Plotting and Visualization:

Plotting and Visualization: A Brief matplotlib API Primer, Figures and Subplots, Colors, Markers, and Line Styles, Ticks, Labels, and Legends, Annotations and Drawing on a Subplot, Saving Plots to File, Plotting Functions in pandas, Line Plots, Bar Plots, Histograms and Density Plots, Scatter Plots.

Introduction to Matplotlib:

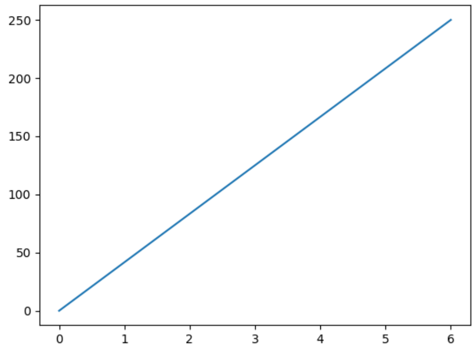
- Matploplib is a low-level library of Python which is used for data visualization. It is easy to use and emulates MATLAB like graphs and visualization. This library is built on the top of NumPy arrays and consists of several plots like line chart, bar chart, histogram, etc.
- Matplotlib is originally written by Dr. John D Hunter.
- We need to install matplotlib in command prompt by using

pip install matplotlib

- Pyplot is a subpackage of matplotlib.
  Example:

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

xpoints = np.array([0, 6])
ypoints = np.array([0, 250])

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

output:



- It provides object oriented API for embedded plots applications by using general purpose GUI like Tk, Wxpython, GTK etc..

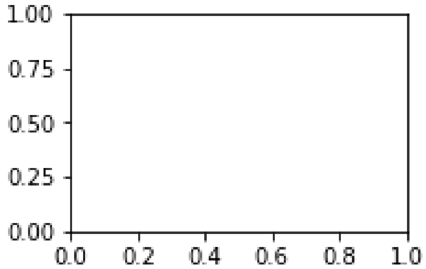
Plotting and Visualization:

Figures:

- The figure () function in pyplot module of matplotlib library is used to create a figure.
- Now let us see a example using figure () function.

```
import matplotlib.pyplot as plt
fig = plt.figure ()
ax1 = fig.add_subplot (2, 2, 1)
```

Output:



Subplots:

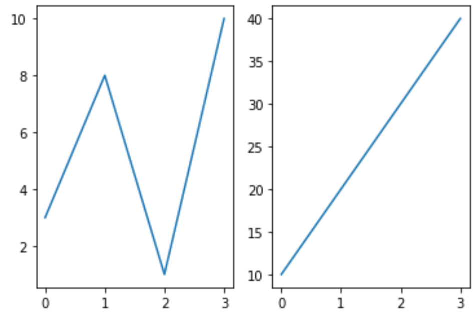
- Basically subplots are used to create many plots in a single plot. Now let see a example to easy of understanding.

Example-1:

```
import matplotlib.pyplot as plt
import numpy as np
#plot 1:
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])
plt.subplot(1, 2, 1)
plt.plot(x,y)
#plot 2:
x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])
plt.subplot(1, 2, 2)
plt.plot(x,y)
plt.show()
```

output:

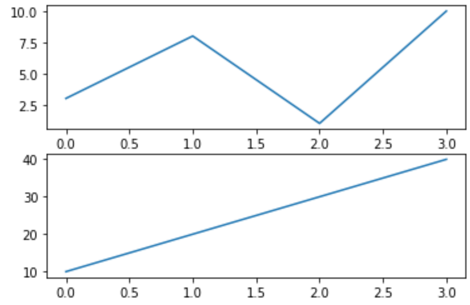
Plotting and Visualization:



Example-2:

```
import matplotlib.pyplot as plt
import numpy as np
#plot 1:
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])
plt.subplot(2, 1, 1)
plt.plot(x,y)
#plot 2:
x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])
plt.subplot(2, 1, 2)
plt.plot(x,y)
plt.show()
```

output:



- For syntax of subplots visit: Click here

Colors:

- We can change the color for the graph in plots by using color attribute in plt.plot() function. We can verify this by seeing the following example.

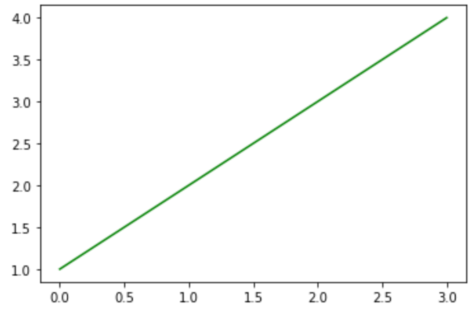
Example:

```
import matplotlib.pyplot as plt

color = 'green'
```

Plotting and Visualization:

```
plt.plot([1, 2, 3, 4], color = color)

plt.show()
```

output:



- We have different color options to change colors in plots and graphs. Now let us see the list of the colors used in changing of color.

Color syntax	Description
‘b’	Blue
‘r’	Red
‘g’	Green
‘c’	Cyan
‘m’	Magenta
‘y’	Yellow
‘k’	Black
‘w’	White

Marker Reference:

- Marker references are used to markers in the graphs. We have different markers as follows:

marker	symbol	description
"."	●	point
","	·	pixel

Plotting and Visualization:

marker	symbol	description
"o"	●	circle
"v"	▼	triangle_down
"^"	▲	triangle_up
"<"	◀	triangle_left
">"	▶	triangle_right
"1"	Y	tri_down
"2"	⅄	tri_up
"3"	≺	tri_left
"4"	≻	tri_right
"8"	⯃	octagon
"s"	■	square
"p"	⬟	pentagon
"P"	✚	plus (filled)
"*"	★	star
"h"	⬢	hexagon1
"H"	⬣	hexagon2
"+"	+	plus
"x"	×	x
"X"	✖	x (filled)
"D"	◆	diamond
"d"	⬧	thin_diamond
"|"	|	vline
"_"	—	hline
0 (TICKLEFT)	−	tickleft
1 (TICKRIGHT)	−	tickright
2 (TICKUP)	╵	tickup
3 (TICKDOWN)	╷	tickdown
4 (CARETLEFT)	◂	caretleft
5 (CARETRIGHT)	▸	caretright
6 (CARETUP)	▴	caretup
7 (CARETDOWN)	▾	caretdown
8 (CARETLEFTBASE)	◂	caretleft (centered at base)
9 (CARETRIGHTBASE)	▸	caretright (centered at base)
10 (CARETUPBASE)	▴	caretup (centered at base)
11 (CARETDOWNBASE)	▾	caretdown (centered at base)
"None", " " or ""		nothing
'$...$'	*f*	Render the string using mathtext. E.g "f" for marker

marker	symbol	description
		showing the letter f.
verts		A list of (x, y) pairs used for Path vertices. The center of the marker is located at (0, 0) and the size is normalized, such that the created path is encapsulated inside the unit cell.
path		A **Path** instance.
(numsides, 0, angle)		A regular polygon with numsides sides, rotated by angle.
(numsides, 1, angle)		A star-like symbol with numsides sides, rotated by angle.
(numsides, 2, angle)		An asterisk with numsides sides, rotated by angle.

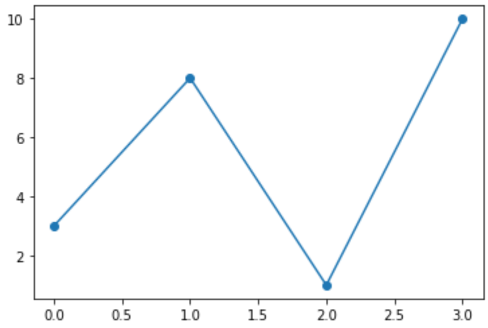
- Now we saw about the marker reference, we can also change the color of the marker reference. To change the color for marker we use an argument “markeredgecolor” or shortly we use “mec” to change outer edge of the marker. We can also change the color of the part inside of the edge of the marker by using an argument “markerfacecolor” or shortly we use “mfc”.
- Now let us see an example for both the above argument.

Example:

```
import matplotlib.pyplot as plt

import numpy as np

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

plt.plot (ypoints, marker = 'o')

plt.show ()
```

output:

Plotting and Visualization:

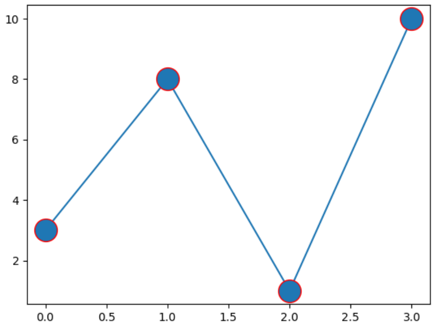


- Now the given below example is for changing the color of the edge of the marker.

Example:

```
import matplotlib.pyplot as plt

import numpy as np

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

plt.plot (ypoints, marker = 'o', ms = 20, mec = 'r')

plt.show ()
```

output:

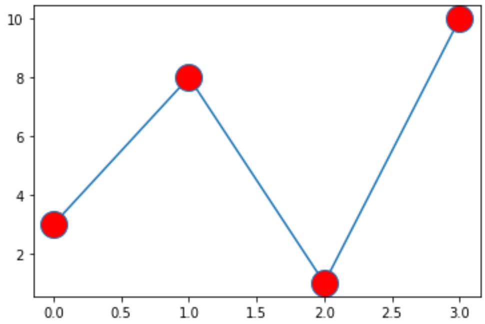


- Now let us see the example for changing the color for inside of the marker.

Example:

```
import matplotlib.pyplot as plt

import numpy as np

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

plt.plot(ypoints, marker = 'o', ms = 20, mfc = 'r')
plt.show()
```

output:

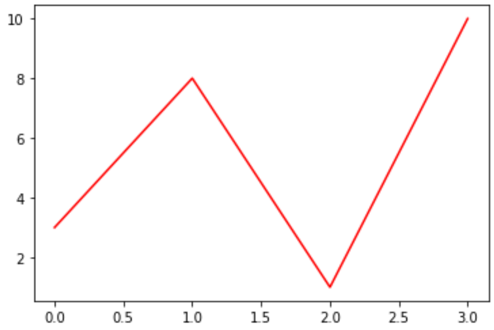


- We can also change the color for the line of the graph by using “color” or shortly we use “c”.

Example:

```
import matplotlib.pyplot as plt

import numpy as np

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

plt.plot(ypoints, color = 'r')

plt.show()
```

output:



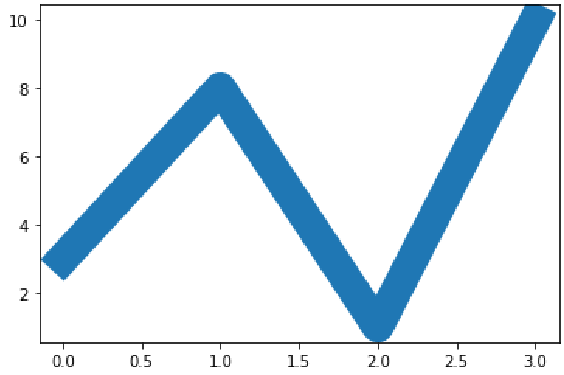
- We can also change the width of the line by using an argument “linewidth” or shortly we use “lw”.

Example:

```
import matplotlib.pyplot as plt

import numpy as np
```

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

plt.plot(ypoints, linewidth = '20.5')

plt.show()
```

output:



Line Styles:

- As we learnt that we can the change the color and the width of the graph but we can also change style of the line in the graph by using an attribute “line style”. Here we use shortly ‘ls’ to change the line style.

- We have different line styles in matplotlib and this are as follows:

Character	**Definition**
–	Solid line
—	Dashed line
-.	dash-dot line
:	Dotted line

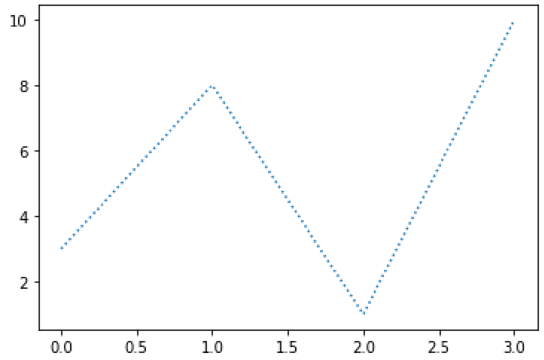
Example-1:

```
import matplotlib.pyplot as plt

import numpy as np

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

plt.plot(ypoints, linestyle = 'dotted')

plt.show()

output:



Example-2:

import matplotlib.pyplot as plt

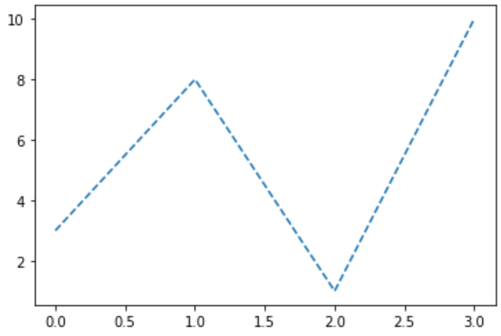
import numpy as np

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

plt.plot(ypoints, linestyle = 'dashed')

plt.show()

output:



Ticks:

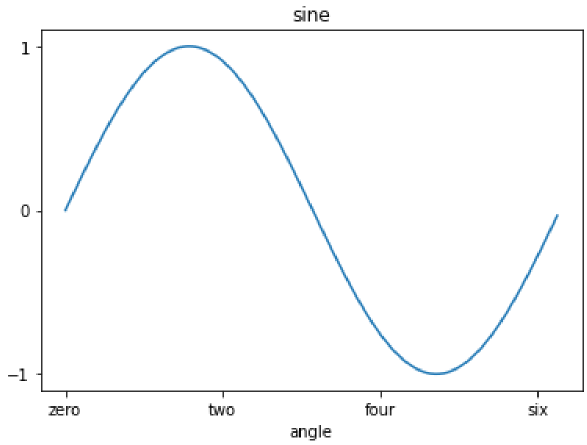
- Ticks are the values used to show specific points on the coordinate axis. It can be a number or a string.
- Position and labels of ticks can be explicitly mentioned to suit specific requirements.
- The xticks () and yticks () function takes a list object as argument. The elements in the list denote the positions on corresponding action where ticks will be displayed.

Plotting and Visualization:

Example:

```
import matplotlib.pyplot as plt

import numpy as np

import math

x = np.arange(0, math.pi*2, 0.05)

fig = plt.figure()

ax = fig.add_axes([0.1, 0.1, 0.8, 0.8]) # main axes

y = np.sin(x)

ax.plot(x, y)

ax.set_xlabel('angle')

ax.set_title('sine')

ax.set_xticks([0,2,4,6])

ax.set_xticklabels(['zero','two','four','six'])

ax.set_yticks([-1,0,1])

plt.show()
```

output:



Labels:

- Labels are used to determine the

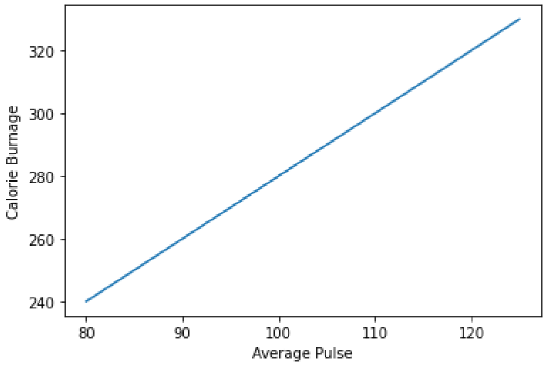
Plotting and Visualization:

- As of now we learnt about the graph but we don’t understand the values of the axes so we use the label for representation of the values of the axes in the graph.
- Labels are of two types: 1.) xlabel 2.)ylabel
- xlabel determines the label for the x-axis and ylabel determines the label for the yaxis. Now let us see an example:

Example:

```
import numpy  as np

import matplotlib.pyplot as plt

x = np.array ([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])

y = np.array ([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

plt.plot(x, y)

plt.xlabel ("Average Pulse")

plt.ylabel ("Calorie Burnage")

plt.show ()
```

output:



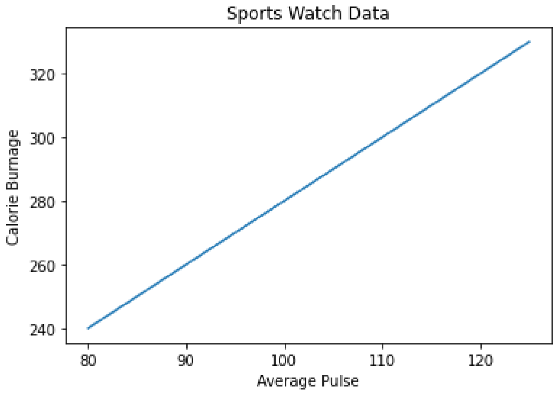
- And we can also give the label for the graph by using a method in the matplotlib.pyplot “title” .

Example:

```
import numpy as np

import matplotlib.pyplot as plt

x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
```

```
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

plt.plot(x, y)

plt.title("Sports Watch Data")

plt.xlabel("Average Pulse")

plt.ylabel("Calorie Burnage")

plt.show()
```

output:



Legend:

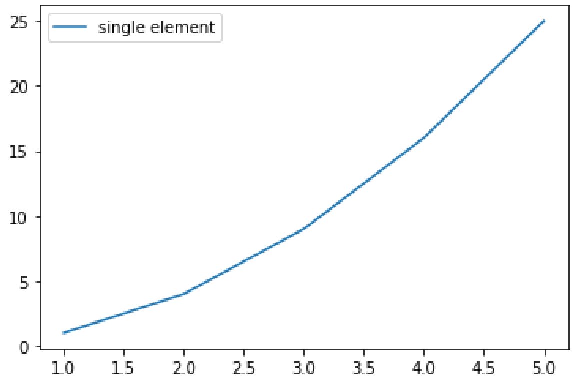
- A legend is an area describing the elements of the graph. In the matplotlib library, there’s a function called **legend ()** which is used to Place a legend on the axes.
- The attribute **Loc** in legend () is used to specify the location of the legend. Default value of loc is loc=”best” (upper left). The strings ‘upper left’, ‘upper right’, ‘lower left’, ‘lower right’ place the legend at the corresponding corner of the axes/figure.
- In simple words ‘legend’ are used to the describe
- Now let us discuss an example based on legend:

Example-1:

```
import numpy as np

import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]

y = [1, 4, 9, 16, 25]
```

```
plt.plot(x, y)

plt.legend(['single element'])

plt.show()
```

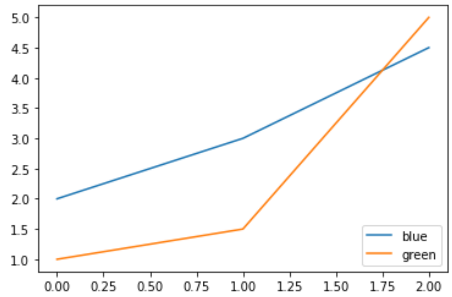
output:



Example-2:

```
import numpy as np
import matplotlib.pyplot as plt
y1 = [2, 3, 4.5]
y2 = [1, 1.5, 5]
plt.plot(y1)
plt.plot(y2)
plt.legend(["blue", "green"], loc ="lower right")
plt.show()
```

output:



Annotations:

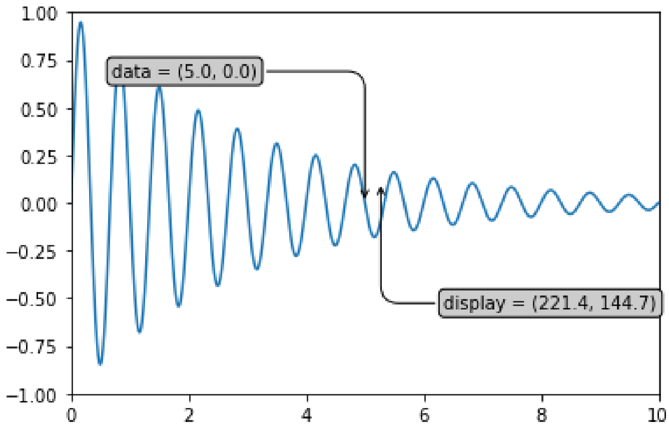
- The annotation() is a function which is used to describe about a particular point in a graph.The syntax of annotation is annotate().
- The annotate() function in pyplot module of matplotlib library is used to annotate the point xy with text s.

Plotting and Visualization:

Example:

```
import numpy as np

import matplotlib.pyplot as plt

x = np.arange(0, 10, 0.005)

y = np.exp(-x / 3.) * np.sin(3 * np.pi * x)

fig, ax = plt.subplots()

ax.plot(x, y)

ax.set_xlim(0, 10)

ax.set_ylim(-1, 1)

# Setting up the parameters

xdata, ydata = 5, 0

xdisplay, ydisplay = ax.transData.transform((xdata, ydata))

bbox = dict(boxstyle ="round", fc ="0.8")

arrowprops = dict(

    arrowstyle = "->",

    connectionstyle = "angle, angleA = 0, angleB = 90,\

    rad = 10")

offset = 72

# Annotation

ax.annotate('data = (%.1f, %.1f)'%(xdata, ydata),

            (xdata, ydata), xytext =(-2 * offset, offset),

            textcoords ='offset points',

            bbox = bbox, arrowprops = arrowprops)

disp = ax.annotate('display = (%.1f, %.1f)'%(xdisplay, ydisplay),

            (xdisplay, ydisplay), xytext =(0.5 * offset, -offset),
            xycoords ='figure pixels',

            textcoords ='offset points',

            bbox = bbox, arrowprops = arrowprops)

# To display the annotation

plt.show()
```

output:



Drawing on a Subplot:

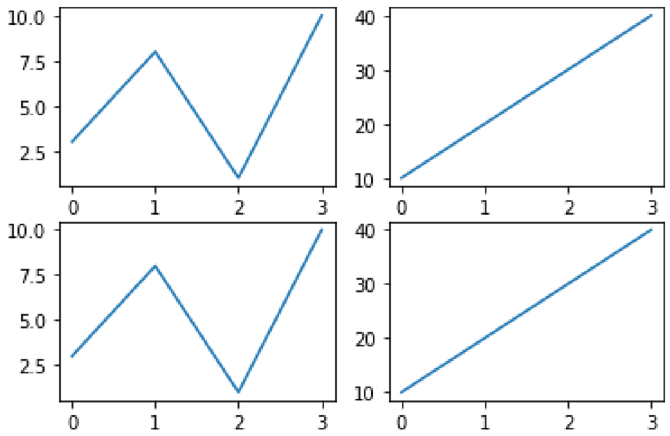
- Matplotlib.pyplot has convenience function called subplot which is used in creating the common layout of subplots, including the enclosing figure object in a single call.
- The syntax of subplots is “plt.subplots(nrows,ncols)” .

Example:

```
import matplotlib.pyplot as plt

import numpy as np

x = np.array([0, 1, 2, 3])

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

plt.subplot(2, 2, 1)

plt.plot(x,y)

x = np.array([0, 1, 2, 3])

y = np.array([10, 20, 30, 40])
```

```
plt.subplot(2, 2, 2)

plt.plot(x,y)

x = np.array([0, 1, 2, 3])

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

plt.subplot(2, 2, 3)

plt.plot(x,y)

x = np.array([0, 1, 2, 3])

y = np.array([10, 20, 30, 40])

plt.subplot(2, 2, 4)

plt.plot(x,y)

plt.show()
```

output:



Saving Plots to File:

- Matplotlib is a widely used python library to plot graphs, plots, charts, etc. show() method is used to display graphs as output, but don’t save it in any file.
- To save generated graphs in a file on storage disk, savefig() method is used.
- Basically we use savefig() method to save the plots or the graphs that generated by the our analysis on the data.

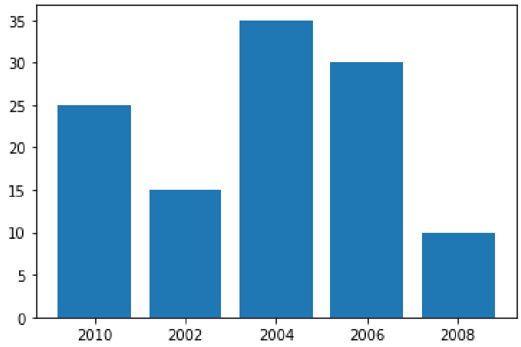
Example : import matplotlib.pyplot as plt

```
# Creating data
```

Plotting and Visualization:

```
year = ['2010', '2002', '2004', '2006', '2008']

production = [25, 15, 35, 30, 10]

# Plotting barchart

plt.bar(year, production)

# Saving the figure.

plt.savefig("output.jpg")

# Saving figure by changing parameter values

plt.savefig("output1", facecolor='y', bbox_inches="tight",

            pad_inches=0.3, transparent=True)
```

ouput:



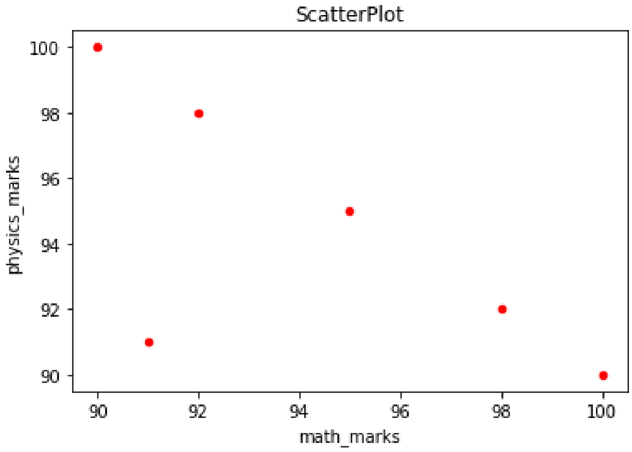
Plotting functions in pandas:

- We have different function for visualization in pandas.
- Here we use the dataset for the data visualization. We can plot line,bar, histograms,scatter plot,box plot, etc …
- We have different functions to plot different graphical structures.
- For plotting different graphical structures we use a function “pd.DataFrame.plot” .

Example:

```
import pandas as pd

import matplotlib.pyplot as plt

data_dict = { 'name':['p1','p2','p3','p4','p5','p6'],

              'age':[20,20,21,20,21,20],
            'math_marks':[100,90,91,98,92,95],

            'physics_marks':[90,100,91,92,98,95],

            'chem_marks' :[93,89,99,92,94,92]

            }

df = pd.DataFrame(data_dict)

df.head()

df.plot(kind = 'scatter',

        x = 'math_marks',

        y = 'physics_marks',

        color = 'red')

plt.title('ScatterPlot')

plt.show()
```

output:



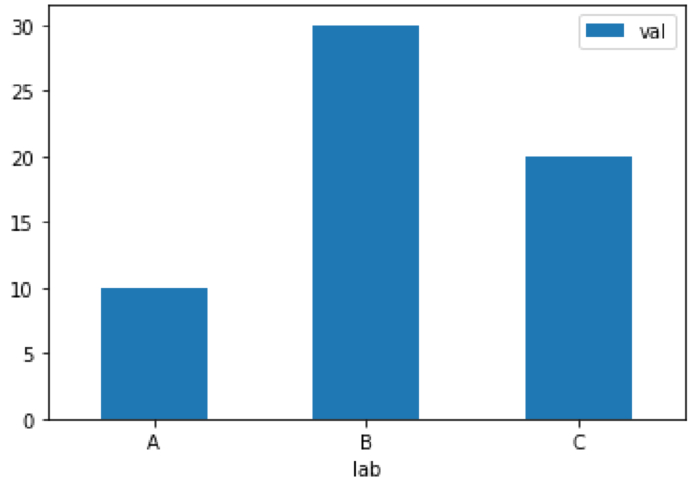
- In the above example there is an attribute called kind in plot() function, it can be assigned with different types of graphical representations.
- To plot a bar graph we use “pd.DataFrame.plot.bar()”.

```
import pandas as pd

import matplotlib.pyplot as plt

df = pd.DataFrame({'lab':['A', 'B', 'C'], 'val':[10, 30, 20]})
```

Plotting and Visualization:

```
ax = df.plot.bar(x='lab', y='val', rot=0)
```

output:

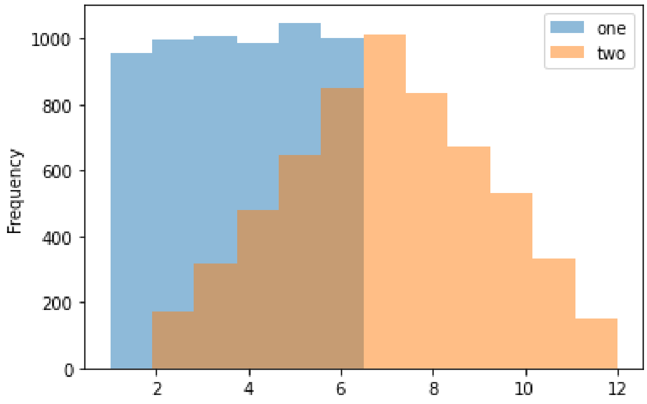


- To plot a histogram we use “pd.DataFrame.plot.hist”

Example:

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.DataFrame(
    np.random.randint(1, 7, 6000),
    columns = ['one'])
df['two'] = df['one'] + np.random.randint(1, 7, 6000)
ax = df.plot.hist(bins=12, alpha=0.5)
```

output:



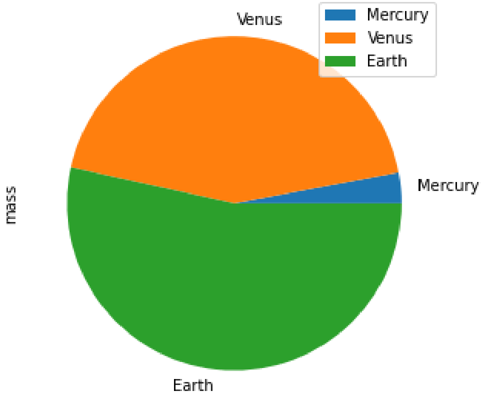
Plotting and Visualization:

- To plot a pie chart we use “pd.DataFrame.plt.pie”

Example:

```
import pandas as pd

import matplotlib.pyplot as plt

df = pd.DataFrame({'mass': [0.330, 4.87 , 5.97],

                   'radius': [2439.7, 6051.8, 6378.1]},

                  index=['Mercury', 'Venus', 'Earth'])

plot = df.plot.pie(y='mass', figsize=(5, 5))
```

output:



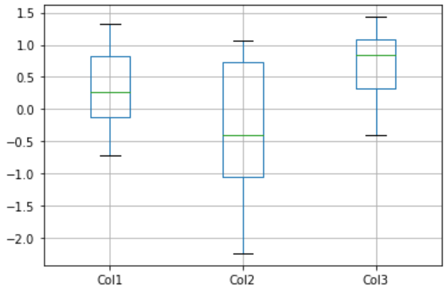
- To plot the boxplot we use “pd.DataFrame.boxplot”

Example:

```
import pandas as pd

import matplotlib.pyplot as plt

np.random.seed(1234)

df = pd.DataFrame(np.random.randn(10, 4),

                  columns=['Col1', 'Col2', 'Col3', 'Col4'])

boxplot = df.boxplot(column=['Col1', 'Col2', 'Col3'])
```

output:

Plotting and Visualization:



- Similarly we can plot all the different types of graphical structures using the pandas data frames and series.

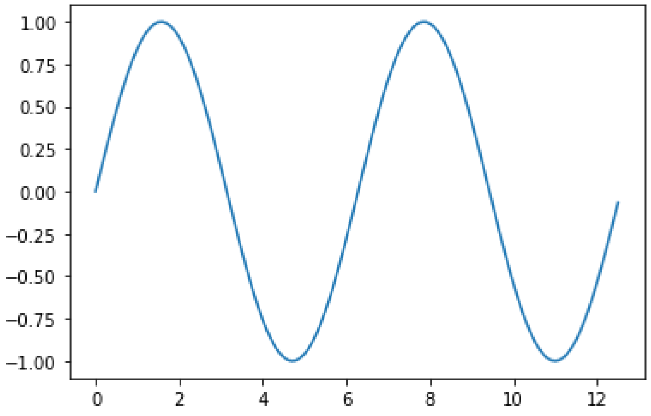
Line plot:

- Line plots can be created in Python with Matplotlib's pyplot library.
- To build a line plot, first import Matplotlib. It is a standard convention to import Matplotlib's pyplot library as plt.
- The plt alias will be familiar to other Python programmers.

Example:

```
import pandas as pd

import matplotlib.pyplot as plt

x = np.arange(0, 4 * np.pi, 0.1)

y = np.sin(x)

plt.plot(x, y)

plt.show()
```

output:



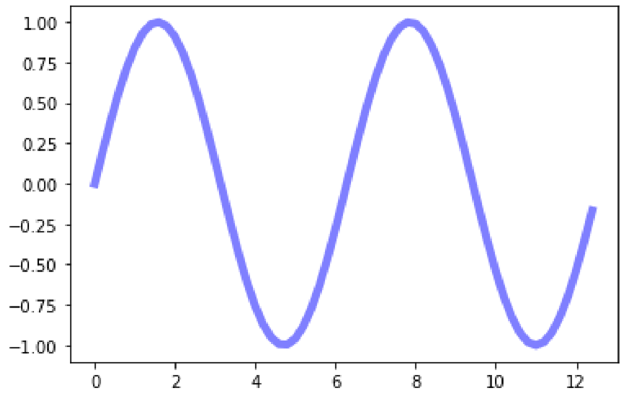
Plotting and Visualization:

- Matplotlib provides us some additonal features like:
  1. Line color.
  2. Line width.
  3. Line opacity.
  4. Line markers.

Example:

```
import numpy as np

import matplotlib.pyplot as plt

x = np.arange(0, 4 * np.pi, 0.2)

y = np.sin(x)

plt.plot(x,y,color='blue',alpha=0.5,lw=5)

plt.show()
```

output:



Bar plot:

- A bar chart or bar graph is a chart or graph that presents categorical data with rectangular bars with heights or lengths proportional to the values that they represent.
- The bar() function takes arguments that describes the layout of the bar.
- The bars can be plotted vertically or horizontally.To plot the bar graph horizontally we use a function called barh().

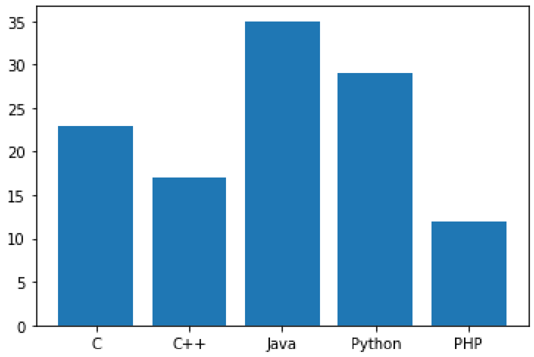
Example-1:

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

Plotting and Visualization:

```
x = ['C', 'C++', 'Java', 'Python', 'PHP']

y = [23,17,35,29,12]

plt.bar(x,y)

plt.show()
```

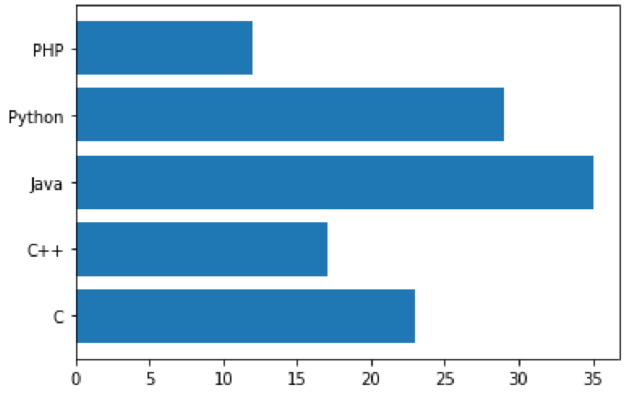
output:



Example-2:

```
import matplotlib.pyplot as plt

x = ['C', 'C++', 'Java', 'Python', 'PHP']

y = [23,17,35,29,12]

plt.barh(x,y)

plt.show()
```

output:



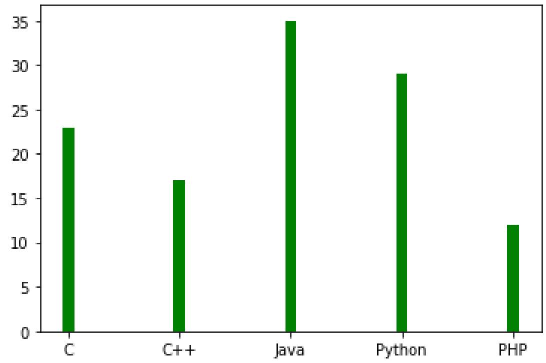
- Same as every plots we can change the Color, Width.

Plotting and Visualization:

Example:

```
import matplotlib.pyplot as plt
x = ['C', 'C++', 'Java', 'Python', 'PHP']
y = [23,17,35,29,12]
plt.bar(x,y,color='green',width=0.1)
plt.show()
```

output:



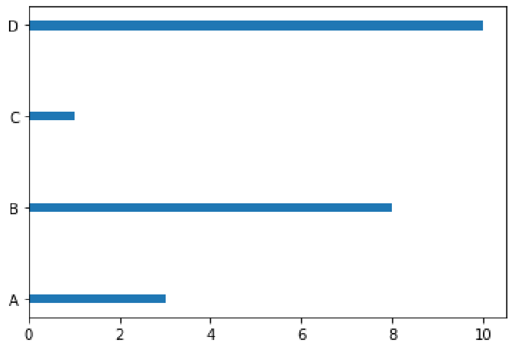
- The barh() function takes a argument called height.

Example:

```
import matplotlib.pyplot as plt
import numpy as np
x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])
plt.barh(x, y, height = 0.1)
plt.show()
```

output:

Plotting and Visualization:



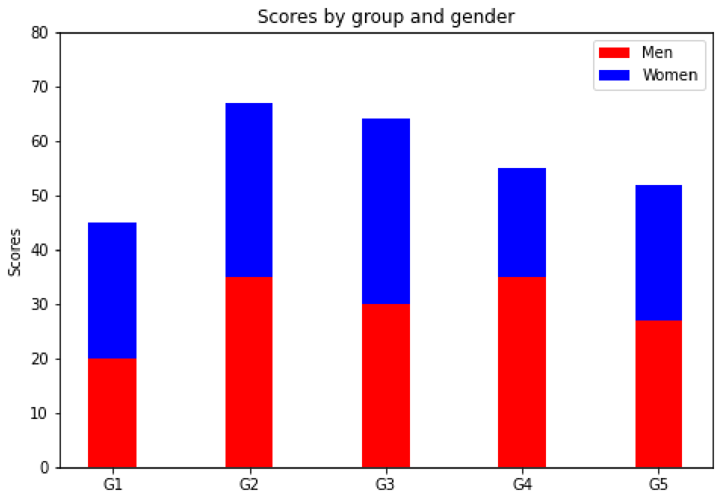
- We can perform every operations like legend, labels, titles and axes.

Example:

```
import numpy as np

import matplotlib.pyplot as plt

N = 5

menMeans = (20, 35, 30, 35, 27)

womenMeans = (25, 32, 34, 20, 25)

ind = np.arange(N) # the x locations for the groups

width = 0.35

fig = plt.figure()

ax = fig.add_axes([0,0,1,1])

ax.bar(ind, menMeans, width, color='r')

ax.bar(ind, womenMeans, width,bottom=menMeans, color='b')

ax.set_ylabel('Scores')

ax.set_title('Scores by group and gender')

ax.set_xticks(ind, ('G1', 'G2', 'G3', 'G4', 'G5'))

ax.set_yticks(np.arange(0, 81, 10))

ax.legend(labels=['Men', 'Women'])

plt.show()
```

output

Plotting and Visualization:



Histograms:

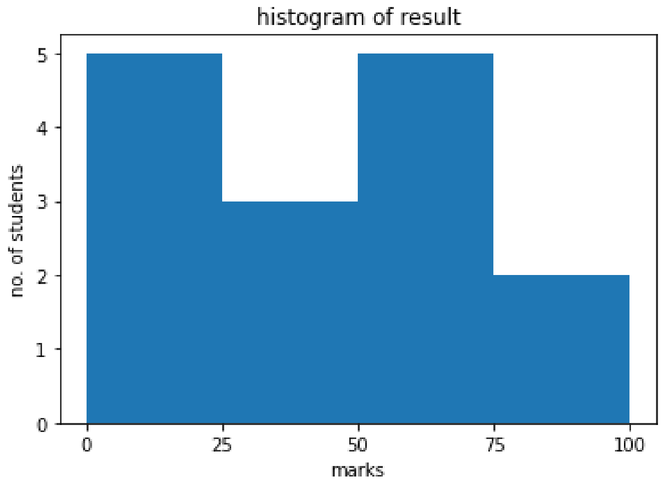
- A histogram is an accurate representation of the distribution of numerical data. It is an estimate of the probability distribution of a continuous variable.
- It is a kind of bar graph.We use a function called hist() to plot the histogram.

Example:

```
import matplotlib.pyplot as plt

import numpy as np

a = np.array([22,87,5,43,56,73,55,54,11,20,51,5,79,31,27])

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

plt.title("histogram of result")

plt.xticks([0,25,50,75,100])

plt.xlabel('marks')

plt.ylabel('no. of students')

plt.show()
```

output:

Plotting and Visualization:



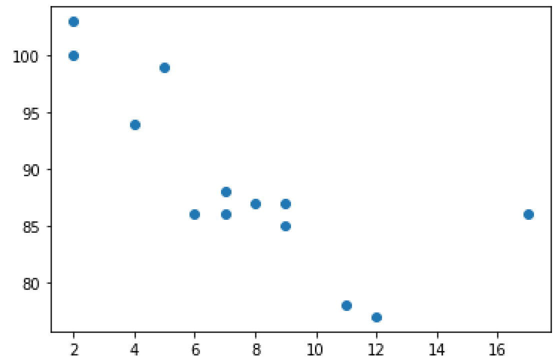
Scatter Plots:

- To plot a scatter plot we use a method called scatter().
- Mainly these scatter plots are used to observer the relationship between the varibles and use dots to represent the relation between them.

Example:

