👋 Hi, I'm **A GOKUL KRISHNA**

📊 **Internship in Data Science**

🏢 at **ShadowFox**

Python Visualization Libraries: Matplotlib and Seaborn

In [ ]:

🎨 Matplotlib

> **Matplotlib** is one of the most popular and powerful Python libraries for data visualization. It allows for creating a wide variety of static, animated, and interactive plots.

🔑 Key Features

✅ Highly customizable plots with detailed control
📁 Supports multiple file formats (PNG, PDF, SVG)
🛠️ Compatible with GUI toolkits (Tkinter, Qt, etc.)
📊 Ideal for both 2D and limited 3D plotting

📚 Typical Use Cases

- 📖 **Academic Research** – Create publication-quality plots.
- 📈 **Data Exploration** – Visualize trends and data distributions.
- 🧩 **Custom Visualizations** – Tailored plots for specific needs.

🌈 Seaborn

> **Seaborn** is a high-level API built on top of Matplotlib that simplifies the process of creating visually attractive and statistically meaningful graphics.

🌟 Key Features

| |
|---|
| 🧠 Simplified syntax for quick and efficient plotting |
| 🎨 Beautiful default themes and color palettes |

📚 Typical Use Cases

- 🔍 **Exploratory Data Analysis (EDA)** – Understand your data quickly.
- 📈 **Statistical Visualization** – Relationships and distributions.
- 🖼️ **Beautiful Plots** – Create insightful and presentable charts fast.

In [1]:
```python
import matplotlib.pyplot as plt

# Sample data
x = [0, 1, 2, 3, 4, 5]
y = [0, 1, 4, 9, 16, 25]

# Create line plot
plt.plot(x, y)
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Line Plot Example')
plt.show()
```
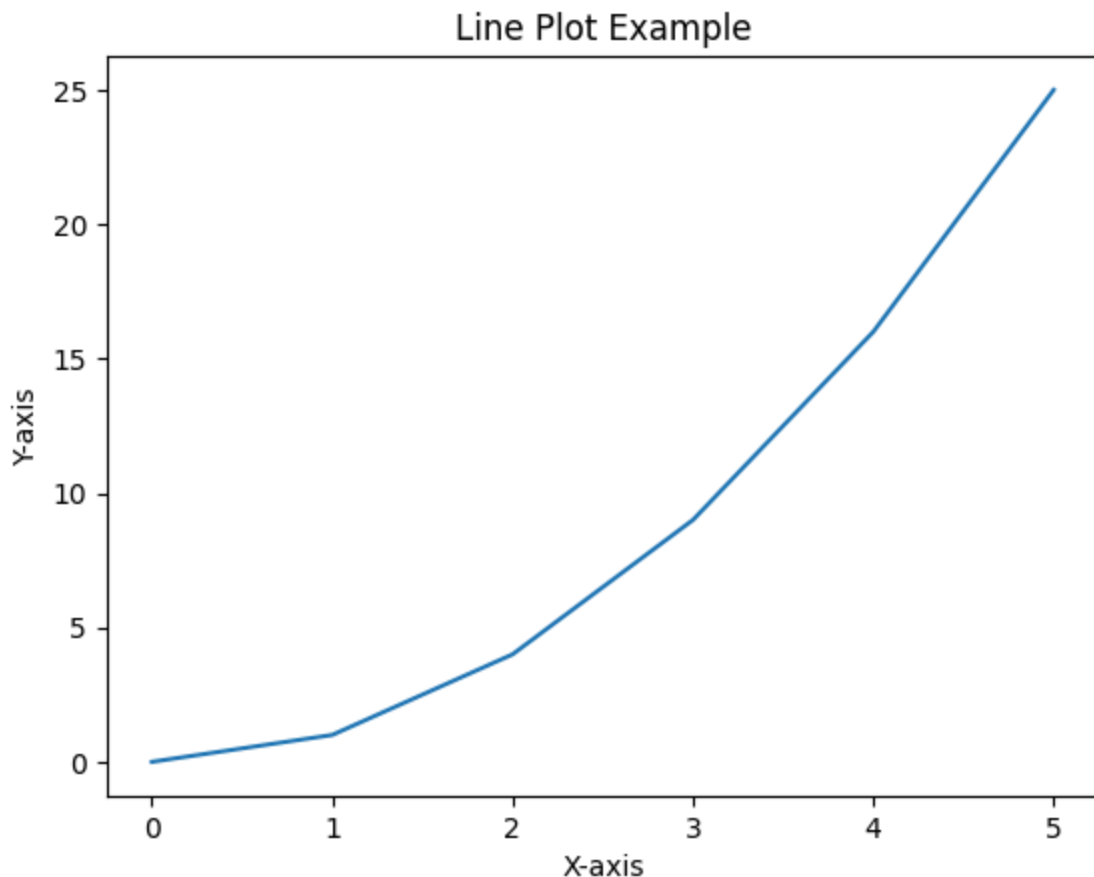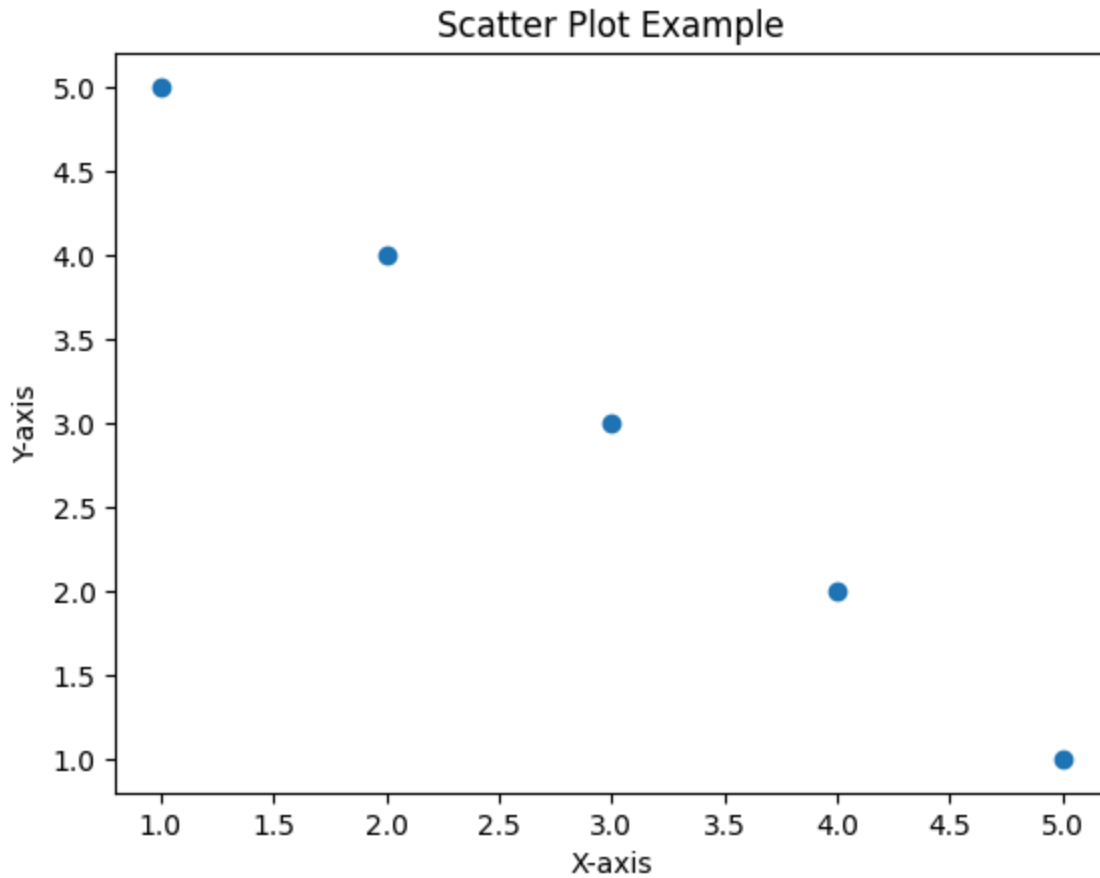
In [2]:
```python
import matplotlib.pyplot as plt

# Sample data
x = [1, 2, 3, 4, 5]
y = [5, 4, 3, 2, 1]

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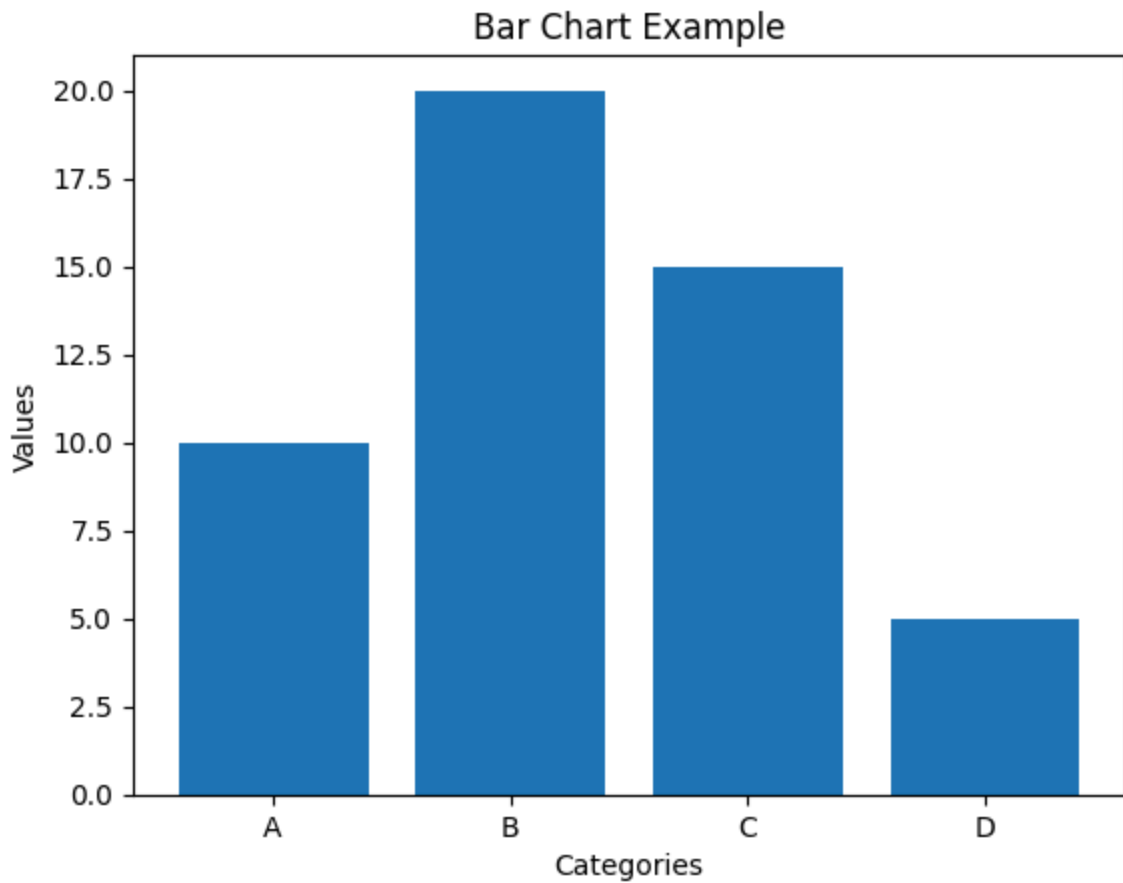

In [3]:
```python
import matplotlib.pyplot as plt

# Sample data
categories = ['A', 'B', 'C', 'D']
values = [10, 20, 15, 5]

# Create bar chart
plt.bar(categories, values)
plt.xlabel('Categories')
plt.ylabel('Values')
plt.title('Bar Chart Example')
plt.show()
```
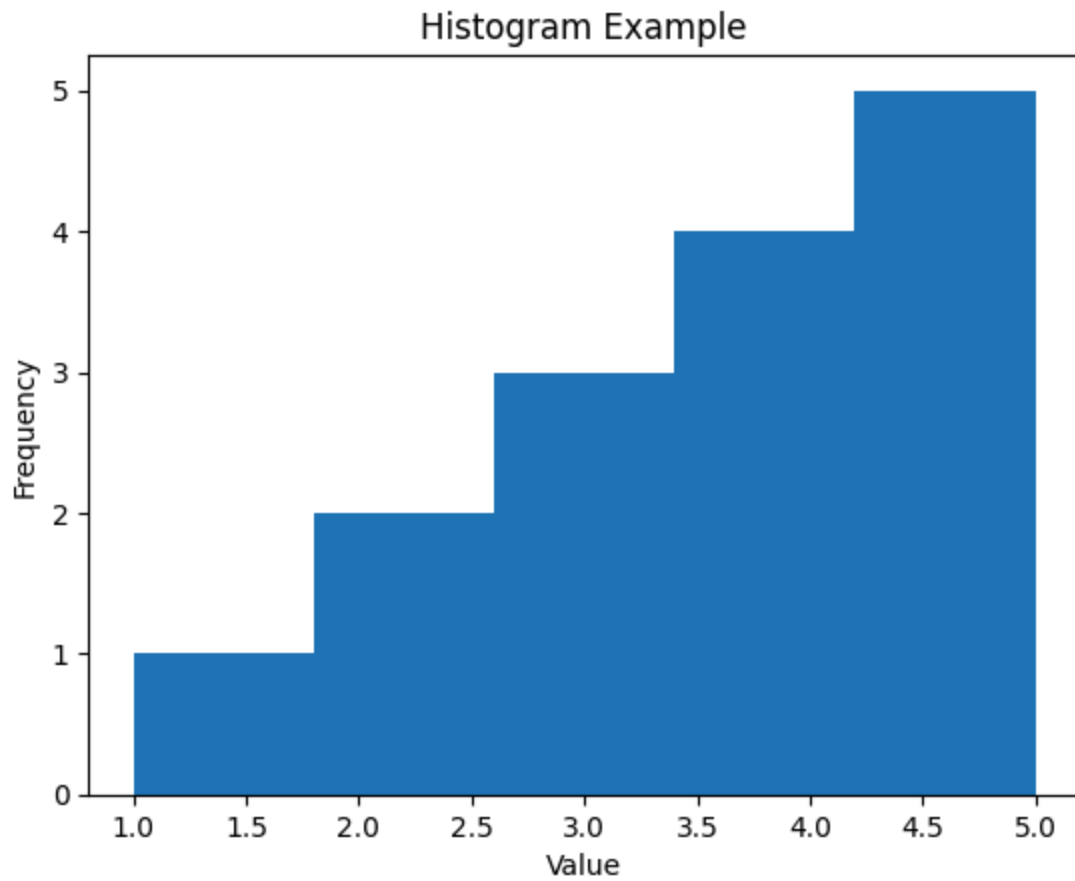
Bar Chart Example


