```
#Basics of the data declaration
```

```
import numpy as np
```

```
a=np.array([[1,2],[3,4]])
print(a)
```

```
[[1 2]
 [3 4]]
```

```
#minimum number of dimensions as 2
a=np.array([1,2,3,4,5],ndmin=3)
print(a)
```

```
[[[1 2 3 4 5]]]
```

```
#numbers as complex numbers: a+bj
np.array([1,2,3,4,5],dtype=complex)
```

```
array([1.+0.j, 2.+0.j, 3.+0.j, 4.+0.j, 5.+0.j])
```

```
#pandas library: pandas series
import numpy as np
import pandas as pd

data=np.array(['a','b','c','d'])
```

```
data
```

```
array(['a', 'b', 'c', 'd'], dtype='<U1')
```

```
pd.Series(data)
```

```
0    a
1    b
2    c
3    d
dtype: object
```

```
data={'Name':['rahul','virat','sachin','ishan'],'scores':[100,85,58,41]}
pd.DataFrame(data)
```

	Name	scores
0	rahul	100
1	virat	85
2	sachin	58
3	ishan	41

```
pd.DataFrame(data,index=['rank 1','rank 2','rank 3','rank 4'])
```

	Name	scores
rank 1	rahul	100
rank 2	virat	85
rank 3	sachin	58
rank 4	ishan	41

Data Cleansing

```
import random


print("Random integers between 0 and 9: ")
for i in range(8):
    y = random.randrange(9)
    print(y)
```

```
Random integers between 0 and 9:
6
5
1
1
3
0
2
1
```

```
#declaration of random numbers
np.random.randn(5,3)
```

```
array([[-0.63337875,  0.33467276,  0.03021479],
       [-0.86249647,  0.4091779 , -1.09622842],
       [ 0.67839501, -0.44646729, -1.24219396],
       [-0.44927079, -0.42717574,  1.9001828 ],
       [-0.5342216 , -1.84751053, -1.49331562]])
```

```
data=pd.DataFrame(np.random.randn(5,3),index=['a','c','d','f','h'],columns=['one','two','three'])
```

```
data
```

	one	two	three
a	0.415133	-0.522397	1.134052
c	0.612289	0.021486	0.628577
d	0.562665	-0.730986	0.710595
f	-0.179665	-0.522519	-0.295811
h	1.416361	-0.817667	-0.159872

```
#create empty cells for missing values
```

```
data=data.reindex(['a','b','c','d','e','f','g','h'])
```

```
data
```

	one	two	three
a	0.415133	-0.522397	1.134052
b	NaN	NaN	NaN
c	0.612289	0.021486	0.628577
d	0.562665	-0.730986	0.710595
e	NaN	NaN	NaN
f	-0.179665	-0.522519	-0.295811
g	NaN	NaN	NaN
h	1.416361	-0.817667	-0.159872

```
data['one'].isnull()
```

```
a    False
b     True
c    False
d    False
e     True
f    False
g     True
h    False
Name: one, dtype: bool
```

```
#NaN: not a number
#fill the NaN field
data.fillna(2)
```

	one	two	three
a	-0.867066	-1.042043	1.227564
b	2.000000	2.000000	2.000000
c	0.054259	1.214242	-1.332039
d	0.034982	-0.047473	-0.903696
e	2.000000	2.000000	2.000000
f	-0.020372	0.085187	-0.252121
g	2.000000	2.000000	2.000000
h	-0.913642	0.704379	-1.295217

```
data.fillna(data.mean())
```

```
data.fillna(data.mean())
```

	one	two	three
a	-0.867066	-1.042043	1.227564
b	-0.342368	0.182858	-0.511102
c	0.054259	1.214242	-1.332039
d	0.034982	-0.047473	-0.903696
e	-0.342368	0.182858	-0.511102
f	-0.020372	0.085187	-0.252121
g	-0.342368	0.182858	-0.511102
h	-0.913642	0.704379	-1.295217

```
data
```

	one	two	three
a	-0.867066	-1.042043	1.227564
b	NaN	NaN	NaN
c	0.054259	1.214242	-1.332039
d	0.034982	-0.047473	-0.903696
e	NaN	NaN	NaN
f	-0.020372	0.085187	-0.252121
g	NaN	NaN	NaN
h	-0.913642	0.704379	-1.295217

```
data.fillna(method='pad') #padding the NaN value with reference to the previous row
```

	one	two	three
a	-0.867066	-1.042043	1.227564
b	-0.867066	-1.042043	1.227564
c	0.054259	1.214242	-1.332039
d	0.034982	-0.047473	-0.903696
e	0.034982	-0.047473	-0.903696
f	-0.020372	0.085187	-0.252121
g	-0.020372	0.085187	-0.252121
h	-0.913642	0.704379	-1.295217

```
data.fillna(method='bfill') #padding the NaN value with reference to the next row
```

	one	two	three
a	-0.867066	-1.042043	1.227564
b	0.054259	1.214242	-1.332039
c	0.054259	1.214242	-1.332039
d	0.034982	-0.047473	-0.903696
e	-0.020372	0.085187	-0.252121
f	-0.020372	0.085187	-0.252121
g	-0.913642	0.704379	-1.295217
h	-0.913642	0.704379	-1.295217

```
data
```

	one	two	three
a	-0.867066	-1.042043	1.227564
b	NaN	NaN	NaN
c	0.054259	1.214242	-1.332039
d	0.034982	-0.047473	-0.903696
e	NaN	NaN	NaN
f	-0.020372	0.085187	-0.252121
g	NaN	NaN	NaN
h	-0.913642	0.704379	-1.295217

```
data.dropna()
```

	one	two	three
a	-0.867066	-1.042043	1.227564
c	0.054259	1.214242	-1.332039
d	0.034982	-0.047473	-0.903696
f	-0.020372	0.085187	-0.252121
h	-0.913642	0.704379	-1.295217

```
data.describe()  #description of the data
```

	one	two	three
count	5.000000	5.000000	5.000000
mean	-0.342368	0.182858	-0.511102

Visualization of the data

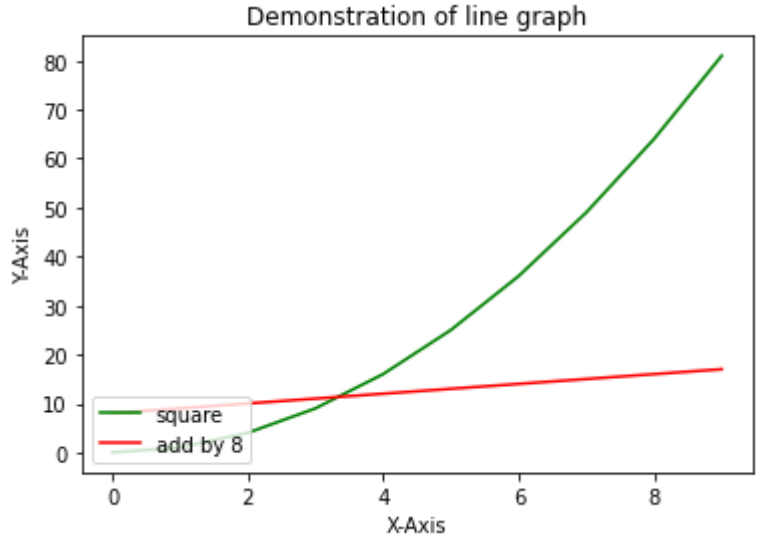
```
#Line graph
```

50%	-0.020372	0.085187	-0.903696

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

```
#line graph showing characteristic of square of a number. X vs Y: X vs X**2
x=np.arange(0,10)
y=x**2
#print(x,y)
plt.plot(x,y,'g')
z=x+8
plt.plot(x,z,'r')
plt.title('Demonstration of line graph')
plt.xlabel('X-Axis')
plt.ylabel('Y-Axis')
plt.legend(['square','add by 8'],loc=3)
```

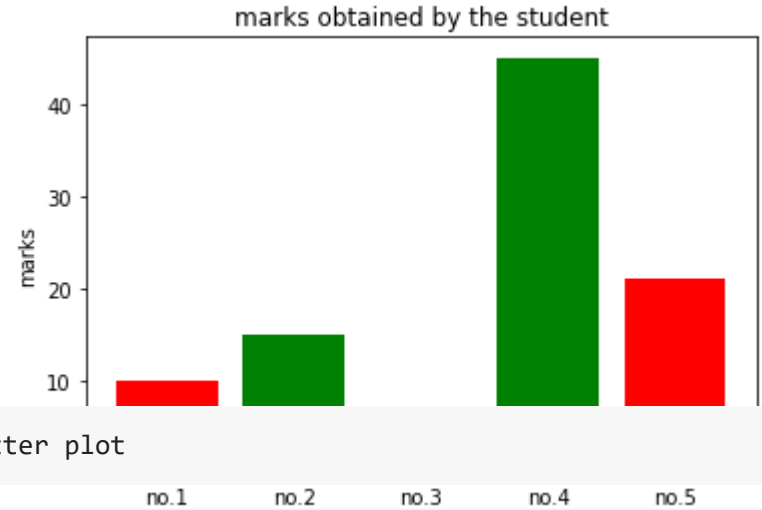
```
<matplotlib.legend.Legend at 0x7f6d04b65150>
```




```
#bar chart
```

```
x=[1,2,3,4,5]
marks=[10,15,7,45,21]
labels=['no.1','no.2','no.3','no.4','no.5']
plt.bar(x,marks,width=0.8,tick_label=labels,color=['red','green'])
plt.xlabel('Roll Number')
plt.ylabel('marks')
plt.title('marks obtained by the student')
```

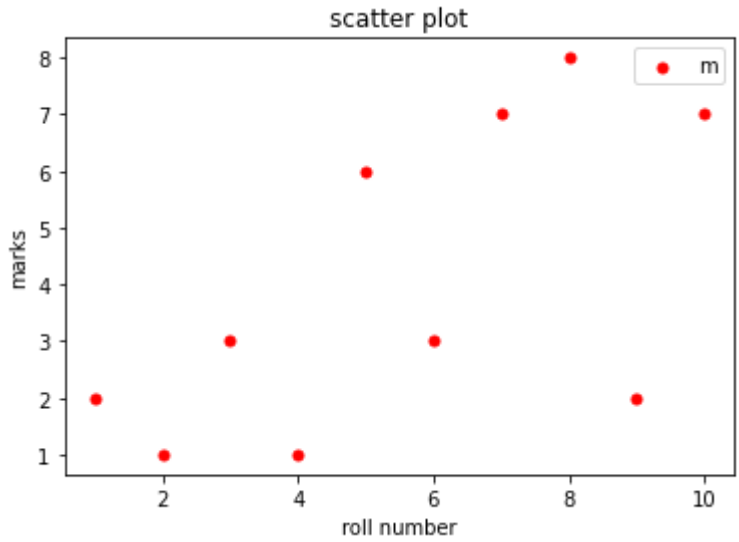
```
Text(0.5, 1.0, 'marks obtained by the student')
```




```
#scatter plot
```

```
x=[1,2,3,4,5,6,7,8,9,10]
y=[2,1,3,1,6,3,7,8,2,7]
plt.scatter(x,y,marker='o',color='red',s=25)  #marker="*"
plt.xlabel('roll number')
plt.ylabel('marks')
plt.title('scatter plot')
plt.legend('marks')
```

```
<matplotlib.legend.Legend at 0x7f6d041d4110>
```




```
#pie chart
```

```
activities=['work','eat','play','sleep','gossip']
hours=[8,2,4,8,2]
plt.pie(hours,labels=activities,radius=1.5,explode=(0,0,0.1,0,0))
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f6d0454d750>
```