NUMPY

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

```
[1 2 3]
```

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

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

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

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

```
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],[4,5,6]])
print(a.shape)
```

```
(2, 3)
```

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

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

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

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

```
a=np.arange(24)
print(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]
```

```
a=np.arange(24)
a.ndim
b=a.reshape(2,4,3)
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]]]
```

```
x=np.array([1,2,3,4,5],dtype=np.int8)
print(x.itemsize)
```

```
1
```

```
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]]
```

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

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

```
#common arrays
d=np.ones((3,3))
d
```

```
     array([[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.]])
```

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

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

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

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

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

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

```
#extracts the diagonal elements of matrix
np.diag(a)
```

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

```
#creates a array using random
a=np.random.rand(4)
a
```

```
array([0.33904929, 0.87896114, 0.97643454, 0.57591848])
```

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

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

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

```
complex128
```

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

```
bool
```

```
a=np.arange(10)
print(a)
print(a[5])
print(a[-1])
```

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

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

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

```
#assigning value to the index
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])#[start_value:end_value(exclusive):step]
```

```
[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
```

```
b[0]=10
print(b)
print(a) #a is also updates, since it shares the same location in memory
```

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

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

```
False
```

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

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

Using Boolean Mask

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

```
[14 15  0  7 16  5 18  3  9  2 16 10 10  5 17]
```

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

```
even_num=a[mask]
print(even_num)
```

```
[14  0 16 18  2 16 10 10]
```

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

```
[14 15  0  7 16  5 18  3  9  2 16 10 10  5 17]
```

Using Integer Array

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

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

```
b=a[[2,3,5,2,4]]
print(b)
```

```
[20 30 50 20 40]
```

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

```
[   0   10   20   30   40   50   60 -200   80 -200]
[20 30 50 20 40]
```

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 Comparison
a=np.array([1,2,3,4])
b=np.array([6,2,9,4])
print(a==b)
```

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

```
print(a>b)
```

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

```
print(a<b)
```

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

```
#array wise comparison
print(np.array_equal(a,b))
```

```
False
```

```
c=np.array([1,2,5,4])
print(np.array_equal(a,c))
```

```
False
```

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))
```

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

```
print(np.logical_and(a,b))
```

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

```
print(np.logical_not(a))
```

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

Transcendental Functions:

```
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)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-f8e2422298ab> in <cell line: 1>()
----> 1 a=np.array([1,2,3,4])
      2 b=np.array([5,10])
      3 print(a+b)

NameError: name 'np' is not defined
```

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)
```

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

```
print(y.sum(axis=0))  #Column wise sum
```

```
[4 6]
```

```
print(y.sum(axis=1))  #Row wise sum
```

```
[3 7]
```

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

```
4
```

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

```
0
```

```
print(y.argmax())  #indexof maximum Element
```

```
     3
```

Logical Reduction

```
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))
```

```
     5.0
```

```
print(np.median(x))
```

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

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

2.Write a NumPy program to create a 3x3 matrix with values ranging from 2 to 10

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

```
     [[ 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=[[1,5],[3,8]]
print(np.sort(a, axis=0))
print(np.sort(a, axis=1))
```

```
[[1 5]
 [3 8]]
[[1 5]
 [3 8]]
```

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 the dates for the month of march,2017.

```
import datetime as dt
start_date = dt.date(2017, 3, 1)
end_date = dt.date(2017, 3, 31)
date_arr = np.arange(np.datetime64(start_date), np.datetime64(end_date) + np.timedelta64(1, 'D'), dtype='datetime64[D]')
print(date_arr)
```

```
['2017-03-01' '2017-03-02' '2017-03-03' '2017-03-04' '2017-03-05'
 '2017-03-06' '2017-03-07' '2017-03-08' '2017-03-09' '2017-03-10'
 '2017-03-11' '2017-03-12' '2017-03-13' '2017-03-14' '2017-03-15'
 '2017-03-16' '2017-03-17' '2017-03-18' '2017-03-19' '2017-03-20'
 '2017-03-21' '2017-03-22' '2017-03-23' '2017-03-24' '2017-03-25'
 '2017-03-26' '2017-03-27' '2017-03-28' '2017-03-29' '2017-03-30'
 '2017-03-31']
```