**Visualization**

In data science, visualization refers to the graphical representation of information and data. By using visual elements like charts, graphs, maps, and plots, data visualization tools make it easier to:

1. Understand complex data
2. Identify patterns and trends
3. Spot outliers or anomalies
4. Communicate results effectively to others

**Common Data Visualization Types:**

1. **Bar Chart** – compare quantities of different categories.
2. **Line Graph** – show trends over time.
3. **Pie Chart** – show proportions or percentages.
4. **Histogram** – show frequency distributions.
5. **Scatter Plot** – show relationships or correlations.
6. **Heatmap** – show magnitude using color.

Python offers several libraries for this purpose, with **Matplotlib** and **Seaborn** being two of the most popular choices.

**1. Matplotlib**

Matplotlib is a foundational Python library for creating 2D plots and graphs. It was developed in 2002 by John Hunter and works well with NumPy arrays and Pandas data.

**Key Components:**

* Figure: A Figure is the container in which plots are present. It can have a single plot or multiple plots.
* Axes: An Axes is an Artist attached to a Figure that contains a region for plotting data, and usually includes two (or three in the case of 3D) Axis objects (be aware of the difference between Axes and Axis) that provide ticks and tick labels to provide scales for the data in the Axes. Each Axes also has a title (set via set\_title()), an x-label (set via set\_xlabel()), and a y-label set via set\_ylabel()).
* Axis: These objects set the scale and limits and generate ticks (the marks on the Axis) and ticklabels (strings labeling the ticks). The location of the ticks is determined by a Locator object and the ticklabel strings are formatted by a Formatter. The combination of the correct Locator and Formatter gives very fine control over the tick locations and labels.
* Artist: Basically, everything visible on the Figure is an Artist (even Figure, Axes, and Axis objects). This includes Text objects, Line2D objects, collections objects, Patch objects, etc. When the Figure is rendered, all of the Artists are drawn to the canvas. Most Artists are tied to an Axes; such an Artist cannot be shared by multiple Axes, or moved from one to another.

**Pyplot:**

Pyplot is a sub-library of Matplotlib. It contains functions for creating different types of plots, such as line plots, bar charts, scatter plots, and histograms. Pie chart, Area charts

We import pyplot from Matplotlib like this:

import matplotlib.pyplot as plt

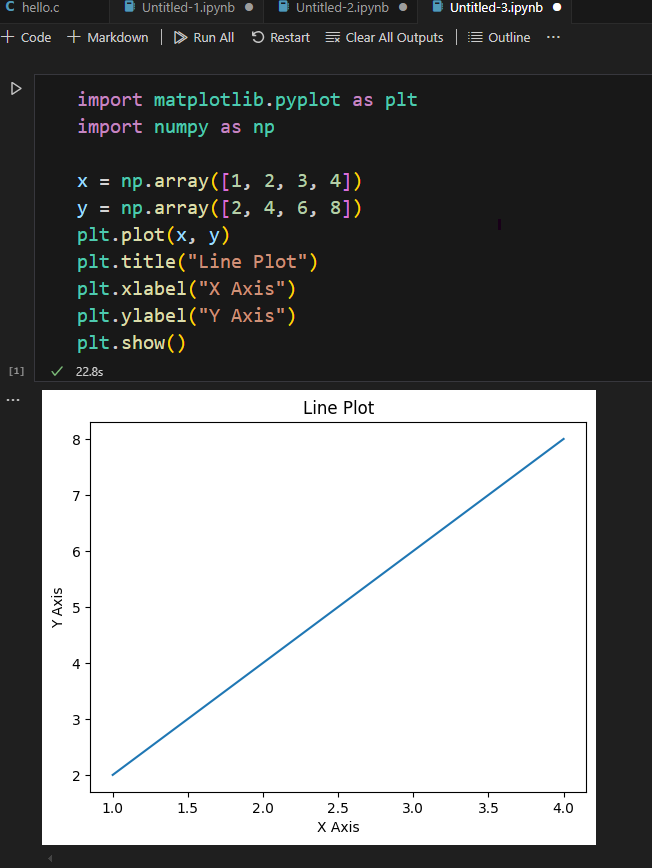
We also use NumPy to generate data:

import numpy as np

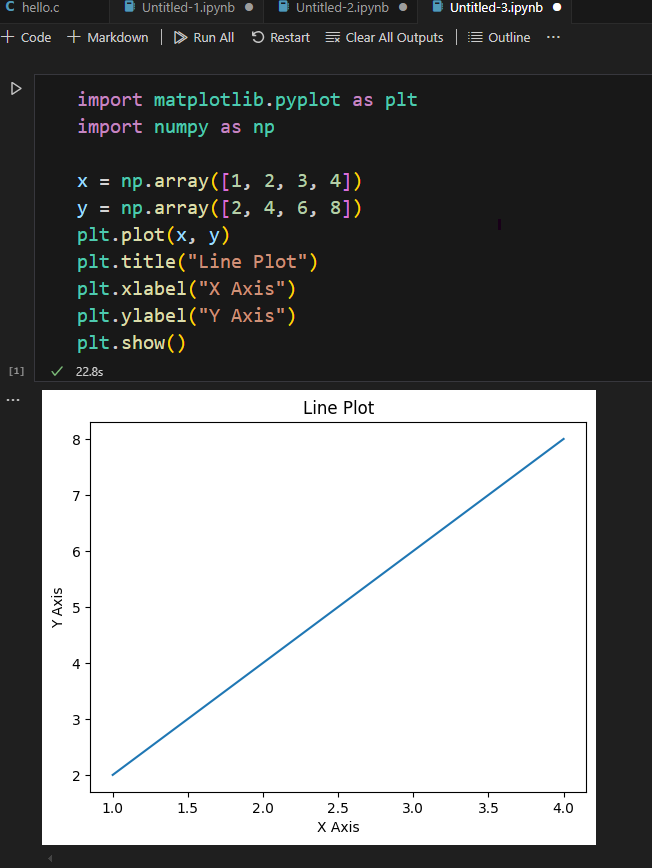
1. **Line Chart:**

Line plots connect individual data points with lines.

Code:



Output:



**Code Description**:

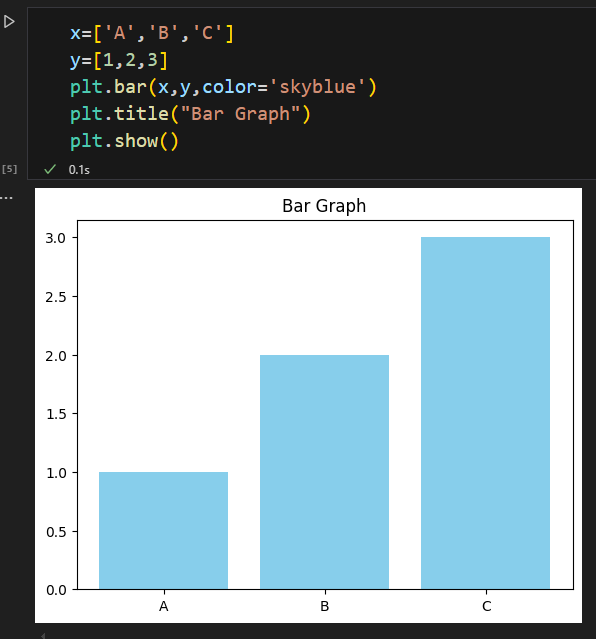
* x and y are data points.
* plt.plot() draws a line connecting the points.
* plt.title() sets the title.
* plt.xlabel() and plt.ylabel() label the axes.
* plt.show() displays the plot

**Use case**: Tracking values over time (e.g., stock prices).

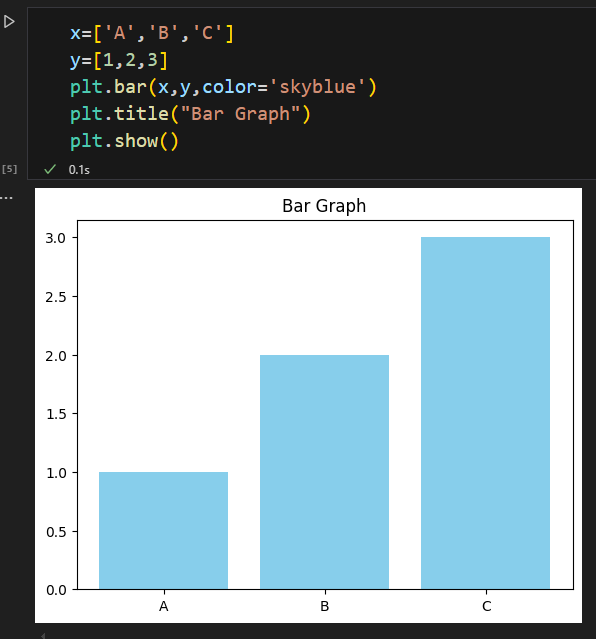
1. **Bar Chart:**

Bar charts represent data using rectangular bars.

**Code :**



**Output:**



**Code Description:**

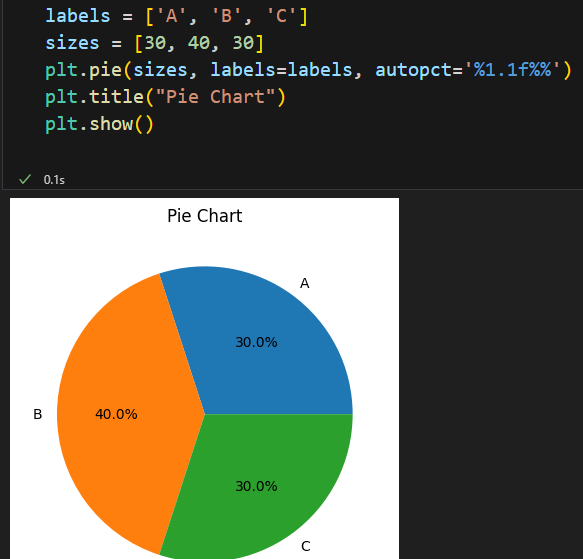
* plt.bar() creates a bar chart using categories A, B, and C.
* color='skyblue' changes the bar color.
* Titles and display follow the same pattern.

**Use case:** Comparing categories (e.g., sales by product).

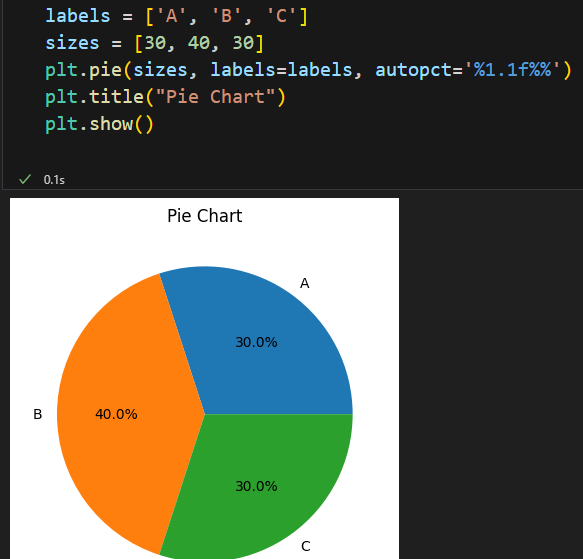
1. **Pie chart:**

A pie chart is a circular statistical graphic that visually represents data proportions using slices of a circle. Each slice corresponds to a category or data point, and its size is proportional to the quantity it represents.

**Code :**



**Output:**



**Code Description**:

* plt.pie() creates a pie chart.
* labels names each slice.
* autopct shows percentages on the slices.

**Use case**: Market share distribution.

**2. Seaborn**

**Seaborn** is a high-level visualization library built on top of Matplotlib. It comes with built-in themes, color palettes, and statistical plotting features.

We import Seaborn like this:

**import seaborn as sns**

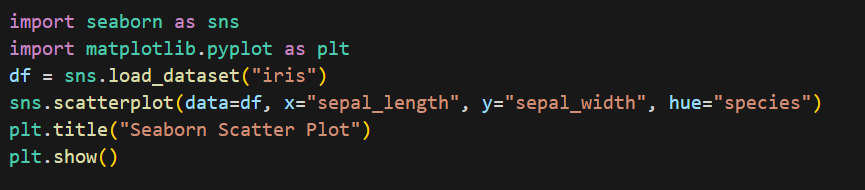
**import matplotlib.pyplot as plt**

It integrates well with **Pandas DataFrames** and is ideal for exploratory data analysis.

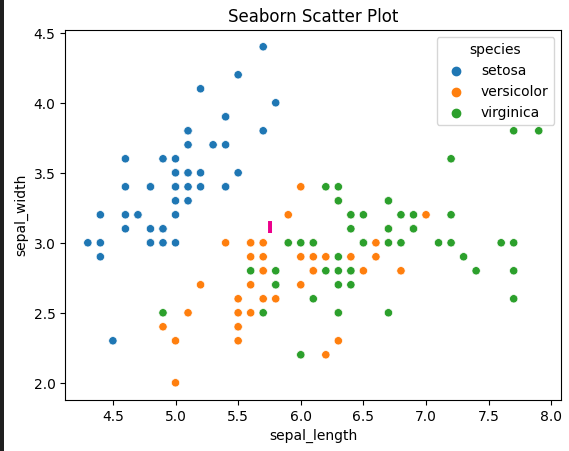
**1.scatter plot (Seaborn):**

A scatter plot, also known as a scatter chart or scatter graph, is a data visualization technique that uses dots to represent values for two different numeric variables.

**Code:**

****

**Output:**

****

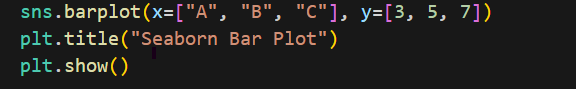
**Code Description:**

* Loads the Iris dataset.
* scatterplot() draws points colored by species.
* hue automatically colors based on categories.

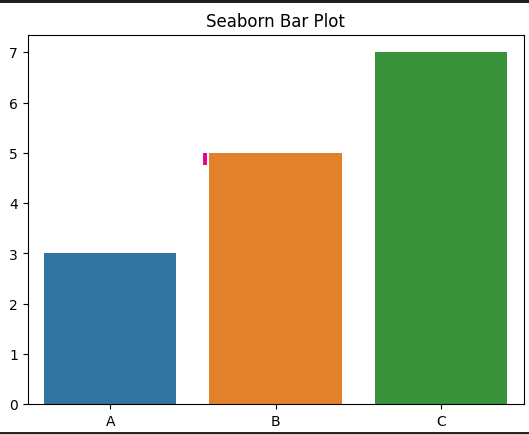
**Use case**: Visualizing clusters in data.

2. **Bar plot**

**Code:**



**Output:**



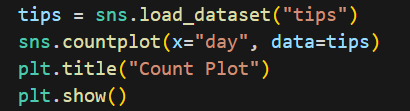
**Code Description**:

* A simple bar plot with labels and values.
* sns.barplot() provides automatic error bars (optional).

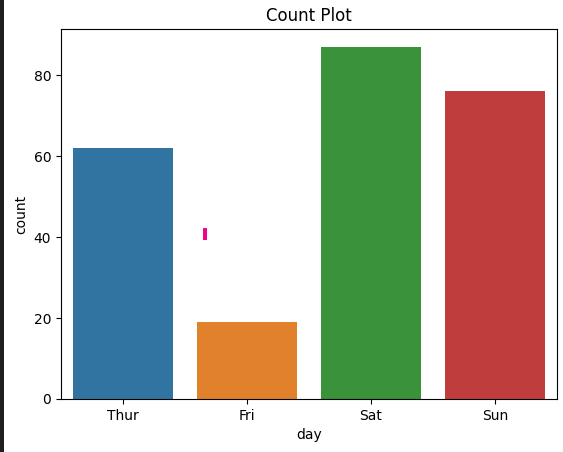
3. **Count plot**

A count plot visualizes the distribution of a categorical variable by displaying the count of observations within each category. It is effectively a histogram for categorical data, where each bar represents a category and its height corresponds to the number of occurrences of that category in the dataset.

**Code:**

****

**Output:**

****

**Code Description:**

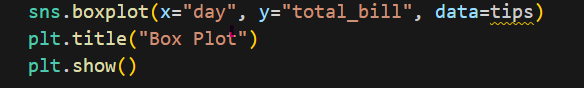
* countplot() counts how many times each day appears in the dataset.
* Useful for categorical frequency.

**Use case:** Count of observations in each category.

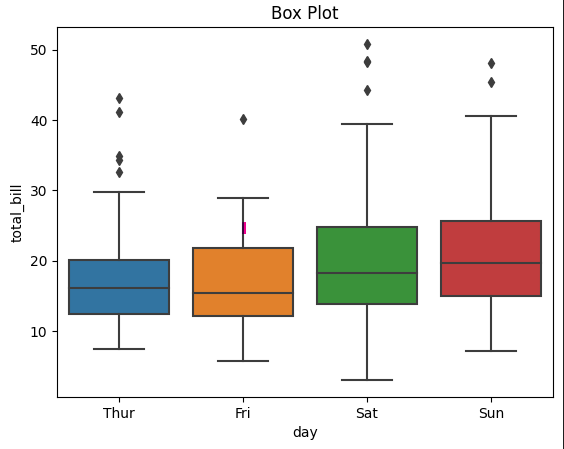
**4.** **Box plot**

A boxplot is a standardized way of displaying the dataset based on the five-number summary: the minimum, the maximum, the sample median, and the first and third quartiles.

**Code:**

****

**Output:**

****

**Code Description:**

* Displays distribution, median, and outliers of total bills by day.
* Ideal for comparing multiple groups.

**Use case:** Checking distribution and outliers.

**Comparison: Matplotlib vs Seaborn**

|  |  |  |
| --- | --- | --- |
| Feature | Matplotlib | Seaborn |
| Level | **Low-level (manual control)** | **High-level (automatic styling)** |
| Syntax | **More code needed** | **Simpler & more readable** |
| Aesthetics | **Basic, customizable** | **Modern, beautiful defaults** |
| Integration | **NumPy, Pandas** | **Pandas-native** |
| Plot Types | **Basic (line, bar, scatter, etc.)** | **Advanced (box, violin, heatmap, etc.)** |
| Interactivity | **Basic** | **Limited (use with Matplotlib/Plotly)** |
| Best For | **Custom plots, fine control** | **Statistical visualizations, EDA** |