```
#1D array
import numpy as np
a = np.array([1,2,3])
print("1D array : ",a)


#2D array
a2 = np.array([[1,2,3],[4,5,6]])
print("2D array : ",a2)

#3D array
a3 = np.array([[[[1,2,3],[4,5,6],[7,8,9]]]])
print("3D array : ",a3)
```

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

```
b = np.arange(24)
print(b)
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23]
```

```
#minimum dimensions
a = np.array([1,2,3,4,5],ndmin = 2)
print(a)
```

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

```
#dtype paramater
a = np.array([1,2,3],dtype = complex)
print(a)
```

```
[1.+0.j 2.+0.j 3.+0.j]
```

```
a = np.array([[1,2],[3,4]])
print(a.shape)
```

```
(2, 2)
```

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

```
[[1 2]
 [3 4]
 [5 6]]
```

```
#1D array
a = np.arange(24)
a.ndim
```

```
1
```

```
#now reshape it
b = a.reshape(2,3,4)
print(b)
#b is having three dimensions
```

```
[[[ 0  1  2  3]
  [ 4  5  6  7]
  [ 8  9 10 11]]

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]
```

```
#dtype of array is int8(1 byte)
x = np.array([1,2,3,4,6],dtype = np.int8)
print(x.itemsize)
```

```
1
```

```
#dtype  of array is now float32(4 bytes)
x = np.array([1,2,3,4,5],dtype = np.float32)
print(x.itemsize)
```

```
     4
```

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

```
       C_CONTIGUOUS : True
       F_CONTIGUOUS : True
       OWNDATA : True
       WRITEABLE : True
       ALIGNED : True
       WRITEBACKIFCOPY : False
```

```
x = np.empty([3,2],dtype = int)
print(x)
```

```
     [[1 2]
      [3 4]
      [5 6]]
```

```
x = np.zeros([3,2],dtype = int)
print(x)
```

```
     [[0 0]
      [0 0]
      [0 0]]
```

```
import numpy as np
c = np.linspace(5,10,5)#start,end,number of points
c
```

```
     array([ 5.  ,  6.25,  7.5 ,  8.75, 10.  ])
```

```
d = np.ones((3,5))
d
```

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

```
x = np.zeros((3,3))
x
```

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

```
y = np.eye(3)#creates a matrix with 1 as the diagonals and 0 as non-diagonals
y
```

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

```
z = np.eye(3,2)
z
```

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

```
a = np.diag([1,2,3,4])#construct a diagonal array
a
```

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

```
a = np.random.rand(4)
a
```

```
     array([0.26855035, 0.48576027, 0.00687081, 0.08636429])
```

```
#we can explicitly specify the required data type
a = np.arange(10,dtype ='float')
a
```

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

```
b = np.array([1+2j, 5+1j])
b
```

```
     array([1.+2.j, 5.+1.j])
```

```
c = np.array([True,False,True])
display(c.dtype)
```

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

```
a = np.arange(10)
print(a)
print(a[5])
print(a[-2])
```

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

```
b = np.diag([1,2,3])
print(b)
print(b[2,2])
```

```
     [[1 0 0]
      [0 2 0]
      [0 0 3]]
     3
```

```
b[2,1]=10
b
```

```
     array([[ 1,  0,  0],
            [ 0,  2,  0],
            [ 0, 10,  3]])
```

```
#slicing
a = np.arange(10)
print(a[1:10:2])
```

```
     [1 3 5 7 9]
```

```
b= np.arange(10)
b[5:] = 10 #assign 10 from index 5 to end
print(b)
```

```
     [ 0  1  2  3  4 10 10 10 10 10]
```

```
a = np.arange(10)
b = a[::2]
np.shares_memory(a,b)
```

```
     True
```

```
c = a[::2].copy()#force the copy
np.shares_memory(a,c)
```

```
     False
```

```
c[0] = 5
print(c)
print(a)
```

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

Using Boolean Mask

```
a = np.random.randint(0,20,15)
print(a)
```

```
[17 11  5  4 13 17 11 10  3  4 16 11 11  9 14]
```

```
mask = (a%2==0)
```

```
even_numbers = a[mask]
even_numbers
```

```
array([ 4, 10,  4, 16, 14])
```

```
a[mask]=-1#it can be very useful to assign a new value to sub array
a
```

```
array([17, 11,  5, -1, 13, 17, 11, -1,  3, -1, -1, 11, 11,  9, -1])
```

Using Integer Array

```
a = np.arange(0,100,10)
print(a)
```

```
[ 0 10 20 30 40 50 60 70 80 90]
```

```
a[[9,7]] = -200
print(a)
print(b)
```

```
[   0   10   20   30   40   50   60 -200   80 -200]
[0 2 4 6 8]
```

NUMERICAL OPERATION ON NUMPY

Element wise Operation

```
a = np.arange(10)
print(a+1)
```

```
[ 1  2  3  4  5  6  7  8  9 10]
```

```
print(a**2)
```

```
[ 0  1  4  9 16 25 36 49 64 81]
```

```
b = np.ones(10)+1
print("b =",b)
print("a-b =",a-b)
```

```
b = [2. 2. 2. 2. 2. 2. 2. 2. 2. 2.]
a-b = [-2. -1.  0.  1.  2.  3.  4.  5.  6.  7.]
```

```
print(a*b)
```

```
[ 0.  2.  4.  6.  8. 10. 12. 14. 16. 18.]
```

```
#Matrix multiplication
c = np.diag([1,2,3,4])
print(c)
print("*"*100)
print(c*c)
print("*"*100)
print(c.dot(c))
```

```
[[1 0 0 0]
 [0 2 0 0]
 [0 0 3 0]
 [0 0 0 4]]
****************************************************************************************************
[[ 1  0  0  0]
 [ 0  4  0  0]
 [ 0  0  9  0]
 [ 0  0  0 16]]
```

```
****************************************************************************************************
[[ 1  0  0  0]
 [ 0  4  0  0]
 [ 0  0  9  0]
 [ 0  0  0 16]]
```

```
#element comparision
a = np.array([1,2,5,4])
b = np.array([6,2,9,4])
print(a==b)
print(a>b)
print(a<b)
```

```
[False  True False  True]
[False False False False]
[ True False  True False]
```

```
#Array wise Comparison
print(np.array_equal(a,b))
c = np.array([1,2,5,4])
print(np.array_equal(a,c))
```

```
False
True
```

Logical Operators

```
a = np.array([1,0,0,1],dtype='bool')
b = np.array([0,1,0,1],dtype='bool')
print(np.logical_or(a,b))
print(np.logical_and(a,b))
print(np.logical_not(a,b))
```

```
[ True  True False  True]
[False False False  True]
[False  True  True False]
```

Transcendental Function

```
a = np.arange(5)+1
print(np.sin(a))
```

```
[ 0.84147098  0.90929743  0.14112001 -0.7568025  -0.95892427]
```

```
print(np.log(a))
```

```
[0.         0.69314718 1.09861229 1.38629436 1.60943791]
```

```
print(np.exp(a))
```

```
[  2.71828183   7.3890561   20.08553692  54.59815003 148.4131591 ]
```

Shape Mismatch

```
a = np.array([1,2,3,4])
b = np.array([5,10])
print(a+b)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-54-1a1bcb51abe4> in <cell line: 3>()
      1 a = np.array([1,2,3,4])
      2 b = np.array([5,10])
----> 3 print(a+b)

ValueError: operands could not be broadcast together with shapes (4,) (2,)
```

Basic Reductions

```
x = np.array([1,2,3,4])
print(np.sum(x))
```

```
10
```

```
y = np.array([[1,2],[3,4]])
print(y)
print("*"*100)
print(y.T) #T = Transpose
```

```
[[1 2]
 [3 4]]
****************************************************************************************************
[[1 3]
 [2 4]]
```

```
print(y.sum(axis = 0)) #column wise sum
print(y.sum(axis = 1)) #row wise sum
```

```
[4 6]
[3 7]
```

```
print(y.max())
```

```
4
```

```
print(y.argmin()) #index of minimum element
```

```
0
```

```
print(y.argmax())#index of maximum element
```

```
3
```

Logical REductions

```
print(np.all([True ,False , False])) #logical and
```

```
False
```

```
print(np.any([True ,False , False]))#logical or
```

```
True
```

```
a = np.zeros([50,50])
print(np.any(a!=0))
```

```
False
```

Statistics

```
x = np.arange(1,10)
print(np.mean(x))#mean
print(np.median(x))#median
```

```
5.0
5.0
```

```
y = np.array([[1,2,3],[4,5,6]])
print(np.mean(y,axis=0))#column wise mean
print(np.mean(y,axis=1))#row wise mean
```

```
[2.5 3.5 4.5]
[2. 5.]
```

```
print(np.std(x))
```

```
2.581988897471611
```

1.Write a Numpy program to convert a list of numeric value into one dimensional Numpy array

```
import numpy as np
lst = [1,2,3,4,5]
arr = np.array(lst)
print(arr)
```

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

2.Wriite a numpy program to create a 3x3 matrix with values ranging from 2 to 10

```
val = np.arange(2,11)
mat = val.reshape(3,3)
print(mat)
```

```
[[ 2  3  4]
 [ 5  6  7]
 [ 8  9 10]]
```

3.Write a numpy program to sort an along the first,last axis of array

```
a = [[3,2],[1,4]]
print(np.sort(a,axis=0))
print(np.sort(a,axis=1))
```

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

4.Write a NumPy program to create a contiguous flattened array

```
a = np.array([[1,2,3],[4,5,6],[7,8,9]])
b = a.flatten()
print(b)
```

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

5.Write a NumPy program to display all dates for the month of march,2017

```
import numpy as np
import datetime
date = datetime.date(2017,3,1)
dates = np.arange(date,date + datetime
print("Display all dates for the month of march,2017:")
print(dates)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-82-9b350446ccec> in <cell line: 5>()
      3 import numpy as np
      4 import datetime
----> 5 dates = np.arange('2017-03-01','2017-03-32',dtype='datetime64[D]')
      6 print("Display all dates for the month of march,2017:")
      7 print(dates)

ValueError: Day out of range in datetime string "2017-03-32"
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```