Chapter 10 :

Informatics Practices

Class XI (As per CBSE Board)

New Syllabus 2018-19

Python Pandas

It is a most famous Python package for data science, which offers powerful and flexible data structures that make data analysis and manipulation easy.Pandas makes data importing and data analyzing much easier. Pandas builds on packages like NumPy and matplotlib to give us a single & convenient place for data analysis and visualization work.

Python Pandas

Basic Features of Pandas

1. Dataframe object help a lot in keeping track of our data.
2. With a pandas dataframe, we can have different data types (float, int, string, datetime, etc) all in one place
3. Pandas has built in functionality for like easy grouping & easy joins of data, rolling windows
4. Good IO capabilities; Easily pull data from a MySQL database directly into a data frame
5. With pandas, you can use patsy for R-style syntax in doing regressions.
6. Tools for loading data into in-memory data objects from different file formats.
7. Data alignment and integrated handling of missing data.
8. Reshaping and pivoting of data sets.
9. Label-based slicing, indexing and subsetting of large data sets.

Python Pandas

Python Pandas – Installation/Environment Setup

Pandas module doesn't come bundled with Standard Python.

1. If we install Anaconda Python package Pandas will be installed by default.

Steps for Anaconda installation & Use

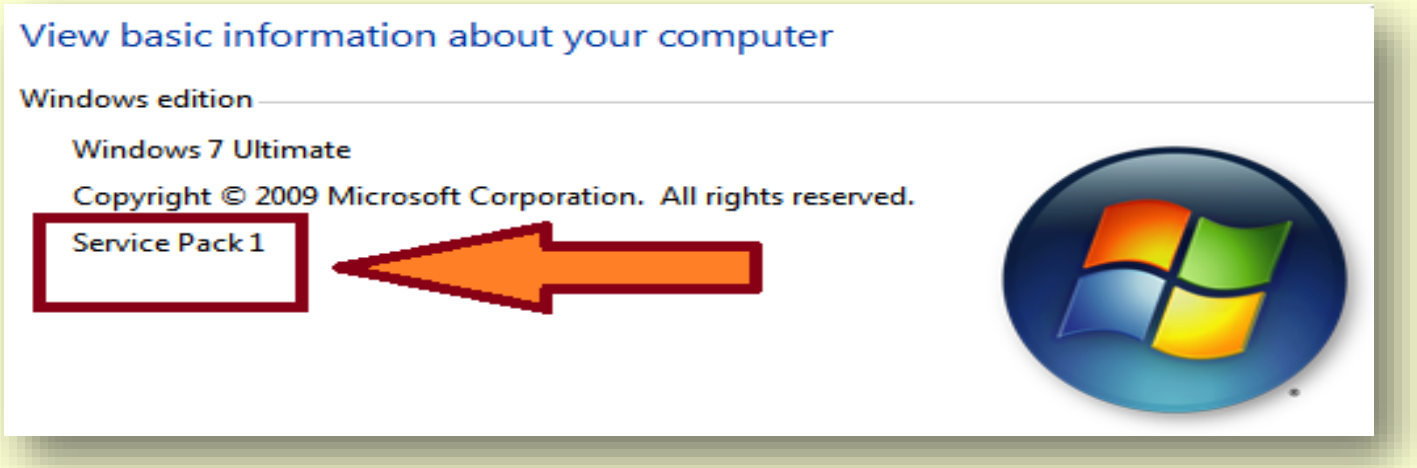
1. visit the site https://www.anaconda.com/download/
2. Download appropriate anaconda installer
3. After download install it.
4. During installation check for set path and all user
5. After installation start spyder utility of anaconda from start menu
6. Type `import pandas as pd` in left pane(temp.py)
7. Then run it.
8. If no error is show then it shows pandas is installed.
9. Like default temp.py we can create another .py file from new window option of file menu for new program.

Python Pandas

Python Pandas – Installation/Environment Setup

2 Pandas installation can be done in Standard Python distribution,using following steps.

1. There must be service pack installed on our computer if we are using windows.If it is not installed then we will not be able to install pandas in existing Standard Python(which is already installed).So install it first(google it).
2. We can check it through properties option of my computer icon.



3. Now install latest version(any one above 3.4) of python.

Python Pandas

Python Pandas – Installation/Environment Setup

2

4. Now move to script folder of python distribution in command prompt (through cmd command of windows).
5. Execute following commands in command prompt serially.

```
>pip install numpy
>pip install six
>pip install pandas
```

Wait after each command for installation

Now we will be able to use pandas in standard python distribution.

6. Type import pandas as pd in python (IDLE) shell.
7. If it executed without error(it means pandas is installed on your system)

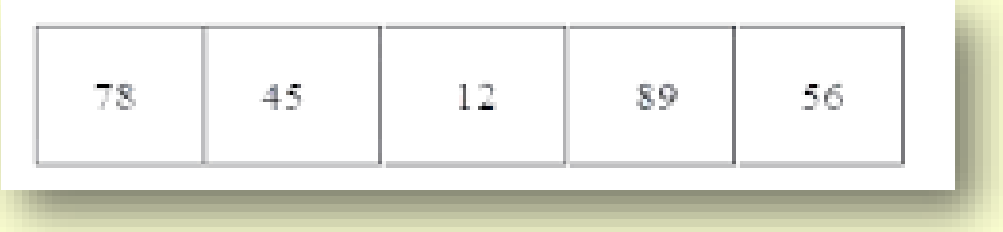
Python Pandas

Data Structures in Pandas

Two important data structures of pandas are–Series, DataFrame

1. Series

Series is like a one-dimensional array like structure with homogeneous data. For example, the following series is a collection of integers.



Basic feature of series are

- Homogeneous data
- Size Immutable
- Values of Data Mutable

Python Pandas

2. DataFrame

DataFrame is like a two-dimensional array with heterogeneous data.

SR. No.	Admn No	Student Name	Class	Section	Gender	Date Of Birth
1	001284	NIDHI MANDAL	I	A	Girl	07/08/2010
2	001285	SOUMYADIP BHATTACHARYA	I	A	Boy	24/02/2011
3	001286	SHREYAANG SHANDILYA	I	A	Boy	29/12/2010

Basic feature of DataFrame are

- ❖ Heterogeneous data
- ❖ Size Mutable
- ❖ Data Mutable

Python Pandas

Pandas Series

It is like one-dimensional array capable of holding data of any type (integer, string, float, python objects, etc.).
Series can be created using constructor.
Syntax :- pandas.Series(data, index, dtype, copy)

Series can be created using

1. Array
2. Dict
3. Scalar value or constant

Python Pandas

Pandas Series

Create an Empty Series

e.g.

```
import pandas as pseries
s = pseries.Series()
print(s)
```

Output

```
Series([], dtype: float64)
```

Python Pandas

Pandas Series

Create a Series from ndarray

Without index

e.g.

```
import pandas as pd1
import numpy as np1
data = np1.array(['a','b','c','d'])
s = pd1.Series(data)
print(s)
```

Output

```
0   a
1   b
2   c
3   d
dtype: object
```

Note : default index is starting from 0

With index position

e.g.

```
import pandas as p1
import numpy as np1
data = np1.array(['a','b','c','d'])
s = p1.Series(data,index=[100,101,102,103])
print(s)
```

Output

```
100   a
101   b
102   c
103   d
dtype: object
```

Note : index is starting from 100

Python Pandas

Pandas Series

Create a Series from dict

Eg.1(without index)

```
import pandas as pd1
import numpy as np1
data = {'a' : 0., 'b' : 1., 'c' : 2.}
s = pd1.Series(data)
print(s)
```

Output

```
a    0.0
b    1.0
c    2.0
dtype: float64
```

Eg.2 (with index)

```
import pandas as pd1
import numpy as np1
data = {'a' : 0., 'b' : 1., 'c' : 2.}
s = pd1.Series(data,index=['b','c','d','a'])
print(s)
```

Output

```
b    1.0
c    2.0
d    NaN
a    0.0
dtype: float64
```

Python Pandas

Pandas Series

Create a Series from Scalar

e.g

```
import pandas as pd1
import numpy as np1
s = pd1.Series(5, index=[0, 1, 2, 3])
print(s)
```

