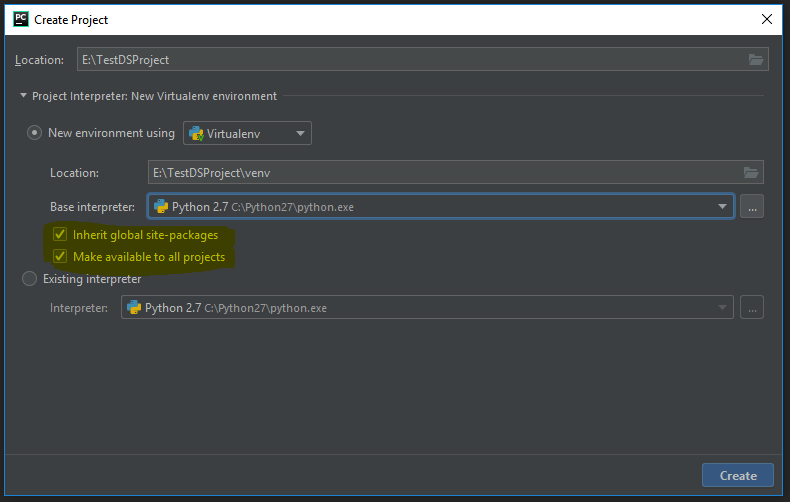
**Environment preparation**

**Pycharm installation:**

Download the exe file from below link and follow the instruction on installation screen.

<https://www.jetbrains.com/pycharm/download/#section=windows>

Remember to tick yellow marked selection. It will allow to install eternal libraries as numpy, scipy and matplotlib.



**Adding numpy, scipy and matplotlib libraries to pycharm.**

Create project and follow the instructions given on below link.

<https://www.jetbrains.com/help/pycharm/installing-uninstalling-and-upgrading-packages.html>

**Errors:**

1. **Invalid syntax.**

|  |
| --- |
| E:\NewDSProject\venv\Scripts\python.exe E:/NewDSProject/main.py  File "E:/NewDSProject/main.py", line 2  import numpy\_data-reader  ^  SyntaxError: invalid syntax  Process finished with exit code 1 |

The error came due to name of file numpy\_data-reader.py. After renaming numpy\_data-reader.py to numpy\_data\_reader.py, error resolved.

**Finding:**

We cannot use ‘-’ (hyphen) in any filename or module name.

2: Print complete output on screen by ***np.set\_printoptions(threshold=np.nan)***

<https://docs.scipy.org/doc/numpy/reference/generated/numpy.set_printoptions.html>

**Libraries**

**Numpy:**

Numpy provides high-performance multidimensional array object and various functions to operate over it.

<http://www.numpy.org/>

<https://docs.scipy.org/doc/numpy/reference/routines.html>

<https://docs.scipy.org/doc/numpy/user/quickstart.html>

<https://docs.scipy.org/doc/numpy/user/>

**Scipy:**

<https://docs.scipy.org/doc/scipy-1.1.0/reference/tutorial/index.html>

**Matplotlib:**

V2.2.3 is LTS (long term support) version.

Matplotlib is a data visualization library built on NumPy arrays in python for 2D plots of arrays. The benefit of visualization is that it allows visual access to huge amounts of data visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

**Basic plots in Matplotlib:**

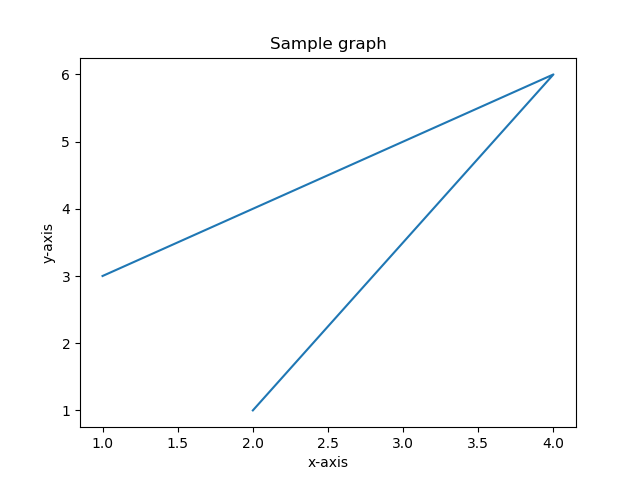
Matplotlib comes with many types of plots. Plots helps to understand trends, patterns, and to make correlations. Some of the sample plots are shown here.

**Line plot:**

|  |
| --- |
| # importing the required module  import matplotlib.pyplot as plt  # x axis values  x = [1, 4, 2]  # corresponding y axis values  y = [3, 6, 1]  # plotting the points  plt.plot(x, y)  # naming the x axis  plt.xlabel('x-axis')  # naming the y axis  plt.ylabel('y-axis')  # giving a title to graph  plt.title('Sample graph')  # function to show the plot  plt.show() |

**Code explanation:**

Define the x-axis and corresponding y-axis values as lists. Plot them on canvas using .plot() function. Give a name to x-axis and y-axis using .xlabel() and .ylabel() functions. Give a title to your plot using .title() function. Finally, to view plot, use .show() function.



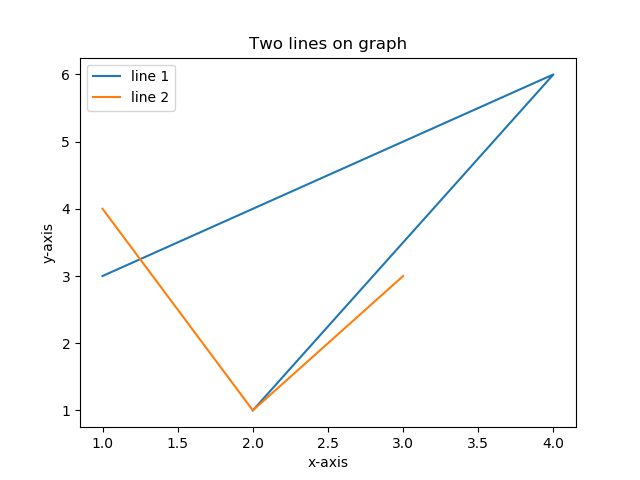
**Plotting two or more lines on same plot:**

|  |
| --- |
| # importing the required module  import matplotlib.pyplot as plt  # line 2 points  x2 = [1, 2, 3]  y2 = [4, 1, 3]  # plotting the line 2 points  plt.plot(x2, y2, label="line 2")  # naming the x axis  plt.xlabel('x-axis')  # naming the y axis  plt.ylabel('y-axis')  # giving a title to graph  plt.title('Two lines on graph')  # show a legend on the plot  plt.legend()  # function to show the plot  plt.show() |

**Code explanation:**

Here, differentiate between line by giving them a name(label) which is passed as an argument of .plot() function.

The small rectangular box giving information about type of line and its color is called legend. We can add a legend to our plot using .legend() function.

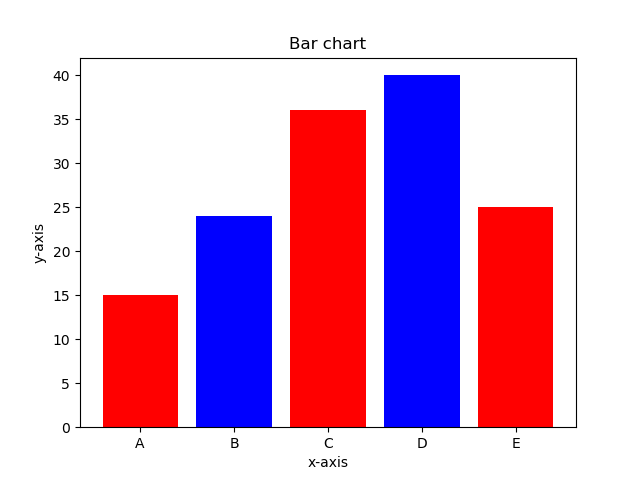


**Bar plot:**

|  |
| --- |
| # importing the required module  import matplotlib.pyplot as plt  # x-coordinates of left sides of bars  left = [1, 2, 3, 4, 5]  # heights of bars  height = [15, 24, 36, 40, 25]  # labels for bars  tick\_label = ['A', 'B', 'C', 'D', 'E']  # plotting a bar chart  plt.bar(left, height, tick\_label=tick\_label, width=0.8, color=['red', 'blue'])  # naming the x-axis  plt.xlabel('x-axis')  # naming the y-axis  plt.ylabel('y-axis')  # plot title  plt.title('Bar chart')  # function to show the plot  plt.show() |

**Code explanation:**

Here, used plt.bar() function to plot a bar chart. X-coordinates of left side of bars are passed along with heights of bars.



**Histogram:**

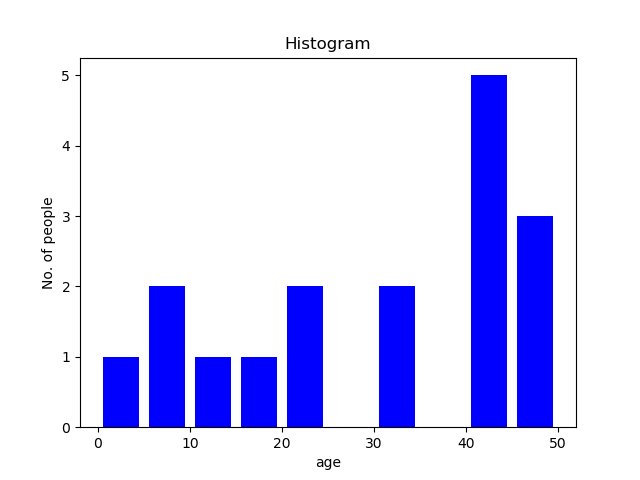
|  |
| --- |
| # importing the required module  import matplotlib.pyplot as plt  # frequencies  ages = [2, 5, 70, 40, 30, 45, 50, 45, 43, 40, 44, 60, 7, 13, 57, 18, 90, 77, 32, 21, 20, 40]  # setting the ranges and no. of intervals  range = (0, 50)  bins = 10  # plotting a histogram  plt.hist(ages, bins, range, color='blue', histtype='bar', rwidth=0.8)  # x-axis label  plt.xlabel('age')  # frequency label  plt.ylabel('No. of people')  # plot title  plt.title('Histogram')  # function to show the plot  plt.show() |

**Code explanation:**

Here, used plt.hist() function to plot a histogram. Frequencies are passed as the ages list.

Range could be set by defining a tuple containing min and max value. These range could be negative to positive.

Next step is to “bin” the range of values—that is, divide the entire range of values into a series of intervals—and then count how many values fall into each interval. Here we have defined bins = 10. So, there are a total of 50/10 = 5 intervals.



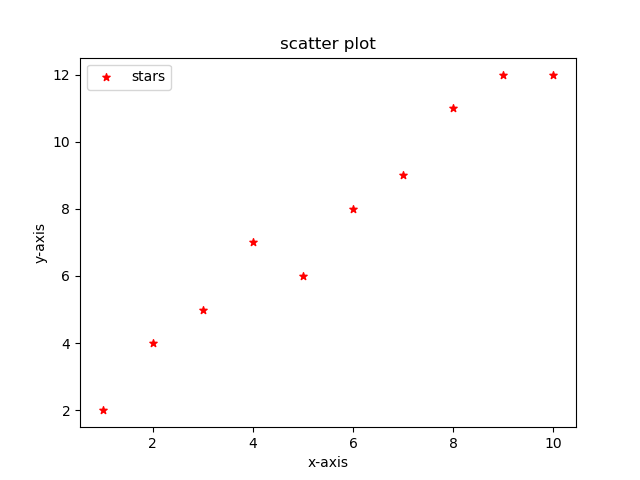
**Scatter Plot:**

|  |
| --- |
| # importing the required module  import matplotlib.pyplot as plt  # x-axis values  x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  # y-axis values  y = [2, 4, 5, 7, 6, 8, 9, 11, 12, 12]  # plotting points as a scatter plot  plt.scatter(x, y, label="stars", color="red", marker="\*", s=30)  # x-axis label  plt.xlabel('x-axis')  # frequency label  plt.ylabel('y-axis')  # plot title  plt.title('scatter plot')  # showing legend  plt.legend()  # function to show the plot  plt.show() |

**Code explanation:**

Here, used plt.scatter() function to plot a scatter plot. Like a line, defined x and corresponding y - axis values as well.

Marker argument is used to set the character to use as marker. Its size can be defined using s parameter.



**Pie-chart:**

|  |
| --- |
| # importing the required module  import matplotlib.pyplot as plt  # defining labels  vehicles = ['taxi', 'bike', 'cars', 'train']  # portion covered by each label  slices = [3, 7, 8, 6]  # color for each label  colors = ['r', 'y', 'g', 'b']  # plotting the pie chart  plt.pie(slices, labels=activities, colors=colors,  startangle=90, shadow=True, explode=(0, 0, 0.1, 0),  radius=1.2, autopct='%1.1f%%')  # plotting legend  plt.legend()  # showing the plot  plt.show() |

**Code explanation:**

Here, plotted a pie chart by using plt.pie() method.

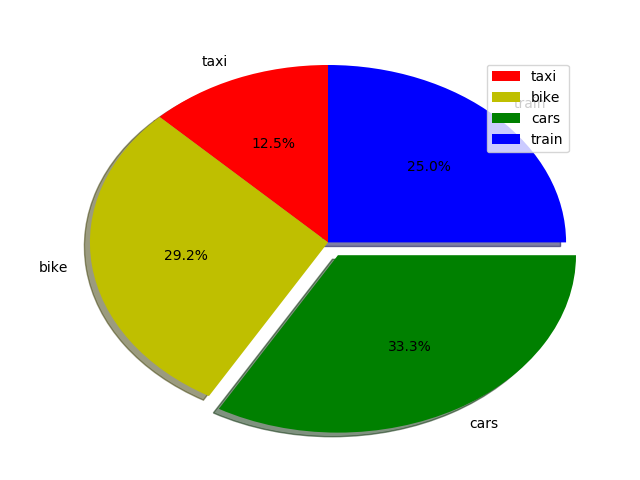
First of all, defined the labels using a list called vehicles. Then, portion of each label can be defined using another list called slices. Color for each label is defined using a list called colors.

shadow = True will show a shadow beneath each label in pie-chart.

startangle rotates the start of the pie chart by given degrees counterclockwise from the x-axis.

explode is used to set the fraction of radius with which we offset each wedge.

autopct is used to format the value of each label. Here, we have set it to show the percentage value only upto 1 decimal place.



**Plotting curves:**

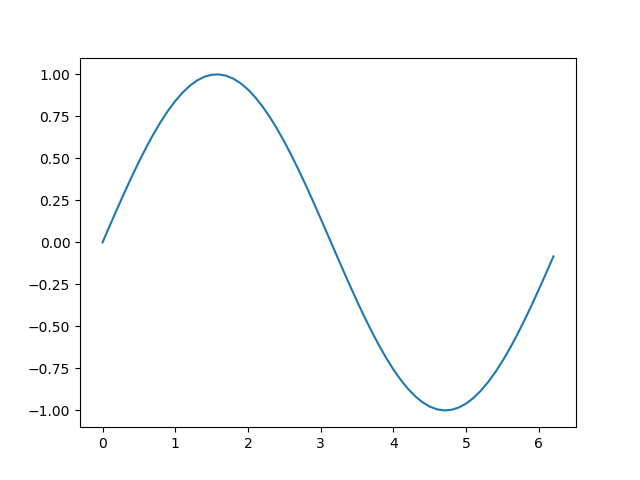
|  |
| --- |
| # importing the required module  import matplotlib.pyplot as plt  import numpy as np  # setting the x - coordinates  x = np.arange(0, 2 \* (np.pi), 0.1)  # setting the corresponding y - coordinates  y = np.sin(x)  # potting the points  plt.plot(x, y)  # function to show the plot  plt.show() |

**Code explanation:**

To set the x - axis values, used np.arange() method in which first two arguments are for range and third one for step-wise increment. The result is a numpy array.

To get corresponding y-axis values, used predefined np.sin() method on the numpy array.

Finally, plot the points by passing x and y arrays to the plt.plot() function.



**References:**

<https://matplotlib.org/2.2.3/index.html>

<https://matplotlib.org/2.2.3/tutorials/index.html>

<https://www.geeksforgeeks.org/python-introduction-matplotlib/>