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 de 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.arrange(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 [12]:
```
# Create an empty series

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

```
Series([], dtype: float64)

<ipython-input-12-06a093c886cc>:4: DeprecationWarning: The default dt
ype for empty Series will be 'object' instead of 'float64' in a futur
e version. Specify a dtype explicitly to silence this warning.
  s=pd.Series()
```

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 [13]:
```
# 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 [14]:
```
#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 sereis 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
```

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 [15]:
```
# Create a series from a dictionary
import pandas as pd
import numpy as np

# Declare a dicctionary 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 [29]:
```
# Create a series from a dictionary

import pandas as pd
import numpy as np

# Declare a dicctionary 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
```

1.5 Create a sereis 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 [30]:
```
# 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
```

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 [24]:
```
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 [31]:
```
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 [32]:
```
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 [28]:
```
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 [ ]:
```

```