MRA DAV PUBLIC SCHOOL

INFORMATICS PRACTICES

Code No-065

CLASS-XII

Blue Print:

Unit No	Unit Name	Marks
1	Data Handling using Pandas and Data Visualization	30
2	Database Query using SQL	25
3	Introduction to Computer Networks	7
4	Societal Impacts	8
	Practical	30
	Total	100

Data Visualisation: Plotting Data Using Matplotlib

By

Rajesh Verma

What is Data Visualization?

We are living in an era where we have access to a significant amount of data. And eventually, this data has been growing increasingly complex every year. The real problem starts when we fail to manage this considerable data resource and cannot use them in the right way. Here comes data visualization into picture.

Data visualization is the graphical representation of information and data i.e it presents quantitative information in a graphical form. In other words, data visualizations turn large and small datasets into visuals that are easier for the human brain to understand and process.

Why Data Visualization?

1. To make data easier to understand, process and remember.
2. To discover unknown facts, outliers, and trends.
3. To visualize relationships and patterns quickly.
4. To make better and meaningful decisions.

Using Matplotlib for Data Visualisation

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. It is a multi-platform data visualization library built on NumPy arrays. It was introduced by John Hunter in the year 2002.

Installation:

Matplotlib can also be installed using the Python package manager, **pip**. To install Matplotlib with **pip**, open a terminal window and type:

```
pip install matplotlib
```

This command installs Matplotlib in the current working Python environment.

Types of Plot

Matplotlib offers a wide range of plots. In this unit we are going to study following types of plots:

1. Line Plot
2. Bar Plot
3. Pie Plot
4. Scatter Plot
5. Histogram Plot
6. Box Plot

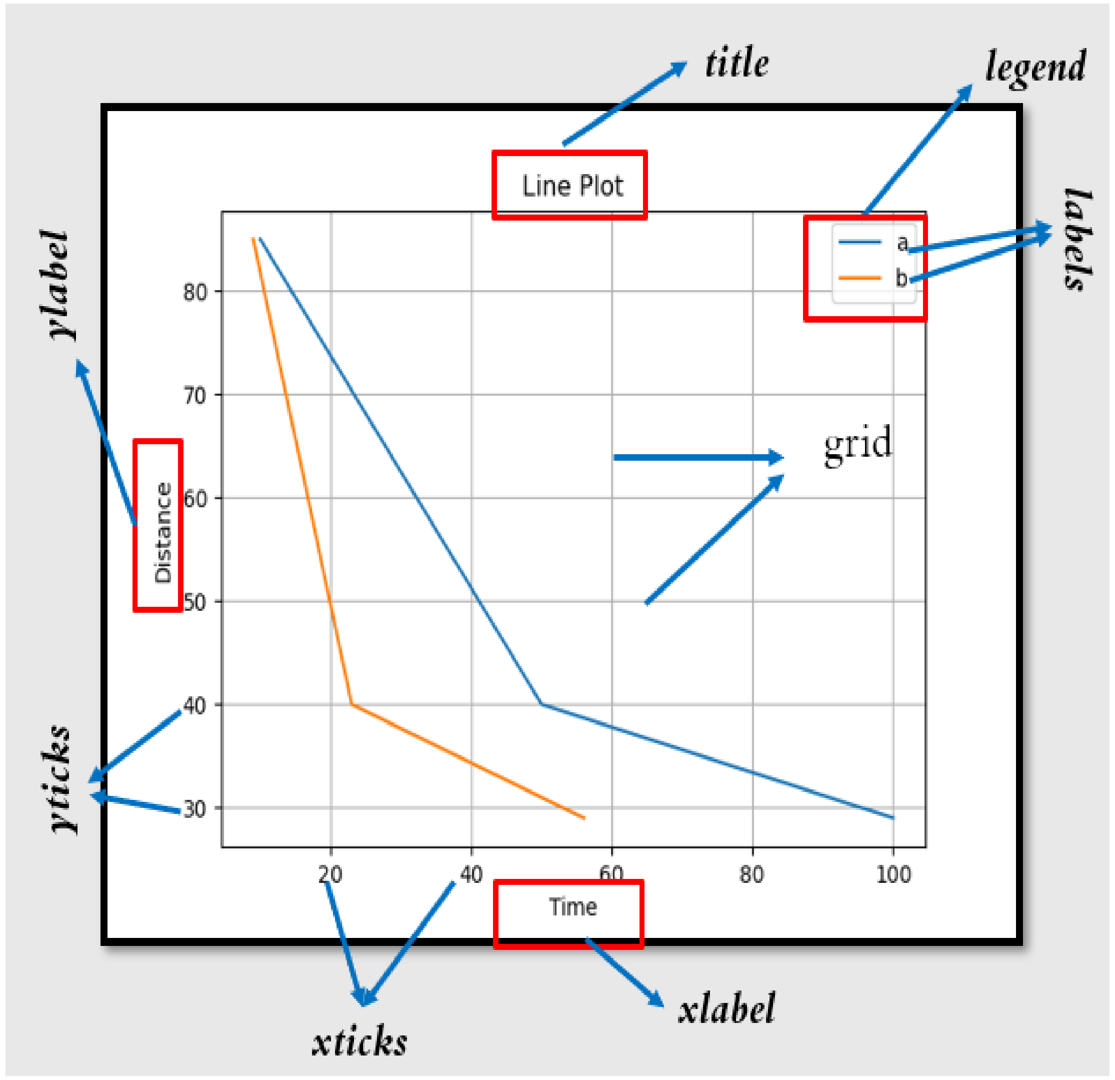
Basic Steps For Plotting A Graph

Import matplotlib.pyplot

•Choose the appropriate plot type like line, bar, pie etc.

•Create a list/array of values according to the type of plot chosen

•Use the built-in functions to set various parameters and to view the plot



Common functions used for plotting graphs through Matplotlib:

Functions	Purpose
xlabel()	to write label name for x-axis
ylabel()	to write label name for y-axis
title()	to write title for the plot
legend()	to show legends
xticks()	To set ticks for x axis
yticks()	To set ticks for y axis
grid()	To show gridlines
show()	to view the plot
savefig()	To save the plot as .png or .pdf at the desired location

All the functions except show() and Savefig() can further use parameters like fontsize=30,color='g',fontname="Arial".

Note: ***Ticks*** are the values used to show specific points on the coordinate axis. It can be a number or a string. Whenever we plot a graph, the axes adjust and take the default ticks. Matplotlib's default ticks are generally sufficient in common situations but are in no way optimal for every plot. The xticks() and yticks() functions are used to customize these ticks as per need.

Integrating Pandas With Matplotlib

The data present in the DataFrame created in pandas can be plotted using the **plot() function** from **matplotlib.pyplot module**.

Syntax:

plt.plot(kind)

Here kind is a string which indicates the type of plot

Kind	Explanation
line	Line plot(default)
bar	Vertical bar plot
barh	Horizontal bar plot
hist	histogram
box	Box plot
pie	Pie plot
scatter	Scatter plot

Line Chart or Line Graph

LINE PLOT

- Line plot is the most common, simplest, and classic type of plot.
- It shows a change in one or more variables over time.
- The plot() method is used to plot the line plot.
- It is often used to visualize a trend in data over intervals of time.

Purpose

A line chart is often used to visualize a trend in data over intervals of time.

Syntax:

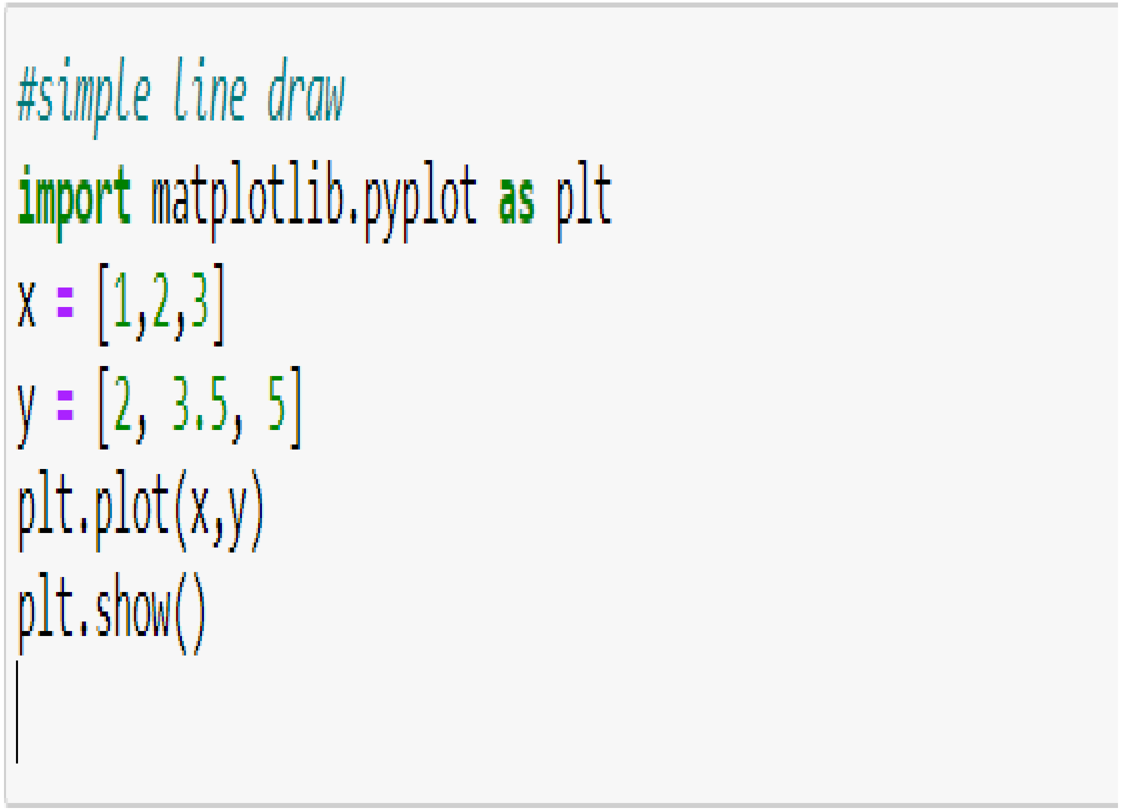
plot(x, y)

here

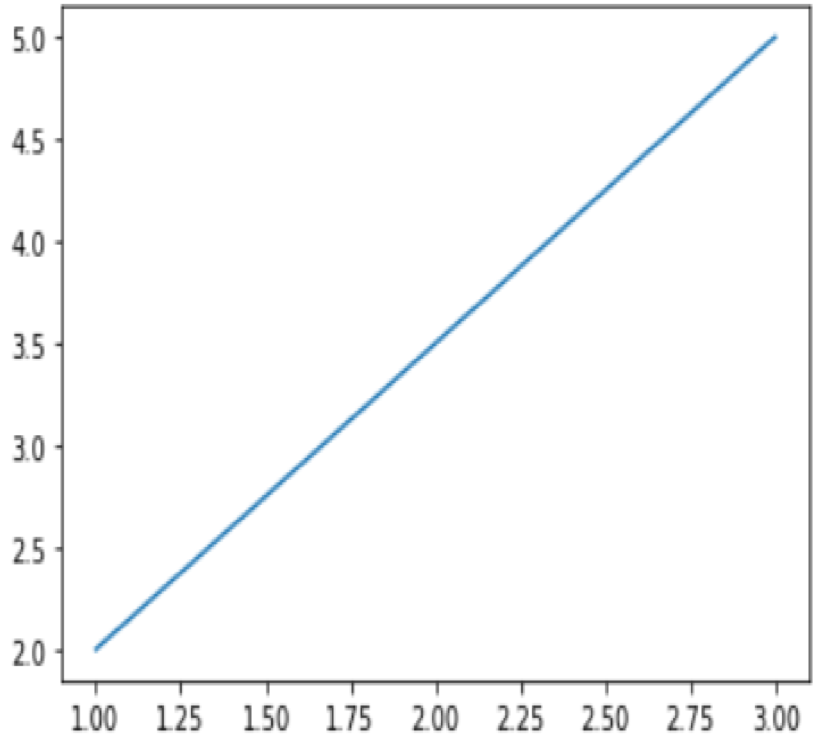
x- list of values for x axis

y- list of values for y axis

Example : Simple Line Draw



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



Customizing The Line Plot

Following are some of the parameters that can be used in the plot function to customize the line chart.

Syntax: plot(x, y, color, marker, markersize, linewidth, linestyle, label)

PARAMETER	PURPOSE
linewidth	to set the width of the line
linestyle	To set line style
label	to set the label for the line chart
marker	To set a symbol that represents a data value
markersize	To set the size of the marker
color	to specify the color of the line

Color

It is possible to format the plot further by changing the colour of the plotted data. We can either use character codes or the color names as values to the parameter color in the plot()

Following is the table of color codes:

Color code	Color Name
b	blue
g	green
r	red
c	cyan
m	magenta
y	yellow
k	black
w	white

Marker

A marker is any symbol that represents a data value in a line chart or a scatter plot

Following is the table of marker codes:

Marker code	Description
s	Square
o	Circle
^	Triangle up
v	Triangle down
>	Triangle right
<	Triangle left
d	Diamond
p	Pentagon
h	Hexagon
8	Octagon
+	Plus
x	Cross

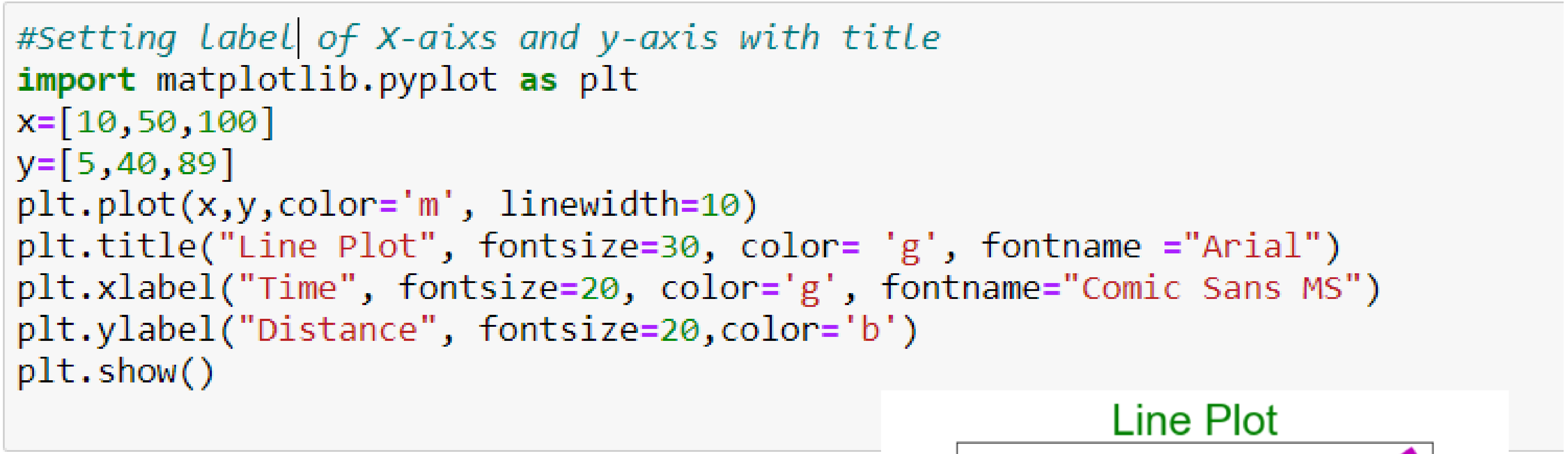
Line Width and Line Style

The linewidth and linestyle property can be used to change the width and the style of the line chart. Linewidth is specified in pixels. The default linewidth is 1 pixel showing a thin line.

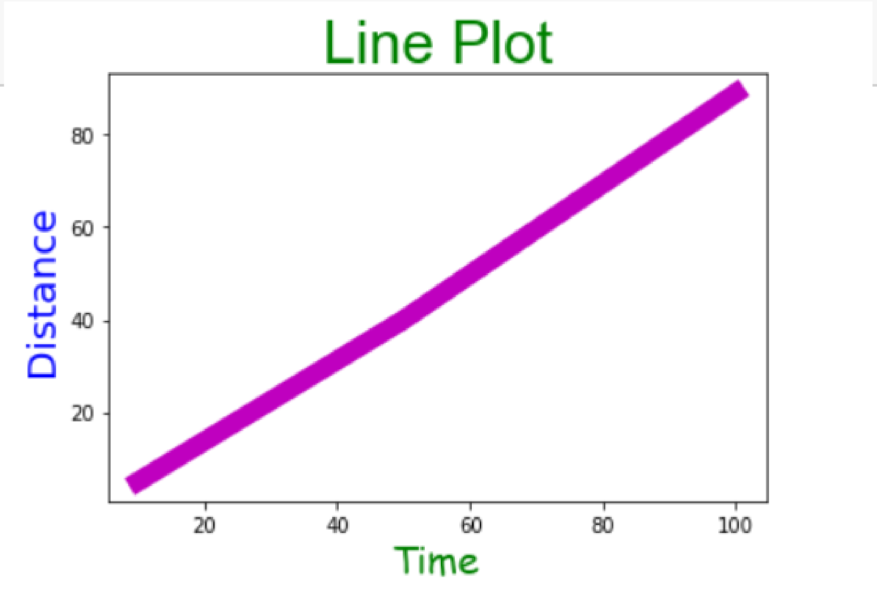
Following is the table of line styles:

Style	PURPOSE
-	Solid Line
--	Dashed Line
-.	Dash-dot Line
:	Dotted Line

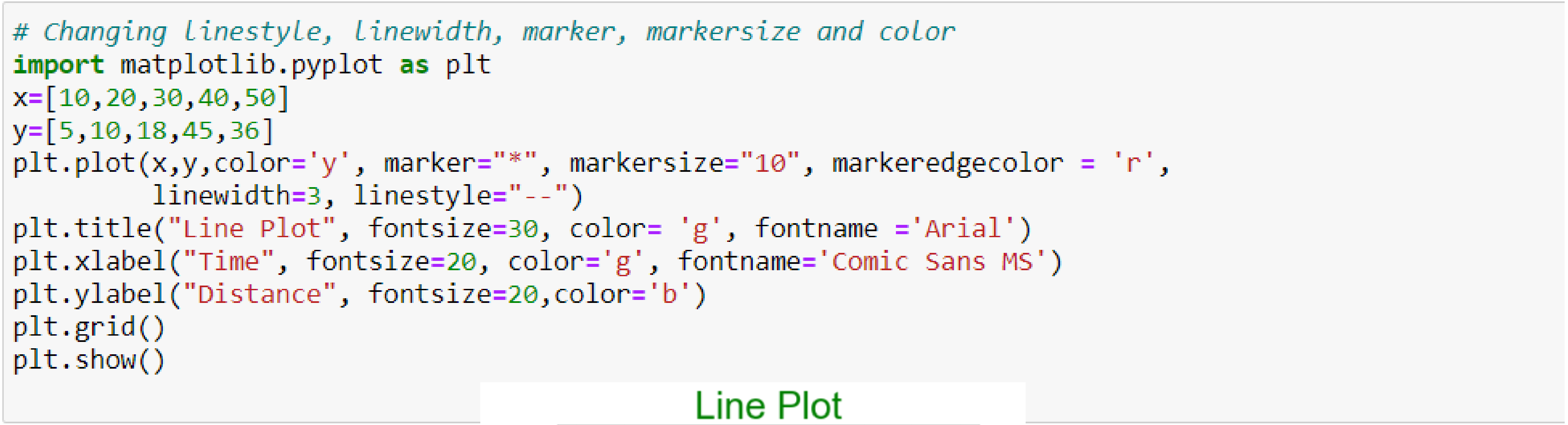
Example : Setting label of X-axis and y-axis with title



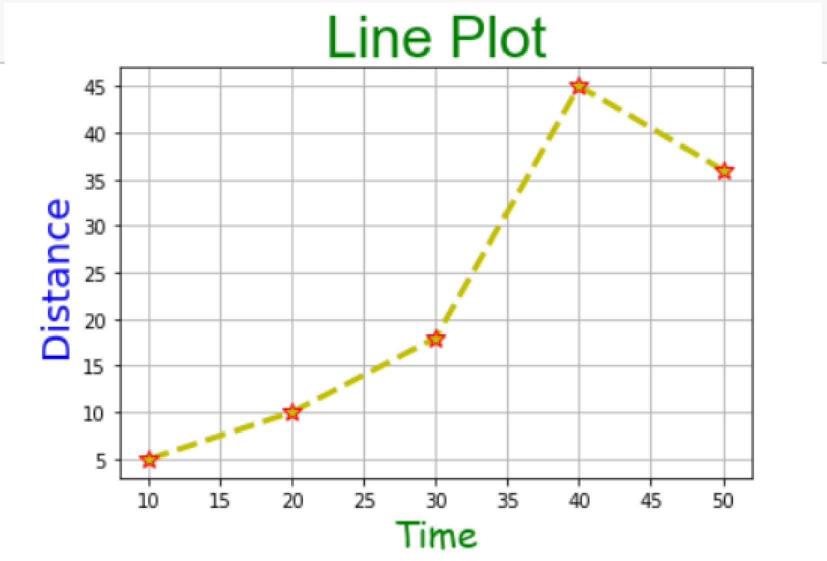
```
#Setting label of X-aixs and y-axis with title
import matplotlib.pyplot as plt
x=[10,50,100]
y=[5,40,89]
plt.plot(x,y,color='m', linewidth=10)
plt.title("Line Plot", fontsize=30, color= 'g', fontname ="Arial")
plt.xlabel("Time", fontsize=20, color='g', fontname="Comic Sans MS")
plt.ylabel("Distance", fontsize=20,color='b')
plt.show()
```



Changing Marker Type, Size and Color

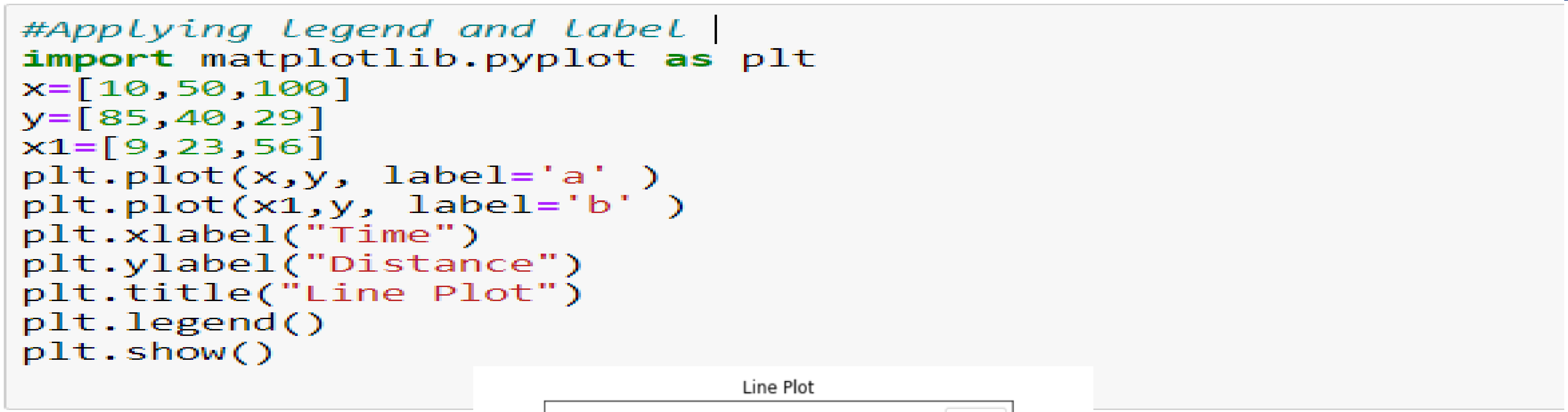


```
# Changing linestyle, linewidth, marker, markersize and color
import matplotlib.pyplot as plt
x=[10,20,30,40,50]
y=[5,10,18,45,36]
plt.plot(x,y,color='y', marker="*", markersize="10", markeredgecolor = 'r',
         linewidth=3, linestyle="--")
plt.title("Line Plot", fontsize=30, color= 'g', fontname ='Arial')
plt.xlabel("Time", fontsize=20, color='g', fontname='Comic Sans MS')
plt.ylabel("Distance", fontsize=20,color='b')
plt.grid()
plt.show()
```

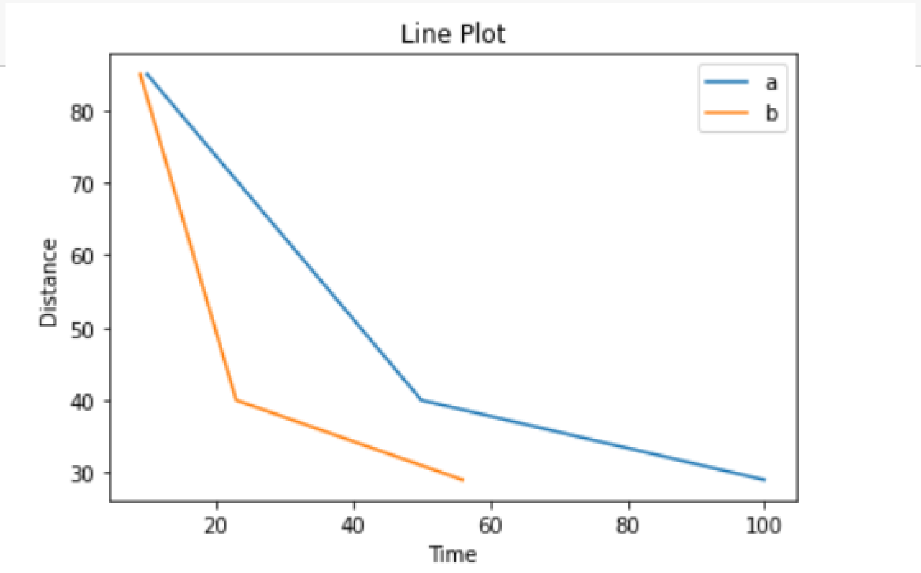


Rajesh Verma

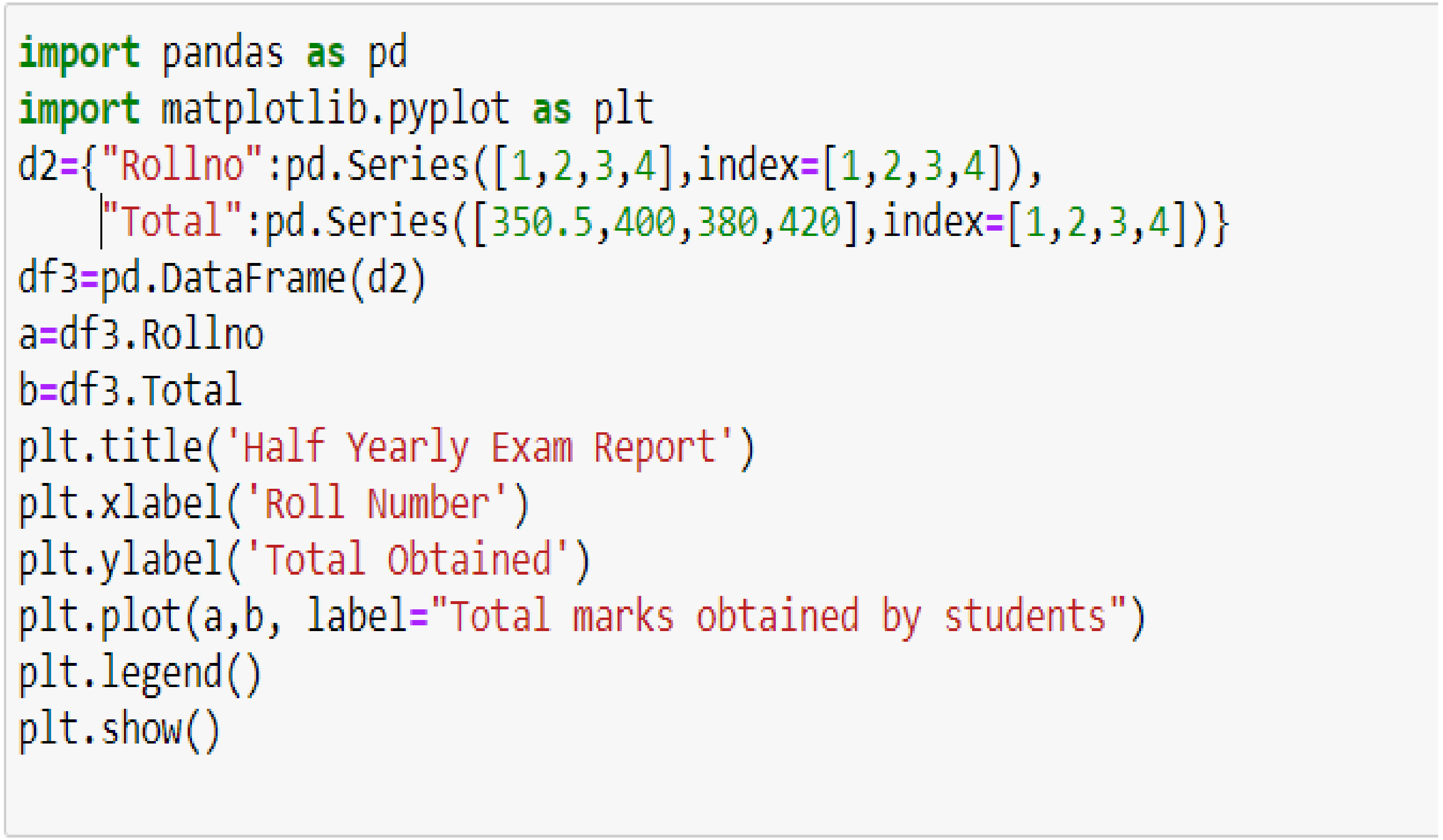
Applying legend and label



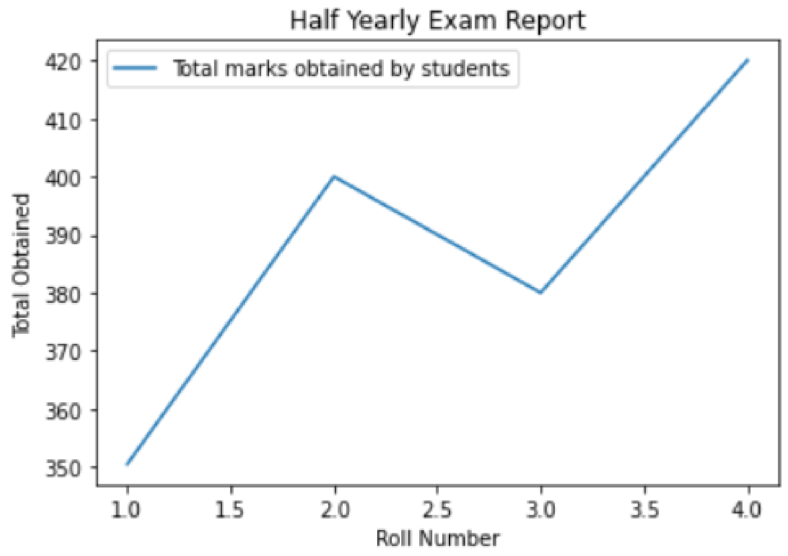
```
#Applying legend and label
import matplotlib.pyplot as plt
x=[10,50,100]
y=[85,40,29]
x1=[9,23,56]
plt.plot(x,y, label='a' )
plt.plot(x1,y, label='b' )
plt.xlabel("Time")
plt.ylabel("Distance")
plt.title("Line Plot")
plt.legend()
plt.show()
```



Plotting Data Stored in a DataFrame Using Line Plot

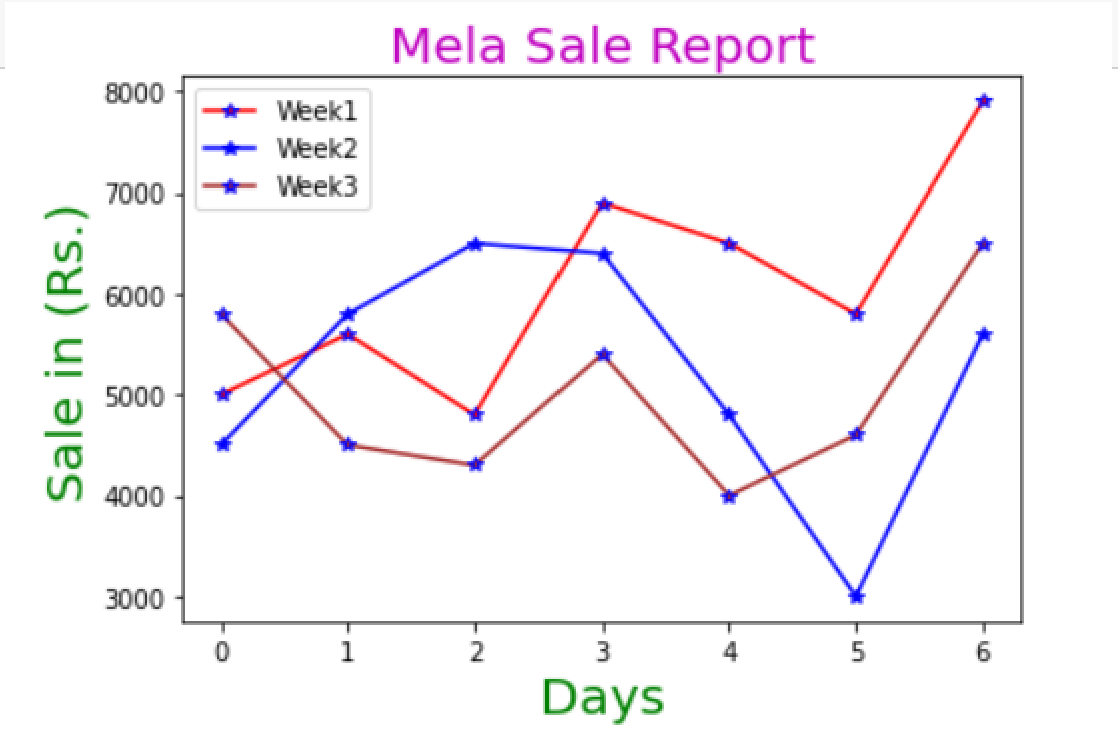


```
import pandas as pd
import matplotlib.pyplot as plt
d2={"Rollno":pd.Series([1,2,3,4],index=[1,2,3,4]),
    "Total":pd.Series([350.5,400,380,420],index=[1,2,3,4])}
df3=pd.DataFrame(d2)
a=df3.Rollno
b=df3.Total
plt.title('Half Yearly Exam Report')
plt.xlabel('Roll Number')
plt.ylabel('Total Obtained')
plt.plot(a,b, label="Total marks obtained by students")
plt.legend()
plt.show()
```



Plotting Data Stored in .CSV File Using Line Plot

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('melasale.csv')
df.plot(kind = 'line',color = ['red','blue','brown'], marker = '*', markeredgecolor = 'b')
plt.title('Mela Sale Report', fontsize = 20, color = 'm')
plt.xlabel('Days', fontsize = 20, color = 'g')
plt.ylabel('Sale in (Rs.)', fontsize = 20, color = 'g')
plt.legend()
plt.show()
```



Rajesh Verma

BAR PLOT

BAR PLOT

•Bar plot presents categorical data in the form of bars with heights or lengths proportional to the values that they represent.
•One axis of the chart shows the specific categories being compared, and the other axis represents a measured value.
•The bars can be plotted horizontally or vertically.
•Bar plots can be either single, stacked, or grouped.

PURPOSE

•Bar plots are used to compare multiple variables in a single timeframe or a single variable in a time series.

•It shows comparisons among discrete categories. One axis of the chart shows the specific categories being compared, and the other axis represents a measured value.

Syntax:

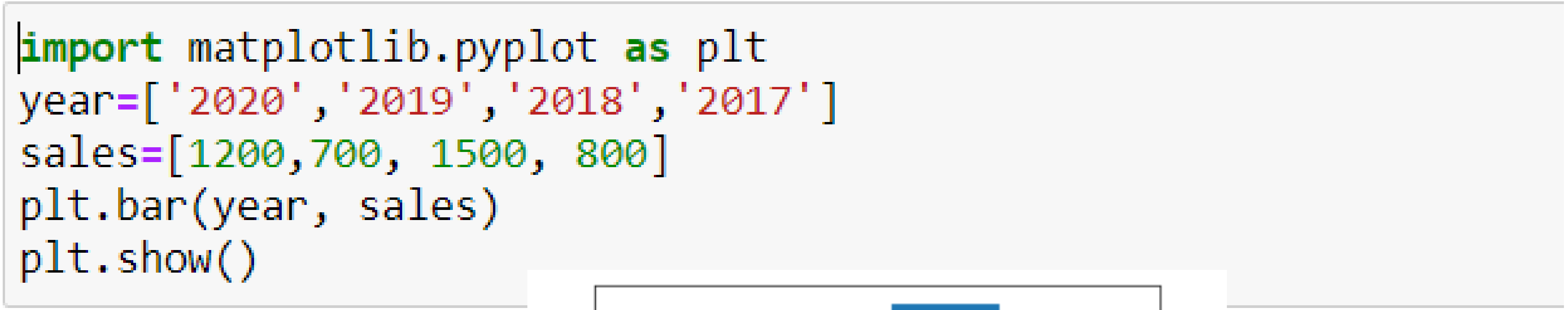
bar(x, y)

here

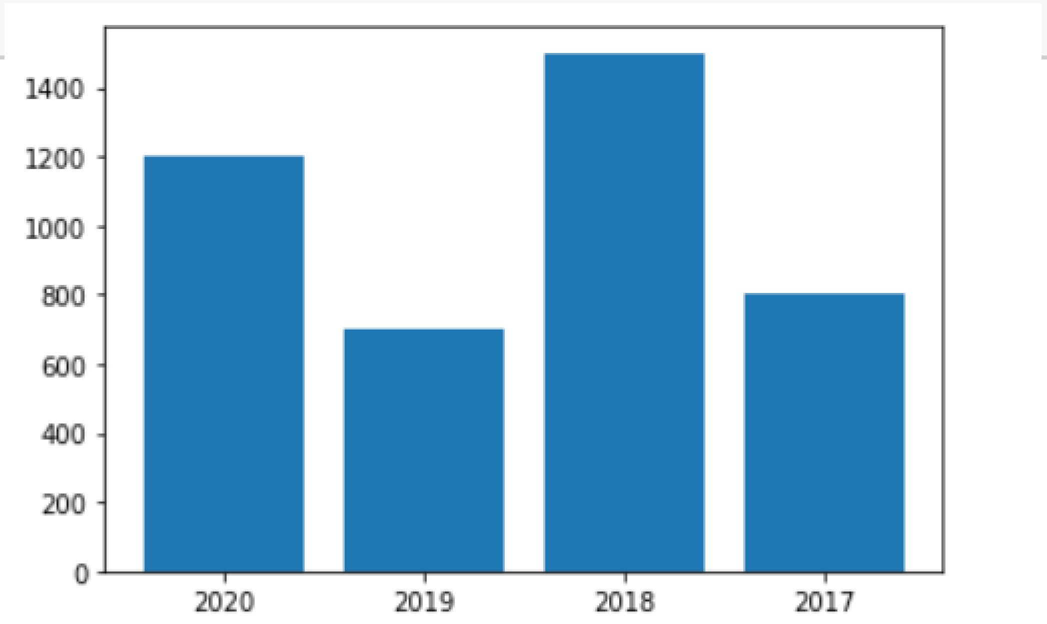
x- list of values for x axis

y- list of values for y axis

Example Bar plot



```
import matplotlib.pyplot as plt
year=['2020','2019','2018','2017']
sales=[1200,700, 1500, 800]
plt.bar(year, sales)
plt.show()
```

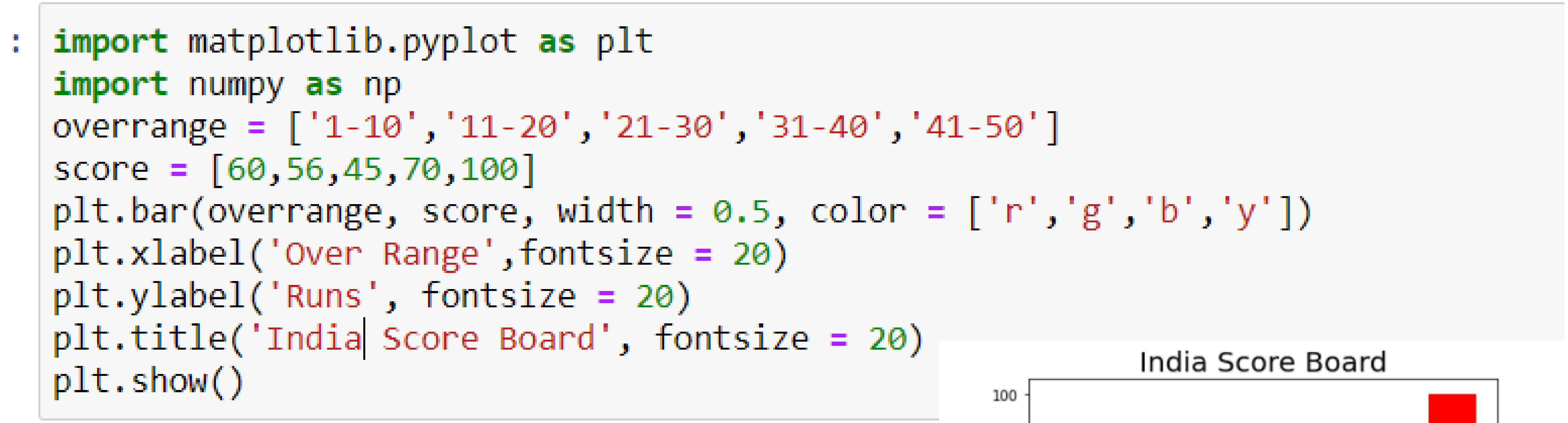


Customizing the Bar Plot

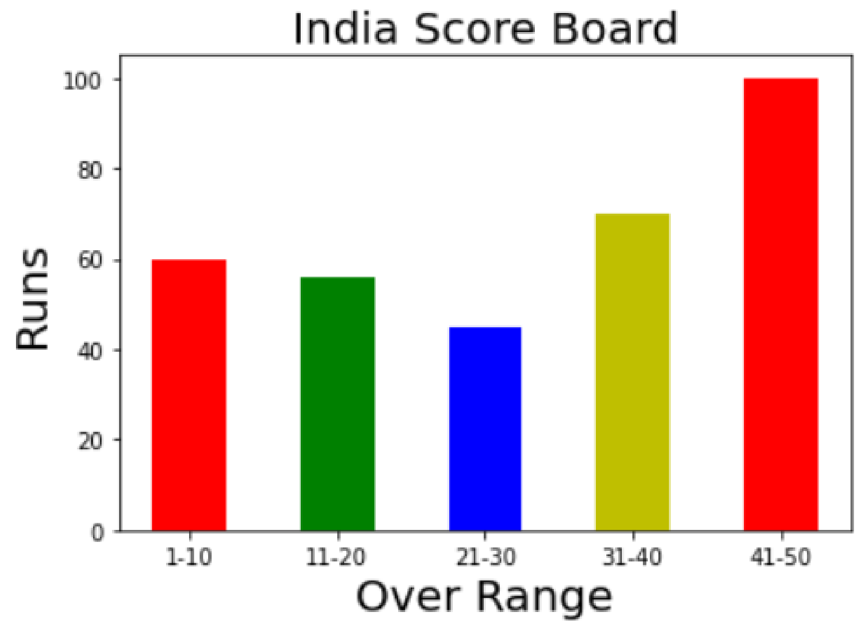
Following are some of the parameters that can be used in the bar function to customize the bar chart.
Syntax:
matplotlib.pyplot.bar(x, height, width=0.8, bottom=None, *, align='center', data=None,)

PARAMETER	**PURPOSE**
height	The height(s) of the bars.
width	to set the width of the bar, default 0.8
label	to set the label for the bar chart
color	to specify the color of the bar
align	it can be ‘center’ or ‘edge’
bottom	to specify a starting value for a bar
edgecolor	The colors of the bar edges
linewidth	Width of the bar edge(s). If 0, don't draw edges.

Changing Width, Color in Bar Chart :



```
import matplotlib.pyplot as plt
import numpy as np
overrange = ['1-10','11-20','21-30','31-40','41-50']
score = [60,56,45,70,100]
plt.bar(overrange, score, width = 0.5, color = ['r','g','b','y'])
plt.xlabel('Over Range',fontsize = 20)
plt.ylabel('Runs', fontsize = 20)
plt.title('India Score Board', fontsize = 20)
plt.show()
```

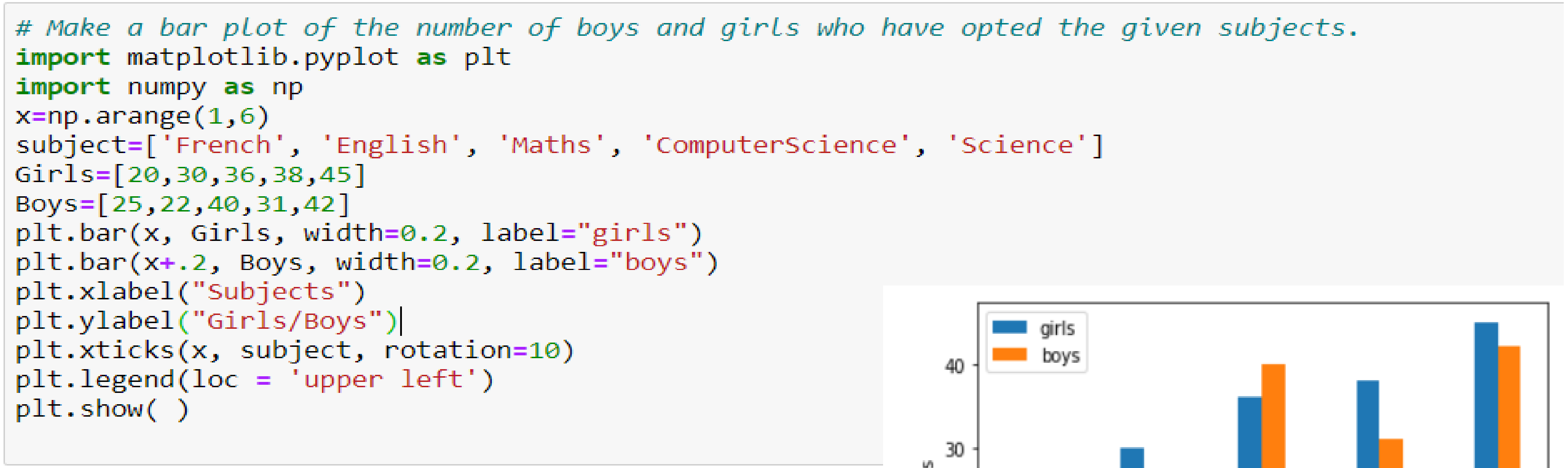


Multiple Bar Graph:

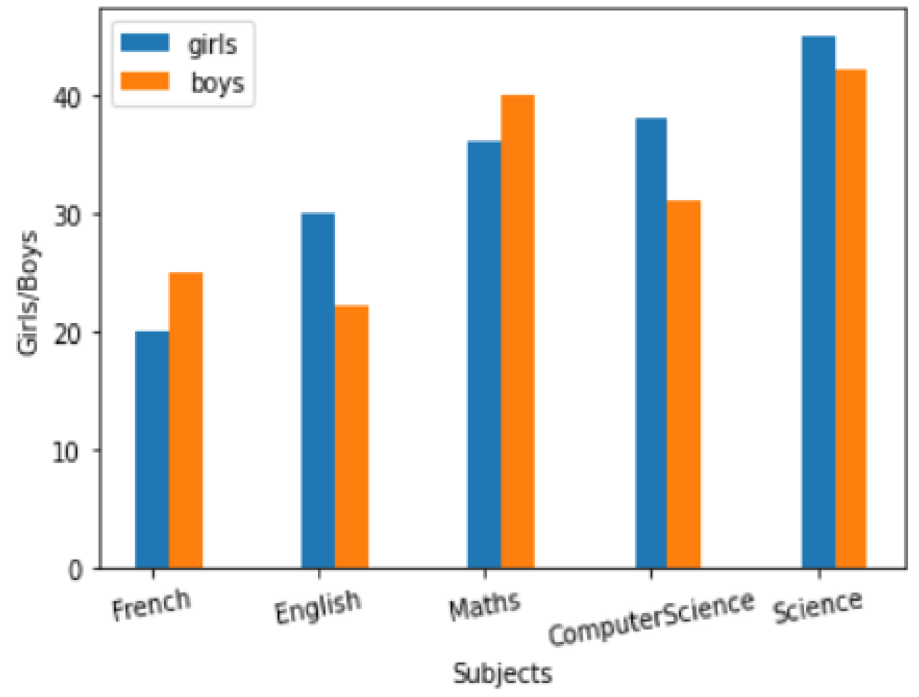
To draw multiple bar chart:

- *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*

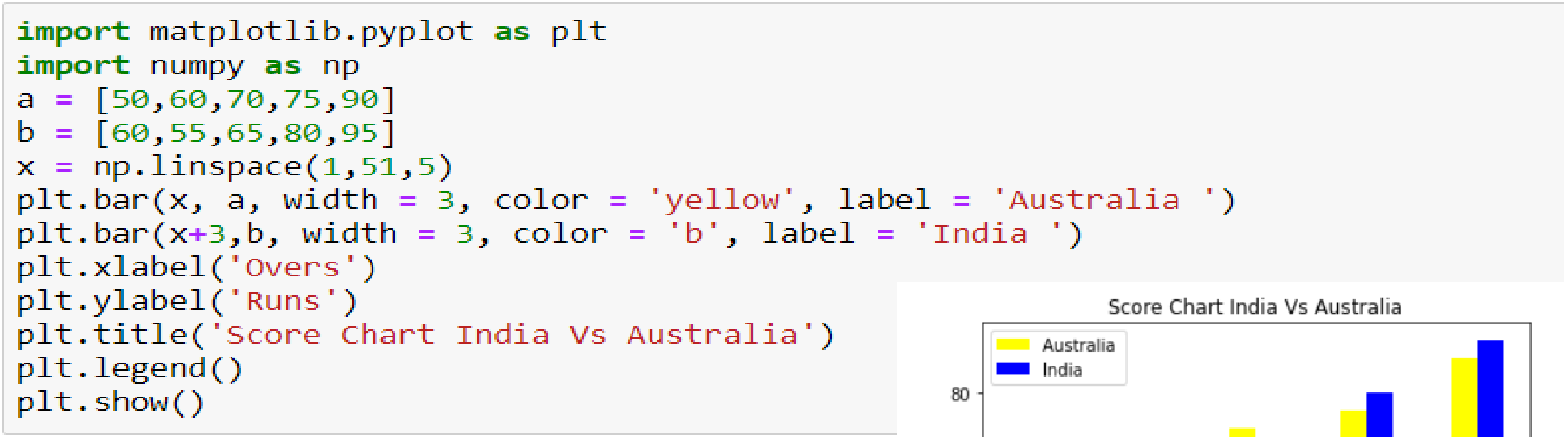
Multiple Bar Graph using arange()



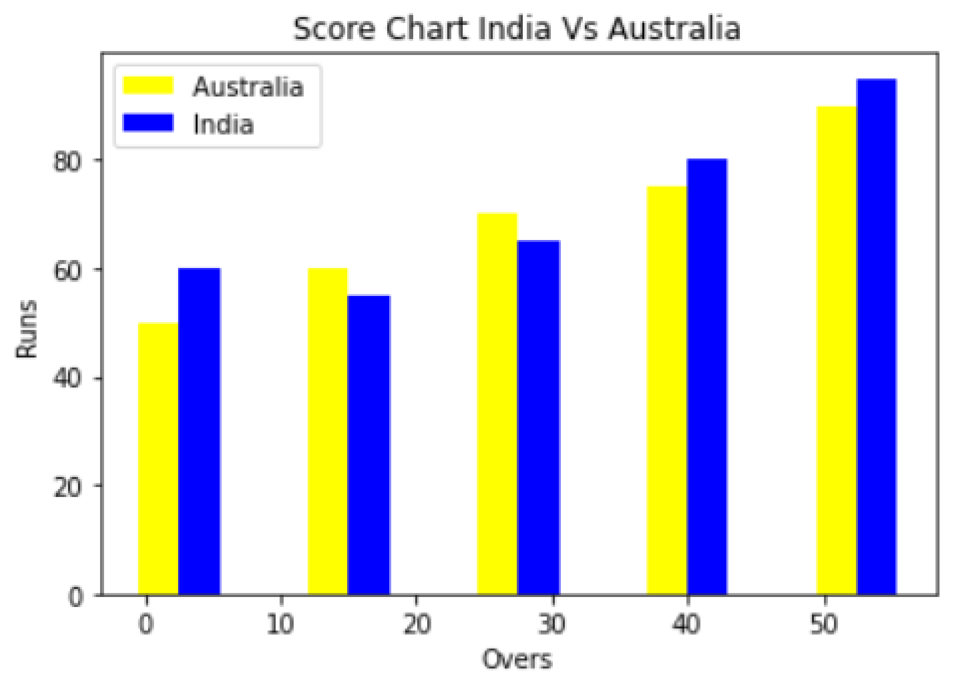
```
# Make a bar plot of the number of boys and girls who have opted the given subjects.
import matplotlib.pyplot as plt
import numpy as np
x=np.arange(1,6)
subject=['French', 'English', 'Maths', 'ComputerScience', 'Science']
Girls=[20,30,36,38,45]
Boys=[25,22,40,31,42]
plt.bar(x, Girls, width=0.2, label="girls")
plt.bar(x+.2, Boys, width=0.2, label="boys")
plt.xlabel("Subjects")
plt.ylabel("Girls/Boys")
plt.xticks(x, subject, rotation=10)
plt.legend(loc = 'upper left')
plt.show( )
```



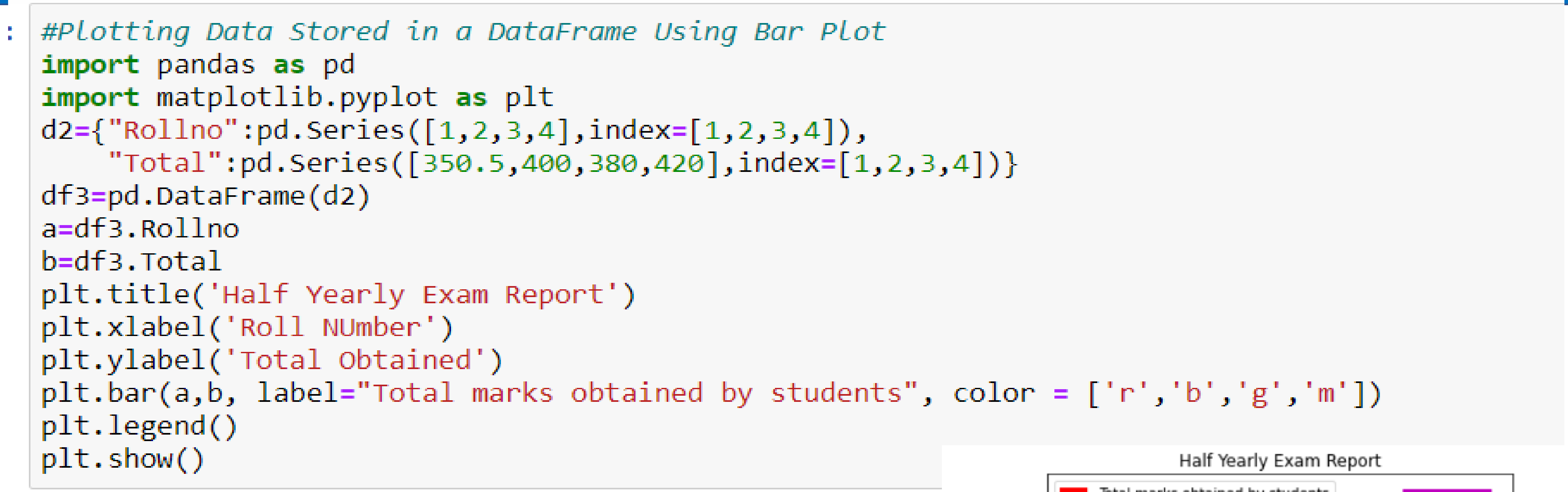
Multiple Bar Graph using linspace()



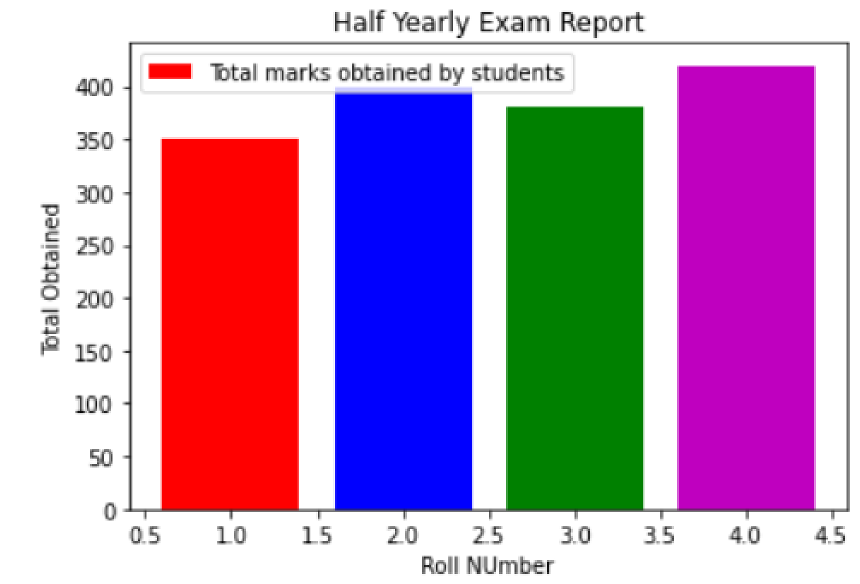
```
import matplotlib.pyplot as plt
import numpy as np
a = [50,60,70,75,90]
b = [60,55,65,80,95]
x = np.linspace(1,51,5)
plt.bar(x, a, width = 3, color = 'yellow', label = 'Australia ')
plt.bar(x+3,b, width = 3, color = 'b', label = 'India ')
plt.xlabel('Overs')
plt.ylabel('Runs')
plt.title('Score Chart India Vs Australia')
plt.legend()
plt.show()
```



Plotting Data Stored in a DataFrame Using Bar Plot

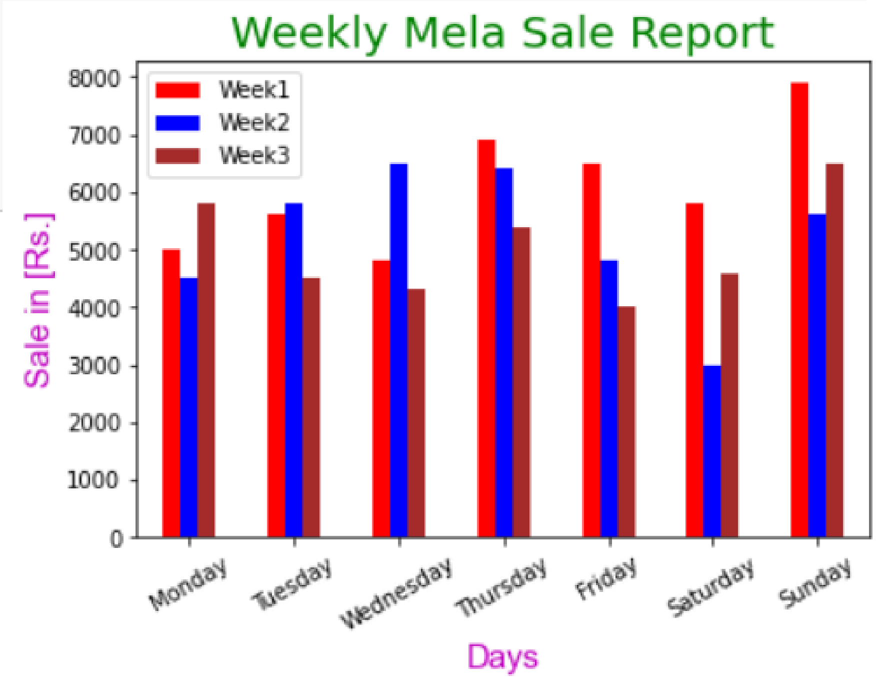


```
#Plotting Data Stored in a DataFrame Using Bar Plot
import pandas as pd
import matplotlib.pyplot as plt
d2={"Rollno":pd.Series([1,2,3,4],index=[1,2,3,4]),
    "Total":pd.Series([350.5,400,380,420],index=[1,2,3,4])}
df3=pd.DataFrame(d2)
a=df3.Rollno
b=df3.Total
plt.title('Half Yearly Exam Report')
plt.xlabel('Roll NUmber')
plt.ylabel('Total Obtained')
plt.bar(a,b, label="Total marks obtained by students", color = ['r','b','g','m'])
plt.legend()
plt.show()
```



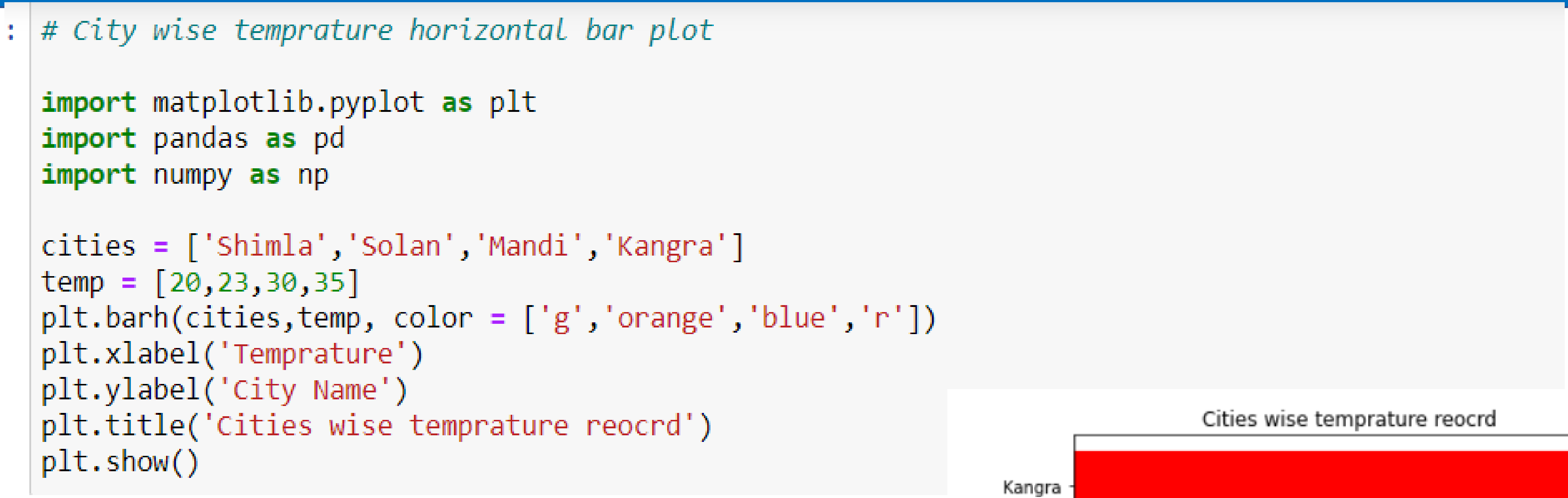
Plotting Data Stored in .CSV File Using Bar Plot

```python
#Plotting Data Stored in .CSV File Using Bar Plot
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
df=pd.read_csv("melasale.csv")
df.plot(kind='bar', width = 0.5,color=['red','blue','brown'])
plt.title('Weekly Mela Sale Report', fontsize = 20, color = 'green')
plt.xlabel('Days', fontsize = 15, fontname = 'Arial', color = 'm')
plt.ylabel('Sale in [Rs.]', fontsize = 15, fontname = 'Arial', color = 'm')
x=np.arange(0,7)
plt.xticks(x,df['Day'], rotation=30)
plt.legend()
plt.show()
```

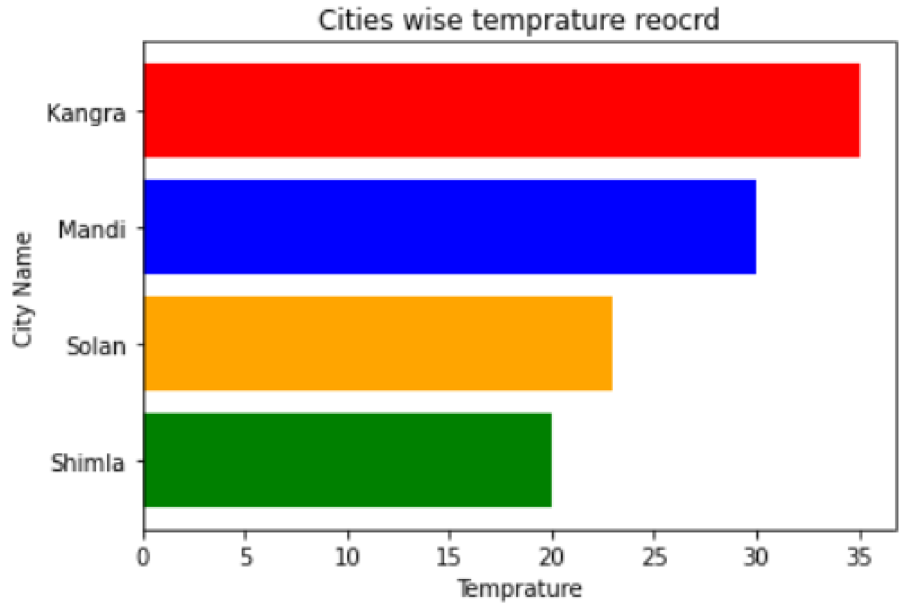


Rajesh Verma

Horizontal Bar Graph

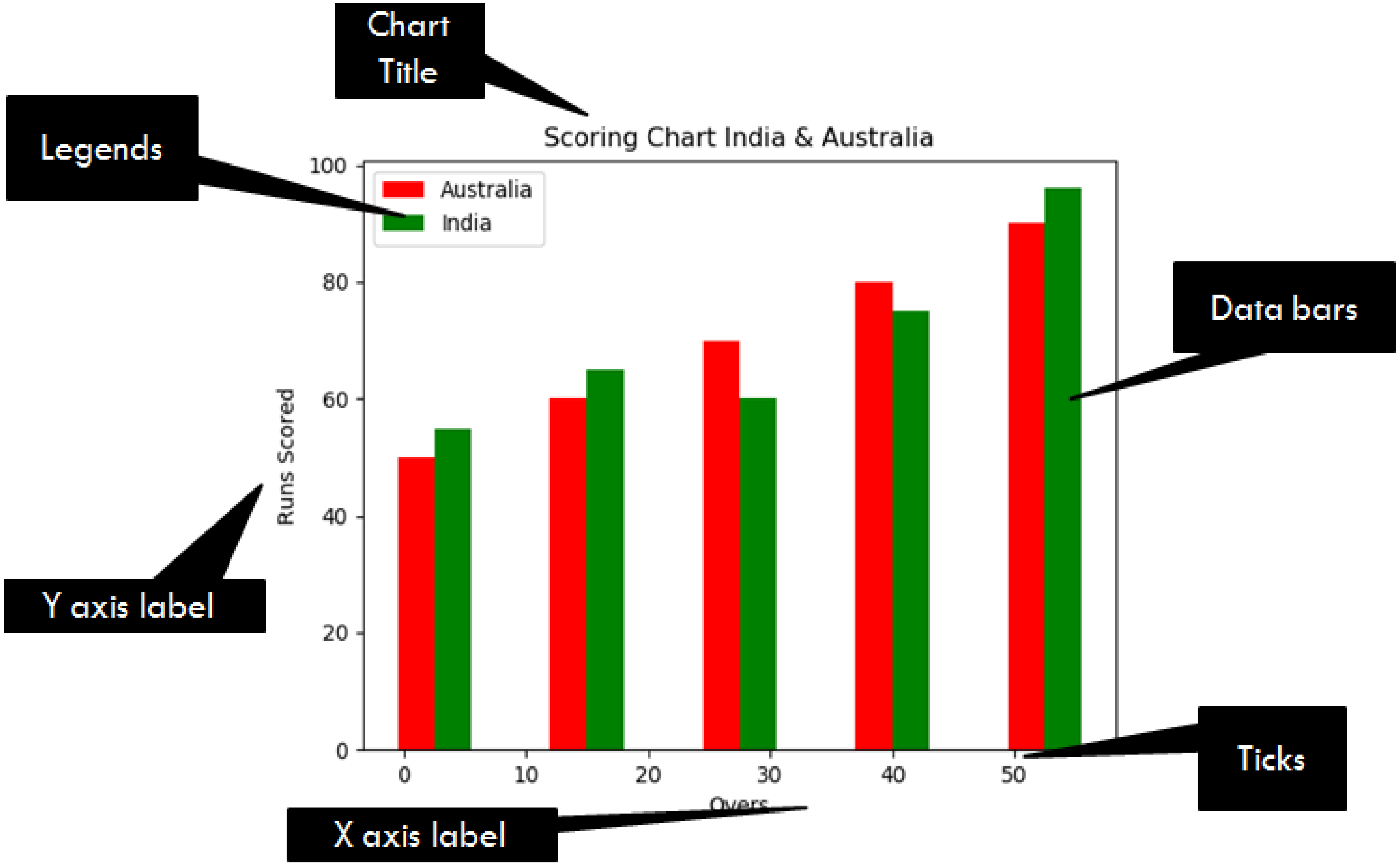


```
# City wise temprature horizontal bar plot

import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

cities = ['Shimla','Solan','Mandi','Kangra']
temp = [20,23,30,35]
plt.barh(cities,temp, color = ['g','orange','blue','r'])
plt.xlabel('Temprature')
plt.ylabel('City Name')
plt.title('Cities wise temprature reocrd')
plt.show()
```

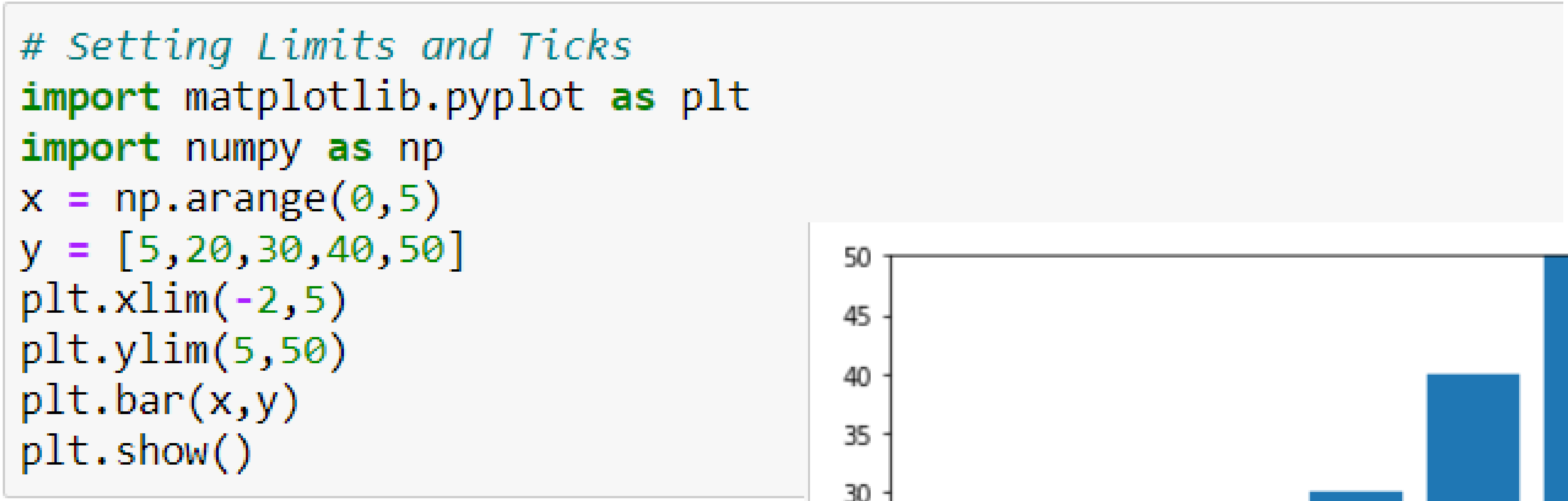


barh() is used to draw horizontal bar graph.

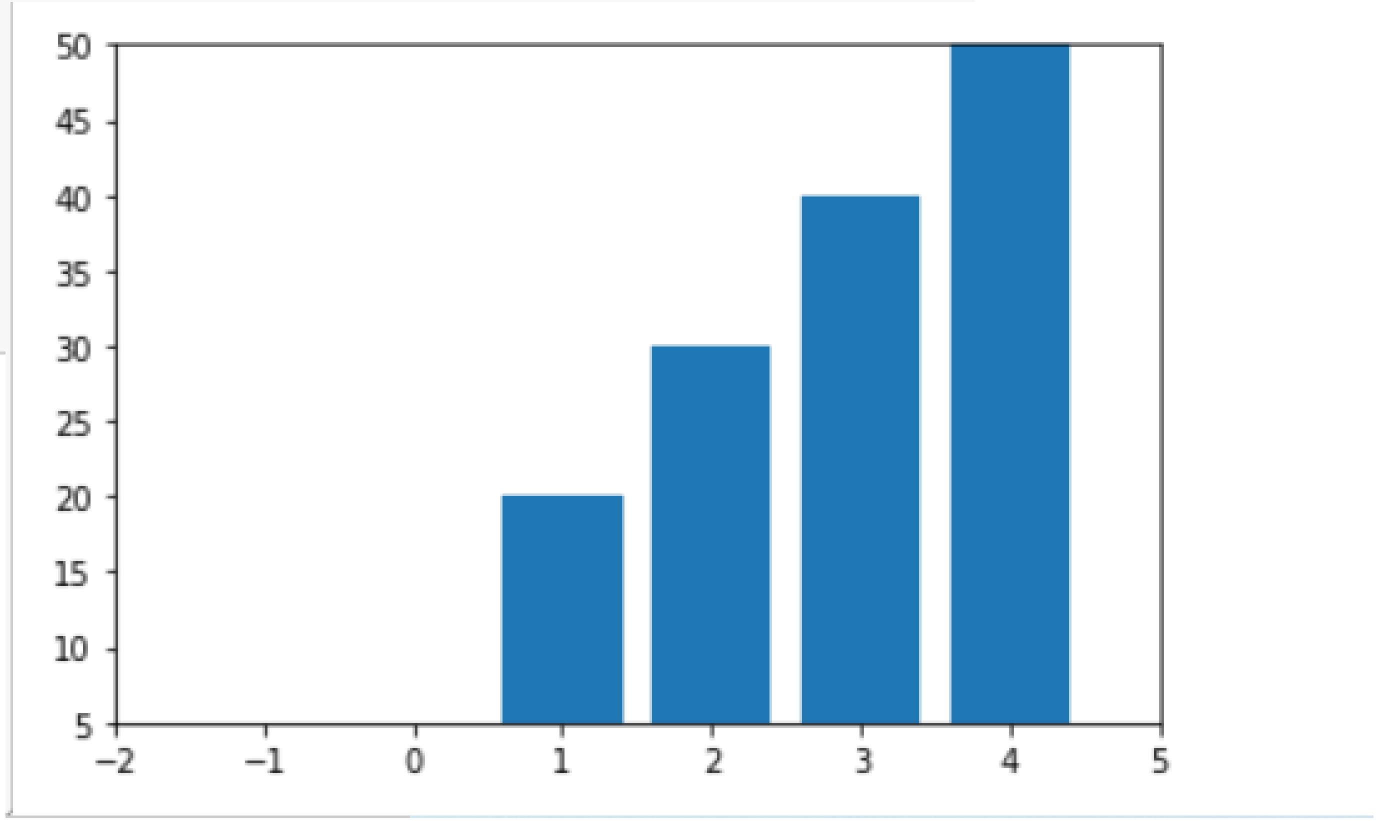
Anatomy of chart



Setting Limits and Ticks



```
# Setting Limits and Ticks
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(0,5)
y = [5,20,30,40,50]
plt.xlim(-2,5)
plt.ylim(5,50)
plt.bar(x,y)
plt.show()
```



Histogram Plot

Histogram Plot

•Histogram is a graphical representation of the distribution of numerical data.

•It takes one numerical variable as input. The variable is cut into several bins. The bins are usually specified as consecutive, non-overlapping intervals of a variable.

•The number of observation per bin is represented by the height of the bar.

Purpose

•Histograms are a great way to show results of continuous data, such as: weight, height, how much time etc.

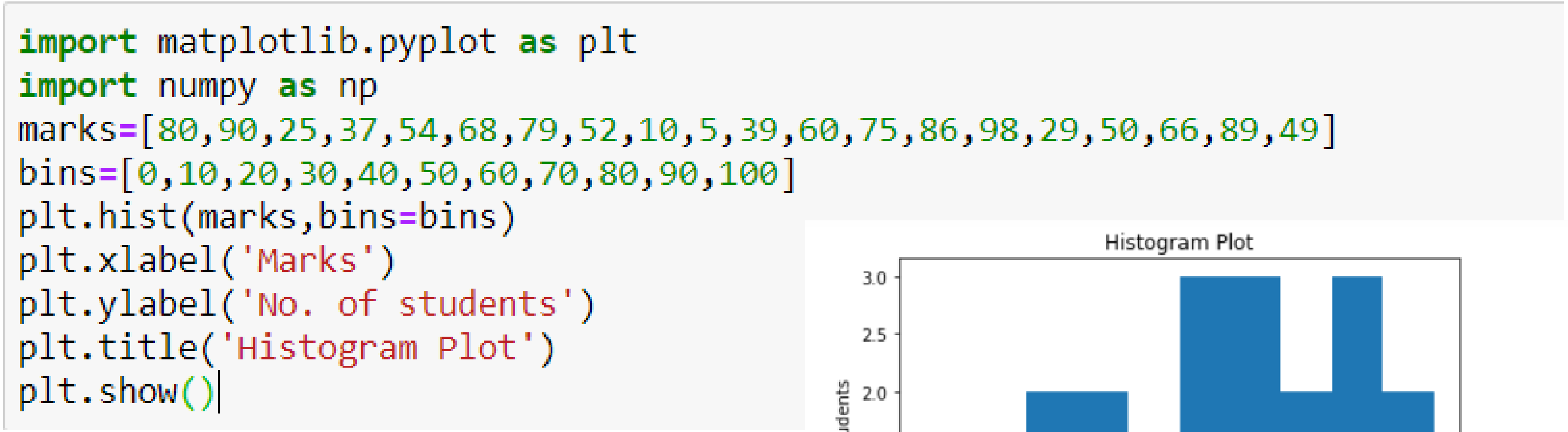
•Histograms are used for data that involve ordinal variables, or things that are not easily quantified.

Difference between Histogram and Bar graph

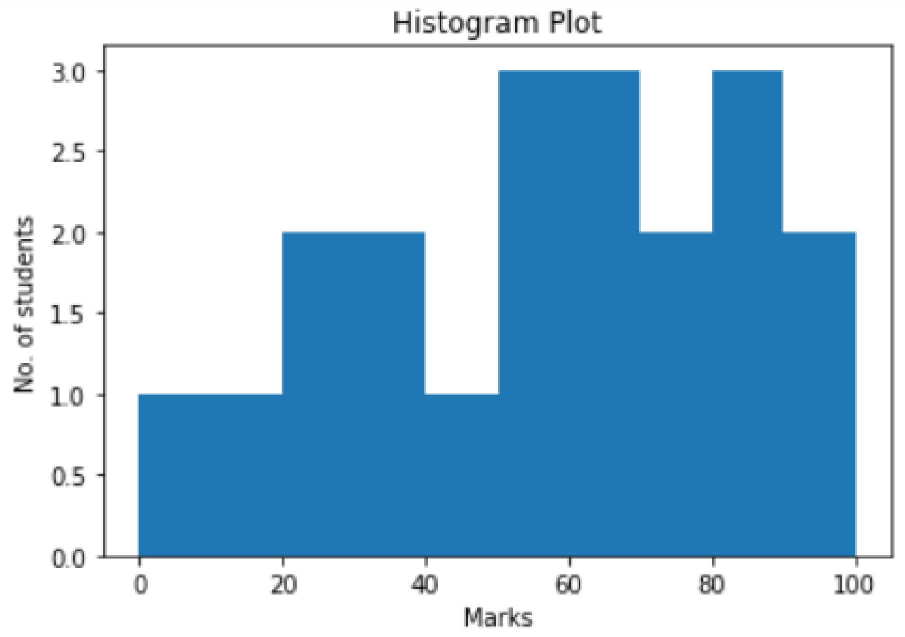
HISTOGRAM	BAR GRAPH
Graphical representation, that uses bars to show the frequency of numerical data.	Pictorial representation of data that uses bars to compare different categories of data.
Indicates distribution of continuous variables	Indicates comparison of discrete variables
Presents quantitative data	Presents categorical data
Bars touch each other	Bars do not touch each other
Elements are grouped together, so that they are considered as ranges.	Elements are taken as individual entities.
Bars need not to be same width	All bars are of same width

Rajesh Verma

Histogram Plot



```
import matplotlib.pyplot as plt
import numpy as np
marks=[80,90,25,37,54,68,79,52,10,5,39,60,75,86,98,29,50,66,89,49]
bins=[0,10,20,30,40,50,60,70,80,90,100]
plt.hist(marks,bins=bins)
plt.xlabel('Marks')
plt.ylabel('No. of students')
plt.title('Histogram Plot')
plt.show()
```



hist(): It is used to create histogram graph.
Syntax: hist(data)
Here data- list of values

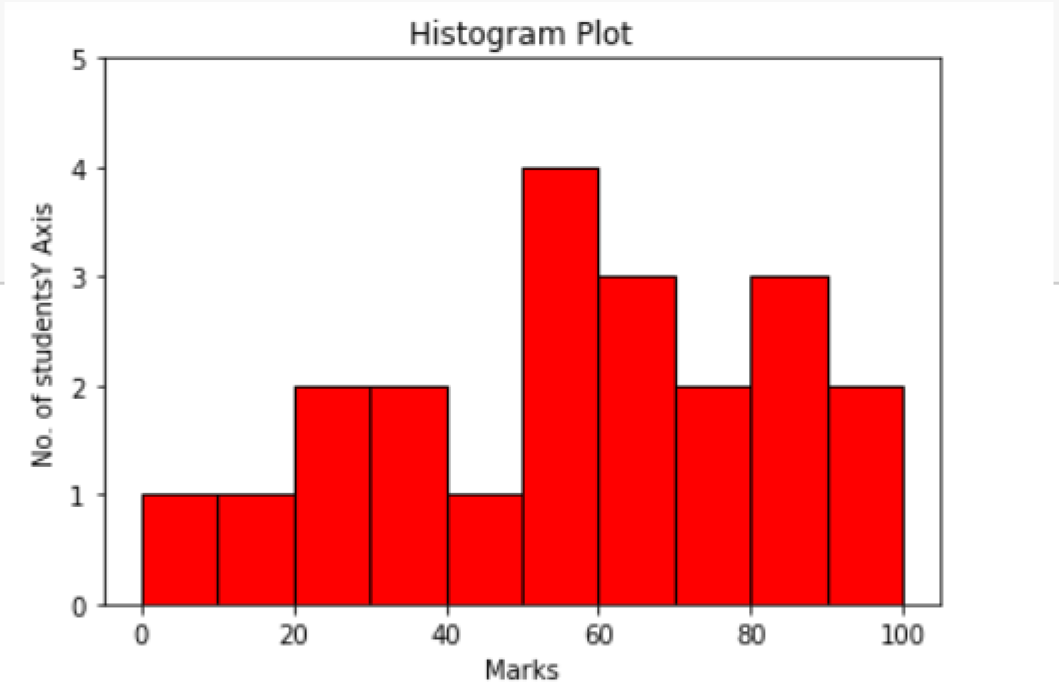
Customizing the Histogram Plot

Following are some of the parameters that can be used in the hist function to customize the histogram chart. Syntax: hist(data, bins, color, edgecolor, hatch, linestyle, fill)

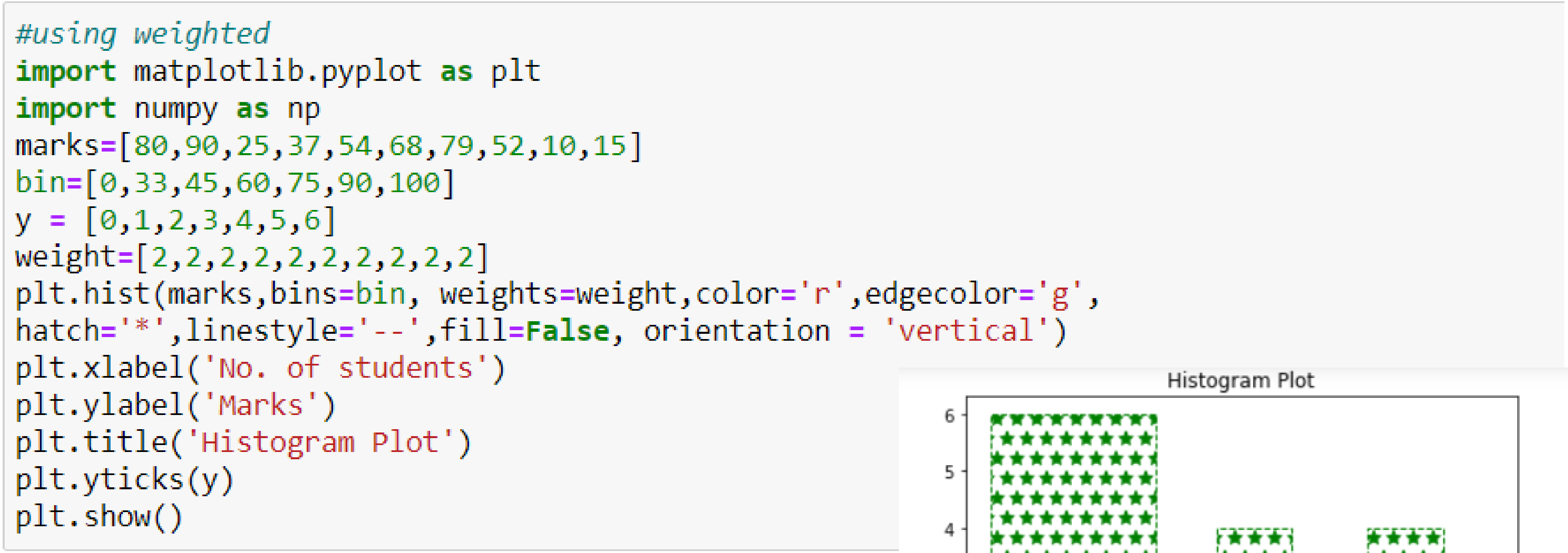
PARAMETER	PURPOSE
color	to specify the color of the bars
edgecolor	To set the color of the edge
hatch	to fill a pattern in a specified area
linestyle	To set line style
fill	set to True to fill the area below the histogram otherwise set to False
bins	To set the number of bins that your data will be divided into.
weights	To set the weight associated with each data value.
histtype	The type of histogram to draw['bar','barstacked','step','stepfillied']

Changing edgecolor of bar

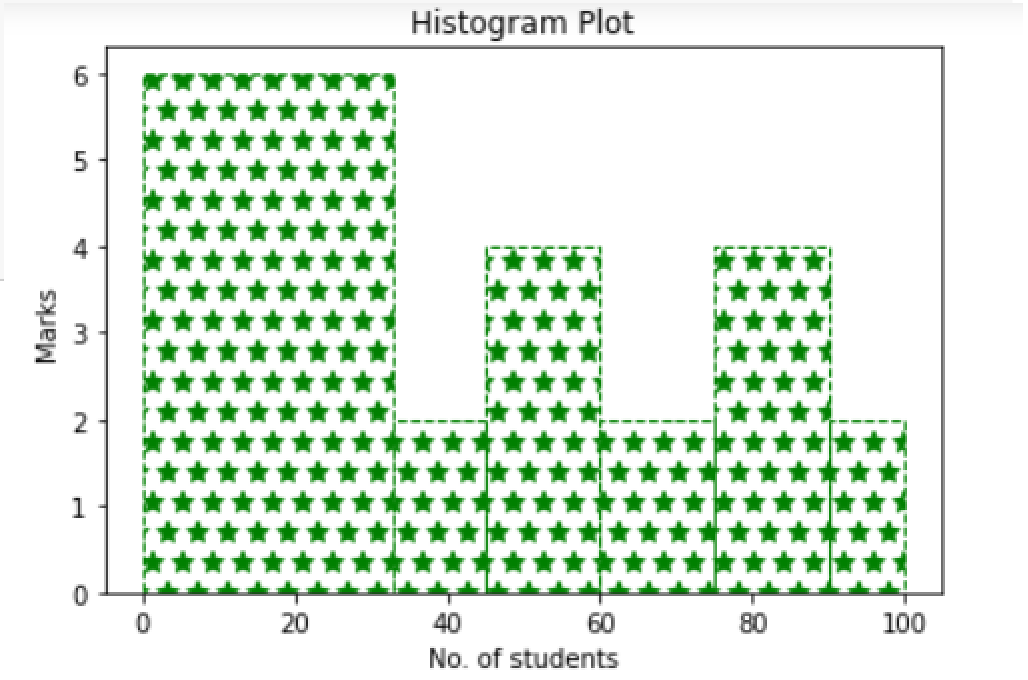
```
import matplotlib.pyplot as plt
import numpy as np
marks=[80,90,25,37,54,68,79,52,10,5,39,60,75,86,98,29,50,66,89,49,51]
bin=[0,10,20,30,40,50,60,70,80,90,100]
plt.hist(marks,bins=bin,color='r', edgecolor='k')
plt.xlabel('Marks')
plt.ylabel('No. of studentsY Axis')
plt.title('Histogram Plot')
plt.yticks([0,1,2,3,4,5])
plt.show()
```



Weigheted Histogram



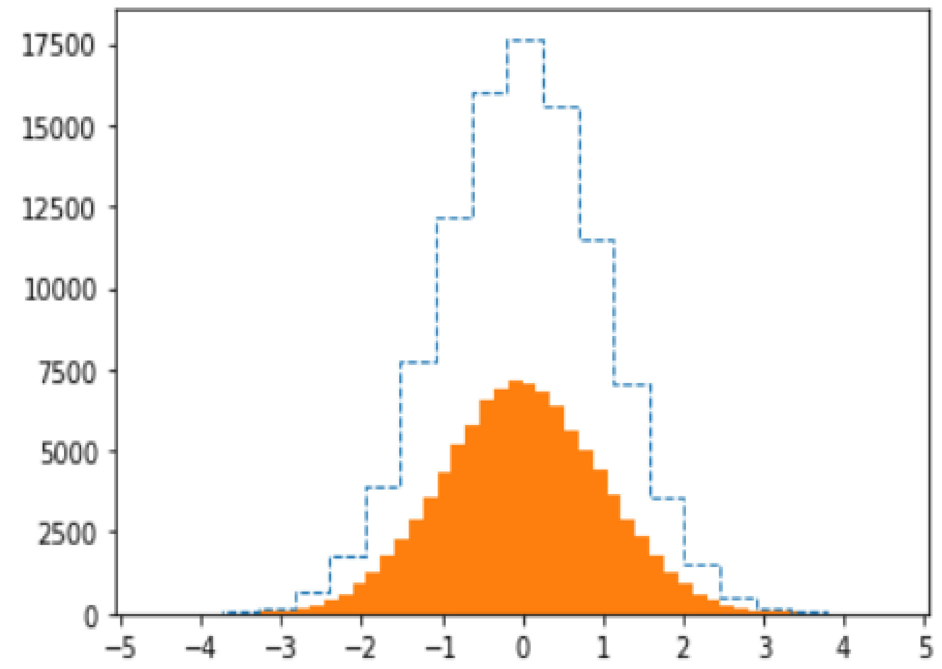
```
#using weighted
import matplotlib.pyplot as plt
import numpy as np
marks=[80,90,25,37,54,68,79,52,10,15]
bin=[0,33,45,60,75,90,100]
y = [0,1,2,3,4,5,6]
weight=[2,2,2,2,2,2,2,2,2,2]
plt.hist(marks,bins=bin, weights=weight,color='r',edgecolor='g',
hatch='*',linestyle='--',fill=False, orientation = 'vertical')
plt.xlabel('No. of students')
plt.ylabel('Marks')
plt.title('Histogram Plot')
plt.yticks(y)
plt.show()
```



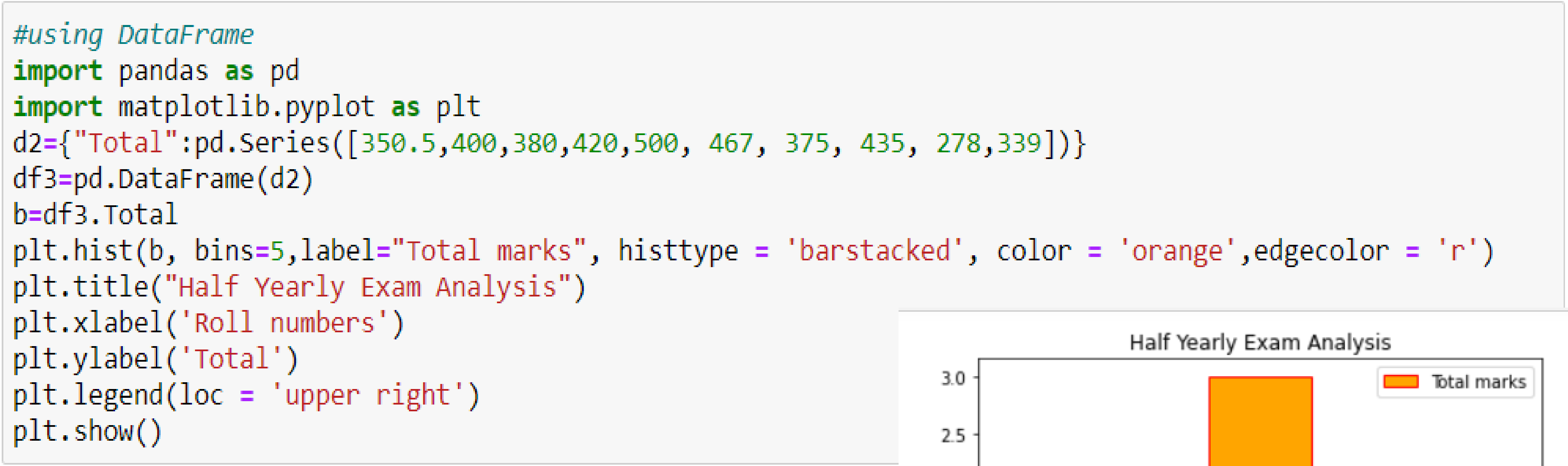
Weigheted Histogram: A weighted histogram is a histogram which uses weights to represent the weighted distribution of the values. Here besides values we need the weights corresponding to each value.

Plotting Histogram by histtype parameter using multiple data

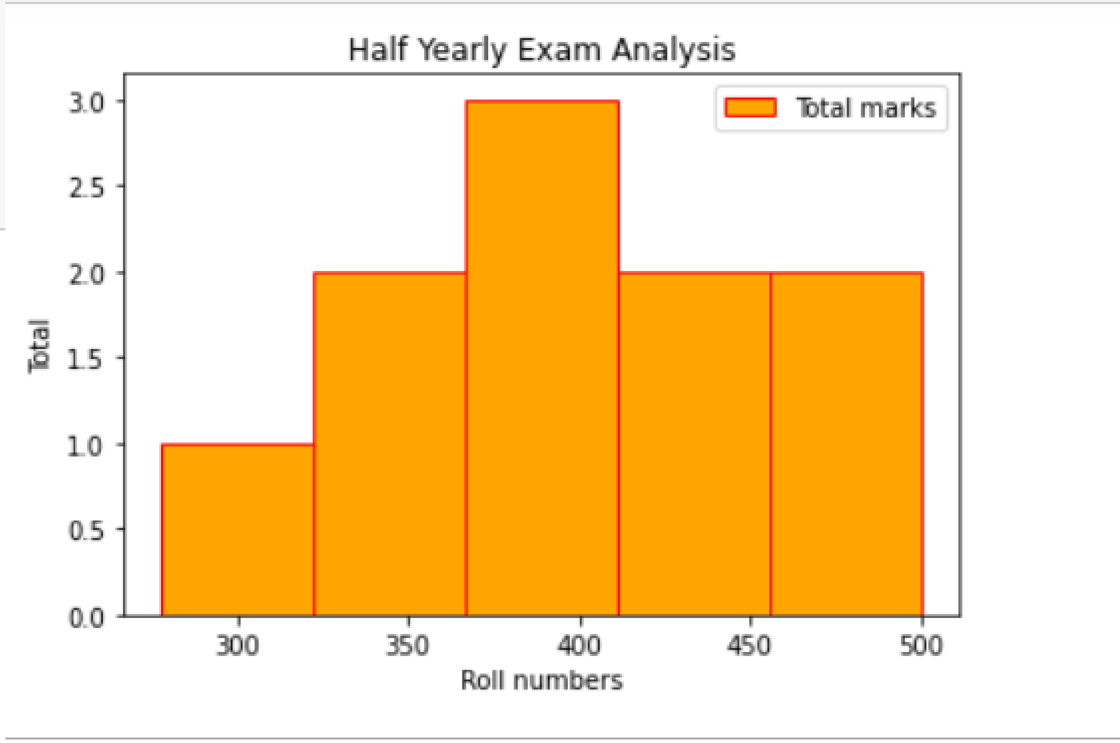
```
import matplotlib.pyplot as plt
import numpy as np
x = np.random.randn(100000)
y = np.random.randn(100000)
plt.hist(x, bins = 20, histtype = 'step', linestyle = '--')
plt.hist(y, bins = 50, histtype = 'barstacked')
plt.xticks([-5,-4,-3,-2,-1,0,1,2,3,4,5])
plt.show()
```



Plotting Data Stored in a DataFrame Using Histogram Plot



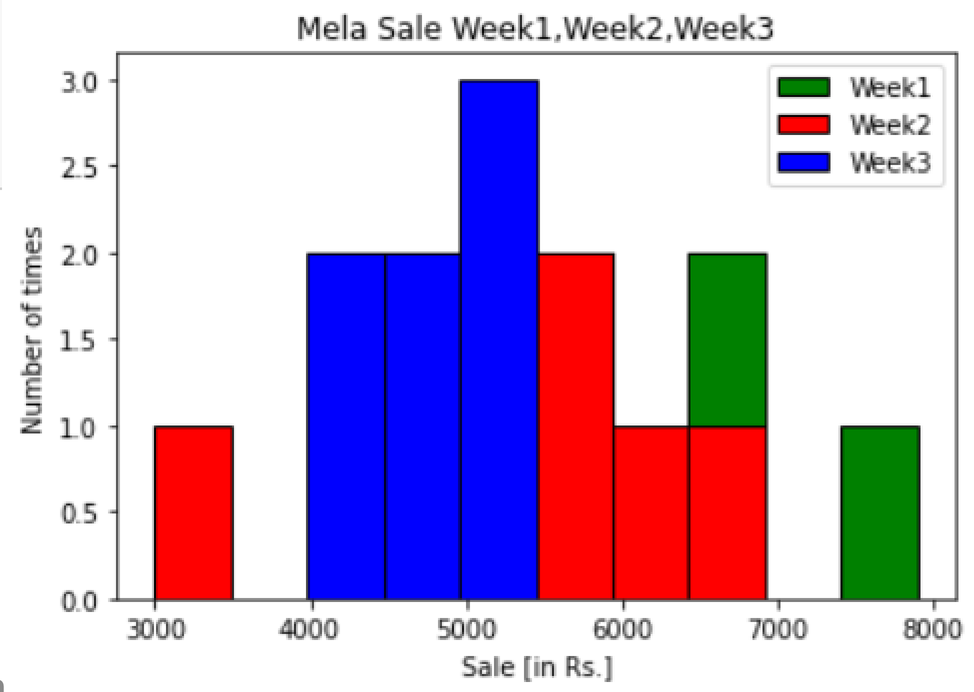
```
#using DataFrame
import pandas as pd
import matplotlib.pyplot as plt
d2={"Total":pd.Series([350.5,400,380,420,500, 467, 375, 435, 278,339])}
df3=pd.DataFrame(d2)
b=df3.Total
plt.hist(b, bins=5,label="Total marks", histtype = 'barstacked', color = 'orange',edgecolor = 'r')
plt.title("Half Yearly Exam Analysis")
plt.xlabel('Roll numbers')
plt.ylabel('Total')
plt.legend(loc = 'upper right')
plt.show()
```



Rajesh Verma

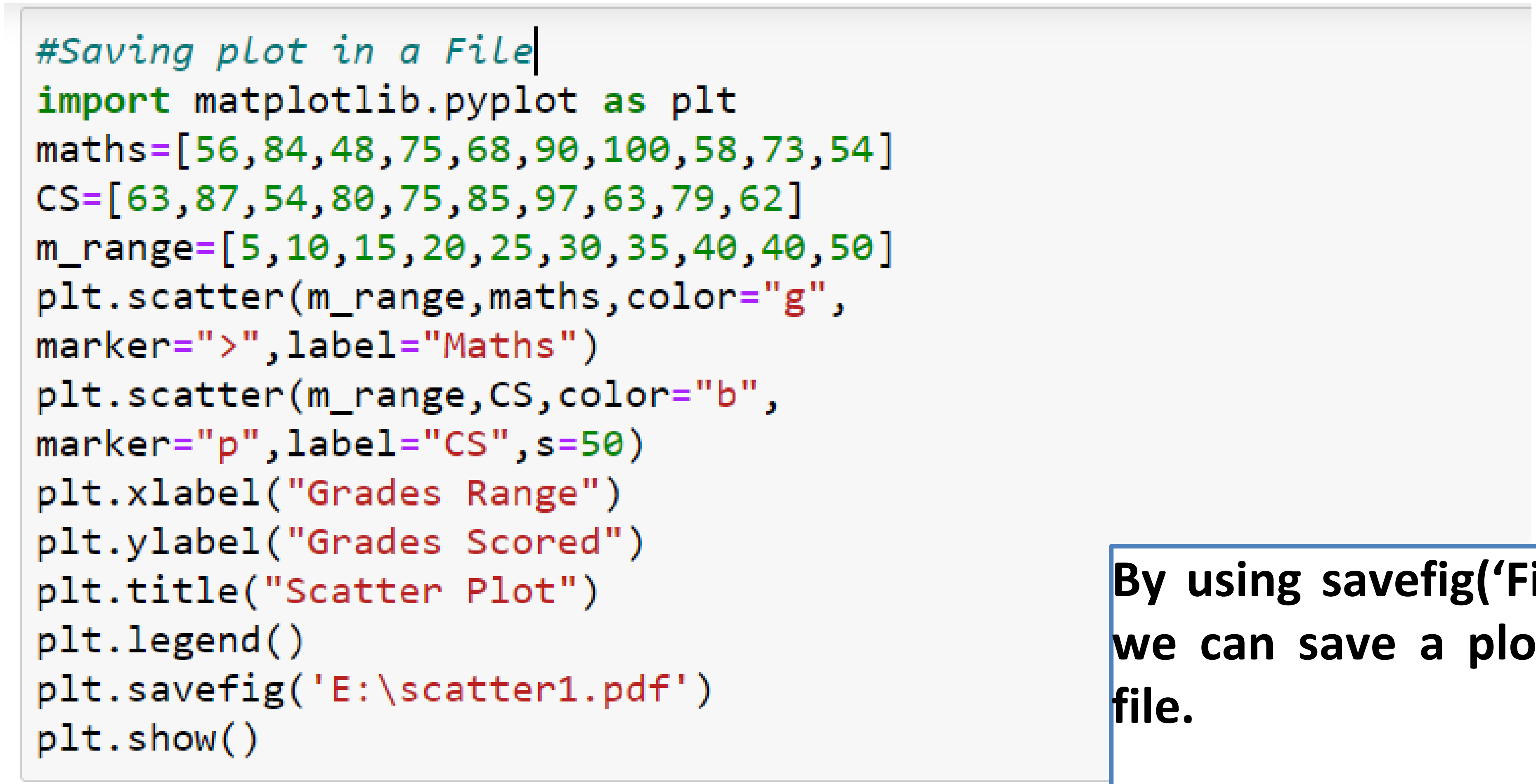
Plotting Data Stored in .CSV File Using Histogram Plot

```python
import pandas as pd
import matplotlib.pyplot as plt
#read the CSV file with specified columns
#using usecols parameter in the variable 'data'
data=pd.read_csv("melasale.csv",
usecols=['Week1','Week2', 'Week3'])
df=pd.DataFrame(data)
#plot histogram for both 'Sale Week1' and 'Sale Week2'
df.plot(kind='hist', title='Mela Sale Week1 and Week2',color=['green','red','b'],edgecolor = 'k')
plt.xlabel('Sale [in Rs.]')
plt.ylabel('Number of times')
plt.legend()
plt.show()
```



Rajesh Verma

Saving Plots or Charts or graph to file



```
#Saving plot in a File
import matplotlib.pyplot as plt
maths=[56,84,48,75,68,90,100,58,73,54]
CS=[63,87,54,80,75,85,97,63,79,62]
m_range=[5,10,15,20,25,30,35,40,40,50]
plt.scatter(m_range,maths,color="g",
marker=">",label="Maths")
plt.scatter(m_range,CS,color="b",
marker="p",label="CS",s=50)
plt.xlabel("Grades Range")
plt.ylabel("Grades Scored")
plt.title("Scatter Plot")
plt.legend()
plt.savefig('E:\scatter1.pdf')
plt.show()
```

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