Python Matplotlib

Python Matplotlib is a library which basically serves the purpose of Data Visualization. The building blocks of Matplotlib library is 2-D NumPy Arrays.

Thus, comparatively huge amount of information/data can be handled and represented through graphs, charts, etc with Python Matplotlib.

we need to import this library whenever we wish to use its built-in functions.

```
1. from matplotlib import pyplot
                or
2. import matplotlib.pyplot
```

matplotlib.pyplot is basically an interface which is used to add style functions to the graphs, charts, etc created using Matplotlib package.

Plotting with Python Matplotlib

Python Matplotlib offers various types of charts to represent and visualize the data.

The following types of graphs/charts can be used to visualize the data using Python Matplotlib:

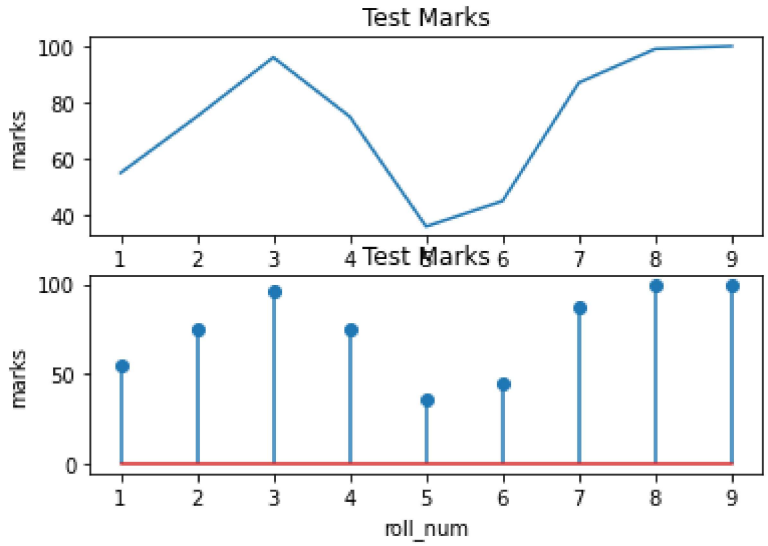
Line Plot Scatter Plot Histogram Bar Chart Pie Chart

```
import matplotlib.pyplot as plt
```

1. Line Plot

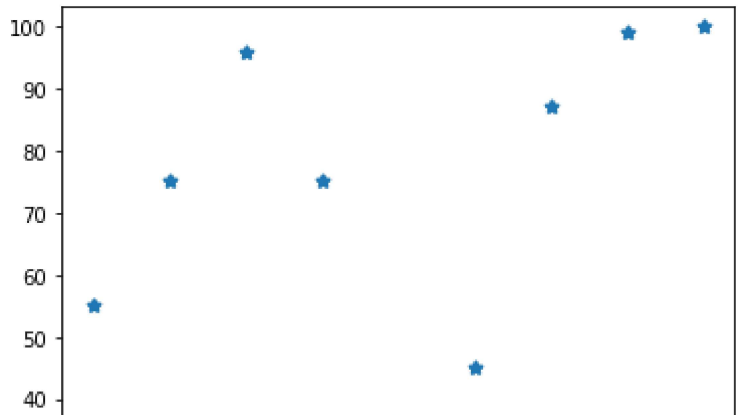
```
# x-axis values
roll_num = [1, 2, 3, 4, 5, 6, 7, 8, 9]

# y-axis values
marks = [55,75,96,75,36,45,87,99,100]

# plt.figure(figsize=(6,2))

plt.subplot(211)
plt.plot(roll_num, marks)
```

```
#matplotlib.pyplot.plot(roll_num, marks)


plt.xlabel('roll_num')      # Set the x axis label of the current axis.
plt.ylabel('marks')         # Set the y axis label of the current axis.
plt.title('Test Marks')     # Set a title


plt.subplot(212)
plt.stem(roll_num,marks)

plt.xlabel('roll_num')      # Set the x axis label of the current axis.
plt.ylabel('marks')         # Set the y axis label of the current axis.
plt.title('Test Marks')     # Set a title


plt.show()
```



2. Scatter Plot

```
# x-axis values
roll_num = [1, 2, 3, 4, 5, 6, 7, 8, 9]

# y-axis values
marks = [55,75,96,75,36,45,87,99,100]

#plt.scatter(roll_num, marks)

plt.scatter(roll_num, marks, marker='*', linewidths=2)
plt.show()
```

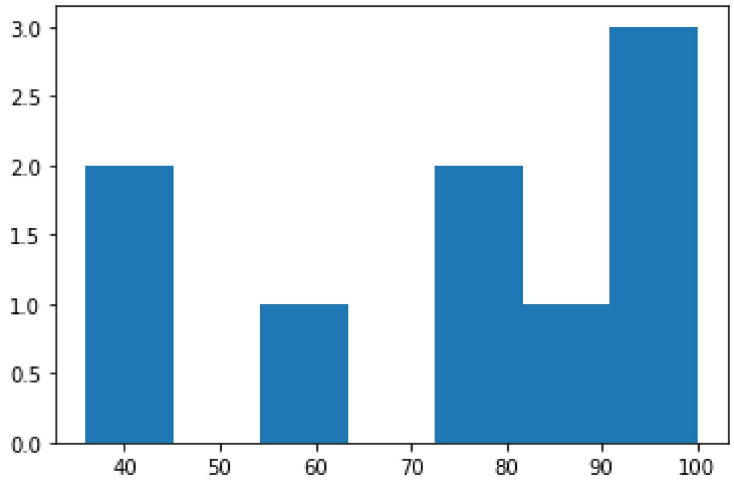


3. Histogram

A graphical display where the data is grouped into ranges (such as "100 to 149", "150 to 199", etc), and then plotted as bars.

The height of each bar shows how many elements are in each range.

```
marks = [55,75,96,75,36,45,87,99,100]

# marks = [45,45,96,75,36,45,87,99,100]

plt.hist(marks, bins = 7)
# plt.hist(marks, bins = [0,20,40,60,80,100])

plt.show()
```

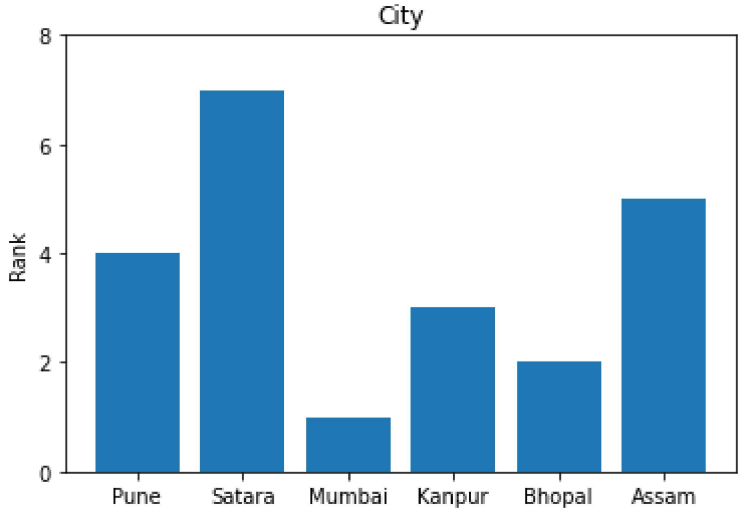


4. Bar Charts

```
import numpy as np
```

```
city = ('Pune', 'Satara', 'Mumbai', 'Kanpur', 'Bhopal', 'Assam')
y_val = np.arange(len(city))

print('y_val = ', y_val)

rank = [4, 7, 1, 3, 2, 5]

plt.bar(y_val, rank, align='center')
plt.xticks(y_val, city)  #is used to set the tick locations for x-axis.
plt.yticks([0,2,4,6,8])
plt.ylabel('Rank') #is used to set a label-text value to the data of y-axis.
plt.title('City')

plt.show()
```

```
y_val =  [0 1 2 3 4 5]
```



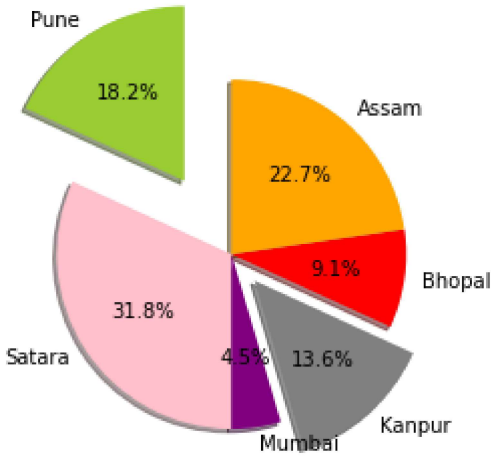

```
import numpy as np
np.arange(5)
```

```
array([0, 1, 2, 3, 4])
```

5. Pie Charts

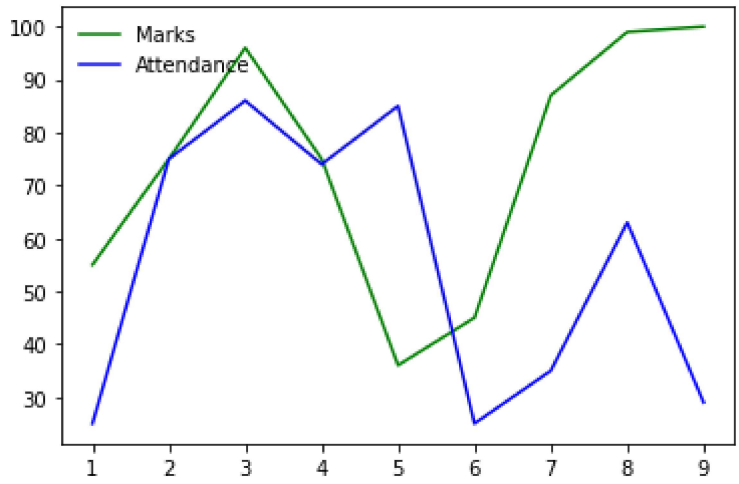
```
city = ('Pune', 'Satara', 'Mumbai', 'Kanpur', 'Bhopal', 'Assam')

rank = [4, 7, 1, 3, 2, 5]
explode = (0.5, 0, 0, 0.2, 0, 0)
colors = ['yellowgreen', 'pink', 'purple', 'grey', 'red', 'orange']

# plt.pie(rank, colors=['yellowgreen', 'pink', 'purple', 'grey', 'red', 'orange'])


plt.pie(rank,  explode=explode, labels=city, colors=colors,
        autopct='%1.1f%%', shadow=True, startangle=90)
```

```
#explode: provides a scalar value to set a fraction of the pie chart apart.
#labels: provides text values to represent each fraction of the chart.
#colors: provides the colors to set to each fraction of the chart.
#autopct: labels the wedges or the fractions of the chart with a numeric value.
#shadow: Accepts boolean values. If set to TRUE, it creates shadow beneath the fractions of t
#startangle: rotates the start of the chart by a particular degree from x-axis.
#pyplot.axis('equal') function enables equal scaling and creates scaled circle charts.


plt.show()
```



Adding features to Charts in Matplotlib

```
# x-axis values
roll_num = [1, 2, 3, 4, 5, 6, 7, 8, 9]

# y-axis values
marks = [55,75,96,75,36,45,87,99,100]
attendance = [25, 75, 86, 74, 85, 25, 35, 63, 29]

plt.plot(roll_num, marks, color = 'green', label = 'Marks')

plt.plot(roll_num, attendance, color = 'blue', label = 'Attendance')

plt.legend(loc='upper left', frameon=False)



#The parameter frameon accepts boolean values. If set to true,
#it creates a rectangular box like border around the labels placed
#by the position set through loc parameter.

plt.show()
```



Plotting with Pandas and Matplotlib

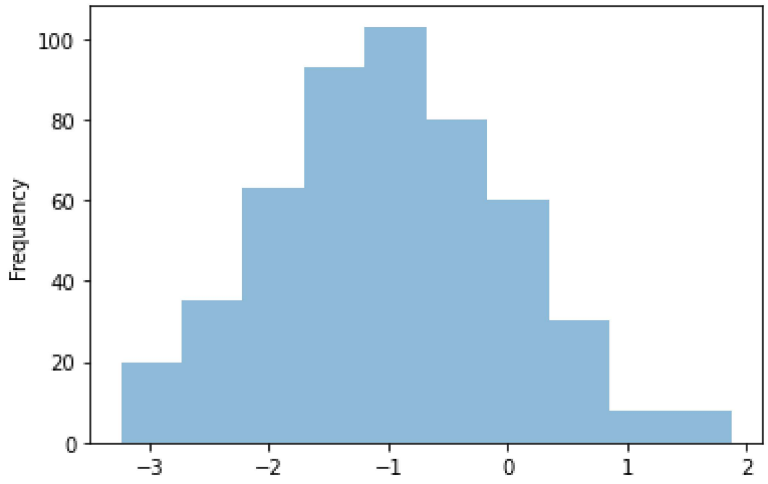
The following are the different types of graphs/charts to be used to plot data in Matplotlib with Pandas Module:

1. Histogram
2. Box Plot
3. Density Plot
4. Hexagonal Bin Plot
5. Scatter Plot
6. Area Plot
7. Pie Plot

1. Histogram

```
import pandas as pd

a=np.random.randn(500)

val = pd.DataFrame({'v1': a + 1, 'v2': a, 'v3': a - 1})


# val.plot.hist(alpha = 0.5)
val['v3'].plot.hist(alpha = 0.5)
#The parameter alpha is basically a float value used to blend the color scale of the plotted

#val1=val['v1']
#val1.plot.hist(alpha = 0.1)

plt.show()
```

```
val
```



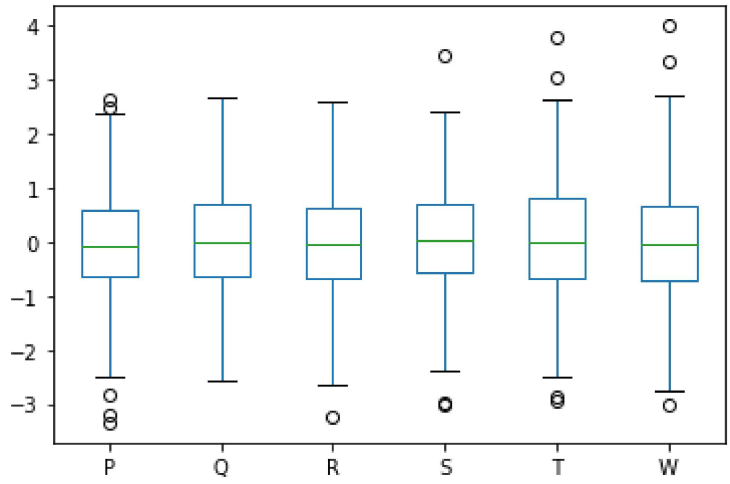
	v1	v2	v3
0	0.351956	-0.648044	-1.648044
1	1.592157	0.592157	-0.407843
2	1.730909	0.730909	-0.269091
3	2.581407	1.581407	0.581407
4	1.216732	0.216732	-0.783268
...	...	...	...
495	1.394459	0.394459	-0.605541
496	1.349369	0.349369	-0.650631
497	0.417171	-0.582829	-1.582829
498	1.005728	0.005728	-0.994272
499	1.053743	0.053743	-0.946257

500 rows × 3 columns

2. Box Plot

```
val = pd.DataFrame(np.random.randn(500,6), columns =['P', 'Q', 'R', 'S', 'T', 'W'])

#print(val)

val.plot.box()
plt.show()

# print(val['P'])
```

```
# print(np.mean(val['P']))
# print(np.std(val['P']))

first quartile
third quartile
median
```




```
np.max(val['P'])
```

```
2.644844155861491
```

Box Plot:

A box plot which is also known as a whisker plot displays a summary of a set of data containing the minimum, first quartile, median, third quartile, and maximum. In a box plot, we draw a box from the first quartile to the third quartile. A vertical line goes through the box at the median. The whiskers go from each quartile to the minimum or maximum.

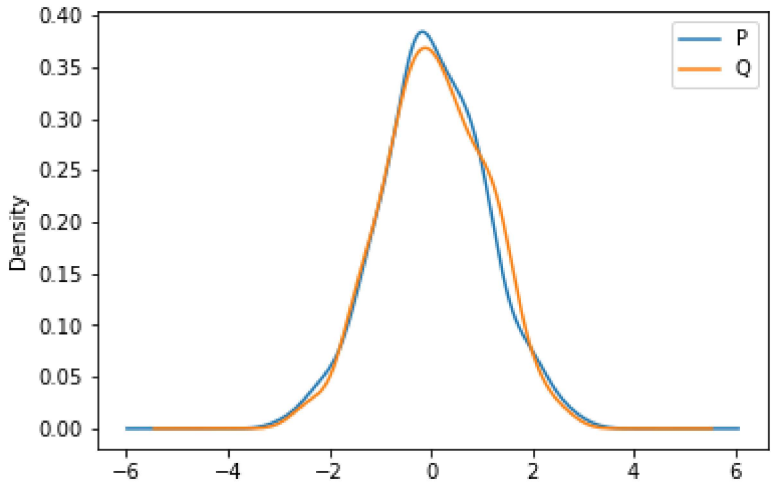
Median divides the lower 50% of the data from the upper 50% data. Quartile1 (Q1) divides the lower 25% of the data from the rest of the data. Quartile3 (Q3) divides the upper 25% of the data from the rest of the data. outliers - data that occurs very very less no. of times may be one time.

3. Density Plot

It is basically a Kernel Density Estimation (KDE) plot. It provides the probability density function of the input values.

```
import numpy as np
import pandas as pd

val = pd.DataFrame(np.random.randn(500,2),
columns =['P', 'Q',])
```

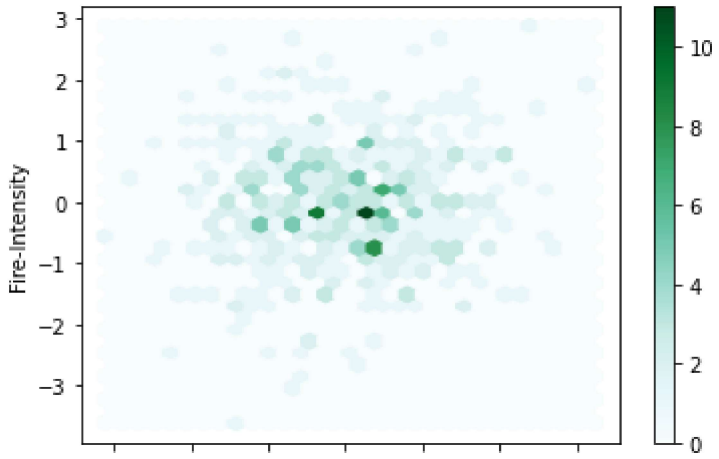
```
val.plot.kde()

plt.show()
# print(np.mean(val['P']))
# print(np.std(val['P']))
```



4. Hexagonal Bin Plot

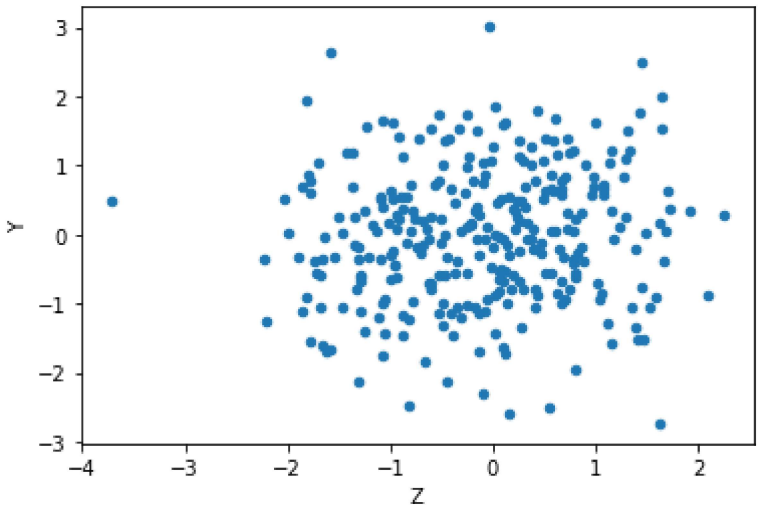
Hexagonal Bin Plot is used to estimate the relationship between two scalar values among a large set of data values.

```
val = pd.DataFrame(np.random.randn(500,2),
columns =['Temperature', 'Fire-Intensity',])
val.plot.hexbin(x ='Temperature', y ='Fire-Intensity', gridsize = 30)

plt.show()
```



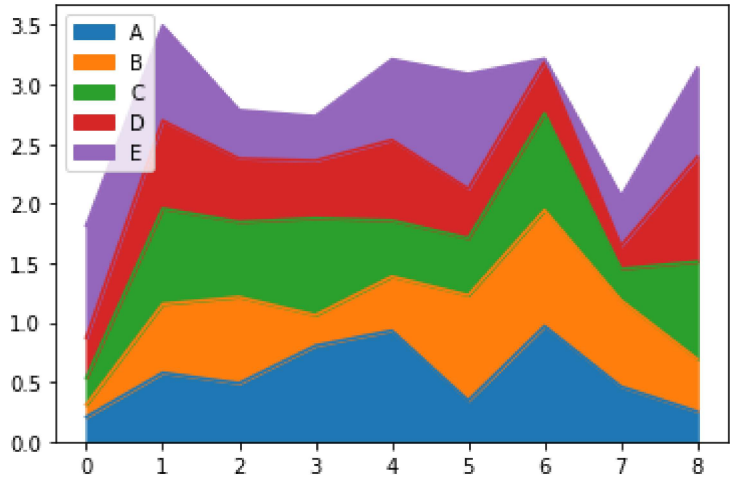
5. Scatter Plot

```
val = pd.DataFrame(np.random.randn(300,5),
columns =['A', 'Z', 'W', 'Y', 'S'])
val.plot.scatter(x='Z', y='Y')
plt.show()
```



6. Area Plot

```
val = pd.DataFrame(np.random.rand(9, 5), columns =['A', 'B', 'C', 'D', 'E'])

val.plot.area()
plt.show()
```



7. Pie Plot

```
val = pd.Series(np.random.rand(5),
index =['w','f', 'e', 'b', 'a'], name ='Pie-Chart')

val.plot.pie(figsize =(5, 5))
```

```
plt.show()
```