# COURSE CONTENT

## LECTURE SESSION - 2

- Introduction to Python/Numpy/Pandas
- Introduction to EDA and Visualizations
- Python hands on exercises

# Python

- Simple programming language to learn
- Yet very powerful. Used in the following industries:
  - Data Science, Machine learning and Deep learning
  - IoT, Arduino, etc.
  - Desktop application development
  - Web applications
- Kept simple to avoid wasting time on cumbersome syntax and language complexities like with Java, .NET, C++
- Used by engineers and scientists to implement their innovation quickly
- Provides a rich set of libraries
- Production ready application

# Installation of Python

- Download Python 3.6 or higher from python.org
- Or Download Anaconda Framework and install
- Or Go to Google Colab and use the notebook from your google account

# Common python libraries for Data Analytics

- NumPy – handling multi-dimensional arrays
- Pandas – Array Series & DataFrames
- Matplotlib, Seaborn – Visualization
- Scipy – Statistical package

# Primitive Data types

- Integer

  ```
  x = 100
  ```
- Float

  ```
  pi = 3.1415
  ```
- String

  ```
  msg = “Hello World”
  ```
- Logical

  ```
  isSuccess = True
  ```

# Structured Data Types in Python

- Apart from data types like int, string, float Python has the below data types which are very useful for data science
  - List

```
arr1 = [ ‘Red’, ‘Green’, ‘Orange’ ]
```

  - Tuples

```
stud = (1092, ‘Albert’, 86.8, ‘PASS’)
```

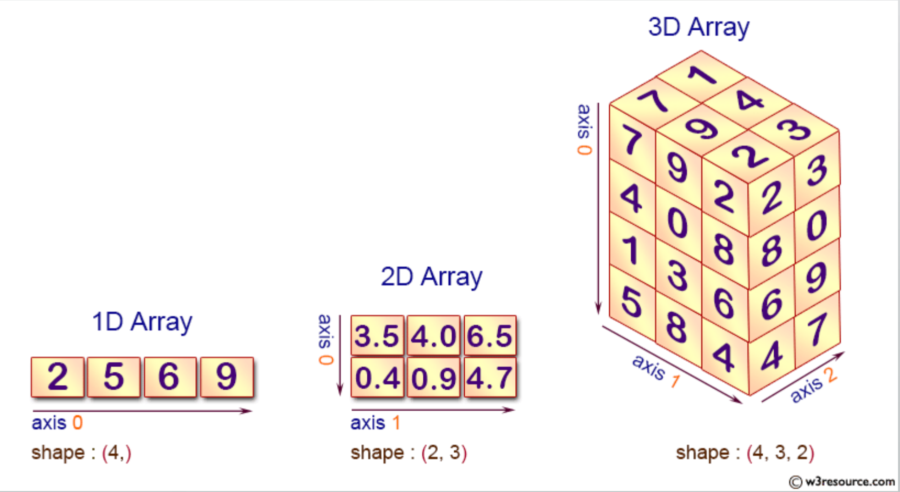
  - Dictionaries

```
planet = {
    “planet”: “Mercury”,
    “moons”: 0,
    “diameter”: 4879
}
```

# INTRO TO NUMPY

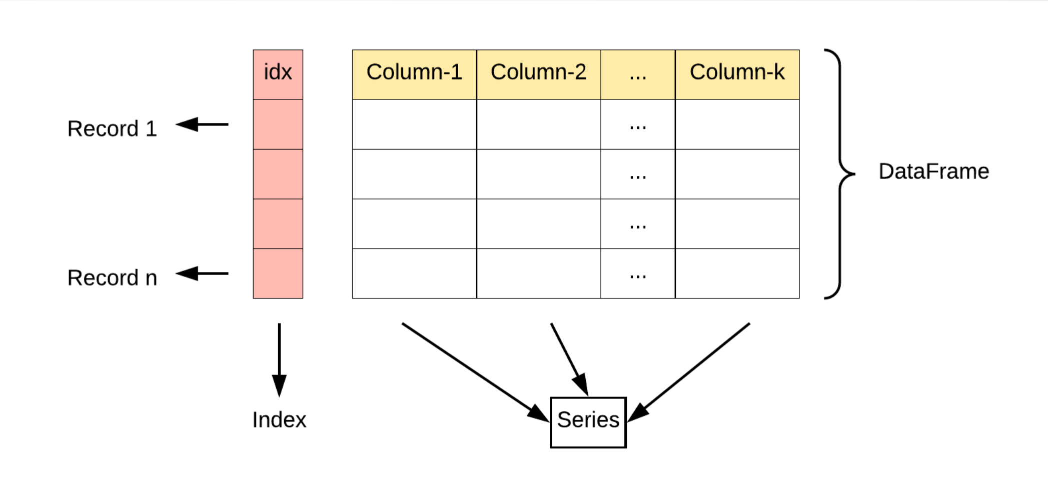
**Numpy** is the basic package for scientific computing with Python. Salient features of numpy:

- **A powerful N-dimensional array object – ndarray**
- **Helpful functions, that eases array operations**
- **Faster than primitive array structure**
- **Used in Linear algebra, Matrix, Fourier transform etc.**



# PANDAS

- A library in Python for data manipulation and analysis
- It offers data structures and operations for manipulating numerical tables and data frames
- Contains two important classes:
  - Series
  - DataFrame
- Meant for storing spreadsheet kind of data



# Case Study

- **Iris Flower Data Analysis**

  To identify the Iris flower species based on a few characteristics of the flower such as Sepal Length, Sepal Width, Petal Length and Petal Width

- **Dataset**

  The dataset contains the above said attributes and the target label is the Species type as a category

| SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm | Species |
|---|---|---|---|---|
| 6.5 | 3.2 | 5.1 | 2 | Iris-virginica |
| 6.1 | 2.8 | 4 | 1.3 | Iris-versicolor |
| 5.1 | 3.5 | 1.4 | 0.3 | Iris-setosa |
| 6.4 | 3.1 | 5.5 | 1.8 | Iris-virginica |
| 6.7 | 3.1 | 4.7 | 1.5 | Iris-versicolor |
| 4.8 | 3.4 | 1.9 | 0.2 | Iris-setosa |
| 6.1 | 2.8 | 4.7 | 1.2 | Iris-versicolor |
| 5.8 | 2.7 | 5.1 | 1.9 | Iris-virginica |

# EDA – Exploratory data analysis

- Import numpy, pandas,matplotlib.pyplot, seaborn packages
- Get the data and read it into a DataFrame
- Perform Univariate analysis
  - Explore the data for non-null and extreme values
  - Populate the null values with interpolation and clean up
  - Find the skewness, frequency distribution
- Perform Bivariate and Multivariate analysis
  - Find the correlation between columns with Pearson correlation coefficient
  - Do a pair plot to visualize the distribution
  - Remove the redundant columns and reduce the dimensionalty

# Example: Using DataFrame, Series & array on a data set

```
import pandas as pd

df1 = pd.read_csv('iris.csv')
df1.head()
```

| | Id | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm | Species |
|---|---|---|---|---|---|---|
| 0 | 1 | 6.5 | 3.2 | 5.1 | 2.0 | Iris-virginica |
| 1 | 2 | 6.1 | 2.8 | 4.0 | 1.3 | Iris-versicolor |
| 2 | 3 | 5.1 | 3.5 | 1.4 | 0.3 | Iris-setosa |
| 3 | 4 | 6.4 | 3.1 | 5.5 | 1.8 | Iris-virginica |
| 4 | 5 | 6.7 | 3.1 | 4.7 | 1.5 | Iris-versicolor |

```
type(df1['SepalLengthCm'].head())
```

```
pandas.core.series.Series
```

```
df1['Species'].head()
```

```
0     Iris-virginica
1    Iris-versicolor
2        Iris-setosa
3     Iris-virginica
4    Iris-versicolor
Name: Species, dtype: object
```

```
list( df1['SepalLengthCm'].head() )
```

```
[6.5, 6.1, 5.1, 6.4, 6.7]
```

```
df1.head().to_dict()
```

```
{'Id': {0: 1, 1: 2, 2: 3, 3: 4, 4: 5},
 'SepalLengthCm': {0: 6.5, 1: 6.1, 2: 5.1, 3: 6.4, 4: 6.7},
 'SepalWidthCm': {0: 3.2, 1: 2.8, 2: 3.5, 3: 3.1, 4: 3.1},
 'PetalLengthCm': {0: 5.1, 1: 4.0, 2: 1.4, 3: 5.5, 4: 4.7},
 'PetalWidthCm': {0: 2.0, 1: 1.3, 2: 0.3, 3: 1.8, 4: 1.5},
 'Species': {0: 'Iris-virginica',
  1: 'Iris-versicolor',
  2: 'Iris-setosa',
  3: 'Iris-virginica',
  4: 'Iris-versicolor'}}
```

# Application of groupby( )

- Similar to pivot tables in excel

| Id | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm | Species |
|---|---|---|---|---|---|
| 45 | 5.8 | 4 | 1.2 | 0.2 | Iris-setosa |
| 66 | 5.7 | 4.4 | 1.5 | 0.4 | Iris-setosa |
| 99 | 5.7 | 3.8 | 1.7 | 0.3 | Iris-setosa |
| 28 | 5.5 | 3.5 | 1.3 | 0.2 | Iris-setosa |

- What is the mean of the Sepal length, width and Petal length, width for each Species of the flower?

```
df1.groupby(by='Species').mean()
```

| Species | Id | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm |
|---|---|---|---|---|---|
| Iris-setosa | 83.50 | 5.006 | 3.418 | 1.464 | 0.244 |
| Iris-versicolor | 66.76 | 5.936 | 2.770 | 4.260 | 1.326 |
| Iris-virginica | 76.24 | 6.588 | 2.974 | 5.552 | 2.026 |

- What is the largest Sepal Length for Setosa?

```
max_df = df1.groupby(by='Species').max()
max_df.loc['Iris-setosa'].SepalLengthCm

```

```
5.8
```

# Merge vs Join operations in DataFrame

- **Merge – Links two DFs matching by a unique column identifier**
- **Join – Links two DFs by their matching index values**

| | City | Country_Code | Population(Mil) |
|---|---|---|---|
| 0 | Tokyo | JA | 37.84 |
| 1 | Jakarta | ID | 30.53 |
| 2 | Delhi | IN | 24.99 |
| 3 | Manila | PH | 24.13 |
| 4 | Seoul | KR | 23.48 |
| 5 | Shanghai | CN | 23.41 |
| 6 | Karachi | PK | 22.12 |
| 7 | Beijing | CN | 21.00 |
| 8 | Mumbai | IN | 17.70 |
| 9 | Chongqing | CN | 15.70 |

| | Country_Code | Country_Name |
|---|---|---|
| 0 | JA | Japan |
| 1 | ID | Indonesia |
| 2 | IN | India |
| 3 | PH | Philippines |
| 4 | KR | S.Korea |
| 5 | CN | China |
| 6 | PK | Pakistan |
| 7 | SG | Singapore |
| 8 | MY | Malaysia |

**3.4.2 Join operation**

```
city_df.join(ctry_df, lsuffix='_city', rsuffix='_ctry')  # how='inr
# Appends the columns based on index key
```

| | City | Country_Code_city | Population(Mil) | Country_Code_ctry | Country_Name |
|---|---|---|---|---|---|
| 0 | Tokyo | JA | 37.84 | JA | Japan |
| 1 | Jakarta | ID | 30.53 | ID | Indonesia |
| 2 | Delhi | IN | 24.99 | IN | India |
| 3 | Manila | PH | 24.13 | PH | Philippines |
| 4 | Seoul | KR | 23.48 | KR | S.Korea |
| 5 | Shanghai | CN | 23.41 | CN | China |
| 6 | Karachi | PK | 22.12 | PK | Pakistan |
| 7 | Beijing | CN | 21.00 | SG | Singapore |
| 8 | Mumbai | IN | 17.70 | MY | Malaysia |
| 9 | Chongqing | CN | 15.70 | NaN | NaN |

**3.4.3 Merge operation**

```
pd.merge(city_df, ctry_df, on='Country_Code', sor
```

| | City | Country_Code | Population(Mil) | Country_Name |
|---|---|---|---|---|
| 0 | Shanghai | CN | 23.41 | China |
| 1 | Beijing | CN | 21.00 | China |
| 2 | Chongqing | CN | 15.70 | China |
| 3 | Jakarta | ID | 30.53 | Indonesia |
| 4 | Delhi | IN | 24.99 | India |
| 5 | Mumbai | IN | 17.70 | India |

# Introduction to Visualization

Data visualization is an important skill in applied statistics and machine learning.

- It provides an important suite of tools for gaining a qualitative understanding. This can be helpful when exploring and getting to know a dataset and can help with identifying patterns, corrupt data, outliers, and much more.
- Visualization is the most important aspect of exploratory data analysis (EDA)

# Matplotlib, Seaborn and Plotly

**Matplotlib**

- The matplotlib is a popular graphical subroutine and is used widely for data visualization applications.
- The matplotlib provides a context, one in which one or more plots can be drawn before the image is shown or saved to file. The context can be accessed via functions on pyplot. There is some convention to import this context and alias it as plt.

```
import matplotlib.pyplot as plt
```

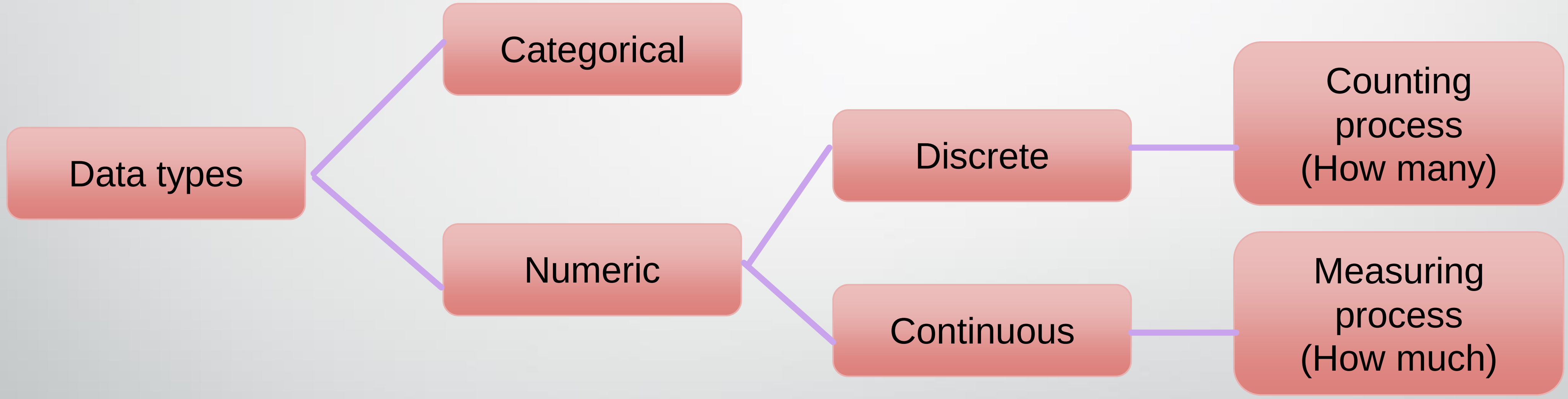
# Seaborn

Seaborn is complementary to Matplotlib and it specifically targets statistical data visualization.

But it goes even further than that: Seaborn extends Matplotlib and that's why it can address the frustrations of working with Matplotlib.

Matplotlib tries to make easy things easier and hard things possible. Seaborn tries to make a well-defined set of hard things easy too.

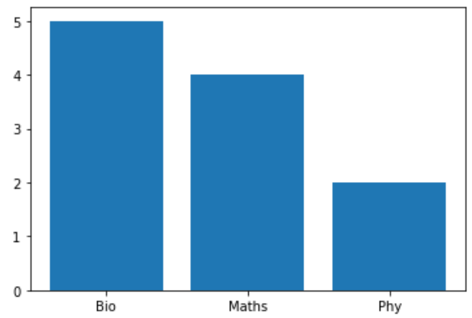
# Types of data



# Different types of plots

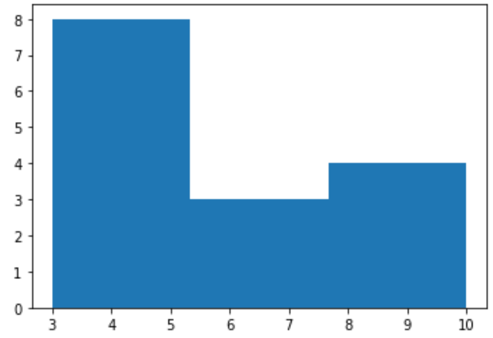
- Line Plot
- Bar Chart
- Histogram Plot
- Box and Whisker Plot
- Scatter Plot

```python
stnames = [ 'Anna', 'Basheer', 'Charan', 'David', 'Emily', 'Feroz',
            'Ganesh', 'Hanifa', 'Irfan', 'Jane', 'Kamal' ]
subject = [ 'Bio', 'Maths', 'Phy', 'Bio', 'Maths', 'Bio',
            'Maths', 'Phy', 'Bio', 'Maths', 'Bio']

df_st = pd.DataFrame( { 'Names': stnames, 'Major': subject } )

stats = df_st.groupby(by='Major').count()
stats.columns = ['Count']

plt.bar(stats.index, stats['Count']);
```

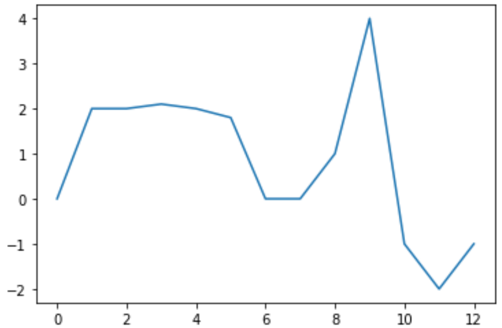



```python
# Histogram
children_ages = [8, 10, 3, 5, 4, 6, 9, 4, 5, 10, 7, 4, 3, 6, 5]
plt.hist(children_ages, bins=3)
# Range=(10 - 3)=7, No. of bins = 3, Bin size = 7/3 = 2.33
# So bin array = [ 3, 3+2.33, 3+2.33+2.33, 10]
```

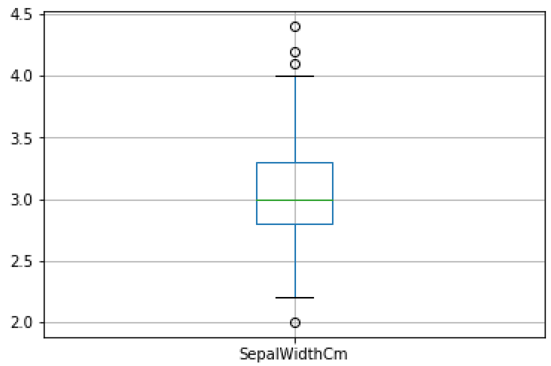
```
(array([8., 3., 4.]),
 array([ 3.        ,  5.33333333,  7.66666667, 10.        ]),
 <a list of 3 Patch objects>)
```



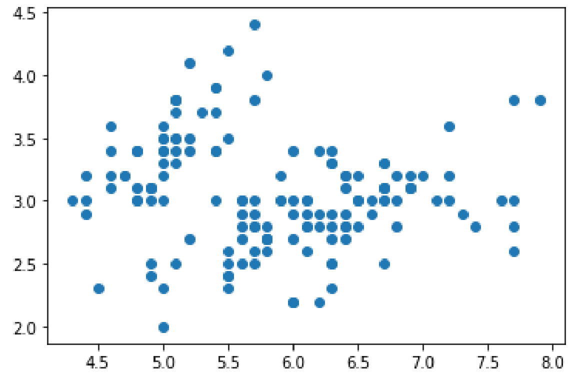
```python
import matplotlib.pyplot as plt

signal = [0, 2, 2, 2.1, 2, 1.8, 0, 0, 1, 4, -1, -2, -1, ];
plt.plot(signal);
plt.show();
```



```python
df1.boxplot( column='SepalWidthCm'  )
plt.show();
```



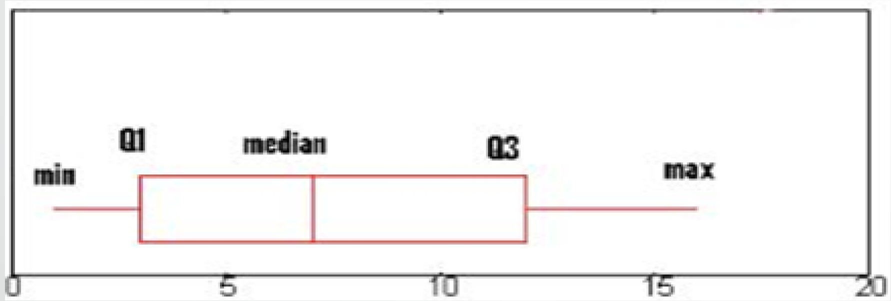

```python
plt.scatter(df1['SepalLengthCm'], df1['SepalWidthCm'])
plt.show();
```

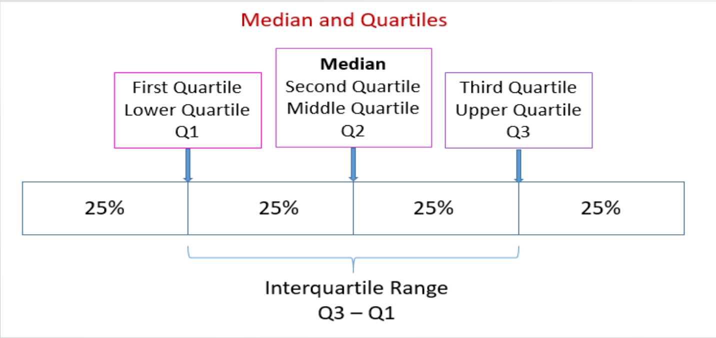


# Practical use cases of various visualization techniques

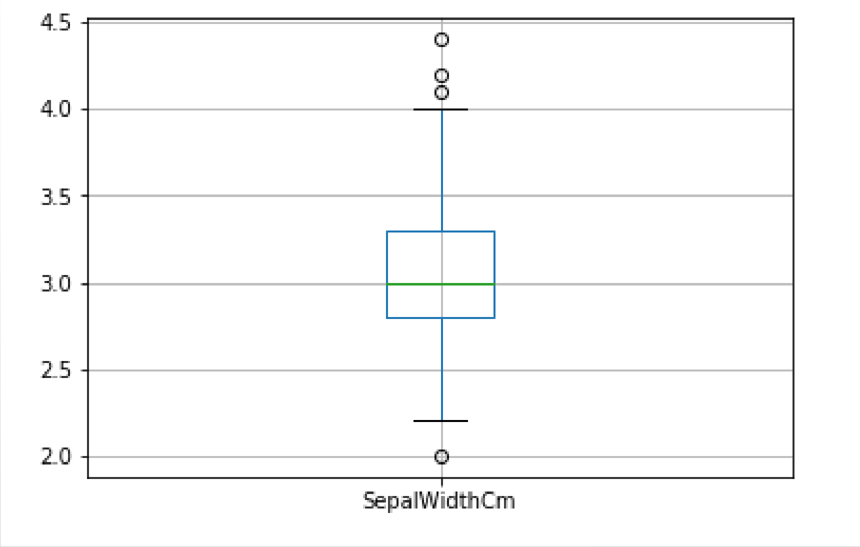
## Box plot

A box plot helps in understanding the distribution of the data at hand. It gives us an understanding of the skewness of the data and provides five-point summary of the data.




```
df1.boxplot( column='SepalWidthCm'  )
plt.show();
```

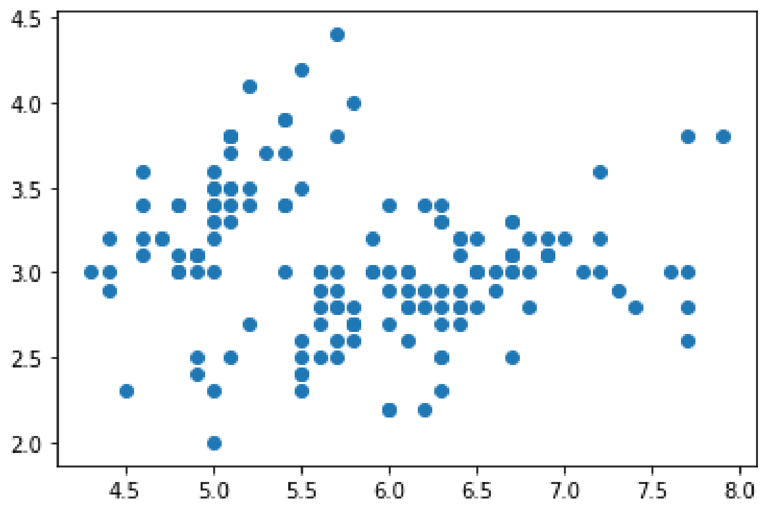


# Practical use cases of various visualization techniques

## Scatter plot

- Relationship between customer age and average call duration in a telecom customer churn dataset
- How width of the petal changes with the length

```
plt.scatter(df1['SepalLengthCm'], df1['SepalWidthCm'])
plt.show();
```

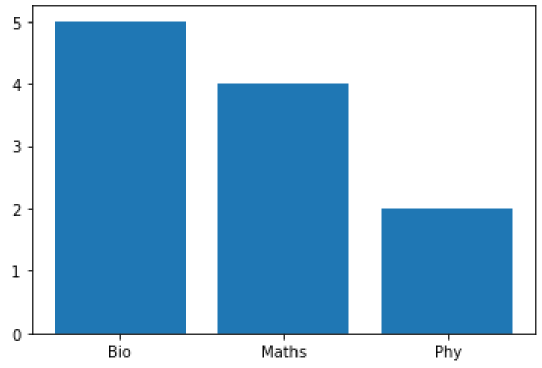


# Practical use cases of various visualization techniques

## Bar plot

- Population statistics between various groups
- Count of different groups

```
stnames = [ 'Anna', 'Basheer', 'Charan', 'David', 'Emily', 'Feroz',
            'Ganesh', 'Hanifa', 'Irfan', 'Jane', 'Kamal' ]
subject = [ 'Bio', 'Maths', 'Phy', 'Bio', 'Maths', 'Bio',
            'Maths', 'Phy', 'Bio', 'Maths', 'Bio']

df_st = pd.DataFrame( { 'Names': stnames, 'Major': subject } )

stats = df_st.groupby(by='Major').count()
stats.columns = ['Count']

plt.bar(stats.index, stats['Count']);
```



# CREDITS

1. Great learning
2. University of Texas at Austin