# Numpy

# Part-01

## Basic of numpy:

Numpy is the core library for scientific and numeric computing in python.
It provides high performance multidimensional array objects and tools for working with arrays.

### Why should we use numpy arrays when we have a python list?

- Fast
- Less memory
- Convenient

## Let's code:

Creating a numpy array:

1. **np.array()**
   Like: a = np.array([12,34,5,6])

2. **Finding numpy array's number**
   A[2] ⇒ 34
   A[0:3] ⇒ [12,34,5]

3. **For check how fast numpy array:**

   %timeit [i+1 for i in range(1000)]

⇒ 140 µs ± 3.97 µs per loop (mean ± std. dev. of 7 runs, 10000 loops each)

%timeit np.arange(1000)+1

⇒ 6.36 µs ± 46.4 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)

4. **Creating two dimensional array**

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

5. **Creating a array in range⇒ use of arange**

```
H = np.arange(1,100,1)
```

1--99 , Gap is one→ 1 or i called that is step size

6. **Also we creating a array using⇒ linspace**

Here , we pass 3 numbers

First two given a range like(1-10)

Last number tells how many numbers you want to create.

Example:

```
h=np.linspace(1,10,9)
```

Here, 1-10 is the range and 9 elements are created.

```
[ 1.    2.125  3.25   4.375  5.5    6.625  7.75   8.875 10.   ]
```

7. **If you need any file data load**

Np.loadtxt

Example:

```
h=np.loadtxt("fileName", delimiter="")
```

8. If you want, a random value np.array⇒ you should use empty

(i) a=np.empty(2) → One -D

Output: array([1.01459836e-311, 0.00000000e+000])

(ii) a=np.empty((2,2)) → Two-D

Output: array([[2.12199579e-314, 1.01431399e-311],
[5.41495948e-321, 1.01431399e-311]])

9. For identityMatrix→ Identity

```
A = np.identity(5)
print(A)
```

```
[[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.]]
```

By default , the data type is float.

10. If you want only one 2-D column show .You can use **arrName[:,column index]**

```
Data = [[1 2 4]
        [4 5 7]]
Data[:,2]
array([4, 7], dtype=int16)
```

## Some mathematical function

1. **Find even Number in np.array**

```
lst=np.array([1,2,3,5,6,7,8])
a= lst%2==0
print(lst[a])
```

```
[2 6 8]
```

2. **Replace the element by condition--**

Like:

```
lst=np.array([1,2,3,5,6,7,8])
lst[lst%2!=0]=1
print(lst)
```

```
[1 2 1 1 6 1 8]
```

**More:**

```
A = np.array(['a','z','b','c'])
A[A!="a"]="m"
print(A)
```

```
['a' 'm' 'm' 'm']
```

## Function/attribute

1. **How to Check numpy array dimension-n-D**

   Example:

   ```
   lst = np.array([[1,2]])
   Lst.ndim
   ```

   Output: 2

2. **If you want to check data type, array size, bytes--**

   (i) arrayName.dtype→ For check data type
   (ii) arrayName.size → For Size
   (iii) arrayName.nbytes → Memory storage

3. **Here I discuss some Important function**

   (i) arrayName.argmin() → return array min number index
   (ii) arrayName.argmax() → return array max number index
   (iii) arrayName.argsort() → return index for sort

***** output in index *****

4. If you want , numpy array with zero or one⟶

```
b=np.zeros(6)
b
```

array([0., 0., 0., 0., 0., 0.])  ** By default data type is float.

```
b=np.ones(5)
b
```

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

Also you can create n-D zeros and ones.

```
b=np.ones((5,5))
b
```

```
[[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.]]
```

5. If you want to reshape a array--

You can use "reshape"⟶ It takes two parameter (one⟶ How many array, two⟶ number of element)

```
a=np.arange(99)
a
```

```
[ 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]
```

```
print(a.reshape(3,33))
```

```
[[ 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]]
```

Return previous array.

**a.ravael()**

6. **sum()--> Sum array value**
7. **sum(axis=0)--> sum 2-D array column value**
8. **sum(axis=1)--> sum 2-D array row value.**

**Example:**

**(i)**

```
a=np.array([1,2,3,4,5])
a.sum()
15
```

(ii)

```
a=np.array([[1,2],[3,4]])
a.sum(axis=0)
[4, 6]
```

(iii)

```
a=np.array([[1,2],[3,4]])
a.sum(axis=1)
[3,7]
```

# Some Arithmetic Function

## 1.Addition,subtraction,Multiplication , Division

```
a=np.array([1,2,3,4])
b=np.array([5,6,7,8])
c=a+b
d=a-b
print(c)
print(d)
```

```
[6  8 10 12]
[-4 -4 -4 -4]
```

## 2. If you want to see value base of condition

```
a=np.array([12,3,4,6]
print(a.where(a>4)
[12,6]
```

*******End********

Ready for second Part??????