NumPy : Numerical Python

- used for creating **multidimensional array**
- **50 times more faster** than python lists
- written in c/c++/python
- used for scientific calculation and machine learning.

In [9]:
```
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

ndarray objects

In [39]:
```
np.array((2,3,4,5))
```

Out[39]:
```
array([2, 3, 4, 5])
```

In [40]:
```
arr = np.array([2,3,4,5])
print(arr)
print(type(arr))
```

```
[2 3 4 5]
<class 'numpy.ndarray'>
```

0-d array / Scalars

In [41]:
```
arr = np.array(23)
arr
```

Out[41]:
```
array(23)
```

ndim : can be used to check the number of dimensions

In [42]:
```
arr.ndim    # number of dimension
```

Out[42]:
```
0
```

1-d array

In [43]:
```
arr = np.array([2,3,4])
print(arr)
print("Number of dimensions : ", arr.ndim)
```

```
[2 3 4]
Number of dimensions :  1
```

2-d array

In [44]:
```
arr = np.array([[2,3,4],[5,6,7]])
print(arr)
print("Number of dimensions : ", arr.ndim)
```

```
[[2 3 4]
 [5 6 7]]
Number of dimensions :  2
```

3-d array

In [45]:
```
arr = np.array([[[2,3,4],[5,6,7]],[[2,3,4],[5,6,7]]])
print(arr)
print("Number of dimensions : ", arr.ndim)
```

```
[[[2 3 4]
  [5 6 7]]

 [[2 3 4]
  [5 6 7]]]
Number of dimensions :  3
```

ndmin : change the dimensions

In [46]:
```
np.array([2,3,4,5], ndmin = 2)
```

Out[46]:
```
array([[2, 3, 4, 5]])
```

In [47]:
```
np.array([2,3,4,5], ndmin = 4)
```

Out[47]:
```
array([[[[2, 3, 4, 5]]]])
```

dtype: change the data type of array

In [51]:
```
np.array([2,3,4.2,5], dtype = np.int)   # integer type
```

Out[51]:
```
array([2, 3, 4, 5])
```

In [52]:
```
1 np.array([2,3,4,5], dtype = np.float)   # float type
```

Out[52]:
```
array([2., 3., 4., 5.])
```

In [50]:
```
1 np.array([2,3,4,5], dtype = np.complex)   # complex type
```

Out[50]:
```
array([2.+0.j, 3.+0.j, 4.+0.j, 5.+0.j])
```

In [35]:
```
1 np.array([2,3,0,5], dtype = np.bool)   # bool type
```

Out[35]:
```
array([ True,  True, False,  True])
```

In [37]:
```
1 np.array([2,3,4,5], dtype = 'U')   # string type
```

Out[37]:
```
array(['2', '3', '4', '5'], dtype='<U1')
```

In [58]:
```
1 np.array("2022-07-04", dtype = np.datetime64)   # datetime
```

Out[58]:
```
array('2022-07-04', dtype='datetime64[D]')
```

In [67]:
```
1 arr1 = np.array([3,4],dtype=np.int)
2
3 arr1.dtype
```

Out[67]:
```
dtype('int32')
```

In [68]:
```
1 arr2 = np.array([3,4],dtype=np.int8)
2
3 arr2.dtype
```

Out[68]:
```
dtype('int8')
```

In [69]:
```
1 import sys
2 sys.getsizeof(arr1)
```

Out[69]:
```
104
```

In [70]:
```
1 sys.getsizeof(arr2)
```

Out[70]:
```
98
```

arange()

In [72]:
```
1 np.arange(1,10,1)
```

Out[72]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

reshape()

In [74]:
```
1 np.arange(1,10,1).reshape(3,3)
```

Out[74]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

In [76]:
```
1 np.arange(1,10,1).reshape(9,1)    # 9 rows & 1 column
```

Out[76]:
```
array([[1],
       [2],
       [3],
       [4],
       [5],
       [6],
       [7],
       [8],
       [9]])
```

MAking 3-d array using arange & reshape

In [81]:
```
1 arr = np.arange(1,37).reshape(3,3,4)
2 arr
```

Out[81]:
```
array([[[ 1,  2,  3,  4],
        [ 5,  6,  7,  8],
        [ 9, 10, 11, 12]],

       [[13, 14, 15, 16],
        [17, 18, 19, 20],
        [21, 22, 23, 24]],

       [[25, 26, 27, 28],
        [29, 30, 31, 32],
        [33, 34, 35, 36]]])
```

flattening of array

In [86]:
```
1 arr.reshape(-1)
```

Out[86]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
       35, 36])
```

In [87]:
```
1 np.arange(1,37).reshape(-1,3)
```

Out[87]:
```
array([[ 1,  2,  3],
       [ 4,  5,  6],
       [ 7,  8,  9],
       [10, 11, 12],
       [13, 14, 15],
       [16, 17, 18],
       [19, 20, 21],
       [22, 23, 24],
       [25, 26, 27],
       [28, 29, 30],
       [31, 32, 33],
       [34, 35, 36]])
```

In [88]:
```
1 np.arange(1,37).reshape(3,-1)
```

Out[88]:
```
array([[ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12],
       [13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24],
       [25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36]])
```

linspace()

- is used to generate sequence of numbers which **automatically takes the step-size** required for generating the numbers.

In [12]:
```
1 np.arange(10,13,1)
```

Out[12]:
```
array([10, 11, 12])
```

In [14]:
```
1 np.arange(10,13,0.5)
```

Out[14]:
```
array([10. , 10.5, 11. , 11.5, 12. , 12.5])
```

In [18]:
```
1 np.linspace(10,13,10)
```

Out[18]:
```
array([10.        , 10.33333333, 10.66666667, 11.        , 11.33333333,
       11.66666667, 12.        , 12.33333333, 12.66666667, 13.        ])
```

In [23]:
```
1 np.linspace(start = 10,
2             stop = 13,
3             num = 10,
4             endpoint = True,
5             retstep = True)
```

Out[23]:
```
(array([10.        , 10.33333333, 10.66666667, 11.        , 11.33333333,
        11.66666667, 12.        , 12.33333333, 12.66666667, 13.        ]),
 0.3333333333333333)
```

Indexing

In [24]:
```
arr = np.array(range(1,6))
arr
```

Out[24]:
```
array([1, 2, 3, 4, 5])
```

In [25]:
```
arr[3]
```

Out[25]:
```
4
```

In [26]:
```
arr = np.array(range(1,17)).reshape(4,4)
arr
```

Out[26]:
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12],
       [13, 14, 15, 16]])
```

In [28]:
```
arr[1][1]
```

Out[28]:
```
6
```

In [29]:
```
arr[1,1]
```

Out[29]:
```
6
```

In [33]:
```
arr[2,[0,2]]
```

Out[33]:
```
array([ 9, 11])
```

Slicing

In [34]:
```
arr
```

Out[34]:
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12],
       [13, 14, 15, 16]])
```

In [35]:
```
arr[2,0::2]    # slicing on second row
```

Out[35]:
```
array([ 9, 11])
```

In [39]:
```
arr[:,1:3]
```

Out[39]:
```
array([[ 2,  3],
       [ 6,  7],
       [10, 11],
       [14, 15]])
```

In [41]:
```
arr[:,[1,2]]
```

Out[41]:
```
array([[ 2,  3],
       [ 6,  7],
       [10, 11],
       [14, 15]])
```

In [42]:
```
arr = np.array(range(1,65)).reshape(8,8)
arr
```

Out[42]:
```
array([[ 1,  2,  3,  4,  5,  6,  7,  8],
       [ 9, 10, 11, 12, 13, 14, 15, 16],
       [17, 18, 19, 20, 21, 22, 23, 24],
       [25, 26, 27, 28, 29, 30, 31, 32],
       [33, 34, 35, 36, 37, 38, 39, 40],
       [41, 42, 43, 44, 45, 46, 47, 48],
       [49, 50, 51, 52, 53, 54, 55, 56],
       [57, 58, 59, 60, 61, 62, 63, 64]])
```

