Contents

2. PANDAS – SERIES

1. Series

- ✓ The Pandas Series is a one-dimensional labeled array.
- ✓ Series can store same and different types of the data.
- ✓ Series stores data in sequential order.
- ✓ Series is like a, one column information.

Series is a pre-defined class

- ✓ Technically speaking **Series** is a pre-defined class in pandas library.

2. Creating Series

- ✓ We can create Series in different ways,
  - Empty series
  - By using list
  - By using an array
  - By accessing single column from DataFrame.

2.1. Empty Series object

- ✓ We need to create object to Series pre-defined class.

Program Name: creating empty Series object
demo1.py

```
import pandas as pd

s = pd.Series()
print(s)
print(type(s))
```

Output

```
Series([], dtype: float64)
<class 'pandas.core.series.Series'>
```

2.2. Creating Series by using list

- ✓ We can create Series by using list.

Program Name: creating Series object using list
demo2.py

```
import pandas as pd

m = [56, 45, 35, 41, 44, 60]
s = pd.Series(m)
print(s)
```

Output

```
0    56
1    45
2    35
3    41
4    44
5    60
dtype: int64
```

Program Name

creating Series object using list, assigning a name
demo3.py

```
import pandas as pd

m = [56, 45, 35, 41, 44, 60]
s = pd.Series(m, name = "marks")
print(s)
```

Output

```
0    56
1    45
2    35
3    41
4    44
5    60
Name: marks, dtype: int64
```

Program Name: creating Series object using list, assigning a name
demo4.py

```
import pandas as pd

n = ["Prasad", "Daniel", "Samuel", "Jeswanth"]
s = pd.Series(n, name = "students")
print(s)
```

Output

```
0       Prasad
1       Daniel
2       Samuel
3    Jeswanth
Name: students, dtype: object
```

Program Name: creating Series object by using range and list.
demo5.py

```
import pandas as pd

r = range(100)
a = list(r)
s = pd.Series(a)

print(s)
```

Output

```
0       0
1       1
2       2
3       3
4       4
       ..
95     95
96     96
97     97
98     98
99     99
Length: 100, dtype: int64
```

2.3. Create a Series from array

- ✓ We can pass array as an argument to the Series.
- ✓ By default, index is assigned to every element.

Program Name: creating ndarray
demo6.py

```
import pandas as pd
import numpy as np

values = [10, 20, 30, 40]
data = np.array(values)

print(data)
print(type(data))
```

Output

```
[10 20 30 40]
<class 'numpy.ndarray'>
```

Program Name: creating ndarray and passing argument to the Series
demo7.py

```
import pandas as pd
import numpy as np

values = [10, 20, 30, 40]
data = np.array(values)

s = pd.Series(data)
print(s)
```

Output

```
0    10
1    20
2    30
3    40
dtype: int32
```

Program Name: Creating Series using ndarray
demo8.py

```
import pandas as pd
import numpy as np

values = ['a', 'b', 'c', 'd']
data = np.array(values)

s = pd.Series(data)
print(s)
```

Output

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

Program Name: Creating Series using ndarray
demo9.py

```
import pandas as pd
import numpy as np

values = ['Vinay', 'Daniel', 'Veeru', 'Arjun']
data = np.array(values)
s = pd.Series(data)
print(s)
```

Output

```
0      Vinay
1     Daniel
2      Veeru
3      Arjun
dtype: object
```

2.4. Creating Series by column from DataFrame

- ✓ If we select single column from DataFrame then it returns Series object.
- ✓ This point we will learn during DataFrame chapter, thanks for understanding.

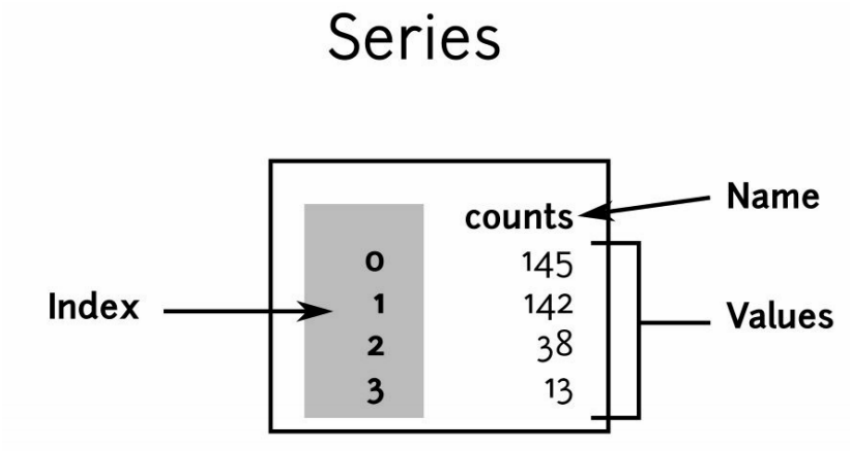
3. Index in Series

3.1. What is index?

- ✓ Index means, the position of value where it stores.
- ✓ The index is a core feature in pandas.
- ✓ By default, index is assigned to every value.
- ✓ From the output, the left most column is the index column.
- ✓ The generic name for an index is an axis.

3.2. Index default value

- ✓ The default values for an index are integers.
- ✓ The index starts from 0, 1, 2, 3, etc.
- ✓ Based on requirement we can customise this index
- ✓ These are called as axis labels



Program Name: Creating Series
demo10.py

```
import pandas as pd

v = [145, 142, 38, 13]
s = pd.Series(v)
print(s)
```

Output

```
0    145
1    142
2     38
3     13
dtype: int64
```

Program Name: Creating Series object and giving name
demo11.py

```
import pandas as pd

v = [145, 142, 38, 13]
s = pd.Series(v, name = 'counts')
print(s)
```

Output

```
0    145
1    142
2     38
3     13
Name: counts, dtype: int64
```

Program Name

Creating Series object and giving name and index
demo12.py

```
import pandas as pd

v = [145, 142, 38, 13]
i = [10, 20, 30, 40]

s = pd.Series(v, name = 'counts', index = i )
print(s)
```

Output

```
10    145
20    142
30     38
40     13
Name: counts, dtype: int64
```

Program Name

Creating Series object and giving name and index
demo13.py

```
import pandas as pd

prices = [1000, 2000, 3000, 4000]
products = ["Nokia", "Samsung", "Oppo", "iPhone 6"]

s = pd.Series(prices, name = 'mobiles', index = products )
print(s)
```

Output

```
Nokia       1000
Samsung     2000
Oppo        3000
iPhone 6    4000
Name: mobiles, dtype: int64
```

4. Accessing values in Series

- ✓ We can access series values by using index

Program Name: Creating Series and accessing values
demo14.py

```
import pandas as pd

v = [56, 45, 35, 41, 44, 60]
s = pd.Series(v, name = "marks")

print(s)
print()
print(s[0])
print(s[1])
```

Output

```
0    56
1    45
2    35
3    41
4    44
5    60
Name: marks, dtype: int64

56
45
```

Program Name: Creating Series and accessing values
demo15.py

```
import pandas as pd

prices = [1000, 2000, 3000, 4000]
products = ["Nokia", "Samsung", "Oppo", "iPhone 6"]

s = pd.Series(prices, name = 'mobiles', index = products )

print(s)
print()
print(s["Nokia"])
print(s["Samsung"])
```

Output

```
Nokia       1000
Samsung     2000
Oppo        3000
iPhone 6    4000
Name: mobiles, dtype: int64

1000
2000
```