Visualization Library Documentation

1. Introduction to Visualization

Visualization is a way of representing data in a graphical or pictorial format for better understanding. Data visualization helps to quickly identify trends, patterns, and outliers in datasets.

In Python, several libraries support data visualization, such as **Matplotlib**, **Seaborn**, **Plotly**, and **Bokeh**.
Here, we will focus on **Matplotlib** and **Seaborn**, two of the most widely used libraries.

2. Matplotlib

Matplotlib, introduced in 2002 by John Hunter, is a powerful library for creating 2D plots. It is built on NumPy and is considered the foundation of Python visualization.

A Matplotlib plot contains four key components:

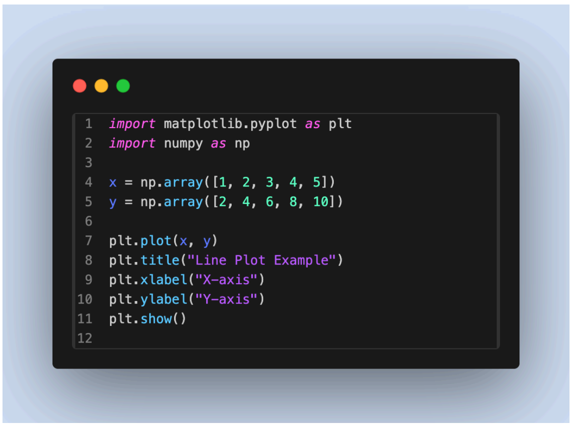
- Figure: The main container for all plots.
- Axes: The plotting area (can have multiple per figure).
- Axis: Handles ticks, labels, and limits.
- Artists: All visible elements (lines, texts, shapes, etc.).

The pyplot sub-library (`import matplotlib.pyplot as plt`) provides easy-to-use functions for common plot types.

Common Plots in Matplotlib

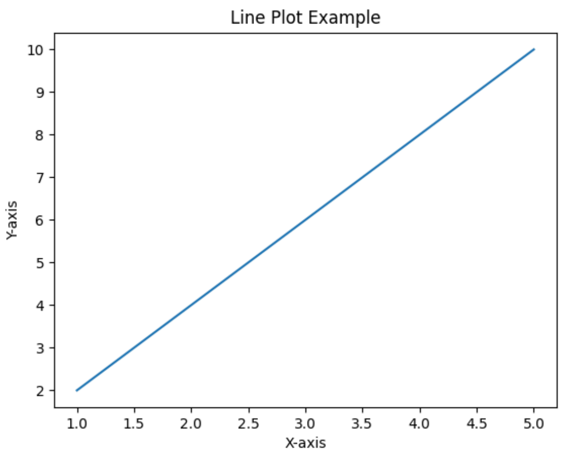
(a) Line Chart

Used to show trends over time or sequential data.



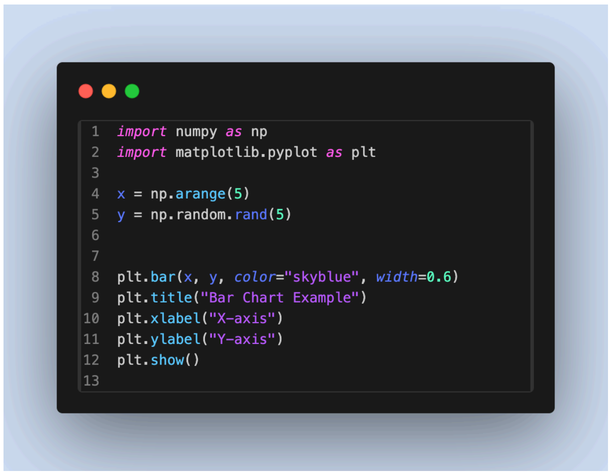
```python
import matplotlib.pyplot as plt
import numpy as np

x = np.array([1, 2, 3, 4, 5])
y = np.array([2, 4, 6, 8, 10])

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

Output:



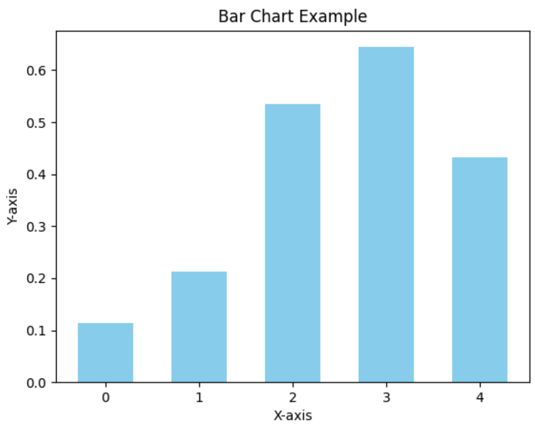
(b) Bar Chart

Used to compare categories.



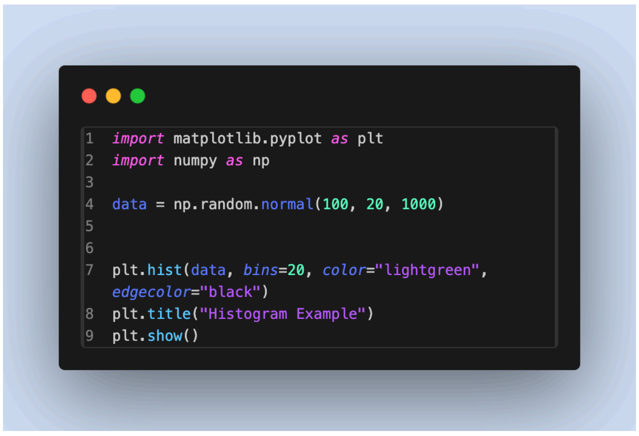
```python
import numpy as np
import matplotlib.pyplot as plt

x = np.arange(5)
y = np.random.rand(5)


plt.bar(x, y, color="skyblue", width=0.6)
plt.title("Bar Chart Example")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()

```

Output:

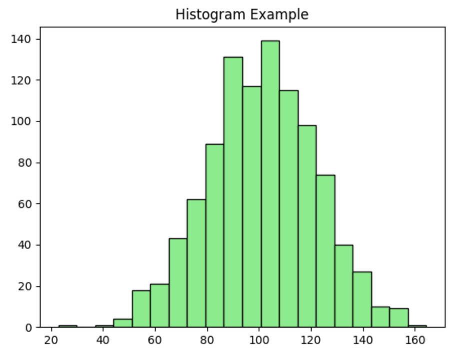


(c) Histogram

Displays the distribution of values.

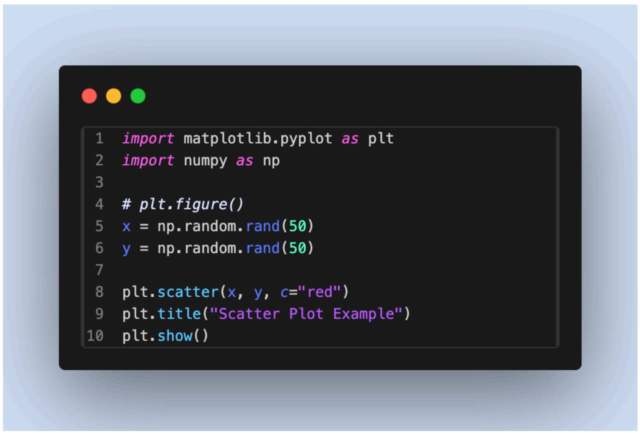


```
import matplotlib.pyplot as plt
import numpy as np

data = np.random.normal(100, 20, 1000)


plt.hist(data, bins=20, color="lightgreen",
edgecolor="black")
plt.title("Histogram Example")
plt.show()
```

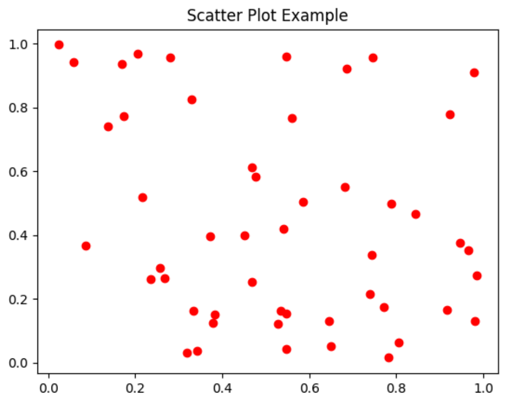


(c) Histogram

Displays the distribution of values.

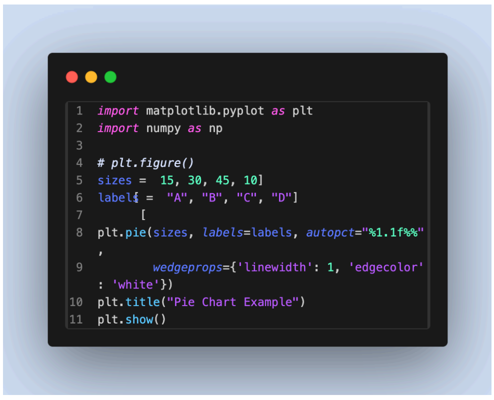


```
import matplotlib.pyplot as plt
import numpy as np

# plt.figure()
x = np.random.rand(50)
y = np.random.rand(50)

plt.scatter(x, y, c="red")
plt.title("Scatter Plot Example")
plt.show()
```

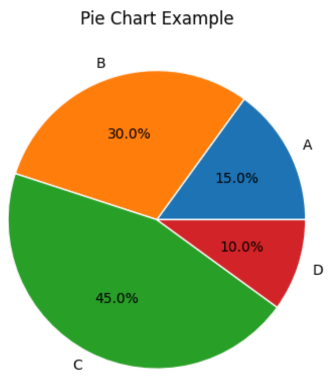


(e) Pie Chart

Represents proportions of a whole.

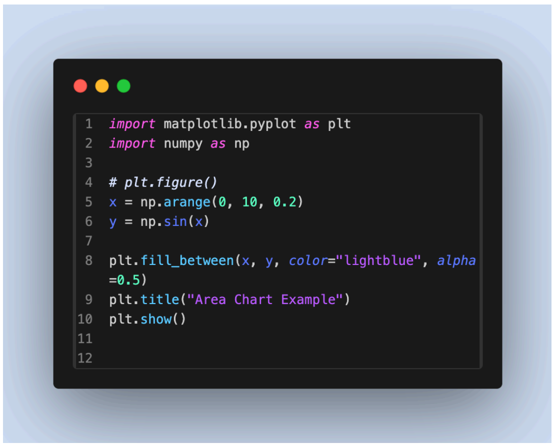


```
import matplotlib.pyplot as plt
import numpy as np

# plt.figure()
sizes =  15, 30, 45, 10]
label$ =  "A", "B", "C", "D"]
      [
plt.pie(sizes, labels=labels, autopct="%1.1f%%"
,
        wedgeprops={'linewidth': 1, 'edgecolor'
: 'white'})
plt.title("Pie Chart Example")
plt.show()
```

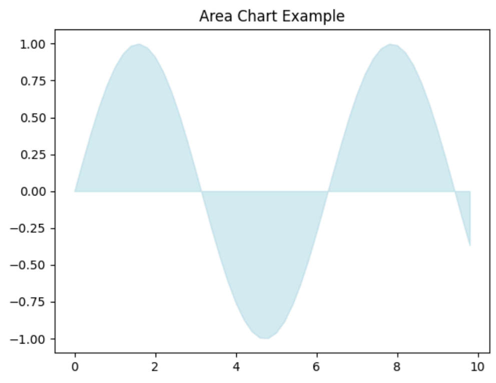


(f) Area Chart

Fills the area under a line plot.



```python
import matplotlib.pyplot as plt
import numpy as np

# plt.figure()
x = np.arange(0, 10, 0.2)
y = np.sin(x)

plt.fill_between(x, y, color="lightblue", alpha
=0.5)
plt.title("Area Chart Example")
plt.show()
```



3. Seaborn

Overview

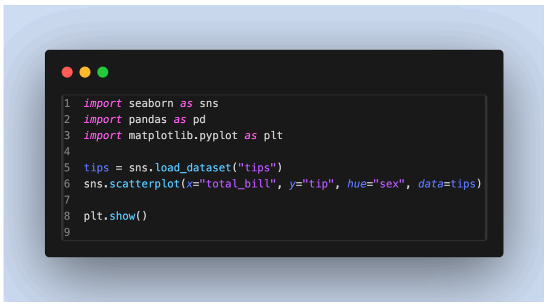
Seaborn is a higher-level visualization library built on top of Matplotlib. It provides beautiful default styles and integrates seamlessly with Pandas DataFrames.

Seaborn simplifies statistical plotting and comes with categories of plots:

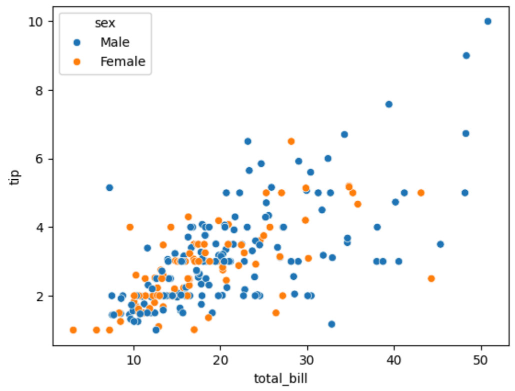
- Relational Plots: `scatterplot()`, `lineplot()`
- Categorical Plots: `barplot()`, `boxplot()`, `violinplot()`
- Distribution Plots: `histplot()`, `kdeplot()`
- Matrix Plots: `heatmap()`, `clustermap()`
- Regression Plots: `regplot()`, `lmplot()`

Common Plots in Seaborn

(a) Scatter Plot – Relationship between two variables.

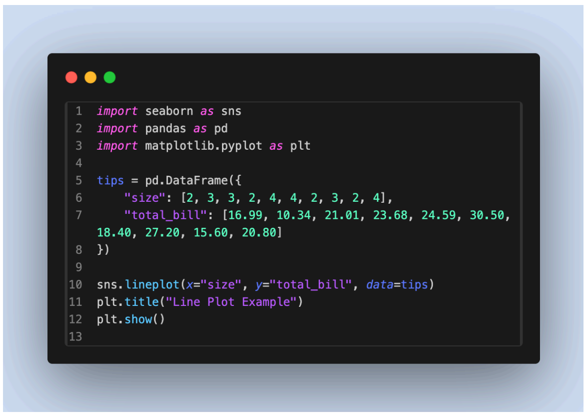


```python
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

tips = sns.load_dataset("tips")
sns.scatterplot(x="total_bill", y="tip", hue="sex", data=tips)

plt.show()
```

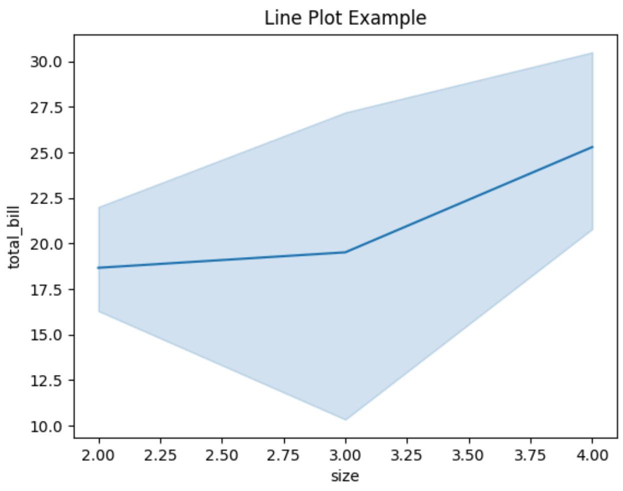


Each point represents a bill and its tip, colored by gender.

(b) Line Plot – Shows trends or patterns over categories.

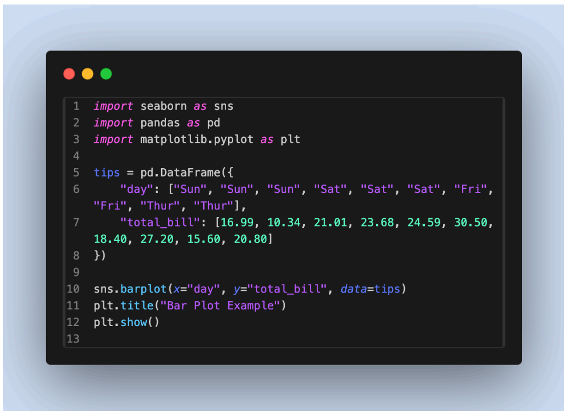


```python
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

tips = pd.DataFrame({
    "size": [2, 3, 3, 2, 4, 4, 2, 3, 2, 4],
    "total_bill": [16.99, 10.34, 21.01, 23.68, 24.59, 30.50, 18.40, 27.20, 15.60, 20.80]
})

sns.lineplot(x="size", y="total_bill", data=tips)
plt.title("Line Plot Example")
plt.show()
```

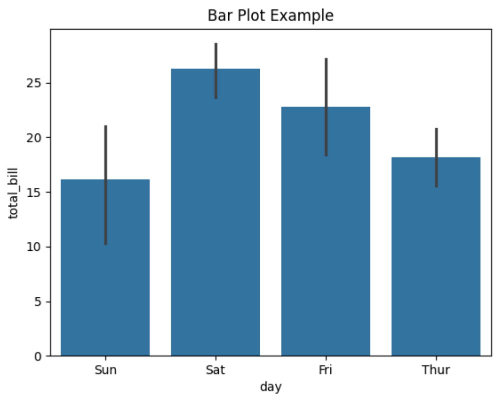


Displays how the bill amount changes with group size.

(c) Bar Plot – Compares averages across categories.



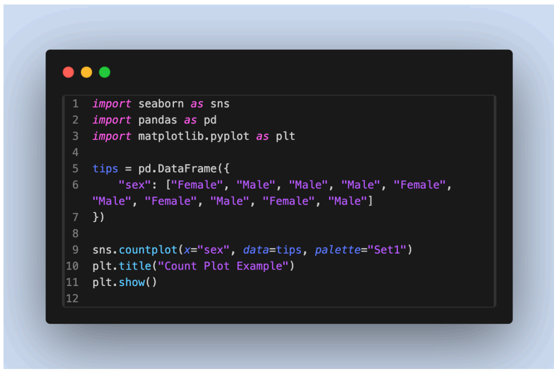
```python
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

tips = pd.DataFrame({
    "day": ["Sun", "Sun", "Sun", "Sat", "Sat", "Sat", "Fri", "Fri", "Thur", "Thur"],
    "total_bill": [16.99, 10.34, 21.01, 23.68, 24.59, 30.50, 18.40, 27.20, 15.60, 20.80]
})

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



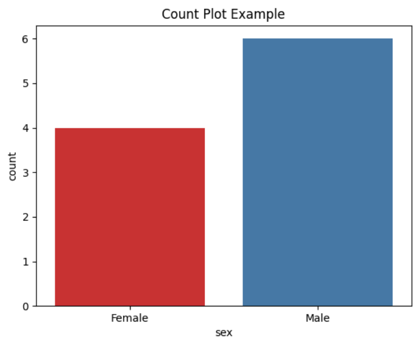
Shows average total bill per day.

(d) Count Plot

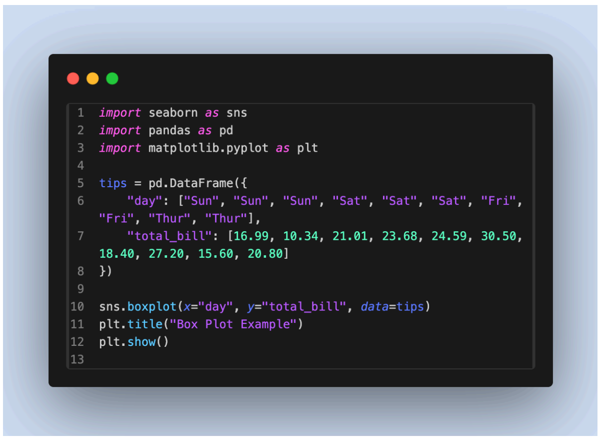
Use case: To count the number of observations in each category.



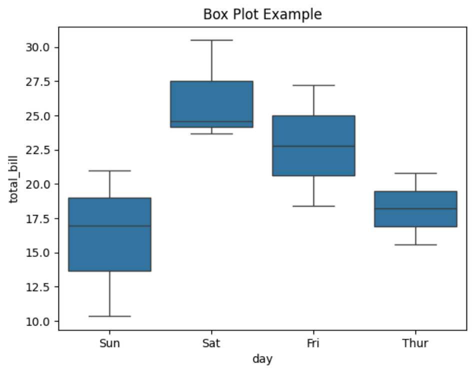
```python
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

tips = pd.DataFrame({
    "sex": ["Female", "Male", "Male", "Male", "Female", "Male", "Female", "Male", "Female", "Male"]
})

sns.countplot(x="sex", data=tips, palette="Set1")
plt.title("Count Plot Example")
plt.show()
```



(e) Box Plot – Shows spread, quartiles, and outliers.

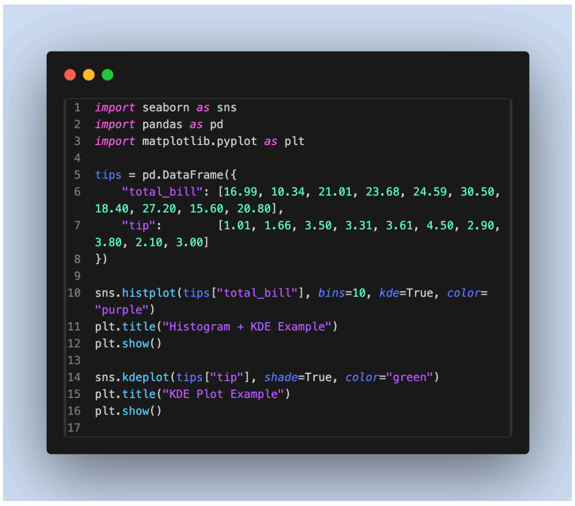


```python
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

tips = pd.DataFrame({
    "day": ["Sun", "Sun", "Sun", "Sat", "Sat", "Sat", "Fri",
"Fri", "Thur", "Thur"],
    "total_bill": [16.99, 10.34, 21.01, 23.68, 24.59, 30.50,
18.40, 27.20, 15.60, 20.80]
})

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

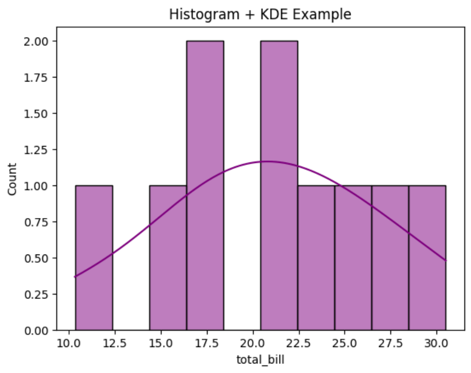


Shows **quartiles, median, and outliers** for bills by day.

(f) Histogram / KDE Plot – Shows distribution of data.

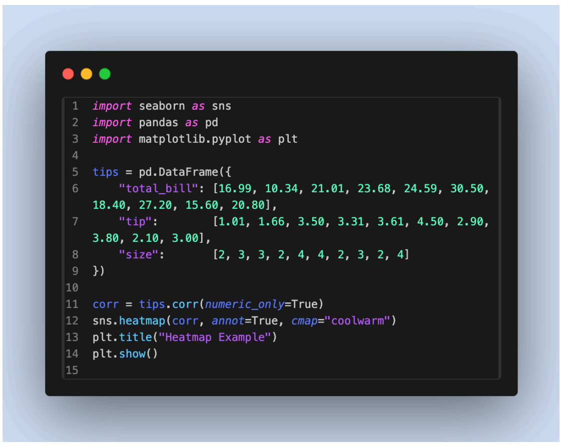


```python
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

tips = pd.DataFrame({
    "total_bill": [16.99, 10.34, 21.01, 23.68, 24.59, 30.50,
18.40, 27.20, 15.60, 20.80],
    "tip":        [1.01, 1.66, 3.50, 3.31, 3.61, 4.50, 2.90,
3.80, 2.10, 3.00]
})

sns.histplot(tips["total_bill"], bins=10, kde=True, color=
"purple")
plt.title("Histogram + KDE Example")
plt.show()

sns.kdeplot(tips["tip"], shade=True, color="green")
plt.title("KDE Plot Example")
plt.show()
```



Histogram shows **distribution** of bills; KDE smooths into a **curve**.

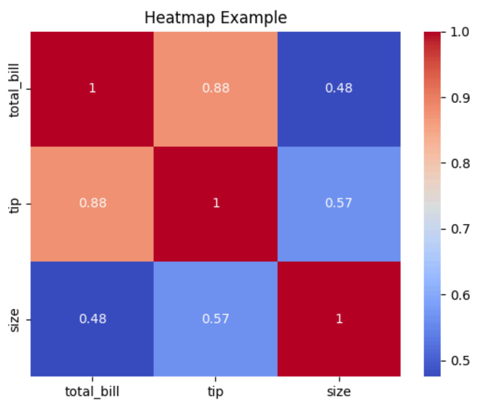
(g) Heatmap – Visualizes correlations or matrix values.



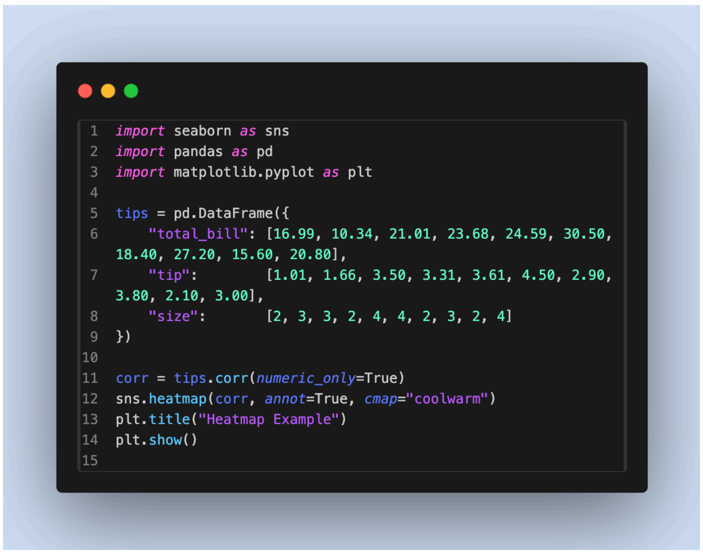
```python
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

tips = pd.DataFrame({
    "total_bill": [16.99, 10.34, 21.01, 23.68, 24.59, 30.50,
18.40, 27.20, 15.60, 20.80],
    "tip":        [1.01, 1.66, 3.50, 3.31, 3.61, 4.50, 2.90,
3.80, 2.10, 3.00],
    "size":       [2, 3, 3, 2, 4, 4, 2, 3, 2, 4]
})

corr = tips.corr(numeric_only=True)
sns.heatmap(corr, annot=True, cmap="coolwarm")
plt.title("Heatmap Example")
plt.show()
```

Visualizes **correlation** between numeric

Columns.

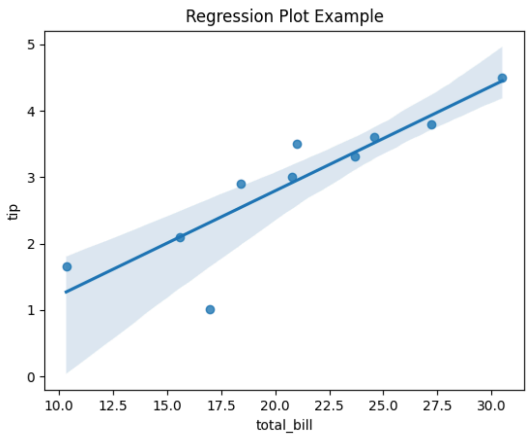


(h) Regression Plot – Scatter + best-fit regression line.



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

tips = pd.DataFrame({
    "total_bill": [16.99, 10.34, 21.01, 23.68, 24.59, 30.50,
18.40, 27.20, 15.60, 20.80],
    "tip":        [1.01, 1.66, 3.50, 3.31, 3.61, 4.50, 2.90,
3.80, 2.10, 3.00],
    "size":       [2, 3, 3, 2, 4, 4, 2, 3, 2, 4]
})

corr = tips.corr(numeric_only=True)
sns.heatmap(corr, annot=True, cmap="coolwarm")
plt.title("Heatmap Example")
plt.show()

```

Shows **scatter plot + regression line**.



COMPARISON OF MATPLOTLIB AND SEABORN

Matplotlib:

Core Library:
Matplotlib is a foundational library for creating static, animated, and interactive visualizations in Python. It provides a **low-level interface** with fine-grained control over every element of a figure, from axes and labels to markers and annotations.

Flexibility:
It supports a wide variety of plots such as line plots, bar charts, scatter plots, histograms, pie charts, 3D plots, and even specialized visualizations like geographical maps. With sufficient customization, nearly any type of visualization can be produced.

Mature Ecosystem:
Matplotlib is one of the oldest and most established plotting libraries in Python. It serves as the base for many other visualization libraries, including Seaborn, which extend its capabilities.

Customization:
Every detail of a visualization can be customized — colors, line styles, fonts, markers, axis ticks, annotations, legends, and more. This makes it suitable for producing publication-quality graphics.

Integration:
It integrates seamlessly with **NumPy, Pandas, SciPy, and IPython/Jupyter**, making it a versatile choice for scientific computing, research, and production-level visualization.

Advantages of Matplotlib:

- Provides **maximum customization and control**.
- Supports a **wide range of visualization types**.
- Backed by a **large community** with extensive tutorials and documentation.
- Ideal for **publication-quality figures** and detailed research graphics.

Seaborn:

High-Level Interface:
Seaborn is built on top of Matplotlib and provides a **simplified, high-level API**. It reduces the amount of code needed to generate complex visualizations by applying sensible defaults.

Statistical Plotting:
It specializes in **statistical graphics**. Seaborn includes built-in functions for plots such as scatter plots, bar plots, violin plots, box plots, pair plots, heatmaps, and distribution plots — tasks that would require more effort with Matplotlib.

Aesthetics:
Seaborn provides attractive **default themes and color palettes**. By default, plots look polished and visually pleasing, reducing the need for heavy customization. It also supports color palettes for categorical and continuous data.

Integration with Pandas:
It works seamlessly with **Pandas DataFrames**. Instead of manually extracting arrays, you can directly specify column names, making the workflow more intuitive for data analysis and exploratory tasks.

Advantages of Seaborn:

- Provides **simplified syntax** for complex plots.
- Comes with **built-in support for statistical analysis**.
- Has **visually appealing defaults** that make plots look professional with minimal effort.
- Works **directly with Pandas DataFrames**, speeding up data exploration.
- Ideal for **exploratory data analysis (EDA)** and visualization-driven insights.

Overall Comparison:

The choice between Matplotlib and Seaborn depends on the use case.

- **Matplotlib** is preferred when **flexibility, customization, or specialized visualizations** are required. It is the backbone of Python visualization and excels in detailed, publication-ready graphics.
- **Seaborn** is preferred when the focus is on **statistical analysis, quick insights, and aesthetic plots** with minimal coding effort.

In practice, many data scientists and analysts use **both libraries together**: Seaborn for quick and attractive statistical plots, and Matplotlib for deeper customization when required.