```
import numpy as np

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

print(arr)
print(type(arr))
```

```
[1 2 3 4 5]
<class 'numpy.ndarray'>
```

```
obj = (1, 2, 3, 4, 5)
print(type(obj))
print(obj)
arr = np.array(obj)
print(arr)
print(type(arr))
```

```
<class 'tuple'>
(1, 2, 3, 4, 5)
[1 2 3 4 5]
<class 'numpy.ndarray'>
```

```
obj = {"CA": 8, "CB": 10}
print(type(obj))

arr = np.array(obj)
print(arr)
print(type(arr))
```

```
<class 'dict'>
{'CA': 8, 'CB': 10}
<class 'numpy.ndarray'>
```

```
numlist = [  [ [5,6,7],[4,3,6,8] ] , [[5,6,7]], [[5,6,7],[4,3],
[6,8]] ]
print(len(numlist))
print(len(numlist[0]))
print(len(numlist[1]))
print(len(numlist[2]))
print(len(numlist[2][1]))
```

```
3
2
1
3
2
```

```
arr = np.array(["Rohan", "Riya"])
print(arr)
print(type(arr))
print(arr.ndim)
```

```
['Rohan' 'Riya']
<class 'numpy.ndarray'>
1

arr = np.array(67)
print(arr)
print(type(arr))
print(arr.ndim)

67
<class 'numpy.ndarray'>
0
```

Dimensions in Arrays

A dimension in arrays is one level of array depth (nested arrays)

```
aarr = np.array(23)
print(aarr)
print("Dimension of aarr = ",aarr.ndim)

23
Dimension of aarr =  0

aaarr = np.array([1, 2, 3,4, 5, 6])

print(aaarr)
print("Dimension of aaarr = ",aaarr.ndim)

[1 2 3 4 5 6]
Dimension of aaarr =  1

barr = np.array([[1, 2, 3], [4, 5, 6]])

print(barr)
print("Dimension of barr = ",barr.ndim)

[[1 2 3]
 [4 5 6]]
Dimension of barr =  2

list = [1,2,"Rahul"]
print(list)

[1, 2, 'Rahul']
```

```
arr = np.array(456)
print(arr)
print(type(arr))
print(arr.ndim)
```

```
456
<class 'numpy.ndarray'>
0
```

```
carr = np.array([
                [ [1,2, 5] ],
                [ [1,2,9] ] ,
                [ [1,2, 7] ]
            ])

print(carr)
print("Dimension of carr = ",carr.ndim)
print(carr.shape)
```

```
[[[1 2 5]]

 [[1 2 9]]

 [[1 2 7]]]
Dimension of carr =  3
(3, 1, 3)
```

```
del list
var = (a*a for a in range(8))
print(var)
numlist = list(var)
print(type(numlist))
print(numlist)
```

```
<generator object <genexpr> at 0x10cf3e4d0>
<class 'list'>
[0, 1, 4, 9, 16, 25, 36, 49]
```

```
carr1 = np.array([
                [ [1,2,3],[4,5,6],[7,8,9] ],
                [ [1,2,3],[4,5,6],[7,8,9] ],
                [ [1,2,3],[4,5,6],[7,8,9] ],
            ])

print(carr1[0][1][2]) #1st 2-D, 2nd row, 3rd element
print(carr1[0,1,2])
carr1[0, 0, 0] = 100
carr1[1, 0:2, 0:2] = 200
carr1[2, 0, 0] = 300
print(carr1)
```

```
print("Dimension of carr1 = ",carr1.ndim)
print(carr1.shape)
```

```
6
6
[[[100   2   3]
  [  4   5   6]
  [  7   8   9]]

 [[200 200   3]
  [200 200   6]
  [  7   8   9]]

 [[300   2   3]
  [  4   5   6]
  [  7   8   9]]]
Dimension of carr1 =  3
(3, 3, 3)
```

```
print(aarr.ndim)
print(barr.ndim)
print(carr.ndim)
```

```
0
2
3
```

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

print(narr)
print('number of dimensions :', narr.ndim)

print(narr[0]) # 4-d
print(narr[0,0]) # 3-d
print(narr[0,0,0]) # 2-d
print(narr[0,0,0,0]) # 1-d
print(narr[0,0,0,0,1]) #2nd element from above 1-d array
```

```
[[[[[1 2 3 4]]]]]
number of dimensions : 5
[[[[1 2 3 4]]]]
[[[1 2 3 4]]]
[[1 2 3 4]]
[1 2 3 4]
2
```

Higher Dimensional Arrays

```
narr = np.array([1, 2, 3, 4], ndmin=5)
print(narr)
print(narr[0])
print('number of dimensions :', narr.ndim)
print(narr[0,0,0])
```

```
[[[[[1 2 3 4]]]]]
[[[[1 2 3 4]]]]
number of dimensions : 5
[[1 2 3 4]]
```

NumPy Array Indexing

Access Array Elements

```
print(barr, end = "\n\n")
print(barr[0])
```

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

[1 2 3]
```

Access 2-D Arrays

```
print(barr[0, 1])
print(barr[1, 2])
```

```
2
6
```

Access 3-D Arrays

```
print(carr1[0, 1, 2])
```

```
6
```

Negative Indexing

```
print('Last element from 2nd dim: ', barr[1, -1])
```

```
Last element from 2nd dim:  6
```

Slicing arrays

```
print(barr, end = "\n\n")
print(barr[0, 1:3])
```

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

[2 3]
```

Data Types in NumPy

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

```
int64
```

```
arr = np.array(['apple', 'banana', 'cherry'])
print(arr.dtype)
```

```
<U6
```

```
arr = np.array([1, 2, 3, 4], dtype='S')
print(arr)
print(arr.dtype)
```

```
[b'1' b'2' b'3' b'4']
|S1
```

```
arr = np.array([1, 2, 3, 4], dtype='i4')
print(arr)
print(arr.dtype)
```

```
[1 2 3 4]
int32
```

Converting Data Type on Existing Arrays

```
arr = np.array([1.1, 2.1, 3.1])
newarr = arr.astype('i')
```

```
print(newarr)
print(newarr.dtype)
```

```
[1 2 3]
int32
```

NumPy Array Copy vs View

```
arr = np.array([1, 2, 3, 4, 5])
x = arr.copy()
arr[0] = 42
print(arr)
print(x)
```

```
[42  2  3  4  5]
[1 2 3 4 5]
```

```
arr = np.array([1, 2, 3, 4, 5])
x = arr.view()
arr[0] = 42
print(arr)
print(x)
```

```
[42  2  3  4  5]
[42  2  3  4  5]
```

Check if Array Owns its Data

```
arr = np.array([1, 2, 3, 4, 5])
x = arr.copy()
y = arr.view()

print(x.base)
print(y.base)
```

```
None
[1 2 3 4 5]
```

Array Shape

```
arr = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])
print(arr.shape)
```

```
(2, 4)
```

Reshaping arrays

```
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8])
newarr = arr.reshape(2, 4)
print(newarr)
```

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

Iterate Arrays

```
arr = np.array([1, 2, 3])

for x in arr:
  print(x)
```

```
1
2
3
```

Iterating 2-D Arrays

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

for x in arr:
  print(x)
```

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

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

for x in arr:
  for y in x:
    print(y)
```

```
1
2
3
4
5
6
```

Searching Arrays

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

x = np.where(arr == 4)

print(x)
```

```
(array([3, 5, 6]),)
```

Sorting Arrays

```
arr = np.array([3, 2, 0, 1])

print(np.sort(arr))
```

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

Filtering Arrays

```
arr = np.array([41, 42, 43, 44])

x = [True, False, True, False]

newarr = arr[x]

print(newarr)
```

```
[41 43]
```

Random Numbers in NumPy

```
from numpy import random

x = random.randint(100)

print(x)
```

```
65
```

```
x = random.rand()

print(x)
```

```
0.9148868843393018
```

Generate Random Array

```
x = random.randint(100, size=(5))

print(x)
```

```
[72 86 34 72 26]
```

```
x = random.randint(100, size=(3, 5))

print(x)
```

```
[[33 82 30 82 21]
 [23 35 13 31  8]
 [11 46 90 96 51]]
```

Generate Random Float

```
x = random.rand(5)

print(x)
```

```
[0.5472966  0.24474207 0.07287419 0.49084068 0.43122348]
```

```
x = random.rand(3, 5)

print(x)
```

```
[[0.0029164  0.37830881 0.45890181 0.17719729 0.83977454]
 [0.69336825 0.47664748 0.91011012 0.98724752 0.31986005]
 [0.3122594  0.62127074 0.55764982 0.47516867 0.42630191]]
```

Random Choice

```
x = random.choice([3, 5, 7, 9])

print(x)
```

```
7
```

```
x = random.choice([3, 5, 7, 9], size=(3, 5))

print(x)
```

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

Create a Random Array

```
x = random.randint(100, size=(3, 5))

print(x)
```

```
[[12 17 46 55 43]
 [22  3 47 10  0]
 [71 65 43 57  0]]
```

NumPy Array Join

```
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]
```

```
arr1 = np.array([[1, 2], [3, 4]])
arr2 = np.array([[5, 6], [7, 8]])

arr = np.concatenate((arr1, arr2), axis=1)

print(arr)
```

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

```
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])

arr = np.stack((arr1, arr2), axis=1)

print(arr)
```

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

```
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])

arr = np.hstack((arr1, arr2))

print(arr)
```

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

```
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])

arr = np.vstack((arr1, arr2))

print(arr)
```

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

```
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])

arr = np.dstack((arr1, arr2))

print(arr)
```

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

NumPy Array Splitting

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

newarr = np.array_split(arr, 3)

print(newarr)
```

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

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

newarr = np.array_split(arr, 4)

print(newarr)
```

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

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

newarr = np.array_split(arr, 3)

print(newarr[0])
print(newarr[1])
print(newarr[2])
```

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

```
arr = np.array([[1, 2], [3, 4], [5, 6], [7, 8], [9, 10], [11, 12]])

newarr = np.array_split(arr, 3)

print(newarr)
```

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

```
arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12], [13,
14, 15], [16, 17, 18]])

newarr = np.array_split(arr, 3, axis=1)

print(newarr)
```

```
[array([[ 1],
       [ 4],
       [ 7],
       [10],
       [13],
       [16]]), array([[ 2],
       [ 5],
       [ 8],
       [11],
       [14],
       [17]]), array([[ 3],
       [ 6],
       [ 9],
       [12],
       [15],
       [18]])]
```

```
arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12], [13,
14, 15], [16, 17, 18]])

newarr = np.hsplit(arr, 3)

print(newarr)
```

```
[array([[ 1],
       [ 4],
       [ 7],
       [10],
       [13],
       [16]]), array([[ 2],
       [ 5],
       [ 8],
       [11],
```

```
       [14],
       [17]]), array([[ 3],
       [ 6],
       [ 9],
       [12],
       [15],
       [18]])]
```

```
arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12], [13,
14, 15], [16, 17, 18]])

newarr = np.vsplit(arr, 3)

print(newarr)
```

```
[array([[1, 2, 3],
       [4, 5, 6]]), array([[ 7,  8,  9],
       [10, 11, 12]]), array([[13, 14, 15],
       [16, 17, 18]])]
```

NumPy Array Searching

```
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8])

x = np.where(arr == 4)

print(x)
```

```
(array([3]),)
```

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

x = np.where(arr%2 == 0)

print(x)
```

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

```
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8])

x = np.where(arr%2 == 1)

print(x)
```

```
(array([0, 2, 4, 6]),)
```

```
arr = np.array([6, 7, 8, 9])

x = np.searchsorted(arr, 7)
```

```
print(x)
```

```
1
```

```
arr = np.array([6, 7, 8, 9])

x = np.searchsorted(arr, 7, side='right')

print(x)
```

```
2
```

```
arr = np.array([1, 3, 5, 7])

x = np.searchsorted(arr, [2, 4, 6])

print(x)
```

```
[1 2 3]
```

NumPy Array Sorting

```
arr = np.array([3, 2, 0, 1])

print(np.sort(arr))
```

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

```
arr = np.array(['banana', 'cherry', 'apple'])

print(np.sort(arr))
```

```
['apple' 'banana' 'cherry']
```

```
arr = np.array([True, False, True])

print(np.sort(arr))
```

```
[False  True  True]
```

NumPy Array Filtering

```
arr = np.array([41, 42, 43, 44])

x = [True, False, True, False]

newarr = arr[x]
```

```
print(newarr)
```

```
[41 43]
```

```
arr = np.array([41, 42, 43, 44])

filter_arr = []

for element in arr:
  if element > 42:
    filter_arr.append(True)
  else:
    filter_arr.append(False)

newarr = arr[filter_arr]

print(filter_arr)
print(newarr)
```

```
[False, False, True, True]
[43 44]
```

```
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])

filter_arr = arr % 2 == 0

newarr = arr[filter_arr]

print(filter_arr)
print(newarr)
```

```
[False  True False  True False  True False  True False  True]
[ 2  4  6  8 10]
```

```
arr = np.array([41, 42, 43, 44])

filter_arr = arr > 42

newarr = arr[filter_arr]

print(filter_arr)
print(newarr)
```

```
[False False  True  True]
[43 44]
```

```
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])

filter_arr = arr > 5

newarr = arr[filter_arr]
```

```
print(filter_arr)
print(newarr)
```

```
[False False False False False  True  True  True  True  True]
[ 6  7  8  9 10]
```