

What is numpy?

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

At the core of the NumPy package, is the ndarray object. This encapsulates n-dimensional arrays of homogeneous data types

Numpy Arrays Vs Python Sequences

- NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of an ndarray will create a new array and delete the original.
- The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory.
- NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python's built-in sequences.
- A growing plethora of scientific and mathematical Python-based packages are using NumPy arrays; though these typically support Python-sequence input, they convert such input to NumPy arrays prior to processing, and they often output NumPy arrays.

✓ Creating Numpy Arrays

```
# np.array
import numpy as np
```

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

```
[1 2 3]
```

```
# 2D and 3D
b = np.array([[1,2,3],[4,5,6]])
print(b)
```

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

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

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

```
[[5 6]
 [7 8]]]
```

```
# dtype
np.array([1,2,3],dtype=float)
```

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

```
# np.arange
np.arange(1,11,2)
```

```
array([1, 3, 5, 7, 9])
```

```
# with reshape
np.arange(16).reshape(2,2,2,2)
```

```
array([[[[ 0,  1],
 [ 2,  3]],
```

```
[[ 4,  5],
 [ 6,  7]]],
```

```
[[[ 8,  9],
 [10, 11]],
```

```
[[12, 13],
 [14, 15]]]])
```

```
# np.ones and np.zeros
np.ones((3,4))
```

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

```
np.zeros((3,4))
```

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

```
[0., 0., 0., 0.]])
```

```
# np.random
```

```
np.random.random((3,4))
```

```
array([[0.85721156, 0.31248316, 0.08807828, 0.35230774],
       [0.96813914, 0.44681708, 0.56396358, 0.53020065],
       [0.03277116, 0.28543753, 0.09521082, 0.87967034]])
```

```
# np.linspace
```

```
np.linspace(-10,10,10,dtype=int)
```

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

```
# np.identity
```

```
np.identity(3)
```

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

✓ Array Attributes

```
a1 = np.arange(10,dtype=np.int32)
```

```
a2 = np.arange(12,dtype=float).reshape(3,4)
```

```
a3 = np.arange(8).reshape(2,2,2)
```

```
a3
```

```
array([[[0, 1],
        [2, 3]],
```

```
       [[4, 5],
        [6, 7]])])
```

```
# ndim
```

```
a3.ndim
```

```
3
```

```
# shape
```

```
print(a3.shape)
```

```
a3
```

```
(2, 2, 2)
array([[[0, 1],
        [2, 3]],
```

```
[[4, 5],  
 [6, 7]])
```

```
# size  
print(a2.size)  
a2
```

```
12  
array([[ 0.,  1.,  2.,  3.],  
       [ 4.,  5.,  6.,  7.],  
       [ 8.,  9., 10., 11.]])
```

```
# itemsize  
a3.itemsize
```

```
8
```

```
# dtype  
print(a1.dtype)  
print(a2.dtype)  
print(a3.dtype)
```

```
int32  
float64  
int64
```

✓ Changing Datatype

```
# astype  
a3.astype(np.int32)  
  
array([[0, 1],  
       [2, 3]],  
  
       [[4, 5],  
       [6, 7]]], dtype=int32)
```

✓ Array Operations

```
a1 = np.arange(12).reshape(3,4)  
a2 = np.arange(12,24).reshape(3,4)
```

```
a2
```

```
array([[12, 13, 14, 15],
       [16, 17, 18, 19],
       [20, 21, 22, 23]])
```

scalar operations

arithmetic

```
a1 ** 2
```

```
array([[ 0,  1,  4,  9],
       [16, 25, 36, 49],
       [64, 81, 100, 121]])
```

relational

```
a2 == 15
```

```
array([[False, False, False,  True],
       [False, False, False, False],
       [False, False, False, False]])
```

vector operations

arithmetic

```
a1 ** a2
```

```
array([[          0,          1,          16384,
         14348907],
       [ 4294967296, 762939453125, 101559956668416,
        11398895185373143],
       [1152921504606846976, -1261475310744950487, 1864712049423024128,
        6839173302027254275]])
```

✓ Array Functions

```
a1 = np.random.random((3,3))
```

```
a1 = np.round(a1*100)
```

```
a1
```

```
array([[43., 28., 71.],
       [27., 93., 36.],
       [31., 18.,  7.]])
```

max/min/sum/prod

0 -> col and 1 -> row

```
np.prod(a1,axis=0)
```

```
array([35991., 46872., 17892.])
```

```
# mean/median/std/var
np.var(a1,axis=1)

array([317.55555556, 854.          , 96.22222222])

# trigonometric functions
np.sin(a1)

array([[ -0.83177474,  0.27090579,  0.95105465],
       [ 0.95637593, -0.94828214, -0.99177885],
       [-0.40403765, -0.75098725,  0.6569866 ]])

# dot product
a2 = np.arange(12).reshape(3,4)
a3 = np.arange(12,24).reshape(4,3)

np.dot(a2,a3)

array([[114, 120, 126],
       [378, 400, 422],
       [642, 680, 718]])

# log and exponents
np.exp(a1)

array([[4.72783947e+18, 1.44625706e+12, 6.83767123e+30],
       [5.32048241e+11, 2.45124554e+40, 4.31123155e+15],
       [2.90488497e+13, 6.56599691e+07, 1.09663316e+03]])

# round/floor/ceil

np.ceil(np.random.random((2,3))*100)

array([[48.,  4.,  6.],
       [ 3., 18., 82.]])
```

▼ Indexing and Slicing

```
a1 = np.arange(10)
a2 = np.arange(12).reshape(3,4)
a3 = np.arange(8).reshape(2,2,2)

a3

array([[[0, 1],
       [2, 3]],
```

```
[[4, 5],  
 [6, 7]])
```

a1

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

a2

```
array([[ 0,  1,  2,  3],  
       [ 4,  5,  6,  7],  
       [ 8,  9, 10, 11]])
```

a2[1,0]

4

a3

```
array([[0, 1],  
       [2, 3]],  
      [[4, 5],  
       [6, 7]])
```

a3[1,0,1]

5

a3[1,1,0]

6

a1

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

a1[2:5:2]

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

a2

```
array([[ 0,  1,  2,  3],  
       [ 4,  5,  6,  7],  
       [ 8,  9, 10, 11]])
```

```
a2[0:2,1::2]
```

```
array([[1, 3],  
       [5, 7]])
```

```
a2[:,1::2]
```

```
array([[ 1,  3],  
       [ 9, 11]])
```

```
a2[1,::3]
```

```
array([4, 7])
```

```
a2[0,:]
```

```
array([0, 1, 2, 3])
```

```
a2[:,2]
```

```
array([ 2,  6, 10])
```

```
a2[1:,1:3]
```

```
array([[ 5,  6],  
       [ 9, 10]])
```

```
a3 = np.arange(27).reshape(3,3,3)
```

```
a3
```

```
array([[[ 0,  1,  2],  
        [ 3,  4,  5],  
        [ 6,  7,  8]],  
       [[ 9, 10, 11],  
        [12, 13, 14],  
        [15, 16, 17]],  
       [[18, 19, 20],  
        [21, 22, 23],  
        [24, 25, 26]]])
```

```
a3[:,2,0::2]
```

```
array([[ 0,  2],  
       [18, 20]])
```



```
a3[2,1:,1:]
```

```
array([[22, 23],  
       [25, 26]])
```

```
a3[0,1,:]
```

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

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

▼ Iterating

```
a1
```

```
for i in a1:  
    print(i)
```

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

```
a2
```

```
array([[ 0,  1,  2,  3],  
       [ 4,  5,  6,  7],  
       [ 8,  9, 10, 11]])
```

```
for i in a2:  
    print(i)
```

```
[0 1 2 3]  
[4 5 6 7]  
[ 8  9 10 11]
```

a3

```
array([[ 0,  1,  2],  
       [ 3,  4,  5],  
       [ 6,  7,  8]],  
  
      [[ 9, 10, 11],  
       [12, 13, 14],  
       [15, 16, 17]],  
  
      [[18, 19, 20],  
       [21, 22, 23],  
       [24, 25, 26]]])
```

```
for i in a3:  
    print(i)
```

```
[[0 1 2]  
 [3 4 5]  
 [6 7 8]]  
[[ 9 10 11]  
 [12 13 14]  
 [15 16 17]]  
[[18 19 20]  
 [21 22 23]  
 [24 25 26]]
```

```
for i in np.nditer(a3):  
    print(i)
```

```
0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15
```

16
17
18
19
20
21
22
23
24
25
26

✓ Reshaping

```
# reshape
```

```
# Transpose
```

```
np.transpose(a2)
```

```
a2.T
```

```
array([[ 0,  4,  8],  
       [ 1,  5,  9],  
       [ 2,  6, 10],  
       [ 3,  7, 11]])
```

```
# ravel
```

```
a3.ravel()
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,  
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26])
```

✓ Stacking

```
# horizontal stacking
```

```
a4 = np.arange(12).reshape(3,4)
```

```
a5 = np.arange(12,24).reshape(3,4)
```

```
a5
```

```
array([[12, 13, 14, 15],  
       [16, 17, 18, 19],  
       [20, 21, 22, 23]])
```

```
np.hstack((a4,a5))
```

```
array([[ 0,  1,  2,  3, 12, 13, 14, 15],  
       [ 4,  5,  6,  7, 16, 17, 18, 19],  
       [ 8,  9, 10, 11, 20, 21, 22, 23]])
```

```
# Vertical stacking
np.vstack((a4,a5))

array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15],
       [16, 17, 18, 19],
       [20, 21, 22, 23]])
```

▼ Splitting

```
# horizontal splitting
a4
```

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

```
np.hsplit(a4,5)
```

```
-----
ValueError                                Traceback (most recent call last)
<ipython-input-227-59485ca7f23c> in <module>
----> 1 np.hsplit(a4,5)

<__array_function__ internals> in hsplit(*args, **kwargs)

----- 1 frames -----
<__array_function__ internals> in split(*args, **kwargs)

/usr/local/lib/python3.8/dist-packages/numpy/lib/shape_base.py 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
```

```
# vertical splitting
```

```
a5
```

```
array([[12, 13, 14, 15],
       [16, 17, 18, 19],
       [20, 21, 22, 23]])
```

```
np.vsplit(a5,2)
```

```
-----
ValueError                                Traceback (most recent call last)
<ipython-input-230-5b73f701499e> in <module>
----> 1 np.vsplit(a5,2)

<__array_function__ internals> in vsplit(*args, **kwargs)

_____ 1 frames _____
<__array_function__ internals> in split(*args, **kwargs)

/usr/local/lib/python3.8/dist-packages/numpy/lib/shape_base.py 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
```

Start coding or [generate](#) with AI.