DATA VISUALIZATION WITH MATPLOTLIB AND SEABORN

In data science, data visualization is essential for understanding patterns, trends, and insights. Two popular Python libraries for data visualization are Matplotlib and Seaborn . This documentation provides an overview of both libraries, explores their capabilities with examples, and compares them side-by-side.

MATPLOTLIB OVERVIEW

Matplotlib is a foundational library for creating static, animated, and interactive visualizations in Python. It provides full control over plots and is highly customizable.

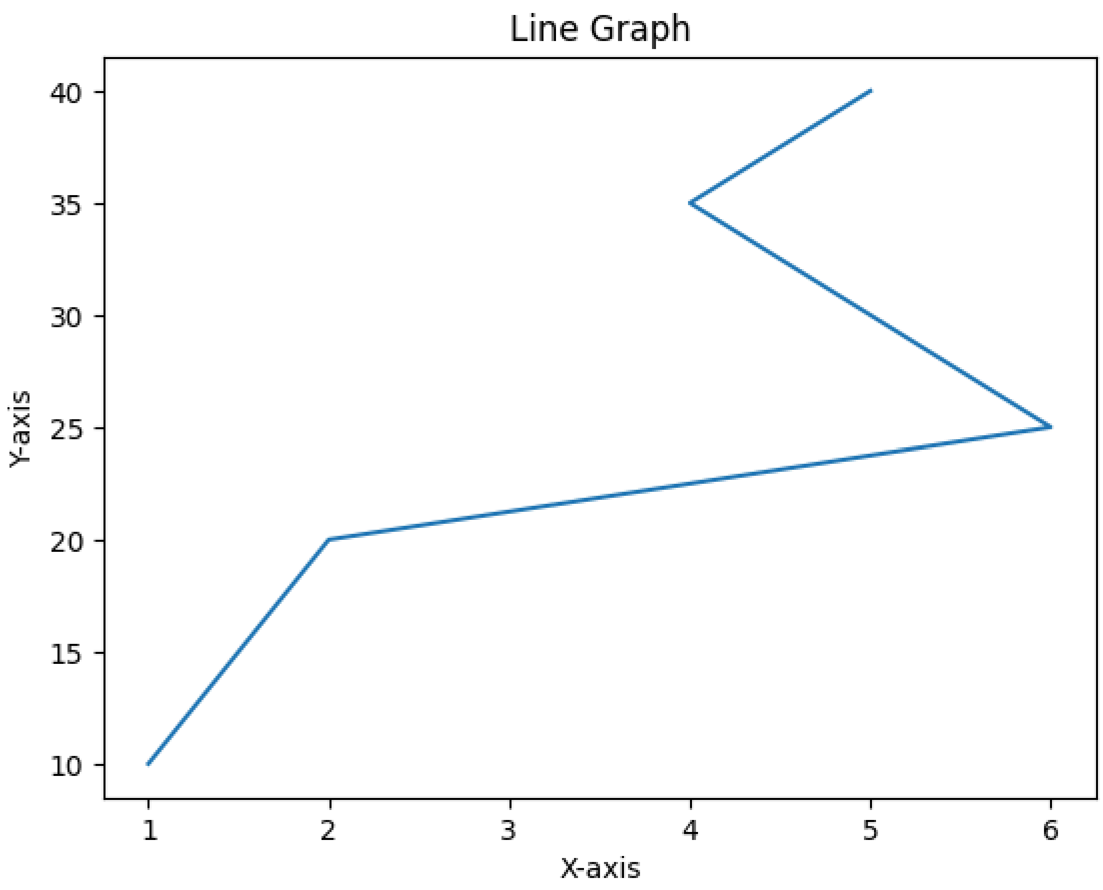
Key Features:

- Basic plotting (line, bar, scatter)
- Subplots
- Custom styling (colors, labels, legends)
- Save plots as images

In [15]:
```
import matplotlib.pyplot as plt

x = [1, 2, 6, 4, 5]
y = [10, 20, 25, 35, 40]

plt.plot(x, y)
plt.title("Line Graph")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```



A line graph is used to display information that changes continuously over time. In this graph, the X-axis represents numbers from 1 to 5, and the Y-axis shows corresponding values increasing steadily.

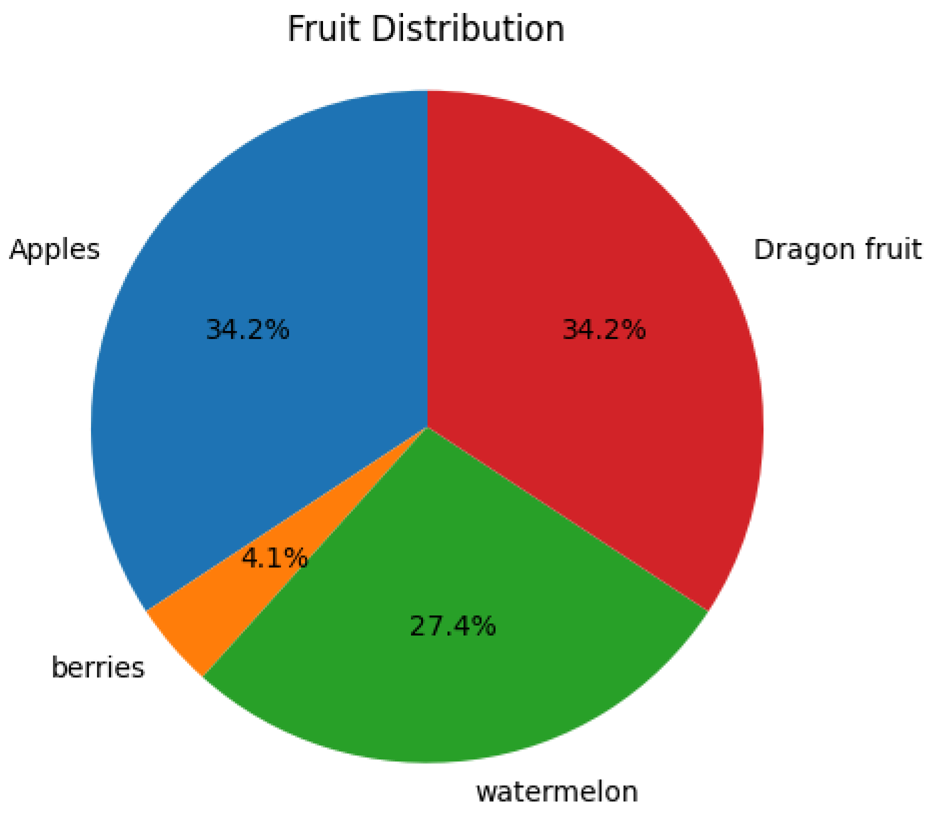
In [14]:
```
import matplotlib.pyplot as plt

# Data
labels = ['Apples', 'berries', 'watermelon', 'Dragon fruit']
sizes = [25, 3, 20, 25]

# Create pie chart
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)

# Equal aspect ratio ensures the pie is drawn as a circle.
plt.axis('equal')
plt.title("Fruit Distribution")

# Show the plot
plt.show()
```



labels: Names of each slice.

sizes: Values corresponding to each label.

autopct='%1.1f%%': Displays percentage values on the chart.

startangle=90: Rotates the chart for better visual alignment.

plt.axis('equal'): Keeps the pie circular instead of oval.

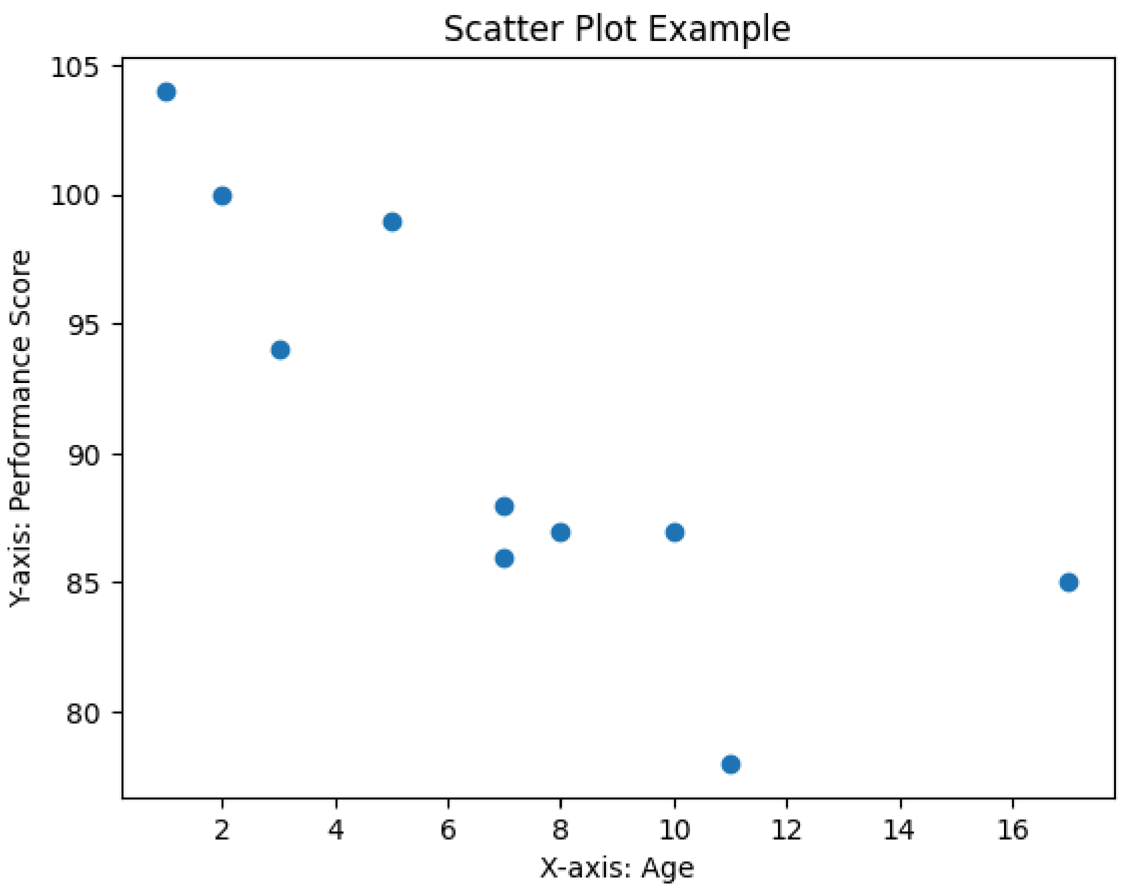
In [12]:
```
import matplotlib.pyplot as plt

# Data
x = [5, 7, 8, 7, 2, 17, 1, 10, 3, 11]
y = [99, 86, 87, 88, 100, 85, 104, 87, 94, 78]

# Create scatter plot
plt.scatter(x, y)

# Add labels and title
plt.xlabel("X-axis: Age")
plt.ylabel("Y-axis: Performance Score")
plt.title("Scatter Plot Example")

# Show the plot
plt.show()
```



plt.scatter(x, y): Plots each pair (x[i], y[i]) as a dot.

This example might represent how age (x) affects a performance score (y).

No lines are drawn between the points—only their positions matter.

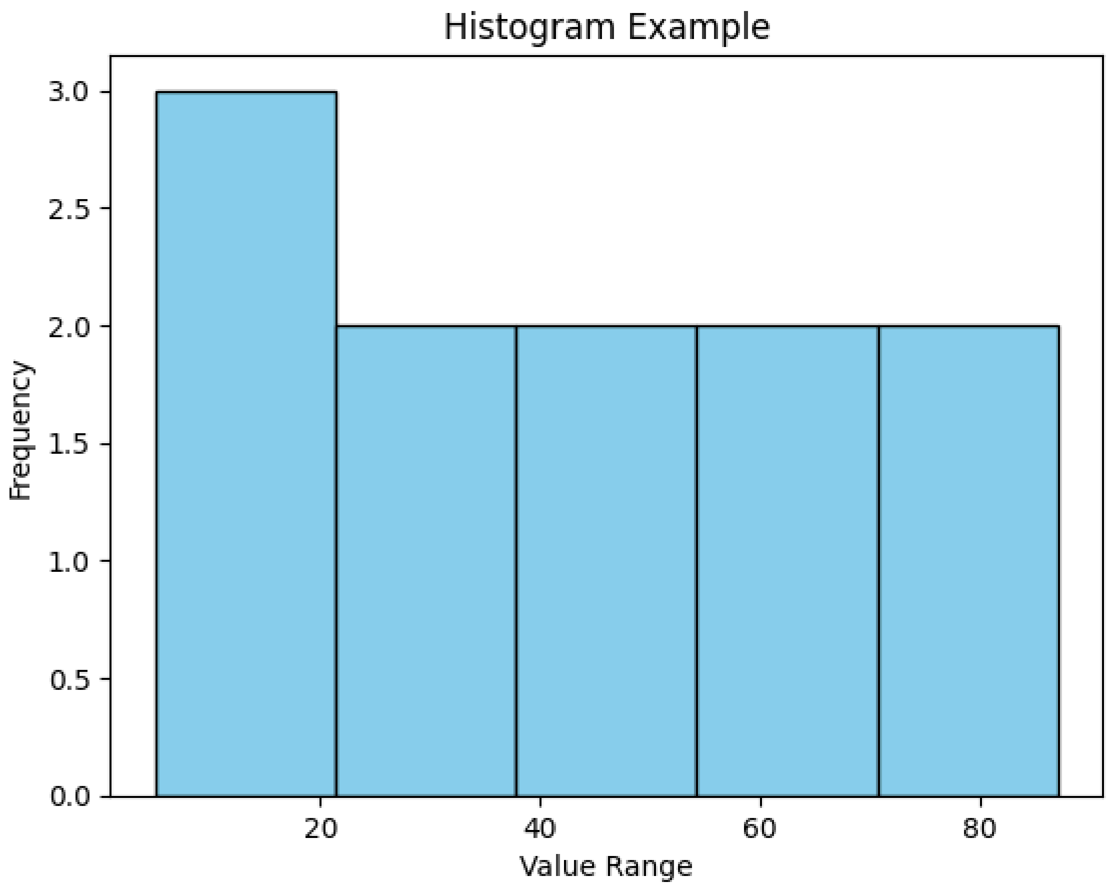
In [11]:
```
import matplotlib.pyplot as plt

# Data
data = [22, 87, 5, 43, 56, 73, 55, 54, 11, 20,  27]

# Create histogram
plt.hist(data, bins=5, color='skyblue', edgecolor='black')

# Add labels and title
plt.xlabel("Value Range")
plt.ylabel("Frequency")
plt.title("Histogram Example")

# Show the plot
plt.show()
```



data: A list of numerical values.

bins=5: The number of intervals (bars) to group the data into.

color and edgecolor: Used for styling the bars.

A histogram shows how frequently values appear within specified ranges.

MATPLOTLIB SUMMARY

Matplotlib is a powerful and widely-used Python library for creating static, animated, and interactive visualizations. It provides fine-grained control over every aspect of a plot, making it ideal for users who need customized and professional-looking charts.

Key Points:

Low-level control: Gives you full control over axes, labels, ticks, legends, and styles. Highly customizable: Great for creating publication-quality plots Versatile: Can produce a wide variety of visualizations like line graphs, bar charts, pie charts, scatter plots, and histograms. Compatible with many tools: Often used in combination with NumPy, Pandas, and Seaborn.

When to Use Matplotlib:

When you need precise customization. When building complex, multi-plot figures. When working in data science, engineering or scientific research where plot accuracy matters.

SEABORN OVERVIEW

Seaborn is a high-level Python data visualization library built on top of Matplotlib. It simplifies the creation of beautiful and informative statistical graphics, especially when working with Pandas DataFrames. Seaborn comes with built-in themes, color palettes, and functions designed to work well with structured datasets.

Key Features:

Simplified syntax: Create complex visualizations with less code. Better aesthetics: Cleaner and more visually appealing by default. Built-in support for Pandas: Easily plots DataFrame columns without extra processing. Advanced statistical plots: Includes box plots, violin plots, heatmaps, and regression plots.

When to Use Seaborn:

When you want to quickly create attractive plots. When working with data frames and categorical/statistical data. When you need built-in themes and color handling.

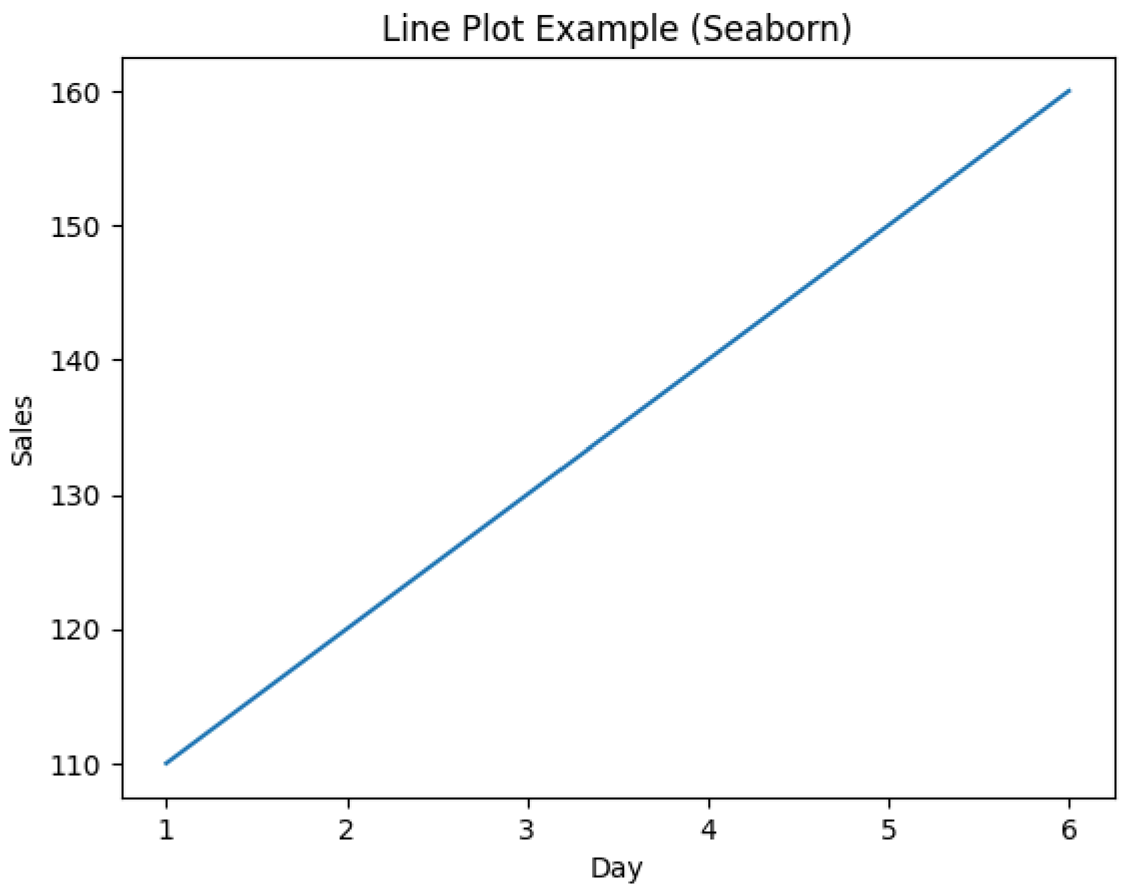
In [10]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# Create data
data = pd.DataFrame({
    "Day": [1, 2, 3, 5, 6],
    "Sales": [110, 120, 130, 150, 160]
})

# Create line plot
sns.lineplot(x="Day", y="Sales", data=data)

# Add labels and title
plt.xlabel("Day")
plt.ylabel("Sales")
plt.title("Line Plot Example (Seaborn)")

# Show plot
plt.show()
```



data: A DataFrame storing the data.

sns.lineplot(): Takes x, y, and data arguments to automatically plot a line chart.

Styling is cleaner out of the box—no need for additional design tweaks.

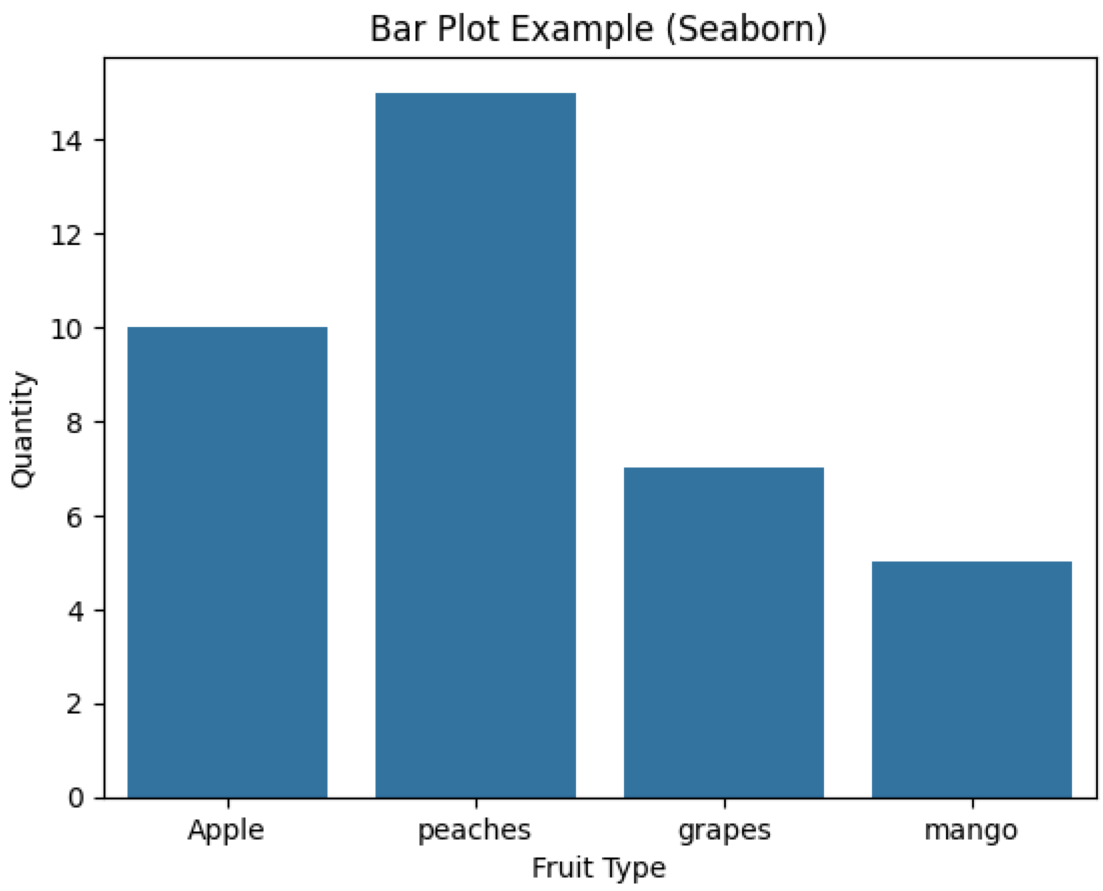
In [9]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# Create data
data = pd.DataFrame({
    "Fruit": ["Apple", "peaches", "grapes", "mango"],
    "Quantity": [10, 15, 7, 5]
})

# Create bar plot
sns.barplot(x="Fruit", y="Quantity", data=data)

# Add labels and title
plt.xlabel("Fruit Type")
plt.ylabel("Quantity")
plt.title("Bar Plot Example (Seaborn)")

# Show plot
plt.show()
```



sns.barplot() automatically calculates the mean (or displays given values if only one data point per category).

Works seamlessly with DataFrames.

Ideal for comparing group-wise statistics.

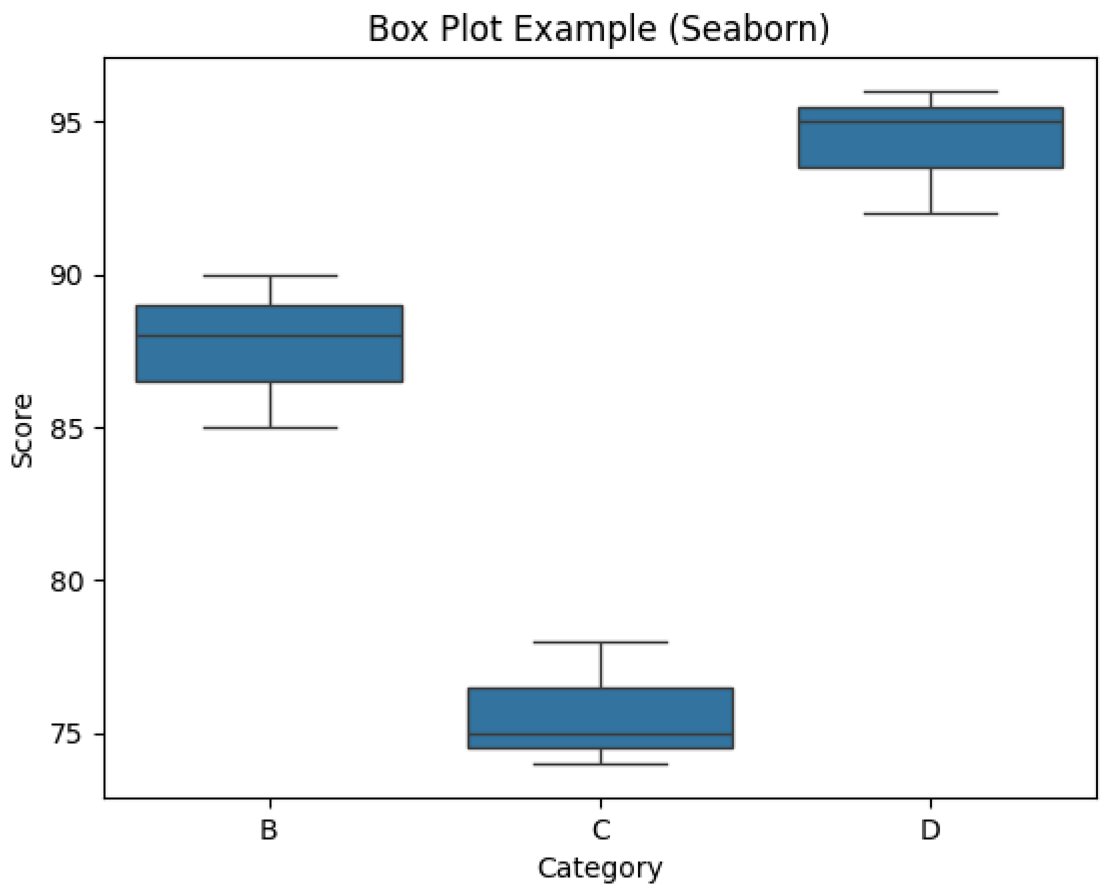
In [8]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# Create data
data = pd.DataFrame({
    "Category": ["B", "B", "B", "C", "C", "C", "D", "D", "D"],
    "Score": [85, 90, 88, 75, 78, 74, 95, 92, 96]
})

# Create box plot
sns.boxplot(x="Category", y="Score", data=data)

# Add labels and title
plt.xlabel("Category")
plt.ylabel("Score")
plt.title("Box Plot Example (Seaborn)")

# Show plot
plt.show()
```



Each box shows the interquartile range (IQR), with a line at the median.

Whiskers show the spread of the rest of the data, and dots indicate outliers.

Great for comparing distributions across categories.

FEATURES MATPLOTLIB SEABORN

EASE OF USE: 1)requires more code for customization and plotting 1)easysyntax, less code for common plots.ideal for quick exploration CUSTOMIZATION OPTIONS: 2)highly coustomization with full control over every plot 2)less control but comes with attractive default styles and themes
(eg.lines,markers etc..)
INTERACTIVITY: 3)can be made with interactive with `mpl_toolkits` 3)primarily static but can be combined with `plotly` GRAPH TYPES: 4) offer a wide range of graph types,including 4) focuses on statistical plots,with some special features like heatmaps. coustomization for complex graphs COMMUNITY& DOCUMENTATION: 5) extensive documentation and a very active community 5) community is smaller but growing fast,and documentation growing steady

MATPLOTLIB:

Strengths: Full control over every element, great for scientific work where precision is critical. Best for creating complex or multi-plot figures.

Weaknesses: More verbose code and requires more effort to make plots visually appealing.

SEABORN:

Strengths: Quick, beautiful, and effective for statistical data analysis. Ideal for users who want to make plots with minimal effort.

Weaknesses: Less flexibility for custom plots and complex customization. Slower with very large datasets.