SANJEEV SHARMA

Informatics Practices(New)

CLASS XII Code No. 065 -2019-20

Unit 1: Data Handling (DH-2)

What is pandas?

Pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

Python with pandas is in use in a wide variety of academic and commercial domains, including Finance, Neuroscience, Economics, Statistics, Advertising, Web Analytics, and more.

What problem does pandas solve?

It enables us to carry out our entire data analysis workflow in Python. Combined with the excellent IPython toolkit and other libraries, the environment for doing data analysis in Python excels in performance, productivity, and the ability to collaborate.

Some of the Highlights of Python pandas

1. A fast and efficient DataFrame object for data manipulation with integrated indexing.
2. Tools for reading and writing data between in-memory data structures and different formats: CSV and text files, Microsoft Excel, SQL databases etc.
3. Flexible reshaping and pivoting of data sets

Installing pandas

The simplest way to install not only pandas, but Python and the most popular packages that is with Anaconda, a cross-platform (Linux, Mac OS X, Windows) Python distribution for data analytics and scientific computing. After running the installer, the user will have access to

pandas and the rest of the stack without needing to install anything else, and without needing to wait for any software to be compiled.

Installation instructions for Anaconda can be found here.

Another advantage to installing Anaconda is that you don't need admin rights to install it. Anaconda can install in the user's home directory, which makes it trivial to delete Anaconda if you decide (just delete that folder).

Note: Each time we need to use pandas in our python program we need to write a line of code at the top of the program:

```
import pandas as <identifier_name>
```

Above statement will import the pandas library to our program.

We will use two different pandas libraries in in our programs

1. Series
2. DataFrames

pandas Series

Series is a one-dimensional labeled array capable of holding any data type (integers, strings, floating point numbers, Python objects, etc.). The axis labels are collectively referred to as the **index**. The basic method to create a Series is to call:

```
import pandas as <identifier name>

<Series_name> = <identifier name>.Series(data, index=index)
```

Data can be many different things:

- a Python dict
- a Python list
- a Python tuple

The passed index is a list of axis labels.

Step by Step method to create a pandas Series

Step 1

Suppose we have a list of games created with following python codes:

```
games_list = ['Cricket', 'Volleyball', 'Judo', 'Hockey']
```

Step 2

Now we create a pandas Series with above list

```
# Python script to generate a Series object from List
import pandas as ps
games_list = ['Cricket', 'Volleyball', 'Judo', 'Hockey']
s= ps.Series(games_list)
print(s)
```

OUTPUT

```
0 Cricket
1 Volleyball
2 Judo
3 Hockey
dtype: object
```

In the above output 0,1,2,3 are the indexes of list values. We can also create our own index for each value. Let us create another series with the same values with our own index values:

```
# Python script to generate a Series object from List using custom Index
import pandas as pd
games_list = ['Cricket', 'Volleyball', 'Judo', 'Hockey']
s= pd.Series(games_list, index =['G1','G2','G3','G4'])
print(s)
```

OUTPUT

```
G1    CRICKET
G2    VOLLEYBALL
G3     JUDO
G4      HOCKEY
dtype: object
```

In the above output Game_1, Game_2, Game_3, Game_4 are our own created indexes of list values.

In the similar manner we can create pandas Series with different data (tuple, dictionary, Object) etc.

Now we will create a Series with a Dictionary

Suppose we have a dictionary of games created with the following Python codes:

```
d = {'Cricket': 1, 'Volleyball': 2, 'Judo': 3 , 'Hockey':4}
```

Now we create a pandas Series with above dictionary

```
# Python script to generate a Dictionary Object
import pandas as pd
games_dict =  {'Cricket': 1, 'Volleyball': 2, 'Judo': 3 , 'Hockey':4}
s= pd.Series(games_dict)
print(s)
```

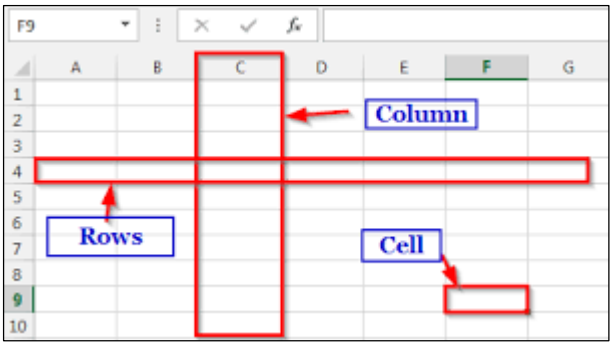
OUTPUT

```
Cricket 1
Volleyball 2
Judo 3
Hockey 4
Dtype : int64
```

The Python Pandas DataFrame

DataFrame is a Two-dimensional size-mutable, potentially heterogeneous tabular data structure. Tabular data structure has rows and columns. DataFrame is a way to represent and work with tabular data.

Pandas DataFrame is similar to excel sheet and looks like this



How to create a Pandas DataFrame?

In the real world, a Panda DataFrame will be created by loading the datasets from the permanent storage, including but not limited to excel, csv and MySQL database.

First we will use Python Data Structures (Dictionary and list) to create DataFrame.

Using Python Dictionary to create a DataFrame object

```
name_dict = {  'name' : ["Anita", "Sajal", "Ayaan", "Abhey"],
'age' : [14,32, 3, 6] }
```

If we print this dictionary using print(name_dict) command, it will show us the output like this:

```
{'name': ['Anita', 'Sajal', 'Ayaan', 'Abhey'], 'age': [14, 32, 3, 6]}
```

We can create a Pandas DataFrame out of this dictionary

```
# Python script to generate a Dictionary Object and print using variable
import pandas as pd
name_dict = {
    'Name' : ["Anita", "Sajal", "Ayaan", "Abhey"],
    'Age' : [14,32, 4, 6]
    }
df = pd.DataFrame(name_dict)
print(df)
```

Output

```
    Name     Age
0  Anita      14
1  Sajal      15
2  Ayaan       4
3  Abhey       6
```

As you can see the output generated for the DataFrame object is look similar to what we have seen in the excel sheet as. Only difference is that the default index value for the first row is 0 in DataFrame whereas in excel sheet this value is 1. We can also customize this index value as per our need.

Note: A side effect of dictionary is that when accessing the same dictionary at two separate times, the order in which the information is returned by the does not remained constant.

One more example of DataFrame with customize index value

```
# Python script to generate a Dictionary Object with custom index
import pandas as pd
name_dict = {
    'Name' : ["Anita", "Sajal", "Ayaan", "Abhey"],
    'Age' : [14,32, 4, 6]      }
df = pd.DataFrame(name_dict , index=[1,2,3,4])
print(df)
```

Output

```
    Name     Age
1  Anita      14
2  Sajal      15
3  Ayaan       4
4  Abhey       6
```

In the preceding output the index values start from 1 instead of 0

Viewing the Data of a DataFrame

To selectively view the rows, we can use head(...) and tail(...) functions, which by default give first or last five rows (if no input is provided), otherwise shows specific number of rows from top or bottom

Here is how it displays the contents

```
df.head()   # Displays first Five Rows
df.tails()   # Displays last Five Rows
print(df.head(2)) # Displays first Two Rows
print(df.tail(1)) #Displays last One Row
print(df.head(-2)) #Displays all rows except last two rows
print(df.tail(-1)) #Displays all rows except first row
```

Advance operations on Data Frames:

Pivoting:

	A	B	C	D	E	F
1	CBSE DUTY CHART 2019				PIVOT CHART	
2	SR NO	NAME OF INVIGILATOR	DATE OF DUTY	AMOUNT PAYABLE	INVIGILATOR	TOTAL AMOUNT PAYABLE
3	1	RAJESH DOGRA	2/3/2019	550	NARESH	1650
4	2	RAJEST KUMAR	2/3/2019	550	2/3/2019	550
5	3	NAVNEET	2/3/2019	550	2/4/2019	550
6	4	SANJEEV	2/3/2019	550	9/3/2019	550
7	5	SHIKHA	2/3/2019	550	NAVNEET	1650
8	6	NARESH	2/3/2019	550	28/4/2019	550
9	7	RC THAKUR	2/3/2019	550	2/3/2019	550
10	8	RAJESH DOGRA	5/3/2019	550	5/3/2019	550
11	9	RAJEST KUMAR	5/3/2019	550	RAJESH DOGRA	1650
12	10	NAVNEET	5/3/2019	550	14/3/2019	550
13	11	SANJEEV	5/3/2019	550	2/3/2019	550
14	12	SHIKHA	9/3/2019	550	5/3/2019	550
15	13	NARESH	9/3/2019	550	RAJEST KUMAR	1650
16	14	RC THAKUR	9/3/2019	550	14/3/2019	550
17	15	RC THAKUR	14/3/2019	550	2/3/2019	550
18	16	RAJESH DOGRA	14/3/2019	550	5/3/2019	550
19	17	RAJEST KUMAR	14/3/2019	550	RC THAKUR	1650
20	18	SHIKHA	28/4/2019	550	14/3/2019	550
21	19	NAVNEET	28/4/2019	550	2/3/2019	550
22	20	SHIKHA	2/4/2019	550	9/3/2019	550
23	21	NARESH	2/4/2019	550	SANJEEV	1100
24					2/3/2019	550
25					5/3/2019	550
26					SHIKHA	2200
27					28/4/2019	550
28					2/3/2019	550
29					2/4/2019	550
30					9/3/2019	550
31					Grand Total	11550

Sample Pivot chart created in Excel

A Pivot Table is an interactive way to quickly summarize large amounts of data. We can use a Pivot Table to analyse numerical data in detail, and answer unanticipated questions about our data. A PivotTable is especially designed for:

1. Querying large amounts of data in many user-friendly ways.
2. Expanding and collapsing levels of data to focus your results.
3. Filtering, sorting, grouping, and conditionally formatting the most useful and interesting subset of data enabling you to focus on just the information you want.

Creating Pivoting Tables with pandas' DataFrame

Pivot Tables in pandas

With pandas' pivot tables we can create a spreadsheet-style pivot table using DataFrame.

Steps to create a pandas' pivot table

Step 1

Create a DataFrame using Dictionary or any other sequence

Step 2

Use previously created DataFrame to generate a Pivot Table

Step 3

Print the Pivot Table

Example 1:

```
# Pyhton script demonstrating the use of  pivot_table() method
import pandas as pd
name_dict = {
    'INVIGILATOR' : ["Rajesh", "Naveen","Anil","Naveen","Rajesh"],
   'AMOUNT' : [550,550,550,550,550],
      }
df = pd.DataFrame(name_dict )
print(df)
pd.pivot_table(df, index = ['INVIGILATOR'],aggfunc='sum')
```

Output

```
      INVIGILATOR  AMOUNT
0       Rajesh          550
```

```
1       Naveen          550
2       Anil            550
3       Naveen          550
4       Rajesh          550
```

Output in pivot table form

```
INVIGILATOR         AMOUNT
Anil                550
Naveen              1100
Rajesh              1100
```

Example 2:

```
# Pyhton script demonstrating the use of  pivot_table() method
import pandas as pd
sale_dict = {
 'ITEM_NAME' : ["NOTEBOOK", "PEN","INKPEN","NOTEBOOK","PEN"],
    'AMOUNT' : [100,50,30,100,50],
    'QUANTITY' :[2,5,3,3,5]
       }
df = pd.DataFrame(sale_dict )
print(df)
pd.pivot_table(df,     index     =     ['ITEM_NAME','AMOUNT','QUANTITY'],
aggfunc='sum')
```

Output :

```
  ITEM_NAME  AMOUNT  QUANTITY
0  NOTEBOOK     100         2
1     PEN        50         5
2    INKPEN      30         3
3  NOTEBOOK     100         3
```

```
4       PEN             50          5
```

Output in pivot table form

```
ITEM_NAME     AMOUNT  QUANTITY
INKPEN          30          3
NOTEBOOK        100         2
                            3
PEN             50          5
```

Descriptive Statistics

After data collection, we generally use different ways to summarise the data. Python pandas provide different methods to generate descriptive statistics. Some of the common methods are:

min, max, mode, mean, count, sum, median

Example 1:

```
#Total sales per employee
import pandas as pd
monthlysale = { 'Salesman' : ["Akshit", "Jaswant","Karan","Akshit", "Jaswant","Karan","Akshit", "Jaswant","Karan","Akshit", "Jaswant","Karan"],
    'Sales' : [1000,300,800,1000,500,60,1000,900,300,1000,900,50],
    'Quarter' :[1,1,1,2,2,2,3,3,3,4,4,4],
            'District': ['Kangra','Hamirpur','Kangra','Mandi','Hamirpur','Kangra','Kangra','Hamirpur','Mandi','Hamirpur','Hamirpur','Kangra']
    }
df = pd.DataFrame(monthlysale )
# Employee wise total sale:


pd.pivot_table(df, index = ['Salesman'], values = ['Sales'],aggfunc='sum')
```

Output:

```
Salesman   Sales
Akshit     4000
Jaswant    2600
Karan      1210
```

Example 2:

```
#Total sales Per District
import pandas as pd
monthlysale = { 'Salesman' : ["Akshit", "Jaswant","Karan","Akshit", "Jaswant","Karan","Akshit", "Jaswant","Karan","Akshit", "Jaswant","Karan"],
    'Sales' : [1000,300,800,1000,500,60,1000,900,300,1000,900,50],
    'Quarter' :[1,1,1,2,2,2,3,3,3,4,4,4],
                'District': ['Kangra','Hamirpur','Kangra','Mandi','Hamirpur','Kangra','Kangra','Hamirpur','Mandi','Hamirpur','Hamirpur','Kangra']
      }
df = pd.DataFrame(monthlysale )
# District wise total sale:

pd.pivot_table(df, index = ['District'], values = ['Sales'],aggfunc='sum')
```

Output:

```
District    Sales
Hamirpur 3600
Kangra    2910
Mandi      1300
```

Example 3:

```
#Total sales per employee and per district
```

```
import pandas as pd
monthlysale = { 'Salesman' : ["Akshit", "Jaswant","Karan","Akshit", "Jaswant","Karan","Akshit", "Jaswant","Karan","Akshit", "Jaswant","Karan"],
    'Sales' : [1000,300,800,1000,500,60,1000,900,300,1000,900,50],
    'Quarter' :[1,1,1,2,2,2,3,3,3,4,4,4],
                'District': ['Kangra','Hamirpur','Kangra','Mandi','Hamirpur','Kangra','Kangra','Hamirpur','Mandi','Hamirpur','Hamirpur','Kangra']
        }
df = pd.DataFrame(monthlysale )
# Employee and district wise total sale:

pd.pivot_table(df, index = ['Salesman','District'], values = ['Sales'],aggfunc='sum')
```

Output:

```
Salesman District    Sales
Akshit      Hamirpur 1000
Kangra     2000
Mandi      1000
Jaswant   Hamirpur 2600
Karan      Kangra    910
Mandi      300
```

Example 4:

Maximum sales District wise

```
import pandas as pd
```

```
monthlysale = { 'Salesman' : ["Akshit", "Jaswant","Karan","Akshit",
"Jaswant","Karan","Akshit", "Jaswant","Karan","Akshit",
"Jaswant","Karan"],
    'Sales' : [1000,300,800,1000,500,60,1200,900,1300,1000,900,50],
    'Quarter' :[1,1,1,2,2,2,3,3,3,4,4,4],
     'District':
['Kangra','Hamirpur','Kangra','Mandi','Hamirpur','Kangra','Kangra','Hami
rpur','Mandi','Hamirpur','Hamirpur','Kangra']
      }
df = pd.DataFrame(monthlysale )
# Maximum sale:
pd.pivot_table(df, index = ['District'], values = ['Sales'],aggfunc='max')
```

Output:

```
District     Sales
Hamirpur 1000
Kangra      1200
Mandi        1300
```

Example 5:

```
# Minimum sale District Wise

import pandas as pd
monthlysale = { 'Salesman' : ["Akshit", "Jaswant","Karan","Akshit",
"Jaswant","Karan","Akshit", "Jaswant","Karan","Akshit",
"Jaswant","Karan"],
    'Sales' : [1000,300,800,1000,500,60,1000,900,300,1000,900,50],
    'Quarter' :[1,1,1,2,2,2,3,3,3,4,4,4],
```

```
            'District':
['Kangra','Hamirpur','Kangra','Mandi','Hamirpur','Kangra','Kangra','Hami
rpur','Mandi','Hamirpur','Hamirpur','Kangra']
    }
# Minimum Sale District wise:
pd.pivot_table(df, index = ['District'], values = ['Sales'],aggfunc='min')
```

Output:

```
District Sales
Hamirpur 300
Kangra     50
Mandi      300
```

Example 6:

```
# Median of sales Distirct wise
import pandas as pd
monthlysale = { 'Salesman' : ["Akshit", "Jaswant","Karan","Akshit",
"Jaswant","Karan","Akshit", "Jaswant","Karan","Akshit",
"Jaswant","Karan"],
   'Sales' : [1000,300,800,1000,500,60,1000,900,300,1000,900,50],
   'Quarter' :[1,1,1,2,2,2,3,3,3,4,4,4],
            'District':
['Kangra','Hamirpur','Kangra','Mandi','Hamirpur','Kangra','Kangra','Hami
rpur','Mandi','Hamirpur','Hamirpur','Kangra']
    }
df = pd.DataFrame(monthlysale )
# Median of sales Distirct wise
pd.pivot_table(df, index = ['District'], values = ['Sales'],aggfunc='median')
```

Output:

```
District Sales
Hamirpur 900
Kangra     800
Mandi      650
```

Complete Example:

```
# Maximum , Minimum , Mean, Mode  , Median  and Count of sales Salesman wise
import pandas as pd
print("\n")
print ( "Dataframe of Values\n")
print("\n")
monthlysale = { 'Salesman' : ["Akshit", "Jaswant","Karan","Akshit", "Jaswant","Karan","Akshit", "Jaswant","Karan","Akshit", "Jaswant","Karan"],
    'Sales' : [1000,300,800,1000,500,60,1000,900,300,1000,900,50],
    'Quarter' :[1,1,1,2,2,2,3,3,3,4,4,4],
                'District': ['Kangra','Hamirpur','Kangra','Mandi','Hamirpur','Kangra','Kangra','Hamirpur','Mandi','Hamirpur','Hamirpur','Kangra']
       }
df = pd.DataFrame(monthlysale )
# Use of mode() method of DataFrame
print("\n")
print ( "Use of mode() method of DataFrame")
print("\n")
print(df.mode())
print("\n")
print ( "Use of max,min,mean,median and count\n")
```

```
pd.pivot_table(df, index = ['Salesman'], values = ['Sales'],aggfunc=
['max','min','mean','median','count])
```

Output:

```
Use of mode() method of DataFrame
  Salesman   Sales  Quarter  District
0   Akshit  1000.0        1  Hamirpur
1  Jaswant     NaN        2    Kangra
2    Karan     NaN        3       NaN
3      NaN     NaN        4       NaN


Use of max, min, mean, median and count
            max  min   mean      median     count
          Sales Sales  Sales      Sales     Sales
Salesman
Akshit     1000 1000 1000.0      1000         4
Jaswant     900  300  650.0       700         4
Karan       800   50  302.5       180         4
```

Aggregation of DataFrame or Sequences

It is the process of turning the values of a dataset into one single value. The most common method to perform aggregation are max, min, sum, count. We have already covered all of these function in earlier examples.

Histogram using Python pandas :

Histogram is a diagram consisting of rectangles whose area is proportional to the frequency of a variable and whose width is equal to the class interval.

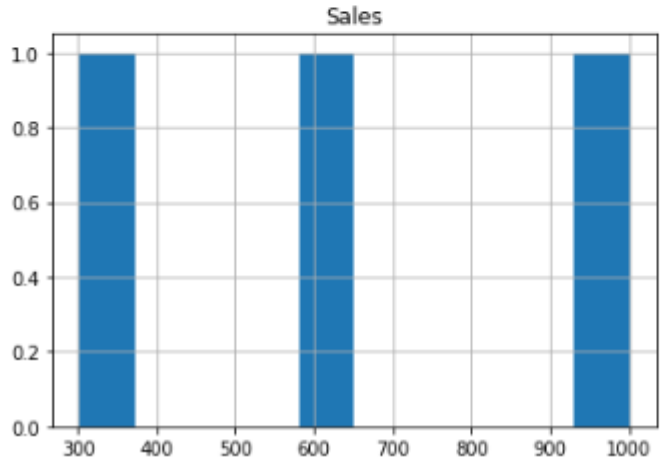
```
# Use of Histogram and hist() method
import pandas as pd
```

```
print("\n")
print ( "Dataframe of Values\n")
monthlysale = { 'Salesman' : ["Akshit", "Jaswant","Karan","Akshit",
"Jaswant","Karan","Akshit", "Jaswant","Karan","Akshit",
"Jaswant","Karan"],
    'Sales' : [1000,300,800,1000,500,60,1000,900,300,1000,900,50],
    'Quarter' :[1,1,1,2,2,2,3,3,3,4,4,4],
                'District':
['Kangra','Hamirpur','Kangra','Mandi','Hamirpur','Kangra','Kangra','Hami
rpur','Mandi','Hamirpur','Hamirpur','Kangra']
        }
df = pd.DataFrame(monthlysale )
print(df)
print("\n")
print ( "Use of Histogram hist() method\n")
pd.pivot_table(df, index = ['Salesman'], values = ['Sales']).hist()
```

Output:

```
Dataframe of Values

   Salesman  Sales  Quarter  District
0    Akshit   1000        1    Kangra
1   Jaswant    300        1  Hamirpur
2     Karan    800        1    Kangra
3    Akshit   1000        2     Mandi
4   Jaswant    500        2  Hamirpur
5     Karan     60        2    Kangra
6    Akshit   1000        3    Kangra
7   Jaswant    900        3  Hamirpur
8     Karan    300        3     Mandi
9    Akshit   1000        4  Hamirpur
10  Jaswant    900        4  Hamirpur
11    Karan     50        4    Kangra

Use of Histogram hist() method
```

Out[35]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7ff2e5932198>]],
      dtype=object)
```

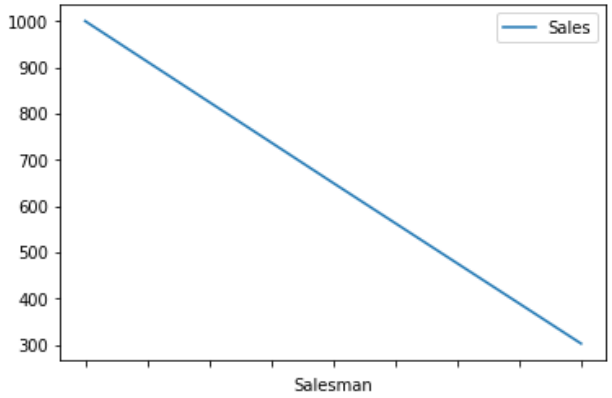



```
# Use of Histogram and plot() method

import pandas as pd

print("\n")

print ( "Dataframe of Values\n")

monthlysale = { 'Salesman' : ["Akshit", "Jaswant","Karan","Akshit",
"Jaswant","Karan","Akshit", "Jaswant","Karan","Akshit",
"Jaswant","Karan"],

    'Sales' : [1000,300,800,1000,500,60,1000,900,300,1000,900,50],

    'Quarter' :[1,1,1,2,2,2,3,3,3,4,4,4],
```

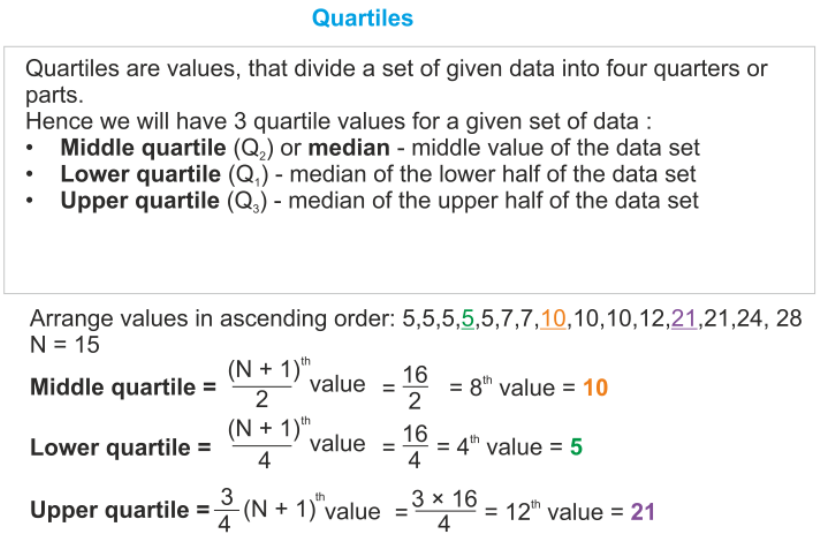
```
            'District':
['Kangra','Hamirpur','Kangra','Mandi','Hamirpur','Kangra','Kangra','Hami
rpur','Mandi','Hamirpur','Hamirpur','Kangra']
    }
df = pd.DataFrame(monthlysale )
print(df)
print("\n")
print ( "Use of Histogram plot() method\n")
pd.pivot_table(df, index = ['Salesman'], values = ['Sales']).plot()
```

Output:

```
Dataframe of Values

   Salesman  Sales  Quarter  District
0    Akshit   1000        1    Kangra
1   Jaswant    300        1  Hamirpur
2     Karan    800        1    Kangra
3    Akshit   1000        2     Mandi
4   Jaswant    500        2  Hamirpur
5     Karan     60        2    Kangra
6    Akshit   1000        3    Kangra
7   Jaswant    900        3  Hamirpur
8     Karan    300        3     Mandi
9    Akshit   1000        4  Hamirpur
10  Jaswant    900        4  Hamirpur
11    Karan     50        4    Kangra


Use of Histogram plot() method

Out[2]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff7d8271588>
```



Quantile



Variance Function in Python pandas

var() – Variance Function in python pandas is used to calculate variance of a given set of numbers, Variance of a Series , DataFrame etc.

```
# Use of Quantile and Variance  Method using Series Object
import pandas as pd
# Create a List of  Values
marks =  [ 34,76,45,90,32,56,93,56,24,12,54,10]
# Soritng of Listy
marks.sort()
print ( "Create a Series from the List values\n")
marks_series=pd.Series(marks)
print(marks_series)

print("\n")
print ( "Q1 , Q2 , Q3 and 100th  Quantiles \n")
print("Q2 quantile of marks_series : ",marks_series.quantile(.50))
```

```
print("Q1 quantile of marks_series : ",marks_series.quantile(.25))
print("Q3 quantile of marks_series : ",marks_series.quantile(.75))
print("100th quantile of marks_series : ",marks_series.quantile(.1))

# Calculate the variance of  Series\n")
print("\nVariance of marks_series : ",marks_series.var())
```

Output:

```
Create a Series from the List values
0     10
1     12
2     24
3     32
4     34
5     45
6     54
7     56
8     56
9     76
10    90
11    93
dtype: int64
Q1 , Q2 , Q3 and 100th  Quantiles
Q2 quantile of marks_series :  49.5
Q1 quantile of marks_series :  30.0
Q3 quantile of marks_series :  61.0
100th quantile of marks_series :  13.200000000000001
```

```
Variance of marks_series :  773.7272727272727
```

Sorting of DataFrame:

```
# Use of Sorting method with DataFrame
import pandas as pd


print("\n")
print ( "Dataframe of Values\n")
monthlysale = { 'Salesman' : ["Akshit", "Jaswant","Karan","Akshit",
"Jaswant","Karan","Akshit", "Jaswant","Karan","Akshit",
"Jaswant","Karan"],
    'Sales' : [1000,300,800,1000,500,60,1000,900,300,1000,900,50]
        }
df = pd.DataFrame(monthlysale )
print(df)
print("\n")
print ( "Sorting of DataFrame using Sales column in Descending
order\n")
sr=df.sort_index(ascending=False)
print(sr)
```

Output:

```
Dataframe of Values
   Salesman  Sales
0    Akshit   1000
1   Jaswant    300
2     Karan    800
3    Akshit   1000
```

```
4   Jaswant    500
5     Karan     60
6    Akshit   1000
7   Jaswant    900
8     Karan    300
9    Akshit   1000
10  Jaswant    900
11    Karan     50
Sorting of DataFrame using Sales column in Descending order
     Salesman  Sales
11    Karan     50
10  Jaswant    900
9    Akshit   1000
8     Karan    300
7   Jaswant    900
6    Akshit   1000
5     Karan     60
4   Jaswant    500
3    Akshit   1000
2     Karan    800
1   Jaswant    300
0    Akshit   1000
```

Function application:

If we want to apply user defined function or we want to use some other library's function Pyhton pandas provide mainly three important functions namely pipe() , Apply() , Applymap. In coming section we will see the use and working of all three functions one by one.

pipe() :

This function performs the custom operation for the entire dataframe. In below example we will using pipe() function to add value 2 to the entire dataframe.

```
# Use of pipe() function with DataFrame
import pandas as pd
import math
# User Defined Functioin
def new_value(dataframe):
    return dataframe.Sales * 2

print("\n")
print ( "Creating a Dataframe of Values with Dictionary \n")

monthlysale = { 'Salesman' : ["Akshit", "Jaswant","Karan"],
    'Sales' : [1000,300,800]
        }
df=pd.DataFrame(monthlysale)
print("The original Dataframe is \n")
print(df)

print("After applying the pipe() function to multiply the sales values with 2 \n")
df.pipe(new_value)
```

```
Creating a Dataframe of Values with Dictionary
The original Dataframe is
  Salesman  Sales
```

```
0   Akshit   1000
1  Jaswant    300
2    Karan    800
After applying the pipe() function to multiply the sales values with 2
```

Output:

```
0    2000
1     600
2    1600
```

apply():

This function performs the custom operation for either row wise or column wise.

```
# Use of apply() function with DataFrame
import pandas as pd
import numpy as np
print("\n")
print ( "Creating a Dataframe of Values with Dictionary \n")
monthlysale = { 'Salesman' : ["Akshit", "Jaswant","Karan"],
    'Sales' : [1000,300,800]
        }
df=pd.DataFrame(monthlysale)
print("The original DataFrame is \n")
print(df)
print("After applying the apply function to find the  Maximum value in DataFrame \n")
print(df.apply(np.max))
```

```
print("After applying the apply function to find the  Minimum value in DataFrame \n")

print(df.apply(np.max)
```

Output:

```
Creating a DataFrame of Values with Dictionary

The original DataFrame is

  Salesman  Sales

0   Akshit   1000

1  Jaswant    300

2    Karan    800

After applying the apply function to find the Maximum value in DataFrame


Salesman    Karan

Sales         1000

After applying the apply function to find the Minimum value in DataFrame

Salesman    Karan

Sales         1000
```

applymap():

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

```
# Use of applymap() function with DataFrame


import pandas as pd

print("\n")

print ( "Creating a Dataframe of Values with Dictionary \n")
```

```
monthlysale = { 'Salesman' : ["Akshit", "Jaswant","Karan"],
    'Sales_March' : [1000,300,800],'Sales_April' : [1500,400,1200]
        }
df=pd.DataFrame(monthlysale)
print("The original Dataframe is \n")
print(df)
print("After applying the applymap() function to multiply both Sales by 2 \n")
print(df.applymap(lambda x:x*2))
```

Output:

```
Creating a Dataframe of Values with Dictionary
The original Dataframe is
  Salesman  Sales_March  Sales_April
0   Akshit         1000         1500
1  Jaswant          300          400
2    Karan          800         1200
After applying the applymap() function to multiply both sales by 2
         Salesman  Sales_March  Sales_April
0    AkshitAkshit         2000         3000
1  JaswantJaswant          600          800
2      KaranKaran         1600         2400
```

Re-indexing:

The reindex() method in Pandas can be used to change the index of rows and columns of a Series or DataFrame.

```
# Use of reindex() function with DataFrame
import pandas as pd
```

```
print("\n")
df=pd.Series([1500,400,1200], index = [1,2,3])
print("The original Series is \n")
print(df)

print("After applying the reindex() function to change the order of index of the Series \n")
df_newindex=df.reindex(index = [3,1,2])
print(df_newindex)
```

Output:

```
The original Series is
1    1500
2     400
3    1200
After applying the reindex() function to change the order of index of the Series
3    1200
1    1500
2     400
```

rename():

Pandas rename() method is used to rename any index, column or row.

```
# Use of rename() function with DataFrame
import pandas as pd
print("\n")
print ( "Creating a Dataframe of Values with Dictionary \n")

monthlysale = { 'Salesman' : ["Akshit", "Jaswant","Karan"],
```

```
    'Sales_March' : [1000,300,800]       }

df=pd.DataFrame(monthlysale)
print("The original Dataframe is \n")
print(df)

print("After applying the rename() function to change the name of one column \n")

df.rename(columns={'Salesman': 'New_Salesman'},inplace=True) # inplace=True mean to make changes in original Dataframe
print(df)
print("No change in the Original Series if we omint inplace parameter\n")# Without using inplace parameter
df.rename(columns={'Sales_March': 'March_Sale'})
print(df)
```

Output:

```
Creating a Dataframe of Values with Dictionary
The original Dataframe is
  Salesman  Sales_March
0   Akshit         1000
1  Jaswant          300
2    Karan          800
After applying the rename() function to change the name of one column
  New_Salesman  Sales_March
0       Akshit         1000
1      Jaswant          300
2        Karan          800
```

```
No change in the Original Series if we omint inplace parameter


  New_Salesman  Sales_March
0       Akshit         1000
1      Jaswant          300
2        Karan          800
```

Group by Function:

By “group by” we are referring to a process involving one or more of the following steps:

- Splitting the data into groups based on some criteria.
- Applying a function to each group independently.

```
# Use of groupby() function with DataFrame
import pandas as pd


print("\n")
print ( "Dataframe of Values\n")


monthlysale = { 'Salesman' : ["Akshit", "Jaswant","Karan","Akshit",
"Jaswant","Karan","Akshit", "Jaswant","Karan","Akshit",
"Jaswant","Karan"],
    'Sales' : [1000,300,800,1000,500,60,1000,900,300,1000,900,50]
      }
df = pd.DataFrame(monthlysale )
print(df)
print ( "Grouping of DataFrame Salesman columns with sum()
function\n")
df1=df.groupby('Salesman').sum()
```

```
print(df1)


print ( "\nGrouping of DataFrame Salesman columns with count() function\n")
df2=df.groupby('Salesman').count()
print(df2)
```

Output:

```
Dataframe of Values
   Salesman  Sales
0    Akshit   1000
1   Jaswant    300
2     Karan    800
3    Akshit   1000
4   Jaswant    500
5     Karan     60
6    Akshit   1000
7   Jaswant    900
8     Karan    300
9    Akshit   1000
10  Jaswant    900
11    Karan     50
Grouping of DataFrame Salesman columns with sum() function
Salesman   Sales
Akshit      4000
Jaswant     2600
Karan       1210
Grouping of DataFrame Salesman columns with count() function
```

```
Salesman  Sales
Akshit        4
Jaswant       4
Karan         4
```

transform() Function:

This function is used to modify values of a Dataframe.

```
# Use of transform() function with DataFrame
import pandas as pd
print("\n")
print ( "Dataframe of Values\n")

monthlysale = { 'Salesman' : ["Akshit", "Jaswant","Karan","Akshit",
"Jaswant","Karan","Akshit", "Jaswant","Karan","Akshit",
"Jaswant","Karan"],
    'Sales' : [1000,300,800,1000,500,60,1000,900,300,1000,900,50]
        }
df = pd.DataFrame(monthlysale )
print(df)

print ( "\nUse of Transform function\n")
df2=df.transform(func = lambda x : x + 10)
print(df2)
print("DataFrame.transform() function has successfully added 10 to
each element of the given Dataframe.")
```

Output:

Dataframe of Values

	Salesman	Sales
0	Akshit	1000
1	Jaswant	300
2	Karan	800
3	Akshit	1000
4	Jaswant	500
5	Karan	60
6	Akshit	1000
7	Jaswant	900
8	Karan	300
9	Akshit	1000
10	Jaswant	900
11	Karan	50

Use of transform function

	Sales
0	1010
1	310
2	810
3	1010
4	510
5	70
6	1010
7	910
8	310
9	1010

10 910

11 60

DataFrame.transform() function has successfully added 10 to each element of the given Dataframe.

To Be Continue .....