DATA SCIENCE

Task Level (Beginner):- Visualization Library Documentation
(Matplotlib, Seaborn)

Name :- Amisha Shivdas Kale

➢ Beginner-Friendly Guide to Python Visualization with **Matplotlib** and **Seaborn**

1. Introduction to Data Visualization:

- What is Data Visualization?

Data visualization is the graphical representation of information and data. By using charts, graphs, and plots, we can better understand trends, patterns, and outliers in datasets.

- Why is it Important?

  ➔ Makes data easier to understand

  ➔ Helps communicate findings clearly

  ➔ Useful for analysis, presentations, and storytelling

- Common Types of Graphs and Their Uses

Graph Type	**Use Case Example**
1. Line Plot	Showing trends over time (e.g. stock prices)
2. Bar Chart	Comparing categories(e.g. Sales by product)
3. Histogram	Understanding distribution (e.g. age of users)
4. Scatter Plot	Showing relationships (e.g. height vs. weight)
5. Box Plot	Summarizing data spread and outliers
6. Heatmap	Visualizing correlations or matrix-style data

2. What are Python Libraries?

- What is a Library in Python?

A **library** in Python is like a toolbox. It contains a collection of pre-written code that you can reuse, so you don't have to write everything from scratch. Libraries help you get things done faster and more efficiently.

For example, instead of writing code to draw a graph pixel by pixel, you can use a **visualization library** to do it in just one line of code.

- Why Use Visualization Libraries?

  ➔ They simplify the process of creating **charts and graphs**

  ➔ Allow you to **focus on your data**, not the drawing

  ➔ Offer many built-in styles, colors, and customization tools

  ➔ Help you produce **professional-quality visualizations**

- Types of Python Visualization Libraries

1. Basic Plotting Libraries

These libraries are great for creating standard charts like line plots, bar charts, pie charts, etc. They give you full control over every part of the plot.

Common Libraries:

- **Matplotlib**
  - The **foundation** for most visualizations in Python
  - Highly customizable
  - Great for static plots in reports and presentations

- **Pandas (built-in plotting)**
    - Easy, quick visualizations from DataFrames
    - Good for simple use cases
    - Based on Matplotlib

Use when:
You want static, high-quality charts and complete control over how they look.

---

2. **Statistical Visualization Libraries**

These are designed to make statistical plots easy and beautiful. They often include built-in themes and handle complex plots with very little code.

Common Libraries:

- **Seaborn**
    - Built on top of Matplotlib
    - Best for **exploratory data analysis (EDA)**
    - Includes advanced plots like violin plots, heatmaps, and pairplots

Use when:
You're analyzing datasets and want **quick, attractive, and informative charts**.

---

3. Interactive Plotting Libraries

These libraries let users interact with the visualizations: hover, zoom, filter, and more.

Common Libraries:

- **Plotly**
    - Makes interactive plots that run in a browser
    - Great for dashboards and web apps
    - Can export to HTML
- **Bokeh**
    - Similar to Plotly
    - Ideal for streaming and real-time data
    - Integrates well with web tools

Use when:

You want to **create dashboards** or let users explore the data by themselves.

---

3. Matplotlib – Your Basic Drawing Board

- What is Matplotlib?

Matplotlib is the **foundational library for plotting in Python**. It gives you full control over how your graphs look. It's flexible, powerful, and supports a wide variety of plot types.

- Key Features:

  ➔ **Highly customizable**: You can tweak every part of the plot

  ➔ **Wide range of plots**: Line, bar, pie, scatter, and more

  ➔ **Compatible** with NumPy and Pandas

  ➔ **Saves** plots to many formats (PNG, SVG, PDF, etc.)

- Best For:

  ➔ Custom and detailed visualizations

  ➔ Reports and presentations

  ➔ Situations where you need control over every visual detail

- Fun Fact:

Many other libraries like **Seaborn**, **Pandas Plot**, and even **Plotly** use Matplotlib behind the scenes for rendering plots!

Reference: To learn more about the Matplotlib library and to explore its visual representations, click on the following link: [Insert Link Here]

---

4. Seaborn – The Beautiful Plot Assistant

- What is Seaborn?

Seaborn is built **on top of Matplotlib** and makes it easier to create beautiful and **statistical plots**. It comes with attractive themes and simplified syntax.

- Key Features:

  ➔ **Stylish default themes**

  ➔ Built-in support for **Pandas DataFrames**

  ➔ Easy-to-use for **statistical visualizations**

  ➔ Great for **exploring data relationships**

- Best For:

  ➔ Data exploration (EDA)

  ➔ Quick, informative plots with little code

  ➔ Visualizing distributions, trends, and relationships

- Tip:

Use Seaborn when you're working with **structured data (like tables)** and want to visualize trends quickly and beautifully.

Reference: To learn more about the Seaborn library and to explore its visual representations, click on the following link: [Insert Link Here]

❖ Graph Types in Matplotlib

1. Line Plot

Description:

Displays information as a series of data points connected by straight lines.

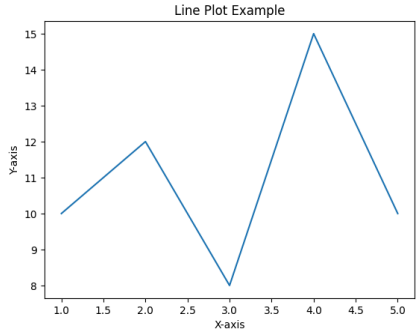
Use Case:

Track monthly sales, temperature changes, or website visitors over days.

Code Example:

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]

y = [10, 12, 8, 15, 10]

plt.plot(x, y)

plt.title("Line Plot Example")

plt.xlabel("X-axis")

plt.ylabel("Y-axis")

plt.show()
```

Output :-



2. Bar Chart

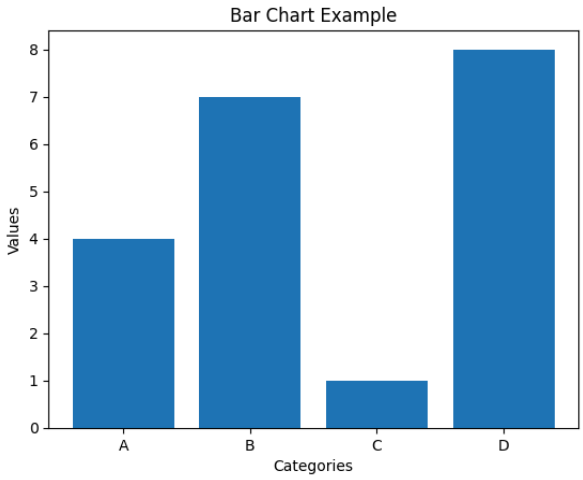
Description:
Uses rectangular bars to show quantities for different categories.

Use Case:
Compare sales across different products or regions.

Code Example:

```
categories = ['A', 'B', 'C', 'D']
values = [4, 7, 1, 8]

plt.bar(categories, values)
plt.title("Bar Chart Example")
plt.xlabel("Categories")
plt.ylabel("Values")
plt.show()
```

Output :-



3. Histogram

Description:

Shows the frequency distribution of a single numeric variable by dividing the data into bins.

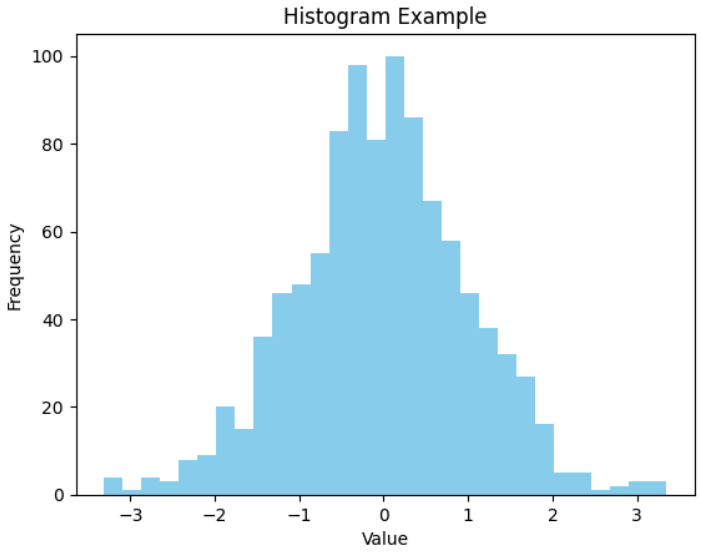
Use Case:

Understand the age distribution of a group of people.

Code Example:

```
import numpy as np

data = np.random.randn(1000)

plt.hist(data, bins=30, color='skyblue')
plt.title("Histogram Example")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()
```

Output:-



4. Scatter Plot

Description:

Displays data points based on two variables. Helps identify relationships or patterns.

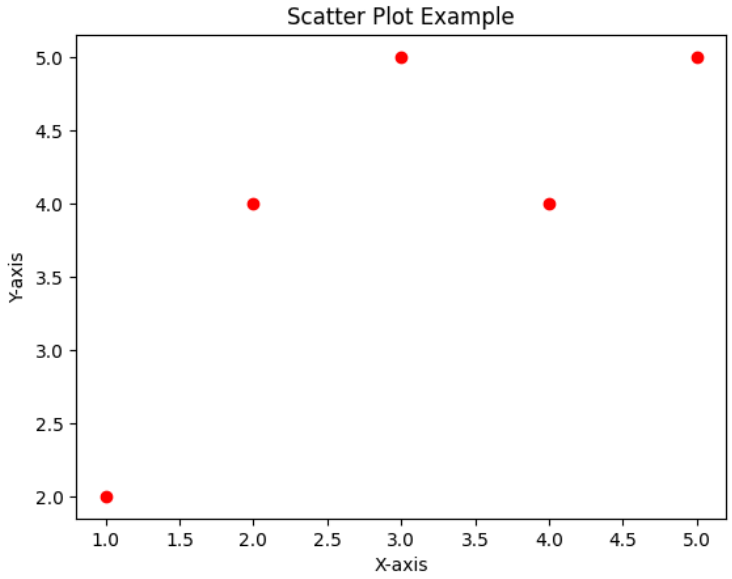
Use Case:

Check if more study hours lead to higher exam scores.

Code Example:

```
x = [1, 2, 3, 4, 5]
y = [2, 4, 5, 4, 5]

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

Output:-



5. Pie Chart

Description:

A circular chart divided into slices to illustrate proportions.

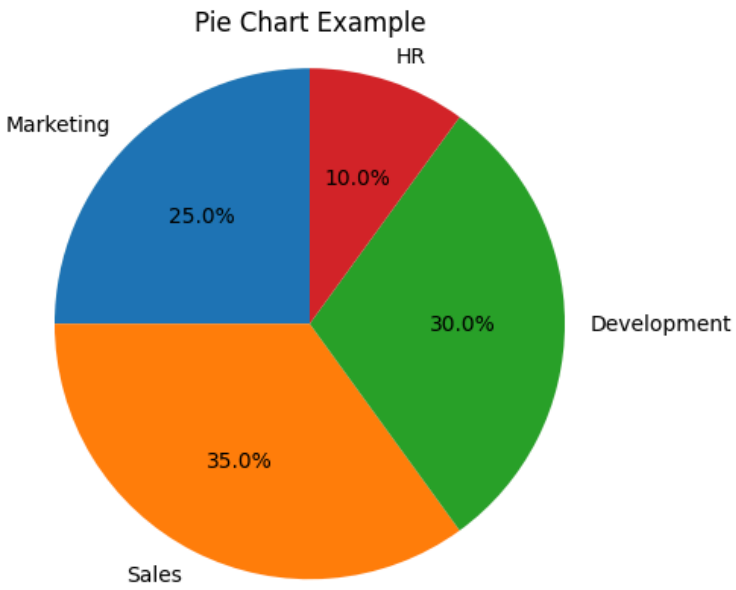
Use Case:

Show how a company budget is divided between departments.

Code Example:

```
labels = ['Marketing', 'Sales', 'Development', 'HR']
sizes = [25, 35, 30, 10]

plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)
plt.title("Pie Chart Example")
plt.axis('equal')
plt.show()
```

Output:-



6. Box Plot

Description:

Displays the distribution of a dataset through quartiles, highlighting medians and outliers.

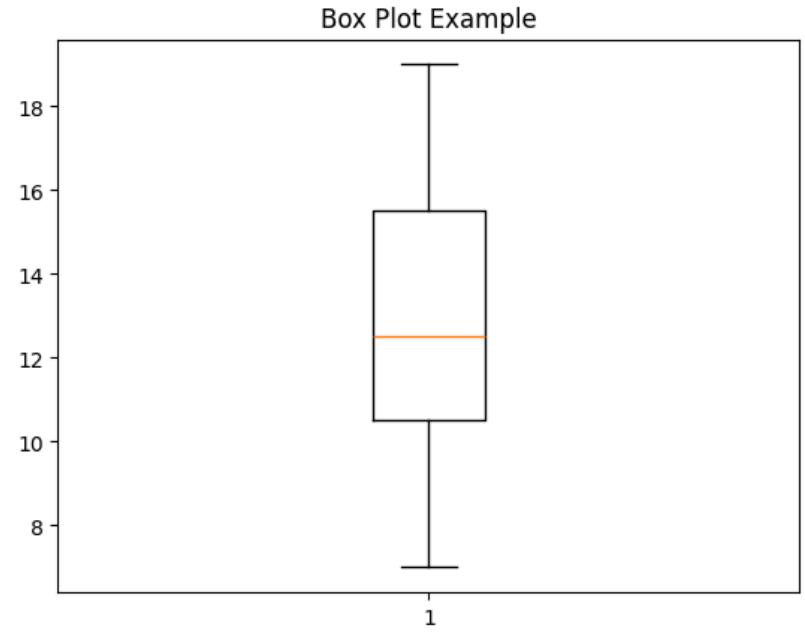
Use Case:

Compare salary distribution between two departments.

Code Example:

```
data = [7, 15, 13, 9, 12, 19, 11, 17]

plt.boxplot(data)
plt.title("Box Plot Example")
plt.show()
```

Output:-



❖ Graph Types in Seaborn

1. Line Plot

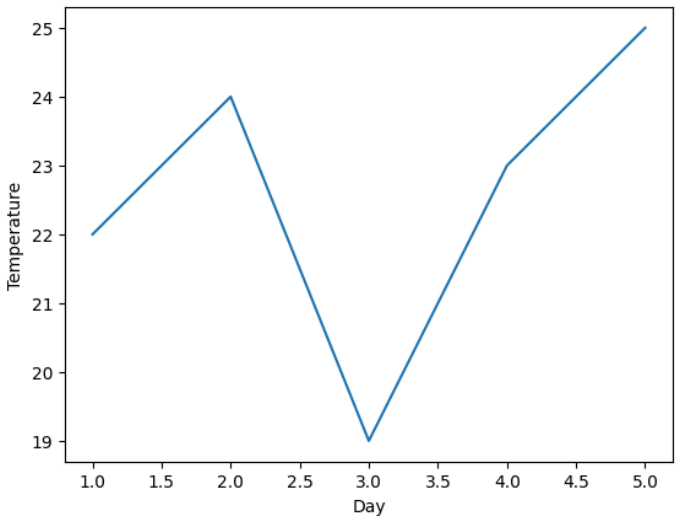
Description:
Displays trends over time or a sequence, similar to Matplotlib but with built-in support for DataFrames.

Use Case:
Visualizing temperature changes across days.

Code Example:

```
import seaborn as sns
import pandas as pd

data = pd.DataFrame({
    'Day': [1, 2, 3, 4, 5],
    'Temperature': [22, 24, 19, 23, 25]
})

sns.lineplot(x='Day', y='Temperature', data=data)
```

Output:-



2. Bar Chart

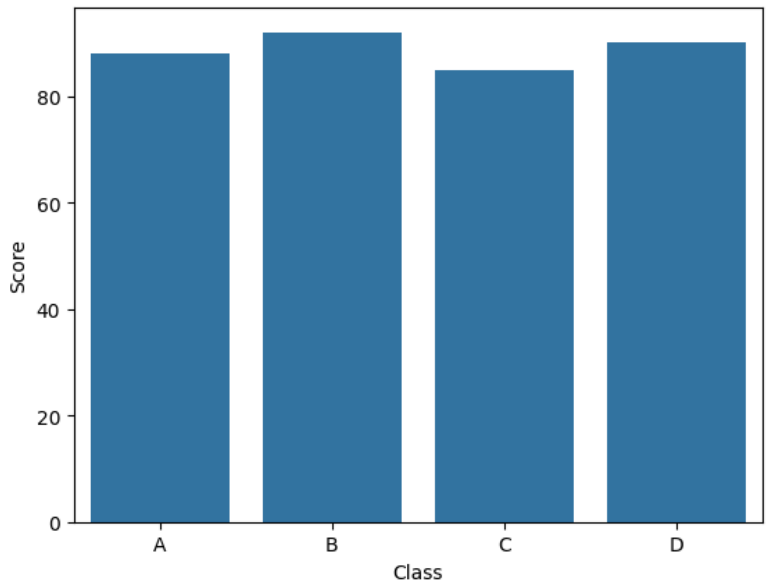
Description:
Shows the average (by default) of a numerical variable for each category.

Use Case:
Compare average test scores between different classes.

Code Example:

```
data = pd.DataFrame({
    'Class': ['A', 'B', 'C', 'D'],
    'Score': [88, 92, 85, 90]
})

sns.barplot(x='Class', y='Score', data=data)
```

Output:-



3. Histogram

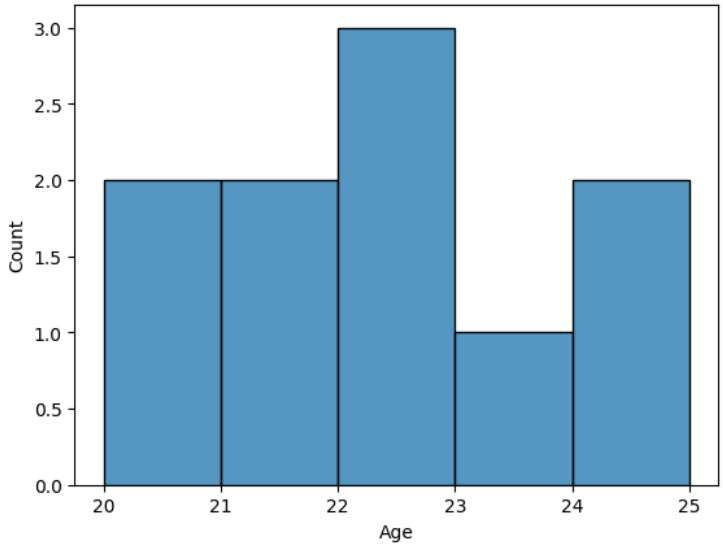
Description:
Displays the distribution of a single variable by counting the number of observations in each bin.

Use Case:
Analyze distribution of student ages.

Code Example:

```
data = pd.DataFrame({'Age': [20, 21, 22, 23, 22, 21, 20, 24, 25, 22]})

sns.histplot(data['Age'], bins=5)
```

Output:-



4. Scatter Plot

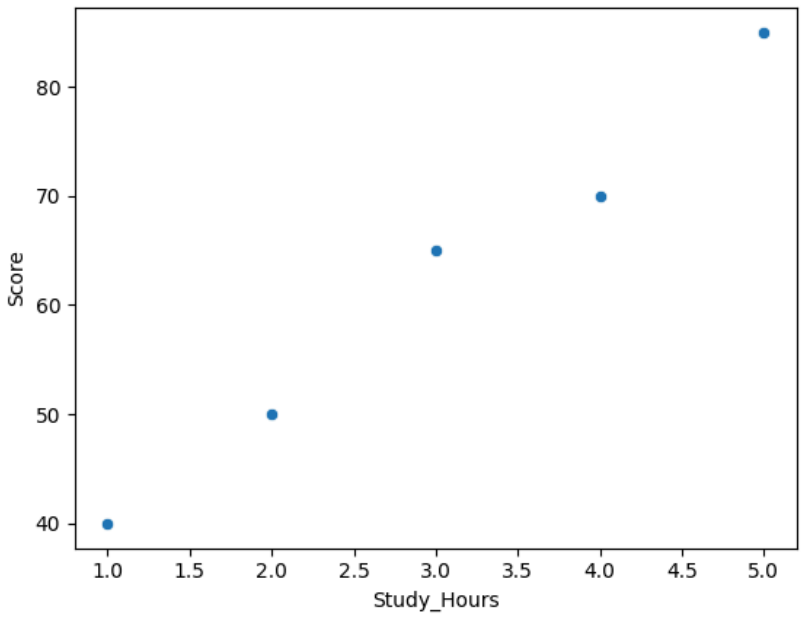
Description:
Plots individual data points to explore relationships between two variables.

Use Case:
Check the relation between study time and marks.

Code Example:

```
data = pd.DataFrame({
    'Study_Hours': [1, 2, 3, 4, 5],
    'Score': [40, 50, 65, 70, 85]
})

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

Output:-



5. Box Plot

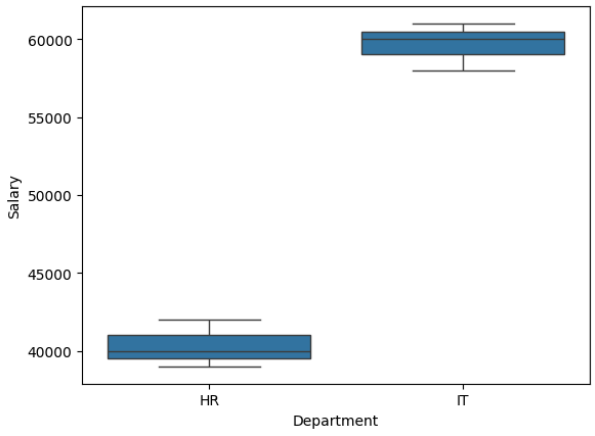
Description:
Shows median, quartiles, and outliers for numeric data grouped by categories.

Use Case:
Compare salary ranges across job titles.

Code Example:

```
data = pd.DataFrame({
    'Department': ['HR', 'IT', 'HR', 'IT', 'HR', 'IT'],
    'Salary': [40000, 60000, 42000, 58000, 39000, 61000]
})

sns.boxplot(x='Department', y='Salary', data=data)
```

Output:-



❖ Comparison: Matplotlib vs. Seaborn

Points	Matplotlib	Seaborn
1. Ease of use	Slightly difficult to use , required more lines of codes and manual settings .	Easy to use , automatically handles styling and layout with less code.
2. Customization	Highly customizable - you can control every small part of the graph .	Limited customization- enough for most cases , but less control.
3. Interactivity	Creates static (non clickable) graphs.	Also creates static graphs , built on Top of matplotlib
4. Graph Appearance	Basic and plain by default , needs manual styling to look good.	Beautiful and stylish by default , no need for extra styling .
5. Learning curve	Steeper-tasks more time to learn for beginners.	Beginner-friendly Easier to learn and use, especially with DataFrame .
6. Performance	Handles large datasets well if optimized.	Work well with small and medium datasets but may slow down with large ones .
7. Type of Graphs	Great for basic plots like line , bar, pie, scatter, etc.	Best for statistical plots like box plot, heatmaps and distribution plot .
8. Data Handling	Works with lists , arrays , and Pandas DataFrame .	Work best with pandas DataFrame
9. Use case	Suitable for detailed and custom visualizations , such as in research .	Suitable for quick analysis and creating clean , professional plots
10. Appearance control	You must manually set colors, titles, axis , etc.	Automatically applies colors, spacing, and themes for a polished look

❖ When to Use What?

- **Use Matplotlib if:**

  ➢ You need **full control** over the plot (like in research papers).

  ➢ You're building **custom or complex visualizations**.

  ➢ You want to **fine-tune every detail**, including axis positions, legends, ticks, etc.

- **Use Seaborn if:**

  ➢ You want **fast and attractive plots with minimal code**.

  ➢ You're doing **exploratory data analysis (EDA)**.

  ➢ You're working with **data stored in DataFrames** and want to analyze patterns and distributions quickly.

❖ Final Thoughts

- **Matplotlib** is like the foundation or "engine" of Python plotting. It's powerful and flexible but requires more effort.
- **Seaborn** is like a stylish upgrade that runs on top of Matplotlib. It helps you create beautiful plots easily, especially for statistics and group-based comparisons.

➔ Think of Seaborn as a **friendly assistant** who handles the design, while Matplotlib is the **toolbox** you use when you want to do things your own way.
