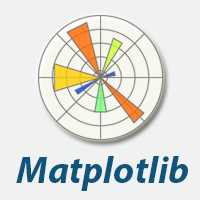
<https://www.w3schools.com/python/matplotlib_intro.asp>

<https://matplotlib.org/stable/tutorials/index.html>

<https://www.geeksforgeeks.org/matplotlib-tutorial/>

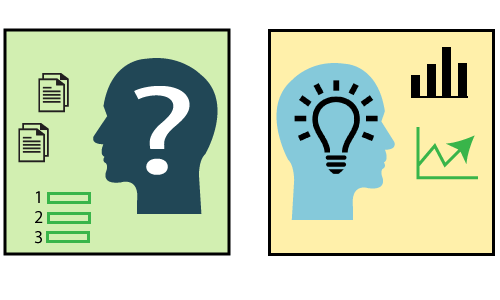
<https://www.tutorialspoint.com/matplotlib/index.htm>

# Matplotlib (Python Plotting Library)



Human minds are more adaptive for the visual representation of data rather than textual data. We can easily understand things when they are visualized. It is better to represent the data through the graph where we can analyze the data more efficiently and make the specific decision according to data analysis. Before learning the matplotlib, we need to understand data visualization and why data visualization is important.

## Data Visualization

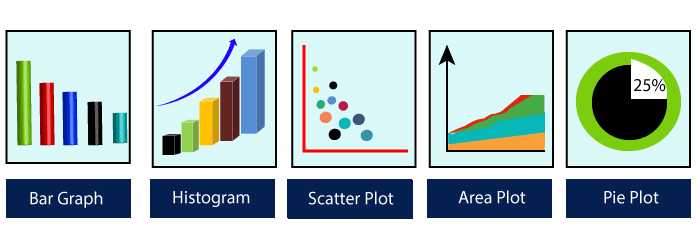


Graphics provides an excellent approach for exploring the data, which is essential for presenting results. Data visualization is a new term. It expresses the idea that involves more than just representing data in the graphical form (instead of using textual form).

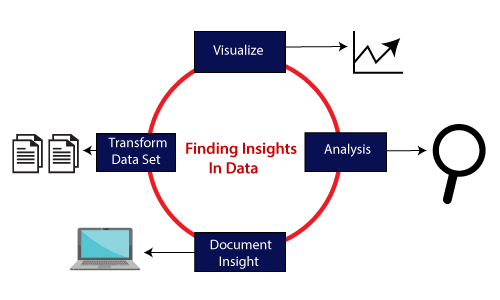
This can be very helpful when discovering and getting to know a dataset and can help with classifying patterns, corrupt data, outliers, and much more. With a little domain knowledge, data visualizations can be used to express and demonstrate key relationships in plots and charts. The static does indeed focus on quantitative description and estimations of data. It provides an important set of tools for gaining a qualitative understanding.

There are five key plots that are used for data visualization.

Play Videox[](https://campaign.adpushup.com/get-started/?utm_source=banner%26utm_campaign=growth_hack)

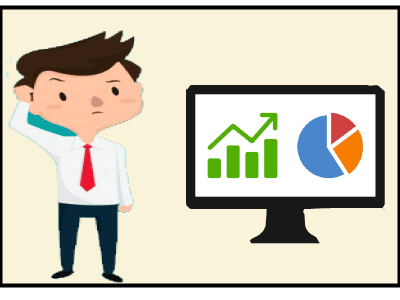


There are five phases which are essential to make the decision for the organization:



* ****Visualize:**** We analyze the raw data, which means it makes complex data more accessible, understandable, and more usable. Tabular data representation is used where the user will look up a specific measurement, while the chart of several types is used to show patterns or relationships in the data for one or more variables.
* ****Analysis:**** Data analysis is defined as cleaning, inspecting, transforming, and modeling data to derive useful information. Whenever we make a decision for the business or in daily life, is by past experience. ****What will happen to choose a particular decision****, it is nothing but analyzing our past. That may be affected in the future, so the proper analysis is necessary for better decisions for any business or organization.
* ****Document Insight:**** Document insight is the process where the useful data or information is organized in the document in the standard format.
* ****Transform Data Set:**** Standard data is used to make the decision more effectively.

## Why need data visualization?



Data visualization can perform below tasks:

* It identifies areas that need improvement and attention.
* It clarifies the factors.
* It helps to understand which product to place where.
* Predict sales volumes.

## Benefit of Data Visualization

Here are some benefits of the data visualization, which helps to make an effective decision for the organizations or business:

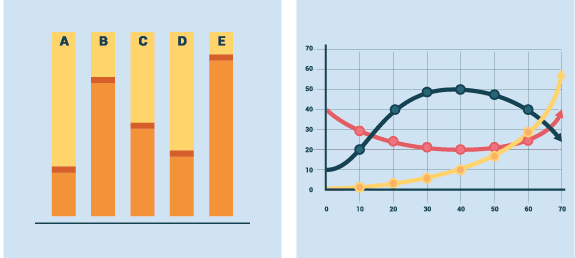
****1. Building ways of absorbing information****

Data visualization allows users to receive vast amounts of information regarding operational and business conditions. It helps decision-makers to see the relationship between multi-dimensional data sets. It offers new ways to analyses data through the use of maps, fever charts, and other rich graphical representations.

Visual data discovery is more likely to find the information that the organization needs and then end up with being more productive than other competitive companies.

****2. Visualize relationship and patterns in Businesses****

The crucial advantage of data visualization is that it is essential to find the correlation between operating conditions and business performance in today's highly competitive business environment.



The ability to make these types of correlations enables the executives to identify the root cause of the problem and act quickly to resolve it.

Suppose a food company is looking their monthly customer data, and the data is presented with bar charts, which shows that the company's score has dropped by five points in the previous months in that particular region; the data suggest that there's a problem with customer satisfaction in this area.

****3. Take action on the emerging trends faster****

Data visualization allows the decision-maker to grasp shifts in customer behavior and market conditions across multiple data sets more efficiently.

Having an idea about the customer's sentiments and other data discloses an emerging opportunity for the company to act on new business opportunities ahead of their competitor.

****4. Geological based Visualization****

Geo-spatial visualization is occurred due to many websites providing web-services, attracting visitor's interest. These types of websites are required to take benefit of location-specific information, which is already present in the customer details.

****Matplotlib**** is a Python library which is defined as a multi-platform data visualization library built on Numpy array. It can be used in python scripts, shell, web application, and other graphical user interface toolkit.

The ****John D. Hunter**** originally conceived the matplotlib in ****2002****. It has an active development community and is distributed under a ****BSD-style license****. Its first version was released in 2003, and the latest ****version 3.1.1**** is released on ****1 July 2019****.

Matplotlib 2.0.x supports Python versions 2.7 to 3.6 till 23 June 2007. Python3 support started with Matplotlib 1.2. Matplotlib 1.4 is the last version that supports Python 2.6.

There are various toolkits available that are used to enhance the functionality of the ****matplotlib****. Some of these tools are downloaded separately, others can be shifted with the matplotlib source code but have external dependencies.

* ****Bashmap:**** It is a map plotting toolkit with several map projections, coastlines, and political boundaries.
* ****Cartopy:**** It is a mapping library consisting of object-oriented map projection definitions, and arbitrary point, line, polygon, and image transformation abilities.
* ****Excel tools:**** Matplotlib provides the facility to utilities for exchanging data with Microsoft Excel.
* ****Mplot3d:**** It is used for 3D plots.
* ****Natgrid:**** It is an interface to the Natgrid library for irregular gridding of the spaced data.

## Matplotlib Architecture

There are three different layers in the architecture of the matplotlib which are the following:

* Backend Layer
* Artist layer
* Scripting layer

****Backend layer****

The backend layer is the bottom layer of the figure, which consists of the implementation of the various functions that are necessary for plotting. There are three essential classes from the backend layer ****FigureCanvas****(The surface on which the figure will be drawn), ****Renderer****(The class that takes care of the drawing on the surface), and ****Event****(It handle the mouse and keyboard events).

****Artist Layer****

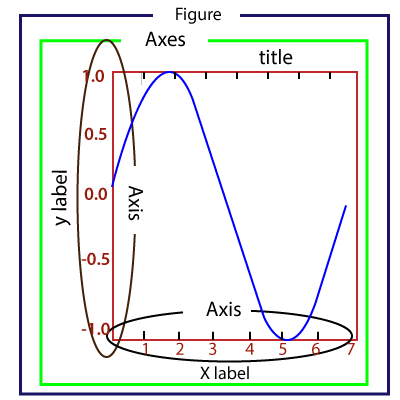
The artist layer is the second layer in the architecture. It is responsible for the various plotting functions, like axis, which coordinates on how to use the renderer on the figure canvas.

****Scripting layer****

The scripting layer is the topmost layer on which most of our code will run. The methods in the scripting layer, almost automatically take care of the other layers, and all we need to care about is the current state (figure & subplot).

## The General Concept of Matplotlib

A Matplotlib figure can be categorized into various parts as below:



****Figure:**** It is a whole figure which may hold one or more axes (plots). We can think of a Figure as a canvas that holds plots.

****Axes:**** A Figure can contain several Axes. It consists of two or three (in the case of 3D) Axis objects. Each Axes is comprised of a title, an x-label, and a y-label.

****Axis:**** Axises are the number of line like objects and responsible for generating the graph limits.

****Artist:**** An artist is the all which we see on the graph like Text objects, Line2D objects, and collection objects. Most Artists are tied to Axes.

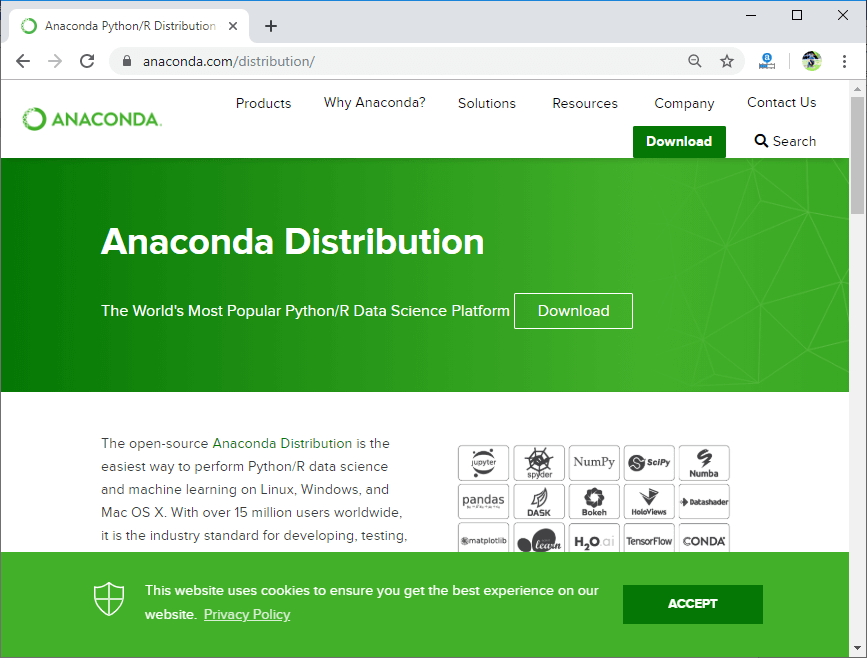
## Installing Matplotlib

Before start working with the Matplotlib or its plotting functions first, it needs to be installed. The installation of matplotlib is dependent on the distribution that is installed on your computer. These installation methods are following:

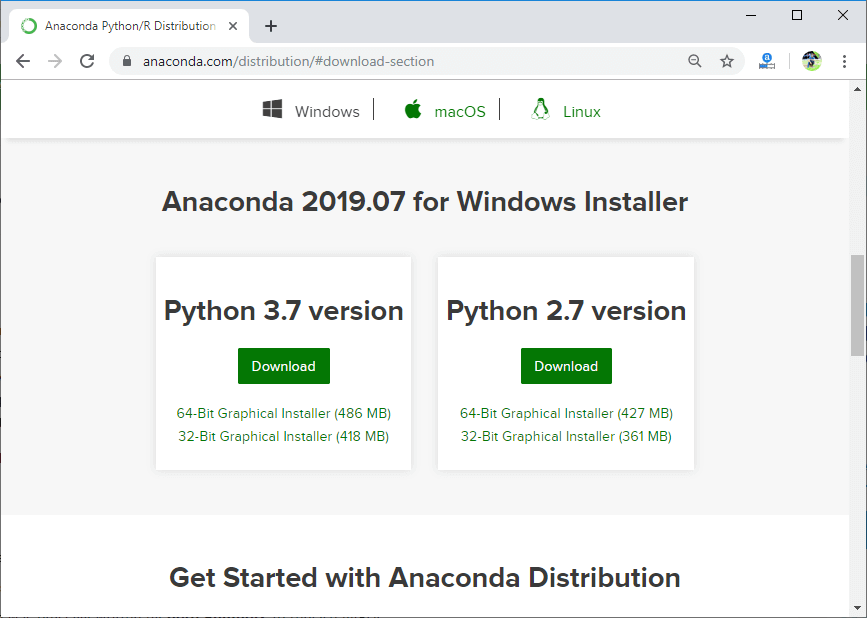
****Use the Anaconda distribution of Python****

The easiest way to install Matplotlib is to download the Anaconda distribution of Python. Matplotlib is pre-installed in the anaconda distribution No further installation steps are necessary.

* Visit the official site of Anaconda and click on the Download Button



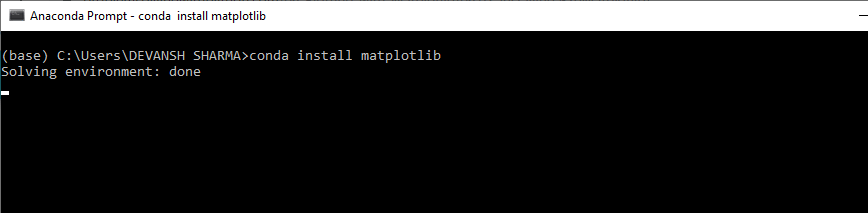
* Choose download according to your Python interpreter configuration.



****Install Matplotlib using with Anaconda Prompt****

Matplotlib can be installed using with the Anaconda Prompt by typing command. To install matplotlib, open Anaconda Prompt and type the following command:

1. conda install matplotlib



****Install Matplotlib with pip****

The python package manager pip is also used to install matplotlib. Open the command prompt window, and type the following command:

1. pip install matplotlib

## Verify the Installation

To verify that matplotlib is installed properly or not, type the following command includes calling .\_\_version \_\_ in the terminal.

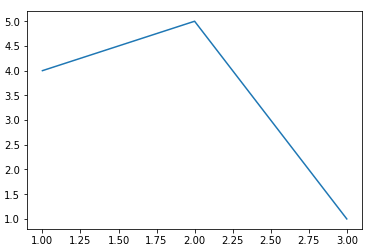
1. **import** matplotlib
2. matplotlib.\_\_version\_\_
3. '3.1.1'

## Basic Example of plotting Graph

Here is the basic example of generating a simple graph; the program is following:

1. from matplotlib **import** pyplot as plt
2. #ploting our canvas
3. plt.plot([1,2,3],[4,5,1])
4. #display the graph
5. plt.show()

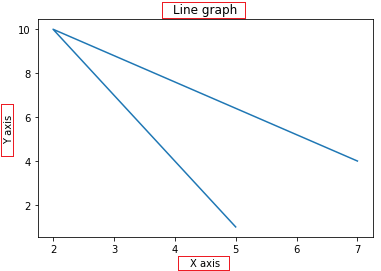
****Output:****



It takes only three lines to plot a simple graph using the Python matplotlib. We can add titles, labels to our chart which are created by Python matplotlib library to make it more meaningful. The example is the following:

1. from matplotlib **import** pyplot as plt
3. x = [5, 2, 7]
4. y = [1, 10, 4]
5. plt.plot(x, y)
6. plt.title('Line graph')
7. plt.ylabel('Y axis')
8. plt.xlabel('X axis')
9. plt.show()

****Output:****



The graph is more understandable from the previous graph.

## Working with Pyplot

The ****matplotlib.pyplot**** is the collection command style functions that make matplotlib feel like working with MATLAB. The pyplot functions are used to make some changes to figure such as create a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot including labels, etc.

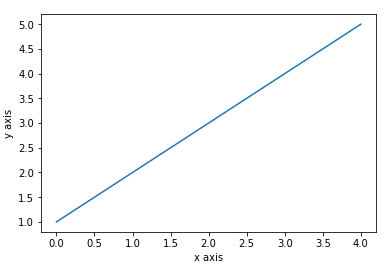
It is good to use when we want to plot something quickly without instantiating any figure or Axes.

While working with ****matplotlib.pyplot****, some states are stored across function calls so that it keeps track of the things like current figure and plotting area, and these plotting functions are directed to the current axes.

The pyplot module provide the ****plot()**** function which is frequently use to plot a graph. Let's have a look on the simple example:

1. from matplotlib **import** pyplot as plt
2. plt.plot([1,2,3,4,5])
3. plt.ylabel("y axis")
4. plt.xlabel('x axis')
5. plt.show()

****Output:****

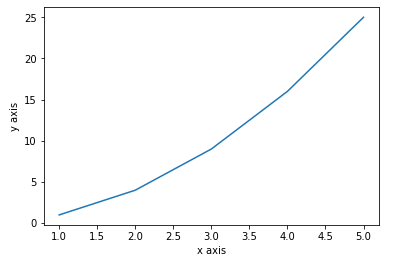


In the above program, it plots the graph x-axis ranges from 0-4 and the y-axis from 1-5. If we provide a single list to the plot(), matplotlib assumes it is a sequence of y values, and automatically generates the x values. Since we know that python index starts at 0, the default x vector has the same length as y but starts at 0. Hence the x data are [0, 1, 2, 3, 4].

We can pass the arbitrary number of arguments to the plot(). For example, to plot x versus y, we can do this following way:

1. from matplotlib **import** pyplot as plt
2. plt.plot([1,2,3,4,5],[1,4,9,16,25])
3. plt.ylabel("y axis")
4. plt.xlabel('x axis')
5. plt.show()

****Output:****

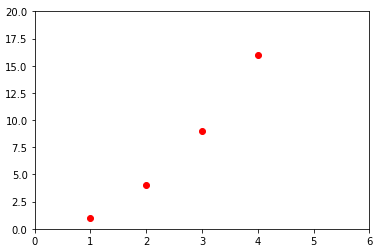


****Formatting the style of the plot****

There is an optional third argument, which is a format string that indicates the color and line type of the plot. The default format string is '****b-****'which is the solid blue as you can observe in the above plotted graph. Let's consider the following example where we plot the graph with the red circle.

1. from matplotlib **import** pyplot as plt
2. plt.plot([1, 2, 3, 4,5], [1, 4, 9, 16,25], 'ro')
3. plt.axis([0, 6, 0, 20])
4. plt.show()

****Output:****



****Example format String****

|  |  |
| --- | --- |
| ****'b'**** | Using for the blue marker with default shape. |
| ****'ro'**** | Red circle |
| ****'-g'**** | Green solid line |
| ****'--'**** | A dashed line with the default color |
| ****'^k:'**** | Black triangle up markers connected by a dotted line |

The matplotlib supports the following color abbreviation:

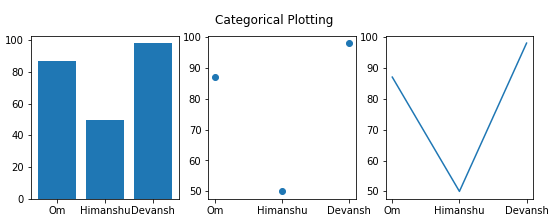
|  |  |
| --- | --- |
| **Character** | **Color** |
| ****'b' Blue**** |  |
| ****'g'**** | Green |
| ****'r'**** | Red |
| ****'c'**** | Cyan |
| ****'m'**** | Magenta |
| ****'y'**** | Yellow |
| ****'k'**** | Black |
| ****'w'**** | White |

### Plotting with categorical variables

Matplotlib allows us to pass categorical variables directly to many plotting functions: consider the following example

1. from matplotlib **import** pyplot
2. names = ['Abhishek', 'Himanshu', 'Devansh']
3. marks= [87,50,98]
5. plt.figure(figsize=(9,3))
7. plt.subplot(131)
8. plt.bar(names, marks)
9. plt.subplot(132)
10. plt.scatter(names, marks)
11. plt.subplot(133)
12. plt.plot(names, marks)
13. plt.suptitle('Categorical Plotting')
14. plt.show()

****Output:****



In the above program, we have plotted the categorical graph using the ****subplot()**** function. Let's a have a look on the subplot() function.

## What is subplot()

The Matplotlib ****subplot()**** function is defined as to plot two or more plots in one figure. We can use this method to separate two graphs which plotted in the same axis Matplotlib supports all kinds of subplots, including 2x1 vertical, 2x1 horizontal, or a 2x2 grid.

It accepts the three arguments: they are ****nrows, ncols, and index****. It denote the number of rows, number of columns and the index.

The subplot() function can be called in the following way:

1. subplot(nrows,ncols,index,\*\*kwargs)
2. subplot(pos,\*\*kwargs)
3. subplot(ax)

****Parameters:****

* ****\*args:****

Three separate integers or three-digit integer describes the position of the subplot. If the three integers are ****nrows, ncols, and index**** in order, the subplot will take the index position on a grid with ****nrows row**** and ****ncol column****.

The argument ****pos**** are a three-digit integer, where the first digit is denoted the number of rows, the second digit denoted the number of columns, and the third represents the index of the subplot. ****For example,**** subplot (1, 3, 2) is the same as the subplot (132).

#### **Note:** Passed integer must be less than 10.

* ****\*\*kwargs****

The subplot() function also accepts the keyword arguments for the returned axes base class.

Consider the following example:

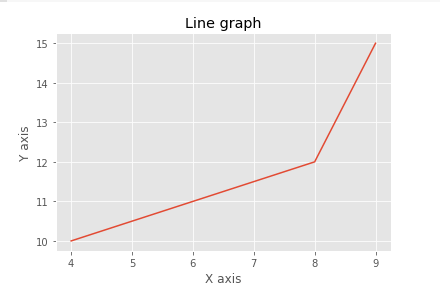
## Creating different types of graph

### 1. Line graph

The line graph is one of charts which shows information as a series of the line. The graph is plotted by the plot() function. The line graph is simple to plot; let's consider the following example:

1. from matplotlib **import** pyplot as plt
3. x = [4,8,9]
4. y = [10,12,15]
6. plt.plot(x,y)
8. plt.title("Line graph")
9. plt.ylabel('Y axis')
10. plt.xlabel('X axis')
11. plt.show()

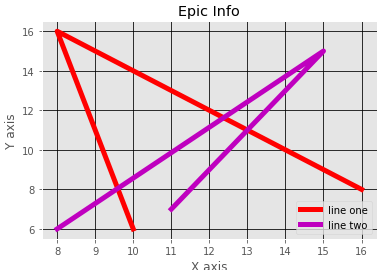
****Output:****



We can customize the graph by importing the style module. The style module will be built into a matplotlib installation. It contains the various functions to make the plot more attractive. In the below program, we are using the style module:

1. from matplotlib **import** pyplot as plt
2. from matplotlib **import** style
4. style.use('ggplot')
5. x = [16, 8, 10]
6. y = [8, 16, 6]
7. x2 = [8, 15, 11]
8. y2 = [6, 15, 7]
9. plt.plot(x, y, 'r', label='line one', linewidth=5)
10. plt.plot(x2, y2, 'm', label='line two', linewidth=5)
11. plt.title('Epic Info')
12. fig = plt.figure()
13. plt.ylabel('Y axis')
14. plt.xlabel('X axis')
15. plt.legend()
16. plt.grid(True, color='k')
17. plt.show()

****Output:****

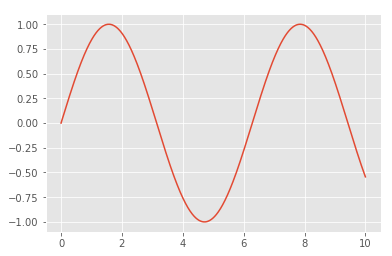


In Matplotlib, the figure (an instance of class plt.Figure) can be supposed of as a single container that consists of all the objects denoting axes, graphics, text, and labels.

****Example-3****

1. **import** numpy as np
2. **import** matplotlib.pyplot as plt
4. fig = plt.figure()
5. ax = plt.axes()
7. x = np.linspace(0, 10, 1000)
8. ax.plot(x, np.sin(x))

****Output:****



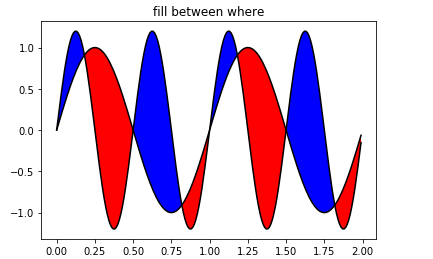
The matplotlib provides the ****fill\_between()**** function which is used to fill area around the lines based on the user defined logic.

### Example-4

1. **import** numpy as np
2. **import** matplotlib.pyplot as plt
4. fig = plt.figure()
5. ax = plt.axes()
7. x = np.linspace(0, 10, 1000)
8. ax.plot(x, np.sin(x))

11. **import** matplotlib.pyplot as plt
12. **import** numpy as np
14. x = np.arange(0.0, 2, 0.01)
15. y1 = np.sin(2 \* np.pi \* x)
16. y2 = 1.2 \* np.sin(4 \* np.pi \* x)
17. fig, ax = plt.subplots(1, sharex=True)
18. ax.plot(x, y1, x, y2, color='black')
19. ax.fill\_between(x, y1, y2, where=y2 >= y1, facecolor='blue', interpolate=True)
20. ax.fill\_between(x, y1, y2, where=y2 <= y1, facecolor='red', interpolate=True)
21. ax.set\_title('fill between where')

****Output:****

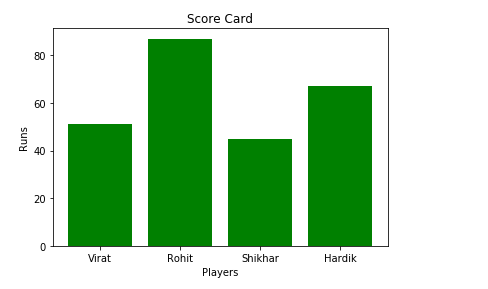


### 2. Bar graphs

Bar graphs are one of the most common types of graphs and are used to show data associated with the categorical variables. Matplotlib provides a ****bar()**** to make bar graphs which accepts arguments such as: categorical variables, their value and color.

1. from matplotlib **import** pyplot as plt
2. players = ['Virat','Rohit','Shikhar','Hardik']
3. runs = [51,87,45,67]
4. plt.bar(players,runs,color = 'green')
5. plt.title('Score Card')
6. plt.xlabel('Players')
7. plt.ylabel('Runs')
8. plt.show()

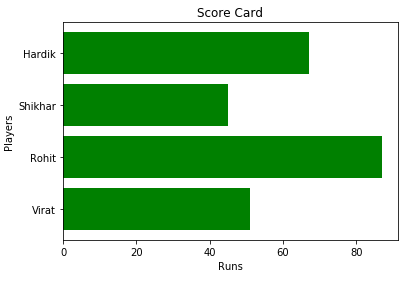
****Output:****



Another function ****barh()**** is used to make horizontal bar graphs. It accepts ****xerr**** or ****yerr**** as arguments (in case of vertical graphs) to depict the variance in our data as follows:

1. from matplotlib **import** pyplot as plt
2. players = ['Virat','Rohit','Shikhar','Hardik']
3. runs = [51,87,45,67]
4. plt.barh(players,runs, color = 'green')
5. plt.title('Score Card')
6. plt.xlabel('Players')
7. plt.ylabel('Runs')
8. plt.show()

****Output:****



Let's have a look on the other example using the ****style()**** function:

1. from matplotlib **import** pyplot as plt
2. from matplotlib **import** style
4. style.use('ggplot')
6. x = [5,8,10]
7. y = [12,16,6]
9. x2 = [6,9,11]
10. y2 = [7,15,7]

13. plt.bar(x, y, color = 'y', align='center')
14. plt.bar(x2, y2, color='c', align='center')
16. plt.title('Information')
18. plt.ylabel('Y axis')
19. plt.xlabel('X axis')

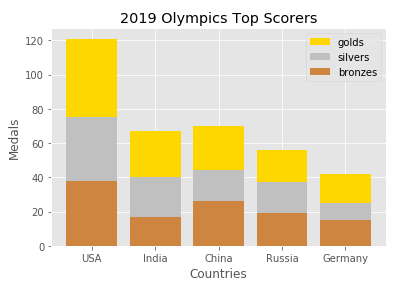
****Output:****



Similarly to vertical stack, the bar graph together by using the bottom argument and define the bar graph, which we want to stack below and its value.

1. from matplotlib **import** pyplot as plt
2. **import** numpy as np
4. countries = ['USA', 'India', 'China', 'Russia', 'Germany']
5. bronzes = np.array([38, 17, 26, 19, 15])
6. silvers = np.array([37, 23, 18, 18, 10])
7. golds = np.array([46, 27, 26, 19, 17])
8. ind = [x **for** x, \_ in enumerate(countries)]
10. plt.bar(ind, golds, width=0.5, label='golds', color='gold', bottom=silvers+bronzes)
11. plt.bar(ind, silvers, width=0.5, label='silvers', color='silver', bottom=bronzes)
12. plt.bar(ind, bronzes, width=0.5, label='bronzes', color='#CD853F')
14. plt.xticks(ind, countries)
15. plt.ylabel("Medals")
16. plt.xlabel("Countries")
17. plt.legend(loc="upper right")
18. plt.title("2019 Olympics Top Scorers")

****Output:****

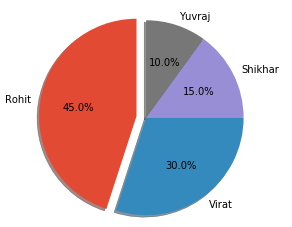


### 3. Pie Chart

A pie chart is a circular graph that is broken down in the segment or slices of pie. It is generally used to represent the percentage or proportional data where each slice of pie represents a particular category. Let's have a look at the below example:

1. from matplotlib **import** pyplot as plt
3. # Pie chart, where the slices will be ordered and plotted counter-clockwise:
4. Players = 'Rohit', 'Virat', 'Shikhar', 'Yuvraj'
5. Runs = [45, 30, 15, 10]
6. explode = (0.1, 0, 0, 0)  # it "explode" the 1st slice
8. fig1, ax1 = plt.subplots()
9. ax1.pie(Runs, explode=explode, labels=Players, autopct='%1.1f%%',
10. shadow=True, startangle=90)
11. ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
13. plt.show()

****Output:****



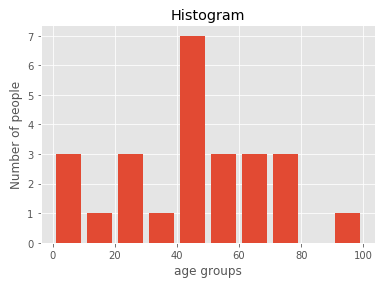
### 4. Histogram

First, we need to understand the difference between the bar graph and histogram. A histogram is used for the distribution, whereas a bar chart is used to compare different entities. A histogram is a type of bar plot that shows the frequency of a number of values compared to a set of values ranges.

****For example**** we take the data of the different age group of the people and plot a histogram with respect to the bin. Now, bin represents the range of values that are divided into series of intervals. Bins are generally created of the same size.

1. from matplotlib **import** pyplot as plt
2. from matplotlib **import** pyplot as plt
3. population\_age = [21,53,60,49,25,27,30,42,40,1,2,102,95,8,15,105,70,65,55,70,75,60,52,44,43,42,45]
4. bins = [0,10,20,30,40,50,60,70,80,90,100]
5. plt.hist(population\_age, bins, histtype='bar', rwidth=0.8)
6. plt.xlabel('age groups')
7. plt.ylabel('Number of people')
8. plt.title('Histogram')
9. plt.show()

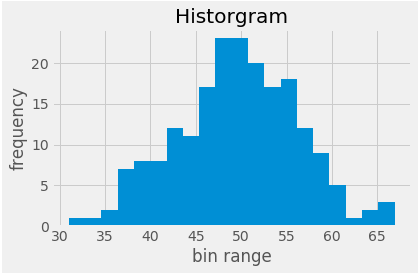
****Output:****



Let's consider the another example of plotting histogram:

1. from matplotlib **import** pyplot as plt
2. # Importing Numpy Library
3. **import** numpy as np
4. plt.style.use('fivethirtyeight')
6. mu = 50
7. sigma = 7
8. x = np.random.normal(mu, sigma, size=200)
9. fig, ax = plt.subplots()
11. ax.hist(x, 20)
12. ax.set\_title('Historgram')
13. ax.set\_xlabel('bin range')
14. ax.set\_ylabel('frequency')
16. fig.tight\_layout()
17. plt.show()

****Output:****



### 5. Scatter plot

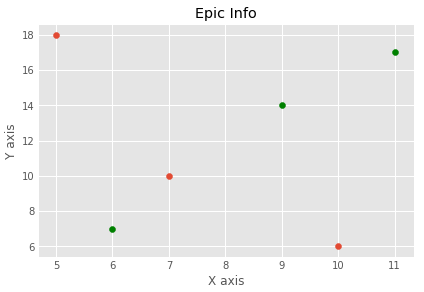
The scatter plots are mostly used for comparing variables when we need to define how much one variable is affected by another variable. The data is displayed as a collection of points. Each point has the value of one variable, which defines the position on the horizontal axes, and the value of other variable represents the position on the vertical axis.

Let's consider the following simple example:

****Example-1:****

1. from matplotlib **import** pyplot as plt
2. from matplotlib **import** style
3. style.use('ggplot')
5. x = [5,7,10]
6. y = [18,10,6]
8. x2 = [6,9,11]
9. y2 = [7,14,17]
11. plt.scatter(x, y)
13. plt.scatter(x2, y2, color='g')
15. plt.title('Epic Info')
16. plt.ylabel('Y axis')
17. plt.xlabel('X axis')
19. plt.show()

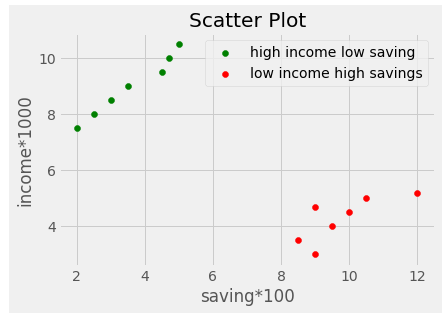
****Output:****



****Example-2****

1. **import** matplotlib.pyplot as plt
2. x = [2, 2.5, 3, 3.5, 4.5, 4.7, 5.0]
3. y = [7.5, 8, 8.5, 9, 9.5, 10, 10.5]
5. x1 = [9, 8.5, 9, 9.5, 10, 10.5, 12]
6. y1 = [3, 3.5, 4.7, 4, 4.5, 5, 5.2]
7. plt.scatter(x, y, label='high income low saving', color='g')
8. plt.scatter(x1, y1, label='low income high savings', color='r')
9. plt.xlabel('saving\*100')
10. plt.ylabel('income\*1000')
11. plt.title('Scatter Plot')
12. plt.legend()
13. plt.show()

****Output:****



### 6. 3D graph plot

Matplotlib was initially developed with only two-dimension plot. Its 1.0 release was built with some of three-dimensional plotting utilities on top of two-dimension display, and the result is a convenient set of tools for 3D data visualization.

Three-dimension plots can be created by importing the ****mplot3d**** toolkit, include with the main Matplotlib installation:

1. from mpl\_toolkits **import** mplot3d

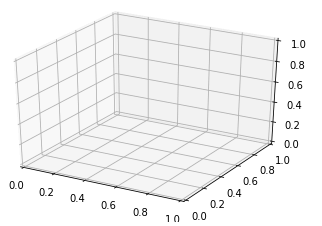
When this module is imported in the program, three-dimension axes can be created by passing the keyword ****projection='3d'**** to any of the normal axes creation routines:

Let's see the simple 3D plot

****Example-1:****

1. from mpltoolkits **import** mplot3d
2. **import** numpy as np
3. **import** matplotlib.pyplot as plt
4. fig = plt.figure()
5. ax = plt.axes(projection='3d')\_

****Output:****

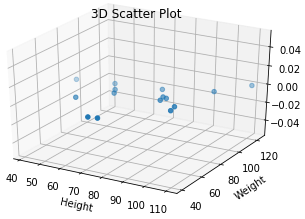


****Example-2:****

1. from mpl\_toolkits **import** mplot3d
2. **import** numpy as np
3. **import** matplotlib.pyplot as plt
5. height = np.array([100,110,87,85,65,80,96,75,42,59,54,63,95,71,86])
6. weight = np.array([105,123,84,85,78,95,69,42,87,91,63,83,75,41,80])

9. scatter(height,weight)
11. fig = plt.figure()
12. ax = plt.axes(projection='3d')
13. # This is used to plot 3D scatter
14. ax.scatter3D(height,weight)
15. plt.title("3D Scatter Plot")
16. plt.xlabel("Height")
17. plt.ylabel("Weight")
18. plt.title("3D Scatter Plot")
19. plt.xlabel("Height")
20. plt.ylabel("Weight")
22. plt.show()

****Output:****

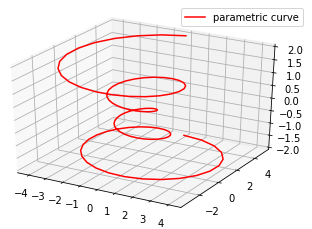


#### **Note:** We can use the plot3D () to plot simple 3D line graph.

****Example-3****

1. **import** matplotlib as mpl
2. from mpl\_toolkits.mplot3d **import** Axes3D
3. **import** numpy as np
4. **import** matplotlib.pyplot as plt
6. mpl.rcParams['legend.fontsize'] = 10
8. fig = plt.figure()
9. ax = fig.gca(projection='3d')
10. theta1 = np.linspace(-4 \* np.pi, 4 \* np.pi, 100)
11. z = np.linspace(-2, 2, 100)
12. r = z\*\*2 + 1
13. x = r \* np.sin(theta1)
14. y = r \* np.cos(theta1)
15. ax.plot3D(x, y, z, label='parametric curve', color = 'red')
16. ax.legend()
18. plt.show()

****Output:****



## Important functions of Matplotlib

|  |  |
| --- | --- |
| **Functions** | **Description** |
| plot(x-axis value, y-axis-values) | It is used to plot a simple line graph with x-axis value against the y-axis values. show() It is used to display the graph. |
| title("string") | It is used to set the title of the plotted graph as specified by the string. |
| xlabel("string") | It is used to set the label for the x-axis as specified by the string. |
| ylabel("string") | It is used to set the label for y-axis as specified by the string. |
| figure() | It is used to control a figure level attributes. |
| subplots(nrows,ncol,index) | It is used to add a subplot to recent figure. |
| subtitle("string") | It adds a common title to the plotted graph specified by the string. |
| subplots(nrows,ncols,figsize) | It provides the simple way to create subplot, in a single call and returns a tuple of a figure and number of axes. |
| set\_title("string") | It is an axes level method which is used to set the title of the subplots. |
| bar(categorical variables, values, color) | It is used to create a vertical bar graph. |
| barh(categorical variables, values, color) | It is used to create horizontal bar graphs. |
| legend(loc) | It is used to make a legend of the graph. |
| xtricks(index, categorical variables) | It is used to set or get the current tick locations labels of the x-axis. |
| pie(value, categorical variables) | It is used to create a pie chart. |
| hist(value, number of bins) | It is used to create a histogram. |
| xlim(start value, end value) | It is used to set the limit of values of the x-axis. |
| ylim(start value, end value) | It is used to set the limit of values of the y-axis. |
| scatter(x-axis values, y-axis values) | It is used to plots a scatter plot with x-axis value against the y-axis values. |
| axes() | It is used to add axes to the recent figure. |
| set\_xlabel("string") | It is an axes level method which is used to set the x-label of the plot specified as a string. |
| set\_ylabel("string") | It is used to set the y-label of the plot specified as a string. |
| scatter3D(x-axis values, y-axis values) | It is used to plot a three-dimension scatter plot with x- axis value against the y-axis. |
| plot3D(x-axis values, y-axis values) | It is used to plots a three-dimension line graph with x- axis values against y-axis values. |

In this tutorial, we have learned about the matplotlib (Python Library), where we covered the brief introduction of data visualization and how data visualization is essential to make a decision for the organization. We have plotted the different types of graphs for the graphical representation of the data.