Pandas Tutorial

July 15, 2023

[1]:
```
import numpy as np
import pandas as pd
```

[2]:
```
dict1 = {
    "Name" : ['Amit','Vicky','Vikash','Sandeep'],
    "Age" : [20,22,25,28],
    "City" : ['Kanpur','Varansi','Odisha','Assam']
}
```

[3]:
```
df = pd.DataFrame(dict1)
#DataFrame me rows and columns hotes hai
```

[4]:
```
df
```

[4]:

	Name	Age	City
0	Amit	20	Kanpur
1	Vicky	22	Varansi
2	Vikash	25	Odisha
3	Sandeep	28	Assam

[5]:
```
df.to_csv('friends.csv')

#Export this data frame into excel
```

[39]:
```
df.to_csv('friends_index_false.csv',index = False)

#Serial number nahi dikhega
```

[7]:
```
df.head(2)
#First two
```

[7]:

	Name	Age	City
0	Amit	20	Kanpur
1	Vicky	22	Varansi

[8]:
```
df.tail(2)
#Last two
```

[8]:
```
      Name  Age    City
2   Vikash   25  Odisha
3  Sandeep   28   Assam
```

[9]:
```
df.describe()
```

[9]:
```
             Age
count   4.00
mean   23.75
std     3.50
min    20.00
25%    21.50
50%    23.50
75%    25.75
max    28.00
```

[10]:
```
amit = pd.read_csv('friends.csv')
```

[11]:
```
amit
```

[11]:
```
   Unnamed: 0     Name  Age     City
0           0     Amit   20   Kanpur
1           1    Vicky   22  Varansi
2           2   Vikash   25   Odisha
3           3  Sandeep   28    Assam
```

[17]:
```
ser = pd.Series(np.random.rand(10))
#Series me 1 row or 1 column hota hai.
```

[18]:
```
ser
```

[18]:
```
0    0.836503
1    0.067171
2    0.105803
3    0.523887
4    0.260700
5    0.098100
6    0.814473
7    0.670541
8    0.161247
9    0.457720
dtype: float64
```

[19]:
```
type(ser)
```

[19]:
```
pandas.core.series.Series
```

[20]:
```
newdf = pd.DataFrame(np.random.rand(10,2))
```

[21]:
```
newdf
```

[21]:

	0	1
0	0.961628	0.718404
1	0.559137	0.665605
2	0.565264	0.953912
3	0.184048	0.705683
4	0.578378	0.774647
5	0.135605	0.788566
6	0.437779	0.476832
7	0.861719	0.809773
8	0.816871	0.432000
9	0.590701	0.136572

[22]:
```
type(newdf)
```

[22]:
```
pandas.core.frame.DataFrame
```

[23]:
```
newdf.describe()
```

[23]:

	0	1
count	10.000000	10.000000
mean	0.569113	0.646199
std	0.269813	0.236085
min	0.135605	0.136572
25%	0.468119	0.524025
50%	0.571821	0.712044
75%	0.760329	0.785086
max	0.961628	0.953912

[24]:
```
newdf.dtypes
```

[24]:
```
0    float64
1    float64
dtype: object
```

[25]:
```
newdf.head()
```

[25]:

	0	1
0	0.961628	0.718404
1	0.559137	0.665605
2	0.565264	0.953912
3	0.184048	0.705683
4	0.578378	0.774647

[26]:
```
newdf.dtypes
```

[26]:
```
0    float64
1    float64
dtype: object
```

[27]:
```
newdf[0][0] = "Amit"
```

[28]:
```
newdf.head()
```

[28]:

	0	1
0	Amit	0.718404
1	0.559137	0.665605
2	0.565264	0.953912
3	0.184048	0.705683
4	0.578378	0.774647

[29]:
```
newdf.dtypes
```

[29]:
```
0     object
1    float64
dtype: object
```

[32]:
```
newdf.index
```

[32]:
```
RangeIndex(start=0, stop=10, step=1)
```

[31]:
```
newdf.columns
```

[31]:
```
RangeIndex(start=0, stop=2, step=1)
```

[33]:
```
newdf.to_numpy()
```

[33]:
```
array([['Amit', 0.718404233018901],
       [0.559136921087114, 0.6656051429179622],
       [0.5652638497666339, 0.9539115666403768],
       [0.18404763082078457, 0.7056827985776976],
       [0.57837774112977, 0.7746472839821646],
       [0.1356049102625494, 0.7885656203024453],
       [0.43777931521836466, 0.47683225224940784],
       [0.8617188234096577, 0.8097729901082146],
       [0.8168711313163136, 0.43200029687297925],
       [0.5907007693097568, 0.13657161857664224]], dtype=object)
```

[34]:
```
newdf[0][0] = 0.22
```

[35]:
```
newdf.to_numpy()
```

[35]:
```
array([[0.22, 0.718404233018901],
       [0.559136921087114, 0.6656051429179622],
       [0.5652638497666339, 0.9539115666403768],
       [0.18404763082078457, 0.7056827985776976],
       [0.57837774112977, 0.7746472839821646],
       [0.1356049102625494, 0.7885656203024453],
       [0.43777931521836466, 0.47683225224940784],
       [0.8617188234096577, 0.8097729901082146],
       [0.8168711313163136, 0.43200029687297925],
       [0.5907007693097568, 0.13657161857664224]], dtype=object)
```

[36]:
```
newdf
```

[36]:

	0	1
0	0.22	0.718404
1	0.559137	0.665605
2	0.565264	0.953912
3	0.184048	0.705683
4	0.578378	0.774647
5	0.135605	0.788566
6	0.437779	0.476832
7	0.861719	0.809773
8	0.816871	0.432000
9	0.590701	0.136572

[38]:
```
newdf.T  #T ranspose kr dega
```

[38]:

	0	1	2	3	4	5	6	\
0	0.22	0.559137	0.565264	0.184048	0.578378	0.135605	0.437779	
1	0.718404	0.665605	0.953912	0.705683	0.774647	0.788566	0.476832	

	7	8	9
0	0.861719	0.816871	0.590701
1	0.809773	0.432	0.136572

[40]:
```
newdf.head()
#Means it will show some datasets from starting
```

[40]:

	0	1
0	0.22	0.718404
1	0.559137	0.665605
2	0.565264	0.953912
3	0.184048	0.705683
4	0.578378	0.774647

[41]:
```
newdf.sort_index(axis=1,ascending=False)
#It will reverse
```

[41]:
```
          1         0
0  0.718404      0.22
1  0.665605  0.559137
2  0.953912  0.565264
3  0.705683  0.184048
4  0.774647  0.578378
5  0.788566  0.135605
6  0.476832  0.437779
7  0.809773  0.861719
8  0.432000  0.816871
9  0.136572  0.590701
```

[42]:
```
newdf[0]
#It is a series
```

[42]:
```
0        0.22
1    0.559137
2    0.565264
3    0.184048
4    0.578378
5    0.135605
6    0.437779
7    0.861719
8    0.816871
9    0.590701
Name: 0, dtype: object
```

[45]:
```
newdf2 = newdf
#Here we can consider newdf2 is pointer
# i.e if we make some changes in newdf2 then newdf also changes
```

[48]:
```
newdf3 = newdf.copy()
#or newdf3 = newdf[:]
#in this ,if we make some changes in newdf3 then no change in newdf
```

[49]:
```
newdf.loc[0,0] = 200111
```

[50]:
```
newdf.head(2)
```

[50]:
```
          0         1
0    200111  0.718404
1  0.559137  0.665605
```

[51]:
```
newdf.columns = list("AB")
```

[53]:
```
newdf.loc[0,'A'] = 200111
```

[54]:
```
newdf
```

[54]:

	A	B
0	200111	0.718404
1	0.559137	0.665605
2	0.565264	0.953912
3	0.184048	0.705683
4	0.578378	0.774647
5	0.135605	0.788566
6	0.437779	0.476832
7	0.861719	0.809773
8	0.816871	0.432000
9	0.590701	0.136572

[59]:
```
newdf.loc[0,0] = 1234
#If column name '0' is not present then it will create a new column with
# column name '0'.
```

[60]:
```
newdf
```

[60]:

	A	B	0
0	200111	0.718404	1234.0
1	0.559137	0.665605	NaN
2	0.565264	0.953912	NaN
3	0.184048	0.705683	NaN
4	0.578378	0.774647	NaN
5	0.135605	0.788566	NaN
6	0.437779	0.476832	NaN
7	0.861719	0.809773	NaN
8	0.816871	0.432000	NaN
9	0.590701	0.136572	NaN

[62]:
```
newdf = newdf.drop(0,axis = 1)
# It will drop '0' column
# axis = 1 means want to delete column
```

[64]:
```
newdf.loc[[0,1],['A','B']]
#It will give row 0 and 1 and column A and B
```

[64]:

	A	B
0	200111	0.718404
1	0.559137	0.665605

[67]:
```
# newdf.loc[[0,1],:]
# Gives all the column
# Similarly newdf.loc[:,['A','B']]
# gives all the row
```

[69]:
```
newdf.loc[(newdf['A']<0.3)]
# in the Ath column, gives value less 0.3
```

[69]:

	A	B
3	0.184048	0.705683
5	0.135605	0.788566

[71]:
```
newdf.iloc[0,1]
# Whether index is named ABC... or 0,1,2.....
#independent of naming of rows or column
```

[71]:
```
0.718404233018901
```