In [ ]:
```
# Access array elements
```

In [2]:
```
# print the first element of an array

import numpy as np
a1=np.array([11,22,33,44,55])
print(a1[0])
```

```
11
```

In [4]:
```
# get second and forth element from the array and add
print(a1[1]+a1[3])
```

```
66
```

In [6]:
```
# get second and forth element from the array and subtract

print(a1[3]-a1[1])
```

```
22
```

In [ ]:
```
# slicing
```

In [8]:
```
a2=np.arange(0,25,2)
a2
```

Out[8]:
```
array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18, 20, 22, 24])
```

In [9]:
```
print(a2[2:])
```

```
[ 4  6  8 10 12 14 16 18 20 22 24]
```

In [10]:
```
print(a2[::-1])
```

```
[24 22 20 18 16 14 12 10  8  6  4  2  0]
```

In [11]:
```
print(a2[1:8:2])
```

```
[ 2  6 10 14]
```

In [12]:
```
a3=np.array([[1,2,3,4,5],[6,7,8,9,10]])
a3
```

Out[12]:
```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10]])
```

In [13]:
```
print(a3[0,1:4])
```

```
[2 3 4]
```

In [14]:
```
print(a3[1,1:4])
```

```
[7 8 9]
```

Data Types in NumPy

- i - integers
- b - boolean
- u - unsigned integers
- f - float
- c - complex

- M - datetime
- O - object
- S - string
- U unicode string

In [17]:
```
a5=np.array(['sandeep','hari','meera'])
print(a5.dtype)
```

```
<U7
```

In [18]:
```
print(a1.dtype)
```

```
int32
```

In [19]:
```
# to specify a data type

arr=np.array([1,2,3,4],dtype='S')
print(arr)
print(arr.dtype)
```

```
[b'1' b'2' b'3' b'4']
|S1
```

In [20]:
```
arr1=np.array([1,2,3,4],dtype='i')
```

In [22]:
```
print(arr1)
print(arr1.dtype)
```

```
[1 2 3 4]
int32
```

operation on array

In [24]:
```
a10=np.arange(0,20)
a10
```

Out[24]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19])
```

In [25]:
```
# sum of all elemnts
a10.sum()
```

Out[25]:
```
190
```

In [26]:
```
# cummulative sum
np.cumsum(a10)
```

Out[26]:
```
array([  0,   1,   3,   6,  10,  15,  21,  28,  36,  45,  55,  66,  78,
        91, 105, 120, 136, 153, 171, 190], dtype=int32)
```

In [28]:
```
# minimum number in an array

a10.min()
```

Out[28]:
```
0
```

In [29]:
```
# to get the maximun
a10.max()
```

Out[29]:
```
19
```

In [30]:
```
# to find the mean of an array
a10.mean()
```

Out[30]: 9.5

In [31]:
```
# to find the median

np.median(a10)
```

Out[31]: 9.5

In [32]:
```
# to find the index of minmum number in an array
a10.argmin()
```

Out[32]: 0

In [33]:
```
# to find the index of maximum number
a10.argmax()
```

Out[33]: 19

In [34]:
```
# variance

np.var(a10)
```

Out[34]: 33.25

In [35]:
```
# standard deviation

np.std(a10)
```

Out[35]: 5.766281297335398

In [37]:
```
# calculate the percentile
np.percentile(a10,30)
```

Out[37]: 5.7

operation on 2-D array

In [40]:
```
a11=np.array([[2,9,34,4,56,7,8],[87,45,16,9,13,5,8]])
a11
```

Out[40]:
```
array([[ 2,  9, 34,  4, 56,  7,  8],
       [87, 45, 16,  9, 13,  5,  8]])
```

In [42]:
```
a11.sum()
```

Out[42]: 303

In [44]:
```
print(a11.cumsum())
```

```
[  2  11  45  49 105 112 120 207 252 268 277 290 295 303]
```

In [45]:
```
a11
```

Out[45]:
```
array([[ 2,  9, 34,  4, 56,  7,  8],
       [87, 45, 16,  9, 13,  5,  8]])
```

In [46]:
```
np.cumsum(a11)
```

Out[46]:
```
array([  2,  11,  45,  49, 105, 112, 120, 207, 252, 268, 277, 290, 295,
       303], dtype=int32)
```

In [47]:
```
a11.max()
```

Out[47]:
```
87
```

In [48]:
```
a11.min()
```

Out[48]:
```
2
```

In [50]:
```
a11.mean()
```

Out[50]:
```
21.642857142857142
```

In [52]:
```
np.median(a11)
```

Out[52]:
```
9.0
```

In [53]:
```
a11.argmin()
```

Out[53]:
```
0
```

In [54]:
```
a11.argmax()
```

Out[54]:
```
7
```

In [55]:
```
a11.var()
```

Out[55]:
```
581.2295918367347
```

In [56]:
```
np.var(a11)
```

Out[56]:
```
581.2295918367347
```

In [57]:
```
np.std(a11)
```

Out[57]:
```
24.10870365317751
```

In [58]:
```
a11.std()
```

Out[58]:
```
24.10870365317751
```

In [59]:
```
np.percentile(a11,100)
```

Out[59]:
```
87.0
```

copy of an array

In [60]:
```
c1=np.array([12,13,14,15,16])
c2=c1.copy()
c1[3]=20

print(c1)
print(c2)
```

```
[12 13 14 20 16]
[12 13 14 15 16]
```

view

- the view() should be affected by the changes made to the original array

In [61]:
```
c3=np.array([12,13,14,15,16])
c4=c3.view()   # ---> if you change in the original array it will reflect in copied
c1[3]=20

print(c3)
print(c4)
```

```
[12 13 14 15 16]
[12 13 14 15 16]
```

Difference between copy and view

- copy is a new array and view is just a view of the original array
- the copy owns the data and any changes made to the copy will not affect the original and vise versa
- the view does not own the data and any changes made to the view will affect the original array and any changes made to the original array will affect the view

Base

- NumPy array has the attribute base that returns None if the array owns the data, otherwise the base attribute refers to the original object

In [82]:
```
ac=np.array([12,23,56,45,15])
x=ac.copy()  # here x owns the data hence the base attribute will return None
y=ac.view()

print(x.base)
print(y.base)
```

```
None
[12 23 56 45 15]
```

In [63]:
```
# reshaping
import numpy as np
ar=np.array([1,2,3,4,5,6,7,8,9])
n=ar.reshape(3,3)
print(n)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

In [64]:
```
# iterating array
for i in ar:
    print(i)
```

```
1
2
3
4
5
6
7
8
9
```

In [65]:
```
# joining 2 array
''' we join the two arrays by concatenate() function along with the axi. if axis is

a=np.array([1,2,3])
b=np.array([4,5,6])
con = np.concatenate((a,b))
print(con)
```

```
[1 2 3 4 5 6]
```

In [71]:
```
aa=np.array([[1,2,3],[4,5,6]])
ab=np.array([[11,12,13],[14,15,16]])
con1=np.concatenate((aa,ab),axis=1)
print(con1)
```

```
[[ 1  2  3 11 12 13]
 [ 4  5  6 14 15 16]]
```

In [ ]:
```
# joining array using stack function
```

In [74]:
```
a12=np.array([[1,2,3],[4,5,6]])
a13=np.array([[11,12,13],[14,15,16]])
con2=np.stack((a12,a13),axis=1)
print(con2)
```

```
[[[ 1  2  3]
  [11 12 13]]

 [[ 4  5  6]
  [14 15 16]]]
```

In [75]:
```
# hstack - to stack along rows
a12=np.array([[1,2,3],[4,5,6]])
a13=np.array([[11,12,13],[14,15,16]])
con3=np.hstack((a12,a13))
print(con3)
```

```
[[ 1  2  3 11 12 13]
 [ 4  5  6 14 15 16]]
```

In [76]:
```
a=np.array([1,2,3])
b=np.array([4,5,6])
con4=np.hstack((a,b))
print(con4)
```

```
[1 2 3 4 5 6]
```

In [ ]:
```
# vstack -- to stack along columns
```

In [77]:
```
a=np.array([1,2,3])
b=np.array([4,5,6])
con5=np.vstack((a,b))
print(con5)
```

```
[[1 2 3]
 [4 5 6]]
```

In [78]:
```
# dstack()
# to stack along height which is the same as depth
a=np.array([1,2,3])
b=np.array([4,5,6])
con6=np.dstack((a,b))
print(con6)
```

```
[[[1 4]
  [2 5]
  [3 6]]]
```

In [80]:
```
con7=np.vstack(a)
print(con7)
```

```
[[1]
 [2]
 [3]]
```

In [ ]: