Numpy简介

NumPy（Numerical Python） 是 Python 语言的一个扩展程序库，支持大量的维度数组与矩阵运算，此外也针对数组运算提供大量的数学函数库。

NumPy 是一个运行速度非常快的数学库，主要用于数组计算，包含：

- 一个强大的N维数组对象 ndarray
- 广播功能函数
- 整合 C/C++/Fortran 代码的工具
- 线性代数、傅里叶变换、随机数生成等功能

NumPy Ndarray 对象

- NumPy 最重要的一个特点是其 N 维数组对象 ndarray，它是一系列同类型数据的集合，以 0 下标为开始进行集合中元素的索引
- ndarray 对象是用于存放同类型元素的多维数组
- ndarray 中的每个元素在内存中都有相同存储大小的区域

numpy对象创建：

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

名称	描述
object	数组或嵌套的数列
dtype	数组元素的数据类型，可选
copy	对象是否需要复制，可选
order	创建数组的样式，C为行方向，F为列方向，A为任意方向（默认）
subok	默认返回一个与基类类型一致的数组
ndmin	指定生成数组的最小维度

数据类型转换

```python
import numpy as np

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

print(a, type(a))
print(a[0])
```

```
[1 2 3] <class 'numpy.ndarray'>
1
```

```python
a = np.array([1.2, 3, 4])
print(a)
```

```
[1.2 3.  4. ]
```

```python
a = np.array([1, 2, 3], dtype='f4')
print(a)
```

```
[1. 2. 3.]
```

```python
a = np.array([1.2, 3, 4, 5.13], dtype='i4')
print(a)
```

```
[1 3 4 5]
```

拷贝

```python
li1 = [1, 2, 3]
li2 = li1

li2[0] = 8

print(li1, li2)
print(id(li1), id(li2))
```

```
[8, 2, 3] [8, 2, 3]
1439199844352 1439199844352
```

```python
a = np.array([1, 2, 3, 4, 5])
b = np.array(a, copy=True)

print(id(a), id(b))
```

```
1439199938192 1439226992656
```

最小维度

```
a = np.array([1, 2, 3, 4, 5], ndmin=2)
print(a)
```

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

subok

```
a = np.mat([1, 2, 3, 4])
print(type(a), a)

a1 = np.array(a, subok=True)
print(type(a1), a1)

a2 = np.array(a)
print(type(a2), a2)
```

```
<class 'numpy.matrix'> [[1 2 3 4]]
<class 'numpy.matrix'> [[1 2 3 4]]
<class 'numpy.ndarray'> [[1 2 3 4]]
```

NumPy 数据类型

名称	描述
bool_	布尔型数据类型（True 或者 False）
int_	默认的整数类型（类似于 C 语言中的 long，int32 或 int64）
intc	与 C 的 int 类型一样，一般是 int32 或 int 64
intp	用于索引的整数类型（类似于 C 的 ssize_t，一般情况下仍然是 int32 或 int64）
int8	字节（-128 to 127）
int16	整数（-32768 to 32767）
int32	整数（-2147483648 to 2147483647）
int64	整数（-9223372036854775808 to 9223372036854775807）
uint8	无符号整数（0 to 255）
uint16	无符号整数（0 to 65535）
uint32	无符号整数（0 to 4294967295）
uint64	无符号整数（0 to 18446744073709551615）
float_	float64 类型的简写
float16	半精度浮点数，包括：1 个符号位，5 个指数位，10 个尾数位
float32	单精度浮点数，包括：1 个符号位，8 个指数位，23 个尾数位
float64	双精度浮点数，包括：1 个符号位，11 个指数位，52 个尾数位
complex_	complex128 类型的简写，即 128 位复数
complex64	复数，表示双 32 位浮点数（实数部分和虚数部分）
complex128	复数，表示双 64 位浮点数（实数部分和虚数部分）

数据类型对象 (dtype)

数据类型对象（numpy.dtype 类的实例）用来描述与数组对应的内存区域是如何使用，它描述了数据的以下几个方面：

- 数据的类型（整数，浮点数或者 Python 对象）
- 数据的大小（例如， 整数使用多少个字节存储）
- 数据的字节顺序（小端法或大端法）
- 在结构化类型的情况下，字段的名称、每个字段的数据类型和每个字段所取的内存块的部分
- 如果数据类型是子数组，那么它的形状和数据类型是什么。

字节顺序是通过对数据类型预先设定 < 或 > 来决定的。 < 意味着小端法(最小值存储在最小的地址，即低位组放在最前面)。> 意味着大端法(最重要的字节存储在最小的地址，即高位组放在最前面)。

dtype 对象是使用以下语法构造的：

```
numpy.dtype(object, align, copy)

object - 要转换为的数据类型对象
align - 如果为 true，填充字段使其类似 C 的结构体。
copy - 复制 dtype 对象 ，如果为 false，则是对内置数据类型对象的引用
```

每个内建类型都有一个唯一定义它的字符代码

字符	对应类型
b	布尔型
i	(有符号) 整型
u	无符号整型 integer
f	浮点型
c	复数浮点型
m	timedelta（时间间隔）
M	datetime（日期时间）
O	(Python) 对象
S, a	(byte-)字符串
U	Unicode
V	原始数据 (void)

```
dt = np.dtype(np.int32)
print(dt)

输出：
int32


dt = np.dtype('i4')
print(dt)

输出：
int32


dt = np.dtype([('age', np.int8)])
print(dt)

输出：
[('age', 'i1')]
```

结构化数据类型

```
student = np.dtype([('name','S20'), ('age','i1'), ('score', 'f4')])
a = np.array([('xm', 10, 98.123456789), ('xh', 8, 99.111111111), ('xl', '9',
100)], dtype=student)
print(a)

输出:
[(b'xm', 10,  98.12346 ) (b'xh',  8,  99.111115) (b'xl',  9, 100.      )]
```