Today's Agenda

- Python Libraries for DA
- Codes and Examples of various statistical measurements
  - Blog Link → https://msameeruddin.hashnode.dev/5-statistical-measures-for-exploratory-data-analysis
- Data Visualization
- Outlier Detection
  - Blog Link → https://msameeruddin.hashnode.dev/6-how-to-detect-outliers-like-a-detective

Python Tools for Data Analysis

SciPy
NumPy
matplotlib
IP[y]: IPython
Interactive Computing
pandas
scikit
learn

Credits - Image from Internet

Other Libraries

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

Basic Statistics

The detailed blog on the same → https://msameeruddin.hashnode.dev/5-statistical-measures-for-exploratory-data-analysis

- **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

Pandas Profiling Example

In [1]:

```
# import pandas_profiling as pp
# import pandas as pd

# source = 'https://raw.githubusercontent.com/msameeruddin/Data-Analysis-Python/main/4_DA_Visualization/stud
# df = pd.read_csv(source)
# # ---------
# profile = pp.ProfileReport(df)
# profile.to_file("output.html")
```

Important Packages

- Installation process

```
py -m pip install scipy numpy pandas matplotlib plotly pandas-profiling statsmodels --user
```

`import` Packages

In [2]:

```
import pandas as pd
import numpy as np
from scipy import stats
```

Mean

In [3]:

```
# show example
data = [5, 6, 1, -10, 4, 8, 10]
mean_val = np.mean(data)
print(mean_val)
```

```
3.4285714285714284
```

Median

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 [4]:

```
# show example
data = [5, 6, 1, -10, 4, 8, 10, 9, 100, 32]
median_val = np.median(data)
print(median_val)
```

```
7.0
```

Mode

In [5]:

```
data = [1, 1, 2, 3, 4, 4, 4, 4, 4, 5, 6, 7, 8, 8, 8, 9, 8, 8]
mode_val = stats.mode(data)
print(mode_val)
```

```
ModeResult(mode=array([4]), count=array([5]))
```

In [6]:

```
mode_val = stats.mode(data)[0]
print(mode_val)
```

```
[4]
```

In [7]:

```
mode_val = stats.mode(data)[0][0]
print(mode_val)
```

```
4
```

Standard Deviation

Formula

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

where

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

In [8]:

```python
data = [5, 6, 1, -10, 4, 8, 10]
std_val = np.std(data)
print(std_val)
```

```
6.091144458665098
```

- variance → σ^2
- median absolute deviation
- covariance
- correlation

Data Visualization

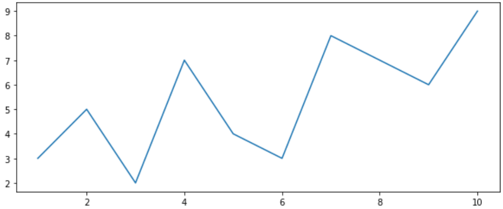
In [9]:

```python
from matplotlib import pyplot as plt
```

Line Plot

In [10]:

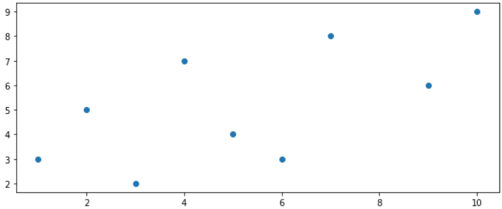
```python
x = [1, 2, 3, 4, 5, 6, 7, 9, 10]
y = [3, 5, 2, 7, 4, 3, 8, 6, 9]

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



Scatter Plot

In [11]:

```python
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

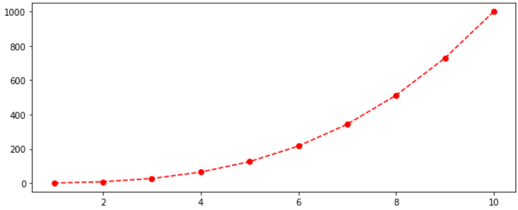
In [12]:

```
x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
y = [i**3 for i in x]

print(x)
print(y)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]
```

In [13]:

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



Read Data

- data source - https://bit.ly/2RU48GX

In [14]:

```
data_source = 'https://bit.ly/2RU48GX'
df = pd.read_csv(data_source)
df.head()
```

Out[14]:

	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 [15]:

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

In [16]:

```
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, 65.98, 68.67, 66.88, 6
7.7, 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, 6
7.01, 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, 69.7
2, 67.73, 68.64, 66.78, 70.05, 66.28, 69.2, 69.13, 67.36, 70.09, 70.18, 68.23, 68.13, 70.24, 71.49, 69.2, 7
0.06, 70.56, 66.29, 63.43, 66.77, 68.89, 64.87, 67.09, 68.35, 65.61, 67.76, 68.02, 67.66, 66.31, 69.44, 63.8
4, 67.72, 70.05, 70.19, 65.95, 70.01, 68.61, 68.81, 69.76, 65.46, 68.83, 65.8, 67.21, 69.42, 68.94, 67.94, 6
5.63, 66.5, 67.93, 68.89, 70.24, 68.27, 71.23, 69.1, 64.4, 71.1, 68.22, 65.92, 67.44, 73.9, 69.98, 69.52, 6
5.18, 68.01, 68.34, 65.18, 68.26, 68.57, 64.5, 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, 67.7
6, 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 [17]:

```python
print(y_list)
```

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

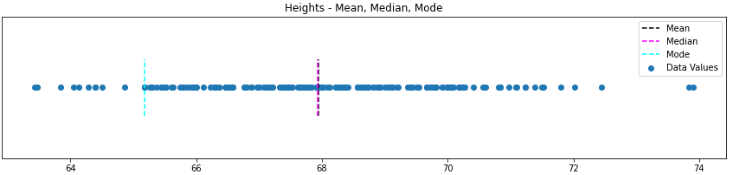
In [18]:

```python
mean_val = np.mean(x_list)
median_val = np.median(x_list)
mode_val = stats.mode(x_list)[0][0]
print(mean_val)
print(median_val)
print(mode_val)
```

```
67.9498
67.935
65.18
```

In [19]:

```python
plt.figure(figsize=(15, 3))

plt.yticks([])

plt.title("Heights - Mean, Median, Mode")

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

plt.axvline(x=mean_val, ymin=0.3, ymax=0.7, ls='--', color='black', label='Mean')
plt.axvline(x=median_val, ymin=0.3, ymax=0.7, ls='--', color='magenta', label='Median')
plt.axvline(x=mode_val, ymin=0.3, ymax=0.7, ls='--', color='cyan', label='Mode')

plt.legend()

plt.show()
```



Weights → Mean, Median, Mode

In [20]:

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

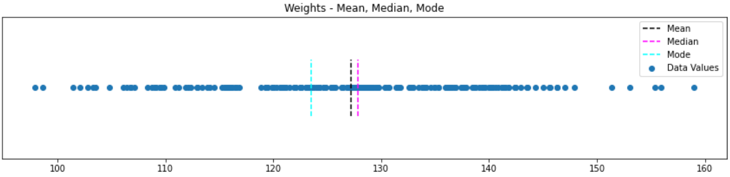
In [21]:

```python
mean_val = np.mean(x_list)
median_val = np.median(x_list)
mode_val = stats.mode(x_list)[0][0]
print(mean_val)
print(median_val)
print(mode_val)
```

```
127.22194999999999
127.875
123.49
```

In [22]:

```
plt.figure(figsize=(15, 3))

plt.yticks([])

plt.title("Weights - Mean, Median, Mode")

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

plt.axvline(x=mean_val, ymin=0.3, ymax=0.7, ls='--', color='black', label='Mean')
plt.axvline(x=median_val, ymin=0.3, ymax=0.7, ls='--', color='magenta', label='Median')
plt.axvline(x=mode_val, ymin=0.3, ymax=0.7, ls='--', color='cyan', label='Mode')

plt.legend()

plt.show()
```



Outliers

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

In [23]:

```
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?

The detailed blog on the same → https://msameeruddin.hashnode.dev/6-how-to-detect-outliers-like-a-detective

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

Formula

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

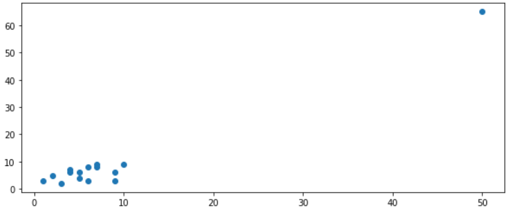
where

- μ = mean
- σ = standard deviation
- x_i = each data value

1 - a) scatter plot

In [24]:

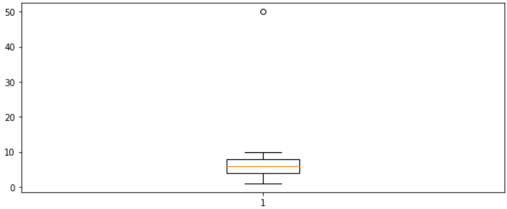
```
# size (10, 4)
plt.figure(figsize=(10, 4))
plt.scatter(x, y)
plt.show()
```



1 - b) box plot

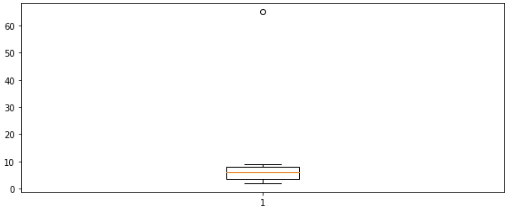
In [25]:

```
# size (10, 4) - x
plt.figure(figsize=(10, 4))
plt.boxplot(x)
plt.show()
```



In [26]:

```
# size (10, 4) - y
plt.figure(figsize=(10, 4))
plt.boxplot(y)
plt.show()
```



2 - zscore method

In [27]:

```
def calculate_zscores(data_values):
    # array
    data_array = np.array(data_values)

    # mean
    mean_val = np.mean(data_array)

    # standard deviation
    std_dev = np.std(data_array)

    # applying the formula for all the values
    z_scores = (data_array - mean_val) / std_dev

    return z_scores
```

In [28]:

```
z_x = calculate_zscores(data_values=x)
print(z_x)
```

```
[-0.66314737 -0.57511896 -0.48709054 -0.39906213 -0.31103372 -0.22300531
 -0.1349769   0.04107993  0.12910834  3.6502448  -0.31103372 -0.22300531
  0.04107993 -0.39906213 -0.1349769 ]
```

In [29]:

```
def get_outlier_vals(data_values):
    # z_score values of all the data values
    z_scores = calculate_zscores(data_values=data_values)

    # get the index of the outlier
    # whose value is > 3
    # whose value is < -3
    inds = list(np.where(z_scores < -3)[0])
    inds.extend(list(np.where(z_scores > 3)[0]))

    outlier_vals = [data_values[i] for i in sorted(inds)]

    return outlier_vals
```

In [30]:

```
out_x = get_outlier_vals(data_values=x)
print(out_x)
```

```
[50]
```

In [31]:

```
out_y = get_outlier_vals(data_values=y)
print(out_y)
```

```
[65]
```

In [32]:

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

In [33]:

```
out_heights = get_outlier_vals(data_values=heights)
print(out_heights)
```

```
[73.9, 73.83]
```

In [34]:

```
out_weights = get_outlier_vals(data_values=weights)
print(out_weights)
```

```
[]
```

What did we learn?

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