```
import pandas as pd
data = [10, 20, 30, 40, 50]
series = pd. Series (data)
print(series)
```

```
0    10
1    20
2    30
3    40
4    50
dtype: int64
```

```
import pandas as pd
a = [1, 7, 21]
myvar = pd.Series (a, index = ["x", "y", "z"])
print(myvar)
```

```
x     1
y     7
z    21
dtype: int64
```

```
gt=series > 25
print("series>25= \n",gt)
series_filtered = series [series > 25]
print("Filtered Series: \n", series_filtered)
# Scalar multiplication
series_multiplied = series * 2
print("\nSeries after Scalar Multiplication: \n", series_multiplied)
```

```
series>25=
 0    False
1    False
2     True
3     True
4     True
dtype: bool
Filtered Series:
 2    30
3    40
4    50
dtype: int64

Series after Scalar Multiplication:
 0     20
1     40
2     60
3     80
4    100
dtype: int64
```

```
import numpy as np
print("series= \n",series)
series_sqrt = np.sqrt(series)
print("\nSeries after Applying Square Root: \n", series_sqrt)
```

```
series=
 0    10
1    20
2    30
3    40
4    50
dtype: int64

Series after Applying Square Root:
 0    3.162278
1    4.472136
2    5.477226
3    6.324555
4    7.071068
dtype: float64
```

```
import pandas as pd
data = {
"calories": [420, 380, 390],
"duration": [50, 40, 45] }
#load data into a DataFrame object:
df = pd.DataFrame(data)
print(df)
```

```
   calories  duration
0       420        50
1       380        40
2       390        45
```

```
import pandas as pd
obj = pd. Series ([4.5, 7.2, -5.3, 3.6], index= ['d', 'b', 'a', 'c'])
print(obj)
```

```
d    4.5
b    7.2
a   -5.3
c    3.6
dtype: float64
```

```
import pandas as pd
df1 = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6]})
df2 = pd.DataFrame({'A': [7, 8, 9], 'B': [10, 11, 12]})
print(df1)
print(df2)
print("Addition: \n", df1+df2)
print("\nSubtraction: \n", df1 - df2)
```

```
print("\nMultiplication: \n", df1*df2)
print("\nDivision: \n", df1 / df2)
```

```
   A  B
0  1  4
1  2  5
2  3  6
   A   B
0  7  10
1  8  11
2  9  12
Addition:
     A   B
0   8  14
1  10  16
2  12  18

Subtraction:
    A  B
0 -6 -6
1 -6 -6
2 -6 -6

Multiplication:
     A   B
0   7  40
1  16  55
2  27  72

Division:
           A         B
0  0.142857  0.400000
1  0.250000  0.454545
2  0.333333  0.500000
```

```
import pandas as pd
df3 = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6]}, index= [0, 1, 2])

df4= pd.DataFrame({'B': [7, 8, 9], 'C': [10, 11, 12]}, index= [1, 2,
3])
print(df3)
print(df4)
result = df3 + df4
print("Result of df3df4 with alignment: \n", result)
```

```
   A  B
0  1  4
1  2  5
2  3  6
   B   C
```

```
1  7  10
2  8  11
3  9  12
Result of df3df4 with alignment:
     A     B   C
0 NaN   NaN NaN
1 NaN  12.0 NaN
2 NaN  14.0 NaN
3 NaN   NaN NaN
```

```
result_add = df3.add(df4, fill_value=0)
print("\nResult of df3.add(df4) with fill_value=0:\n", result_add)
```

```
Result of df3.add(df4) with fill_value=0:
      A     B     C
0  1.0   4.0   NaN
1  2.0  12.0  10.0
2  3.0  14.0  11.0
3  NaN   9.0  12.0
```

```
import pandas as pd
obj = pd.Series (range(4), index=['d', 'a', 'b', 'c'])
df_sort_series = obj.sort_index()
print (df_sort_series)
```

```
a    1
b    2
c    3
d    0
dtype: int64
```

```
import pandas as pd
obj = pd.Series ([7, -5, 7, 3, 2, 0, 41])
df_rank =obj.rank()
print("Rank \n", df_rank)
```

```
Rank
 0    5.5
1    1.0
2    5.5
3    4.0
4    3.0
5    2.0
6    7.0
dtype: float64
```

```
import pandas as pd
obj = pd.Series (np.arange(4.), index=['a', 'b', 'c', 'd'])
print("obj ['b']= \n",obj ['b'])
print("obj [1]= \n", obj [1])
```

```
print("obj [2:4]= \n",obj [2:4])
print("obj [['b', 'a', 'd']]= \n", obj [['b', 'a', 'd']])
print("obj [b:d]= \n",obj ['b': 'd']) #no n-1 concept
print("obj [[1, 3]]= \n", obj [[1, 3]])
print("obj [obj < 2]= \n", obj [obj < 2])
```

```
obj ['b']=
 1.0
obj [1]=
 1.0
obj [2:4]=
 c    2.0
d    3.0
dtype: float64
obj [['b', 'a', 'd']]=
 b    1.0
a    0.0
d    3.0
dtype: float64
obj [b:d]=
 b    1.0
c    2.0
d    3.0
dtype: float64
obj [[1, 3]]=
 b    1.0
d    3.0
dtype: float64
obj [obj < 2]=
 a    0.0
b    1.0
dtype: float64

C:\Users\rasir\AppData\Local\Temp\ipykernel_14136\3635843064.py:4:
FutureWarning: Series.__getitem__ treating keys as positions is
deprecated. In a future version, integer keys will always be treated
as labels (consistent with DataFrame behavior). To access a value by
position, use `ser.iloc[pos]`
  print("obj [1]= \n", obj [1])
C:\Users\rasir\AppData\Local\Temp\ipykernel_14136\3635843064.py:8:
FutureWarning: Series.__getitem__ treating keys as positions is
deprecated. In a future version, integer keys will always be treated
as labels (consistent with DataFrame behavior). To access a value by
position, use `ser.iloc[pos]`
  print("obj [[1, 3]]= \n", obj [[1, 3]])
```

```
import pandas as pd
obj3 = pd.Series(['blue', 'purple', 'yellow'], index=[0, 2, 4])
print(obj3)
obj4= obj3. reindex (range(6), method='ffill') #forward-fills the
```

```
values
obj4

0      blue
2    purple
4    yellow
dtype: object

0      blue
1      blue
2    purple
3    purple
4    yellow
5    yellow
dtype: object
```