DATA TYPES FOR DATA SCIENCE IN PYTHON

- Module (sometimes called packages or libraries) groups functions together
- Example:
    - **statsmodels**: used in machine learning, usually aliased as sm
    - **seaborn**: a visualization library, usually aliased as sns
    - **numpy**: perform mathematical operations on lists of data, usually aliased as np
    - **panda** is a module that uses tools like DataFrame to examine and modify tabular data
- Add module by using 'import'
- Variable store date: string, float
- Functions are like machine which turn input to output

- Use the **.info()** method to inspect the DataFrame
- Two methods for selecting columns
- Using brackets and string notation
    - locations = **crime_data['location']**
- Using dot notation
    - locations = **crime_data.location**
- ***If a column name contains a space, then it needs to be in brackets and string notation***

Creating line plots

Matplotlib is a module for creating charts and visualization

From matplotlib, import pyplot aliased as plt: **from mathplotlib import pyplot as plt**

Pick a style: **plt.style.use(" ")**

Plot, add a label, line color, linestyle, marker to the plot

plt.plot(x, y, label = " ", color = " ", linestyle = ":"/"- -"/" ", marker = 'o'/'d'/'s')

Add a title: **plt.title(" ")**

Add y-axis label: **plt.ylabel(" ")**

Add x-axis label: **plt.xlabel(" ")**

#Add notation: **plt.text(x_1, y_1, " ")**

Add a legend: **plt.legend()**

Display a plot: **plt.show()**

Creating scatter plot

# From matplotlib, import pyplot aliased as plt:	**from mathplotlib import pyplot as plt**
# Create a scatter plot:	**plt.scatter(x, y, color = " ")**
# Add labels	**plt.ylabel(" ")**
	plt.xlabel(" ")
# Display the plot:	**plt.plot()**

Data Type

List is a collection of ordered data

- To add data elements to a list — **.append()**
- Method to combine a list with another array type (list, set, tuple) — **.extend()**
- Method to find the position of an item in a list — **.index()**
- Method to remove an item in a list — **.pop()**
- Use a **for** loop to iterate through all the items in a list
- Function is for sorting the data in a list from lowest to highest — **sorted()**

Tuples are ordered collection of data. Tuples are made of several items just like a list, but they cannot be modified in any way

  - You can use an index just like a list
  - You can also unpack the tuple into multiple variables
- Function is used to pair up multiple array data types — **zip()**
- When looping over a list, you can also track your position in the list by using the **enumerate()** function. This function will return the index of the list item

A Set is an unordered collection of data. Some common functions used in set:

  - **.union()**
  - **.intersection()**
  - **len()** function is to compute the number of names
  - **difference()**
  - **add():** used to add items to a set

A dictionary is an unordered collection of data that stores in key-value pairs

  - **sorted():** sort by the keys of the dictionary. Reverse the order by passing reverse = True as a keyword argument

- **.get()** method allows you to supply the name of a key and optionally, what you'd like to have returned if the key is not found
- **.keys()** method is used to explore a new dictionary
- If you want to add data to a dictionary, you can simply create a new key and assign the data you desire to it.
  - Use **.update()** method to update a dictionary with keys and values from another dictionary, tuples or keyword arguments
    - Ex: names[2021].update([(1, 'Aaron'), (2, 'Bob')])
- **.item()** method is used for iterating over items in a dictionary and this returns each key and value from the dictionary as a tuple which you can unpack in a **for** loop.
- Check to see if a key exists in a dictionary by using the **in** expression

---

Working with csv file

- Python **csv** module
- **open()** function is to create a Python file object which accepts a file name and a mode. The mode is 'r' for read and 'w' for write
- **[1:]** to skip the header row
- **csv.reader()** reads a file object and returns the lines from the file as tuples
- The **csv** module also provides a way to directly create a dictionary from a csv file with the **DictReader** class. If the file has a header row, that row will automatically be used as the keys for the dictionary.
- **.close()** method closes file objects

Using Counter on Lists

- **Counter** is a powerful tool for counting, validating and learning more about the elements within a dataset that is found in the **collections** module.
- Pass an iterable (list, set, tuple) or a dictionary to the Counter
- Use **Counter** object similarity to a dictionary with key/value assignment: Counter[key] = value
  - Finding the most common elements in a list: **.most_common()**
- **defaultdict** passes it a default type that every key will have even if it doesn't currently exist

- Work exactly like a dictionary. You can pass it the type you want it to be such as a list, tuple, set, int, string, dictionary or any other valid type object.

OrderedDict

- OrderedDict is a dictionary subclass that remembers the order in which its contents are added
- If your code is heavily base on dictionaries and you're dealing with missing keys all the time

```
# Import OrderedDict from collections

from collections import OrderedDict
```

Dates and Times

Import the datetime object from datetime

```
from datetime import datetime
```

- Strings to DateTimes by using **.strptime()**
- Converting to a String by using **.strftime()**
- The **.now()** method on the datetime object in the datetime module returns the current local time on the machine on which it is run
- **.utcnow()** does the same thing but returns the values in UTC time

Timezones from **pytz** import **timezone**

- **.replace(tzinfor=" ")** makes a datetime object 'aware'
- **.astimezone()** method accepts a timezone object and returns a new datetime object in the desired timezone

Timedelta from **datetime** import **timedelta**

- The timedelta object from the datetime module is used to represent differences in datetime object

Pendulum

- Pendulum is a Python package to ease datetimes manipulation import **pendulum**
- Create a now datetime **pendulum.now(" ")**
- Convert to another timezone **.in_timezone(" ")**

- Convert to ISO 8601 string **.to_iso8601_string()**
- Convert strings to datetime **.parse(, strict = False)**