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

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

```
import sys
S=range(1000)
print("Size of each element of list in bytes: ",sys.getsizeof(S))
print("Size of the whole list in bythes: ",sys.getsizeof(S)*len(S))
D=np.arange(1000)
print("Size of each element of Numpy array in bytes: ",D.itemsize)
print("Size of the whole Numpy array in bytes: ",D.size*D.itemsize)
```

```
Size of each element of list in bytes:  48
Size of the whole list in bythes:  48000
Size of each element of Numpy array in bytes:  8
Size of the whole Numpy array in bytes:  8000
```

```
#0-D arrays
arr = np.array(45)
print(arr)
```

```
45
```

```
#2D Arrays
arr=np.array([[1,2,3], [4,5,6]])
print(arr)
```

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

```
#3D Arrays
arr = np.array([[[1,2,3], [4,5,6]],  [[1,2,3], [4,5,6]]])
print(arr)
```

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

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

```
import numpy as np
zeros_arr =np.zeros((2,3))
print(zeros_arr)
print("Data type of the array: ",zeros_arr.dtype)
zeros_arr_int = np.zeros((3,3), dtype=np.int16)
print("\n array of zeros with integer elements: \n",zeros_arr_int)
```

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

```
Data type of the array:  float64

 array of zeros with integer elements:
 [[0 0 0]
 [0 0 0]
 [0 0 0]]
```

```
import numpy as np
ones_arr =np.ones(6)
print(ones_arr)
print("\nData type of the array: ",ones_arr.dtype)
```

```
[1. 1. 1. 1. 1. 1.]

Data type of the array:  float64
```

```
import numpy as np
empty_arr = np.empty((3,2))
print(empty_arr)
print("\nData type of the array: ",empty_arr.dtype)
```

```
[[1. 1.]
 [1. 1.]
 [1. 1.]]

Data type of the array:  float64
```

```
import numpy as np
identity_matrix = np.eye(3)
print(identity_matrix)
identity_matrix = np.eye(4)
print("\n",identity_matrix)
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]

 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
```

```
import numpy as np
arr1 =np.array([1,2,3], dtype=np.float64)
arr2 =np.array([1,2,3], dtype=np.int32)
print(arr1.dtype)
print(arr2.dtype)
```

```
float64
int32
```

```
import numpy as np
arr = np.array([1,2,3])
print("Before converting: ", arr, arr.dtype)
new_arr = arr.astype('f')
print("After converting: ", new_arr, new_arr.dtype)
```

```
Before converting:  [1 2 3] int64
After converting:  [1. 2. 3.] float32
```

```
import numpy as np
arr1 =np.arange(10)
arr2 =np.random.randn(10)
print("array 1\n",arr1, arr1.dtype)
print("array \2n",arr1, arr2.dtype)
new_arr =arr1.astype(arr2.dtype)
print("new array \n",new_arr, new_arr.dtype)
```

```
array 1
 [0 1 2 3 4 5 6 7 8 9] int64
array n [0 1 2 3 4 5 6 7 8 9] float64
new array
 [0. 1. 2. 3. 4. 5. 6. 7. 8. 9.] float64
```

```
import numpy as np
str_arr = np.array(['hello', 'world', 'numpy'])
print(str_arr)
```

```
['hello' 'world' 'numpy']
```

```
import numpy as np
a=np.array([1.,2.,3.,4.])
b=np.array([5.,6.,7.,8.])
print(a+b)
print(a-b)
print(a*b)
print(a/b)
print(a**b)
```

```
[ 6.  8. 10. 12.]
[-4. -4. -4. -4.]
[ 5. 12. 21. 32.]
[0.2        0.33333333 0.42857143 0.5       ]
[1.0000e+00 6.4000e+01 2.1870e+03 6.5536e+04]
```

```
import numpy as np
arr = np.arange(12)
newarr = arr.reshape(4,3)
print("\n",newarr)
newarr =arr.reshape((4,3),order='F')
print("\n",newarr)
```

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

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

```
import numpy as np
arr = np.array([[1,2,3,4,9,10], [5,6,7,8,11,12]])
print(arr)
newarr =arr.reshape(3,2,2)
print("\n",newarr)
newarr1 = newarr.reshape(3, -1)
print("\n reshape(3,-1) \n",newarr1)
```

```
[[ 1  2  3  4  9 10]
 [ 5  6  7  8 11 12]]

 [[[ 1  2]
  [ 3  4]]

 [[ 9 10]
  [ 5  6]]

 [[ 7  8]
  [11 12]]]

 reshape(3,-1)
 [[ 1  2  3  4]
 [ 9 10  5  6]
 [ 7  8 11 12]]
```

```
#FLATTENING OR REVALING:
import numpy as np
original_array= np.array([[10,20], [30,40]])
print("original_array:\n",original_array)
flattened_array = original_array.flatten()
flattened_array[0] =999
print("Flattened array:\n", flattened_array)
print("Original array:\n", original_array)
reveled_array =original_array.ravel()
raveled_array =888
print("Ravelled array:\n",raveled_array)
print("Original arra:\n", original_array)
```

```
original_array:
 [[10 20]
```

```
 [30 40]]
Flattened array:
 [999  20  30  40]
Original array:
 [[10 20]
 [30 40]]
Ravelled array:
 888
Original arra:
 [[10 20]
 [30 40]]
```

```
import numpy as np
arr1 = np.array([[1,2,], [3,4]])
arr2 = np.array([[2,1], [3,5]])
greater_than =arr1>arr2
less_than =arr1<arr2
equal_to =arr1==arr2
print("Greater than:\n", greater_than)
print("\nLess than:\n", less_than)
print("\nEqual to:\n", equal_to)
arr3=np.array([1,2,3])
arr4=np.array([1,2,3])
arr5=np.array([1,2,4])
result1=np.array_equal(arr3,arr4)
result2=np.array_equal(arr3,arr5)
print("Array-wise comparison 1:", result1)
print("Array-wise comparison 2:",result2)
```

```
Greater than:
 [[False  True]
 [False False]]

Less than:
 [[ True False]
 [False  True]]

Equal to:
 [[False False]
 [ True False]]
Array-wise comparison 1: True
Array-wise comparison 2: False
```

```
import numpy as np
x=np.arange(10)
print("x=",x)
print("\n x[:5]=",x[:5])
print("\n x[5:]=",x[5:])
print("\n x[4:7]=",x[4:7])
print("\n x[::2]=",x[::2])
```

```
print("\n x[1::2]=",x[1::2])
print("\n x[::-1]=",x[::-1])
print("\n x[5::-2]=",x[5::-2])
```

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

 x[:5]= [0 1 2 3 4]

 x[5:]= [5 6 7 8 9]

 x[4:7]= [4 5 6]

 x[::2]= [0 2 4 6 8]

 x[1::2]= [1 3 5 7 9]

 x[::-1]= [9 8 7 6 5 4 3 2 1 0]

 x[5::-2]= [5 3 1]
```

```
import numpy as np
x= np.arange(10)
print(x)
x[5:] = 2
print("\n After broadcasting1",x)
x[::2]=100
print("\n After broadcasting2",x)
```

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

 After broadcasting1 [0 1 2 3 4 2 2 2 2 2]

 After broadcasting2 [100   1 100   3 100   2 100   2 100   2]
```

```
x2 = np.array([[12,5,2,4], [7,6,8,8], [1,6,7,7]])
print(x2[0])
print(x2[2,0])
print(x2[1][2])
print(x2[2,-1])
print(x2[0])
```

```
[12  5  2  4]
1
8
7
[12  5  2  4]
```

```
#slicing n-d array
import numpy as np
x2=np.array([[12,5,2,4], [7,6,8,8], [1,6,7,7]])
```

```
print("x2=",x2)
print("\n x2[:2, :3]=\n",x2[:2, :3])
print("\n x2[:3, ::2]=\n",x2[:3, ::2])
print("\n x2[::-1, ::-1]=\n",x2[::-1, ::-1])
print("\n x2[::-1]=\n",x2[::-1])
```

```
x2= [[12  5  2  4]
 [ 7  6  8  8]
 [ 1  6  7  7]]

 x2[:2, :3]=
 [[12  5  2]
 [ 7  6  8]]

 x2[:3, ::2]=
 [[12  2]
 [ 7  8]
 [ 1  7]]

 x2[::-1, ::-1]=
 [[ 7  7  6  1]
 [ 8  8  6  7]
 [ 4  2  5 12]]

 x2[::-1]=
 [[ 1  6  7  7]
 [ 7  6  8  8]
 [12  5  2  4]]
```

```
import numpy as np
a= np.arange(12).reshape(3,4)
print("a= \n",a)
b=a>4
print("b= \n",a[b])
a[b]=1
print("a=\n",a)
```

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

```
 [4 1 1 1]
 [1 1 1 1]]
```

```
import numpy as np
names =np.array(['Bob','Joe','Will','Bob','Will','Joe','Joe'])
data= np.arange(28).reshape(7,4)
print(names)
print("data=\n", data)
print("names bool == \n",data[names =='B0b'])
print("names == 'Bob'\n",data[names =='B0b'])
print("names == 'Bob',2: \n",data[names =='B0b',2:])
print("names == 'Bob',3\n", data[names =='B0b',3])
```

```
['Bob' 'Joe' 'Will' 'Bob' 'Will' 'Joe' 'Joe']
data=
 [[ 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]]
names bool ==
 []
names == 'Bob'
 []
names == 'Bob',2:
 []
names == 'Bob',3
 []
```

```
import numpy as np
arr=np.arange(12).reshape((3,4))
print(arr)
print("Transpose using .T")
print(arr.T)
print("Transpose using the transpose( function")
print(np.transpose(arr))
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
Transpose using .T
[[ 0  4  8]
 [ 1  5  9]
 [ 2  6 10]
 [ 3  7 11]]
Transpose using the transpose( function
[[ 0  4  8]
```

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

```
import numpy as np
arr1=np.transpose(arr,(1,0))
print(arr1)
arr2=np.transpose(arr,(0,1))
print(arr2)
```

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

```
import numpy as np
arr = np.arange(12).reshape(3,4)
print("Dot product of arr and arr2:")
print(np.dot(arr,arr.T))
```

```
Dot product of arr and arr2:
[[ 14  38  62]
 [ 38 126 214]
 [ 62 214 366]]
```

```
import numpy as np
arr= np.arange(12).reshape((3,4))
print(arr)
print("Swape axes 0 and 1:")
print(np.swapaxes (arr, 0,1))
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
Swape axes 0 and 1:
[[ 0  4  8]
 [ 1  5  9]
 [ 2  6 10]
 [ 3  7 11]]
```

```
import numpy as np
arr1= np.array([[1,2], [3,3]])
arr2=np.array([[5,6], [7,8]])
result =np.multiply(arr1,arr2)
print("Result of element - wise
print(result)
result = np.matmul(arr1,arr2)
print("Result of matrix multiplication:")
```

```
print(result)
result=np.dot(arr1,arr2)
print("Result of matrix multiplication:")
print(result)
```

```
Result of element-wise multiplicatipon:
[[ 5 12]
 [21 24]]
Result of ematrix multiplication:
[[19 22]
 [36 42]]
```