MATPLOTLIB

Matplotlib is the foundational plotting library in Python, often used for creating static, interactive, and animated visualizations. It provides a high level of control over every aspect of a figure.

The Pyplot is a sub library of matplotlib where all the utilities lie under. It has different types of plots including bar graphs, scatter plots, pie charts, histograms, area charts.

```
#How to import the library
import matplotlib.pyplot as plt
```

By using this library we can visualize the data and we can plot the data on different types of graphs.

The different types of graphs are:

1.Line Plot

2.Bar Chart

3.Histogram

4.Pie Chart

5.Scatter Plot

6.Box Plot

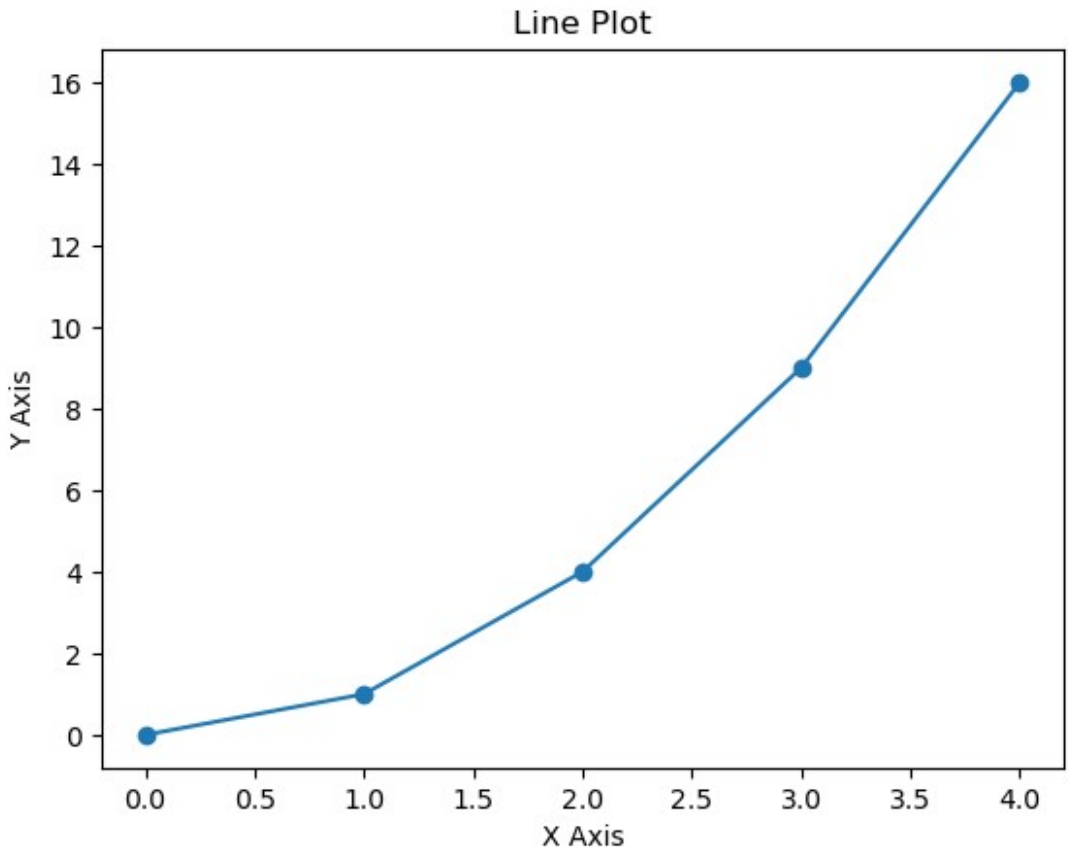
7.Stack Plot

#LINE PLOT

Definition: A line plot is used to represent data points connected by straight lines, typically to show trends over time or ordered data.

```
x = [0, 1, 2, 3, 4]
y = [0, 1, 4, 9, 16]

plt.plot(x, y, marker='o', label='y = x²')
plt.title('Line Plot')
plt.xlabel('X Axis')
plt.ylabel('Y Axis')
plt.show()
```

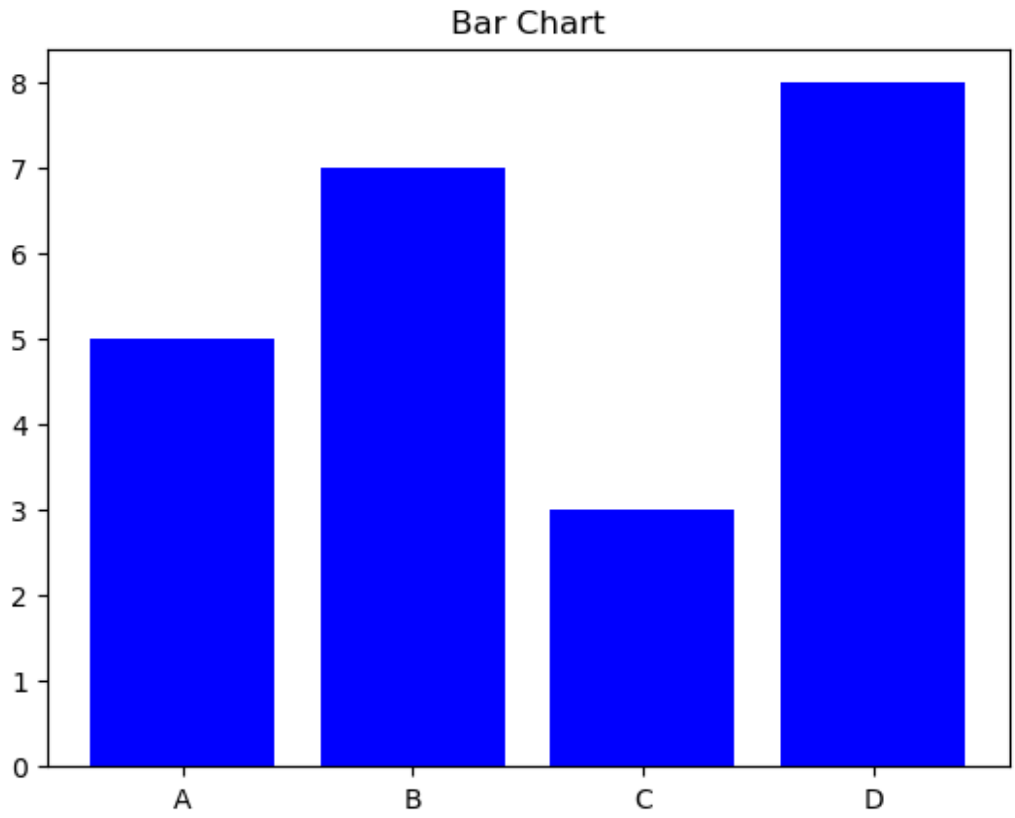


#BAR CHART

Definition: A bar chart displays categorical data with rectangular bars, where the height or length of each bar corresponds to the value.

```
categories = ['A', 'B', 'C', 'D']
values = [5, 7, 3, 8]

plt.bar(categories, values, color='blue')
plt.title('Bar Chart')
plt.show()
```



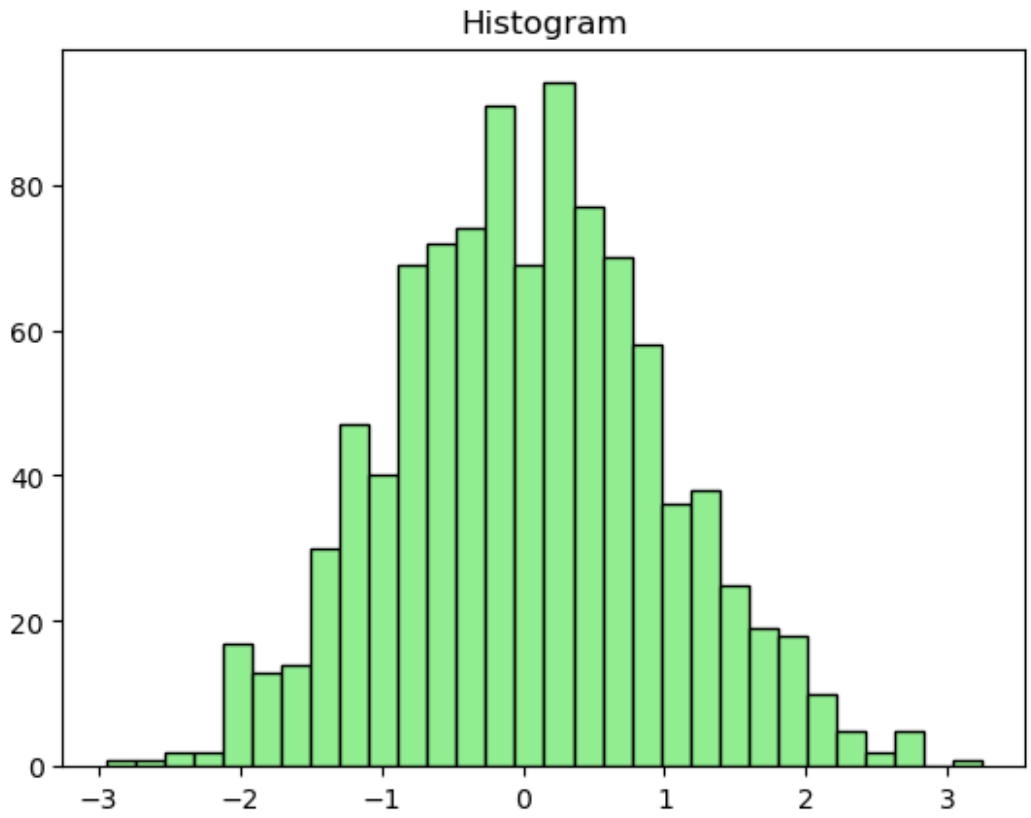
#HISTOGRAM

Definition: A histogram represents the distribution of numerical data by grouping it into bins. It is useful for visualizing the frequency of data ranges.

```
import numpy as np

data = np.random.randn(1000)

plt.hist(data, bins=30, color='lightgreen', edgecolor='black')
plt.title('Histogram')
plt.show()
```

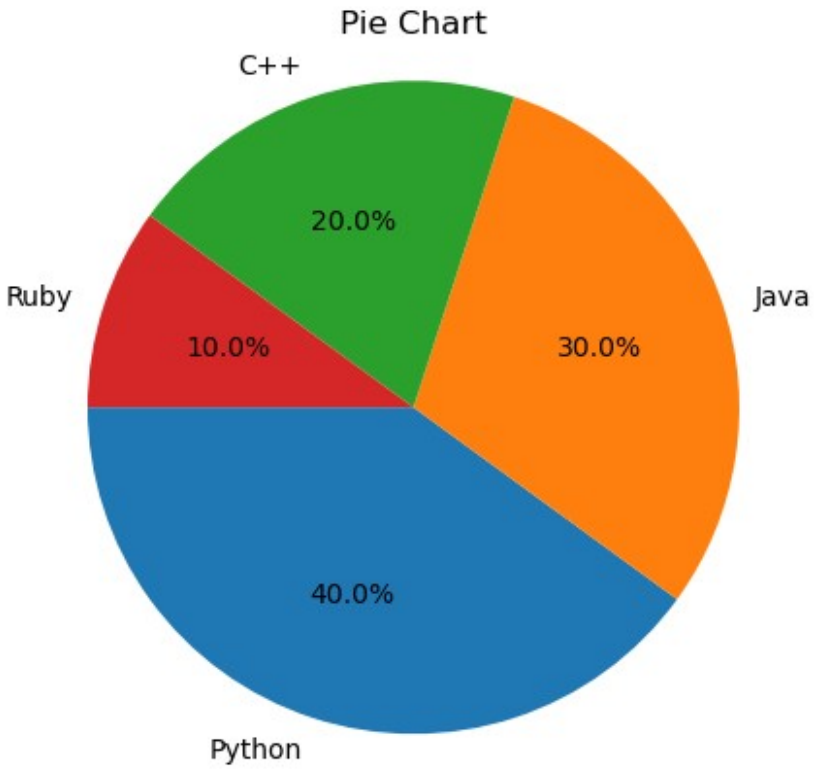


#PIE-CHART

Definition: A circular chart divided into slices to illustrate numerical proportions. Best used for showing percentage or part-to-whole relationships.

```
labels = ['Python', 'Java', 'C++', 'Ruby']
sizes = [40, 30, 20, 10]

plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=180)
plt.title('Pie Chart')
plt.axis('equal')
plt.show()
```

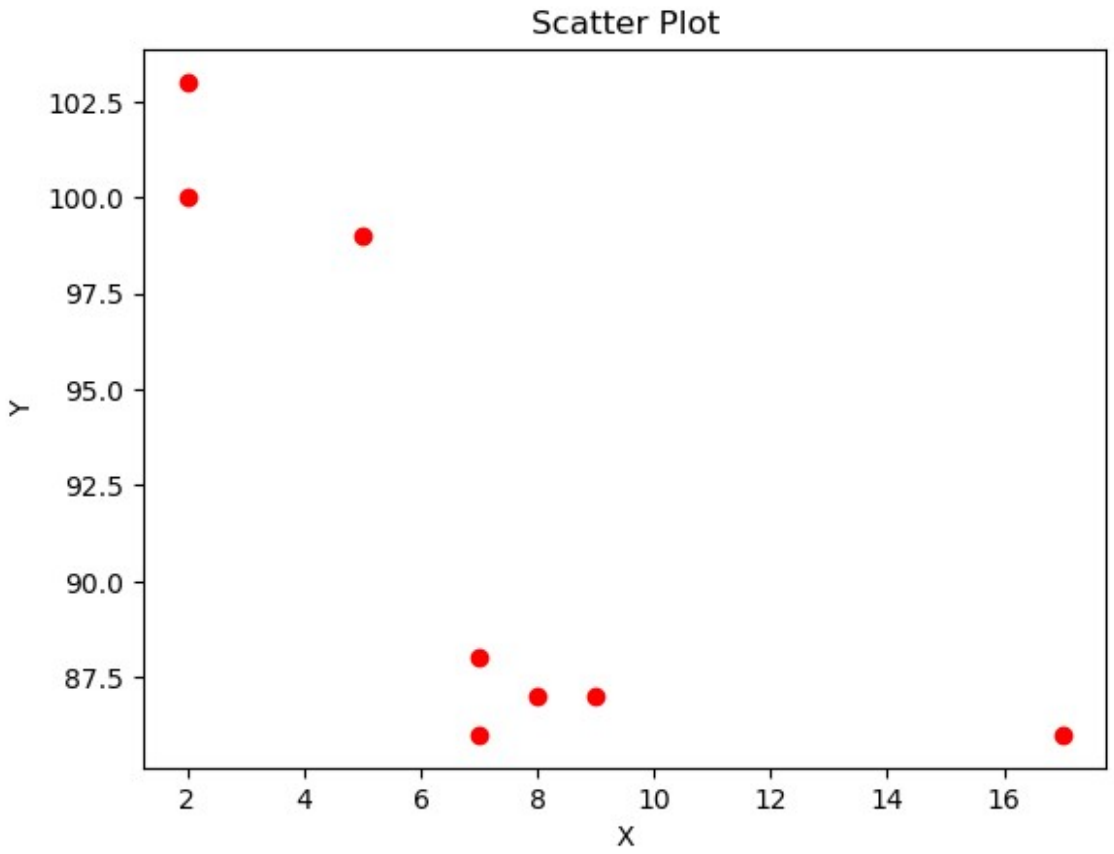


#SCATTER PLOT

Definition: A scatter plot displays values for two variables as a collection of points. It's ideal for showing relationships or correlations.

```
x = [5, 7, 8, 7, 2, 17, 2, 9]
y = [99, 86, 87, 88, 100, 86, 103, 87]

plt.scatter(x, y, color='red')
plt.title('Scatter Plot')
plt.xlabel('X')
plt.ylabel('Y')
plt.show()
```

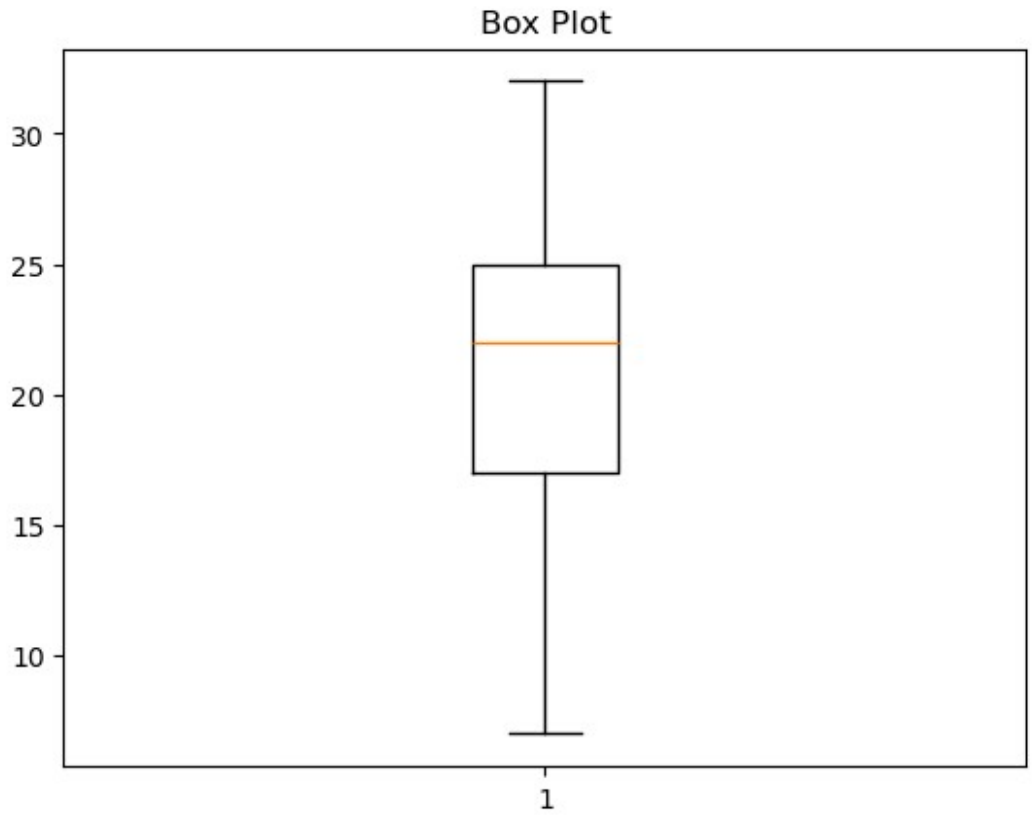


#BOX PLOT

Definition: A box plot (or box-and-whisker plot) summarizes the distribution of a dataset, highlighting the median, quartiles, and outliers.

```
data = [7, 15, 13, 19, 21, 22, 24, 25, 25, 30, 32]

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

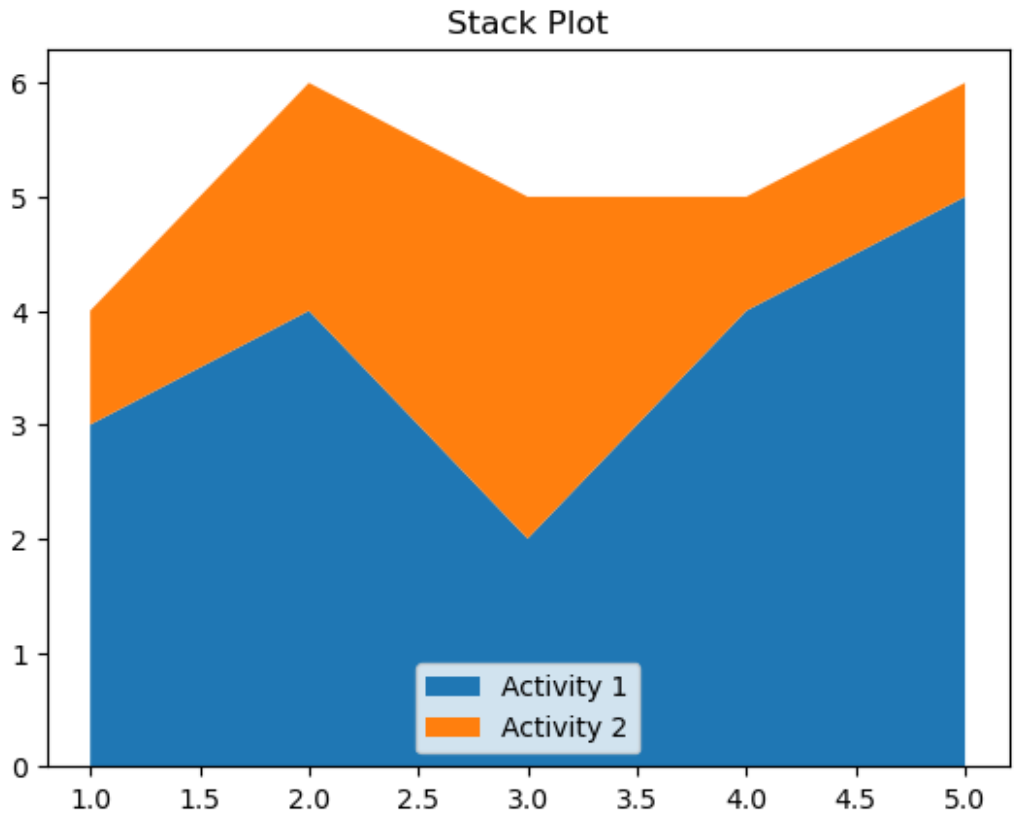


#STACK PLOT

Definition: A stack plot shows different data series stacked on top of one another. It's ideal for representing cumulative totals over time.

```
days = [1, 2, 3, 4, 5]
activity1 = [3, 4, 2, 4, 5]
activity2 = [1, 2, 3, 1, 1]

plt.stackplot(days, activity1, activity2, labels=['Activity 1',
'Activity 2'])
plt.title('Stack Plot')
plt.legend()
plt.show()
```



SEABORN

Seaborn is a Python data visualization library built on top of Matplotlib. It provides a high-level interface for creating informative and aesthetically pleasing statistical graphics. Seaborn excels at visualizing relationships between multiple variables, handling complex datasets, and offering attractive default styles. It is commonly used in data science and machine learning for exploratory data analysis and data presentation.

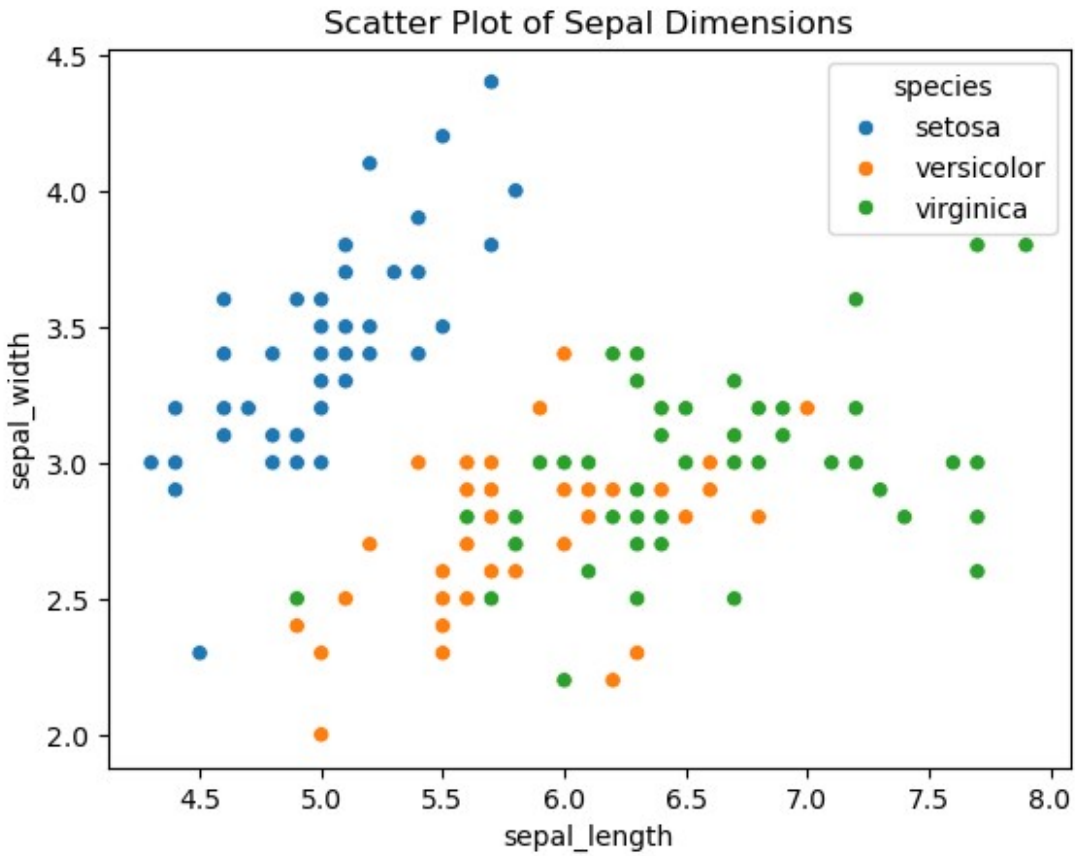
```
#how to import the library
import seaborn as sns
import pandas as pd

#For plotting graphs we are going to use built-in datasets like:
tips = sns.load_dataset("tips")
iris = sns.load_dataset("iris")
```

#SCATTER PLOT

Definition: Shows the relationship between two numerical variables using points.

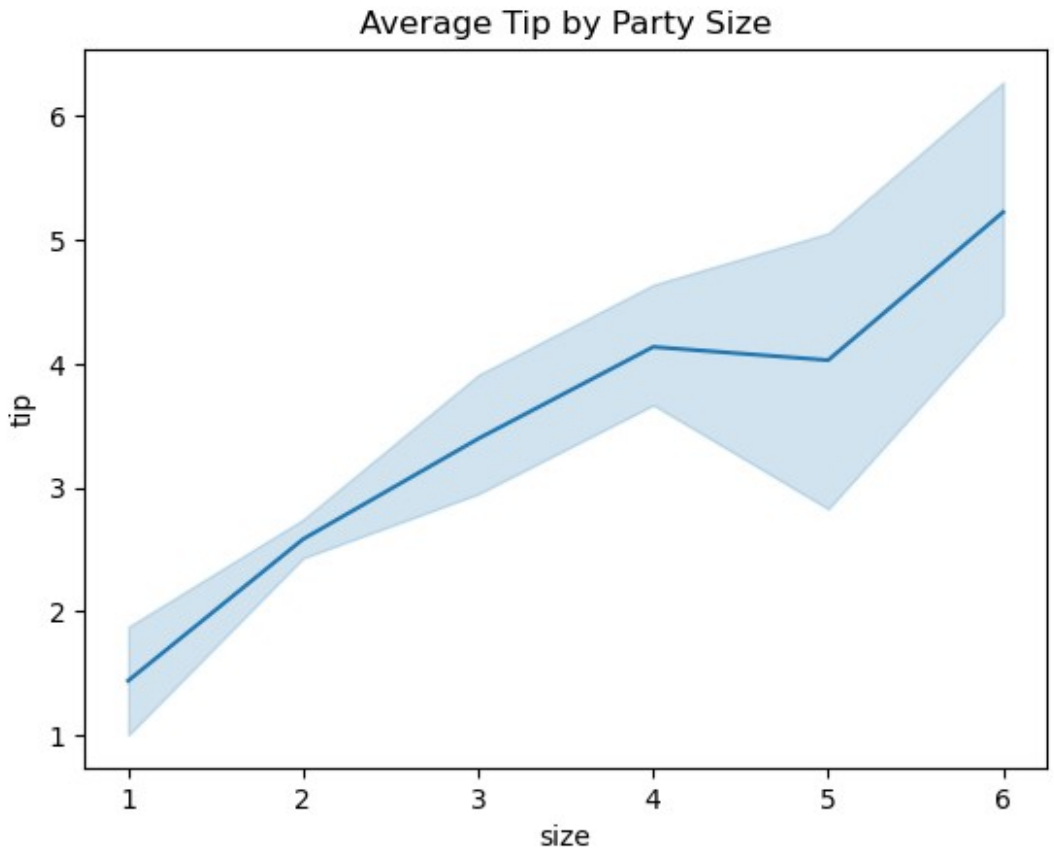
```
sns.scatterplot(data=iris, x='sepal_length', y='sepal_width',
hue='species')
plt.title('Scatter Plot of Sepal Dimensions')
plt.show()
```



#LINE PLOT

Definition: Displays data points connected by lines, often used to show trends.

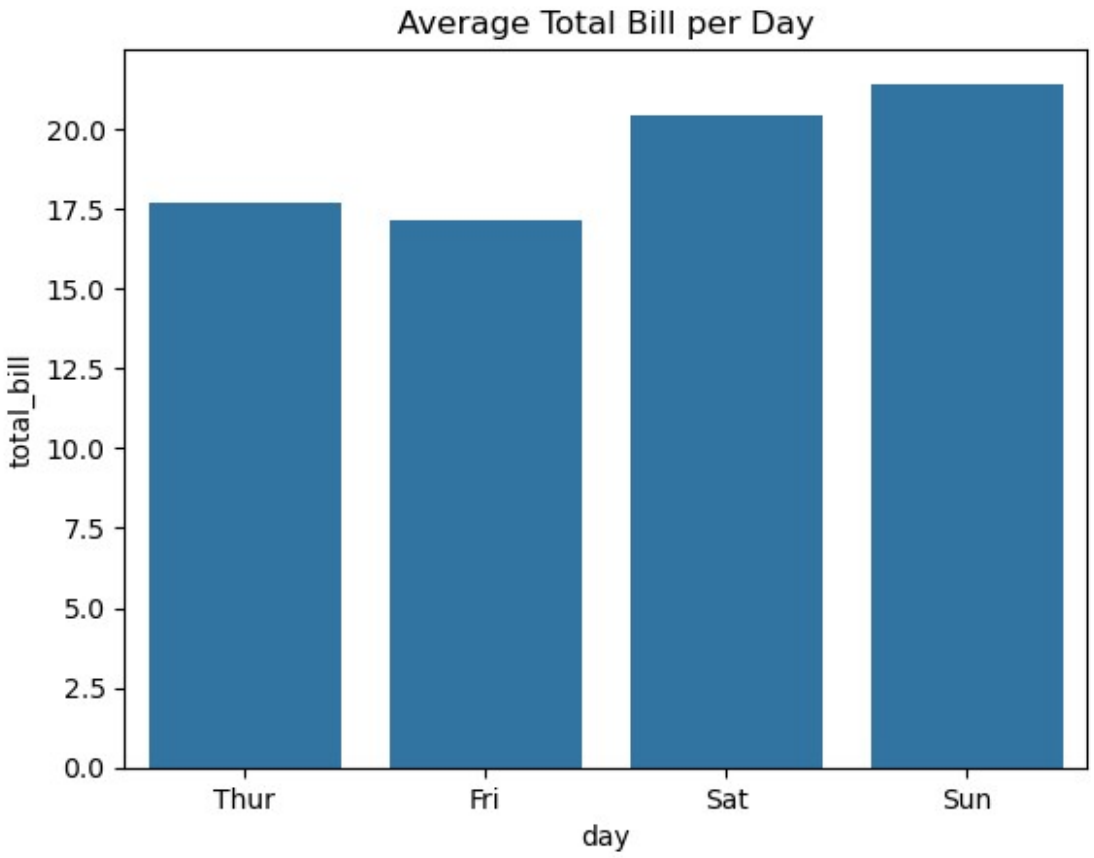
```
sns.lineplot(data=tips, x='size', y='tip', estimator='mean')
plt.title('Average Tip by Party Size')
plt.show()
```



#BAR PLOT

Definition: Shows the average (or other estimator) of a categorical variable.

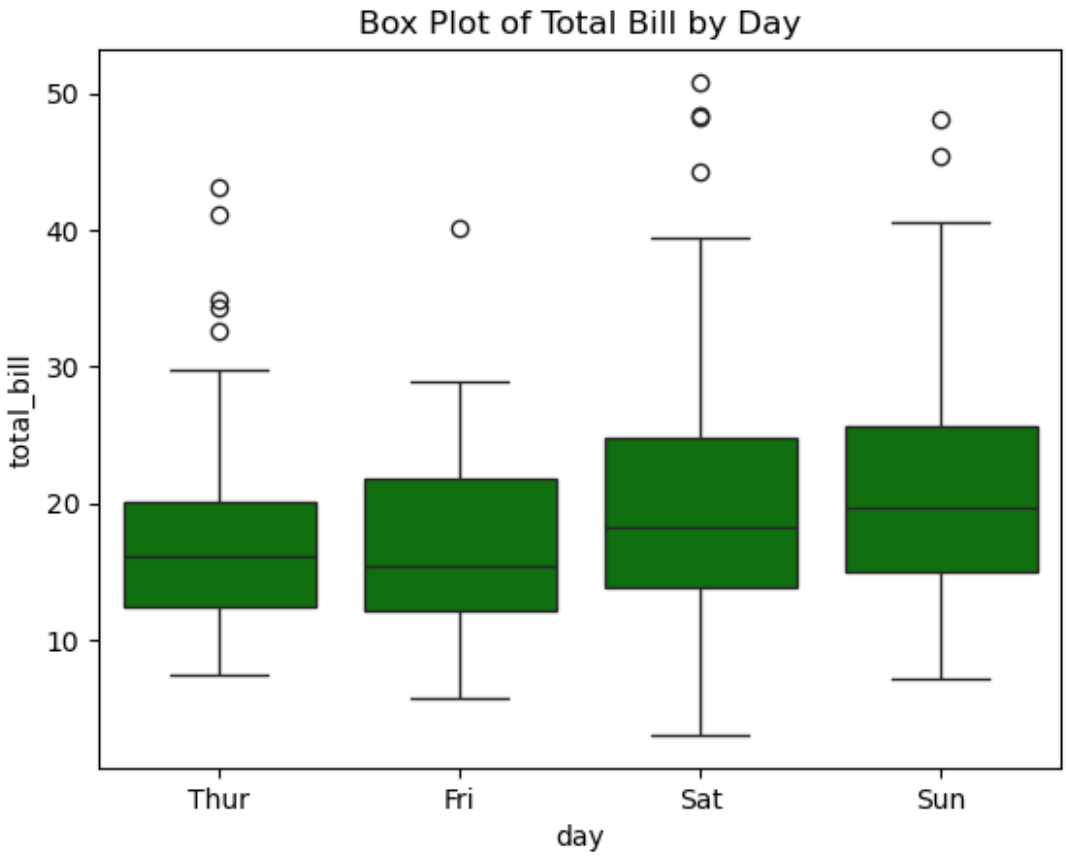
```
sns.barplot(data=tips, x='day', y='total_bill', errorbar=None)
plt.title('Average Total Bill per Day')
plt.show()
```



#BOX PLOT

Definition: Shows the distribution of data based on five summary statistics: min, Q1, median, Q3, and max.

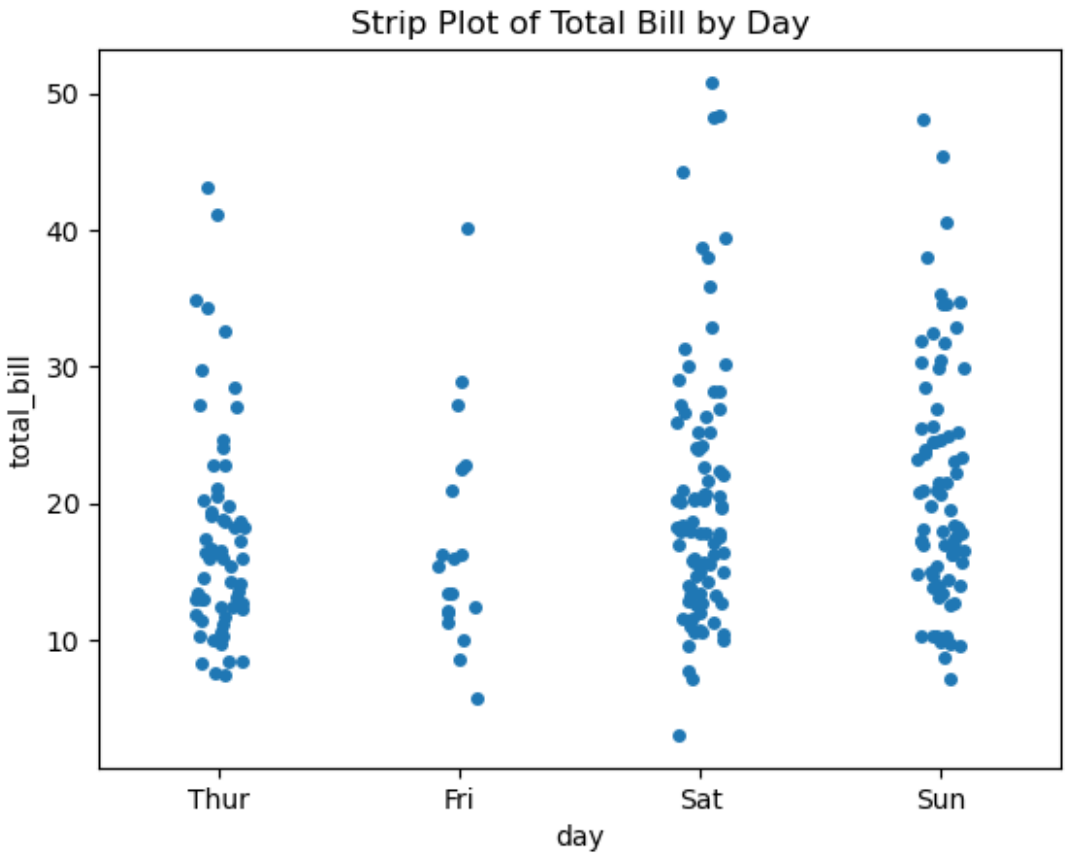
```
sns.boxplot(data=tips, x='day', y='total_bill',color='green')
plt.title('Box Plot of Total Bill by Day')
plt.show()
```



#STRIP PLOT

Definition: Displays individual observations for small datasets.

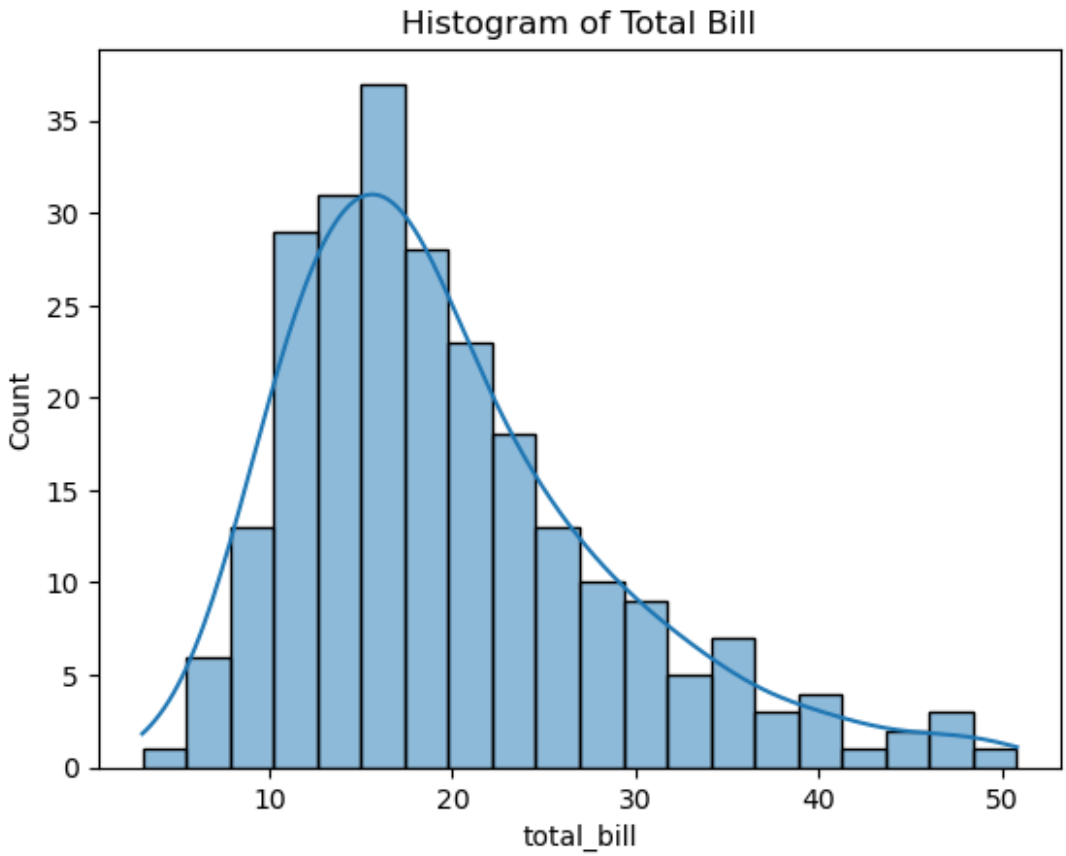
```
sns.stripplot(data=tips, x='day', y='total_bill', jitter=True)
plt.title('Strip Plot of Total Bill by Day')
plt.show()
```



#HISTOGRAM

Definition: Displays the distribution of a single variable by dividing into bins.

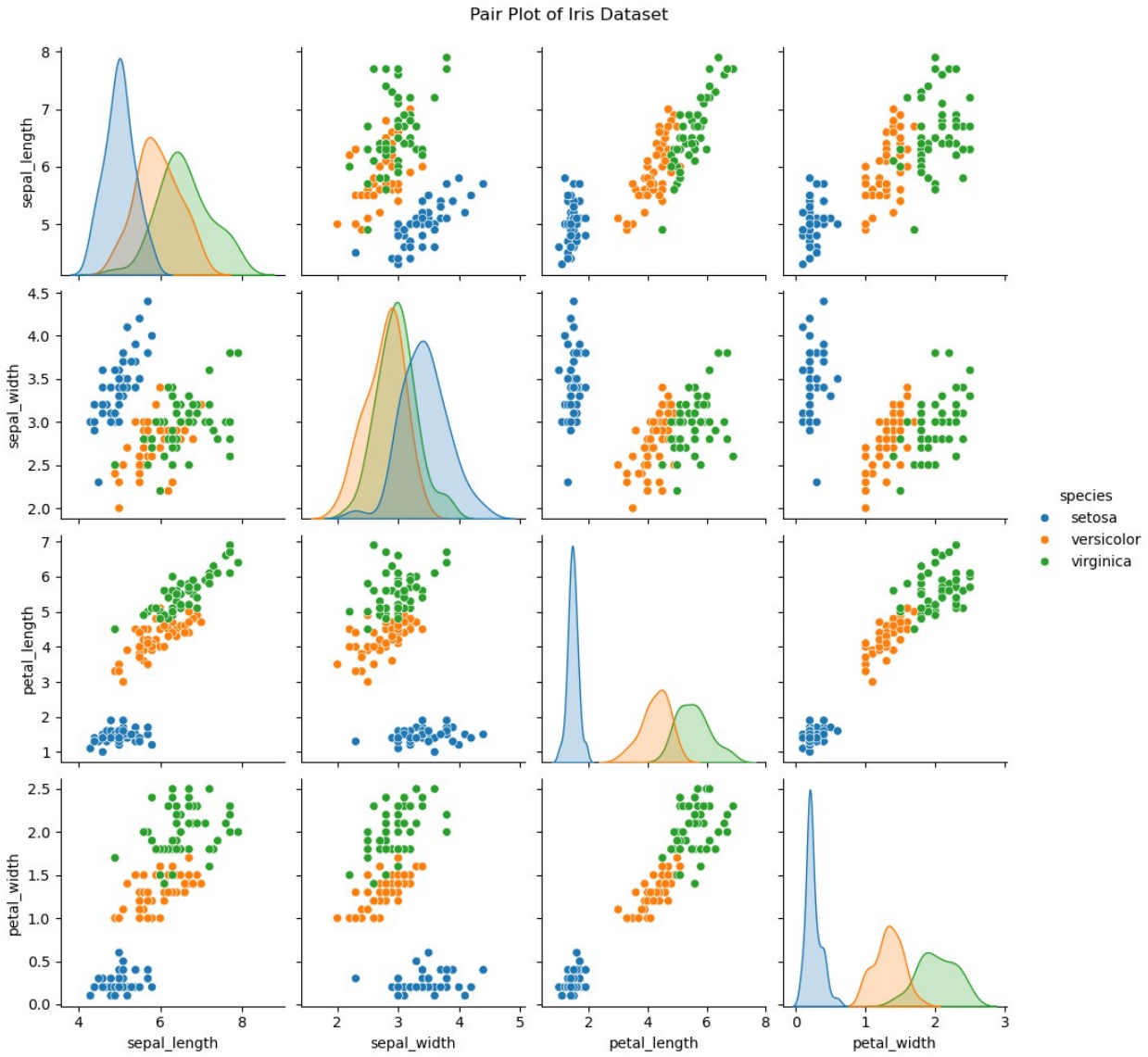
```
sns.histplot(data=tips, x='total_bill', bins=20, kde=True)
plt.title('Histogram of Total Bill')
plt.show()
```



#PAIR PLOT

Definition: Creates a matrix of scatter plots to visualize pairwise relationships in a dataset.

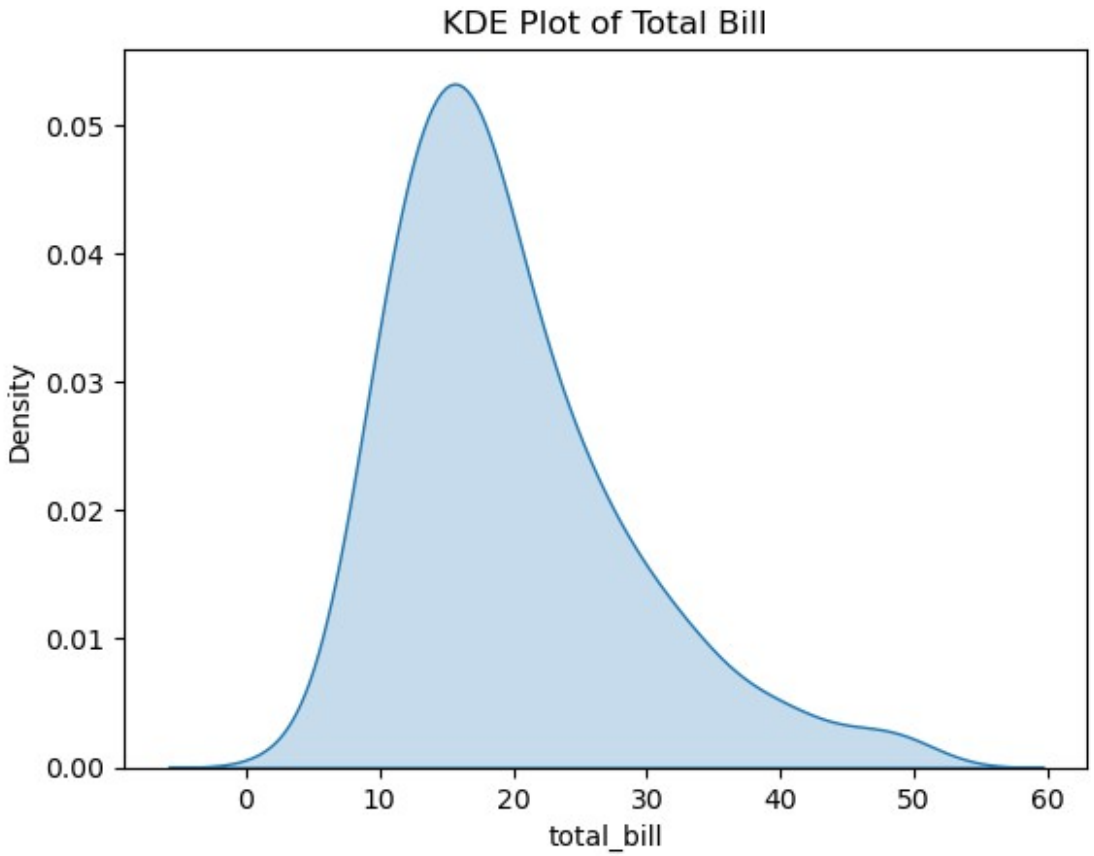
```
sns.pairplot(data=iris, hue='species')
plt.suptitle('Pair Plot of Iris Dataset', y=1.02)
plt.show()
```



#KDE PLOT

Definition: A smooth curve (Kernel Density Estimate) to visualize the probability density function of a variable.

```
sns.kdeplot(data=tips['total_bill'], fill=True)
plt.title('KDE Plot of Total Bill')
plt.show()
```



Aspect	Matplotlib	Seaborn
Type of Library	A low-level plotting tool where you build everything manually	A high-level tool built on top of Matplotlib for easier, prettier plots
Main Purpose	Great for making fully customized and detailed plots	Best for quick and beautiful statistical plots
Ease of Use	Harder to learn, requires more lines of code	Easier to use, with simpler and shorter code
Customization	Very flexible – you can tweak almost every part of a plot	Offers good control, but not as detailed as Matplotlib
Visual Style	Looks basic unless you customize it	Nice-looking plots out of the box
Working With Data	Handles lists, arrays, and NumPy arrays	Works directly with Pandas DataFrames (perfect for data analysis)
Stats Features	No built-in stats; you calculate everything yourself	Built-in features like average lines, confidence intervals, and KDE
Default Themes	Simple and plain look by default	Comes with attractive themes and color palettes
Compatibility	Standard plotting tool used in almost all Python setups	Depends on Matplotlib underneath to generate plots
Types of Charts	Line, bar, pie, scatter, etc. – covers basics	Adds advanced charts like violin, box, heatmaps, and pair plots

Aspect	Matplotlib	Seaborn
Complex Plots	Perfect for highly custom or complex visualizations	Best for quick data exploration and analytics
Speed/ Performance	Slightly faster when making basic plots	May be slower due to added features
Learning Difficulty	Takes time to master	Great for beginners
Interactivity	Needs extra tools to add interactivity	Mostly static plots – no built-in interactivity
3D Plot Support	Yes – can make 3D plots using special toolkit	No native support for 3D plots
Animations	Yes – supports animated plots	Limited – depends on Matplotlib to animate