***1)Give an introduction to plotting using Matplotlib?***

Matplotlib is **a open-source** plotting library for Python.

It is designed to help users **visualize data** in a variety of formats.

It is Developed by **John D. Hunter** in 2003.

it enables users to graphically represent data, facilitating easier analysis and understanding.

It is used to **convert data** into interactive plots and graphs.

**Key Features of Matplotlib**

* **Versatile Plotting**: Create a wide variety of visualizations, including line plots, scatter plots, bar charts, and histograms.
* **Extensive Customization**: Control every aspect of your plots, from colors and markers to labels and annotations.
* **Seamless Integration with NumPy**: Effortlessly plot data arrays directly, enhancing data manipulation capabilities.
* **High-Quality Graphics**: Generate publication-ready plots with precise control over aesthetics.
* **Cross-Platform Compatibility**: Use Matplotlib on Windows, macOS, and Linux without issues.
* **Interactive Visualizations**: Engage with your data dynamically through interactive plotting features.

**Uses of MatplotLib:**

Matplotlib is a Python library for data visualization, primarily used to create static, animated, and interactive plots. It provides a wide range of plotting functions to visualize data effectively.

**Key Uses of Matplotlib:**

* **Basic Plots:**Line plots, bar charts, histograms, scatter plots, etc.
* **Statistical Visualization:** Box plots, error bars, and density plots.
* **Customization:**Control over colors, labels, gridlines, and styles.
* **Subplots & Layouts:**Create multiple plots in a single figure.
* **3D Plotting:**Surface plots and 3D scatter plots using mpl\_toolkits.mplot3d.
* **Animations & Interactive Plots:** Dynamic visualizations with FuncAnimation.
* **Integration:** Works well with Pandas, NumPy and Jupyter Notebooks.

**Applications of MatplotLib:**

Matplotlib is used in

-scientific research,

-finance for market analysis

- data analysis in data science

-education for teaching complex concepts

-engineering for visualizing simulations.

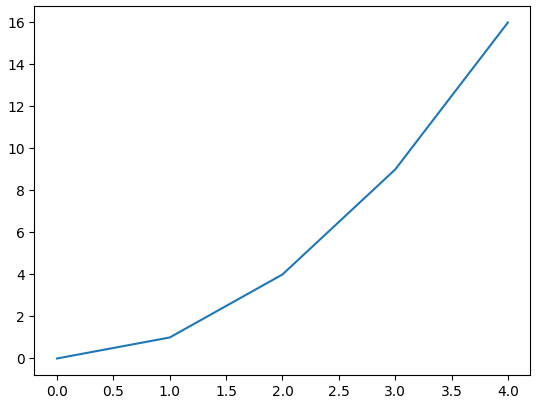
**import** **matplotlib.pyplot** **as** **plt**

x = [0, 1, 2, 3, 4]

y = [0, 1, 4, 9, 16]

plt.plot(x, y)

plt.show()



The parts of a Matplotlib figure include:

* **Figure**: The overarching container that holds all plot elements, acting as the canvas for visualizations.
* **Axes**: The areas within the figure where data is plotted; each figure can contain multiple axes.
* **Axis**: Represents the x-axis and y-axis, defining limits, tick locations, and labels for data interpretation.
* **Lines and Markers**: Lines connect data points to show trends, while markers denote individual data points in plots like scatter plots.
* **Title and Labels**: The title provides context for the plot, while axis labels describe what data is being represented on each axis.

***Explain Plotting using MatplotLib?***

**Different Types of Plots in Matplotlib:**

* 1. Line Graph
* 2. Bar Chart
* 3. Histogram
* 4. Scatter Plot
* 5. Pie Chart
* 6. 3D Plot

**Matplotlib Plotting:Plotting x and y points:**

The **plot()** function is used to draw points (markers) in a diagram.

By default, the plot() function draws a line from point to point.

The function takes parameters for specifying points in the diagram.

Parameter 1 is an array containing the points on the x-axis.

Parameter 2 is an array containing the points on the y-axis.

The parts of a Matplotlib figure include:

* **Figure**: The overarching container that holds all plot elements, acting as the canvas for visualizations.
* **Axes**: The areas within the figure where data is plotted; each figure can contain multiple axes.
* **Axis**: Represents the x-axis and y-axis, defining limits, tick locations, and labels for data interpretation.
* **Lines and Markers**: Lines connect data points to show trends, while markers denote individual data points in plots like scatter plots.
* **Title and Labels**: The title provides context for the plot, while axis labels describe what data is being represented on each axis.

**Linechart:**

*# importing the required libraries*

**import** **matplotlib.pyplot** **as** **plt**

**import** **numpy** **as** **np**

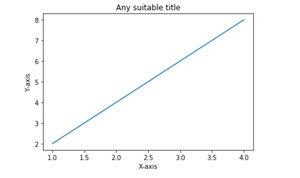
*# define data values*

x = np.array([1, 2, 3, 4]) *# X-axis points*

y = x\*2 *# Y-axis points*

plt.plot(x, y) *# Plot the chart*

plt.show() *# display*



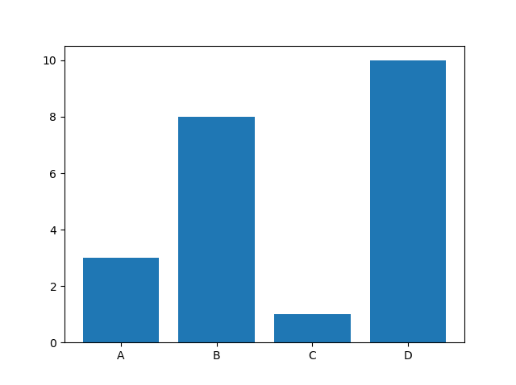
In this example, a simple line chart is generated using [NumPy](https://www.geeksforgeeks.org/numpy-tutorial/)to define data values. The x-values are evenly spaced points, and the y-values are calculated as twice the corresponding x-values.

**Bar chart:**

With Pyplot, you can use the bar() function to draw bar graphs

# Draw 4 bars

import matplotlib.pyplot as plt  
import numpy as np  
  
x = np.array(["A", "B", "C", "D"])  
y = np.array([3, 8, 1, 10])  
  
plt.bar(x,y)  
plt.show()



The bar() function takes arguments that describes the layout of the bars.

The categories and their values represented by the *first*and *second*argument as arrays.

**Histogram:**

A histogram is a graph showing *frequency* distributions.

It is a graph showing the number of observations within each given interval.

In Matplotlib, we use the**hist()**function to create histograms.

A [Histogram](https://www.geeksforgeeks.org/matplotlib-pyplot-hist-in-python/) represents data provided in the form of some groups. It is an accurate method for the graphical representation of numerical data distribution.

**import** matplotlib.pyplot as plt

**import** numpy as np

# Generate random data for the histogram

data **=** np.random.randn(1000)

# Plotting a basic histogram

plt.hist(data, bins**=**30, color**=**'skyblue', edgecolor**=**'black')

# Adding labels and title

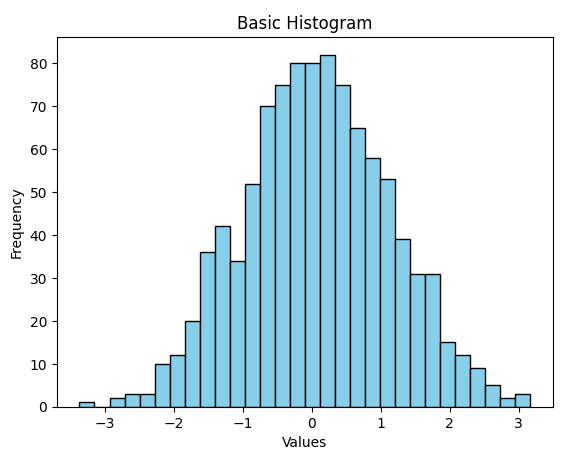
plt.xlabel('Values')

plt.ylabel('Frequency')

plt.title('Basic Histogram')

# Display the plot

plt.show()



**Scatter Chart:**

**matplotlib.pyplot.scatter()** is used to create [scatter plots](https://www.geeksforgeeks.org/scatter-plot/).

It is essential for visualizing relationships between numerical variables.

Scatter plots help illustrate how changes in one variable can influence another, making them invaluable for data analysis.

A basic scatter plot can be created using matplotlib.pyplot.scatter() by plotting two sets of data points on the x and y axes:

**import** **matplotlib.pyplot** **as** **plt**

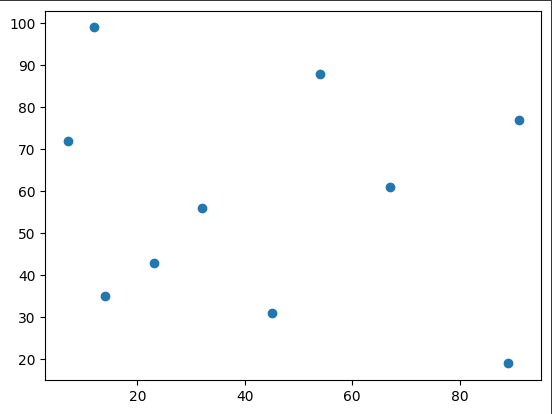
**import** **numpy** **as** **np**

x = np.array([12, 45, 7, 32, 89, 54, 23, 67, 14, 91])

y = np.array([99, 31, 72, 56, 19, 88, 43, 61, 35, 77])

plt.scatter(x, y)

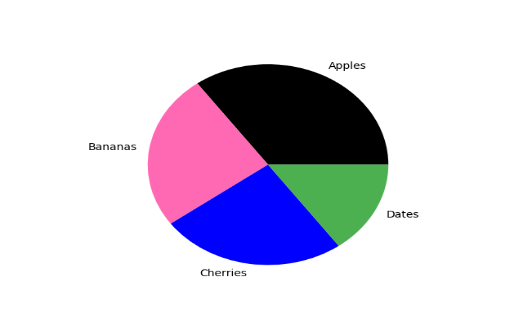
plt.show()



**Pie Chart:**

With Pyplot, you can use the pie() function to draw pie charts.

import matplotlib.pyplot as plt  
import numpy as np  
  
y = np.array([35, 25, 25, 15])  
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]  
mycolors = ["black", "hotpink", "b", "#4CAF50"]  
  
plt.pie(y, labels = mylabels, colors = mycolors)  
plt.show()



the pie chart draws one piece (called a wedge) for each value in the array ([35, 25, 25, 15]).

By default the plotting of the first wedge starts from the x-axis and moves ***counterclockwise***.

***Explain Importing MySQL for Python?***

While working with Python we need to work with databases.

Python can be used in database applications.

Databases may be of different types like MySQL, SQLite, NoSQL, etc.

One of the most popular databases is MySQL.

**MySQL Database:** **You can download a MySQL database at**[**https://www.mysql.com/downloads/**](https://www.mysql.com/downloads/)**.**

**Install MySQL Driver:**

Python needs a MySQL driver "MySQL Connector" to access the MySQL database.

Use PIP to install "MySQL Connector".

PIP is most likely already installed in the Python environment.

Navigate command line to the location of PIP, and type the following:

Download and install "MySQL Connector":

C:\Users\*Your Name*\AppData\Local\Programs\Python\Python36-32\Scripts>python -m pip install mysql-connector-python

**Connecting to the MySQLDatabase:**

The **mysql.connector** provides the **connect()** method used to create a connection between the MySQL database and the Python application.

**Syntax:**

Conn\_obj= mysql.connector.connect(host = <hostname>, user = <username>, passwd = <password>)

**Program:**

# Python program to connect to mysql database

**import** mysql.connector

# Connecting from the server

conn **=** mysql.connector.connect(user **=** 'username',

                                host **=** 'localhost',

password=’password’,

                               # database **=** 'database\_name' # if database is existed)

 print(conn)

# Disconnecting from the server

conn.close()

we can use connection.MySQLConnection() class instead of connect():

# Python program to connect to mysql database

**from** mysql.connector **import** connection

# Connecting to the server

conn **=** connection.MySQLConnection(user **=** 'username',

                               host **=** 'localhost',

                               # database **=** 'database\_name' # if database is existed)

 print(conn)

 # Disconnecting from the server

conn.close()

**Create a Database:**

import mysql.connector  
mydb = mysql.connector.connect(  
  host="localhost",  
  user="*yourusername*",  
  password="*yourpassword*")  
mycursor = mydb.cursor()  
mycursor.execute("CREATE DATABASE mydatabase")

# to show databases

#mycursor.execute("SHOW DATABASES")  
#for x in mycursor:  
 # print(x)

**To create a table** **in MySQL**, use the "CREATE TABLE" statement.

|  |
| --- |
| import mysql.connector mydb = mysql.connector.connect(   host="localhost",   user="*yourusername*",   password="*yourpassword*",   database="mydatabase") mycursor = mydb.cursor() mycursor.execute("CREATE TABLE customers (name VARCHAR(255), address VARCHAR(255))")  #to show tables  #mycursor.execute("SHOW TABLES") #for x in mycursor:  #print(x) |

**To insert into a table:**

To fill a table in MySQL, use the "INSERT INTO" statement.

|  |
| --- |
| import mysql.connector mydb = mysql.connector.connect(   host="localhost",   user="*yourusername*",   password="*yourpassword*",   database="mydatabase" ) mycursor = mydb.cursor() sql = "INSERT INTO customers (name, address) VALUES (%s, %s)" val = ("John", "Highway 21") mycursor.execute(sql, val) **mydb.commit()** print(mycursor.rowcount, "record inserted.") |

**To select from a table:**

To select from a table in MySQL, use the "SELECT" statement.

import mysql.connector  
mydb = mysql.connector.connect(  
  host="localhost",  
  user="*yourusername*",  
  password="*yourpassword*",  
  database="mydatabase")  
mycursor = mydb.cursor()  
mycursor.execute("SELECT \* FROM customers")  
myresult = mycursor.fetchall()  
for x in myresult:  
  print(x)