```
import numpy as np
```

```
np.version
```

```
<module 'numpy.version' from '/usr/local/lib/python3.7/dist-packages/numpy/version.py'>
```

```
np.zeros(10, dtype=int)
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

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

```
int64
<class 'numpy.ndarray'>
```

```
n=np.array([1,2,3,4,5])
print(n.size)
print(n.itemsize)
```

```
5
8
```

```
x=np.zeros(10, dtype=int)
x[np.array(4)]=1
x
```

```
array([0, 0, 0, 0, 1, 0, 0, 0, 0, 0])
```

```
np.arange(10,49)
```

```
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26,
       27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43,
       44, 45, 46, 47, 48])
```

```
n=np.array([1,2,3,4,5])
n[::-1]
```

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

```
n=np.random.randint(1,3,27)
n.reshape(3,3,3)
```

```
array([[[1, 2, 1],
        [2, 1, 1],
```

```
               [2, 2, 1]],

              [[2, 2, 2],
               [2, 1, 1],
               [2, 2, 1]],

              [[2, 2, 1],
               [2, 2, 2],
               [1, 2, 1]]])
```

```
np.random.randint(1,10,(10,10))
```

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

```
n=np.random.randn(30)
print(n)
np.average(n)
```

```
[ 0.60999998  1.47753475  1.24212419 -0.53021029  2.4803373   0.52944693
  1.12542536 -0.80213262  0.41471694 -1.16987641  0.13158443 -0.69054518
  0.16748574  0.08774217  0.18516063 -1.17503128 -0.03197022 -0.58025889
 -1.59485727 -0.30473542  1.2835495  -0.52937077 -0.22494666  1.42115711
 -1.40065829 -0.30283935  0.8048991   0.55340976 -1.11054099  0.90231254]
0.09896375943236384
```

```
n=np.ones((10,10), dtype=int)
n[1:-1,1:-1]=0
n
```

```
array([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
       [1, 0, 0, 0, 0, 0, 0, 0, 0, 1],
       [1, 0, 0, 0, 0, 0, 0, 0, 0, 1],
       [1, 0, 0, 0, 0, 0, 0, 0, 0, 1],
       [1, 0, 0, 0, 0, 0, 0, 0, 0, 1],
       [1, 0, 0, 0, 0, 0, 0, 0, 0, 1],
       [1, 0, 0, 0, 0, 0, 0, 0, 0, 1],
       [1, 0, 0, 0, 0, 0, 0, 0, 0, 1],
       [1, 0, 0, 0, 0, 0, 0, 0, 0, 1],
       [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]])
```

```
n=np.random.randint(1,100,(10,10))
np.pad(n,pad_width=1,mode='constant',constant_values=0)
```

```
array([[ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0, 24, 36, 22, 69,  2, 59, 81, 62, 10, 53,  0],
       [ 0, 47, 34, 87, 51, 31, 18, 73, 21, 43, 74,  0],
       [ 0,  9, 38, 28, 84, 38, 14, 49, 81, 13, 53,  0],
       [ 0, 23, 40, 60, 99, 57, 40, 39, 99, 24, 83,  0],
       [ 0, 94, 23, 30, 89, 36, 22, 49,  9, 29, 26,  0],
       [ 0, 32, 71, 60, 29,  9, 95, 21, 51, 32, 19,  0],
       [ 0,  7, 38, 63, 52,  3, 62, 71, 99, 20, 49,  0],
       [ 0, 35, 24, 47, 10, 53, 65, 56, 87, 66, 72,  0],
       [ 0, 70, 11, 42,  9, 87, 19, 91, 83, 57, 11,  0],
       [ 0, 63, 83, 77, 76, 57, 96, 53, 65, 24, 83,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0]])
```

```
x=np.array([[1,2,3,4,5,6,7],[8,9,10,11,12,13,14]])
print(x[1][5])
print(x[0])
print(x[:,3])
x[1][5]=20
print(x)
```

```
13
[1 2 3 4 5 6 7]
[ 4 11]
[[ 1  2  3  4  5  6  7]
 [ 8  9 10 11 12 20 14]]
```

```
n=np.ones(16,dtype=int)
n.reshape(2,-1)
```

```
array([[1, 1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1, 1]])
```

```
a=np.array([1,2,3])
n=np.array([])
for i in range(len(a)):
  for j in range(len(a)):
    n=np.append(n, a[i])
for i in range(len(a)):
  for j in range(len(a)):
    n=np.append(n, a[j])
n.astype(int)
```

```
array([1, 1, 1, 2, 2, 2, 3, 3, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3])
```

```
import sys
s=range(100)
print(sys.getsizeof(s))
n=np.arange(100)
print(n.itemsize)
# Therefore we can make a validation that lists takes more memory than numpy.
```

```
48
8
```

```
import time
n=100
l1,l2=range(n),range(n)
a1,a2=np.arange(n),np.arange(n)
initialTime=time.time()
resList=[i*j for i,j in zip(l1,l2)]
print(time.time()-initialTime)
initialTime=time.time()
resArr=a1*a2
print(time.time()-initialTime)
# Therefor we can make an another validation that lists takes more time than numpy.
```

```
8.678436279296875e-05
0.00011277198791503906
```

✓ 0s completed at 21:37