

Batch: C5-1 Roll No.: 16010123293

Experiment / assignment / Tutorial: Exp8

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)`

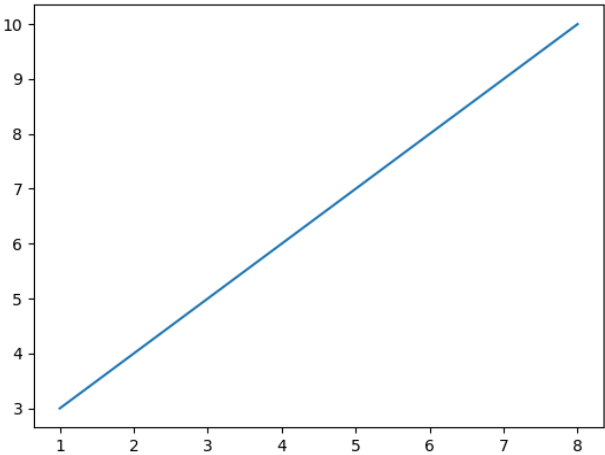
- `x, y`: These parameter are the horizontal and vertical coordinates of the data points. `x` values are optional.
- `fmt`: This parameter is an optional parameter and it contains the string value.
- `data`: This parameter is an optional parameter and it is an object with labelled data.

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
<pre>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()</pre>	

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.



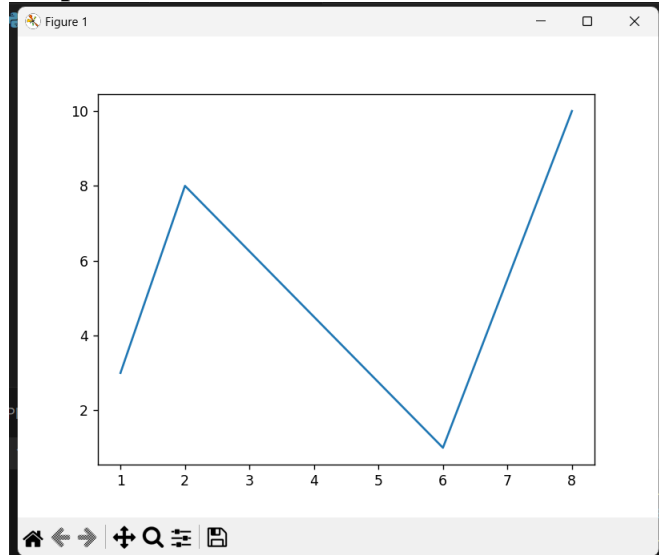
Program:

```
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

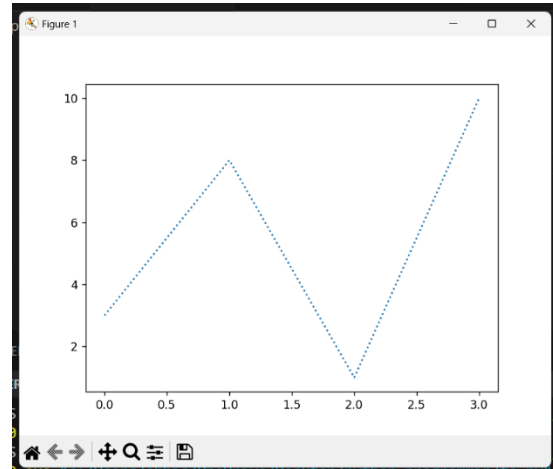
Syntax: plt.plot(xdata, ydata, linestyle='dotted')

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,
```

Output:--

```
300, 310, 320, 330])
```

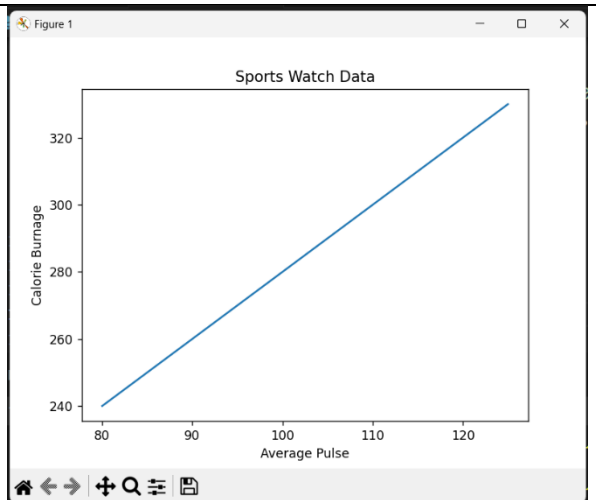
```
plt.plot(x, y)
```

```
plt.title("Sports Watch Data")
```

```
plt.xlabel("Average Pulse")
```

```
plt.ylabel("Calorie Burnage")
```

```
plt.show()
```



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.

```

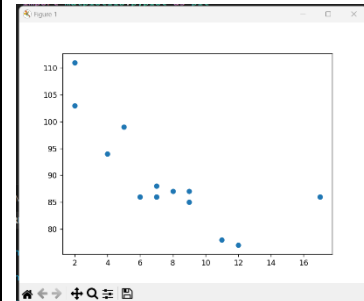
import matplotlib.pyplot as plt
import numpy as np

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")

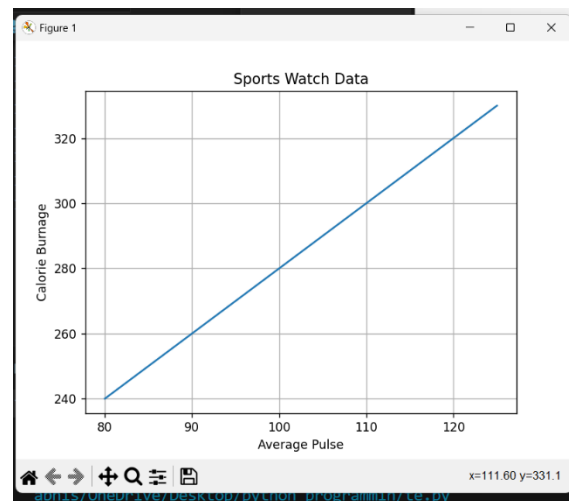
plt.plot(x, y)

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)
```

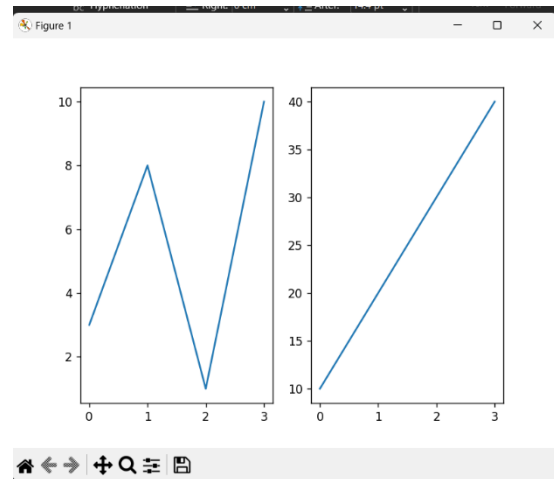
#plot 2:

```
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)
```

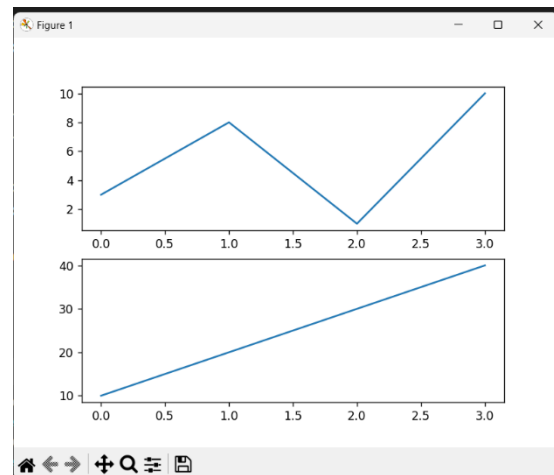
#plot 2:

```
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

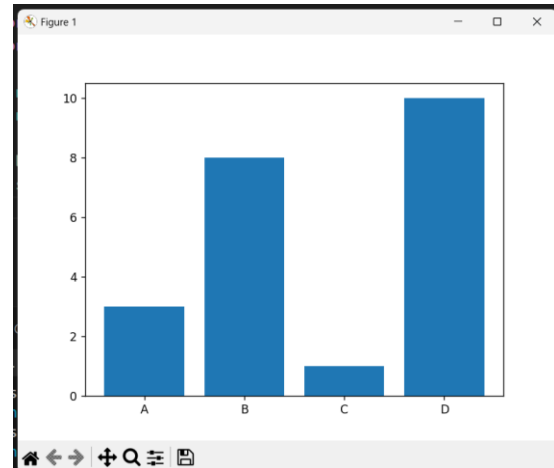
With Pyplot, you can use the `bar()` function to draw bar graphs.


```
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()
```

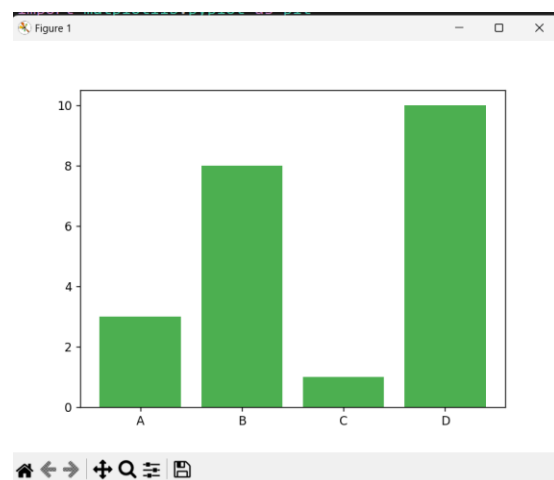
Output:--



```
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()
```



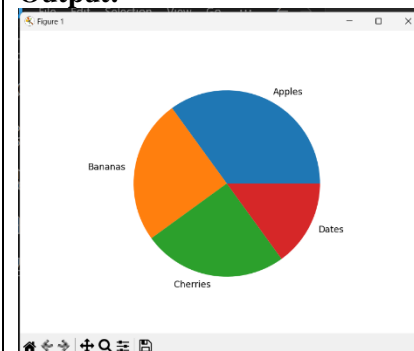
7) Creating Pie Chart with Labels:

```
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) Swarm chart	6) Regression plot
2) Pair chart	7) Count plot
3) Pair grid	8) Bar plot
4) Facet Grid	9) Violin plot
5) Scatter plot	10) Heat map

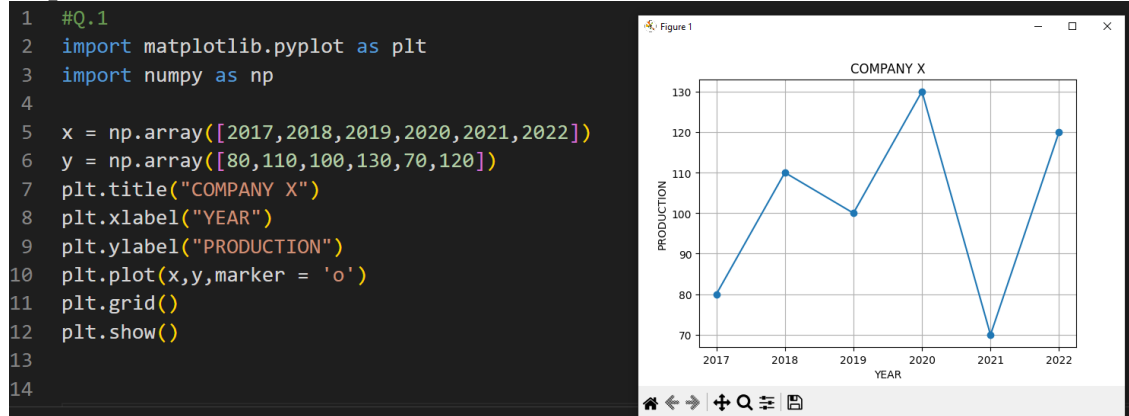
2) What is the Seaborn library? What are Different categories of plot in Seaborn.

Books/ Journals/ Websites referred:

1. [Matplotlib Plotting \(w3schools.com\)](https://www.w3schools.com/matplotlib/matplotlib_intro.asp) – Reference website.
2. Reema Thareja, Python Programming: Using Problem Solving Approach, Oxford University Press, First Edition 2017, India

- Sheetal Taneja and Naveen Kumar, Python Programming: A modular Approach, Pearson India, Second Edition 2018, India

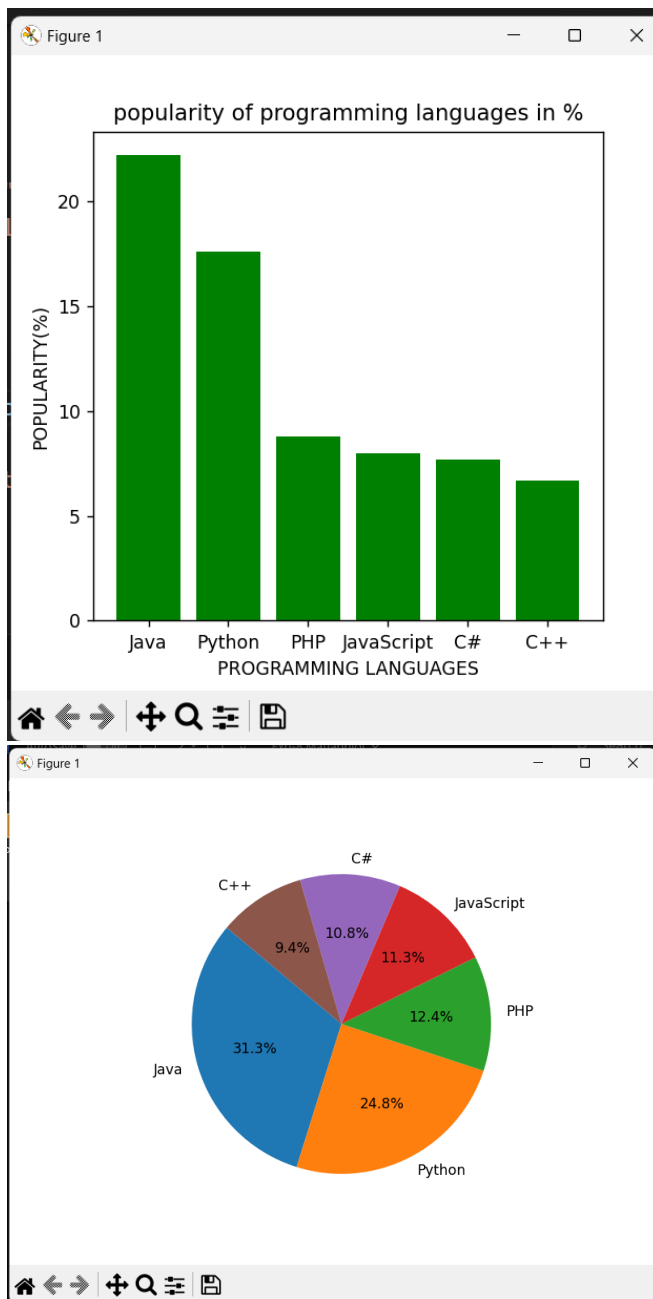
Implementation details:

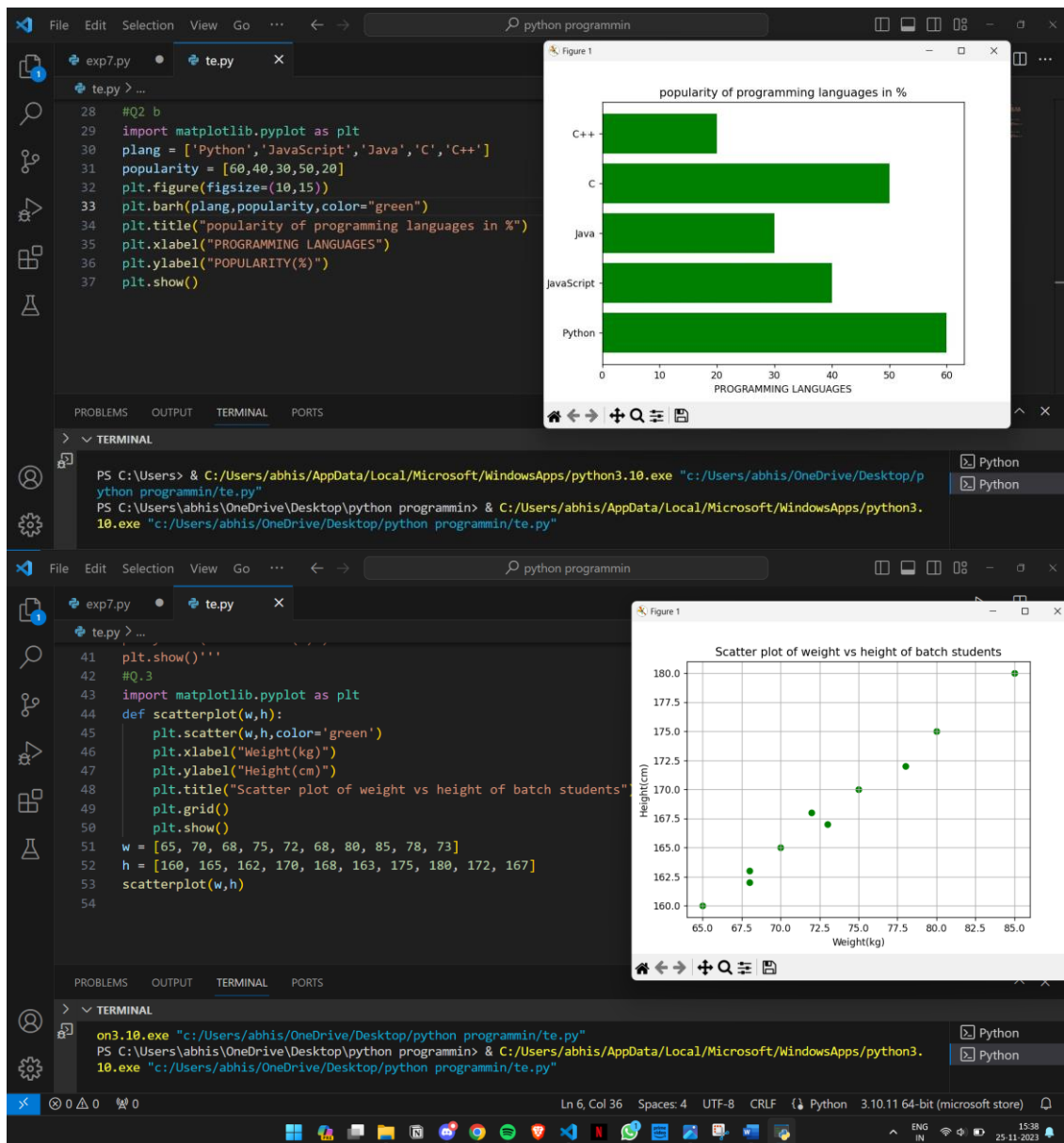


```

#Q.2 a
import matplotlib.pyplot as plt
def bar_chart(plang,popularity):
    plt.figure(figsize=(10,15))
    plt.bar(plang,popularity,color="green")
    plt.title("popularity of programming languages in %")
    plt.xlabel("PROGRAMMING LANGUAGES")
    plt.ylabel("POPULARITY(%)")
    plt.show()
def pie_chart(plang,popularity):
    plt.pie(popularity,labels = plang,autopct = "%1.1f%%",startangle = 140)
    plt.show()
plang = ['Java','Python','PHP','JavaScript','C#','C++']
popularity = [22.2,17.6,8.8,8.7,7.7,6.7]
bar_chart(plang,popularity)
pie_chart(plang,popularity)
  
```

Output:





Post lab qn1 Ans:

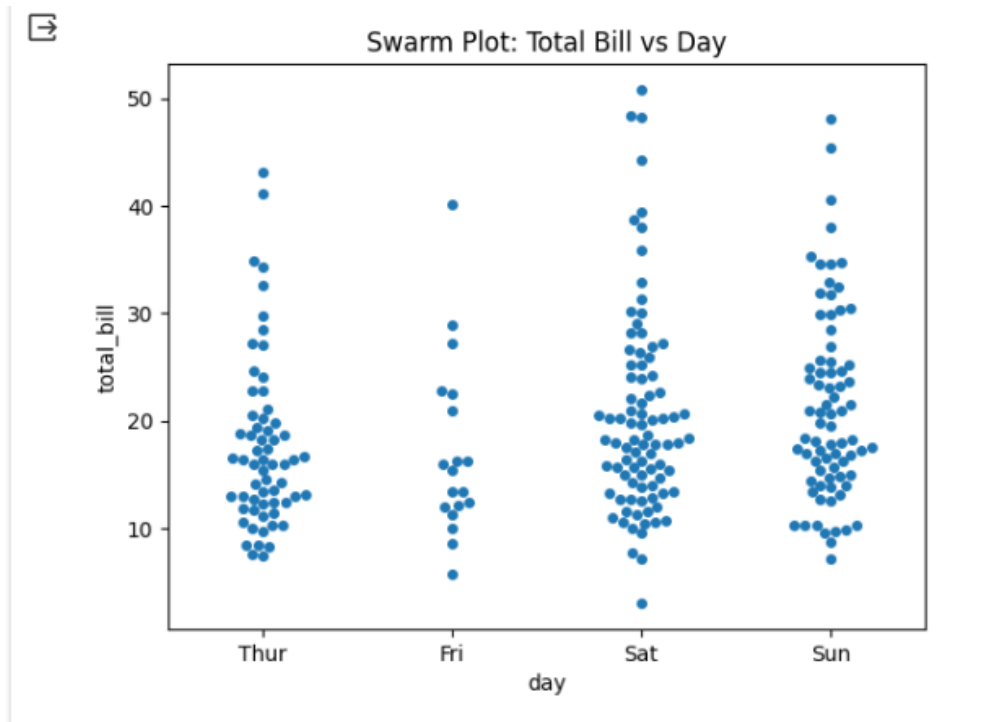
Swarm Chart:

This swarm plot visualizes the distribution of total bills on different days in a restaurant. It helps to see how bills are concentrated and if there are any outliers.

```
import seaborn as sns
import matplotlib.pyplot as plt

tips = sns.load_dataset('tips')
sns.swarmplot(x='day', y='total_bill' data= tips)
plt.title('Swarm Plot : Total Bill vs Day')
```

```
plt.show
```

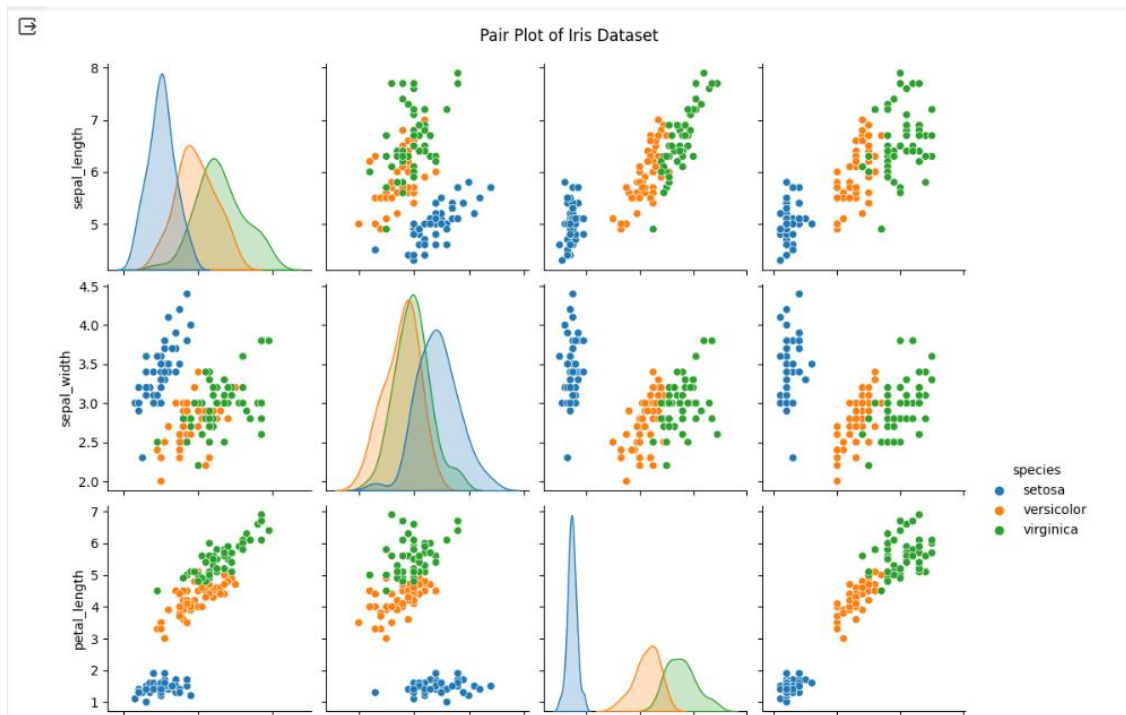


Pair Chart:

This pair plot shows the relationships between different measurements (sepal length, sepal width, petal length, petal width) for three species of iris flowers.

```
import seaborn as sns
import matplotlib.pyplot as plt

iris = sns.load_dataset('iris')
sns.pairplot(iris, hue= 'species')
plt.suptitle('Pair plot of iris dataset', y=1.02)
plt.show()
```



Pair Grid :

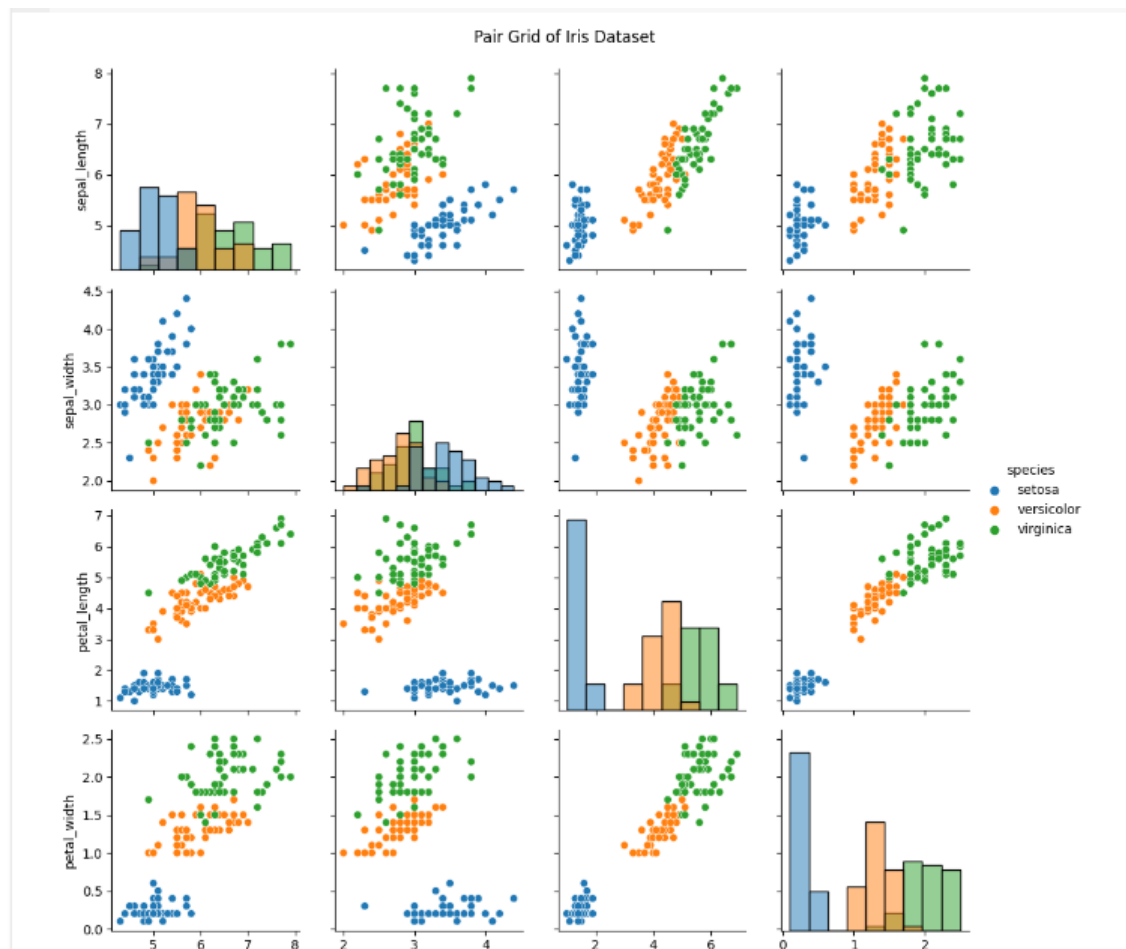
This pair grid provides a more customizable way to create pairwise plots. It's especially useful when you want different types of plots on the diagonal and off-diagonal.

```

import seaborn as sns
import matplotlib.pyplot as plt

iris = sns.load_dataset("iris")

g = sns.PairGrid(iris, hue="species")
g.map_diag(sns.histplot)
g.map_offdiag(sns.scatterplot)
g.add_legend()
plt.suptitle('Pair Grid of Iris Dataset', y=1.02)
plt.show()
  
```



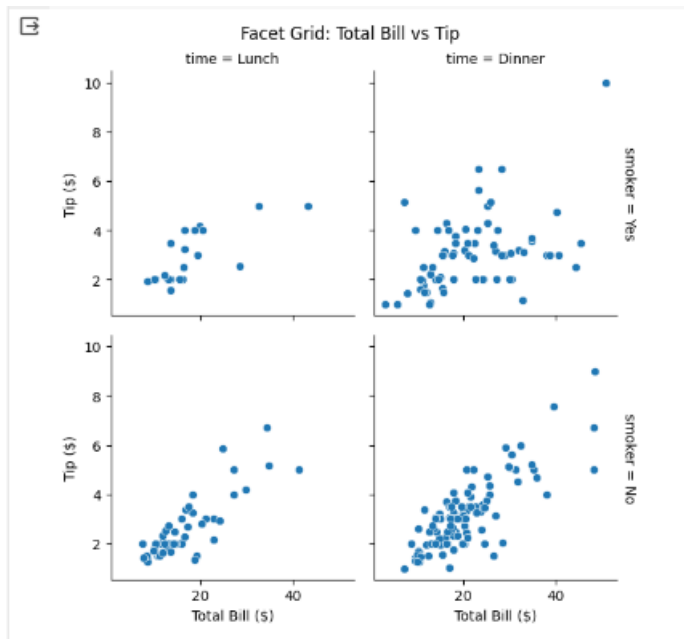
Facer Grid :

This facet grid separates the scatter plots of total bill vs tip based on time (lunch or dinner) and smoker status. It helps to analyze the relationship in different subsets of the data.

```

import seaborn as sns
import matplotlib.pyplot as plt

iris = sns.load_dataset('iris')
g = sns.FacetGrid(tips, col='time', row='smoker', margin_titles=True)
g.map(sns.scatterplot, 'total_bill', 'tip')
g.set_axis_labels('Total Bill($)', 'Tip($)')
plt.suptitle('Facet Grid : Total Bill vs Tip', y=1.02)
plt.show()
  
```

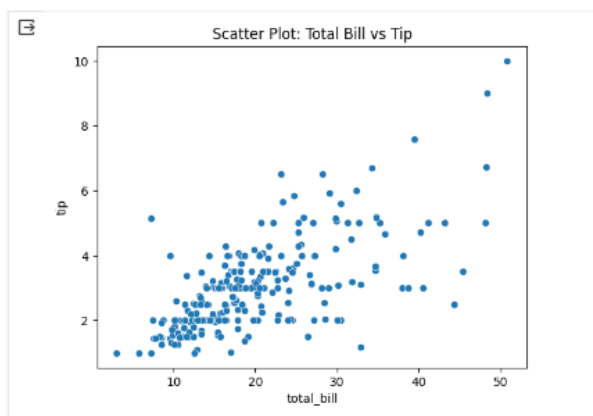



Scatter plot :

This simple scatter plot visualizes the relationship between total bill and tip in a restaurant, helping to identify patterns or trends.

```
import seaborn as sns
import matplotlib.pyplot as plt

tips = sns.load_dataset('tips')
sns.scatterplot(x='total_bill', y='tip', data=tips)
plt.title('Scatter plot: Total Bill vs tip')
plt.show()
```

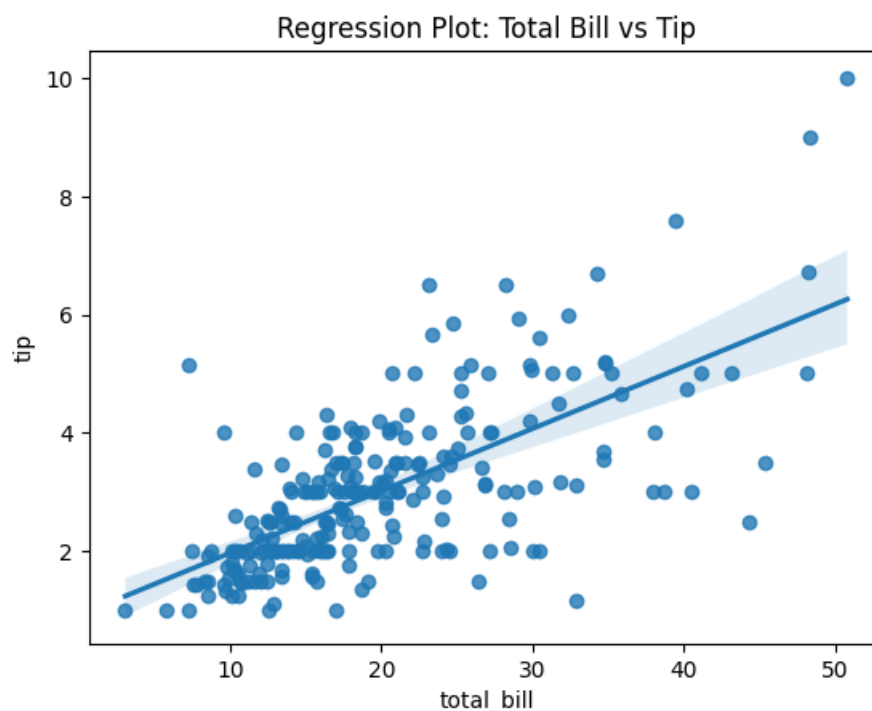


Regression plot

Regression plot not only shows the scatter plot but also fits a linear regression line to represent the trend in the data.

```
import matplotlib.pyplot as plt
import seaborn as sns

tips = sns.load_dataset('tips')
sns.regplot(x='total_bill', y='tip', data=tips)
plt.title('Regression Plot')
plt.show()
```

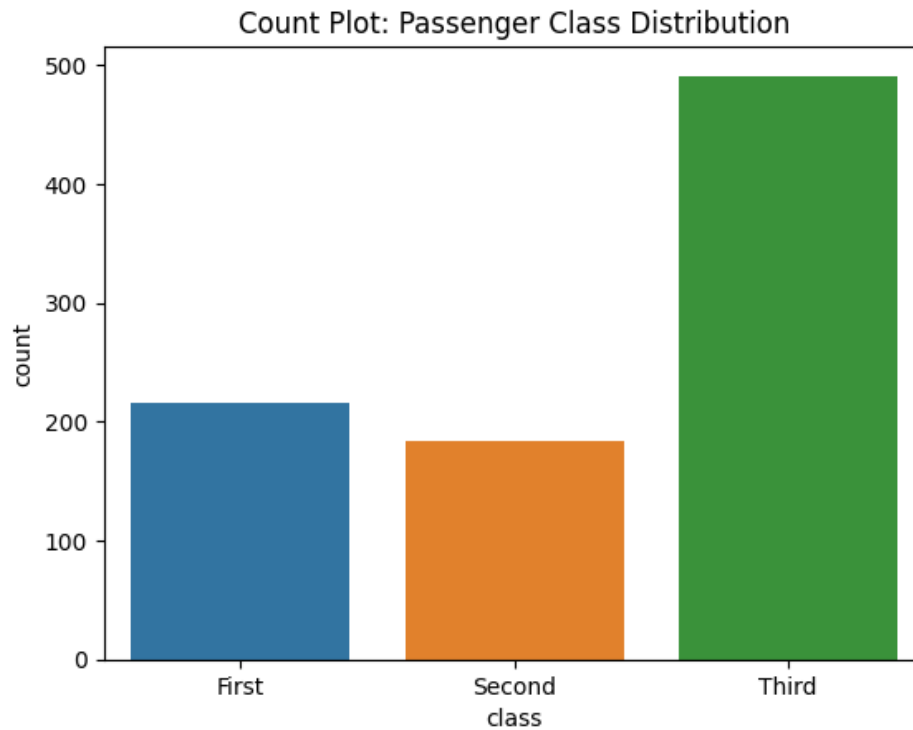


Count plot

`seaborn.countplot()` method is used to Show the counts of observations in each categorical bin using bars.

```
import matplotlib.pyplot as plt
import seaborn as sns

titanic = sns.load_dataset('titanic')
sns.countplot(x='class', data=titanic)
plt.title('Count Plot')
plt.show()
```



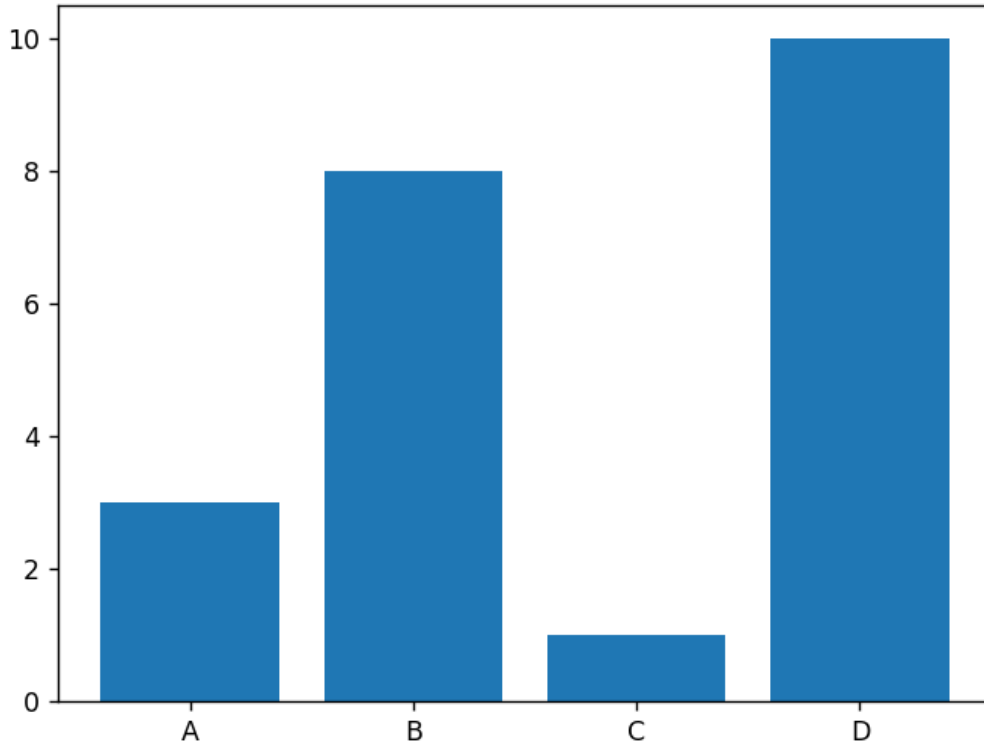
Bar Plot

The bar() function is used 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()
```



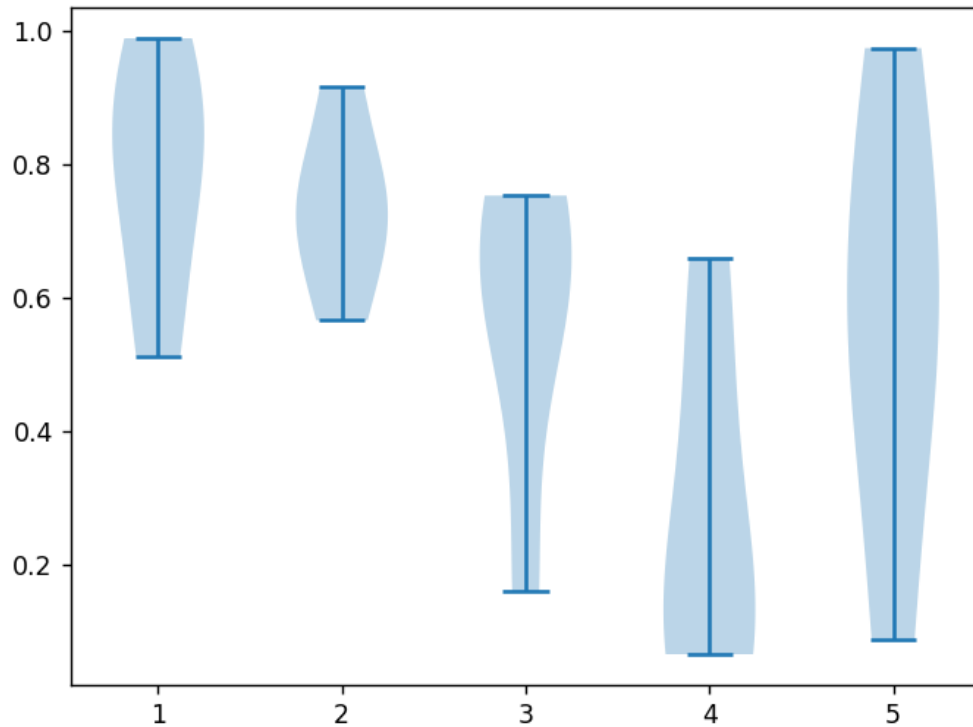
Violin Plot

Through this function, you can make a violin plot for every column of the dataset or each vector in the dataset sequence. All filled areas extend to show the entire data range with lines that are optional at the mean, the median, the maximum and the minimum.

```
import numpy as np
import matplotlib.pyplot as plt

data = np.random.random(( 5,5))

plt.violinplot(data)
plt.show()
```



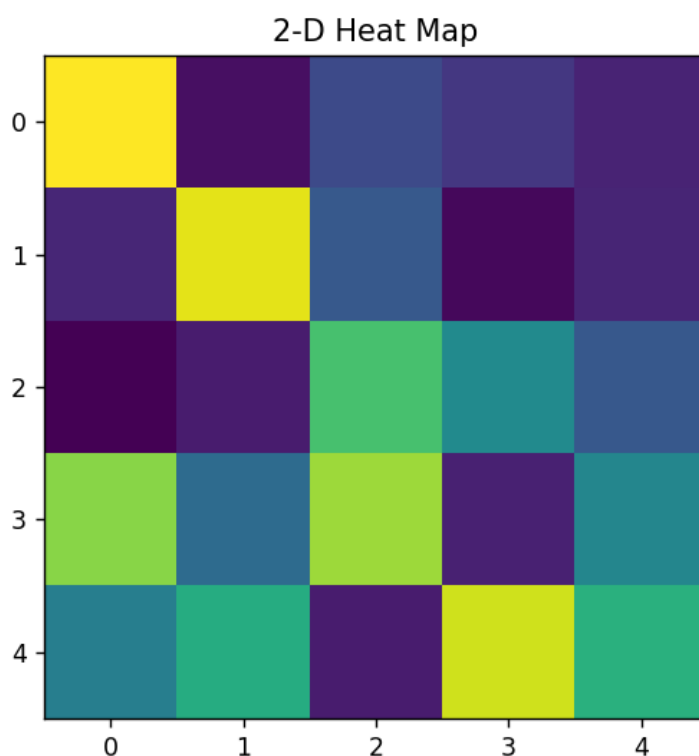
Heat map

A 2-D Heatmap is a data visualization tool that helps to represent the magnitude of the matrix in form of a colored table. It is used to see the correlation between columns of a dataset where we can use a darker color for columns having a high correlation.

```
import numpy as np
import matplotlib.pyplot as plt

data = np.random.random(( 5,5))
plt.imshow( data )

plt.title( "2-D Heat Map" )
plt.show()
```



Post lab qn2 Ans:

Python Seaborn library is a widely popular data visualization library that is commonly used for data science and machine learning tasks. You build it on top of the matplotlib data visualization library and can perform exploratory analysis. You can create interactive plots to answer questions about your data.

Conclusion: In this experiment we learned how to install matplotlib, how to use it for data visualisation in the form of graphs, bar charts, pie charts etc.

Date: 20 -10-2023

Signature of faculty in-charge