(1)
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
print(plt.style.available)
plt.style.use('classic')
#plt.style.use('ggplot')
%matplotlib inline
# static images of the plot embedded in the notebook
```

Session on 2/12/2021

(2)
```
#Line color and marker
x = np.linspace(0, 10, 100)
plt.plot(x, np.sin(x - 0), color='blue', marker ='+')   # color by
name
plt.plot(x, np.sin(x - 1), color='g', marker = '*')      # short color
code (rgbcmyk)
plt.plot(x, np.sin(x - 2), color='0.75', marker = 's')  # Grayscale
between 0 and 1
plt.plot(x, np.sin(x - 4), color=(1.0,0.2,0.3), marker = 'd') # RGB
tuple, values 0 and 1
```

(3)
```
# An equal aspect ratio i.e. one unit in x is equal to one unit in y
import numpy as np
x = np.linspace(0, 10, 100)
plt.plot(x, np.sin(x))
plt.axis('tight')
#plt.axis('equal')
```

(4)
```
import numpy as np
x = np.linspace(0, 10, 100)
plt.plot(x, np.sin(x))
plt.axis('tight')
#plt.axis('equal')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Sine Graph')
plt.savefig('plot.png')
plt.show()

from IPython.display import Image
Image('plot.png')
```

(5)
```
#Subplots in Matplotlib
plt.style.use('seaborn-whitegrid')
import numpy as np
x = np.arange(5)           #a numpy array [0,1,2,3,4]
#plt.subplot(nrow, ncol, pos)
plt.subplot(1,2,1)     #Divides the figure window into 1 row and 2
columns, polt goes to position 1
```

```
plt.plot(x, x, label='linear')    # plot x versus x
plt.subplot(1,2,2)
plt.plot(x, x*x, label='square')
```

```
(6)
#Subplots in Matplotlib
import numpy as np
x = np.arange(5)
plt.subplot(2,2,1)     #Divides the figure window into 1 row and 2
columns
plt.plot(x, x, label='linear')
plt.subplot(2,2,2)
plt.plot(x, x*x, label='square')
plt.subplot(2,2,3)
plt.plot(x, x*x*x, label='cube')
plt.subplot(2,2,4)
plt.plot(x, x*x*x*x, label='fourth')
#add a x-axis lablels and title to each
```

```
(7)
#grid of subplots
for i in range(1, 7):
    plt.subplot(2, 3, i)
    plt.text(0.5, 0.5, str((2,3,i)), fontsize = 18, ha = 'center')
```

```
(8)
# Bar Plots
#When both x and Y are discrete or categorical
x=[2,8,10]                    #x takes discrete values
y=[11,15,9]     #height or frequency of each bar
plt.bar(x,y, color='r', label='red')

x2=[12,5,15]
y2=[6,8,10]
plt.bar(x2,y2,color='g', label ='green')
plt.legend()        # use this whenever  labels are used
plt.title("Bar plot")
plt.xlabel("X axis")
plt.ylabel("Y axis")
```

```
(9)
# Simple bar plot
import numpy as np
import matplotlib.pyplot as plt
cities=['Delhi','Mumbai','Bangalore','Hydrabad']
population=[23456123,10083104,18456123,13411093]
plt.bar(cities,population,width=[.3,.1,.7,.4],color=['r','b','g','y'])

#plt.barh(cities, population,color=['r','b','g','y'])
plt.xlabel("Cities")
plt.ylabel("Population")
Exercise:
#plotting revenue and profit side by side
company=['GOOGL','AMZN','MSFT','FB']
revenue=[90,136,89,27]
profit=[40,2,34,12]
```

```
xpos=np.arange(len(company))
plt.xticks(xpos,company)
plt.bar(xpos-0.2,revenue,width=0.3,label='revenue')
plt.bar(xpos+0.2,profit,color='blue',width=0.3,label='profit')
plt.legend()
plt.xlabel("Stocks")
plt.ylabel("revenue and profit")


(10)
#Recall of Univariate Line plot
plt.plot([1, 2, 3, 4, 8])   #default values on x-axis are 0,1,2,3,4

(11)#HistoGram
# Histogram shows the quantitave values
ages = [32, 67, 113, 89, 65, 102, 93, 54, 69, 103, 74, 87, 105, 78,
116, 28, 92, 108, 29, 118, 135, 76, 84, 78, 23, 104, 75, 81]
age_groups = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120,
130, 140]
plt.hist(ages, age_groups, rwidth = 0.9)

plt.xlabel('age_groups')
plt.ylabel('population_ages')
plt.title("Histogram")


(12)
data=np.random.randn(100)
plt.hist(data)
bins=[-3,-2,-1,1,2,3]   #integer or sequence or 'auto', optional these
nos are on the left of the bin
plt.hist(data, bins=bins, color='blue',  edgecolor='green')

(13)
#hist function has many options to tune calculations and display
data=np.random.randn(100)
plt.hist(data,bins=30,
histtype='stepfilled',color='steelblue',alpha=0.5)
 #alpha lightens the color of hist

(14)
np.random.seed(100)
n_bins = 5
x = np.random.randn(100, 3)    #three data could be say Revenue Profit
and Sales of a Company
print(x.shape)
colors = ['red', 'tan', 'lime']
#plt.hist(x, n_bins, density= None, histtype='bar', color = colors,
label = colors)
#plt.hist(x, n_bins, density= None, histtype='bar',  color = colors,
label = colors, stacked = True)
plt.legend()
plt.xlabel("Stocks")
plt.ylabel("revenue and profit")
#plt.show()
#explain 3 arrays and bin intervals in output
Scatter Plot(15)
```

```
import matplotlib.pyplot as plt
import numpy as np
x = np.random.randn(10)
y = np.random.randn(10)
plt.scatter(x,y)             # Just marks the coordinates on the x-
axis.
#Understand why we dont use a Line plot here
#plt.plot(x,y)

(16)
# Scatter plot using plt.plot()
x = np.linspace(0, 10, 50)
y = np.sin(x)
plt.plot(x, y, 'o', color = 'blue')

(17)
plt.plot(x , y, '-og')

(18)
plt.plot(x, y, '-p', color='gray',
         markersize=15, linewidth=4,
         markerfacecolor='white',
         markeredgecolor='gray',
         markeredgewidth=2)
plt.ylim(-1.2, 1.2);

(19)
x = [1, 2, 3, 4, 5, 6, 7, 8]
#y = [5, 2, 4, 2, 1, 4, 5, 2]
y=[2,4,6,8,10,13,15,16]
plt.scatter(x, y, label = 'cat', color = 'k', s=130 , marker = "^")  #
s is marker size
plt.xlabel("X")
plt.ylabel("Y")
plt.title("Scatter Plot")
plt.legend()
plt.show()

(20)Pie Chart/Pie Plot
import matplotlib.pyplot as plt, numpy
plt.figure(figsize=(3,3))
x=(40,20,5)
labels=['Bikes', 'Cars', 'Buses']    #what is the % of each vehicle in
the whole
plt.pie(x, labels = labels)   #plot x with labels given
plt.show()

(21)
from matplotlib import pyplot as plt
import numpy as np
plt.figure(figsize=(4,4))
langs = ['C', 'C++', 'Java', 'Python', 'PHP']
students = [23,17,35,29,12]
cols = ['c', 'm', 'r', 'b', 'g']
#plt.pie(students, labels = langs, colors = cols, explode = (0, 0.1,
0, 0,0), autopct='%1.2f%%')
plt.pie(students , labels = langs, colors = cols, startangle = 90,
```

```
shadow = False, explode = (0, 0.1, 0, 0,0), autopct = '%1.1f%%')
plt.show()

BoxPlot(22)
#Generating two random normal variates
#We use the numpy.random.normal() function to create the fake data.
#arguments are mean and standard deviation of the normal distribution,
and the number of values desired.
np.random.seed(10)
data1 = np.random.normal(100, 10, 200)
data2 = np.random.normal(80, 30, 200)
data3 = np.random.normal(100, 20, 200)
# Create the boxplot
data=[data1, data2, data3]
plt.boxplot(data)
plt.show()
#Observe the mean and standard deviation in the graph

#Error Bar(23)
#Error Bar added to line plot
plt.style.use('seaborn-whitegrid')
x = np.arange(10)/10
y = (x + 0.1)**2     # instead of x2 we have x+0.1 to power 2
# defing our error
y_error = np.linspace(0.05, 0.2, 10)    #error is between 0.05 and 0.2
in the square

# ploting our function and  error bar
#plt.plot(x, y)
plt.errorbar(x, y, yerr = y_error, fmt ='or')
plt. xlabel('x-axis')
plt.ylabel('y-axis')
plt.title('Line plot with error bars')

(24)
# Error can be added to the x-value too then the error bars are
horizontal
x = np.arange(10)/10
x_error=0.1
y = (x)**2     # instead of x2 we have x+0.1 to power 2
# defing our error
#y_error = np.linspace(0.05, 0.2, 10)    #error is between 0.05 and
0.2 in the square
# ploting our function and  error bar
plt.plot(x, y)
plt.errorbar(x, y, xerr = x_error, fmt ='or')
plt. xlabel('x-axis')
plt.ylabel('y-axis')
plt.title('Line plot with error bars')
```