**Matplot Library**

Matplotlib is a powerful plotting library in Python used for creating static, animated, and interactive visualizations. Matplotlib’s primary purpose is to provide users with the tools and functionality to represent data graphically, making it easier to analyze and understand. It was originally developed by John D. Hunter in 2003 and is now maintained by a large community of developers

Matplotlib’s built-in styles include classic styles reminiscent of traditional scientific plots, modern styles with vibrant colors and sleek lines, and specialized styles tailored for specific purposes such as presentation or grayscale printing. Additionally, Matplotlib allows you to customize plot styles to match your preferences or corporate branding, ensuring that your visualizations are both informative and visually appealing.

**Here are five different types of plots you can create using the Matplotlib library in Python:**

1. Line plot
2. Bar plot
3. Scatter plot
4. Histogram
5. Pie chart

**Each graph type serves a specific purpose in data analysis:**

Line plots reveal trends and continuous changes over time.

Bar plots are useful for comparative analysis between discrete groups.

Scatter plots identify relationships or correlations between two variables.

Histograms are ideal for examining the distribution and variability of data.

Pie charts effectively show proportions but should be used sparingly when there are only a few distinct categories.

Matplot Library is an excellent library for exploratory data analysis and publication quality plotting. It’s won its popularity by offering an easy-to-use procedural interface through a pyplot state machine. At the same time, it also allows to control all aspects of plotting for advanced visualizations through its main object-oriented interface, which facilitates the creation of maintainable, modular code. Because it is so easy to start using matplotlib, it’s almost universally taught as the first graphics library in universities, so it’s safe to say it won’t be going anywhere soon. That being said, matplotlib is quite old and might feel clunky at times. Add-on libraries such as seaborn try to smooth the rough edges of matplotlib by offering an arsenal of advanced visualizations out of the box, better default settings, and extended procedural interfaces to aid with the more common tasks encountered during fine-tuning the appearance of the plots.

**LINE PLOT**

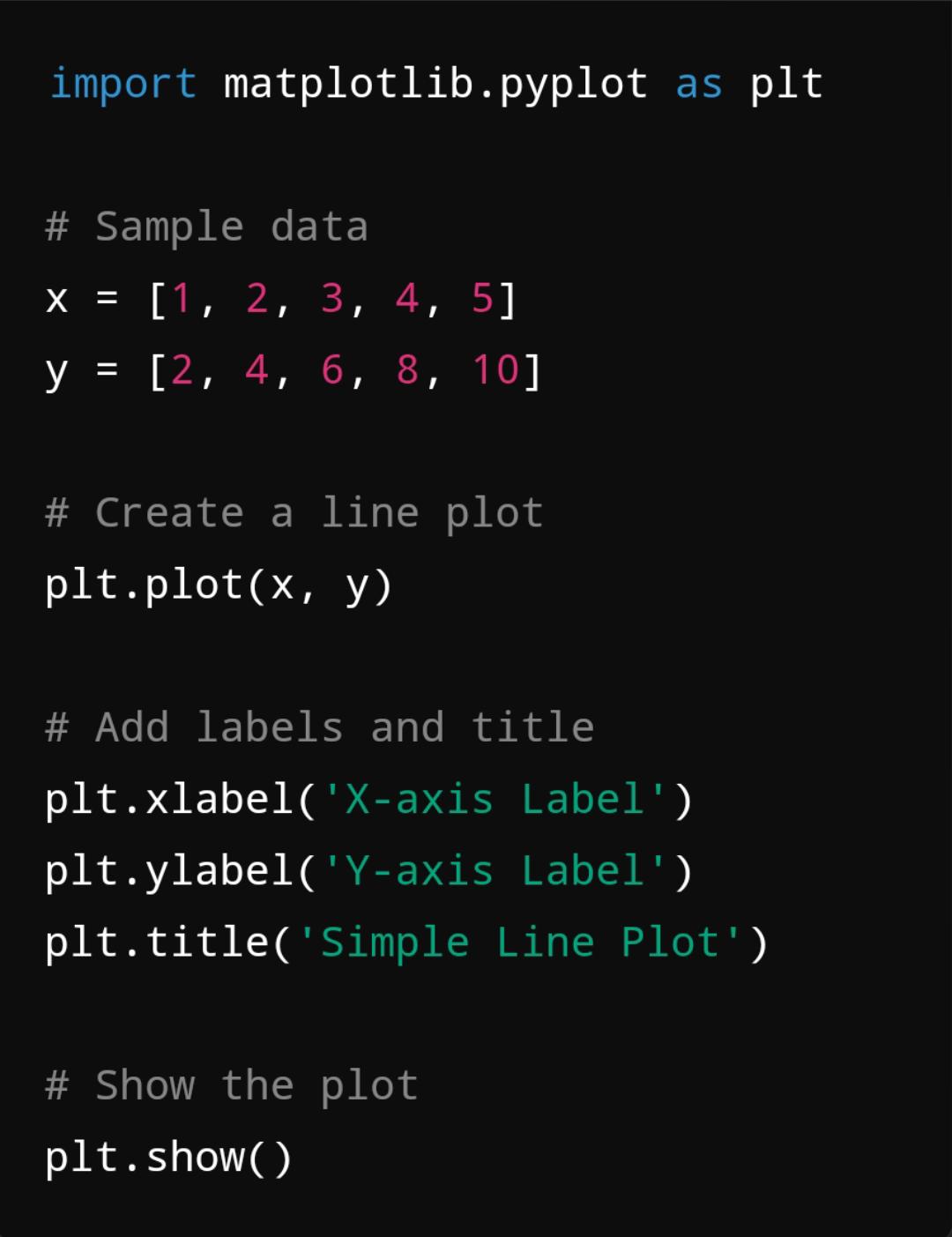
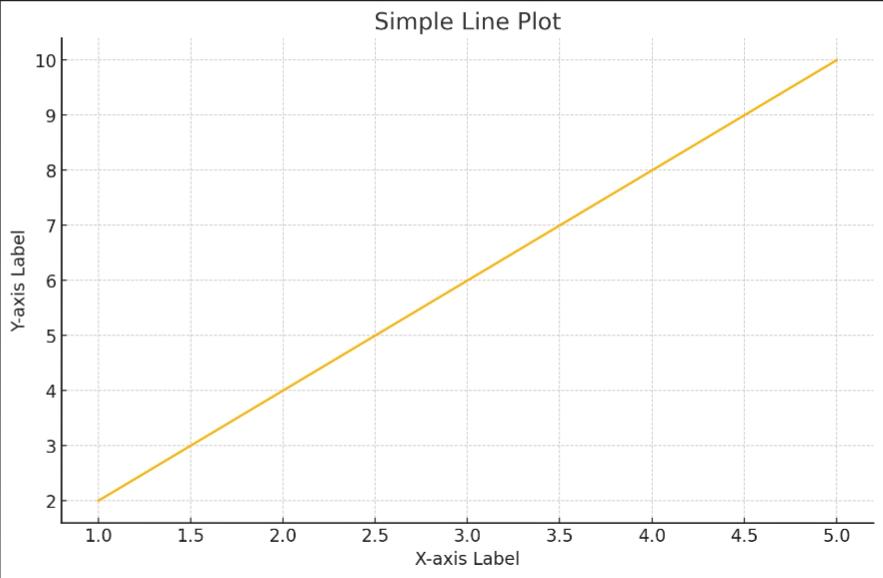
A Line plot is a basic type of data visualization used to show the relationship between two variables, often representing how one variable changes over time or along a specific range.

**PURPOSE OF LINE PLOT:**

**Trend Analysis:** Line plots are commonly used to analyze trends over time, such as tracking stock prices or temperature changes.

**Comparing Variables:** They can help compare the relationship between two variables to see if one increases, decreases, or fluctuates as the other changes.

**CODE: OUTPUT:**

****

**BAR PLOT**

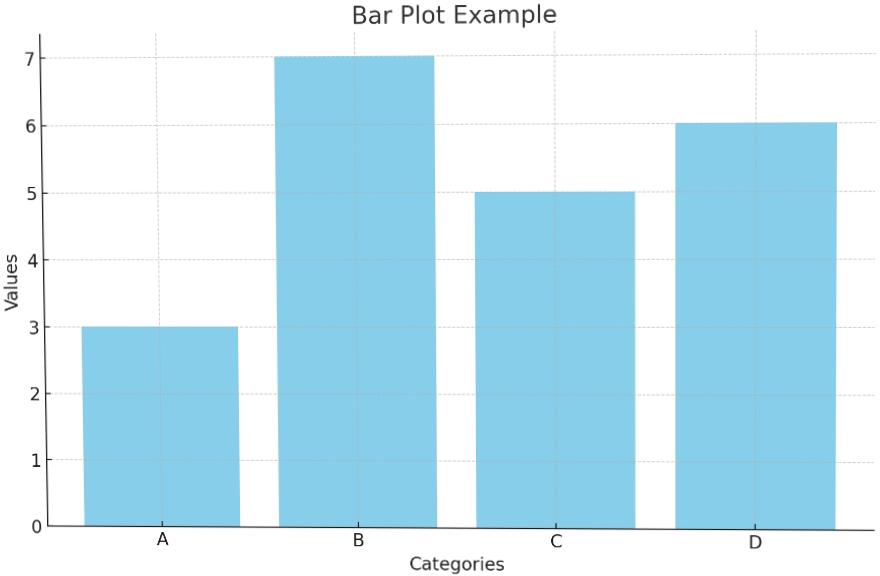
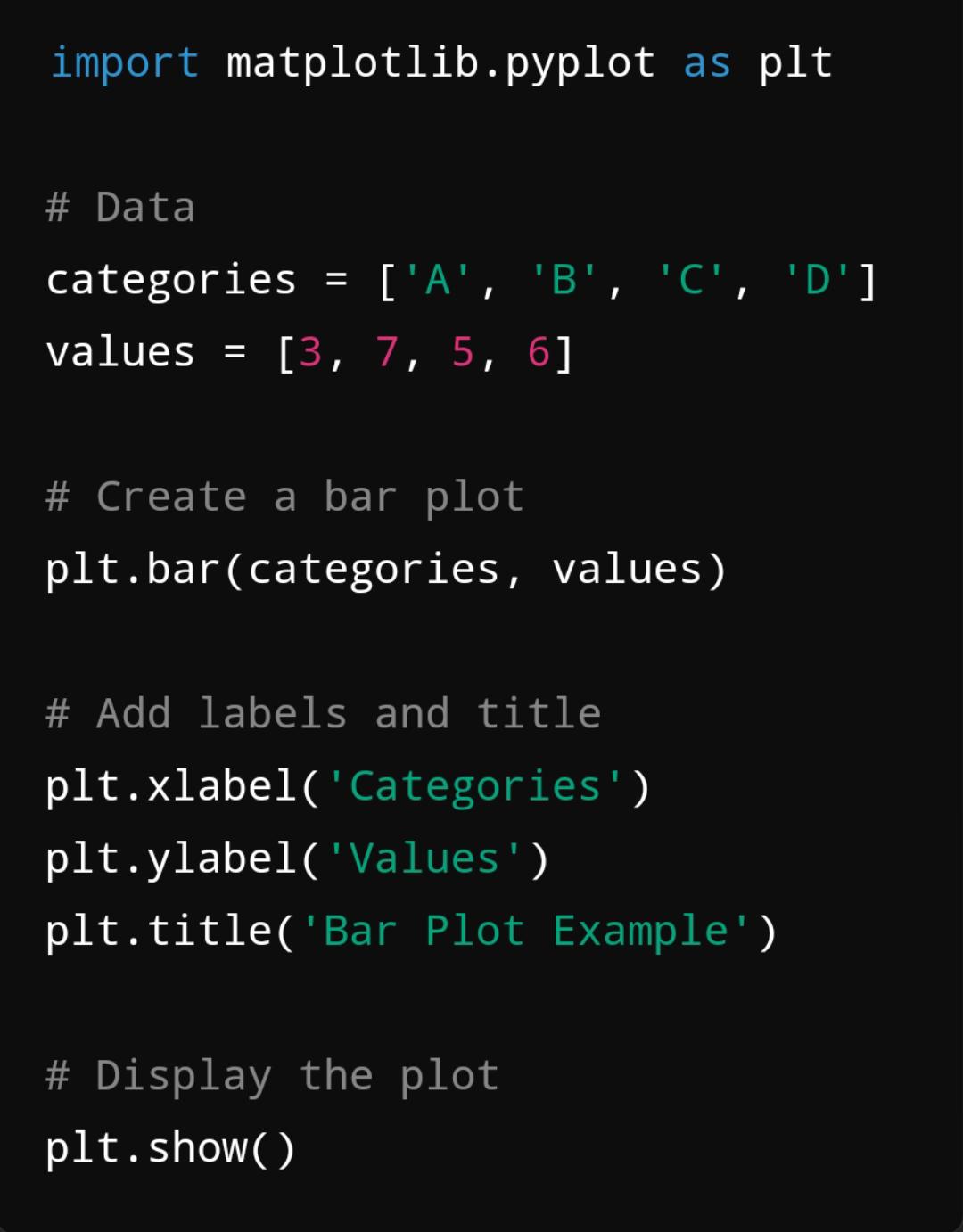
A bar plot is a type of chart that represents categorical data with rectangular bars. In the Matplotlib library, you can easily create a bar plot using the bar() function.

**PURPOSE OF BAR PLOTS:**

**Comparison:** They are great for comparing the quantities of different categories.

**Clear visualization:** The differences between categories are easy to see, making them ideal for presentations or reports.

**CODE: OUTPUT:**



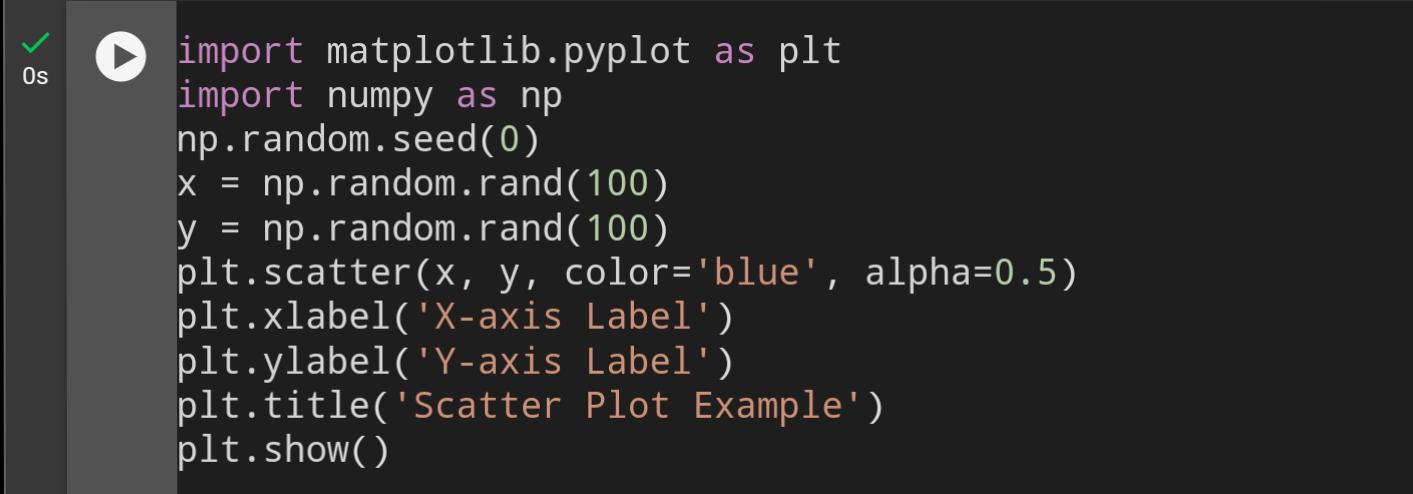
**SCATTER PLOT**

A scatter plot is a type of data visualization that displays values for typically two variables for a set of data. The points are plotted on a two-dimensional axis, where each axis represents one of the variables. Scatter plots are useful for determining relationships or correlations between the variables.

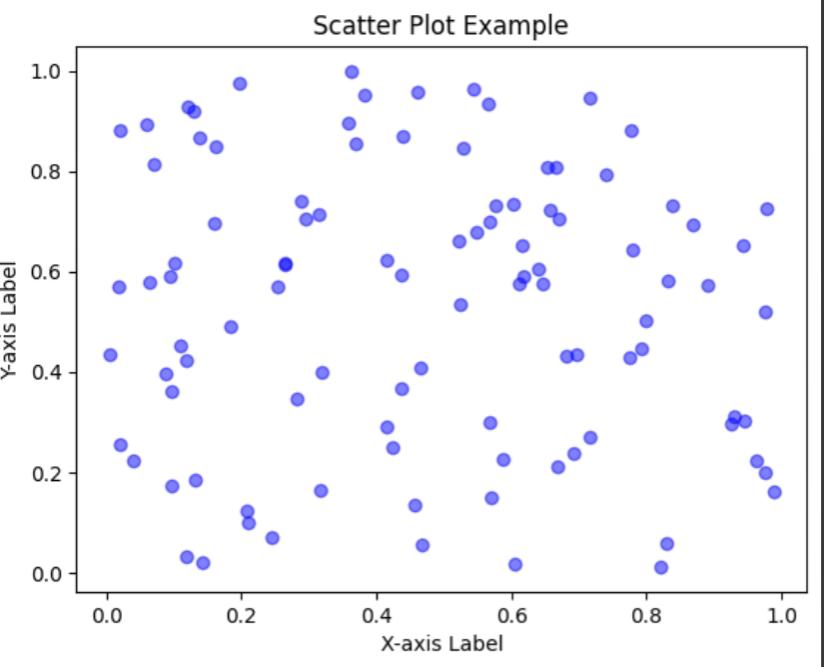
**PURPOSE OF SCATTER PLOT:**

**Visualizing Distributions:** By plotting individual data points, scatter plots provide a visual representation of the distribution of data within the two variables, allowing for a better understanding of how data is spread.

**Detecting Trends:** Scatter plots can reveal trends or patterns in the data, such as linear or nonlinear relationships. This can be particularly useful in regression analysis to determine the nature of the relationship.

**CODE:**

**OUTPUT:**

****

**HISTOGRAM**

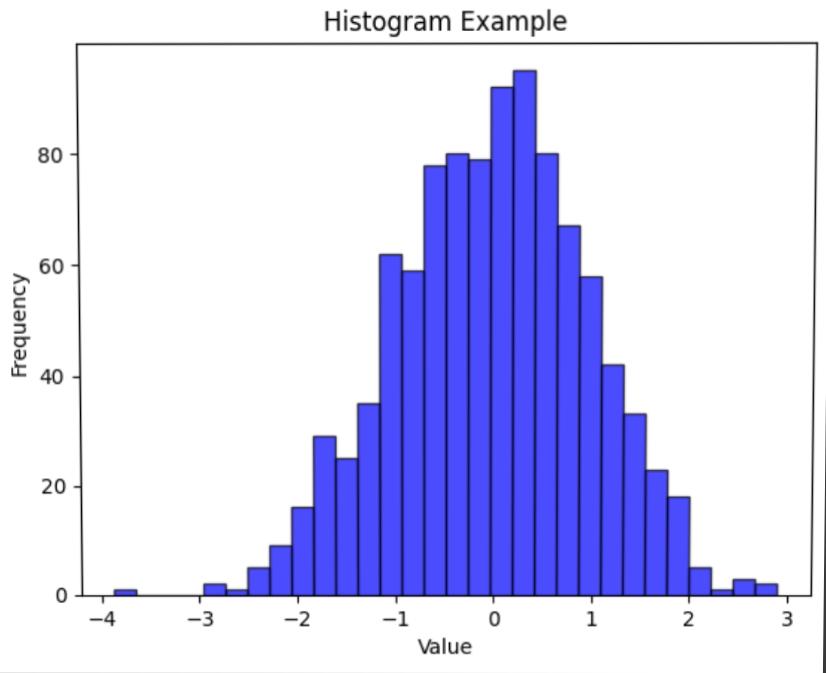
A histogram is a type of bar chart that represents the distribution of a dataset. It displays the frequency of data points falling within specified ranges (called bins) on the x-axis.

**PURPOSE OF HISTOGRAM:**

**Understanding Data Distribution:** Histograms help visualize how data points are distributed across different ranges, making it easier to identify patterns and trends.

**Exploratory Data Analysis:** They are a key tool in exploratory data analysis (EDA) for summarizing the characteristics of a dataset before applying statistical tests.

**CODE:**

**OUTPUT:**

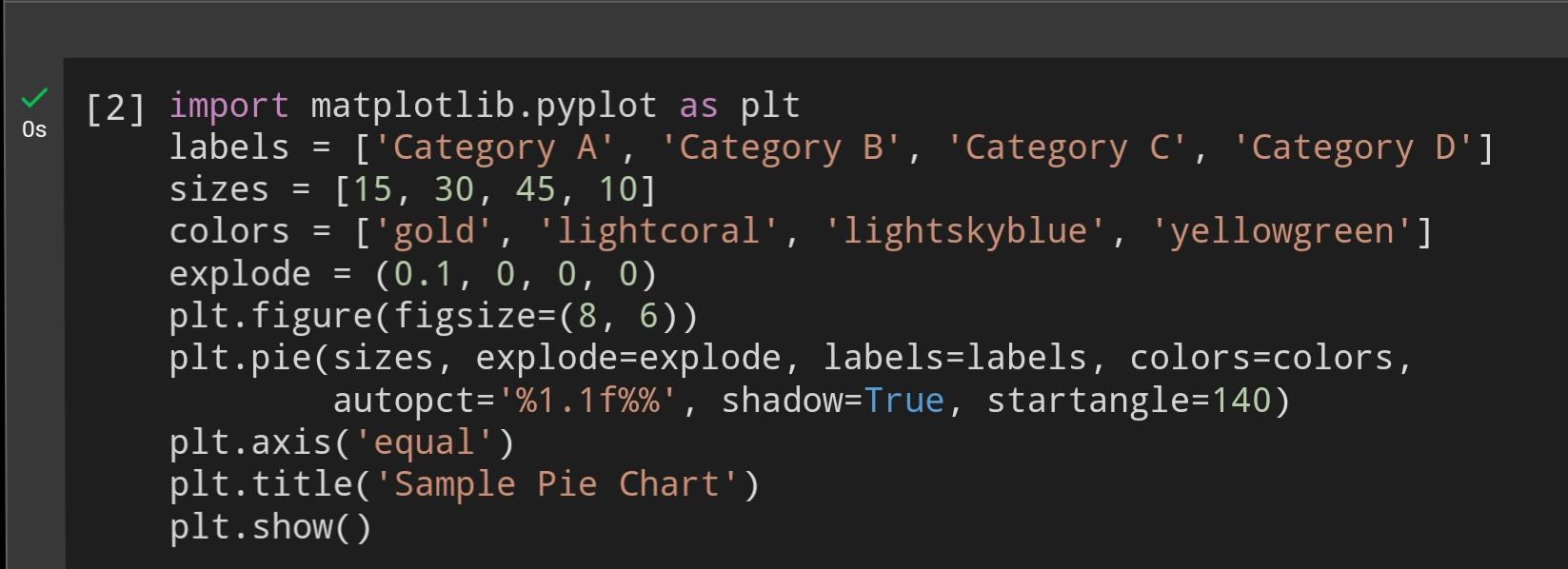
**PIE CHART**

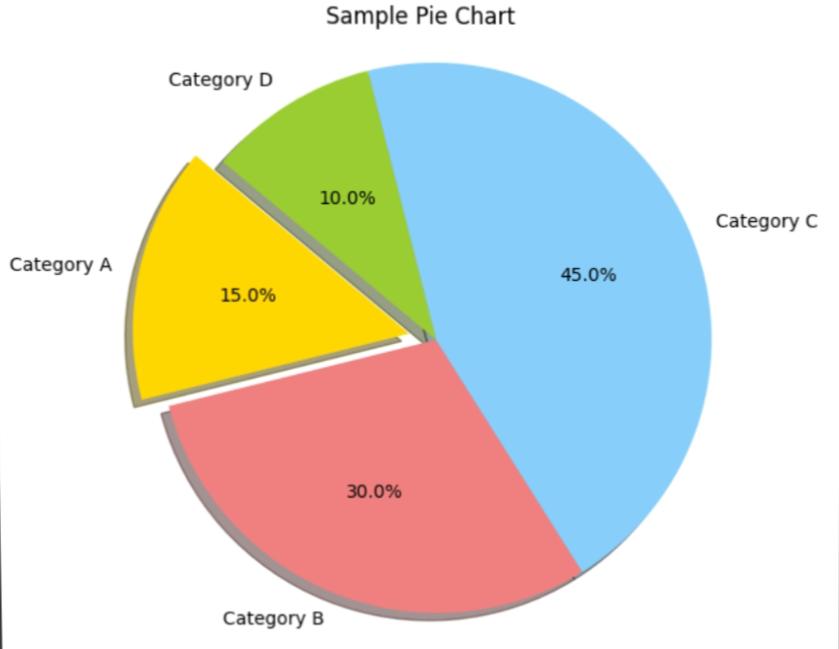
A pie chart is a circular statistical graphic that is divided into slices to illustrate numerical proportions. Each slice of the pie represents a category’s contribution to the whole, making it easy to compare relative sizes at a glance. The entire pie chart represents 100% of the data, and the size of each slice corresponds to its percentage of the total. Pie charts are commonly used in business, education, and media to present data in a visually appealing and easily understandable format.

**PURPOSE OF PIE CHART :**

**Data Simplification:** Pie charts simplify complex data sets by condensing them into a single visual, making it easier for audiences to grasp key insights quickly.

**Decision-Making Aid:** In business and organizational contexts, pie charts can aid in decision-making by illustrating the distribution of resources, sales, or market share.

**CODE:**

**OUTPUT:**

**SEABORN LIBRARY**

Seaborn is an amazing visualization library for statistical graphics plotting in Python. It provides beautiful default styles and color palettes to make statistical plots more attractive. It is built on top matplotlib library and is also closely integrated with the data structures from pandas.

Seaborn aims to make visualization the central part of exploring and understanding data. It provides dataset-oriented APIs so that we can switch between different visual representations for the same variables for a better understanding of the dataset.

**Different categories of plot in Seaborn :**

Plots are basically used for visualizing the relationship between variables. Those variables can be either completely numerical or a category like a group, class, or division. Seaborn divides the plot into the below categories :

**Scatter Plot (sns.scatterplot):**

Visualizes the relationship between two continuous variables. It shows individual data points in a Cartesian plane.

**Line Plot (sns.lineplot):**

Displays information as a series of data points called ‘markers’ connected by straight line segments. It’s useful for visualizing trends over time.

**Bar Plot (sns.barplot):**

Represents categorical data with rectangular bars. The length of each bar is proportional to the value it represents.

**Count Plot (sns.countplot):**

Similar to a bar plot but specifically for displaying the count of observations in each categorical bin.

**Box Plot (sns.boxplot):**

Summarizes the distribution of a dataset by displaying the median, quartiles, and outliers. It’s useful for understanding the spread and skewness of the data.

Seaborn is a powerful Python data visualization library that builds on top of Matplotlib and is tightly integrated with pandas. It simplifies the process of creating informative and aesthetically appealing statistical plots by offering built-in themes, color palettes, and functions for visualizing complex datasets. Key features include its ease of use, ability to handle DataFrames directly, and its support for advanced visualizations such as heatmaps, pair plots, and categorical plots. Seaborn is an essential tool for data scientists and analysts aiming to explore and present data effectively.

**SCATTER PLOT**

Creating a scatter plot in Seaborn is straightforward and can effectively visualize the relationship between two continuous variables. Below is a detailed explanation, including examples of how to create and customize scatter plots using Seaborn.

**PURPOSE OF SCATTER PLOT:**

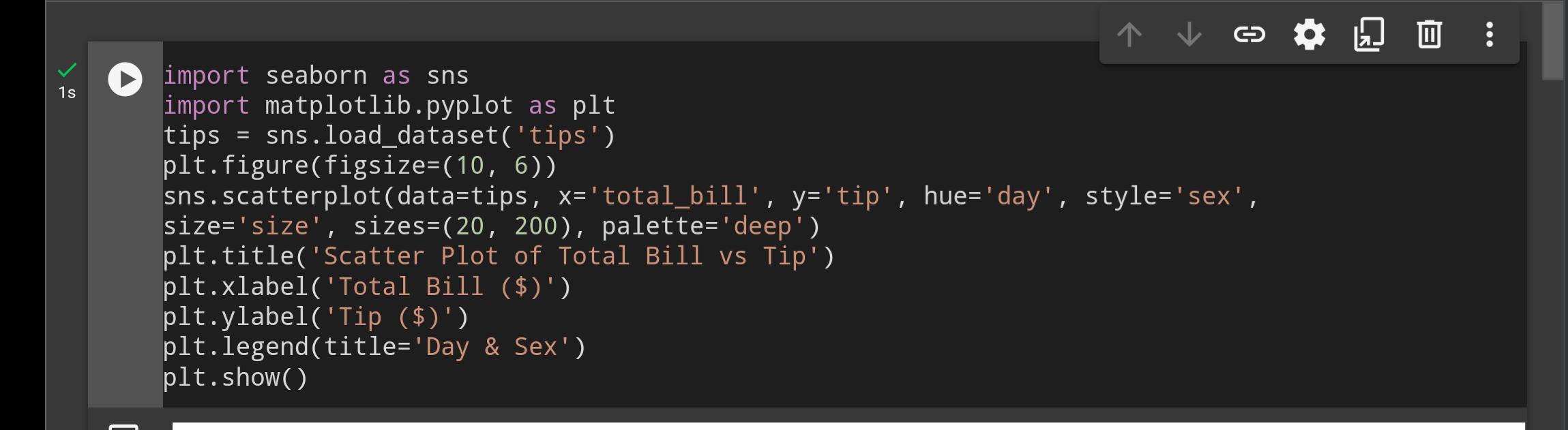
**Assessing Model Fit:**

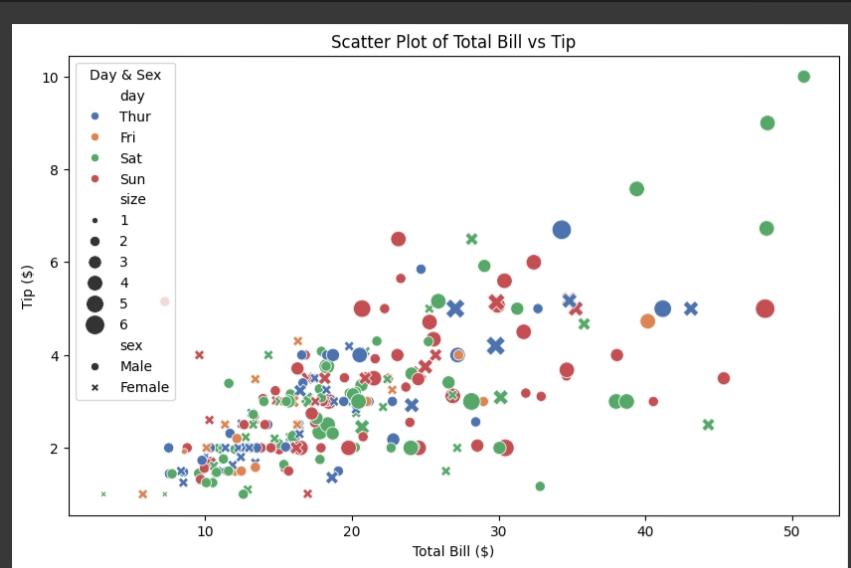
In statistical modeling, scatter plots can be used to visualize the relationship between predicted and actual values, helping assess how well a model fits the data

**Visualizing Relationships Between Variables:**

Correlation: Scatter plots help identify relationships between two continuous variables, showing whether they have a positive, negative, or no correlation.

**CODE:**

****

**OUTPUT:**

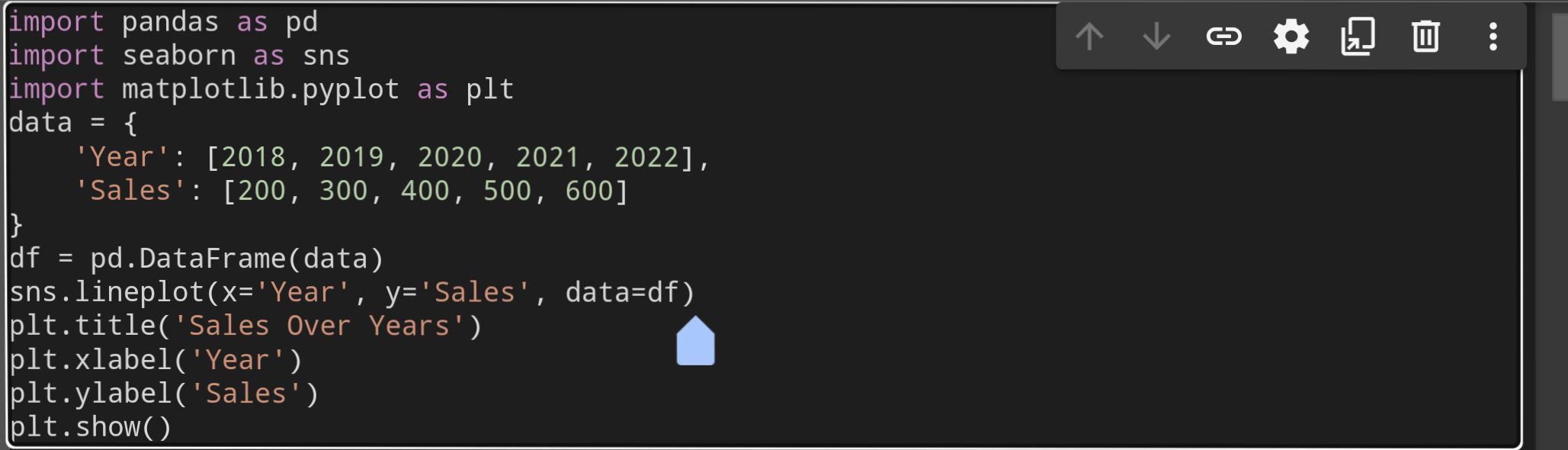
**LINE PLOT**

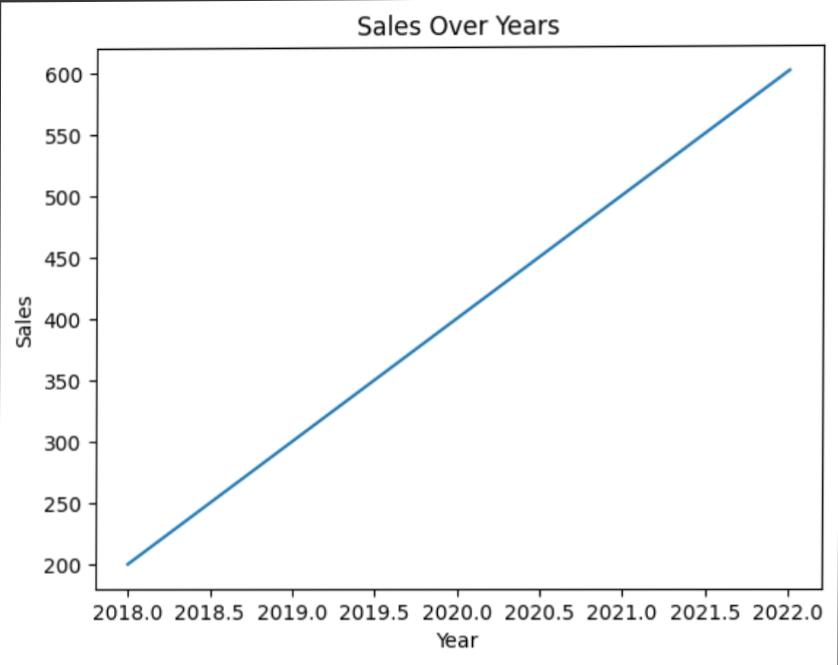
Creating a line plot in Seaborn is straightforward and allows for clear visualization of data trends over a continuous variable. Below are the steps and an example to create a basic line plot using Seaborn.Import the necessary librariesns.lineplot() to create the plot.

**PURPOSE OF LINE PLOT:**

**Trend Visualization:** Line plots effectively display trends and patterns in data, making it easy to see whether values are increasing, decreasing, or remaining constant over time.

**Time Series Analysis:** They are particularly useful for time series data, where the x-axis represents time (e.g., days, months, years) and the y-axis represents a variable of interest (e.g., sales, temperature).

**CODE:**

**OUTPUT:**

**BAR PLOT**

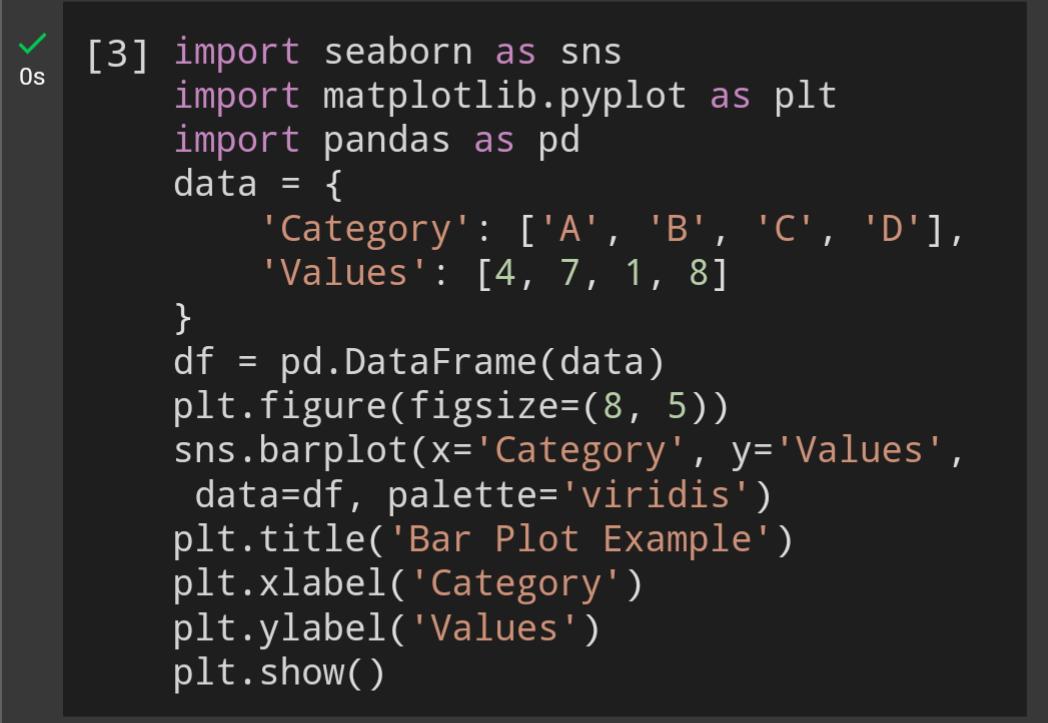
Creating a bar plot using Seaborn is straightforward. Below is a step-by-step guide along with sample codeStart by importing the necessary libraries, including Seaborn and Matplotlib.Use Seaborn’s barplot function to create the plot.sns.barplot(): This function creates the bar plot. The x parameter specifies the category, and the y parameter specifies the values to be plotted.

**PURPOSE OF BAR PLOTS:**

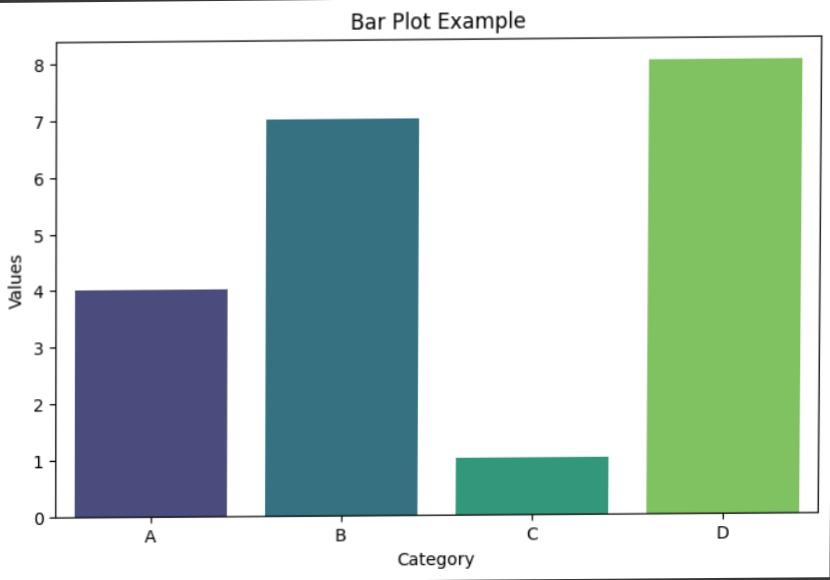
**Displaying Aggregated Data:** They are useful for displaying aggregated data, such as sums or averages, derived from raw data. For example, they can show the total sales for different product categories.

**Easy Interpretation:** Bar plots are straightforward and easy to interpret, making them a popular choice for presentations and reports where clarity is essential.

**CODE:**

****

**OUTPUT:**

****

**COUNT PLOT**

A count plot in Seaborn is used to show the counts of observations in each categorical bin using bars. It’s a useful tool to visualize the frequency distribution of categorical data. You can create a count plot using the seaborn.countplot() function.x=’day’: Specifies the categorical variable for the x-axis (in this case, the day of the week). Here is the explanation of how create count plot

Import seaborn library

Data=tips: Specifies the dataset to use.

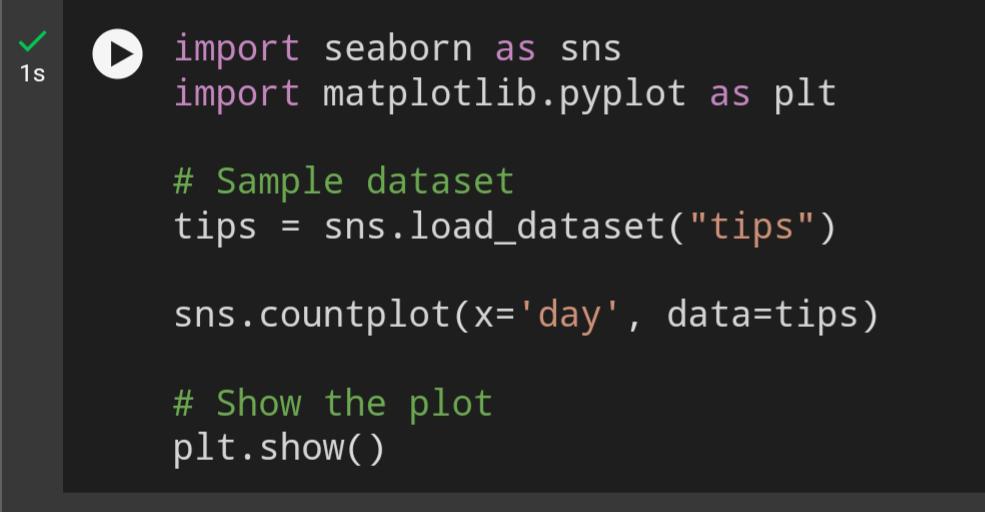
Plt.show(): Displays the plot.

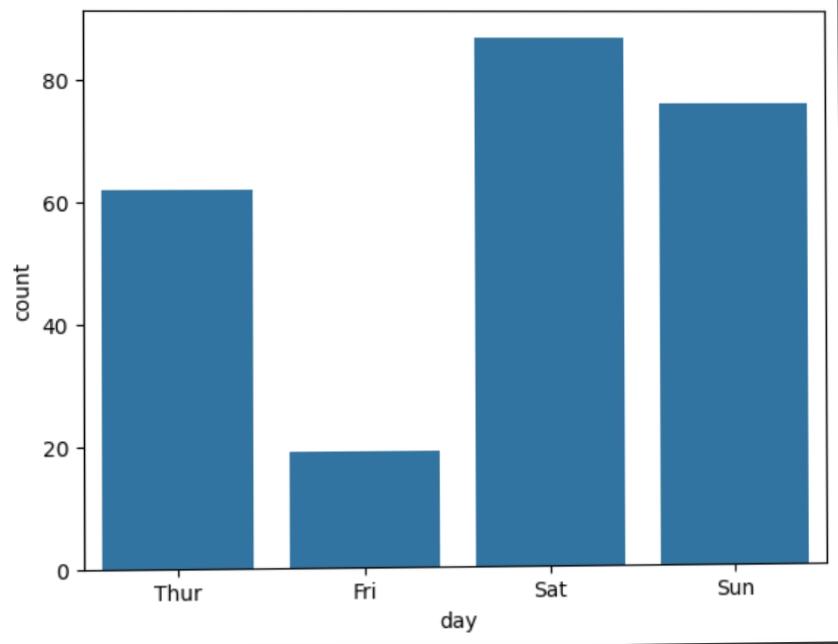
You can also customize the count plot by adding hue, adjusting colors, and changing axis labels.

**PURPOSE OF COUNT PLOT:**

**Comparison between groups:** When using hue or color distinction, a count plot can compare how different subgroups (e.g., male vs. female) behave within each category.

**Spotting imbalances:** It can reveal imbalances in the dataset, such as if some categories are significantly underrepresented.

**CODE:**

**OUTPUT:**

**BOX PLOT**

A box plot (or box-and-whisker plot) in Seaborn is a type of plot that provides a visual summary of the distribution of a dataset. It shows the median, quartiles, and potential outliers. Box plots are especially useful for comparing distributions across multiple categories or groups. Here is the explanation of how to create Box polt

Import seaborn library

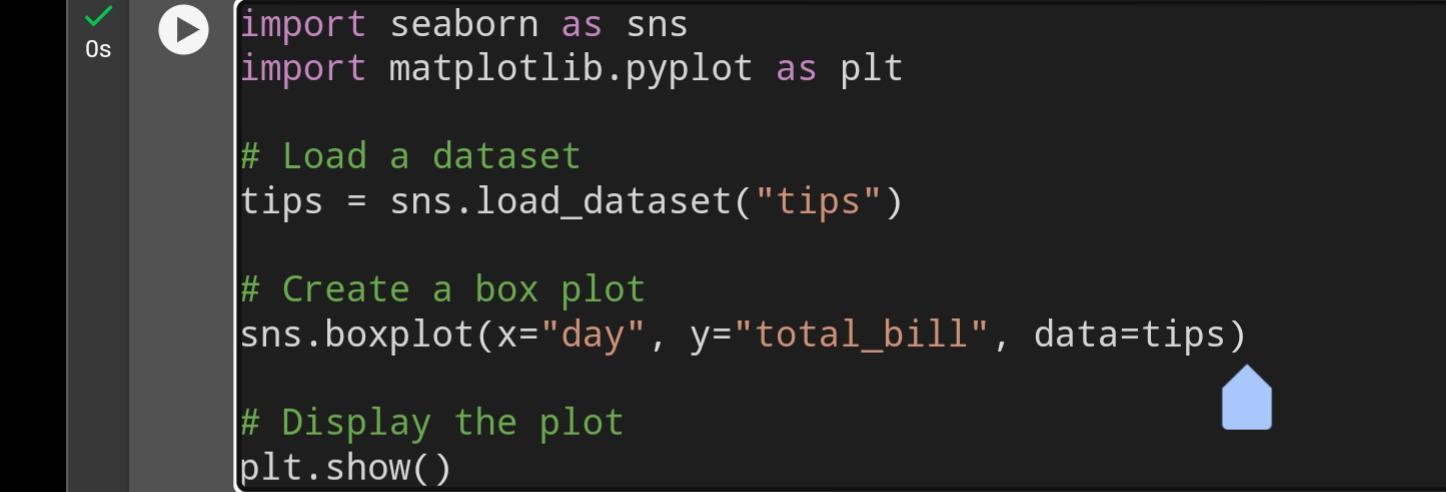
X and y: Define the categorical and numerical variables

showfliers=False: You can choose to hide the outliers.

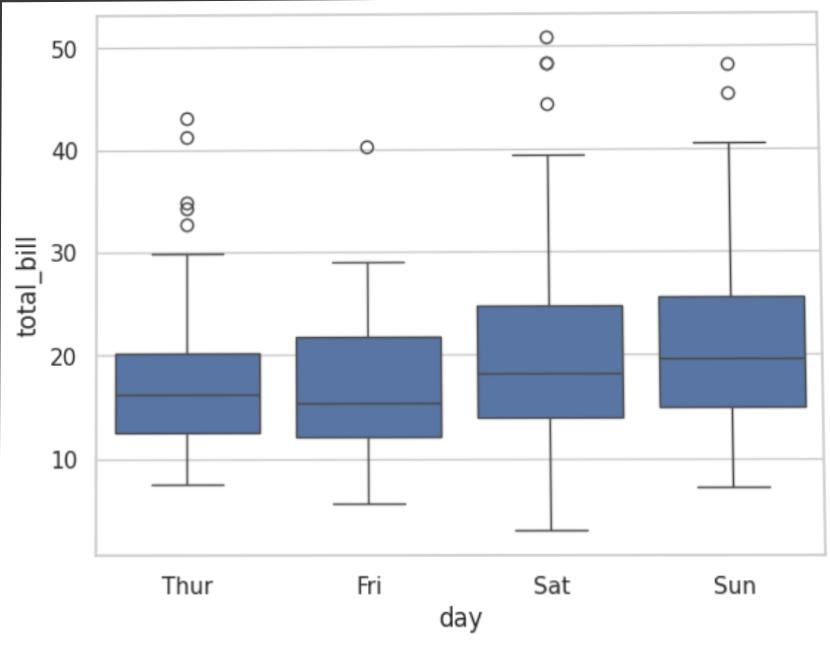
**PURPOSE OF BOX PLOT:**

**Summarizing data distribution:** Box plots display the range, quartiles, and median, providing a concise visual representation of the data’s spread.

**Spotting outliers:** Outliers are plotted as individual points outside the whiskers, making them easy to detect.

**CODE:**

**OUTPUT:**

****

**COMPARISON OF MATPLOT LIBRARY AND SEABORN**

**MATPLOT LIBRARY:**

It is a Python library used for plotting graphs with the help of other libraries like Numpy and Pandas. It is a powerful tool for visualizing data in Python. It is used for creating statistical inferences and plotting 2D graphs of arrays. It was first introduced by John D. Hunter in 2002. It uses Pyplot to provide a MATLAB-like interface free and open-source. It is capable of dealing with various operating systems and their graphical backends.

**SEABORN LIBRARY:**

It is also a Python library used for plotting graphs with the help of Matplotlib, Pandas, and Numpy. It is built on the roof of Matplotlib and is considered as a superset of the Matplotlib library. It helps in visualizing univariate and bivariate data. It uses beautiful themes for decorating Matplotlib graphics. It acts as an important tool in picturing Linear Regression Models. It serves in making graphs of statical Time-Series data. It eliminates the overlapping of graphs and also aids in their beautification.

**Here’s a comparison between \*\*Matplotlib\*\* and \*\*Seaborn\*\* based on ease of use, customization options, interactivity, and performance with large datasets:**

**1. \*\*Ease of Use\*\***

**- \*\*Matplotlib\*\*:**

- It is a low-level library that provides fine control over every aspect of a plot, but this can make it more complex and require more code to generate basic plots.

- Its syntax can feel verbose, requiring multiple steps to achieve even simple visualizations.

**- \*\*Seaborn\*\*:**

- Built on top of Matplotlib, Seaborn simplifies many aspects of plotting by offering high-level commands for more complex statistical plots.

- It is more intuitive and easier to use for creating visually attractive plots with less code, particularly for statistical graphics.

**2. \*\*Customization Options\*\***

**- \*\*Matplotlib\*\*:**

- It provides extensive customization options, allowing control over every visual aspect of the plot (axes, labels, ticks, colors, markers, etc.).

- You can customize plots at a very granular level, which is powerful but can lead to more complicated code.

**- \*\*Seaborn\*\*:**

- While it is easier to use, Seaborn’s customization is somewhat limited compared to Matplotlib.

- Seaborn can handle some customization, but for advanced or specific tweaks, users often need to switch back to Matplotlib functions for more granular control.

**3. \*\*Interactivity\*\***

**- \*\*Matplotlib\*\*:**

- It supports basic interactivity through functions like zooming, panning, and tooltips using its `plt.show()` or by integrating with interactive environments (like Jupyter notebooks).

- Additional libraries such as `mplcursors`, `ipympl`, and `matplotlib.widgets` can enhance interactivity.

**- \*\*Seaborn\*\*:**

- Since it is built on Matplotlib, Seaborn inherits the same interactivity options.

- Seaborn itself doesn’t add new interactivity options beyond what Matplotlib offers.

**4. \*\*Performance with Large Datasets\*\***

**- \*\*Matplotlib\*\***:

- Performance with large datasets can be a concern. Matplotlib can handle a decent amount of data, but as the dataset grows, it may slow down.

- Using `agg` backends or simplifying plots (reducing detail or limiting markers) can help improve performance.

**- \*\*Seaborn\*\*:**

- Since Seaborn is built on Matplotlib, its performance with large datasets is similar. However, some Seaborn functions, particularly statistical plots (like `sns.pairplot`), can become slow with very large datasets due to the complexity of the calculations involved.

- For large datasets, Seaborn’s performance might degrade faster than Matplotlib’s due to the added complexity of Seaborn’s statistical routines.

**Summary:**

**Ease of Use:** Seaborn is easier to use, especially for complex statistical plots, while Matplotlib requires more manual work.

**Customizatio:** Matplotlib offers more extensive customization than Seaborn.

**Interactivity:** Both libraries are similar in interactivity, as Seaborn relies on Matplotlib’s capabilities.

**Performance with Large Datasets:** Both libraries face performance challenges with very large datasets, though Matplotlib may handle simpler visualizations of large datasets slightly better.

Ultimately, Seaborn is great for quick, aesthetically pleasing statistical plots, while Matplotlib is more versatile for complex, highly customized visualizations.