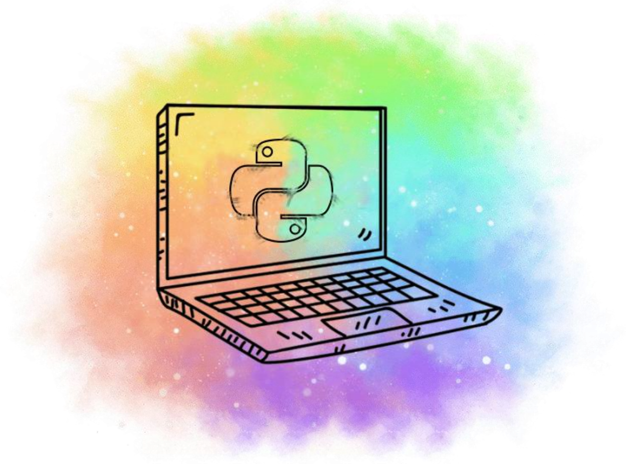


Divyanshu_Pal

Python Pandas -1

CBSE/Any Board Class 12

With **NOTES!**

Outline

- Using Pandas
- Pandas Data Structure
- Series Data Structure
- Dataframe Data Structure
- Creating and displaying dataframe
- Dataframe Attributes
- Selecting and Accessing data
- Adding/Modifying row's/column's values
- Deleting/Renaming rows/columns
- Dataframe Indexing and Boolean indexing

Using Pandas

Pandas is an open source, BSD library build for python. It offers High-performance, easy-to-use, data structures and data analysis tool.

syntax: `import pandas as <any alias name>`

example:

```
import pandas as pd
import pandas
import pandas as xyz
```

Pandas Data Structure

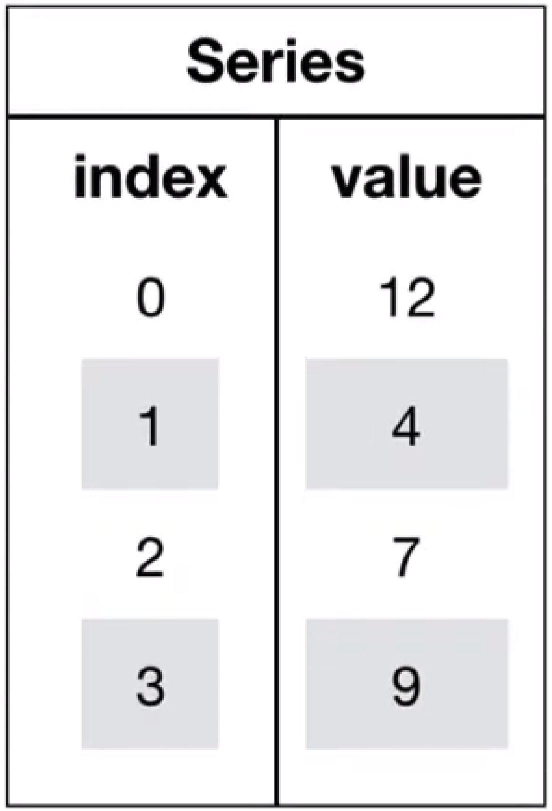
Data Structure: A *data structure* is a particular way of storing and organizing data in a computer to suite a specific purpose.

Data structures of Pandas:

1. Series
2. Dataframe
3. Panels(Not covered in syllabus)

Series

Series is a data structure in pandas. It contains 1-D array of indexed data.

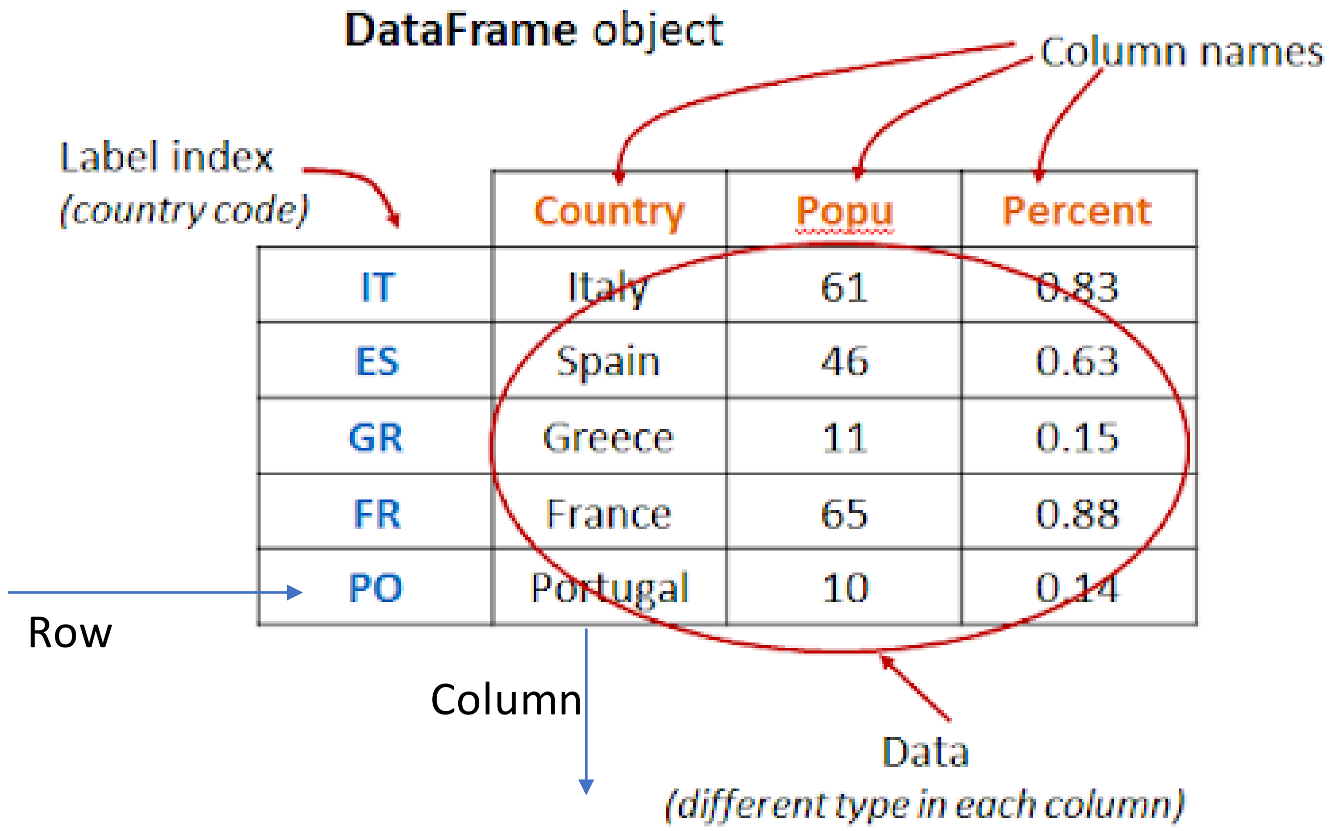


Series	
index	value
0	12
1	4
2	7
3	9

- 1-dimensional data structure
- Values are mapped to index
- Can have same index for various values
- Value mutable(value can be changed in the original object)
- Size immutable(new object will be created with new values)
- unique datatype

Data Frame

Is an another data structure of pandas, stores data in 2-D labelled array



	Country	Popu	Percent
IT	Italy	61	0.83
ES	Spain	46	0.63
GR	Greece	11	0.15
FR	France	65	0.88
PO	Portugal	10	0.14

- 2-Dimensional Data Structure.
- Value are stored in rows and columns.
- Cannot have same row-index and column-label.
- Value mutable(value can be changed in the original object).
- Size mutable(new object will not be created changes will be made in original one).
- Columns can have different datatypes.

Divyanshu_Pal

Creating a series object

Pandas module is already imported

Alias name if used

Syntax: <series object> = pandas.Series(data,index)

S must be capital

1. Creating Empty Series

```
S1 = pandas.Series()
```

2. Creating non-empty series object.

```
S2 = pandas.Series([3,4,7,4,9])
print(S1)
```

Output:

```
Series object 2:
 0   3
 1   4
 2   7
 3   4
 4   9
dtype: int64
```

Divyanshu_Pal

Ways of creating non-empty series object.

I. Specify data as python sequence

giving data using python sequence functions and characters such as range().

Here pd is used as alias name

```
import pandas as pd
s3 = pd.Series(range(0,6,2))
print(s3)
```

Here pd is used on the place of pandas

Output:

```
s3:
 0  0
 1  2
 2  4
 3  6
 dtype: int64
```

Divyanshu_Pal

II. Specify data as an ndarray

we give data as any numpy array like np.arrange(), np.linspace(), np.title(), etc.

```
import pandas as pd
import numpy as np
S4 = pd.Series(np.arrange(1,3,0.5))
print(S4)
```

arange() function will give data from 1 to 3 with a gap or step of 0.5 with float datatype.

Output:

```
S4:
 0  1.0
 2  1.5
 3  2.0
 4  2.5
 dtype: float32
```

III. Specify data as Python dictionary.

here we will give the data using a python dictionary.

Here data is specified as a python dictionary

```
import pandas as pd
dict1={‘A’:1, ‘B’:5, ‘C’:9, ‘D’:6 }
S5 = pd.Series(dict1)
# we can pass the dictionary on the place of dict1 like
# S5 = pd.Series({‘A’:1, ‘B’:5, ‘C’:9, ‘D’:6})
print(S5)
```

Remember these are comments

Output:

```
S5:
 A  1
 B  5
 C  9
 D  6
 dtype: int64
```

Divyanshu_Pal

IV. Specify Data as scalar value.

here we will specify the data as any scalar value like 1,9,10,453,etc.

```
import pandas as pd
S6 = pd.Series(6,index=range(0,3))
print(S6)

S7=pd.Series(8)
print(S7)
```

Output:

```
S6:
 0  6
 1  6
 2  6
 dtype: int64

S7:
 dtype:int64
```

See, the output of S6 object value 6 is given to all specified indexes

Here, no value is displayed because indexes were not given.

Divyanshu_Pal

Creating series object(Additional Functionality)

I. Specifying or adding NaN values.

here we will include NaN values in our data array or input.

```
import pandas as pd
import numpy as np
S8 = pd.Series([2,5,np.NaN,7])
print(S8)
```

Output:

```
S8:
 0  2.0
 1  5.0
 2  NaN
 3  7.0
 dtype: float64
```

Observe here datatype is float64 not float32 because nan is a 64-bit datatype

Divyanshu_Pal

II. Specifying index(es) as well as data with Series().

Here we will provide data with any method but along with custom indexes.

Note these points:

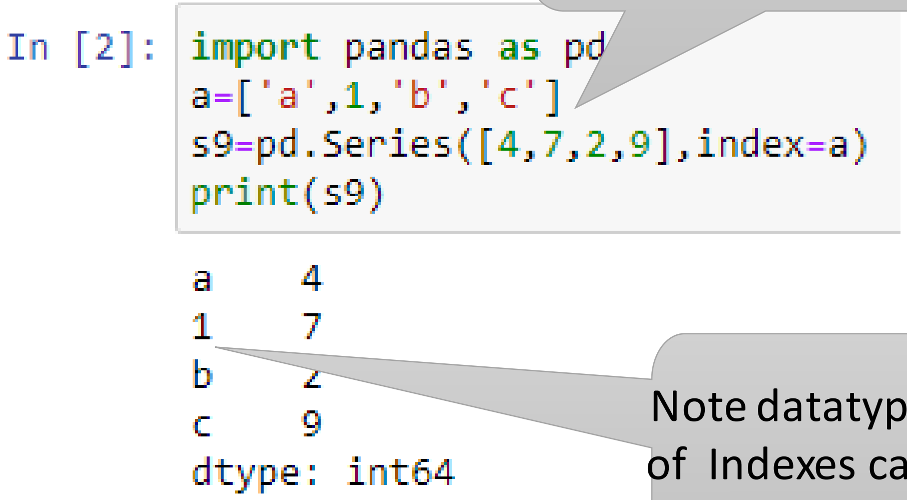
1. **No. of indexes must be equal to the no. of values.**
2. **Error will be displayed if 6 indexes are given for 5 data points.**

Recall the syntax of series:

syntax: <series object> = pandas.Series(data,index=[])

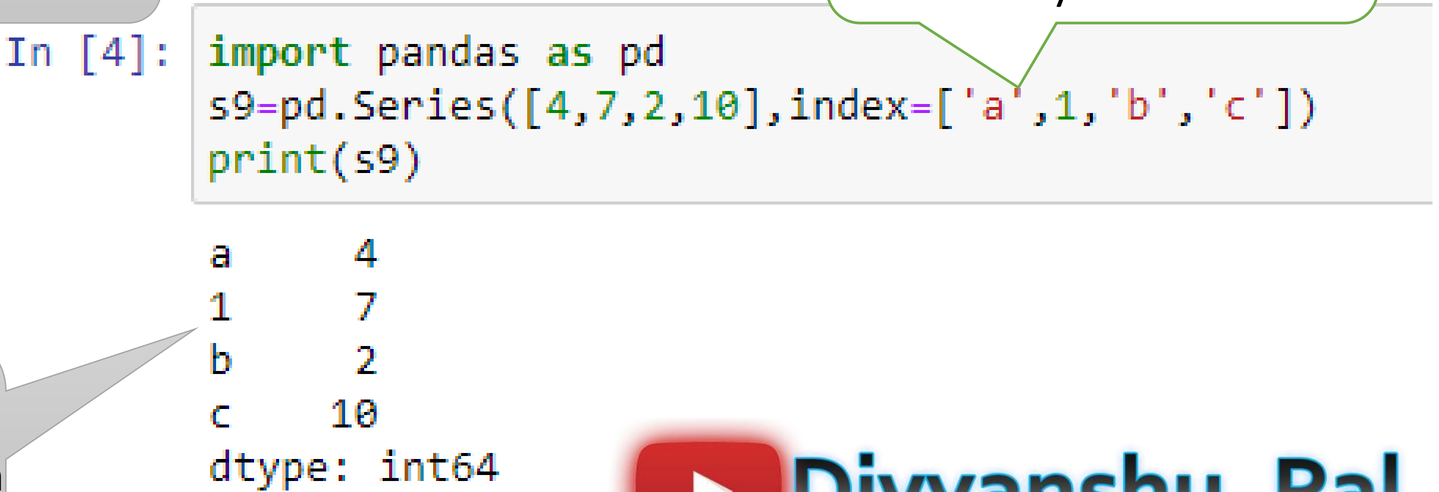
Here list of indexes are stored in different variable

Here the list of indexes is directly passed to the syntax.



```
In [2]: import pandas as pd
        a=['a',1,'b','c']
        s9=pd.Series([4,7,2,9],index=a)
        print(s9)

a     4
1     7
b     2
c     9
dtype: int64
```



```
In [4]: import pandas as pd
        s9=pd.Series([4,7,2,10],index=['a',1,'b','c'])
        print(s9)

a      4
1      7
b      2
c     10
dtype: int64
```

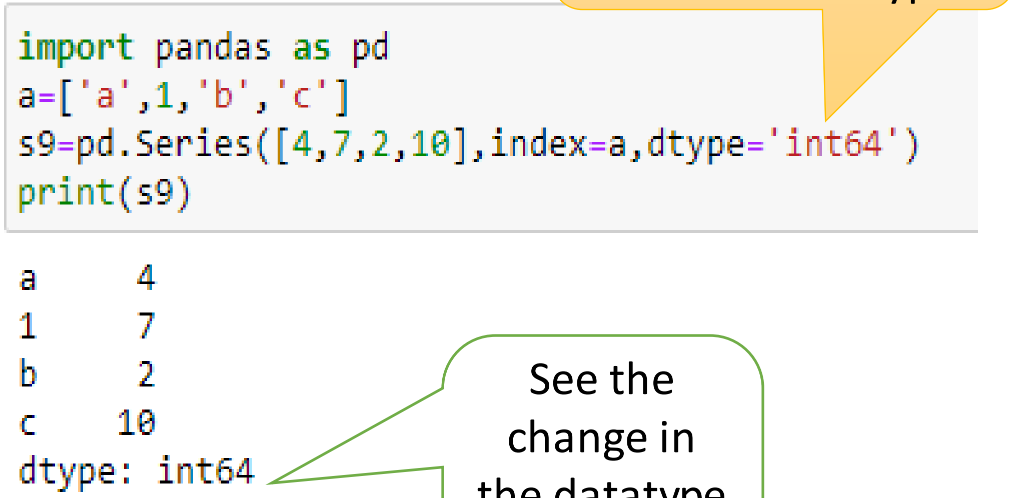
Note datatype of Indexes can be different.

Divyanshu_Pal

III. Specify datatype along with data and indexes.

Here we will specify our custom datatype in the series object.

Here we have specified int64 as the new datatype

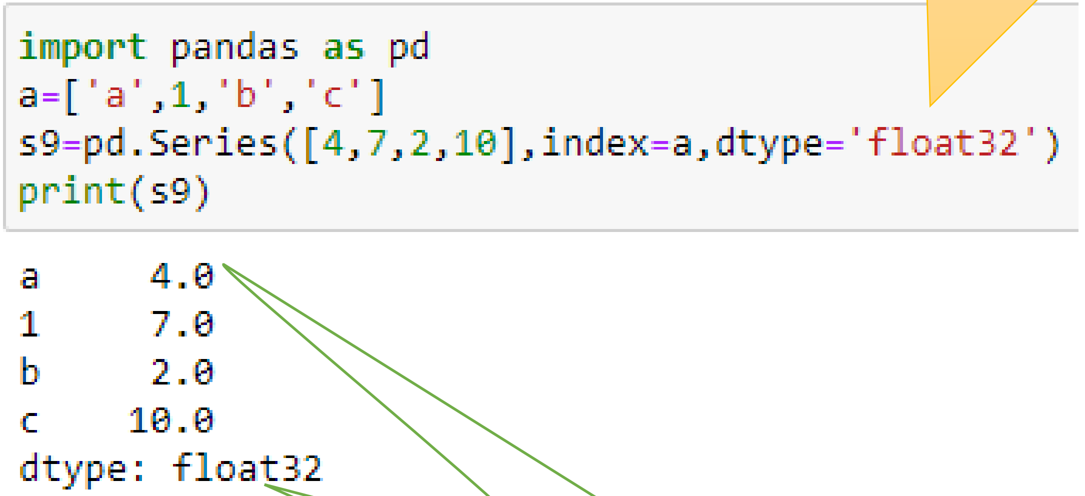


```
import pandas as pd
a=['a',1,'b','c']
s9=pd.Series([4,7,2,10],index=a,dtype='int64')
print(s9)
```

```
a     4
1     7
b     2
c    10
dtype: int64
```

See the change in the datatype of series object.

Here we have specified float32 as the new datatype



```
import pandas as pd
a=['a',1,'b','c']
s9=pd.Series([4,7,2,10],index=a,dtype='float32')
print(s9)
```

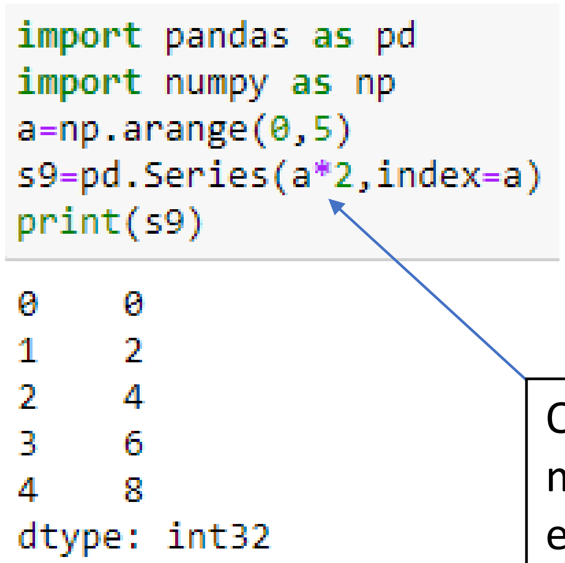
```
a     4.0
1     7.0
b     2.0
c    10.0
dtype: float32
```

Observe the change in the method of displaying Float datatype

Divyanshu_Pal

IV. Using a mathematical Function/Expression to create Data Array in Series().

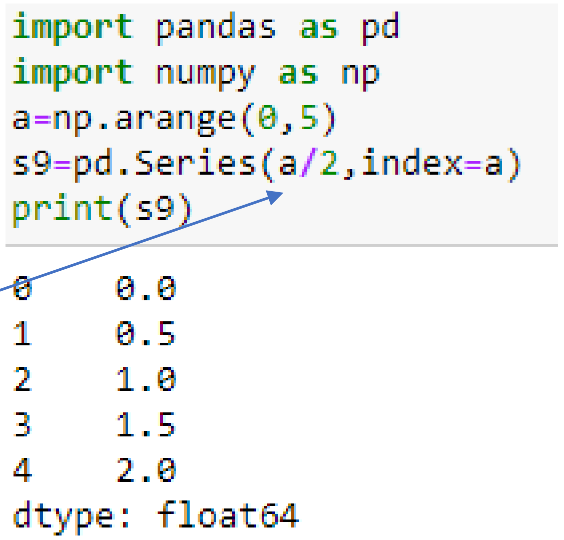
Here we will not give data as a list, dictionary or any python sequence but as a mathematical function/expression.



```
import pandas as pd
import numpy as np
a=np.arange(0,5)
s9=pd.Series(a*2,index=a)
print(s9)
```

```
0    0
1    2
2    4
3    6
4    8
dtype: int32
```

Observe here a mathematical expression is given as the data input.



```
import pandas as pd
import numpy as np
a=np.arange(0,5)
s9=pd.Series(a/2,index=a)
print(s9)
```

```
0    0.0
1    0.5
2    1.0
3    1.5
4    2.0
dtype: float64
```

Divyanshu_Pal

* Points to remember

- Syntax of series: <object> = pandas.Series(data,index=[],dtype=“ “)

- S must be capital.
- Datatype must be enclosed in quotation marks like ‘int64’ or “int64”.
- Indexes can have different data types.
- Values must have unique datatype.

Divyanshu_Pal