

Numpy 使用教程

一、实验介绍

1.1 实验内容

如果你使用 Python 语言进行科学计算，那么一定会接触到 Numpy。Numpy 是支持 Python 语言的数值计算扩充库，其拥有强大的高维度数组处理与矩阵运算能力。除此之外，Numpy 还内建了大量的函数，方便你快速构建数学模型。

1.2 实验知识点

- Numpy 数组 ndarray
- ndarray 数组创建方法
- ndarray 数组属性

1.3 实验环境

- python2.7
- Xfce 终端
- ipython 终端

1.4 适合人群

本课程难度为一般，属于初级级别课程，适合具有 Python 基础，并对使用 Numpy 进行科学计算感兴趣的用戶。

二、Numpy 多维数组

2.1 ndarray 介绍

在 python 内建对象中，数组有三种形式：

1. list 列表：`[1, 2, 3]`
2. Tuple 元组：`(1, 2, 3, 4, 5)`
3. Dict 字典：`{A:1, B:2}`

其中，元组与列表相似，不同之处在于元组的元素不能修改。而字典由键和值构成。

python 标准类针对数组的处理局限于 1 维，并仅提供少量的功能。

而 Numpy 最核心且最重要的一个特性就是 ndarray 多维数组对象，它区别于 python 的标准类，拥有对高维数组的处理能力，这也是数值计算过程中缺一不可的重要特性。

Numpy 中，ndarray 类具有六个参数，它们分别为：

1. shape：数组的形状。
2. dtype：数据类型。
3. buffer：对象暴露缓冲区接口。
4. offset：数组数据的偏移量。
5. strides：数据步长。
6. order：{'C', 'F'}，以行或列为主排列顺序。

下面，我们来了解创建 ndarray 的一些方法。在 numpy 中，我们主要通过以下 5 种途径创建数组，它们分别是：

1. 从 Python 数组结构列表，元组等转换。
2. 使用 np.arange、np.ones、np.zeros 等 numpy 原生方法。
3. 从存储空间读取数组。
4. 通过使用字符串或缓冲区从原始字节创建数组。

5. 使用特殊函数，如 random。

2.2 从列表或元组转换

在 numpy 中，我们使用 `numpy.array` 将列表或元组转换为 `ndarray` 数组。其方法为：

```
numpy.array(object, dtype=None, copy=True, order=None, subok=False, ndmin=0)
```

其中，参数：

- `object`：列表、元组等。
- `dtype`：数据类型。如果未给出，则类型为被保存对象所需的最小类型。
- `copy`：布尔来写，默认 `True`，表示复制对象。
- `order`：顺序。
- `subok`：布尔类型，表示子类是否被传递。
- `ndmin`：生成的数组应具有的最小维数。

下面，通过列表创建一个 `ndarray` 数组：

```
import numpy as np

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

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

      [[1, 2, 3],
       [1, 2, 3],
       [1, 2, 3]],

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



或者是列表和元组：

```
import numpy as np

np.array([(1,2),(3,4),(5,6)])
```

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



2.3 arange 方法创建

除了直接使用 array 方法创建 ndarray，在 numpy 中还有一些方法可以创建一些有规律性的多维数。首先，我们来看一看 arange()。arange() 的功能是在给定区间内创建一系列均匀间隔的值。方法如下：

```
numpy.arange(start, stop, step, dtype=None)
```

你需要先设置值所在的区间，这里为 `[开始, 停止)`，你应该能发现这是一个半开半闭区间。然后，在设置 `step` 步长用于设置值之间的间隔。最后的可选参数 `dtype` 可以设置返回 `ndarray` 的值类型。

举个例子：

```
import numpy as np

# 在区间 [3, 7) 中以 0.5 为步长新建数组
np.arange(3, 7, 0.5, dtype='float32')
```

```
In [4]: np.arange(3, 7, 0.5, dtype='float32')
Out[4]: array([ 3. ,  3.5,  4. ,  4.5,  5. ,  5.5,  6. ,  6.5], dtype=float32)
```

2.4 `linspace` 方法创建

`linspace` 方法也可以像 `arange` 方法一样，创建数值有规律的数组。`linspace` 用于在指定的区间内返回间隔均匀的值。其方法如下：

```
numpy.linspace(start, stop, num=50, endpoint=True, retstep=False, dtype=None)
```

- `start`：序列的起始值。
- `stop`：序列的结束值。
- `num`：生成的样本数。默认值为50。
- `endpoint`：布尔值，如果为真，则最后一个样本包含在序列内。
- `retstep`：布尔值，如果为真，返回间距。
- `dtype`：数组的类型。


举个例子：

```
import numpy as np

np.linspace(0, 10, 10, endpoint=True)
np.linspace(0, 10, 10, endpoint=False)
```

```
In [12]: np.linspace(0, 10, 10, endpoint=True)
Out[12]:
array([ 0., 1.11111111, 2.22222222, 3.33333333,
        4.44444444, 5.55555556, 6.66666667, 7.77777778,
        8.88888889, 10.])

