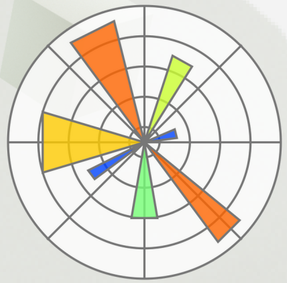


Kampus Merdeka X MyEduSolve

# Python Week #8

DATA SCIENCE A - TEAM 3

# Outline

Pandas

- Install & Import
- Reading tabular data
- Create a dataframe object
- Attributes, indexing, and slicing
- Descriptive statistics
- Filtering
- Aggregation and combining
- Data type conversion and missing value handling

Matplotlib

- Install & Import
- Basic plot


# Pandas

**Pandas is a Python library focused on data analysis. Often referred to as library data analysis because there are many modules to process data.**

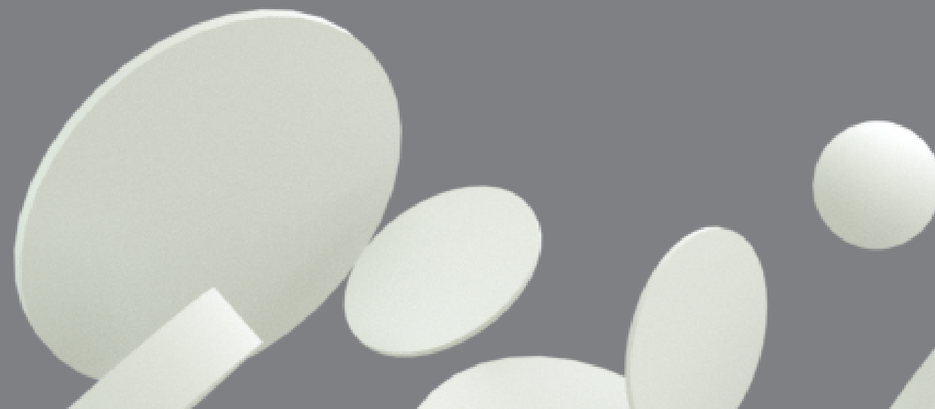
## Install

using pip:
pip install pandas

using anaconda:
conda install pandas

## Import

import pandas as pd



# Pandas

## Read Tabular Data

## df = pd.read_csv(csv_file_name)


**pandas supports many file formats examples are csv, excel, sql, json, and parquet. If you want to access a file format, you only need to use read_(file format).**

# Pandas

## Create a dataframe object

Dataframes are two-dimensional, size-mutable, potentially heterogeneous tabular data.

Syntax

pandas.DataFrame(data=None, index=None, columns=None, dtype=None, copy=None)

**Examples: Constructing DataFrame from a dictionary.**

```
>>> d = {'col1': [1, 2], 'col2': [3, 4]}
>>> df = pd.DataFrame(data=d)
>>> df
   col1  col2
0     1     3
1     2     4
```

# Pandas

## Attributes

Attributes are similar to methods, but they are used to assist in finding information about data.

### pandas attribute example

**Attributes**

| | |
|---|---|
| `at` | Access a single value for a row/column label pair. |
| `attrs` | Dictionary of global attributes of this dataset. |
| `axes` | Return a list representing the axes of the DataFrame. |
| `columns` | The column labels of the DataFrame. |

### How to use attributes

```
df.columns
```

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'Sib
Sp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

# Pandas

## Indexing and Slicing

Indexing is similar to an address, where the address is useful to assist us in accessing data. Indexing in pandas can be done by **.loc**, **.iloc**, and **[]**.

Slicing is the selection of a set of rows and/or columns from a DataFrame. Slicing can be used with the **[start:stop]** operator.

## Examples of using Indexing

| Object Type | Indexers |
| --- | --- |
| Series | `s.loc[indexer]` |
| DataFrame | `df.loc[row_indexer,column_indexer]` |

```
df.iloc[2:4]
```

| | PassengerId | Survived | Pclass | Name |
| --- | --- | --- | --- | --- |
| **2** | 3 | 1 | 3 | Heikkinen, Miss. Laina |
| **3** | 4 | 1 | 1 | Futrelle, Mrs. Jacques Heath (Lily May Peel) |

# Pandas

## Descriptive Statistics

Descriptive statistics are a summary of the central tendency, dispersion, and distribution of the data set, excluding NaN values.

### How to use Descriptive Statistics

DataFrame.describe(
percentiles=None,
include=None, exclude=None,
datetime_is_numeric=False)

### Examples of using Descriptive Statistics

```
>>> s = pd.Series(['a', 'a', 'b', 'c'])
>>> s.describe()
count      4
unique     3
top        a
freq       2
dtype: object
```

# Pandas

## Filtering

Pandas makes it easy to modify data. We can do filtering according to the conditions we want in a fairly easy way, namely by slicing on the desired conditions. Examples of parameters for filtering are **items**, **like**, and **regex**.

### Examples of using Filter

```
>>> df = pd.DataFrame(np.array(([1, 2, 3], [4, 5, 6])),
...                   index=['mouse', 'rabbit'],
...                   columns=['one', 'two', 'three'])
>>> df
        one  two  three
mouse     1    2      3
rabbit    4    5      6
```

```
>>> # select columns by name
>>> df.filter(items=['one', 'three'])
        one  three
mouse     1      3
rabbit    4      6
```

# Pandas
## Aggregation

Aggregation can make it possible to apply functions to be executed along any of the DataFrame's axes.

### How to use Aggregation

DataFrame.aggregate(func=None, axis=0, *args, **kwargs)

### Examples of using Aggregation

```
df = pd.DataFrame([[1, 2, 3],
                   [4, 5, 6],
                   [7, 8, 9],
                   [np.nan, np.nan, np.nan]],
                  columns=['A', 'B', 'C'])
```

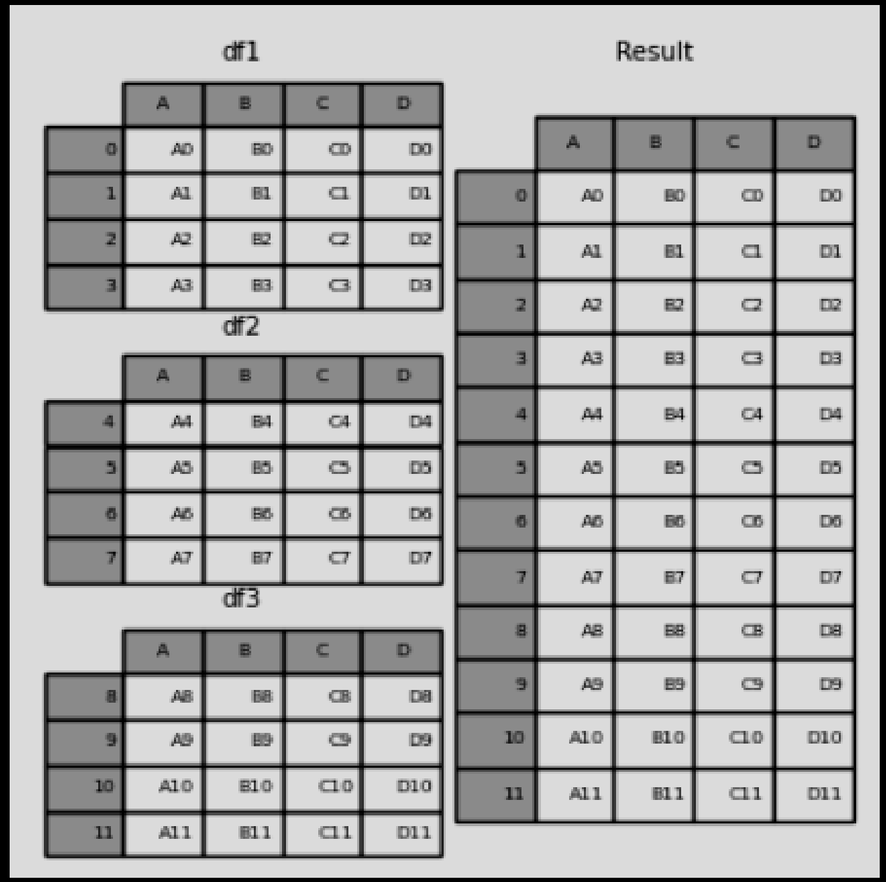
```
>>> df.agg(['sum', 'min'])
        A     B     C
sum  12.0  15.0  18.0
min   1.0   2.0   3.0
```

# Pandas

## Combining

```
In [1]: df1 = pd.DataFrame(
   ...:     {
   ...:         "A": ["A0", "A1", "A2", "A3"],
   ...:         "B": ["B0", "B1", "B2", "B3"],
   ...:         "C": ["C0", "C1", "C2", "C3"],
   ...:         "D": ["D0", "D1", "D2", "D3"],
   ...:     },
   ...:     index=[0, 1, 2, 3],
   ...: )
   ...:

In [2]: df2 = pd.DataFrame(
   ...:     {
   ...:         "A": ["A4", "A5", "A6", "A7"],
   ...:         "B": ["B4", "B5", "B6", "B7"],
   ...:         "C": ["C4", "C5", "C6", "C7"],
   ...:         "D": ["D4", "D5", "D6", "D7"],
   ...:     },
   ...:     index=[4, 5, 6, 7],
   ...: )
   ...:

In [3]: df3 = pd.DataFrame(
   ...:     {
   ...:         "A": ["A8", "A9", "A10", "A11"],
   ...:         "B": ["B8", "B9", "B10", "B11"],
   ...:         "C": ["C8", "C9", "C10", "C11"],
   ...:         "D": ["D8", "D9", "D10", "D11"],
   ...:     },
   ...:     index=[8, 9, 10, 11],
   ...: )
   ...:

In [4]: frames = [df1, df2, df3]

In [5]: result = pd.concat(frames)
```

Pandas has data merging features like **join**, **append**, **merge**, **concat**, **pivot_table**, etc.




| | A | B | C | D |
|---|---|---|---|---|
| 0 | A0 | B0 | C0 | D0 |
| 1 | A1 | B1 | C1 | D1 |
| 2 | A2 | B2 | C2 | D2 |
| 3 | A3 | B3 | C3 | D3 |

| | A | B | C | D |
|---|---|---|---|---|
| 4 | A4 | B4 | C4 | D4 |
| 5 | A5 | B5 | C5 | D5 |
| 6 | A6 | B6 | C6 | D6 |
| 7 | A7 | B7 | C7 | D7 |

| | A | B | C | D |
|---|---|---|---|---|
| 8 | A8 | B8 | C8 | D8 |
| 9 | A9 | B9 | C9 | D9 |
| 10 | A10 | B10 | C10 | D10 |
| 11 | A11 | B11 | C11 | D11 |

| | A | B | C | D |
|---|---|---|---|---|
| 0 | A0 | B0 | C0 | D0 |
| 1 | A1 | B1 | C1 | D1 |
| 2 | A2 | B2 | C2 | D2 |
| 3 | A3 | B3 | C3 | D3 |
| 4 | A4 | B4 | C4 | D4 |
| 5 | A5 | B5 | C5 | D5 |
| 6 | A6 | B6 | C6 | D6 |
| 7 | A7 | B7 | C7 | D7 |
| 8 | A8 | B8 | C8 | D8 |
| 9 | A9 | B9 | C9 | D9 |
| 10 | A10 | B10 | C10 | D10 |
| 11 | A11 | B11 | C11 | D11 |

# Pandas

## Data Type Conversion and Missing Value Handling

This is the process of checking each data type in the column in the DataFrame, if there is data that is not correct (according to the data description), then we must change the data type as a form of handling.

**Example**: The "Survived" column should be of object data type, but the existing data is integer data type, so we have to change the data type to match the data description.

```
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
```

```
df["Survived"].replace({0: "Survive", 1: "Death"})
```

```
0       Survive
1         Death
2         Death
3         Death
4       Survive
         ...
886     Survive
887       Death
888     Survive
889       Death
890     Survive
Name: Survived, Length: 891, dtype: object
```

# Missing Value Type

## MAR ( Missing at Random )

the data that may be missing is most likely related to the variable, or usually in the process of sampling / data collection, the respondent intentionally does not answer the question for this variable, either because it is privacy or something else, so it becomes blank.

## MCAR ( Missing Completely at Random )

missing data but not related to a variable, so in most cases this happens because of an error, it could be human error or data corruption.

## MNAR ( Missing Not at Random )

the missing data is not due to an error or related to certain variables such as MCAR or MAR. The occurrence of missing data in a variable is related to the variable itself, so it cannot be predicted from other variables in a dataset. The missing data in this case could indeed be the value of a data (value = Null).

# Matplotlib

Matplotlib is a Python library used for plotting 2D charts and figures.

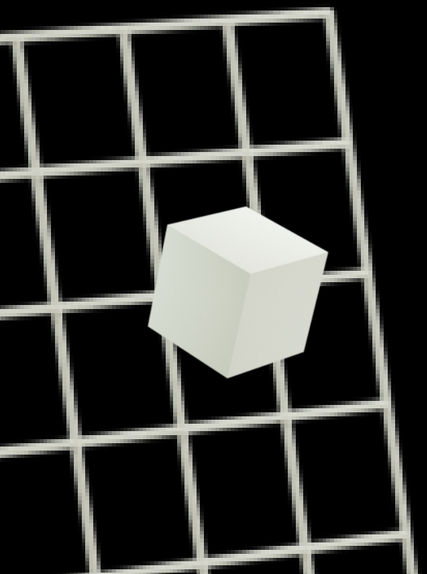
## Install

**Using pip**

python -m pip install -U pip
python -m pip install -U matplotlib

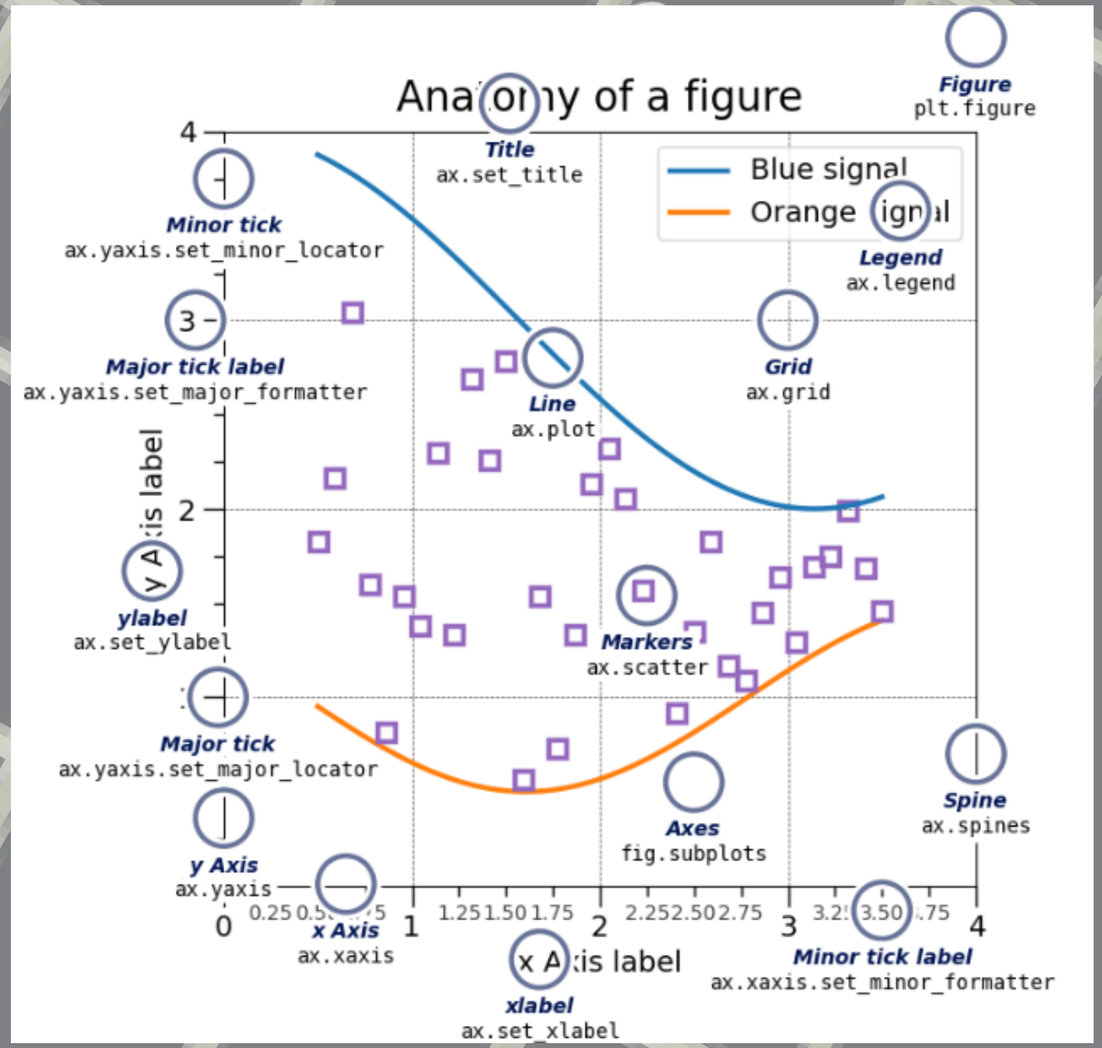
**Using Anaconda**

1. via the anaconda main channel
   conda install matplotlib
2. via the conda-forge community channel
   conda install -c conda-forge matplotlib



## Import

```
import matplotlib.pyplot as plt
%matplotlib inline
```




# Anatomy of a Figure

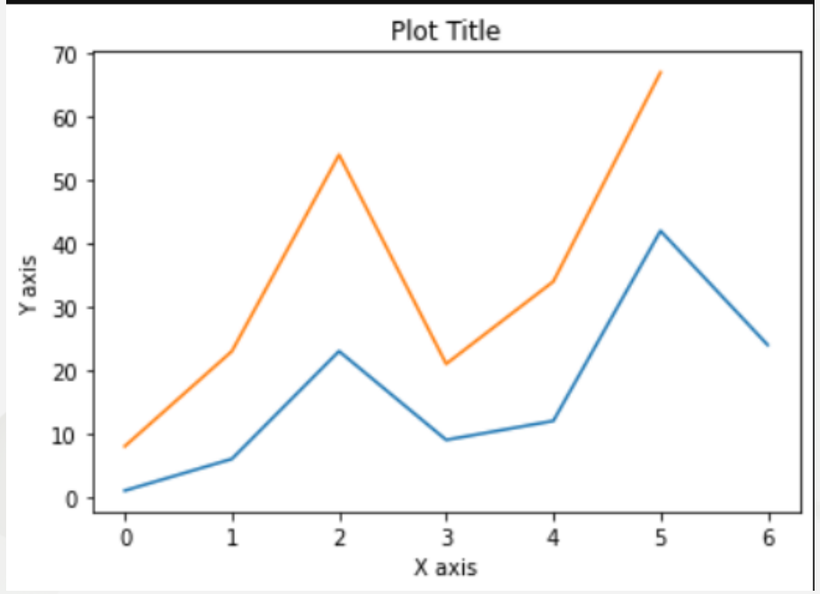
Examples of using functions, methods, classes, and modules:

- matplotlib.pyplot.figure
- matplotlib.axes.Axes.text
- matplotlib.axis.Axis.set_minor_formatter
- matplotlib.axis.Axis.set_major_locator
- matplotlib.axis.Axis.set_minor_locator
- matplotlib.patches.Circle
- matplotlib.patheffects.withStroke
- matplotlib.ticker.FuncFormatter

# Matplotlib

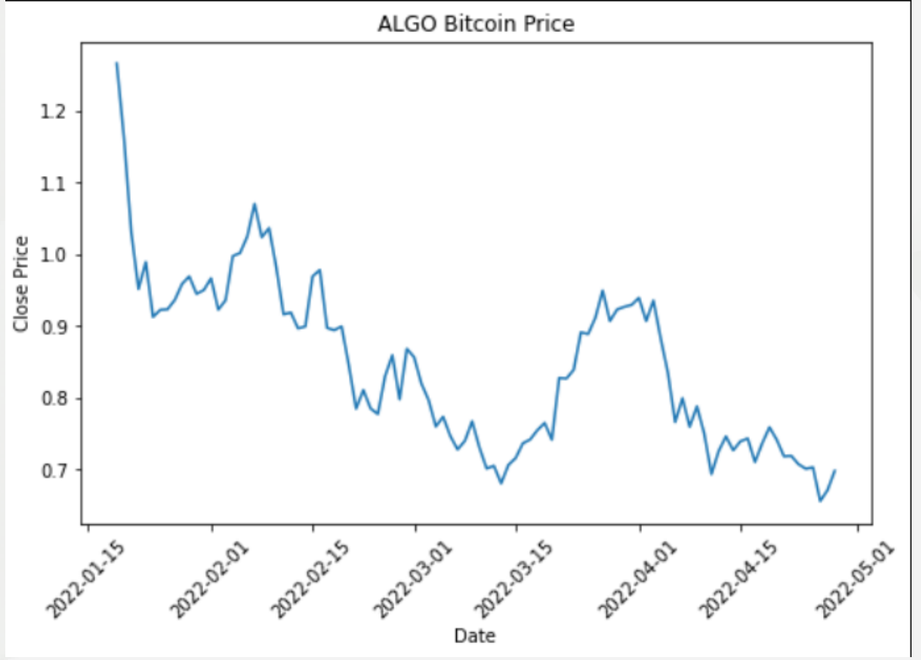
## Line plot for series data

```
first_list = [1, 6, 23, 9, 12, 42, 24]
second_list = [8, 23, 54, 21, 34, 67]
plt.plot(first_list)
plt.plot(second_list)
plt.title("Plot Title")
plt.xlabel("X axis")
plt.ylabel("Y axis")
plt.show()
```




## Line plot for time series data

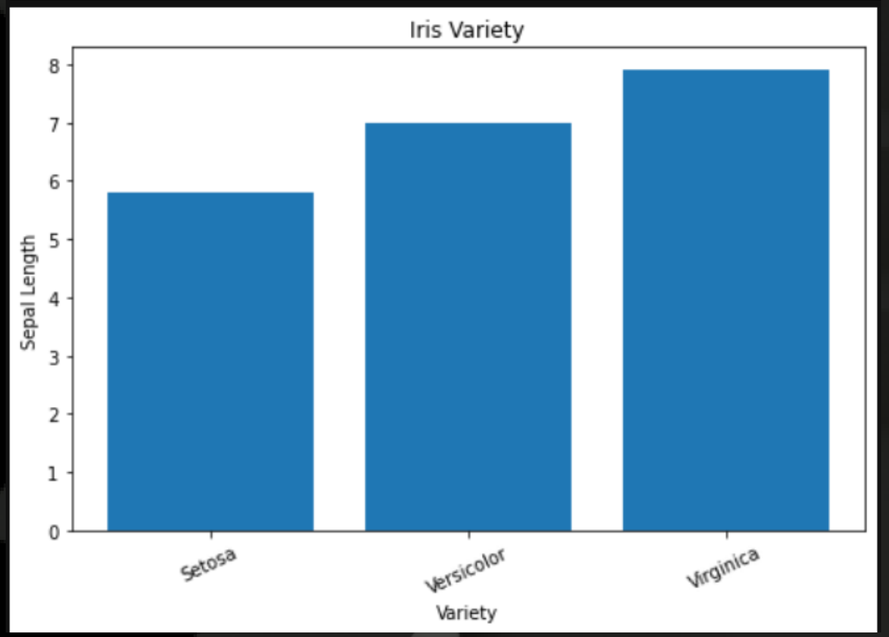
```
plt.figure(figsize=(8, 5))
plt.plot(stock2["date"][:100], stock2["close"][:100])
plt.title("ALGO Bitcoin Price")
plt.xlabel("Date")
plt.ylabel("Close Price")
plt.xticks(rotation=45)
plt.show()
```



# Matplotlib

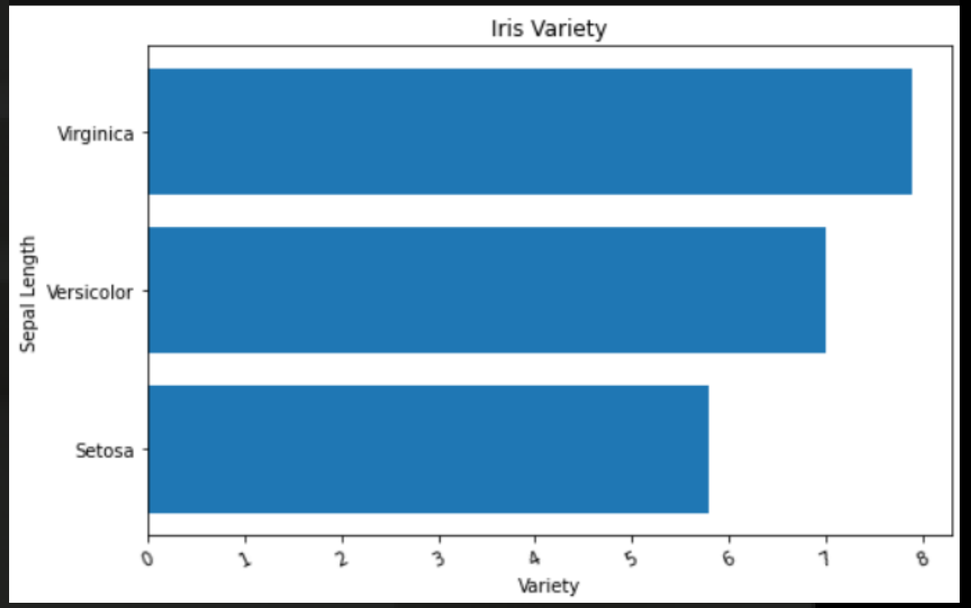
## Vertical bar plot

```
plt.figure(figsize=(8, 5))
plt.bar(iris["variety"], iris["sepal_length"])
plt.title("Iris Variety")
plt.xlabel("Variety")
plt.ylabel("Sepal Length")
plt.xticks(rotation=25)
plt.show()
```




## Horizontal bar plot

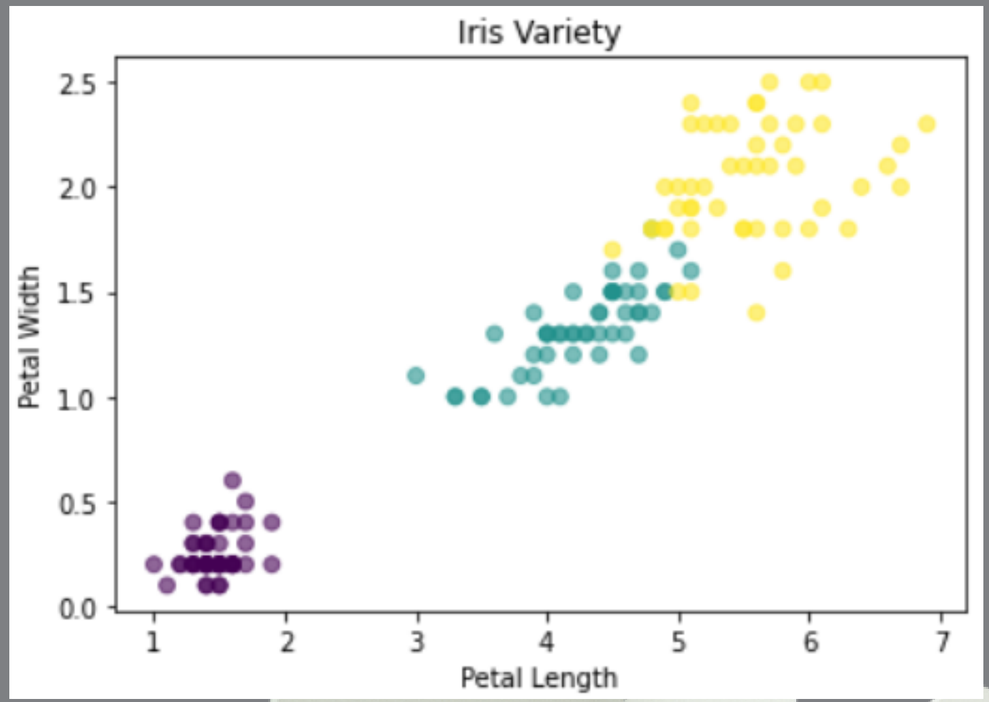
```
plt.figure(figsize=(8, 5))
plt.barh(iris["variety"], iris["sepal_length"])
plt.title("Iris Variety")
plt.xlabel("Variety")
plt.ylabel("Sepal Length")
plt.xticks(rotation=25)
plt.show()
```



# Matplotlib

Scatter Plot

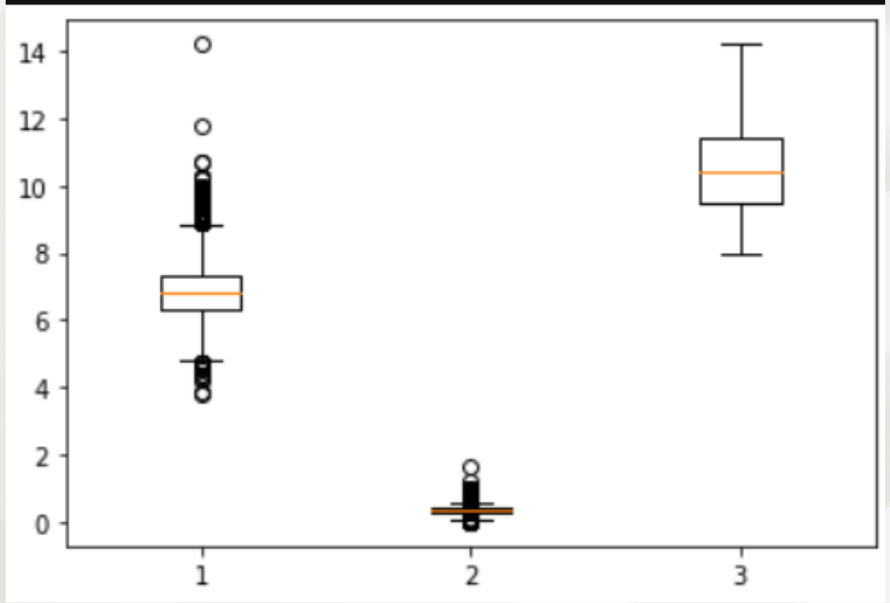
```
plt.scatter(iris["petal_length"],
            iris["petal_width"],
            c=iris["variety_enc"],
            alpha=0.6)
plt.title("Iris Variety")
plt.xlabel("Petal Length")
plt.ylabel("Petal Width")
plt.show()
```



# Matplotlib

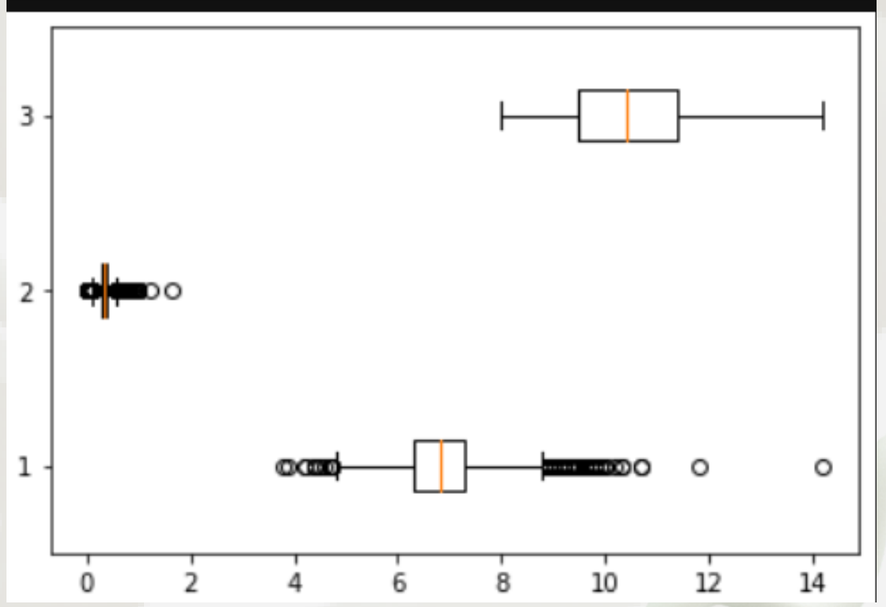
## Vertical Box Plot

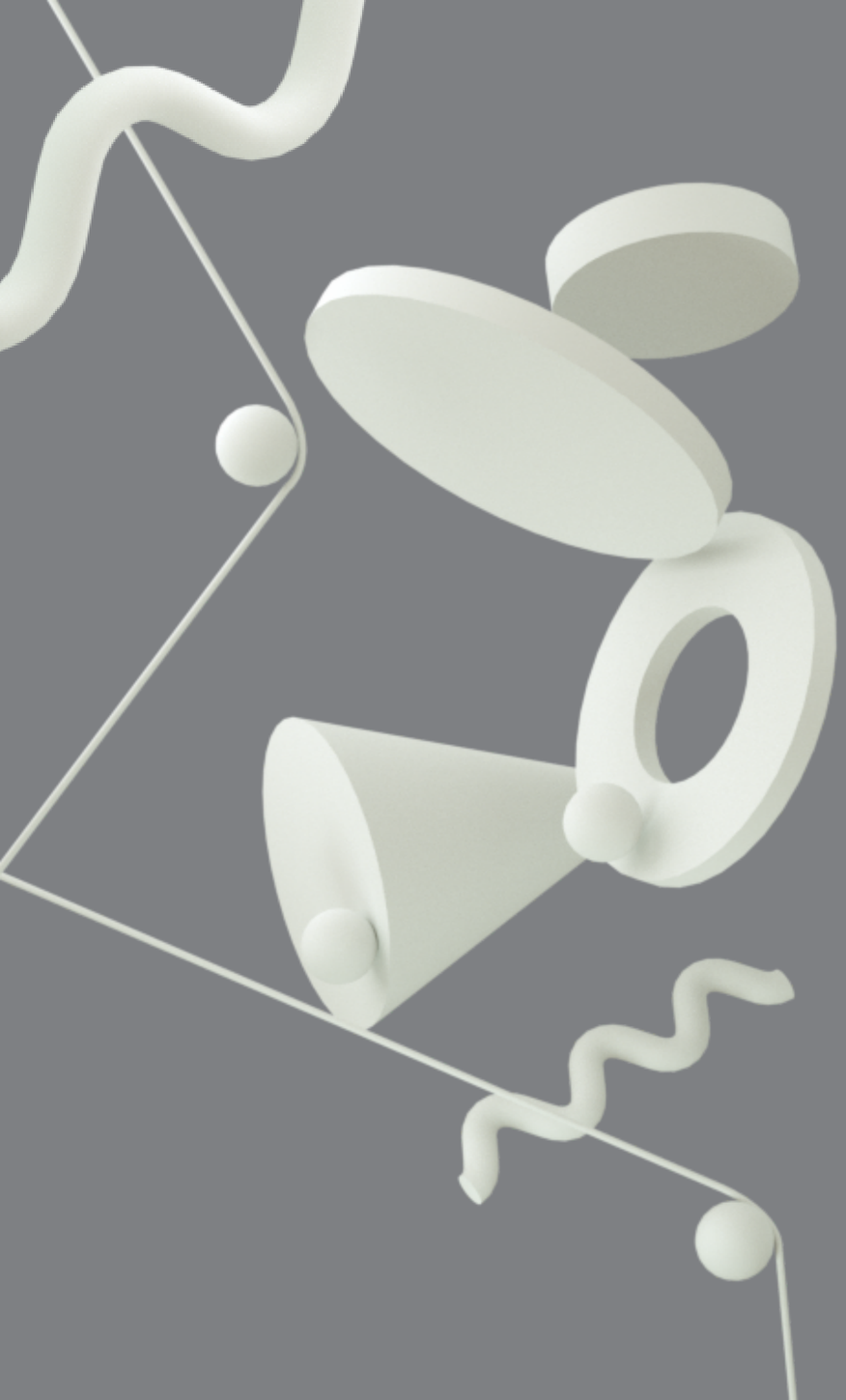
```
plt.boxplot([wine["fixed acidity"],
             wine["citric acid"],
             wine["alcohol"]])
plt.show()
```




## Horizontal Box Plot

```
plt.boxplot([wine["fixed acidity"],
             wine["citric acid"],
             wine["alcohol"]], vert=False)
plt.show()
```

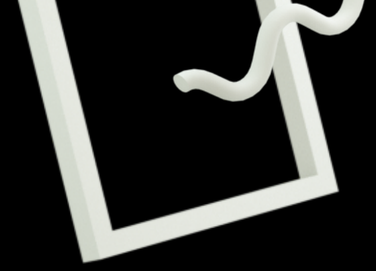




Want to know more about **Pandas** and **Matplotlib** libraries? Visit their official website to see the documentation!

https://pandas.pydata.org/

https://matplotlib.org/



# Thank You!