```
In [2]: import pandas as pd
```

```
In [3]: marks = [80,90,100,40]
        labels = ["maths","science","physics","chem"]
```

```
In [4]: ser = pd.Series(marks,labels)
```

```
In [5]: ser
```

```
Out[5]: maths       80
        science     90
        physics    100
        chem        40
        dtype: int64
```

```
In [6]: ser = pd.Series(data = marks,index = labels)
```

```
In [7]: ser
```

```
Out[7]: maths       80
        science     90
        physics    100
        chem        40
        dtype: int64
```

```
In [8]: type(ser)
```

```
Out[8]: pandas.core.series.Series
```

```
In [9]: ser['maths']
```

```
Out[9]: 80
```

```
In [10]: ser['maths']
```

```
Out[10]: 80
```

```
In [11]: import numpy as np
         p = np.arange(16).reshape(4,4)
         labels = ['a','b','c','d']
         label2 = ['A','B','C','D']
```

```
In [12]: p
```

```
Out[12]: array([[ 0,  1,  2,  3],
                [ 4,  5,  6,  7],
                [ 8,  9, 10, 11],
                [12, 13, 14, 15]])
```

```
In [13]: df = pd.DataFrame(p,index = labels,columns = label2)
```

```
In [14]: df
```

```
Out[14]:
            A   B   C   D
         a  0   1   2   3
         b  4   5   6   7
         c  8   9  10  11
         d 12  13  14  15
```

```
In [15]: df1 = pd.DataFrame([[1,2,3,4],[4,5,6,7],[8,9,10,11],[12,13,14,15,]], index = "A B C D".split(),columns = "a b c d".split())
```

```
In [16]: "A B C D".split()
```

```
Out[16]: ['A', 'B', 'C', 'D']
```

```
In [17]: "a b c d".split()
```

```
Out[17]: ['a', 'b', 'c', 'd']
```

In [18]:
```
df1
```

Out[18]:

	a	b	c	d
A	1	2	3	4
B	4	5	6	7
C	8	9	10	11
D	12	13	14	15

In [19]:
```
df1[['b','d']]
```

Out[19]:

	b	d
A	2	4
B	5	7
C	9	11
D	13	15

In [20]:
```
df1['a']
```

Out[20]:
```
A     1
B     4
C     8
D    12
Name: a, dtype: int64
```

In [21]:
```
df1.loc['C']
```

Out[21]:
```
a     8
b     9
c    10
d    11
Name: C, dtype: int64
```

In [22]:
```
df1.iloc[0]
```

Out[22]:
```
a    1
b    2
c    3
d    4
Name: A, dtype: int64
```

In [23]:
```
df1['new'] =df1['a']+df1['b']
```

In [24]:
```
df1
```

Out[24]:

	a	b	c	d	new
A	1	2	3	4	3
B	4	5	6	7	9
C	8	9	10	11	17
D	12	13	14	15	25

In [25]:
```
df1['new2'] = [1,2,4,6]
```

In [26]:
```
df1
```

Out[26]:

	a	b	c	d	new	new2
A	1	2	3	4	3	1
B	4	5	6	7	9	2
C	8	9	10	11	17	4
D	12	13	14	15	25	6

```
In [27]: df1.drop('new2',axis =1)
```

```
Out[27]:
      a   b   c   d  new
  A   1   2   3   4    3
  B   4   5   6   7    9
  C   8   9  10  11   17
  D  12  13  14  15   25
```

```
In [28]: df1
```

```
Out[28]:
      a   b   c   d  new  new2
  A   1   2   3   4    3     1
  B   4   5   6   7    9     2
  C   8   9  10  11   17     4
  D  12  13  14  15   25     6
```

```
In [29]: df1.drop('new2',axis = 1,inplace =  True)
```

```
In [30]: df1
```

```
Out[30]:
      a   b   c   d  new
  A   1   2   3   4    3
  B   4   5   6   7    9
  C   8   9  10  11   17
  D  12  13  14  15   25
```

```
df1.drop('D')
```

In [31]:
```
df1
```

Out[31]:

	a	b	c	d	new
A	1	2	3	4	3
B	4	5	6	7	9
C	8	9	10	11	17
D	12	13	14	15	25

```
df1.drop('D',inplace = True)
```

In [32]:
```
df1
```

Out[32]:

	a	b	c	d	new
A	1	2	3	4	3
B	4	5	6	7	9
C	8	9	10	11	17
D	12	13	14	15	25

In [55]:
```
df1.reset_index()
```

Out[55]:

	newindex	a	b	c	d	new
0	CA	1	2	3	4	3
1	NY	4	5	6	7	9
2	WY	8	9	10	11	17
3	DR	12	13	14	15	25

In [41]:
```
newind = "CA NY  WY DR".split()
```

In [42]:
```
newind
```

Out[42]:
```
['CA', 'NY', 'WY', 'DR']
```

In [43]:
```
df1
```

Out[43]:

	a	b	c	d	new
A	1	2	3	4	3
B	4	5	6	7	9
C	8	9	10	11	17
D	12	13	14	15	25

In [44]:
```
df1["newindex"] = newind
```

In [45]:
```
df1
```

Out[45]:

	a	b	c	d	new	newindex
A	1	2	3	4	3	CA
B	4	5	6	7	9	NY
C	8	9	10	11	17	WY
D	12	13	14	15	25	DR

In [49]:
```
df1.set_index("newindex",inplace  = True)
```

In [50]:
```
df1
```

Out[50]:

	a	b	c	d	new
newindex					
CA	1	2	3	4	3
NY	4	5	6	7	9
WY	8	9	10	11	17
DR	12	13	14	15	25

In [52]:
```
df1.head(2) # will return first 5 rows of a dataet
```

Out[52]:

	a	b	c	d	new
newindex					
CA	1	2	3	4	3
NY	4	5	6	7	9

In [54]:
```
df1.tail(2)
```

Out[54]:

	a	b	c	d	new
newindex					
WY	8	9	10	11	17
DR	12	13	14	15	25

In [56]:
```
df1['a'].unique()
```

Out[56]:
```
array([ 1,  4,  8, 12], dtype=int64)
```

In [57]:
```
df1['a'].value_counts()
```

Out[57]:
```
12    1
4     1
1     1
8     1
Name: a, dtype: int64
```

In [58]:
```
df1.describe()
```

Out[58]:

	a	b	c	d	new
count	4.000000	4.000000	4.000000	4.000000	4.000000
mean	6.250000	7.250000	8.250000	9.250000	13.500000
std	4.787136	4.787136	4.787136	4.787136	9.574271
min	1.000000	2.000000	3.000000	4.000000	3.000000
25%	3.250000	4.250000	5.250000	6.250000	7.500000
50%	6.000000	7.000000	8.000000	9.000000	13.000000
75%	9.000000	10.000000	11.000000	12.000000	19.000000
max	12.000000	13.000000	14.000000	15.000000	25.000000

In [59]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 4 entries, CA to DR
Data columns (total 5 columns):
a      4 non-null int64
b      4 non-null int64
c      4 non-null int64
d      4 non-null int64
new    4 non-null int64
dtypes: int64(5)
memory usage: 192.0+ bytes
```

```
1- to  replace the missing values with mean
2- replace the missingvalues with mode
3- replce it with median
4- your own value
```

In [65]:
```
df = pd.DataFrame({'A':[1,2,np.NaN],
                   'B':[5,np.NaN,np.NaN],
                   'C':[1,2,3]})
```

In [66]:
```
df
```

Out[66]:

	A	B	C
0	1.0	5.0	1
1	2.0	NaN	2
2	NaN	NaN	3

In [67]:
```
df.isnull().any
```

Out[67]:
```
<bound method DataFrame.any of        A      B      C
0  False  False  False
1  False   True  False
2   True   True  False>
```

In [68]:
```
df.isnull().any()
```

Out[68]:
```
A     True
B     True
C    False
dtype: bool
```

In [69]:
```
df.isnull().sum()
```

Out[69]:
```
A    1
B    2
C    0
dtype: int64
```

In [70]:
```
df.dropna()
```

Out[70]:

	A	B	C
0	1.0	5.0	1

In [71]:
```
df.dropna(axis = 1)
```

Out[71]:
```
   C
0  1
1  2
2  3
```

In [73]:
```
df.dropna(thresh = 2,axis=1)
```

Out[73]:
```
     A  C
0  1.0  1
1  2.0  2
2  NaN  3
```

In [74]:
```
df.fillna(value= 60)
```

Out[74]:
```
      A     B  C
0   1.0   5.0  1
1   2.0  60.0  2
2  60.0  60.0  3
```

In [112]:
```
df['A'].fillna(df["A"].mean(),inplace = True)
```

In [109]:
```
df['B'].fillna(df["B"].mean(),inplace = True)
```

In [110]:
```
df['B'].fillna(df["B"].median(),inplace = True)
```

In [78]:
```
p = df['A'].mode()
```

```
In [79]: p
```

```
Out[79]: 0    1.0
         1    2.0
         dtype: float64
```

```
In [81]: p[1]
```

```
Out[81]: 2.0
```

```
In [85]: df['A'].fillna(df["A"].mode()[0])
```

```
Out[85]: 0    1.0
         1    2.0
         2    1.0
         Name: A, dtype: float64
```

```
In [99]: df1 = pd.DataFrame(["Aindia" , np.nan, "Aindia","Australia","Australia","us","canada"],columns = ['country'])
```

```
In [100]: df1
```

Out[100]:

	country
0	Aindia
1	NaN
2	Aindia
3	Australia
4	Australia
5	us
6	canada

```
In [101]: q = df1['country'].mode()
```

```
In [103]: q[0]
```

```
Out[103]: 'Aindia'
```

In [106]:

```
df1['country'].fillna(df1['country'].mode()[1])
```

Out[106]:

```
0       Aindia
1    Australia
2       Aindia
3    Australia
4    Australia
5           us
6       canada
Name: country, dtype: object
```

In [113]:

```
df
```

Out[113]:

	A	B	C
0	1.0	5.0	1
1	2.0	5.0	2
2	1.5	5.0	3

```
mean- ideal condition is if colums data is ina range -  age (20,80)
median()- lot of variation salary (1000 to 1000000)
mode  - for textual data
```

```
create a  dataframe with 10 rows anf seven colums two colums should be textual
in that make some missing values in four of the colums one in textual colum
apply all the functions which are taugut
```

In [114]:

```
df2 = pd.DataFrame(np.arange(16).reshape(4,4),,"p q r s".split())
```

In [115]:

```
df2
```

Out[115]:

	p	q	r	s
a	0	1	2	3
b	4	5	6	7
c	8	9	10	11
d	12	13	14	15

```
In [124]: "a b c d".split()

Out[124]: ['a', 'b', 'c', 'd']
```

```
In [ ]: pd.DataFrame(data,index list format,columslist format)
```

```
In [117]: newindex = ["row1","row2","row3","row4"]
```

```
In [118]: df2["newcolumn"] = newindex
```

```
In [119]: df2

Out[119]:
      p   q   r   s  newcolumn
  a   0   1   2   3       row1
  b   4   5   6   7       row2
  c   8   9  10  11       row3
  d  12  13  14  15       row4
```

```
In [122]: df2.set_index("newcolumn",inplace = True)
```

```
In [123]: df2

Out[123]:
             p   q   r   s
 newcolumn
      row1   0   1   2   3
      row2   4   5   6   7
      row3   8   9  10  11
      row4  12  13  14  15
```

```
In [134]: dataset = pd.read_csv(r"D:\pythonbasics\50_Startups.csv")
In [127]: dataset
```

Out[127]:

	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	123456.00	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	12345.00	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94
5	131876.90	99814.71	362861.36	New York	156991.12
6	134615.46	147198.87	127716.82	California	156122.51
7	130298.13	145530.06	323876.68	Florida	155752.60
8	120542.52	148718.95	311613.29	New York	152211.77
9	123334.88	108679.17	304981.62	California	149759.96
10	101913.08	110594.11	229160.95	Florida	146121.95
11	100671.96	91790.61	249744.55	California	144259.40
12	93863.75	127320.38	249839.44	Florida	141585.52
13	91992.39	135495.07	252664.93	California	134307.35
14	119943.24	156547.42	256512.92	Florida	132602.65
15	114523.61	122616.84	261776.23	New York	129917.04
16	78013.11	121597.55	264346.06	California	126992.93
17	94657.16	145077.58	282574.31	New York	125370.37
18	91749.16	114175.79	294919.57	Florida	124266.90
19	86419.70	153514.11	0.00	New York	122776.86
20	76253.86	113867.30	298664.47	California	118474.03
21	78389.47	153773.43	299737.29	New York	111313.02
22	73994.56	122782.75	303319.26	Florida	110352.25
23	67532.53	105751.03	304768.73	Florida	108733.99
24	77044.01	99281.34	140574.81	New York	108552.04
25	64664.71	139553.16	137962.62	California	107404.34

	R&D Spend	Administration	Marketing Spend	State	Profit
26	75328.87	144135.98	134050.07	Florida	105733.54
27	72107.60	127864.55	353183.81	New York	105008.31
28	66051.52	182645.56	118148.20	Florida	103282.38
29	65605.48	153032.06	107138.38	New York	101004.64
30	61994.48	115641.28	91131.24	Florida	99937.59
31	61136.38	152701.92	88218.23	New York	97483.56
32	63408.86	129219.61	46085.25	California	97427.84
33	55493.95	103057.49	214634.81	Florida	96778.92
34	46426.07	157693.92	210797.67	California	96712.80
35	46014.02	85047.44	205517.64	New York	96479.51
36	28663.76	127056.21	201126.82	Florida	90708.19
37	44069.95	51283.14	197029.42	California	89949.14
38	20229.59	65947.93	185265.10	New York	81229.06
39	38558.51	82982.09	174999.30	California	81005.76
40	28754.33	118546.05	172795.67	California	78239.91
41	27892.92	84710.77	164470.71	Florida	77798.83
42	23640.93	96189.63	148001.11	California	71498.49
43	15505.73	127382.30	35534.17	New York	69758.98
44	22177.74	154806.14	28334.72	California	65200.33
45	1000.23	124153.04	1903.93	New York	64926.08
46	1315.46	115816.21	297114.46	Florida	49490.75
47	0.00	135426.92	0.00	California	42559.73
48	542.05	51743.15	0.00	New York	35673.41
49	0.00	116983.80	45173.06	California	14681.40

In [129]:
```
dataset.isnull().sum()
```

Out[129]:
```
R&D Spend          0
Administration     0
Marketing Spend    0
State              0
Profit             0
dtype: int64
```

In [131]:
```
dataset.tail()
```

Out[131]:

	R&D Spend	Administration	Marketing Spend	State	Profit
45	1000.23	124153.04	1903.93	New York	64926.08
46	1315.46	115816.21	297114.46	Florida	49490.75
47	0.00	135426.92	0.00	California	42559.73
48	542.05	51743.15	0.00	New York	35673.41
49	0.00	116983.80	45173.06	California	14681.40

In [132]:
```
dataset['State'].unique()
```

Out[132]:
```
array(['New York', 'California', 'Florida'], dtype=object)
```

In [133]:
```
dataset['State'].value_counts()
```

Out[133]:
```
New York      17
California    17
Florida       16
Name: State, dtype: int64
```

In [136]:

```
data = pd.DataFrame({'1':  ['h', 'e','l','l','o',np.NaN,'n','a','i','t'],
        '2':  ['h', 'e','l','l','o','n','a','i',np.NaN,'t'],
        '3':  ['h', 'e','l','l','o',np.NaN,'n','a','i','t'],
        '4':  ['h', 'e','l','l','o',np.NaN,'n','a','i','t'],
        '5':  ['h', 'e',np.NaN,'l','o','n','a','i','t','k'],
        '6':  ['h',np.NaN,'l','l','o','k','n','a','i','t'],
        '7':  ['h', 'e','l','l','o',np.NaN,'n','a','i','t']})
data
```

Out[136]:

	1	2	3	4	5	6	7
0	h	h	h	h	h	h	h
1	e	e	e	e	e	NaN	e
2	l	l	l	l	NaN	l	l
3	l	l	l	l	l	l	l
4	o	o	o	o	o	o	o
5	NaN	n	NaN	NaN	n	k	NaN
6	n	a	n	n	a	n	n
7	a	i	a	a	i	a	a
8	i	NaN	i	i	t	i	i
9	t	t	t	t	k	t	t

In [140]:

```
data['1'].mode()
```

Out[140]:

```
0    l
dtype: object
```

In [158]:

```
data['1'].fillna(data['1'].mode()[0],inplace = True)
data['2'].fillna(data['2'].mode()[0],inplace = True)
data['3'].fillna(data['3'].mode()[0],inplace = True)
```

In [159]:
```
data
```

Out[159]:

	1	2	3	4	5	6	7
0	h	h	h	h	h	h	h
1	e	e	e	e	e	NaN	e
2	l	l	l	l	NaN	l	l
3	l	l	l	l	l	l	l
4	o	o	o	o	o	o	o
5	l	n	l	NaN	n	k	NaN
6	n	a	n	n	a	n	n
7	a	i	a	a	i	a	a
8	i	l	i	i	t	i	i
9	t	t	t	t	k	t	t

In [143]:
```
d3 = df
```

In [144]:
```
d3
```

Out[144]:

	A	B	C
0	1.0	5.0	1
1	2.0	5.0	2
2	1.5	5.0	3

In [145]:
```
d3 = pd.DataFrame({'A':[1,2,np.NaN],
                   'B':[5,np.NaN,np.NaN],
                   'C':[1,2,3]})
```

```
In [146]: d3
```

```
Out[146]:
        A    B  C
0     1.0  5.0  1
1     2.0  NaN  2
2     NaN  NaN  3
```

```
In [151]: d3['A'].fillna(d3['A'].mean(),inplace = True)
```

```
In [152]: d3['B'].fillna(d3['B'].median(),inplace = True)
```

```
In [153]: d3
```

```
Out[153]:
     A    B  C
0  1.0  5.0  1
1  2.0  5.0  2
2  1.5  5.0  3
```

```
In [168]: pdataset = pd.read_excel("datasetexcel.xlsx")
```

In [169]:
```
pdataset
```

Out[169]:

	maths	roll	tutorname	op	std (mean - value)2
0	40.0	1.0	1	3.000000	NaN
1	3.0	3.0	3	NaN	NaN
2	5.0	NaN	3	55.000000	NaN
3	4.0	4.0	3	5.000000	NaN
4	3.0	3.0	3	31.000000	NaN
5	3.0	3.0	3	35.285714	NaN
6	3.0	3.0	3	39.571429	NaN
7	NaN	NaN	3	43.857143	NaN
8	2.0	2.0	3	48.142857	NaN
9	2.0	2.0	3	52.428571	NaN
10	2.0	2.0	3	56.714286	NaN
11	1.0	1.0	3	61.000000	NaN
12	5.0	5.0	3	65.285714	NaN
13	6.0	NaN	3	69.571429	NaN
14	2.0	2.0	3	73.857143	NaN
15	1.0	1.0	3	78.142857	NaN
16	3.0	NaN	3	82.428571	NaN
17	2.0	2.0	3	86.714286	NaN

In [ ]: