Matplotlib in Python:

Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. Matplotlib is written in Python and makes use of NumPy, the numerical mathematics extension of Python. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPythonotTkinter. It can be used in Python and IPython shells, Jupyter notebook and web application servers also.

Matplotlib has a procedural interface named the Pylab, which is designed to resemble MATLAB, a proprietary programming language developed by MathWorks. Matplotlib along with NumPy can be considered as the open source equivalent of MATLAB.

Matplotlib was originally written by John D. Hunter in 2003. The current stable version is 2.2.0 released in January 2018.

Installation of Matplotlib

```
pip install matplotlib
```

Importing and checking matplotlib version:

Program:

```
import matplotlib
print(matplotlib.__version__)
```

Output:

```
2.0.0
```

PyPlot

Plots (graphics), also known as charts, are a visual representation of data in the form of colored (mostly) graphics. It tells its audience the story about the data relationship through data points, lines, symbols, labels, and numbers so that professionals and anyone with limited knowledge of reading data can get a fair idea of what the data is trying to show. We can use the Matplotlib visualization library in Python to portray the graphs.

Most of the *Matplotlib* utilities lies under the *pyplot* submodule, and are usually imported under the *plt* alias:

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

Plot Types:

The six most commonly used Plots come under Matplotlib. These are:

- Bar Plot
- Line Plot
- Scatter Plot
- Pie Plot
- Histogram Plot

➢ Bar plot:

A bar plot or bar chart is a graph that represents the category of data with rectangular bars.A bar chart describes the comparisons between the discrete categories. One of the axis of the plot represents the specific categories being compared, while the other axis represents the measured values corresponding to those categories.

The **matplotlib** API in Python provides the bar() function which can be used in MATLAB style use or as an object-oriented API. The syntax of the bar() function to be used with the axes is as follows:-

Syntax:

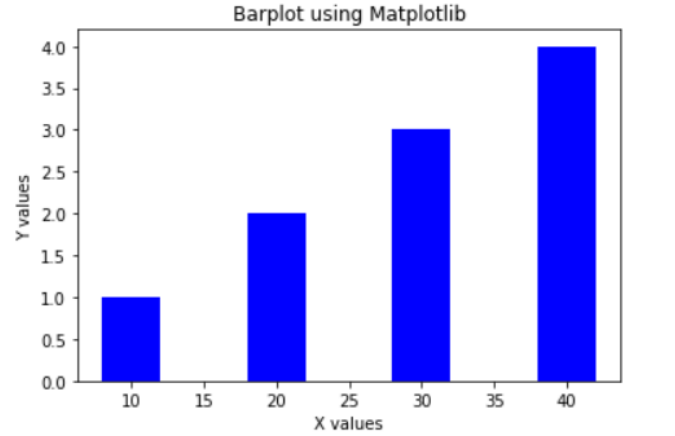
```
plt.bar(x, height, width, bottom, align)
```

The function creates a bar plot bounded with a rectangle depending on the given parameters.

Program:

```
import numpy as np
import matplotlib.pyplot as plt
x=[10,20,30,40]
y=[1,2,3,4]
plt.bar(x,y,color="blue",width= 4)
plt.xlabel("X values")
plt.ylabel("Y values")
plt.title("Barplot using Matplotlib")
plt.show()
```

Output:



- **Line Plot:**

The plot() function is used to draw points (markers) in a diagram. By default, the plot() function draws a line from point to point. The function takes parameters for specifying points in the diagram. Parameter 1 is an array containing the points on the x-axis. Parameter 2 is an array containing the points on the y-axis.

Syntax:

```
plt.plot(x,y, scalex=True, scaley=True)
```

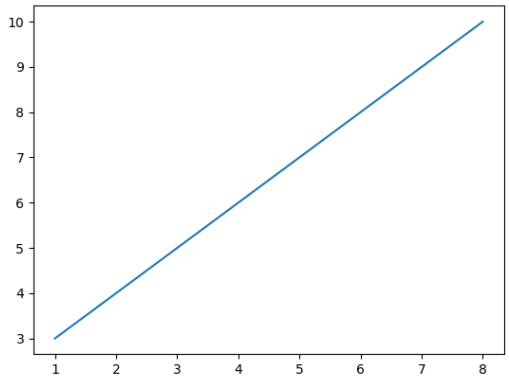
Program:

```
import matplotlib.pyplot as plt
import numpy as np

xpoints = np.array([1, 8])
ypoints = np.array([3, 10])

plt.plot(xpoints, ypoints)
plt.show()
```

Output:



- **Scatter Plot:**

Scatter plots are used to observe relationship between variables and uses dots to represent the relationship between them. The scatter() method in the matplotlib library is used to draw a scatter plot. Scatter plots are widely used to represent relation among variables and how change in one affects the other.

Syntax:

```
plt.scatter(x_axis_data, y_axis_data,)
```

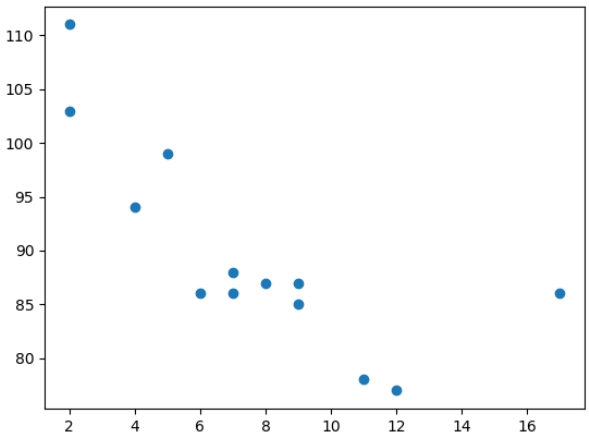
Program:

```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])

plt.scatter(x, y)
plt.show()
```

Output:



➢ **Pie Plot:**

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. The area of a wedge represents the relative percentage of that part with respect to whole data. Pie charts are commonly used in business presentations like sales, operations, survey results, resources, etc as they provide a quick summary.

Syntax:

```
plt.pie(data)
```

Program:

```
import matplotlib.pyplot as plt
import numpy as np

y = np.array([35, 25, 25, 15])

plt.pie(y)
plt.show()
```

Output:

➢ **Histogram Plot:**

A histogram is an accurate representation of the distribution of numerical data. It is an estimate of the probability distribution of a continuous variable. It is a kind of bar graph.

To construct a histogram, follow these steps −

- **Bin** the range of values.
- Divide the entire range of values into a series of intervals.
- Count how many values fall into each interval.

The bins are usually specified as consecutive, non-overlapping intervals of a variable.

The **matplotlib.pyplot.hist()** function plots a histogram. It computes and draws the histogram of x.

Program:

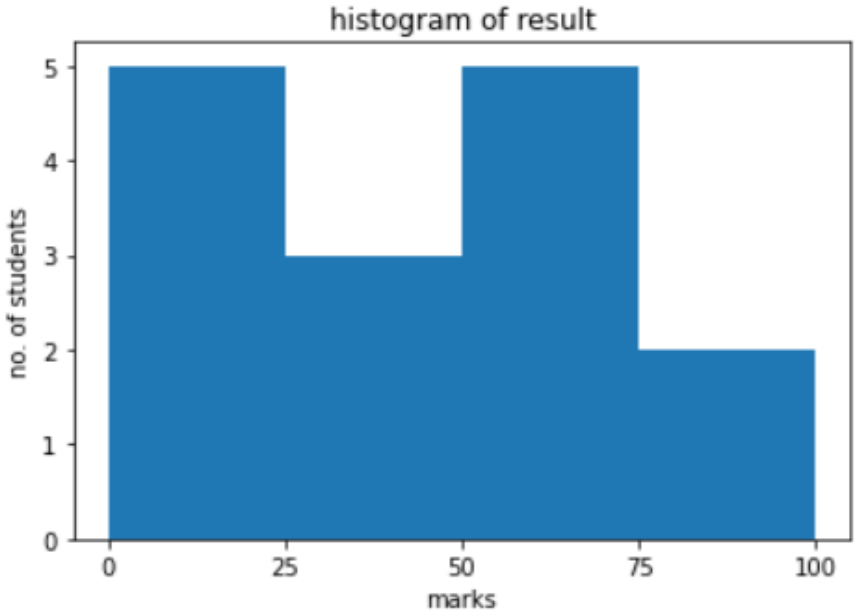
```
from matplotlib import pyplot as plt
import numpy as np
fig,ax = plt.subplots(1,1)
a = np.array([22,87,5,43,56,73,55,54,11,20,51,5,79,31,27])
```

```
ax.hist(a, bins = [0,25,50,75,100])

ax.set_title("histogram of result")
ax.set_xticks([0,25,50,75,100])
ax.set_xlabel('marks')
ax.set_ylabel('no. of students')

plt.show()
```

Output:



Polygons:

To plot shapely polygons and objects using matplotlib, the steps are as follows −

- Create a polygon object using (x, y) data points.
- Get x and y, the exterior data, and the array using polygon.exterior.xy.
- Plot x and y data points using plot() method with red color.

Syntax:

```
patches.Polygon(xy, *, closed=True, **kwargs)
```

Program:

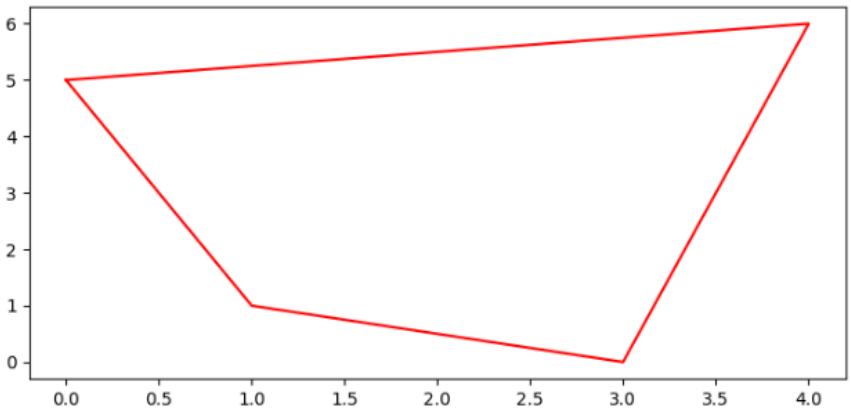
```
from shapely.geometry import Polygon
import matplotlib.pyplot as plt

plt.rcParams["figure.figsize"] = [7.00, 3.50]
plt.rcParams["figure.autolayout"] = True

polygon1 = Polygon([ (0, 5), (1, 1), (3, 0), (4, 6) ])
x, y = polygon1.exterior.xy

plt.plot(x, y, c="red")
plt.show()
```

Output:



➢ Box plots / Quartiles:

A Box Plot is also known as Whisker plot is created to display the summary of the set of data values having properties like minimum, first quartile, median, third ***quartile*** and maximum.

Creating Box Plot:

The *matplotlib.pyplot* module of *matplotlib* library provides *boxplot( )* function with the help of which we can create box plots.

Syntax:

```
plt.boxplot(data, patch_artist=None, widths=None)
```

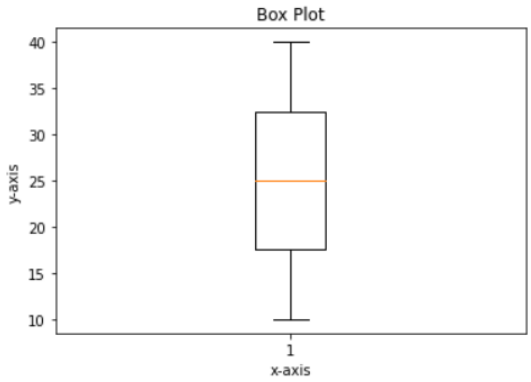
Program:

```
import matplotlib.pyplot as plt
import numpy as np

data=[10,20,30,40]
plt.boxplot(data)

plt.show()
```

Output:



Heat Maps:

A 2-D Heatmap is a data visualization tool that helps to represent the magnitude of the phenomenon in form of colors. In python, we can plot 2-D Heatmaps using Matplotlib package. There are different methods to plot 2-D Heatmaps.

Method 1: Using *matplotlib.pyplot.imshow()* Function

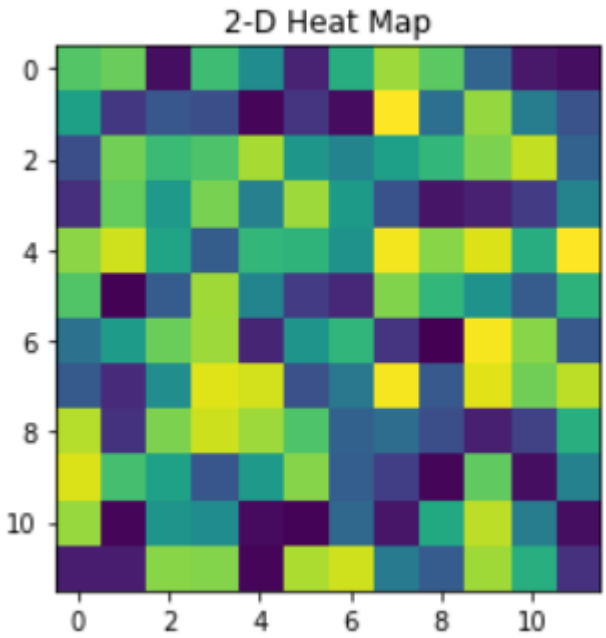
Program:

```
import numpy as np
import matplotlib.pyplot as plt

data = np.random.random(( 12 , 12 ))
plt.imshow( data)

plt.title( "2-D Heat Map" )
plt.show()
```

Output:



Method 2: Using Seaborn Library

Program:

```
import numpy as np
import seaborn as sns
import matplotlib.pylab as plt

data_set = np.random.rand( 10 , 10 )
ax = sns.heatmap( data_set , linewidth = 0.5 , cmap = 'coolwarm' )
plt.title( "2-D Heat Map" )

plt.show()
```

Output: