Pandas

In [1]:
```
# single dimension= series object
# multi dimension = dataframe
```

Series

In [2]:
```
# series object
import pandas as pd
Parray=pd.Series([1,2,3,4,5])
Parray
# here 0 is index of 1 and so on
```

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

In [3]:
```
type(Parray)
```

Out[3]:
```
pandas.core.series.Series
```

In [4]:
```
# changing index
Changing_Index=pd.Series([5,7,6,5,2,3,2],index=["A","B","C","D","E","F","G"])
Changing_Index
```

Out[4]:
```
A    5
B    7
C    6
D    5
E    2
F    3
G    2
dtype: int64
```

In [5]:
```
# series with dictonaries
STdictonirseries=pd.Series({"A":1,"B":2,"C":3,"D":4})
print(STdictonirseries)
```

```
A    1
B    2
C    3
D    4
dtype: int64
```

In [6]:
```
# series with dictonaries change index
STdictonirIndexEtrypd.Series({"A":1,"B":2,"C":3,"D":4},index=["A","B","D","E"])
print(STdictonirIndexEt)
```

```
A    1.0
B    2.0
D    4.0
E    NaN
dtype: float64
```

In [7]:
```
# extracting  Single Element
SimpleSeriesOfPDpd.Series([1,5,7,93,5,41,5,6,9,7])
SimpleSeriesOfPD[5]
```

Out[7]:
```
93
```

In [8]:
```
SimpleSeriesOfPD[-3:]
```

Out[8]:
```
7    6
8    9
9    7
dtype: int64
```

In [9]:
```
SimpleSeriesOfPD[:4]
```

Out[9]:
```
0    1
1    5
2    7
3    93
dtype: int64
```

In [10]:
```
# math operation on series
# Sum
MathSeries=pd.Series([10,50,70,62,41])
MathSeries+3
```

Out[10]:
```
0    13
1    53
2    81
3    65
4    44
dtype: int64
```

In [11]:
```
#Sub
MathSeries-3
```

Out[11]:
```
0     7
1    47
2    75
3    59
4    38
dtype: int64
```

In [12]:
```
# divide
MathSeries/10
```

Out[12]:
```
0    1.0
1    5.0
2    7.0
3    6.2
4    4.1
dtype: float64
```

In [13]:
```
# Adding two Series
SeriesOneregd.Series([14,75,35,41,85,32,41,62,30])
SeriesTworegd.Series([51,65,72,68,15,32,15,17,86])
SeriesOneregd+SeriesTwo
```

Out[13]:
```
0    65
1    140
2    107
3    109
4    100
5     64
6     56
7     79
8    116
dtype: int64
```

Data frame

In [15]:
```
# creating Daa frame using dictionaries
DataFrameregd.DataFrame({"Name":["kaushik","Harsh","Abhinesh","Reena"],"ID":[101,102,103,104]})
print(DataFrame)
```

```
     Name   ID
0  kaushik  101
1    Harsh  102
2  Abhinesh  103
3    Reena  104
```

In [17]:
```
# dataFrame Builtin function
# l[head] to get first five rows in data set
# l[describe()] to get general information abut data aset
# l[tail()] to get last five rows in data set
# l[shape()] to get rows and colum in dat set
```

In [23]:
```
# load dataset
irisnpd.read_csv('iris.csv')
```

In [22]:
```
# first five record of data set
iris.head()
```

Out[22]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [24]:
```
iris.tail()
```

Out[24]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

In [25]:
```
iris.describe()
```

Out[25]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

In [27]:
```
iris.shape
```

Out[27]:
```
(150, 6)
```

In [34]:
```
# like[rows,colums] to extract value from data set using index
iris.iloc[5:11,2:]
```

Out[34]:

	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
5	3.9	1.7	0.4	Iris-setosa
6	3.4	1.4	0.3	Iris-setosa
7	3.4	1.5	0.2	Iris-setosa
8	2.9	1.4	0.2	Iris-setosa
9	3.1	1.5	0.1	Iris-setosa
10	3.7	1.5	0.2	Iris-setosa

In [38]:
```
# loc[colums names] to extract value from data set using colum names
iris.loc[0:3,["SepalWidthCm","PetalLengthCm","Species"]] # here 0:3 is rows and both inclusive
```

Out[38]:

	SepalWidthCm	PetalLengthCm	Species
0	3.5	1.4	Iris-setosa
1	3.0	1.4	Iris-setosa
2	3.2	1.3	Iris-setosa
3	3.1	1.5	Iris-setosa

In [40]:
```
iris.head()
```

Out[40]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [42]:
```
# Droping colums
iris.drop("PetalWidthCm",axis=1) # 1= drop column and 0 = drop rows
```

Out[42]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	Species
0	1	5.1	3.5	1.4	Iris-setosa
1	2	4.9	3.0	1.4	Iris-setosa
2	3	4.7	3.2	1.3	Iris-setosa
3	4	4.6	3.1	1.5	Iris-setosa
4	5	5.0	3.6	1.4	Iris-setosa
...	...	...	...	...	...
145	146	6.7	3.0	5.2	Iris-virginica
146	147	6.3	2.5	5.0	Iris-virginica
147	148	6.5	3.0	5.2	Iris-virginica
148	149	6.2	3.4	5.4	Iris-virginica
149	150	5.9	3.0	5.1	Iris-virginica

150 rows × 5 columns

In [45]:
```
# droping rows
# iris.drop([index of rows],axis=0)
iris.drop([1,2,3],axis=0)
```

Out[45]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
5	6	5.4	3.9	1.7	0.4	Iris-setosa
6	7	4.6	3.4	1.4	0.3	Iris-setosa
7	8	5.0	3.4	1.5	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

147 rows × 6 columns

In [46]:
```
# dat aset mean median min max
iris.mean()
```

Out[46]:
```
Id               75.500000
SepalLengthCm     5.843333
SepalWidthCm      3.054000
PetalLengthCm     3.758667
PetalWidthCm      1.198667
dtype: float64
```

In [48]:
```
iris.median()
```

Out[48]:
```
Id               75.50
SepalLengthCm     5.80
SepalWidthCm      3.00
PetalLengthCm     4.35
PetalWidthCm      1.30
dtype: float64
```

In [49]:
```
iris.max()
```

Out[49]:
```
Id                   150
SepalLengthCm        7.9
SepalWidthCm         4.4
PetalLengthCm        6.9
PetalWidthCm         2.5
Species    Iris-virginica
dtype: object
```

In [50]:
```
iris.min()
```

Out[50]:
```
Id                   1
SepalLengthCm      4.3
SepalWidthCm         2
PetalLengthCm        1
PetalWidthCm       0.1
Species    Iris-setosa
dtype: object
```

In [58]:
```
# apply  function . user defined function apply on dat aset or its clouns "half is user define function"
def double_make(s):
    return s*2
```

In [59]:
```
iris.head()
```

Out[59]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [60]:
```
iris[["PetalWidthCm","PetalLengthCm"]].apply(double_make)
```

Out[60]:

	PetalWidthCm	PetalLengthCm
0	0.4	2.8
1	0.4	2.8
2	0.4	2.6
3	0.4	3.0
4	0.4	2.8
...	...	...
145	4.6	10.4
146	3.8	10.0
147	4.0	10.4
148	4.6	10.8
149	3.6	10.2

150 rows × 2 columns

In [ ]:
```
# value count
# sort values
```

In [63]:
```
iris["Species"].value_counts()
```

Out[63]:
```
Iris-versicolor    50
Iris-setosa        50
Iris-virginica     50
Name: Species, dtype: int64
```

In [64]:
```
iris.sort_values(by="SepalWidthCm")
```

Out[64]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
60	61	5.0	2.0	3.5	1.0	Iris-versicolor
62	63	6.0	2.2	4.0	1.0	Iris-versicolor
119	120	6.0	2.2	5.0	1.5	Iris-virginica
68	69	6.2	2.2	4.5	1.5	Iris-versicolor
41	42	4.5	2.3	1.3	0.3	Iris-setosa
...	...	...	...	...	...	...
16	17	5.4	3.9	1.3	0.4	Iris-setosa
14	15	5.8	4.0	1.2	0.2	Iris-setosa
32	33	5.2	4.1	1.5	0.1	Iris-setosa
33	34	5.5	4.2	1.4	0.2	Iris-setosa
15	16	5.7	4.4	1.5	0.4	Iris-setosa

150 rows × 6 columns