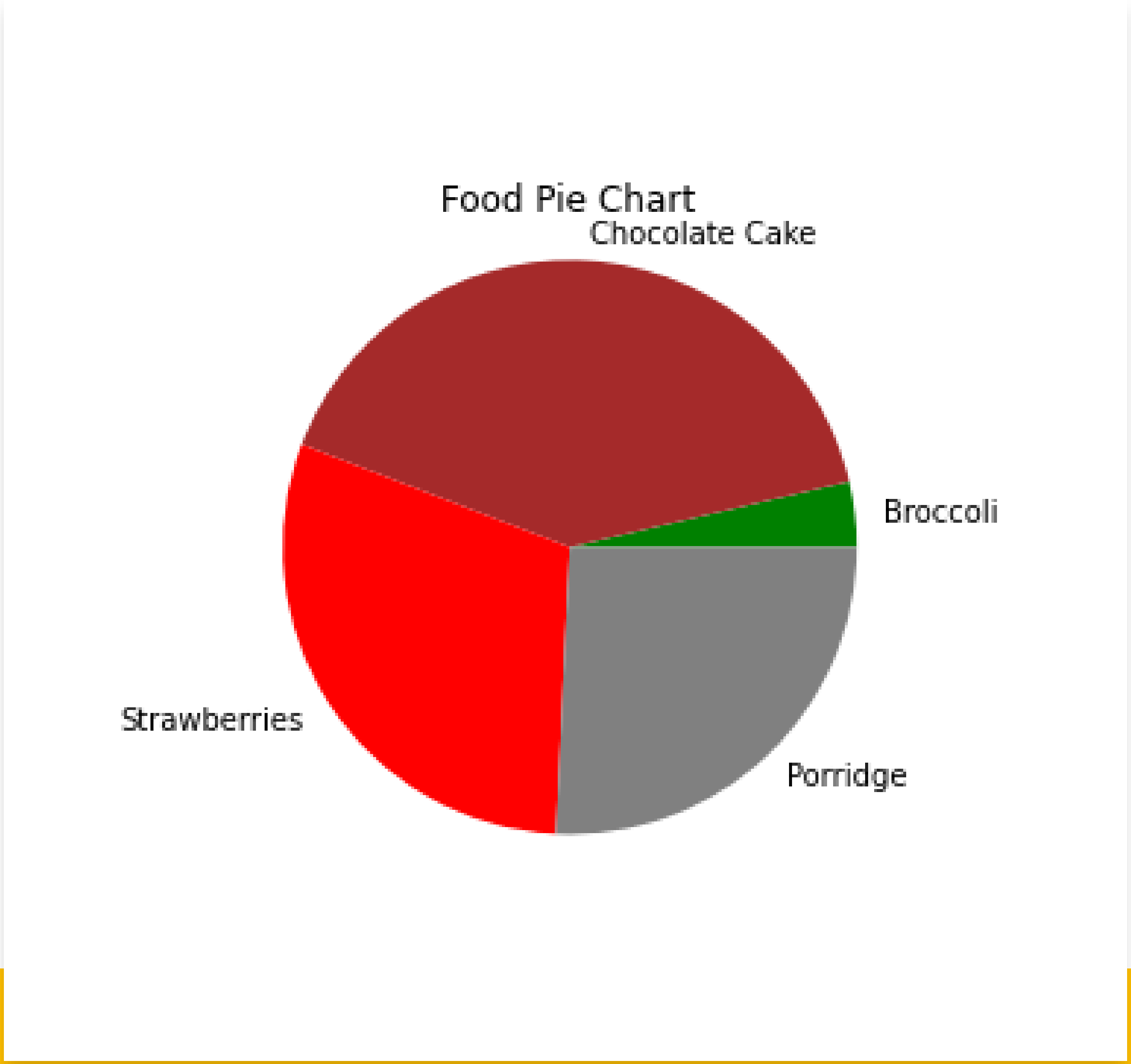
- Note the use of labels, we will retain the same idea of labels when we use Machine Learning.



```
import matplotlib.pyplot as plt

# Data labels, sizes, and colors are defined:
labels = 'Broccoli', 'Chocolate Cake',
'Blueberries', 'Raspberries'
sizes = [30, 330, 245, 210]
colors = ['green', 'brown', 'purple', 'red']

# Data is plotted:
plt.pie(sizes, labels=labels, colors=colors)

plt.axis('equal')
plt.title("Food Pie Chart")
plt.show()
```

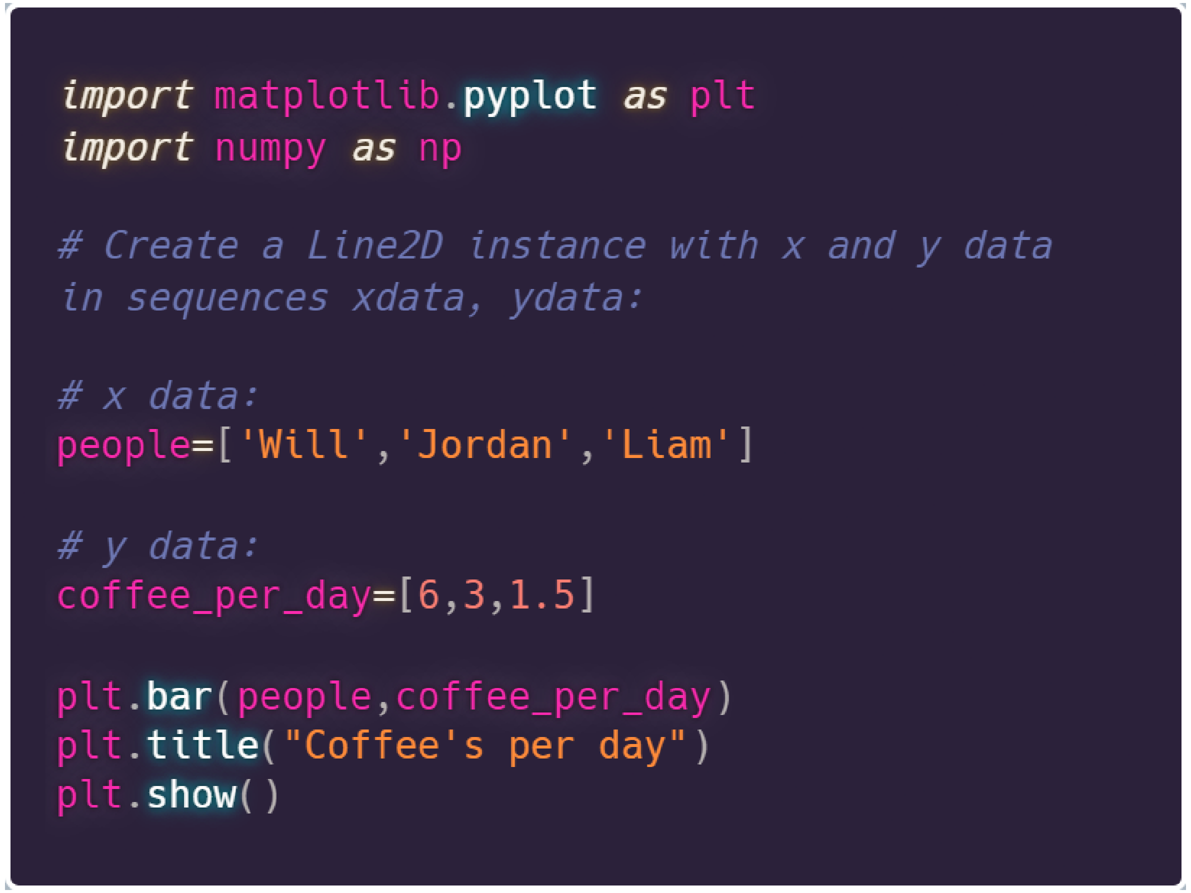
Matplotlib basics

- We should get a nice pie chart like this.
- Or maybe more people have lied about preferring Broccoli…



Matplotlib basics

- Or if we looked at how many coffee’s per day, we had…
- We use plt.bar() to show us a bar histogram!

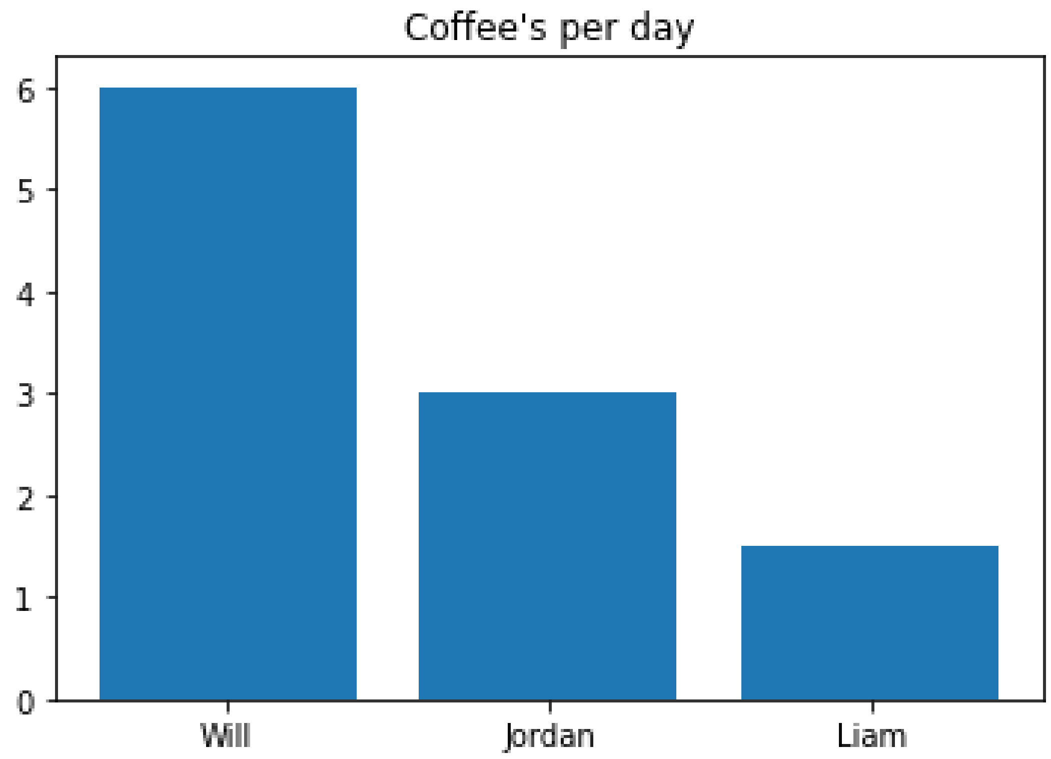


```
import matplotlib.pyplot as plt
import numpy as np

# Create a Line2D instance with x and y data
in sequences xdata, ydata:

# x data:
people=['Will','Jordan','Liam']

# y data:
coffee_per_day=[6,3,1.5]

plt.bar(people,coffee_per_day)
plt.title("Coffee's per day")
plt.show()
```

Matplotlib basics

- We should see a simple histogram similar to this example one.

