

```
In [2]: import numpy as np  
print(np.__version__)
```

1.24.3

```
In [10]: arr1=np.array([1,2,3,4,5],dtype= np.float32)  
arr1
```

Out[10]: array([1., 2., 3., 4., 5.], dtype=float32)

```
In [21]: #Create a 3x3 numpy array of all True's  
  
nparr2=np.full((3,3),True,dtype=bool)  
nparr2
```

Out[21]: array([[ True, True, True],  
[ True, True, True],  
[ True, True, True]])

```
In [29]: # Extract all odd numbers from arr  
arr=np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])  
arr[arr%2==1]
```

Out[29]: array([1, 3, 5, 7, 9])

```
In [4]: import numpy as np  
arr4=np.array([1,2,3])
```

```
In [6]: arr4
```

Out[6]: array([1, 2, 3])

```
In [7]: arr5=np.fromiter((a for a in range(8)),float)
```



```
In [8]: arr5
```

```
Out[8]: array([0., 1., 2., 3., 4., 5., 6., 7.])
```

```
In [9]: my_list=[1,2,3,4,5]
arr6=np.array(my_list)
```

```
In [10]: arr6
```

```
Out[10]: array([1, 2, 3, 4, 5])
```

```
In [11]: aterable=[a for a in range(8)]
```



```
In [12]: aterable
```

```
Out[12]: [0, 1, 2, 3, 4, 5, 6, 7]
```

```
In [16]: aterable3=(a for a in range(8))
```

```
In [18]: print(aterable3)
```

```
<generator object <genexpr> at 0x000001258B417440>
```

```
In [19]: iterable = (a for a in range(8))
print(np.fromiter(iterable, float))
```

```
[0. 1. 2. 3. 4. 5. 6. 7.]
```

```
In [20]: iterable
```

```
Out[20]: <generator object <genexpr> at 0x000001258B417510>
```

```
In [23]: arr5=np.array([[1, 2, 3, 4],[5, 6, 7, 8]])
arr5
```

```
Out[23]: array([[1, 2, 3, 4],
               [5, 6, 7, 8]])
```

```
In [28]: emtarr=np.empty([3,4],dtype=int)
```

```
In [27]: emtarr
```

```
Out[27]: array([[0, 0, 0, 0],  
               [0, 0, 0, 0],  
               [0, 0, 0, 0]])
```

```
In [31]: list_1 = [1, 12, 3, 14]  
list_2 = [51, 6, 72, 8]  
list_3 = [92, 101, 11, 12]  
my_arr=np.array([list_1, list_2, list_3])  
my_arr
```

```
Out[31]: array([[ 1,  2,  3,  4],  
               [ 5,  6,  7,  8],  
               [ 9, 10, 11, 12]])
```

```
In [34]: my_arr.ndim
```

```
Out[34]: 2
```

## Inspecting properties

```
In [35]: my_arr.size
```

```
Out[35]: 12
```

```
In [36]: len(my_arr)
```

```
Out[36]: 3
```

```
In [37]: my_arr.shape
```

```
Out[37]: (3, 4)
```

```
In [38]: my_arr.dtype
```

```
Out[38]: dtype('int32')
```

```
In [39]: my_arr.astype('float')
```

```
Out[39]: array([[ 1.,  2.,  3.,  4.],
                [ 5.,  6.,  7.,  8.],
                [ 9., 10., 11., 12.]])
```

```
In [40]: my_arr.tolist()
```

```
Out[40]: [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]
```

## saving and loading files

```
In [41]: np.save("file", np.arange(5))
```

```
In [42]: np.load("file.npy")
```

```
Out[42]: array([0, 1, 2, 3, 4])
```

## sorting arrays

```
In [9]: oned_arr=np.array([8,4,6,9,3,5,1,0,7,-1])
```

```
In [10]: oned_arr
```

```
Out[10]: array([ 8,  4,  6,  9,  3,  5,  1,  0,  7, -1])
```

```
In [12]: print(oned_arr.sort())
```

None

```
In [5]: sorted_one
```

```
In [6]: print(sorted_one)
```

None

```
In [50]: print(sorted_one)
```

None

```
In [13]: oned_arr
```

```
Out[13]: array([-1,  0,  1,  3,  4,  5,  6,  7,  8,  9])
```

```
In [14]: np.sort(oned_arr)
```

```
Out[14]: array([-1,  0,  1,  3,  4,  5,  6,  7,  8,  9])
```

```
In [52]: my_arr
```

```
Out[52]: array([[ 1,  2,  3,  4],
                [ 5,  6,  7,  8],
                [ 9, 10, 11, 12]])
```

```
In [ ]:
```

```
In [53]: np.sort(my_arr, axis=0)
```

```
Out[53]: array([[ 1,  2,  3,  4],
                [ 5,  6,  7,  8],
                [ 9, 10, 11, 12]])
```

```
In [54]: np.sort(my_arr, axis=1)
```

```
Out[54]: array([[ 1,  2,  3,  4],
                [ 5,  6,  7,  8],
                [ 9, 10, 11, 12]])
```

```
In [55]: list_1 = [1, 12, 3, 14]
list_2 = [51, 6, 72, 8]
list_3 = [92, 101, 11, 12]
my_arr=np.array([list_1, list_2, list_3])
my_arr
```

```
Out[55]: array([[ 1, 12,  3, 14],
 [ 51,  6, 72,  8],
 [ 92, 101, 11, 12]])
```

```
In [56]: np.sort(my_arr,axis=0)
```



```
Out[56]: array([[ 1,  6,  3,  8],
 [ 51, 12, 11, 12],
 [ 92, 101, 72, 14]])
```

```
In [57]: my_arr
```

```
Out[57]: array([[ 1, 12,  3, 14],
 [ 51,  6, 72,  8],
 [ 92, 101, 11, 12]])
```

```
In [58]: np.sort(my_arr,axis=1)
```

```
Out[58]: array([[ 1,  3, 12, 14],
 [ 6,  8, 51, 72],
 [ 11, 12, 92, 101]])
```

## Initial place holders

```
In [15]: np.arange(1,10)
```

```
Out[15]: array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
In [16]: np.linspace(1,10,3)
```

```
Out[16]: array([ 1. ,  5.5, 10. ])
```

```
In [17]: np.linspace(1,10,5)
```

```
Out[17]: array([ 1. ,  3.25,  5.5 ,  7.75, 10.  ])
```

```
In [18]: np.linspace(0,10,5)
```

```
Out[18]: array([ 0. ,  2.5,  5. ,  7.5, 10.  ])
```

```
In [20]: np.zeros((2,4),dtype=int)
```

```
Out[20]: array([[0, 0, 0, 0],  
               [0, 0, 0, 0]])
```

```
In [21]: np.ones((2,4),dtype=int)
```

```
Out[21]: array([[1, 1, 1, 1],  
               [1, 1, 1, 1]])
```

```
In [22]: np.random.rand(5)
```

```
Out[22]: array([0.73209706, 0.87472594, 0.22264876, 0.76564433, 0.85383799])
```

```
In [23]: np.random.rand(5,2)
```

```
Out[23]: array([[0.3166636 , 0.71368581],  
               [0.63808592, 0.75198728],  
               [0.49219031, 0.1735933 ],  
               [0.33478624, 0.25236605],  
               [0.64008798, 0.52966437]])
```

```
In [24]:
```

```
Out[24]: array([1, 4, 2, 4, 4, 0, 0, 4, 4, 1])
```

In [25]:

```
-----  
ValueError                                Traceback (most recent call last)  
Cell In[25], line 1  
----> 1 np.random.randint(5,2)  
  
File mtrand.pyx:763, in numpy.random.mtrand.RandomState.randint()  
  
File _bounded_integers.pyx:1338, in numpy.random._bounded_integers._rand_int32()  
  
ValueError: low >= high
```

In [26]: `np.zeros((2,4),dtype=int)`

Out[26]: `array([[0, 0, 0, 0],  
[0, 0, 0, 0]])`

In [27]: `np.zeros([2,4],dtype=int)`

Out[27]: `array([[0, 0, 0, 0],  
[0, 0, 0, 0]])`

In [28]: `np.eye(4)`

Out[28]: `array([[1., 0., 0., 0.],  
[0., 1., 0., 0.],  
[0., 0., 1., 0.],  
[0., 0., 0., 1.]])`

## Array manipulations



```
In [29]: arr
```

```
-----  
NameError                                Traceback (most recent call last)  
Cell In[29], line 1  
----> 1 arr  
  
NameError: name 'arr' is not defined
```

```
In [30]: my_arr
```

```
-----  
NameError                                Traceback (most recent call last)  
Cell In[30], line 1  
----> 1 my_arr  
  
NameError: name 'my_arr' is not defined
```

```
In [31]: list_1 = [1, 12, 3, 14]  
list_2 = [51, 6, 72, 8]  
list_3 = [92, 101, 11, 12]  
my_arr=np.array([list_1, list_2, list_3])  
my_arr
```

```
Out[31]: array([[ 1, 12,  3, 14],  
               [ 51,  6, 72,  8],  
               [ 92, 101, 11, 12]])
```

```
In [32]: my_arr
```

```
Out[32]: array([[ 1, 12,  3, 14],  
               [ 51,  6, 72,  8],  
               [ 92, 101, 11, 12]])
```

```
In [33]: new_arr=np.arange(1,6)
```

```
In [34]: new_arr
```

```
Out[34]: array([1, 2, 3, 4, 5])
```

```
In [35]: np.append(new_arr,[9])
```

```
Out[35]: array([1, 2, 3, 4, 5, 9])
```

```
In [37]: new_arr=np.append(new_arr,[10])
```

```
In [38]: new_arr
```

```
Out[38]: array([ 1,  2,  3,  4,  5,  9, 10])
```

```
In [39]: my_arr=np.arange(1,13).reshape(2,6)
```

```
In [40]: my_arr
```

```
Out[40]: array([[ 1,  2,  3,  4,  5,  6],
                [ 7,  8,  9, 10, 11, 12]])
```

```
In [42]: col=np.arange(20,26).reshape(1,6)
```

```
In [43]: my_arr_col=np.append(my_arr,col,axis=0)
```

```
In [44]: my_arr
```

```
Out[44]: array([[ 1,  2,  3,  4,  5,  6],
                [ 7,  8,  9, 10, 11, 12]])
```

```
In [45]: my_arr_col
```

```
Out[45]: array([[ 1,  2,  3,  4,  5,  6],
                [ 7,  8,  9, 10, 11, 12],
                [20, 21, 22, 23, 24, 25]])
```

```
In [46]: row=np.arange(80,83).reshape(3,1)
```

```
In [48]: my_arr_row=np.append(my_arr_col,row,axis=1)
```

```
In [49]: my_arr_row
```

```
Out[49]: array([[ 1,  2,  3,  4,  5,  6, 80],  
               [ 7,  8,  9, 10, 11, 12, 81],  
               [20, 21, 22, 23, 24, 25, 82]])
```

```
In [50]: arr = np.asarray([1, 2, 3, 4])
```

```
In [51]: arr
```

```
Out[51]: array([1, 2, 3, 4])
```

```
In [52]: a = np.insert(arr, 1, 9)  
         print("\nArray after insertion:", a)
```

```
Array after insertion: [1 9 2 3 4]
```

```
In [53]: a = np.delete(arr, 3)
```

```
In [54]: a
```

```
Out[54]: array([1, 2, 3])
```

```
In [55]: np.delete(a, 0)
```

```
Out[55]: array([2, 3])
```

## Reshaping array

```
In [56]: array = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9,  
                        10, 11, 12, 13, 14, 15, 16])
```

```
In [57]: array
```

```
Out[57]: array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16])
```

```
In [58]: print("Array: " + str(array))
```

```
Array: [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16]
```

```
In [59]: reshaped1 = array.reshape((4, array.size//4))
```

```
In [60]: reshaped1
```

```
Out[60]: array([[ 1,  2,  3,  4],  
               [ 5,  6,  7,  8],  
               [ 9, 10, 11, 12],  
               [13, 14, 15, 16]])
```

```
In [61]: reshaped2 = np.reshape(array, (2, 8))
```

```
In [62]: reshaped2
```

```
Out[62]: array([[ 1,  2,  3,  4,  5,  6,  7,  8],  
               [ 9, 10, 11, 12, 13, 14, 15, 16]])
```

```
In [64]: arr = np.array([1, 2, 3, 4, 5, 6])
```

```
In [66]: arr.resize(3, 4)  
print(arr)
```

```
[[1 2 3 4]  
 [5 6 0 0]  
 [0 0 0 0]]
```

```
In [68]: list_1 = [1, 2, 3, 4]
list_2 = [5, 6, 7, 8]
arr = np.array([list_1, list_2])
```

```
In [69]: arr
```

```
Out[69]: array([[1, 2, 3, 4],
               [5, 6, 7, 8]])
```

```
In [71]: print(arr.flatten())

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

```
In [72]: gfg = np.array([[1, 2],
                        [4, 5],
                        [7, 8]])
```

```
In [73]: gfg
```

```
Out[73]: array([[1, 2],
               [4, 5],
               [7, 8]])
```

```
In [74]: gfg.transpose(1, 0)
```

```
Out[74]: array([[1, 4, 7],
               [2, 5, 8]])
```

```
In [90]: newarr1=np.ones((3,4))
```

```
In [86]: newarr1
```

```
Out[86]: array([[1., 1., 1., 1.],
               [1., 1., 1., 1.],
               [1., 1., 1., 1.]])
```

```
In [91]: newarr2=np.zeros((3,2))
```

```
In [92]: np.concatenate((newarr1, newarr2), axis = 1)
```

```
Out[92]: array([[1., 1., 1., 1., 0., 0.],
               [1., 1., 1., 1., 0., 0.],
               [1., 1., 1., 1., 0., 0.]])
```

```
In [93]: np.split(newarr1, 3, 1)
```

```
-----
ValueError                                Traceback (most recent call last)
Cell In[93], line 1
----> 1 np.split(newarr1, 3, 1)

File <__array_function__ internals>:200, in split(*args, **kwargs)

File ~\anaconda3\Lib\site-packages\numpy\lib\shape_base.py:872, in split(ary, indices_or_sections, axis)
    870     N = ary.shape[axis]
    871     if N % sections:
--> 872         raise ValueError(
    873             'array split does not result in an equal division') from None
    874 return array_split(ary, indices_or_sections, axis)

ValueError: array split does not result in an equal division
```

```
In [105]: my_arr=np.arange(0,9)
```

```
my_arr
```

```
In [106]: print(my_arr)
```

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

```
In [112]: x = np.arange(16.0).reshape(4, 4)
```

```
In [113]: x
```

```
Out[113]: array([[ 0.,  1.,  2.,  3.],
                 [ 4.,  5.,  6.,  7.],
                 [ 8.,  9., 10., 11.],
                 [12., 13., 14., 15.]])
```

```
In [116]: np.vsplit(x, 1)
```

```
Out[116]: [array([[ 0.,  1.,  2.,  3.],
                 [ 4.,  5.,  6.,  7.],
                 [ 8.,  9., 10., 11.],
                 [12., 13., 14., 15.]])]
```

```
In [118]: a_new = np.array([[1 ,2 ],[3 ,4 ],[5 ,6 ]])
          print(a_new[[0 ,1 ,2 ],[0 ,0 ,1]])
```

```
[1 3 6]
```

```
In [121]: a = np.array([10, 40, 80, 50, 100])
          print(a[a>50])
```

```
[ 80 100]
```

```
In [122]: import numpy as np
```

```
In [124]: arr1=np.arange(1,9)
          arr1
```

```
Out[124]: array([1, 2, 3, 4, 5, 6, 7, 8])
```

```
In [125]: arr2 = np.asarray([1, 2, 3, 4])
```

```
In [127]: arr2
```

```
Out[127]: array([1, 2, 3, 4])
```

```
In [128]: arr2*2
```

```
Out[128]: array([2, 4, 6, 8])
```

```
In [129]: arr2
```

```
Out[129]: array([1, 2, 3, 4])
```

```
In [130]: arr1
```

```
Out[130]: array([1, 2, 3, 4, 5, 6, 7, 8])
```

```
In [131]: arr1*2
```

```
Out[131]: array([ 2,  4,  6,  8, 10, 12, 14, 16])
```

```
In [132]: arr1
```

```
Out[132]: array([1, 2, 3, 4, 5, 6, 7, 8])
```

```
In [136]: arr1.T
```

```
Out[136]: array([1, 2, 3, 4, 5, 6, 7, 8])
```

```
In [137]: arr3=np.arange(1,10).reshape(3,3)
```

```
In [138]: arr3
```

```
Out[138]: array([[1, 2, 3],
                 [4, 5, 6],
                 [7, 8, 9]])
```

```
In [139]: arr3.T
```

```
Out[139]: array([[1, 4, 7],
                 [2, 5, 8],
                 [3, 6, 9]])
```



```
In [140]: arr3
```

```
Out[140]: array([[1, 2, 3],  
                [4, 5, 6],  
                [7, 8, 9]])
```

```
In [161]: arr3=np.asarray([[ -1, 6],[9, 7]])  
arr3
```

```
Out[161]: array([[ -1,  6],  
                [ 9,  7]])
```

```
In [144]: arr3.T
```

```
Out[144]: array([[5, 9],  
                [6, 7]])
```

```
In [145]: arr3
```

```
Out[145]: array([[5, 6],  
                [9, 7]])
```

```
In [146]: arr3.mean
```

```
Out[146]: <function ndarray.mean>
```

```
In [162]: np.sqrt(arr3)
```

```
C:\Users\mrchi\AppData\Local\Temp\ipykernel_9452\2692725025.py:1: RuntimeWarning: invalid value encountered in sqrt  
  np.sqrt(arr3)
```

```
Out[162]: array([[      nan,  2.44948974],  
                [3.      ,  2.64575131]])
```

```
In [159]: np.std(arr3,dtype=np.float64)
```

```
Out[159]: 1.479019945774904
```

In [ ]: