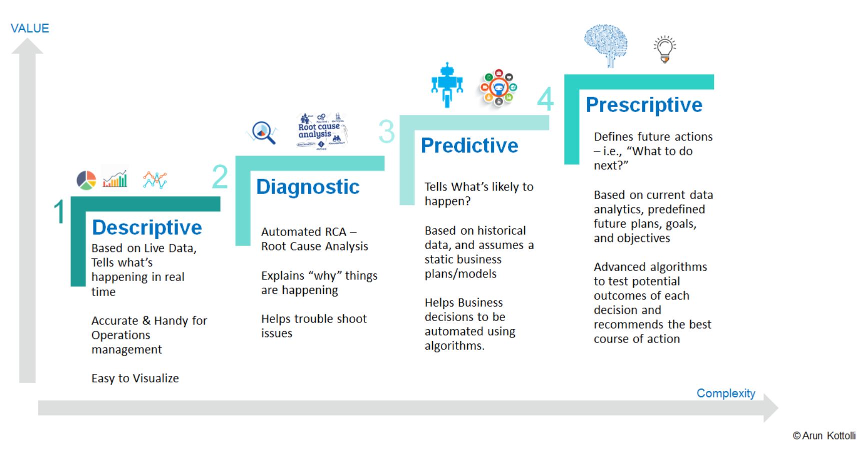
Today's Agenda

- Types of Data Analytics
- Python Libraries for DA
- Codes and Examples of various statistical measurements
- Data Visualization
- Outlier Detection

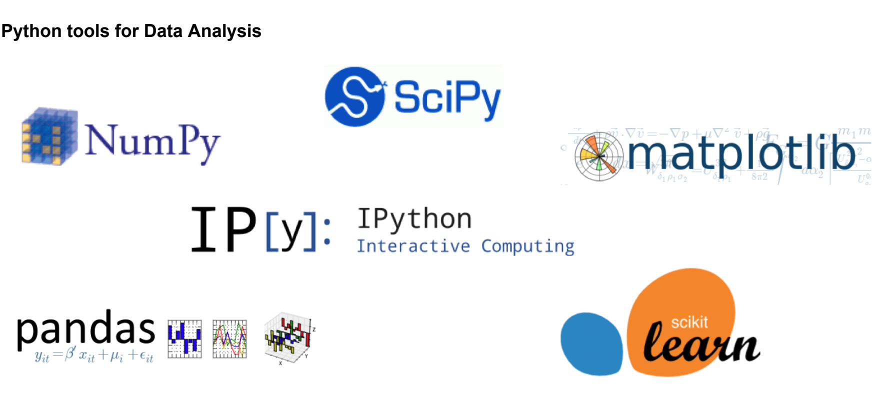
```
In [ ]:
```



Credits - Image from Internet

```
In [ ]:
```

Python tools for Data Analysis



Credits - Image from Internet

Other libraries

- OpenCV
- Pandas Profiling
- GeoPandas
- BioPandas
- Plotly
- Statsmodels
- Statistics
- Tensorflow
- Pytorch
- -
- Dash
- Flask

Basic Statistics

- Mean → Average of all the data values
  - Sum of all data values divided by total number of data values
- Median → The value separating the higher half from the lower half of the data
- Mode → The value that appears most frequently in the data set
- Standard deviation → Used to measure of the amount of variation or dispersion of set of values
  - Low standard deviation → All values very close to mean
  - High standard deviation → All values are far from the mean

`import` the necessary packages

```
In [1]: import pandas as pd
        import numpy as np
        from collections import Counter
```

Mean

```
In [2]: def calculate_mean(data_values):
            return sum(data_values) / len(data_values)
```

```
In [3]: # show example
        x = [5, 6, 1, -10, 4, 8, 10]
        mean_value = calculate_mean(data_values=x)
        print(mean_value)

        3.4285714285714284
```

Median

```
In [4]: 5 / 2
Out[4]: 2.5

In [5]: int(5 / 2)
Out[5]: 2

In [6]: 5 // 2
Out[6]: 2
```

Procedure -

- First sort the values
- If the total number of values is odd
  - Take the middle value
- If the total number of values is even
  - Take the two middle values
  - Find the average of those two middle values

```
In [7]: def calculate_median(data_values):
            sorted_values = sorted(data_values)
            mid_index = len(data_values) // 2

            # odd case
            if len(data_values) % 2 != 0:
                median = sorted_values[mid_index]
            # even case
            else:
                mid_index_l = mid_index - 1
                mini_data = [sorted_values[mid_index_l], sorted_values[mid_index]]
                median = calculate_mean(mini_data)

            return median
```

```
In [8]: # show example
        x = [5, 6, 1, -10, 4, 8, 10, 9, 100, 32]
        median_value = calculate_median(data_values=x)
        print(median_value)

        7.0
```

Mode

```
In [9]: d = [1, 1, 2, 3, 4, 4, 5, 6, 7, 8, 9, 8]
        print(Counter(d))
        print( list( Counter(d).keys() ) )
        print( list( Counter(d).values() ) )

        Counter({1: 2, 4: 2, 8: 2, 2: 1, 3: 1, 5: 1, 6: 1, 7: 1, 9: 1})
        [1, 2, 3, 4, 5, 6, 7, 8, 9]
        [2, 1, 1, 2, 1, 1, 1, 2, 1]
```

```
In [10]: # show dictionary example
         d = [1, 2, 3, 4, 5, 6, 7, 9, 8]
         Counter(d).items()
         # in case repetition is similar, take the the minimum value from the repetition (key - original)
Out[10]: dict_items([(1, 1), (2, 1), (3, 1), (4, 1), (5, 1), (6, 1), (7, 1), (9, 1), (8, 1)])
```

```
In [11]: def calculate_mode(data_values):
             data_counter = Counter(data_values)
             max_freq = max(list(data_counter.values()))

             if max_freq == 1:
                 return "Mode doesn't exist"

             mode = [i for i, j in data_counter.items() if j == max_freq]
             return min(mode)
```

```
In [12]: # show example
         x = [1, 1, 2, 3, 4, 4, 4, 4, 5, 6, 7, 8, 8, 9, 8]
         mode_value = calculate_mode(data_values=x)
         print(mode_value)

         4
```

Standard deviation

Formula

$$\sigma = \sqrt{\frac{\sum(x_i - \mu)^2}{N}} \rightarrow i = 1, 2, 3, \dots n$$

where

- σ = Standard deviation
- x_i = each data value
- μ = Mean
- N = Total size of the data

```
In [13]: def calculate_stddev(data_values):
             return np.std(a=data_values)
```

```
In [14]: # show example

         x_list = [1, 1, 2, 3, 4, 4, 5, 6, 7]
         std_v = calculate_stddev(data_values=x_list)
         print(std_v)

         2.0
```

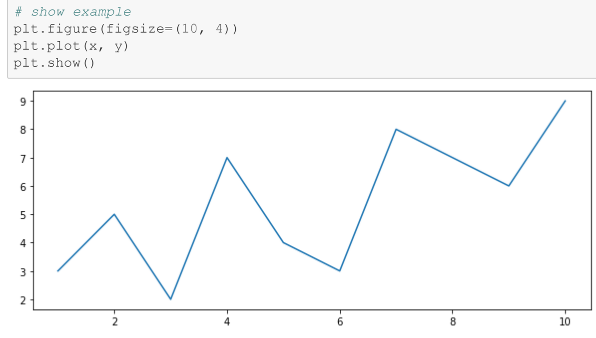
```
In [ ]:
```

Data visualization

```
In [15]: from matplotlib import pyplot as plt
```

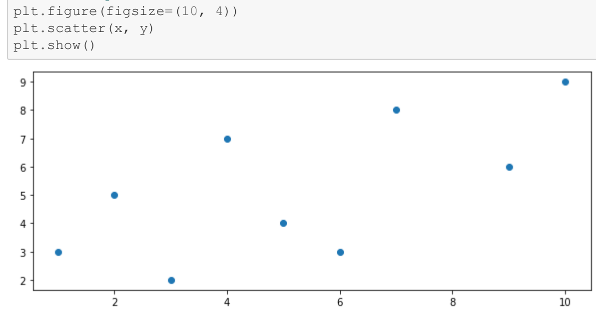
Line Plot

```
In [16]: x = [1, 2, 3, 4, 5, 6, 7, 9, 10]
         y = [3, 5, 2, 7, 4, 3, 8, 6, 9]

         # show example
         plt.figure(figsize=(10, 4))
         plt.plot(x, y)
         plt.show()
```



