CLASS NOTES

Class: XII	**Date: 17-04-2021**
Subject: **Informatics Practices**	**Topic:** Chapter-1 Python Pandas - I

Some Facts :

A. PANDAS has derived its name from "**PAN**el **DA**ta **S**ystem".
B. The main author of Pandas is "**Wes McKinney**".
C. It is used as Python's library for data analysis.
D. It is an open source.
E. It offers high performance, easy to use data structure and data analysis tools.
F. In order to use Pandas we need to import by using "import pandas as pd".

Why PANDAS?

1. To read and write in many different data formats.
2. It can calculate in all the possible ways data is organized.
3. It can easily select subsets of data from bulky data.
4. It allows you to apply operation to independent groups within the data.
5. It supports reshaping of data into different forms.
6. Using its advance time-series functionality one can predict the future values.

Pandas data structure

Data structure : It refers to specialized way of storing data so as to apply a specific type of functionality on them. Out of the many data structure two basic data structures – Series and DataFrame are universally popular for their dependability.

Comparison between Series and DataFrame :

Property	**Series**	**DataFrame**
Dimensions	1-Dimentional	2-Dimentional
Type of Data	Homogeneous – All elements must be of same data type.	Heterogeneous – DataFrame objects can have different data types.
Mutability	Size- Immutable: Size of a series object, once created, cannot change.	Size – Mutable : We can drop or add the elements in an existing dataframe object.

Series Data Structure:

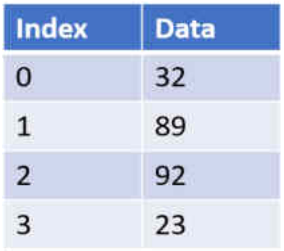
It represents a one dimensional array of indexed data.

It has two main components :

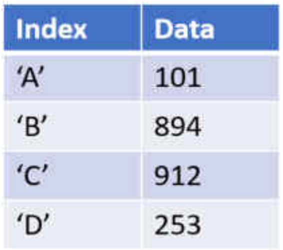
1. An array of actual data.
2. An associated array of indexes or data labels

The index is used to access individual data values.

Example:



Index	Data
0	32
1	89
2	92
3	23



Index	Data
'A'	101
'B'	894
'C'	912
'D'	253

Creating Series Object:

Make sure that you have imported pandas module.

import pandas as pd

To create an empty series just issue the following command:

<Series Object>=pandas.Series()

To create a non-empty Series object, you need to specify arguments for data and indexes as per following syntax :

<Series Object>=pd.Series(data, index=idx)

data: It can be a python sequence, an ndarray, a python dictionary or a scalar value.

idx: It is a valid numpy datatype.

Series() method with arguments:

Syntax is

<Series Object> = pandas.Series(data, index=idx)

1. **Creating series using List:**

```
Ser2 = pd.Series([12,23,34,45,67])
>>> print(Ser2)
0   12
1   23
2   34
3   45
4   67
```

Creating a series using range() method.

```
>>> ser3=pd.Series(range(4))
>>> print(ser3)
0   0
1   1
2   2
3   3
```

The default index can be changed also. Following example explains :

```
>>> ser3.index= ['One','Two','Three','Four']
>>> print(ser3)
One     0
Two     1
Three   2
Four    3
```

To create a series with explicitly defined index values.

```
>>> ser3=pd.Series([12,23,34,45,67], index = ['A','B','C','D','E'])
>>> print(ser3)
A   12
B   23
C   34
D   45
E   67
```

Accessing single or multiple values based on index from Series :

```
>>> ser3['B']
23
>>> ser3[['A','C','D']]
```

```
          A    12
          C    34
          D    45
>>> ser3[['B','D','A’]]
          B    23
          D    45
          A    12
```

To perform indexing, slicing and accessing data from a series:

```
          ser3 =
          >>> print(ser3[1])              To extract the second value
          23
          >>> print(ser3[:3])             To extract the first three values
          A    12
          B    23
          C    34
          >>> print(ser3[-3:])            To extract the last three values
          C    34
          D    45
          E    67
```

To create a series from a scalar or constant value:

```
          >>> ser4 = pd.Series(25 , index = ['a','b','c','d'])
          >>> print(ser4)
          a    25
          b    25
          c    25
          d    25
>>> ser5=pd.Series(25, index=[0,1,2,3,4])
>>> print(ser5)
          0    25
          1    25
          2    25
          3    25
          4    25
```

Creating a series with range() and for loop:

```
          >>> ser6=pd.Series(range(1,20,4), index=[x for x in 'abcde'])
          >>> print(ser6)
          a     1
          b     5
          c     9
          d    13
          e    17
```

Creating a series from two different list:

```
          >>> rollno=[1,2,3,4,5]
          >>> marks=[12,13,16,18,20]
          >>> ser7=pd.Series(marks, index=rollno)
          >>> print(ser7)
          1    12
          2    13
          3    16
          4    18
          5    20
```

Creating a series from Dictionary :

```
          >>> ser8=pd.Series({1:34,2:43,3:31,4:55})
          >>> print(ser8)
          1    34
          2    43
```

```
3    31
4    4 55
```

Extracting values from a series using head() and tail() functions:

```
>>> print(ser8.head(2))                First two values from the series
Rollno
1    34
2    43
>>> print(ser8.tail(2))                  Last two values from the series
Rollno
3    31
4    55
```

Extracting value using condition : Assume that we have a series ser7

```
>>> ser7
1    12
2    13
3    16
4    18
                      5 20
>>> ser7<17            Condition to extract the values below 17(Vectorised)
1     True
2     True
3     True
4    False
          5 False
```

```
>>> ser7[ser7<17]    Condition to extract the values below 17(Filter)
            1    12
            2    13
            3    16
```

To delete an element usinf drop() method:

```
>>> ser7
1    12
2    13
3    16
4    18
5    20
>>> ser7.drop(2)
1    12
3    16
4    18
5    20
```

Creating a series with range() and for loop:

```
>>> ser6=pd.Series(range(1,20,4), index=[x for x in 'abcde'])
>>> print(ser6)
a     1
b     5
c     9
d    13
e    17
```

Creating a series from two different list:

```
>>> rollno=[1,2,3,4,5]
>>> marks=[12,13,16,18,20]
>>> ser7=pd.Series(marks, index=rollno)
>>> print(ser7)
1    12
```

```
2   13
3   16
4   18
5   20
```

Extracting values from a series using head() and tail() functions:

```
>>> print(ser8.head(2))             First two values from the series
Rollno
1   34
2   43
>>> print(ser8.tail(2))              Last two values from the series
Rollno
3   31
4   55
```

Assignment Questions to be done in IP Notebook:

1. In Science Olympiad 2020 the total collection of fee in Rs. is 1200,3450,5210,2410 and 2870 from the sections A, B ,C, D and E. Write code to create a Series object that store the fee amount as the value and section name as the index.
2. People have contributed towards charity funds for the victim of COVID-19. The State-wise contribution are Rs. 132, 165,128,142 and 117 in lakh from the Indian states MP, UP, AP, HP and Delhi. Central government had announced that they will give the contribution amount double the collected amount state wise. write coding to make a Series object for the same.
3. Write a python code to create a Series object Temp1 that stores temperature of seven days in it. Take any random seven temperature.

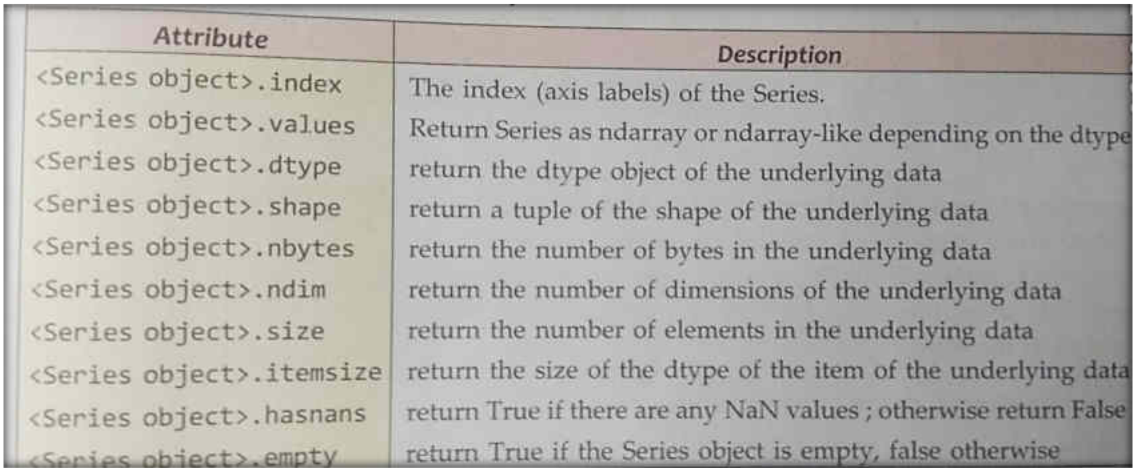
3. Write code to create a series object that stores total number of medals 450 to be won every year in Olympics held in the decade 2020-2030.

4. Write python code to distribute Rs. 500 crore for education in the budget equally for four quarters Q1,Q2,Q3 and Q4 of the year.

5. Create a Python Series which includes monthly expenditure for the 6 Months. Specify the datatype as float64.

6. A Python list namely player stores the names of cricket players i.e. Player[‘Rohit’, ‘Virat’, ‘Rahul’, ‘MSD’, ‘Raina’]. Another list Run_scr stores the runs scored by the players in overall one day cricket. Create a series in Python that stores Player as the index values and Run_scr as the data.

Read and Test the following ATTRIBUTES of SERIES:



Attribute	Description
<Series object>.index	The index (axis labels) of the Series.
<Series object>.values	Return Series as ndarray or ndarray-like depending on the dtype
<Series object>.dtype	return the dtype object of the underlying data
<Series object>.shape	return a tuple of the shape of the underlying data
<Series object>.nbytes	return the number of bytes in the underlying data
<Series object>.ndim	return the number of dimensions of the underlying data
<Series object>.size	return the number of elements in the underlying data
<Series object>.itemsize	return the size of the dtype of the item of the underlying data
<Series object>.hasnans	return True if there are any NaN values ; otherwise return False
<Series object>.empty	return True if the Series object is empty, false otherwise