# Append

```
appends the values at the last index of the array
```

In [3]:
```
import numpy as np
array = np.array([2,1,4,5,7])
np.append(array,100)
```

Out[3]:
```
array([  2,   1,   4,   5,   7, 100])
```

In [4]:
```
array = np.array([2,1,4,5,7])
array2 = np.array([100,200])
new_array = np.append(array,array2)
new_array
```

Out[4]:
```
array([  2,   1,   4,   5,   7, 100, 200])
```

## for multidimantional array

#### adding all elements from both matrix

In [17]:
```
array1 = np.array([[2,1],
                   [4,5]])
array2 = np.array([[100,200,300]])

np.append(array1,array2)
```

Out[17]:
```
array([  2,   1,   4,   5, 100, 200, 300])
```

#### adding elents row-wise

```
# for row wise addtion, number of columns from both the array or matrix has to be equal
```

In [19]:
```
array1 = np.array([[2,1],
                   [4,5]])
array2 = np.array([[100,200]])

np.append(array1,array2,axis=0)
```

Out[19]:
```
array([[  2,   1],
       [  4,   5],
       [100, 200]])
```

In [20]:
```
array1 = np.array([[2,1],
                   [4,5]])
array2 = np.array([[100,200,300]])

np.append(array1,array2,axis=0)
```

...

**adding elements column wise**

```
# for column wise addition, number of rows has to be equal
```

In [146]:
```
array1 = np.array([[2,1],
                   [4,5]])
array2 = np.array([[100],
                   [200]])

np.append(array1,array2,axis=1)
```

Out[146]:
```
array([[  2,   1, 100],
       [  4,   5, 200]])
```

In [24]:
```
array1 = np.array([[2,1],
                   [4,5]])
array2 = np.array([[100],
                   [200],
                   [300]])

np.append(array1,array2,axis=1)
```

...

# concatenate()

```
concates the arrays along the axis
```

**for row_wise addition**

```
# number of columns in both arrays has to be equal
```

In [35]:
```
array1 = np.array([[2,1],
                   [4,5]])
array2 = np.array([[100,200]])

np.concatenate([array1,array2],axis=0)
```

Out[35]:
```
array([[  2,   1],
       [  4,   5],
       [100, 200]])
```

**for column wise addition**

```
# number of rows in both arrays has to be equal
```

In [40]:
```
array1 = np.array([[2,1],
                   [4,5]])
array2 = np.array([[100],
                   [200]])

np.concatenate([array1,array2],axis=1)
```

Out[40]:
```
array([[  2,   1, 100],
       [  4,   5, 200]])
```

# for loop

In [52]:
```
array = np.array([2,1,4,5,7])
arr2 = np.array([[  2,   1, 100],
                 [  4,   5, 200]])

for i in array:
    print(i)
print()
for i in arr2:
    print(i)
```

```
2
1
4
5
7

[  2   1 100]
[  4   5 200]
```

## np.nditer()

In [55]:
```
for i in np.nditer(array):
    print(i)

print()
for i in np.nditer(arr2):
    print(i)
```

```
2
1
4
5
7

2
1
100
4
5
200
```

In [47]:
```
for i in range(0,len(array)):
    print(array[i])
```

```
2
1
4
5
7
```

In [49]:
```

```

```
2
1
4
5
7
```

## np.ndenumerate()

In [58]:
```
array = np.array([2,1,4,5,7])
arr2 = np.array([[  2,   1, 100],
                 [  4,   5, 200]])

for i,j in enumerate(array):
    print(i,j)

print()
for i,j in enumerate(arr2):
    print(i,'>>',j)
```

```
0 2
1 1
2 4
3 5
4 7

0 >> [  2   1 100]
1 >> [  4   5 200]
```

In [59]:
```
for index, value in np.ndenumerate(array):
    print(index,value)

print()
for index, value in np.ndenumerate(arr2):
    print(index,value)
```

```
(0,) 2
(1,) 1
(2,) 4
(3,) 5
(4,) 7

(0, 0) 2
(0, 1) 1
(0, 2) 100
(1, 0) 4
(1, 1) 5
(1, 2) 200
```

# ceil()

```
will yield out the maximum whole number that is closest to that decimal number
```

In [61]:
```
x = 2.756
np.ceil(x)
```

Out[61]: 3.0

In [62]:
```
x = -2.756
np.ceil(x)
```

Out[62]: -2.0

In [63]:
```
x = 2.5
np.ceil(x)
```

Out[63]: 3.0

In [64]:
```
x = 2.00000001
np.ceil(x)
```

Out[64]: 3.0

In [65]:
```
x = np.array([-2.574,9,4,2])
np.ceil(x)
```

Out[65]: array([-2., 9., 4., 2.])

In [66]:
```
x = np.array([[-2.5,9],
              [4.8,2]])
np.ceil(x)
```

Out[66]:
```
array([[-2.,  9.],
       [ 5.,  2.]])
```

# floor

```
will yield out the minimum closest whole number to that decimal number
```

In [67]:
```
x = 2.756
np.floor(x)
```

Out[67]: 2.0

In [68]:
```
x = -2.756
np.floor(x)
```

Out[68]: -3.0

In [69]:
```
x = 2.5
np.floor(x)
```

Out[69]: 2.0

In [70]:
```
x = 2.9999999
np.floor(x)
```

Out[70]: 2.0

In [71]:
```
x = np.array([[-2.5,9],
              [4.8,2]])
np.floor(x)
```

Out[71]:
```
array([[-3.,  9.],
       [ 4.,  2.]])
```

# around()

```
works exactly same as round function
for decimal part less than pr equal to .5, it choses the closest minimum whole number & for otherwise it choses maximum closest whole number
```

**round & around being used for rounding of to closest whole number**

In [72]:
```
round(2.2)
```

Out[72]: 2

In [73]:
```
np.around(2.2)
```

Out[73]: 2.0

In [74]:
```
round(2.8)
```

Out[74]: 3

In [75]:
```
np.around(2.8)
```

Out[75]: 3.0

In [76]:
```
round(2.5)
```

Out[76]: 2

In [79]:
```
np.around(2.5)
```

Out[79]: 2.0

In [81]:
```
x = [[2.5, -6.7],
     [8.9, -1.4]]
np.around(x)
```

Out[81]:
```
array([[ 2., -7.],
       [ 9., -1.]])
```

**round & around being used for rounding off to specific decimals**

In [82]:
```
round(3.4394623948,2)
```

Out[82]: 3.44

In [84]:
```
np.around(3.4394623948,2)
```

Out[84]: 3.44

In [86]:
```
x = [[2.52083, -6.7300874],
     [8.93279, -1.4397439]]

x = np.around(x,3)
x
```

Out[86]:
```
array([[ 2.521, -6.73 ],
       [ 8.933, -1.44 ]])
```

## copy & deepcopy

```
# for numpy array copy & deepcopy both works the same as deepcopy
```

In [93]:
```
x = np.array([[2.52083, -6.7300874],
              [8.93279, -1.4397439]])

y = np.copy(x)
y[0,0]=100
print(x)
print(y)
```

```
[[ 2.52083   -6.7300874]
 [ 8.93279   -1.4397439]]
[[100.         -6.7300874]
 [  8.93279    -1.4397439]]
```

In [94]:
```
x = np.array([[2.52083, -6.7300874],
              [8.93279, -1.4397439]])

y = x.copy()
y[0,0]=100
print(x)
print(y)
```

```
[[ 2.52083   -6.7300874]
 [ 8.93279   -1.4397439]]
[[100.         -6.7300874]
 [  8.93279    -1.4397439]]
```

In [95]:
```
from copy import copy
x = np.array([[2.52083, -6.7300874],
              [8.93279, -1.4397439]])

y = copy(x)
y[0,0]=100
print(x)
print(y)
```

```
[[ 2.52083   -6.7300874]
 [ 8.93279   -1.4397439]]
[[100.         -6.7300874]
 [  8.93279    -1.4397439]]
```

# np.where()

```
# it returns the index of values that meet the given condition
```

In [96]:
```
arr = np.array([2,3,5,8,5,1,0])
np.where(arr>4)
```

Out[96]:
```
(array([2, 3, 4], dtype=int64),)
```

In [97]:
```
arr = np.array([2,3,5,8,5,1,0])
np.where(arr==5)
```

Out[97]:
```
(array([2, 4], dtype=int64),)
```

In [104]:
```
arr = np.array([2,3,5,8,5,1,0])
print('indexes that meet condition are: ',np.where(arr%2==0))
print('values in array that meet conditions are ',arr[np.where(arr%)]2==0)
```

```
indexes that meet condition are:  (array([0, 3, 6], dtype=int64),)
values in array that meet conditions are  [2 8 0]
```

In [100]:
```
np.array([i for i in np.nditer(arr) if i%2==0])
```

Out[100]:
```
array([2, 8, 0])
```

### replacing values in array using np.where function

```
Syntax:
    np.where(condition,if meets,if does not meet)
    np.where(condition,True,False)
    output are the elements of the array & not the indices
```

In [120]:
```
arr = np.array([1,2,3,4,5,6])

arr = np.where(arr%2==0,'even','odd')
arr
```

Out[120]:
```
array(['odd', 'even', 'odd', 'even', 'odd', 'even'], dtype='<U4')
```

In [119]:
```
players = np.array(['sachin','sehvag','tendulkar'])

np.where(players=='tendulkar','10dulkar',players)
```

Out[119]:
```
array(['sachin', 'sehvag', '10dulkar'], dtype='<U9')
```

In [122]:
```
arr = np.array([-1,2,3,-4,5,6,3,-2,0])
np.where(arr<0,0,arr)
```

Out[122]:
```
array([0, 2, 3, 0, 5, 6, 3, 0, 0])
```

In [125]:
```
array = np.array([0, 2, 3, 0, 5, 6, 3, 0, 0])

df1 = pd.DataFrame(array,columns={'values'})
df1.to_csv('df1.csv')

array2 = np.where(array==0,np.NaN,array)
array2

df2 = pd.DataFrame(array2,columns={'VALUES'})
df2.to_csv('df2.csv')
```

In [117]:
```
arr = np.array([2,3,5,8,5,1,0])
import pandas as pd
df = pd.DataFrame(arr,columns={'values'})
df.loc[np.where(df['values']==5)]
```

Out[117]:

| | values |
|---|---|
| **2** | 5 |
| **4** | 5 |

In [111]:
```
df[df['values']==5]
```

Out[111]:

| | values |
|---|---|
| **2** | 5 |
| **4** | 5 |

In [118]:
```
df['values'] = np.where(df['values']==5,True,False)
df
```

...

**argwhere functions**

```
returns the indices of the values that meets the condition in the array
```

In [130]:
```
import numpy as np
arr = np.array([9,5,3,1,7,8,4])
np.argwhere(arr>2)
```

Out[130]:
```
array([[0],
       [1],
       [2],
       [4],
       [5],
       [6]], dtype=int64)
```

In [131]:
```
(arr[np.argwhere(arr>2)]).flatten()
```

Out[131]:
```
array([9, 5, 3, 7, 8, 4])
```

## Size() & Shape()

In [133]:
```
arr = np.array([9,5,3,1,7,8,4])
np.size(arr)
```

Out[133]:
```
7
```

In [144]:
```
arr = np.array([[9,5,3],[1,7,8],[4,7,8]])
print(len(arr))
print(np.shape(arr))
np.size(arr)
```

```
3
(3, 3)
```

Out[144]:
```
9
```

## statistical functions

```
min,max,mean,avergae,median,mode,sum,len,std,var
```

In [135]:
```
arr = np.array([9,5,3,1,7,8,4])
np.min(arr)
```

Out[135]:
```
1
```

In [136]:
```
arr = np.array([9,5,3,1,7,8,4])
np.max(arr)
```

Out[136]:
```
9
```

In [137]:
```
arr = np.array([9,5,3,1,7,8,4])
np.sum(arr)
```

Out[137]:
```
37
```

In [138]:
```
arr = np.array([9,5,3,1,7,8,4])
np.mean(arr)
```

Out[138]: 5.285714285714286

In [139]:
```
arr = np.array([9,5,3,1,7,8,4])
np.average(arr)
```

Out[139]: 5.285714285714286

In [140]:
```
arr = np.array([9,5,3,1,7,8,4])
np.median(arr)
```

Out[140]: 5.0

In [141]:
```
arr = np.array([9,5,3,1,7,8,4])
np.std(arr)
```

Out[141]: 2.657296462534039

$$\sigma = \sqrt{\frac{\sum (x - mean)^2}{n}}$$

x is a set of numbers

mean is the average of the set of numbers

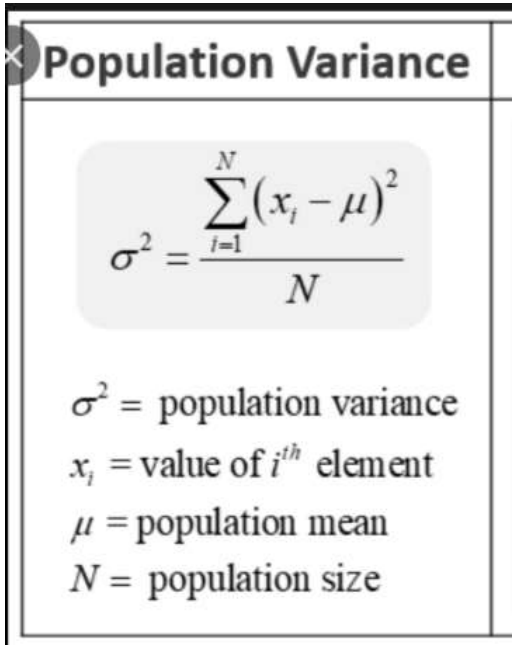
n is the size of the set

σ is the standard deviation

]

In [142]:
```
arr = np.array([9,5,3,1,7,8,4])
np.var(arr)
```

Out[142]: 7.061224489795918




In [145]:
```
1 np.shape(arr)[0]  # equivalent to len function
```

Out[145]: 3

In [ ]:
```
1
```