

Visualization

Contents

- Visualization
 - ♦ Matplotlib Basics
 - ♦ Two Interfaces in Matplotlib
- Example Plots in Matplotlib
 - ♦ Customizing Plots
- ♦ Seaborn Basics
 - ♦ Common Seaborn Plots
 - ♦ Seaborn Styles & Themes
- When to Use Seaborn

matplotlib




seaborn


Data visualization is a **core step in data analysis and machine learning**, helping us reveal **patterns, trends, and insights** hidden in raw data.

♦ Matplotlib

- **History:** Created by **John D. Hunter** in **2003**.

- **Purpose:** Provide a MATLAB-like plotting interface for Python.
- **Key Features:**
 - Highly customizable (line plots, scatter plots, histograms, 3D, etc.)
 - Fine-grained control over every element of a plot.
 - Widely used as the **foundation library** for visualization in Python.
- **Use Cases:** Academic research, reports, scientific visualization, general-purpose plotting.
-  Official Website: <https://matplotlib.org/>

◆ Seaborn

- **History:** Developed by **Michael Waskom** in **2012**.
- **Purpose:** Built on top of Matplotlib to simplify **statistical visualization**.
- **Key Features:**
 - Beautiful default styles and color palettes.
 - High-level API for statistical graphics (distributions, correlations, categorical plots).
 - Works seamlessly with **pandas DataFrames**.
- **Use Cases:** Exploratory Data Analysis (EDA), statistical reports, quick insights.
-  Official Website: <https://seaborn.pydata.org/>

◆ Matplotlib vs Seaborn

Feature	Matplotlib	Seaborn
Level	Low-level (more code, more control)	High-level (less code, simpler syntax)
Customization	Very flexible, detailed	Limited but visually appealing defaults
Data Integration	Works with arrays, NumPy	Works directly with pandas DataFrames
Plot Types	General plotting (line, scatter...)	Statistical plots (box, violin, heatmap)

◆ Relationship Between Them

- **Seaborn is built on top of Matplotlib** → every Seaborn plot is internally a Matplotlib figure.
- You can **customize Seaborn plots using Matplotlib commands**.
- Think of Matplotlib as the **engine**, and Seaborn as the **beautiful interface** on top.

👉 Together, they form the **most widely used visualization ecosystem in Python**.

Matplotlib is the most fundamental plotting library in Python. It is highly customizable but requires more code compared to Seaborn.

Why use Matplotlib?

- Foundation for most Python visualization libraries.
- Provides **low-level control** over every element in a figure.
- Suitable for research, reports, and precise figure customization.

Basic Import

```
import matplotlib.pyplot as plt
```

```
import matplotlib
import seaborn

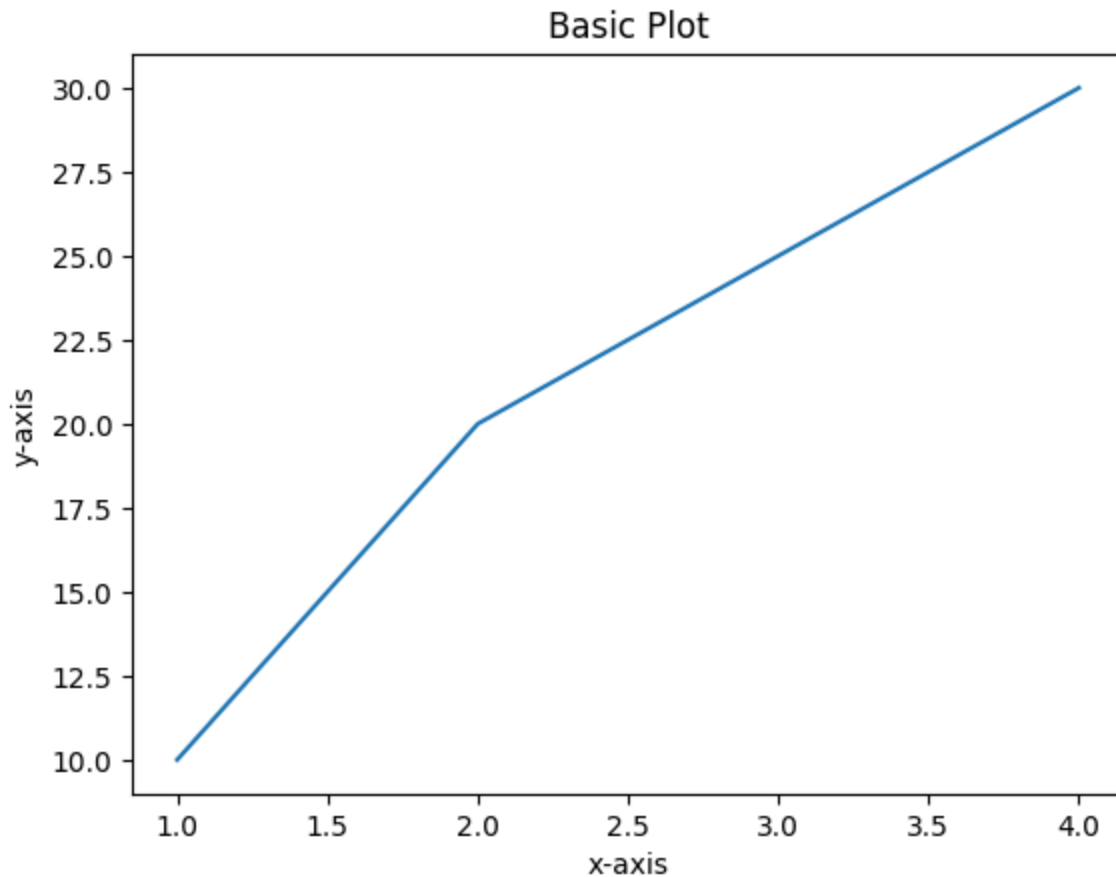
print("Matplotlib version:", matplotlib.__version__)
print("Seaborn version:", seaborn.__version__)
```

```
Matplotlib version: 3.10.1
Seaborn version: 0.13.2
```

```
import matplotlib.pyplot as plt

# Basic plotting workflow
x = [1, 2, 3, 4]
y = [10, 20, 25, 30]

plt.plot(x, y)
plt.title("Basic Plot")
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.show()
```



Matplotlib provides **two main ways** to create visualizations, and understanding the difference is key to writing clean, maintainable code.

Pyplot Interface (Stateful)

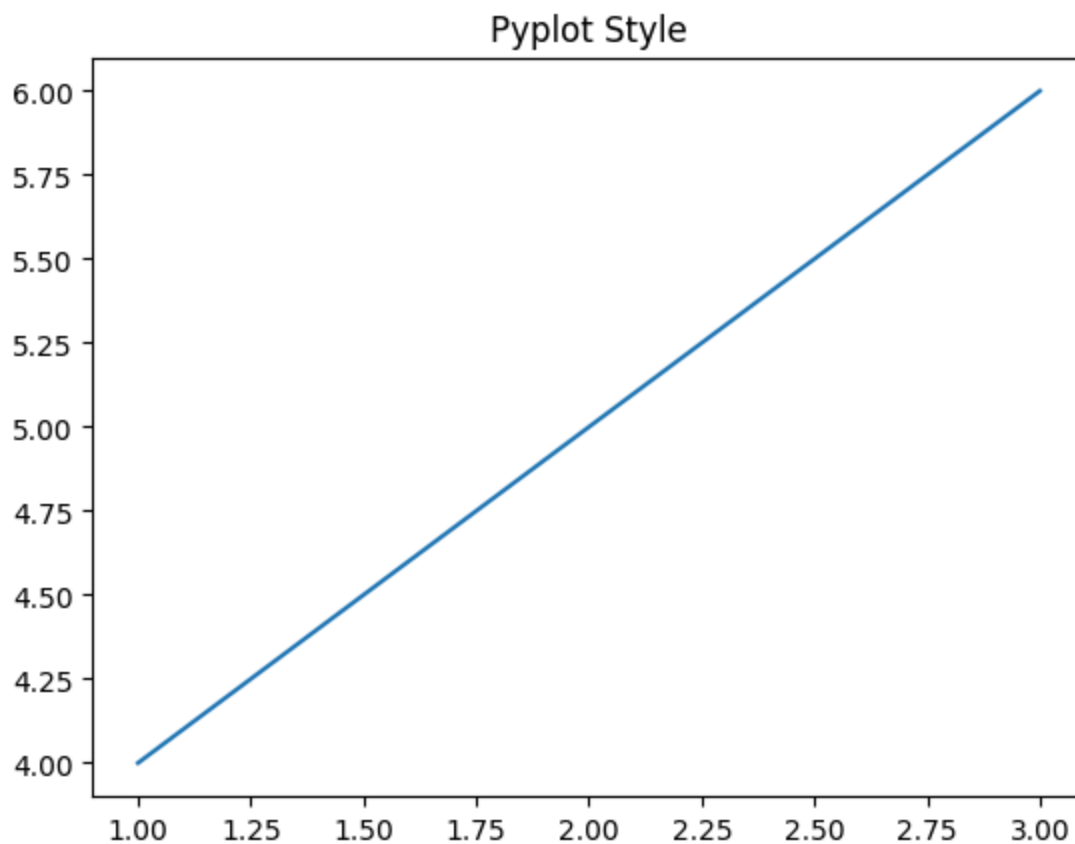
- Works like MATLAB: it **keeps track of the current figure and axes** automatically.
- Simpler syntax → good for **quick and exploratory plots**.
- Less flexible for complex, multi-plot layouts.

Pros: Easy to write, great for small scripts and interactive use (e.g., Jupyter).

Cons: Can get messy when managing multiple figures or subplots.

```
import matplotlib.pyplot as plt

# Pyplot Interface (stateful)
plt.plot([1, 2, 3], [4, 5, 6])
plt.title("Pyplot Style")
plt.show()
```



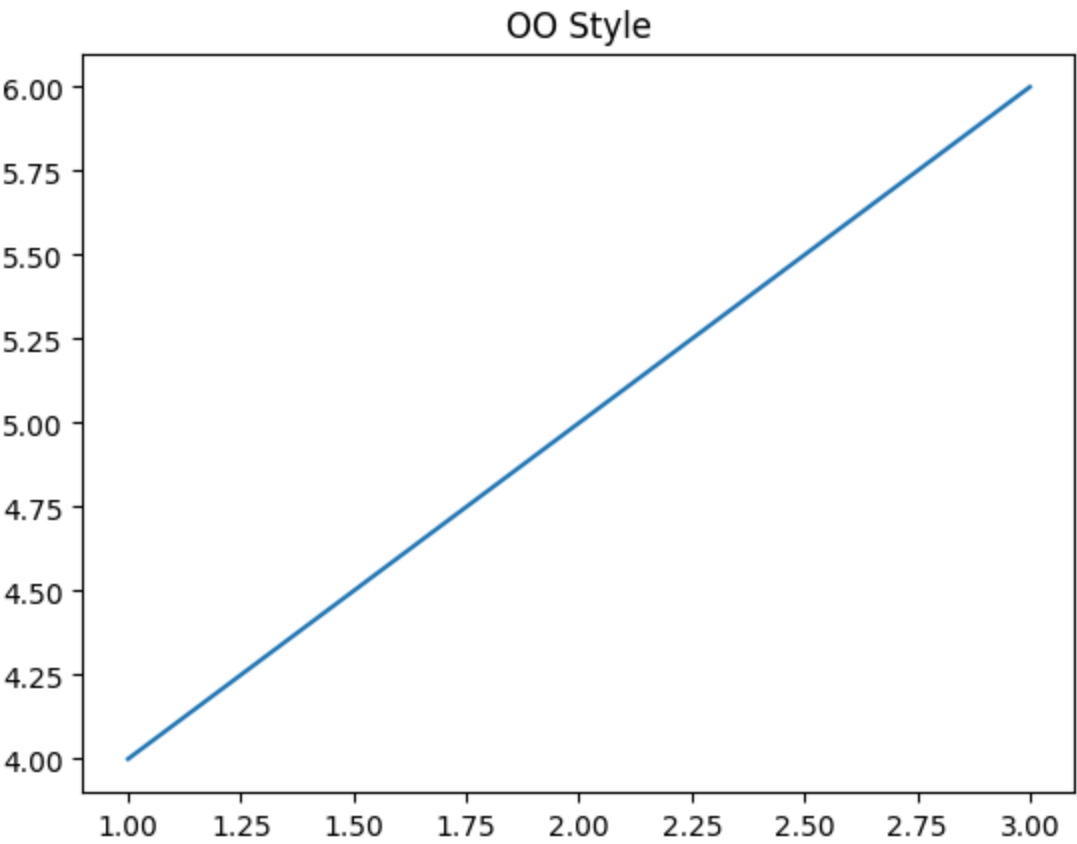
Object-Oriented (OO) Interface

- You **explicitly create a Figure and Axes object** and then call methods on them.
- Recommended for **complex, reusable, or production-ready visualizations**.
- More explicit and avoids confusion when working with multiple plots.

Pros: Full control, clearer structure, better for dashboards and reports.


Cons: Slightly more code for simple plots.

```
fig, ax = plt.subplots()
ax.plot([1, 2, 3], [4, 5, 6])
ax.set_title("OO Style")
plt.show()
```



Key Differences

Feature	Pyplot (Stateful)	Object-Oriented (OO)
Style	MATLAB-like, implicit	Explicit, Pythonic
Use Case	Quick, interactive plotting	Complex, reusable plots
Control over Figures	Limited	Full control
Best for	Fast prototyping	Reports, dashboards, scripts

 **Best Practice:** Use Pyplot for quick visual checks, but switch to the OO interface when building anything more complex or shareable.

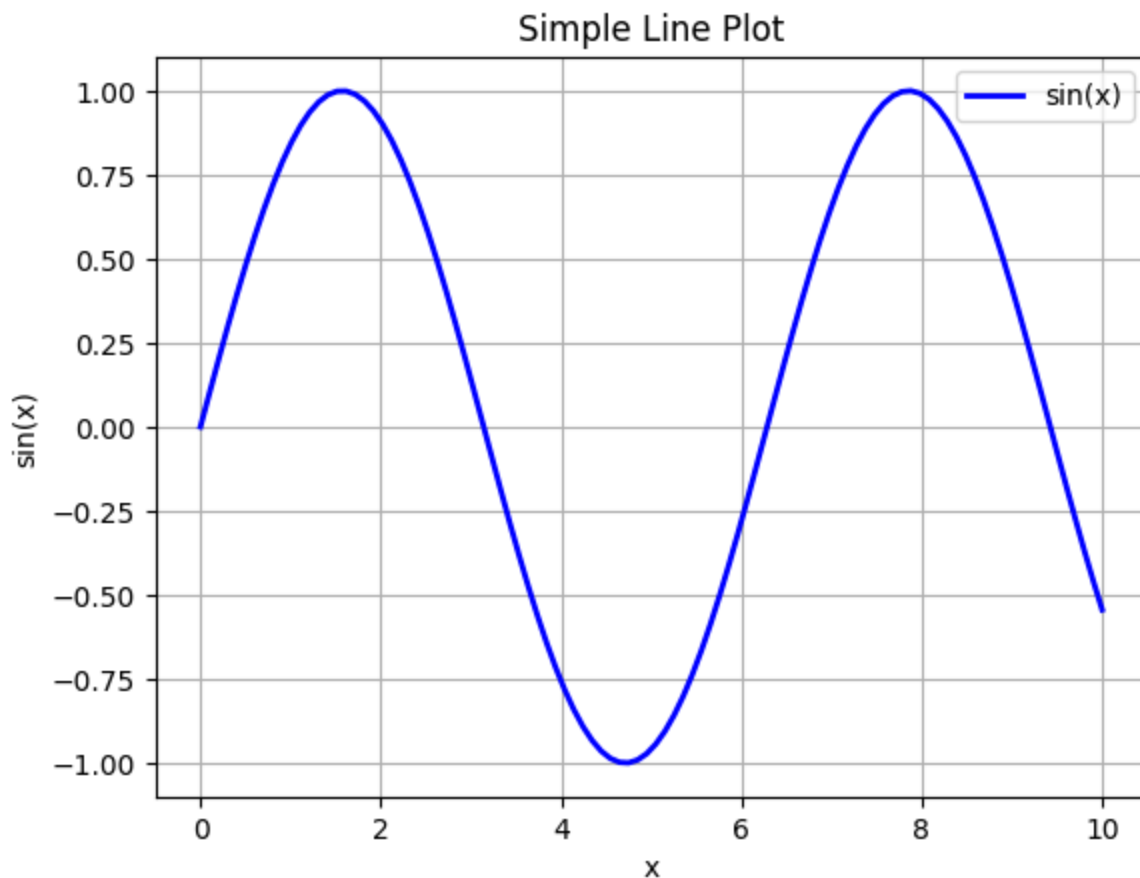
Below are some of the most common types of plots you will use with **Matplotlib**. Each example shows how to create the plot and includes explanations of the key functions.

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

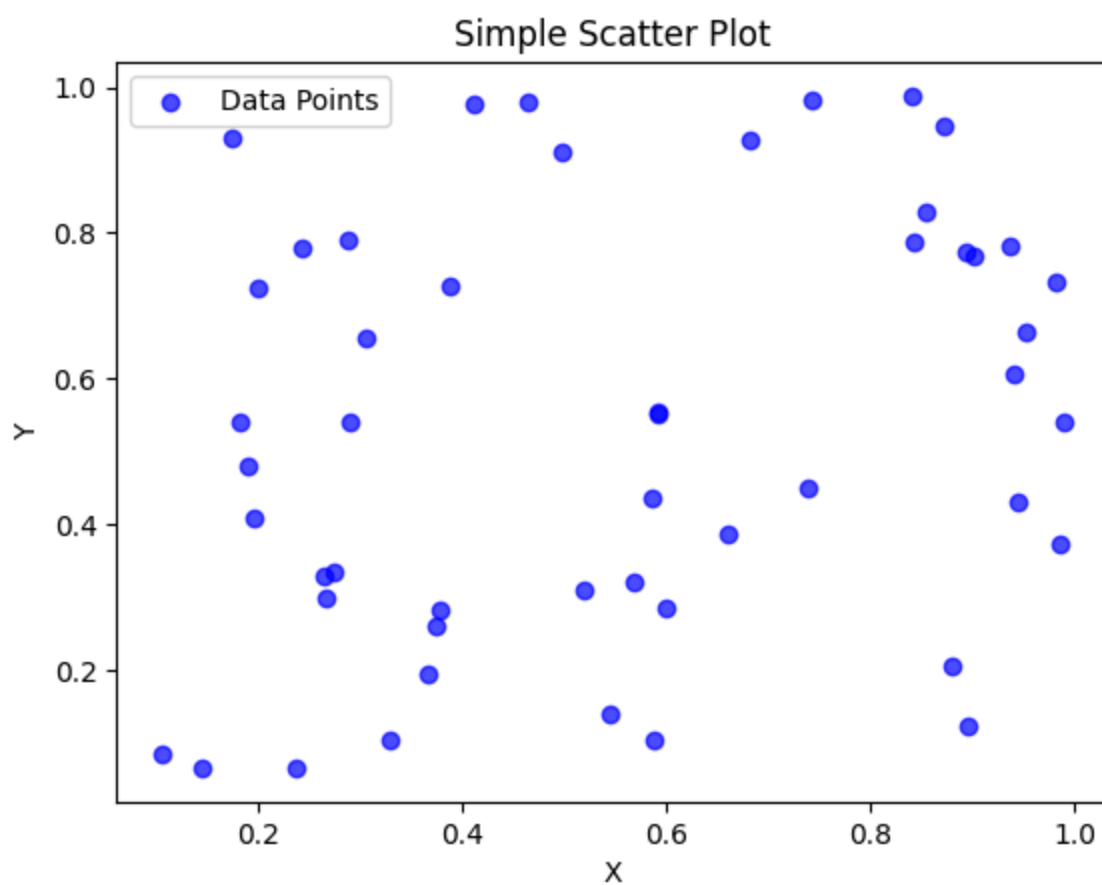
# -----
# Line Plot (Trends over data)
# -----

# Create evenly spaced values from 0 to 10
x = np.linspace(0, 10, 100)
# Compute sine function values
y = np.sin(x)

# Plot line graph
plt.plot(x, y, color='blue', linestyle='-', linewidth=2, label='sin(x)')
plt.title("Simple Line Plot") # Title of the plot
plt.xlabel("x")               # Label for x-axis
plt.ylabel("sin(x)")          # Label for y-axis
plt.grid(True)                # Add background grid
plt.legend()                  # Add legend to identify the line
plt.show()
```



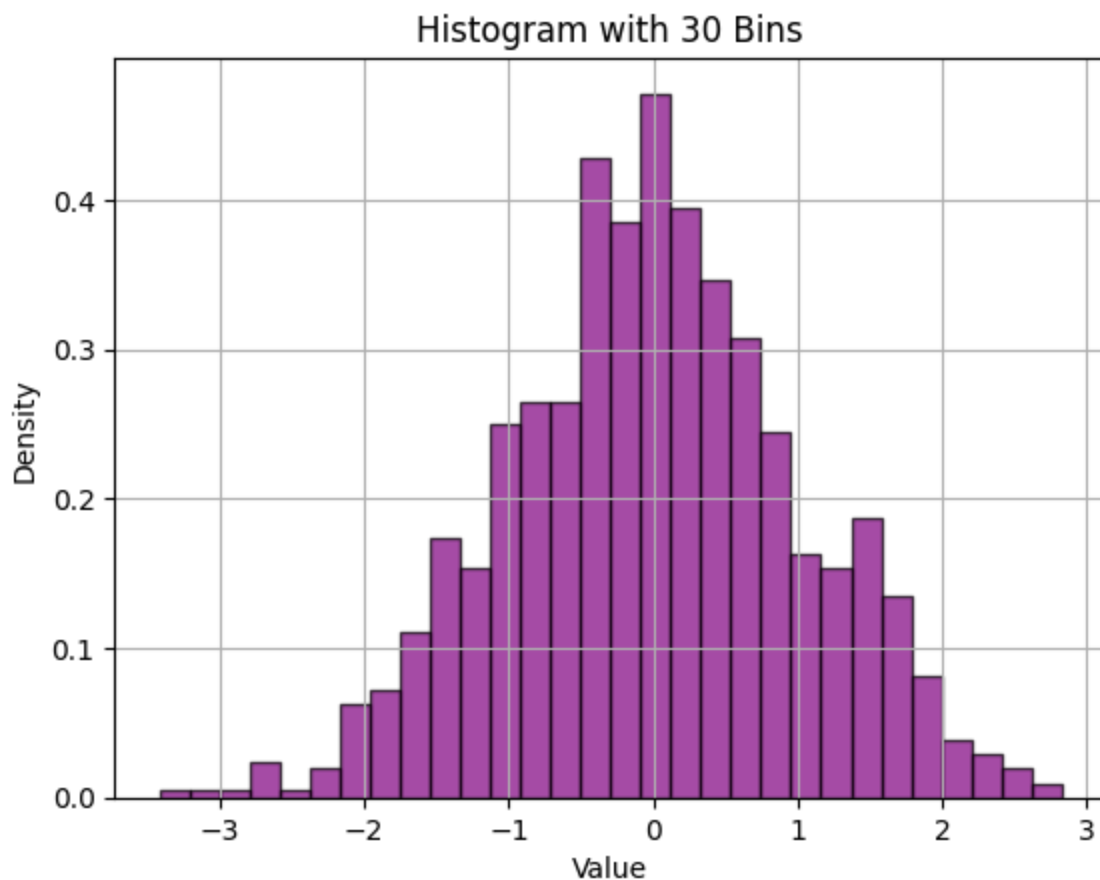
```
# -----  
# Scatter Plot (Relationships between variables)  
# -----  
  
