# **Data Visualization with Python**

In today’s world, a lot of data is being generated on a daily basis. And sometimes to analyze this data for certain trends, patterns may become difficult if the data is in its raw format. To overcome this data visualization comes into play. Data visualization provides a good, organized pictorial representation of the data which makes it easier to understand, observe, analyze. In this tutorial, we will discuss how to visualize data using Python.

Python provides various libraries that come with different features for visualizing data. All these libraries come with different features and can support various types of graphs. In this tutorial, we will be discussing four such libraries.

* Matplotlib
* Seaborn
* Bokeh
* Plotly

[tips database](https://media.geeksforgeeks.org/wp-content/uploads/tips.csv)

## **Database Used**

### **Tips Database**

Tips database is the record of the tip given by the customers in a restaurant for two and a half months in the early 1990s. It contains 6 columns such as total\_bill, tip, sex, smoker, day, time, size.

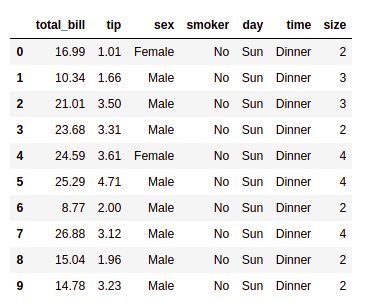
You can download the tips database from [here](https://media.geeksforgeeks.org/wp-content/uploads/tips.csv).

**Example:**

* Python3

| import pandas as pd      # reading the database  data = pd.read\_csv("tips.csv")    # printing the top 10 rows  display(data.head(10)) |
| --- |

**Output:**

****

## **Matplotlib**

### **Matplotlib is an easy-to-use, low-level data visualization library that is built on NumPy arrays. It consists of various plots like scatter plot, line plot, histogram, etc. Matplotlib provides a lot of flexibility.**

### 

### **Scatter Plot**

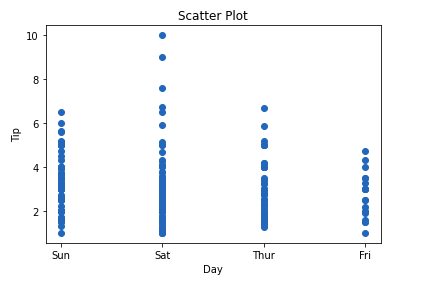
Scatter plots are used to observe relationships between variables and uses dots to represent the relationship between them. The [**scatter()**](https://www.geeksforgeeks.org/matplotlib-pyplot-scatter-in-python/) method in the matplotlib library is used to draw a scatter plot.

**Example:**

* Python3

| import pandas as pd  import matplotlib.pyplot as plt      # reading the database  data = pd.read\_csv("tips.csv")    # Scatter plot with day against tip  plt.scatter(data['day'], data['tip'])    # Adding Title to the Plot  plt.title("Scatter Plot")    # Setting the X and Y labels  plt.xlabel('Day')  plt.ylabel('Tip')    plt.show() |
| --- |
|  |

**Output:**

****

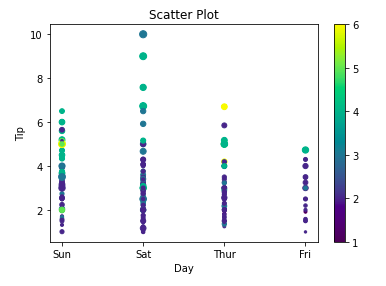
This graph can be more meaningful if we can add colors and also change the size of the points. We can do this by using the **c and s** parameterrespectivelyof the scatter function. We can also show the color bar using the [colorbar()](https://www.geeksforgeeks.org/matplotlib-pyplot-colorbar-function-in-python/) method.

**Example:**

* Python3

| import pandas as pd  import matplotlib.pyplot as plt      # reading the database  data = pd.read\_csv("tips.csv")    # Scatter plot with day against tip  plt.scatter(data['day'], data['tip'], c=data['size'],  s=data['total\_bill'])    # Adding Title to the Plot  plt.title("Scatter Plot")    # Setting the X and Y labels  plt.xlabel('Day')  plt.ylabel('Tip')    plt.colorbar()    plt.show() |
| --- |
|  |

**Output:**

****

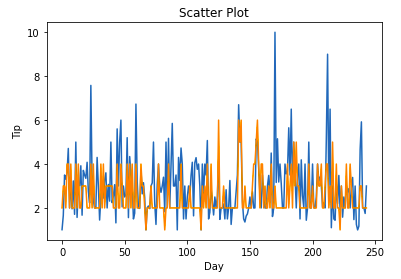
### **Line Chart**

[**Line Chart**](https://www.geeksforgeeks.org/line-chart-in-matplotlib-python/) **is used to represent a relationship between two data X and Y on a different axis. It is plotted using the plot() function. Let’s see the below example.**

**Example:**

| **import pandas as pd**  **import matplotlib.pyplot as plt**      **# reading the database**  **data = pd.read\_csv("tips.csv")**    **# Scatter plot with day against tip**  **plt.plot(data['tip'])**  **plt.plot(data['size'])**    **# Adding Title to the Plot**  **plt.title("Scatter Plot")**    **# Setting the X and Y labels**  **plt.xlabel('Day')**  **plt.ylabel('Tip')**    **plt.show()** |
| --- |

**Output:**

****

### **Bar Chart**

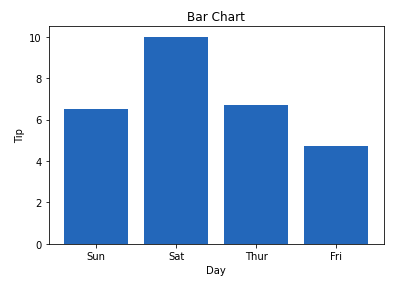
A [bar plot](https://www.geeksforgeeks.org/bar-plot-in-matplotlib/) or bar chart is a graph that represents the category of data with rectangular bars with lengths and heights that is proportional to the values which they represent. It can be created using the **bar()** method.

**Example:**

* Python3

| import pandas as pd  import matplotlib.pyplot as plt      # reading the database  data = pd.read\_csv("tips.csv")    # Bar chart with day against tip  plt.bar(data['day'], data['tip'])    plt.title("Bar Chart")    # Setting the X and Y labels  plt.xlabel('Day')  plt.ylabel('Tip')    # Adding the legends  plt.show() |
| --- |

**Output:**

****

### **Histogram**

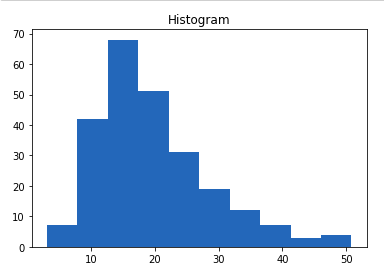
A [histogram](https://www.geeksforgeeks.org/plotting-histogram-in-python-using-matplotlib/) is basically used to represent data in the form of some groups. It is a type of bar plot where the X-axis represents the bin ranges while the Y-axis gives information about frequency. The [**hist()**](https://www.geeksforgeeks.org/matplotlib-pyplot-hist-in-python/) function is used to compute and create a histogram. In histogram, if we pass categorical data then it will automatically compute the frequency of that data i.e. how often each value occurred.

**Example:**

* Python3

| import pandas as pd  import matplotlib.pyplot as plt      # reading the database  data = pd.read\_csv("tips.csv")    # histogram of total\_bills  plt.hist(data['total\_bill'])    plt.title("Histogram")    # Adding the legends  plt.show() |
| --- |

**Output:**

****

**Seaborn** is a high-level interface built on top of the Matplotlib. It provides beautiful design styles and color palettes to make more attractive graphs.

To install seaborn type the below command in the terminal.

Seaborn is built on the top of Matplotlib, therefore it can be used with the Matplotlib as well. Using both Matplotlib and Seaborn together is a very simple process. We just have to invoke the Seaborn Plotting function as normal, and then we can use Matplotlib’s customization function.

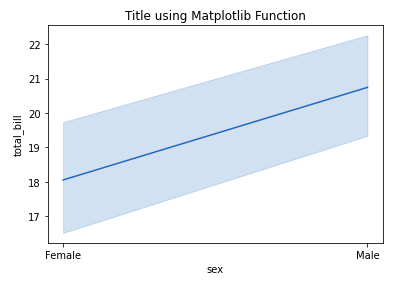
**Note:** Seaborn comes loaded with dataset such as tips, iris, etc. but for the sake of this tutorial we will use Pandas for loading these datasets.

**Example:**

* Python3

| # importing packages  import seaborn as sns  import matplotlib.pyplot as plt  import pandas as pd      # reading the database  data = pd.read\_csv("tips.csv")    # draw lineplot  sns.lineplot(x="sex", y="total\_bill", data=data)    # setting the title using Matplotlib  plt.title('Title using Matplotlib Function')    plt.show() |
| --- |

**Output:**

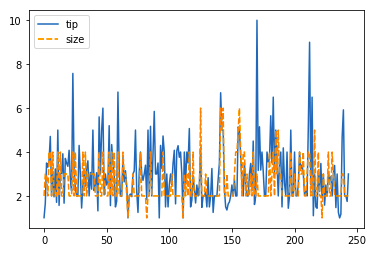
****

**Example 2:**

* Python3

| # importing packages  import seaborn as sns  import matplotlib.pyplot as plt  import pandas as pd      # reading the database  data = pd.read\_csv("tips.csv")    # using only data attribute  sns.lineplot(data=data.drop(['total\_bill'], axis=1))  plt.show() |
| --- |

**Output:**

****

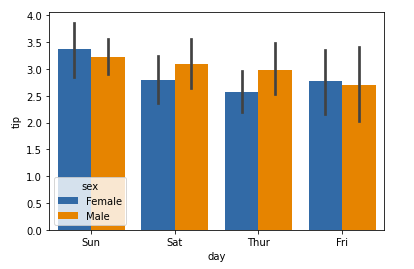
### **Bar Plot**

[Bar Plot](https://www.geeksforgeeks.org/barplot-using-seaborn-in-python/) in Seaborn can be created using the [**barplot()**](https://www.geeksforgeeks.org/seaborn-barplot-method-in-python/) method.

**Example:**

| # importing packages  import seaborn as sns  import matplotlib.pyplot as plt  import pandas as pd      # reading the database  data = pd.read\_csv("tips.csv")    sns.barplot(x='day',y='tip', data=data,  hue='sex')    plt.show() |
| --- |

**Output:**

****

### **Histogram**

The histogram in Seaborn can be plotted using the **histplot()** function.

**Example:**

| # importing packages  import seaborn as sns  import matplotlib.pyplot as plt  import pandas as pd      # reading the database  data = pd.read\_csv("tips.csv")    sns.histplot(x='total\_bill', data=data, kde=True, hue='sex')    plt.show()  A kernel density estimate (KDE) plot is a method for visualizing the distribution of observations in a dataset, analogous to a histogram. KDE represents the data using a continuous probability density curve in one or more dimensions.  The approach is explained further in the [user guide](https://seaborn.pydata.org/tutorial/distributions.html#tutorial-kde).  Relative to a histogram, KDE can produce a plot that is less cluttered and more interpretable, especially when drawing multiple distributions. But it has the potential to introduce distortions if the underlying distribution is bounded or not smooth. Like a histogram, the quality of the representation also depends on the selection of good smoothing parameters. |
| --- |

**Output:**

[**https://www.geeksforgeeks.org/data-visualization-with-python/**](https://www.geeksforgeeks.org/data-visualization-with-python/)