Pandas Basics ¶

Python Pandas is defined as an open-source library that provides high-performance data manipulation in Python.It is used for data analysis in Python and developed by Wes McKinney in 2008.

Importing the libraries

In [1]:
```
import pandas as pd
```

In [2]:
```
import numpy as np
```

Pandas Series

Create Series using list

In [3]:
```
a=[1,2,3,4,5,6]
b=pd.Series(a)
b
```

Out[3]:
```
0    1
1    2
2    3
3    4
4    5
5    6
dtype: int64
```

In [4]:
```
a=['apple','banana','pineapple','orange']
b=pd.Series(a,index=['1','2','3','4'])
b
```

Out[4]:
```
1        apple
2       banana
3    pineapple
4       orange
dtype: object
```

Create Series using dictionary

In [5]:
```
a=pd.Series({'apple':20,'banana':35,'pineapple':15,'orange':30})
```

In [6]:
```
1 a
```

Out[6]:
```
apple         20
banana        35
pineapple     15
orange        30
dtype: int64
```

Data Frame-->Pandas .Dataframe

In [7]:
```
1 a={'col1':['apple'],'col2':['banana'],'col3':['pineapple'],'col4':['orange']}#using
2 b=pd.DataFrame(a)
3 b
```

Out[7]:

	col1	col2	col3	col4
0	apple	banana	pineapple	orange

In [8]:
```
1 a=[[1,2,3],[4,5,6]]
2 b=pd.DataFrame(a,columns=['col1','col2','col3'])
3 b
```

Out[8]:

	col1	col2	col3
0	1	2	3
1	4	5	6

Add a column

In [9]:
```
1 b['col4']=[7,8]
2 b
```

Out[9]:

	col1	col2	col3	col4
0	1	2	3	7
1	4	5	6	8

In [10]:
```
1 tem_df=pd.DataFrame({'city':['mumbai','delhi','banglore','hyderabad'],'tem':[45,40,
2 tem_df
```

Out[10]:

	city	tem
0	mumbai	45
1	delhi	40
2	banglore	48
3	hyderabad	46

In [11]:
```
hum_df=pd.DataFrame({'city':['mumbai','delhi','chennai','hyderabad'],'hum':[50,55,5
hum_df
```

Out[11]:

	city	hum
0	mumbai	50
1	delhi	55
2	chennai	54
3	hyderabad	60

Combining Dataframes

In [12]:
```
df=pd.concat([tem_df,hum_df]) # concat
df
```

Out[12]:

	city	tem	hum
0	mumbai	45.0	NaN
1	delhi	40.0	NaN
2	banglore	48.0	NaN
3	hyderabad	46.0	NaN
0	mumbai	NaN	50.0
1	delhi	NaN	55.0
2	chennai	NaN	54.0
3	hyderabad	NaN	60.0

In [13]:
```
df=pd.concat([tem_df,hum_df],ignore_index=True)
df
```

Out[13]:

	city	tem	hum
0	mumbai	45.0	NaN
1	delhi	40.0	NaN
2	banglore	48.0	NaN
3	hyderabad	46.0	NaN
4	mumbai	NaN	50.0
5	delhi	NaN	55.0
6	chennai	NaN	54.0
7	hyderabad	NaN	60.0

In [14]:
```
df=pd.concat([tem_df,hum_df],axis=1)
df
```

Out[14]:

	city	tem	city	hum
0	mumbai	45	mumbai	50
1	delhi	40	delhi	55
2	banglore	48	chennai	54
3	hyderabad	46	hyderabad	60

Merging of Dataframes

In [15]:
```
#Inner Join
df=pd.merge(tem_df,hum_df,on='city',how='inner')
df
```

Out[15]:

	city	tem	hum
0	mumbai	45	50
1	delhi	40	55
2	hyderabad	46	60

In [16]:
```
#Outer join
df=pd.merge(tem_df,hum_df,on='city',how='outer')
df
```

Out[16]:

	city	tem	hum
0	mumbai	45.0	50.0
1	delhi	40.0	55.0
2	banglore	48.0	NaN
3	hyderabad	46.0	60.0
4	chennai	NaN	54.0

In [17]:
```
#Left join
df=pd.merge(tem_df,hum_df,on='city',how='left')
df
```

Out[17]:

	city	tem	hum
0	mumbai	45	50.0
1	delhi	40	55.0
2	banglore	48	NaN
3	hyderabad	46	60.0

In [18]:
```
#Right join
df=pd.merge(tem_df,hum_df,on='city',how='right')
df
```

Out[18]:

	city	tem	hum
0	mumbai	45.0	50
1	delhi	40.0	55
2	chennai	NaN	54
3	hyderabad	46.0	60

In [ ]:
```
1
```

Pandas Advance

Import data/Load Data

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

In [2]:
```
df=pd.read_excel('C:\\Users\\dell\\Music\\Book new.xlsx')
df
```

Out[2]:

	Age	Gender	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	AG	LiverPatient
0	65	Female	0.7	0.1	187	16	18	6.8	3.3	0.90	Yes
1	62	Male	10.9	5.5	699	64	100	7.5	3.2	0.74	Yes
2	62	Male	7.3	4.1	490	60	68	7.0	3.3	0.89	Yes
3	58	Male	1.0	0.4	182	14	20	6.8	3.4	NaN	Yes
4	72	NaN	3.9	2.0	195	27	59	7.3	2.4	0.40	Yes
5	46	Male	1.8	0.7	208	19	14	7.6	4.4	1.30	Yes
6	26	NaN	0.9	0.2	154	16	12	7.0	3.5	NaN	Yes
7	29	Female	NaN	0.3	202	14	11	6.7	3.6	1.10	Yes
8	17	Male	0.9	0.3	202	22	19	7.4	4.1	1.20	No

First Five Observation

In [3]:
```
df.head()
```

Out[3]:

	Age	Gender	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	AG	LiverPatient
0	65	Female	0.7	0.1	187	16	18	6.8	3.3	0.90	Yes
1	62	Male	10.9	5.5	699	64	100	7.5	3.2	0.74	Yes
2	62	Male	7.3	4.1	490	60	68	7.0	3.3	0.89	Yes
3	58	Male	1.0	0.4	182	14	20	6.8	3.4	NaN	Yes
4	72	NaN	3.9	2.0	195	27	59	7.3	2.4	0.40	Yes

Last Five Observation

In [4]:
```
1 df.tail()
```

Out[4]:

	Age	Gender	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	AG	LiverPatient
4	72	NaN	3.9	2.0	195	27	59	7.3	2.4	0.4	Yes
5	46	Male	1.8	0.7	208	19	14	7.6	4.4	1.3	Yes
6	26	NaN	0.9	0.2	154	16	12	7.0	3.5	NaN	Yes
7	29	Female	NaN	0.3	202	14	11	6.7	3.6	1.1	Yes
8	17	Male	0.9	0.3	202	22	19	7.4	4.1	1.2	No

Column Names/ Variables

In [5]:
```
1 df.keys
```

Out[5]:
```
<bound method NDFrame.keys of    Age  Gender    TB   DB  Alkphos  Sgpt  Sgot   TP  ALB
AG LiverPatient
0   65  Female   0.7  0.1      187    16    18  6.8  3.3  0.90          Yes
1   62    Male  10.9  5.5      699    64   100  7.5  3.2  0.74          Yes
2   62    Male   7.3  4.1      490    60    68  7.0  3.3  0.89          Yes
3   58    Male   1.0  0.4      182    14    20  6.8  3.4   NaN          Yes
4   72     NaN   3.9  2.0      195    27    59  7.3  2.4  0.40          Yes
5   46    Male   1.8  0.7      208    19    14  7.6  4.4  1.30          Yes
6   26     NaN   0.9  0.2      154    16    12  7.0  3.5   NaN          Yes
7   29  Female   NaN  0.3      202    14    11  6.7  3.6  1.10          Yes
8   17    Male   0.9  0.3      202    22    19  7.4  4.1  1.20           No>
```

Concise summary of Dataframes

In [6]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9 entries, 0 to 8
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Age           9 non-null      int64
 1   Gender        7 non-null      object
 2   TB            8 non-null      float64
 3   DB            9 non-null      float64
 4   Alkphos       9 non-null      int64
 5   Sgpt          9 non-null      int64
 6   Sgot          9 non-null      int64
 7   TP            9 non-null      float64
 8   ALB           9 non-null      float64
 9   AG            7 non-null      float64
 10  LiverPatient  9 non-null      object
dtypes: float64(5), int64(4), object(2)
memory usage: 920.0+ bytes
```

Describing the whole data

In [7]:
```
1 df.describe()
```

Out[7]:

	Age	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	AG
count	9.000000	8.000000	9.000000	9.000000	9.000000	9.000000	9.000000	9.000000	7.000000
mean	48.555556	3.425000	1.511111	279.888889	28.000000	35.666667	7.122222	3.466667	0.932857
std	19.887880	3.776147	1.980811	186.536487	19.742087	32.019525	0.334581	0.565685	0.304998
min	17.000000	0.700000	0.100000	154.000000	14.000000	11.000000	6.700000	2.400000	0.400000
25%	29.000000	0.900000	0.300000	187.000000	16.000000	14.000000	6.800000	3.300000	0.815000
50%	58.000000	1.400000	0.400000	202.000000	19.000000	19.000000	7.000000	3.400000	0.900000
75%	62.000000	4.750000	2.000000	208.000000	27.000000	59.000000	7.400000	3.600000	1.150000
max	72.000000	10.900000	5.500000	699.000000	64.000000	100.000000	7.600000	4.400000	1.300000

In [8]:
```
1 df.columns ## Column names
```

Out[8]:
```
Index(['Age', 'Gender', 'TB', 'DB', 'Alkphos', 'Sgpt', 'Sgot', 'TP', 'ALB',
       'AG', 'LiverPatient'],
      dtype='object')
```

Data types of each column

In [9]:
```
1 df.dtypes
```

Out[9]:
```
Age              int64
Gender          object
TB             float64
DB             float64
Alkphos          int64
Sgpt             int64
Sgot             int64
TP             float64
ALB            float64
AG             float64
LiverPatient    object
dtype: object
```

Checking for missing value

In [10]:
```
1 df.isnull().sum()
```

Out[10]:
```
Age             0
Gender          2
TB              1
DB              0
Alkphos         0
Sgpt            0
Sgot            0
TP              0
ALB             0
AG              2
LiverPatient    0
dtype: int64
```

Accessing single columns

In [11]:
```
1 df.TB
```

Out[11]:
```
0     0.7
1    10.9
2     7.3
3     1.0
4     3.9
5     1.8
6     0.9
7     NaN
8     0.9
Name: TB, dtype: float64
```

In [12]:
```
1 df['TB']
```

Out[12]:
```
0     0.7
1    10.9
2     7.3
3     1.0
4     3.9
5     1.8
6     0.9
7     NaN
8     0.9
Name: TB, dtype: float64
```

Accessing multiple columns

In [13]:
```
1 df[['TB','DB']]
```

Out[13]:

	TB	DB
0	0.7	0.1
1	10.9	5.5
2	7.3	4.1
3	1.0	0.4
4	3.9	2.0
5	1.8	0.7
6	0.9	0.2
7	NaN	0.3
8	0.9	0.3

In [14]:
```
1 df['TB'].values
```

Out[14]:
```
array([ 0.7, 10.9,  7.3,  1. ,  3.9,  1.8,  0.9,  nan,  0.9])
```

Drop a row

In [15]:
```
1 df1=df.drop(0,axis=0)
```

In [16]:
```
1 df1
```

Out[16]:

	Age	Gender	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	AG	LiverPatient
1	62	Male	10.9	5.5	699	64	100	7.5	3.2	0.74	Yes
2	62	Male	7.3	4.1	490	60	68	7.0	3.3	0.89	Yes
3	58	Male	1.0	0.4	182	14	20	6.8	3.4	NaN	Yes
4	72	NaN	3.9	2.0	195	27	59	7.3	2.4	0.40	Yes
5	46	Male	1.8	0.7	208	19	14	7.6	4.4	1.30	Yes
6	26	NaN	0.9	0.2	154	16	12	7.0	3.5	NaN	Yes
7	29	Female	NaN	0.3	202	14	11	6.7	3.6	1.10	Yes
8	17	Male	0.9	0.3	202	22	19	7.4	4.1	1.20	No

Filling the missing values

In [17]:
```
df1['AG'].fillna(df['AG'].median(),inplace=True)
df1
```

Out[17]:

	Age	Gender	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	AG	LiverPatient
1	62	Male	10.9	5.5	699	64	100	7.5	3.2	0.74	Yes
2	62	Male	7.3	4.1	490	60	68	7.0	3.3	0.89	Yes
3	58	Male	1.0	0.4	182	14	20	6.8	3.4	0.90	Yes
4	72	NaN	3.9	2.0	195	27	59	7.3	2.4	0.40	Yes
5	46	Male	1.8	0.7	208	19	14	7.6	4.4	1.30	Yes
6	26	NaN	0.9	0.2	154	16	12	7.0	3.5	0.90	Yes
7	29	Female	NaN	0.3	202	14	11	6.7	3.6	1.10	Yes
8	17	Male	0.9	0.3	202	22	19	7.4	4.1	1.20	No

In [18]:
```
df1['TB'].fillna(df1['TB'].median(),inplace=True)
df1
```

Out[18]:

	Age	Gender	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	AG	LiverPatient
1	62	Male	10.9	5.5	699	64	100	7.5	3.2	0.74	Yes
2	62	Male	7.3	4.1	490	60	68	7.0	3.3	0.89	Yes
3	58	Male	1.0	0.4	182	14	20	6.8	3.4	0.90	Yes
4	72	NaN	3.9	2.0	195	27	59	7.3	2.4	0.40	Yes
5	46	Male	1.8	0.7	208	19	14	7.6	4.4	1.30	Yes
6	26	NaN	0.9	0.2	154	16	12	7.0	3.5	0.90	Yes
7	29	Female	1.8	0.3	202	14	11	6.7	3.6	1.10	Yes
8	17	Male	0.9	0.3	202	22	19	7.4	4.1	1.20	No

In [19]:
```
df1['Gender'].fillna(df['Gender'].mode()[0],inplace=True)
df1
```

Out[19]:

	Age	Gender	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	AG	LiverPatient
1	62	Male	10.9	5.5	699	64	100	7.5	3.2	0.74	Yes
2	62	Male	7.3	4.1	490	60	68	7.0	3.3	0.89	Yes
3	58	Male	1.0	0.4	182	14	20	6.8	3.4	0.90	Yes
4	72	Male	3.9	2.0	195	27	59	7.3	2.4	0.40	Yes
5	46	Male	1.8	0.7	208	19	14	7.6	4.4	1.30	Yes
6	26	Male	0.9	0.2	154	16	12	7.0	3.5	0.90	Yes
7	29	Female	1.8	0.3	202	14	11	6.7	3.6	1.10	Yes
8	17	Male	0.9	0.3	202	22	19	7.4	4.1	1.20	No

In [20]:
```
1 df1
```

Out[20]:

	Age	Gender	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	AG	LiverPatient
1	62	Male	10.9	5.5	699	64	100	7.5	3.2	0.74	Yes
2	62	Male	7.3	4.1	490	60	68	7.0	3.3	0.89	Yes
3	58	Male	1.0	0.4	182	14	20	6.8	3.4	0.90	Yes
4	72	Male	3.9	2.0	195	27	59	7.3	2.4	0.40	Yes
5	46	Male	1.8	0.7	208	19	14	7.6	4.4	1.30	Yes
6	26	Male	0.9	0.2	154	16	12	7.0	3.5	0.90	Yes
7	29	Female	1.8	0.3	202	14	11	6.7	3.6	1.10	Yes
8	17	Male	0.9	0.3	202	22	19	7.4	4.1	1.20	No

Droping multiple rows

In [ ]:
```
1 df1=df.drop([0,1,2,3],axis=0)
2 df1
```

In [ ]:
```
1 df1=df.drop([5,6],axis=0)
2 df1
```

In [ ]:
```
1 df1=df.drop(index=[0,1,2,3])
2 df1
```

Droping multiple columns

In [ ]:
```
1 df1=df.drop(['TB'],axis=1)
```

In [ ]:
```
1 df1
```

In [ ]:
```
1 df1=df.drop(['TB','DB'],axis=1)
2 df1
```

In [ ]:
```
1 df1=df.drop(columns=['TB',"AG"])
2 df1
```

In [ ]:
```
1 df['IB']=df['TB']-df['DB']
2 df
```

In [ ]:
```
1 df=df.drop(columns=['IB'])
2 df
```

shorting in ascending order

In [ ]:
```
df.sort_values(by='TB',ascending=True)
```

In [ ]:
```
df.sort_values(by='DB',ascending=True)
```

Data Extraction

Can be done by 3 ways

-By using Conditions -By using iloc() & loc() -Combination of condtions # loc()

In [ ]:
```
df['Gender']=='Male'
```

In [ ]:
```
df[df['Gender']=='Male']#Conditions
```

In [ ]:
```
df[df['Age']>=70]
```

Multiple Conditions

In [ ]:
```
filter=df[(df['Gender']=='Male') | (df['Age']>=40)]
filter
```

In [ ]:
```
filter2=df[(df['LiverPatient']=='Yes') & (df['Age']>=50)]
filter2
```

In [ ]:
```
df.iloc[[1,2,3]]
```

In [ ]:
```
df.iloc[[3,4,5],[6]]
```

Displaying specific column of range of rows

In [ ]:
```
df.loc[1,'TB':'AG']
```

In [ ]:
```
df.loc[[1,2,3,4],'TB':'DB']
```

In [ ]:
```
df.loc[[1,2,3,4],'TB':'AG']
```

In [ ]:
```
df.iloc[[1,2,3],[4,5]]
```

In [ ]:
```

```