Data Science Internship

Beginner Level : Visualization Library Documentation

Python Visualization :

Python visualization is the process of the representing data in graphical or visual formats such as charts, graphs, and plots using python libraries.

It helps in analyzing, understanding, and interpreting data patterns more easily the viewing raw numbers.

Libraries chosen : matplotlib and seaborn

Matplotlib : It is one of the most widely used and foundational python libraries for data visualization. It provides fine-grained control over every aspect of figure, making it ideal for creating both simple and highly customized static plots. Introduced in 2002 by John Hunter.

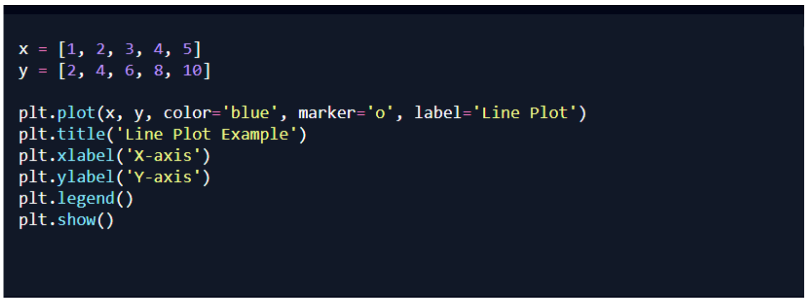
Pyplot is a sub library of matplotlib where all the utilites lie under and has types of plots including line, bar, scatter, pie, histogram charts.

Some of the plots in matplotlib

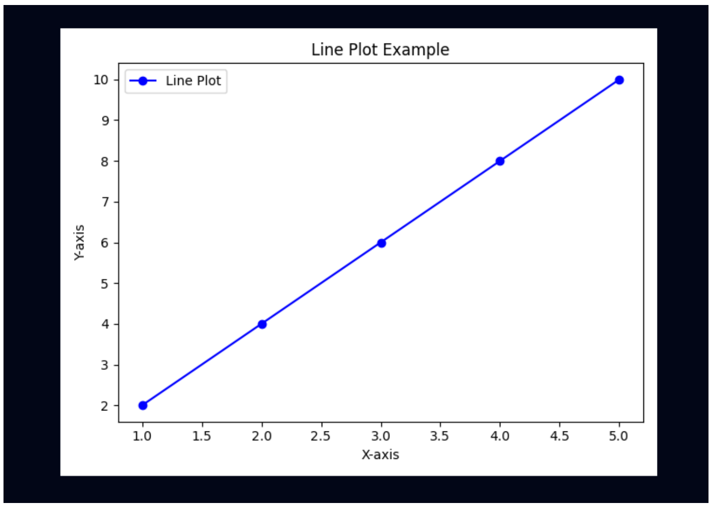
1.Line plot: Dispalys information as a series of data points connected by straight lines.

Use: Visualizing trends over time.

Code :



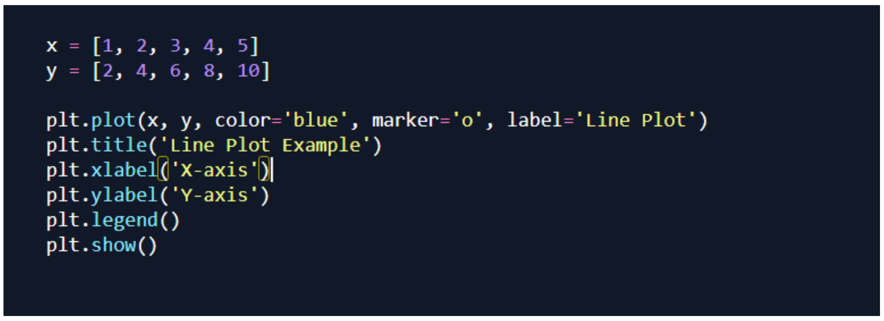
```python
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]

plt.plot(x, y, color='blue', marker='o', label='Line Plot')
plt.title('Line Plot Example')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.legend()
plt.show()
```



2.Scatter Plot : Displays individual data points on a two-dimensional axis to reveal relationships or correlations.

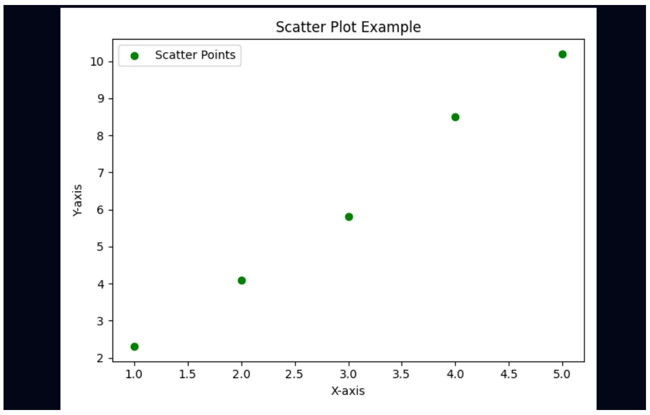
Use: Exploring relationships between two numerical variables(eg., height vs weight)

Code :



```
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]

plt.plot(x, y, color='blue', marker='o', label='Line Plot')
plt.title('Line Plot Example')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.legend()
plt.show()
```

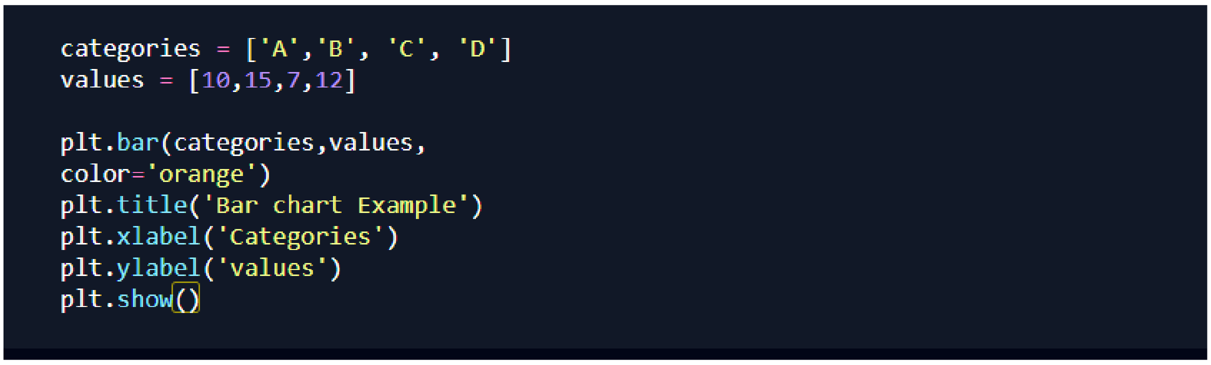
Output:



3.Bar Chart :Represents categorical data with rectangular bars showing values or frequencies.

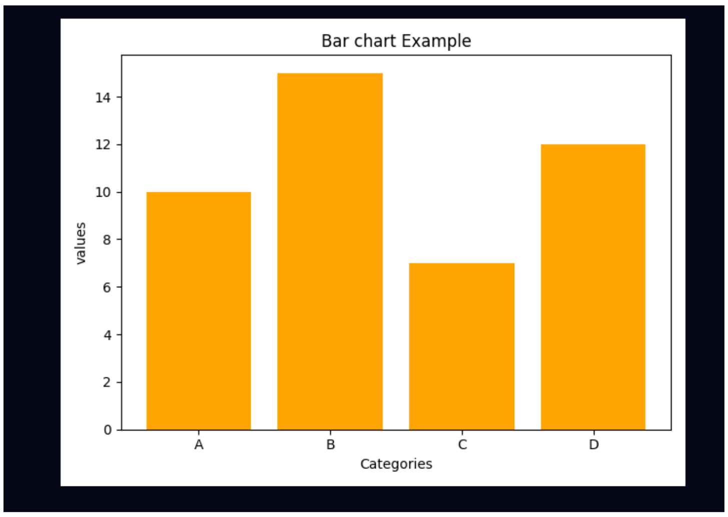
Use : comparing values between different groups(eg., sales by product category).

Code :



```
categories = ['A','B', 'C', 'D']
values = [10,15,7,12]

plt.bar(categories,values,
color='orange')
plt.title('Bar chart Example')
plt.xlabel('Categories')
plt.ylabel('values')
plt.show()
```

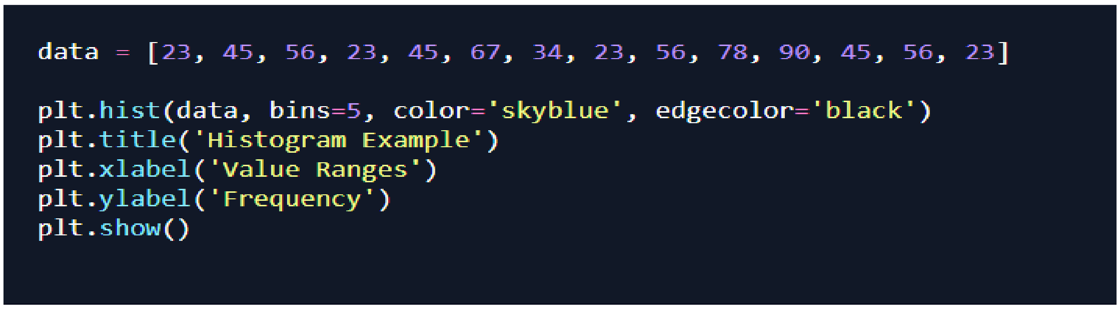
Output:



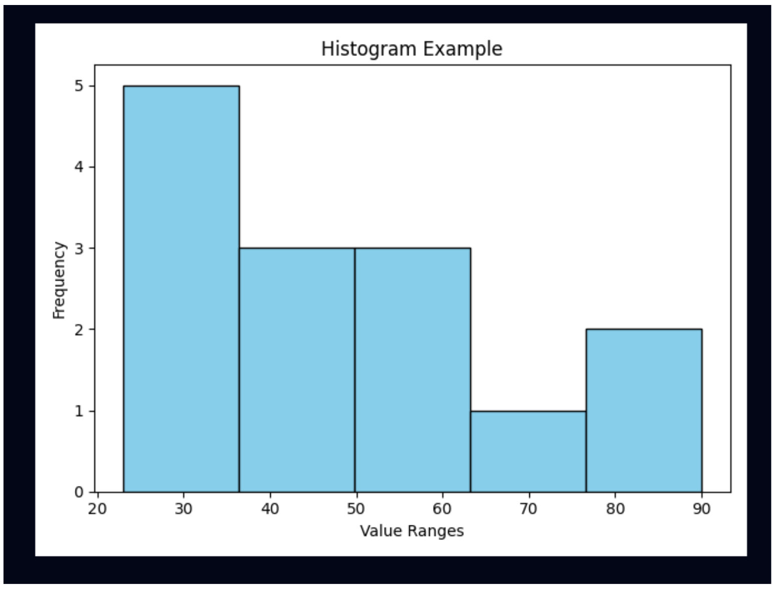
4.Histogram:Dispalys the distribution of numerical data by dividing values into bins

Use: understanding data distribution(eg.,age distribution in a survey).

Code:



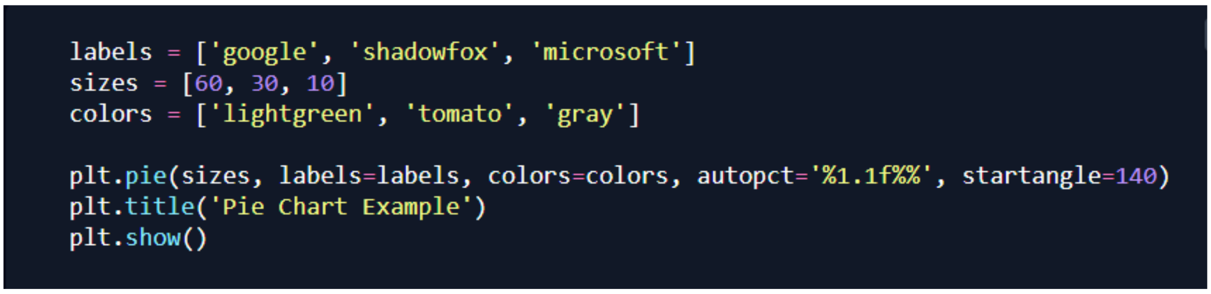
```
data = [23, 45, 56, 23, 45, 67, 34, 23, 56, 78, 90, 45, 56, 23]

plt.hist(data, bins=5, color='skyblue', edgecolor='black')
plt.title('Histogram Example')
plt.xlabel('Value Ranges')
plt.ylabel('Frequency')
plt.show()
```



5.pie chart : Represents categorical data as slices of a circle.showing the proportion of each catrgory.

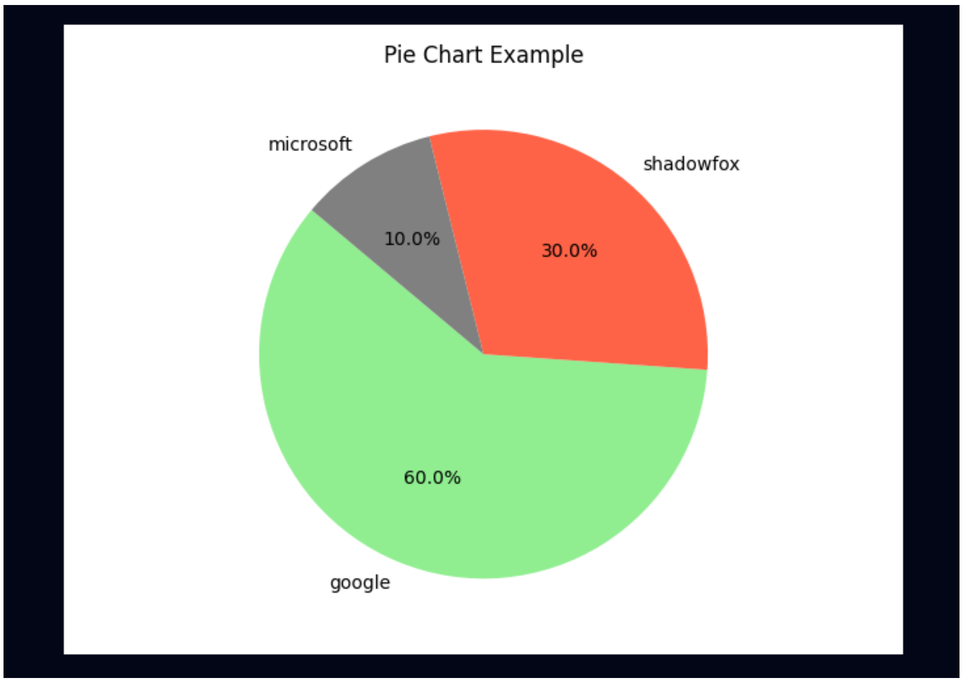
Use: Showing percentage composition(eg.,market share of companies).

Code:



```
labels = ['google', 'shadowfox', 'microsoft']
sizes = [60, 30, 10]
colors = ['lightgreen', 'tomato', 'gray']

plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=140)
plt.title('Pie Chart Example')
plt.show()
```

Output:



Seaborn : It is a python library built on matplotlib that makes it easy to create attractive and informative statistical graphs. It simplifies plotting complex data patterns with minimal code.it was introduced in **2012** by **Michael Waskon.**

Setup: Seaborn is built on top matplotlib and provides beautiful, high-level visualizations with simpler syntax. You must import both Seaborn and matplotlib, and optionally load a dataset to plot.

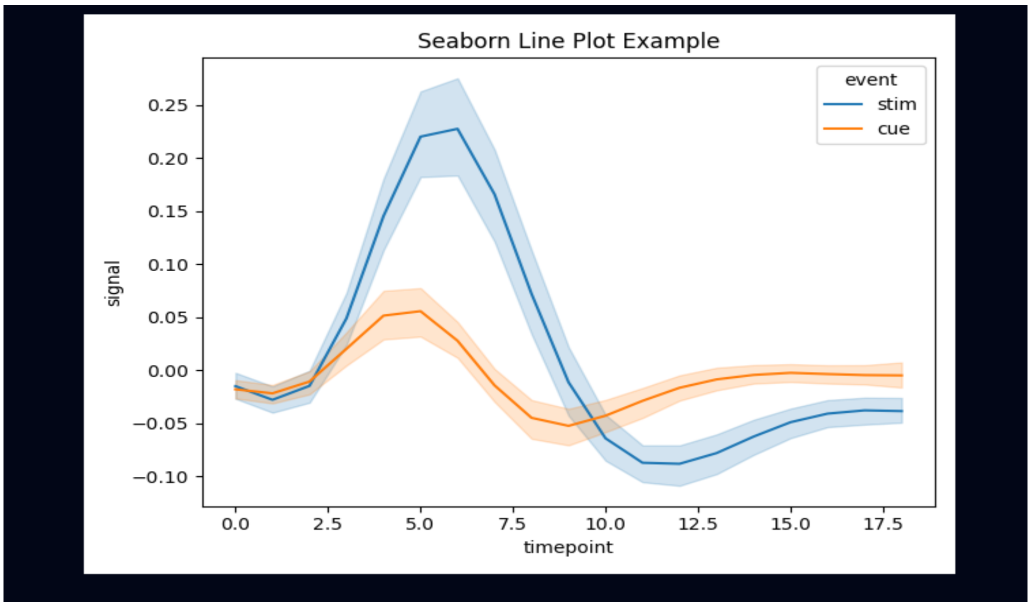
Import seaborn as sns
Import matplotlib.pyplot as plt

Some of the plots in Seaborn

1.Line plot : Shows trends or changes over a continuous variable(like time)

Use: it is useful for analyzing patterns, growth, or decline. It handle multiple lines and add confidence intervals.

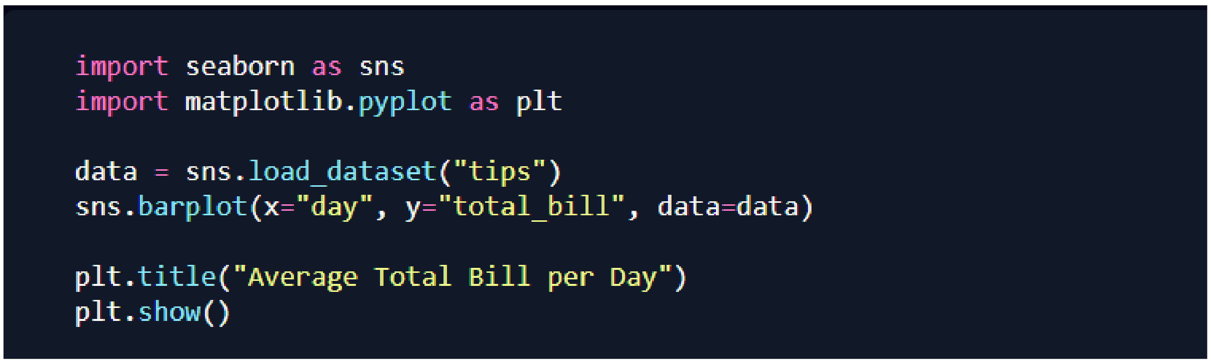
```
import seaborn as sns
import matplotlib.pyplot as plt

# Sample data
data = sns.load_dataset("fmri")

# Create a line plot
sns.lineplot(x="timepoint", y="signal", hue="event", data=data)

plt.title("Seaborn Line Plot Example")
plt.show()
```



2.Bar plot: Shows comparison among different categories. It can automatically calculate mean or other aggregations if there are multiple data points per category.

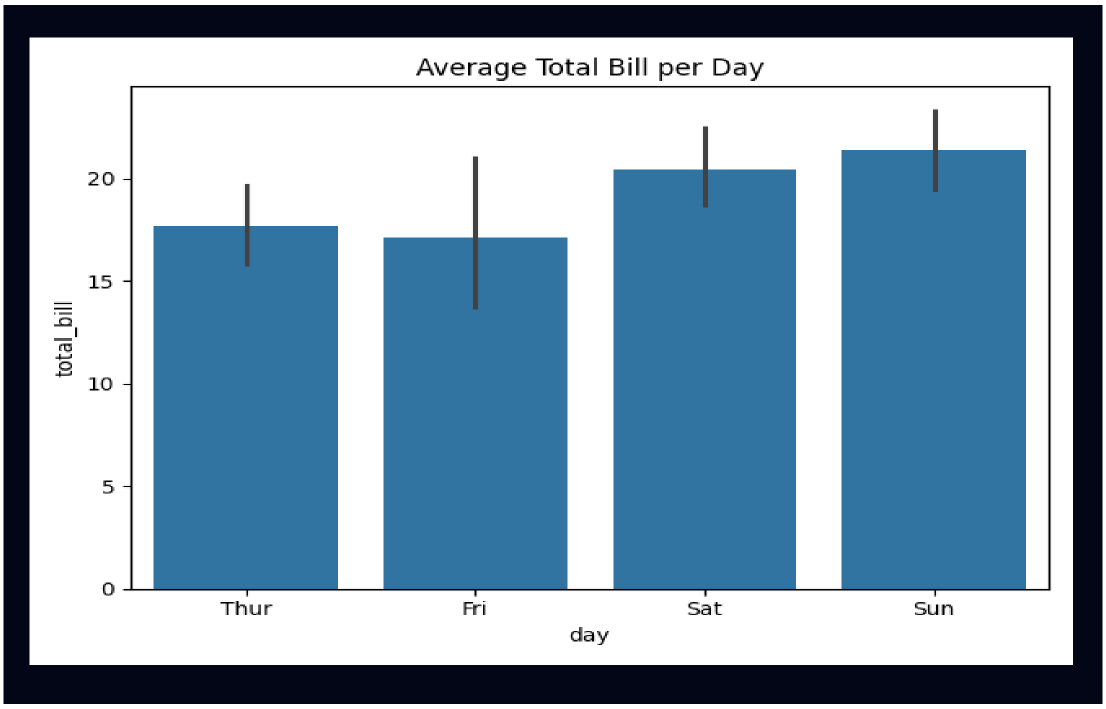
Use: comparing quantities like sales, counts, or averages.

Code:



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

data = sns.load_dataset("tips")
sns.barplot(x="day", y="total_bill", data=data)

plt.title("Average Total Bill per Day")
plt.show()
```

Output:

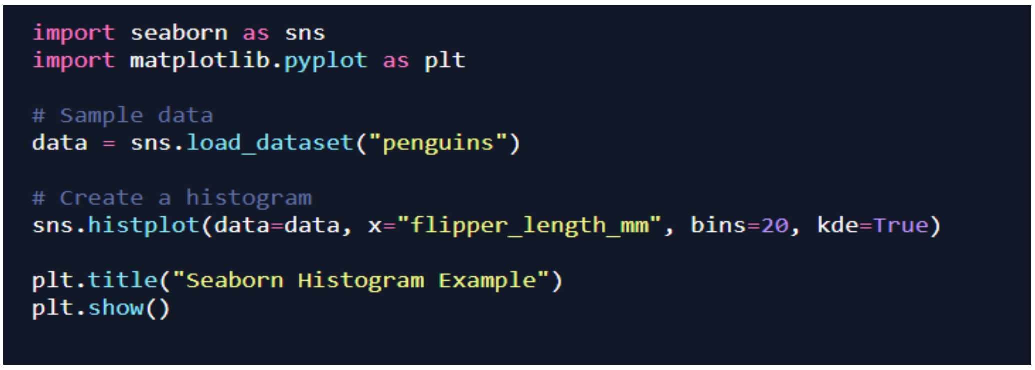


3.Histogram/Distribution Plot : Displays the frequency distribution of a numerical variable.

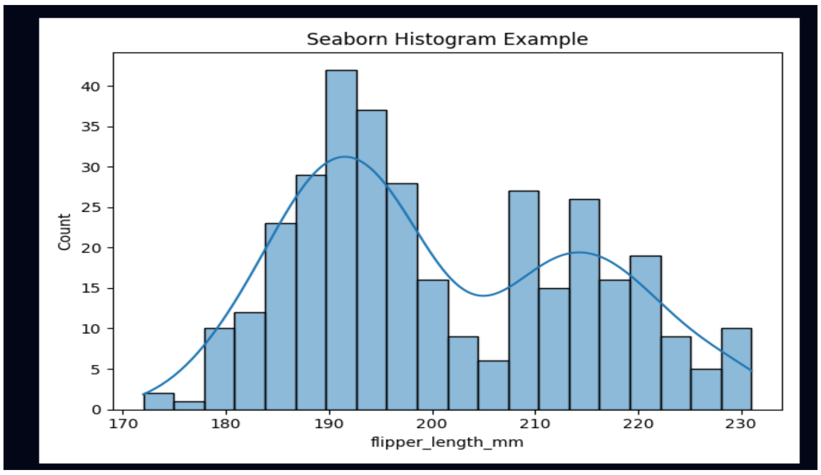
Kde=True adds a smooth density curve over the histogram.

Use: To understand data spread, skewness, and patterns.

Code:


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

# Sample data
data = sns.load_dataset("penguins")

# Create a histogram
sns.histplot(data=data, x="flipper_length_mm", bins=20, kde=True)

plt.title("Seaborn Histogram Example")
plt.show()
```

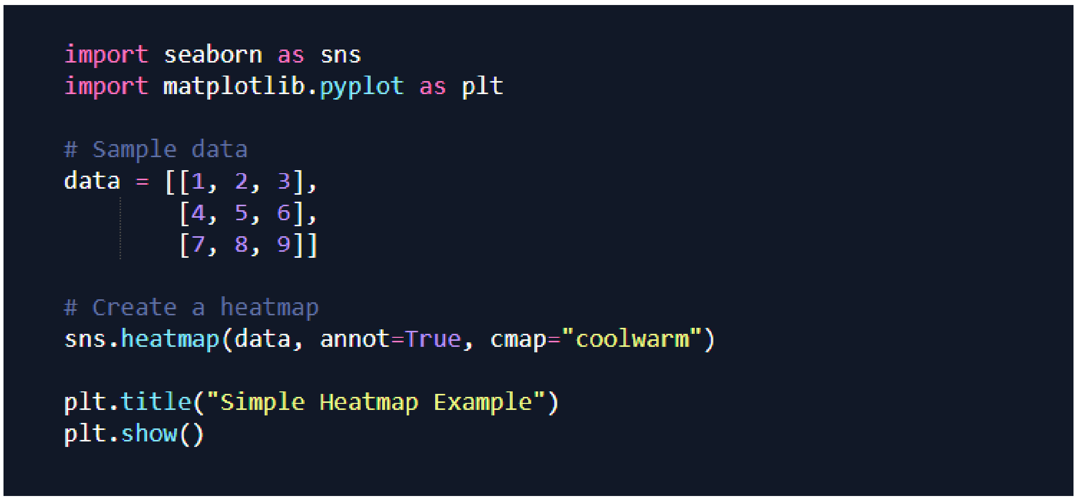
Output:



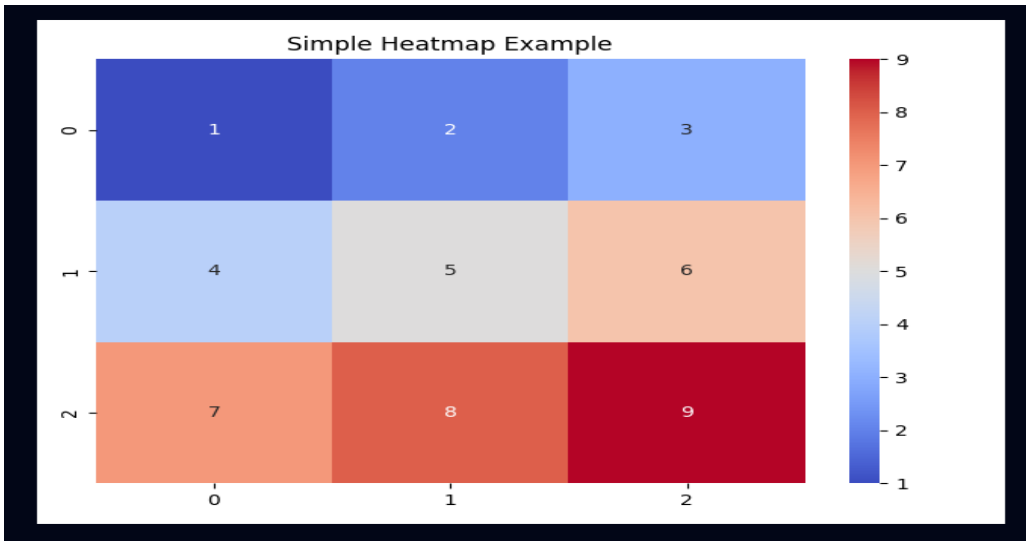
4.Heatmap : Dispalys a matrix of values as colors, often used for correlation matrices.

Use: useful for spotting patterns,tends.

Code:


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

# Sample data
data = [[1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]]

# Create a heatmap
sns.heatmap(data, annot=True, cmap="coolwarm")

plt.title("Simple Heatmap Example")
plt.show()
```

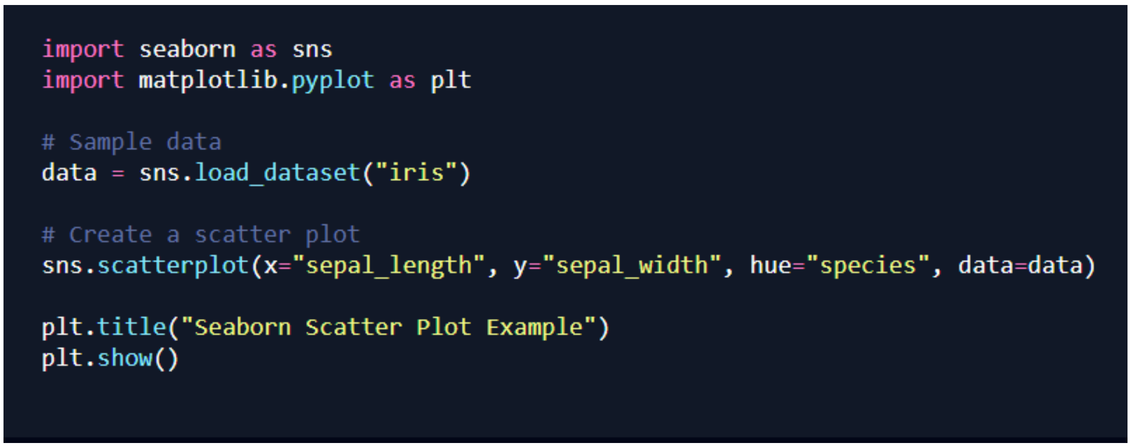
Output:



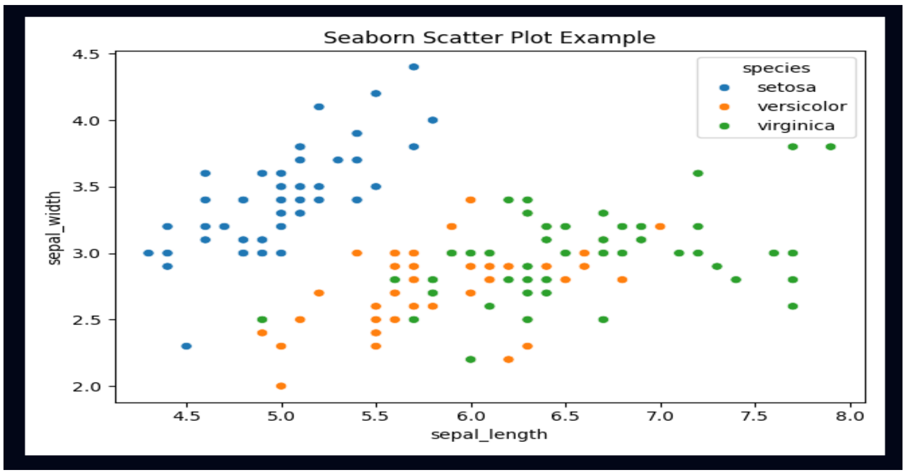
5.Scatter Plot : It shows the relationship between two numerical variables by displaying points on a 2D plane.

Use: To identify patterns, trends, or correlations in data.

Code:



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

# Sample data
data = sns.load_dataset("iris")

# Create a scatter plot
sns.scatterplot(x="sepal_length", y="sepal_width", hue="species", data=data)

plt.title("Seaborn Scatter Plot Example")
plt.show()
```



COMPARISION OF MATPLOTLIB AND SEABORN

Matplotlib :

- It is type of Low-level plotting library.
- Primary use of matplotlib is it completely controls over plot design.
- Import : import matplotlib.pyplot as plt
- **Flexibility** : It is extremely flexible - you can customize every element of the plot.
- **Integration** : It works with NumPy, and SciPy directly.
- It supports all standard plots(lines, bar, pie, scatter, etc.)
- **Weaknesses :** It requires more code for simple plots. Default style is plain: needs manual styling.
- It can be overwhelming for beginners.
- **Performance** : Faster and more efficient for very large datasets.
- **Memory use** : More optimized for rendering large amounts of points.
- **Scalability** : Suitable for high-volume data and custom optimization.
- Ideal for fundamental charts(line, bar, scatter).

Seaborn :

- It is type of High-level statistical plotting library(built on Matplotlib)
- Primary use of seaborn is quick and elegant visualization of statistical data.
- Import : import seaborn as sns
- **Integration** : Integrates tightly with Pandas DataFrames --works great with tabular data.
- It supports advanced plots(heatmap, pairplot, violin, jointplot, ect.)
- **Weaknesses** :less flexible for fine-tuned customization.Limited customization once you reach matplotlib-level details.
- **Dependent on matplotlib** – advanced tweaks requires matplotlib knowledge.
- **Performance** :Slightly slower –adds abstraction and statistical computation overhead.
- **Memory use**: Consumes more memory due to automatic aggregation and styling.
- **Scalability**: Suitable for moderate-size datasets; may lag on millions of rows.