In [11]:
```python
#operating on data in numpy
import pandas as pd
import numpy as np
ser=pd.Series([1,4,7,0])#creating series object
dataf=pd.DataFrame(np.arange(0,12).reshape(3,4),columns=["a","b","c","d"])#creating dataframe object
print("Series object in pandas=")
print(ser)
print("Dataframe object in pandas=")
print(dataf)
#performing basic ufuncs on pandas
print("Dividing each element by 2 in series object")
print(ser/2)
#multiplying each element by 3 in dataframe object
print("Multiplying each element by 3 in dataframe object")
print(dataf*3)
```

```
Series object in pandas=
0    1
1    4
2    7
3    0
dtype: int64
Dataframe object in pandas=
   a  b   c   d
0  0  1   2   3
1  4  5   6   7
2  8  9  10  11
Dividing each element by 2 in series object
0    0.5
1    2.0
2    3.5
3    0.0
dtype: float64
Multiplying each element by 3 in dataframe object
    a   b   c   d
0   0   3   6   9
1  12  15  18  21
2  24  27  30  33
```

In [17]:
```
#operations on more than one dataframes and index alignment
height={"a":6.1,"b":5.4,"c":6,"d":5.7}
height_obj=pd.Series(height)
weight={"a":75,"e":84,"c":67,"f":90}
weight_obj=pd.Series(weight)
print("Height series object=")
print(height_obj)
print("Weight series object=")
print(weight_obj)
print("Dividing height and weight object=")
print(height_obj/weight_obj)
#first union of indeices are taken and then if both values are there divided else NaN
#index matching is implemented using NaN(not a number) value
ser1=pd.Series(["a","b","c","d"],index=[1,2,3,4])
ser2=pd.Series(["a","b","c","d"],index=[2,3,4,5])
print("ser1=")
print(ser1)
print("ser2=")
print(ser2)
print("adding ser1 and ser2=")
print(ser1+ser2)
```

```
Height series object=
a    6.1
b    5.4
c    6.0
d    5.7
dtype: float64
Weight series object=
a    75
e    84
c    67
f    90
dtype: int64
Dividing height and weight object=
a    0.081333
b         NaN
c    0.089552
d         NaN
e         NaN
f         NaN
dtype: float64
ser1=
1    a
2    b
3    c
4    d
dtype: object
ser2=
2    a
3    b
4    c
5    d
dtype: object
adding ser1 and ser2=
1    NaN
2     ba
3     cb
4     dc
5    NaN
dtype: object
```

In [20]:
```
#instead of NaN we can also fill in approproate values by calling ufunc as attribute and passing fill value
ser1=pd.Series([1,2,3,4],index=["a","b","c","d"])
ser2=pd.Series([5,6,7,8],index=["a","b","e","f"])
print("ser1=")
print(ser1)
print("ser2=")
print(ser2)
print("adding ser1 and ser2=")
print(ser1.add(ser2,fill_value=0))#if no value 0 is subsituted
```

```
ser1=
a    1
b    2
c    3
d    4
dtype: int64
ser2=
a    5
b    6
e    7
f    8
dtype: int64
adding ser1 and ser2=
a    6.0
b    8.0
c    3.0
d    4.0
e    7.0
f    8.0
dtype: float64
```

In [35]:
```
#allignment in dataframe
A=pd.DataFrame(np.random.randint(0,10,(2,5)),columns=list('ABCDE'))
print("A=")
print(A)
B=pd.DataFrame(np.random.randint(0,10,(2,5)),columns=list('BCDFG'))
print("B=")
print(B)
print("A+B=")
print(A+B)
print("A*B with 0 when data is missing=")
print(A.multiply(B,fill_value=0))
```

```
A=
   A  B  C  D  E
0  8  9  6  2  5
1  3  1  5  7  8
B=
   B  C  D  F  G
0  3  9  8  8  3
1  2  7  0  6  3
A+B=
    A   B   C   D   E   F   G
0 NaN  12  15  10 NaN NaN NaN
1 NaN   3  12   7 NaN NaN NaN
A*B with 0 when data is missing=
     A   B   C   D    E    F    G
0  0.0  27  54  16  0.0  0.0  0.0
1  0.0   2  35   0  0.0  0.0  0.0
```