In [3]:

```
import numpy as np
```

1D Array Numpy

In [1]:

```
arr1d=[1,2,3,4,5]
print(arr1d)
print(type(arr1d))
```

```
[1, 2, 3, 4, 5]
<class 'list'>
```

1D numpy

In [4]:

```
arr1d=np.array([1,2,3,4,5])
print(arr1d)
print(type(arr1d))
```

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

dimension

In [27]:

```
arr1d.ndim
```

Out[27]:

```
1
```

In [11]:

```
arr1d.size
```

Out[11]:

```
5
```

In [13]:

```
arr1d.shape
```

Out[13]:

```
(5,)
```

2D numpy

In [32]:

```
arr2d=np.array([[1,2,3],[4,5,6],[7,8,9],[2,3,4]])
print(arr2d)
print(type(arr2d))
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]
 [2 3 4]]
<class 'numpy.ndarray'>
```

In [18]:

```
arr2d.ndim
```

Out[18]:

```
2
```

In [22]:

```
arr2d.size
```

Out[22]:

```
9
```

In [36]:

```
arr2d.shape
```

Out[36]:

```
(4, 3)
```

Multi-D numpy

In [48]:

```
arrmd=np.array([[[1,2,3,4],[4,5,6,7],[7,8,9,0]],
                [[9,8,7,6],[6,5,4,3],[3,2,1,0]],
               [[1,2,3,4],[4,5,6,7],[7,8,9,0]],
               [[9,8,7,6],[6,5,4,3],[3,2,1,0]]])
print(arrmd)
print(type(arrmd))
```

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

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

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

 [[9 8 7 6]
  [6 5 4 3]
  [3 2 1 0]]]
<class 'numpy.ndarray'>
```

Numpy_arr_funciton

In [69]:

```
arr_print=np.array([[[1,1,1,1],[1,1,1,1],[1,1,1,1]],
                [[1,1,1,1],[1,1,1,1],[1,1,1,1]],
               [[1,1,1,1],[1,1,1,1],[1,1,1,1]],
                [[1,1,1,1],[1,1,1,1],[1,1,1,1]],
                [[1,1,1,1],[1,1,1,1],[1,1,1,1]]])
print(max,arr_print)
```

```
<built-in function max> [[[1 1 1 1]
  [1 1 1 1]
  [1 1 1 1]]

 [[1 1 1 1]
  [1 1 1 1]
  [1 1 1 1]]

 [[1 1 1 1]
  [1 1 1 1]
  [1 1 1 1]]

 [[1 1 1 1]
  [1 1 1 1]
  [1 1 1 1]]

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

By default float

In [61]:

```
arr=np.ones((5,5))
print(arr)
arr=np.ones((5,5),dtype=int)
print(arr)
```

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

In [65]:

```
arr=np.ones((5,5), dtype=bool)
print(arr)
arr=np.zeros((5,5), dtype=int)
print(arr)
```

```
[[ True  True  True  True  True]
 [ True  True  True  True  True]
 [ True  True  True  True  True]
 [ True  True  True  True  True]
 [ True  True  True  True  True]]
[[0 0 0 0 0]
 [0 0 0 0 0]
 [0 0 0 0 0]
 [0 0 0 0 0]
 [0 0 0 0 0]]
```

Identical

In [70]:

```
e_arr=np.eye((5),dtype=int)
print(e_arr)
```

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

```
#empty function-> random vvalue,garbage
arr_emp=np.empty((5,5))
```

```
print(arr_emp)
```

Full set array

In [82]:

```
f=np.full((2,4,4),3.2)
print(f)
```

```
[[[3.2 3.2 3.2 3.2]
  [3.2 3.2 3.2 3.2]
  [3.2 3.2 3.2 3.2]
  [3.2 3.2 3.2 3.2]]

 [[3.2 3.2 3.2 3.2]
  [3.2 3.2 3.2 3.2]
  [3.2 3.2 3.2 3.2]
  [3.2 3.2 3.2 3.2]]]
```

Random

In [87]:

```
from numpy import random
r=np.random.randint(30,size=10)
print(r)
```

```
[15 28 24 26 24 22  2 12 13 21]
```

In [95]:

```
#make list
x=[22,23,45,67,68,54,32,43,22,11]
#random.choice(x)
random.choice(x,size=5)
```

Out[95]:

```
array([45, 54, 68, 11, 54])
```

In [132]:

```
#t=random.rand(4,4)
#print(t)
#fix the value

random.seed(7)
t=random.rand(4,4)
print(t)
```

```
[[0.07630829 0.77991879 0.43840923 0.72346518]
 [0.97798951 0.53849587 0.50112046 0.07205113]
 [0.26843898 0.4998825  0.67923    0.80373904]
 [0.38094113 0.06593635 0.2881456  0.90959353]]
```

Slicing and Indexing

In [161]:

```
SI=np.array([[1,2,3,4],
             [5,6,7,8],
             [9,10,11,12]])
#print(arr)
#print(type(arr))
#arr[0]
#arr[0][2]

#arr(row:col)
#SI[0:3,1:3]
#print(SI[0:2,2:4])

#SI[0:3,0:1]

SI[1:3,1:3]
```

Out[161]:

```
array([[ 6,  7],
       [10, 11]])
```

In [163]:

```
#Functions
#range -> arange

ar1d= np.arange(1,10)
print(ar1d)
```

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

In [168]:

```
ar1d= np.arange(1,10,2)
print(ar1d)
```

```
[1 3 5 7 9]
```

linspace

In [181]:

```
#generate values between x & y with equal dist.

arl=np.linspace(1,30,10,dtype=int)
print(arl)

#one to multi dimension
arl=arl.reshape(5,2)
print(arl)
```

```
[ 1  4  7 10 13 17 20 23 26 30]
[[ 1  4]
 [ 7 10]
 [13 17]
 [20 23]
 [26 30]]
```

Multi dimension to one

In [184]:

```
ar1=np.linspace(1,30,10,dtype=int)
print(ar1)

#one to multi dimension
ar2=ar1.reshape(5,2)
#multi dimension to one
print(ar2)
print(ar2.ravel())
```

```
[ 1  4  7 10 13 17 20 23 26 30]
[[ 1  4]
 [ 7 10]
 [13 17]
 [20 23]
 [26 30]]
[ 1  4  7 10 13 17 20 23 26 30]
```

Ques

In [193]:

```
arq= np.arange(1,21)
print(arq)
rshape=arq.reshape(4,5)
print(rshape)
rshape[1:3,1:4]
```

```
[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20]
[[ 1  2  3  4  5]
 [ 6  7  8  9 10]
 [11 12 13 14 15]
 [16 17 18 19 20]]
```

Out[193]:

```
array([[ 7,  8,  9],
       [12, 13, 14]])
```

In [ ]: