TASK 1

DATA VISUALISATIONS

Ancilla Teresa Dsouza
ShadowFox

DATA VISUALISATION

Data visualisation is the representation of information and data using charts, graphs, maps and other visual tools. It helps to spot patterns and trends quickly, makes complex data easier to interpret and is also useful for decision-making, reporting and storytelling.

To create effective visualizations in Python, two widely used libraries are **Matplotlib and Seaborn**. Both offer powerful tools to turn raw data into clear and meaningful visuals, each with its own strengths and specialities.

Matplotlib is one of the oldest and most versatile visualization libraries in Python. It allows users to build a wide range of plots from scratch and offers full control over every element of a figure — including layout, colours, fonts, and more. It supports line plots, bar charts, pie charts, scatter plots, histograms and many other types of visualisations. It is particularly useful when detailed customization is needed or when building complex, publication-quality plots.

Seaborn is built on top of Matplotlib and is designed to make visualization simpler and more attractive. It focuses on statistical graphics and works efficiently with datasets, especially those in pandas DataFrames. Seaborn provides built-in styling options and integrated colour palettes optimized for statistical data visualization. It makes it easy to create complex visualizations like heatmaps, boxplots, and violin plots with just a few lines of code. It is commonly used in exploratory data analysis to quickly understand relationships and patterns in data.

---

1. LINE PLOT

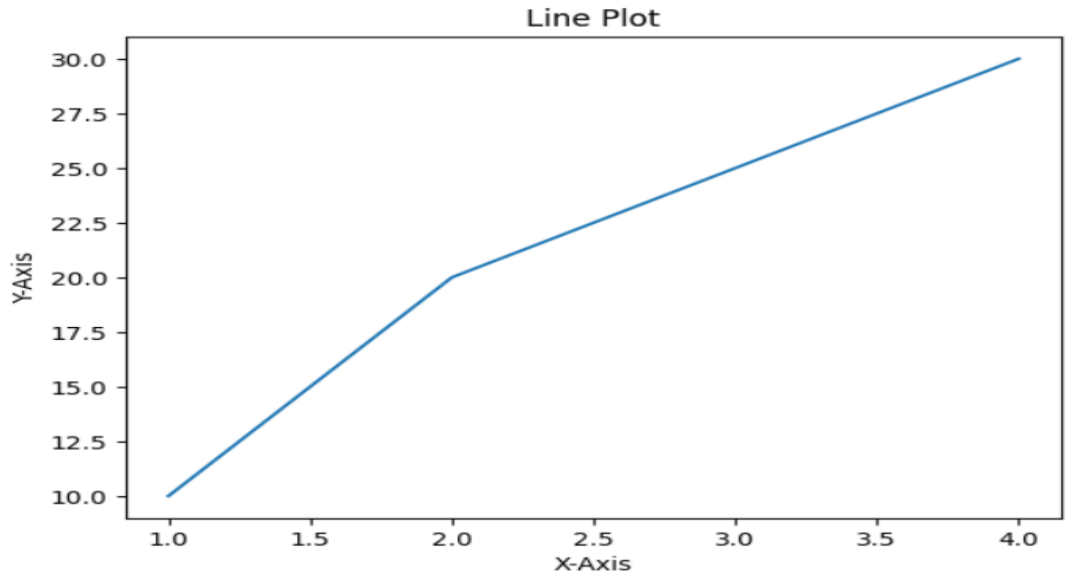
MATPLOTLIB

Displays information as a series of data points connected by straight lines.
Use Case: Trend analysis over time.

