**Matplotlib**

**import matplotlib.pyplot as plt**

from matplotlib.pylab import rcParams

rcParams['figure.figsize'] = 15, 6

**from matplotlib import pyplot as plt1**

plt.plot([1,2,3,4,5])

plt.show()

plt.plot([1,2,1.5,3,2])

plt.show()

plt.plot([2,4,6,8,10],[1,2,1.5,5,3])

plt.show()

**Graph with title**

x = [2,4,6,8,10]

y = [1,2,1.5,5,3]

plt.plot (x,y)

plt.title ('First Graph')

plt.xlabel ('Month ')

plt.ylabel ('Sales')

plt.show()

**Graph generation with Square**

x= np.arange (0,16,2)

plt.plot(x, [x1\*\*2 for x1 in x])

plt.show()

**Multi Line Graph**

x= np.arange (0,16,2)

plt.plot(x, [x1 for x1 in x])

plt.plot(x, [x1\*\*2 for x1 in x])

plt.plot(x, [x1\*\*3 for x1 in x])

plt.show()

**Limit the x and Y axis values**

x= np.arange (0,100,2)

plt.plot(x, [x1 for x1 in x])

plt.plot(x, [x1\*\*2 for x1 in x])

plt.plot(x, [x1\*\*3 for x1 in x])

plt.axis ([0,20,0,80])

plt.show()

**Using XLIM and Y LIM**

x= np.arange (0,100,2)

plt.plot(x, [x1 for x1 in x])

plt.plot(x, [x1\*\*2 for x1 in x])

plt.plot(x, [x1\*\*3 for x1 in x])

plt.xlim ([0,20])

plt.ylim ([0,80])

plt.show()

**Style.use**

<https://matplotlib.org/tutorials/introductory/customizing.html>

from matplotlib import style

import matplotlib.style

style.use ('ggplot')

x =[10,12,14,16]

y= [20,30,15,25]

x1= [11,13,15,17]

y1= [32,28,34,24]

plt.plot (x,y,'g',label = 'Sales',linewidth = 10)

plt.plot (x1,y1,'y',label = 'Profit',linewidth = 3)

plt.title ('First Graph')

plt.xlabel ('X -Axis')

plt.ylabel ('Y-Axis')

plt.legend()

plt.grid(True, color = 'k')

**Line Chart with Label**

x = ['10','12','14','16','18']

y = [15,10,12,18,20]

plt.plot (x,y)

#plt.text (x - axis position , y axis position ie hight , value/text to be displayed ,

# ha = Horizontal alignment va = vertical alignment)

for i in range(len(x)):

print ('i=',i, 'and y[i] =',y[i])

plt.text(i,y[i],y[i],ha ='center', va= 'bottom')

**Bar Graph**

plt.bar ([10,12,14,16,18],[15,10,12,18,20],label = 'Example 1',color = 'g')

plt.bar ([11,13,15,17,19],[20,17,25,12,28], label = 'Example 2',color = 'b')

plt.title ('Graph Two')

plt.xlabel ('X- Axis')

plt.ylabel ('Y-Axis')

plt.legend()

plt.show()

**Bar Graph**

plt.bar ([10,12,14,16,18],[15,10,12,18,20],label = 'Example 1',color = 'g')

plt.bar ([11,13,15,17,19],[20,17,25,12,28], label = 'Example 2',color = 'b')

plt.bar ([11.1,13.1,15.1,17.1,19.1],[12,20,20,22,28], label = 'Example 3',color = 'r')

plt.title ('Graph Two')

plt.xlabel ('X- Axis')

plt.ylabel ('Y-Axis')

plt.legend()

plt.show()

**Two Bars Overlapping**

plt.bar ([10,12,14,16,18],[15,10,12,18,20],label = 'Example 1',color = 'g')

plt.bar ([11,13,15,17,19],[20,17,25,12,28], label = 'Example 2',color = 'b')

plt.bar ([11,13,15,17,19],[12,20,20,22,28], label = 'Example 3',color = 'r')

plt.title ('Graph Two')

plt.xlabel ('X- Axis')

plt.ylabel ('Y-Axis')

plt.legend()

plt.show()

**Bar Graph with width – Example 1**

plt.bar ([10,12,14,16,18],[15,10,12,18,20],label = 'Example 1',color = 'g')

plt.bar ([11,13,15,17,19],[20,17,25,12,28], label = 'Example 2',width=1,color = 'b')

plt.bar ([11,13,15,17,19],[12,20,20,22,28], label = 'Example 3',width=0.7,color = 'c')

plt.title ('Graph Two')

plt.xlabel ('X- Axis')

plt.ylabel ('Y-Axis')

plt.legend()

plt.show()

**Bar Graph with width – Example 2**

plt.bar ([10,12,14,16,18],[15,10,12,18,20],label = 'Example 1',width=1.2,color = 'g')

plt.bar ([10,12,14,16,18],[20,17,25,12,28], label = 'Example 2',width=0.7,color = 'b')

plt.title ('Graph Two')

plt.xlabel ('X- Axis')

plt.ylabel ('Y-Axis')

plt.legend()

plt.show()

**Bar graph with Label**

# Fc = Face color # ec = Edge Color

x = ['10','12','14','16','18']

y = [15,10,12,18,20]

plt.bar (x,y,label = 'Example 1',width=.8,fc = 'lightgreen',ec = 'r')

#plt.text (x - axis position , y axis position ie hight , value/text to be displayed ,

# ha = Horizontal alignment va = vertical alignment )

for i in range(len(x)):

plt.text(i,y[i],y[i],ha ='center', va= 'bottom')

plt.legend()

plt.show()

**Histogram**

age = [10,12,45,32,42,22,21,28,35,14,16,20,33,33,35,38,32,42,45,48,18,19,26]

age\_group= [0,10,20,30,40,50]

plt.hist (age,age\_group)

plt.title ('Graph Three')

plt.xlabel ('X- Axis')

plt.ylabel ('Y-Axis')

plt.show()

**Histogram With Formatting**

age = [10,12,45,32,42,22,21,28,35,14,16,20,33,33,35,38,32,42,45,48,18,19,26]

age\_group= [0,10,20,30,40,50]

plt.hist (age,age\_group,histtype = 'bar',rwidth =0.5,color = 'b',label ='Age Group')

plt.title ('Graph Three')

plt.xlabel ('X- Axis')

plt.ylabel ('Y-Axis')

plt.legend()

plt.show()

**Histogram With different Bin Size**

age = [10,12,45,32,42,22,21,28,35,14,16,20,33,33,35,38,32,42,45,48,18,19,26]

age\_group= [10,30,30,35,40,50]

plt.hist (age,age\_group,histtype = 'bar',rwidth =0.5,color = 'b',label ='Age Group')

plt.title ('Graph Three')

plt.xlabel ('X- Axis')

plt.ylabel ('Y-Axis')

plt.legend()

plt.show()

**Histogram with Steps**

age = [10,12,45,32,42,22,21,28,35,14,16,20,33,33,35,38,32,42,45,48,18,19,26]

age\_group= [10,20,30,40,50]

plt.hist (age,age\_group,histtype = 'step',color = 'b',label ='Age Group')

plt.title ('Graph Three')

plt.xlabel ('X- Axis')

plt.ylabel ('Y-Axis')

plt.legend()

plt.show()

**Histogram from Numpy Series**

y=np.random.randn (10,10)

plt.hist (y)

plt.show ()

**Scatter plot**

x = [1,2,3,4,5,6,7,8,9,10]

y = [18,16,11,15,13,18,13,16,14,18]

plt.scatter (x,y,label = 'Sales',color ='k')

plt.title ('Graph Four')

plt.xlabel ('X- Axis')

plt.ylabel ('Y-Axis')

# Legend Positions - ``'upper left', 'upper right', 'lower left', 'lower right'``

plt.legend(loc= 'upper right')

plt.show()

**Area Graph- Basic**

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

sleep = [7,6,4,5,8]

eat = [2,3,4,2,1]

work =[8,9,10,8,9]

play=[4,5,3,3,2]

plt.stackplot (days,sleep,eat,work,play)

plt.title ('Graph Five')

plt.xlabel ('X- Axis')

plt.ylabel ('Y-Axis')

plt.show()

**Area Graph**

**Example 1**

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

sleep = [7,6,4,5,8]

eat = [2,3,4,2,1]

work =[8,9,10,8,9]

play=[4,5,3,3,2]

plt.stackplot (days,sleep,eat,work,play,labels= ('sleep','eat','work','play'),

colors= ['m','r','k','g'])

plt.title ('Graph Five')

plt.xlabel ('X- Axis')

plt.ylabel ('Y-Axis')

plt.legend()

plt.show()

**Example 2**

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

sleep = [7,6,4,5,8]

eat = [2,3,4,2,1]

work =[8,9,10,8,9]

play=[4,5,3,3,2]

plt.stackplot (days,sleep,eat,work,play,colors= ['m','r','k','g'])

plt.plot ([],[],color = 'm',label='sleep',linewidth = 2)

plt.plot ([],[],color = 'r',label='eat',linewidth = 2)

plt.plot ([],[],color = 'k',label='work',linewidth = 2)

plt.plot ([],[],color = 'g',label='play',linewidth = 2)

plt.title ('Graph Five')

plt.xlabel ('X- Axis')

plt.ylabel ('Y-Axis')

plt.legend()

plt.show()

**Pie Chart – Basic**

hours = [5,8,3,4]

plt.pie (hours)

plt.show()

**Pie Chart – autopct with one decimal point**

hours = [5,8,3,4]

activity = ['Sleep','Work','Eat','Play']

col=['g','r','c','y']

plt.pie(hours,labels = activity,colors = col,startangle = 90,shadow = True,

explode = (0,0,0,0.1),autopct = '%1.1f%%')

plt.title ('Pie Chart')

plt.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.

'''

Other Parameters for plt.axis

**'scaled'** Set equal scaling (i.e., make circles circular) by

changing dimensions of the plot box. This is the same as

``ax.set\_aspect('equal', adjustable='box', anchor='C')``.

Additionally, further autoscaling will be disabled.

**'tight'** Set limits just large enough to show all data, then

disable further autoscaling.

**'auto'** Automatic scaling (fill plot box with data).

'image' 'scaled' with axis limits equal to data limits.

'square' Square plot; similar to 'scaled', but initially forcing

``xmax-xmin == ymax-ymin``.

'''

plt.show()

**Pie Chart – autopct and Lebel examples**

hours = [5,8,3,4]

activity = ['Sleep','Work','Eat','Play']

col=['g','r','c','y']

plt.subplot (221) # autopct with three decimal point

plt.pie(hours,labels = activity,autopct = '%1.3f%%')

plt.axis('equal')

plt.legend(loc='upper left')

plt.subplot (222)# autopct with integer

plt.pie(hours,labels = activity,autopct = '%d%%')

plt.title ('Pie Chart')

plt.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.

plt.legend(loc='upper right')

plt.show()

**Donut Chart**

plt.figure(figsize= (5,5))

hours = [5,8,3,4]

plt.pie (hours)

inner\_circle = plt.Circle((0,0), 0.70, fc = 'w')

'''

**matplotlib.pyplot.gca()**

The gca() function in pyplot module of matplotlib library

is used to get the current Axes instance on the current figure

matching the given keyword args, or create one.

'''

plt.gca().add\_artist(inner\_circle)

plt.show()

**Plot Circles**

import matplotlib.pyplot as plt

# (0,0) - Centre of Circle. .2- Radius of Circle

# clip\_on=False - Otherwise only portion of circle visible

circle1=plt.Circle((0,0),.2,color='r',clip\_on=False)

circle2=plt.Circle((.5,.5),.2,color='b')

circle3=plt.Circle((1,1),.2,color='g',clip\_on=False)

# gcf ()- get current figure

fig = plt.gcf()

# gca () - get current axis

# add\_artist() The Axes. add\_artist() function is used to add an Artist to the axes.

fig.gca().add\_artist(circle1)

fig.gca().add\_artist(circle2)

fig.gca().add\_artist(circle3)

plt.show()

**Donut Chart**

# library

import matplotlib.pyplot as plt

# create data

names = ['Group A', 'Group B', 'Group C', 'Group D']

size = [12,11,3,30]

# Create a circle at the center of the plot

my\_circle = plt.Circle( (0,0), 0.7, color='white')

# Give color names

plt.pie(size, labels=names, colors=['red','green','blue','skyblue'])

# gcf – Get Current Figure and gca – Get Current axis

p = plt.gcf()

p.gca().add\_artist(my\_circle)

# Show the graph

plt.show()

**Donut Chart**

sleep = [7,6,4,5,8]

eat = [2,3,4,2,1]

work =[8,9,10,8,9]

play=[4,5,3,3,2]

plt.figure(figsize= (10,10))

plt.pie(sleep,pctdistance = 0.92,autopct = '%1.1f%%, \n (sleep)', radius = 1.0)

plt.pie(eat,pctdistance = 0.88,autopct = '%1.1f%% \n (eat)', radius = 0.80)

plt.pie(work,pctdistance = 0.85,autopct = '%1.1f%% \n (work', radius = 0.60)

plt.pie(play,pctdistance = 0.80,autopct = '%1.1f%% \n (play)', radius = 0.40)

inner\_circle = plt.Circle((0,0), 0.20, fc = 'white')

plt.gca().add\_artist(inner\_circle)

plt.show()

**Multiple Graph**

**Example 1**

x1=np.arange(10,30,2)

y1 = [1,5,6,7,8,3,4,6,5,4]

x2= np.arange (10,30,2)

y2 = [5,4,2,8,4,6,3,4,5,4]

plt.subplot (211)

plt.plot(x1,y1)

plt.title ('Graph 1')

plt.xlabel ('X-Axis')

plt.ylabel ('Y-Axis')

plt.subplot(212)

plt.plot(x2,y2)

plt.title ('Graph 2')

plt.xlabel ('X-Axis')

plt.ylabel ('Y-Axis')

plt.show()

**Example 2 b = colour o -circle**

def f(t):

return (np.exp(-t)\*np.cos(2\*np.pi\*t))

t1=np.arange(10,20,2)

t2= np.arange (11,21,2)

plt.subplot (211)

plt.plot(t1,f(t1),**'bo'**,t2,f(t2))

plt.title ('Graph 1')

plt.xlabel ('X-Axis')

plt.ylabel ('Y-Axis')

plt.subplot(212)

plt.plot(t2,np.cos(2\*np.pi\*t2))

plt.title ('Graph 2')

plt.xlabel ('X-Axis')

plt.ylabel ('Y-Axis')

plt.show()

tight\_layout()

The tight\_layout() function in pyplot module of matplotlib library is used to automatically adjust subplot parameters to give specified padding.

x1=np.arange(10,30,2)

y1 = [1,5,6,7,8,3,4,6,5,4]

x2= np.arange (10,30,2)

y2 = [5,4,2,8,4,6,3,4,5,4]

plt.subplot (211)

plt.plot(x1,y1)

plt.title ('Graph 1')

plt.xlabel ('X-Axis')

plt.ylabel ('Y-Axis')

plt.subplot(212)

plt.plot(x2,y2)

plt.title ('Graph 2')

plt.xlabel ('X-Axis')

plt.ylabel ('Y-Axis')

plt.tight\_layout(pad=1, h\_pad=1, w\_pad=1) # The labels are not overlapping

plt.show()

**Create Graph from data set**

import pandas as pd

from matplotlib import pyplot as plt

plt.style.use('seaborn')

data =pd.read\_csv('C:\\Noble\\Training\\Top Mentor\\Training\\Data Set\\Matplotlib\_data.csv')

ages = data['Age']

dev\_salaries = data['All\_Devs']

py\_salaries = data['Python']

js\_salaries = data['JavaScript']

plt.plot(ages, py\_salaries, label='Python')

plt.plot(ages, js\_salaries, label='JavaScript')

plt.plot(ages, dev\_salaries, color='#444444',

linestyle='--', label='All Devs')

plt.legend()

plt.title('Median Salary (USD) by Age')

plt.xlabel('Ages')

plt.ylabel('Median Salary (USD)')

plt.tight\_layout()

plt.show()

**Create Subplot with Figure and Axis**

import pandas as pd

from matplotlib import pyplot as plt

plt.style.use('seaborn')

data = pd.read\_csv('C:\\Noble\\Training\\Top Mentor\\Training\\Data Set\\Matplotlib\_data.csv')

ages = data['Age']

dev\_salaries = data['All\_Devs']

py\_salaries = data['Python']

js\_salaries = data['JavaScript']

# Fig1 and fig 2 are figure

# ax1 and ax2 are axis

fig1, ax1 = plt.subplots()

fig2, ax2 = plt.subplots()

print ('Axis 1 ',ax1)

print ('Axis 2 ',ax2)

ax1.plot(ages, dev\_salaries, color='#444444',

linestyle='--', label='All Devs')

ax2.plot(ages, py\_salaries, label='Python')

ax2.plot(ages, js\_salaries, label='JavaScript')

ax1.legend()

ax1.set\_title('Median Salary (USD) by Age')

ax1.set\_ylabel('Median Salary (USD)')

ax2.legend()

ax2.set\_xlabel('Ages')

ax2.set\_ylabel('Median Salary (USD)')

plt.tight\_layout()

plt.show()

fig1.savefig('fig1.png')

fig2.savefig('fig2.png')

**Graph with different line style**

y = np.arange (10)

plt.plot(y,'--',y\*2, ':', y\*3,'-', y\*4,'-.')

plt.show()

**Control Marker style**

y = np.arange (10)

plt.plot(y,'\*',y+5, 'o',y\*2,'d')

plt.show()

**Save Graph**

x= np.arange (0,100,2)

plt.plot(x, [x1 for x1 in x], label = 'Linear Graph')

plt.plot(x, [x1\*\*2 for x1 in x], label = 'Square Graph')

plt.plot(x, [x1\*\*3 for x1 in x], label = 'Three times of x')

plt.xlim ([0,20])

plt.ylim ([0,80])

plt.xlabel ('X-Axis')

plt.ylabel ('Y-Axis')

plt.legend()

plt.savefig ('Multi Line Graph.png')

plt.show()