**Matplotlib Tutorial**

* Difficulty Level : [Easy](https://www.geeksforgeeks.org/easy/)
* Last Updated : 18 Nov, 2022

 Read

 Discuss

 Courses

 Practice

 Video

**Matplotlib**is easy to use and an amazing visualizing library in Python. It is built on NumPy arrays and designed to work with the broader SciPy stack and consists of several plots like line, bar, scatter, histogram, etc.

In this article, we will learn about ***Python plotting with Matplotlib*** from basics to advance with the help of a huge dataset containing information about different types of plots and their customizations.

**Table Of Content**

* [*Getting Started*](https://www.geeksforgeeks.org/matplotlib-tutorial/#getting%20started)
* [*Pyplot*](https://www.geeksforgeeks.org/matplotlib-tutorial/#pyplot)
* [*Figure class*](https://www.geeksforgeeks.org/matplotlib-tutorial/#figure%20class)
* [*Axes Class*](https://www.geeksforgeeks.org/matplotlib-tutorial/#axes%20class)
* [*Setting Limits and Tick labels*](https://www.geeksforgeeks.org/matplotlib-tutorial/#limits)
* [*Multiple Plots*](https://www.geeksforgeeks.org/matplotlib-tutorial/#multiple)
* [*What is a Legend?*](https://www.geeksforgeeks.org/matplotlib-tutorial/#what)
* [*Creating Different Types of Plots*](https://www.geeksforgeeks.org/matplotlib-tutorial/#creating)
  + [*Line Graph*](https://www.geeksforgeeks.org/matplotlib-tutorial/#line)
  + [*Bar chart*](https://www.geeksforgeeks.org/matplotlib-tutorial/#bar)
  + [*Histograms*](https://www.geeksforgeeks.org/matplotlib-tutorial/#histogram)
  + [*Scatter Plot*](https://www.geeksforgeeks.org/matplotlib-tutorial/#scatter)
  + [*Pie Chart*](https://www.geeksforgeeks.org/matplotlib-tutorial/#pie)
  + [*3D Plots*](https://www.geeksforgeeks.org/matplotlib-tutorial/#3d)
* [*Working with Images*](https://www.geeksforgeeks.org/matplotlib-tutorial/#images)
* [*Customizing Plots in Matplotlib*](https://www.geeksforgeeks.org/matplotlib-tutorial/#customizing)
* [*More articles on Matplotlib*](https://www.geeksforgeeks.org/matplotlib-tutorial/#more)
* [*Exercises, Applications, and Projects*](https://www.geeksforgeeks.org/matplotlib-tutorial/#exercises)

[**Recent Articles on Matplotlib !!!**](https://www.geeksforgeeks.org/tag/python-matplotlib/)

**Getting Started**

Before we start learning about Matplotlib we first have to set up the environment and will also see how to use Matplotlib with Jupyter Notebook:

* [Environment Setup for Matplotlib](https://www.geeksforgeeks.org/environment-setup-for-matplotlib/)
* [Using Matplotlib with Jupyter Notebook](https://www.geeksforgeeks.org/using-matplotlib-with-jupyter-notebook/)

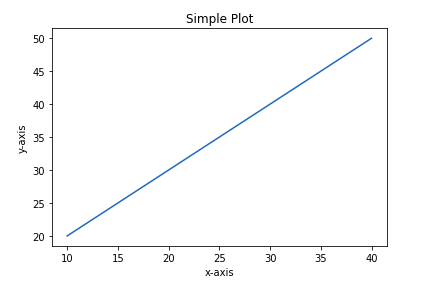
After learning about the environment setup and how to use Matplotlib with Jupyter let’s create a [simple plot](https://www.geeksforgeeks.org/simple-plot-in-python-using-matplotlib/). We will be plotting two lists containing the X, Y coordinates for the plot.

**Example:**

* Python3

|  |
| --- |
| **import** matplotlib.pyplot as plt    # initializing the data  x **=** [10, 20, 30, 40]  y **=** [20, 30, 40, 50]    # plotting the data  plt.plot(x, y)    # Adding the title  plt.title("Simple Plot")    # Adding the labels  plt.ylabel("y-axis")  plt.xlabel("x-axis")  plt.show() |

**Output:**



In the above example, the elements of **X** and **Y** provides the coordinates for the **x axis and y axis**and a straight line is plotted against those coordinates. For a detailed introduction to Matplotlib and to see how basic charts are plotted refer to the below article.

* [Introduction to Matplotlib](https://www.geeksforgeeks.org/python-introduction-matplotlib/)
* [Simple Plot in Python using Matplotlib](https://www.geeksforgeeks.org/simple-plot-in-python-using-matplotlib/)

In the above article, you might have seen **Pyplot** was imported in code and must have wondered what is Pyplot. Don’t worry we will discuss the Pyplot in the next section.

**Pyplot**

[**Pyplot**](https://www.geeksforgeeks.org/pyplot-in-matplotlib/) is a Matplotlib module that provides a MATLAB-like interface. Pyplot provides functions that interact with the figure i.e. creates a figure, decorates the plot with labels, and creates a plotting area in a figure.

**Syntax:**

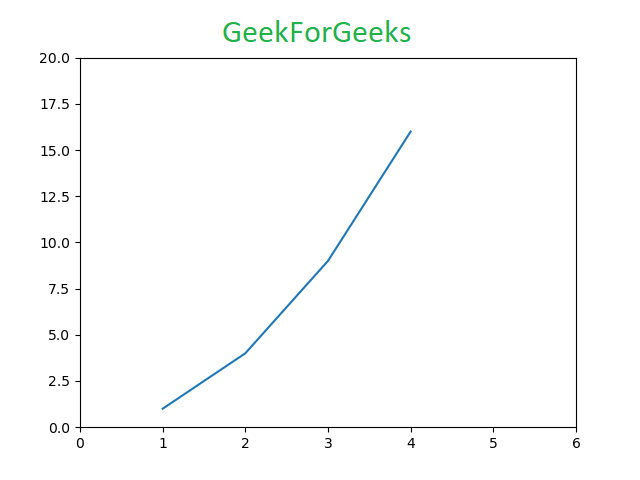
*matplotlib.pyplot.plot(\*args, scalex=True, scaley=True, data=None, \*\*kwargs)*

**Example:**

* Python3

|  |
| --- |
| # Python program to show pyplot module  **import** matplotlib.pyplot as plt  plt.plot([1, 2, 3, 4], [1, 4, 9, 16])  plt.axis([0, 6, 0, 20])  plt.show() |

**Output:**



Refer to the below articles to get detailed information about Pyplot and functions associated with this class.

* [Pyplot in Matplotlib](https://www.geeksforgeeks.org/pyplot-in-matplotlib/)
* [Matplotlib.pyplot.plot() function in Python](https://www.geeksforgeeks.org/matplotlib-pyplot-plot-function-in-python/)
* [Matplotlib.pyplot.title() in Python](https://www.geeksforgeeks.org/matplotlib-pyplot-title-in-python/)
* [matplotlib.pyplot.imshow() in Python](https://www.geeksforgeeks.org/matplotlib-pyplot-imshow-in-python/)
* [Matplotlib.pyplot.legend() in Python](https://www.geeksforgeeks.org/matplotlib-pyplot-legend-in-python/)
* [Matplotlib.pyplot.subplots() in Python](https://www.geeksforgeeks.org/matplotlib-pyplot-subplots-in-python/)
* [Matplotlib.pyplot.colors() in Python](https://www.geeksforgeeks.org/matplotlib-pyplot-colors-in-python/)
* [Matplotlib.pyplot.grid() in Python](https://www.geeksforgeeks.org/matplotlib-pyplot-grid-in-python/)

[**>>> More Functions on Pyplot class**](https://www.geeksforgeeks.org/tag/matplotlib-pyplot-class/)

Matplotlib take care of the creation of inbuilt defaults like **Figure and Axes.**Don’t worry about these terms we will study them in detail in the below section but let’s take a brief about these terms.

* **Figure:**This class is the top-level container for all the plots means it is the overall window or page on which everything is drawn. A figure object can be considered as a box-like container that can hold one or more axes.
* **Axes:**This class is the most basic and flexible component for creating sub-plots. You might confuse axes as the plural of axis but it is an individual plot or graph. A given figure may contain many axes but given axes can only be in one figure.

**Figure class**

Figure class is the top-level container that contains one or more axes. It is the overall window or page on which everything is drawn.

**Syntax:**

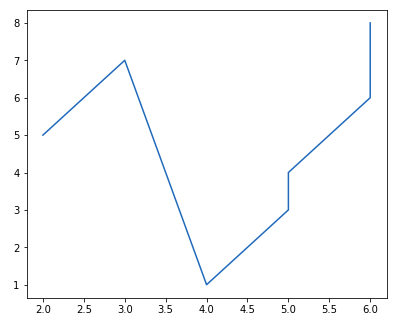
*class matplotlib.figure.Figure(figsize=None, dpi=None, facecolor=None, edgecolor=None, linewidth=0.0, frameon=None, subplotpars=None, tight\_layout=None, constrained\_layout=None)*

**Example 1:**

* Python3

|  |
| --- |
| # Python program to show pyplot module  **import** matplotlib.pyplot as plt  **from** matplotlib.figure **import** Figure    # Creating a new figure with width = 5 inches  # and height = 4 inches  fig **=** plt.figure(figsize **=**(5, 4))    # Creating a new axes for the figure  ax **=** fig.add\_axes([1, 1, 1, 1])    # Adding the data to be plotted  ax.plot([2, 3, 4, 5, 5, 6, 6],          [5, 7, 1, 3, 4, 6 ,8])  plt.show() |

**Output:**

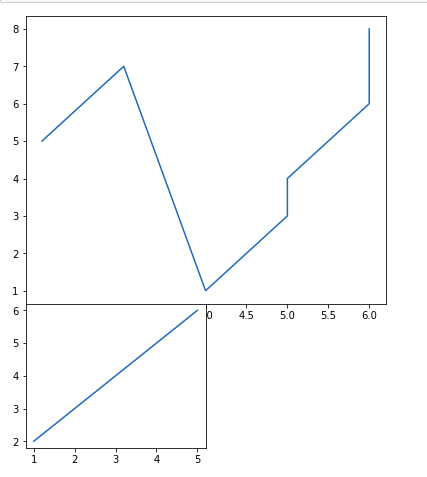


**Example 2:**Creating multiple plots

* Python3

|  |
| --- |
| # Python program to show pyplot module  **import** matplotlib.pyplot as plt  **from** matplotlib.figure **import** Figure    # Creating a new figure with width = 5 inches  # and height = 4 inches  fig **=** plt.figure(figsize **=**(5, 4))    # Creating first axes for the figure  ax1 **=** fig.add\_axes([1, 1, 1, 1])    # Creating second axes for the figure  ax2 **=** fig.add\_axes([1, 0.5, 0.5, 0.5])    # Adding the data to be plotted  ax1.plot([2, 3, 4, 5, 5, 6, 6],           [5, 7, 1, 3, 4, 6 ,8])  ax2.plot([1, 2, 3, 4, 5],           [2, 3, 4, 5, 6])    plt.show() |

**Output:**



Refer to the below articles to get detailed information about the Figure class and functions associated with it.

* [Matplotlib.figure.Figure() in Python](https://www.geeksforgeeks.org/matplotlib-figure-figure-in-python/)
* [Matplotlib.figure.Figure.add\_axes() in Python](https://www.geeksforgeeks.org/matplotlib-figure-figure-add_axes-in-python/)
* [Matplotlib.figure.Figure.clear() in Python](https://www.geeksforgeeks.org/matplotlib-figure-figure-clear-in-python/)
* [Matplotlib.figure.Figure.colorbar() in Python](https://www.geeksforgeeks.org/matplotlib-figure-figure-colorbar-in-python/)
* [Matplotlib.figure.Figure.get\_figwidth() in Python](https://www.geeksforgeeks.org/matplotlib-figure-figure-get_figwidth-in-python/)
* [Matplotlib.figure.Figure.get\_figheight() in Python](https://www.geeksforgeeks.org/matplotlib-figure-figure-get_figheight-in-python/)
* [Matplotlib.figure.Figure.subplots() in Python](https://www.geeksforgeeks.org/matplotlib-figure-figure-subplots-in-python/)

[**>>> More Functions in Figure Class**](https://www.geeksforgeeks.org/tag/matplotlib-figure-class/)

**Axes Class**

[**Axes class**](https://www.geeksforgeeks.org/matplotlib-axes-class/) is the most basic and flexible unit for creating sub-plots. A given figure may contain many axes, but a given axes can only be present in one figure. The axes() function creates the axes object. Let’s see the below example.

**Syntax:**

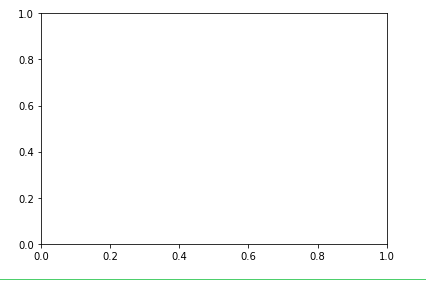
*matplotlib.pyplot.axis(\*args, emit=True, \*\*kwargs)*

**Example 1:**

* Python3

|  |
| --- |
| # Python program to show pyplot module  **import** matplotlib.pyplot as plt  **from** matplotlib.figure **import** Figure  # Creating the axes object with argument as  # [left, bottom, width, height]  ax **=** plt.axes([1, 1, 1, 1]) |

**Output:**

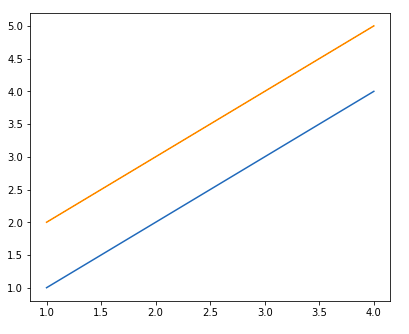


**Example 2:**

* Python3

|  |
| --- |
| # Python program to show pyplot module  **import** matplotlib.pyplot as plt  **from** matplotlib.figure **import** Figure  fig **=** plt.figure(figsize **=** (5, 4))    # Adding the axes to the figure  ax **=** fig.add\_axes([1, 1, 1, 1])    # plotting 1st dataset to the figure  ax1 **=** ax.plot([1, 2, 3, 4], [1, 2, 3, 4])    # plotting 2nd dataset to the figure  ax2 **=** ax.plot([1, 2, 3, 4], [2, 3, 4, 5])  plt.show() |

**Output:**



Refer to the below articles to get detailed information about the axes class and functions associated with it.

* [Matplotlib – Axes Class](https://www.geeksforgeeks.org/matplotlib-axes-class/)
* [Matplotlib.axes.Axes.update() in Python](https://www.geeksforgeeks.org/matplotlib-axes-axes-update-in-python/)
* [Matplotlib.axes.Axes.draw() in Python](https://www.geeksforgeeks.org/matplotlib-axes-axes-draw-in-python/)
* [Matplotlib.axes.Axes.get\_figure() in Python](https://www.geeksforgeeks.org/matplotlib-axes-axes-get_figure-in-python/)
* [Matplotlib.axes.Axes.set\_figure() in Python](https://www.geeksforgeeks.org/matplotlib-axes-axes-set_figure-in-python/)
* [Matplotlib.axes.Axes.properties() in Python](https://www.geeksforgeeks.org/matplotlib-axes-axes-properties-in-python/)

[**>>> More Functions on Axes Class**](https://www.geeksforgeeks.org/matplotlib-axes-class/)

**Setting Limits and Tick labels**

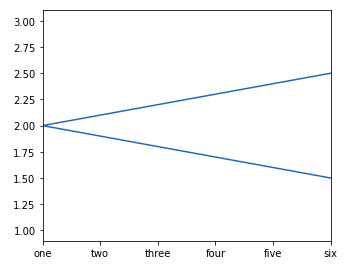
You might have seen that Matplotlib automatically sets the values and the markers(points) of the x and y axis, however, it is possible to set the limit and markers manually. **[set\_xlim()](https://www.geeksforgeeks.org/matplotlib-axes-axes-set_xlim-in-python/)** and **[set\_ylim()](https://www.geeksforgeeks.org/matplotlib-axes-axes-set_ylim-in-python/)** functions are used to set the limits of the x-axis and y-axis respectively. Similarly, **[set\_xticklabels()](https://www.geeksforgeeks.org/matplotlib-axes-axes-set_xticklabels-in-python/)** and **[set\_yticklabels()](https://www.geeksforgeeks.org/matplotlib-axes-axes-set_yticklabels-in-python/)** functions are used to set tick labels.

