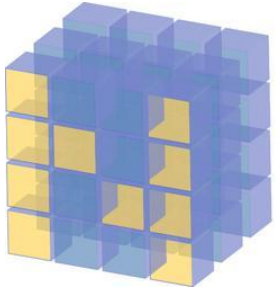
5.2 创建NumPy数组



NumPy ——Numeric Python

- 提供了**多维数组**、**矩阵**的常用操作和一些高效的**科学计算函数**。
- 底层运算通过C语言实现，处理**速度快**、**效率高**，适用于**大规模多维数组**。
- 可以直接完成**数组**和**矩阵**运算，**无需循环**。

安装NumPy库

- Anaconda：在Anaconda中，已经被安装了NumPy
- pip安装

```
pip install numpy
```

导入NumPy库

```
import numpy as np
```

在调用Numpy中的函数时，一定要加上前缀**np**

```
from numpy import *
```

在调用Numpy中的函数时，可以不加前缀

创建数组

array([列表]/(元组))

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

>>> a
array([0, 1, 2,3])

>>>print(a)
[0,1,2,3]

>>>type(a)
<class 'numpy.ndarray '>
```

输出指定的数组元素

```
>>>a[0]
0
>>>print(a[1],a[2],a[3])
1  2  3

>>>a[ 0:3 ]
array([ 0, 1, 2 ])
>>>a[ :3]
array([ 0, 1, 2 ])
>>>a[0: ]
array([ 0, 1, 2 ,3])
```

数组的属性

属　性	描　述
ndim	数组的维数
shape	数组的形状
size	数组元素的总个数
dtype	数组中元素的数据类型
itemsize	数组中每个元素的字节数

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

>>>a.ndim
1
>>>a.shape
(4,)
>>>a.size
4
```

二维数组

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

```
>>>b.ndim
2
>>>b.shape
(3, 4)
>>>b.size
12
```

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

```
>>>b[0].ndim
1
>>>b[0].shape
(4, )
>>>b[0].size
4
```

二维数组

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

```
>>>b[0][0]
0
>>>b[0][1]
1
```

```
>>>b[0,0]
0
>>>b[0,1]
1
```

三维数组

```
>>>t= np.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]]])
>>> t
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]]])
```

t[0]

t[1]

```
>>>t.ndim
3
>>>t.shape
(2, 3, 4)
>>>t.size
24
```

```
>>>t[0].ndim
2
>>>t[0].shape
(3,4)
>>>t[0].size
12
```

三维数组

```
>>>t= np.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]]])
>>> t
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]]])
```

```
>>>t[0][0]
array([0, 1, 2, 3])
>>>t[0][0].ndim
1
```

```
>>>t[0][0].shape
(4,)
>>>t[0][0].size
4
```

```
>>>t[0][0][0]
0
>>>t[0,0,0]
0
```

数组元素的数据类型

NumPy要求数组中所有元素的**数据类型**必须是**一致**的

- int8、uint8、int16、uint16、int32、uint32、int64、uint64
- float16、float32、float64、float128
- complex64、complex128、complex256
- bool、object、string_、unicode_

array([列表]/(元组), dtype=数据类型)

int64、'int64'、"np.int64"

```
>>>a= np.array( [0, 1,2, 3 ], dtype=np.int64)
>>>a
array([0,1, 2, 3], dtype=int64)
```

```
>>> a.dtype
dtype('int64')
>>>a.itemsize
8
```

每个元素所占的字节数

```
>>>c=np.array([1.2, 3.5, 5.1])
>>>c.dtype
dtype('float64')
```

在使用Python列表或元组、创建NumPy数组时，所创建的数组类型，**由原来的元素类型推导而来**。

创建特殊的数组

函　数	功能描述
np.arange()	创建数字序列数组
np.ones()	创建全1数组
np.zeros()	创建全0数组
np.eye()	创建单位矩阵
np.linspace()	创建等差数列
np.logspace()	创建等比数列

- arange()函数：创建一个由**数字序列**构成的数组。

np.arange(起始数字, 结束数字, 步长, dtype)

前闭后开：数字序列中**不包括结束数字**
起始数字省略时，默认从0开始
步长省略时，默认为1

```
>>> np.arange(4)
array([0, 1, 2, 3])
```

```
>>>d = np.arange(0, 2, 0.3)
array([ 0. ,  0.3,  0.6,  0.9,  1.2,  1.5,  1.8])
```

- ones()函数：创建一个**元素全部为1**的数组

np.ones(shape, dtype)

```
>>> np.ones((3,2),dtype=np.int16)
array([[1, 1],
       [1, 1],
       [1, 1]], dtype=int16)
```

```
>>> np.ones((3,2))
array([[1., 1.],
       [1., 1.],
       [1., 1.]])
```

当数据类型省略时，这里仍然有**2层括号**

- zeros()函数：创建一个**元素全部为0**的数组

```
np.zeros( shape, dtype )
```

```
>>> np.zeros((2,3))
array([[0., 0., 0.],
       [0., 0., 0.]])
```

- eye()函数：创建一个**单位矩阵**。

```
np.eye( shape, dtype )
```

```
>>> np.eye(3)
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

```
>>> np.eye(2,3)
array([[1., 0., 0.],
       [0., 1., 0.]])
```

- linspace()函数：创建**等差数列**。

参数：起始数字、结束数字、元素个数、元素数据类型

```
np.linspace(start, stop, num=50,  dtype)
```

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

- logspace()函数：创建一个**等比数列**。

参数：起始指数、结束指数、元素个数、基、 、元素数据类型

```
np.logspace(start, stop, num=50, base=10,  dtype)
```

```
>>> np.logspace(1,5,5, base=2)
array([  2.,   4.,   8.,  16.,  32.])
```

- **asarray()函数：** 将**列表**或**元组**转化为数组对象。

```
import numpy as np

list1=[[1,1,1],[1,1,1],[1,1,1]]
arr1=np.array(list1)
arr2=np.asarray(list1)

list1[0][0]=3

print('list1:\n',list1)
print('arr1:\n',arr1)
print('arr2:\n',arr2)
```

运行结果：

```
list1:
[[3, 1, 1], [1, 1, 1], [1, 1, 1]]
arr1:
[[1 1 1]
 [1 1 1]
 [1 1 1]]
arr2:
[[1 1 1]
 [1 1 1]
 [1 1 1]]
```

当数据源本身是一个**ndarray对象**时，array()会**复制**出一个副本，占用新的内存，而asarray()则**不复制副本**，它直接引用**原数组**。

```
import numpy as np

arr1=np.ones((3,3))
arr2=np.array(arr1)
arr3=np.asarray(arr1)

arr1[0][0]=3

print('arr1:\n',arr1)
print('arr2:\n',arr2)
print('arr3:\n',arr3)
```

```
arr1:
[[ 3.  1.  1.]
 [ 1.  1.  1.]
 [ 1.  1.  1.]]
arr2:
[[ 1.  1.  1.]
 [ 1.  1.  1.]
 [ 1.  1.  1.]]
arr3:
[[ 3.  1.  1.]
 [ 1.  1.  1.]
 [ 1.  1.  1.]]
```