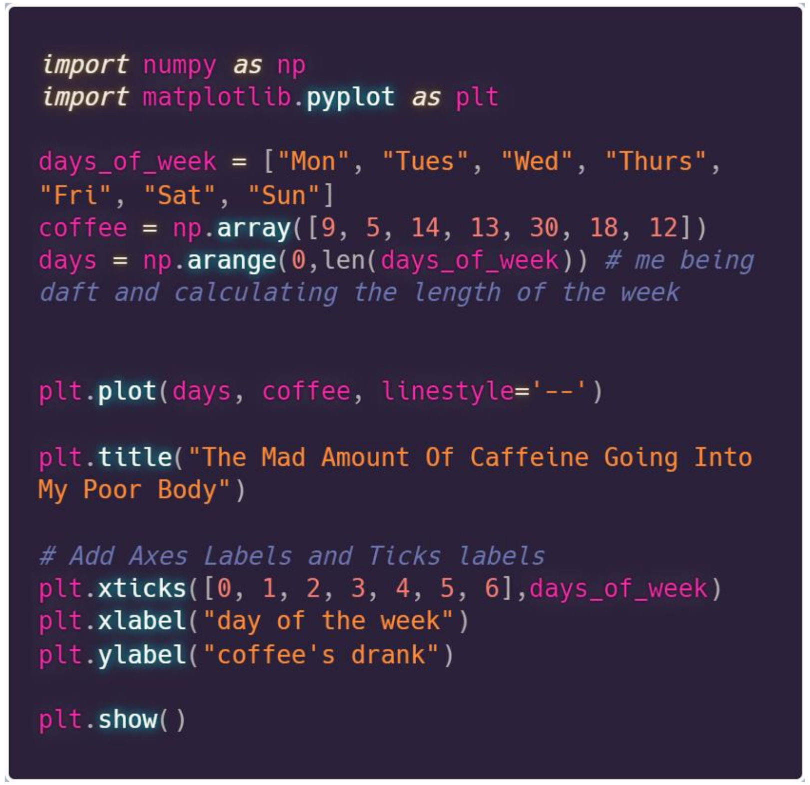
- These plotting methods can be really handy for Light Data Exploration



NumPy & Matplotlib

- So, let’s put our libraries together and see how we can plot using matplotlib and our NumPy arrays!

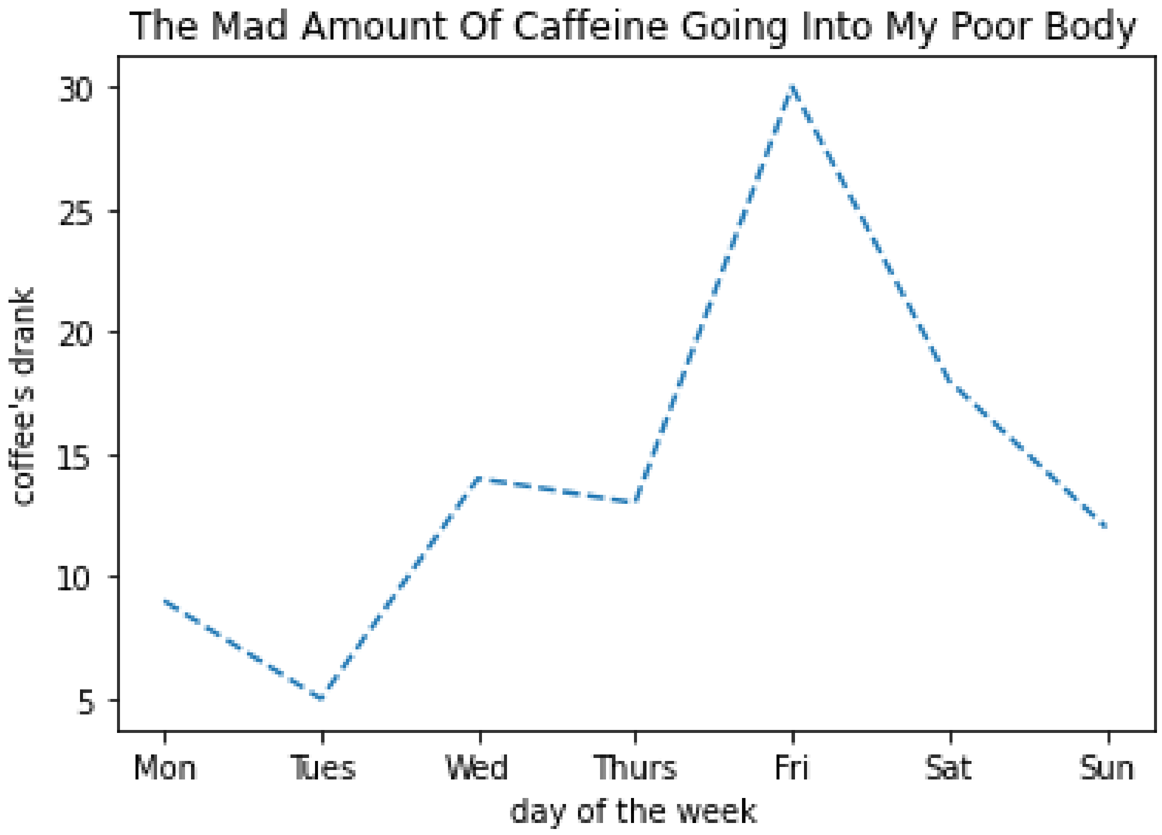
- Notice the use of xticks() in order to have custom x ticks along the x axis.



```
import numpy as np
import matplotlib.pyplot as plt

days_of_week = ["Mon", "Tues", "Wed", "Thurs",
"Fri", "Sat", "Sun"]
coffee = np.array([9, 5, 14, 13, 30, 18, 12])
days = np.arange(0,len(days_of_week)) # me being
daft and calculating the length of the week


plt.plot(days, coffee, linestyle='--')

plt.title("The Mad Amount Of Caffeine Going Into
My Poor Body")

# Add Axes Labels and Ticks labels
plt.xticks([0, 1, 2, 3, 4, 5, 6],days_of_week)
plt.xlabel("day of the week")
plt.ylabel("coffee's drank")

plt.show()
```

NumPy & Matplotlib

- I recorded the amount of caffeine I'd ingested over the course of the week, and wanted to see how it fared against days of the week.
- Let's take a look. Feel free to adjust the numbers if you're not quite as addicted to coffee as we are…
- From the data we can see that I increase caffeine intake a lot through the working week.



“Maybe stories are just data with a soul.”
— Brené Brown

Manipulating Data

- Let’s look at some larger data sets.

- On Kaggle there is a data set called ‘Hollywood Theatrical Market Synopsis 1995-2021’.

- You can either find it at https://www.kaggle.com/johnharshith/hollywood-theatrical-market-synopsis-1995-to-2021

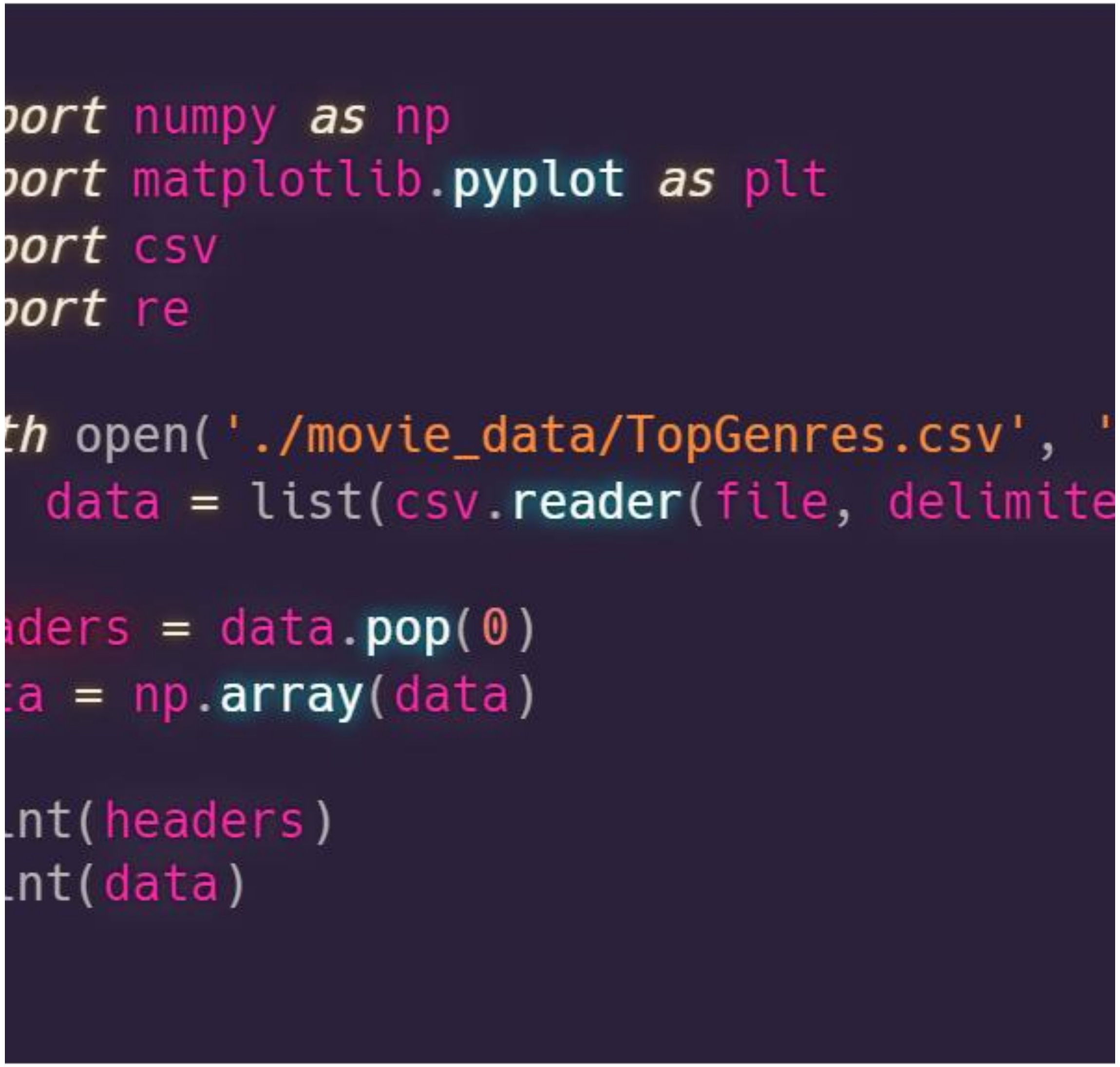
- *Or at* https://github.com/seriouswill/movie_data

Manipulating Data

- In our jupyter notebooks, let’s take a look at one of the .csv files we downloaded. We picked the TopGenre.csv

- We’re going to need to import csv, and re as well as numpy and matplotlib

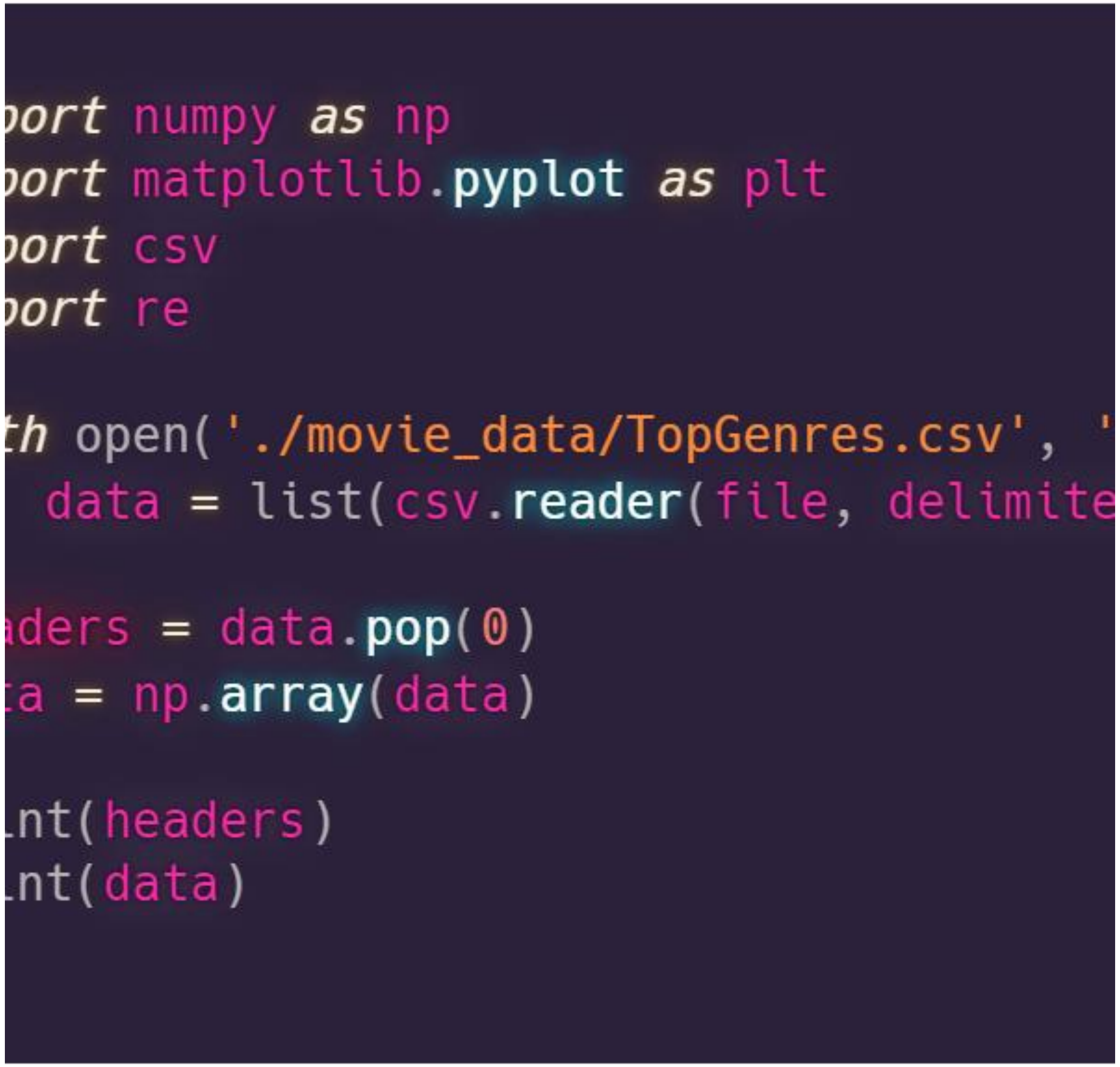
- The re library is for ReGex.



```
port numpy as np
port matplotlib.pyplot as plt
port csv
port re

th open('./movie_data/TopGenres.csv', '
 data = list(csv.reader(file, delimite

aders = data.pop(0)
a = np.array(data)

nt(headers)
nt(data)
```

Manipulating Data

- We’ll use the with open method and the csv library to read the csv file.
- Then we’ll put the first row (the headers) in a seperate variable with .pop(). And then print them out to have a look at the data.

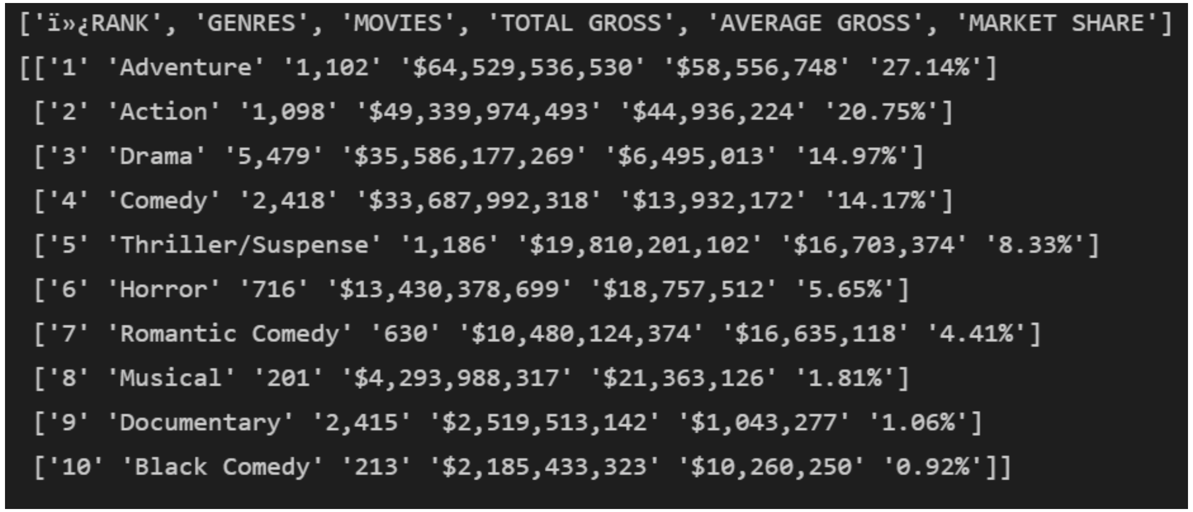


```
port numpy as np
port matplotlib.pyplot as plt
port csv
port re

th open('./movie_data/TopGenres.csv', '
 data = list(csv.reader(file, delimite

aders = data.pop(0)
ta = np.array(data)

nt(headers)
nt(data)
```

Manipulating Data

- The data should look something like this.

- What can you spot about the data that we might need to change?



```
['ï»¿RANK', 'GENRES', 'MOVIES', 'TOTAL GROSS', 'AVERAGE GROSS', 'MARKET SHARE']
[['1' 'Adventure' '1,102' '$64,529,536,530' '$58,556,748' '27.14%']
 ['2' 'Action' '1,098' '$49,339,974,493' '$44,936,224' '20.75%']
 ['3' 'Drama' '5,479' '$35,586,177,269' '$6,495,013' '14.97%']
 ['4' 'Comedy' '2,418' '$33,687,992,318' '$13,932,172' '14.17%']
 ['5' 'Thriller/Suspense' '1,186' '$19,810,201,102' '$16,703,374' '8.33%']
 ['6' 'Horror' '716' '$13,430,378,699' '$18,757,512' '5.65%']
 ['7' 'Romantic Comedy' '630' '$10,480,124,374' '$16,635,118' '4.41%']
 ['8' 'Musical' '201' '$4,293,988,317' '$21,363,126' '1.81%']
 ['9' 'Documentary' '2,415' '$2,519,513,142' '$1,043,277' '1.06%']
 ['10' 'Black Comedy' '213' '$2,185,433,323' '$10,260,250' '0.92%']]
```

Discuss

Manipulating Data

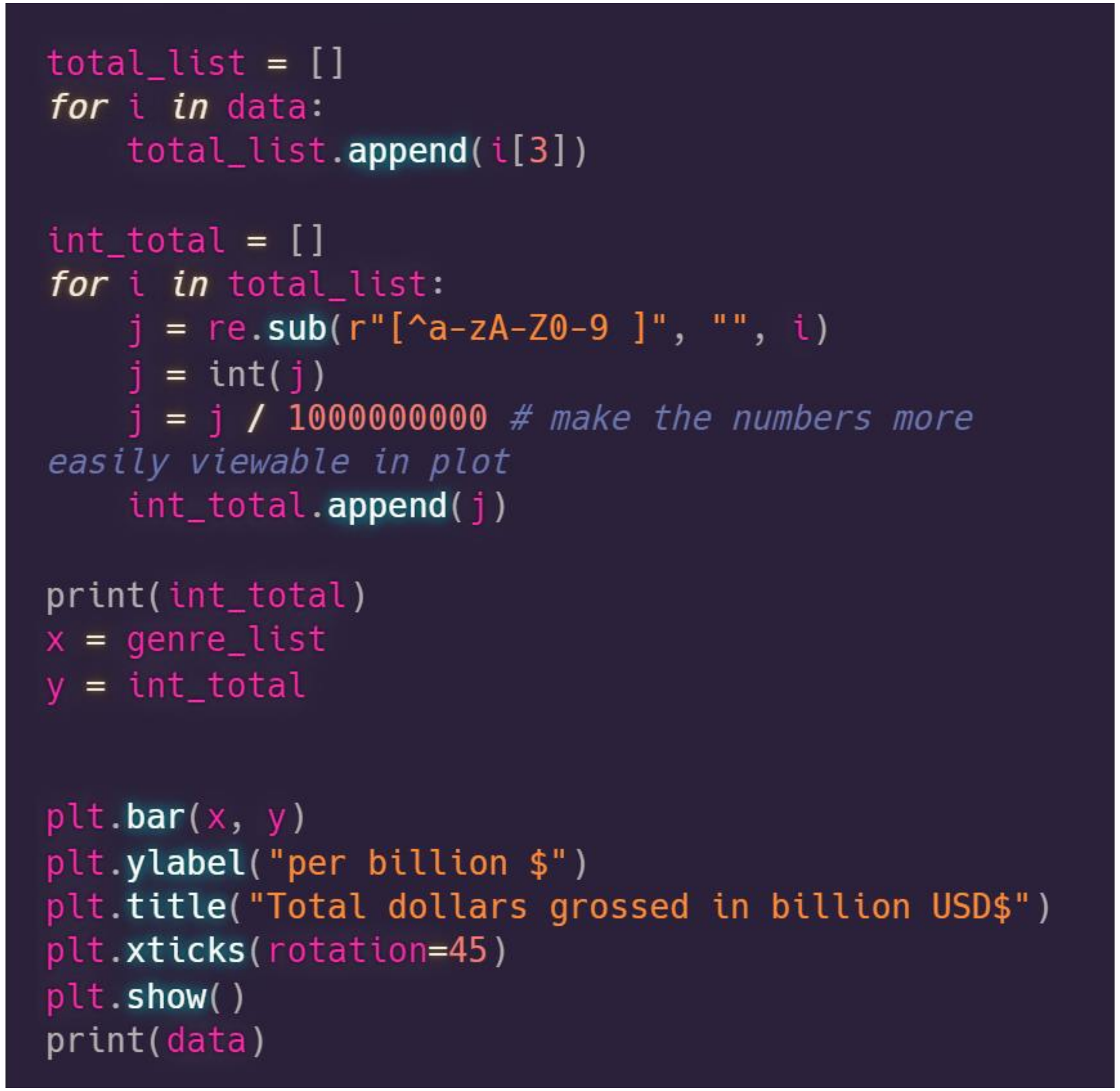
- We can manipulate the data like this.

- We can make an x axis out of the genre, by .append() to an empty list.

- But to strip the special characters out of the movie total box office we need to use a bit of ReGex, turn each item into an integer, then put those edited entries in a new list!

- Then we can make a bar chart to view our data!

- (We divided the total gross entries by a billion to make it more easily viewable in plot)



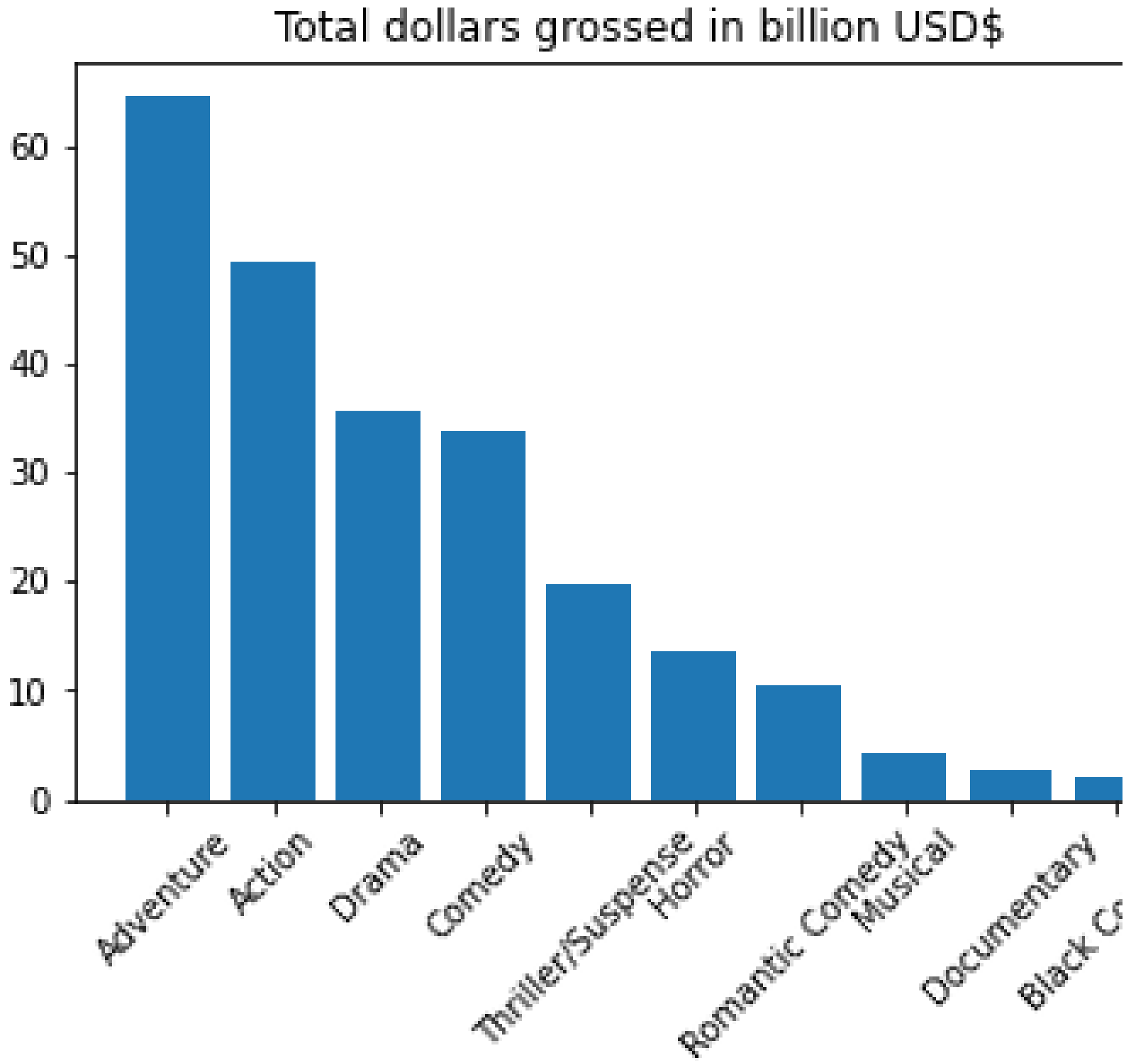
```
total_list = []
for i in data:
    total_list.append(i[3])

int_total = []
for i in total_list:
    j = re.sub(r"[^a-zA-Z0-9 ]", "", i)
    j = int(j)
    j = j / 1000000000 # make the numbers more
easily viewable in plot
    int_total.append(j)

print(int_total)
x = genre_list
y = int_total


plt.bar(x, y)
plt.ylabel("per billion $")
plt.title("Total dollars grossed in billion USD$")
plt.xticks(rotation=45)
plt.show()
print(data)
```

