My Pandas Series Hands on Codes

. . . by swarnadeep

Importing Libraries

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

Making a Series database `planetData` with two 1D array `distance_from_sun` and `planets`

In [2]:
```
distance_from_sun = [149.6, 1433.5, 227.9, 108.2, 778.6]
```

In [3]:
```
planets = ['Earth','Saturn', 'Mars','Venus', 'Jupiter']
```

In [4]:
```
planetData = pd.Series(distance_from_sun,index = planets)
```

In [5]:
```
planetData
```

Out[5]:
```
Earth       149.6
Saturn     1433.5
Mars        227.9
Venus       108.2
Jupiter     778.6
dtype: float64
```

Printing Shape , Dimension and Size of Pandas Series

In [6]:
```
planetData.shape
```

Out[6]:
```
(5,)
```

In [7]:
```
planetData.ndim
```

Out[7]:
```
1
```

In [8]:
```
planetData.size
```

Out[8]:
```
5
```

Printing `.values` and `.index` of Series and Checking if an index is present or not by `in` operator

In [9]:
```
print('The data in planetData is:', planetData.values)
print('The index of planetData is:', planetData.index)
```

```
The data in planetData is: [ 149.6 1433.5  227.9  108.2  778.6]
The index of planetData is: Index(['Earth', 'Saturn', 'Mars', 'Venus', 'Jupiter'], dtype='object')
```

In [10]:
```
print('Is Earth an index label in planetData:', 'Earth' in planetData)
print('Is Sun an index label in planetData:', 'Sun' in planetData)
```

```
Is Earth an index label in planetData: True
Is Sun an index label in planetData: False
```

Accessing, Changing Data in Pandas Series

In [11]:
```
# Accessing ELement by Single index label
print('Distance of Saturn : ',planetData['Saturn'])

# Or we can use nummerical indexing by default
print('Distance of Saturn(by Nummerical Indexing) : ',planetData[1])

# We can also access multiple elements
print("The Distance of Earth & Mars is :\n",planetData[['Earth','Mars']])
```

```
Distance of Saturn :  1433.5
Distance of Saturn(by Nummerical Indexing) :  1433.5
The Distance of Earth & Mars is :
 Earth    149.6
Mars     227.9
dtype: float64
```

In [12]:
```
# Changing a Data Value by its index
planetData['Saturn'] = 1200
```

In [13]:
```
planetData
```

Out[13]:
```
Earth       149.6
Saturn     1200.0
Mars        227.9
Venus       108.2
Jupiter     778.6
dtype: float64
```

In [14]:
```
# Printing where distance is greater than 500
planetData[planetData > 500]
```

Out[14]:
```
Saturn     1200.0
Jupiter     778.6
dtype: float64
```

Using `.loc[]` and `.iloc[]` for Accessing Element (Recommended)

In [15]:
```
# loc uses actual index to access elements
planetData.loc['Mars']
```

Out[15]:
```
227.9
```

In [16]:
```
planetData.loc[['Mars','Earth']]
```

Out[16]:
```
Mars     227.9
Earth    149.6
dtype: float64
```

In [17]:
```
# iloc uses equivalent nummerical position of index to access elements
planetData.iloc[0]
```

Out[17]:
```
149.6
```

In [18]:
```
planetData.iloc[[3,4]]
```

Out[18]:
```
Venus      108.2
Jupiter    778.6
dtype: float64
```

Deleting Elements in Pandas Series by index label

In [19]:
```
planetData.drop('Venus')
```

Out[19]:
```
Earth       149.6
Saturn     1200.0
Mars        227.9
Jupiter     778.6
dtype: float64
```

In [20]:
```
planetData.drop(['Venus','Jupiter'])
```

Out[20]:
```
Earth      149.6
Saturn    1200.0
Mars       227.9
dtype: float64
```

In [21]:
```
# To update the `planetData` instantly, we use `inplace`
planetData.drop('Venus', inplace = True)
```

In [22]:
```
planetData
```

Out[22]:
```
Earth       149.6
Saturn     1200.0
Mars        227.9
Jupiter     778.6
dtype: float64
```

Arithmetic Operations in Pandas Series

Just like with NumPy ndarrays, we can perform element-wise arithmetic operations on Pandas Series.

In [23]:
```
fruits= pd.Series(data = [10, 6, 3,], index = ['apples', 'oranges', 'bananas'])
```

In [24]:
```
fruits
```

Out[24]:
```
apples     10
oranges     6
bananas     3
dtype: int64
```

We can use `+` , `-` , `*` , `/` to perform basic element-wise operations

In [25]:
```
print('fruits + 2:\n', fruits + 2)
print('Similarly we can do substraction, multiplication & division')
```

```
fruits + 2:
 apples     12
oranges     8
bananas     5
dtype: int64
Similarly we can do substraction, multiplication & division
```

We can also apply mathematical functions from NumPy like `EXP()` , `SQRT()` , `POW()`

In [26]:
```
print('EXP(X) = \n', np.exp(fruits))
```

```
EXP(X) =
 apples      22026.465795
oranges       403.428793
bananas        20.085537
dtype: float64
```

In [27]:
```
print('SQRT(X) =\n', np.sqrt(fruits))
```

```
SQRT(X) =
 apples      3.162278
oranges     2.449490
bananas     1.732051
dtype: float64
```

In [28]:
```
print('POW(X,2) =\n',np.power(fruits,2))
```

```
POW(X,2) =
 apples      100
oranges      36
bananas       9
dtype: int64
```

Pandas also allows us to only apply arithmetic operations on selected items.

In [29]:
```
print('We half the amount of apples and oranges:\n', fruits.loc[['apples', 'oranges']] / 2)
```

```
We half the amount of apples and oranges:
 apples     5.0
oranges    3.0
dtype: float64
```

In [30]:
```
print('Amount of apples - 2 = ', fruits.iloc[0] - 2)
print()
```

```
Amount of apples - 2 =  8
```

In [ ]:
```

```