```
import numpy as np

import matplotlib.pyplot as plt

x =[5, 7, 8, 7, 2, 17, 2, 9,4, 11, 12, 9, 6]

y =[99, 86, 87, 88, 100, 86,103, 87, 94, 78, 77, 85, 86]

plt.scatter(x, y)

plt.show()
```

output:



- The main important factor in the process of plotting the scatter plot we need to ensure that the size of the two arrays are same.

Plotting and Visualization:

- Now let us see what will happen if we give the array of different sizes.

Example:

```
import numpy as np

import matplotlib.pyplot as plt

x =[5, 7, 8, 7, 2, 17, 2, 9,4, 11, 12, 9, 6]

y =[99, 86, 87, 88, 100, 86,103, 87, 94, 78, 77]

plt.scatter(x, y)

plt.show()
```

output:

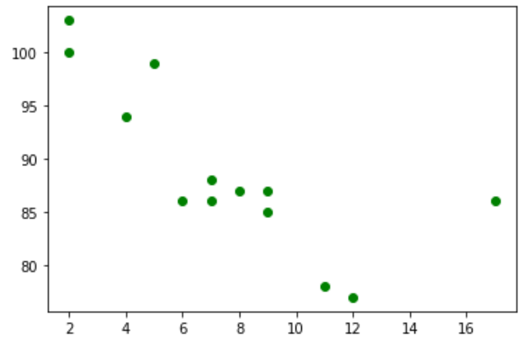
```
ValueError: x and y must be the same size.
```

- By default we have blue and orange colors in the scatter plot.
- We can change the color of the dots in scatter plot we use “color” or “c” argument in scatter() method.

Example:

```
import numpy as np

import matplotlib.pyplot as plt

x =[5, 7, 8, 7, 2, 17, 2, 9,4, 11, 12, 9, 6]

y =[99, 86, 87, 88, 100, 86,103, 87, 94, 78, 77, 85, 86]

plt.scatter(x, y,color='green')

plt.show()
```

output:

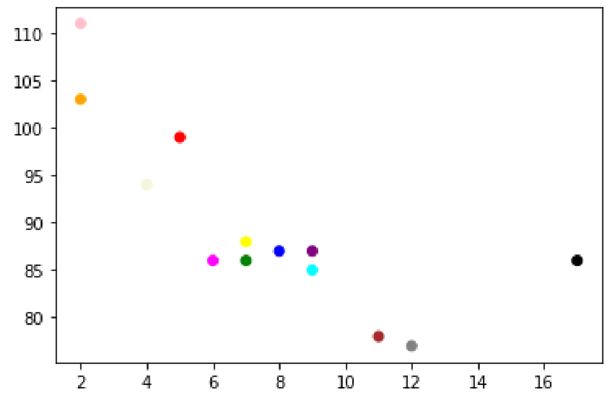


- We can also assign each dot a different color.

Example:

```
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])

colors = np.array(["red","green","blue","yellow","pink","black","orange","purple","beige","brown","gray","cyan","magenta"])

plt.scatter(x, y, c=colors)

plt.show()
```

output:



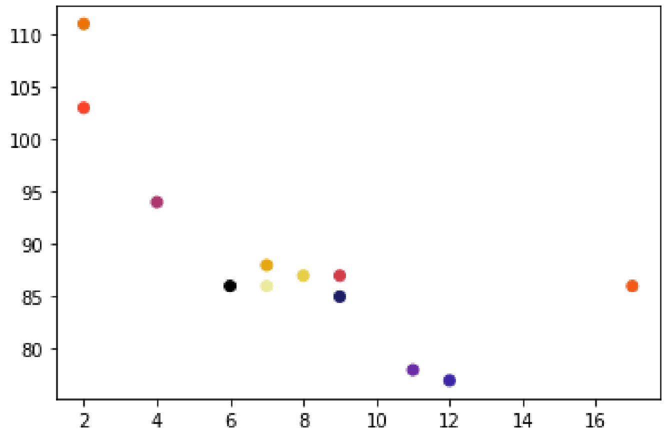
- In matplotlib module contains many types of color maps.
- To plot the scatter plot using the color map we use “cmap”

Example:

Plotting and Visualization:

```
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])

colors = np.array([0, 10, 20, 30, 40, 45, 50, 55, 60, 70, 80, 90, 100])

plt.scatter(x, y, c=colors, cmap='CMRmap_r')

plt.show()
```

output:

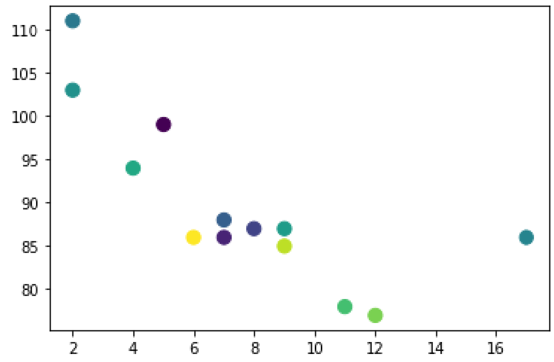


- We can also change the size of the dots in scatter plot by using “s” argument in the scatter() method.

Example:

```
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])

colors = np.array([0, 10, 20, 30, 40, 45, 50, 55, 60, 70, 80, 90, 100])

plt.scatter(x, y, c=colors, s=85)

plt.show()
```

output:

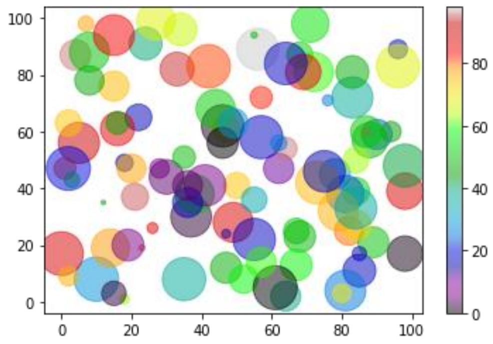


- As in the line chart we can also change the transparency of the dots in the scatter plot we use “alpha” argument.

Example:

```
import matplotlib.pyplot as plt

import numpy as np

x = np.random.randint(100, size=(100))

y = np.random.randint(100, size=(100))

colors = np.random.randint(100, size=(100))

sizes = 10 * np.random.randint(100, size=(100))

plt.scatter(x, y, c=colors, s=sizes, alpha=0.5, cmap='nipy_spectral')

plt.colorbar()

plt.show()
```

output:



-

Plotting and Visualization: