Rabindranath World School,
DLF Phase III, Gurgaon. 122002

Subject – IP
Class – XII
HANDOUTS

Data Visualization

When data is shown in the form of pictures, it becomes easy for the user to understand it. So representing the data in the form of pictures or graph is called “data visualization”. It represents (patterns, trends, correlations etc.) in data and thereby helps decision makers to understand the meaning of data for making decision in business.

- Matplotlib is a python library which provides many interfaces and function to present data in 2D graphics. We can say, Matplotlib is a high quality plotting library of Python.
- Matplotlib library offers many different collections of sub modules; Pyplot is one such sub module.
- Pyplot is a collection of methods within Matplotlib library which allows user to construct 2D plots easily.

Installing and importing Matplotlib-

With Anaconda : if we have installed python using Anaconda, then Matplotlib is already installed on your computer. We can

check this Anaconda Navigator, by Clicking on Environment and then scroll down to find Matplotlib.

With Standard Installation : First we need to download wheel package of Matplotlib as per Python's version installed and platform (OS).

With Standard Installation : Next we need to install it by giving following command:

python –m pip install –U pip python –m

pip install –U matplotlib

To use Pyplot for data visualization, we have to first import it in our python environment.

import matplotlib.pyplot

But this method will require to type every command as -

matplotlib.pyplot.Command

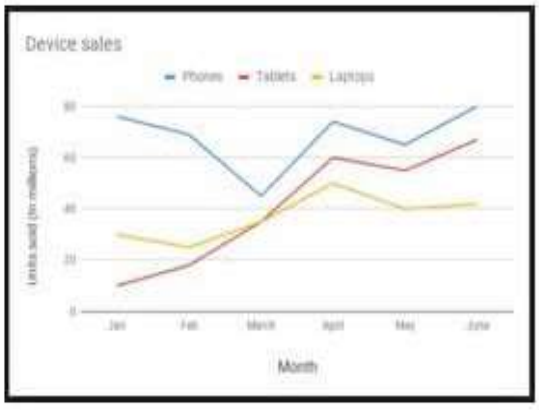
Another method is-

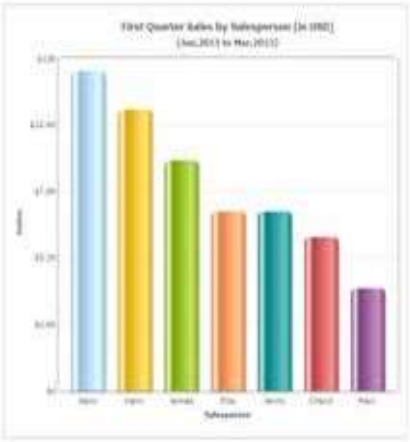
import matplotlib.pyplot as plt

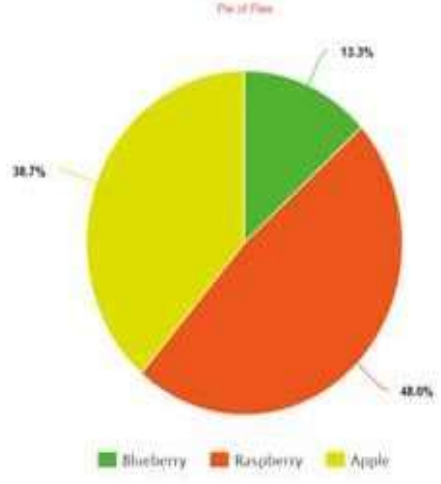
(Now we can qualify command as plt.Command) (plt is just identifier we can take any name)

Basics of Simple Plotting

- There are various types of chart we can use to visualize the data elements like:
- **Line Chart:** it displays information as a series of data points called 'markers' connected by straight line
- **Bar Chart:** it present category wise data in rectangular bars with length proportional to the values. It can be horizontal and vertical.
- **Pie Chart:** is a circular chart divided into slices to represent the value/percentage.







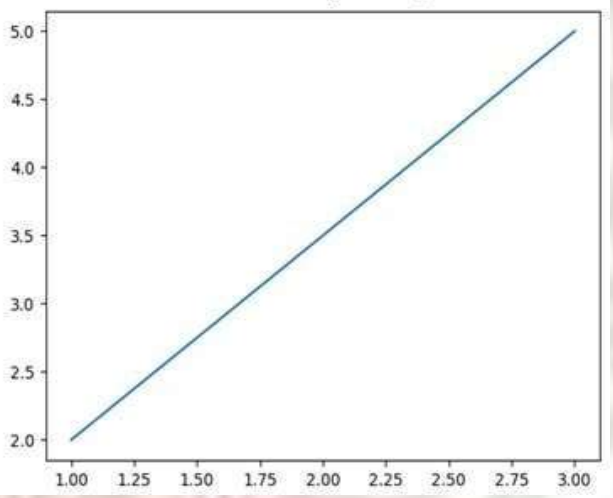
Line Chart or Line Graph

Line graph is a simple graph that shows the result in the form of lines. To create a line graph we need x and y coordinates. For example-

plt.plot(x, y, 'colorname')

plot() function is used to draw line chart. In previous examples we already observed this. Let us draw and use various attributes available with plot().

```
#Simple line draw
import matplotlib.pyplot as plt
x=[1,2,3]
y=[2,3.5,5]
plt.plot(x,y)
plt.show()
```

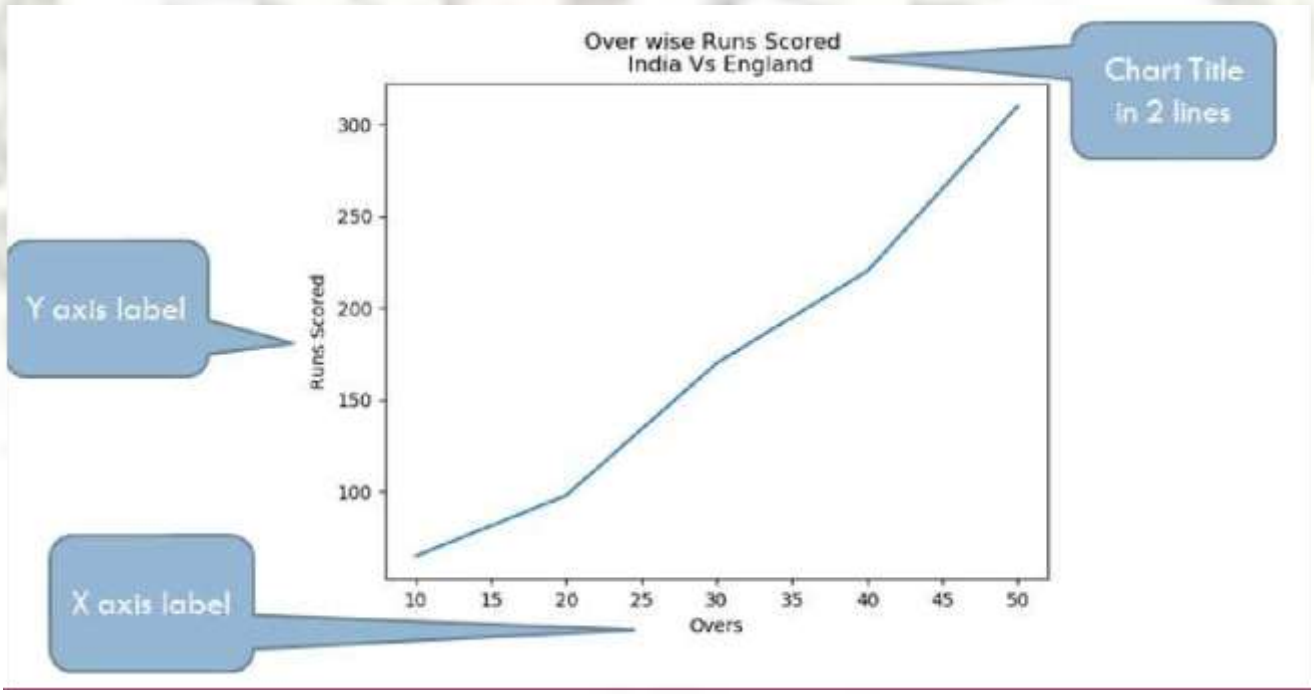



```
# Setting Label of X and Y axis and also title for chart
import matplotlib.pyplot as plt
x = [ 10, 20, 30, 40, 50]
y = [65, 98, 170, 220, 310]
plt.xlabel('Overs')
plt.ylabel('Runs Scored')
plt.title('Over wise Runs Scored \n India Vs England')
plt.plot(x,y)
plt.show()
```

Program:

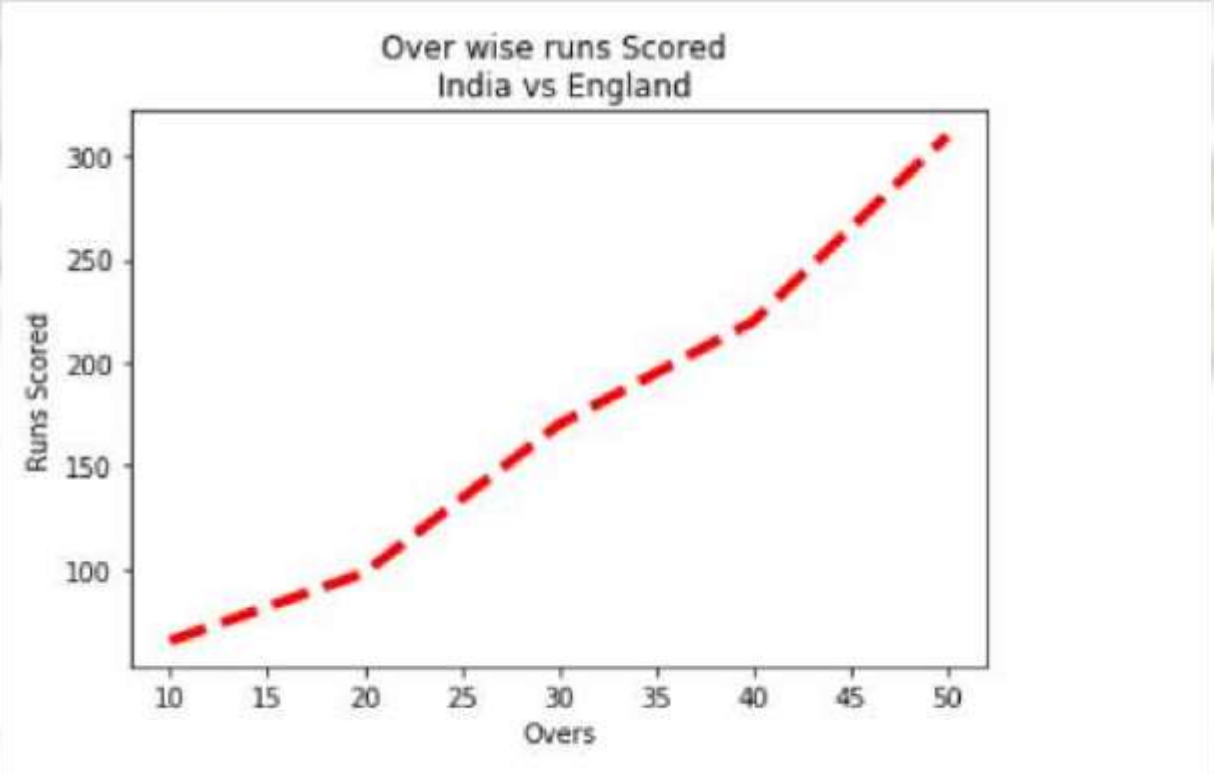
```
# Setting Label of X and Y axis and also title for chart
import matplotlib.pyplot as plt
x = [ 10, 20, 30, 40, 50]
y = [65, 98, 170, 220, 310]
plt.xlabel('Overs')
plt.ylabel('Runs Scored')
plt.title('Over wise Runs Scored \n India Vs England')
plt.plot(x,y)
plt.show()
```

Output:



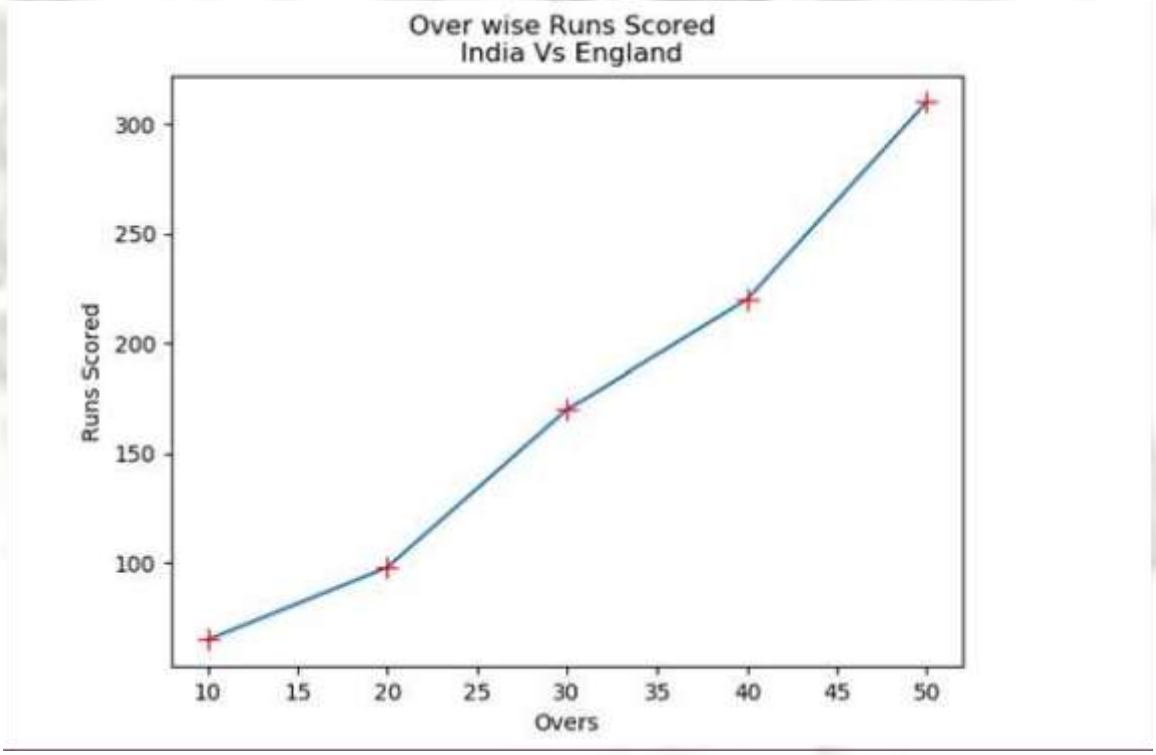
Changing line color and line width and line style :

```
import matplotlib.pyplot as plt
x=[10,20,30,40,50]
y=[65,98,170,220,310]
plt.xlabel('Overs')
plt.ylabel('Runs Scored')
plt.title('Over wise runs Scored \n India vs England')
plt.plot(x,y,'r',linewidth=4,linestyle='dashed')
plt.show()
```



Changing Marker Type, Size and Color

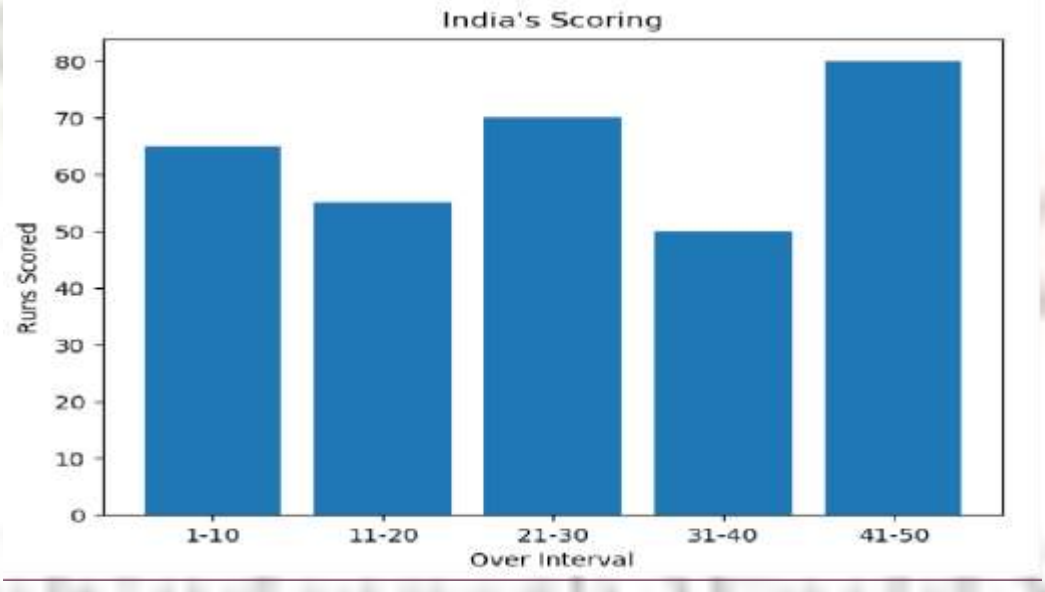
```
# Setting Label of X and Y axis and also title for chart
import matplotlib.pyplot as plt
x = [ 10, 20, 30, 40, 50]
y = [65, 98, 170, 220, 310]
plt.xlabel('Overs')
plt.ylabel('Runs Scored')
plt.title('Over wise Runs Scored \n India Vs England')
plt.plot(x,y,marker='+',markersize='10', markeredgecolor='red')
plt.show()
```



Bar Graph

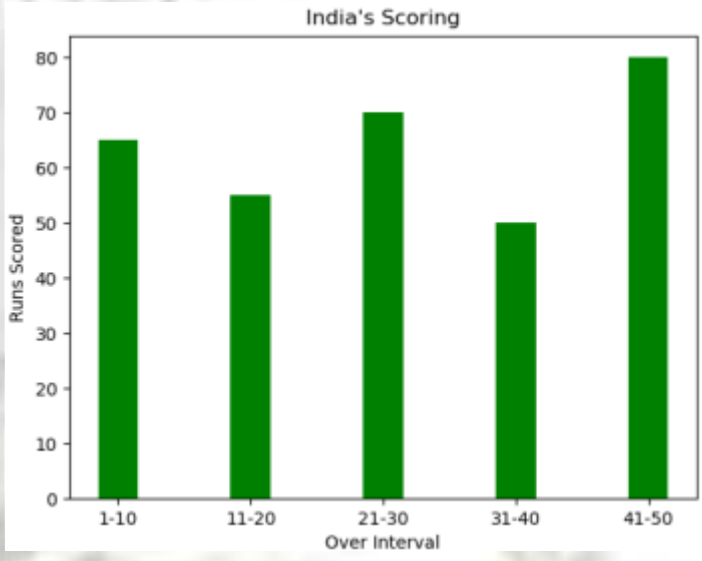
A bar graph is used to represents data in the form of vertical or horizontal bars. It is useful to compare the quantities.

```
# Bar chart example
import matplotlib.pyplot as pl
import numpy as np
OverRange1=['1-10','11-20','21-30','31-40','41-50']
RunsScored1=[65,55,70,50,80]
pl.bar(OverRange1,RunsScored1)
pl.xlabel('Over Interval')
pl.ylabel('Runs Scored')
pl.title('India\'s Scoring')
pl.show()
```

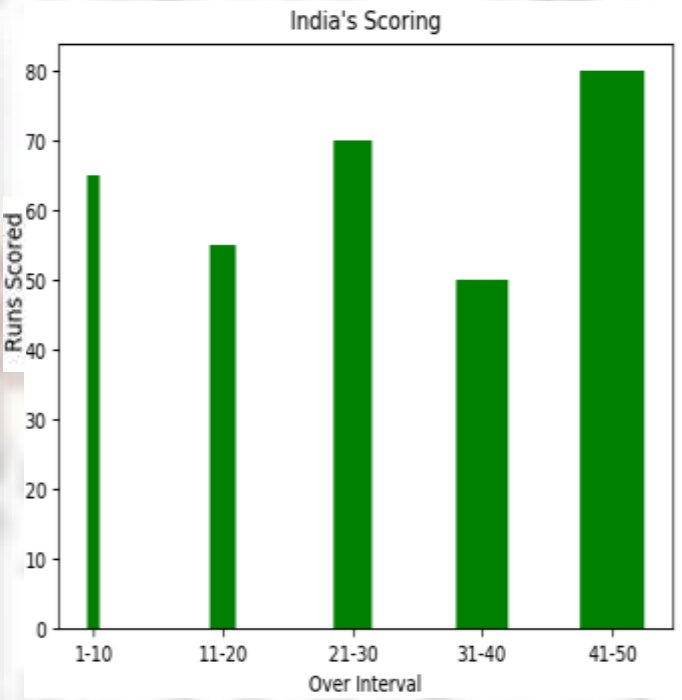


Changing Width, Color in Bar Chart :

```
# Bar chart example
import matplotlib.pyplot as pl
import numpy as np
OverRange1=['1-10','11-20','21-30','31-40','41-50']
RunsScored1=[65,55,70,50,80]
pl.bar(OverRange1,RunsScored1,width=0.3,color='g')
#default width=0.5
pl.xlabel('Over Interval')
pl.ylabel('Runs Scored')
pl.title('India\'s Scoring')
pl.show()
```

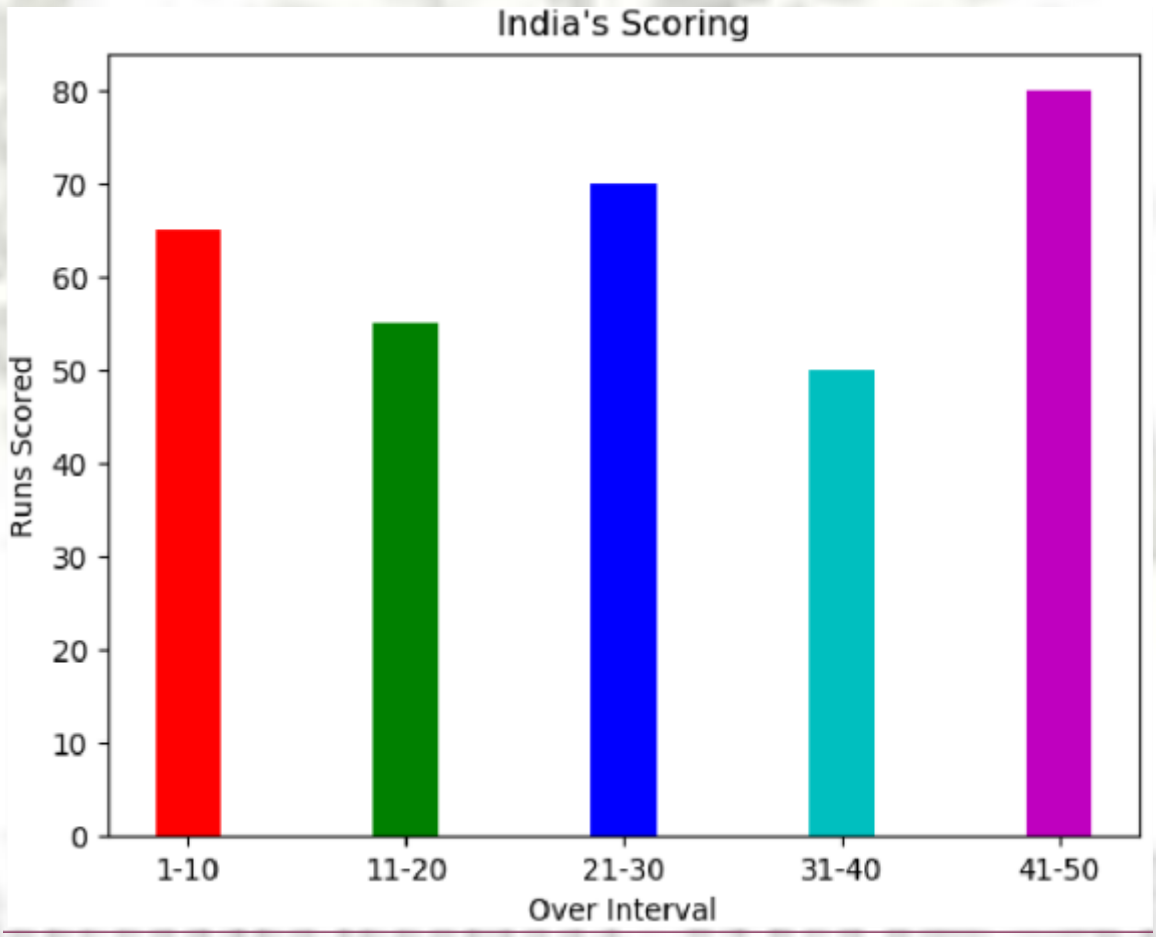



```
# Bar chart example
import matplotlib.pyplot as pl
import numpy as np
OverRange1=['1-10','11-20','21-30','31-40','41-50']
RunsScored1=[65,55,70,50,80]
pl.bar(OverRange1,RunsScored1,width=[0.1,0.2,0.3,0.4,0.5],color='g')
#default width=0.5
pl.xlabel('Over Interval')
pl.ylabel('Runs Scored')
pl.title('India\'s Scoring')
pl.show()
```



Example 2-

```
# Bar chart example
import matplotlib.pyplot as pl
import numpy as np
OverRange1=['1-10','11-20','21-30','31-40','41-50']
RunsScored1=[65,55,70,50,80]
pl.bar(OverRange1,RunsScored1,width=0.3,color=['r','g','b','c','m'])
#default width=0.5
pl.xlabel('Over Interval')
pl.ylabel('Runs Scored')
pl.title('India\'s Scoring')
pl.show()
```

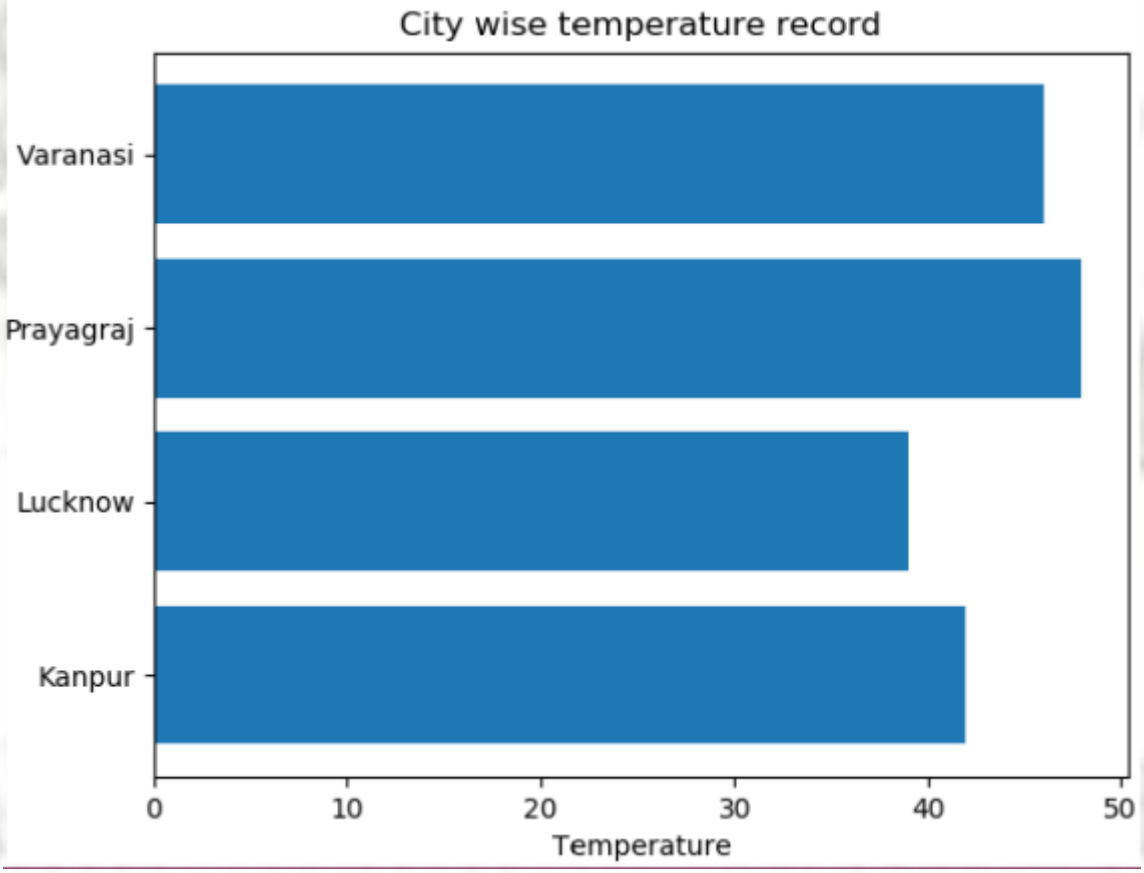


Horizontal Bar Graph:

barh() is used to draw horizontal bar graph.

```
import matplotlib.pyplot as pl
import numpy as np
Cities=['Kanpur','Lucknow','Prayagraj','Varanasi']
Temp=[42,39,48,46]
pl.barh(Cities,Temp)
pl.xlabel('Temperature')
pl.ylabel('Cities')
pl.title('City wise temperature record')
pl.show()
```

Output-



Multiple Bar Graph:

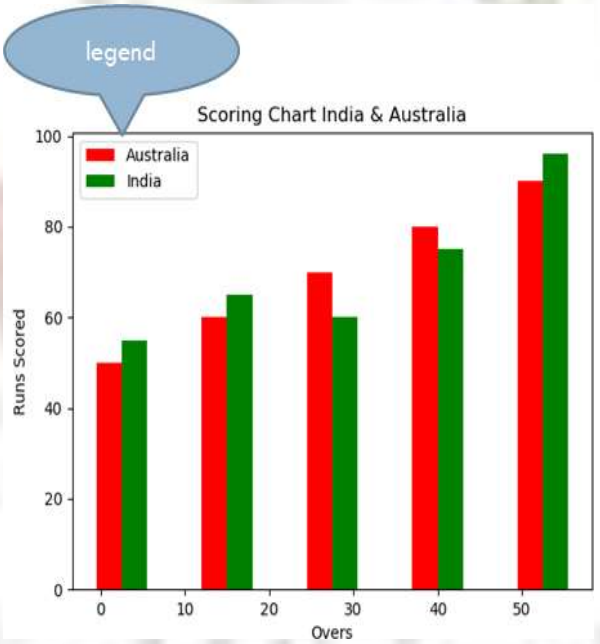
To draw multiple bar chart:

- Decide the no. of X points, we can use arange() or linspace() function to find no. of points based on the length of values in sequence.
- Decide the thickness of each bar and accordingly adjust X point on X-axis
- Give different color to different data ranges
- The width remains the same for all ranges being plotted
- Call plot() for each data range

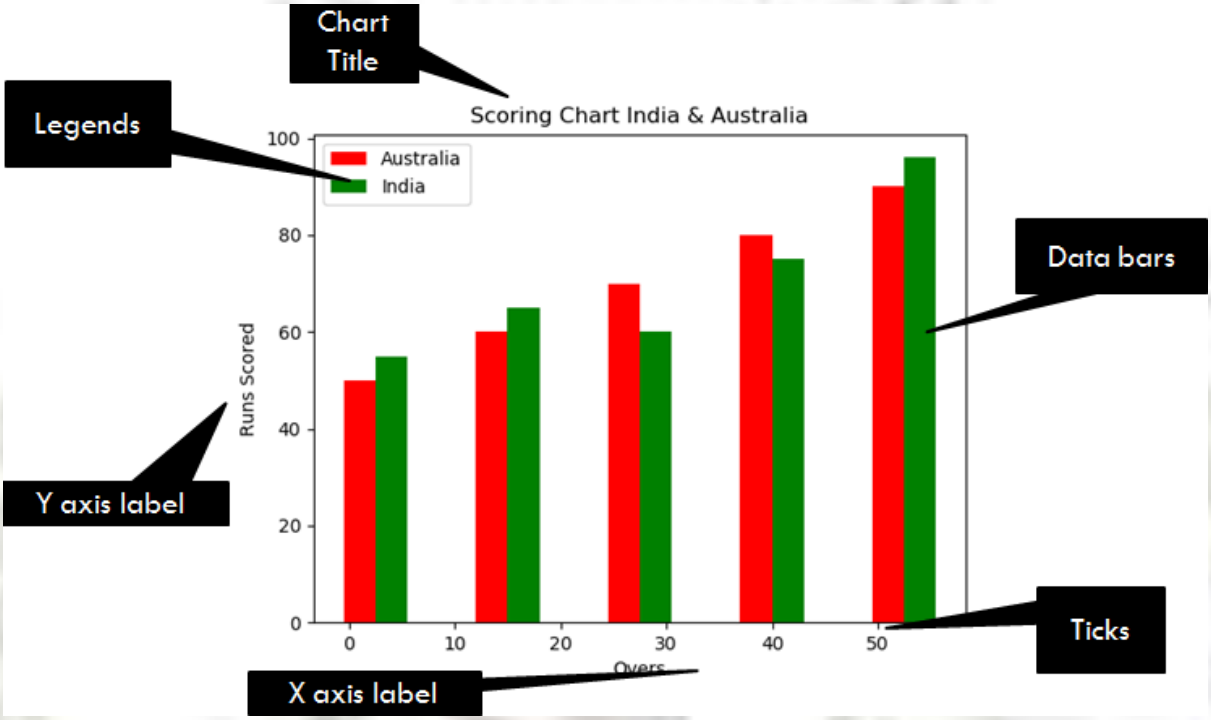
```
# Multiple Bar
import matplotlib.pyplot as pl
import numpy as np
a=[50,60,70,80,90]
b=[55,65,60,75,96]
x = np.linspace(1,51,5)
pl.bar(x,a,width=3,color='r',label='Australia')
pl.bar(x+3,b,width=3,color='g',label='India')
pl.xlabel('Overs')
pl.ylabel('Runs Scored')
pl.title('Scoring Chart India & Australia')
pl.legend()
pl.show()
```

Legend label

To display legend

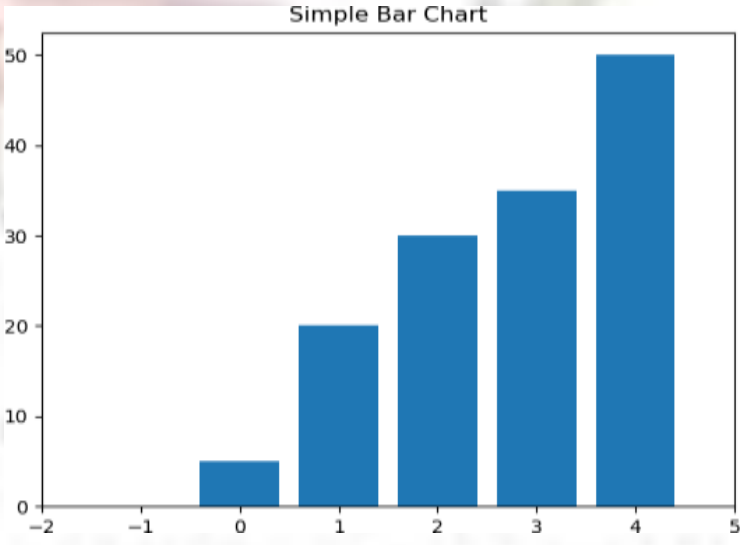


Anatomy of chart:-



Setting Limits and Ticks

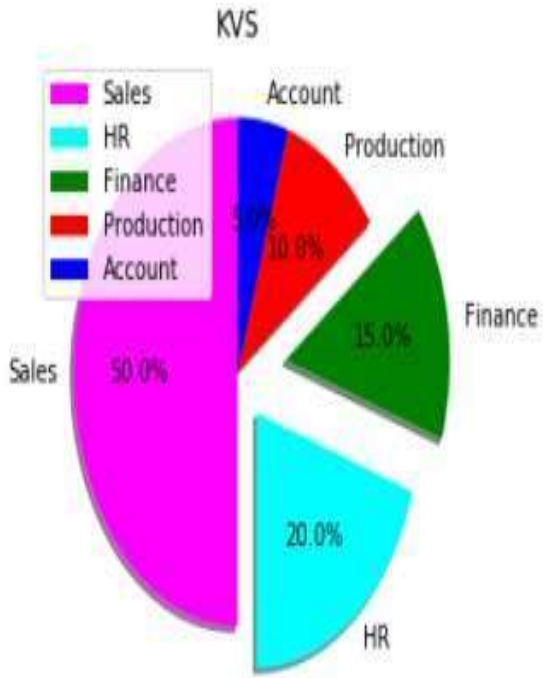
```
import matplotlib.pyplot as pl
import numpy as np
x = np.arange(5)
y = [5.0,20.0, 30.0,35.0,50.0]
pl.xlim(-2.0,5.0)
pl.bar(x,y)
pl.title("Simple Bar Chart")
pl.show()
```



Pie Chart

A pie chart shows a circle that is divided into sectors and each sector represents a proportion of the whole.

```
import matplotlib.pyplot as plt
slices=[50,20,15,10,5]
dept=['Sales','HR','Finance','Production','Account']
cols=['magenta','cyan','green','red','blue']
exp=[0,0.2,0.3,0,0]
plt.pie(slices, labels=dept, colors=cols, startangle=90, explode=exp, shadow=True, autopct='%.1f%%')
plt.title('KVS')
plt.legend()
plt.show()
```



- **Sometimes we want to emphasize on one or more slice and show them little pulled out. This feature is called explode in pie chart.**

- **If we want to explode or stand out 2nd and 3rd slice out of 5 slices to 0.2 and 0.3 unit respectively , explode will be [0,0.2,0.3,0,0]. The value of explode vary from 0.1 to 1 to show that how much a slice will come out of pie chart.**

autopct : allows to view percentage of share in a pie chart-

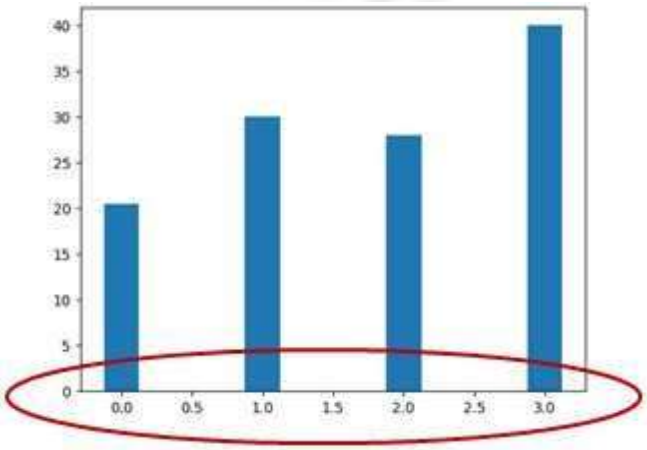
The option autopct='%.1f %%' indicates how to display the percentages on the slices. Here %.1 shows that the percentage value should be displayed with 1 digit after decimal point. The next two % symbols indicates that only one symbol is to be displayed.

Shadow option-

Shadow= True indicates that the pie chart should be displayed with a shadow. This will improve the look of the chart.

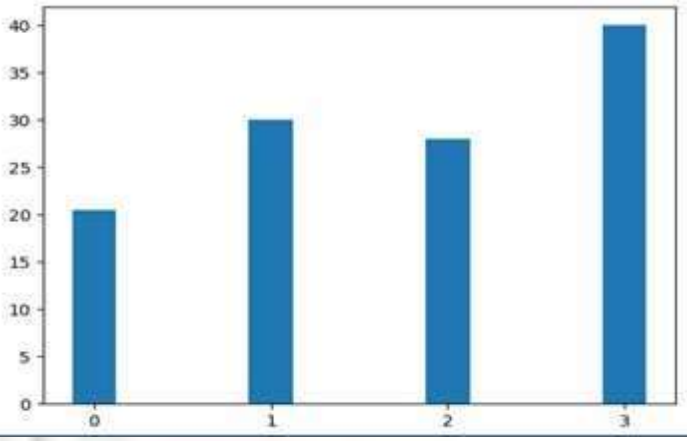
Setting ticks of Bar Graph:-

```
# Example ticks
import matplotlib.pyplot as pl
x = range(4)
y = [20.5,30,28,40]
pl.bar(x,y,width=0.25)
pl.show()
```



By default the ticks are appearing at data point 0.5 apart

```
# Example ticks
import matplotlib.pyplot as pl
x = range(4)
y = [20.5,30,28,40]
pl.xticks([0,1,2,3])
pl.bar(x,y,width=0.25)
pl.show()
```



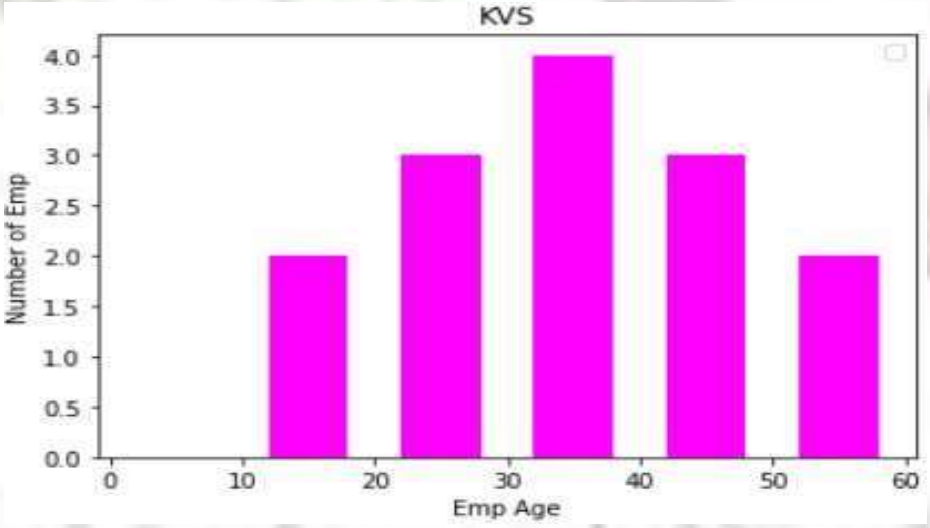
Histogram

Histogram shows distribution of values. Histogram is similar to bar graph but it is useful to show values grouped in bins or intervals.

For example- we can collect the age of each employee in an office and show it in the form of a histogram to know how many employees are there in the range 0-10 years, 10-20 years and so on. For this we can create histogram like this-

```
import matplotlib.pyplot as plt
age=[22,32,35,45,55,14,26,19,56,44,48,33,38,28]
years=[0,10,20,30,40,50,60]
plt.hist(age, bins=years, color='magenta', histtype='bar', rwidth=.6)
plt.xlabel('Emp Age')
plt.ylabel('Number of Emp')
plt.title('KVS')
plt.legend()
plt.show()
```

Output-

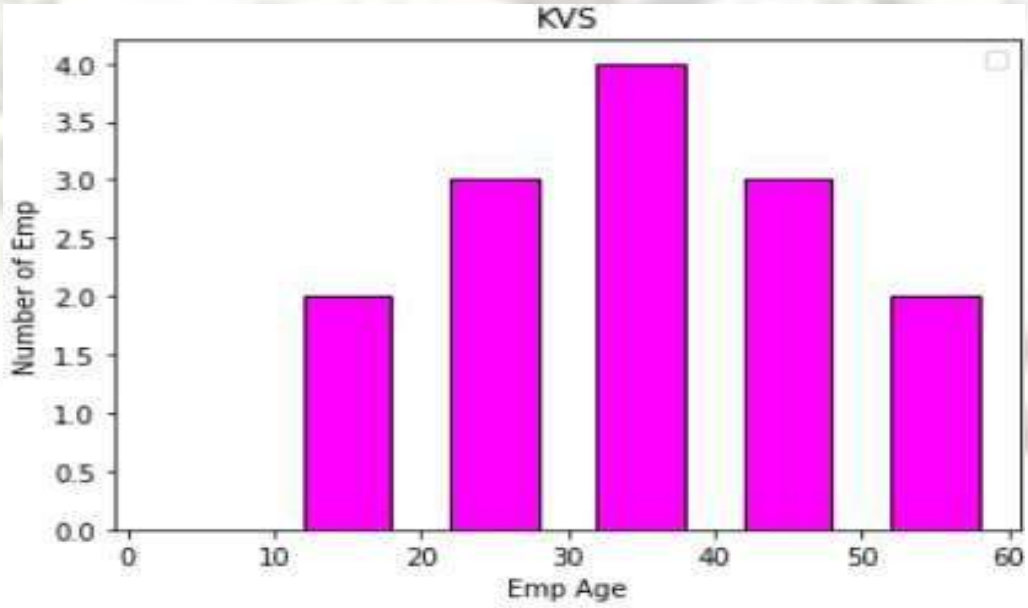


rwidth=0.6 means that the bars width is 60%. There will be a gap of 40% space before and after the bar.

Example 2-

```
import matplotlib.pyplot as plt
age=[22,32,35,45,55,14,26,19,56,44,48,33,38,28]
years=[0,10,20,30,40,50,60]
plt.hist(age, bins=years, color='magenta', histtype='bar', edgecolor='black',rwidth=.6)
plt.xlabel('Emp Age')
plt.ylabel('Number of Emp')
plt.title('KVS')
plt.legend()
plt.show()
```

Output-



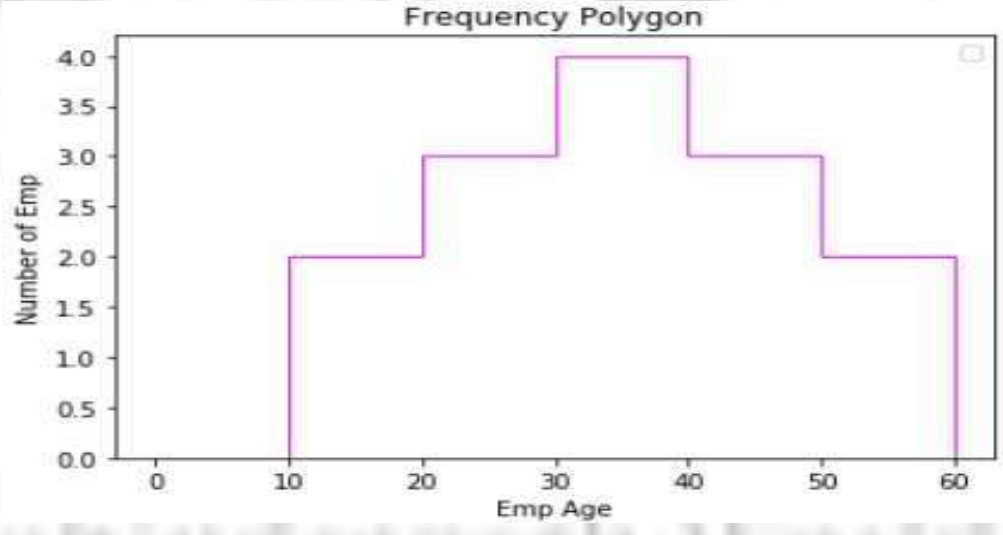
Note- edgecolor is used to define the color of edge around bar.

Frequency Polygons

Frequency polygon is a way for understanding the shape of distributions. It connects the top center point of each bins and then we get the relative frequency polygon. It has the same purpose as the histogram have but is used specially for comparing sets of data.

```
import matplotlib.pyplot as plt
age=[22,32,35,45,55,14,26,19,56,44,48,33,38,28]
years=[0,10,20,30,40,50,60]
plt.hist(age, bins=years, color='magenta', histtype='step')
plt.xlabel('Emp Age')
plt.ylabel('Number of Emp')
plt.title('Frequency Polygon')
plt.legend()
plt.show()
```

Output-

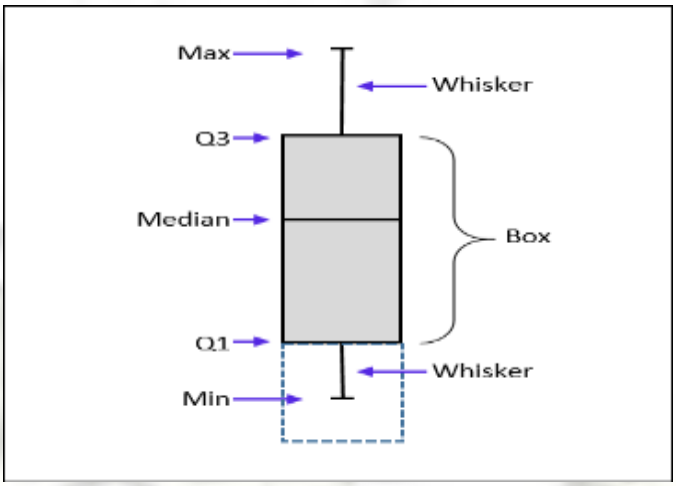


histtype=’step’ creates frequency polygon by using hist().

Box Plot

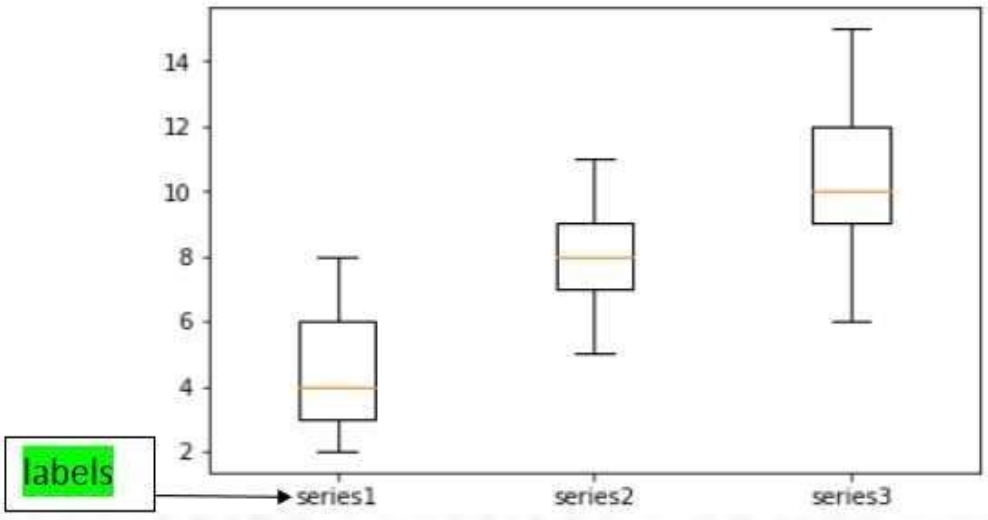
A Box plot is graphical representation of the five number summary of given data set. It includes-

1. Maximum
2. 2. Minimum
3. 1st Quartile
4. 2ND Quartile (Median)
5. 3RD Quartile



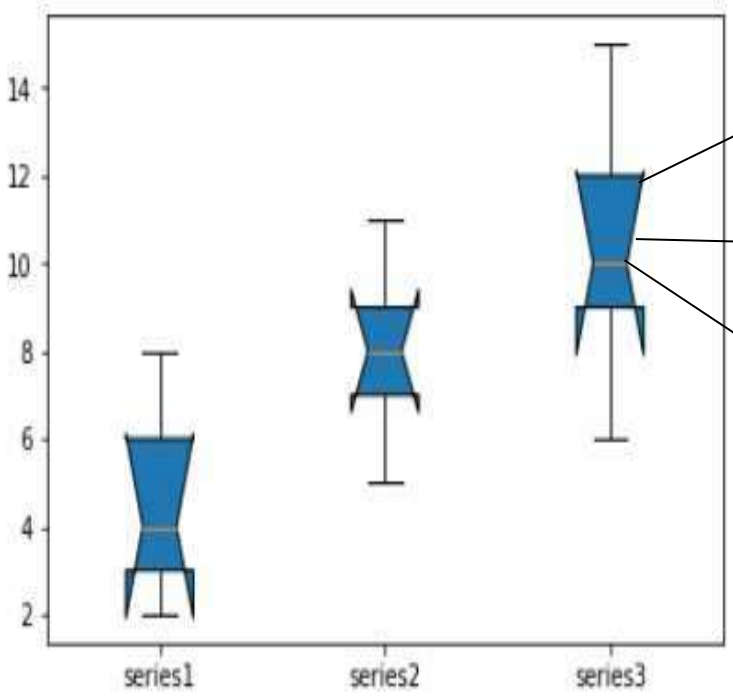
Example 1-

```
import matplotlib.pyplot as plt
val1=[2,3,4,6,8]
val2=[5,7,8,9,11]
val3=[6,9,10,12,15]
data=[val1,val2,val3]
plt.boxplot(data, labels=['series1','series2','series3'])
plt.show()
```



Example 2-

```
import matplotlib.pyplot as plt
val1=[2,3,4,6,8]
val2=[5,7,8,9,11]
val3=[6,9,10,12,15]
data=[val1,val2,val3]
plt.boxplot(data, labels=['series1','series2','series3'],patch_artist=True, notch=True)
plt.show()
```



If notch=True creates a notched box plot otherwise creates rectangular box plot

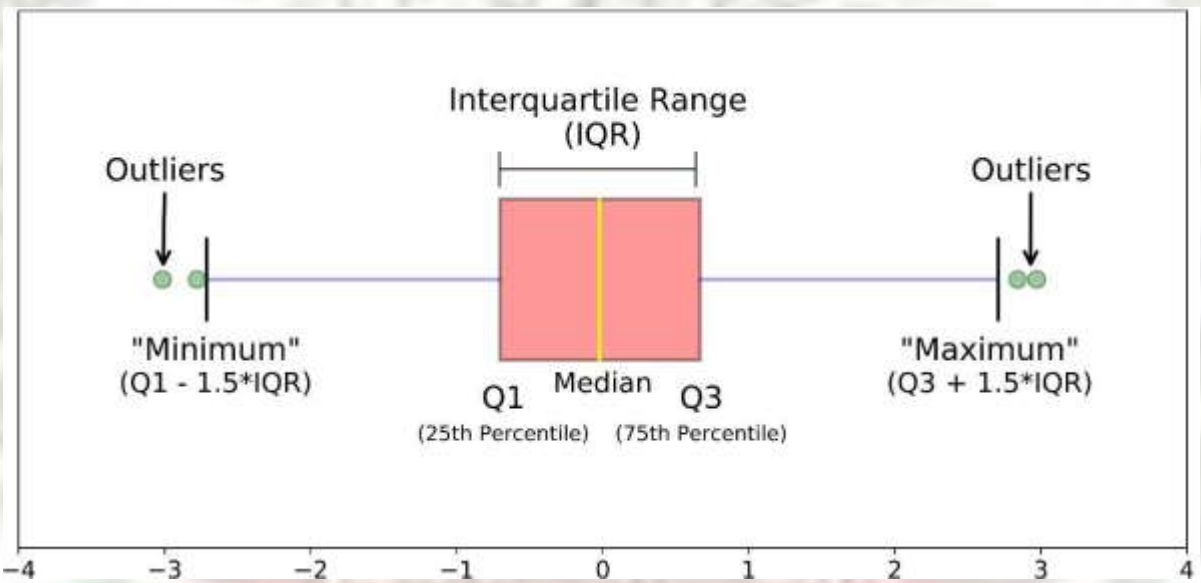
Patch_artist=True fills the box plot with color

More about Box Plot:

IQR (Inter Quartile Range) = It always lies between 25^{th} to 75^{th} percentile. i.e. (Q3 – Q1)

Minimum= (Q1 - 1.5 * IQR)

Maximum= (Q3 + 1.5 * IQR)



Scatter Chart

A **scatter plot** is a type of **plot** that shows the data as a collection of points in the form of dots, and shows the relationship between two variables - one plotted along the x- axis and the other plotted along y-axis.

Syntax-

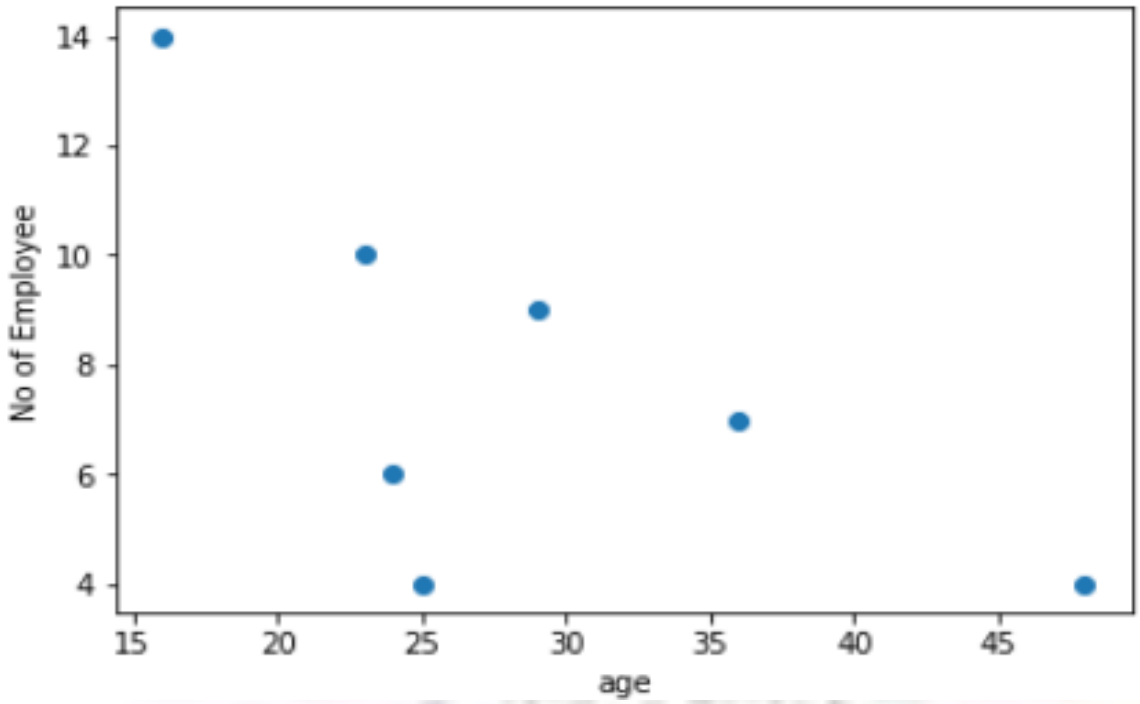
Scatter(x, y, color, marker)

Marker- is a symbol (style) for representing data point. Following is a list of valid marker style-

Marker	Description
's'	**Square Marker**
'o'	**Circle Marker**
'd'	**Diamond Marker**
'x'	**Cross Marker**
'+'	**Plus Marker**
'^'	**Triangle down**
'v'	**Triangle Up**

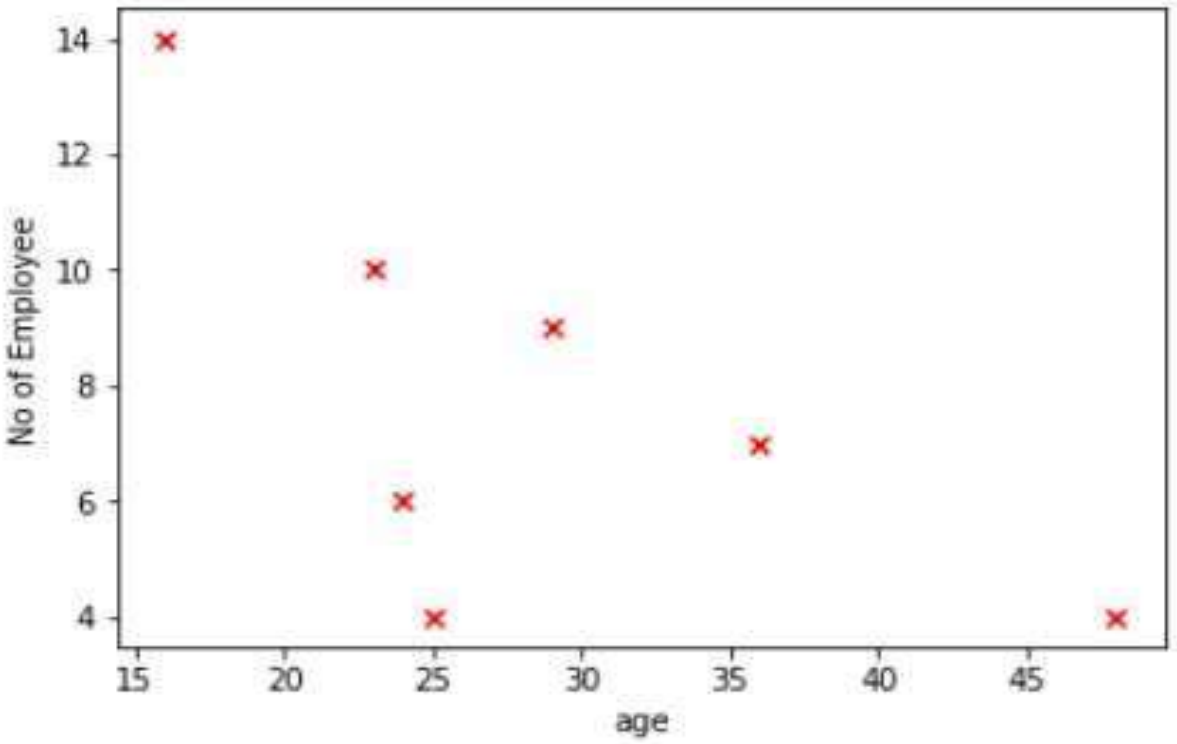
Example 1-

```
import matplotlib.pyplot as plt

x = [25,24,23,16,29,36,48]
y = [4,6,10,14,9,7,4]
plt.scatter(x, y)
plt.xlabel('age')
plt.ylabel('No of Employee')
plt.show()
```



Example -2

```
import matplotlib.pyplot as plt

x = [25,24,23,16,29,36,48]
y = [4,6,10,14,9,7,4]
plt.scatter(x, y, color='red',marker='x')
plt.xlabel('age')
plt.ylabel('No of Employee')
plt.show()
```



Saving Plots or Chartsor graph to file

```
import matplotlib.pyplot as plt

x = [25,24,23,16,29,36,48]
y = [4,6,10,14,9,7,4]
plt.scatter(x, y, color='red',marker='x')
plt.xlabel('age')
plt.ylabel('No of Employee')
plt.savefig('E:\scatter.pdf')
plt.show()
```

By using savefig('Filepath') we can save a plot into a file.