```
import numpy as np
a=np.array([[1,2,4],[5,8,7]])
print("Array created:\n",a)
```

```
Array created:
 [[1 2 4]
 [5 8 7]]
```

```
b=np.zeros((3,4))
print("Array with all zeros:\n",b)
```

```
Array with all zeros:
 [[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
```

```
c=np.full((3,3),6)
print("Array with all 6S:\n",c)
```

```
Array with all 6S:
 [[6 6 6]
 [6 6 6]
 [6 6 6]]
```

```
d=np.random.random((2,2))
print("Random array:\n",d)
```

```
Random array:
 [[0.5516059  0.21704829]
 [0.71387749 0.07496801]]
```

```
import numpy as np
e=np.arange(0,30,5)
print(e)
```

```
[ 0  5 10 15 20 25]
```

```
#Reshaping 3x4 array to 2x2x3 array
arr=np.array([[1,2,3,4],[5,2,4,2],[1,2,0,1]])
newarr=arr.reshape(4,3)
print("\nOriginal array:\n",arr)
print("Reshaped array[4,3]:\n",newarr)
```

```
Original array:
 [[1 2 3 4]
 [5 2 4 2]
 [1 2 0 1]]
Reshaped array[4,3]:
 [[1 2 3]
 [4 5 2]
```

```
 [4 2 1]
 [2 0 1]]
```

```
#flatten array
flarr=arr.flatten()
print("\nOriginal array:\n",arr)
print("\nFlattened array:\n",flarr)
```

```
Original array:
 [[1 2 3 4]
 [5 2 4 2]
 [1 2 0 1]]

Flattened array:
 [1 2 3 4 5 2 4 2 1 2 0 1]
```

```
import numpy as np
arr=np.array([[1,2,3,4],[5,2,4,2],[1,2,0,1]])
print("\nNo.of dimensions:\n",arr.ndim)
print("\nShape of array:\n",arr.shape)
print("\nSize of array:\n",arr.size)
print("\nArray type:\n",arr.dtype)
newtype=arr.astype('f')
print("\nConverted array element:\n",newtype)
print("\nConverted Array type:\n",newtype.dtype)
```

```
No.of dimensions:
 2

Shape of array:
 (3, 4)

Size of array:
 12

Array type:
 int32

Converted array element:
 [[1. 2. 3. 4.]
 [5. 2. 4. 2.]
 [1. 2. 0. 1.]]

Converted Array type:
 float32
```

```
import numpy as np
p=np.array([[1,2,3],[4,5,6],[7,8,9],[10,11,12],[13,14,15]])
print(p[3:0:-1])
```

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

In [1]:
```
import numpy as np
arr=np.array([[-1,2,0,4],[4,-0.5,6,0],[2.6,0,7,8],[3,-7,4,2.0]])
print("Original array:\n",arr)
```

```
Original array:
 [[-1.   2.   0.   4. ]
 [ 4.  -0.5  6.   0. ]
 [ 2.6  0.   7.   8. ]
 [ 3.  -7.   4.   2. ]]
```

In [2]:
```
#returns every other rows
print("\nEvery other rows:\n",arr[0:3:2])
```

```
Every other rows:
 [[-1.   2.   0.   4. ]
 [ 2.6  0.   7.   8. ]]
```

In [3]:
```
#return every other element from entire array
arr=np.array([1,2,3,4,5,6,7])
print("\nOriginal array:",arr)
print("\nReturns every other elements in the array:arr[::2]",arr[::2])
```

```
Original array: [1 2 3 4 5 6 7]

Returns every other elements in the array:arr[::2] [1 3 5 7]
```

In [4]:
```
#slicing array
arr=np.array([[-1,2,0,4],[4,-0.5,6,0],[2.6,0,7,8],[3,-7,4,2.0]])
temp=arr[:2,:3]
print("\nArray with first 2 rows and 3 columns:\n",temp)
```

```
Array with first 2 rows and 3 columns:
 [[-1.   2.   0. ]
 [ 4.  -0.5  6. ]]
```

In [5]:
```
#Integer array indexing
apple=arr[[0,1,2,3],[3,2,1,0]]
print("\nElements at indices(0,3),(1,2),(2,1),""(3,0):\n",apple)
```

```
Elements at indices(0,3),(1,2),(2,1),(3,0):
 [4. 6. 0. 3.]
```

In [6]:
```
#Boolean array indexing
cond=arr>2
#cond is a boolean array
temp=arr[cond]
print("\nElements greater than 2:\n",temp)
```

```
Elements greater than 2:
 [4.  4.  6.  2.6 7.  8.  3.  4. ]
```

In [7]:
```
#joining two arrays
import numpy as np
arr1=np.array([1,2,3])
arr2=np.array([4,5,6])
arr=np.concatenate((arr1,arr2))
```

```
print(arr)
```

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

In [9]:
```
#Horizontal join
arr3=np.hstack((arr1,arr2))
print(arr3)
```

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

In [10]:
```
#vertical join
arr4=np.vstack((arr1,arr2))
print(arr4)
```

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

In [11]:
```
#Depth join
arr5=np.dstack((arr1,arr2))
print(arr5)
```

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

In [16]:
```
#Splitting of array
arr=np.array([1,2,3,4,5,6])
arr6=np.array_split(arr,3)
print(arr)
print(arr6)
print(arr6[0])
print(arr6[1])
print(arr6[2])
```

```
[1 2 3 4 5 6]
[array([1, 2]), array([3, 4]), array([5, 6])]
[1 2]
[3 4]
[5 6]
```

In [ ]:
```
```

In [1]:
```
import numpy as np
arr=np.array([1,2,3,4,5,4,6,4])
print(arr)
x=np.where(arr==4)
print(x)
```

```
[1 2 3 4 5 4 6 4]
(array([3, 5, 7]),)
```

In [2]:
```
#indexes where the values are even
arr=np.array([1,2,3,4,5,6,7,8])
x=np.where(arr%2==0)
print(arr)
print(x)
```

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

In [3]:
```
#sorting the array
arr=np.array([3,2,0,1])
print("\nOriginal array:",arr)
print("\nSorted array:",np.sort(arr))
arr=np.array([[3,2,4],[5,0,1]])
print("\nOriginal array:",arr)
print("\nSorted Array:",np.sort(arr))
```

```
Original array: [3 2 0 1]

Sorted array: [0 1 2 3]

Original array: [[3 2 4]
 [5 0 1]]

Sorted Array: [[2 3 4]
 [0 1 5]]
```

In [4]:
```
import numpy as np
sorted_array=np.array([1,3,5,7])
values=[2,4,6]
insert_indices=np.searchsorted(sorted_array,values)
print(sorted_array)
print(values)
print(insert_indices)
```

```
[1 3 5 7]
[2, 4, 6]
[1 2 3]
```

In [5]:
```
#Filter
arr=np.array([41,42,43,44])
x=[True,False,True,False]
newarr=arr[x]
print("\nOriginal array:",arr)
print("\nFilter index:",x)
print("\nFilter array:",newarr)
```

```
Original array: [41 42 43 44]

Filter index: [True, False, True, False]

Filter array: [41 43]
```

In [6]:
```
arr=np.array([41,42,43,44])
filter_arr=arr>42
newarr=arr[filter_arr]
print("\nOriginal array:",arr)
print("\nFilter array:condition->42:",filter_arr)
print("nNew array:",newarr)
```

```
Original array: [41 42 43 44]

Filter array:condition->42: [False False  True  True]
nNew array: [43 44]
```

In [ ]:
```
```

In [3]:

```
import numpy as np
arr1=[10,20,30,40,50]
arr2=[2,4,5,8,10]
a=np.array(arr1)
b=np.array(arr2)
print("Original arrays")
print(a)
print(b)
print("\nVector addition")
print(a+b)
print("\nVector subtraction")
print(a-b)
print("\nVector multiplication")
print(a*b)
print("\nVector Division")
print(a/b)
print("Vector dot product")
print(a.dot(b))
print("\nScalar multiplication")
sclr=5
print("scalar value=",sclr)
print("array=",a)
print("result=",a*sclr)
#Numpy.vectorize method
def my_func(x,y):
#return x-y if x>y,otherwise return x+y
    if x>y:
        return x-y
    else:
        return x+y
print("\n\nNumpy.vectorize method")
print("(Return x-y if x>y,otherwise return x+y)")
arr1=[10,4,20]
arr2=[2,3,30]
vec_func=np.vectorize(my_func)
print("array1:",arr1)
print("array2:",arr2)
print("result:",vec_func(arr1,arr2))
```

```
Original arrays
[10 20 30 40 50]
[ 2  4  5  8 10]

Vector addition
[12 24 35 48 60]

Vector subtraction
[ 8 16 25 32 40]

Vector multiplication
[ 20  80 150 320 500]

Vector Division
[5. 5. 6. 5. 5.]
Vector dot product
1070

Scalar multiplication
scalar value= 5
array= [10 20 30 40 50]
result= [ 50 100 150 200 250]


Numpy.vectorize method
(Return x-y if x>y,otherwise return x+y)
array1: [10, 4, 20]
array2: [2, 3, 30]
result: [ 8  1 50]
```

In [ ]: