In [ ]:
```
#Agenda of the Today :
                        1. Continue to Numpy Package
                        2. Pandas Package
                        3. Matplotlib Package
```

In [1]:
```
#How to generate numpy arrays using random module:
#syntax:
import numpy as np
np.random.rand()--- Its generates numbers the uniformly b/w 0 and 1
np.random.randn()---its generate normally distributed b/w 0 and 1
np.random.ranint()---Its generates uniformly distributed numbers b/w 0 and
                        given number.
```

In [20]:
```
#Example:  (package-module-methods/functions)
import numpy as np
print(np.random.rand())
print(np.random.rand(3,3))
print(np.random.rand() +100 * 50)
print(np.random.randn())
print(np.random.randn(3,2))
```

```
0.1526715606130259
[[0.7604665  0.42460856 0.69226305]
 [0.39536926 0.78666812 0.87718008]
 [0.93236702 0.27132827 0.84509864]]
5000.3362883818
0.43682736586836035
[[-0.09352949 -1.14685522]
 [-1.15590282 -0.6575425 ]
 [ 0.27235055  0.53217814]]
```

In [32]:
```
print(np.random.randint(30))
print(np.random.randint(50000000,size = (4,5)))
```

```
24
[[12152729 10604912 27416602  3522912 27771544]
 [13222447 13661838 35895139 18054563  7027272]
 [ 4003537 17122977 48814610 24205310 41544496]
 [20335826  9225542  2502934 35804158 39793786]]
```

In [39]:
```
#Numpy operations on numpy arrays: (indexing ,slicing)
import numpy as np
a1=np.array([1,23,50,90,"surya","python"])   #single-dimension array
print(a1[0])
print(a1[4])
print(a1[5])
print(a1[0:5])
```

```
1
surya
python
['1' '23' '50' '90' 'surya']
```

In [51]:
```
#Indexing on multi-dimesional array
a1 = np.array([[50,60,"surya"],[100,500,"python"],[90.3,600,"Data Science"]])
```

```
print(a1[1][2])
print(a1[1,2])
print(a1[2,1])
print(a1[:,2])
print(a1[0:3,2])
```

```
python
python
600
['surya' 'python' 'Data Science']
['surya' 'python' 'Data Science']
```

In [64]:

```
#Stacking on Numpy arrays (hstacking and vstacking)
import numpy as np
a1 = np.array([[1,2,3],[4,5,6]])
a2 = np.array([[7,8,9],[10,11,12]])
print(a1)
print(a2)
print(a1.shape,a2.shape)
a3=np.hstack((a1,a2))
print(a3)
a4=np.vstack((a2,a1))
print(a4)
```

```
[[1 2 3]
 [4 5 6]]
[[ 7  8  9]
 [10 11 12]]
(2, 3) (2, 3)
[[ 1  2  3  7  8  9]
 [ 4  5  6 10 11 12]]
[[ 7  8  9]
 [10 11 12]
 [ 1  2  3]
 [ 4  5  6]]
```

In [67]:

```
#matrix Transpose
import numpy as np
a1 = np.array([[1,2,3],[4,5,6]])
print("Original array:",a1)
aT = a1.transpose()
print("Transpose array:",aT)
```

```
Original array: [[1 2 3]
 [4 5 6]]
Transpose array: [[1 4]
 [2 5]
 [3 6]]
```

In [ ]:

```
#Pandas in Python:
   --- Its an open source library in python that allows to perform
      - (i) Data manipulation
        (ii) Data analysis
        (iii) Data Cleaning
#Why we use pandas?
     Data Scientities use pandas for
      1. easily handling missing data
      2. flexible way to slice the data.
      3. Its provides the times series tool for work with time series data.
```

In [69]:
```
#How to install pandas?
#pip install pandas
```

In [70]:
```
#How to import pandas ?
import pandas as pd
```

In [ ]:
```
#How to use pandas?
 - in pandas package we have 2 powerfull data structures
     Those are
         (i) Data Series (1d array, Its can store any type of values.)
          (ii) Data Frames (2d array,its in the form of rows and columns.)
```

In [79]:
```
#Data series: (pd.Series(data,index))
import pandas as pd
s1=pd.Series([1,2,3.5,"python",True,False],index=["A","B","C","D","E","F"])
print(s1)
```

```
A         1
B         2
C       3.5
D    python
E      True
F     False
dtype: object
```

In [90]:
```
#Data Frames:(syntax: pd.DataFrame(data,index))
#(2d array,its in the form of rows and columns.)
df=pd.DataFrame([[1,3],[5,6]])
#print(df)
#Dataframe creation using Dictionary input data:
d1 = {"Name":["surya","apssdc","python"],"values":[500,600,900]}
df=pd.DataFrame(data=d1,index=["1","2","3"])
df
```

Out[90]:

	Name	values
1	surya	500
2	apssdc	600
3	python	900

In [92]:
```
#Inspecting the data:
print(df.head(1))    #to view the data from dataframe (in top to bottom)
print(df.tail(1))  #to view th data from data frame(bottom to top)
```

Out[92]:

	Name	values
3	python	900

In [95]:
```
#How to read datasets(.csv) files:
import pandas as pd
```

```
df=pd.read_csv("Employee_Data.csv")  #50000 rows X 37 Columns
```

In [101…

```
#Inspecting the data: (head() and tail())
#by defaultly head() methods gives first 5 rows only.
df.head(5000)
```

Out[101…

	Emp ID	Name Prefix	First Name	Middle Initial	Last Name	Gender	E Mail	Father's Name	Mot N
0	158108	Mr.	Bernard	N	Weatherly	M	bernard.weatherly@gmail.com	Wilbur Weatherly	Weat
1	863655	Ms.	Cordia	M	Knopp	F	cordia.knopp@hotmail.com	Wilson Knopp	D K
2	343966	Mr.	Burton	C	Jin	M	burton.jin@yahoo.com	Richard Jin	Wi
3	389958	Mr.	Lauren	O	Guzzi	M	lauren.guzzi@yahoo.com	Blair Guzzi	Sh
4	291208	Prof.	Carter	C	Hunt	M	carter.hunt@hotmail.com	Jamal Hunt	
…	…	…	…	…	…	…	…	…	
4995	569666	Hon.	Erick	L	Geil	M	erick.geil@gmail.com	Leopoldo Geil	Karar
4996	550695	Prof.	Vince	X	Dimick	M	vince.dimick@gmail.com	Doyle Dimick	Ken D
4997	128820	Ms.	Denna	I	Arbour	F	denna.arbour@aol.com	Eldon Arbour	N Ar
4998	553029	Ms.	Deloras	X	Sonntag	F	deloras.sonntag@gmail.com	Grover Sonntag	Ca Sor
4999	637987	Ms.	Laraine	E	Bixler	F	laraine.bixler@hotmail.com	Keenan Bixler	

5000 rows × 37 columns

In [102…

```
#to view the data from bottom to top
df.tail(10)
```

Out[102…

	Emp ID	Name Prefix	First Name	Middle Initial	Last Name	Gender	E Mail	Father's Name
49990	963209	Mrs.	Billy	I	Phillip	F	billy.phillip@gmail.com	Lupe Phillip
49991	931753	Ms.	Grayce	J	Vella	F	grayce.vella@gmail.com	Ernie Vella
49992	578366	Mrs.	Nicolasa	I	Hearns	F	nicolasa.hearns@hotmail.com	Tommy Hearns
49993	481787	Ms.	Estella	N	Gelinas	F	estella.gelinas@gmail.com	Terence Gelinas
49994	186184	Mr.	Donald	J	Hunsberger	M	donald.hunsberger@aol.com	Milford Hunsberger
49995	773461	Hon.	Felton	J	Hohl	M	felton.hohl@yahoo.com	Gustavo Hohl
49996	743563	Ms.	Launa	N	Staudt	F	launa.staudt@aol.com	Dee Staudt
49997	248566	Ms.	Caryl	S	Ezzell	F	caryl.ezzell@aol.com	Maxwell Ezzell
49998	320960	Mr.	Xavier	U	Deberry	M	xavier.deberry@hotmail.com	Alfred Deberry
49999	508239	Mrs.	Blanch	I	Tarleton	F	blanch.tarleton@ntlworld.com	Landon Tarleton

10 rows × 37 columns

In [103…

```
#describe(to perform counting,mean,std,min,max and percentice of dataset)
df.describe()
```

Out[103…

	Emp ID	Age in Yrs.	Weight in Kgs.	Year of Joining	Month of Joining	Day of Joining	Age Compa (Yea
count	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	50000.0000
mean	555457.763620	40.561104	59.982920	2007.306600	6.480380	15.736820	9.7754
std	255906.633279	11.272835	13.711586	8.602125	3.415472	8.780311	8.5969
min	111112.000000	21.000000	40.000000	1978.000000	1.000000	1.000000	0.0000

	Emp ID	Age in Yrs.	Weight in Kgs.	Year of Joining	Month of Joining	Day of Joining	Age Compa (Yea
25%	334764.250000	30.790000	50.000000	2002.000000	4.000000	8.000000	2.6800
50%	554324.500000	40.560000	57.000000	2010.000000	6.000000	16.000000	7.3200
75%	776663.250000	50.340000	70.000000	2014.000000	9.000000	23.000000	14.9400
max	999982.000000	60.000000	90.000000	2017.000000	12.000000	31.000000	38.7000

In [109…

```
#slicng on Datasets:
df[["Emp ID","Salary"]]  #by calling the columns names
df[50:100]     #By calling the middle index values.
df[5000:10000]
```

Out[109…

	Emp ID	Name Prefix	First Name	Middle Initial	Last Name	Gender	E Mail	Father's Name	M
5000	699247	Mr.	Jarod	I	Hooten	M	jarod.hooten@aol.com	Faustino Hooten	
5001	622619	Mr.	Damian	J	Popp	M	damian.popp@gmail.com	Donny Popp	
5002	919938	Mr.	Broderick	H	Leeds	M	broderick.leeds@msn.com	Deangelo Leeds	
5003	436781	Hon.	Nickole	Y	Bonham	F	nickole.bonham@btinternet.com	Morton Bonham	E
5004	604298	Mr.	Jackson	U	Hudak	M	jackson.hudak@gmail.com	Robin Hudak	
...	...	...	...	...	...	...	...	...	
9995	781202	Mr.	Ezekiel	H	Hafner	M	ezekiel.hafner@gmail.com	Gil Hafner	
9996	147423	Ms.	Shayla	X	Clemente	F	shayla.clemente@yahoo.co.in	Basil Clemente	Cl
9997	200182	Mrs.	Rutha	F	Dominick	F	rutha.dominick@gmail.com	Leopoldo Dominick	D
9998	643195	Ms.	Kendall	F	Brewster	F	kendall.brewster@hotmail.com	Ernesto Brewster	E

	Emp ID	Name Prefix	First Name	Middle Initial	Last Name	Gender	E Mail	Father's Name	M
9999	788061	Mr.	Matt	G	Branco	M	matt.branco@comcast.net	Chauncey Branco	

5000 rows × 37 columns

In [114…

```
#Drop the column: pd.drop()
df.drop(columns=["Emp ID","Name Prefix","First Name"])
#sort the values:
df.sort_values("Salary")
df.sort_values("Emp ID")
```

Out[114…

	Emp ID	Name Prefix	First Name	Middle Initial	Last Name	Gender	E Mail	Father's Name
35753	111112	Drs.	Arlean	E	Barley	F	arlean.barley@bp.com	Howard Barley
11161	111142	Mr.	Dirk	B	Dalzell	M	dirk.dalzell@gmail.com	Melvin Dalzell
48981	111154	Mr.	Darin	N	Keenum	M	darin.keenum@hotmail.com	Bruce Keenum
25213	111189	Mr.	Hosea	I	Broom	M	hosea.broom@gmail.com	Evan Broom
7324	111243	Ms.	Roselia	V	Carrasquillo	F	roselia.carrasquillo@aol.com	Olen Carrasquillo
...	...	...	...	...	...	...	...	...
21652	999888	Mr.	Hayden	O	Chafin	M	hayden.chafin@verizon.net	Chuck Chafin
18218	999916	Ms.	Rochelle	M	Tinker	F	rochelle.tinker@aol.com	Byron Tinker
28620	999918	Mr.	Aubrey	Z	Pavon	M	aubrey.pavon@shaw.ca	Paul Pavon
37823	999954	Ms.	Erminia	F	Viers	F	erminia.viers@verizon.net	Felton Viers

	Emp ID	Name Prefix	First Name	Middle Initial	Last Name	Gender	E Mail	Father's Name
24770	999982	Drs.	Melisa	J	Rademacher	F	melisa.rademacher@shell.com	Marco Rademacher

50000 rows × 37 columns

In [116…

```
#rename: change of index name:
df.rename(columns={"Emp ID":"My ID","Salary":"Credit","E Mail":"G-Mail"})
```

Out[116…

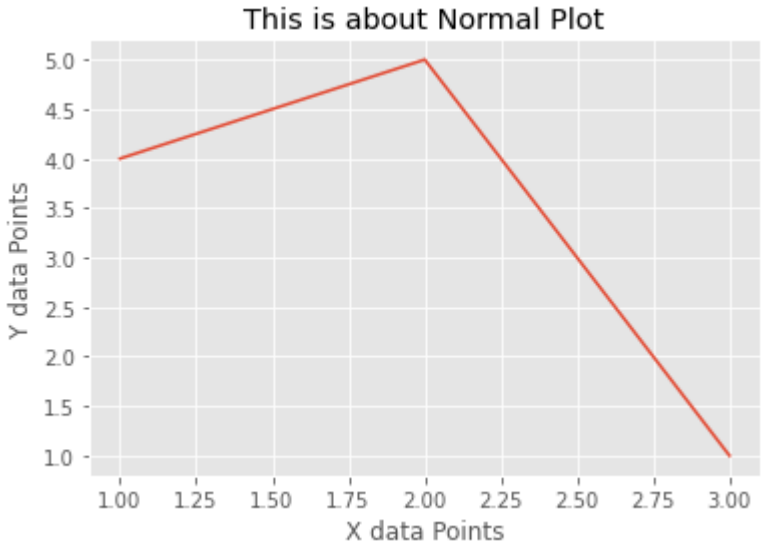
	My ID	Name Prefix	First Name	Middle Initial	Last Name	Gender	G-Mail	Father's Name	Mc
0	158108	Mr.	Bernard	N	Weatherly	M	bernard.weatherly@gmail.com	Wilbur Weatherly	We
1	863655	Ms.	Cordia	M	Knopp	F	cordia.knopp@hotmail.com	Wilson Knopp	[
2	343966	Mr.	Burton	C	Jin	M	burton.jin@yahoo.com	Richard Jin	V
3	389958	Mr.	Lauren	O	Guzzi	M	lauren.guzzi@yahoo.com	Blair Guzzi	S
4	291208	Prof.	Carter	C	Hunt	M	carter.hunt@hotmail.com	Jamal Hunt	
…	…	…	…	…	…	…	…	…	
49995	773461	Hon.	Felton	J	Hohl	M	felton.hohl@yahoo.com	Gustavo Hohl	Eri
49996	743563	Ms.	Launa	N	Staudt	F	launa.staudt@aol.com	Dee Staudt	L
49997	248566	Ms.	Caryl	S	Ezzell	F	caryl.ezzell@aol.com	Maxwell Ezzell	(
49998	320960	Mr.	Xavier	U	Deberry	M	xavier.deberry@hotmail.com	Alfred Deberry	Jea D
49999	508239	Mrs.	Blanch	I	Tarleton	F	blanch.tarleton@ntlworld.com	Landon Tarleton	Ros Ta

50000 rows × 37 columns

In [ ]:
```
#Matplotlib in Python: (plotting library)
 Its used for 2d graphics in python programming to represent the complex data.
```

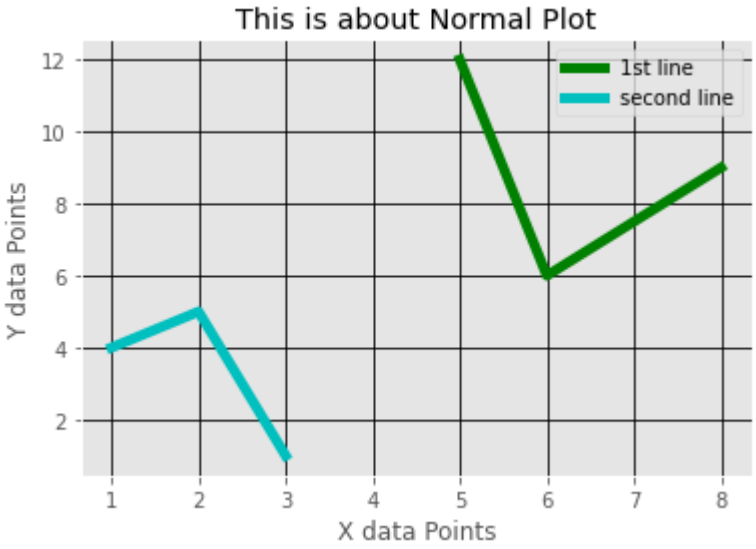
In [ ]:
```
#types of plots:
1. plots
2. pie charts
3. Bar graphs
4. area graphs
5. scatter plots
6. Histograms
```

In [121…
```
#How to generate normal plot:
from matplotlib import pyplot as plt
from matplotlib import style
style.use("ggplot")
x1 = [1,2,3]
y1 = [4,5,1]
plt.plot(x1,y1)
plt.title("This is about Normal Plot")
plt.xlabel("X data Points")
plt.ylabel("Y data Points")

#to view plot
plt.show()
```



In [122…
```
from matplotlib import pyplot as plt
from matplotlib import style
style.use("ggplot")
x = [5,6,8]
y = [12,6,9]
x1 = [1,2,3]
y1 = [4,5,1]
plt.plot(x,y,"g",label="1st line",linewidth=5)
plt.plot(x1,y1,"c",label="second line",linewidth=5)
```

```
plt.title("This is about Normal Plot")
plt.xlabel("X data Points")
plt.ylabel("Y data Points")
plt.legend()
plt.grid(True,color="k")

#to view plot
plt.show()
```



In [ ]:

```
#I will post remaining plots in Github
```

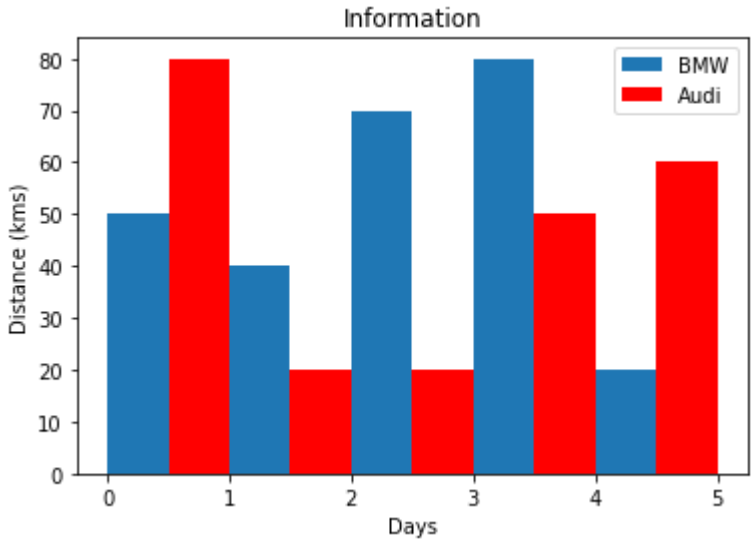
In [ ]:

```
#Python Matplotlib: Bar Graph
First, let us understand why do we need a bar graph. A bar graph uses bars to compare d
It is well suited when you want to measure the changes over a period of time. It can be
```

In [1]:

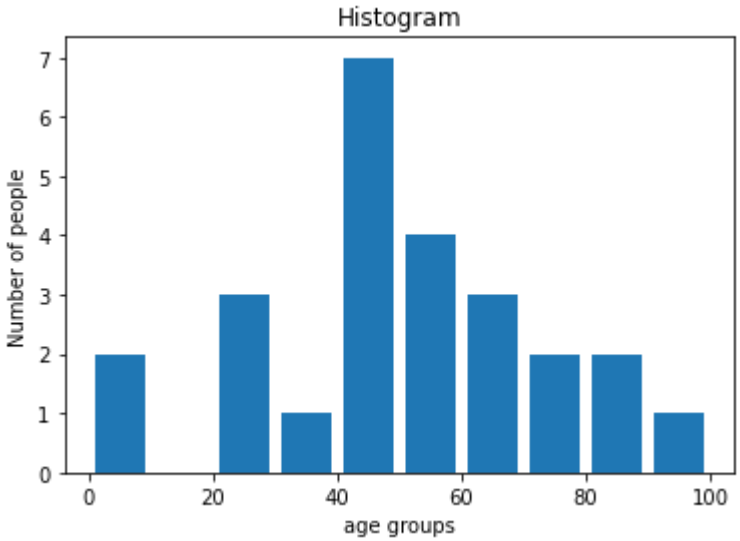
```
#Example: Bar Graph
from matplotlib import pyplot as plt

plt.bar([0.25,1.25,2.25,3.25,4.25],[50,40,70,80,20],
label="BMW",width=.5)
plt.bar([.75,1.75,2.75,3.75,4.75],[80,20,20,50,60],
label="Audi", color='r',width=.5)
plt.legend()
plt.xlabel('Days')
plt.ylabel('Distance (kms)')
plt.title('Information')
plt.show()
```

```
Matplotlib is building the font cache; this may take a moment.
```



In [ ]:
```
#Python Matplotlib – Histogram
Let me first tell you the difference between a bar graph and a histogram.
Histograms are used to show a distribution whereas a bar chart is used to compare diffe
Histograms are useful when you have arrays or a very long list.
```

In [2]:
```
#Example: Histogram
import matplotlib.pyplot as plt
population_age = [22,55,62,45,21,22,34,42,42,4,2,102,95,85,55,110,120,70,65,55,111,115,
bins = [0,10,20,30,40,50,60,70,80,90,100]
plt.hist(population_age, bins, histtype='bar', rwidth=0.8)
plt.xlabel('age groups')
plt.ylabel('Number of people')
plt.title('Histogram')
plt.show()
```

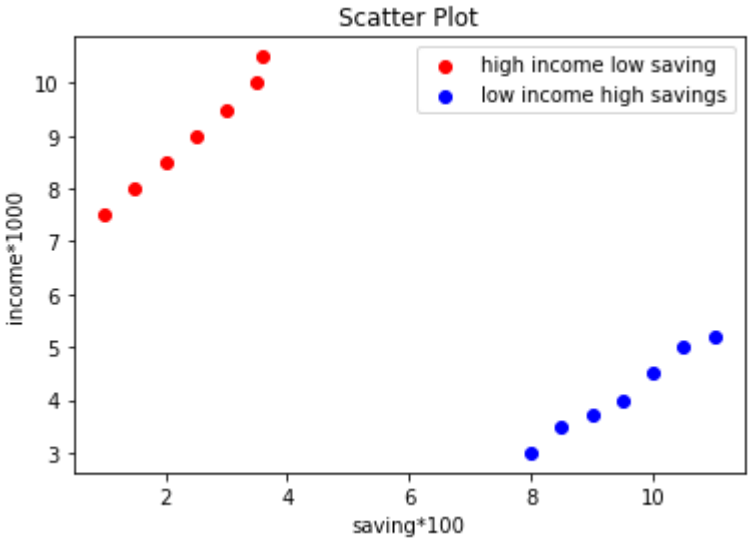


In [ ]:
```
#Python Matplotlib : Scatter Plot
Usually we need scatter plots in order to compare variables,
for example,
how much one variable is affected by another variable to build a relation out of it.
The data is displayed as a collection of points, each having the value of one variable
```

which determines the position on the horizontal axis and the value of other variable de
the position on the vertical axis.

In [1]:
```
#Example: Sactter Plot
import matplotlib.pyplot as plt
x = [1,1.5,2,2.5,3,3.5,3.6]
y = [7.5,8,8.5,9,9.5,10,10.5]

x1=[8,8.5,9,9.5,10,10.5,11]
y1=[3,3.5,3.7,4,4.5,5,5.2]

plt.scatter(x,y, label='high income low saving',color='r')
plt.scatter(x1,y1,label='low income high savings',color='b')
plt.xlabel('saving*100')
plt.ylabel('income*1000')
plt.title('Scatter Plot')
plt.legend()
plt.show()
```

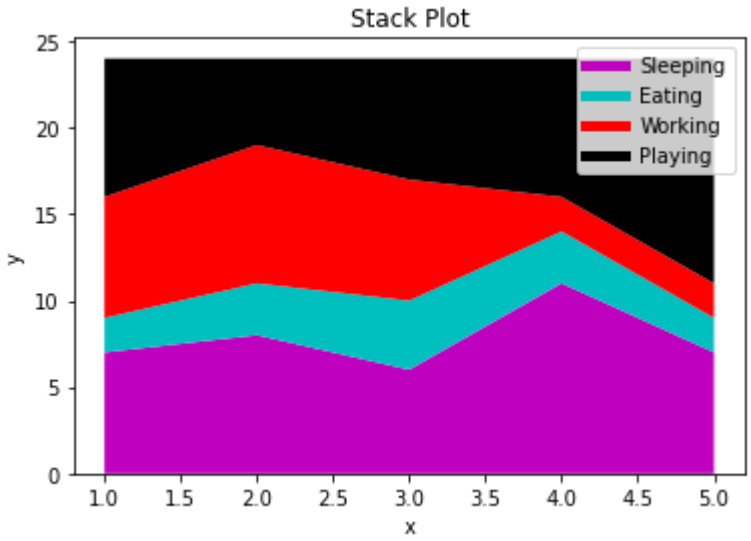
```
Matplotlib is building the font cache; this may take a moment.
```



In [ ]:
```
#Python Matplotlib : Area Plot
Area plots are pretty much similar to the line plot. They are also known as stack plots
These plots can be used to track changes over time for two or more related groups that
```

In [2]:
```
#Example: Area Plot
import matplotlib.pyplot as plt
days = [1,2,3,4,5]

sleeping =[7,8,6,11,7]
eating = [2,3,4,3,2]
working =[7,8,7,2,2]
playing = [8,5,7,8,13]

plt.plot([],[],color='m', label='Sleeping', linewidth=5)
plt.plot([],[],color='c', label='Eating', linewidth=5)
plt.plot([],[],color='r', label='Working', linewidth=5)
plt.plot([],[],color='k', label='Playing', linewidth=5)
```

```
plt.stackplot(days, sleeping,eating,working,playing, colors=['m','c','r','k'])

plt.xlabel('x')
plt.ylabel('y')
plt.title('Stack Plot')
plt.legend()
plt.show()
```



In [ ]:

```
#Python Matplotlib : Pie Chart
A pie chart refers to a circular graph which is broken down into segments i.e. slices o
It is basically used to show the percentage or proportional data where each slice of pi
```

In [3]:

```
#Example : Pie Chart
import matplotlib.pyplot as plt

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

sleeping =[7,8,6,11,7]
eating = [2,3,4,3,2]
working =[7,8,7,2,2]
playing = [8,5,7,8,13]
slices = [7,2,2,13]
activities = ['sleeping','eating','working','playing']
cols = ['c','m','r','b']

plt.pie(slices,
  labels=activities,
  colors=cols,
  startangle=90,
  shadow= True,
  explode=(0,0.1,0,0),
  autopct='%1.1f%%')

plt.title('Pie Plot')
plt.show()
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: