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

In [22]:
```
a=pd.Series([1,3,5,7,9],index=["a","b","c","d","e"])
print("Series object in pandas")
print(a)
print("Element with index 'a':",a["a"])#accessing element with index
print("Is index b in Series object:","b" in a)#contains that particular key or not
print("Is index i in Series object:","i" in a)#contains that particular key or not
print("List of key values:",a.keys())#list of key values is printed by keys function
print("list of keys with values:",list(a.items()))#we use list of functio to connvert items object to list
```

```
Series object in pandas
a    1
b    3
c    5
d    7
e    9
dtype: int64
Element with index 'a': 1
Is index b in Series object: True
Is index i in Series object: False
List of key values: Index(['a', 'b', 'c', 'd', 'e'], dtype='object')
list of keys with values: [('a', 1), ('b', 3), ('c', 5), ('d', 7), ('e', 9)]
```

In [34]:
```
#series as oe dimensional array
#slicing by explicit array
print("Index slicing from 'b' to 'd':")#slicing using indexes,unlike normal integer slcing last element is included
print(a["b":"d"])
# slicing by implicit integer index
print("First 3 elements of series objects")#slicing using integer indexes
print(a[0:3])
#masking
print("All elements above 5:")
print(a[a>5])#printing all elements
#fancy indexing
print("Printing elements at index 'a' and 'e':")
print(a[["a","e"]])#printing elements at index "a" and "e"
```

```
Index slicing from 'b' to 'd':
b    3
c    5
d    7
dtype: int64
First 3 elements of series objects
a    1
b    3
c    5
dtype: int64
All elements above 5:
d    7
e    9
dtype: int64
Printing elements at index 'a' and 'e':
a    1
e    9
dtype: int64
```

In [50]:
```
a=pd.Series(["a","b","c","d","e"],index=[1,3,5,7,9])
print("Series object in pandas")
print(a)
#loc attribute-references explicit indexing
print("Value at key 1 using loc:")
print(a.loc[1])#returns value at key 1
print("Values from key 1 to 5")
print(a.loc[1:5])#returns values from key 1 and 5
```

```
Series object in pandas
1    a
3    b
5    c
7    d
9    e
dtype: object
Value at key 1 using loc:
a
Values from key 1 to 5
1    a
3    b
5    c
dtype: object
```

In [55]:
```
a=pd.Series([1,3,5,7,9],index=["a","b","c","d","e"])
print("Series object in pandas")
print(a)
#iloc attribute-references implicit indexing
print("Value at second key  using iloc:")
print(a.iloc[1])#returns value at second key
print("Values from second key to fourth key")
print(a.iloc[1:5])#returns values from second key to fourth key
```

```
Series object in pandas
a    1
b    3
c    5
d    7
e    9
dtype: int64
Value at second key  using iloc:
3
Values from second key to fourth key
b    3
c    5
d    7
e    9
dtype: int64
```

In [98]:
```
#dataframe as dictionary
name={1:"dairy milk",2:"5star",3:"shots",4:"mars"}
price={1:10,2:15,3:5,4:50}
name_object=pd.Series(name)
price_object=pd.Series(price)
choc_dataframe=pd.DataFrame({"name":name_object,"price":price_object})
print("Chocolate dataframe=")
print(choc_dataframe)
print("Name list=")
print(choc_dataframe["name"])#using indexing
print("Name list=")
print(choc_dataframe.name)#using object method
#adding new column in dataframe object
choc_dataframe["GST"]=choc_dataframe["price"]//10#adding a new column in dataframe
print("Dataframe after adding new column=")
print(choc_dataframe)
#all masking and slicing are implemented row wise
```

```
Chocolate dataframe=
         name  price
1  dairy milk     10
2       5star     15
3       shots      5
4        mars     50
Name list=
1    dairy milk
2         5star
3         shots
4          mars
Name: name, dtype: object
Name list=
1    dairy milk
2         5star
3         shots
4          mars
Name: name, dtype: object
Dataframe after adding new column=
         name  price  GST
1  dairy milk     10    1
2       5star     15    1
3       shots      5    0
4        mars     50    5
```

In [101]:
```
#dataframe as two dimesional array
print("All values as two dimensional array=")
print(choc_dataframe.values)
#transpose of dataframe
print("Transpose of dataframe=")
print(choc_dataframe.T)
#iloc and loc indexing in dataframe
#labels are maintained despite indexing
print("Details of chocolate with index 1:")
print(choc_dataframe.loc[1])#using explicit indexing
print("Details of second to third chocolate")
print(choc_dataframe.iloc[1:3])#using implicit indexing
#ix indexing is a hybrid of both loc and iloc
#print("Details from second chocolate to chocolate with index '4':")
#print(choc_dataframe.ix[2])#prone to confusions
```

```
All values as two dimensional array=
[['dairy milk' 10 1]
 ['5star' 15 1]
 ['shots' 5 0]
 ['mars' 50 5]]
Transpose of dataframe=
                1      2      3     4
name   dairy milk  5star  shots  mars
price          10     15      5    50
GST             1      1      0     5
Details of chocolate with index 1:
name     dairy milk
price            10
GST               1
Name: 1, dtype: object
Details of second to third chocolate
    name  price  GST
2  5star     15    1
3  shots      5    0
```