Matplotlib and Seaborn Visualization Guide

MATPLOTLIB

Introduction to Matplotlib

Matplotlib is the foundational plotting library for Python, renowned for its flexibility and extensive support for 2D visualizations.

Unique Features:

- Highly customizable plots.
- "pyplot" interface emulates MATLAB-like plotting.
- Wide range of supported plot types.

Typical Use Cases:

- Data exploration
- Publication-quality figures
- Serving as a base for other libraries (e.g., Seaborn)

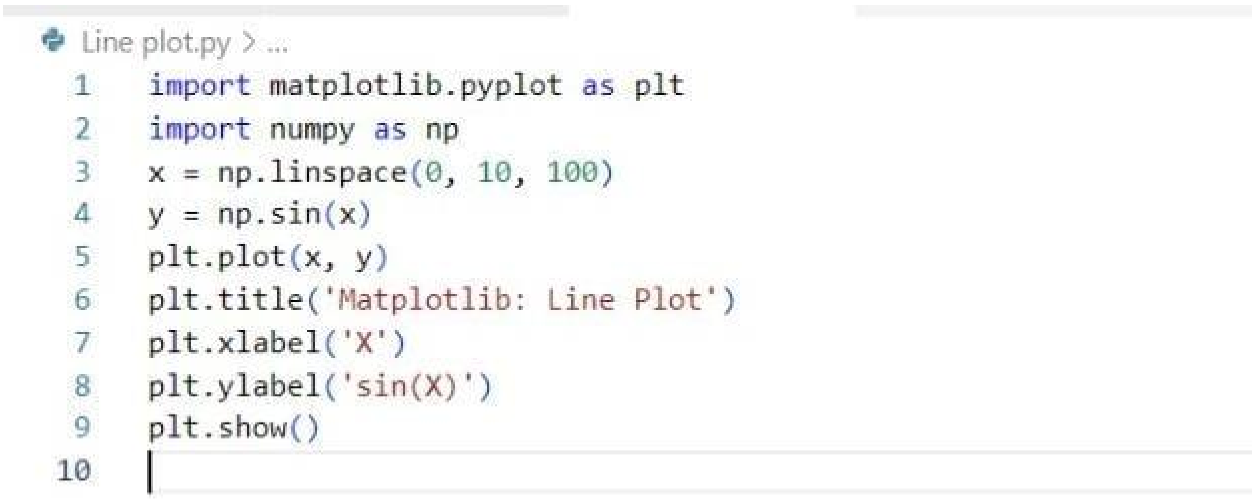
Matplotlib Examples

Below are some common types of graphs generated using Matplotlib, based on the example code provided.

- Line Plot
- Scatter Plot
- Bar Chart
- Histogram
- Pie Chart
- Box Plot

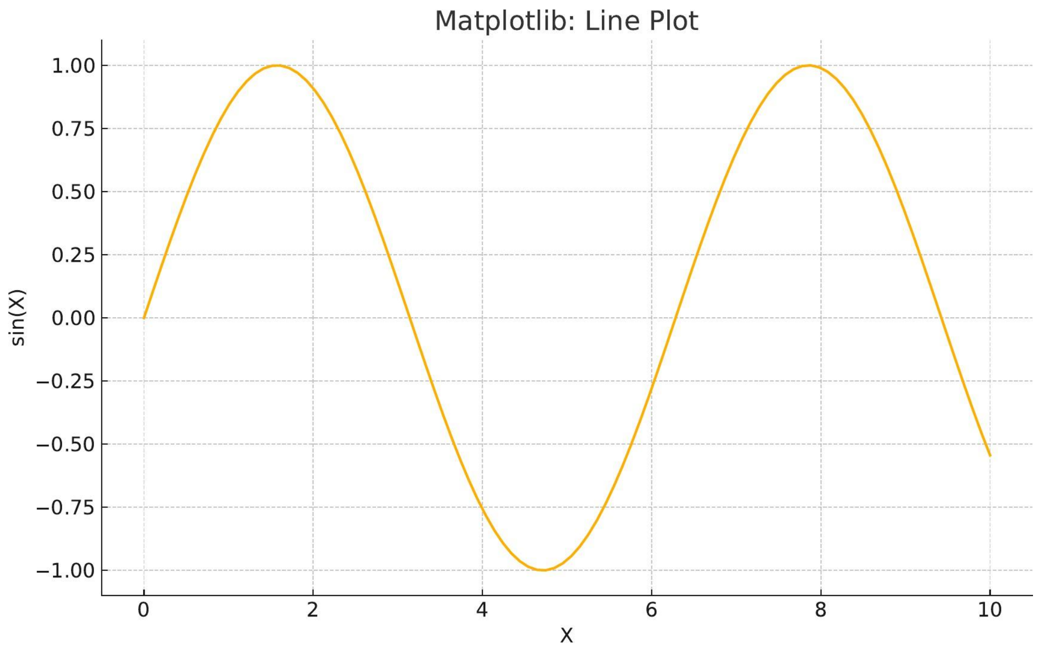
Line Plot

Trend over continous variables(eg:time series)


Line plot.py > ...

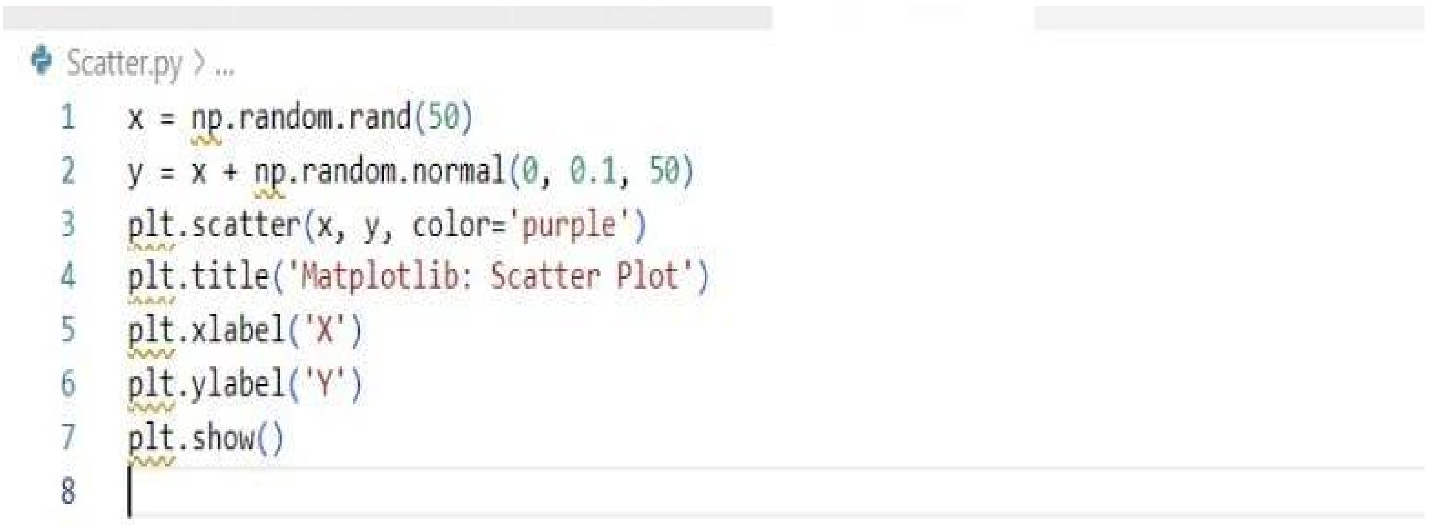
```python
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(0, 10, 100)
y = np.sin(x)
plt.plot(x, y)
plt.title('Matplotlib: Line Plot')
plt.xlabel('X')
plt.ylabel('sin(X)')
plt.show()

```

OUTPUT:



Scatter Plot

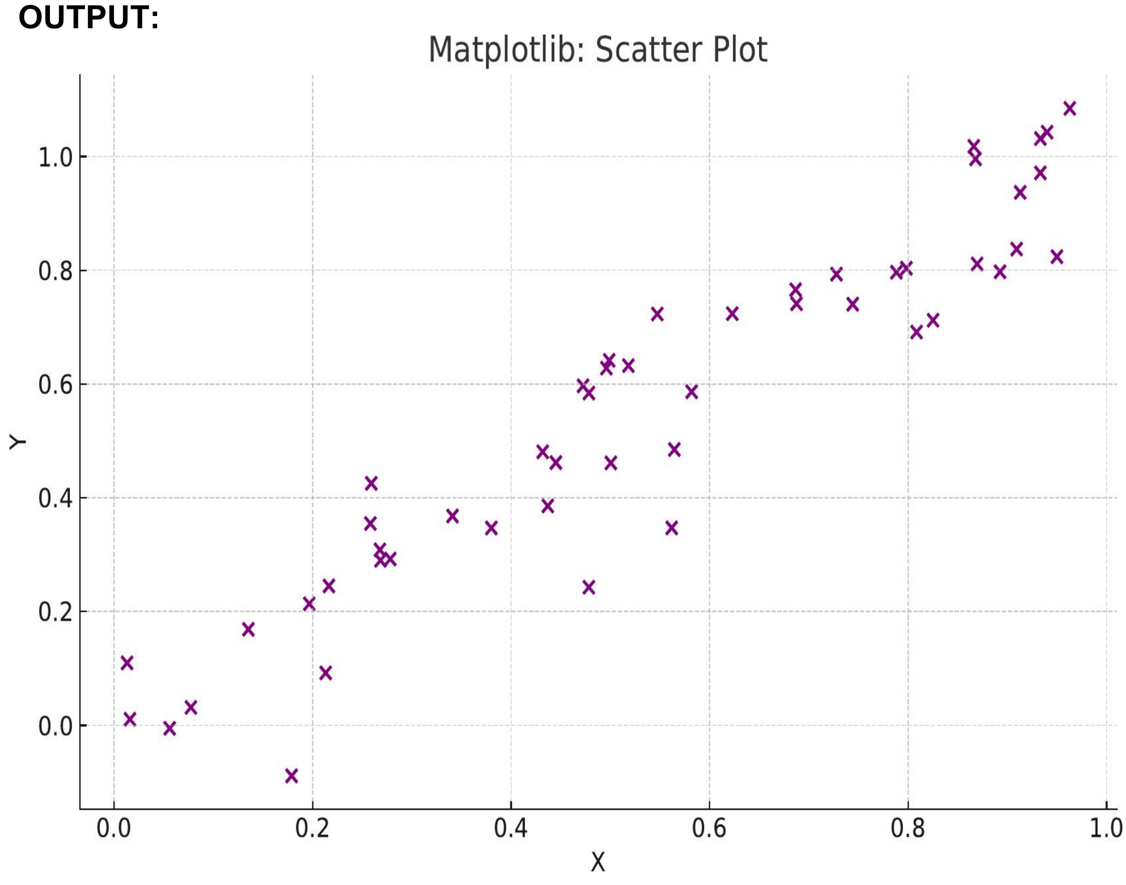
Relationship between two variables



Scatter.py > ...

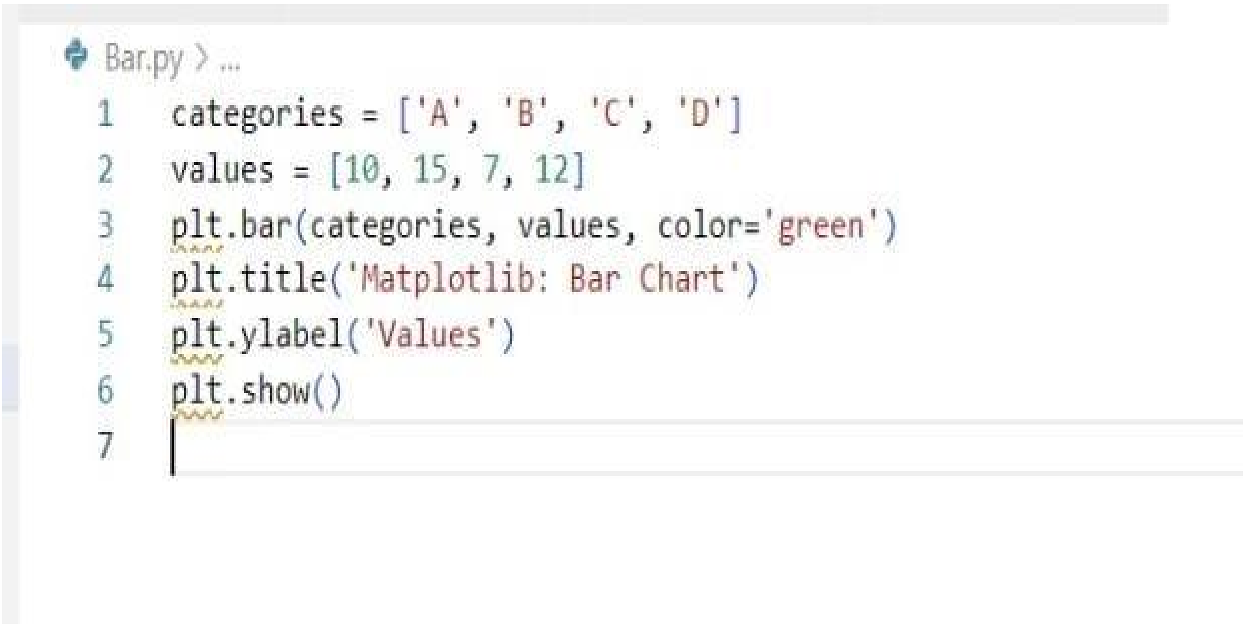
```python
x = np.random.rand(50)
y = x + np.random.normal(0, 0.1, 50)
plt.scatter(x, y, color='purple')
plt.title('Matplotlib: Scatter Plot')
plt.xlabel('X')
plt.ylabel('Y')
plt.show()

```

OUTPUT:



Bar Chart

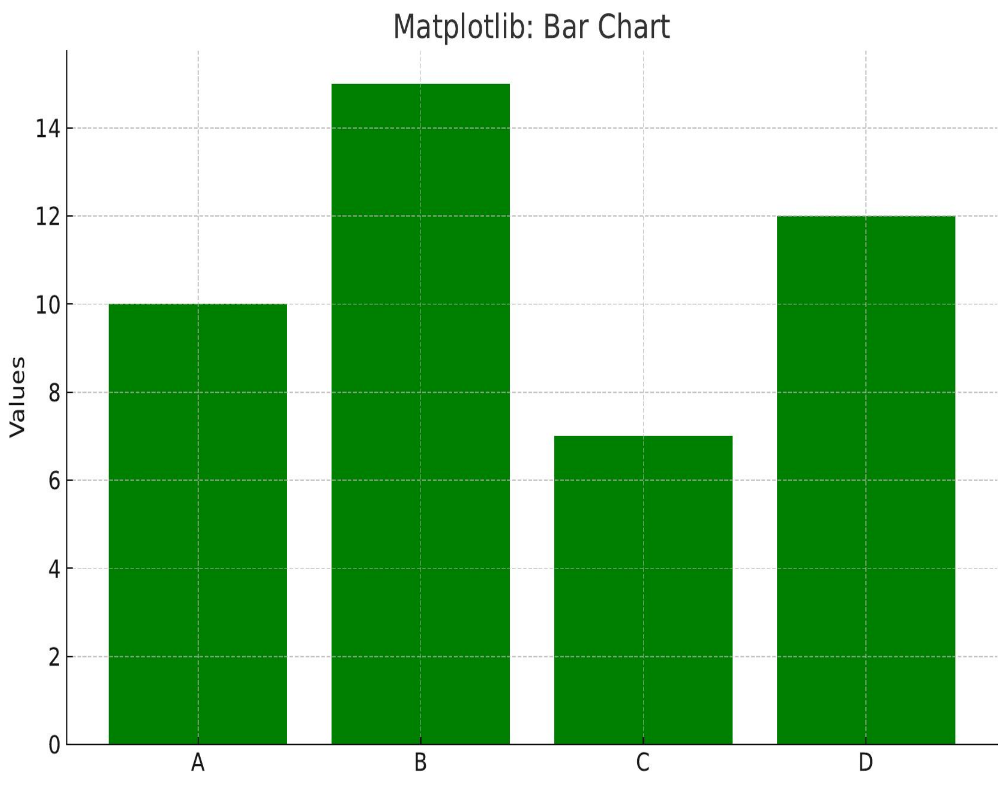
Comparing Categories



Bar.py > ...

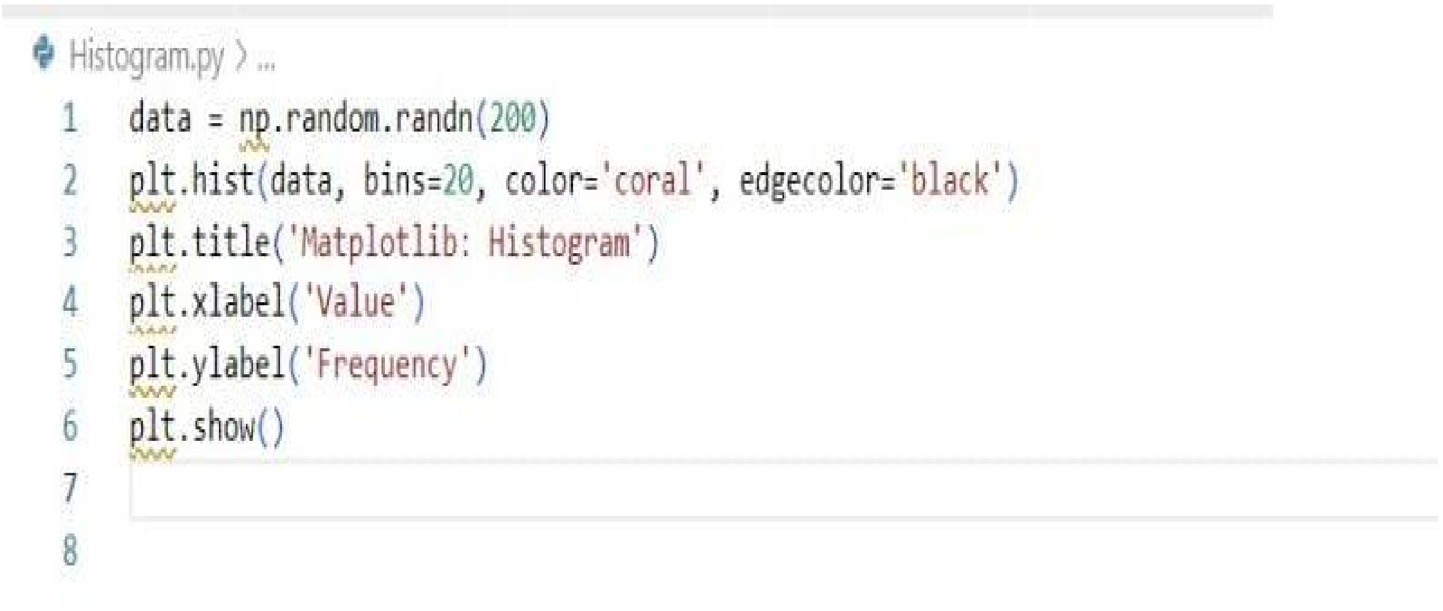
```python
categories = ['A', 'B', 'C', 'D']
values = [10, 15, 7, 12]
plt.bar(categories, values, color='green')
plt.title('Matplotlib: Bar Chart')
plt.ylabel('Values')
plt.show()
```

OUTPUT:



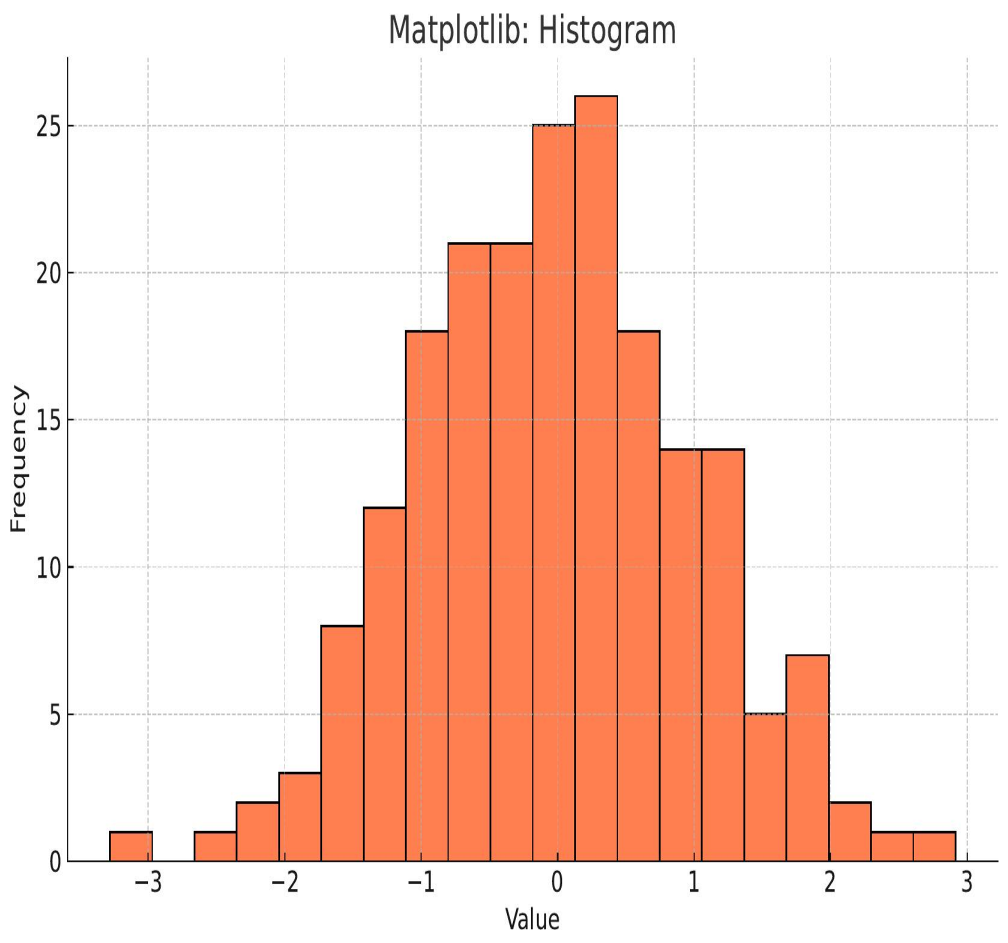
Histogram

Distribution of numerical data



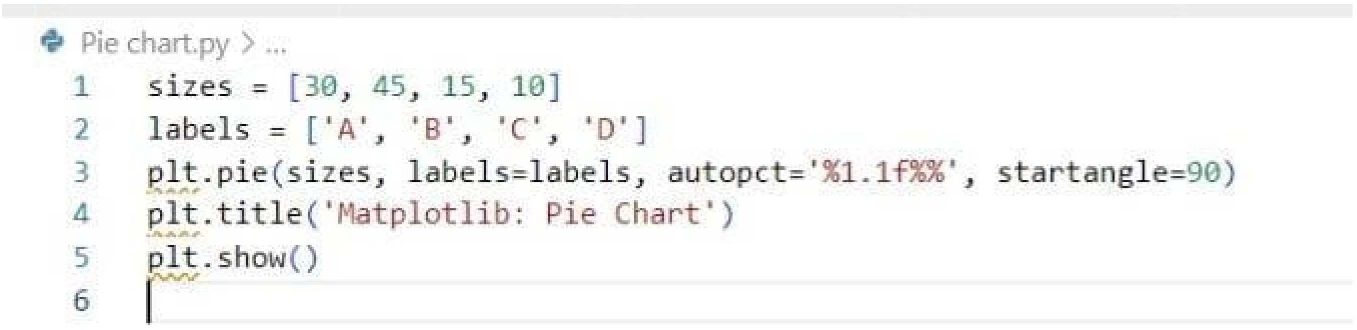
Histogram.py > ...

```python
data = np.random.randn(200)
plt.hist(data, bins=20, color='coral', edgecolor='black')
plt.title('Matplotlib: Histogram')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.show()
```



Pie Chart

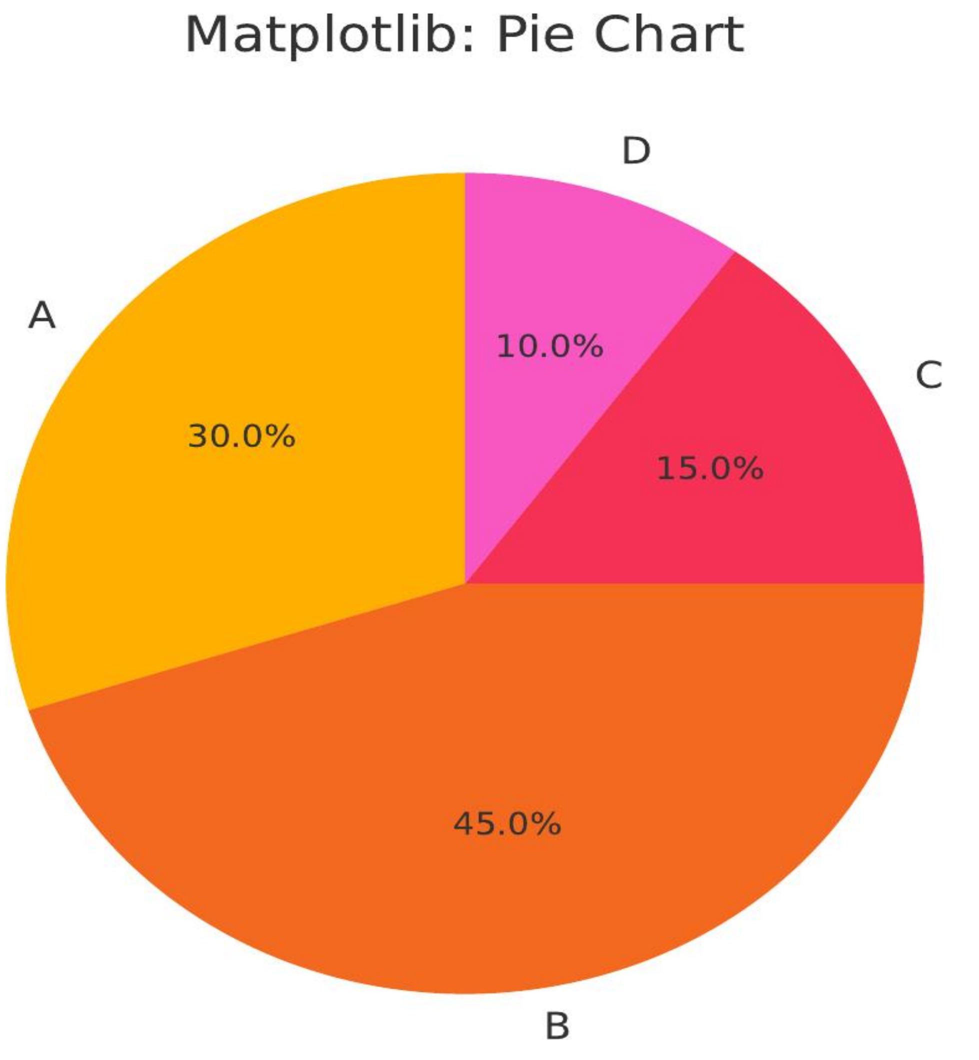
Proportions within a while



Pie chart.py > ...

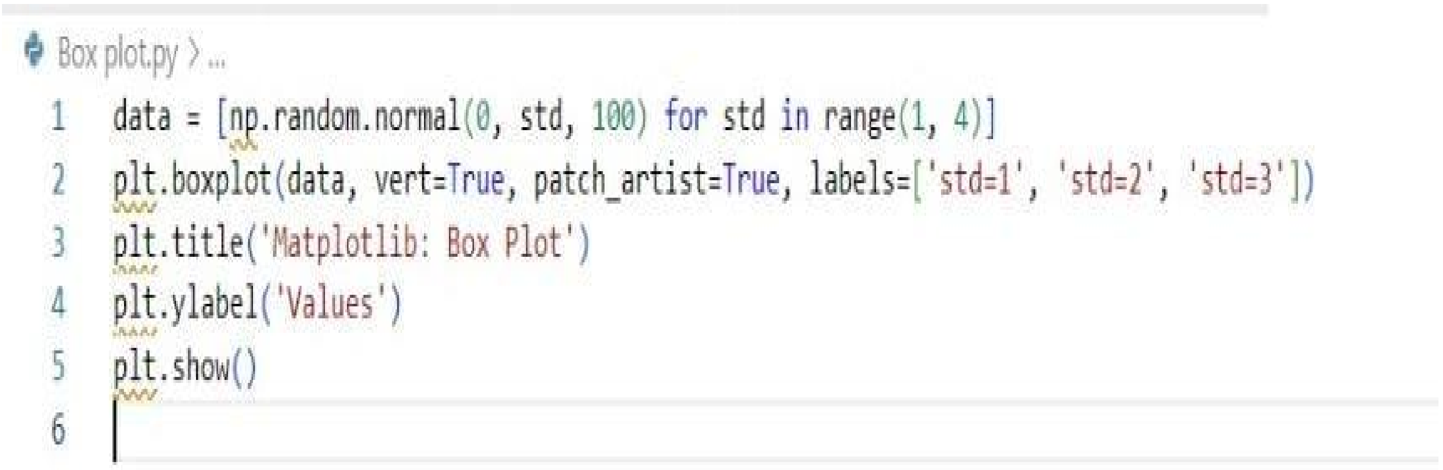
```python
sizes = [30, 45, 15, 10]
labels = ['A', 'B', 'C', 'D']
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)
plt.title('Matplotlib: Pie Chart')
plt.show()

```

OUTPUT:



BOX PLOT

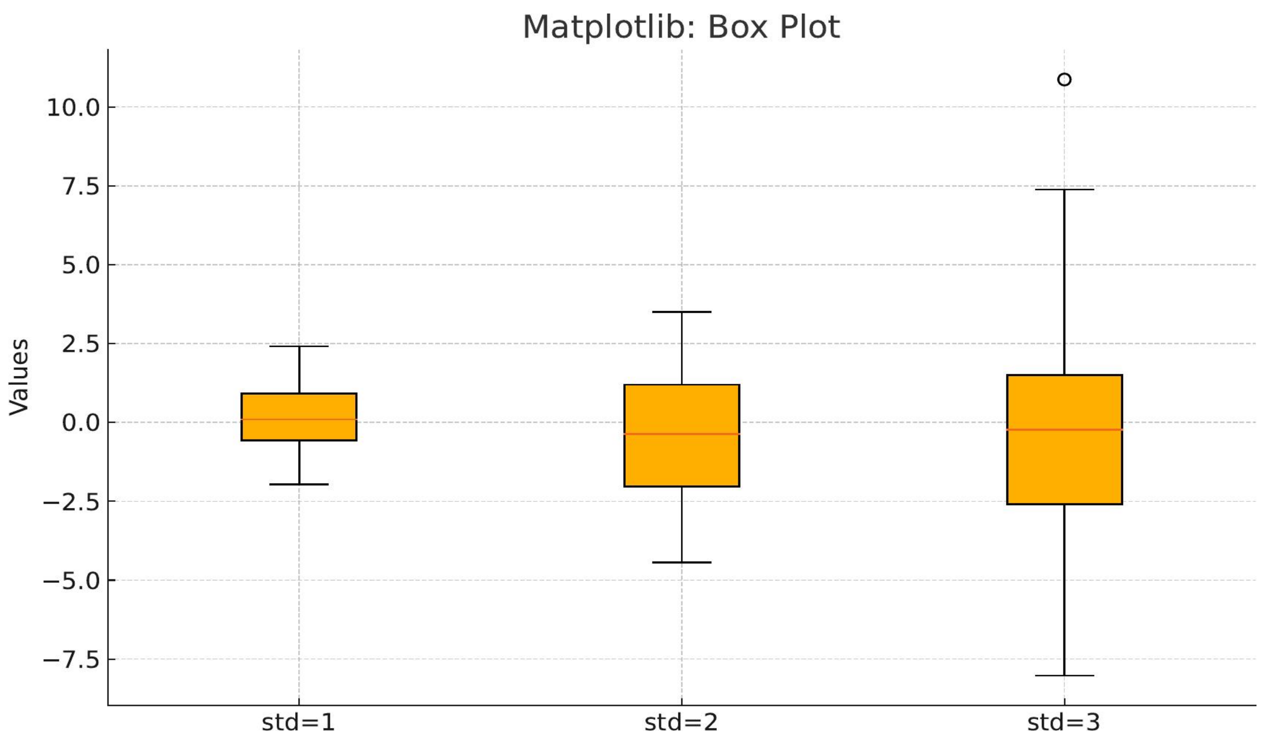
Showing distribution and outliers



Box plot.py > ...

```python
data = [np.random.normal(0, std, 100) for std in range(1, 4)]
plt.boxplot(data, vert=True, patch_artist=True, labels=['std=1', 'std=2', 'std=3'])
plt.title('Matplotlib: Box Plot')
plt.ylabel('Values')
plt.show()

```

OUTPUT:



SEABORN

Introduction to Seaborn

Seaborn is built on top of Matplotlib and is designed for statistical data visualization with a focus on attractive defaults and simplification.

Unique Features:
- Simplified syntax for creating complex statistical plots.
- Built-in themes, color palettes, and integration with pandas DataFrames.
- Automatically manages plot aesthetics for clarity.

Typical Use Cases:
- Statistical data analysis
- Exploring and visualizing relationships and distributions

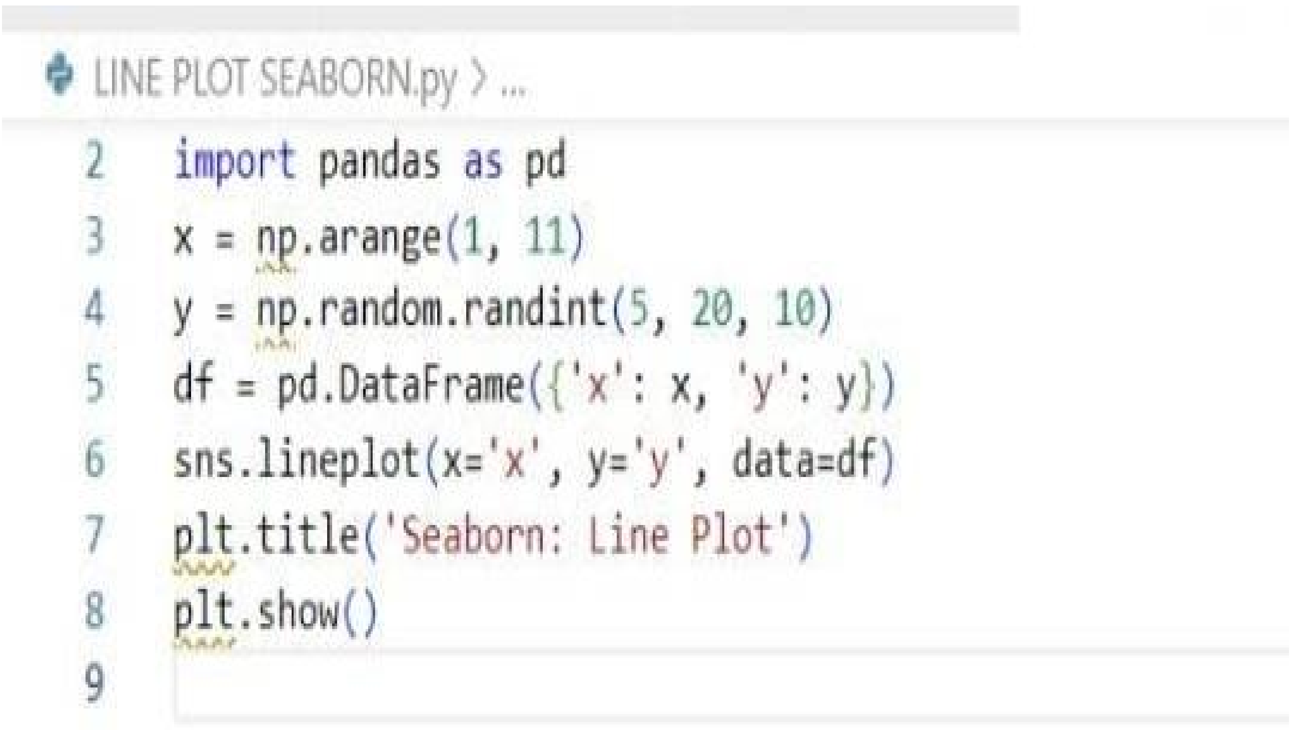
Seaborn Examples

Below are various statistical plots created using Seaborn.

- Line Plot
- Grouped Scatter Plot
- Bar Plot
- Histogram with KDE
- Box Plot
- Vilion Plot

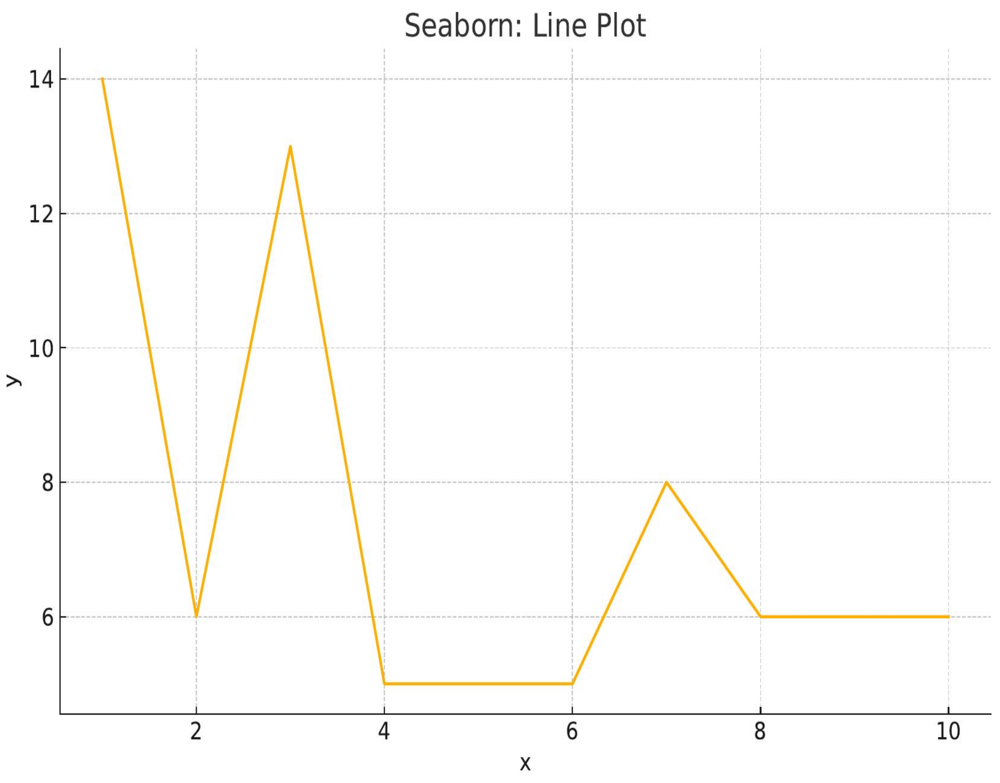
LINE PLOT

Time series with confidence interval



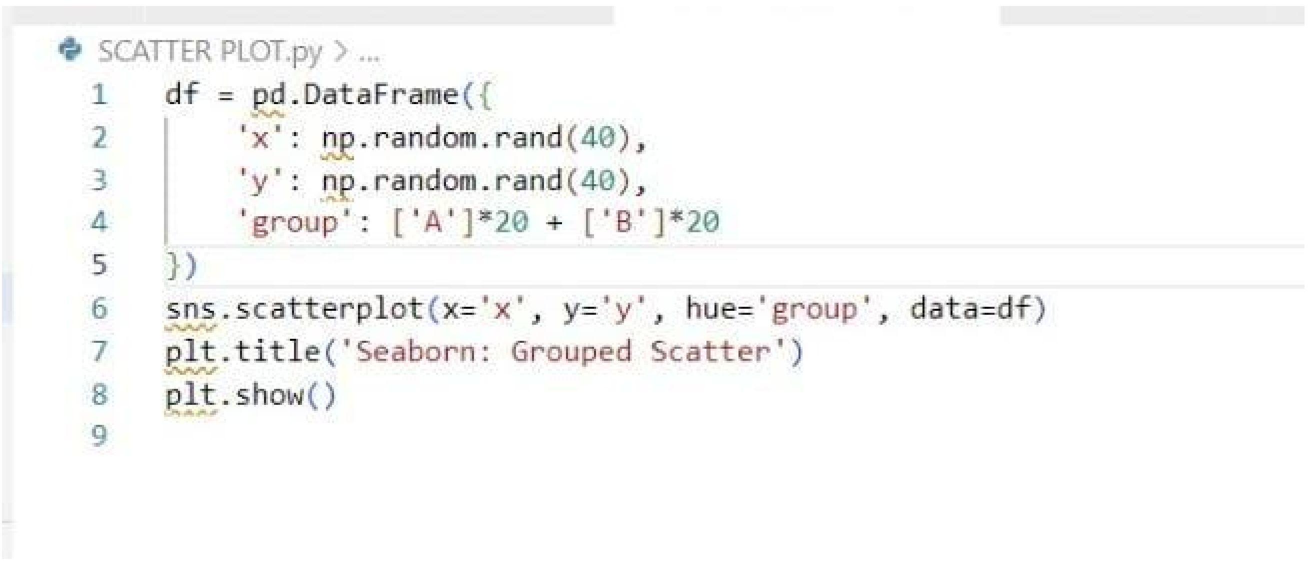
```python
import pandas as pd
x = np.arange(1, 11)
y = np.random.randint(5, 20, 10)
df = pd.DataFrame({'x': x, 'y': y})
sns.lineplot(x='x', y='y', data=df)
plt.title('Seaborn: Line Plot')
plt.show()

```

OUTPUT:



GROUPED SCATTER

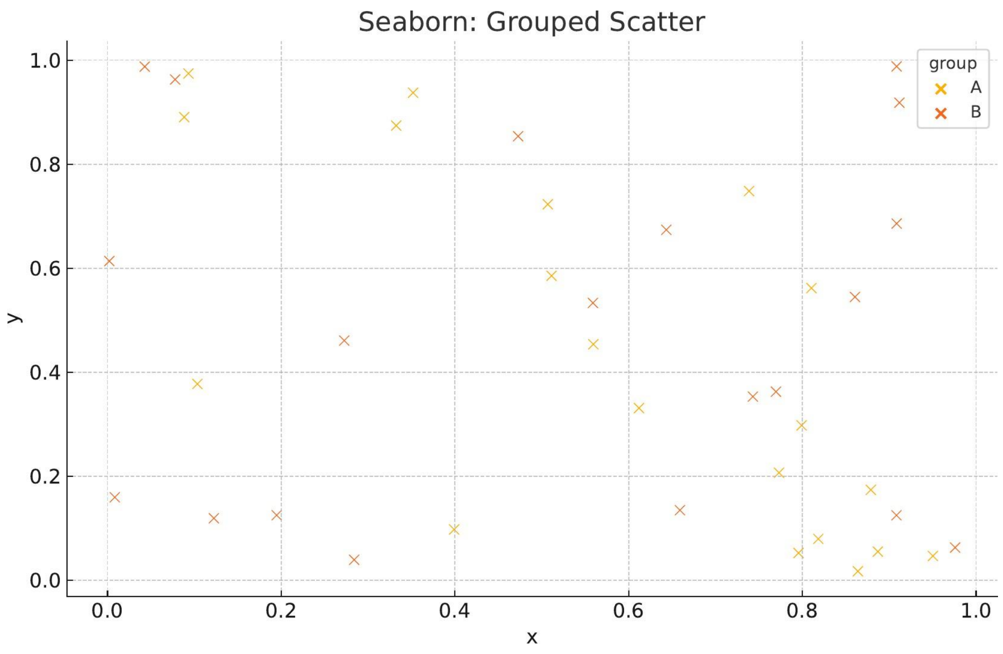
Relationship, with hue for categories



SCATTER PLOT.py > ...

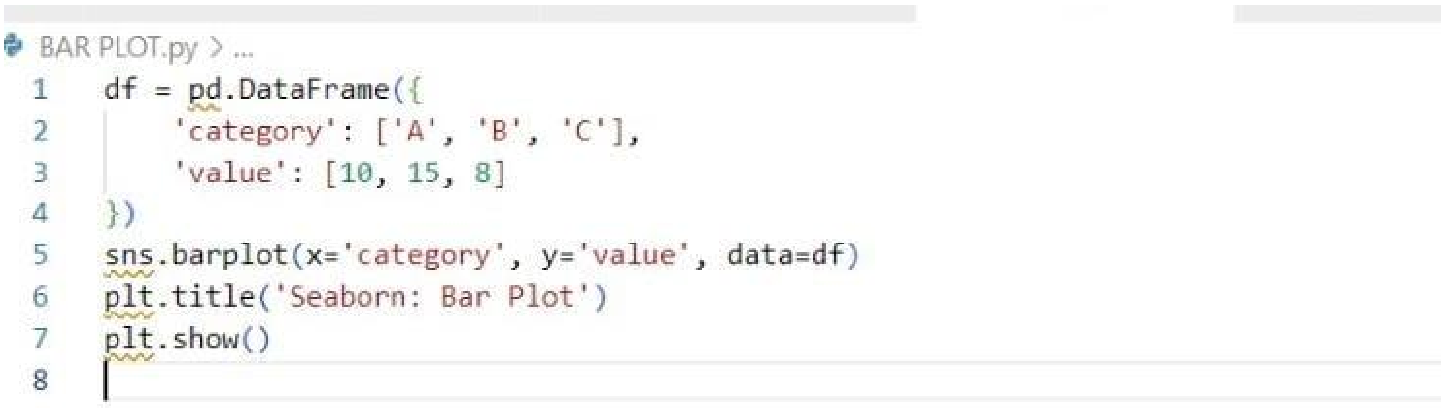
```python
df = pd.DataFrame({
    'x': np.random.rand(40),
    'y': np.random.rand(40),
    'group': ['A']*20 + ['B']*20
})
sns.scatterplot(x='x', y='y', hue='group', data=df)
plt.title('Seaborn: Grouped Scatter')
plt.show()
```

OUTPUT:



BAR PLOT

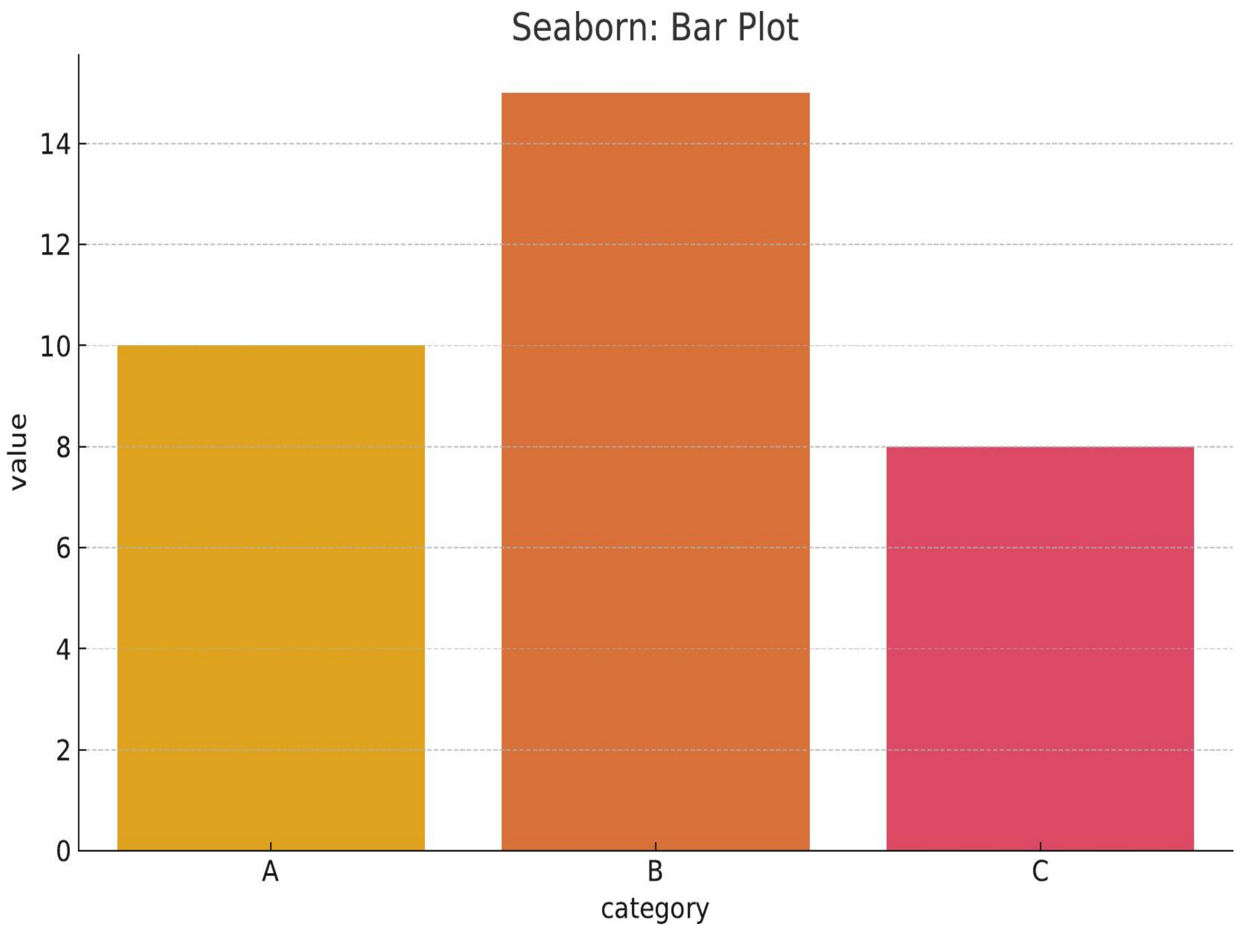
Category-wise mean/CI visualization



BAR PLOT.py > ...

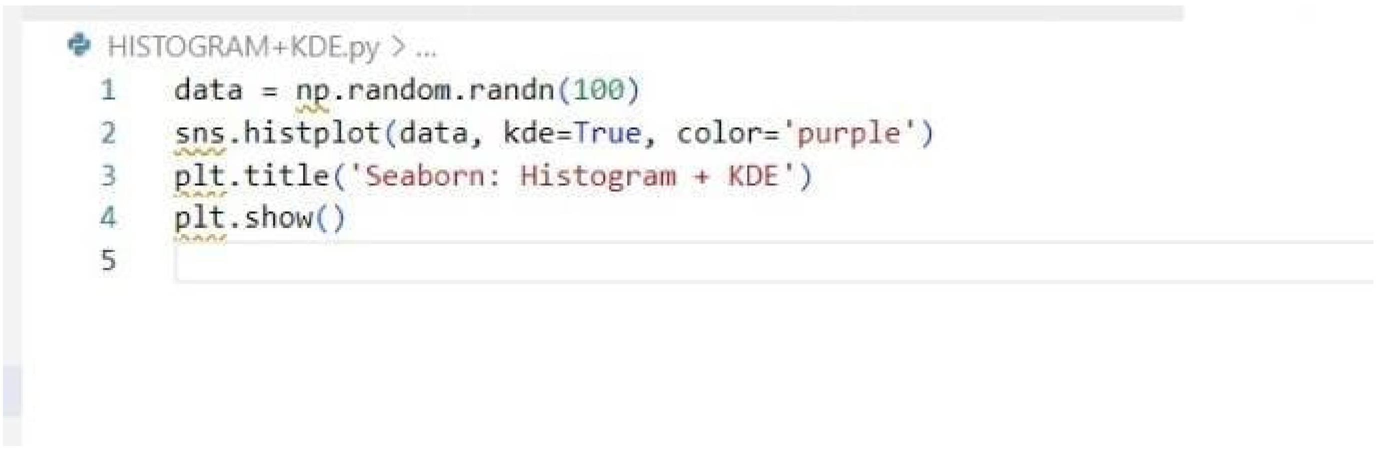
```python
df = pd.DataFrame({
    'category': ['A', 'B', 'C'],
    'value': [10, 15, 8]
})
sns.barplot(x='category', y='value', data=df)
plt.title('Seaborn: Bar Plot')
plt.show()

```

OUTPUT:



HISTOGRAM+KDE

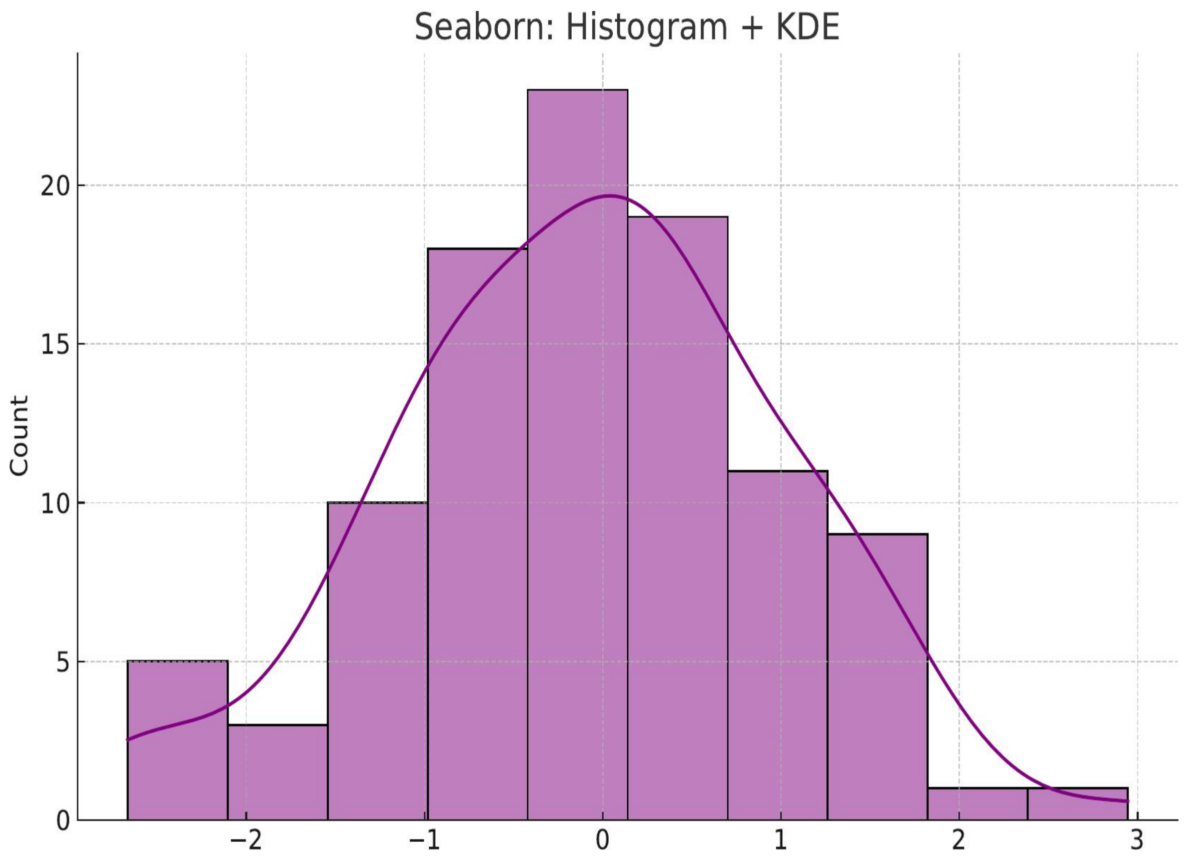
Distribution with density estimation



HISTOGRAM+KDE.py > ...

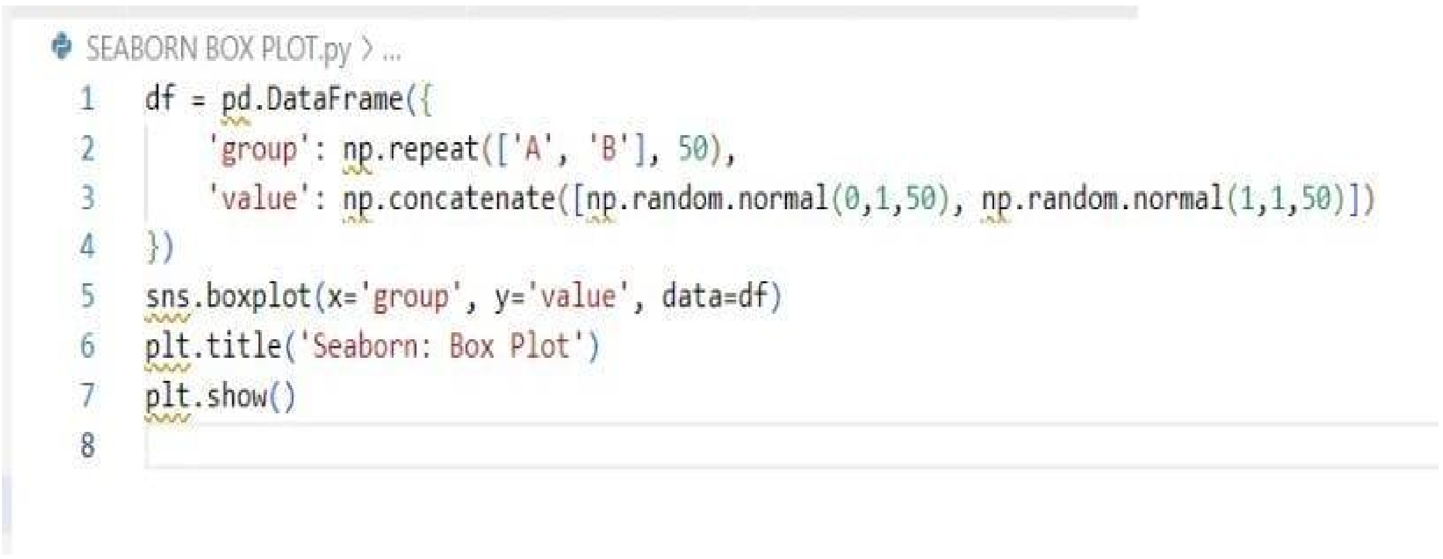
```python
data = np.random.randn(100)
sns.histplot(data, kde=True, color='purple')
plt.title('Seaborn: Histogram + KDE')
plt.show()

```

OUTPUT:



BOX PLOT

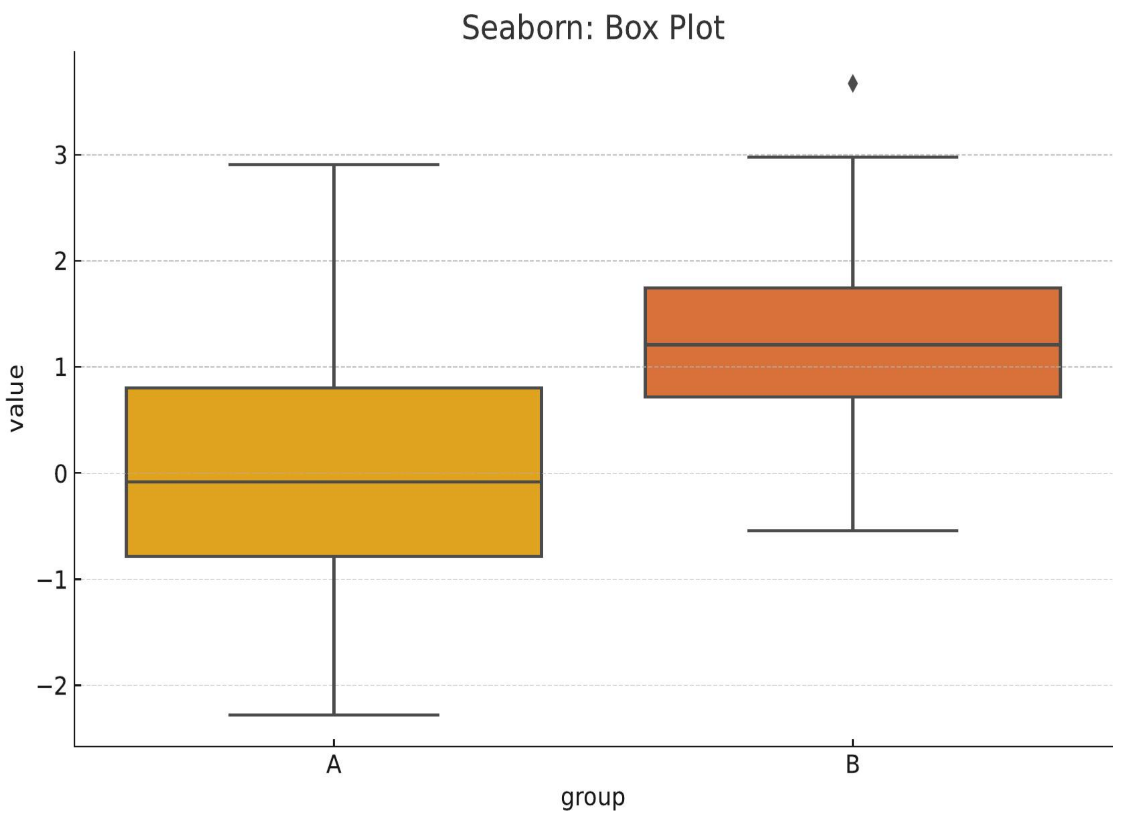
Distribution with groups,outlier detection



SEABORN BOX PLOT.py > ...

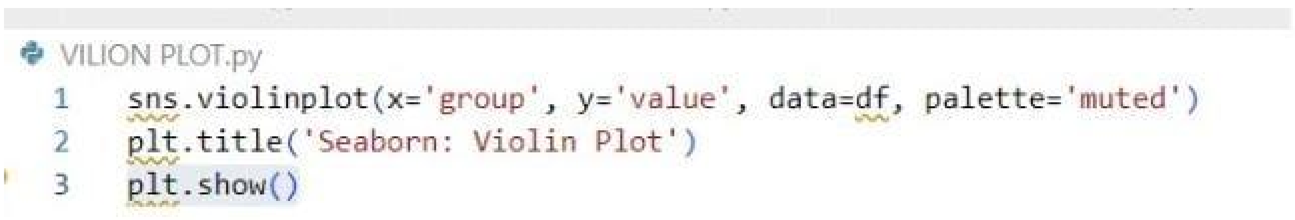
```python
df = pd.DataFrame({
    'group': np.repeat(['A', 'B'], 50),
    'value': np.concatenate([np.random.normal(0,1,50), np.random.normal(1,1,50)])
})
sns.boxplot(x='group', y='value', data=df)
plt.title('Seaborn: Box Plot')
plt.show()
```

OUTPUT:



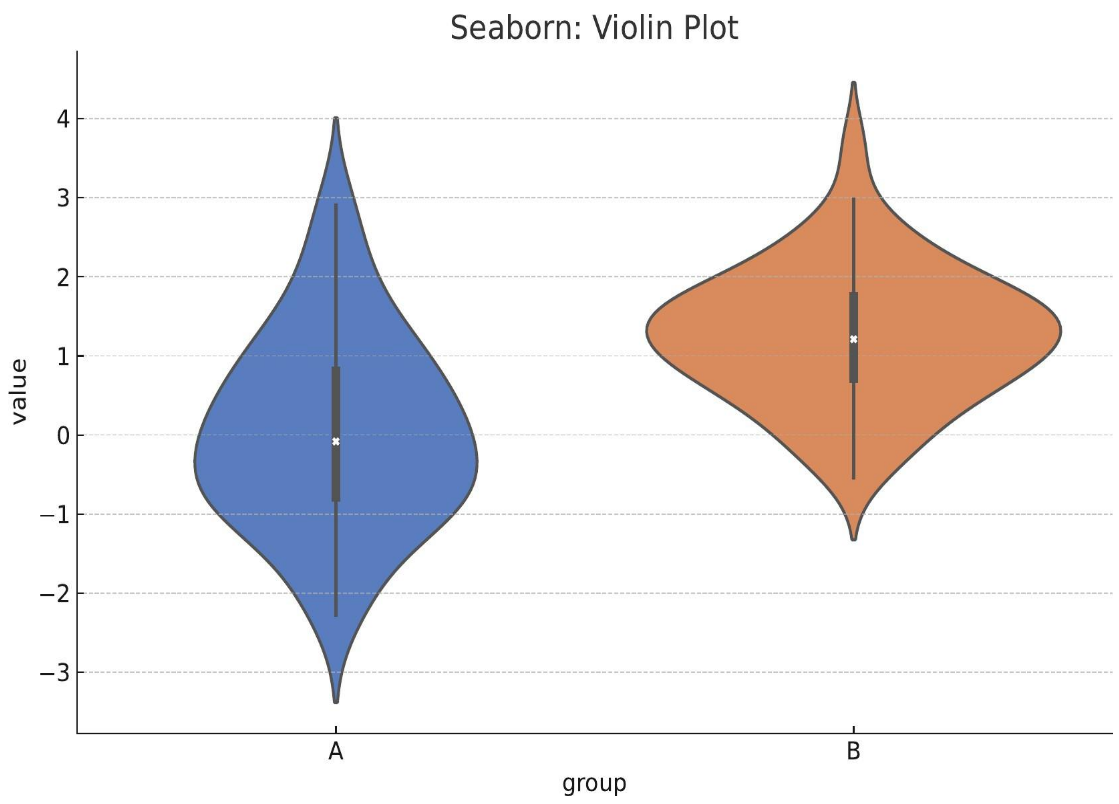
VIOLIN PLOT

Distribution shape and summary with groups


VILION PLOT.py

```python
sns.violinplot(x='group', y='value', data=df, palette='muted')
plt.title('Seaborn: Violin Plot')
plt.show()
```

OUTPUT:



Matplotlib vs Seaborn: Comparison

Feature Comparison:

Ease of Use:

- Matplotlib: Steeper learning curve, very flexible
- Seaborn: Simpler syntax for statistical plots

Customization:

- Matplotlib: Highly configurable (fine-tuned control)
- Seaborn: Limited, but can be adjusted via Matplotlib

Graph Variety:

- Matplotlib: Supports most 2D plots; 3D via mpl_toolkits
- Seaborn: Focuses on statistical data visualization

Interactivity:

- Matplotlib: Basic (plugins/extensions available)
- Seaborn: Inherits Matplotlib's interactivity

Dataset Handling:

- Matplotlib: Manual handling, works with numpy, pandas
- Seaborn: Natively works with DataFrames; easy grouping

Performance:

- Matplotlib: Handles large datasets well
- Seaborn: Slightly less performant for very large data