```
In [4]:  import matplotlib.pyplot as plt

         # Sample data
         data = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5]

         # Create histogram
         plt.hist(data, bins=5)
         plt.xlabel('Value')
         plt.ylabel('Frequency')
         plt.title('Histogram Example')
         plt.show()
```
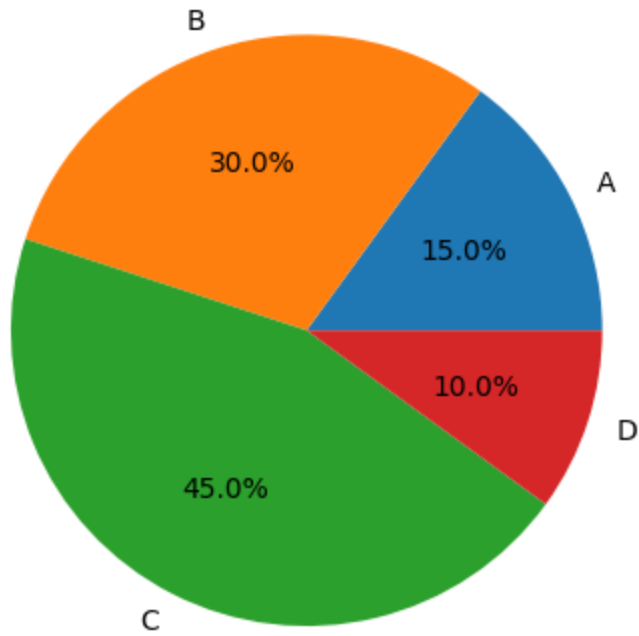
Histogram Example

```python
import matplotlib.pyplot as plt

# Sample data
labels = ['A', 'B', 'C', 'D']
sizes = [15, 30, 45, 10]

# Create pie chart
plt.pie(sizes, labels=labels, autopct='%1.1f%%')
plt.title('Pie Chart Example')
plt.show()
```
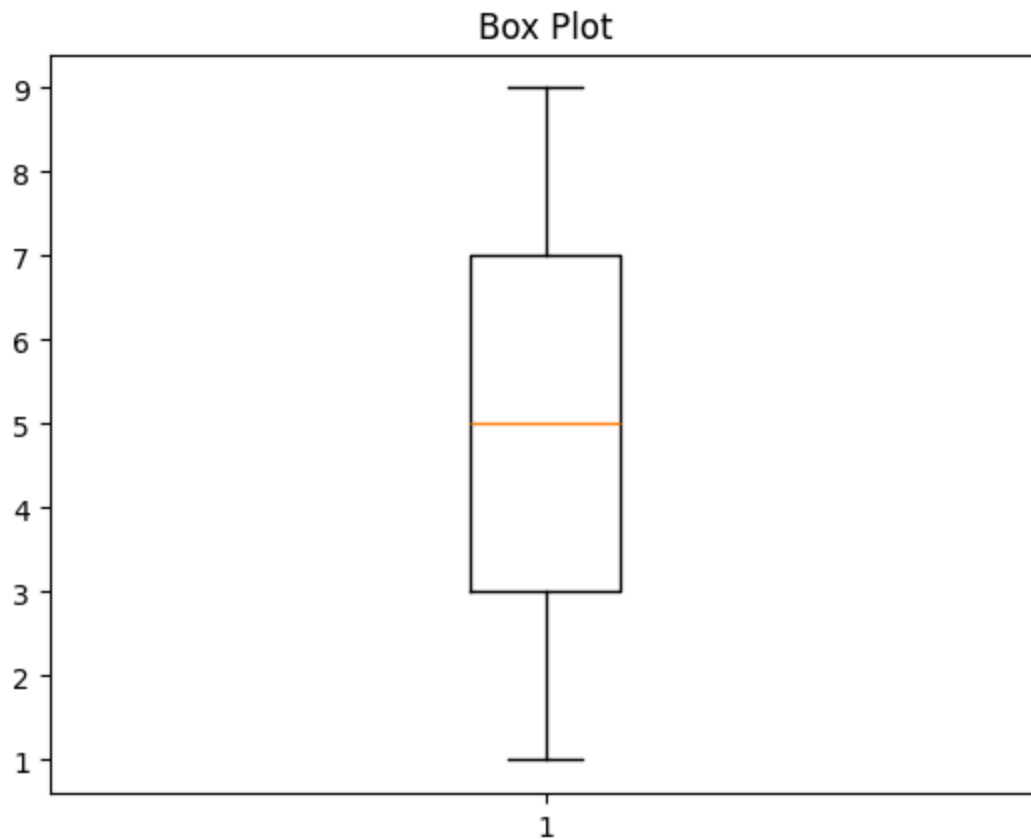
Pie Chart Example

In [6]:
```python
import matplotlib.pyplot as plt

data = [1, 2, 3, 4, 5, 6, 7, 8, 9]

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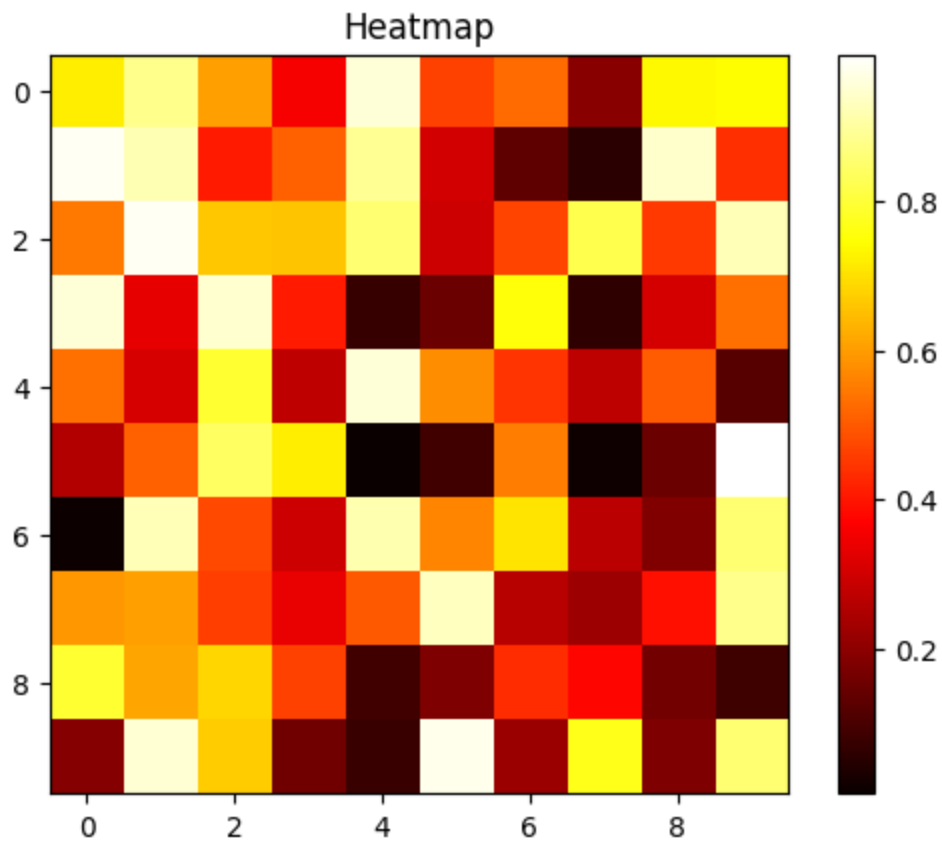
Box Plot


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

data = np.random.rand(10, 10)

plt.imshow(data, cmap='hot', interpolation='nearest')
plt.title("Heatmap")
plt.colorbar()
plt.show()
```
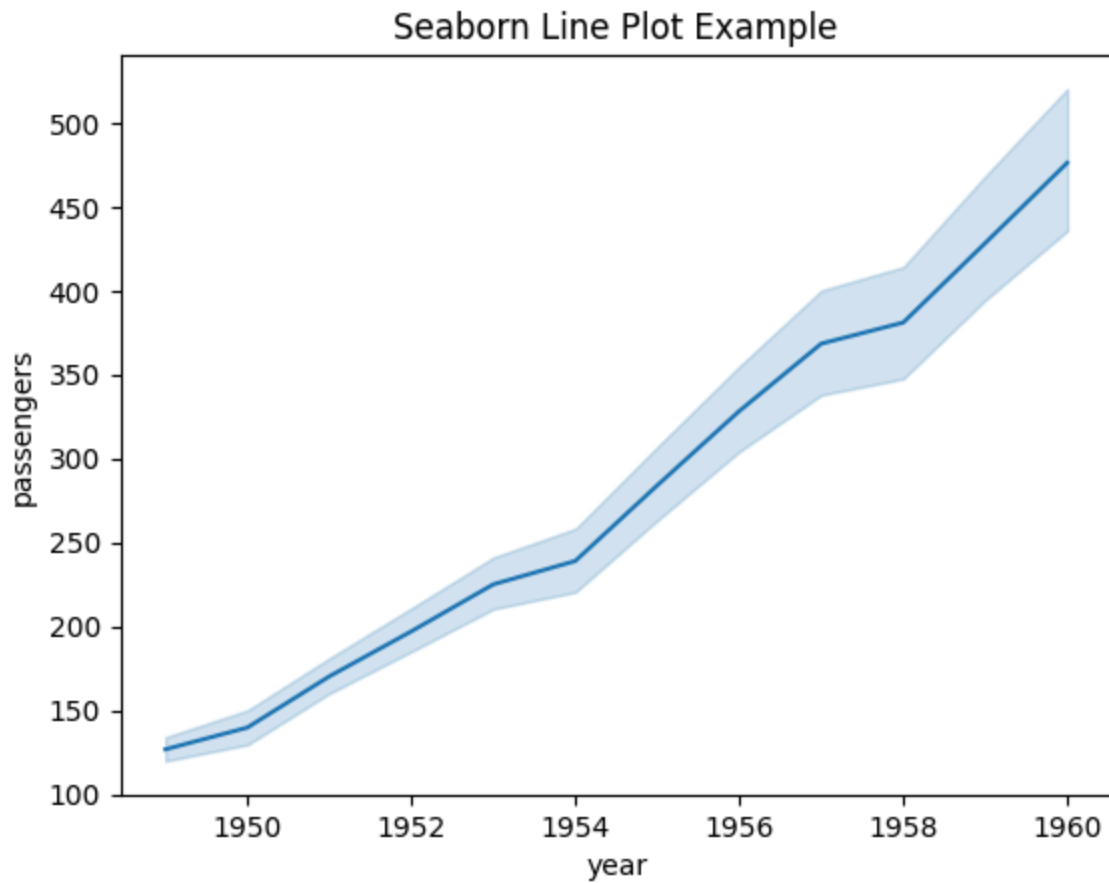
Heatmap

```
In [8]:  import seaborn as sns
         import matplotlib.pyplot as plt

         # Sample data
         data = sns.load_dataset("flights")

         # Create line plot
         sns.lineplot(x="year", y="passengers", data=data)
         plt.title('Seaborn Line Plot Example')
         plt.show()
```

Seaborn Line Plot Example

In [9]:
```python
import seaborn as sns
import matplotlib.pyplot as plt

# Sample data
data = sns.load_dataset("tips")

# Create scatter plot
sns.scatterplot(x="total_bill", y="tip", data=data)
plt.title('Seaborn Scatter Plot Example')
plt.show()
```

Seaborn Scatter Plot Example
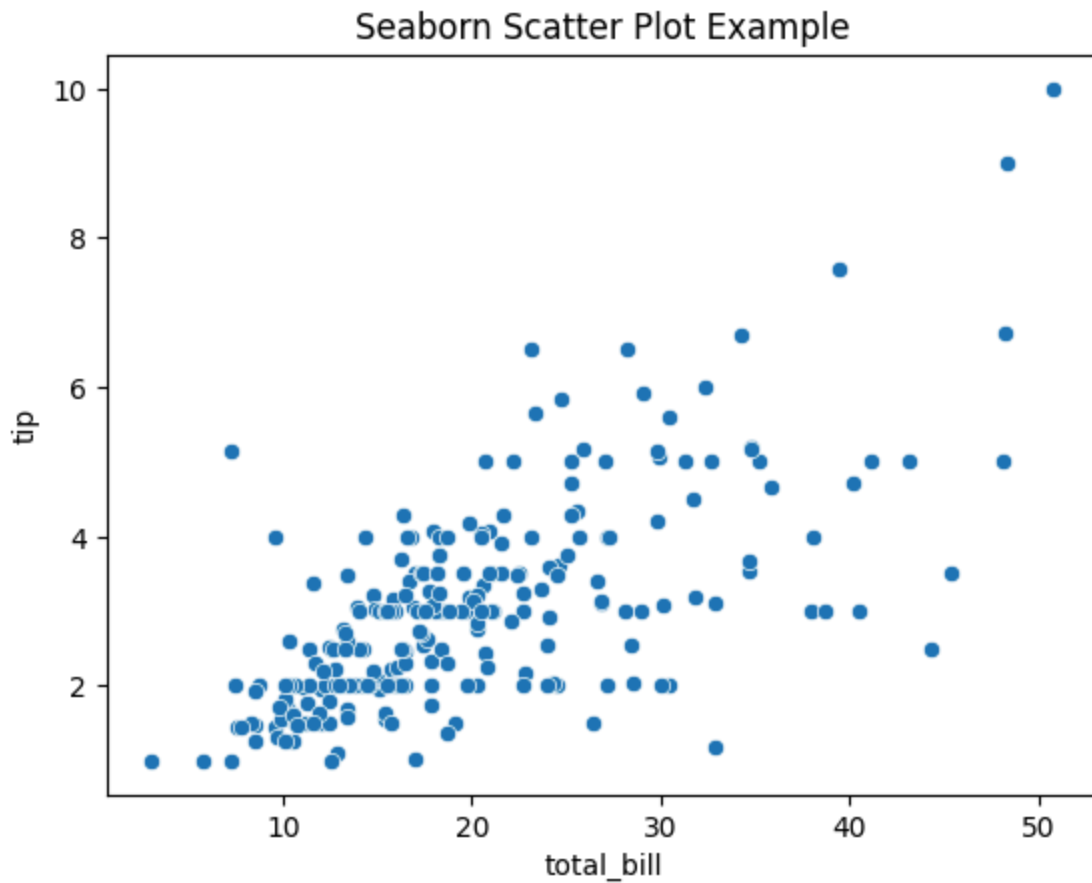