Code Snippet:

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4]
y = [10, 20, 25, 30]
plt.plot(x, y)
```

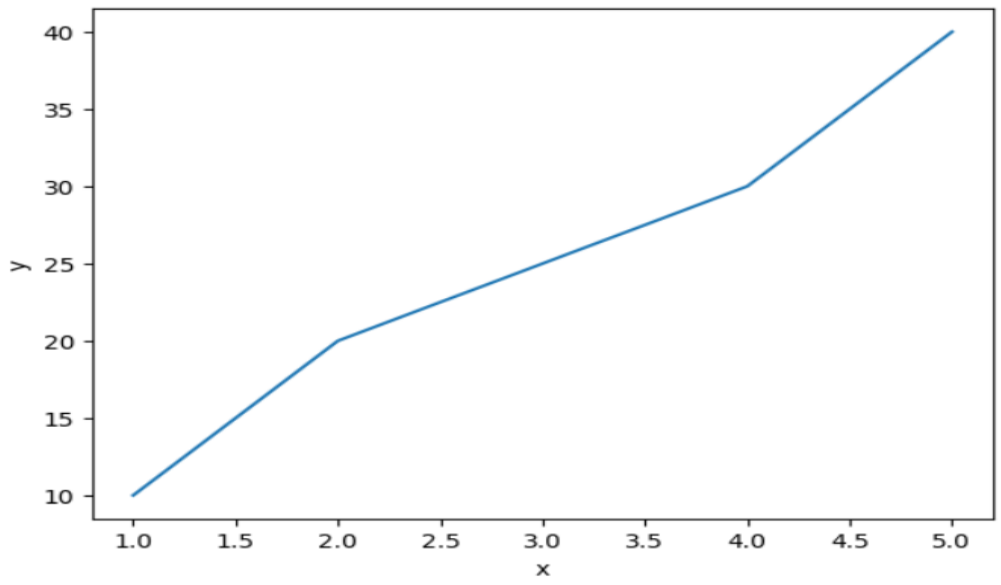
```
plt.title("Line Plot")
plt.xlabel("X-Axis")
plt.ylabel("Y-Axis")
plt.show()
```



SEABORN

<u>Code Snippet:</u>

```
import seaborn as sns
import pandas as pd
df = pd.DataFrame({'x': [1, 2, 3, 4, 5], 'y': [10, 20, 25, 30, 40]})
sns.lineplot(data=df, x='x', y='y')
```



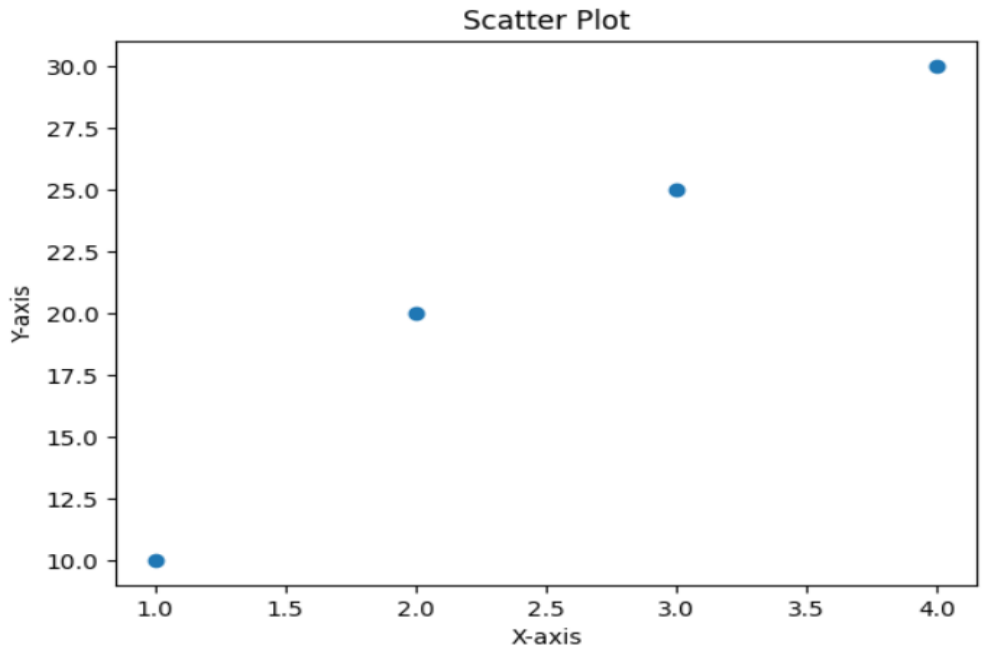
2. SCATTER PLOT

Shows the relationship between two numeric variables.
Use Case: Correlation analysis.

MATPLOTLIB

Code Snippet:

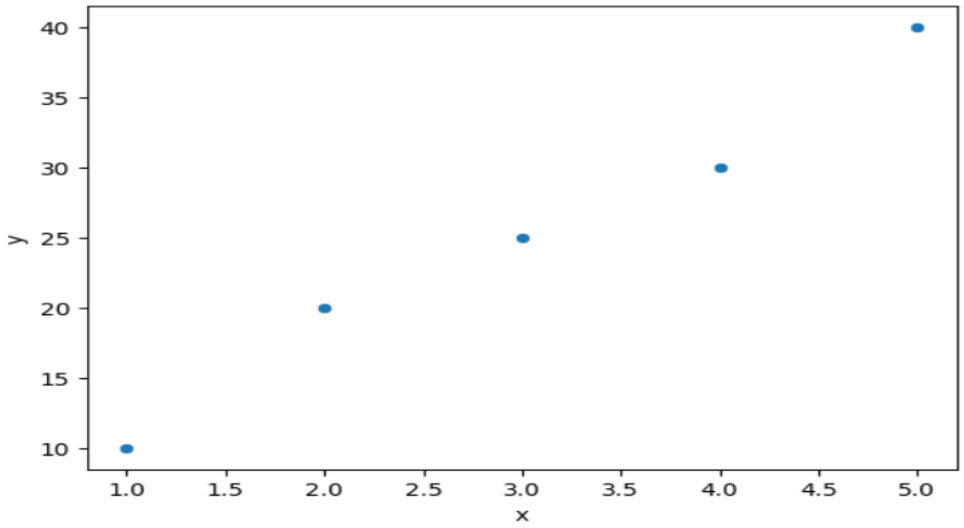
```
plt.scatter(x, y)
plt.title('Scatter Plot')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.show()
```



SEABORN

Code Snippet:

```
sns.scatterplot(data=df, x='x', y='y')
```



---

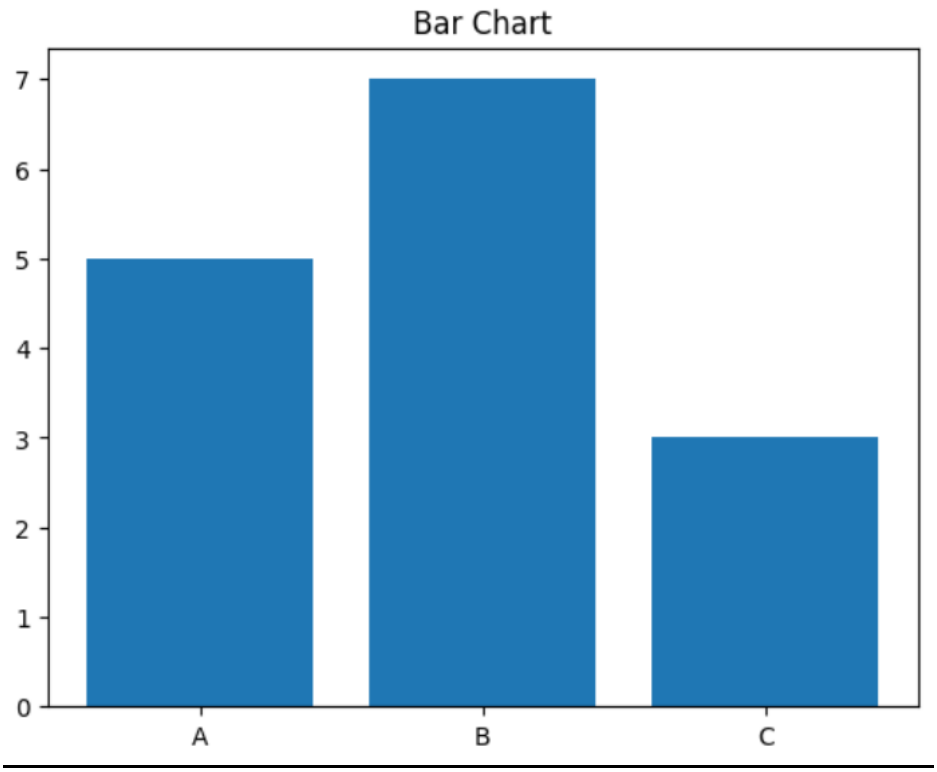
3. BAR CHART

Represents categorical data with rectangular bars.
Use Case: Comparing quantities across categories.

MATPLOTLIB

Code Snippet:

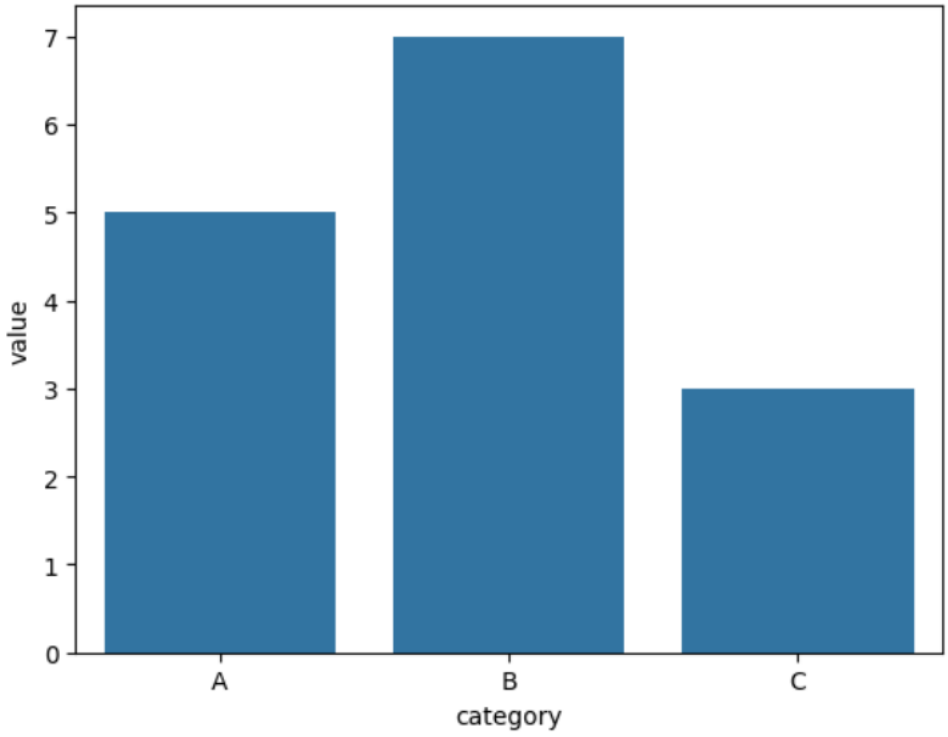
```
categories = ['A', 'B', 'C']
values = [5, 7, 3]
plt.bar(categories, values)
plt.title('Bar Chart')
plt.show()
```



SEABORN

Code Snippet:

```
df = pd.DataFrame({'category': ['A', 'B', 'C'], 'value': [5, 7, 3]})
sns.barplot(data=df, x='category', y='value')
```



---

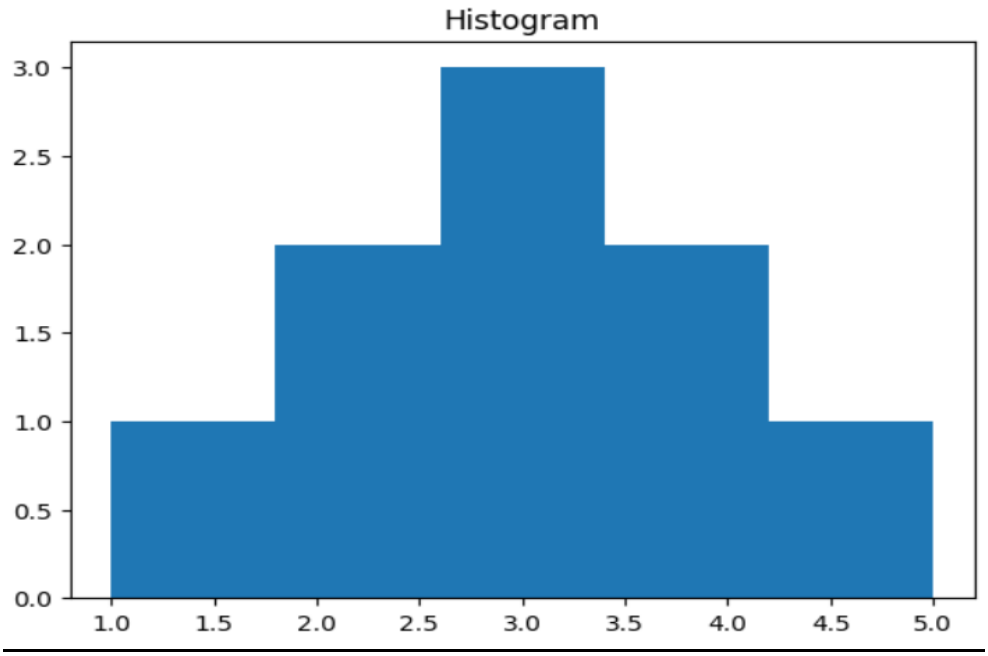
4. HISTOGRAM

Shows the distribution of a numeric variable.
Use Case: Understanding the distribution and frequency of data.

MATPLOTLIB

Code Snippet:

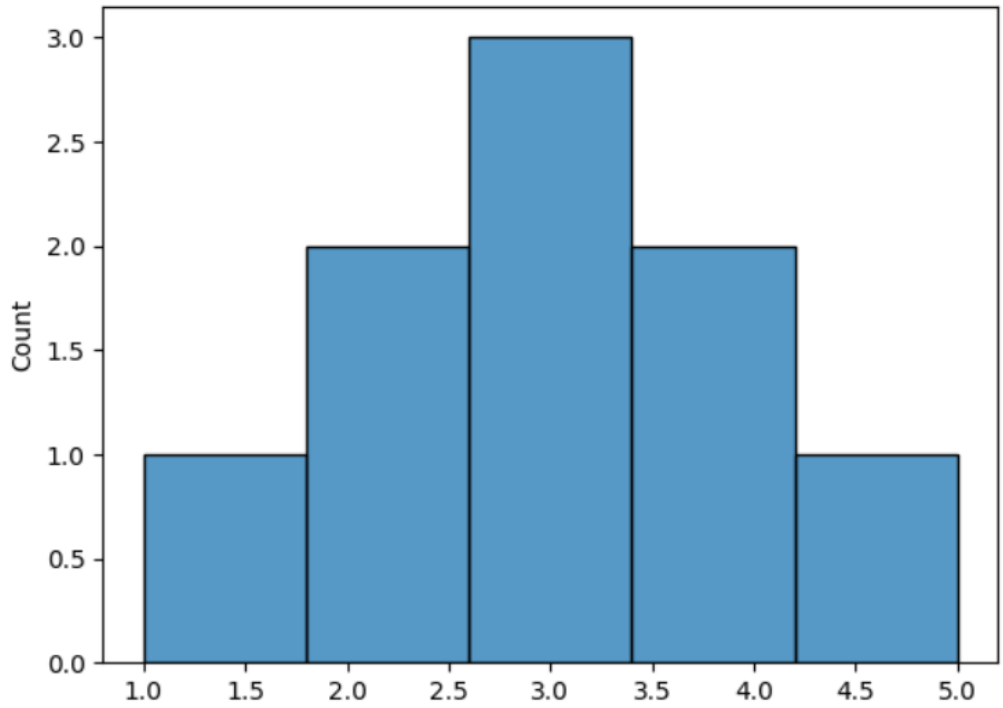
```
data = [1, 2, 2, 3, 3, 3, 4, 4, 5]
plt.hist(data, bins=5)
plt.title('Histogram')
plt.show()
```



SEABORN

Code Snippet:

sns.histplot(data, bins=5)



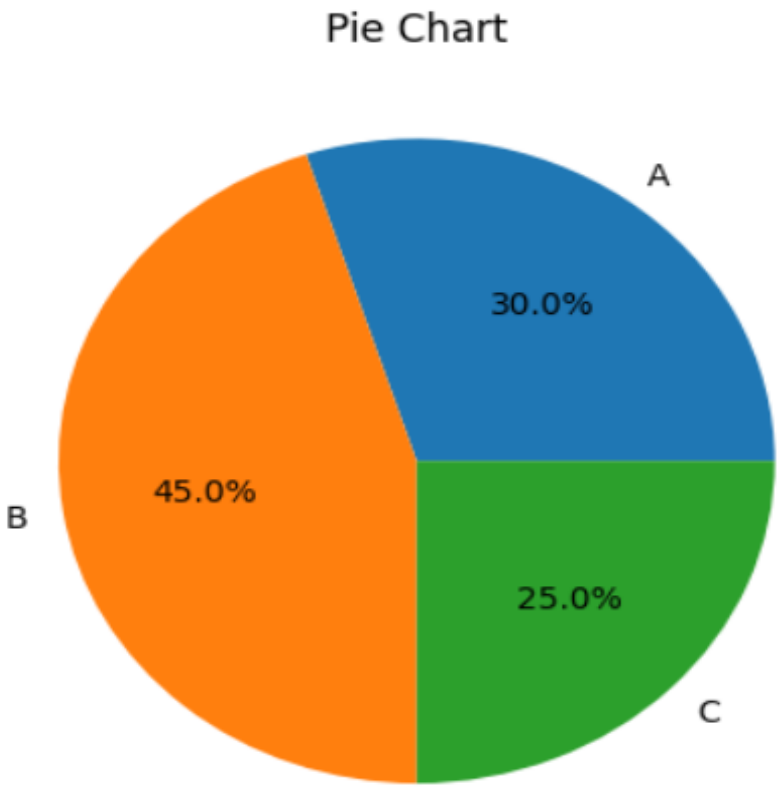
5. PIE CHART

Displays data as slices of a pie.
Use Case: Showing part-to-whole relationships.

MATPLOTLIB

Code Snippet:

```
labels = ['A', 'B', 'C']
sizes = [30, 45, 25]
plt.pie(sizes, labels=labels, autopct='%1.1f%%')
plt.title('Pie Chart')
plt.show()
```



Seaborn doesn't accept pie charts.

Differences between Matplotlib and Seaborn

Feature	Matplotlib	Seaborn
Ease of Use	Moderate	High
Customization	Very High	Moderate
Interactivity	Low (static plots)	Low
Performance	High	High
Dataset Handling	Manual	Integrates with pandas

Conclusion:
Both Matplotlib and Seaborn are essential tools for data visualization in Python.

- Use Seaborn for quick, clean, and insightful plots — especially when working with pandas' data. It is ideal for exploratory analysis.
- Use Matplotlib when you need full control and customization, such as for detailed reports or presentations.

In most cases, combining both libraries offers the perfect combination: ease of use with Seaborn, and precision with Matplotlib.