NumPy

- NumPy is a python package that stands for 'Numerical Python'. It is the library we use for scientific computing which contains a powerful n-dimentional array object.

uses

- Mumpy arrays provide tools for integrating C,C++ etc
- It is also useful in linear algebra, random number capability

why NumPy is used in Python?

- we use python NumPy array instead of a list because of the following reasons:

1. Less Memory
2. Fast
3. Convenient

In [1]:

```python
import numpy as np
```

In [2]:

```python
l=[12,23,34,54,67,78]
```

In [3]:

```python
type(l)
```

Out[3]:

```
list
```

In [4]:

```python
a1=np.array(l) # ver list to numpy array
a1
```

Out[4]:

```
array([12, 23, 34, 54, 67, 78])
```

In [5]:

```python
type(a1)
```

Out[5]:

```
numpy.ndarray
```

In [6]:

```python
# memory address of an array object
a1.dtype
```

Out[6]:

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

In [7]:

```python
# convert integer array to float
a1.astype(float)
```

Out[7]:

```
array([12., 23., 34., 54., 67., 78.])
```

In [8]:

```python
a1.data
```

Out[8]:

```
<memory at 0x0000023FCF700280>
```

arange

```
In [9]:  # generate evenly spaced number between 0 to 20
         np.arange(0,20)

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

In [10]: # generate number between 0 to 50 with a space of 5
         np.arange(0,50,5)

Out[10]: array([ 0,  5, 10, 15, 20, 25, 30, 35, 40, 45])

In [12]: # shape of an array
         a1.shape

Out[12]: (6,)

In [13]: # datatype of object
         a1.dtype

Out[13]: dtype('int32')

In [14]: # bytes consumed by array
         a1.nbytes

Out[14]: 24

In [15]: #Length of array
         len(a1)

Out[15]: 6

In [16]: # generate array of zeros

         np.zeros(12)

Out[16]: array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])

In [18]: # generate array of ones
         np.ones(12)

Out[18]: array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])

In [19]: # repeat 5 six times in an array
         np.repeat(5,6)

Out[19]: array([5, 5, 5, 5, 5, 5])

In [21]: # repeat each element in an array five times
         a3=np.array([12,23,55])
         np.repeat(a3,5)

Out[21]: array([12, 12, 12, 12, 12, 23, 23, 23, 23, 23, 55, 55, 55, 55, 55])

In [23]: # generate array of even numbers
         np.arange(0,100,2)

Out[23]: array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32,
                34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66,
                68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98])
```

```
In [24]:  # generate array of odd numbers
          np.arange(1,100,2)

Out[24]: array([ 1,  3,  5,  7,  9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33,
                35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67,
                69, 71, 73, 75, 77, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97, 99])
```

```
In [25]:  # generate array of even numbers
          a4=np.arange(1,100)
          a4[a4%2==0]

Out[25]: array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32,
                34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66,
                68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98])
```

```
In [26]:  # generate array of odd numbers
          a5=np.arange(1,100)
          a5[a5%2==1]

Out[26]: array([ 1,  3,  5,  7,  9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33,
                35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67,
                69, 71, 73, 75, 77, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97, 99])
```

```
In [27]:  # generate evenly spaced 4 numbers between 10 to 20
          np.linspace(10,20,4)

Out[27]: array([10.        , 13.33333333, 16.66666667, 20.        ])
```

```
In [28]:  # create an array of random values
          np.random.random(4)

Out[28]: array([0.82774763, 0.13669453, 0.62665672, 0.56139139])
```

```
In [29]:  # create an array of random integer numbers
          np.random.randint(0,500,5)

Out[29]: array([393, 440, 268, 452, 271])
```

```
In [33]:  np.random.randint(0,4,2)

Out[33]: array([1, 0])
```

```
In [38]:  a6=np.random.random(8)  # a6 is array of 8 random numbers
          a6

Out[38]: array([0.71317011, 0.12651365, 0.14273755, 0.25888958, 0.67237969,
                0.2360219 , 0.78731734, 0.95515595])
```

```
In [39]:  # generate an array of random integer numbers
          np.random.randint(0,500,8)  #---> generate integer only

Out[39]: array([ 65,   4, 297, 334, 185, 134, 273, 383])
```

```
In [41]:  a7=np.random.uniform(5,10,8)
          a7

Out[41]: array([9.54556131, 9.32615299, 8.15454748, 9.5348872 , 7.02982583,
                9.70815939, 5.89915051, 6.74760015])
```

```
In [42]:  np.floor(a7)  # --> to remove the decimal part

Out[42]: array([9., 9., 8., 9., 7., 9., 5., 6.])
```

```
In [43]: a7
```

```
Out[43]: array([9.54556131, 9.32615299, 8.15454748, 9.5348872 , 7.02982583,
                9.70815939, 5.89915051, 6.74760015])
```

```
In [44]: np.trunc(a7) # --> to remove the decimal part
```

```
Out[44]: array([9., 9., 8., 9., 7., 9., 5., 6.])
```

```
In [45]: print(np.__version__)  # shows the version of NumPy
```

```
1.19.2
```

Note

- The array object in NumPY is called ndarray.
- We can create a NumPy ndarray object by using the array() function

```
In [47]: b=np.array([12,13,45,78,59,42]) # create array from a List
         print(b)
         print(type(b))
```

```
[12 13 45 78 59 42]
<class 'numpy.ndarray'>
```

```
In [49]: bt=np.array((12,13,45,78,59,42)) # create array from a Tuple
         print(bt)
         print(type(bt))
```

```
[12 13 45 78 59 42]
<class 'numpy.ndarray'>
```

Dimension

- A dimension in array is one level of array depth(nested array)

0-D array

- 0-D array or scalars are the elements in an arrary. Each value in an array is a 0-D array.

```
In [50]: # example of 0-D array
         ae=np.array(5)
         print(ae)
```

```
5
```

1-D array

- an array that has 0-D arrays as its elements is called uni-dimentional or 1-D array

```
In [51]: # ex of 1-D array
         u1=np.array([1,2,3,4,5])
         print(u1)
```

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

2-D array

- an array that has 1-D arrays as its elements is called a 2-D array.
- these are often used to represent matrix

```
In [53]: u2=np.array([[1,2,3],[4,5,6],[7,8,9]])
         print(u2)
```

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

3-D array

- an array that has 2-D array (matrix) as its elements is called 3-d array

```
In [55]: u3=np.array([[[1,2,3],[4,5,6]],[[10,20,30],[40,50,60]]])
         u3
```

```
Out[55]: array([[[ 1,  2,  3],
                 [ 4,  5,  6]],

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

```
In [ ]:
```