Output

```
0    5
1    5
2    5
3    5
dtype: int64
```

Note :- here 5 is repeated for 4 times (as per no of index)

Python Pandas

Pandas Series

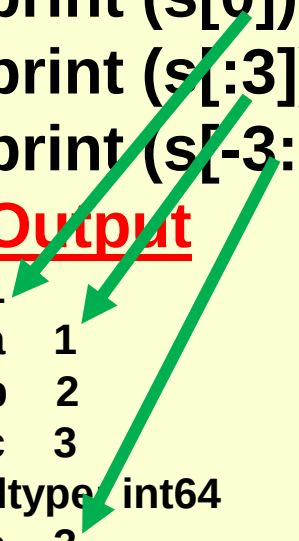
Accessing Data from Series with Position

e.g.

```
import pandas as pd1
s = pd1.Series([1,2,3,4,5],index = ['a','b','c','d','e'])
print (s[0])# for 0 index position
print (s[:3]) #for first 3 index values
print (s[-3:]) #for last 3 index values
```

Output

```
1
a   1
b   2
c   3
dtype: int64
c   3
d   4
e   5
dtype: int64
```



Python Pandas

Pandas Series

Retrieve Data Using Label (Index)

e.g.

```
import pandas as pd1
s = pd1.Series([1,2,3,4,5],index = ['a','b','c','d','e'])
print (s[['c','d']])
```

Output

```
c    3
d    4
dtype: int64
```

Python Pandas

Pandas Series

Head function

e.g

```
import pandas as pd1
s = pd1.Series([1,2,3,4,5],index = ['a','b','c','d','e'])
print (s.head(3))
```

Output

```
a   1
b   2
c   3
dtype: int64
```

Return first 3 elements

Python Pandas

Pandas Series

tail function

e.g

```
import pandas as pd1
s = pd1.Series([1,2,3,4,5],index = ['a','b','c','d','e'])
print (s.tail(3))
```

Output

```
c    3
d    4
e    5
dtype: int64
```

Return last 3 elements

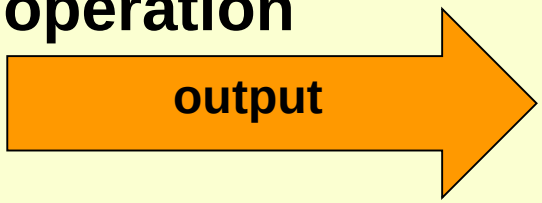
Python Pandas

Pandas Series

Vectorized operations with Series

e.g.

```
import pandas as pd1
s = pd1.Series([1,2,3])
t = pd1.Series([1,2,4])
u=s+t #addition operation
print (u)
u=s*t # multiplication operation
print (u)
```




```
0    2
1    4
2    7
dtype: int64

0     1
1     4
2    12
dtype: int64
```

Python Pandas

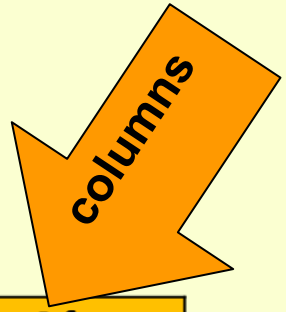
Pandas DataFrame

It is a two-dimensional data structure, just like any table (with rows & columns).

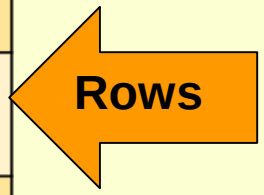
Basic Features of DataFrame

- Columns may be of different types
- Size can be changed(Mutable)
- Labeled axes (rows / columns)
- Arithmetic operations on rows and columns

Structure



SR. No.	Admn No	Student Name	Class	Section	Gender	Date Of Birth
1	001284	NIDHI MANDAL	I	A	Girl	07/08/2010
2	001285	SOUMYADIP BHATTACHARYA	I	A	Boy	24/02/2011
3	001286	SHREYAANG SHANDILYA	I	A	Boy	29/12/2010



It can be created using constructor

pandas.DataFrame(data, index, columns, dtype, copy)

Python Pandas

Pandas DataFrame

Create DataFrame

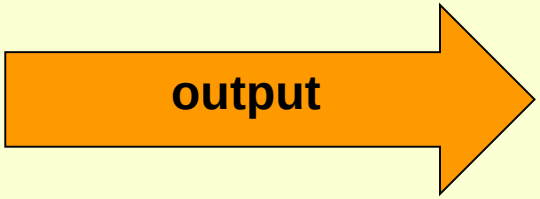
It can be created with followings

- **Lists**
- **dict**
- **Series**
- **Numpy ndarrays**
- **Another DataFrame**

Create an Empty DataFrame

e.g.

```
import pandas as pd1
df1 = pd1.DataFrame()
print(df1)
```




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

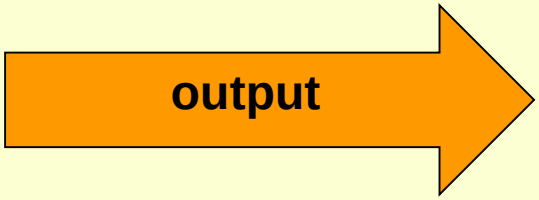
Python Pandas

Pandas DataFrame

Create a DataFrame from Lists

e.g.1

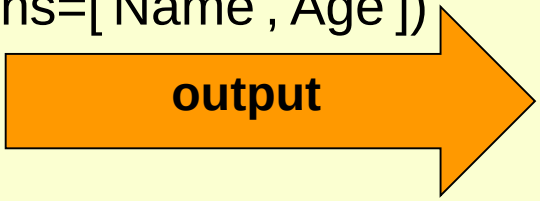
```
import pandas as pd1
data1 = [1,2,3,4,5]
df1 = pd1.DataFrame(data1)
print (df1)
```




```
   0
0  1
1  2
2  3
3  4
4  5
```

e.g.2

```
import pandas as pd1
data1 = [['Freya',10],['Mohak',12],['Dwivedi',13]]
df1 = pd1.DataFrame(data1,columns=['Name','Age'])
print (df1)
```




```
      Name  Age
0    Freya   10
1    Mohak   12
2  Dwivedi   13
```

Write below for numeric value as float

```
df1 = pd1.DataFrame(data,columns=['Name','Age'],dtype=float)
```

Python Pandas

Pandas DataFrame

Create a DataFrame from Dict of ndarrays / Lists

e.g.1

```
import pandas as pd1
data1 = {'Name':['Freya', 'Mohak'],'Age':[9,10]}
df1 = pd1.DataFrame(data1)
print (df1)
```

Output

```
    Name  Age
0   Freya    9
1   Mohak  10
```

Write below as 3rd statement in above prog for indexing

```
df1 = pd1.DataFrame(data1, index=['rank1','rank2','rank3','rank4'])
```

Python Pandas

Pandas DataFrame

Create a DataFrame from List of Dicts

e.g.1

```
import pandas as pd1
data1 = [{'x': 1, 'y': 2},{'x': 5, 'y': 4, 'z': 5}]
df1 = pd1.DataFrame(data1)
print (df1)
```

Output

```
   x  y   z
0  1  2  NaN
1  5  4  5.0
```

Write below as 3rd stmnt in above program for indexing

```
df = pd.DataFrame(data, index=['first', 'second'])
```

Python Pandas

Pandas DataFrame

Create a DataFrame from Dict of Series

e.g.1

```
import pandas as pd1
d1 = {'one' : pd1.Series([1, 2, 3], index=['a', 'b', 'c']),
      'two' : pd1.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}
df1 = pd1.DataFrame(d1)
print (df1)
```

Output

```
   one    two
a  1.0     1
b  2.0     2
c  3.0     3
d  NaN     4
```

Column Selection -> print (df ['one'])

Adding a new column by passing as Series: ->

```
df1['three']=pd1.Series([10,20,30],index=['a','b','c'])
```

Adding a new column using the existing columns values

```
df1['four']=df1['one']+df1['three']
```

Python Pandas

Pandas DataFrame

Column Deletion

```
del df1['one'] # Deleting the first column using DEL function
df.pop('two') #Deleting another column using POP function
```

Row Selection, Addition, and Deletion

```
#Selection by Label
import pandas as pd1
d1 = {'one' : pd1.Series([1, 2, 3], index=['a', 'b', 'c']),
    'two' : pd1.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}
df1 = pd1.DataFrame(d1)
print (df1.loc['b'])
```

Output

```
one 2.0
two 2.0
Name: b, dtype: float64
```

Python Pandas

Pandas DataFrame

```
#Selection by integer location
import pandas as pd1
d1 = {'one' : pd1.Series([1, 2, 3], index=['a', 'b', 'c']),
    'two' : pd1.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}
df1 = pd1.DataFrame(d1)
print (df1.iloc[2])
```

Output

```
one   3.0
two   3.0
Name: c, dtype: float64
```

Slice Rows : Multiple rows can be selected using ‘ : ’ operator.

```
print (df1[2:4])
```

Python Pandas

Pandas DataFrame

Addition of Rows

```
import pandas as pd1

df1 = pd1.DataFrame([[1, 2], [3, 4]], columns = ['a','b'])
df2 = pd1.DataFrame([[5, 6], [7, 8]], columns = ['a','b'])

df1 = df1.append(df2)
print (df1)
```

Deletion of Rows

```
# Drop rows with label 0
df1 = df1.drop(0)
```

Python Pandas

Pandas DataFrame

Iterate over rows in a dataframe

e.g.

```
import pandas as pd1
import numpy as np1
raw_data1 = {'name': ['freya', 'mohak'],
                                'age': [10, 1],
                                'favorite_color': ['pink', 'blue'],
                                'grade': [88, 92]}
df1 = pd1.DataFrame(raw_data1, columns = ['name', 'age',
'favorite_color', 'grade'])
for index, row in df1.iterrows():
    print (row["name"], row["age"])
```

Output

```
freya 10
mohak 1
```

Python Pandas

Pandas DataFrame

Using itertuples:

e.g.

```
for row in df1.itertuples(index=True, name='Pandas'):
    print (getattr(row, "name"), getattr(row, "age"))
```

Output

```
freya 10
mohak 1
```

modify the rows iterating over

```
def valuation_formula(x):
    return x * 0.5

df1['age_half'] = df1.apply(lambda row: valuation_formula(row['age']), axis=1)
```

Python Pandas

Pandas DataFrame

Binary operation over dataframe with series

e.g.

```
import pandas as pd
x = pd.DataFrame({0: [1,2,3], 1: [4,5,6], 2: [7,8,9] })
y = pd.Series([1, 2, 3])
new_x = x.add(y, axis=0)
print(new_x)
```

Output

```
   0   1   2
0  1   4   7
1  4  10  16
2  9  18  27
```

Python Pandas

Pandas DataFrame

Binary operation over dataframe with datafram

e.g.

```
import pandas as pd
x = pd.DataFrame({0: [1,2,3], 1: [4,5,6], 2: [7,8,9] })
y = pd.DataFrame({0: [1,2,3], 1: [4,5,6], 2: [7,8,9] })
new_x = x.add(y, axis=0)
print(new_x)
```

Output

```
   0   1   2
0  2   8  14
1  4  10  16
2   6  12  18
```

Note :- similarly we can use sub,mul,div functions

Python Pandas

Pandas DataFrame

Handle Right-Side Addition/Subtraction etc.

These functions(starting with ‘r’ like radd,rsub etc) are only called if the left operand does not support the corresponding operation. For instance, to evaluate the expression x-y, where y is an instance of a class that has an __rsub__() method, y.__rsub__(x) is called. Note that ternary pow() will not try calling __rpow__() (the coercion rules would become too complicated).

Python Pandas

Pandas DataFrame

Handle Right-Side Addition/Subtraction etc.

e.g.

```
class Commuter:
    def __init__(self, val):
        self.val = val
    def __add__(self, other):
        print ('add', self.val, other )
    def __radd__(self, other):
        print ('radd', self.val, other )
x = Commuter(88)
y = Commuter(99)
x + 1                   # __add__: instance + noninstance
1 + y                   # __radd__: noninstance + instance
x + y                   # __add__: instance + instance
```

Output

```
add 88 1
radd 99 1
add 88 <__main__.Commuter object at 0x02181370>
```

Python Pandas

Pandas DataFrame

Handle Right-Side Addition/Subtraction etc.

e.g.

```
class Commuter:
    def __init__(self, val):
        self.val = val
    def __add__(self, other):
        print ('add', self.val, other )
    def __radd__(self, other):
        print ('radd', self.val, other )
x = Commuter(88)
y = Commuter(99)
x + 1                   # __add__: instance + noninstance
1 + y                   # __radd__: noninstance + instance
x + y                   # __add__: instance + instance
```

Output

```
add 88 1
radd 99 1
add 88 <__main__.Commuter object at 0x02181370>
```

Python Pandas

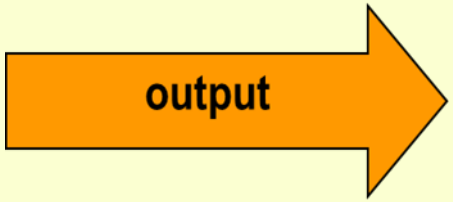
Pandas DataFrame

What does the term “broadcasting” mean in Pandas ?

The term broadcasting comes from numpy, simply put the output that will result when we perform operations between n-dimensional arrays (could be panels, dataframes, series) or scalar values.

e.g.

```
import pandas as pd
import numpy as np
s = pd.Series([1,2,3])
s=s*2
print(s)
```




```
0   2
1   4
2   6
dtype: int64
```

So what is technically happening here is that the scalar value has been broadcasted along the same dimensions of the Series.Same can be done in frame also.

Python Pandas

Pandas DataFrame

What does the term “matching” mean in Pandas ?

Extraction of matching values from series/dataframe.

e.g.

```
import pandas as pd
data = {'name': ['freya', 'mohak'],
        'age': [10,1]}
df = pd.DataFrame(data, columns = ['name', 'age'])
df=df[df['name'].str.contains('ya')]
print(df)
```

OUTPUT

```
   name  age
0  freya   10
```

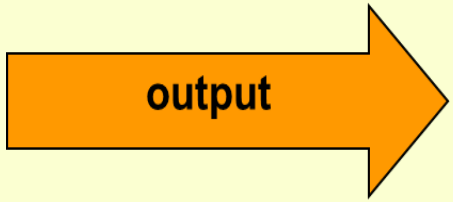
Python Pandas

Pandas DataFrame

Filling the missing data

Eg.

```
import pandas as pd
import numpy as np
raw_data = {'name': ['freya', 'mohak', 'rajesh'],
        'age': [42, np.nan, 36 ] }
df = pd.DataFrame(raw_data, columns = ['name',
'age'])
print(df)
df['age']=df['age'].fillna(0)
print(df)
```




```
   name   age
0  freya  42.0
1  mohak   NaN
2 rajesh  36.0
   name   age
0  freya  42.0
1  mohak   0.0
2 rajesh  36.0
```

In above e.g. age of mohak is filled with 0

Python Pandas

Pandas DataFrame

Comparing two series

e.g.

```
import pandas as pd
ds1 = pd.Series([2, 4, 6, 8, 10])
ds2 = pd.Series([1, 3, 5, 7, 10])
print("Greater than:")
print(ds1 > ds2) # compare two series
```

Output

```
Greater than:
0     True
1     True
2     True
3     True
4    False
dtype: bool
```

Python Pandas

Pandas DataFrame

Merging/combining dataframe

e.g.

```
import pandas as pd
left = pd.DataFrame({
        'id':[1,2],
        'Name': ['anil', 'vishal'],
        'subject_id':['sub1','sub2']})
right = pd.DataFrame(
        {'id':[1,2],
        'Name': ['sumer', 'salil'],
        'subject_id':['sub2','sub4']})
print (pd.merge(left,right,on='id'))
```

Output

```
   id  Name_x    subject_id_x Name_y subject_id_y
0   1    anil         sub1  sumer         sub2
1   2  vishal         sub2  salil         sub4
```

Python Pandas

Pandas DataFrame

Merging/combining dataframe(different styles)

```
pd.merge(left, right, on='subject_id', how='left')  #left join
pd.merge(left, right, on='subject_id', how='right') #right join
pd.merge(left, right, how='outer', on='subject_id') #outer join
pd.merge(left, right, on='subject_id', how='inner') # inner join
```