In [44]:
```
arr[1::2,0::3]
```

Out[44]:
```
array([[ 9, 12, 15],
       [25, 28, 31],
       [41, 44, 47],
       [57, 60, 63]])
```

In [45]:
```
arr = np.array(range(1,65)).reshape(4,4,4)
arr
```

Out[45]:
```
array([[[ 1,  2,  3,  4],
        [ 5,  6,  7,  8],
        [ 9, 10, 11, 12],
        [13, 14, 15, 16]],

       [[17, 18, 19, 20],
        [21, 22, 23, 24],
        [25, 26, 27, 28],
        [29, 30, 31, 32]],

       [[33, 34, 35, 36],
        [37, 38, 39, 40],
        [41, 42, 43, 44],
        [45, 46, 47, 48]],

       [[49, 50, 51, 52],
        [53, 54, 55, 56],
        [57, 58, 59, 60],
        [61, 62, 63, 64]]])
```

In [48]:
```
arr[:,[1,3],:]
```

Out[48]:
```
array([[[ 5,  6,  7,  8],
        [13, 14, 15, 16]],

       [[21, 22, 23, 24],
        [29, 30, 31, 32]],

       [[37, 38, 39, 40],
        [45, 46, 47, 48]],

       [[53, 54, 55, 56],
        [61, 62, 63, 64]]])
```

Updating the array

In [50]:
```
arr = np.array(range(1,65)).reshape(8,8)
arr[0] = 10   # updating all elemnts at row 0 to 10
arr
```

Out[50]:
```
array([[10, 10, 10, 10, 10, 10, 10, 10],
       [ 9, 10, 11, 12, 13, 14, 15, 16],
       [17, 18, 19, 20, 21, 22, 23, 24],
       [25, 26, 27, 28, 29, 30, 31, 32],
       [33, 34, 35, 36, 37, 38, 39, 40],
       [41, 42, 43, 44, 45, 46, 47, 48],
       [49, 50, 51, 52, 53, 54, 55, 56],
       [57, 58, 59, 60, 61, 62, 63, 64]])
```

In [54]:
```
arr[1::2,0::3] = 1000
```

In [55]:
```
arr
```

Out[55]:
```
array([[  10,   10,   10,   10,   10,   10,   10,   10],
       [1000,   10,   11, 1000,   13,   14, 1000,   16],
       [  17,   18,   19,   20,   21,   22,   23,   24],
       [1000,   26,   27, 1000,   29,   30, 1000,   32],
       [  33,   34,   35,   36,   37,   38,   39,   40],
       [1000,   42,   43, 1000,   45,   46, 1000,   48],
       [  49,   50,   51,   52,   53,   54,   55,   56],
       [1000,   58,   59, 1000,   61,   62, 1000,   64]])
```

ones()

In [61]:
```
1 np.ones((3,4,2), dtype = 'i')
```

Out[61]:
```
array([[[1, 1],
        [1, 1],
        [1, 1],
        [1, 1]],

       [[1, 1],
        [1, 1],
        [1, 1],
        [1, 1]],

       [[1, 1],
        [1, 1],
        [1, 1],
        [1, 1]]], dtype=int32)
```

zeros()

In [60]:
```
1 np.zeros((3,4), dtype = 'i')
```

Out[60]:
```
array([[0, 0, 0, 0],
       [0, 0, 0, 0],
       [0, 0, 0, 0]], dtype=int32)
```

eye()

In [65]:
```
1 np.eye(5, dtype = 'i')
```

Out[65]:
```
array([[1, 0, 0, 0, 0],
       [0, 1, 0, 0, 0],
       [0, 0, 1, 0, 0],
       [0, 0, 0, 1, 0],
       [0, 0, 0, 0, 1]], dtype=int32)
```

In [69]:
```
1 np.eye(5,4, dtype = 'i')
```

Out[69]:
```
array([[1, 0, 0, 0],
       [0, 1, 0, 0],
       [0, 0, 1, 0],
       [0, 0, 0, 1],
       [0, 0, 0, 0]], dtype=int32)
```

full()

In [71]:
```
1 np.full((3,3),10)
```

Out[71]:
```
array([[10, 10, 10],
       [10, 10, 10],
       [10, 10, 10]])
```

Random Module

rand()

In [73]:
```
1 np.random.rand()   # any random float number between 0 & 1
```

Out[73]:
```
0.13299069104331462
```

In [85]:
```
1 np.random.rand(4)
```

Out[85]:
```
array([0.73950882, 0.38324661, 0.1988676 , 0.72195547])
```

In [87]:
```
1 np.random.rand(3,3)
```

Out[87]:
```
array([[0.55526914, 0.27944708, 0.7696632 ],
       [0.11131064, 0.24804922, 0.33814519],
       [0.23327455, 0.17701541, 0.45788737]])
```

random()

In [93]:
```
1 np.random.random(2)   # 10 random numbers between 0 & 1
```

Out[93]:
```
array([0.90312055, 0.15098425])
```

In [92]:
```
1 np.random.random((2,2))   # 10 random numbers between 0 & 1
```

Out[92]:
```
array([[0.7079987 , 0.11278811],
       [0.08558949, 0.09067525]])
```

randint()

In [99]:
```
1 np.random.randint(2)   # 10 random numbers between 0 & 1
```

Out[99]:
```
0
```

In [81]:
```
1 np.random.randint(5,10)   # any random number between 5(inclusive) & 10(excl
```

Out[81]:
```
6
```

In [84]:
```
1 np.random.randint(5,10,6)  # 6 random numbers between 5 & 10
```

Out[84]:
```
array([8, 6, 8, 8, 8, 9])
```

In [100]:
```
np.random.randint(5,10,(2,2))
```

Out[100]:
```
array([[5, 5],
       [5, 8]])
```

choice()

In [103]:
```
np.random.choice([12,20,45,60])
```

Out[103]:
```
60
```

we can also define the probability of values being selected

In [119]:
```
np.random.choice([12,20,45,60], p = [0.2,0.1,0.2,0.5], size = 10)
```

Out[119]:
```
array([20, 12, 45, 60, 12, 20, 60, 60, 60, 60])
```

In [120]:
```
np.random.choice([12,20,45,60], p = [0.2,0.1,0.2,0.5], size = (2,2,2))
```

Out[120]:
```
array([[[60, 12],
        [60, 45]],

       [[12, 12],
        [60, 45]]])
```

In [ ]:
```
    *
   * *
  * * *
```

In [126]:
```
# trick
n = 3
for row in range(1,5):
    print(" "*n, "* "*row,sep="")
    n-=1
```

```
   *
  * *
 * * *
* * * *
```

In [132]:
```
for row in range(1,5):
    for s in range(4,row,-1):
        print('-', end = "")
    for j in range(0,row):
        print("*",end=" ")
    print()
```

```
---*
--* *
-* * *
* * * *
```

In [142]:
```
for x in range(1,2):
    print(" "*8,"\/")
for i in range(1,3):
    print(" "*8,"||")
for j in range(1,5):
    if j == 2:
        print("***Happy Birthday***")
        continue
    print("*"*20)
```

```
        \/
        ||
        ||
********************
***Happy Birthday***
********************
********************
```

trunc()

In [143]:
```
arr = np.array([2.567,3.21,-5.78])
arr
```

Out[143]:
```
array([ 2.567,  3.21 , -5.78 ])
```

In [145]:
```
np.trunc(arr)    # removes all decimal part
```

Out[145]:
```
array([ 2.,  3., -5.])
```

floor()

In [147]:
```
arr = np.array([2.567,3.21,-5.78])
np.floor(arr)     # returns next smallest number
```

Out[147]:
```
array([ 2.,  3., -6.])
```

ceil()

In [148]:
```
arr = np.array([2.567,3.21,-5.78])
np.ceil(arr)       # returns next largest number
```

Out[148]:
```
array([ 3.,  4., -5.])
```

around()

In [149]:
```
arr = np.array([2.567,3.21,-5.78])
np.around(arr)    # round off to 0 decimal places
```

Out[149]:
```
array([ 3.,  3., -6.])
```

In [151]:
```
arr = np.array([2.567,3.21,-5.78999])
np.around(arr,2)    # round off to 2 decimnal places
```

Out[151]:
```
array([ 2.57,  3.21, -5.79])
```

astype()

In [152]:
```
arr = np.array([3.2,4,5.6,9.99])
arr.dtype
```

Out[152]:
```
dtype('float64')
```

In [154]:
```
# change the data type using astype function

new_arr = arr.astype('i')
new_arr
```

Out[154]:
```
array([3, 4, 5, 9], dtype=int32)
```

In [155]:
```
new_arr.dtype
```

Out[155]:
```
dtype('int32')
```

In [157]:
```
arr1 = np.array([1,2,3,4])
arr2 = np.array([5,6,7,8])
```

add(), subtract(), multiply(),divide(),divmod()

In [158]:
```
np.add(arr1,arr2)
```

Out[158]:
```
array([ 6,  8, 10, 12])
```

In [162]:
```
np.subtract(arr1,arr2)
```

Out[162]:
```
array([-4, -4, -4, -4])
```

In [163]:
```
np.multiply(arr1,arr2)
```

Out[163]:
```
array([ 5, 12, 21, 32])
```

In [164]:
```
np.divide(arr1,arr2)
```

Out[164]:
```
array([0.2       , 0.33333333, 0.42857143, 0.5       ])
```

In [165]:
```
1 np.divmod(arr1,arr2)
```

Out[165]:
```
(array([0, 0, 0, 0], dtype=int32), array([1, 2, 3, 4], dtype=int32))
```

sum()

In [167]:
```
1 arr1 = np.array([1,2,3,4])
2 arr2 = np.array([5,6,7,8])
3
4 np.sum([arr1,arr2])
```

Out[167]:
```
36
```

In [168]:
```
1 np.sum([arr1,arr2], axis = 0)
```

Out[168]:
```
array([ 6,  8, 10, 12])
```

In [169]:
```
1 np.sum([arr1,arr2], axis = 1)
```

Out[169]:
```
array([10, 26])
```

cumsum()

- cumulative sum

In [171]:
```
1 arr1 = np.array([1,2,3,4])
2 arr2 = np.array([5,6,7,8])
3
4 np.cumsum([arr1,arr2])
```

Out[171]:
```
array([ 1,  3,  6, 10, 15, 21, 28, 36], dtype=int32)
```

In [172]:
```
1 np.cumsum([arr1,arr2], axis = 0)
```

Out[172]:
```
array([[ 1,  2,  3,  4],
       [ 6,  8, 10, 12]], dtype=int32)
```

In [173]:
```
1 np.cumsum([arr1,arr2], axis = 1)
```

Out[173]:
```
array([[ 1,  3,  6, 10],
       [ 5, 11, 18, 26]], dtype=int32)
```

diff(), prod(), cumprod()

In [174]:
```
arr1 = np.array([1,2,3,4])
arr2 = np.array([5,6,7,8])

np.diff([arr1,arr2])
```

Out[174]:
```
array([[1, 1, 1],
       [1, 1, 1]])
```

In [181]:
```
np.diff([arr1,arr2],axis = 1)
```

Out[181]:
```
array([[1, 1, 1],
       [1, 1, 1]])
```

In [175]:
```
arr1 = np.array([1,2,3,4])
arr2 = np.array([5,6,7,8])

np.prod([arr1,arr2])
```

Out[175]:
```
40320
```

In [178]:
```
np.cumprod([arr1,arr2])
```

Out[178]:
```
array([    1,     2,     6,    24,   120,   720,  5040, 40320],
      dtype=int32)
```

absolute()

In [182]:
```
np.absolute(np.array([-3,4,-6]))
```

Out[182]:
```
array([3, 4, 6])
```

insert()

In [185]:
```
arr=np.array([2,3,5,6])
arr
```

Out[185]:
```
array([2, 3, 5, 6])
```

In [191]:
```
np.insert(arr,2,4)    # inserting 4 at index 2 in arr
```

Out[191]:
```
array([2, 3, 4, 5, 6])
```

np.nan

- nan mean **Not a Number**

In [196]:
```
arr = np.array([3,4,np.nan,5,7])
arr
```

Out[196]:
```
array([ 3.,  4., nan,  5.,  7.])
```

isnan()

- returns True where null value is present, False otherwise

In [198]:
```
np.isnan(arr)
```

Out[198]:
```
array([False, False,  True, False, False])
```

In [201]:
```
arr = np.random.randint(20,50,size = (3,3))
arr
```

Out[201]:
```
array([[22, 26, 33],
       [49, 20, 38],
       [21, 20, 48]])
```

where()

- is used to match a condition in an array
- it returns the indexes where the condition is matched

In [205]:
```
arr
```

Out[205]:
```
array([[22, 26, 33],
       [49, 20, 38],
       [21, 20, 48]])
```

In [204]:
```
np.where(arr>25)
```

Out[204]:
```
(array([0, 0, 1, 1, 2], dtype=int64), array([1, 2, 0, 2, 2], dtype=int64))
```

In [206]:
```
arr[np.where(arr>25)]  # finding numbers greater than 25
```

Out[206]:
```
array([26, 33, 49, 38, 48])
```

In [207]:
```
np.where(arr%2==0)     # finding even numbers in arr
```

Out[207]:
```
(array([0, 0, 1, 1, 2, 2], dtype=int64),
 array([0, 1, 1, 2, 1, 2], dtype=int64))
```

In [208]:
```
arr[np.where(arr%2==0)]
```

Out[208]:
```
array([22, 26, 20, 38, 20, 48])
```

In [209]:
```
arr[[0, 0, 1, 1, 2, 2],[0, 1, 1, 2, 1, 2]]
```

Out[209]:
```
array([22, 26, 20, 38, 20, 48])
```

Boolean Indexing

In [210]:
```
arr = np.array([20,40,50,11,33])
arr
```

Out[210]:
```
array([20, 40, 50, 11, 33])
```

In [211]:
```
arr>35
```

Out[211]:
```
array([False,  True,  True, False, False])
```

In [212]:
```
arr[[False,  True,  True, False, False]]    # boolean indexing
```

Out[212]:
```
array([40, 50])
```

In [213]:
```
arr[arr>35]
```

Out[213]:
```
array([40, 50])
```

In [216]:
```
arr = np.random.randint(10,100,100).reshape(10,10)
arr
```

Out[216]:
```
array([[89, 45, 35, 26, 18, 56, 27, 55, 91, 79],
       [50, 54, 84, 25, 99, 91, 75, 91, 95, 55],
       [54, 93, 38, 52, 21, 36, 69, 74, 67, 60],
       [42, 43, 29, 63, 67, 37, 74, 69, 50, 77],
       [84, 48, 40, 72, 42, 68, 16, 35, 56, 69],
       [75, 16, 23, 51, 87, 97, 52, 29, 24, 10],
       [67, 99, 81, 30, 57, 83, 29, 10, 12, 26],
       [22, 28, 27, 41, 28, 45, 51, 71, 80, 74],
       [42, 95, 86, 80, 24, 67, 81, 55, 43, 58],
       [99, 80, 36, 45, 27, 96, 95, 87, 47, 63]])
```

In [218]:
```
arr[arr%2==0]     # boolean indexing
```

Out[218]:
```
array([26, 18, 56, 50, 54, 84, 54, 38, 52, 36, 74, 60, 42, 74, 50, 84, 48,
       40, 72, 42, 68, 16, 56, 16, 52, 24, 10, 30, 10, 12, 26, 22, 28, 28,
       80, 74, 42, 86, 80, 24, 58, 80, 36, 96])
```

In [219]:
```
arr[np.where(arr%2==0)]    # getting elements through indexes
```

Out[219]:
```
array([26, 18, 56, 50, 54, 84, 54, 38, 52, 36, 74, 60, 42, 74, 50, 84, 48,
       40, 72, 42, 68, 16, 56, 16, 52, 24, 10, 30, 10, 12, 26, 22, 28, 28,
       80, 74, 42, 86, 80, 24, 58, 80, 36, 96])
```

In [223]:
```
arr[~arr%2==0]   # getting odd numbers through boolean indexing
```

Out[223]:
```
array([89, 45, 35, 27, 55, 91, 79, 25, 99, 91, 75, 91, 95, 55, 93, 21, 69,
       67, 43, 29, 63, 67, 37, 69, 77, 35, 69, 75, 23, 51, 87, 97, 29, 67,
       99, 81, 57, 83, 29, 27, 41, 45, 51, 71, 95, 67, 81, 55, 43, 99, 45,
       27, 95, 87, 47, 63])
```

np.sort()

- sorts the array into ascending order by default

In [235]:
```
np.sort(arr)  # see the ouput by printing the arr
```

Out[235]:
```
array([[18, 26, 27, 35, 45, 55, 56, 79, 89, 91],
       [25, 50, 54, 55, 75, 84, 91, 91, 95, 99],
       [21, 36, 38, 52, 54, 60, 67, 69, 74, 93],
       [29, 37, 42, 43, 50, 63, 67, 69, 74, 77],
       [16, 35, 40, 42, 48, 56, 68, 69, 72, 84],
       [10, 16, 23, 24, 29, 51, 52, 75, 87, 97],
       [10, 12, 26, 29, 30, 57, 67, 81, 83, 99],
       [22, 27, 28, 28, 41, 45, 51, 71, 74, 80],
       [24, 42, 43, 55, 58, 67, 80, 81, 86, 95],
       [27, 36, 45, 47, 63, 80, 87, 95, 96, 99]])
```

descending order using sort()

In [238]:
```
-np.sort(-arr)    # sort the array into descending order
```

Out[238]:
```
array([[91, 89, 79, 56, 55, 45, 35, 27, 26, 18],
       [99, 95, 91, 91, 84, 75, 55, 54, 50, 25],
       [93, 74, 69, 67, 60, 54, 52, 38, 36, 21],
       [77, 74, 69, 67, 63, 50, 43, 42, 37, 29],
       [84, 72, 69, 68, 56, 48, 42, 40, 35, 16],
       [97, 87, 75, 52, 51, 29, 24, 23, 16, 10],
       [99, 83, 81, 67, 57, 30, 29, 26, 12, 10],
       [80, 74, 71, 51, 45, 41, 28, 28, 27, 22],
       [95, 86, 81, 80, 67, 58, 55, 43, 42, 24],
       [99, 96, 95, 87, 80, 63, 47, 45, 36, 27]])
```

np.flip()

In [239]:
```
arr = np.flip( np.sort(arr) )    # do ascending and then reverse the array
arr                              # array flipped using flipped function
```

Out[239]:
```
array([[99, 96, 95, 87, 80, 63, 47, 45, 36, 27],
       [95, 86, 81, 80, 67, 58, 55, 43, 42, 24],
       [80, 74, 71, 51, 45, 41, 28, 28, 27, 22],
       [99, 83, 81, 67, 57, 30, 29, 26, 12, 10],
       [97, 87, 75, 52, 51, 29, 24, 23, 16, 10],
       [84, 72, 69, 68, 56, 48, 42, 40, 35, 16],
       [77, 74, 69, 67, 63, 50, 43, 42, 37, 29],
       [93, 74, 69, 67, 60, 54, 52, 38, 36, 21],
       [99, 95, 91, 91, 84, 75, 55, 54, 50, 25],
       [91, 89, 79, 56, 55, 45, 35, 27, 26, 18]])
```

Assignment :

Take a 5x5 zeros array as input:

```
00000
00000
00000
00000
00000
```

output:1

```
11111
10001
10001
10001
11111
```

output:2

```
10001
01010
00100
01010
10001
```

output:3

```
11111
00010
00100
01000
11111
```

In [ ]:
```
1
```