SERIES

Series is a one-dimensional array like structure with homogeneous data, which can be used to handle and manipulate data. What makes it special is its index attribute, which has incredible functionality and is heavily mutable.

It has two parts

1. Data part (An array of actual data)
2. Associated index with data (associated array of indexes or data labels).

✓ Data of Series is always mutable, means it can be changed. ✓ But the size of Data of Series is always immutable, means it cannot be changed.

In [9]:
```
# Syntax : pd.Series(data, index=(optional))
# data --> ndarray(or numpy array), scalar quan., python sequence( lists), dictiona

# Series with ndarray

import pandas as pd
import numpy as np

arr1 = np.array([10,20,30,40,50])
df = pd.Series(arr1)
print(df)

# Series with Mutable index.
arr2 = np.array([10,20,30,40,50])
df1 = pd.Series(arr2, index=['a','b','c','d','e'])
print(df1)

# Creating from scalar values.

#  --> To create a series from scalar value, an index must be provided. The
#  --> scalar value will be repeated as per the length of index.

df3 = pd.Series(50, index=['a','b','c','d','e'])
print(df3)

# Creating Series from Dictionary.

dic = {'Name':'Bhaskar','Branch':'C.S.E','Age':20}
df4 = pd.Series(dic)
print(df4)
```

```
0    10
1    20
2    30
3    40
4    50
dtype: int32
a    10
b    20
c    30
d    40
e    50
dtype: int32
a    50
b    50
c    50
d    50
e    50
dtype: int64
Name      Bhaskar
Branch      C.S.E
Age            20
dtype: object
```

In [8]:
```
# Mathematical operations in Series.

s = pd.Series([1,2,3,4,5])

# Print all the values of the Series by multiplying them by 2.

print(s*2)

# Print Square of all the values of the series.

print(s**2)

# Print all the values of the Series that are greater than 2

print(s[s>2])
```

```
0     2
1     4
2     6
3     8
4    10
dtype: int64
0     1
1     4
2     9
3    16
4    25
dtype: int64
2    3
3    4
4    5
dtype: int64
```

In [16]:
```
import pandas as pd

s1 = pd.Series([1,2,3,4,5], index=['a','b','c','d','e'])
s2 = pd.Series([1,2,3,4,5], index=['a','b','c','d','e'])
s3 = pd.Series([1,2,3,4], index=['a','b','c','d'])

# Adding two Series

print(s1+s2)

# While adding two series, if Non-Matching Index is found in either of the
# Series, Then NaN will be printed corresponds to Non-Matching Index.

print(s2+s3)

#If Non-Matching Index is found in either of the series,
#then this NonMatching Index corresponding value of that series will be filled as 0

print(s2.add(s3,fill_value=0))
```

```
a     2
b     4
c     6
d     8
e    10
dtype: int64
a    2.0
b    4.0
c    6.0
d    8.0
e    NaN
dtype: float64
a    2.0
b    4.0
c    6.0
d    8.0
e    5.0
dtype: float64
```

In [26]:
```
# Creating a sample DataFrame

data = np.array([10,20,30,40,50,60,70])

df5 = pd.DataFrame(data)

# Printing the first 3 rows using head() in Python

print(df5.head(3))

# Printing the first 5 rows using head() in Python

print("\n\n",df5.head())

# Printing the last 2 rows using tail() in Python

print("\n",df.tail(2))
```

```
   0
0  10
1  20
2  30


    0
0  10
1  20
2  30
3  40
4  50

 3    40
4    50
dtype: int32
```

In [23]:
```
# Selection in Series.

# 1. loc index label :-
# Syntax:-series_name.loc[StartRange: StopRange]

s = pd.Series([1,2,3,4,5])

# To Print Values from Index 0 to 2.
print(s.loc[:2])
print(s[0:3])

# To Print Values from Index 3 to 4.
print(s.loc[3:4])
print(s[3:])
```

```
0    1
1    2
2    3
dtype: int64
0    1
1    2
2    3
dtype: int64
3    4
4    5
dtype: int64
3    4
4    5
dtype: int64
```

In [28]:
```
# Indexing in Series.
# Pandas provide index attribute to get or set the index of entries or values in se

s4 = pd.Series([1,2,3,4,5], index=['a','b','c','d','e'])

print(s4.index)
print(s4.values)
```

```
Index(['a', 'b', 'c', 'd', 'e'], dtype='object')
[1 2 3 4 5]
```

In [34]:
```
# Slicing is a way to retrieve subsets of data from a pandas object. A
# slice object syntax is –

# SERIES_NAME [start:end: step]

s4 = pd.Series([1,2,3,4,5], index=['a','b','c','d','e'])

print("\n",s4[1:5:2])
print("\n",s4[0:6:2])

print("\n",s4[-1])
print("\n",s4[-4:-1])
```

```
 b    2
d    4
dtype: int64

 a    1
c    3
e    5
dtype: int64

 5

 b    2
c    3
d    4
dtype: int64
```

In [ ]: