# VISUALIZATION

**Visualization** in data science refers to the graphical representation of data and information using visual elements like charts, graphs, and maps. It helps simplify complex data and reveals patterns, trends, and outliers that may not be obvious in raw datasets. By converting numbers and text into visuals, it allows for quicker understanding, better communication of insights, and more informed decision-making. Visualization tools and libraries like Matplotlib, Seaborn, Plotly, and Pandas enable data scientists to explore data interactively and present findings effectively to both technical and non-technical audiences.

Now lets discuss Pandas and Seaborn library deeply

## LIBRARY OVERVIEW

**Pandas** is a Python library used for data manipulation and analysis, offering powerful tools to work with structured data using DataFrames.
**Seaborn** is a visualization library built on Matplotlib that simplifies the creation of attractive and informative statistical graphics, with built-in themes and support for pandas DataFrames.

## GRAPHS

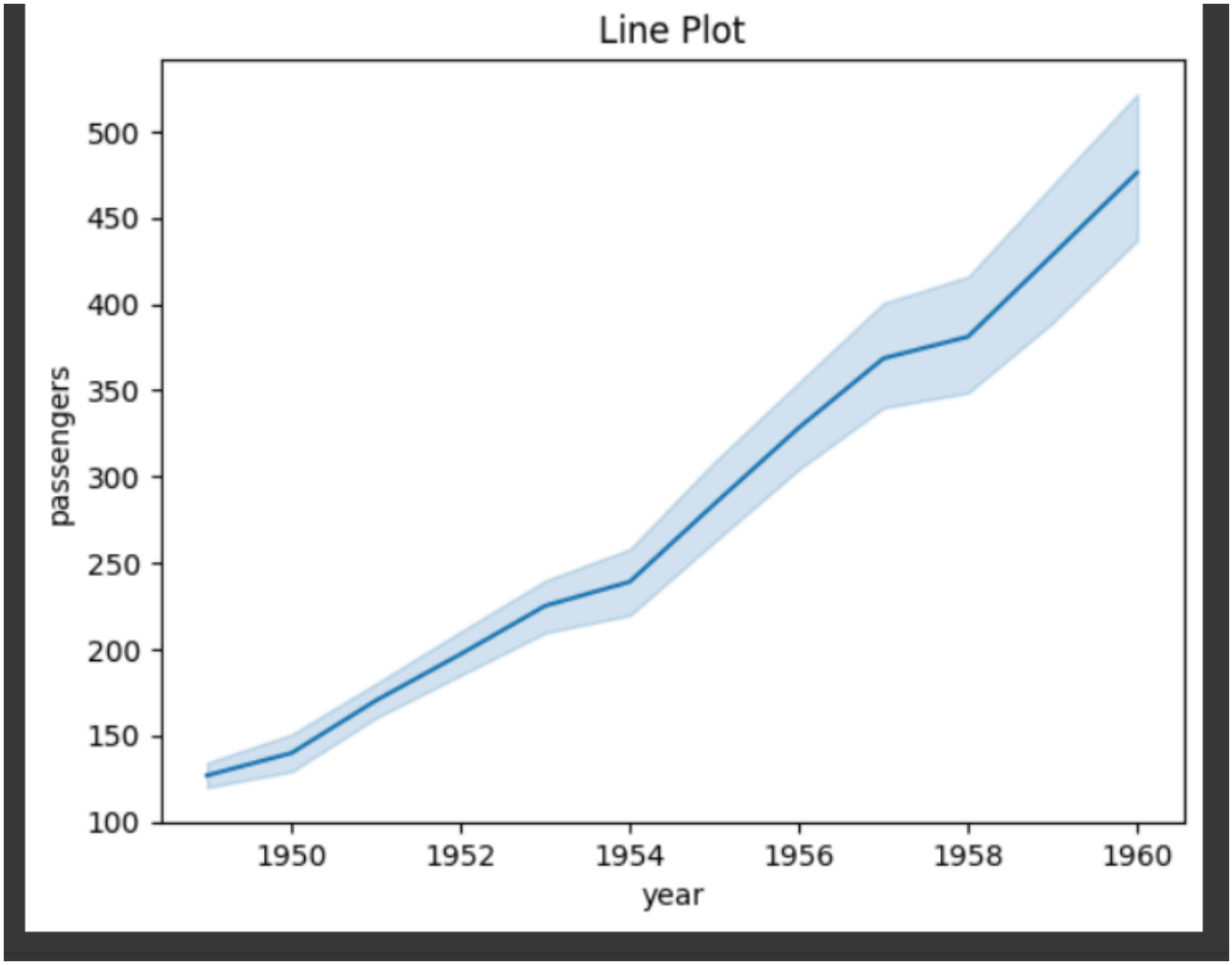
### SEABORN

#### 1.Line Plot

**Description:** Shows the relationship between two continuous variables.
**Use Case:** Trend over time, temperature change, stock price, etc.

```
import seaborn as sns
import matplotlib.pyplot as plt

flights = sns.load_dataset("flights")
sns.lineplot(x="year", y="passengers", data=flights)
plt.title("Line Plot")
plt.show()
```

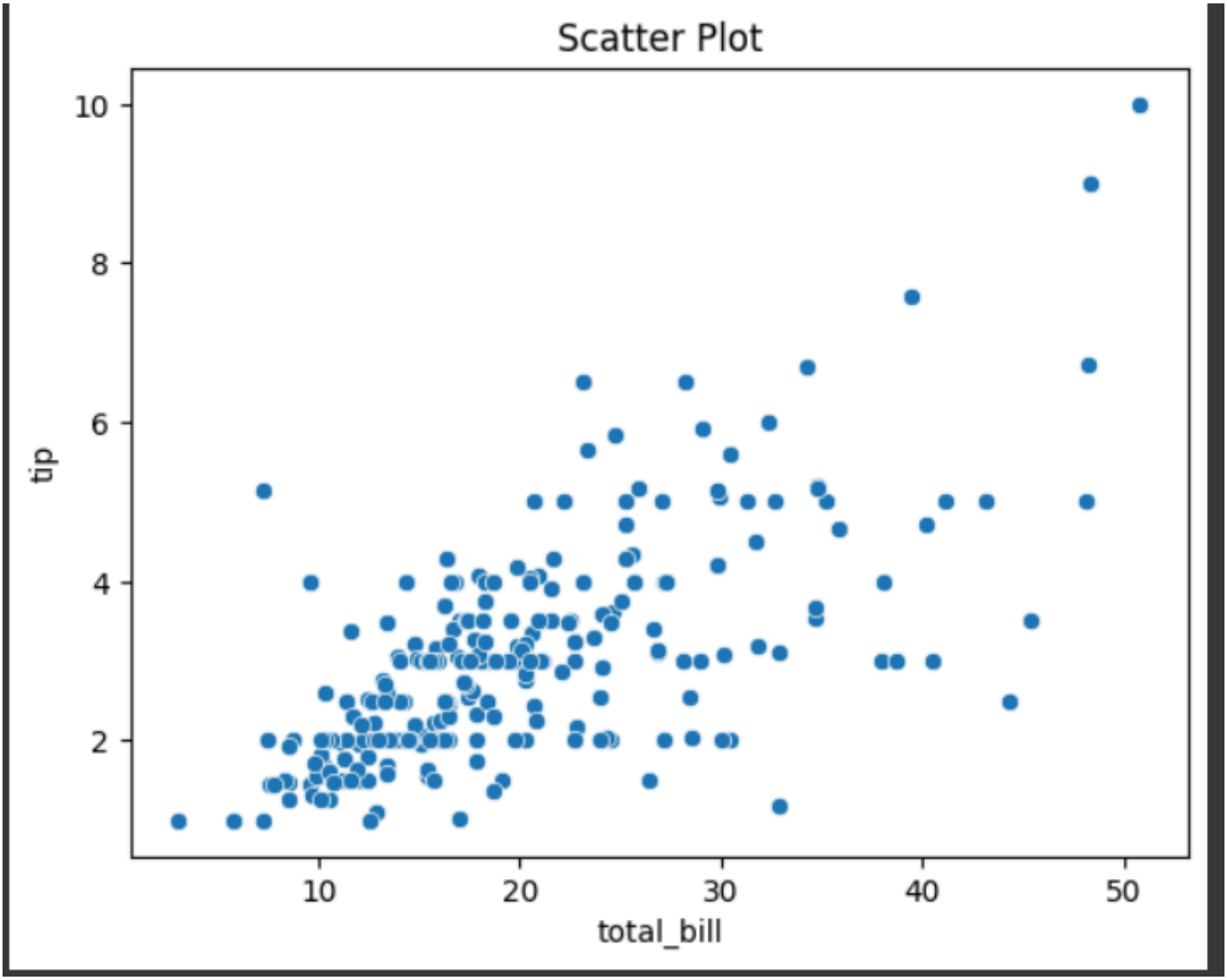
and the output it gives




2. **Scatter Plot**

**Description:** Plots individual data points for two continuous variables.
**Use Case:** Correlation between height and weight, income vs age, etc.

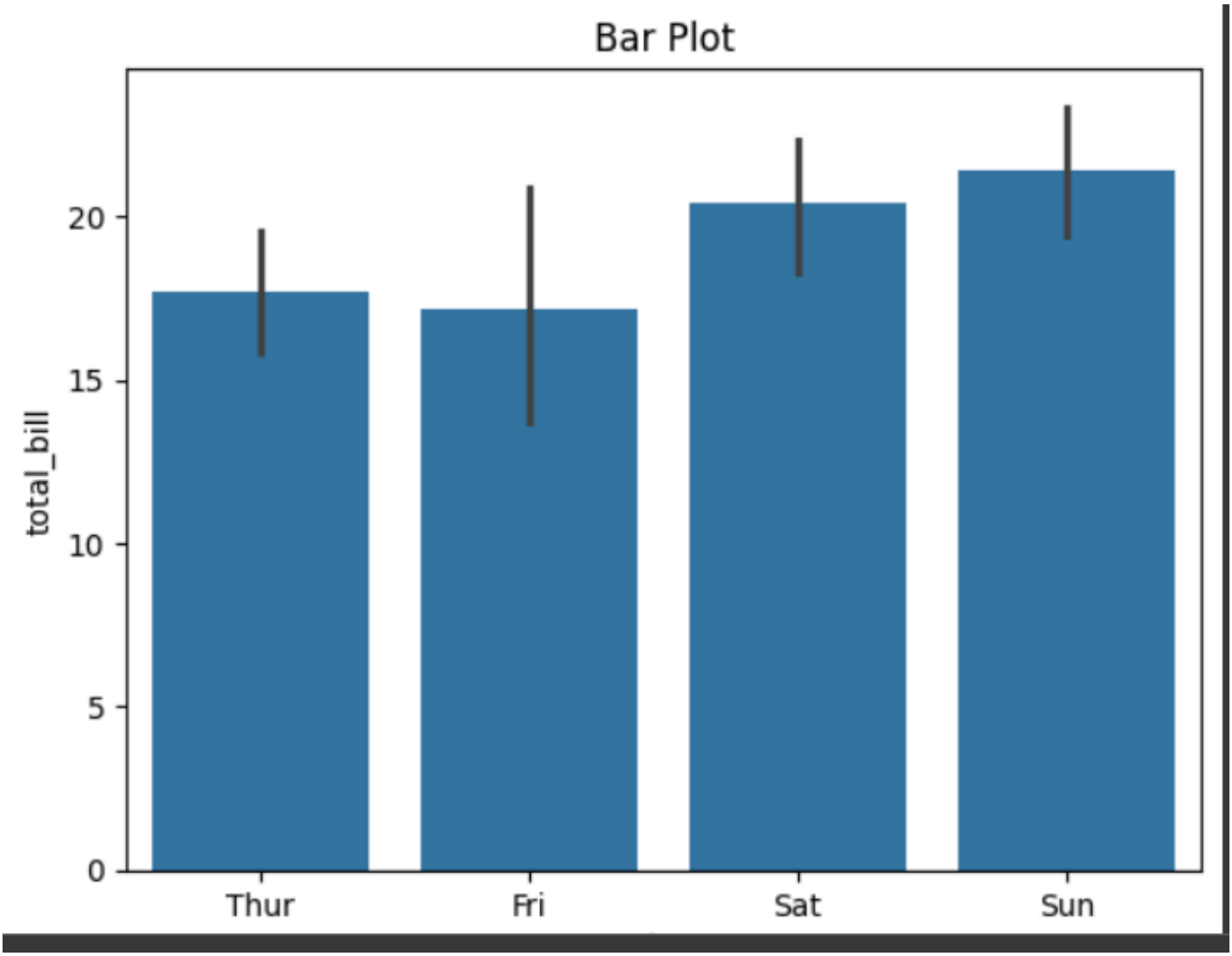
```
tips = sns.load_dataset("tips")
sns.scatterplot(x="total_bill", y="tip", data=tips)
plt.title("Scatter Plot")
plt.show()
```




3. **Bar Plot**

**Description:** Displays mean or other aggregation of values across categories.
**Use Case:** Average salary by department, average tip by day.

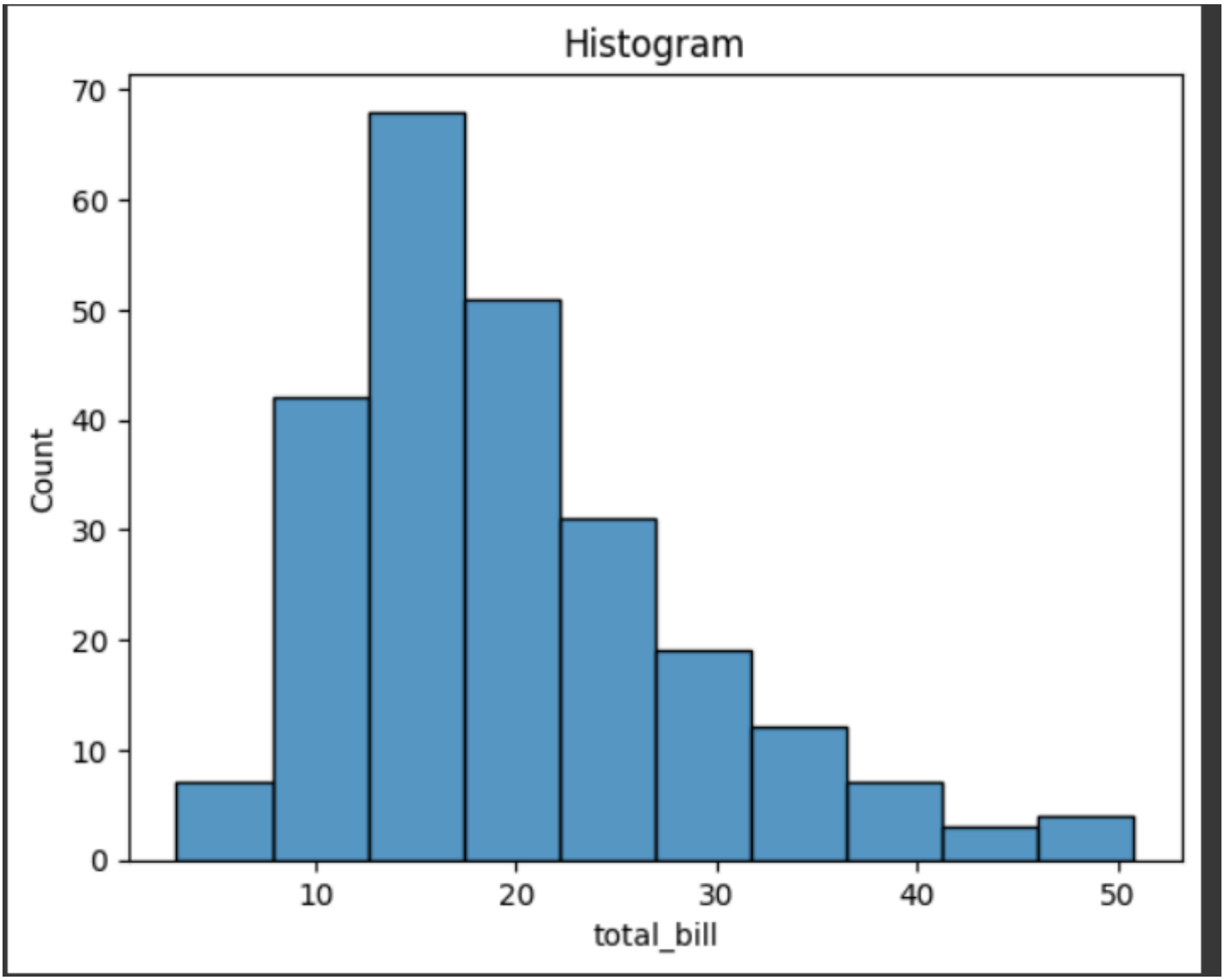
```
sns.barplot(x="day", y="total_bill", data=tips)
plt.title("Bar Plot")
plt.show()
```




4. **Histogram**

**Description:** Shows the distribution of a single variable by dividing into bins.
**Use Case:** Age distribution, exam score frequencies.

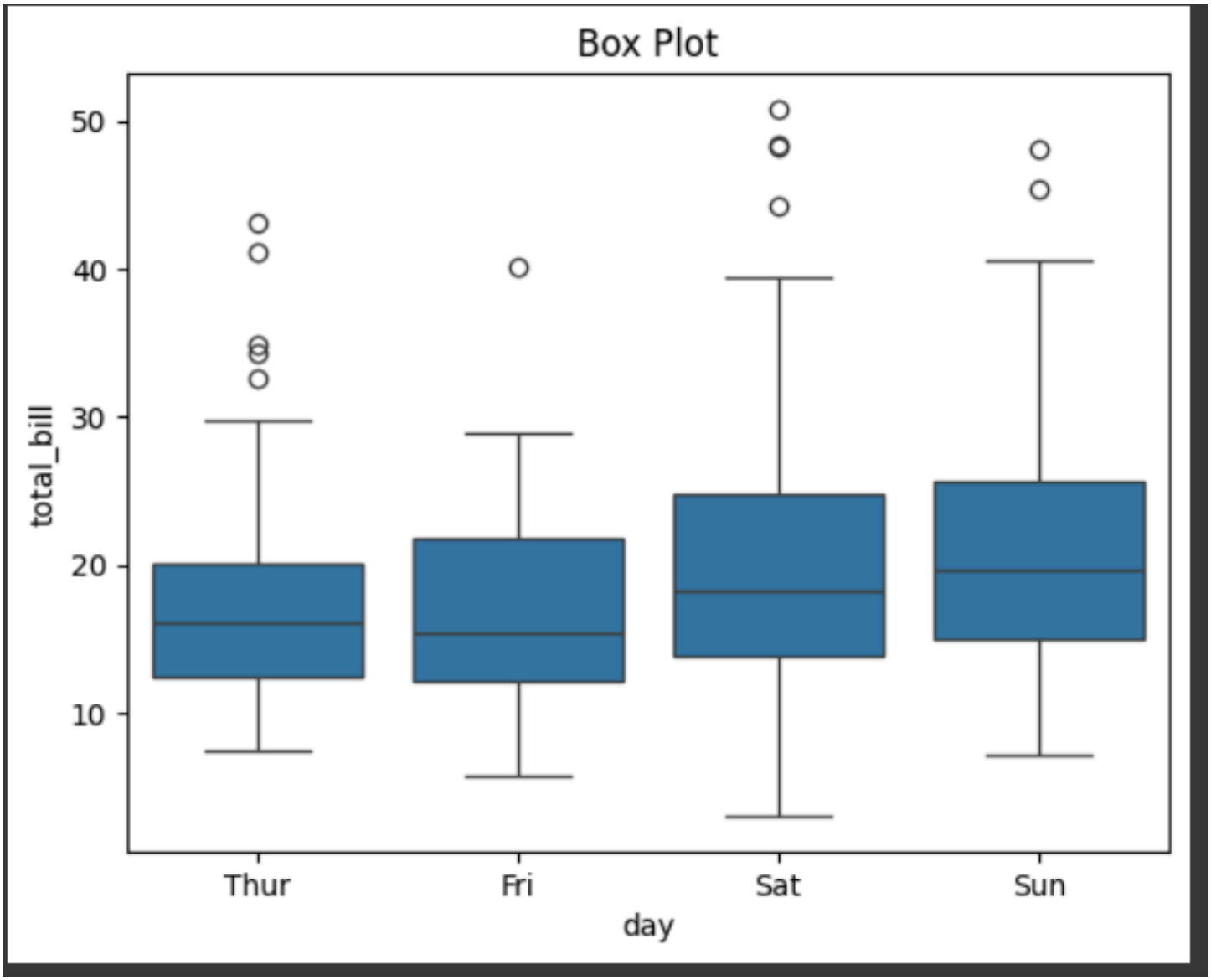
```
sns.histplot(data=tips["total_bill"], bins=10)
plt.title("Histogram")
plt.show()
```




5. **Box Plot**

**Description:** Shows distribution, median, and outliers using quartiles.
**Use Case:** Income variation, bill amounts by day.

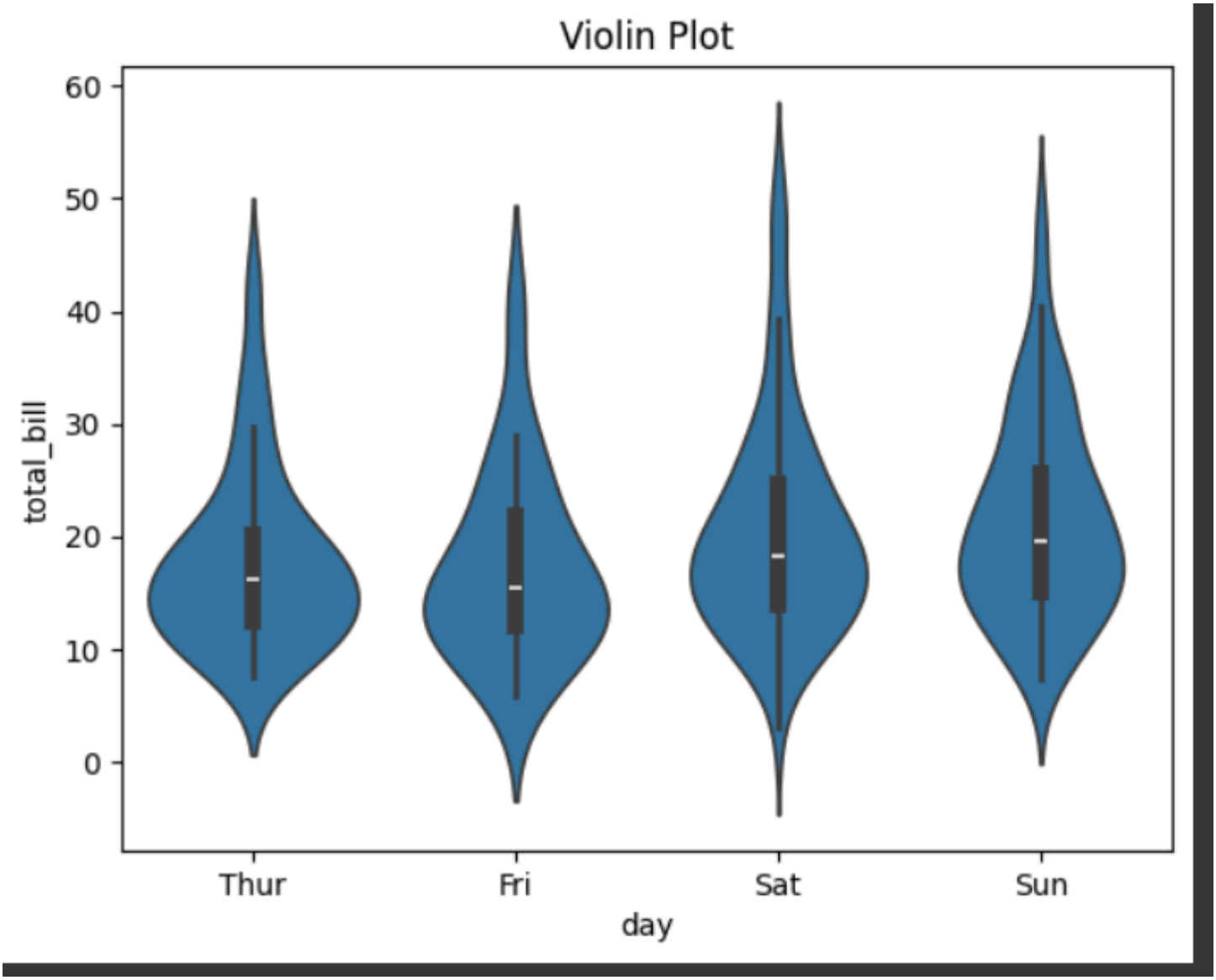
```
sns.boxplot(x="day", y="total_bill", data=tips)
plt.title("Box Plot")
plt.show()
```




## 6. Violin Plot

**Description: Combines boxplot and KDE to show distribution and density.**
**Use Case: Compare distribution of marks or income.**

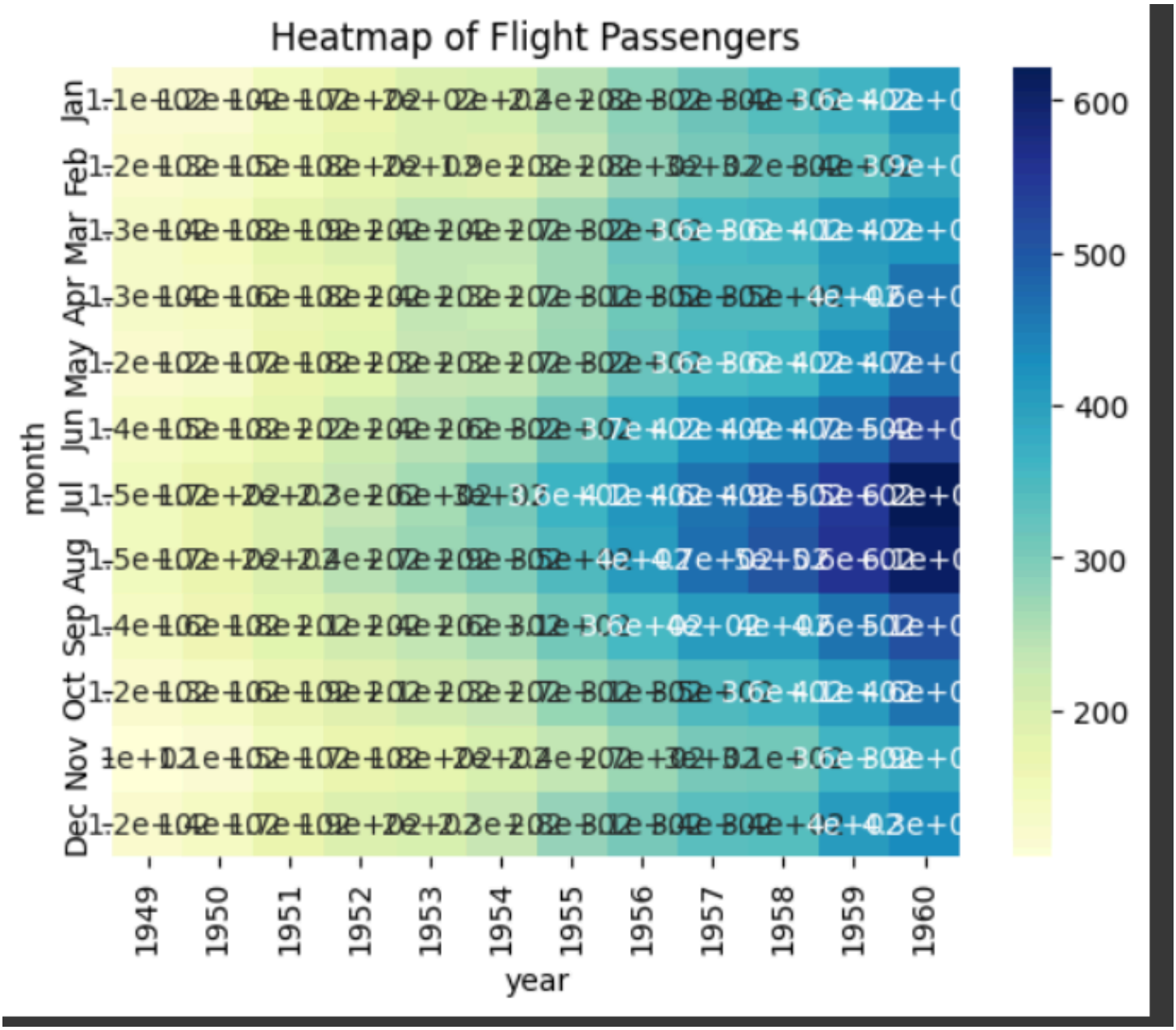
```
sns.violinplot(x="day", y="total_bill", data=tips)
plt.title("Violin Plot")
plt.show()
```




## 7. Heatmap

**Description: Uses color to show correlation or matrix values.**
**Use Case: Correlation matrix, flight frequency by month/year.**

```
flights = sns.load_dataset("flights").pivot(index="month",columns= "year", values="passengers")
sns.heatmap(flights, cmap="YlGnBu", annot=True)
plt.title("Heatmap")
plt.show()
```

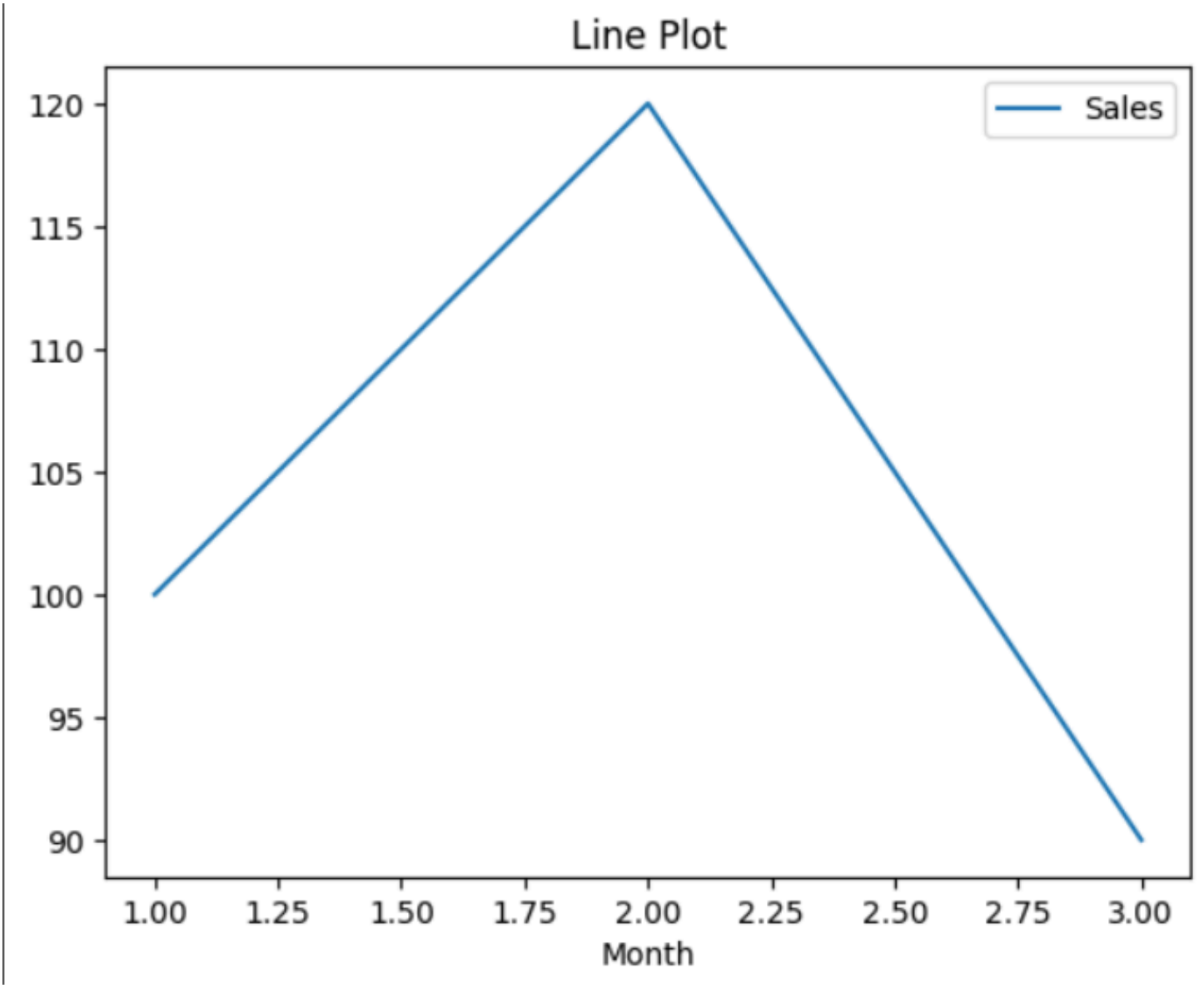



Pandas

## 1. Line Plot

**Description: Default plot in Pandas for continuous data.**
**Use Case: Sales over months, population growth.**

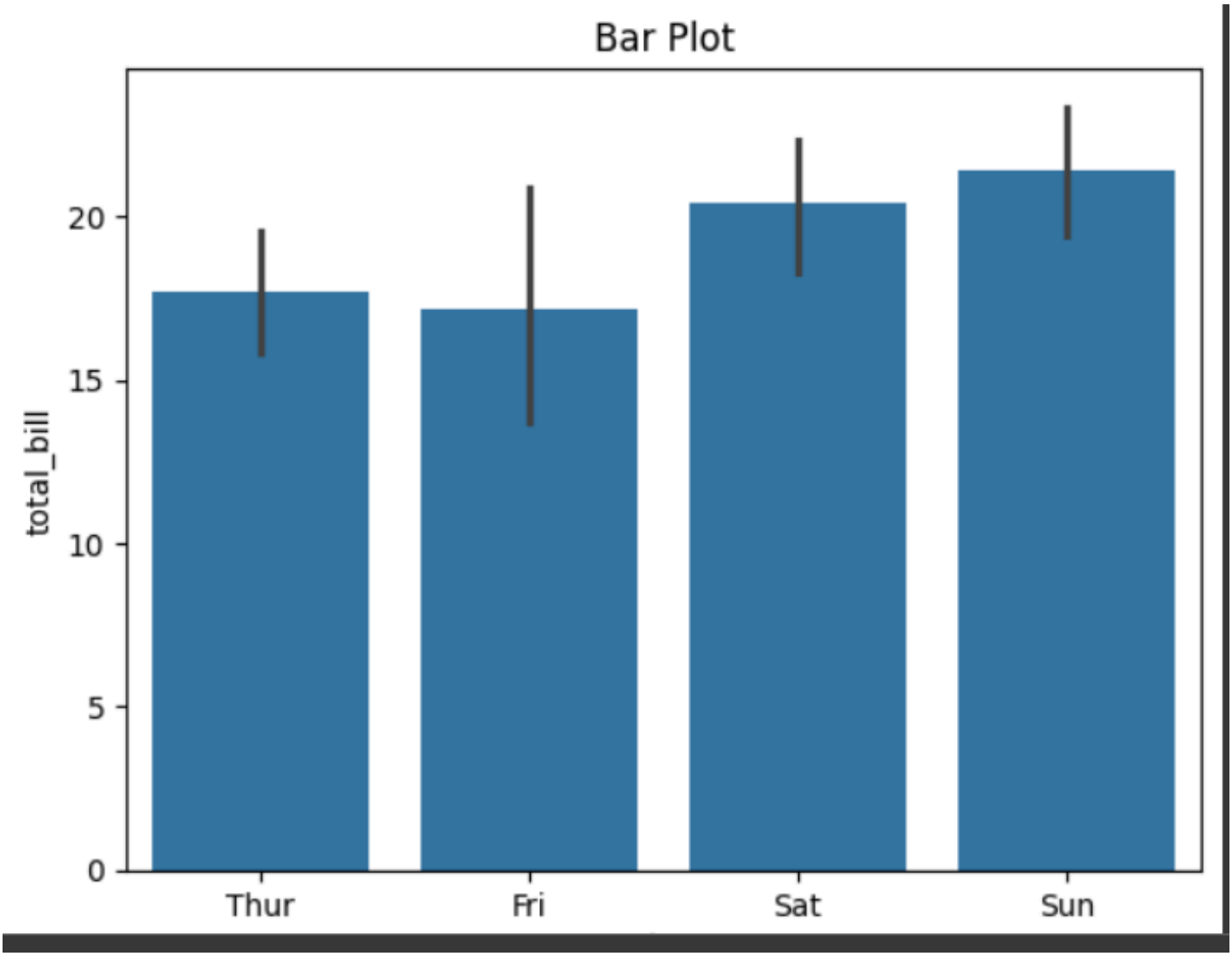
```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.DataFrame({'Month': [1, 2, 3], 'Sales': [100, 120, 90]})
df.plot(x='Month', y='Sales')
plt.title("Line Plot")
plt.show()
```




## 2. Bar Plot

**Description: Represents categorical data with rectangular bars.**
**Use Case: Revenue by region, count of categories.**

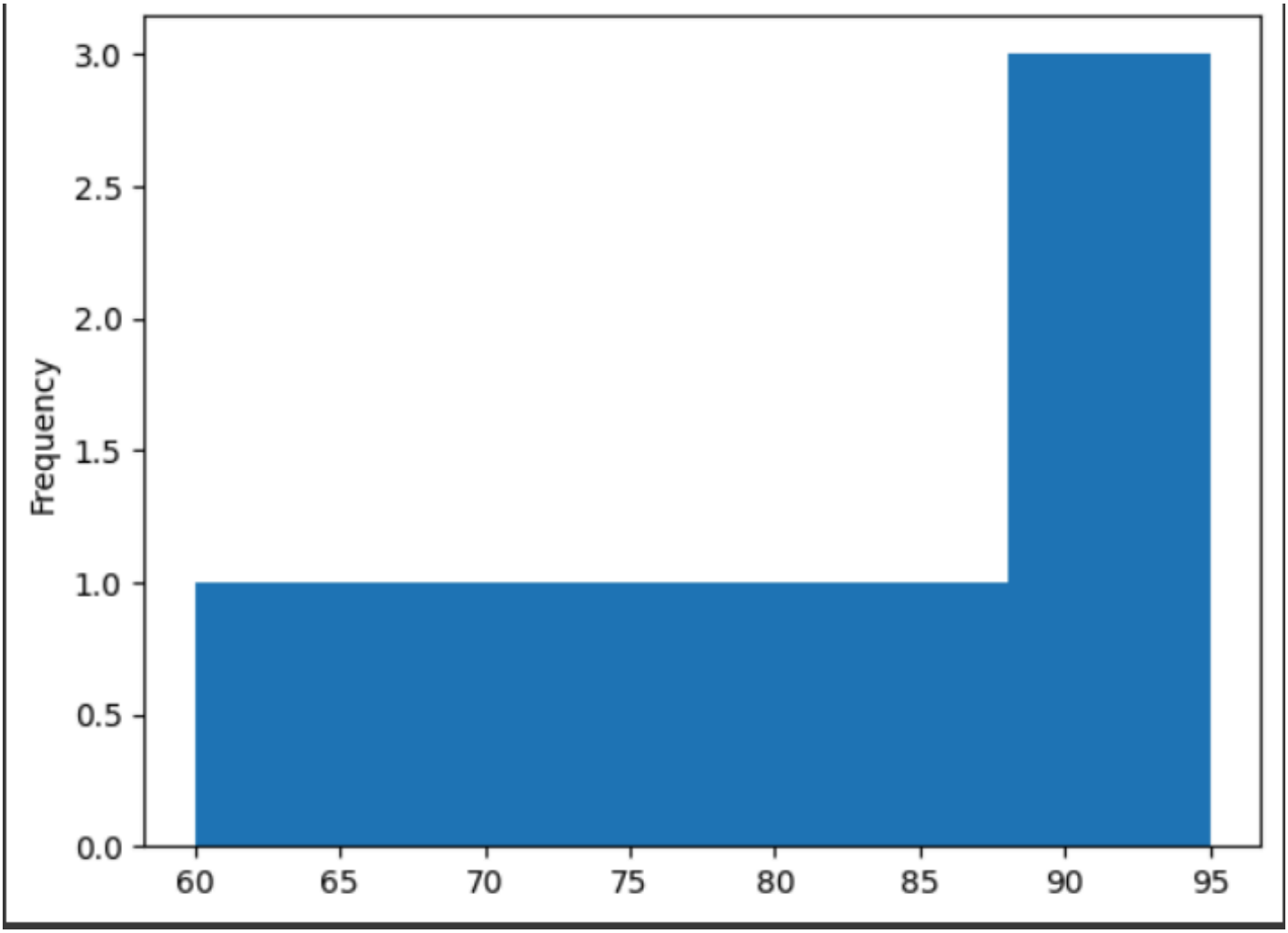
```
sns.barplot(x="day", y="total_bill", data=tips)
plt.title("Bar Plot")
plt.show()
```




### 3. Histogram

**Description: Frequency distribution of a numerical column.**
**Use Case: Distribution of test scores, salaries.**

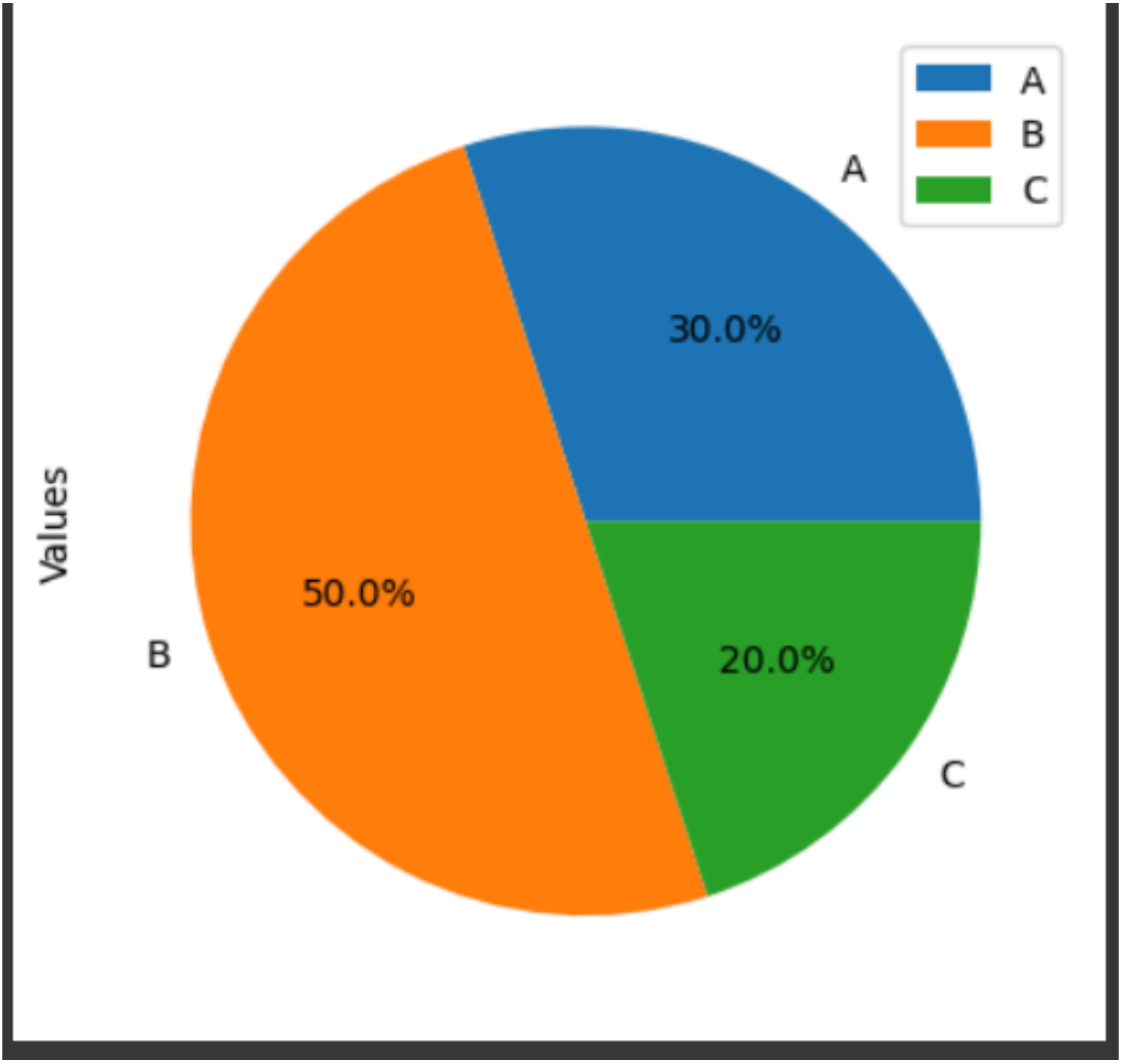
```
import pandas as pd

df = pd.DataFrame({'Score': [70, 85, 90, 75, 60, 95, 88]})
df['Score'].plot(kind='hist', bins=5)
plt.title("Histogram")
plt.show()
```




## 4. Pie Chart

**Description: Circular chart divided into slices to show proportions.**
**Use Case: Market share, budget allocation.**

```
import pandas as pd

df = pd.DataFrame({'Category': ['A', 'B', 'C'], 'Values': [30, 50, 20]})
df.set_index('Category').plot(kind='pie', y='Values', autopct='%1.1f%%')
plt.title("Pie Chart")
plt.ylabel("")
plt.show()
```




COMPARISON BETWEEN LIBRARIES

| Feature | Seaborn | Pandas Visualization |
|---|---|---|
| **Purpose** | Specialized statistical and attractive visualizations | Quick, basic plotting from DataFrames |
| **Ease of Use** | Easy for statistical and complex plots | Very easy for simple plots |
| **Customization** | Moderate (extendable via Matplotlib) | Limited (requires switching to Matplotlib for more) |
| **DataFrame Support** | Excellent, works directly with Pandas DataFrames | Native (since it’s part of Pandas) |
| **Statistical Plots** | Extensive (violin, box, KDE, pairplot, etc.) | Very limited |
| **Built-in Themes** | Yes, multiple style options | No |
| **Interactivity** | Static by default, can integrate with interactive backends | Static plots only |
| **Best Use Case** | EDA, statistical analysis, presentation-ready visuals | Quick data exploration, simple charts |
| **Learning Curve** | Gentle for basic usage, some learning for advanced plots | Minimal for basic plots |
| **Underlying Library** | Matplotlib | Matplotlib |