01-Series

October 19, 2022

Copyright Pierian Data

For more information, visit us at www.pieriandata.com

1 Series

The first main data type we will learn about for pandas is the Series data type. Let's import Pandas and explore the Series object.

A Series is very similar to a NumPy array (in fact it is built on top of the NumPy array object). What differentiates the NumPy array from a Series, is that a Series can have axis labels, meaning it can be indexed by a label, instead of just a number location. It also doesn't need to hold numeric data, it can hold any arbitrary Python Object.

Let's explore this concept through some examples:

[1]:
```
import numpy as np
import pandas as pd
```

1.1 Creating a Series

You can convert a list,numpy array, or dictionary to a Series:

[3]:
```
labels = ['a','b','c']
my_list = [10,20,30]
arr = np.array([10,20,30])
d = {'a':10,'b':20,'c':30}
```

1.1.1 Using Lists

[4]:
```
pd.Series(data=my_list)
```

[4]:
```
0    10
1    20
2    30
dtype: int64
```

[5]:
```
pd.Series(data=my_list,index=labels)
```

[5]:
```
a    10
b    20
c    30
dtype: int64
```

[6]:
```
pd.Series(my_list,labels)
```

[6]:
```
a    10
b    20
c    30
dtype: int64
```

1.1.2 Using NumPy Arrays

[7]:
```
pd.Series(arr)
```

[7]:
```
0    10
1    20
2    30
dtype: int64
```

[8]:
```
pd.Series(arr,labels)
```

[8]:
```
a    10
b    20
c    30
dtype: int64
```

1.1.3 Using Dictionaries

[9]:
```
pd.Series(d)
```

[9]:
```
a    10
b    20
c    30
dtype: int64
```

1.1.4 Data in a Series

A pandas Series can hold a variety of object types:

[10]:
```
pd.Series(data=labels)
```

[10]:
```
0    a
1    b
2    c
dtype: object
```

[11]:
```
# Even functions (although unlikely that you will use this)
pd.Series([sum,print,len])
```

[11]:
```
0      <built-in function sum>
1    <built-in function print>
2      <built-in function len>
dtype: object
```

1.2 Using an Index

The key to using a Series is understanding its index. Pandas makes use of these index names or numbers by allowing for fast look ups of information (works like a hash table or dictionary).

Let's see some examples of how to grab information from a Series. Let us create two sereis, ser1 and ser2:

[12]:
```
ser1 = pd.Series([1,2,3,4],index = ['USA', 'Germany','USSR', 'Japan'])
```

[13]:
```
ser1
```

[13]:
```
USA        1
Germany    2
USSR       3
Japan      4
dtype: int64
```

[14]:
```
ser2 = pd.Series([1,2,5,4],index = ['USA', 'Germany','Italy', 'Japan'])
```

[15]:
```
ser2
```

[15]:
```
USA        1
Germany    2
Italy      5
Japan      4
dtype: int64
```

[16]:
```
ser1['USA']
```

[16]:
```
1
```

Operations are then also done based off of index:

[17]:
```
ser1 + ser2
```

[17]:
```
Germany    4.0
Italy      NaN
Japan      8.0
USA        2.0
USSR       NaN
dtype: float64
```

Let's stop here for now and move on to DataFrames, which will expand on the concept of Series! # Great Job!