WELCOM TO PANDAS TUTORIAL

In [1]:

```
import numpy as np
import pandas as pd
```

In [2]:

```
dict1 = {
    "name":['hary','rohan','skillf','shubh'],
    "marks" : [15,34,24,25],
    "city":['rampur','kolkata','bareli','nashik']
}
```

In [3]:

```
df = pd.DataFrame(dict1)                    ### makes data frame of any type of data         [DATAFRAME]
df
```

Out[3]:

	name	marks	city
0	hary	15	rampur
1	rohan	34	kolkata
2	skillf	24	bareli
3	shubh	25	nashik

In [4]:

```
df.to_csv('friends.csv')                    ### convert data frame into csv                 [TO_CSV]
```

In [5]:

```
df.to_csv('friends.csv',index=False)        ### INDEX= FALSE : DONT SHOW INDEX NUMBER
```

In [6]:

```
df.index = ['a','b','c','d']                ### changed index name                        [INDEX]
df.head(2)
```

Out[6]:

	name	marks	city
a	hary	15	rampur
b	rohan	34	kolkata

In [7]:
```
df.tail(2)                                   ### show the tail of data                  [TAIL]
```

Out[7]:

	name	marks	city
c	skillf	24	bareli
d	shubh	25	nashik

In [8]:
```
df.describe()                                ### gives all stats of data                [DESCRIBE]
```

Out[8]:

	marks
count	4.000000
mean	24.500000
std	7.767453
min	15.000000
25%	21.750000
50%	24.500000
75%	27.250000
max	34.000000

In [9]:
```
data = pd.read_csv(r"D:\harsh work\data_science\datasets\titanic passenger list.csv")

### loading csv data in pandas       [READ_CSV]
```

In [10]:
```
data.head(2)
```

Out[10]:

	Unnamed: 0	Unnamed: 0.1	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.
0	0	0	1	1	Allen, Miss. Elisabeth Walton	male	29.00	0	0	24160	211.3375	B5	S	2	NaN	St L
1	1	1	1	1	Allison, Master. Hudson Trevor	male	0.92	1	2	113781	151.5500	C22 C26	S	11	NaN	Mon Chester

In [11]:
```
data['sex'][0] = 'male'
### data:your data , ['sex'] : column name , [0] : index , = 'male' : chnage orignal value as male
```

```
<ipython-input-11-828b7f79d737>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#re
turning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-
a-view-versus-a-copy)
  data['sex'][0] = 'male'
```

```
In [12]: data.head(2)                              ### column sex , index 0 : vlaue changed from female to male
```

Out[12]:

	Unnamed: 0	Unnamed: 0.1	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.
0	0	0	1	1	Allen, Miss. Elisabeth Walton	male	29.00	0	0	24160	211.3375	B5	S	2	NaN	St L
1	1	1	1	1	Allison, Master. Hudson Trevor	male	0.92	1	2	113781	151.5500	C22 C26	S	11	NaN	Mon Chester

Deep Understanding Of Pandas

In [13]:
```
ser = pd.Series(np.random.rand(34))
print(ser)
type(ser)                                   ### it gives 0 to 34 serires          [SEIRES]
```

```
0     0.253411
1     0.374433
2     0.004047
3     0.666668
4     0.464212
5     0.898917
6     0.467395
7     0.259257
8     0.846452
9     0.381481
10    0.780828
11    0.421783
12    0.001903
13    0.008908
14    0.028225
15    0.418502
16    0.500866
17    0.895689
18    0.361485
19    0.763013
20    0.023943
21    0.410698
22    0.300423
23    0.680264
24    0.041265
25    0.425172
26    0.997111
27    0.879135
28    0.920318
29    0.692707
30    0.544604
31    0.893604
32    0.445550
33    0.990854
dtype: float64
```

Out[13]:
```
pandas.core.series.Series
```

In [14]:
```
newdf = pd.DataFrame(np.random.rand(334,5),index=np.arange(334))      ### makes any dataframe
newdf.head(5)                                                          ###       [DATAFRAME]
```

Out[14]:

	0	1	2	3	4
0	0.525418	0.984169	0.074613	0.910435	0.904506
1	0.797874	0.151738	0.131649	0.071332	0.701692
2	0.679583	0.626458	0.546089	0.732970	0.516948
3	0.673509	0.001938	0.047301	0.777832	0.707877
4	0.892942	0.087179	0.085141	0.226777	0.371006

In [15]:
```
type(newdf)                                 ### dataframe
```

Out[15]:
```
pandas.core.frame.DataFrame
```

In [16]:
```
newdf.describe()
newdf.dtypes
```

Out[16]:
```
0    float64
1    float64
2    float64
3    float64
4    float64
dtype: object
```

In [17]:
```
newdf.index                                 ### all index of dataframe
```

Out[17]:
```
Int64Index([  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,
            ...
            324, 325, 326, 327, 328, 329, 330, 331, 332, 333],
           dtype='int64', length=334)
```

In [18]:
```
newdf.columns
```

Out[18]:
```
RangeIndex(start=0, stop=5, step=1)
```

In [19]:
```
newdf.to_numpy()                                   ### convert data into array
```

Out[19]:
```
array([[5.25418293e-01, 9.84169359e-01, 7.46126597e-02, 9.10435381e-01,
        9.04506476e-01],
       [7.97873947e-01, 1.51737820e-01, 1.31649098e-01, 7.13322185e-02,
        7.01691919e-01],
       [6.79582513e-01, 6.26457541e-01, 5.46089482e-01, 7.32970294e-01,
        5.16947539e-01],
       ...,
       [1.31519540e-01, 6.31355605e-01, 8.34375455e-01, 8.14823628e-01,
        3.73775929e-01],
       [4.07084132e-01, 8.33584482e-01, 5.96161067e-01, 7.07779900e-04,
        6.23575117e-01],
       [6.90824726e-01, 1.82876296e-01, 5.68372277e-02, 1.94712933e-01,
        5.55203185e-01]])
```

In [20]:
```
newdf[0][0] = 0.3
newdf.head()
```

Out[20]:

	0	1	2	3	4
0	0.300000	0.984169	0.074613	0.910435	0.904506
1	0.797874	0.151738	0.131649	0.071332	0.701692
2	0.679583	0.626458	0.546089	0.732970	0.516948
3	0.673509	0.001938	0.047301	0.777832	0.707877
4	0.892942	0.087179	0.085141	0.226777	0.371006

Atributes of pandas

In [21]:
```
newdf.T                                          ### transforse of data          [T]
```

Out[21]:

	0	1	2	3	4	5	6	7	8	9	...	324	325	326
0	0.300000	0.797874	0.679583	0.673509	0.892942	0.793515	0.014928	0.085228	0.506408	0.372945	...	0.894051	0.053568	0.492877
1	0.984169	0.151738	0.626458	0.001938	0.087179	0.907141	0.894558	0.369409	0.353545	0.332607	...	0.063603	0.496439	0.263600
2	0.074613	0.131649	0.546089	0.047301	0.085141	0.453583	0.980700	0.230816	0.070685	0.754502	...	0.960727	0.782086	0.928174
3	0.910435	0.071332	0.732970	0.777832	0.226777	0.867880	0.393523	0.435636	0.895311	0.281589	...	0.331002	0.748980	0.744856
4	0.904506	0.701692	0.516948	0.707877	0.371006	0.204566	0.631299	0.411256	0.735970	0.673478	...	0.755850	0.951105	0.233996

5 rows × 334 columns

In [22]:
```
newdf.head(5)
```

Out[22]:

	0	1	2	3	4
0	0.300000	0.984169	0.074613	0.910435	0.904506
1	0.797874	0.151738	0.131649	0.071332	0.701692
2	0.679583	0.626458	0.546089	0.732970	0.516948
3	0.673509	0.001938	0.047301	0.777832	0.707877
4	0.892942	0.087179	0.085141	0.226777	0.371006

In [23]:

```
newdf.sort_index(axis=0,ascending=False)          ### revers the indexes          [SORT_INDEX]
newdf.head(5)
```

Out[23]:

	0	1	2	3	4
0	0.300000	0.984169	0.074613	0.910435	0.904506
1	0.797874	0.151738	0.131649	0.071332	0.701692
2	0.679583	0.626458	0.546089	0.732970	0.516948
3	0.673509	0.001938	0.047301	0.777832	0.707877
4	0.892942	0.087179	0.085141	0.226777	0.371006

In [24]:

```
newdf.loc[0][0] = 735                             ### change any value in data          [LOC]
                                                  ### ideal way
```

In [25]:

```
newdf.columns = list("abcde")                     ### change coulmn name          [COLUMN]
newdf.head(2)
```

Out[25]:

	a	b	c	d	e
0	735.000000	0.984169	0.074613	0.910435	0.904506
1	0.797874	0.151738	0.131649	0.071332	0.701692

In [26]:

```
newdf = newdf.drop(['e'],axis=1)                   ### drops any column or row          [DROP]
newdf.head()
```

Out[26]:

	a	b	c	d
0	735.000000	0.984169	0.074613	0.910435
1	0.797874	0.151738	0.131649	0.071332
2	0.679583	0.626458	0.546089	0.732970
3	0.673509	0.001938	0.047301	0.777832
4	0.892942	0.087179	0.085141	0.226777

In [27]:
```
newdf.loc[[1,2],['a','b']]                                   ### show only particular line and row
```

Out[27]:

	a	b
1	0.797874	0.151738
2	0.679583	0.626458

In [28]:
```
newdf.loc[:,['a','b']].head(5)                                ### : gives 'all'
```

Out[28]:

	a	b
0	735.000000	0.984169
1	0.797874	0.151738
2	0.679583	0.626458
3	0.673509	0.001938
4	0.892942	0.087179

In [29]:
```
newdf.loc[(newdf['a']<0.5)].head(5)                          ### a th coumne where value are bigger than 0.5
```

Out[29]:

	a	b	c	d
6	0.014928	0.894558	0.980700	0.393523
7	0.085228	0.369409	0.230816	0.435636
9	0.372945	0.332607	0.754502	0.281589
10	0.357618	0.481962	0.822404	0.718223
11	0.086787	0.146956	0.430164	0.704789

```
In [30]: newdf.loc[(newdf['a']<0.5)  & (newdf['b']>0.3)]          ### more complex condtions
```

Out[30]:

	a	b	c	d
6	0.014928	0.894558	0.980700	0.393523
7	0.085228	0.369409	0.230816	0.435636
9	0.372945	0.332607	0.754502	0.281589
10	0.357618	0.481962	0.822404	0.718223
14	0.051548	0.712569	0.793383	0.181648
...	...	...	...	...
315	0.442578	0.450831	0.243229	0.264932
325	0.053568	0.496439	0.782086	0.748980
329	0.160003	0.961212	0.687316	0.171135
331	0.131520	0.631356	0.834375	0.814824
332	0.407084	0.833584	0.596161	0.000708

114 rows × 4 columns

```
In [31]: newdf.head(2)
```

Out[31]:

	a	b	c	d
0	735.000000	0.984169	0.074613	0.910435
1	0.797874	0.151738	0.131649	0.071332

```
In [32]: newdf.iloc[0,2]                                  ### can target any particular value   [ILOC]
```

```
Out[32]: 0.07461265973930176
```

```
In [33]: newdf = newdf.drop([1,2,5],axis=0)
```

In [34]:
```
newdf.head()
```

Out[34]:

	a	b	c	d
0	735.000000	0.984169	0.074613	0.910435
3	0.673509	0.001938	0.047301	0.777832
4	0.892942	0.087179	0.085141	0.226777
6	0.014928	0.894558	0.980700	0.393523
7	0.085228	0.369409	0.230816	0.435636

In [35]:
```
newdf.reset_index(drop=True,inplace=True)        ### reset number of index    [RESET_INDEX]
```

In [36]:
```
newdf.head()
```

Out[36]:

	a	b	c	d
0	735.000000	0.984169	0.074613	0.910435
1	0.673509	0.001938	0.047301	0.777832
2	0.892942	0.087179	0.085141	0.226777
3	0.014928	0.894558	0.980700	0.393523
4	0.085228	0.369409	0.230816	0.435636

In [37]:
```
newdf.dropna()                                  ### drop all null values           [DROPNA]
```

Out[37]:

	a	b	c	d
0	735.000000	0.984169	0.074613	0.910435
1	0.673509	0.001938	0.047301	0.777832
2	0.892942	0.087179	0.085141	0.226777
3	0.014928	0.894558	0.980700	0.393523
4	0.085228	0.369409	0.230816	0.435636
...	...	...	...	...
326	0.160003	0.961212	0.687316	0.171135
327	0.481626	0.233440	0.475106	0.909601
328	0.131520	0.631356	0.834375	0.814824
329	0.407084	0.833584	0.596161	0.000708
330	0.690825	0.182876	0.056837	0.194713

331 rows × 4 columns

In [38]:
```
newdf['b'].isnull()                             ### show all null values     [.ISNULL]
```

Out[38]:
```
0      False
1      False
2      False
3      False
4      False
       ...  
326    False
327    False
328    False
329    False
330    False
Name: b, Length: 331, dtype: bool
```

In [39]:

```
newdf['a'] = None
```

In [40]:

```
newdf.head(2)                                   ### replace all values as none
```

Out[40]:

	a	b	c	d
0	None	0.984169	0.074613	0.910435
1	None	0.001938	0.047301	0.777832

In [41]:

```
newdf['a'].isnull()
```

Out[41]:

```
0      True
1      True
2      True
3      True
4      True
       ...
326    True
327    True
328    True
329    True
330    True
Name: a, Length: 331, dtype: bool
```

In [42]:
```
df =pd.DataFrame({
    "name":['hary','rohan','skillf','shubh'],
    "marks" : [15,34,24,25],
    "city":['rampur','kolkata','bareli','nashik']
})

df.head()
```

Out[42]:

	name	marks	city
0	hary	15	rampur
1	rohan	34	kolkata
2	skillf	24	bareli
3	shubh	25	nashik

In [43]:
```
df.shape                                          ### shape of dataframe           [SHAPE]
```

Out[43]:
```
(4, 3)
```

In [44]:
```
df.info()                                        ### information about dataset      [INFO]
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   name    4 non-null      object
 1   marks   4 non-null      int64 
 2   city    4 non-null      object
dtypes: int64(1), object(2)
memory usage: 224.0+ bytes
```

In [45]:
```
df['name'].value_counts()                    ### counts the value of every parameter     [VALUE_COUNTS]
```

Out[45]:
```
shubh     1
rohan     1
skillf    1
hary      1
Name: name, dtype: int64
```

In [46]:

```
df.notnull()                                ### if null then false otherwise true          [NOTNULL]
```

Out[46]:

	name	marks	city
0	True	True	True
1	True	True	True
2	True	True	True
3	True	True	True

In [47]:

```
data1 = pd.read_excel(r"D:\harsh work\data_science\datasets\pixel-spreadsheet.xlsx")
data1.head(2)
###  reading excel file                   [READ_EXCEL]
```

Out[47]:

	Unnamed: 0	Unnamed: 0.1	Unnamed: 0.1.1	187	188	190	190.1	190.2	191	193	...	175	174	173	172	173.1	172.1	169	169.1	168
0	0	120000	120000	120000	180	182	182	182	183	183	...	167	166	165	164	162	161	161	161	160
1	1	1	1	166	167	171	171	169	170	171	...	148	147	146	145	144	143	142	142	139

2 rows × 99 columns

In [48]:

```
data1.iloc[0][0] = 120000
```

```
In [49]: data1.head()                                  ### value has been changed
```

Out[49]:

	Unnamed: 0	Unnamed: 0.1	Unnamed: 0.1.1	187	188	190	190.1	190.2	191	193	...	175	174	173	172	173.1	172.1	169	169.1	168
0	120000	120000	120000	120000	180	182	182	182	183	183	...	167	166	165	164	162	161	161	161	160
1	1	1	1	166	167	171	171	169	170	171	...	148	147	146	145	144	143	142	142	139
2	2	2	2	187	189	191	189	189	191	192	...	176	175	174	174	175	175	171	171	170
3	3	3	3	179	181	183	181	181	183	184	...	168	167	166	166	164	164	163	163	162
4	4	4	4	166	168	172	170	168	170	171	...	149	148	147	147	146	146	144	144	143

5 rows × 99 columns

```
In [50]: data1.to_excel(r"D:\harsh work\data_science\datasets\pixel-spreadsheet.xlsx")
         ### changes has been saved in excel file
```

END