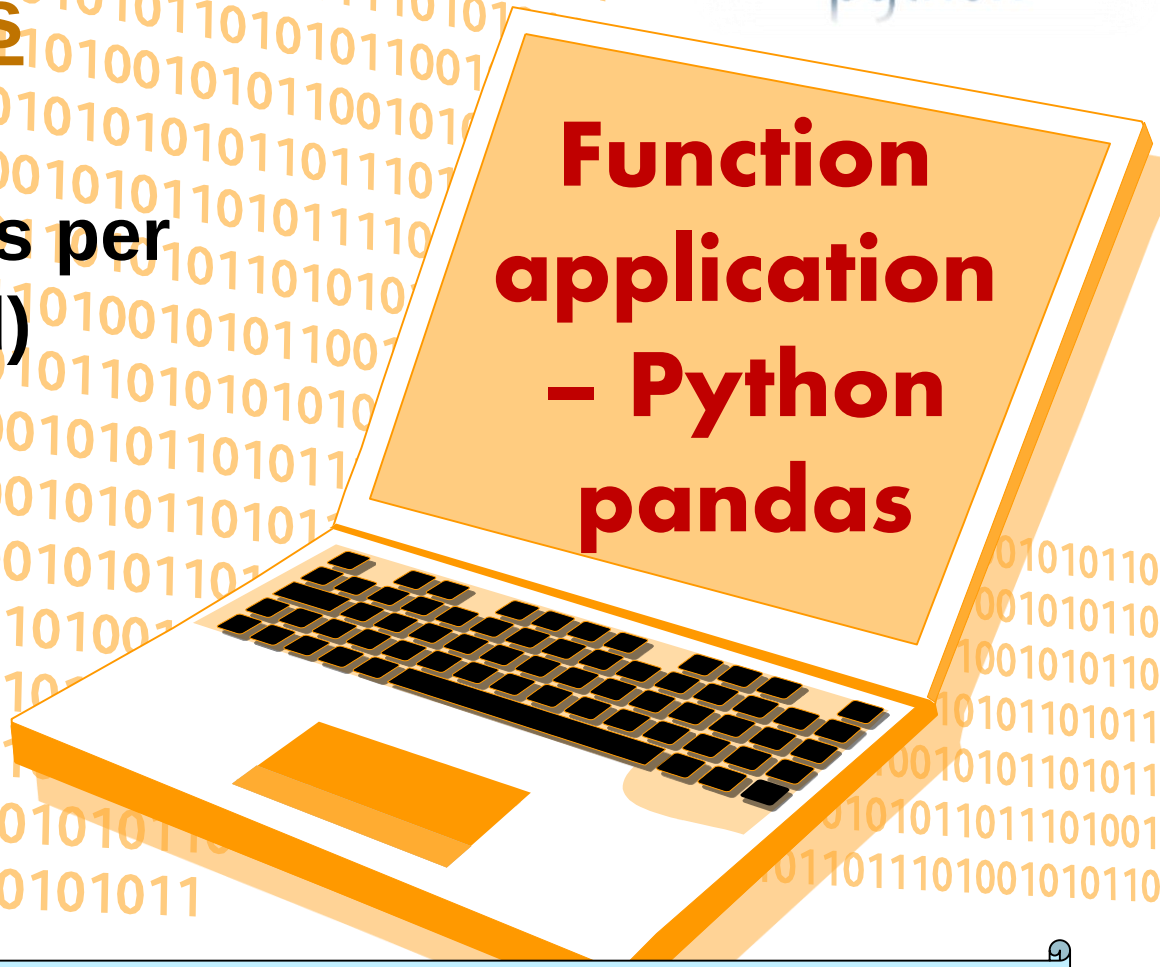
Chapter 3 :

Informatics Practices

Class XII (As per CBSE Board)



New Syllabus 2019-20

Visit : python.mykvs.in for regular updates

Function application

Pandas provide three important functions namely pipe(), apply() and applymap() ,to apply our own function or some other library's function.Use of these functions depend on entire dataframe,row-coloumn elements or element wise.

- Table wise Function Application: pipe()
- Row or Column Wise Function Application: apply()
- Element wise Function Application: applymap()

Function application

Table wise Function Application: pipe()

Pipe() function performs the operation for the entire dataframe with the help of user defined or library function. In below example we are using pipe() Function to add 5 value to the entire dataframe.

e.g.program.

```
import pandas as pd
import numpy as np
import math
# own function
def adder(adder1,adder2):
   return adder1+adder2

#Create a Dictionary of series
d = {'science_marks':pd.Series([22,55,63,85,47]),
   'english_marks':pd.Series([89,87,67,55,47])}

df = pd.DataFrame(d)
df1=df.pipe(adder,5)
print (df1)
```

OUTPUT

	science_marks	english_marks
0	27	94
1	60	92
2	68	72
3	90	60
4	52	52

Function application

Table wise Function Application: pipe()

Basic idea behind pipe() function is like,we want to apply a function to a data frame or series, to then apply other, other, … It will be sandwich like structure:

df = fun3(fun2(fun1(df, arg1= 1), arg2= 2), arg3=3)

In essence here we want is pipelining just similar to chaining.

e.g.program

```
import pandas as pd
import numpy as np
import math
# own functions
def adder(adder1,adder2):
   return adder1+adder2
def divide(adder1,adder2):
   return adder1/adder2
#Create a Dictionary of series
d = {'science_marks':pd.Series([22,55]),
   'english_marks':pd.Series([89,87])}
df = pd.DataFrame(d)
print(df)
df1=df.pipe(adder,5).pipe(divide,2)
print (df1)
```

```
 science_marks  english_marks
0             22             89
1             55             87

After adder () call
 science_marks  english_marks
0             27             94
1             60             92

After divide() call
 science_marks  english_marks
0          13.5           47.0
1          30.0           46.0
```

Function application

Row or Column Wise Function Application: apply()

apply() function performs the operation over either row wise or column wise data

e.g. of **Row wise Function in python pandas : Apply()**

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

d = {'science_marks':pd.Series([22,55]),
   'english_marks':pd.Series([89,87])}
df = pd.DataFrame(d)
print(df)
r=df.apply(np.mean,axis=1)
print (r)
```

OUTPUT

```
  science_marks  english_marks
0             22             89
1             55             87
0    55.5
1    71.0
dtype: float64
```

Function application

e.g. of **Column wise Function in python pandas : Apply()**

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

d = {'science_marks':pd.Series([22,55]),
   'english_marks':pd.Series([89,87])}
df = pd.DataFrame(d)
print(df)
r=df.apply(np.mean,axis=0)
print (r)
```

OUTPUT

```
 science_marks  english_marks
0             22             89
1             55             87
science_marks    38.5
english_marks    88.0
dtype: float64
```

Function application

Element wise Function Application in python pandas: applymap()

applymap() Function performs the specified operation for all the elements the dataframe.

e.g.program

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

d = {'science_marks':pd.Series([22,55]),
   'english_marks':pd.Series([89,87])}
df = pd.DataFrame(d)
print(df)
r=df.applymap(lambda x:x+2)
print (r)
```

OUTPUT

```
 science_marks  english_marks
0             22             89
1             55             87
   science_marks  english_marks
0             24             91
1             57             89
```

Function application

aggregation (group by) in pandas –

Data aggregation- Aggregation is the process of finding the values of a dataset (or a subset of it) into one single value. Let us go through the DataFrame like…

DATA OF DATAFRAME

	name	age	weight	height	runsscored
0	vishal	15	51	5.1	55
1	anil	16	48	5.2	25
2	mayur	15	49	5.1	71
3	viraj	17	51	5.3	53
4	mahesh	16	48	5.1	51

…then a simple aggregation method is to calculate the sum of the runsscored, which is 55+25+71+53+51=255. Or a different aggregation method would be to count the number of the name, which is 5. So the aggregation is not too complicated. Let's see the rest in practice…

Function application

aggregation (group by) in pandas –

Pandas Data Aggregation #1: .count() –

e.g.program

```
from collections import OrderedDict
from pandas import DataFrame
import pandas as pd
import numpy as np

table = OrderedDict((
    ("name", ['vishal', 'anil', 'mayur', 'viraj','mahesh']),
    ('age',[15, 16, 15, 17,16]),
    ('weight',  [51, 48, 49, 51,48]),
    ('height',   [5.1, 5.2, 5.1, 5.3,5.1]),
    ('runsscored',   [55,25, 71, 53,51])
))
d = DataFrame(table)
print("DATA OF DATAFRAME")
print(d)
print(d.count())
print("total names in dataframe are",d.name.count())
```

#note:- count() function can be used directly with dataframe or with the specific field of data frame

OUTPUT

```
DATA OF DATAFRAME
     name  age  weight  height  runsscored
0  vishal   15      51     5.1          55
1    anil   16      48     5.2          25
2   mayur   15      49     5.1          71
3   viraj   17      51     5.3          53
4  mahesh   16      48     5.1          51
name          5
age           5
weight        5
height        5
runsscored    5
dtype: int64
total names in dataframe are 5
```

Function application

aggregation (group by) in pandas –
Pandas Data Aggregation #2: .sum() –
e.g.program

```
from collections import OrderedDict
from pandas import DataFrame
import pandas as pd
import numpy as np
table = OrderedDict((
    ("name", ['vishal', 'anil', 'mayur', 'viraj','mahesh']),
    ('age',[15, 16, 15, 17,16]),
    ('weight',  [51, 48, 49, 51,48]),
    ('height',   [5.1, 5.2, 5.1, 5.3,5.1]),
    ('runsscored',   [55,25, 71, 53,51])
))
d = DataFrame(table)
print("DATA OF DATAFRAME")
print(d)
print(d.sum())
print("sum of score",d.runsscored.sum())
#note:- sum() function concates the string if field contain string value,otherwise sums the numbers . it can be used directly with dataframe or with the field of data frame
```

OUTPUT

```
DATA OF DATAFRAME
     name  age  weight  height  runsscored
0   vishal   15      51     5.1          55
1    anil    16      48     5.2          25
2   mayur    15      49     5.1          71
3   viraj    17      51     5.3          53
4  mahesh    16      48     5.1          51
name          vishalanilmayurvirajmahesh
age                                   79
weight                               247
height                              25.8
runsscored                           255
dtype: object
sum of score 255
```

Function application

aggregation (group by) in pandas –
Pandas Data Aggregation #3 and #4:
.min() and .max()
e.g.program

```
from collections import OrderedDict
from pandas import DataFrame
import pandas as pd
import numpy as np

table = OrderedDict((
    ("name", ['vishal', 'anil', 'mayur', 'viraj','mahesh']),
    ('age',[15, 16, 15, 17,16]),
    ('weight',  [51, 48, 49, 51,48]),
    ('height',   [5.1, 5.2, 5.1, 5.3,5.1]),
    ('runsscored',   [55,25, 71, 53,51])
))
d = DataFrame(table)
print("DATA OF DATAFRAME")
print(d)
print(d.max())
print("max score is",d.runsscored.max())
nm=d.name[d['runsscored'].idxmax()]
print("maximum scorer is ",nm)
#note:- in above program only max() function is used,on need min() can be used.idmax returns the index of maximum scorer ,which is further used to extract that name
```

```
OUTPUT
DATA OF DATAFRAME
     name  age  weight  height  runsscored
0  vishal    15      51     5.1          55
1    anil    16      48     5.2          25
2   mayur    15      49     5.1          71
3   viraj    17      51     5.3          53
4  mahesh    16      48     5.1          51
name         vishal
age              17
weight           51
height          5.3
runsscored       71
dtype: object
max score is 71
maximum scorer is  mayur
```

Function application

aggregation (group by) in pandas –
Pandas Data aggregation #5 and #6:
.mean() and .median()
e.g.program

```
from collections import OrderedDict
from pandas import DataFrame
import pandas as pd
import numpy as np

table = OrderedDict((
    ("name", ['vishal', 'anil', 'mayur', 'viraj','mahesh']),
    ('age',[15, 16, 15, 17,16]),
    ('weight',  [51, 48, 49, 51,48]),
    ('height',   [5.1, 5.2, 5.1, 5.3,5.1]),
    ('runsscored',   [55,25, 71, 53,51])
))
d = DataFrame(table)
print("DATA OF DATAFRAME")
print(d)
print("mean value of the score is",d.runsscored.mean())
print("mean value of the score is",d.runsscored.median())
```

OUTPUT

```
DATA OF DATAFRAME
     name  age  weight  height  runsscored
0  vishal   15      51     5.1          55
1    anil   16      48     5.2          25
2   mayur   15      49     5.1          71
3   viraj   17      51     5.3          53
4  mahesh   16      48     5.1          51
mean value of the score is 51.0
mean value of the score is 53.0
```

Function application

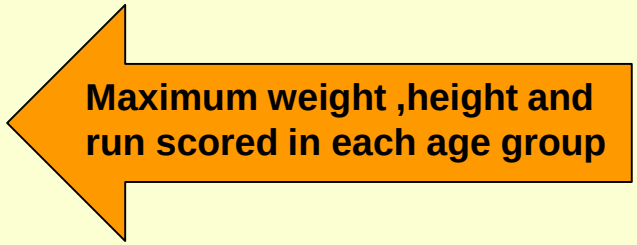
Grouping in pandas

For Data Analysis we will probably do segmentations many times. For instance, it's nice to know the max for all age groups, then grouping is to be done for each age value(group).
e.g. given below.

DATA OF DATAFRAME

	name	age	weight	height	runsscored
0	vishal	15	51	5.1	55
1	anil	16	48	5.2	25
2	mayur	15	49	5.1	71
3	viraj	17	51	5.3	53
4	mahesh	16	48	5.1	51

age	name	weight	height	runsscored
15	vishal	51	5.1	71
16	mahesh	48	5.2	51
17	viraj	51	5.3	53



Function application

Grouping in pandas

e.g.program.

```
from collections import OrderedDict
from pandas import DataFrame
import pandas as pd
import numpy as np

table = OrderedDict((
    ("name", ['vishal', 'anil', 'mayur', 'viraj','mahesh']),
    ('age',[15, 16, 15, 17,16]),
    ('weight',  [51, 48, 49, 51,48]),
    ('height',   [5.1, 5.2, 5.1, 5.3,5.1]),
    ('runsscored',   [55,25, 71, 53,51])
))
d = DataFrame(table)
print("DATA OF DATAFRAME")
print(d)
print(d.groupby('age').max())
```

```
# for individual column value  we can
use stmt like print(d.groupby('age').max().name)
```

```
OUTPUT
DATA OF DATAFRAME
     name  age  weight  height  runsscored
0  vishal  15      51     5.1          55
1    anil  16      48     5.2          25
2   mayur  15      49     5.1          71
3   viraj  17      51     5.3          53
4  mahesh  16      48     5.1          51
       name  weight  height  runsscored
age
15   vishal      51     5.1          71
16   mahesh      48     5.2          51
17    viraj      51     5.3          53
```

Function application

Transform –

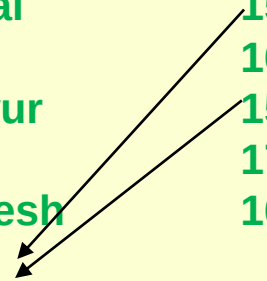
Transform is an operation used in conjunction with groupby.It is used in given pattern.

Dataframe -> grouping -> aggregate function on each group value -> then transform that value in each group value.

e.g.

DATA OF DATAFRAME

	name	age	weight	height	runsscored
0	vishal	15	51	5.1	55
1	anil	16	48	5.2	25
2	mayur	15	49	5.1	71
3	viraj	17	51	5.3	53
4	mahesh	16	48	5.1	51



0	126
1	76
2	126
3	53
4	76

In above example sum of score of each age group is applied over in order of age.

Function application

Transform –

e.g. program –

```
from collections import OrderedDict
from pandas import DataFrame
import pandas as pd
import numpy as np

table = OrderedDict((
    ("name", ['vishal', 'anil', 'mayur', 'viraj','mahesh']),
    ('age',[15, 16, 15, 17,16]),
    ('weight',  [51, 48, 49, 51,48]),
    ('height',   [5.1, 5.2, 5.1, 5.3,5.1]),
    ('runsscored',   [55,25, 71, 53,51])
))
d = DataFrame(table)
print("DATA OF DATAFRAME")
print(d)
print(d.groupby('age')["runsscored"].transform('sum'))
```

OUTPUT

DATA OF DATAFRAME

	name	age	weight	height	runsscored
0	vishal	15	51	5.1	55
1	anil	16	48	5.2	25
2	mayur	15	49	5.1	71
3	viraj	17	51	5.3	53
4	mahesh	16	48	5.1	51

0	126
1	76
2	126
3	53
4	76