# Generate 50 random points for x and y  
x = np.random.rand(50)  
y = np.random.rand(50)  
  
# Create scatter plot  
plt.scatter(x, y, alpha=0.7, color='blue', marker='o', label='Data Points')  
plt.title("Simple Scatter Plot")  
plt.xlabel("X")  
plt.ylabel("Y")  
plt.legend()  
plt.show()
```




```
# -----
# Histogram (Distribution of data)
# -----

# Generate 1000 random numbers from a normal distribution
data = np.random.randn(1000)

# Plot histogram with 30 bins
plt.hist(data, bins=30, alpha=0.7, density=True, color='purple', edgecolor='black')
plt.title("Histogram with 30 Bins")
plt.xlabel("Value")
plt.ylabel("Density")
plt.grid(True)
plt.show()
```



Matplotlib allows you to **customize almost everything** in your visualizations:

- **Legends** → use `plt.legend(loc='best')` to automatically place legend.
- **Ticks** → change axis labels with `plt.xticks()` and `plt.yticks()`.
- **Annotations** → highlight key points with `plt.annotate()`.
- **Styles** → use predefined themes like `plt.style.use('seaborn-v0_8')`.
- **Saving figures** → export plots using `plt.savefig('figure.png')`.

👉 These options make your plots not only **informative** but also **professional and presentation-ready**.

Seaborn is a Python data visualization library built on top of Matplotlib.

- It is mainly designed for **statistical plots**.
- Provides **beautiful default styles** with minimal code.
- Works seamlessly with **pandas DataFrames**.

```
# Check Seaborn Version
import seaborn as sns
import matplotlib.pyplot as plt
print("Seaborn version:", sns.__version__)
```

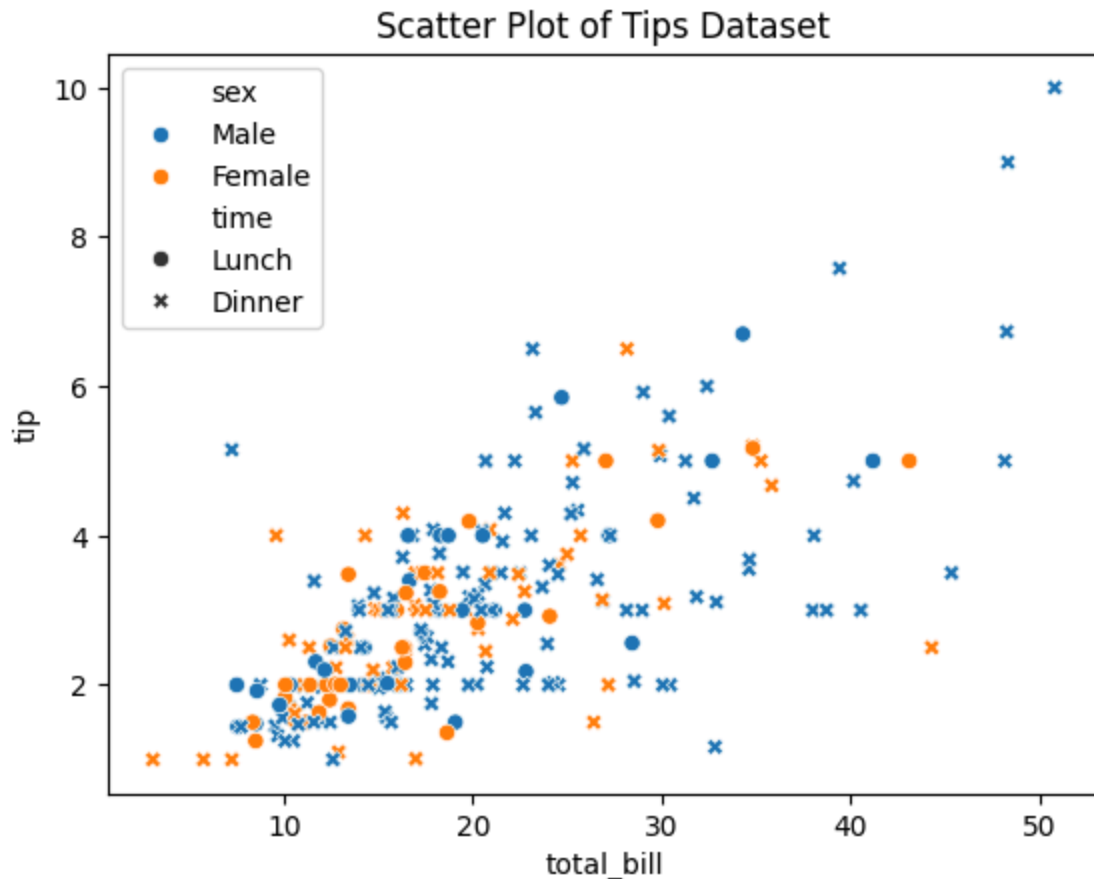
Seaborn version: 0.13.2

Example: Scatter Plot

Seaborn makes scatter plots simple and stylish by default.

```
tips = sns.load_dataset("tips") # Load example dataset (restaurant bills + tips)

# Create scatter plot with color and style differentiation
sns.scatterplot(data=tips, x="total_bill", y="tip", hue="sex", style="time")
plt.title("Scatter Plot of Tips Dataset")
plt.show()
```



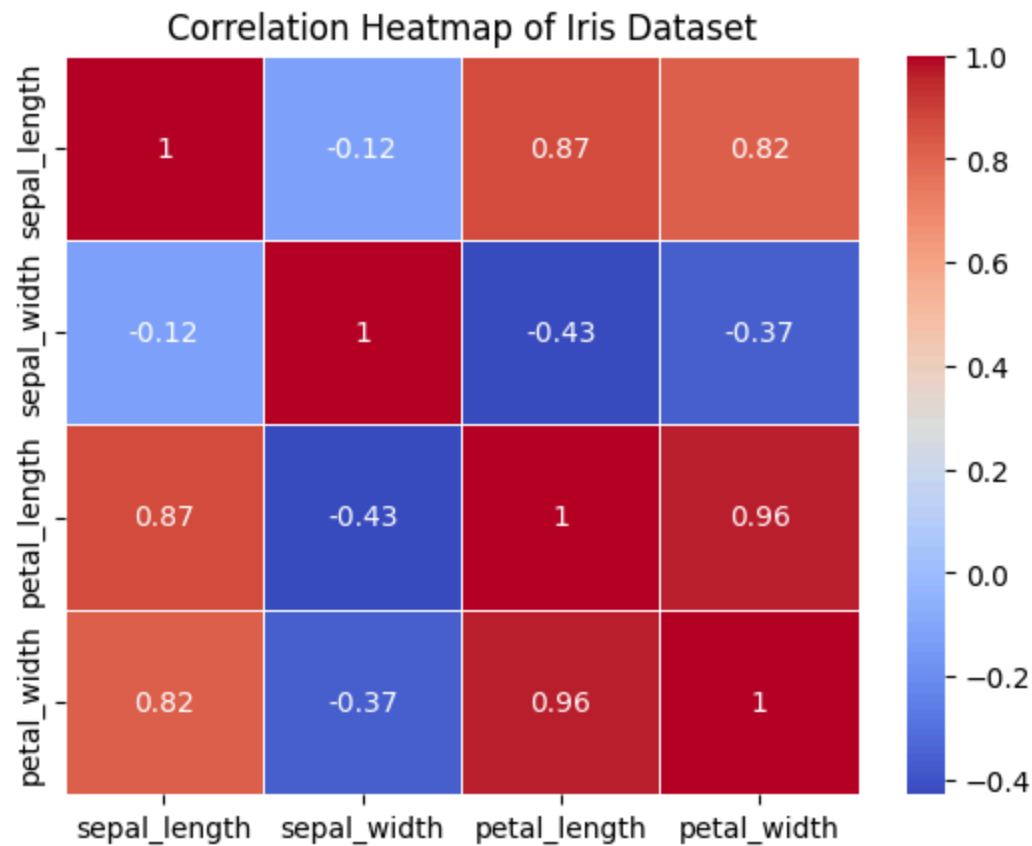
Seaborn provides many built-in statistical plots:

- `sns.histplot()` → Histogram
- `sns.boxplot()` → Box plot (spread, outliers)
- `sns.violinplot()` → Distribution + density
- `sns.heatmap()` → Correlation or matrix visualization
- `sns.pairplot()` → Multi-variable relationships
- `sns.barplot()` → Bar chart with confidence intervals
- `sns.countplot()` → Counts of categorical values

Example: Heatmap (Correlation Matrix)

```
iris = sns.load_dataset("iris") # Famous Iris dataset
corr = iris.corr(numeric_only=True) # Compute correlation between numeric columns

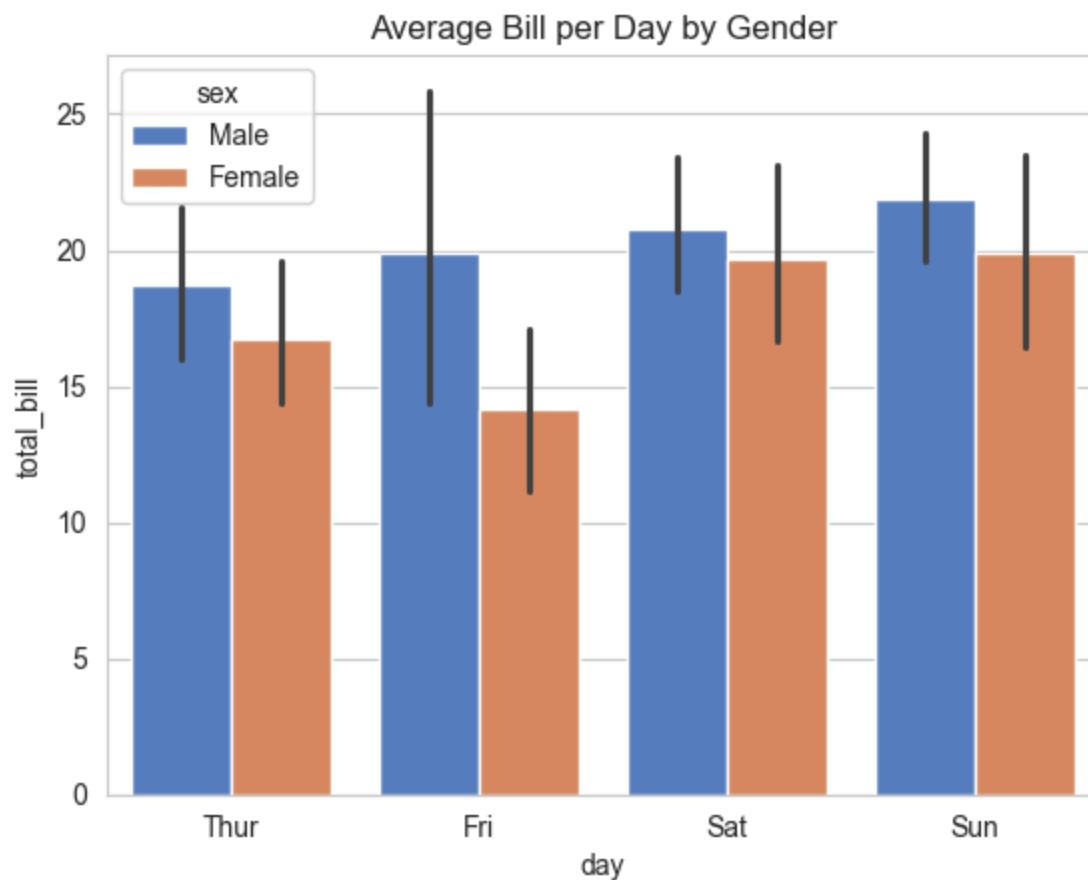
# Draw heatmap with annotations for clarity
sns.heatmap(corr, annot=True, cmap="coolwarm", linewidths=0.5)
plt.title("Correlation Heatmap of Iris Dataset")
plt.show()
```



Seaborn allows you to change global themes with just one line.

```
sns.set_style("whitegrid") # Background style
sns.set_palette("muted")   # Color palette

sns.barplot(data=tips, x="day", y="total_bill", hue="sex")
plt.title("Average Bill per Day by Gender")
plt.show()
```



- Quick **Exploratory Data Analysis (EDA)**
- Statistical plots (distributions, correlations, categories)
- Cleaner plots with **less code** compared to Matplotlib

👉 Remember: Seaborn is built on Matplotlib, so you can always customize further using Matplotlib commands.

```
# Example of mixing Seaborn with Matplotlib
ax = sns.histplot(data=tips, x="total_bill", bins=20)
ax.set_title("Customized Histogram") # Matplotlib method used on Seaborn plot
plt.show()
```