**Example:**

* Python3

|  |
| --- |
| # Python program to show pyplot module  **import** matplotlib.pyplot as plt  **from** matplotlib.figure **import** Figure  x **=** [3, 1, 3]  y **=** [3, 2, 1]    # Creating a new figure with width = 5 inches  # and height = 4 inches  fig **=** plt.figure(figsize **=**(5, 4))    # Creating first axes for the figure  ax **=** fig.add\_axes([0.1, 0.1, 0.8, 0.8])    # Adding the data to be plotted  ax.plot(x, y)  ax.set\_xlim(1, 2)  ax.set\_xticklabels((    "one", "two", "three", "four", "five", "six"))  plt.show() |

**Output:**



**Multiple Plots**

Till now you must have got a basic idea about Matplotlib and plotting some simple plots, now what if you want to plot multiple plots in the same figure. This can be done using multiple ways. One way was discussed above using the [add\_axes()](https://www.geeksforgeeks.org/matplotlib-figure-figure-add_axes-in-python/) method of the figure class. Let’s see various ways multiple plots can be added with the help of examples.

**Method 1: Using the**[add\_axes()](https://www.geeksforgeeks.org/matplotlib-figure-figure-add_axes-in-python/)**method**

The add\_axes() method figure module of matplotlib library is used to add an axes to the figure.

**Syntax:**

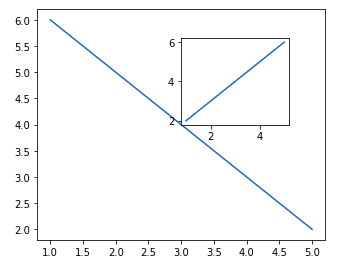
*add\_axes(self, \*args, \*\*kwargs)*

**Example:**

* Python3

|  |
| --- |
| # Python program to show pyplot module  **import** matplotlib.pyplot as plt  **from** matplotlib.figure **import** Figure    # Creating a new figure with width = 5 inches  # and height = 4 inches  fig **=** plt.figure(figsize **=**(5, 4))    # Creating first axes for the figure  ax1 **=** fig.add\_axes([0.1, 0.1, 0.8, 0.8])    # Creating second axes for the figure  ax2 **=** fig.add\_axes([0.5, 0.5, 0.3, 0.3])    # Adding the data to be plotted  ax1.plot([5, 4, 3, 2, 1], [2, 3, 4, 5, 6])  ax2.plot([1, 2, 3, 4, 5], [2, 3, 4, 5, 6])  plt.show() |

**Output:**



The add\_axes() method adds the plot in the same figure by creating another axes object.

**Method 2: Using**[subplot()](https://www.geeksforgeeks.org/matplotlib-pyplot-subplot-function-in-python/)**method.**

This method adds another plot to the current figure at the specified grid position.

**Syntax:**

*subplot(nrows, ncols, index, \*\*kwargs)*

*subplot(pos, \*\*kwargs)*

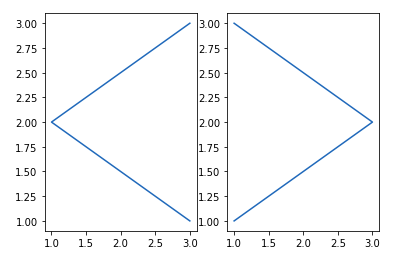
*subplot(ax)*

**Example:**

* Python3

|  |
| --- |
| **import** matplotlib.pyplot as plt  # data to display on plots  x **=** [3, 1, 3]  y **=** [3, 2, 1]  z **=** [1, 3, 1]    # Creating figure object  plt.figure()    # addind first subplot  plt.subplot(121)  plt.plot(x, y)    # addding second subplot  plt.subplot(122)  plt.plot(z, y) |

**Output:**



**Note:**Subplot() function have the following disadvantages –

* It does not allow adding multiple subplots at the same time.
* It deletes the preexisting plot of the figure.

**Method 3: Using**[subplots()](https://www.geeksforgeeks.org/matplotlib-pyplot-subplots-in-python/)**method**

This function is used to create figure and multiple subplots at the same time.

**Syntax:**

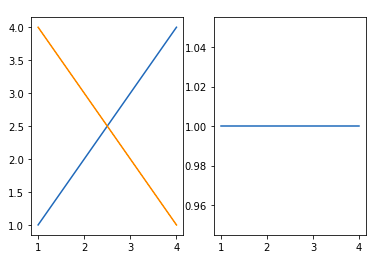
*matplotlib.pyplot.subplots(nrows=1, ncols=1, sharex=False, sharey=False, squeeze=True, subplot\_kw=None, gridspec\_kw=None, \*\*fig\_kw)*

**Example:**

* Python3

|  |
| --- |
| **import** matplotlib.pyplot as plt    # Creating the figure and subplots  # according the argument passed  fig, axes **=** plt.subplots(1, 2)    # plotting the data in the 1st subplot  axes[0].plot([1, 2, 3, 4], [1, 2, 3, 4])    # plotting the data in the 1st subplot only  axes[0].plot([1, 2, 3, 4], [4, 3, 2, 1])    # plotting the data in the 2nd subplot only  axes[1].plot([1, 2, 3, 4], [1, 1, 1, 1]) |

**Output:**



**Method 4: Using**[subplot2grid()](https://www.geeksforgeeks.org/matplotlib-pyplot-subplot2grid-in-python/)**method**

This function give additional flexibility in creating axes object at a specified location inside a grid. It also helps in spanning the axes object across multiple rows or columns. In simpler words, this function is used to create multiple charts within the same figure.

**Syntax:**

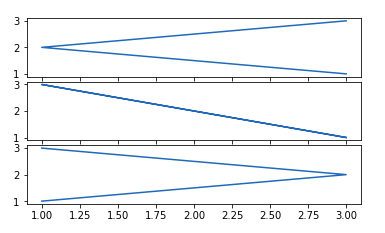
*Plt.subplot2grid(shape, location, rowspan, colspan)*

**Example:**

* Python3

|  |
| --- |
| **import** matplotlib.pyplot as plt  # data to display on plots  x **=** [3, 1, 3]  y **=** [3, 2, 1]  z **=** [1, 3, 1]    # adding the subplots  axes1 **=** plt.subplot2grid (    (7, 1), (0, 0), rowspan **=** 2,  colspan **=** 1)  axes2 **=** plt.subplot2grid (    (7, 1), (2, 0), rowspan **=** 2, colspan **=** 1)  axes3 **=** plt.subplot2grid (    (7, 1), (4, 0), rowspan **=** 2, colspan **=** 1)    # plotting the data  axes1.plot(x, y)  axes2.plot(x, z)  axes3.plot(z, y) |

**Output:**



Refer to the below articles to get detailed information about subplots

* [How to create multiple subplots in Matplotlib in Python?](https://www.geeksforgeeks.org/how-to-create-multiple-subplots-in-matplotlib-in-python/)
* [How to Add Title to Subplots in Matplotlib?](https://www.geeksforgeeks.org/how-to-add-title-to-subplots-in-matplotlib/)
* [How to Set a Single Main Title for All the Subplots in Matplotlib?](https://www.geeksforgeeks.org/how-to-set-a-single-main-title-for-all-the-subplots-in-matplotlib/)
* [How to Turn Off the Axes for Subplots in Matplotlib?](https://www.geeksforgeeks.org/how-to-turn-off-the-axes-for-subplots-in-matplotlib/)
* [How to Create Different Subplot Sizes in Matplotlib?](https://www.geeksforgeeks.org/how-to-create-different-subplot-sizes-in-matplotlib/)
* [How to set the spacing between subplots in Matplotlib in Python?](https://www.geeksforgeeks.org/how-to-set-the-spacing-between-subplots-in-matplotlib-in-python/)
* [Matplotlib Sub plotting using object oriented API](https://www.geeksforgeeks.org/python-matplotlib-sub-plotting-using-object-oriented-api/)
* [Make subplots span multiple grid rows and columns in Matplotlib](https://www.geeksforgeeks.org/make-subplots-span-multiple-grid-rows-and-columns-in-matplotlib/)

**What is a Legend?**

A legend is an area describing the elements of the graph. In simple terms, it reflects the data displayed in the graph’s Y-axis. It generally appears as the box containing a small sample of each color on the graph and a small description of what this data means.

**Creating the Legend**

A Legend can be created using the [**legend()**](https://www.geeksforgeeks.org/matplotlib-pyplot-legend-in-python/) method. The attribute **Loc** in the legend() is used to specify the location of the legend. The default value of loc is loc=”best” (upper left). The strings ‘upper left’, ‘upper right’, ‘lower left’, ‘lower right’ place the legend at the corresponding corner of the axes/figure.

The attribute **bbox\_to\_anchor=(x, y)** of legend() function is used to specify the coordinates of the legend, and the attribute ncol represents the number of columns that the legend has. Its default value is 1.

**Syntax:**

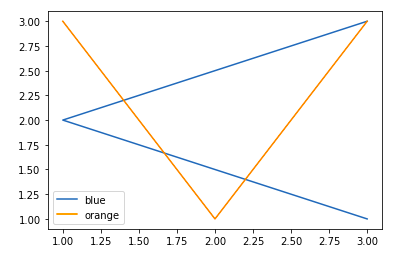
*matplotlib.pyplot.legend([“blue”, “green”], bbox\_to\_anchor=(0.75, 1.15), ncol=2)*

**Example:**

* Python3

|  |
| --- |
| **import** matplotlib.pyplot as plt    # data to display on plots  x **=** [3, 1, 3]  y **=** [3, 2, 1]  plt.plot(x, y)  plt.plot(y, x)    # Adding the legends  plt.legend(["blue", "orange"])  plt.show() |

**Output:**



Refer to the below articles to get detailed information about the legend –

* [Matplotlib.pyplot.legend() in Python](https://www.geeksforgeeks.org/matplotlib-pyplot-legend-in-python/)
* [Matplotlib.axes.Axes.legend() in Python](https://www.geeksforgeeks.org/matplotlib-axes-axes-legend-in-python/)
* [Change the legend position in Matplotlib](https://www.geeksforgeeks.org/change-the-legend-position-in-matplotlib/)
* [How to Change Legend Font Size in Matplotlib?](https://www.geeksforgeeks.org/how-to-change-legend-font-size-in-matplotlib/)
* [How Change the vertical spacing between legend entries in Matplotlib?](https://www.geeksforgeeks.org/how-change-the-vertical-spacing-between-legend-entries-in-matplotlib/)
* [Use multiple columns in a Matplotlib legend](https://www.geeksforgeeks.org/use-multiple-columns-in-a-matplotlib-legend/)
* [How to Create a Single Legend for All Subplots in Matplotlib?](https://www.geeksforgeeks.org/how-to-create-a-single-legend-for-all-subplots-in-matplotlib/)
* [How to manually add a legend with a color box on a Matplotlib figure ?](https://www.geeksforgeeks.org/how-to-manually-add-a-legend-with-a-color-box-on-a-matplotlib-figure/)
* [How to Place Legend Outside of the Plot in Matplotlib?](https://www.geeksforgeeks.org/how-to-place-legend-outside-of-the-plot-in-matplotlib/)
* [How to Remove the Legend in Matplotlib?](https://www.geeksforgeeks.org/how-to-remove-the-legend-in-matplotlib/)
* [Remove the legend border in Matplotlib](https://www.geeksforgeeks.org/remove-the-legend-border-in-matplotlib/)

**Creating Different Types of Plots**

**Line Graph**

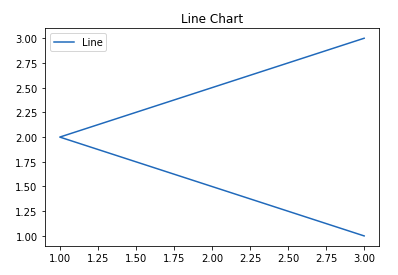
Till now you all must have seen that we are working with only the **line charts** as they are easy to plot and understand. Line Chart is used to represent a relationship between two data X and Y on a different axis. It is plotted using the pot() function. Let’s see the below example

**Example:**

* Python3

|  |
| --- |
| **import** matplotlib.pyplot as plt  # data to display on plots  x **=** [3, 1, 3]  y **=** [3, 2, 1]    # This will plot a simple line chart  # with elements of x as x axis and y  # as y axis  plt.plot(x, y)  plt.title("Line Chart")    # Adding the legends  plt.legend(["Line"])  plt.show() |

**Output:**



Refer to the below article to get detailed information about line chart.

* [Line chart in Matplotlib](https://www.geeksforgeeks.org/line-chart-in-matplotlib-python/)
* [Line plot styles in Matplotlib](https://www.geeksforgeeks.org/line-plot-styles-in-matplotlib/)
* [Plot a Horizontal line in Matplotlib](https://www.geeksforgeeks.org/plot-a-horizontal-line-in-matplotlib/)
* [Plot a Vertical line in Matplotlib](https://www.geeksforgeeks.org/plot-a-vertical-line-in-matplotlib/)
* [Plot Multiple lines in Matplotlib](https://www.geeksforgeeks.org/plot-multiple-lines-in-matplotlib/)
* [Change the line opacity in Matplotlib](https://www.geeksforgeeks.org/change-the-line-opacity-in-matplotlib/)
* [Increase the thickness of a line with Matplotlib](https://www.geeksforgeeks.org/increase-the-thickness-of-a-line-with-matplotlib/)
* [Plot line graph from NumPy array](https://www.geeksforgeeks.org/plot-line-graph-from-numpy-array/)
* [How to Fill Between Multiple Lines in Matplotlib?](https://www.geeksforgeeks.org/how-to-fill-between-multiple-lines-in-matplotlib/)

**Bar chart**

A **bar plot** 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. The bar plots can be plotted horizontally or vertically. A bar chart describes the comparisons between the discrete categories. It can be created using the **bar()** method.

**Syntax:**

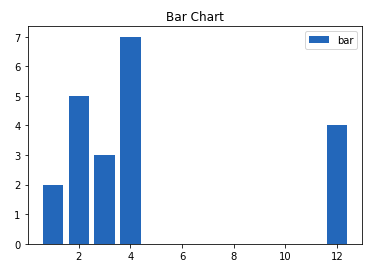
*plt.bar(x, height, width, bottom, align)*

**Example:**

* Python3

|  |
| --- |
| **import** matplotlib.pyplot as plt  # data to display on plots  x **=** [3, 1, 3, 12, 2, 4, 4]  y **=** [3, 2, 1, 4, 5, 6, 7]    # This will plot a simple bar chart  plt.bar(x, y)    # Title to the plot  plt.title("Bar Chart")    # Adding the legends  plt.legend(["bar"])  plt.show() |

**Output:**



Refer to the below articles to get detailed information about Bar charts –

* [Bar Plot in Matplotlib](https://www.geeksforgeeks.org/bar-plot-in-matplotlib/)
* [Draw a horizontal bar chart with Matplotlib](https://www.geeksforgeeks.org/draw-a-horizontal-bar-chart-with-matplotlib/)
* [Create a stacked bar plot in Matplotlib](https://www.geeksforgeeks.org/create-a-stacked-bar-plot-in-matplotlib/)
* [Stacked Percentage Bar Plot In MatPlotLib](https://www.geeksforgeeks.org/stacked-percentage-bar-plot-in-matplotlib/)
* [Plotting back-to-back bar charts Matplotlib](https://www.geeksforgeeks.org/plotting-back-to-back-bar-charts-matplotlib/)
* [How to display the value of each bar in a bar chart using Matplotlib?](https://www.geeksforgeeks.org/how-to-display-the-value-of-each-bar-in-a-bar-chart-using-matplotlib/)
* [How To Annotate Bars in Barplot with Matplotlib in Python?](https://www.geeksforgeeks.org/how-to-annotate-bars-in-barplot-with-matplotlib-in-python/)
* [How to Annotate Bars in Grouped Barplot in Python?](https://www.geeksforgeeks.org/how-to-annotate-bars-in-grouped-barplot-in-python/)

**Histograms**

A **histogram** 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. To create a histogram the first step is to create a bin of the ranges, then distribute the whole range of the values into a series of intervals, and count the values which fall into each of the intervals. Bins are clearly identified as consecutive, non-overlapping intervals of variables. The [hist()](https://www.geeksforgeeks.org/matplotlib-pyplot-hist-in-python/) function is used to compute and create histogram of x.

**Syntax:**

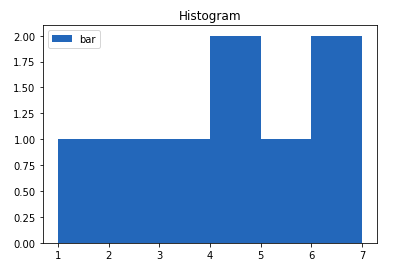
*matplotlib.pyplot.hist(x, bins=None, range=None, density=False, weights=None, cumulative=False, bottom=None, histtype=’bar’, align=’mid’, orientation=’vertical’, rwidth=None, log=False, color=None, label=None, stacked=False, \\*, data=None, \\*\\*kwargs)*

**Example:**

* Python3

|  |
| --- |
| **import** matplotlib.pyplot as plt  # data to display on plots  x **=** [1, 2, 3, 4, 5, 6, 7, 4]  # This will plot a simple histogram  plt.hist(x, bins **=** [1, 2, 3, 4, 5, 6, 7])  # Title to the plot  plt.title("Histogram")  # Adding the legends  plt.legend(["bar"])  plt.show() |

**Output:**



Refer to the below articles to get detailed information about histograms.

* [Plotting Histogram in Python using Matplotlib](https://www.geeksforgeeks.org/plotting-histogram-in-python-using-matplotlib/)
* [Create a cumulative histogram in Matplotlib](https://www.geeksforgeeks.org/create-a-cumulative-histogram-in-matplotlib/)
* [How to plot two histograms together in Matplotlib?](https://www.geeksforgeeks.org/how-to-plot-two-histograms-together-in-matplotlib/)
* [Overlapping Histograms with Matplotlib in Python](https://www.geeksforgeeks.org/overlapping-histograms-with-matplotlib-in-python/)
* [Bin Size in Matplotlib Histogram](https://www.geeksforgeeks.org/bin-size-in-matplotlib-histogram/)
* [Compute the histogram of a set of data using NumPy in Python](https://www.geeksforgeeks.org/compute-the-histogram-of-a-set-of-data-using-numpy-in-python/)
* [Plot 2-D Histogram in Python using Matplotlib](https://www.geeksforgeeks.org/plot-2-d-histogram-in-python-using-matplotlib/)

**Scatter Plot**

Scatter plots are used to observe the relationship between variables and use 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.

**Syntax:**

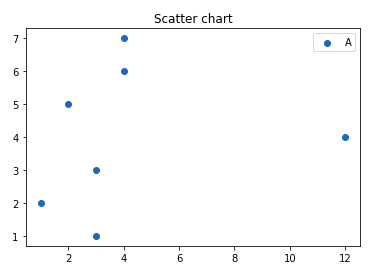
*matplotlib.pyplot.scatter(x\_axis\_data, y\_axis\_data, s=None, c=None, marker=None, cmap=None, vmin=None, vmax=None, alpha=None, linewidths=None, edgecolors=None)*

**Example:**

* Python3

|  |
| --- |
| **import** matplotlib.pyplot as plt  # data to display on plots  x **=** [3, 1, 3, 12, 2, 4, 4]  y **=** [3, 2, 1, 4, 5, 6, 7]    # This will plot a simple scatter chart  plt.scatter(x, y)    # Adding legend to the plot  plt.legend("A")    # Title to the plot  plt.title("Scatter chart")  plt.show() |

**Output:**



Refer to the below articles to get detailed information about the to scatter plot.

* [matplotlib.pyplot.scatter() in Python](https://www.geeksforgeeks.org/matplotlib-pyplot-scatter-in-python/)
* [How to add a legend to a scatter plot in Matplotlib ?](https://www.geeksforgeeks.org/how-to-add-a-legend-to-a-scatter-plot-in-matplotlib/)
* [How to Connect Scatterplot Points With Line in Matplotlib?](https://www.geeksforgeeks.org/how-to-connect-scatterplot-points-with-line-in-matplotlib/https:/www.geeksforgeeks.org/how-to-connect-scatterplot-points-with-line-in-matplotlib/)
* [How to create a Scatter Plot with several colors in Matplotlib?](https://www.geeksforgeeks.org/how-to-create-a-scatter-plot-with-several-colors-in-matplotlib/)
* [How to increase the size of scatter points in Matplotlib ?](https://www.geeksforgeeks.org/how-to-increase-the-size-of-scatter-points-in-matplotlib/)

**Pie Chart**

A **Pie Chart** is a circular statistical plot that can display only one series of data. The area of the chart is the total percentage of the given data. The area of slices of the pie represents the percentage of the parts of the data. The slices of pie are called wedges. The area of the wedge is determined by the length of the arc of the wedge. It can be created using the pie() method.

