Creating Pandas Series

Pandas Series

A Pandas series is a *one-dimensional* array-like object that can hold many data types, such as numbers or strings, and has an option to provide axis labels.

Difference between NumPy ndarrays and Pandas Series

1. One of the main differences between Pandas Series and NumPy ndarrays is that you can assign an index label to each element in the Pandas Series. In other words, you can name the indices of your Pandas Series anything you want.
2. Another big difference between Pandas Series and NumPy ndarrays is that Pandas Series can hold data of different data types.

Let's start by importing Pandas into Python. It has become a convention to import Pandas as pd, therefore, you can import Pandas by typing the following command in your Jupyter notebook:

```
import pandas as pd
```

Let's begin by creating a Pandas Series. You can create Pandas Series by using the command pd.Series(data, index), where index is a list of index labels. Let's use a Pandas Series to store a grocery list. We will use the food items as index labels and the quantity we need to buy of each item as our data.

Example 1 - Create a Series

```
# We import Pandas as pd into Python

import pandas as pd


# We create a Pandas Series that stores a grocery list

groceries = pd.Series(data = [30, 6, 'Yes', 'No'], index = ['eggs', 'apples', 'milk', 'bread'])


# We display the Groceries Pandas Series

groceries
```

```
eggs          30
apples         6
milk         Yes
bread         No
dtype: object
```

We see that Pandas Series are displayed with the indices in the first column and the data in the second column. Notice that the data is not indexed 0 to 3 but rather it is indexed with the names of the food we

put in, namely eggs, apples, etc... Also, notice that the data in our Pandas Series has both integers and strings.

Just like NumPy ndarrays, Pandas Series have attributes that allow us to get information from the series in an easy way. Let's see some of them:

Example 2 - Print attributes - shape, ndim,and size

```
# We print some information about Groceries

print('Groceries has shape:', groceries.shape)

print('Groceries has dimension:', groceries.ndim)

print('Groceries has a total of', groceries.size, 'elements')
```

Groceries has shape: (4,)
Groceries has dimension: 1
Groceries has a total of 4 elements

We can also print the index labels and the data of the Pandas Series separately. This is useful if you don't happen to know what the index labels of the Pandas Series are.

Example 3 - Print attributes - values, and index

```
# We print the index and data of Groceries

print('The data in Groceries is:', groceries.values)

print('The index of Groceries is:', groceries.index)
```

The data in Groceries is: [30 6 'Yes' 'No']
The index of Groceries is: Index(['eggs', 'apples', 'milk', 'bread'], dtype='object')

If you are dealing with a very large Pandas Series and if you are not sure whether an index label exists, you can check by using the in command

Example 4 - Check if an index is available in the given Series

```
# We check whether bananas is a food item (an index) in Groceries

x = 'bananas' in groceries


# We check whether bread is a food item (an index) in Groceries

y = 'bread' in groceries


# We print the results

print('Is bananas an index label in Groceries:', x)
```

```
print('Is bread an index label in Groceries:', y)
```

```
Is bananas an index label in Groceries: False
Is bread an index label in Groceries: True
```

Additional Reading - Pandas Series Documentation

Refer to the Series documentation to have a quick glance at the different attributes of Series, and the optional arguments of the constructor.