

Python Data Structures: Series

Programming for Data Science with Python

Overview

Series is a **one-dimensional labeled NumPy array**

- capable of holding any data type (integers, strings, floating point numbers, Python objects, etc.)
 - The axis labels are collectively referred to as the index.
 - All of the values have the same data type, similar to Numpy ndarrays

The basic method to create a series is to call: `**s = pandas.Series(data, index=index)`

Here "data" can be a ndarray, or a dictionary, or a scalar value, etc.

1. Create Series

1.1 Series Constructor

A pandas Series can be created with the following constructor: **pandas.Series(data, Index, dtype, copy)**

data: constants, ndarray, list, dictionary, etc **index:** Index values must be unique and hashable, same length as data.

- index: passed parameter is a list of axis label
- Default: `np.arange(n)` if no index is passed.

dtype: dtype is for data type. If None, data type will be inferred.

- copy: Copy data. Default: False

1.2 Create empty series

****Run the following code block:****

```
In [1]: # Create an empty series

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

Series([], dtype: object)

```
In [2]: import pandas as pd
world_series = pd.Series()
print(world_series)
```

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

1.3 Create a series from an ndarray

If data is an ndarray, then index passed must be of the same length.

- If no index is passed, then by default index will be range(n) where n is array length, i.e., [0, 1, 2, 3 ... range (len(array))-1].

****Run the following 2 code blocks:****

```
In [3]: # Example 1: Create a series from an ndarray
import pandas as pd
import numpy as np
# Array is created from a list
data = np.array(['a', 'b', 'c', 'd'])
# A series is created from the array with the default index
s = pd.Series(data)
print(s)
```

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

```
In [7]: import pandas as pd
import numpy as np
series_data = np.array(['Texas Rangers', 'Houston Astros', 'Atlanta Braves', 'Los Angeles Dodgers'])
world_series = pd.Series(series_data)
print(world_series)
```

```
0    Texas Rangers
1    Houston Astros
2    Atlanta Braves
3    Los Angeles Dodgers
dtype: object
```

```
In [8]: #Example 2: Create a series from an ndarray
import pandas as pd
import numpy as np
#Array is created from a list
data = np.array(['a', 'b', 'c', 'd'])
# A series is created from the array with specific indices
s = pd.Series(data, index=[100, 101, 102, 103])
print(s)
```

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

```
In [11]: import pandas as pd
import numpy as np
series_data = np.array(['Texas Rangers', 'Houston Astros', 'Atlanta Braves', 'Los Angeles Dodgers'])
world_series = pd.Series(series_data, index=[2023, 2022, 2021, 2020])
print(world_series)
```

```
2023      Texas Rangers
2022      Houston Astros
2021      Atlanta Braves
2020  Los Angeles Dodgers
dtype: object
```

1.4 Create a series from a dictionary

A dict can be passed as input.

- If no index is specified, then the dictionary keys are taken in a sorted order to construct the index.
- If index is passed, the values in data corresponding to the labels in the index will be pulled out.

****Run the following 2 code blocks:****

```
In [12]: # Create a series from a dictionary
import pandas as pd
import numpy as np

# Declare a dictionary with keys: 'a', 'b', 'c'
aDict = {'a': 0., 'b': 1., 'c': 2.}

# Create a series from this dictionary
s = pd.Series(aDict)
print(s)
```

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

```
In [15]: import pandas as pd
import numpy as np

series_Dict = {'Season': 2023, 'Winner': 'Texas Rangers', 'Losers': 'Arizona Diamondbacks'}
world_series = pd.Series(series_Dict)
print(world_series)
```

```
Season      2023
Winner      Texas Rangers
Losers      Arizona Diamondbacks
dtype: object
```

```
In [16]: # Create a series from a dictionary

import pandas as pd
import numpy as np

# Declare a dictionary with keys: 'a', 'b', 'c'
```

```
data = {'a': 0., 'b' : 1., 'c' :2.}

# Create a series from this dictionary with specific indices
# The dict has only three items
s = pd.Series(data, index=['b','c','d','a'])
print(s)

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

```
In [17]: import pandas as pd
import numpy as np

series_Dict = {'Season': 2023, 'Winner': 'Texas Rangers', 'Losers': 'Arizona Diamondbacks', 'Series/Score': 100}
world_series = pd.Series(series_Dict, index=['Season','Winner', 'Losers','Series/Score'])
print(world_series)

Season                2023
Winner                Texas Rangers
Losers                Arizona Diamondbacks
Series/Score                NaN
dtype: object
```

1.5 Create a series from scalar values

If data is a **scalar** value, an **index must be provided**.

- The value will be repeated to match the length of index.

Run the following code block:

```
In [18]: # Create a series from scalar values
import pandas as pd
import numpy as np
# Create a series
s = pd.Series(5, index=[0, 1, 2, 3])
print(s)

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

```
In [22]: import pandas as pd
import numpy as np

world_series = pd.Series(10, index=[0,1,2,3])
print(world_series)

0    10
1    10
2    10
3    10
dtype: int64
```

1.6 Accessing Data from Series with Position

Data in the series can be accessed similar to that ndarray.

****Run the following code block:****

```
In [23]: import pandas as pd
s = pd.Series([1,2,3,4,5], index = ['a','b','c','d','e'])

#retrieve the first element
print(s[0])
```

1

```
In [32]: import pandas as pd
world_series = pd.Series([2023,2022,2021,2020], index = ['Texas Rangers', 'Houston Ast
print(world_series[3])
```

2020

```
In [33]: import pandas as pd
s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])

# Retrieve the first 3 elements: from 0 - 3, not including 3
# i.e., retrieve 0,1,2
print(s[:3])
```

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

```
In [34]: import pandas as pd
world_series = pd.Series([2023,2022,2021,2020], index = ['Texas Rangers', 'Houston Ast
print(world_series[:3])
```

```
Texas Rangers    2023
Houston Astros    2022
Atlanta Braves    2021
dtype: int64
```

```
In [35]: import pandas as pd
s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])

# Retrieve the last 3 elements:
print(s[-3:])
```

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

```
In [36]: import pandas as pd
world_series = pd.Series([2023,2022,2021,2020], index = ['Texas Rangers', 'Houston Ast
print(world_series[-3:])
```

```
Houston Astros      2022
Atlanta Braves      2021
Los Angeles Dodgers  2020
dtype: int64
```

```
In [37]: import pandas as pd
s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])
```

```
# Retrieve a single element at a specific index
print(s['a'])
```

```
1
```

```
In [38]: import pandas as pd
world_series = pd.Series([2023,2022,2021,2020], index = ['Texas Rangers', 'Houston Ast
print(world_series['Texas Rangers'])
```

```
2023
```

```
In [39]: import pandas as pd
s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])

# Retrieve multiple elements using a list of index label values
print(s[['a','c','d']])
```

```
a    1
```

```
c    3
```

```
d    4
```

```
dtype: int64
```

```
In [40]: import pandas as pd
world_series = pd.Series([2023,2022,2021,2020], index = ['Texas Rangers', 'Houston Ast
print(world_series[['Texas Rangers', 'Houston Astros', 'Los Angeles Dodgers']])
```

```
Texas Rangers      2023
```

```
Houston Astros     2022
```

```
Los Angeles Dodgers 2020
```

```
dtype: int64
```