Numpy which stands for Numerical Python is a library consisting of multi-dimesnional array object and a collection of routine processing of those arrays.

Using Numpy mathematical and logical operations on an array can be performed.

In [ ]:

```
1
```

To install numpy pip install numpy

In [ ]:

```
1
```

import library

In [1]:

```
1 import numpy as np
```

In [2]:

```
1 np.__version__
```

Out[2]:

```
'1.20.1'
```

ndarray

```
1 The most important object defined in Numpy is N-dimensional array called as ndarry
2
3 It describes the collection of items of the same type.
4
5 Items in the collection can be accessed using zero based indexing technique
6
7 Every item in an ndarry takes the same size of block in the memory
8
9 Each element in ndarray is an object of data-type called as dtype
```

In [ ]:

```
1
```

creating numpy array

In [3]:

```
1 my_array = np.array([10,20,30])
```

In [4]:

```
1 print(my_array)
```

```
[10 20 30]
```

In [5]:

```
1 type(my_array)
```

Out[5]:

```
numpy.ndarray
```

In [ ]:

```
1
```

In [6]:

```
1 #more than one dimesnional array
```

In [7]:

```
1 arr = np.array([[10,20],[30,40]])
```

In [8]:

```
arr
```

Out[8]:

```
array([[10, 20],
       [30, 40]])
```

In [9]:

```
type(arr)
```

Out[9]:

```
numpy.ndarray
```

08th August 2021

Numpy's main object is the homogeneous multi-dimesnional array. It is a table of elements(usually numbers) all of the same type, indexed by a tuple of non-negative integers.

2) In Numpy dimensions are called as axes

3) ndarray.ndim: The number of axes (dimensions) of the array

4) ndarray.shape : It tells us the dimesnions of the array in the form of a tuple

5) ndarray.size : The total number of elements present in the array.

6) ndarray.dtype: The object describing the type of the element in the array

In [9]:

```
arr
```

Out[9]:

```
array([[10, 20],
       [30, 40]])
```

In [10]:

```
arr.ndim
```

Out[10]:

```
2
```

In [11]:

```
arr.shape
```

Out[11]:

```
(2, 2)
```

In [12]:

```
arr.size
```

Out[12]:

```
4
```

In [13]:

```
arr.dtype
```

Out[13]:

```
dtype('int64')
```

In [ ]:

```
49849.7474900083
```

```
1) np.bool_ : bool  Boolean (True or False)
2) np.byte  : signed char   (dependent on OS architecture)
3) np.ubyte : unsigned byte (dependent on OS)
4) np.int : Default integer type
5) np.int8 : Byte(-128 to 127)
6) np.int16: integer(-32768 to 32767)
7) np.int32: integer(-2147483648 to 214783647)
8) np.int64: integer(-9223372036854775808 to 9223372036854775807)
7) np.float16:sign bit, 5 bits exponent, 10 bits mantissa
8) np.float32: sign bit, 8 bits exponent, 23 bits mantissa
9) np.float64: sign bit 11 bits exponent, 52 bits mantissa
10) complex64: complex number represented by 32 bit float (real and imaginary)
11) complex128: complex number represented by 64 bit float (real and imaginary)
```

In [ ]:

```
1
```

Creating numpy Array

In [16]:

```
1 arr = np.array([[100,200,300],[500,600,7000]])
```

In [17]:

```
1 arr
```

Out[17]:

```
array([[ 100,  200,  300],
       [ 500,  600, 7000]])
```

In [18]:

```
1 arr.shape
```

Out[18]:

```
(2, 3)
```

In [19]:

```
1 arr.size
```

Out[19]:

```
6
```

In [23]:

```
1 arr = arr.reshape(3,2)
```

In [24]:

```
1 arr
```

Out[24]:

```
array([[ 100,  200],
       [ 300,  500],
       [ 600, 7000]])
```

In [ ]:

```
1
```

In [25]:

```
1 np.array(10,20,30) #error
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-25-c10f88926b6f> in <module>
----> 1 np.array(10,20,30) #error

TypeError: array() takes from 1 to 2 positional arguments but 3 were given
```

In [26]:

```
1 np.array([10,20,30]) #correct way
```

Out[26]:

```
array([10, 20, 30])
```

In [ ]:

```
1
```

arange

In [31]:

```
1 #this function creates the numbers between a range, you have to provide the start point, end point, step
2
3 res = np.arange(1,21)
```

In [30]:

```
res
```

Out[30]:

```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19, 20])
```

In [32]:

```
res1 = np.arange(1,21,2)
```

In [33]:

```
res1
```

Out[33]:

```
array([ 1,  3,  5,  7,  9, 11, 13, 15, 17, 19])
```

In [ ]:

```

```

In [34]:

```
res2 = np.arange(21)
```

In [35]:

```
res2
```

Out[35]:

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20])
```

In [36]:

```
res
```

Out[36]:

```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19, 20])
```

In [37]:

```
res.ndim
```

Out[37]:

```
1
```

In [39]:

```
res.size
```

Out[39]:

```
20
```

In [43]:

```
res.reshape(2,10)
```

Out[43]:

```
array([[ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10],
       [11, 12, 13, 14, 15, 16, 17, 18, 19, 20]])
```

In [ ]:

```

```

In [ ]:

```

```

In [52]:

```
res = np.arange(1,26).reshape(5,5)
```

In [ ]:

```

```

In [53]:

```
1 np.arange(0,3,0.3)
```

Out[53]:

```
array([0. , 0.3, 0.6, 0.9, 1.2, 1.5, 1.8, 2.1, 2.4, 2.7])
```

In [ ]:

```
1
```

In [ ]:

```
1
```

In [45]:

```
1 res
```

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]])
```

In [ ]:

```
1
```

In [49]:

```
1 np.empty([3,2],dtype=complex)
```

Out[49]:

```
array([[4.67629672e-310+0.j, 0.00000000e+000+0.j],
       [0.00000000e+000+0.j, 0.00000000e+000+0.j],
       [0.00000000e+000+0.j, 0.00000000e+000+0.j]])
```

In [50]:

```
1 np.zeros([5,5])
```

Out[50]:

```
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.]])
```

In [ ]:

```
1
```

In [51]:

```
1 np.ones([5,4])
```

Out[51]:

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

In [ ]:

```
1
```

In [ ]:

```
1
```

In [59]:

```
np.linspace(0,2,50)
```

Out[59]:

```
array([0.        , 0.04081633, 0.08163265, 0.12244898, 0.16326531,
       0.20408163, 0.24489796, 0.28571429, 0.32653061, 0.36734694,
       0.40816327, 0.44897959, 0.48979592, 0.53061224, 0.57142857,
       0.6122449 , 0.65306122, 0.69387755, 0.73469388, 0.7755102 ,
       0.81632653, 0.85714286, 0.89795918, 0.93877551, 0.97959184,
       1.02040816, 1.06122449, 1.10204082, 1.14285714, 1.18367347,
       1.2244898 , 1.26530612, 1.30612245, 1.34693878, 1.3877551 ,
       1.42857143, 1.46938776, 1.51020408, 1.55102041, 1.59183673,
       1.63265306, 1.67346939, 1.71428571, 1.75510204, 1.79591837,
       1.83673469, 1.87755102, 1.91836735, 1.95918367, 2.        ])
```

In [56]:

```
0 + 0.25
```

Out[56]:

```
0.25
```

In [57]:

```
0.25 + 0.25
```

Out[57]:

```
0.5
```

In [58]:

```
0.5 + 0.25
```

Out[58]:

```
0.75
```

In [62]:

```
import sys
np.set_printoptions(threshold=sys.maxsize)
```

In [ ]:

```

```

In [63]:

```
np.arange(10000).reshape(100,100)
```

Out[63]:

```
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,
```

In [ ]:

```

```

In [64]:

```
arr1 = np.array([20,30,40,50])
arr2 = np.arange(4)
```

In [65]:

```
arr1
```

Out[65]:

```
array([20, 30, 40, 50])
```

In [66]:

```
1 arr2
```

Out[66]:

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

In [ ]:

```
1
```

In [67]:

```
1 arr1 - arr2
```

Out[67]:

```
array([20, 29, 38, 47])
```

In [ ]:

```
1
```

In [68]:

```
1 arr1 + arr2
```

Out[68]:

```
array([20, 31, 42, 53])
```

In [ ]:

```
1
```

In [69]:

```
1 arr1 * arr2
```

Out[69]:

```
array([  0,  30,  80, 150])
```

In [ ]:

```
1
```

In [70]:

```
1 arr1 / arr2
```

```
<ipython-input-70-c69aea946715>:1: RuntimeWarning: divide by zero encountered in true_divide
  arr1 / arr2
```

Out[70]:

```
array([        inf, 30.        , 20.        , 16.66666667])
```

In [ ]:

```
1
```

In [ ]:

```
1
```

In [71]:

```
1 arr2
```

Out[71]:

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

In [72]:

```
1 np.sin(arr2)
```

Out[72]:

```
array([0.        , 0.84147098, 0.90929743, 0.14112001])
```

In [ ]:

```
1
```

In [73]:

```
1 np.cos(arr2)
```

Out[73]:

```
array([ 1.        ,  0.54030231, -0.41614684, -0.9899925 ])
```

In [ ]:

```
1
```

In [74]:

```
1 np.tan(arr2)
```

Out[74]:

```
array([ 0.        ,  1.55740772, -2.18503986, -0.14254654])
```

In [ ]:

```
1
```

In [75]:

```
1 np.cosh(arr2)
```

Out[75]:

```
array([ 1.        ,  1.54308063,  3.76219569, 10.067662  ])
```

In [ ]:

```
1
```

In [ ]:

```
1
```

In [79]:

```
1 arr = np.ones((2,3),dtype=int)
```

In [80]:

```
1 arr
```

Out[80]:

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

In [81]:

```
1 arr
```

Out[81]:

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

In [84]:

```
1 arr = arr*3
```

In [85]:

```
1 arr
```

Out[85]:

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

In [ ]:

```
1
```

In [86]:

```
1 #to generate numbers in a random fashion we have a function np.random.random
2
3 arr_random = np.random.random((3,3))
```

In [87]:

```
1 arr_random
```

Out[87]:

```
array([[0.06902664, 0.3179992 , 0.65851072],
       [0.55165629, 0.99494097, 0.33297209],
       [0.02577726, 0.76636974, 0.70936703]])
```

In [88]:

```
1 arr_random.max()
```

Out[88]:

```
0.9949409716218361
```

In [89]:

```
1 arr_random.min()
```

Out[89]:

```
0.025777260622127818
```

In [91]:

```
1 arr_random.mean()
```

Out[91]:

```
0.4918466593384639
```

In [ ]:

```
1
```

In [92]:

```
1 arr = np.arange(10,21)
```

In [93]:

```
1 arr
```

Out[93]:

```
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20])
```

In [ ]:

```
1
```

In [99]:

```
1 arr[5]
```

Out[99]:

```
15
```

In [104]:

```
1 arr[3:10]
```

Out[104]:

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

In [105]:

```
1 arr[3:10:2]
```

Out[105]:

```
array([13, 15, 17, 19])
```

In [106]:

```
1 arr
```

Out[106]:

```
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20])
```

In [109]:

```
1 arr[3:10]
```

Out[109]:

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

In [110]:

```
1 arr[3:10] + 10
```

Out[110]:

```
array([23, 24, 25, 26, 27, 28, 29])
```

In [111]:

```
1 arr[3:10] * 4
```

Out[111]:

```
array([52, 56, 60, 64, 68, 72, 76])
```

In [ ]:

```
1
```

In [112]:

```
1 arr = arr[3:10]
```

In [113]:

```
1 arr
```

Out[113]:

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

In [ ]:

```
1
```