Python Pandas Series

The Pandas Series can be defined as a one-dimensional array that is capable of storing various data types. We can easily convert the list, tuple, and dictionary into series using "series' method. The row labels of series are called the index. A Series cannot contain multiple columns. It has the following parameter:

- **data:** It can be any list, dictionary, or scalar value.
- **index**: The value of the index should be unique and hashable. It must be of the same length as data. If we do not pass any index, default np.arrange(n) will be used.
- **dtype:** It refers to the data type of series.
- **copy**: It is used for copying the data.

Creating a Series:

We can create a Series in two ways:

Create an empty Series

Create a Series using inputs.

Create an Empty Series:

We can easily create an empty series in Pandas which means it will not have any value.

The syntax that is used for creating an Empty Series:

```
<series object> = pandas.Series()
```

The below example creates an Empty Series type object that has no values and having default datatype, i.e., float64.

Example

```
import pandas as pd

x = pd.Series()

print (x)
```

Creating a Series using inputs:

We can create Series by using various inputs:

- Array
- Dict
- Scalar value

Creating Series from Array:

Before creating a Series, firstly, we have to import the numpy module and then use array() function in the program. If the data is ndarray, then the passed index must be of the same length.

If we do not pass an index, then by default index of range(n) is being passed where n defines the length of an array, i.e., [0,1,2,....range(len(array))-1].

Example

```
import pandas as pd

import numpy as np

info = np.array(['P','a','n','d','a','s'])

a = pd.Series(info)

print(a)
```

Create a Series from dict

We can also create a Series from dict. If the dictionary object is being passed as an input and the index is not specified, then the dictionary keys are taken in a sorted order to construct the index.

If index is passed, then values correspond to a particular label in the index will be extracted from the dictionary.

```
#import the pandas library

import pandas as pd

import numpy as np

info = {'x' : 0., 'y' : 1., 'z' : 2.}

a = pd.Series(info)

print (a)
```

Create a Series using Scalar:

If we take the scalar values, then the index must be provided. The scalar value will be repeated for matching the length of the index.

```
#import pandas library

import pandas as pd

import numpy as np

x = pd.Series(4, index=[0, 1, 2, 3])

print (x)
```

Accessing data from series with Position:

Once you create the Series type object, you can access its indexes, data, and even individual elements.

The data in the Series can be accessed similar to that in the ndarray.

```
import pandas as pd

x = pd.Series([1,2,3],index = ['a','b','c'])

#retrieve the first element

print (x[0])
```

Retrieving Index array and data array of a series object

We can retrieve the index array and data array of an existing Series object by using the attributes index and values.

```
import numpy as np

import pandas as pd

x=pd.Series(data=[2,4,6,8])

y=pd.Series(data=[11.2,18.6,22.5], index=['a','b','c'])

print(x.index)

print(x.values)

print(y.index)
```

```
print(y.values)
```

Retrieving Types (dtype) and Size of Type (itemsize)

You can use attribute dtype with Series object as <objectname> dtype for retrieving the data type of an individual element of a series object, you can use the itemsize attribute to show the number of bytes allocated to each data item.

```
import numpy as np
import pandas as pd
a=pd.Series(data=[1,2,3,4])
b=pd.Series(data=[4.9,8.2,5.6],
index=['x','y','z'])
print(a.dtype)
print(a.itemsize)
print(b.dtype)
print(b.itemsize)
```

Retrieving Shape

The shape of the Series object defines total number of elements including missing or empty values(NaN).

```
import numpy as np
import pandas as pd
a=pd.Series(data=[1,2,3,4])
b=pd.Series(data=[4.9,8.2,5.6],index=['x','y','z'])
print(a.shape)
print(b.shape)
```

Retrieving Dimension, Size and Number of bytes:

```
import numpy as np
import pandas as pd
a=pd.Series(data=[1,2,3,4])
b=pd.Series(data=[4.9,8.2,5.6],
index=['x','y','z'])
print(a.ndim, b.ndim)
print(a.size, b.size)
print(a.nbytes, b.nbytes)
```