**Batch: C4 Roll No.: 16010123217**

**Experiment / assignment / Tutorial 8**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **TITLE: Matplotlib library in Python** |

**AIM:** Write a program to explore the Matplotlib library

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:** To demonstrate Matplot library in python

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Resource Needed: Python IDE**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Theory:**

## What is Matplotlib?

|  |
| --- |
| 1. Matplotlib  Matplotlib is a data visualization library and 2-D plotting library of Python It was initially released in 2003 and it is the most popular and widely-used plotting library in the Python community. It comes with an interactive environment across multiple platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, etc. It can be  used to embed plots into applications using various GUI toolkits like Tkinter, GTK+, wxPython, Qt, etc. So you can use Matplotlib to create plots, bar charts, pie charts, histograms, scatterplots, error charts, power spectra, stemplots, and whatever other visualization charts you want! The Pyplot module also provides a MATLAB-like interface that is just as versatile and useful as MATLAB while being free and open source.  2. Plotly  Plotly is a free open-source graphing library that can be used to form data visualizations. Plotly (plotly.py) is built on top of the Plotly JavaScript library (plotly.js) and can be used to create web-based data visualizations that can be displayed in Jupyter notebooks or web applications using Dash or saved as individual HTML files. Plotly provides more than 40 unique chart types like scatter plots, histograms, line charts, bar charts, pie charts, error bars, box plots, multiple axes, sparklines, dendrograms, 3-D charts, etc. Plotly also provides contour plots, which are not that common in other data visualization libraries. In addition to all this, Plotly can be used offline with no internet connection |

## 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.

**Syntax:**

matplotlib.pyplot.plot(\\*args, scalex=True, scaley=True, data=None, \\*\\*kwargs)

* x, y: These parameter are the horizontal and vertical coordinates of the data points. x values are optional.
* fmt: This parameter is an optional parameter and it contains the string value.
* data: This parameter is an optional parameter and it is an object with labelled data.

Returns:

This returns the following:

**lines :** This returns the list of Line2D objects representing the plotted data.

### Example:-

|  |  |
| --- | --- |
| **Draw a line in a diagram from position (1, 3) to position (8, 10):** | **Output** |
| import matplotlib.pyplot as plt import numpy as np  xpoints = np.array([1, 8]) ypoints = np.array([3, 10])  plt.plot(xpoints, ypoints) plt.show() | https://www.w3schools.com/python/img_matplotlib_plotting1.png |

## 1) Multiple Points

You can plot as many points as you like, just make sure you have the same number of points in both axis.

|  |  |
| --- | --- |
| **Program:**  import matplotlib.pyplot as plt import numpy as np  xpoints = np.array([1, 2, 6, 8]) ypoints = np.array([3, 8, 1, 10])  plt.plot(xpoints, ypoints) plt.show() | **Output:--** |

## 2) Matplotlib Line

## Linestyle:--- You can use the keyword argument linestyle, or shorter ls, to change the style of the plotted line:

Following are the linestyles available in ***matplotlib*:**

**Using *linestyle* Argument:**

* Solid
* Dashed
* Dotted
* Dashdot
* None

|  |  |
| --- | --- |
| **Syntax:** plt.plot(xdata, ydata, linestyle='dotted') | |
| **Program** | **Output:** | |
| **Use a dotted line:**  import matplotlib.pyplot as plt import numpy as np ypoints = np.array([3, 8, 1, 10]) plt.plot(ypoints, linestyle = 'dotted') plt.show() |  | |

## 3)Matplotlib Labels and Title

## a.Create Labels for a Plot

With Pyplot, you can use the xlabel() and ylabel() functions to set a label for the x- and y-axis.

The xlabel() function in pyplot module of matplotlib library is used to set the label for the x-axis.

|  |
| --- |
| **Syntax:** matplotlib.pyplot.xlabel(xlabel, fontdict=None, labelpad=None, \*\*kwargs) |

## b. Create a Title for a Plot

With Pyplot, you can use the title() function to set a title for the plot.

|  |  |
| --- | --- |
| **Program:--**  import numpy as np import matplotlib.pyplot as plt  x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125]) y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])  plt.plot(x, y)  plt.title("Sports Watch Data") plt.xlabel("Average Pulse") plt.ylabel("Calorie Burnage")  plt.show() | **Output:--** |

## 4) Matplotlib Scatter

## Creating Scatter Plots

With Pyplot, you can use the scatter() function to draw a scatter plot.

The scatter() function plots one dot for each observation. It needs two arrays of the same length, one for the values of the x-axis, and one for values on the y-axis:

|  |  |
| --- | --- |
| Syntax:-- matplotlib.pyplot.scatter(x\_axis\_data, y\_axis\_data, s=None, c=None, marker=None, cmap=None, vmin=None, vmax=None, alpha=None, linewidths=None, edgecolors=None)   * **x\_axis\_data-** An array containing x-axis data * **y\_axis\_data-** An array containing y-axis data * **s-** marker size (can be scalar or array of size equal to size of x or y) * **c-** color of sequence of colors for markers * marker- marker style * **cmap-** cmap name * **linewidths-** width of marker border * **edgecolor-** marker border color * **alpha-** blending value, between 0 (transparent) and 1 (opaque)   Except x\_axis\_data and y\_axis\_data all other parameters are optional and their default value is None. Below are the scatter plot examples with various parameters. | |
| import matplotlib.pyplot as plt import numpy as np  x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6]) y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])  plt.scatter(x, y) plt.show() | **Output:--** |

## Add Grid Lines to a Plot

With Pyplot, you can use the grid() function to add grid lines to the plot.

|  |  |
| --- | --- |
| import numpy as np import matplotlib.pyplot as plt  x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125]) y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])  plt.title("Sports Watch Data") plt.xlabel("Average Pulse") plt.ylabel("Calorie Burnage")  plt.plot(x, y)  plt.grid()  plt.show() | **Output:** |

## 5) Display Multiple Plots

With the subplot() function you can draw multiple plots in one figure.

|  |
| --- |
| subplot(nrows, ncols, index, \*\*kwargs)  **The layout is organized in rows and columns, which are represented by the *first*and *second*argument.**  **The third argument represents the index of the current plot.** |

|  |  |
| --- | --- |
| Program:-import matplotlib.pyplot as plt import numpy as np  #plot 1: x = np.array([0, 1, 2, 3]) y = np.array([3, 8, 1, 10])  plt.subplot(1, 2, 1) plt.plot(x,y)  #plot 2: x = np.array([0, 1, 2, 3]) y = np.array([10, 20, 30, 40])  plt.subplot(1, 2, 2) plt.plot(x,y)  plt.show() | **Output:--** |
| import matplotlib.pyplot as plt import numpy as np  #plot 1: x = np.array([0, 1, 2, 3]) y = np.array([3, 8, 1, 10])  plt.subplot(2, 1, 1) plt.plot(x,y)  #plot 2: x = np.array([0, 1, 2, 3]) y = np.array([10, 20, 30, 40])  plt.subplot(2, 1, 2) plt.plot(x,y)  plt.show() | **Output:--** |

## 6) Creating Bars

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

|  |  |
| --- | --- |
| 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() | **Output:--** |
| 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, color = "#4CAF50") plt.show() |  |

## 7) Creating Pie Chart with Labels:

|  |  |
| --- | --- |
| import matplotlib.pyplot as plt import numpy as np  y = np.array([35, 25, 25, 15]) mylabels = ["Apples", "Bananas", "Cherries", "Dates"]  plt.pie(y, labels = mylabels) plt.show() | **Output:** |

**Problem Definition**:

**Note:-- All plot should be labelled on X-axis and Y-axis with Grid for each program.**

1.Write a Python program to draw a line using given axis values with suitable label in the x axis, y axis and a title.

2. a)Write a Python programming to display a bar chart of the popularity of programming Languages. Also draw Pie chart for **popularity** Data values.

**Sample data:**

**Programming languages: Java, Python, PHP, JavaScript, C#, C++**

**Popularity: 22.2, 17.6, 8.8, 8, 7.7, 6.7**

b) Write a Python program to display a horizontal bar chart of the popularity of programming Languages. **Hint: use the barh() function**

3.Prepare a dataset using list as **Weight** and **height** parameters for your batch students

and draw a scatter plot with appropriate label and title.

**Post Lab Questions:--**

1. Considering datasets of your choice, create and explain the utility of following charts:

|  |  |
| --- | --- |
| 1. Swarn chart 2. Pair chart 3. Pair grid 4. Facet Grid 5. Scatter plot | 1. Regression plot 2. Count plot 3. Bar plot 4. Violin plot 5. Heat map |

## ANSWER:-

Swarm Chart: Adjusts data points to prevent overlap, enhancing visualization of individual points in a categorical distribution.

Pair Chart: Plots each pair of variables against each other, providing a quick overview of potential correlations.

Pair Grid: Similar to pair charts but offers more customization and control over the appearance of the plots.

Facet Grid: Creates a grid of subplots based on categorical variables, making it easier to compare different categories.

Scatter Plot: Visualizes the relationship between two continuous variables, great for identifying patterns, clusters, or outliers.

Regression Plot: Displays the relationship between two variables along with a regression line, useful for trend analysis and predictions.

Count Plot: Displays the count of observations in each category of a categorical variable, effective for showing the distribution of categorical data.

Bar Plot: Compares the quantities of different categories, especially effective when the categories are nominal or ordinal.

Violin Plot: Combines aspects of box plots and kernel density plots to provide a summary of the distribution of a continuous variable across different categories.

Heat Map: Uses color gradients to represent the values in a matrix, effective for visualizing the magnitude of a variable across two dimensions.

## What is the Seaborn library? What are Different categories of plot in Seaborn.

ANSWER:- Seaborn is a Python data visualization library based on Matplotlib. It provides a high-level interface for creating attractive and informative statistical graphics, making it easier to visualize complex datasets with multiple variables. Seaborn comes with built-in themes and color palettes to enhance the aesthetics of the plots.

The main categories of plots in Seaborn are:

Relational Plots: These plots show the relationship between two variables. They include scatter and line plots.

Categorical Plots: These plots are used to visualize categorical data. They include bar, count, box, violin, and swarm plots.

Distribution Plots: These plots visualize the distribution of a dataset. They include histogram, kde, and rug plots.

Regression Plots: These plots are used to visualize the relationship between two variables and fit a regression model to the data.

Matrix Plots: These plots are used to visualize data in matrix form. They include heatmap and clustermap.

Multi-Plot Grids: These are used to create complex visualizations by drawing multiple instances of the same plot on different subsets of your dataset.

Axis Grids: These are used to draw multiple instances of the same plot on different subsets of your dataset, with the subsets defined by one or two categorical variables.

**Books/ Journals/ Websites referred:**

1. [Matplotlib Plotting (w3schools.com)](https://www.w3schools.com/python/matplotlib_plotting.asp) – Reference website.
2. Reema Thareja, Python Programming: Using Problem Solving Approach, Oxford University Press, First Edition 2017, India
3. Sheetal Taneja and Naveen Kumar, Python Programming: A modular Approach, Pearson India, Second Edition 2018,India

**Implementation details:**

**1)** Code:

import matplotlib.pyplot as plt

x = [1,2,3]

y = [2,3,4]

plt.grid()

plt.plot(x, y)

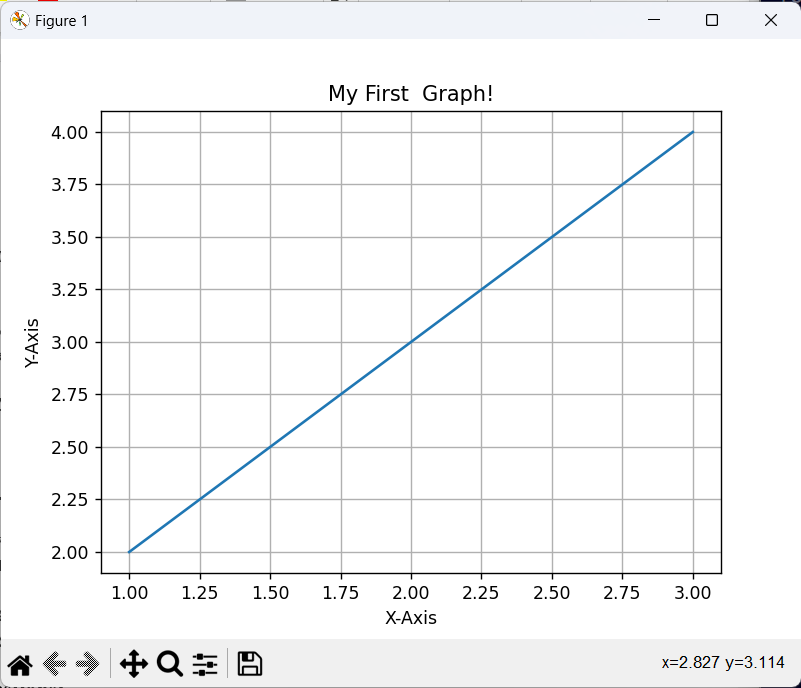
plt.xlabel('X-Axis')

plt.ylabel('Y-Axis')

plt.title('My First  Graph!')

plt.show()

Output:



**2)a)Code:**

**For Bar Graph**

import matplotlib.pyplot as plt

x = ["Java", "Python", "PHP", "JavaScript", "C#", "C++" ]

y = [22.2,17.6,8.8,8,7.7,6.7]

plt.xlabel("Programming languages" )

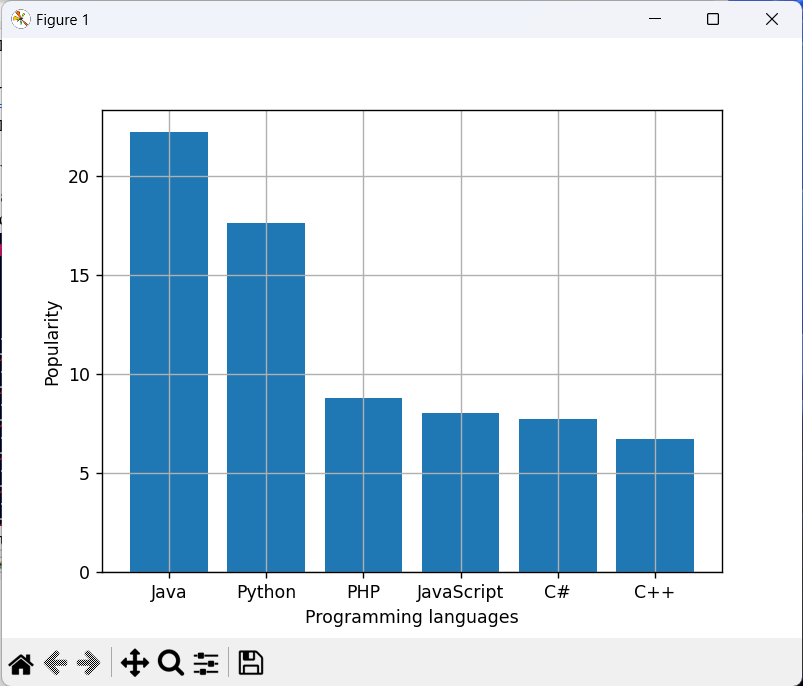
plt.ylabel("Popularity")

plt.bar(x,y)

plt.grid()

plt.show()

**Output:**



**For Pie Chart:**

import matplotlib.pyplot as plt

x = ["Java", "Python", "PHP", "JavaScript", "C#", "C++" ]

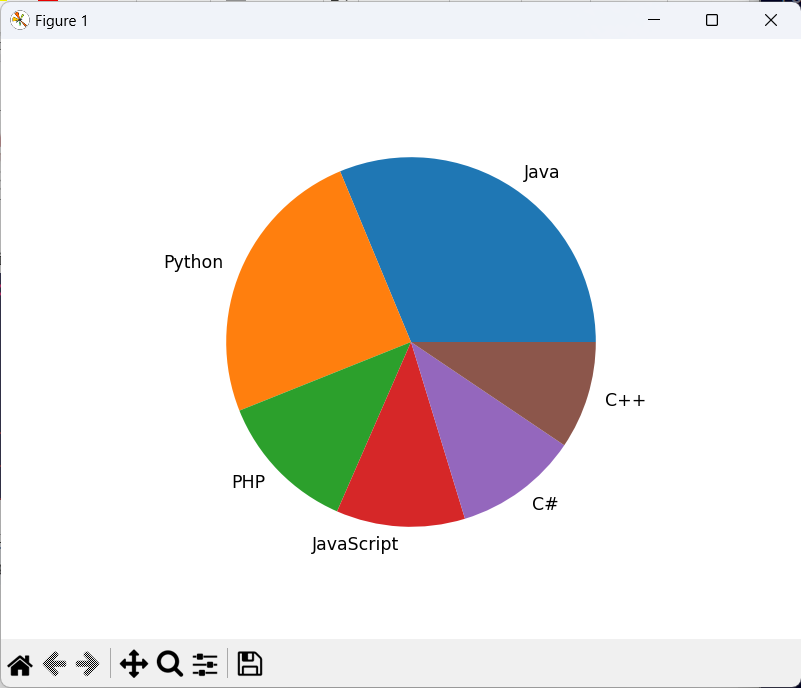
y = [22.2,17.6,8.8,8,7.7,6.7]

plt.pie(y, labels=x)

plt.grid()

plt.show()

**Output:**

****

**2.b)Code:**

import matplotlib.pyplot as plt

x = ["Java", "Python", "PHP", "JavaScript", "C#", "C++" ]

y = [22.2,17.6,8.8,8,7.7,6.7]

plt.xlabel("Programming languages" )

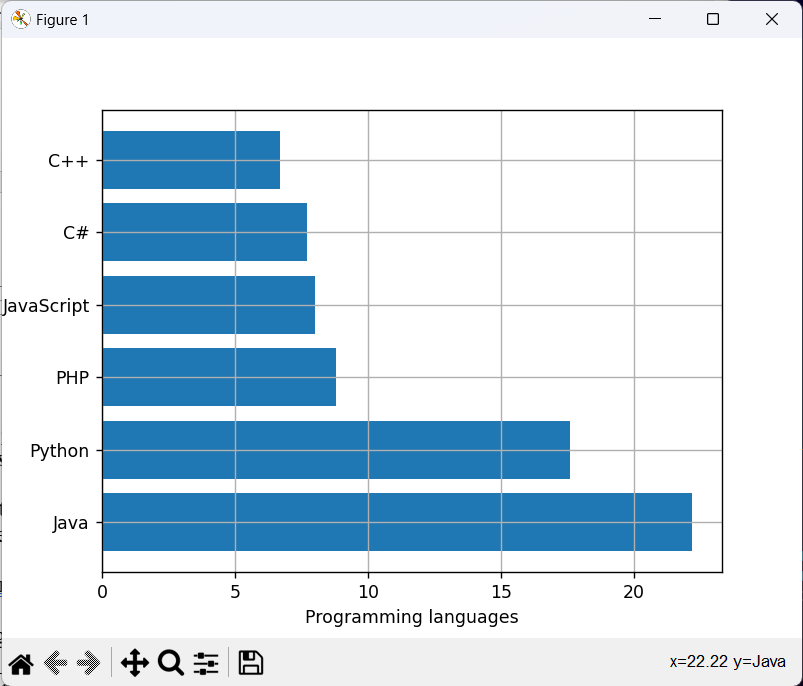
plt.ylabel("Popularity")

plt.barh(x,y)

plt.grid()

plt.show()

Output:



3) Code:

import matplotlib.pyplot as plt

weights = []

heights = []

num\_students = int(input("Enter the number of students: "))

for i in range(num\_students):

    weight = float(input(f"Enter the weight of student {i+1} in kg: "))

    height = float(input(f"Enter the height of student {i+1} in cm: "))

    weights.append(weight)

    heights.append(height)

plt.scatter(weights, heights)

plt.xlabel('Weights (kg)')

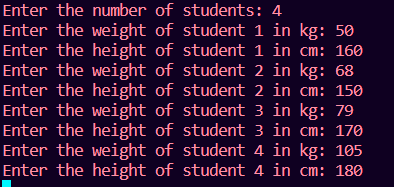
plt.ylabel('Heights (cm)')

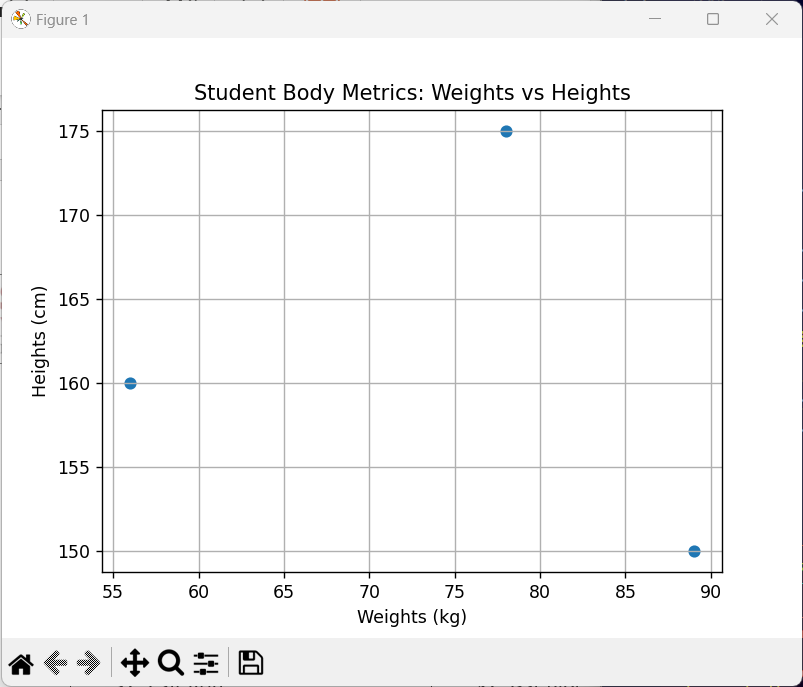
plt.title("Student Body Metrics: Weights vs Heights")

plt.grid()

plt.show()

Output:





**Conclusion:**

In this experiment we learnt how to represent data into graphs, pie charts, etc using matplotlib library in python. Matplotlib is a simple, flexible, and efficient Python library for high-quality data visualization, offering control over figure elements, compatibility with other tools, a variety of plots, and strong community support.

**Date: 20-11-23 Signature of faculty in-charge**