

|  |
| --- |
| **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:--OP1.JPG** |

## 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() | **OutputOP2.JPG** |

## 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:**  **OP3.JPG** |

## 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:--**  **OP4.JPG** |
| 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:--**  **OP5.JPG** |

## 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:--OP6.JPG** |
| 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() | **OP7.JPG** |

## 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:**  **OP8.JPG** |

**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 |

1)Swarm Plot

A Swarm plot is a type of categorical scatter plot that shows individual data points for different categories along a continuous axis. It helps visualize the distribution of data within each category without overlapping points, making it easier to observe patterns and densities in the dataset.

Input:

import seaborn as sns

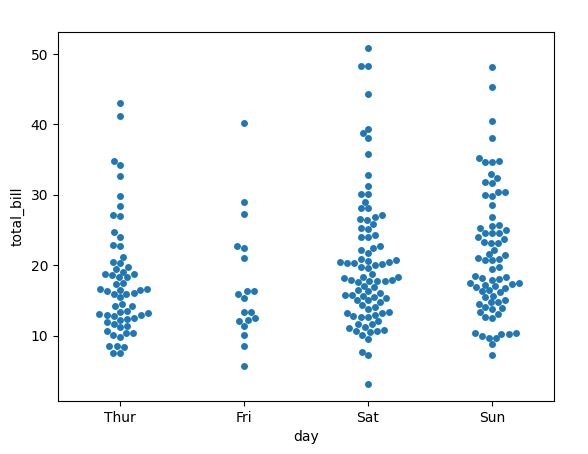
import matplotlib.pyplot as plt

tipdata = sns.load\_dataset('tips')

sns.swarmplot(x='day', y='total\_bill', data=tipdata)

plt.show()

Output:



2) Pair chart

A Pair chart displays pairwise relationships among variables in a dataset. It shows scatterplots or other types of plots for every possible combination of variables, helping to understand correlations and patterns between them.

Input:

import seaborn as sns

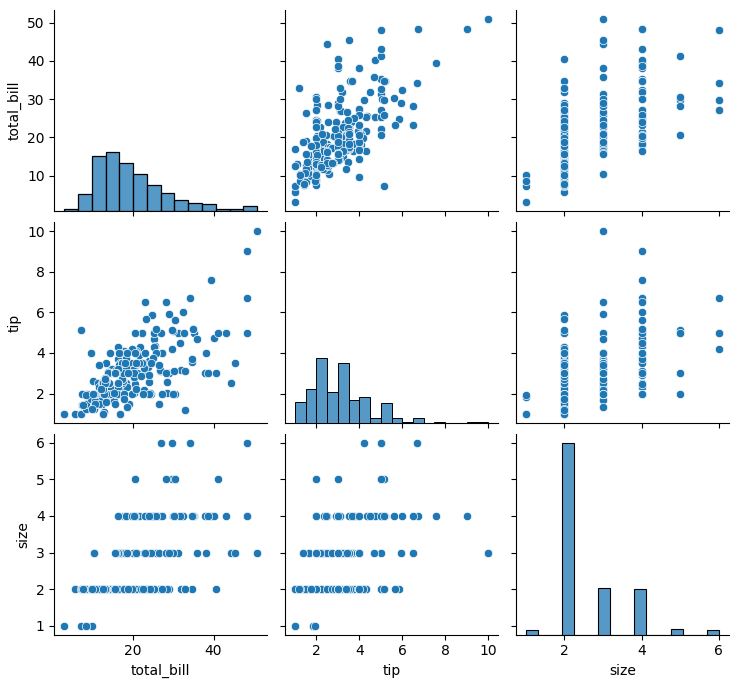
import matplotlib.pyplot as plt

tips\_data = sns.load\_dataset('tips')

sns.pairplot(tips\_data)

plt.show()

Output:



3) Pair grid

A Pair grid is a grid of Pair plots used to visualize multiple pairwise relationships between variables. It's a structured arrangement of Pair plots for better comparison and analysis of various interactions within a dataset.

Input:

import seaborn as sns

import matplotlib.pyplot as plt

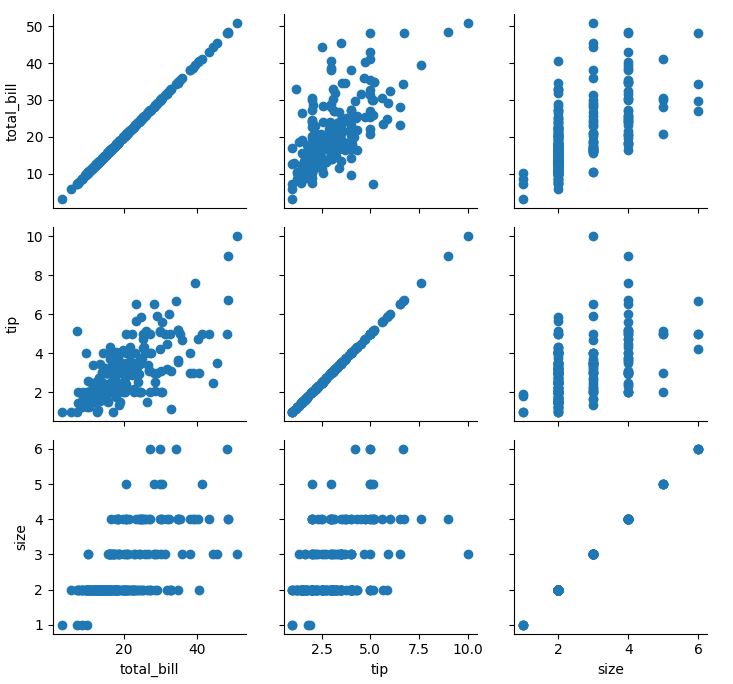
tips\_data = sns.load\_dataset('tips')

gr = sns.PairGrid(tips\_data)

g.map(plt.scatter)

plt.show()

Output:



4) Facet Grid

A Facet Grid is a way to create multiple plots (like histograms or scatter plots) arranged in a grid based on the categories or groups in the data. It allows for easy comparison of subsets of the data.

Input:

import seaborn as sns

import matplotlib.pyplot as plt

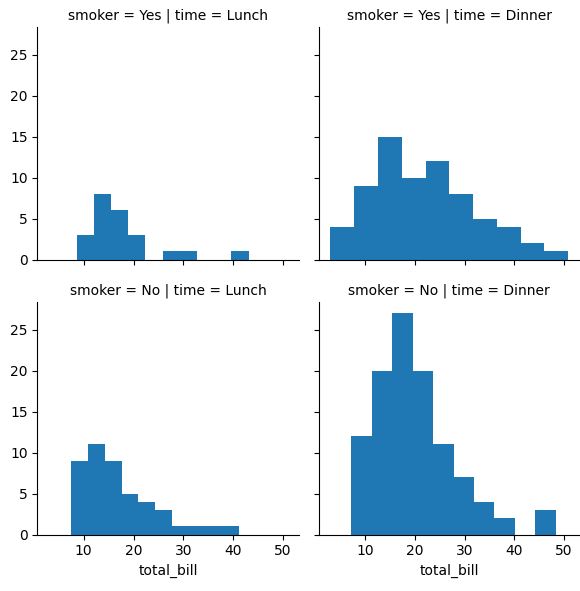
tips\_data = sns.load\_dataset('tips')

g = sns.FacetGrid(tips\_data, col='time', row='smoker')

g.map(plt.hist, 'total\_bill')

plt.show()

Output:



5) Scatter plot

A Scatter plot is a graph that displays individual data points as dots on a two-dimensional plane. It's used to observe and visualize the relationship or correlation between two variables.

Input:

import seaborn as sns

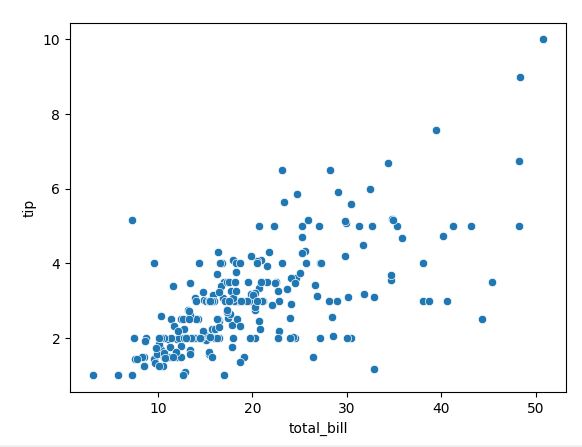
import matplotlib.pyplot as plt

tips\_data = sns.load\_dataset('tips')

sns.scatterplot(x='total\_bill', y='tip', data=tips\_data)

plt.show()

Output:



6) Regression plot

A Regression plot is a type of graph that shows the relationship between two variables, typically with a straight line representing the best-fit linear regression. It helps in understanding the trend or predictive nature of the data.

Input:

import seaborn as sns

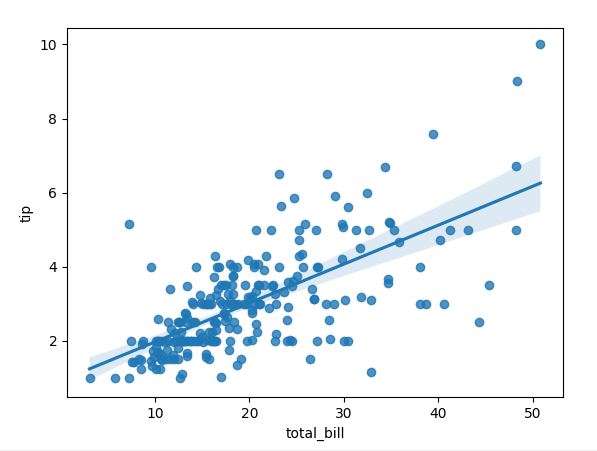
import matplotlib.pyplot as plt

tips\_data = sns.load\_dataset('tips')

sns.regplot(x='total\_bill', y='tip', data=tips\_data)

plt.show()

Output:



7) Count plot

A Count plot represents the count of occurrences of different categories or items in a dataset. It's a simple histogram-like plot that shows the frequency of each category.

Input:

import seaborn as sns

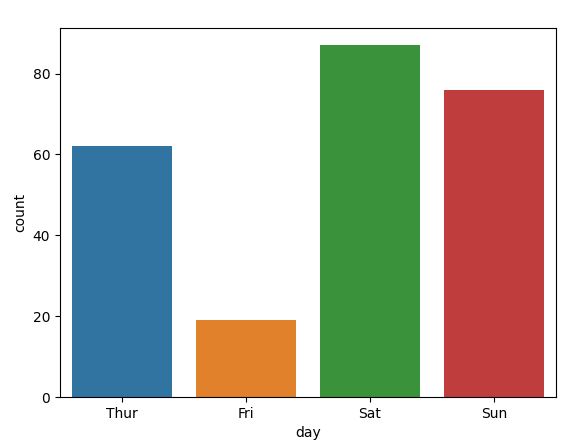
import matplotlib.pyplot as plt

tips\_data = sns.load\_dataset('tips')

sns.countplot(x='day', data=tips\_data)

plt.show()

Output:



8) Bar plot

A Bar plot is a graph that uses bars to represent different categories of data. It's suitable for displaying and comparing the quantities of different groups or items.

Input:

import seaborn as sns

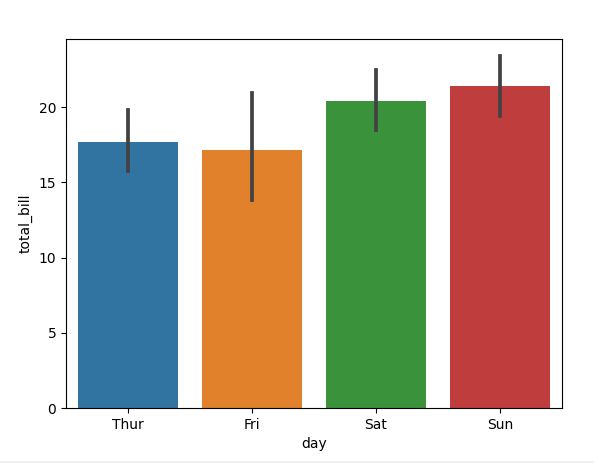
import matplotlib.pyplot as plt

tips\_data = sns.load\_dataset('tips')

sns.barplot(x='day', y='total\_bill', data=tips\_data)

plt.show()

Output:



9) Violin plot

A Violin plot is a visual representation of the distribution of numerical data across different categories. It combines the features of a box plot and a density plot, displaying the distribution's shape and spread.

Input:

import seaborn as sns

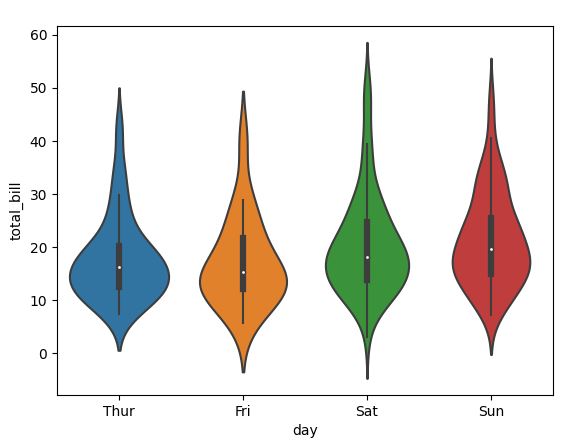
import matplotlib.pyplot as plt

tips\_data = sns.load\_dataset('tips')

sns.violinplot(x='day', y='total\_bill', data=tips\_data)

plt.show()

Output:



10) Heat map

A Heat map is a graphical representation of data where values are depicted using colors on a grid. It's commonly used to visualize correlations or patterns in large datasets, with colors indicating the intensity or magnitude of values in the grid cells.

Input:

import seaborn as sns

import matplotlib.pyplot as plt

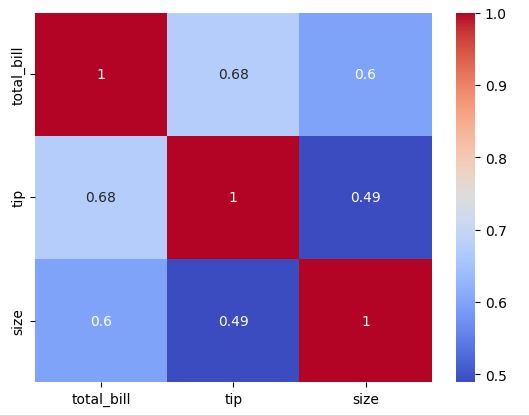
tips\_data = sns.load\_dataset('tips')

correlation\_matrix = tips\_data.corr()

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm')

plt.show()

Output:



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

A) Seaborn is a Python data visualization library used to create appealing and informative statistical graphics. It's built on Matplotlib and works well with Pandas data.

Different categories of plots in Seaborn include:

* Relational Plots: Show relationships between variables (e.g., scatter plots, line plots).
* Categorical Plots: Display relationships among categorical variables (e.g., bar plots, box plots).
* Distribution Plots: Visualize data distributions (e.g., histograms, kernel density plots).
* Matrix Plots: Display 2D data or matrices (e.g., heatmaps, pair plots).
* Regression Plots: Explore linear relationships between variables.
* Time Series Plots: Visualize time-based data using line plots.

These categories offer a variety of ways to visualize different aspects of your data effectively and attractively.

**Books/ Journals/ Websites referred:**

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

**Implementation details:**

**1)**

import numpy as np

import matplotlib.pyplot as plt

x = np.array(['January','February','March','April'])

y = np.array([10,15,20,25])

plt.title("Average monthly temperature")

plt.xlabel("Months")

plt.ylabel("Temperature")

plt.plot(x, y)

plt.grid()

plt.show()

2)

a)

Bar Graph-

import matplotlib.pyplot as plt

import numpy as np

x = np.array(["Java","Python","PHP","Javascript","C#","C++"])

y = np.array([22.2,17.6,8.8,8,7.7,6.7])

plt.bar(x,y,color="#FFA500")

plt.show()

Pie Chart-

import matplotlib.pyplot as plt

import numpy as np

mylabels = np.array(["Java","Python","PHP","Javascript","C#","C++"])

y = np.array([22.2,17.6,8.8,8,7.7,6.7])

plt.pie(y, labels = mylabels)

plt.show()

**b)**

import matplotlib.pyplot as plt

import numpy as np

x = np.array(["Java","Python","PHP","Javascript","C#","C++"])

y = np.array([22.2,17.6,8.8,8,7.7,6.7])

plt.barh(x,y,color="#FFA500")

plt.show()

**3)**

import matplotlib.pyplot as plt

import numpy as np

weight = np.array([55,57,95,49,70,63,81,57])

height = np.array([160,172,183,155,172,167,171,163])

plt.scatter(weight, height, color='orange')

plt.xlabel('Weight (kg)')

plt.ylabel('Height (cm)')

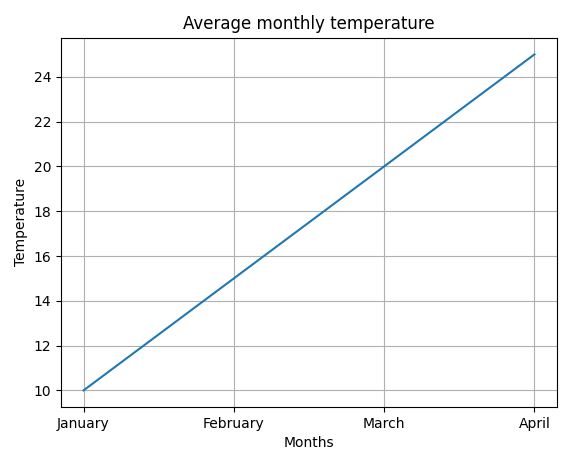
plt.title('Scatter Plot of Weight vs Height of Students')

plt.grid()

plt.show()

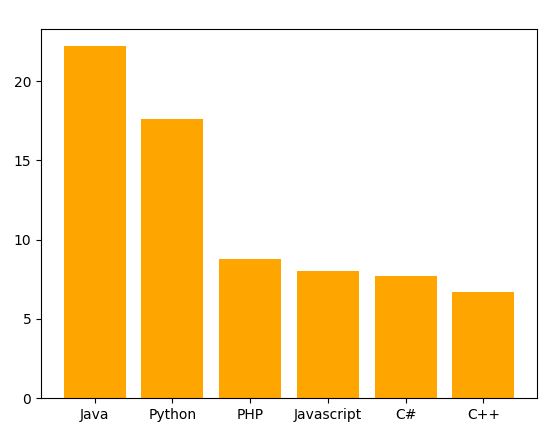
**Output:**

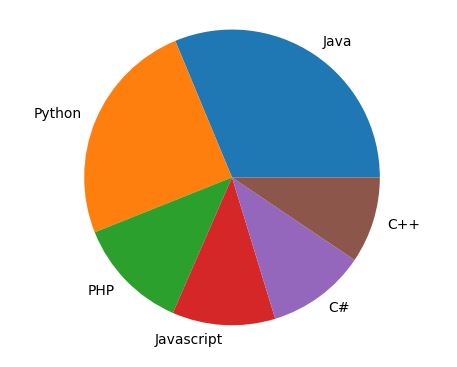
**1)**

****

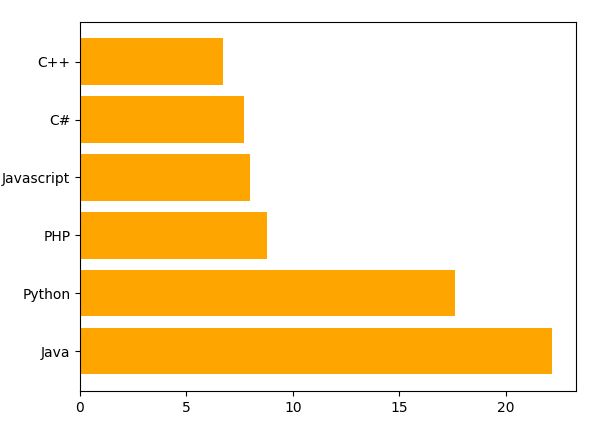
**2)**

**a)**

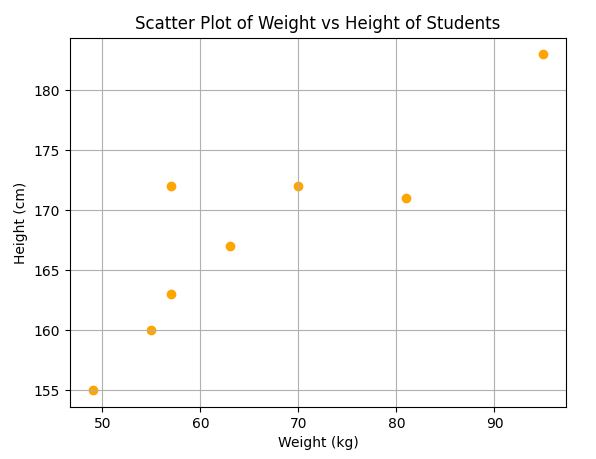
****

****

**b)**

****

**3)**

****

**Conclusion:** Wrote programs to get a better understanding of the Matplotlib library of Python and its various uses to make real life data analysis easier by combining it with Numpy.

**Date: 20 -10-2023 Signature of faculty in-charge**