On the object of Ndarray we can perform the following Statistical Operations:

Note: Here Axis 0= Column, Axis 1= Row

1. amax()

2. amin()

3. mean()

4. median()

5. var()

6. Std()

In [1]:
```
import numpy as np
a=np.array([[10,50,30],[20,40,60],[15,25,35]])
print(a,type(a))
```

```
[[10 50 30]
 [20 40 60]
 [15 25 35]] <class 'numpy.ndarray'>
```

In [ ]:
```
#Statistical Operations
```

1. amax()

Syntax1: varname= numpy.amax(ndarrayobj) or

Syntax2: varname= numpy.amax(ndarrayobj, axis=0/ axis =1)

In [2]:
```
maxv= np.amax(a)
print("max val= ",maxv)
```

```
max val=  60
```

In [ ]:
```
# to calculate the mac value from each column separately
```

In [3]:
```
colmaxv= np.amax(a,axis=0)
print("Column Max Value= ",colmaxv)
```

```
Column Max Value=  [20 50 60]
```

In [5]:
```
rowmaxv= np.amax(a,axis=1)
print("Row Max Value= ",rowmaxv)
```

```
Row Max Value=  [50 60 35]
```

2. amin()

Syntax1: varname= numpy.amin(ndarrayobj) or

Syntax2: varname= numpy.amin(ndarrayobj, axis=0/ axis =1)

In [6]:
```
print(a)
```

```
[[10 50 30]
 [20 40 60]
 [15 25 35]]
```

In [7]:
```
minv= np.amax(a)
print("Min value in Matrix =", minv)
```

```
Min value in Matrix = 60
```

In [8]:
```
colminv= np.amax(a,axis =0)
print("Column min value in Matrix = ",colminv)
```

```
Column min value in Matrix =  [20 50 60]
```

In [9]:
```
rowminv= np.amax(a,axis =1)
print("Row min value in Matrix =  ",rowminv)
```

```
Row min value in Matrix =  [50 60 35]
```

In [12]:
```
minv= np.amin(a)
colminv= np.amin(a,axis =0)
rowminv= np.amin(a,axis =1)
print("Min Value= ",minv)
print("Column min values in Matrix = ",colminv)
print("Row min values in Matrix = ",rowminv)
```

```
Min Value=  10
Column min values in Matrix =  [10 25 30]
Row min values in Matrix =  [10 20 15]
```

3. mean()

Syntax1: varname= numpy.mean(ndarrayobj) or

Syntax2: varname= numpy.mean(ndarrayobj,axis =1)=====> axis=0 for column values, axis=1 for rows values

In [13]:
```
a=np.array([[3,2],[1,4]])
print(a,type(a))
```

```
[[3 2]
 [1 4]] <class 'numpy.ndarray'>
```

In [14]:
```
a.mean()
```

Out[14]:
```
2.5
```

In [17]:
```
meanval = np.mean(a)
print("Mean Value = ",meanval)
```

```
Mean Value =  2.5
```

In [19]:
```
colmean= np.mean(a,axis=0)
rowmean= np.mean(a,axis=1)
print("Column Mean value = ",colmean)
print("Row Mean Value = ",rowmean)
```

```
Column Mean value =  [2. 3.]
Row Mean Value =  [2.5 2.5]
```

4. median()

Syntax1: varname= numpy.median(ndarrayobj) or

Syntax2: varname= numpy.median(ndarrayobj,axis=0)=====> axis=0 for column values, axis=1 for rows values

In [20]:
```
a=np.array([[10,50,30],[20,40,60],[15,25,35]])
print(a,type(a))
```

```
[[10 50 30]
 [20 40 60]
 [15 25 35]] <class 'numpy.ndarray'>
```

In [23]:
```
medv=np.median(a)  # It will sort data in ascending/decending order and pull the middle element is have odd number,
print("Median Value =",medv)   #if even then middle two--> add and divide by 2
```

```
Median Value = 30.0
```

In [24]:
```
a=np.array([[50,30],[20,60],[15,35]])
print(a,type(a))
```

```
[[50 30]
 [20 60]
 [15 35]] <class 'numpy.ndarray'>
```

In [25]:
```
medv= np.median(a)
print("Median value = ",medv)
```

```
Median value =  32.5
```

In [26]:
```
colmedv= np.median(a,axis=0)
print("Column Median value = ",colmedv)
rowmedv= np.median(a,axis=1)
print("Row Median Value = ",rowmedv)
```

```
column Median value =  [20. 35.]
Row Median Value =  [40. 40. 25.]
```

5. variance()

variance= sqr(mean-xi)/Total number of elements ===> where "xi" represent each element of matrix

Syntax1: varname= numpy.variance(ndarrayobj) or

Syntax2: varname= numpy.variamce(ndarrayobj,axis=0)=====> axis=0 for column values, axis=1 for rows values

In [27]:
```
a=np.array([[3,2],[1,4]])
print(a,type(a))
```

```
[[3 2]
 [1 4]] <class 'numpy.ndarray'>
```

In [28]:
```
v=np.var(a)
print("Variance = ",v)
```

```
Variance =  1.25
```

In [30]:
```
cv= np.var(a,axis=0) # cv = Column wise variance
rv= np.var(a,axis=1) # rv= Row wise variance
print("column wise variance = ",cv)
print("Row wise variance = ",rv)
```

```
column wise variance =  [1. 1.]
Row wise variance =  [0.25 2.25]
```

6. std()

Standard Deviation= Sqrt(variance)

Syntax1: varname= numpy.std(ndarrayobj) or

Syntax2: varname= numpy.std(ndarrayobj,axis=0)=====> axis=0 for column values, axis=1 for rows values

In [31]:
```
a=np.array([[3,2],[1,4]])
print(a,type(a))
```

```
[[3 2]
 [1 4]] <class 'numpy.ndarray'>
```

In [33]:
```
sd=np.std(a)  # sd means value of Standard deviation
print("Standard Deviation of a =",sd)
```

```
Standard Deviation of a = 1.118033988749895
```

In [35]:
```
csd= np.std(a,axis=0) # csd = Column wise standard Deviation
rsd= np.std(a,axis=1) # rv= Row wise standard Deviation
print("column wise standard Deviation = ",csd)
print("Row wise standard Deviation = ",rsd)
```

```
column wise standard Deviation =  [1. 1.]
Row wise standard Deviation =  [0.5 1.5]
```

In [37]:
```
np.mode(a)  # Mode Attribute not avaiable in Numpy . It is present in module called statistics
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_24192\769789981.py in <module>
----> 1 np.mode(a)  # Mode Attribute not avaiable in Numpy . It is present in module called statistics

C:\ProgramData\Anaconda3\lib\site-packages\numpy\__init__.py in __getattr__(attr)
    311                 return Tester
    312
--> 313             raise AttributeError("module {!r} has no attribute "
    314                                  "{!r}".format(__name__, attr))
    315

AttributeError: module 'numpy' has no attribute 'mode'
```

In [38]:
```
#mode() is used to get the frequenly repeated element. we need to import statistics to use mode()
import statistics as s
```

In [40]:
```
a=np.array([10,20,30,10,40,50,60,10])
print(a,type(a))
```

```
[10 20 30 10 40 50 60 10] <class 'numpy.ndarray'>
```

In [41]:
```
mv=s.mode(a)  #mv= mode value
print("Mode value = ",mv)
```

```
Mode value =  10
```

In [46]:
```
a=np.array([10,20,30,10,40,20,50,60,20,10])    # we have multiple repeated values, equal number of times. so use multimode concept
print(a,type(a))
```

```
[10 20 30 10 40 20 50 60 20 10] <class 'numpy.ndarray'>
```

In [47]:
```
mv=s.mode(a)  #mv= mode value , we have 10 and 20 repeatimg equal number of times so we shouldnot use mode().
print("Mode value = ",mv)
```

```
Mode value =  10
```

In [48]:
```
mv=s.multimode(a)  #mv= mode value
print("multiMode value = ",mv)
```

```
multiMode value =  [10, 20]
```

In [ ]:
```

```