In [1]:
```
#Pandas provide 2 types of data structures:
# 1. Series() --->1D array --> Contains homogeneous data(int,float,string)--
# 2. DataFrame() --->2D array (we will read the ipl data which is of rows &
# --> Size mutable (can add or remove cols)
```

In [7]:
```
import numpy as np
a1=np.array([1,2,3])
print(a1)
a1.dtype
```

```
[1 2 3]
```

Out[7]:
```
dtype('int32')
```

In [9]:
```
import pandas as pd
import numpy as np
```

Series

In [15]:
```
series1=pd.Series({"A":[1,2,3,4],"B":[6,4,2,8],"C":[65,34,21,32]}) #empty se
print(series1)
print(type(series1))
```

```
A        [1, 2, 3, 4]
B        [6, 4, 2, 8]
C    [65, 34, 21, 32]
dtype: object
<class 'pandas.core.series.Series'>
```

In [14]:
```
df1=pd.DataFrame({"A":[1,2,3,4],"B":[6,4,2,8]}) #empty series
print(df1)
print(type(df1))
```

```
   A  B
0  1  6
1  2  4
2  3  2
3  4  8
<class 'pandas.core.frame.DataFrame'>
```

In [18]:
```
s=pd.Series(data=(2,3))
s
```

Out[18]:
```
0    2
1    3
dtype: int64
```

In [19]:
```
#Label-based indexing  #here we have converted a list to series
s1=pd.Series([100,200,300,400,500],index=['a','b','c','d','e'])
s1
```

Out[19]:
```
a    100
b    200
c    300
d    400
e    500
dtype: int64
```

In [6]:
```
# converting dictionary to series
d={'a':100,'b':200,'c':300}
# s3=pd.Series({'a':100,'b':200,'c':300})
# s3
# s3=pd.Series(d)
# s3
s4=pd.Series(d,index=['b','d','e','a'])
s4
```

Out[6]:
```
b    200.0
d      NaN
e      NaN
a    100.0
dtype: float64
```

In [7]:
```
#numpy array ---->series
ar=np.array([100,200,300,400])
ar
xx=np.array(['a','b','c','d'])
s5=pd.Series(data=ar,index=xx)
s5
```

Out[7]:
```
a    100
b    200
c    300
d    400
dtype: int32
```

In [8]:
```
#create a series of 8 similar elements-->5,5,5,5,5,5,5,5

s6=pd.Series(5,index=[1,2,3,4,5])
s6
```

Out[8]:
```
1    5
2    5
3    5
4    5
5    5
dtype: int64
```

In [9]:
```
#list-->series
#dict-->series
```

In [10]:
```
# Indexing and Slicing
print(s5)
s5[2]
```

```
a    100
b    200
c    300
d    400
dtype: int32
```

Out[10]:
```
300
```

In [11]:
```
s5[1:4]
```

Out[11]:
```
b    200
c    300
d    400
dtype: int32
```

Functions in Series

In [12]:
```
s5
```

Out[12]:
```
a    100
b    200
c    300
d    400
dtype: int32
```

In [13]:
```
#axes: Returns the list of the labels of the series.
s5.axes
```

Out[13]:
```
[Index(['a', 'b', 'c', 'd'], dtype='object')]
```

In [14]:
```
#ndim: Returns the number of dimensions
s5.ndim
```

Out[14]:
```
1
```

In [15]:
```
#size:Returns the size(length) of the series.
s5.size
```

Out[15]:
```
4
```

In [16]:
```
#values: Returns the actual data in the series as an array.
s5.values
```

Out[16]:
```
array([100, 200, 300, 400])
```

In [17]:
```
#head: head() returns the first n rows
s5.head(2)
```

Out[17]:
```
a    100
b    200
dtype: int32
```

In [18]:
```
#tail: tail() returns the last n rows
s5.tail(2)
```

Out[18]:
```
c    300
d    400
dtype: int32
```

DataFrame

In [19]:
```
#dataframe is two dimensional-->size is mutable-->contains homogeneous data
```

In [20]:
```
#Empty DataFrame
df=pd.DataFrame()
print(df)
```

```
Empty DataFrame
Columns: []
Index: []
```

In [21]:
```
#converting 'list' into a dataframe
mylist=[[1,2,3,4],[5,6,7,8]]
df=pd.DataFrame(mylist,columns=['a','b','c','d'])
df
```

Out[21]:

	a	b	c	d
0	1	2	3	4
1	5	6	7	8

In [22]:
```
#converting 'dictionary' to dataframe
d={'name':['prachiti','riya','aditi','mayur'],'age':[20,22,24,25]}
df1=pd.DataFrame(d,index=['rank1','rank2','rank3','rank4'])
df1
```

Out[22]:

	name	age
rank1	prachiti	20
rank2	riya	22
rank3	aditi	24
rank4	mayur	25

In [23]:
```
#converting numpy array--->dataframe
n=pd.DataFrame(np.arange(1,5).reshape(2,2),index=['A','B'],columns=['AA','BB
n
```

Out[23]:

	AA	BB
A	1	2
B	3	4

In [24]:
```
# list-->df
# dict-->df
# columns
# index
# 2D numpy-->df
```

Functions in DataFrame

In [25]:
```
dt={'name':['smith','jhon','tom','micky','belly','fizz','ricky'],'age':[25,2
    'rating':[4.23,3.24,3.98,2.56,3.20,4.60,3.80]}
df3=pd.DataFrame(dt)
df3
```

Out[25]:

	name	age	rating
0	smith	25	4.23
1	jhon	26	3.24
2	tom	25	3.98
3	micky	23	2.56
4	belly	30	3.20
5	fizz	29	4.60
6	ricky	23	3.80

In [26]:
```
df3.T
```

Out[26]:

	0	1	2	3	4	5	6
name	smith	jhon	tom	micky	belly	fizz	ricky
age	25	26	25	23	30	29	23
rating	4.23	3.24	3.98	2.56	3.2	4.6	3.8

In [27]:
```
df3
```

Out[27]:

	name	age	rating
0	smith	25	4.23
1	jhon	26	3.24
2	tom	25	3.98
3	micky	23	2.56
4	belly	30	3.20
5	fizz	29	4.60
6	ricky	23	3.80

axes

In [28]:
```
df3.axes
```

Out[28]:
```
[RangeIndex(start=0, stop=7, step=1),
 Index(['name', 'age', 'rating'], dtype='object')]
```

dtypes : datatypes of each column

In [29]:
```
df3.dtypes
```

Out[29]:
```
name       object
age         int64
rating    float64
dtype: object
```

In [30]:
```
df3.ndim
```

Out[30]:
```
2
```

In [31]:
```
df3.shape
```

Out[31]:
```
(7, 3)
```

In [32]:
```
df3.size
```

Out[32]:
```
21
```

In [33]:
```
df3.values
```

Out[33]:
```
array([['smith', 25, 4.23],
       ['jhon', 26, 3.24],
       ['tom', 25, 3.98],
       ['micky', 23, 2.56],
       ['belly', 30, 3.2],
       ['fizz', 29, 4.6],
       ['ricky', 23, 3.8]], dtype=object)
```

In [34]:
```
df3.columns
```

Out[34]:
```
Index(['name', 'age', 'rating'], dtype='object')
```

In [35]:
```
df3.head() # display first 5 records
```

Out[35]:

	name	age	rating
0	smith	25	4.23
1	jhon	26	3.24
2	tom	25	3.98
3	micky	23	2.56
4	belly	30	3.20

In [36]:
```
df3.head(3)
```

Out[36]:

	name	age	rating
0	smith	25	4.23
1	jhon	26	3.24
2	tom	25	3.98

In [37]:
```
df3.tail()
```

Out[37]:

	name	age	rating
2	tom	25	3.98
3	micky	23	2.56
4	belly	30	3.20
5	fizz	29	4.60
6	ricky	23	3.80

In [38]:
```
df3.tail(3)
```

Out[38]:

	name	age	rating
4	belly	30	3.2
5	fizz	29	4.6
6	ricky	23	3.8

In [39]:
```
print(df3)
df3.sum()
```

```
    name  age  rating
0  smith   25    4.23
1   jhon   26    3.24
2    tom   25    3.98
3  micky   23    2.56
4  belly   30    3.20
5   fizz   29    4.60
6  ricky   23    3.80
```

Out[39]:
```
name      smithjhontommickybellyfizzricky
age                                   181
rating                              25.61
dtype: object
```

In [40]:
```
#Displaying a single column
df3['rating']
```

Out[40]:
```
0    4.23
1    3.24
2    3.98
3    2.56
4    3.20
5    4.60
6    3.80
Name: rating, dtype: float64
```

In [41]:
```
#Displaying multiple columns
df3[['age','rating']]
```

Out[41]:

	age	rating
0	25	4.23
1	26	3.24
2	25	3.98
3	23	2.56
4	30	3.20
5	29	4.60
6	23	3.80

In [42]:
```
df3['age'].sum()
```

Out[42]:
```
181
```

In [43]:
```
df3.sum(axis=1)
```

```
C:\Users\prach\AppData\Local\Temp\ipykernel_23712\2779808981.py:1: FutureWa
rning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_
only=None') is deprecated; in a future version this will raise TypeError.
Select only valid columns before calling the reduction.
  df3.sum(axis=1)
```

Out[43]:
```
0    29.23
1    29.24
2    28.98
3    25.56
4    33.20
5    33.60
6    26.80
dtype: float64
```

In [44]:
```
print(df3)
df3.min()
```

```
    name  age  rating
0  smith   25    4.23
1   jhon   26    3.24
2    tom   25    3.98
3  micky   23    2.56
4  belly   30    3.20
5   fizz   29    4.60
6  ricky   23    3.80
```

Out[44]:
```
name      belly
age          23
rating     2.56
dtype: object
```

In [45]:
```
df3.max()
```

Out[45]:
```
name      tom
age        30
rating    4.6
dtype: object
```

In [46]:
```
#Q find min value of age column
df3['age'].min()
```

Out[46]:
```
23
```

In [47]:
```
#Q find mean of age column
df3['age'].max()
```

Out[47]:
```
30
```

In [48]:
```
#print information of data frame
df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7 entries, 0 to 6
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   name    7 non-null      object 
 1   age     7 non-null      int64  
 2   rating  7 non-null      float64
dtypes: float64(1), int64(1), object(1)
memory usage: 296.0+ bytes
```

In [49]:
```
df3
```

Out[49]:

	name	age	rating
0	smith	25	4.23
1	jhon	26	3.24
2	tom	25	3.98
3	micky	23	2.56
4	belly	30	3.20
5	fizz	29	4.60
6	ricky	23	3.80

In [50]:
```
df3.describe()
```

Out[50]:

	age	rating
count	7.000000	7.000000
mean	25.857143	3.658571
std	2.734262	0.698628
min	23.000000	2.560000
25%	24.000000	3.220000
50%	25.000000	3.800000
75%	27.500000	4.105000
max	30.000000	4.600000

In [51]:
```
df3.describe(include=['object'])
```

Out[51]:

	name
count	7
unique	7
top	smith
freq	1

In [52]:
```
df3
```

Out[52]:

	name	age	rating
0	smith	25	4.23
1	jhon	26	3.24
2	tom	25	3.98
3	micky	23	2.56
4	belly	30	3.20
5	fizz	29	4.60
6	ricky	23	3.80

In [53]:
```
#add column
df3['marks']=[89,45,78,89,77,85,87]
df3
```

Out[53]:

	name	age	rating	marks
0	smith	25	4.23	89
1	jhon	26	3.24	45
2	tom	25	3.98	78
3	micky	23	2.56	89
4	belly	30	3.20	77
5	fizz	29	4.60	85
6	ricky	23	3.80	87

In [54]:
```
df3.rename(columns={'age':'AGE','name':'Name'})
```

Out[54]:

	Name	AGE	rating	marks
0	smith	25	4.23	89
1	jhon	26	3.24	45
2	tom	25	3.98	78
3	micky	23	2.56	89
4	belly	30	3.20	77
5	fizz	29	4.60	85
6	ricky	23	3.80	87

In [55]:
```
df3
```

Out[55]:

	name	age	rating	marks
0	smith	25	4.23	89
1	jhon	26	3.24	45
2	tom	25	3.98	78
3	micky	23	2.56	89
4	belly	30	3.20	77
5	fizz	29	4.60	85
6	ricky	23	3.80	87

In [56]:
```
df3.rename(columns={'age':'AGE','name':'Name'},inplace=True)
```

In [57]:
```
df3
```

Out[57]:

	Name	AGE	rating	marks
0	smith	25	4.23	89
1	jhon	26	3.24	45
2	tom	25	3.98	78
3	micky	23	2.56	89
4	belly	30	3.20	77
5	fizz	29	4.60	85
6	ricky	23	3.80	87

In [58]:
```
df3.drop([2])
```

Out[58]:

	Name	AGE	rating	marks
0	smith	25	4.23	89
1	jhon	26	3.24	45
3	micky	23	2.56	89
4	belly	30	3.20	77
5	fizz	29	4.60	85
6	ricky	23	3.80	87

In [59]:
```
df3
```

Out[59]:

	Name	AGE	rating	marks
0	smith	25	4.23	89
1	jhon	26	3.24	45
2	tom	25	3.98	78
3	micky	23	2.56	89
4	belly	30	3.20	77
5	fizz	29	4.60	85
6	ricky	23	3.80	87

In [60]:
```
df3.drop([2],inplace=True)
```

In [61]:
```
df3
```

Out[61]:

	Name	AGE	rating	marks
0	smith	25	4.23	89
1	jhon	26	3.24	45
3	micky	23	2.56	89
4	belly	30	3.20	77
5	fizz	29	4.60	85
6	ricky	23	3.80	87

In [62]:
```
df3.drop(['marks'],axis=1)
```

Out[62]:

	Name	AGE	rating
0	smith	25	4.23
1	jhon	26	3.24
3	micky	23	2.56
4	belly	30	3.20
5	fizz	29	4.60
6	ricky	23	3.80

In [63]:
```
df3.drop(['marks'],axis=1,inplace=True)
```

In [64]:
```
df3
```

Out[64]:

	Name	AGE	rating
0	smith	25	4.23
1	jhon	26	3.24
3	micky	23	2.56
4	belly	30	3.20
5	fizz	29	4.60
6	ricky	23	3.80

In [ ]: