##Array Creation function

```
import numpy as np

a=np.array([1,2,3])     #create an array from the list
print("Array a:",a)

Array a: [1 2 3]

b=np.arange(0,10,2)     #create an array with evenly spaced values
print("Array b:",b)      #values from 0 to 10 with the step count 2

Array b: [0 2 4 6 8]

d=np.zeros((2,3))      #creating an array filled with zeros
print("Array d:\n",d)          #2*3 matrix of zeros

Array d:
 [[0. 0. 0.]
 [0. 0. 0.]]

e=np.ones((3,2))       #creating an array filled with ones
print("Array e:\n",e)     #3*2 matrix of array of ones

Array e:
 [[1. 1.]
 [1. 1.]
 [1. 1.]]

f=np.eye(4)         #creating an identity matrix
print("Identity matrix f:\n",f)     #4*4 identity matrix

Identity matrix f:
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
```

##Array manipulatiion function

```
a1=np.array([1,2,3])    #Reshape of an array
reshaped=np.reshape(a1,(1,3))     #resahpe of an arraqy to 1*3 matix
print("Reshaped array:",reshaped)

Reshaped array: [[1 2 3]]

f1=np.array([[1,2],[3,4]])   #flatten of an array
flattened = np.ravel(f1)        #flatten to 1Darray
print("Flattened array:",flattened)

Flattened array: [1 2 3 4]
```

```
e1=np.array([[1,2],[3,4]])    #Transpose an array
transposed=np.transpose(e1)     #Transpose the array
print("Transposed array:\n",transposed)
```

```
Transposed array:
 [[1 3]
 [2 4]]
```

```
a2=np.array([1,2])        #Stack arrays vertically
b2=np.array([3,4])
stacked=np.vstack([a2,b2])      #Stack a and b vertically
print("Stacked arrays:\n",stacked)
```

```
Stacked arrays:
 [[1 2]
 [3 4]]
```

##Mathematical functions

```
g=np.array([1,2,3,4])    #Add two arrays
added=np.add(g,2)            #adding 2 to each element
print("Added 2 to g:",added)
```

```
Added 2 to g: [3 4 5 6]
```

```
squared=np.power(g,2)       #Square of each element
print("Squared g:",squared)
```

```
Squared g: [ 1  4  9 16]
```

```
sqrt_val=np.sqrt(g)      #Square root of each element
print("Square root of g:",sqrt_val)
```

```
Square root of g: [1.         1.41421356 1.73205081 2.        ]
```

```
print(a1)
print(g)
```

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

```
print(a)
print(a1)
```

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

```
a3=np.array([1,2,3])
dot_product=np.dot(a1,a)          #dot product of a and g
print("Dot product of a1 and a:",dot_product)
```

```
Dot product of a1 and a: 14
```

##Statistical functions

```
s=np.array([1,2,3,4])
mean=np.mean(s)
print("Mean of s:",mean)
```

```
Mean of s: 2.5
```

```
std_dev=np.std(s)        #Standard deviation of an array
print("Standard deviation of s:",std_dev)
```

```
Standard deviation of s: 1.118033988749895
```

```
minimum=np.min(s)      #Minimum element of an array
print("Min of s:",minimum)
```

```
Min of s: 1
```

```
maximum = np.max(s)      #Maximum element pf an array
print("Max of s:",maximum)
```

```
Max of s: 4
```

##Linear algebra functions

```
matric=np.array([[1,2],[3,4]])    #creating a matrix
```

##Random sampling functions

```
random_vals=np.random.rand(3)  #Generating random values between 0 and
1
print("Random values:",random_vals)    #Array of 3 random values
between 0 and 1
```

```
Random values: [0.00190675 0.6774193  0.91445251]
```

```
np.random.seed(0)   #ser seeed for reproducibility
random_vals=np.random.rand(3)   #generate random values between 0 and
1
print("Random values:",random_vals)         #array of 3 random values
between 0 and 1
```

```
Random values: [0.5488135  0.71518937 0.60276338]
```

```
rand_ints=np.random.randint(0,10,size=5)   #generate random integers
print("Random integers:",rand_ints)      #random integers between 0
and 10
```

```
Random integers: [3 7 9 3 5]
```

```
np.random.seed(0)       #set seed for reproducibility
rand_ints=np.random.randint(0,10,size=5)    #generate random integers
```

```
print("Random integers:",rand_ints)       #random integers between 0
and 10
```

```
Random integers: [5 0 3 3 7]
```

##Boolean & Logical functions

```
logical_test=np.array([True,False,True])     #check if all elements
are true
all_true=np.all(logical_test)                #all
print("All elements True:",all_true)
```

```
All elements True: False
```

```
logical_test=np.array([True,False,True])     #check if all elements
are true
all_true=np.all(logical_test)
print("All elements True:",all_true)
```

```
All elements True: False
```

```
logical_test=np.array([False,False,False])     #check if all elements
are true
all_true=np.all(logical_test)
print("All elements True:",all_true)
```

```
All elements True: False
```

```
any_true=np.any(logical_test)     #check if any elemets are true
print("Any elements True:",any_true)
```

```
Any elements True: False
```

##Set operations

```
set_a=np.array([1,2,3,4])      #intersection of two arrays
set_b=np.array([3,4,5,6])
intersection=np.intersect1d(set_a,set_b)
print("Intersectionof a and b:",intersection)
```

```
Intersectionof a and b: [3 4]
```

```
union=np.union1d(set_a,set_b)         #union of two arrays
print("Union of a and b:", union)
```

```
Union of a and b: [1 2 3 4 5 6]
```

##Array attribute functions

```
import numpy as np
```

```
#array attributes
a=np.array([1,2,3])
shape=a.shape     #shape of the array
size=a.size       #number of elements
dimensions=a.ndim      ##number of dimensions
dtype=a.dtype      #data type of the array
print("Shape of a:", shape)
print("Size of a:",size)
print("Number of dimensions of a:",dimensions)
print("Data type of a:", dtype)
```

```
Shape of a: (3,)
Size of a: 3
Number of dimensions of a: 1
Data type of a: int32
```

##Other functions

```
a=np.array([1,2,3])      #create a copy of an array
copied_array=np.copy(a)        #create a copy of an array a
print("Copied array:",copied_array)
```

```
Copied array: [1 2 3]
```

```
#Size in bytes of an array
array_size_in_bytes=a.nbytes       #size in bytes
print("Size of a in bytes:",array_size_in_bytes)
```

```
Size of a in bytes: 12
```

```
#Check if two arrays share memory
shared=np.shares_memory(a,copied_array)     #check if arrays share
memory
print("Do a and copied_array share memory?",shared)
```

```
Do a and copied_array share memory? False
```