```
In [10]:  import seaborn as sns
          import matplotlib.pyplot as plt

          # Sample data
          data = sns.load_dataset("tips")

          # Create bar chart
          sns.barplot(x="day", y="total_bill", data=data)
          plt.title('Seaborn Bar Chart Example')
          plt.show()
```

Seaborn Bar Chart Example

```
In [11]:  import seaborn as sns
          import matplotlib.pyplot as plt

          # Sample data
          data = sns.load_dataset("iris")

          # Create histogram
          sns.histplot(data["sepal_length"], kde=True)
          plt.title('Seaborn Histogram Example')
          plt.show()
```
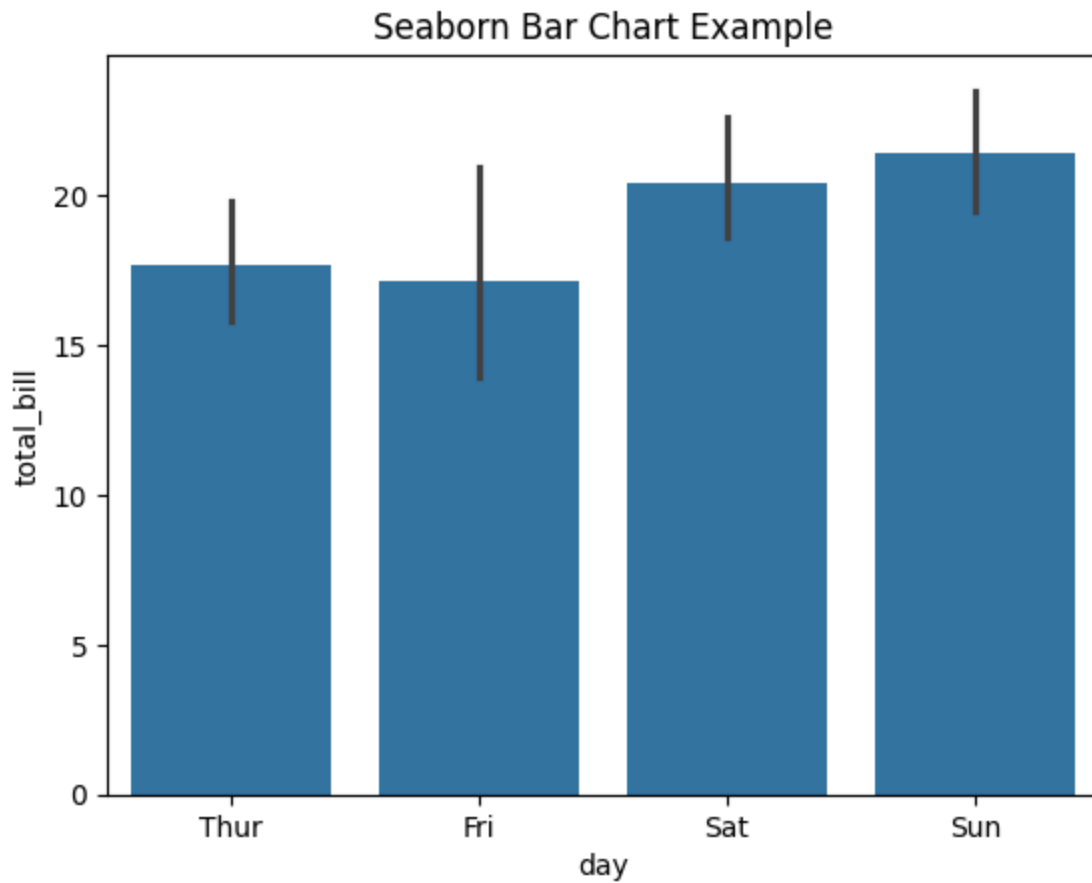
Seaborn Histogram Example


```
In [12]:  import seaborn as sns
          import matplotlib.pyplot as plt

          # Sample data
          data = sns.load_dataset("flights")
          pivot_data = data.pivot(index="month", columns="year", values="passengers")

          # Create heatmap
          sns.heatmap(pivot_data, annot=True, fmt="d")
          plt.title('Seaborn Heatmap Example')
          plt.show()
```
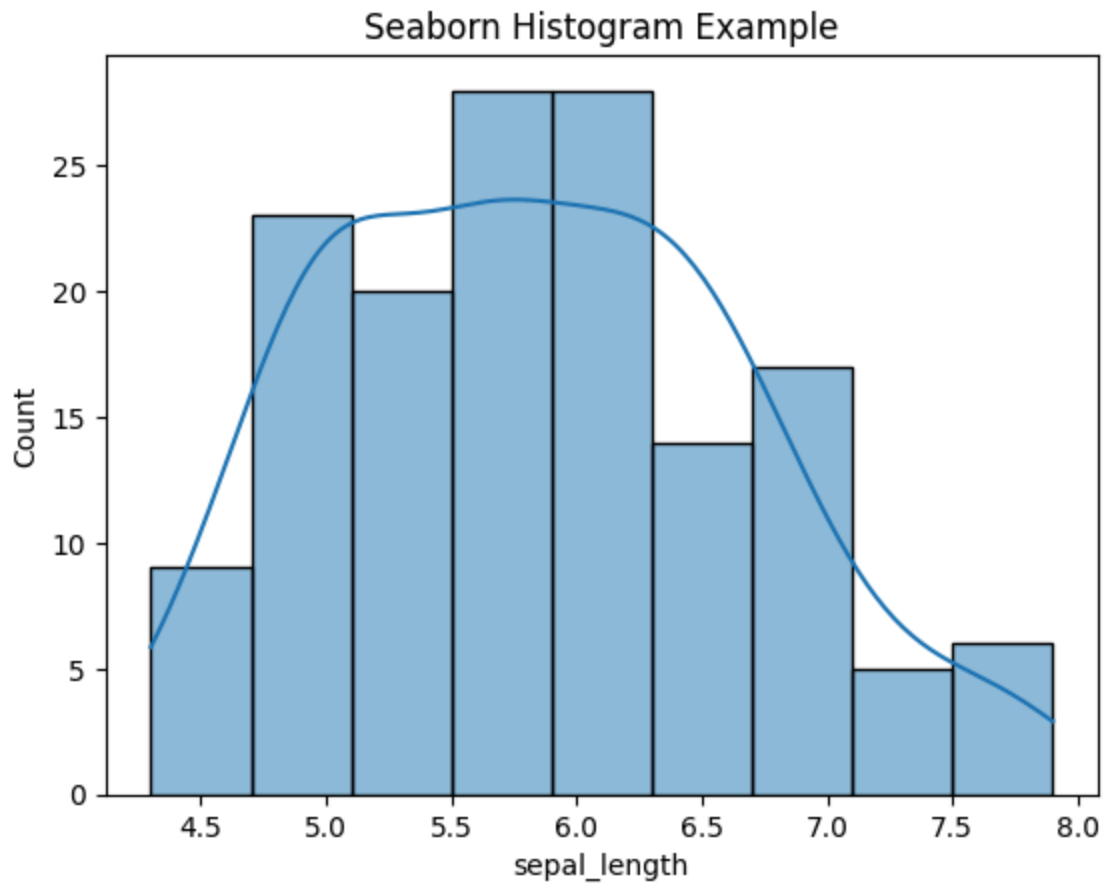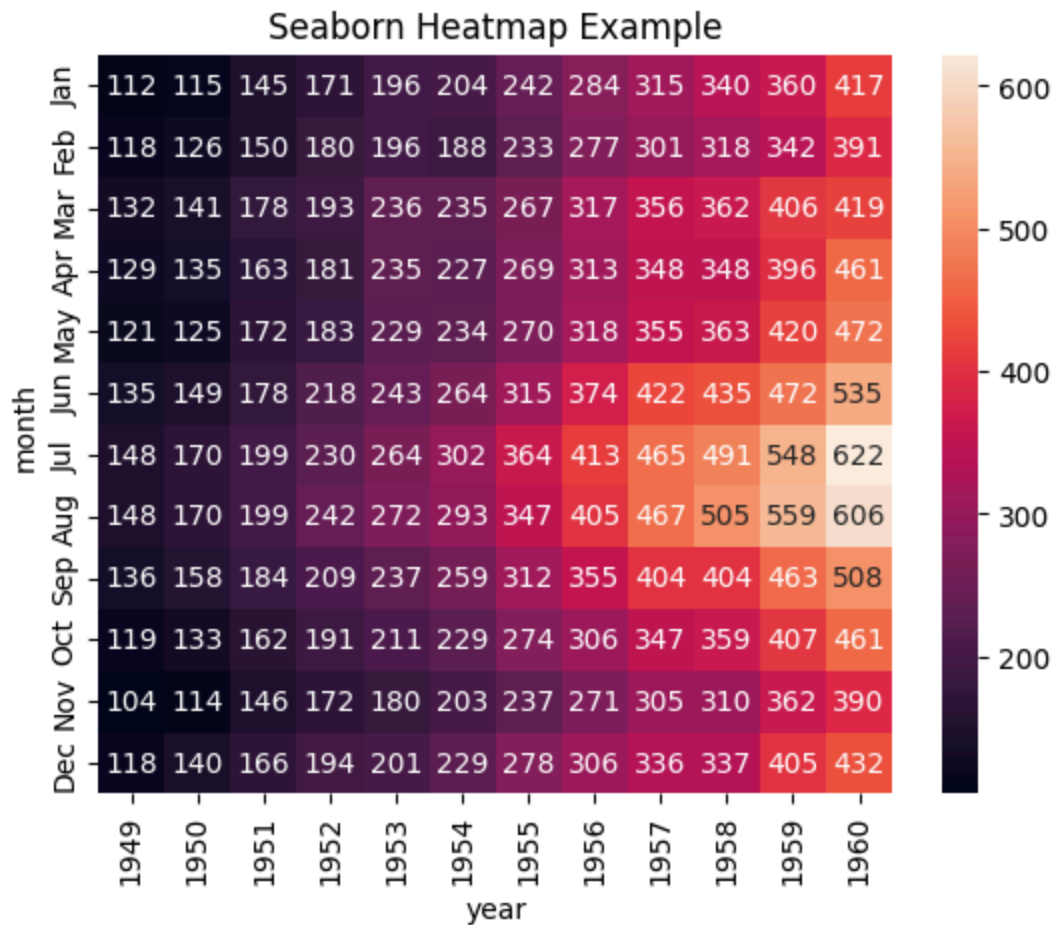
Seaborn Heatmap Example

	1949	1950	1951	1952	1953	1954	1955	1956	1957	1958	1959	1960
Jan	112	115	145	171	196	204	242	284	315	340	360	417
Feb	118	126	150	180	196	188	233	277	301	318	342	391
Mar	132	141	178	193	236	235	267	317	356	362	406	419
Apr	129	135	163	181	235	227	269	313	348	348	396	461
May	121	125	172	183	229	234	270	318	355	363	420	472
Jun	135	149	178	218	243	264	315	374	422	435	472	535
Jul	148	170	199	230	264	302	364	413	465	491	548	622
Aug	148	170	199	242	272	293	347	405	467	505	559	606
Sep	136	158	184	209	237	259	312	355	404	404	463	508
Oct	119	133	162	191	211	229	274	306	347	359	407	461
Nov	104	114	146	172	180	203	237	271	305	310	362	390
Dec	118	140	166	194	201	229	278	306	336	337	405	432

month / year

In [13]:

```python
import seaborn as sns
import pandas as pd

data = pd.DataFrame({
    "category": ['A']*10 + ['B']*10,
    "value": [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
})

sns.boxplot(data=data, x="category", y="value")
plt.title("Box Plot")
plt.show()
```
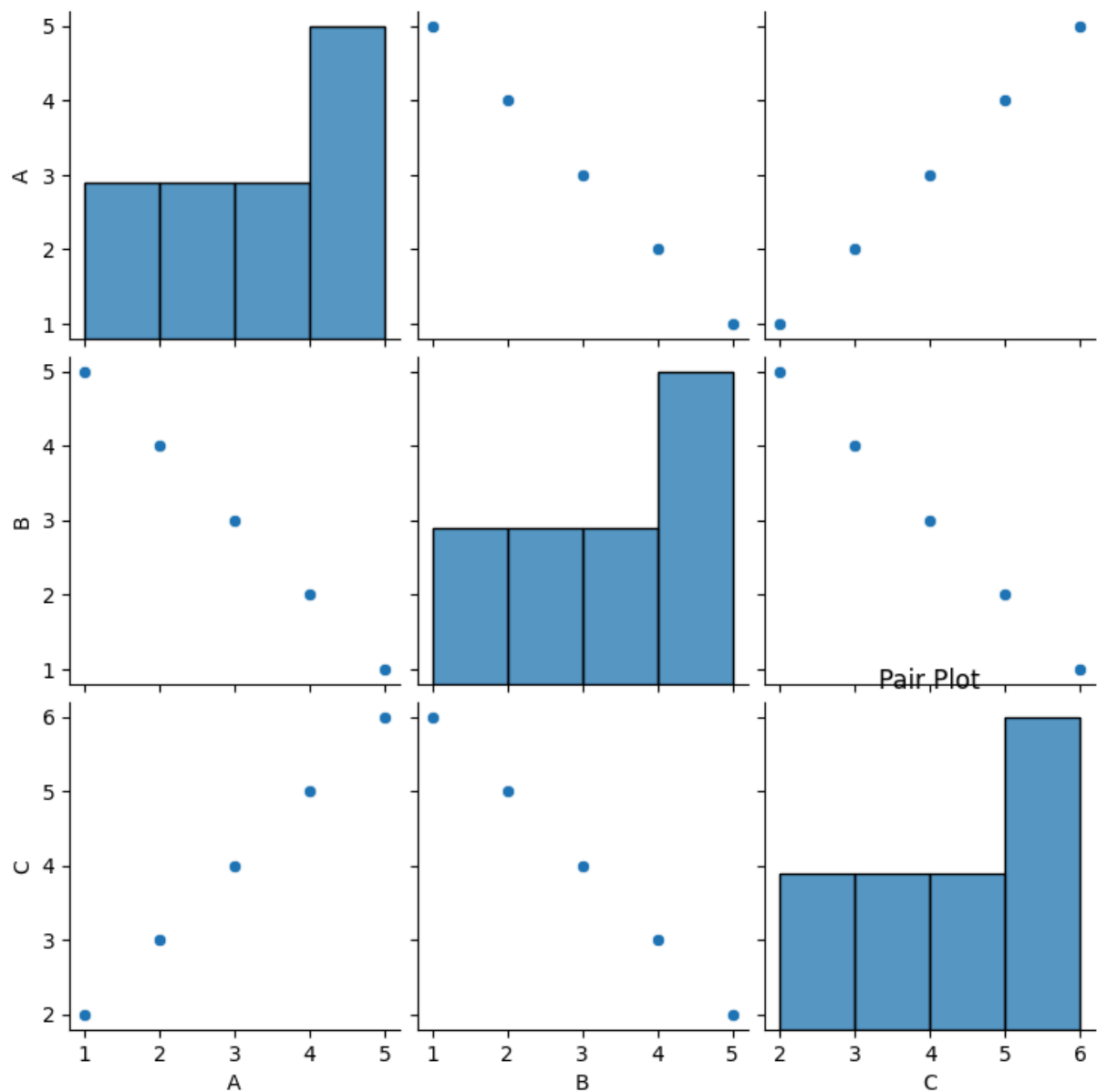
Box Plot
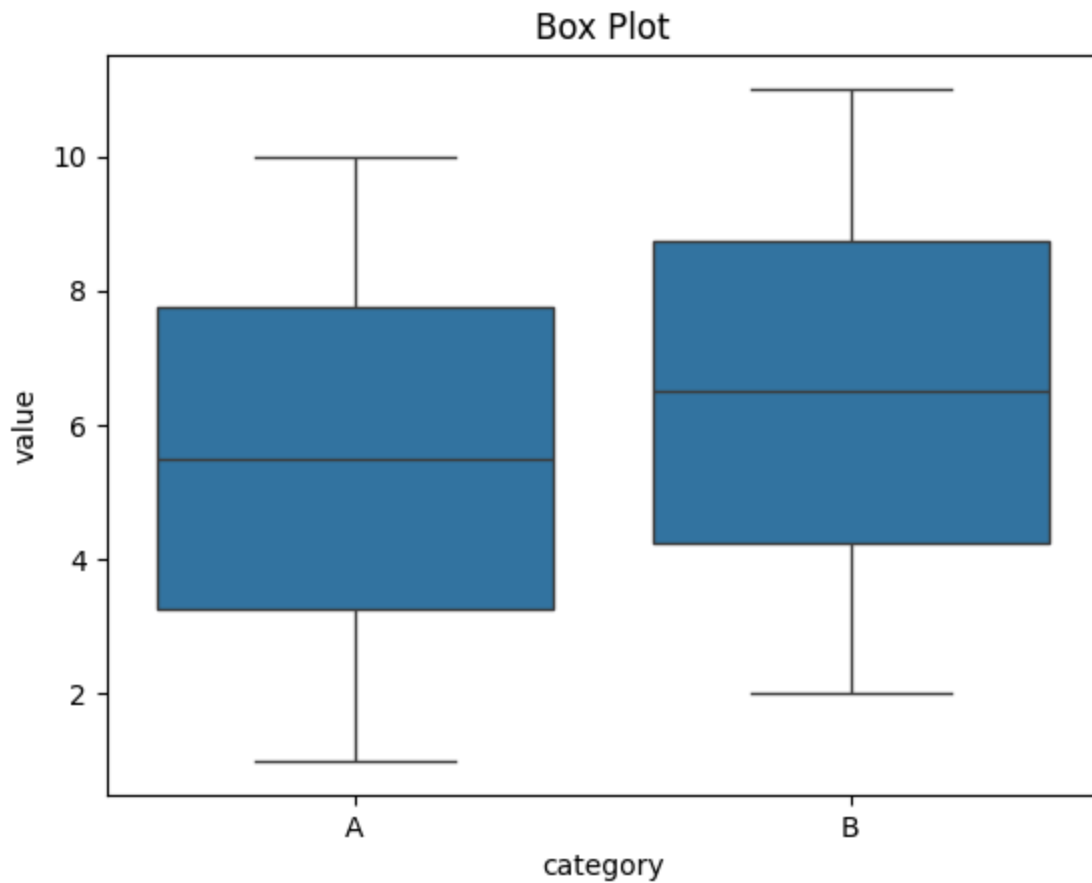
```python
import seaborn as sns
import pandas as pd

data = pd.DataFrame({
    "A": [1, 2, 3, 4, 5],
    "B": [5, 4, 3, 2, 1],
    "C": [2, 3, 4, 5, 6]
})

sns.pairplot(data)
plt.title("Pair Plot")
plt.show()
```
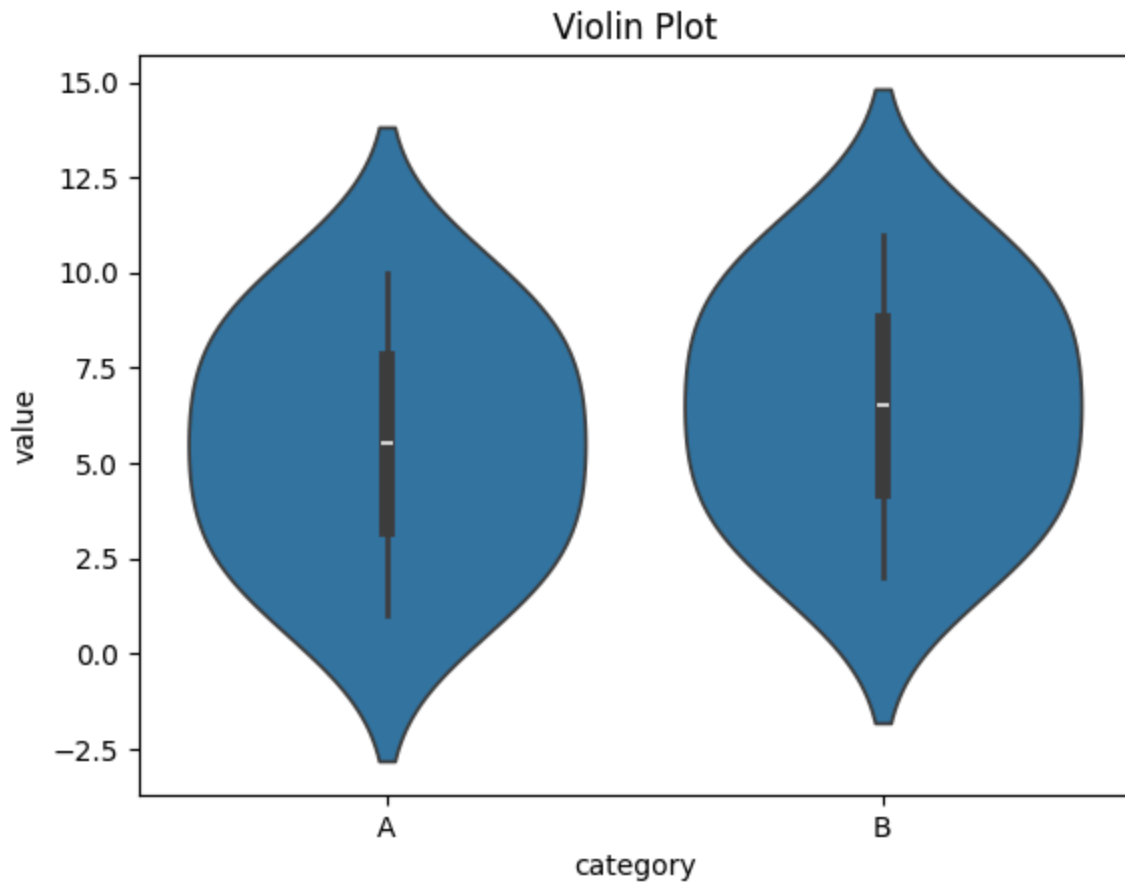
Pair Plot

```
In [15]:  import seaborn as sns
          import pandas as pd

          data = pd.DataFrame({
              "category": ['A']*10 + ['B']*10,
              "value": [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
          })

          sns.violinplot(data=data, x="category", y="value")
          plt.title("Violin Plot")
          plt.show()
```

Violin Plot

```python
import seaborn as sns
import pandas as pd

data = pd.DataFrame({
    "category": ['A']*10 + ['B']*10,
    "value": [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
})

sns.swarmplot(data=data, x="category", y="value")
plt.title("Swarm Plot")
plt.show()
```

Swarm Plot
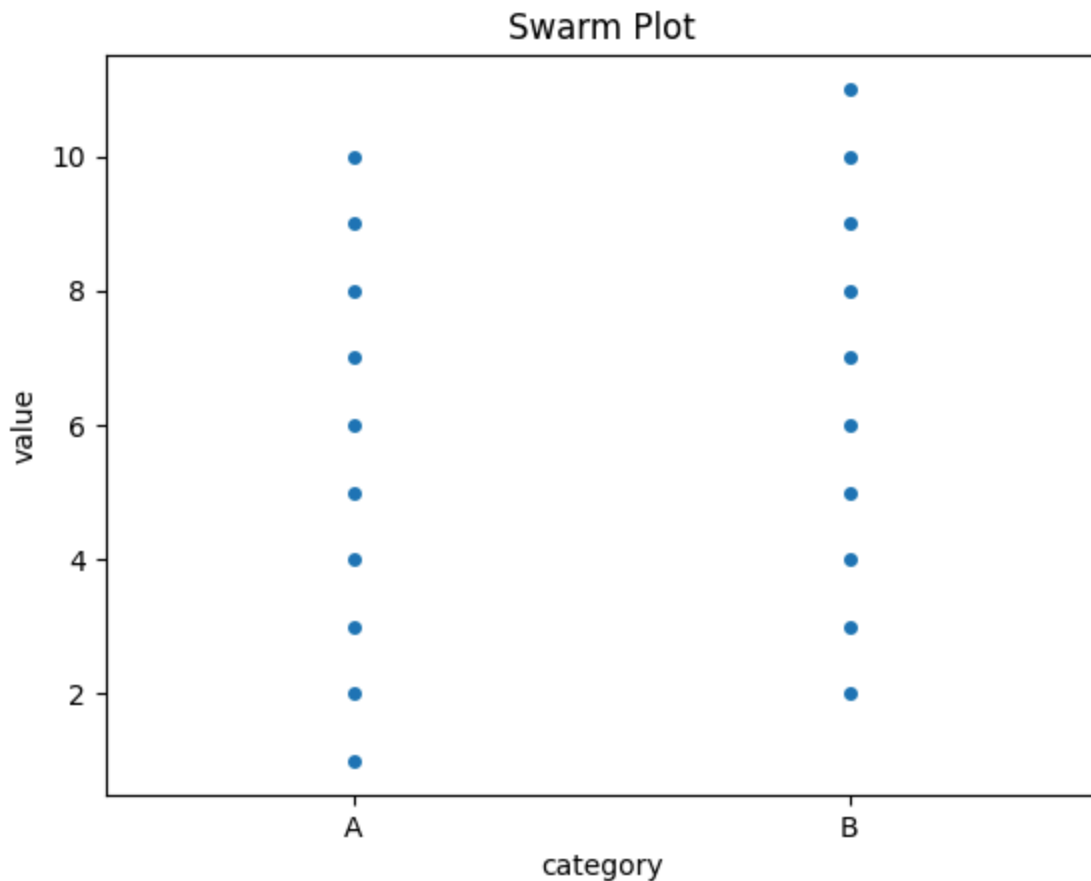
🧑‍💻 Strengths and Weaknesses of Matplotlib and Seaborn

🎨 Matplotlib

Strengths

🔑 Strength	📊 Details
Flexibility and Control	🔧 Matplotlib offers complete control over every aspect of a plot, allowing highly customized visualizations.
Versatility	🔄 Supports a wide variety of plots, including line, scatter, bar, histograms, 3D plots, and more. Also supports animations and interactive plots.
Integration	🔗 Works seamlessly with libraries like NumPy, Pandas, SciPy, and GUI toolkits such as Tkinter, wxPython, and Qt.
Large Community & Documentation	📚 Extensive resources, tutorials, and support from a large user community.

Weaknesses

❌ Weakness	💬 Details
Complexity	🧠 Requires more code for simple plots and can become verbose due to its extensive customization options.
Default Styles	💻 The default aesthetic of plots can appear outdated and less attractive compared to more modern libraries like Seaborn.
Interactivity	🖱️ Basic interactivity is supported, but it lacks the advanced interactive capabilities of libraries like Plotly or Bokeh.
Performance with Large Datasets	📉 Can be slow with very large datasets, particularly for interactive visualizations.

🌈 Seaborn

Strengths

🔑 Strength	📊 Details
Ease of Use	✨ Simplifies the creation of complex statistical plots with minimal code, ideal for quick data visualization and exploration.
Aesthetics	🌟 Beautiful default styles and color palettes enhance visual appeal without needing customization.
Integration with Pandas	🐼 Works seamlessly with Pandas DataFrames, making it easy to create visualizations directly from structured data.
Statistical Visualization	📈 Specialized in statistical plots like violin plots, box plots, and heatmaps.

Weaknesses

❌ Weakness	💬 Details
Limited Customization	⚙️ Provides fewer customization options compared to Matplotlib, limiting detailed adjustments for complex visualizations.
Dependency on Matplotlib	🔗 Built on top of Matplotlib, so advanced customization requires understanding Matplotlib's features.
Performance with Large Datasets	📈 Struggles with very large datasets due to high-level statistical computations that can be resource-intensive.
Interactivity	🖱️ Primarily focused on static plots, requiring additional tools for interactivity.

⚖️ Comparison

🏷️ Aspect	🎨 Matplotlib	🌈 Seaborn
Ease of Use	👨‍💻 Highly customizable but requires more code for basic plots.	👨‍💻 Simplifies the process of creating beautiful, informative plots with minimal code.
Customization Options	⚙️ Extensive customization options for detailed visual adjustments.	🎨 Provides fewer customization options, but with beautiful defaults.
Interactivity	🖱️ Basic interactivity; needs additional libraries like Plotly or Bokeh for advanced features.	🔒 Primarily static plots, but can integrate with Matplotlib's interactive capabilities.
Performance with Large Datasets	📊 Handles large datasets well but can slow down with very large data.	📈 Struggles with very large datasets due to its high-level nature and statistical features.

📝 Summary

- **Matplotlib** is ideal for users who need **full control** over their plots and want to create **publication-quality** figures. It is best suited for complex, highly customized visualizations, and integration with other Python libraries.

- **Seaborn** is best for users who need **quick, beautiful visualizations** and **statistical analysis** with **minimal code**. It excels in **exploratory data analysis (EDA)** and works seamlessly with Pandas.