**Syntax:**

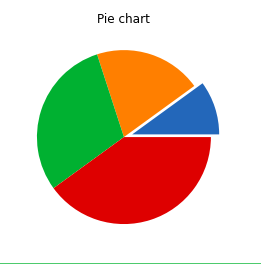
*matplotlib.pyplot.pie(data, explode=None, labels=None, colors=None, autopct=None, shadow=False)*

**Example:**

* Python3

|  |
| --- |
| **import** matplotlib.pyplot as plt  # data to display on plots  x **=** [1, 2, 3, 4]    # this will explode the 1st wedge  # i.e. will separate the 1st wedge  # from the chart  e  **=**(0.1, 0, 0, 0)    # This will plot a simple pie chart  plt.pie(x, explode **=** e)    # Title to the plot  plt.title("Pie chart")  plt.show() |

**Output:**



Refer to the below articles to get detailed information about pie charts.

* [matplotlib.axes.Axes.pie() in Python](https://www.geeksforgeeks.org/matplotlib-axes-axes-pie-in-python/)
* [Plot a pie chart in Python using Matplotlib](https://www.geeksforgeeks.org/plot-a-pie-chart-in-python-using-matplotlib/)
* [How to set border for wedges in Matplotlib pie chart?](https://www.geeksforgeeks.org/how-to-set-border-for-wedges-in-matplotlib-pie-chart/)
* [Radially displace pie chart wedge in Matplotlib](https://www.geeksforgeeks.org/radially-displace-pie-chart-wedge-in-matplotlib/)

**3D Plots**

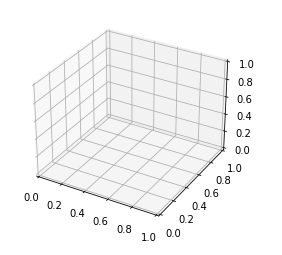
Matplotlib was introduced keeping in mind, only two-dimensional plotting. But at the time when the release of 1.0 occurred, the 3D utilities were developed upon the 2D and thus, we have a 3D implementation of data available today.

**Example:**

* Python3

|  |
| --- |
| **import** matplotlib.pyplot as plt  # Creating the figure object  fig **=** plt.figure()    # keeping the projection = 3d  # ctreates the 3d plot  ax **=** plt.axes(projection **=** '3d') |

**Output:**



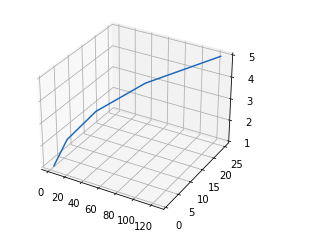
The above code lets the creation of a 3D plot in Matplotlib. We can create different types of 3D plots like scatter plots, contour plots, surface plots, etc. Let’s create a simple 3D line plot.

**Example:**

* Python3

|  |
| --- |
| **import** matplotlib.pyplot as plt  x **=** [1, 2, 3, 4, 5]  y **=** [1, 4, 9, 16, 25]  z **=** [1, 8, 27, 64, 125]  # Creating the figure object  fig **=** plt.figure()  # keeping the projection = 3d  # ctreates the 3d plot  ax **=** plt.axes(projection **=** '3d')  ax.plot3D(z, y, x) |

**Output:**



Refer to the below articles to get detailed information about 3D plots.

* [Three-dimensional Plotting in Python using Matplotlib](https://www.geeksforgeeks.org/three-dimensional-plotting-in-python-using-matplotlib/)
* [3D Scatter Plotting in Python using Matplotlib](https://www.geeksforgeeks.org/3d-scatter-plotting-in-python-using-matplotlib/)
* [3D Surface plotting in Python using Matplotlib](https://www.geeksforgeeks.org/3d-surface-plotting-in-python-using-matplotlib/)
* [3D Wireframe plotting in Python using Matplotlib](https://www.geeksforgeeks.org/3d-wireframe-plotting-in-python-using-matplotlib/)
* [3D Contour Plotting in Python using Matplotlib](https://www.geeksforgeeks.org/3d-contour-plotting-in-python-using-matplotlib/)
* [Tri-Surface Plot in Python using Matplotlib](https://www.geeksforgeeks.org/tri-surface-plot-in-python-using-matplotlib/)
* [Surface plots and Contour plots in Python](https://www.geeksforgeeks.org/surface-plots-and-contour-plots-in-python/)
* [How to change angle of 3D plot in Python?](https://www.geeksforgeeks.org/how-to-change-angle-of-3d-plot-in-python/)
* [How to animate 3D Graph using Matplotlib?](https://www.geeksforgeeks.org/how-to-animate-3d-graph-using-matplotlib/)
* [Draw contours on an unstructured triangular grid in Python using Matplotlib](https://www.geeksforgeeks.org/draw-contours-on-an-unstructured-triangular-grid-in-python-using-matplotlib/)

**Working with Images**

The image module in matplotlib library is used for working with images in Python. The image module also includes two useful methods which are **imread** which is used to read images and **imshow** which is used to display the image.

**Example:**

* Python3

|  |
| --- |
| # importing required libraries  **import** matplotlib.pyplot as plt  **import** matplotlib.image as img  # reading the image  testImage **=** img.imread('g4g.png')  # displaying the image  plt.imshow(testImage) |

**Output:**