Manipulating Data

- So, this is what our cleaned data looks like when plotted in a bar chart.

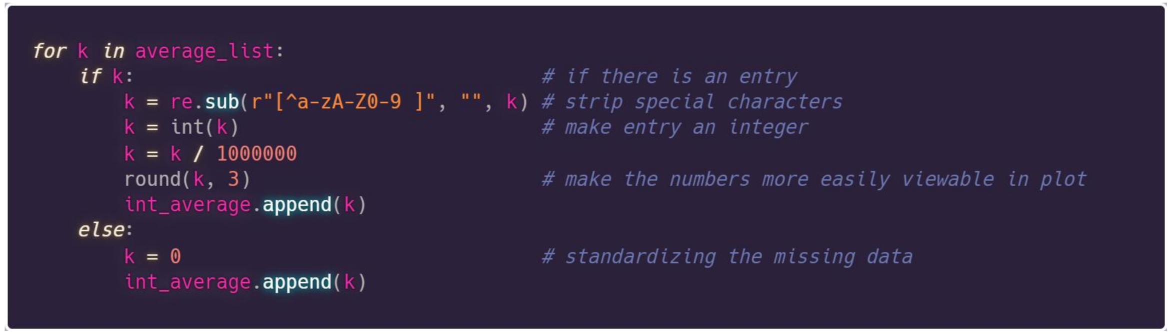
- Luckily, we didn’t need to engineer or clean our data too much. If we take a look at some of the other .csv files in the data package, we can see there are some which have entries missing.

- What do you do in that case!?

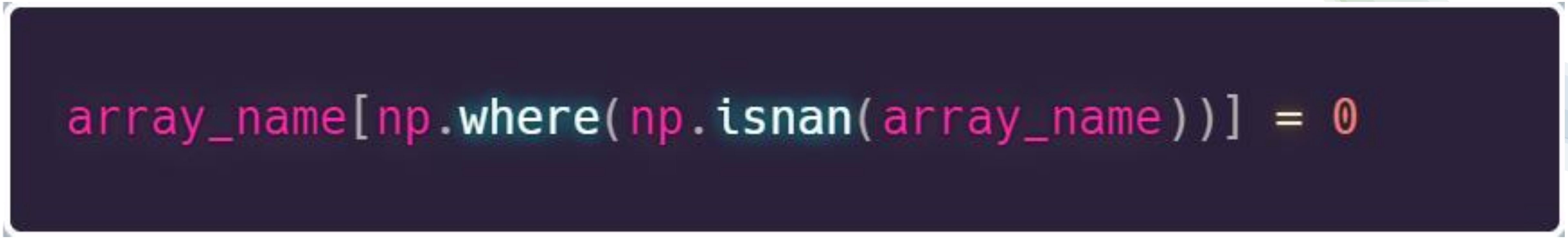


When we manipulate the data to prepare it, we can attribute the missing data to be 0:

Manipulating Data



```
for k in average_list:
    if k:                                    # if there is an entry
        k = re.sub(r"[^a-zA-Z0-9 ]", "", k) # strip special characters
        k = int(k)                           # make entry an integer
        k = k / 1000000
        round(k, 3)                          # make the numbers more easily viewable in plot
        int_average.append(k)
    else:
        k = 0                                # standardizing the missing data
        int_average.append(k)
```



```
array_name[np.where(np.isnan(array_name))] = 0
```

Manipulating Data

Or NumPy has a function to do it for us:

“Information wants to be free.”
— Stewart Brand

Learning Objectives

A basic understanding of NumPy

Understand the difference between arrays in NumPy

Understand axis and shape properties for arrays

An understanding of Matplotlib, and how to plot

Activity Goals

- **Activity 1:**
- Have a look at some datasets on kaggle.com, and see if you can explore them, and plot the data using matplotlib inside your jupyter notebook.

- **Activity 2:**
- Export your jupyter notebook to a pdf to make it ready for presentation.