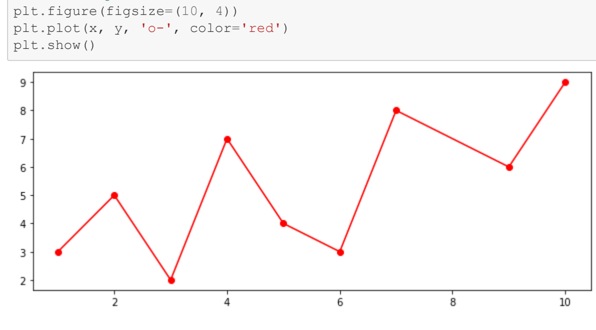
Scatter Plot

```
In [17]: x = [1, 2, 3, 4, 5, 6, 7, 9, 10]
         y = [3, 5, 2, 7, 4, 3, 8, 6, 9]

         # show example
         plt.figure(figsize=(10, 4))
         plt.scatter(x, y)
         plt.show()
```



Line and Scatter together

```
In [18]: x = [1, 2, 3, 4, 5, 6, 7, 9, 10]
         y = [3, 5, 2, 7, 4, 3, 8, 6, 9]

         # show example
         plt.figure(figsize=(10, 4))
         plt.plot(x, y, 'o-', color='red')
         plt.show()
```



Handwritten plots

```
In [19]: # x = [1, 2, 3, 4, 5, 6, 7, 9, 10]
         # y = [3, 5, 2, 7, 4, 3, 8, 6, 9]

         # # show example
         # plt.figure(figsize=(10, 4))
         # plt.xkcd()
         # plt.plot(x, y, 'o-r')
         # plt.show()
```

Read data

```
In [20]: df = pd.read_csv('students_hw.csv')
         df.head()
Out[20]:
```

	Height(Inches)	Weight(Pounds)
0	65.78	112.99
1	71.52	136.49
2	69.40	153.03
3	68.22	142.34
4	67.79	144.30

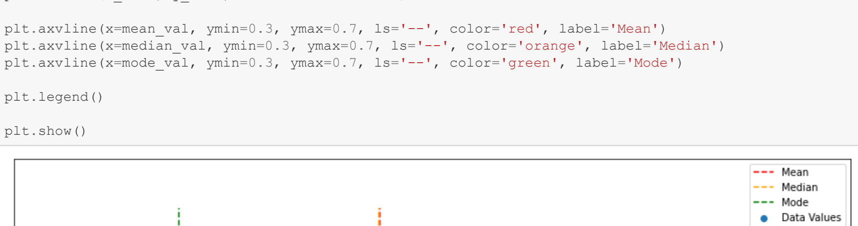
Heights → Mean, Median, Mode

```
In [21]: x_list = df['Height(Inches)'].to_list()
         y_list = [0 for i in range(len(x_list))]
```

```
In [22]: print(x_list)

         [65.78, 71.52, 69.4, 68.22, 67.79, 68.7, 69.8, 70.01, 67.9, 66.78, 66.49, 67.62, 68.3, 67.12, 68.28,
         71.09, 66.46, 68.65, 71.23, 67.13, 67.83, 68.88, 63.48, 68.42, 67.63, 67.21, 70.84, 67.49, 66.53, 65.
         44, 69.52, 65.81, 67.82, 70.6, 71.8, 69.21, 66.8, 67.66, 67.81, 64.05, 68.57, 65.18, 69.66, 67.97, 6
         5.98, 68.67, 66.88, 67.7, 69.82, 69.09, 69.91, 67.33, 70.27, 69.1, 65.38, 70.18, 70.41, 66.54, 66.36,
         67.54, 66.5, 69.0, 68.3, 67.01, 70.81, 68.22, 69.06, 67.73, 67.22, 67.37, 65.27, 70.84, 69.92, 64.29,
         68.25, 66.36, 68.36, 65.48, 69.72, 67.73, 68.64, 66.78, 70.05, 66.28, 69.2, 69.13, 67.36, 70.09, 70.1
         8, 68.23, 68.13, 70.24, 71.49, 69.2, 70.06, 70.56, 66.29, 63.43, 66.77, 68.89, 64.87, 67.09, 68.35, 6
         5.61, 67.76, 68.02, 67.66, 66.31, 69.44, 63.84, 67.72, 70.05, 70.19, 65.95, 70.01, 68.61, 68.81, 69.7
         6, 65.46, 68.83, 65.8, 67.21, 69.42, 68.94, 67.94, 65.63, 66.5, 67.93, 68.89, 70.24, 68.27, 71.23, 6
         9.1, 64.4, 71.1, 68.22, 65.92, 67.44, 73.9, 69.98, 69.52, 65.18, 68.01, 68.34, 65.18, 68.26, 68.57, 6
         4.5, 68.71, 68.89, 69.54, 67.4, 66.48, 66.01, 72.44, 64.13, 70.98, 67.5, 72.02, 65.31, 67.08, 64.39,
         69.37, 68.38, 65.31, 67.14, 68.39, 66.29, 67.19, 65.99, 69.43, 67.97, 67.76, 65.28, 73.83, 66.81, 66.
         89, 65.74, 65.98, 66.58, 67.11, 65.87, 66.78, 68.74, 66.23, 65.96, 68.58, 66.59, 66.97, 68.08, 70.19,
         65.52, 67.46, 67.41, 69.66, 65.8, 66.11, 68.24, 68.02, 71.39]
```

```
In [23]: mean_val = calculate_mean(data_values=x_list)
         median_val = calculate_median(data_values=x_list)
         mode_val = calculate_mode(data_values=x_list)
         print(mean_val)
         print(median_val)
         print(mode_val)

         67.94979999999998
         67.935
         65.18
```

```
In [24]: plt.figure(figsize=(15, 3))

         plt.yticks([])

         plt.scatter(x_list, y_list, label='Data Values')

         plt.axvline(x=mean_val, ymin=0.3, ymax=0.7, ls='--', color='red', label='Mean')
         plt.axvline(x=median_val, ymin=0.3, ymax=0.7, ls='--', color='orange', label='Median')
         plt.axvline(x=mode_val, ymin=0.3, ymax=0.7, ls='--', color='green', label='Mode')

         plt.legend()

         plt.show()
```



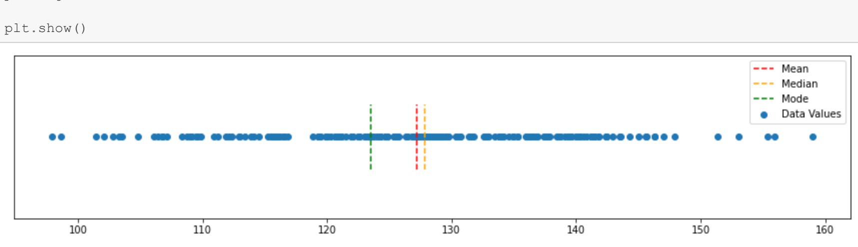

```
In [ ]:
```

Weights → Mean, Median, Mode

```
In [25]: x_list = df['Weight(Pounds)'].to_list()
         y_list = [0 for i in range(len(x_list))]
```

```
In [26]: mean_val = calculate_mean(data_values=x_list)
         median_val = calculate_median(data_values=x_list)
         mode_val = calculate_mode(data_values=x_list)
         print(mean_val)
         print(median_val)
         print(mode_val)

         127.2219500000001
         127.875
         123.49
```

```
In [27]: plt.figure(figsize=(15, 3))

         plt.yticks([])

         plt.scatter(x_list, y_list, label='Data Values')

         plt.axvline(x=mean_val, ymin=0.3, ymax=0.7, ls='--', color='red', label='Mean')
         plt.axvline(x=median_val, ymin=0.3, ymax=0.7, ls='--', color='orange', label='Median')
         plt.axvline(x=mode_val, ymin=0.3, ymax=0.7, ls='--', color='green', label='Mode')

         plt.legend()

         plt.show()
```




```
In [ ]:
```

Outliers

- An outlier is a data point that differs significantly from other data values

```
In [28]: x = [1, 2, 3, 4, 5, 6, 7, 9, 10, 50, 5, 6, 9, 4, 7]
         y = [3, 5, 2, 7, 4, 3, 8, 6, 9, 65, 6, 8, 3, 6, 9]
```

How can we detect outliers?

1. By graphing
   - scatter plot
   - box plot
2. By calculating z_score values
   - if z_score value is $> 3 \rightarrow$ reject
   - if z_score value is $< -3 \rightarrow$ reject

Formula

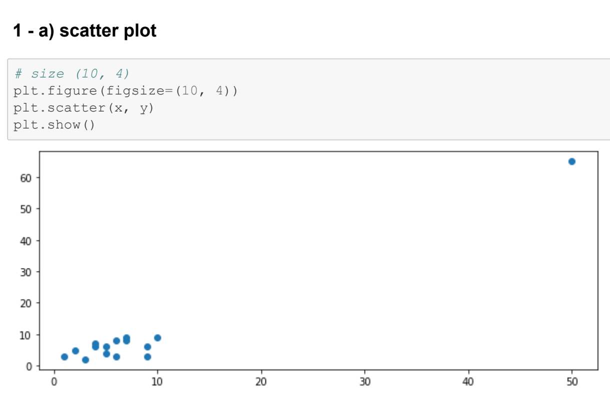
$$z = \frac{(x_i - \mu)}{\sigma} \rightarrow i = 1, 2, 3 \dots n$$

where

- μ = Mean
- σ = Standard deviation
- x_i = each data value

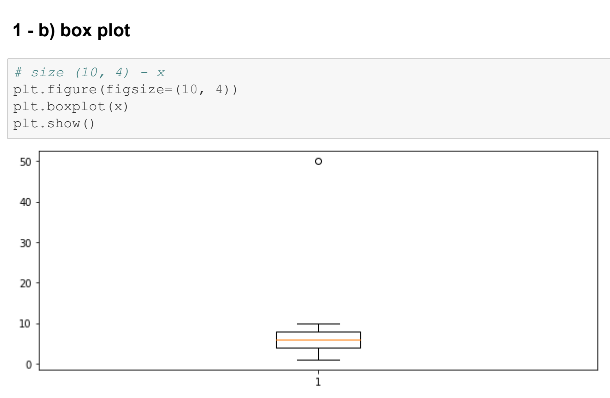
1 - a) scatter plot

```
In [29]: # size (10, 4)
         plt.figure(figsize=(10, 4))
         plt.scatter(x, y)
         plt.show()
```

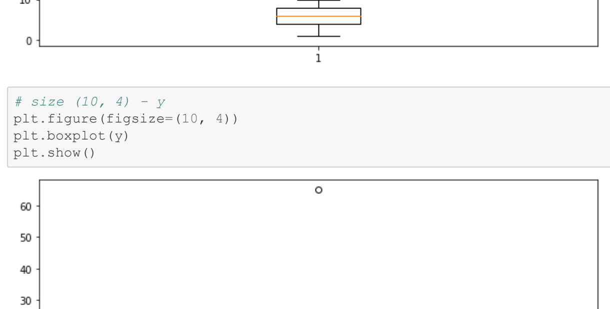


1 - b) box plot

```
In [30]: # size (10, 4) - x
         plt.figure(figsize=(10, 4))
         plt.boxplot(x)
         plt.show()
```



```
In [31]: # size (10, 4) - y
         plt.figure(figsize=(10, 4))
         plt.boxplot(y)
         plt.show()
```



2 - zscore method

```
In [32]: def calculate_zscore(data_values):
             # mean
             mean_vals = calculate_mean(data_values)

             # standard deviation
             std_dev = calculate_stddev(data_values=data_values)

             # applying the formula for all the values
             zscore = [(i - mean_vals)/std_dev for i in data_values]

             return zscore
```

```
In [33]: z_x = calculate_zscore(data_values=x)
         print(z_x)

         [-0.6631473670024701, -0.5751189554534697, -0.4870905439044692, -0.3990621323554687, -0.3110337208064
         683, -0.2230053092574678, -0.13497689770846735, 0.041079925389533554, 0.12910833693853402, 3.65024479
         88985525, -0.3110337208064683, -0.2230053092574678, 0.041079925389533554, -0.3990621323554687, -0.134
         97689770846735]
```

```
In [34]: def get_outlier_indices(data_values):
             # z_score values of all the data values
             z_score = calculate_zscore(data_values=data_values)

             # get the index of the outlier
             # whose value is > 3
             # whose value is < -3
             outlier_indices = [i for i, j in enumerate(z_score) if j > 3 or j < -3]

             return outlier_indices
```

```
In [35]: x_index = get_outlier_indices(data_values=x)
         print(x_index)

         [9]
```

```
In [36]: y_index = get_outlier_indices(data_values=y)
         print(y_index)

         [9]
```

```
In [37]: z_x = x[x_index[0]]

In [38]: z_x
Out[38]: 50
```

```
In [39]: heights = df['Height(Inches)'].to_list()
         weights = df['Weight(Pounds)'].to_list()
```

```
In [40]: out_heights = get_outlier_indices(data_values=heights)
         print(out_heights)

         [138, 174]
```

```
In [41]: heights[138]
Out[41]: 73.9

In [42]: heights[174]
Out[42]: 73.83
```

```
In [43]: out_weights = get_outlier_indices(data_values=weights)
         print(out_weights)

         []
```

```
In [ ]:
```

What did we learn?

- Statistical Measurements
- Python libraries for DA
- Visualization
- Visualization of the dataset
- Outlier detection