Task Level (Beginner):

Python Visualization Libraries Documentation

Guide Selected Libraries: Matplotlib and Plotly

Libraries Covered:
1. Matplotlib
2. Seaborn

1. Library Overview

1.1 Matplotlib

Description:
Matplotlib is the most widely used and foundational Python visualization library. It provides extensive control over plot appearance and supports multiple backends for rendering. It's highly customizable and forms the basis for many other visualization libraries (including Seaborn).

Key Features:
-Low-level control for fine-tuned customization.
- Wide range of static, animated, and interactive plots.
- Works well with NumPy and Pandas.

Typical Use Cases:
- Creating static publication-quality plots.
- Customizing every element of a plot.
- Integrating plots into GUIs and applications.

1.2 Seaborn

Description:
Seaborn is built on top of Matplotlib, offering a high-level API for statistical data visualization. It comes with built-in themes, color palettes, and functions for easily creating complex visualizations with minimal code.

Key Features:
- Beautiful default themes and color styles.
- Specialized functions for statistical visualization.
- Integrates smoothly with Pandas DataFrames.

Typical Use Cases:
- Quick and attractive statistical plots.
- Visualizing relationships between variables.
- Producing heatmaps, categorical plots, and distribution plots.

2. Graph Types & Examples

2.1 Matplotlib Graph Types

a) Line Plot

Displays data points connected by straight lines. Use Case: Time series visualization.

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]
y = [2, 5, 7, 1, 6]

plt.plot(x, y, marker='o')
plt.title("Line Plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```

b) Scatter Plot

Displays data points as dots in a Cartesian plane. Use Case: Analyzing relationships between two numeric variables.

```
import matplotlib.pyplot as plt

x = [5, 7, 8, 7, 6, 9, 5, 8, 7]
y = [99, 86, 87, 88, 100, 86, 103, 87, 94]

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

c) Bar Chart

Uses rectangular bars to represent data values. Use Case: Comparing categorical data.

```
categories = ['A', 'B', 'C', 'D']
values = [3, 7, 2, 5]

plt.bar(categories, values, color='skyblue')
plt.title("Bar Chart")
plt.show()
```

d) Histogram

Shows distribution of numerical data. Use Case: Frequency distribution analysis.

```
data = [22, 87, 5, 43, 56, 73, 55, 54, 11, 20, 51, 5, 79, 31, 27]

plt.hist(data, bins=5, color='green', edgecolor='black')
plt.title("Histogram")
plt.show()
```

e) Pie Chart

Displays proportions as slices of a circle. Use Case: Showing percentage composition.

```
sizes = [215, 130, 245, 210]
labels = ['A', 'B', 'C', 'D']

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

2.2 Seaborn Graph Types

a) Line Plot

```
import seaborn as sns
import pandas as pd

df = pd.DataFrame({
    "x": [1, 2, 3, 4, 5],
    "y": [2, 5, 7, 1, 6]
})

sns.lineplot(data=df, x="x", y="y", marker="o")
```

b) Scatter Plot

```
df = pd.DataFrame({
    "x": [5, 7, 8, 7, 6, 9, 5, 8, 7],
    "y": [99, 86, 87, 88, 100, 86, 103, 87, 94]
})

sns.scatterplot(data=df, x="x", y="y", color="red")
```

c) Bar Plot

```
df = pd.DataFrame({
    "category": ["A", "B", "C", "D"],
    "values": [3, 7, 2, 5]
})

sns.barplot(data=df, x="category", y="values", palette="Blues_d")
```

d) Histogram

```
data = [22, 87, 5, 43, 56, 73, 55, 54, 11, 20, 51, 5, 79, 31, 27]

sns.histplot(data, bins=5, kde=True, color='green')
```

e) Heatmap

```
import numpy as np

matrix_data = np.random.rand(4, 4)
sns.heatmap(matrix_data, annot=True, cmap="YlGnBu")
```

3. Comparison Table

Feature	Matplotlib	Seaborn
Ease of Use	Steeper learning curve, more verbose	High-level API, simpler syntax

Customization	Very high, control over every element	Limited but sufficient for most use cases
Aesthetics	Plain by default	Beautiful default themes
Interactivity	Limited without extra tools	Similar limitations
Performance	Good for large datasets	Slightly slower for huge datasets
Best For	Publication-quality plots	Quick statistical graphics