Visualization Library Documentation: Matplotlib and Pandas.

1.Matplotlib:

Matplotlib is a library for creating interactive visualizations in Python.Matplotlib provides control over every aspect of a figure and allows for detailed customization. It is commonly used in data science, academic research, and engineering for creating publication-quality plots.

Features of Matplotlib

1.Extensive range of plot types.

2.High level of customization.

3.Integration with other libraries such as NumPy and pandas.

4.Ability to create complex subplots and multi-figure plots.

5.Support for exporting plots in various formats (PNG, PDF, SVG, etc.).

Use case: 1.Scientific research and engineering. 2.Data analysis and exploration. 3.Publication-quality figures and charts.

Graph Types in Matplotlib

1.Line plot, 2.Scatter plot, 3.Bar chart, 4.Histogram, 5.Pie chart, 6.Box plot, 7.Heatmap, 8.Area plot, 9.Stem Plot,10.3D plot

1.Line Plot

A line plot is used to display data points connected by a line, showing trends over time.

Use Case: Visualizing time series data, trends, or continuous data.

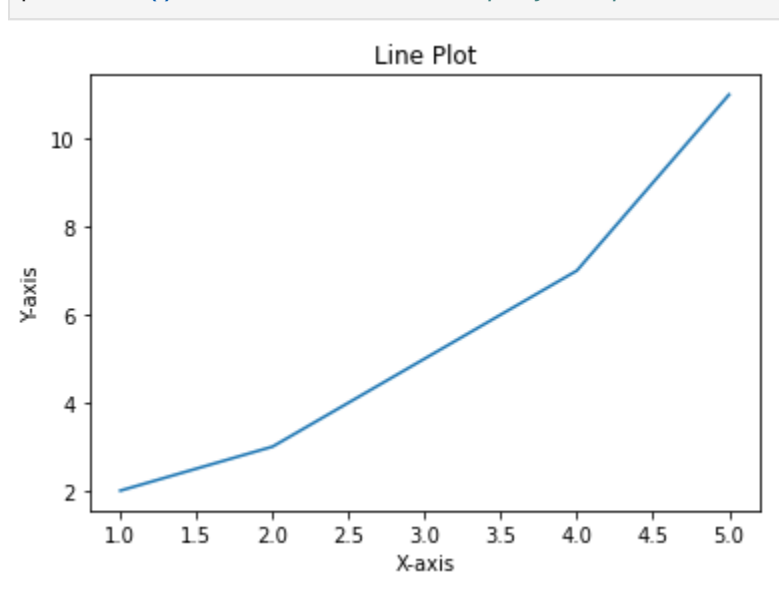
In [1]:
```
#example

#to work with matplotlib we have to install it using command "!pip install matplotlib"

#importing the required library
import matplotlib.pyplot as plt

#x and y are sample data
x = [1, 2, 3, 4, 5]
y = [2, 3, 5, 7, 11]


plt.plot(x, y)
plt.title("Line Plot") #title of the line plot
plt.xlabel("X-axis")   #x label name
plt.ylabel("Y-axis")   #y label name
plt.show()             #to display output of the line plot
```



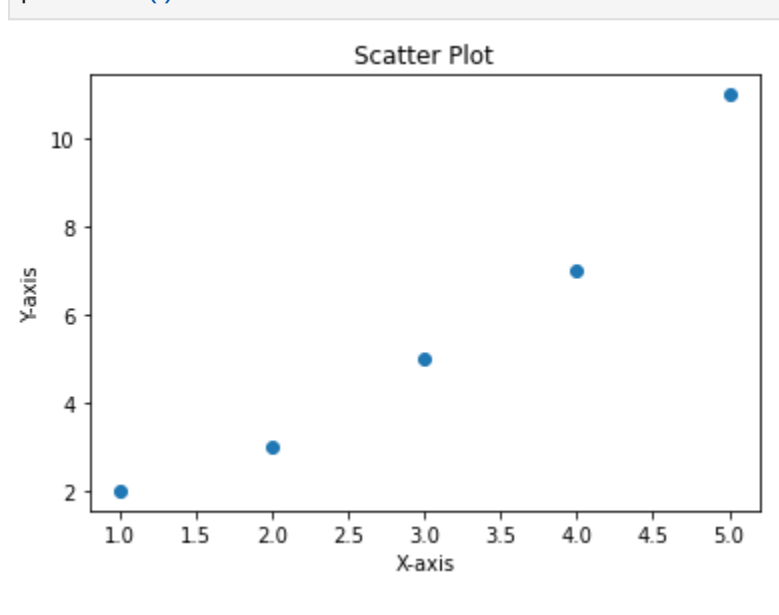
2.Scatter Plot

A scatter plot uses dots to represent values for two different numeric variables.

Use Case: Identifying relationships or correlations between two variables.

In [2]:
```
x = [1, 2, 3, 4, 5]
y = [2, 3, 5, 7, 11]

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



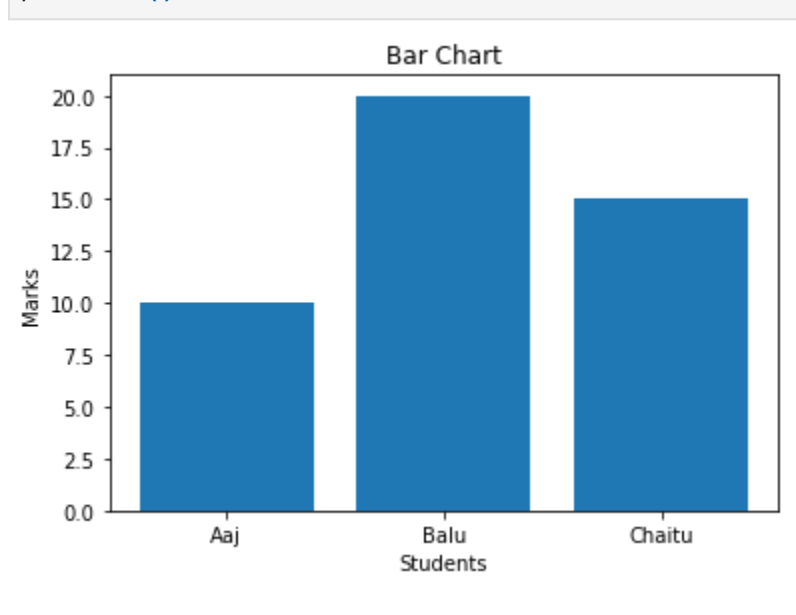
3.Bar Chart

A bar chart presents categorical data with rectangular bars.

Use Case: Comparing different categories or groups

In [5]:
```
Students = ['Aaj', 'Balu', 'Chaitu']
Marks = [10, 20, 15]

plt.bar(Students, Marks)
plt.title("Bar Chart")
plt.xlabel("Students")
plt.ylabel("Marks")
plt.show()
```



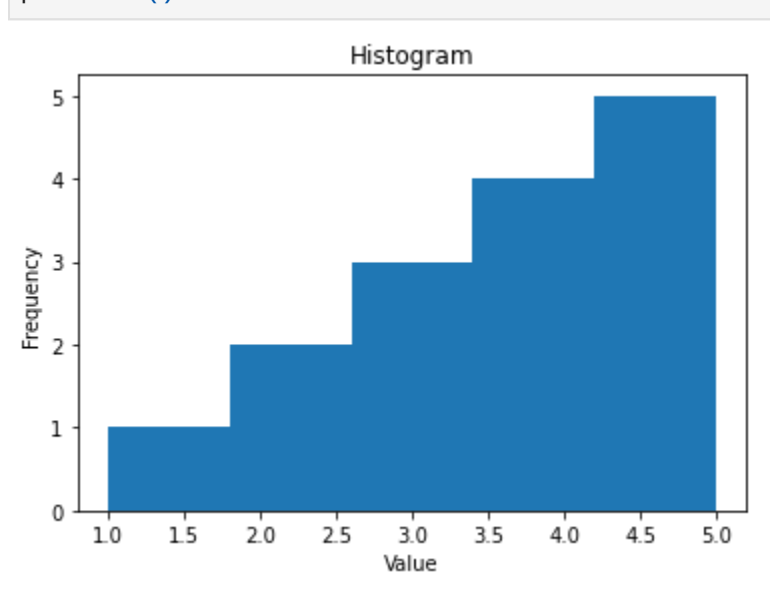
4.Histogram

A histogram displays the distribution of a dataset.

Use Case: Understanding the distribution and frequency of data points.

In [6]:
```
data = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5]

plt.hist(data, bins=5)
plt.title("Histogram")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()
```



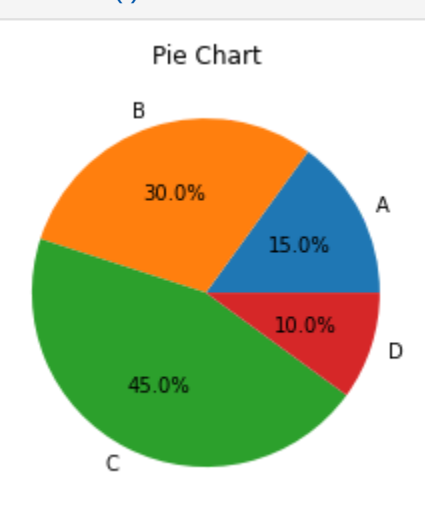
5.Pie Chart

A pie chart represents data as slices of a circle, showing proportions.

Use Case: Displaying the composition of a whole.*

In [7]:
```
sizes = [15, 30, 45, 10]
labels = ['A', 'B', 'C', 'D']

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



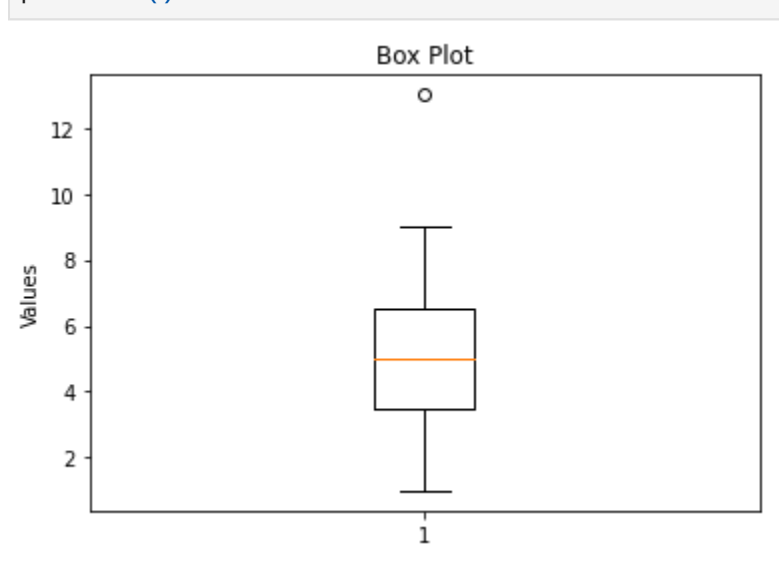
6.Boxplot

Displays the distribution of data based on a five-number summary, highlighting outliers.

Use Case: Identifying outliers and visualizing the distribution of exam scores across different classrooms.

In [22]:
```
data = [1, 2, 2, 3, 4, 4, 4, 5, 5, 5, 6, 7, 8, 9, 13]

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



7.Heatmap

Uses colors to represent values in a matrix, showing relationships between variables.

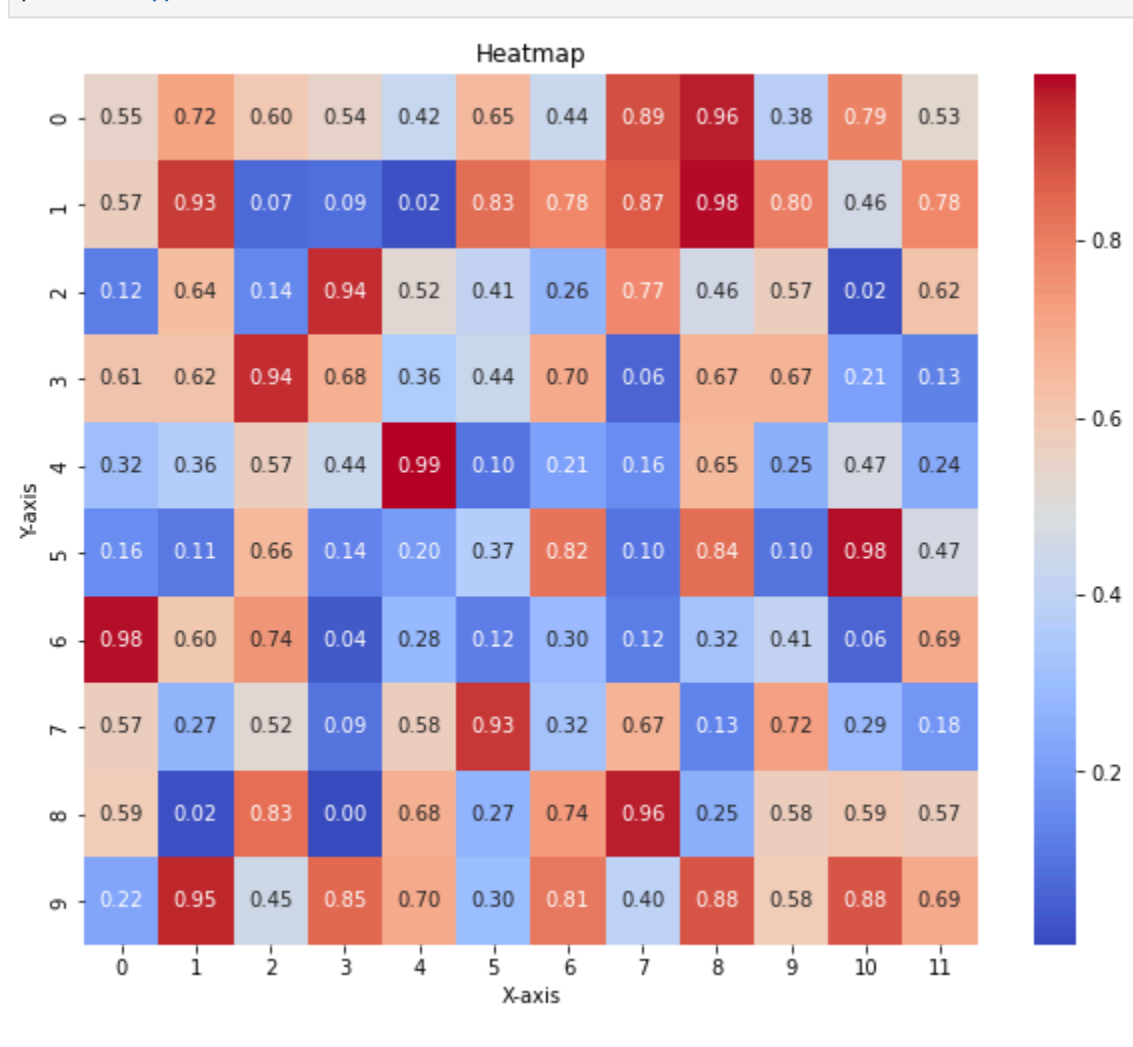
Use Case: Displaying correlation between different variables in a dataset for easier identification of patterns.

In [17]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

np.random.seed(0)  #used to set random numbers
data = np.random.rand(10, 12)

# Creating the heatmap with labels
plt.figure(figsize=(10, 8))
heatmap = sns.heatmap(data, cmap='coolwarm', annot=True, fmt=".2f")

plt.title("Heatmap")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```



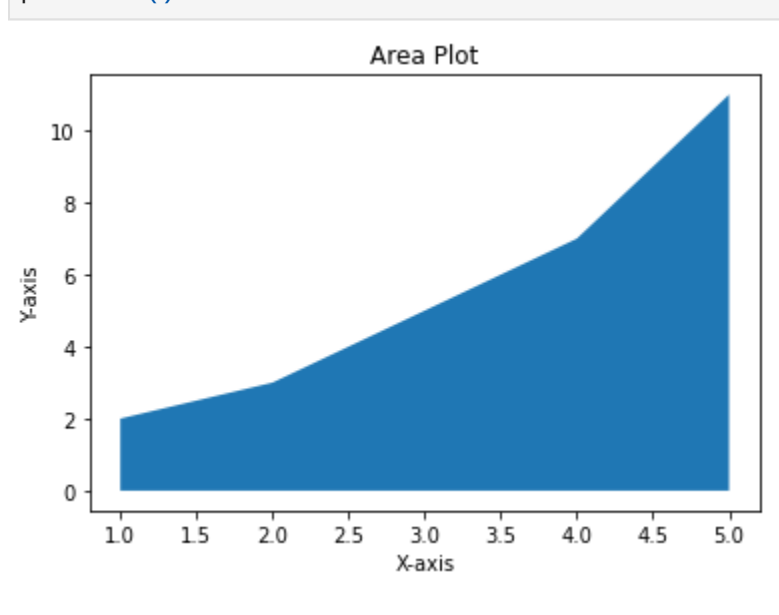
8.Area Plot

Similar to a line plot but fills the area under the line, often used to show cumulative data.

Use Case: Visualizing cumulative revenue over time to show overall growth trends.

In [10]:
```
x = [1, 2, 3, 4, 5]
y = [2, 3, 5, 7, 11]

plt.fill_between(x, y)
plt.title("Area Plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```



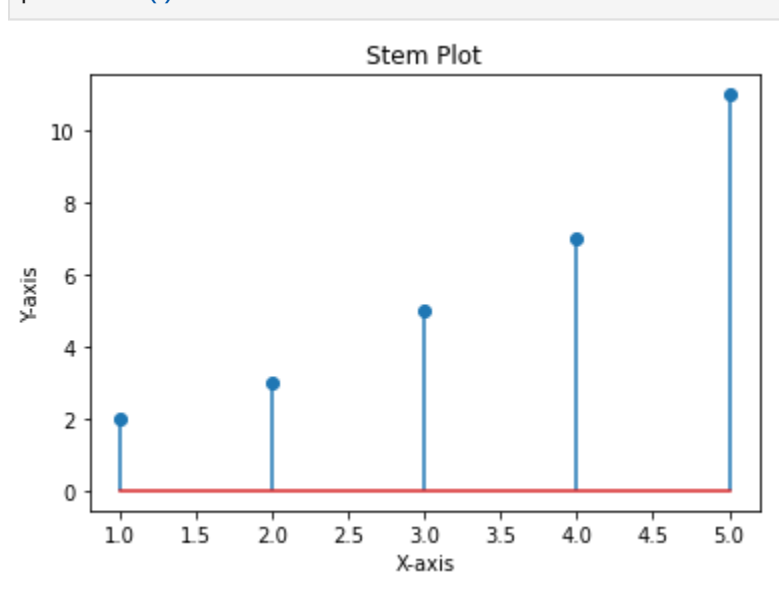
9.Stem Plot

Displays data points with lines extending from a baseline to the data points, useful for visualizing discrete data.

Use Case: Representing discrete signal data in a time series for digital signal processing.

In [11]:
```
x = [1, 2, 3, 4, 5]
y = [2, 3, 5, 7, 11]

plt.stem(x, y)
plt.title("Stem Plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```



10.3D Plot:

A 3D plot displays data points in three dimensions, providing a visual representation of complex data with an additional depth dimension.

Use Case: Visualizing the topography of a geographical area to analyze elevation changes.

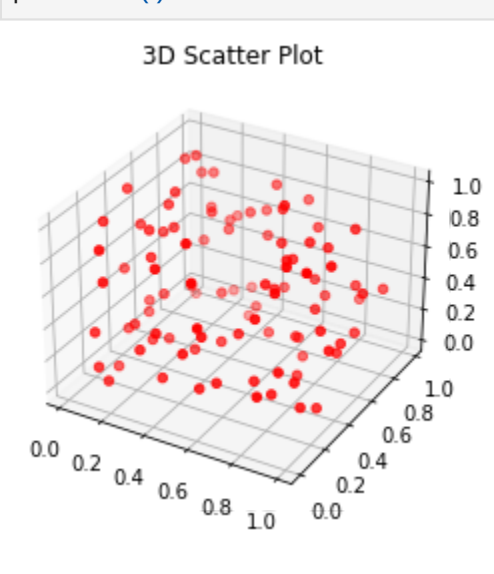
In [14]:
```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Generate data
x = np.random.rand(100)
y = np.random.rand(100)
z = np.random.rand(100)

# Plot 3D scatter
ax.scatter(x, y, z, c='r', marker='o')

plt.title("3D Scatter Plot")
plt.show()
```



2.Seaborn

Seaborn is a statistical data visualization library based on Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Features of Seaborn:

1.Built-in themes for improved aesthetics

2.Simplifies complex visualizations

3.Integrates well with pandas DataFrames

4.Statistical plotting functions

Use Cases:

Exploratory data analysis, Statistical modeling, Data visualization in data science projects.

Graph Type in Seaborn:

Line plot,scatter plot,bar chart,histogram,Box plot,violin plot,Heatmap

import the seaborn as sns to work with seaborn library

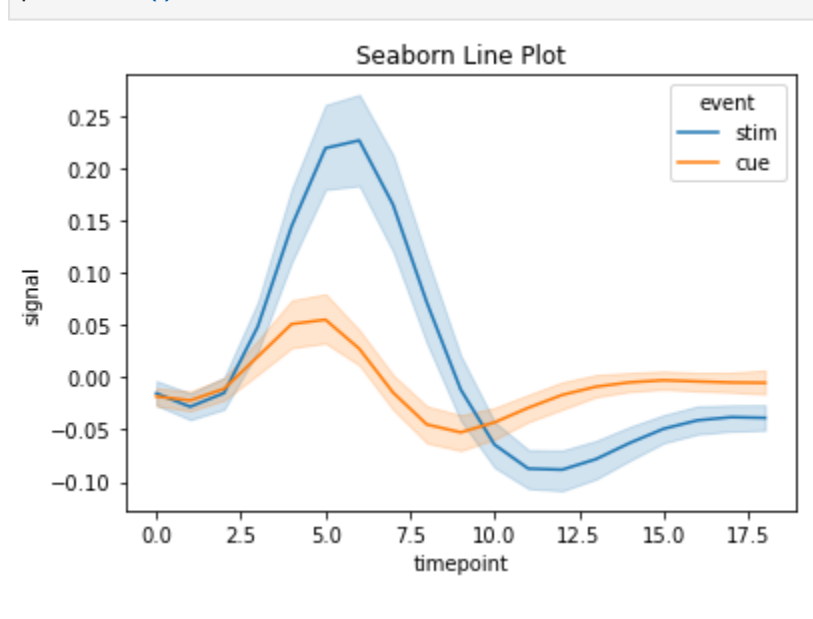
1.Line plot

Line plots in Seaborn enhance Matplotlib line plots with additional functionalities and improved aesthetics. They are useful for visualizing trends over time.

Use Case: Ideal for time series data, such as tracking changes in stock prices over time

In [27]:
```
import seaborn as sns
import matplotlib.pyplot as plt

data = sns.load_dataset("fmri")
sns.lineplot(x="timepoint", y="signal", hue="event", data=data)
plt.title("Seaborn-Line Plot")
plt.show()
```

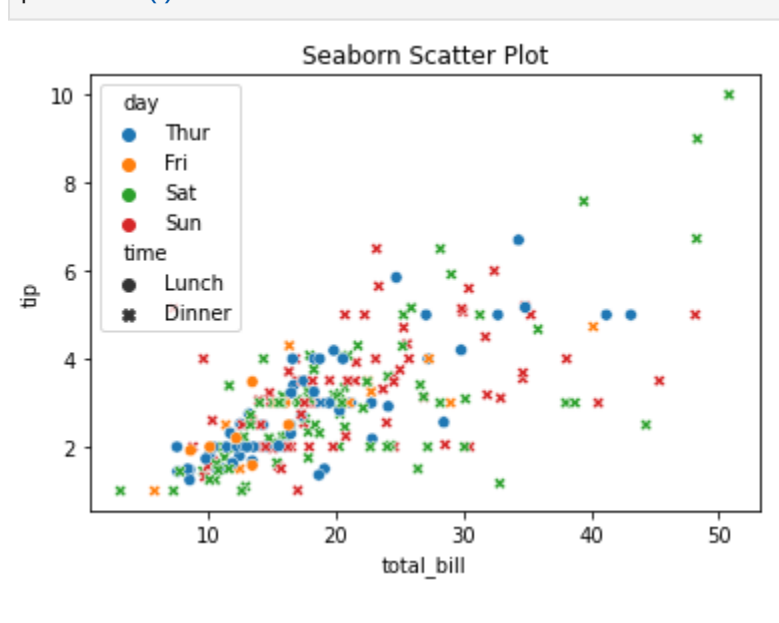


2.Scatter Plots:

Seaborn scatter plots offer enhanced capabilities with support for multiple variables, allowing for the differentiation of data points by color and shape.

Use Case: Useful for examining relationships between two or more variables, such as comparing total bill and tip amount in a restaurant dataset.

In [29]:
```
data = sns.load_dataset("tips")
sns.scatterplot(x="total_bill", y="tip", hue="day", style="time", data=data)
plt.title("Seaborn Scatter Plot")
plt.show()
```

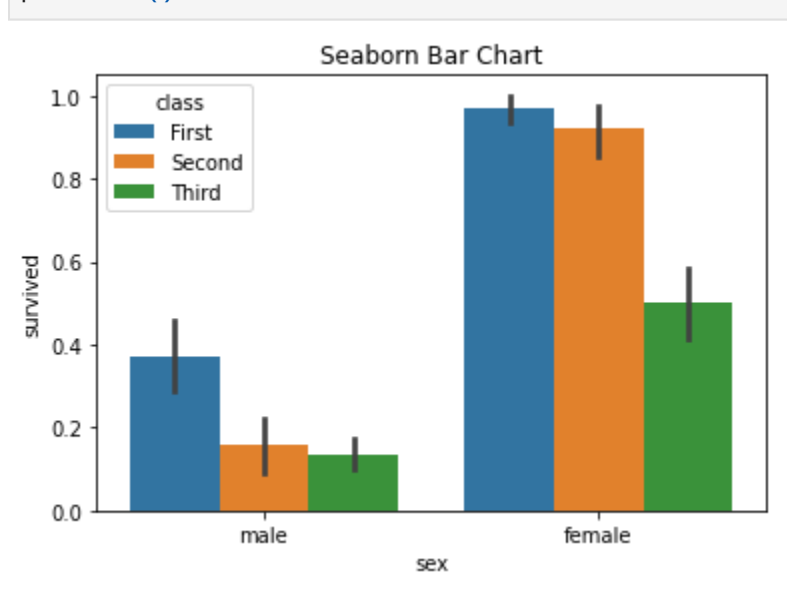


3.Bar Charts:

Bar charts in Seaborn provide additional statistical estimations and improved aesthetics. They can show counts or other summary statistics.

*Use Case:*Ideal for comparing different categories, such as the survival rate of different passenger classes on the Titanic.

In [31]:
```
data = sns.load_dataset("titanic")
sns.barplot(x="sex", y="survived", hue="class", data=data)
plt.title("Seaborn Bar Chart")
plt.show()
```

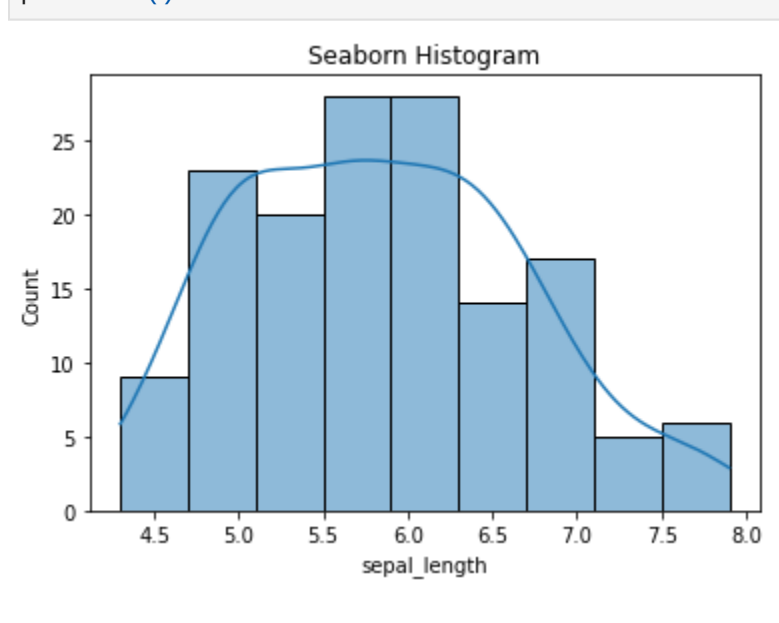


4.Histograms:

Seaborn histograms facilitate the creation of histograms with additional features like kernel density estimation (KDE), which shows the distribution of a dataset.

Use Case: Useful for understanding the distribution of a single variable, such as the length of sepal in the iris dataset.

In [33]:
```
data = sns.load_dataset("iris")
sns.histplot(data["sepal_length"], kde=True)
plt.title("Seaborn Histogram")
plt.show()
```

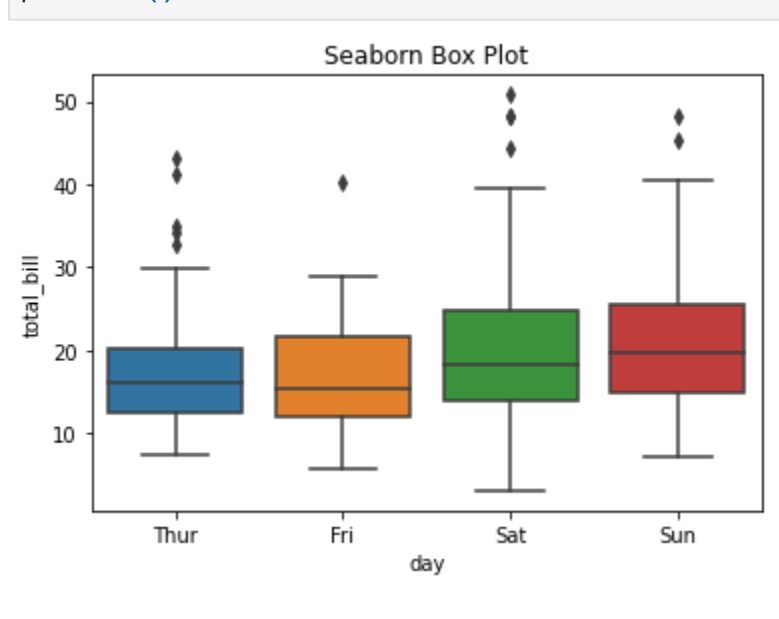


5.Box Plots:

Box plots display the distribution of data based on a five-number summary (minimum, first quartile, median, third quartile, and maximum).

Use Case: Useful for identifying outliers and comparing distributions across multiple groups.

In [34]:
```
data = sns.load_dataset("tips")
sns.boxplot(x="day", y="total_bill", data=data)
plt.title("Seaborn Box Plot")
plt.show()
```



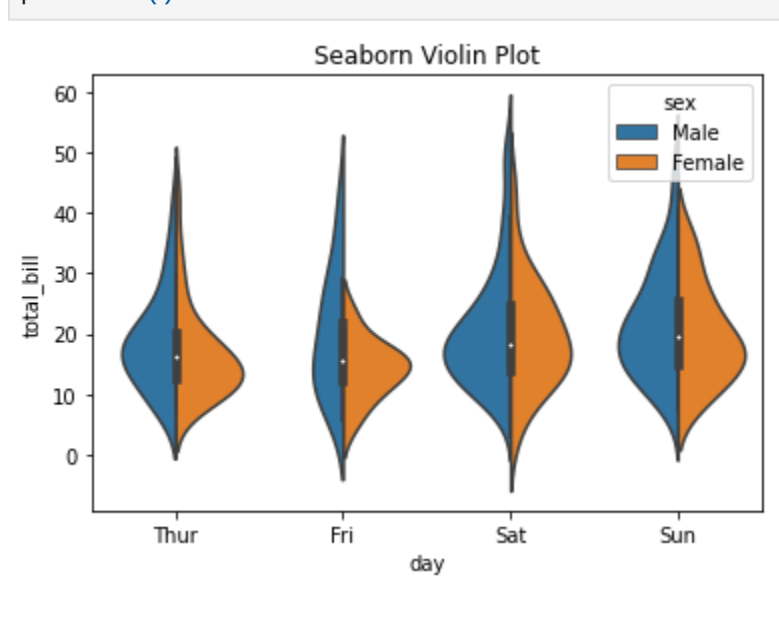
6.Violin Plots

Violin plots combine aspects of box plots and KDE plots. They provide a deeper understanding of the data distribution.

Use Case: Useful for visualizing the distribution and probability density of the data across different categories.

In [37]:
```
import seaborn as sns
import matplotlib.pyplot as plt

data = sns.load_dataset("tips")
sns.violinplot(x="day", y="total_bill", hue="sex", data=data, split=True)
plt.title("Seaborn Violin Plot")
plt.show()
```

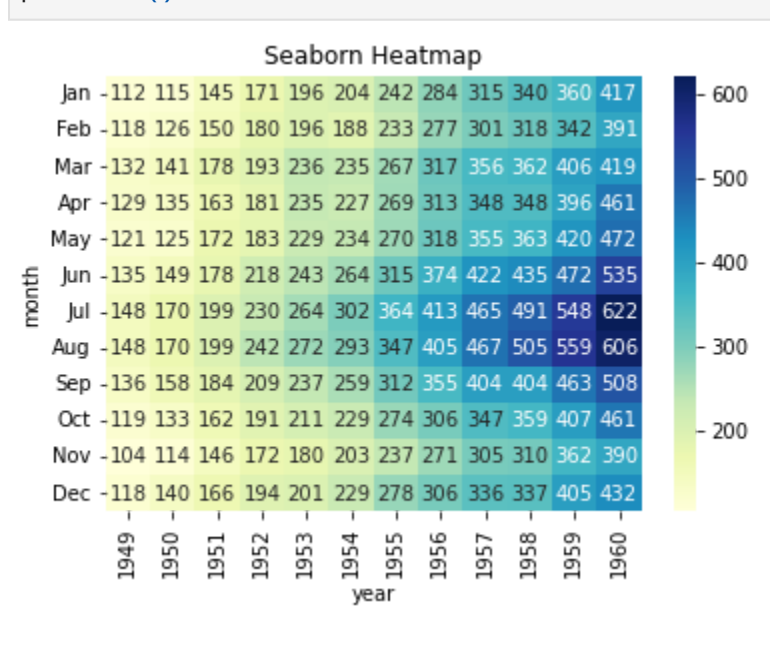


7.Heatmaps

Heatmaps show data in a matrix format where values are represented by color. They are useful for identifying patterns and correlations.

Use Case: Ideal for correlation matrices, showing the relationship between different variables in a dataset.

In [38]:
```
data = sns.load_dataset("flights")
data_pivot = data.pivot("month", "year", "passengers")
sns.heatmap(data_pivot, annot=True, fmt="d", cmap="YlGnBu")
plt.title("Seaborn Heatmap")
plt.show()
```



Comparison of Matplotlib and Seaborn

1.Ease of Use:

Matplotlib: Requires more code and configuration to achieve desired results. Provides detailed control over every aspect of the plot.

Seaborn: Simplifies the process of creating complex visualizations. Built on top of Matplotlib, it allows for quicker and more aesthetic plotting.

2.Customization Options:

Matplotlib: Offers extensive customization options. Users can control every element of the plot, making it highly flexible.

Seaborn: Provides fewer customization options directly but integrates well with Matplotlib for advanced customizations.

3.Interactivity:

Matplotlib: Supports interactive plotting through integrations with interactive backends like ipympl.

Seaborn: Primarily focused on static plots but can be combined with interactive libraries for enhanced interactivity.

4.Performance with Large Datasets

Matplotlib: Handles large datasets reasonably well, though performance can degrade with very large data.

Seaborn: Similar performance to Matplotlib since it is built on top of it, but may offer slight improvements due to optimized functions for statistical plots