In [13]: np.linspace(0, 10, 10, endpoint=False)
Out[13]: array([ 0., 1., 2., 3., 4., 5., 6., 7., 8., 9.])
```



2.5 ones 方法创建

`numpy.ones` 用于快速创建数值全部为 1 的多维数组。其方法如下：

```
numpy.ones(shape, dtype=None, order='C')
```

其中：


- `shape`：用于指定数组形状，例如 (1, 2) 或 3。
- `dtype`：数据类型。
- `order`：{'C', 'F'}，按行或列方式储存数组。

举个例子：

```
import numpy as np

np.ones((2,3))
```

```
In [5]: np.ones((2,3))
Out[5]:
array([[ 1.,  1.,  1.],
       [ 1.,  1.,  1.]])
```



2.6 zeros 方法创建

`zeros` 方法和上面的 `ones` 方法非常相似，不同的地方在于，这里全部填充为 0。
`zeros` 方法和 `ones` 是一致的。

```
numpy.zeros(shape, dtype=None, order='C')
```

其中：

- shape：用于指定数组形状，例如 (1, 2) 或 3。
- dtype：数据类型。
- order：{'C', 'F'}，按行或列方式储存数组。

举个例子：

```
import numpy as np
```

```
np.zeros((3,2))
```

```
In [7]: np.zeros((3,2))  
Out[7]:  
array([[ 0.,  0.],  
       [ 0.,  0.],  
       [ 0.,  0.]])
```



2.7 eye 方法创建

numpy.eye 用于创建一个二维数组，其特点是 k 对角线上的值为 1，其余值全部为 0。方法如下：

```
numpy.eye(N, M=None, k=0, dtype=<type 'float'>)
```

其中：

- N：输出数组的行数。
- M：输出数组的列数。
- k：对角线索引：0（默认）是指主对角线，正值是指上对角线，负值是指下对角线。

举个例子：

```
import numpy as np
```

```
np.eye(5, 4, 3)
```

```
In [9]: np.eye(5, 4, 1)
Out[9]:
array([[ 0.,  1.,  0.,  0.],
       [ 0.,  0.,  1.,  0.],
       [ 0.,  0.,  0.,  1.],
       [ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.]])
```



2.8 从已知数据创建

我们还可以从已知数据文件、函数中创建 `ndarray`。numpy 提供了下面 5 个方法：

1. `frombuffer (buffer)` : 将缓冲区转换为 1 维数组。
2. `fromfile (file, dtype, count, sep)` : 从文本或二进制文件中构建多维数组。
3. `fromfunction (function, shape)` : 通过函数返回值来创建多维数组。
4. `fromiter (iterable, dtype, count)` : 从可迭代对象创建 1 维数组。
5. `fromstring (string, dtype, count, sep)` : 从字符串中创建 1 维数组。

举个例子：

```
import numpy as np
```

```
np.fromfunction(lambda a, b: a + b, (5, 4))
```

```
In [11]: np.fromfunction(lambda a, b: a + b, (5, 4))
Out[11]:
array([[ 0.,  1.,  2.,  3.],
       [ 1.,  2.,  3.,  4.],
       [ 2.,  3.,  4.,  5.],
       [ 3.,  4.,  5.,  6.],
       [ 4.,  5.,  6.,  7.]])
```



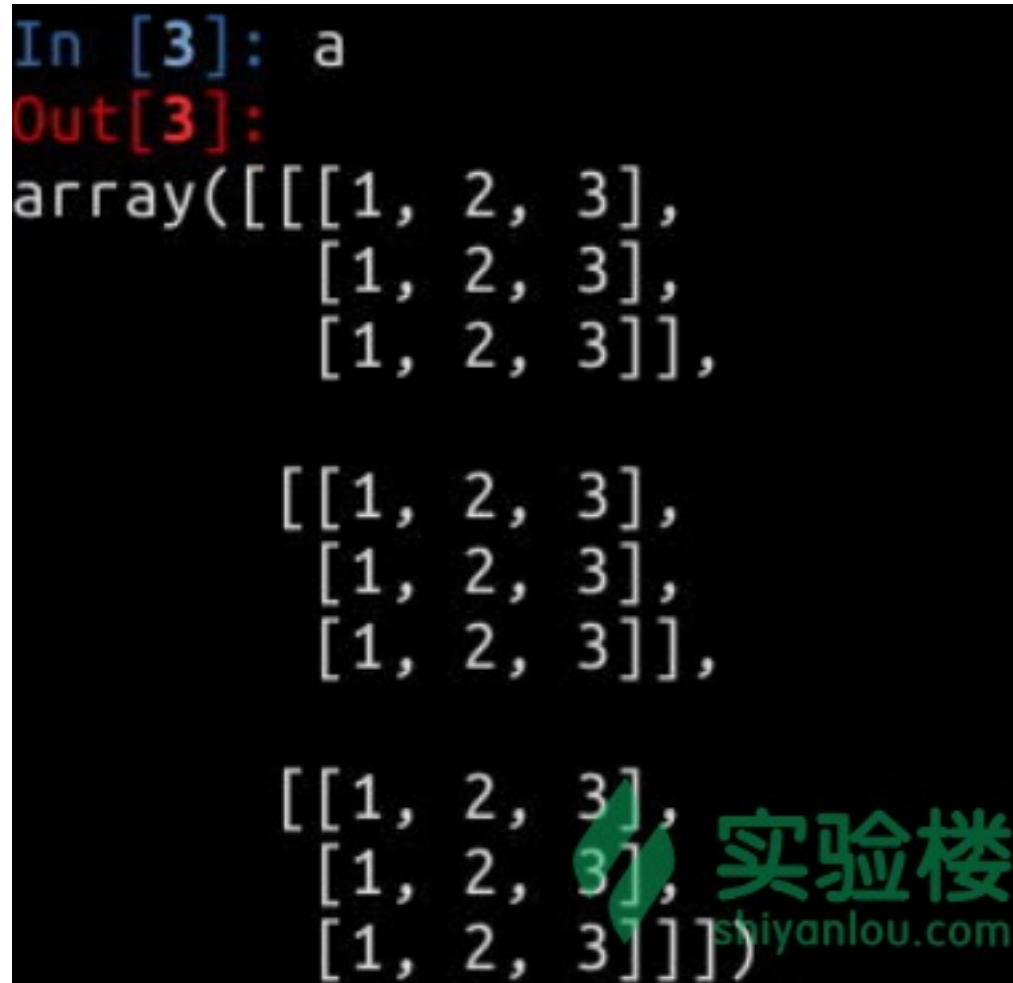
三、ndarray 数组属性

首先，我们创建一个 ndarray 数组，这里还是沿用本章节开头的例子。

```
import numpy as np

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

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



ndarray 多维数组支持下面这些属性：

3.1 ndarray.T

ndarray.T 用于数组的转置，与 .transpose() 相同。

```
import numpy as np

a.T
```

```
In [5]: a.T
Out[5]:
array([[1, 1, 1],
       [1, 1, 1],
       [1, 1, 1]],

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

      [[3, 3, 3],
       [3, 3, 3],
       [3, 3, 3]])
```



3.2 ndarray.dtype

`ndarray.dtype` 用来输出数组包含元素的数据类型。

```
import numpy as np

a.dtype
```

```
In [7]: a.dtype
Out[7]: dtype('int64')
```

3.3 ndarray.imag

`ndarray.imag` 用来输出数组包含元素的虚部。

```
import numpy as np

a.imag
```

```
In [10]: a.imag  
Out[10]:  
array([[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, 0, 0]])
```

实验楼
shiyanolou.com

3.4 ndarray.real

ndarray.real 用来输出数组包含元素的实部。

```
import numpy as np  
  
a.real
```

```
In [19]: a.real
Out[19]:
array([[ [1, 2, 3],
         [1, 2, 3],
         [1, 2, 3]],

       [ [1, 2, 3],
         [1, 2, 3],
         [1, 2, 3]],

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



3.5 ndarray.size

`ndarray.size` 用来输出数组中的总包含元素数。

```
import numpy as np

a.size
```

```
In [11]: a.size
Out[11]: 27
```

3.6 ndarray.itemsize

`ndarray.itemsize` 输出一个数组元素的字节数。

```
import numpy as np

a.itemsize
```

```
In [12]: a.itemsize  
Out[12]: 8
```

3.7 ndarray.nbytes

`ndarray.nbytes` 用来输出数组的元素总字节数。

```
import numpy as np  
  
a.nbytes
```

```
In [13]: a.nbytes  
Out[13]: 216
```

3.8 ndarray.ndim

`ndarray.ndim` 用来输出数组尺寸。

```
import numpy as np  
  
a.ndim
```

```
In [14]: a.ndim  
Out[14]: 3
```

3.9 ndarray.shape

`ndarray.shape` 用来输出数组维数组。

```
import numpy as np  
  
a.shape
```

```
In [15]: a.shape  
Out[15]: (3, 3, 3)
```

3.10 ndarray.strides

`ndarray.strides` 用来遍历数组时，输出每个维度中步进的字节数组。

```
import numpy as np  
  
a.strides
```

```
In [16]: a.strides  
Out[16]: (72, 24, 8)
```

四、实验总结

Ndarray 是 numpy 的灵魂和核心，本章节介绍了 Ndarray 的生成或转换方法，这是了解并熟练使用 numpy 的前提。

**本课程内容，由作者授权实验楼发布，未经允许，禁止转载、下载及非法传播。*

上一节：[Numpy 安装及数值类型介绍 \(/courses/912/labs/3405/document\)](/courses/912/labs/3405/document)

下一节：[Numpy 数组操作及随机抽样 \(/courses/912/labs/3408/document\)](/courses/912/labs/3408/document)