What is NumPy?

- NumPy stands for Numerical Python. NumPy is a Python library used for working with arrays.
- It is an open source project and you can use it freely. NumPy can be used to perform a wide variety of mathematical operations on arrays.
- It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices.

Why we use NumPy?

- In Python we have lists that serve the purpose of arrays, but they are slow to process.
- NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.
- The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.
- Arrays are very frequently used in data science, where speed and resources are very important.

What is NumPy Array?

- An array is a central data structure of the NumPy library. An array is a grid of values and it contains information about the raw data, how to locate an element, and how to interpret an element.
- The most important object defined in NumPy is an N-dimensional array type called ndarray. It describes the collection of items of the same type.

In [ ]:

```
```

In [1]:

```
!pip install numpy
```

```
Requirement already satisfied: numpy in c:\users\admin\anaconda3\lib\site-packages (1.23.5)
```

In [1]:

```
import numpy as np
```

```
In [3]:   lst = [4,3,6,7,8,1,9]
          lst

Out[3]:   [4, 3, 6, 7, 8, 1, 9]
```

```
In [4]:   type(lst)

Out[4]:   list
```

```
In [5]:   arr = np.array([4,3,6,7,8,1,9])
          print(arr)

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

```
In [6]:   arr

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

```
In [8]:   print(type(arr))

          <class 'numpy.ndarray'>
```

```
In [9]:   arr.ndim

Out[9]:   1
```

```
In [ ]:
```

Difference between Numpy Array and Python List

- **Data Type Storage**
    - Homogeneous (Array)
    - Heterogeneous (List)
- **Numerical Operations**
    - Vectorized (Array)
    - Iterative (List)
- **Performance and Speed**

- High Speed and Less Memory (Array)
- Low Speed and More Memory (List)

In [ ]:

diff 1

In [11]:

```
lst = [2,1.44,'3',5,67, True]
lst
```

Out[11]:

```
[2, 1.44, '3', 5, 67, True]
```

In [ ]:

In [12]:

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

Out[12]:

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

In [13]:

```
arr = np.array([1,2,3.8,4,5])
arr
```

Out[13]:

```
array([1. , 2. , 3.8, 4. , 5. ])
```

In [14]:

```
arr = np.array([1,2,3.8,4,'5'])
arr
```

Out[14]:

```
array(['1', '2', '3.8', '4', '5'], dtype='<U32')
```

In [ ]:

diff 2

In [18]:

```
lst = [1,2,3,4,5]
lst
```

Out[18]:

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

In [16]:

```
lst = [1,2,3,4,5] + 5
lst
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[16], line 1
----> 1 lst = [1,2,3,4,5] + 5
      2 lst

TypeError: can only concatenate list (not "int") to list
```

In [17]:

```
lst = [1,2,3,4,5] * 5
lst
```

Out[17]:

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

In [22]:

```
l = []

for i in lst:
    l.append(i*5)

l
```

Out[22]:

```
[5, 10, 15, 20, 25]
```

In [ ]:

```

```

In [23]:

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

Out[23]:

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

```
In [24]:  arr = np.array([1,2,3,4,5]) * 5
          arr

Out[24]:  array([ 5, 10, 15, 20, 25])
```

```
In [25]:  arr = np.array([1,2,3,4,5]) + 5
          arr

Out[25]:  array([ 6,  7,  8,  9, 10])
```

```
In [ ]:
```

```
In [ ]:
```

diff 3

```
In [26]:  [i for i in range(1, 11)]

Out[26]:  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
In [27]:  [i**2 for i in range(1, 11)]

Out[27]:  [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

```
In [34]:  %timeit [i**2 for i in range(1, 1001)]

          109 µs ± 7.78 µs per loop (mean ± std. dev. of 7 runs, 10,000 loops each)
```

```
In [ ]:
```

```
In [29]:  np.arange(1, 11)

Out[29]:  array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

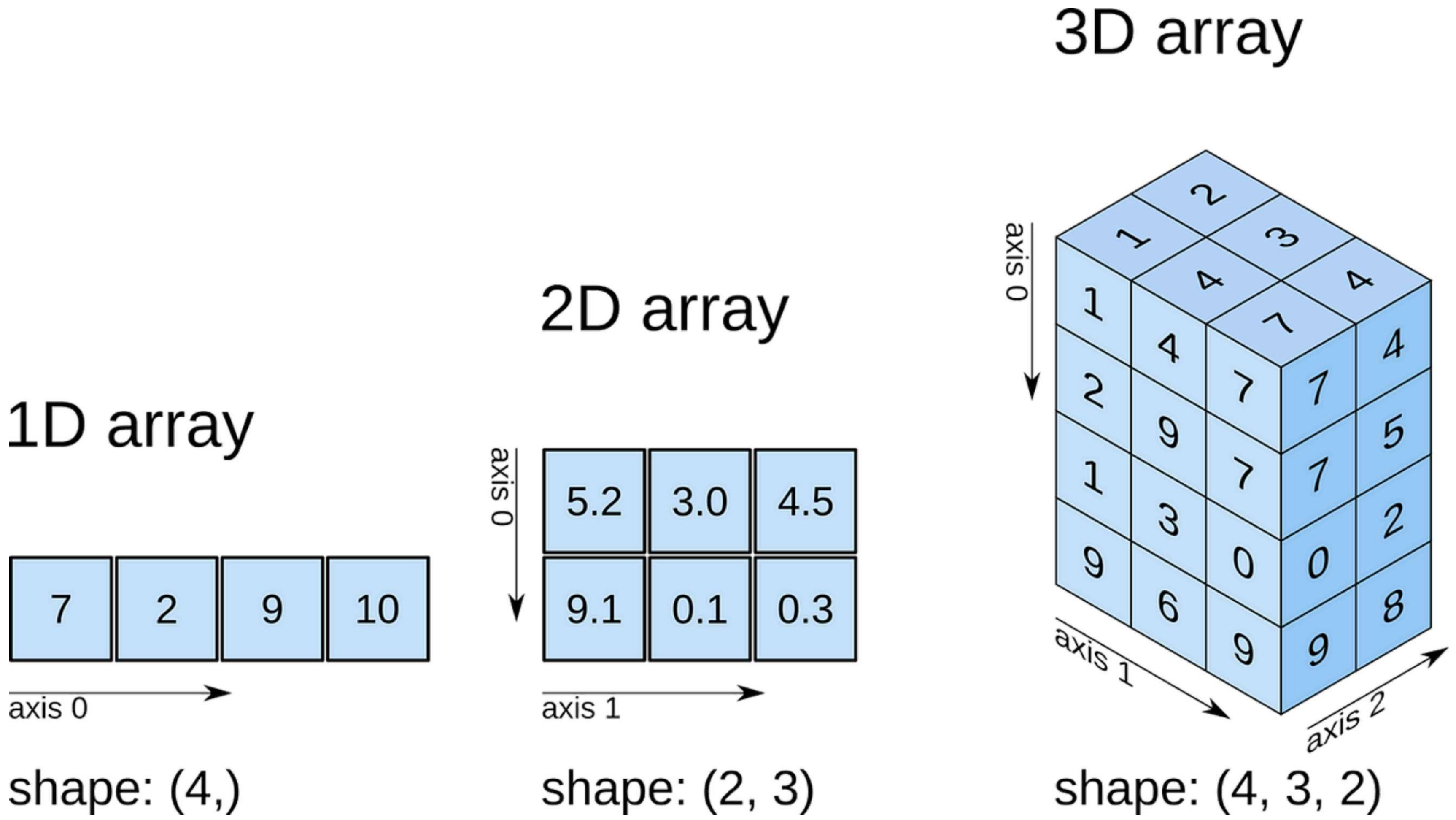
```
In [30]: np.arange(1, 11) ** 2

Out[30]: array([  1,   4,   9,  16,  25,  36,  49,  64,  81, 100])
```

```
In [35]: %timeit np.arange(1, 1001) ** 2

         3.61 µs ± 82.1 ns per loop (mean ± std. dev. of 7 runs, 100,000 loops each)
```

```
In [ ]:
```

```
In [ ]:
```

1D, 2D and 3D Array



In [ ]:

In [ ]:

1D Array

```
In [36]: arr = np.array([1,2,3,4,5])
         arr

Out[36]: array([1, 2, 3, 4, 5])
```

```
In [37]: arr.ndim

Out[37]: 1
```

```
In [38]: arr.shape

Out[38]: (5,)
```

```
In [ ]: 
```

2D Array

```
In [39]: arr = np.array([[1,2,3,4,5]])
         arr

Out[39]: array([[1, 2, 3, 4, 5]])
```

```
In [40]: arr.ndim

Out[40]: 2
```

```
In [41]: arr.shape

Out[41]: (1, 5)
```

```
In [43]: arr = np.array([[1,2,3],[4,5,6]])
         arr

Out[43]: array([[1, 2, 3],
                [4, 5, 6]])
```

```
In [44]:  arr.ndim

Out[44]:  2
```

```
In [45]:  arr.shape

Out[45]:  (2, 3)
```

```
In [ ]:
```

```
In [47]:  arr = np.array([[1,2,3], [4,5,6], [7,8,9]])
          arr

Out[47]:  array([[1, 2, 3],
                 [4, 5, 6],
                 [7, 8, 9]])
```

```
In [48]:  arr.ndim

Out[48]:  2
```

```
In [49]:  arr.shape

Out[49]:  (3, 3)
```

```
In [ ]:
```

3D Array

```
In [50]:  arr = np.array([[[1,2,3,4,5]]])
          arr

Out[50]:  array([[[1, 2, 3, 4, 5]]])
```

In [51]:

```
arr.ndim
```

Out[51]:

```
3
```

In [52]:

```
arr.shape
```

Out[52]:

```
(1, 1, 5)
```

In [ ]:

```

```

In [53]:

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

Out[53]:

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

In [54]:

```
arr.ndim
```

Out[54]:

```
3
```

In [55]:

```
arr.shape
```

Out[55]:

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

In [ ]:

```

```

In [56]:

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

Out[56]:

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

       [[5, 6],
        [7, 8]]])
```

In [57]:

```
arr.shape
```

Out[57]:

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

In [ ]:

```

```

In [58]:

```
arr = np.array([1,2,3,4,5], ndmin=10)
arr
```

Out[58]:

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

In [59]:

```
arr.ndim
```

Out[59]:

```
10
```

In [60]:

```
arr.shape
```

Out[60]:

```
(1, 1, 1, 1, 1, 1, 1, 1, 1, 5)
```

In [ ]:

```

```

In [63]:

```
arr = np.array([1,2,3,4,5], ndmin=33)
arr
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[63], line 1
----> 1 arr = np.array([1,2,3,4,5], ndmin=33)
      2 arr

ValueError: ndmin bigger than allowable number of dimensions NPY_MAXDIMS (=32)
```

In [64]:
```
arr = np.array([1,2,3,4,5], ndmin=32)
arr
```

Out[64]:
```
array([[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[1, 2,
                                  3, 4,
                                  5]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]])
```

In [ ]:
```

```

Special Numpy Functions

In [65]:
```
np.zeros(5)
```

Out[65]:
```
array([0., 0., 0., 0., 0.])
```

In [68]:
```
np.zeros([5, 3], dtype=int)
```

Out[68]:
```
array([[0, 0, 0],
       [0, 0, 0],
       [0, 0, 0],
       [0, 0, 0],
       [0, 0, 0]])
```

In [69]:
```
np.zeros([3, 3, 3], dtype=int)
```

Out[69]:
```
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]]])
```

In [ ]:

```

```

In [71]:

```
np.ones(3)
```

Out[71]:

```
array([1., 1., 1.])
```

In [72]:

```
np.ones([4,4])
```

Out[72]:

```
array([[1., 1., 1., 1.],
       [1., 1., 1., 1.],
       [1., 1., 1., 1.],
       [1., 1., 1., 1.]])
```

In [73]:

```
np.ones([2,4,4], dtype=str)
```

Out[73]:

```
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'],
        ['1', '1', '1', '1'],
        ['1', '1', '1', '1']]], dtype='<U1')
```

In [74]:

```
np.ones([2,4,4], dtype=int)
```

Out[74]:

```
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],
        [1, 1, 1, 1],
        [1, 1, 1, 1]]])
```

In [ ]:

```

```

In [75]:

```python
np.full(5, 3)
```

Out[75]:

```
array([3, 3, 3, 3, 3])
```

In [76]:

```python
np.full([5,5], 2.66)
```

Out[76]:

```
array([[2.66, 2.66, 2.66, 2.66, 2.66],
       [2.66, 2.66, 2.66, 2.66, 2.66],
       [2.66, 2.66, 2.66, 2.66, 2.66],
       [2.66, 2.66, 2.66, 2.66, 2.66],
       [2.66, 2.66, 2.66, 2.66, 2.66]])
```

In [77]:

```python
np.full([3,3,5], 'a')
```

Out[77]:

```
array([[['a', 'a', 'a', 'a', 'a'],
        ['a', 'a', 'a', 'a', 'a'],
        ['a', 'a', 'a', 'a', 'a']],

       [['a', 'a', 'a', 'a', 'a'],
        ['a', 'a', 'a', 'a', 'a'],
        ['a', 'a', 'a', 'a', 'a']],

       [['a', 'a', 'a', 'a', 'a'],
        ['a', 'a', 'a', 'a', 'a'],
        ['a', 'a', 'a', 'a', 'a']]], dtype='<U1')
```

In [ ]:

```

```

In [78]:

```python
np.arange(1, 11)
```

Out[78]:

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

```
In [79]:  np.arange(1, 101)

Out[79]: 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,  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])
```

```
In [81]:  np.arange(0, 101, 5)

Out[81]: array([  0,   5,  10,  15,  20,  25,  30,  35,  40,  45,  50,  55,  60,
                 65,  70,  75,  80,  85,  90,  95, 100])
```

```
In [ ]:
```

Random Number Generation

```
In [2]:  from numpy import random
```

```
In [20]:  random.randint(10, 20, size=1)

Out[20]: array([18])
```

```
In [39]:  random.randint(10, 20, size=10)

Out[39]: array([14, 14, 18, 19, 10, 14, 15, 10, 18, 14])
```

```
In [35]:  random.randint(100, size=(3,3))

Out[35]: array([[17, 66, 15],
                [27,  2, 91],
                [40, 71, 34]])
```

```
In [40]:  random.randint(1, 1000, size=(3,3,3))

Out[40]:  array([[[207, 468, 864],
                  [509, 305, 680],
                  [885, 383, 994]],

                 [[244, 481, 330],
                  [647, 403, 585],
                  [909, 726, 916]],

                 [[214, 139, 102],
                  [120, 923, 139],
                  [631, 802, 814]]])
```

```
In [ ]:
```

```
In [41]:  random.rand(5)

Out[41]:  array([0.47289535, 0.83604223, 0.50309988, 0.27298254, 0.66197567])
```

```
In [46]:  random.rand(3,3)

Out[46]:  array([[0.10595346, 0.08977847, 0.67381134],
                 [0.65867669, 0.356396  , 0.32349365],
                 [0.85932129, 0.76002527, 0.0593251 ]])
```

```
In [47]:  random.rand(2,3,3)

Out[47]:  array([[[0.24310452, 0.19003631, 0.12005265],
                  [0.57682612, 0.69014372, 0.7146583 ],
                  [0.53254379, 0.34729173, 0.86612787]],

                 [[0.73366613, 0.20746809, 0.58170696],
                  [0.60549376, 0.46976956, 0.29867199],
                  [0.44396572, 0.8943039 , 0.95267749]]])
```

In [48]:

```
random.uniform(25000.0, 75000.0, size=5)
```

Out[48]:

```
array([70312.27834884, 67265.17359148, 56555.26459271, 54200.78693539,
       37613.53924801])
```

In [50]:

```
random.uniform(25000.0, 75000.0, size=(5,3))
```

Out[50]:

```
array([[69542.96936656, 47288.48500936, 55308.20989354],
       [67959.81348678, 60212.92525838, 34108.12352746],
       [69332.93187495, 72394.80550523, 31219.051271  ],
       [55922.24772529, 55303.25049668, 40939.1063006 ],
       [58390.84082212, 65794.59033947, 42166.59308679]])
```

In [ ]:

```

```

In [72]:

```
random.choice([3,5,7,9,11,13], size=5)
```

Out[72]:

```
array([ 7,  3,  3, 13,  5])
```

In [73]:

```
random.choice([3,5,7,9,11,13], size=[3,3])
```

Out[73]:

```
array([[ 7,  5,  5],
       [13, 11,  3],
       [ 9,  9,  5]])
```

In [83]:

```
random.choice([3,5,7,9], p=[0, 0.2, 0.8, 0], size=[3,3])
```

Out[83]:

```
array([[7, 5, 7],
       [7, 7, 7],
       [7, 5, 7]])
```

In [85]:

```
random.choice([3,5,7,9], p=[0.7, 0.1, 0.0, 0.2], size=100)
```

Out[85]:

```
array([3, 3, 3, 3, 3, 3, 3, 3, 3, 9, 5, 5, 3, 5, 3, 3, 3, 3, 3, 3, 3, 3,
       9, 3, 3, 3, 3, 9, 9, 9, 3, 3, 3, 3, 3, 9, 3, 3, 3, 3, 9, 5, 3, 3,
       5, 9, 9, 3, 5, 3, 3, 9, 3, 3, 3, 3, 9, 3, 3, 9, 9, 3, 3, 9, 3, 3,
       3, 3, 3, 3, 9, 9, 3, 5, 3, 3, 3, 5, 9, 3, 3, 9, 5, 3, 3, 3, 3, 9,
       3, 3, 3, 3, 5, 3, 3, 3, 3, 3, 9, 9])
```

In [ ]:

```

```

Vectorized Operation

In [87]:

```
arr1 = np.array([1, 2, 3, 4, 5])
arr2 = np.array([1, 2, 3, 4, 5])

print(arr1 + arr2)
```

```
[ 2  4  6  8 10]
```

In [88]:

```
arr1 = np.array([1, 2, 3, 4, 5]) * 5
arr1
```

Out[88]:

```
array([ 5, 10, 15, 20, 25])
```

In [ ]:

```

```

In [89]:

```
arr1 = np.array([1, 2, 3, 4, 5])
arr2 = np.array([1, 2, 3, 4, 5])

np.add(arr1, arr2)
```

Out[89]:

```
array([ 2,  4,  6,  8, 10])
```

In [90]:

```
arr1 = np.array([1, 2, 3, 4, 5])
arr2 = np.array([1, 2, 3, 4, 5])

np.subtract(arr1, arr2)
```

Out[90]:

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

In [91]:

```
arr1 = np.array([1, 2, 3, 4, 5])
arr2 = np.array([1, 2, 3, 4, 5])

np.multiply(arr1, arr2)
```

Out[91]:

```
array([ 1,  4,  9, 16, 25])
```

In [92]:

```
arr1 = np.array([1, 2, 3, 4, 5])
arr2 = np.array([1, 2, 3, 4, 5])

np.divide(arr1, arr2)
```

Out[92]:

```
array([1., 1., 1., 1., 1.])
```

In [93]:

```
arr1 = np.array([1, 2, 3, 4, 5])
arr2 = np.array([1, 2, 3, 4, 5])

np.floor_divide(arr1, arr2)
```

Out[93]:

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

In [94]:

```
arr1 = np.array([1, 2, 3, 4, 5])
arr2 = np.array([1, 2, 3, 4, 5])

np.mod(arr1, arr2)
```

Out[94]:

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

In [95]:

```
arr1 = np.array([1, 2, 3, 4, 5])
arr2 = np.array([1, 2, 3, 4, 5])

np.power(arr1, arr2)
```

Out[95]:

```
array([   1,    4,   27,  256, 3125])
```

In [ ]:

Data Type Conversion

In [98]:

```
arr = np.array([1, 2, 3, 4, 5])
print(arr, '\n')
print(arr.dtype)
```

```
[1 2 3 4 5]

int32
```

In [100]:

```
arr = np.array([1, 2, 3, 4, 5], dtype=float)

print(arr, '\n')
print(arr.dtype)
```

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

float64
```

In [104]:

```
arr = np.array([1, 2, 3, 4, 5], dtype='float16')

print(arr, '\n')
print(arr.dtype)
```

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

float16
```

In [101]:

```
arr = np.array([1, 2, 3, 4, 5], dtype='float')

print(arr, '\n')
print(arr.dtype)
```

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

float64
```

In [102]:

```
arr = np.array([1, 2, 3, 4, 5], dtype='f')

print(arr, '\n')
print(arr.dtype)
```

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

float32
```

In [ ]:

```
```

In [105]:

```
arr = np.array([1, 2, 3, 4, 5], dtype=str)

print(arr, '\n')
print(arr.dtype)
```

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

<U1
```

In [107]:

```
arr = np.array([1, 2, 3, 4, 5], dtype='object')

print(arr, '\n')
print(arr.dtype)
```

```
[1 2 3 4 5]

object
```

In [ ]:

```
```

Memory Management

In [113]:

```
arr = np.array([1, 2, 3, 4, 5], dtype=None)

print(arr, '\n')
print(arr.dtype)
```

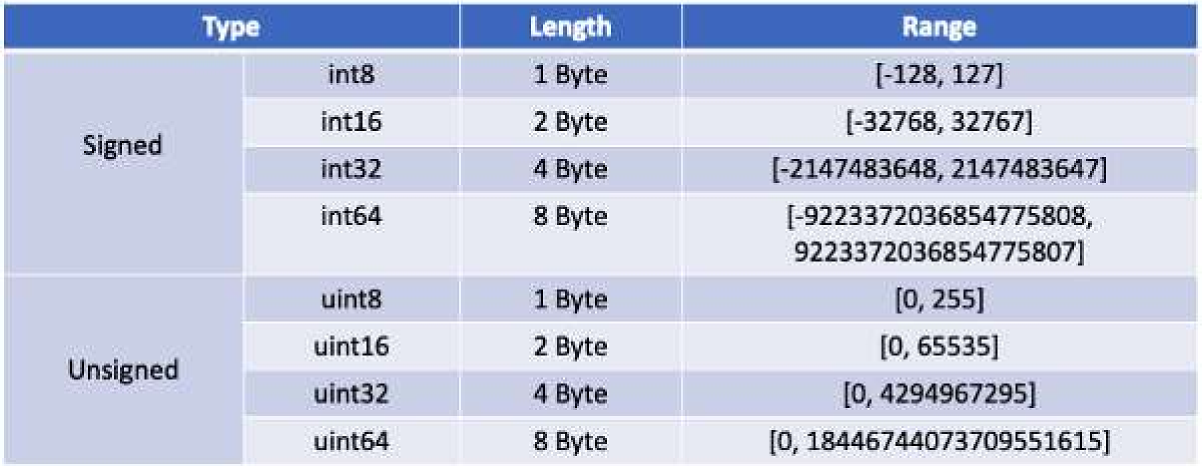
```
[1 2 3 4 5]

int32
```

In [111]:

```
arr
```

Out[111]:

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



Type		Length	Range
Signed	int8	1 Byte	[-128, 127]
	int16	2 Byte	[-32768, 32767]
	int32	4 Byte	[-2147483648, 2147483647]
	int64	8 Byte	[-9223372036854775808, 9223372036854775807]
Unsigned	uint8	1 Byte	[0, 255]
	uint16	2 Byte	[0, 65535]
	uint32	4 Byte	[0, 4294967295]
	uint64	8 Byte	[0, 18446744073709551615]

In [ ]:

```
int8 = -128 +127
```

In [116]:

```
arr = np.array([1, 2, 3, 4, 5], dtype='int8')

print(arr, '\n')
print(arr.dtype)
```

```
[1 2 3 4 5]

int8
```

In [121]:

```
arr = np.array([1, 2, 135, 4, 5], dtype=np.int8)

print(arr, '\n')
print(arr.dtype)
```

```
[   1    2 -121    4    5] 

int8
```

In [122]:

```
arr = np.array([1, 2, 13455, 4, 5], dtype=np.int8)

print(arr, '\n')
print(arr.dtype)
```

```
[   1    2 -113    4    5] 

int8
```

In [123]:

```
arr = np.array([1, 2, 13455, 4, 5], dtype=np.int16)

print(arr, '\n')
print(arr.dtype)
```

```
[    1     2 13455     4     5] 

int16
```

In [128]:

```
arr = np.array([1, 2, 32767151, 4, 5], dtype=np.int16)

print(arr, '\n')
print(arr.dtype)
```

```
[   1    2 -849    4    5] 

int16
```

In [130]:

```
arr = np.array([1, 2, 32767151, 4, 5], dtype = np.int32)

print(arr, '\n')
print(arr.dtype)
```

```
[       1        2 32767151        4        5] 

int32
```

In [ ]:

```

```

Statistical Operation

In [131]:

```
arr = np.array([3,34,5,7,9,4,5,435,46,54,645,46,54,3,4,6])
arr
```

Out[131]:

```
array([  3,  34,   5,   7,   9,   4,   5, 435,  46,  54, 645,  46,  54,
         3,   4,   6])
```

In [132]:

```
len(arr)
```

Out[132]:

```
16
```

In [133]:

```
arr = np.sum([3,34,5,7,9,4,5,435,46,54,645,46,54,3,4,6])
arr
```

Out[133]:

```
1360
```

In [134]:

```
arr = np.min([3,34,5,7,9,4,5,435,46,54,645,46,54,3,4,6])
arr
```

Out[134]:

```
3
```

```
In [142]:  arr = np.argmin([3,34,5,7,9,4,5,435,46,54,645,46,54,3,4,6])
            arr

Out[142]:   0
```

```
In [135]:  arr = np.max([3,34,5,7,9,4,5,435,46,54,645,46,54,3,4,6])
            arr

Out[135]:   645
```

```
In [143]:  arr = np.argmax([3,34,5,7,9,4,5,435,46,54,645,46,54,3,4,6])
            arr

Out[143]:   10
```

```
In [136]:  arr = np.mean([3,34,5,7,9,4,5,435,46,54,645,46,54,3,4,6])
            arr

Out[136]:   85.0
```

```
In [137]:  arr = np.median([3,34,5,7,9,4,5,435,46,54,645,46,54,3,4,6])
            arr

Out[137]:   8.0
```

```
In [138]:  arr = np.sort([3,34,5,7,9,4,5,435,46,54,645,46,54,3,4,6])
            arr

Out[138]:   array([  3,   3,   4,   4,   5,   5,   6,   7,   9,  34,  46,  46,  54,
                    54, 435, 645])
```

```
In [139]:  arr = np.std([3,34,5,7,9,4,5,435,46,54,645,46,54,3,4,6])
            arr

Out[139]:   176.9773996870787
```

```
In [140]:  arr = np.var([3,34,5,7,9,4,5,435,46,54,645,46,54,3,4,6])
           arr
```

```
Out[140]: 31321.0
```

```
In [141]:  176.9773996870787**2
```

```
Out[141]: 31321.000000000004
```

```
In [145]:  arr = np.array([3,34,5,7,9,4,5,435,46,54,645,46,54,3,4,6])
           arr
```

```
Out[145]: array([  3,  34,   5,   7,   9,   4,   5, 435,  46,  54, 645,  46,  54,
                   3,   4,   6])
```

```
In [ ]:
```

```
In [148]:  arr = np.sort([3,34,5,7,9,4,5,435,46,54,645,46,54,3,4,6])
           arr
```

```
Out[148]: array([  3,   3,   4,   4,   5,   5,   6,   7,   9,  34,  46,  46,  54,
                  54, 435, 645])
```

```
In [155]:  minn = np.quantile([3,34,5,7,9,4,5,435,46,54,645,46,54,3,4,6], 0.0)
           minn
```

```
Out[155]: 3.0
```

```
In [152]:  Q1 = np.quantile([3,34,5,7,9,4,5,435,46,54,645,46,54,3,4,6], 0.25)
           Q1
```

```
Out[152]: 4.75
```

```
In [153]:  Q2 = np.quantile([3,34,5,7,9,4,5,435,46,54,645,46,54,3,4,6], 0.5)
           Q2
```

```
Out[153]: 8.0
```

In [154]:

```
Q3 = np.quantile([3,34,5,7,9,4,5,435,46,54,645,46,54,3,4,6], 0.75)
Q3
```

Out[154]:

```
48.0
```

In [156]:

```
maxx = np.quantile([3,34,5,7,9,4,5,435,46,54,645,46,54,3,4,6], 1.0)
maxx
```

Out[156]:

```
645.0
```

In [ ]:

Other Operations

In [158]:

```
arr = random.randint(15, 50, size=[3,3])
arr
```

Out[158]:

```
array([[34, 33, 19],
       [47, 34, 39],
       [22, 46, 33]])
```

In [159]:

```
arr
```

Out[159]:

```
array([[34, 33, 19],
       [47, 34, 39],
       [22, 46, 33]])
```

In [160]:

```
arr.flatten()
```

Out[160]:

```
array([34, 33, 19, 47, 34, 39, 22, 46, 33])
```

In [161]:

```
arr.ravel()
```

Out[161]:

```
array([34, 33, 19, 47, 34, 39, 22, 46, 33])
```

In [ ]:

```

```

In [162]:

```
arr
```

Out[162]:

```
array([[34, 33, 19],
       [47, 34, 39],
       [22, 46, 33]])
```

In [163]:

```
arr.transpose()
```

Out[163]:

```
array([[34, 47, 22],
       [33, 34, 46],
       [19, 39, 33]])
```

In [164]:

```
arr.T
```

Out[164]:

```
array([[34, 47, 22],
       [33, 34, 46],
       [19, 39, 33]])
```

In [ ]:

```

```

In [165]:

```
arr
```

Out[165]:

```
array([[34, 33, 19],
       [47, 34, 39],
       [22, 46, 33]])
```

In [166]:

```
arr.shape
```

Out[166]:

```
(3, 3)
```

In [167]:

```
arr.reshape(9, 1)
```

Out[167]:

```
array([[34],
       [33],
       [19],
       [47],
       [34],
       [39],
       [22],
       [46],
       [33]])
```

In [168]:

```
arr.reshape(1, 9)
```

Out[168]:

```
array([[34, 33, 19, 47, 34, 39, 22, 46, 33]])
```

In [ ]:

In [ ]:

In [170]:

```
arr
```

Out[170]:

```
array([[34, 33, 19],
       [47, 34, 39],
       [22, 46, 33]])
```

In [171]:

```
arr.reshape(1, -1)
```

Out[171]:

```
array([[34, 33, 19, 47, 34, 39, 22, 46, 33]])
```

In [172]:

```
arr.reshape(-1, 1)
```

Out[172]:

```
array([[34],
       [33],
       [19],
       [47],
       [34],
       [39],
       [22],
       [46],
       [33]])
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: