Numpy

- It stands for Numerical Python.
- It comes under Open-Source category.
- Numpy is homogenous in nature.(It can be of only single data type.)

```
#Installing Numpy
!pip install numpy
```

```
Requirement already satisfied: numpy in c:\users\ayushsaxena\appdata\
local\anaconda3\lib\site-packages (1.26.4)
```

```
import numpy as np
```

```
np.array([1,2,3,4])
```

```
array([1, 2, 3, 4])
```

```
np.array([1,2.4,3,5,6]) #It will convert all int values to float.
# Question is why the complete array is not converted into int?
# Ans. Beacuse of data loss if it convert to int intstead of float it
will lost the data from 2.4 to 2.
```

```
array([1. , 2.4, 3. , 5. , 6. ])
```

```
np.array([True,False,10])
```

```
array([ 1,  0, 10])
```

```
np.array([1.0,2.0,3,5,6,True])
```

```
array([1., 2., 3., 5., 6., 1.])
```

```
arr1=np.array([1,2,3.0,5.0,True])
```

```
#to get the data type of arr1
arr1.dtype
```

```
dtype('float64')
```

```
arr2=np.array([1,2,3,4,5,6,7])
arr2.dtype
```

```
dtype('int32')
```

```
arr3=np.array([1,2,3,4,'apple'])
arr3
```

```
array(['1', '2', '3', '4', 'apple'], dtype='<U11')
```

```
arr3.dtype
```

```
dtype('<U11')
```

```
arr4=np.array([True,True,False])
arr4
```

```
array([ True,  True, False])
```

```
arr4.dtype
```

```
dtype('bool')
```

```
arr2
```

```
array([1, 2, 3, 4, 5, 6, 7])
```

```
#gettine size of array will give u number of element
arr2.size
```

```
7
```

```
arr4
```

```
array([ True,  True, False])
```

```
print(arr4.dtype)
print(arr4.size)
```

```
bool
3
```

```
#getting dimension
arr5=np.array([1,2,3,4,5,6])
print(arr5.ndim)
```

```
1
```

```
arr6=np.array([[1,2],[3,4]])
arr6
```

```
array([[1, 2],
       [3, 4]])
```

```
arr7=np.array([[[1,2,3],[4,5,6]],[[7,8,9],[10,11,12]]])
```

```
arr7
```

```
array([[[ 1,  2,  3],
        [ 4,  5,  6]],

       [[ 7,  8,  9],
        [10, 11, 12]]])
```

```
arr7.ndim
```

```
3
```

```
arr6.ndim
```

```
2
```

```
arr8=np.array([[1,2,3,4,5,6],[7,8,9,10,11,12]])
```

```
arr8.ndim
```

```
2
```

```
arr9=np.array([[[1],[2],[3]],[[4],[5],[6]]])
```

```
arr9
```

```
array([[[1],
        [2],
        [3]],

       [[4],
        [5],
        [6]]])
```

```
print(arr9.size)
print(arr9.ndim)
print(arr9.dtype)
```

```
6
3
int32
```

```
arr7
```

```
array([[[ 1,  2,  3],
        [ 4,  5,  6]],

       [[ 7,  8,  9],
        [10, 11, 12]]])
```

```
arr7.size
```

```
12
```

```
arr7.shape # explain how elemts are organised
```

```
(2, 2, 3)
```

```
arr6=np.array([[1,2],[3,4]])
arr6
```

```
array([[1, 2],
       [3, 4]])
```

```
arr6.shape
```

```
(2, 2)
```

```
arr9=np.array([[[1],[2],[3]],[[4],[5],[6]]])
arr9.shape
```

```
(2, 3, 1)
```

```
arr5
```

```
array([1, 2, 3, 4, 5, 6])
```

```
arr5.shape # it value tell you values in 1d array
```

```
(6,)
```

```
arr10=np.array([[[1,2,3],[4,5,6],[7,8,9]],[[10,11,12],[13,14,15],
[16,17,18]]])
arr10
```

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

       [[10, 11, 12],
        [13, 14, 15],
        [16, 17, 18]]])
```

```
arr10.shape
```

```
(2, 3, 3)
```

```
arr10.size
```

```
18
```

```
arr10.dtype
```

```
dtype('int32')
```

```
arr10.ndim
```

```
3
```

```
arr11=np.array([[1,2,3,4],[5,6,7]])
```

```
---------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
Cell In[57], line 1
----> 1 arr11=np.array([[1,2,3,4],[5,6,7]])
```

```
ValueError: setting an array element with a sequence. The requested
array has an inhomogeneous shape after 1 dimensions. The detected
shape was (2,) + inhomogeneous part.
```

Arange

```
np.arange(5)
```

```
array([0, 1, 2, 3, 4])
```

```
np.arange(10)
```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
np.arange(100)
```

```
array([ 0,  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, 65, 66,
67,
       68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83,
84,
       85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99])
```

```
np.arange(100,0,-1)
```

```
array([100,  99,  98,  97,  96,  95,  94,  93,  92,  91,  90,  89,
88,
        87,  86,  85,  84,  83,  82,  81,  80,  79,  78,  77,  76,
75,
        74,  73,  72,  71,  70,  69,  68,  67,  66,  65,  64,  63,
62,
        61,  60,  59,  58,  57,  56,  55,  54,  53,  52,  51,  50,
49,
        48,  47,  46,  45,  44,  43,  42,  41,  40,  39,  38,  37,
36,
        35,  34,  33,  32,  31,  30,  29,  28,  27,  26,  25,  24,
23,
        22,  21,  20,  19,  18,  17,  16,  15,  14,  13,  12,  11,
10,
         9,   8,   7,   6,   5,   4,   3,   2,   1])
```

```
np.arange(10,20)
```

```
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19])
```

```
arr12=np.arange(1000)
print(arr12.size)
print(arr12.ndim)
print(arr12.shape)
```

```
1000
1
(1000,)
```

Reshape

```
arr13=np.arange(100)
arr13.reshape(10,10)
```

```
array([[ 0,  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, 65, 66, 67, 68, 69],
       [70, 71, 72, 73, 74, 75, 76, 77, 78, 79],
       [80, 81, 82, 83, 84, 85, 86, 87, 88, 89],
       [90, 91, 92, 93, 94, 95, 96, 97, 98, 99]])
```

Question-: Create numpy array with 50 element in the shape of 5,5,2

```
arr14=np.arange(50)
arr14.reshape(5,5,2)
```

```
array([[[ 0,  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]]])

arr15=np.arange(20).reshape(2,5,2)
arr15

array([[[ 0,  1],
        [ 2,  3],
        [ 4,  5],
        [ 6,  7],
        [ 8,  9]],

       [[10, 11],
        [12, 13],
        [14, 15],
        [16, 17],
        [18, 19]]])

arr16=np.arange(20).reshape(5,2,2)
arr16

array([[[ 0,  1],
        [ 2,  3]],

       [[ 4,  5],
        [ 6,  7]],

       [[ 8,  9],
        [10, 11]],

       [[12, 13],
        [14, 15]],

       [[16, 17],
        [18, 19]]])
```

Zeros

```python
np.zeros(100) # it will generate hundred zero's
```

```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

```python
np.zeros(100).reshape(10,2,5)
```

```
array([[[0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0.]],

       [[0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0.]],

       [[0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0.]],

       [[0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0.]],

       [[0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0.]],

       [[0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0.]],

       [[0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0.]],

       [[0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0.]],

       [[0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0.]],

       [[0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0.]]])
```

```python
np.zeros(100).reshape(20,10)
```

```
---------------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
Cell In[72], line 1
----> 1 np.zeros(100).reshape(20,10)

ValueError: cannot reshape array of size 100 into shape (20,10)
```

Ravel

- it will convert the array into 1 dimension

```
arr17=np.arange(200).reshape(20,10)

arr17

array([[  0,   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,  65,  66,  67,  68,  69],
       [ 70,  71,  72,  73,  74,  75,  76,  77,  78,  79],
       [ 80,  81,  82,  83,  84,  85,  86,  87,  88,  89],
       [ 90,  91,  92,  93,  94,  95,  96,  97,  98,  99],
       [100, 101, 102, 103, 104, 105, 106, 107, 108, 109],
       [110, 111, 112, 113, 114, 115, 116, 117, 118, 119],
       [120, 121, 122, 123, 124, 125, 126, 127, 128, 129],
       [130, 131, 132, 133, 134, 135, 136, 137, 138, 139],
       [140, 141, 142, 143, 144, 145, 146, 147, 148, 149],
       [150, 151, 152, 153, 154, 155, 156, 157, 158, 159],
       [160, 161, 162, 163, 164, 165, 166, 167, 168, 169],
       [170, 171, 172, 173, 174, 175, 176, 177, 178, 179],
       [180, 181, 182, 183, 184, 185, 186, 187, 188, 189],
       [190, 191, 192, 193, 194, 195, 196, 197, 198, 199]])

arr17.ravel()

array([  0,   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,
        65,  66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,
77,
        78,  79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,
90,
        91,  92,  93,  94,  95,  96,  97,  98,  99, 100, 101, 102,
103,
       104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115,
116,
       117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128,
129,
       130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141,
142,
       143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154,
155,
       156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167,
168,
       169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180,
181,
       182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193,
194,
       195, 196, 197, 198, 199])

arr18=np.zeros(50).reshape(10,5)
arr18

array([[0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.]])
```

Floor, Ceil and Round

```
arr1=np.array([1.1,2.2,3.3,4.4])
np.floor(arr1) # it bound the number to lower bound

array([1., 2., 3., 4.])

np.ceil(arr1) #it round the number to upper bound

array([2., 3., 4., 5.])

arr1
```

```
array([1.1, 2.2, 3.3, 4.4])

arr2=np.array([3.14,5.6,4.5,8.4,9.8])

arr2

array([3.14, 5.6 , 4.5 , 8.4 , 9.8 ])

np.round(arr2)

array([ 3.,  6.,  4.,  8., 10.])

arr3=np.array([2.2,3.6,4.2])
print("The Floor values is displayed as-: ",np.floor(arr3))
print("The round values is displayed as-: ",np.round(arr3))
print("The ceil values is displayed as-: ",(np.ceil(arr3)))

The Floor values is displayed as-:  [2. 3. 4.]
The round values is displayed as-:  [2. 4. 4.]
The ceil values is displayed as-:  [3. 4. 5.]
```

Stacking

- It combines two numpy array into singel one.
- It is of two types-"
-
    a. Horizontal Stack
    b. Vertical stack
- It will work on 2d Array only.

```
arr4=np.arange(10).reshape(2,5)
arr4

array([[0, 1, 2, 3, 4],
       [5, 6, 7, 8, 9]])

arr5= np.arange(20).reshape(4,5)
arr5

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

#vertical stack
np.vstack((arr4,arr5))

array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [ 0,  1,  2,  3,  4],
```

```
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19]])

arr6= np.arange(20).reshape(2,10)
arr6

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

np.hstack((arr4,arr6))

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

np.hstack((arr4,arr5))

---------------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
Cell In[103], line 1
----> 1 np.hstack((arr4,arr5))

File ~\AppData\Local\anaconda3\Lib\site-packages\numpy\core\
shape_base.py:359, in hstack(tup, dtype, casting)
    357     return _nx.concatenate(arrs, 0, dtype=dtype,
casting=casting)
    358 else:
--> 359     return _nx.concatenate(arrs, 1, dtype=dtype,
casting=casting)

ValueError: all the input array dimensions except for the
concatenation axis must match exactly, but along dimension 0, the
array at index 0 has size 2 and the array at index 1 has size 4

np.hstack((arr4,arr6)).astype('float') # converting into float data
type

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

np.hstack((arr4,arr6)).astype('bool') # converting into boolean data
type

array([[False,  True,  True,  True,  True, False,  True,  True,  True,
         True,  True,  True,  True,  True,  True],
```

```
       [ True,  True,  True,  True,  True,  True,  True,  True,  True,
         True,  True,  True,  True,  True,  True]])
```

Split

- Opposite of Vstack and Hstack
- It allows us to split numpy arry into one or more numpy array.
- It is of two type-:
-
    a. Vertical
    b. Horizontal
- All the array should be of same size while spliting.
- It will return back as a list not array.
- Indexing can be used here.
- vsplit only works on arrays of 2 or more dimensions

```
arr7= np.arange(100).reshape(10,10)
arr7
```

```
array([[ 0,  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, 65, 66, 67, 68, 69],
       [70, 71, 72, 73, 74, 75, 76, 77, 78, 79],
       [80, 81, 82, 83, 84, 85, 86, 87, 88, 89],
       [90, 91, 92, 93, 94, 95, 96, 97, 98, 99]])
```

```
res=np.vsplit(arr7, 2)
```

```
res
```

```
[array([[ 0,  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]]),
 array([[50, 51, 52, 53, 54, 55, 56, 57, 58, 59],
        [60, 61, 62, 63, 64, 65, 66, 67, 68, 69],
        [70, 71, 72, 73, 74, 75, 76, 77, 78, 79],
        [80, 81, 82, 83, 84, 85, 86, 87, 88, 89],
        [90, 91, 92, 93, 94, 95, 96, 97, 98, 99]])]
```

```
res[0]
```

```
array([[ 0,  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]])
```

```
res[1]
```

```
array([[50, 51, 52, 53, 54, 55, 56, 57, 58, 59],
       [60, 61, 62, 63, 64, 65, 66, 67, 68, 69],
       [70, 71, 72, 73, 74, 75, 76, 77, 78, 79],
       [80, 81, 82, 83, 84, 85, 86, 87, 88, 89],
       [90, 91, 92, 93, 94, 95, 96, 97, 98, 99]])
```

```
res1=np.vsplit(arr7, 10)
res1
```

```
[array([[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]]),
 array([[10, 11, 12, 13, 14, 15, 16, 17, 18, 19]]),
 array([[20, 21, 22, 23, 24, 25, 26, 27, 28, 29]]),
 array([[30, 31, 32, 33, 34, 35, 36, 37, 38, 39]]),
 array([[40, 41, 42, 43, 44, 45, 46, 47, 48, 49]]),
 array([[50, 51, 52, 53, 54, 55, 56, 57, 58, 59]]),
 array([[60, 61, 62, 63, 64, 65, 66, 67, 68, 69]]),
 array([[70, 71, 72, 73, 74, 75, 76, 77, 78, 79]]),
 array([[80, 81, 82, 83, 84, 85, 86, 87, 88, 89]]),
 array([[90, 91, 92, 93, 94, 95, 96, 97, 98, 99]])]
```

```
res2=np.hsplit(arr7,2)
```

```
res2
```

```
[array([[ 0,  1,  2,  3,  4],
        [10, 11, 12, 13, 14],
        [20, 21, 22, 23, 24],
        [30, 31, 32, 33, 34],
        [40, 41, 42, 43, 44],
        [50, 51, 52, 53, 54],
        [60, 61, 62, 63, 64],
        [70, 71, 72, 73, 74],
        [80, 81, 82, 83, 84],
        [90, 91, 92, 93, 94]]),
 array([[ 5,  6,  7,  8,  9],
        [15, 16, 17, 18, 19],
        [25, 26, 27, 28, 29],
        [35, 36, 37, 38, 39],
        [45, 46, 47, 48, 49],
        [55, 56, 57, 58, 59],
        [65, 66, 67, 68, 69],
        [75, 76, 77, 78, 79],
```

```
        [85, 86, 87, 88, 89],
        [95, 96, 97, 98, 99]])]

res3=np.hsplit(arr7,5)
res3

[array([[ 0,  1],
        [10, 11],
        [20, 21],
        [30, 31],
        [40, 41],
        [50, 51],
        [60, 61],
        [70, 71],
        [80, 81],
        [90, 91]]),
 array([[ 2,  3],
        [12, 13],
        [22, 23],
        [32, 33],
        [42, 43],
        [52, 53],
        [62, 63],
        [72, 73],
        [82, 83],
        [92, 93]]),
 array([[ 4,  5],
        [14, 15],
        [24, 25],
        [34, 35],
        [44, 45],
        [54, 55],
        [64, 65],
        [74, 75],
        [84, 85],
        [94, 95]]),
 array([[ 6,  7],
        [16, 17],
        [26, 27],
        [36, 37],
        [46, 47],
        [56, 57],
        [66, 67],
        [76, 77],
        [86, 87],
        [96, 97]]),
 array([[ 8,  9],
        [18, 19],
        [28, 29],
        [38, 39],
```

```
        [48, 49],
        [58, 59],
        [68, 69],
        [78, 79],
        [88, 89],
        [98, 99]])]
```

```
arr7
```

```
array([[ 0,  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, 65, 66, 67, 68, 69],
       [70, 71, 72, 73, 74, 75, 76, 77, 78, 79],
       [80, 81, 82, 83, 84, 85, 86, 87, 88, 89],
       [90, 91, 92, 93, 94, 95, 96, 97, 98, 99]])
```

```
arr7.T #Transpone
```

```
array([[ 0, 10, 20, 30, 40, 50, 60, 70, 80, 90],
       [ 1, 11, 21, 31, 41, 51, 61, 71, 81, 91],
       [ 2, 12, 22, 32, 42, 52, 62, 72, 82, 92],
       [ 3, 13, 23, 33, 43, 53, 63, 73, 83, 93],
       [ 4, 14, 24, 34, 44, 54, 64, 74, 84, 94],
       [ 5, 15, 25, 35, 45, 55, 65, 75, 85, 95],
       [ 6, 16, 26, 36, 46, 56, 66, 76, 86, 96],
       [ 7, 17, 27, 37, 47, 57, 67, 77, 87, 97],
       [ 8, 18, 28, 38, 48, 58, 68, 78, 88, 98],
       [ 9, 19, 29, 39, 49, 59, 69, 79, 89, 99]])
```

Indexing

```
arr = np.array([10, 20, 30, 40, 50])
print(arr[0])   # First element → 10
print(arr[2])   # Third element → 30
print(arr[-1])  # Last element → 50
```

```
10
30
50
```

```
arr2d = np.array([[1, 2, 3],
                  [4, 5, 6],
                  [7, 8, 9]])

print(arr2d[0, 0])  # Element at 1st row, 1st column → 1
```

```
print(arr2d[1, 2])  # Element at 2nd row, 3rd column → 6
print(arr2d[2, -1]) # Last element in last row → 9
```

```
1
6
9
```

```
arr3d = np.array([
    [[1, 2, 3],
     [4, 5, 6]],

    [[7, 8, 9],
     [10, 11, 12]]
])
print(arr3d[0, 0, 0])  # 1st layer, 1st row, 1st column → 1
print(arr3d[0, 1, 2])  # 1st layer, 2nd row, 3rd column → 6
print(arr3d[1, 0, 1])  # 2nd layer, 1st row, 2nd column → 8
print(arr3d[1, 1, 2])  # 2nd layer, 2nd row, 3rd column → 12
```

```
1
6
8
12
```

Array Slicing

- Slicing allows you to extract a subpart of the array
- Syntax:

1. array[start:end:step]

```
arr = np.array([10, 20, 30, 40, 50, 60])
print(arr[1:4])     # Elements from index 1 to 3 → [20, 30, 40]
print(arr[:3])      # From start to index 2 → [10, 20, 30]
print(arr[3:])      # From index 3 to end → [40, 50, 60]
print(arr[::2])     # Every 2nd element → [10, 30, 50]
```

```
[20 30 40]
[10 20 30]
[40 50 60]
[10 30 50]
```

```
arr2d = np.array([[1, 2, 3, 4],
                  [5, 6, 7, 8],
                  [9, 10, 11, 12]])

print(arr2d[0:2, 1:3])  # Rows 0-1 and Columns 1-2
# Output:
# [[2 3]
#  [6 7]]
```

```
print(arr2d[:, 2])      # All rows, column 2 → [3, 7, 11]
print(arr2d[1, :])      # Entire 2nd row → [5, 6, 7, 8]
```

```
[[2 3]
 [6 7]]
[ 3  7 11]
[5 6 7 8]
```

```
arr3d = np.array([
    [[1, 2, 3],
     [4, 5, 6]],

    [[7, 8, 9],
     [10, 11, 12]]
])


# Take both layers, first row only, all columns
print(arr3d[:, 0, :])
# Output:
# [[1 2 3]
#  [7 8 9]]

# Take second layer only, all rows, first two columns
print(arr3d[1, :, 0:2])
# Output:
# [[7 8]
#  [10 11]]
```

```
[[1 2 3]
 [7 8 9]]
[[ 7  8]
 [10 11]]
```

What is Broadcasting?

- Broadcasting allows NumPy to perform operations on arrays of different shapes — by "stretching" the smaller array to match the shape of the larger one without copying data.
- Think of it as automatic alignment of array shapes during math operations.

```
arr = np.array([1, 2, 3])
result = arr + 5

print(result)
# Output: [6 7 8]
```

```
[6 7 8]
```

Broadcasting Between 1D and 2D Arrays

```
arr2d = np.array([[1, 2, 3],
                  [4, 5, 6]])
arr1d = np.array([10, 20, 30])

result = arr2d + arr1d
print(result)
```

```
[[11 22 33]
 [14 25 36]]
```

Mathematical and Statistical Operations in NumPy

```
arr = np.array([10, 20, 30, 40, 50])
print(arr + 5)   # Add 5 to each element → [15 25 35 45 55]
print(arr - 2)   # Subtract 2 from each → [8 18 28 38 48]
print(arr * 2)   # Multiply each by 2 → [20 40 60 80 100]
print(arr / 10)  # Divide each by 10 → [1. 2. 3. 4. 5.]
```

```
[15 25 35 45 55]
[ 8 18 28 38 48]
[ 20  40  60  80 100]
[1. 2. 3. 4. 5.]
```

```
print(np.exp(arr))     # Exponential of each element
print(np.sqrt(arr))    # Square root
print(np.power(arr, 2))# arr^2
```

```
[2.20264658e+04 4.85165195e+08 1.06864746e+13 2.35385267e+17
 5.18470553e+21]
[3.16227766 4.47213595 5.47722558 6.32455532 7.07106781]
[ 100  400  900 1600 2500]
```

```
data = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9])
print(np.sum(data))     # 45
print(np.mean(data))    # 5.0
print(np.median(data))  # 5.0
print(np.std(data))     # 2.581...
print(np.var(data))     # 6.666...
print(np.min(data))     # 1
print(np.max(data))     # 9
```

```
45
5.0
5.0
2.581988897471611
6.666666666666667
1
9
```

```
data = np.array([1, 2, 3, 4])

print(np.cumsum(data))  # Cumulative sum → [1 3 6 10]
print(np.cumprod(data)) # Cumulative product → [1 2 6 24]
print(np.diff(data))    # Difference between consecutive elements → [1
1 1]
```

```
[ 1  3  6 10]
[ 1  2  6 24]
[1 1 1]
```

```
x = np.array([1, 2, 3, 4, 5])
y = np.array([5, 4, 3, 2, 1])

print(np.corrcoef(x, y))  # Correlation matrix
print(np.cov(x, y))       # Covariance matrix
```

```
[[ 1. -1.]
 [-1.  1.]]
[[ 2.5 -2.5]
 [-2.5  2.5]]
```

```
arr = np.array([2, 8, 15, 25])
print(np.clip(arr, 5, 20))  # Values below 5→5, above 20→20
# Output: [5 8 15 20]
```

```
[ 5  8 15 20]
```