第2章 科学计算库NumPy

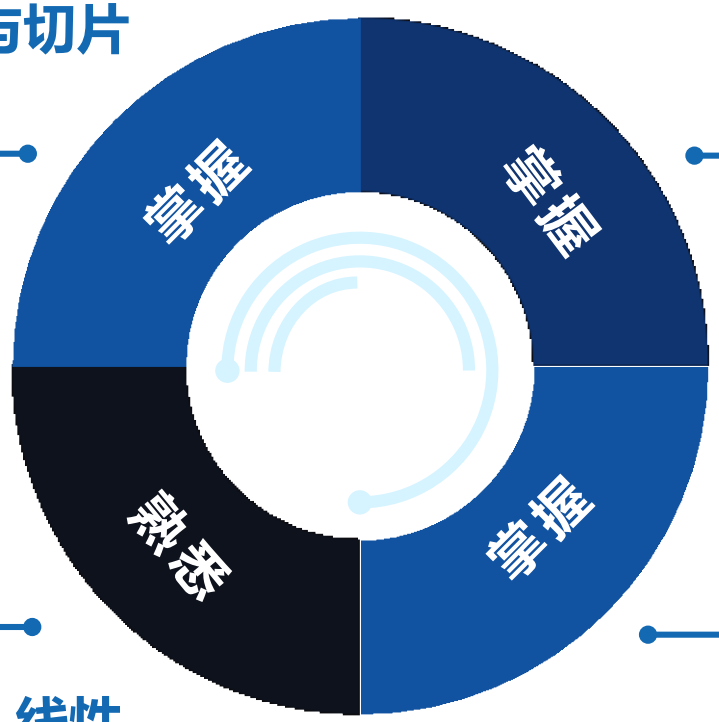
python™

- 认识和创建数组
- 数组的数据类型
- 数组运算
- 索引和切片操作
- 转置和轴对称
- 通用函数
- 数组处理数据
- 线性代数模块
- 随机数模块

学习目标

1 掌握 创建数组，数组运算，索引与切片

2 掌握 利用数组进行数据处理



3 掌握 转置和轴对称，通用函数

4 熟悉 数据类型，线性代数模块，随机数模块

目录页

目录页

过渡页

01 认识NumPy数组对象

02 创建NumPy数组

03 ndarray对象的数据类型

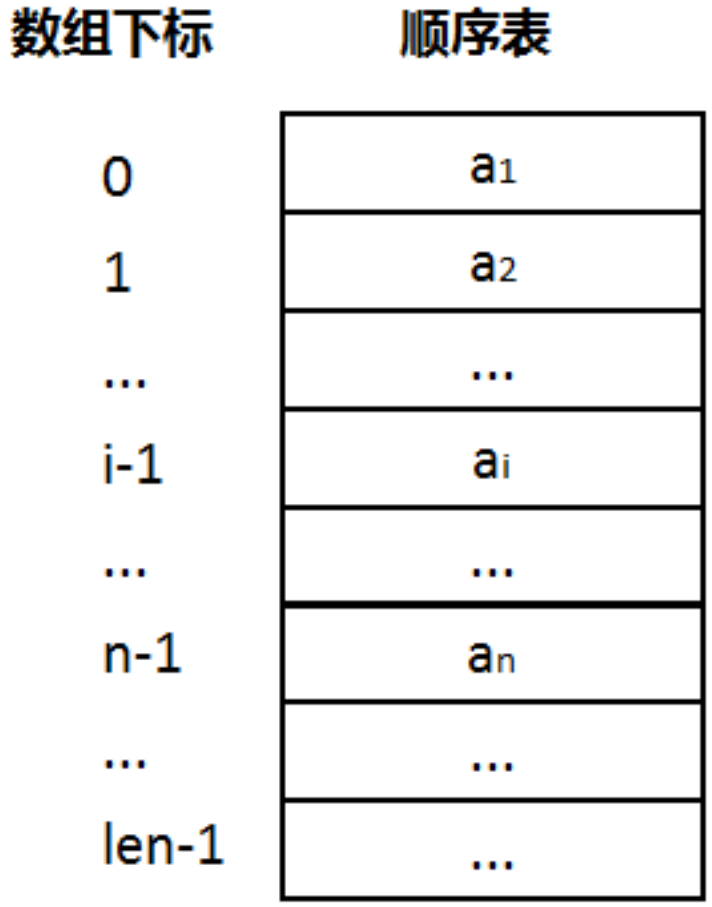
04 数组运算

05 ndarray的索引和切片

06 数组的转置和轴对称

认识NumPy数组对象

NumPy中最重要的一个特点就是其N维数组对象，即ndarray（别名array）对象，该对象可以执行一些科学计算。



认识NumPy数组对象

ndarray对象中定义了一些重要的属性。

属性	具体说明
ndarray.ndim	维度个数，也就是数组轴的个数，比如一维、二维、三维等
ndarray.shape	数组的维度。这是一个整数的元组，表示每个维度上数组的大小。例如，一个 n 行和 m 列的数组，它的 shape 属性为(n,m)
ndarray.size	数组元素的总个数，等于 shape 属性中元组元素的乘积
ndarray.dtype	描述数组中元素类型的对象，既可以使用标准的 Python 类型创建或指定，也可以使用 NumPy 特有的数据类型来指定，比如 numpy.int32、numpy.float64 等
ndarray.itemsize	数组中每个元素的字节大小。例如，元素类型为 float64 的数组有 8（64/8）个字节，这相当于 ndarray.dtype.itemsize

过渡页

创建NumPy数组

最简单的创建ndarray对象的方式是使用array()函数，在调用该函数时传入一个列表或者元组。

array([1, 2, 3])

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

```
# 创建一个一维数组
data1 = np.array([1, 2, 3])
# 创建一个二维数组
data2 = np.array([[1, 2, 3], [4, 5, 6]])
```

创建NumPy数组

通过zeros()函数创建元素值都是0的数组；通过ones()函数创建元素值都为1的数组。

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

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

```
# 创建元素值全是0的数组
np.zeros((3, 4))
# 创建元素值全是1的数组
np.ones((3, 4))
```

创建NumPy数组

通过empty()函数创建一个新的数组，该数组只分配了内存空间，它里面填充的元素都是随机的。

```
# 创建元素值全是随机数的数组
np.empty((5, 2))
```

```
array([[-2.00000000e+000, -2.00390463e+000],
       [ 2.37663529e-312,  2.56761491e-312],
       [ 8.48798317e-313,  9.33678148e-313],
       [ 8.70018275e-313,  2.12199581e-314],
       [ 0.00000000e+000,  6.95335581e-309]])
```

创建NumPy数组

通过arange()函数可以创建一个等差数组，它的功能类似于range()，只不过arange()函数返回的结果是数组，而不是列表。

```
np.arange(1, 20, 5)
```

```
array([ 1,  6, 11, 16])
```

创建NumPy数组

大家可能注意到，有些数组元素的后面会跟着一个小数点，而有些元素后面没有，比如1和1.，产生这种现象，主要是因为**元素的数据类型不同**所导致的。

过渡页

01 认识NumPy数组对象

02 创建NumPy数组

03 ndarray对象的数据类型

04 数组运算

05 ndarray的索引和切片

06 数组的转置和轴对称

查看数据类型

ndarray.dtype可以创建一个表示数据类型的对象，如果希望获取数据类型的名称，则需要访问name属性进行获取。

```
data_one = np.array([[1, 2, 3], [4, 5, 6]])
data_one.dtype.name
```

'int32'

查看数据类型

NumPy的数据类型是由一个类型名和元素位长的数字组成。

- 通过zeros()、ones()、empty()函数创建的数组，默认的数据类型为float64。
- 默认情况下，64位windows系统输出的结果为int32，64位Linux或macOS系统输出结果为int64，当然也可以通过dtype来指定数据类型的长度。

查看数据类型

NumPy中常用的数据类型如右表所示。

数据类型	含义
bool	布尔类型，值为 True 或 False
int8、uint8	有符号和无符号的 8 位整数
int16、uint16	有符号和无符号的 16 位整数
int32、uint32	有符号和无符号的 32 位整数
int64、uint64	有符号和无符号的 64 位整数
float16	半精度浮点数（16 位）
float32	半精度浮点数（32 位）
float64	半精度浮点数（64 位）
complex64	复数，分别用两个 32 位浮点数表示实部和虚部
complex128	复数，分别用两个 64 位浮点数表示实部和虚部
object	Python 对象
string_	固定长度的字符串类型
unicode	固定长度的 unicode 类型

查看数据类型

每一个NumPy内置的数据类型都有一个特征码，它能唯一标识一种数据类型。

特征码	含义
b	布尔型
u	无符号整型
c	复数类型
S，a	字节字符串
V	原始数据
i	有符号整型
f	浮点型
O	Python 对象
U	unicode 字符串

转换数据类型

ndarray对象的数据类型可以通过astype()方法进行转换。

dtype('int64')

dtype('float64')

```
data = np.array([[1, 2, 3], [4, 5, 6]])
data.dtype
# 数据类型转换为float64
float_data = data.astype(np.float64)
float_data.dtype
```

过渡页

01 认识NumPy数组对象

02 创建NumPy数组

03 ndarray对象的数据类型

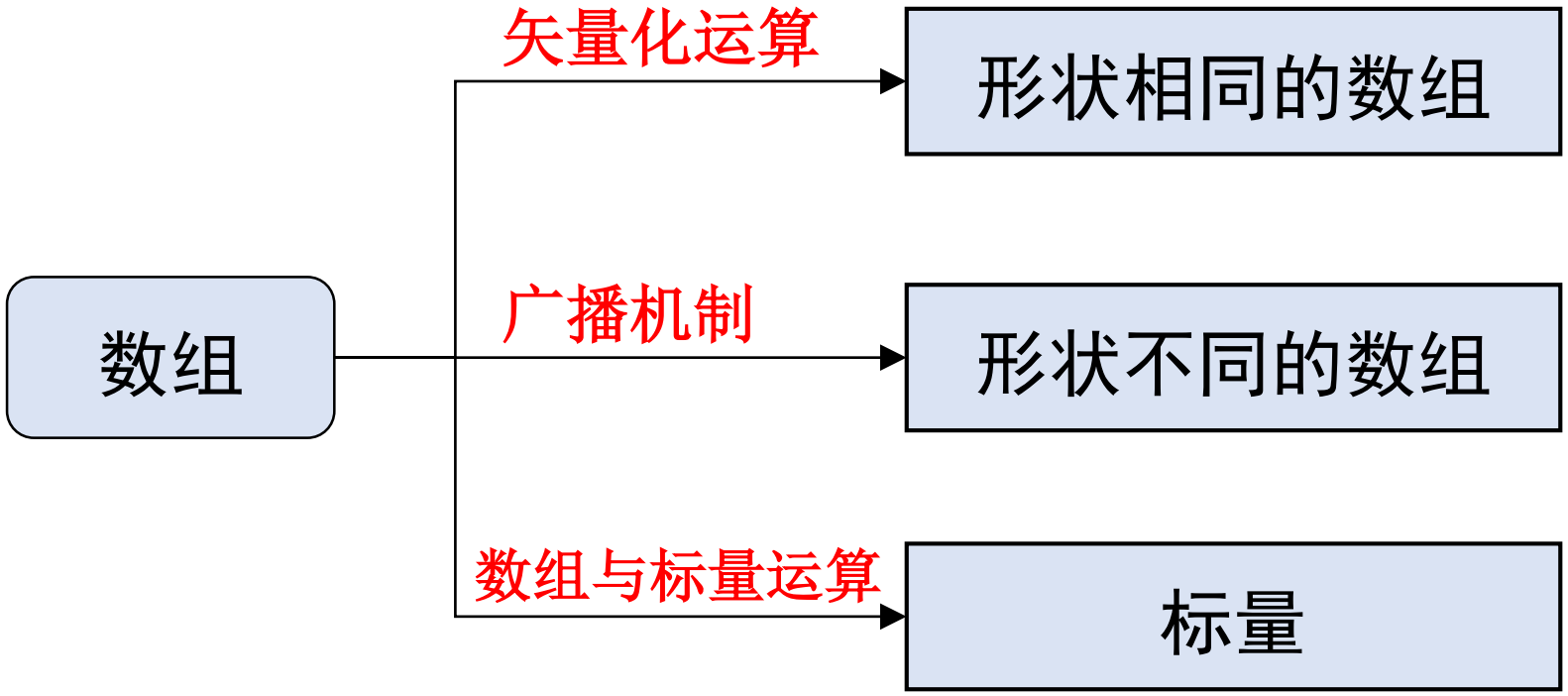
04 数组运算

05 ndarray的索引和切片

06 数组的转置和轴对称

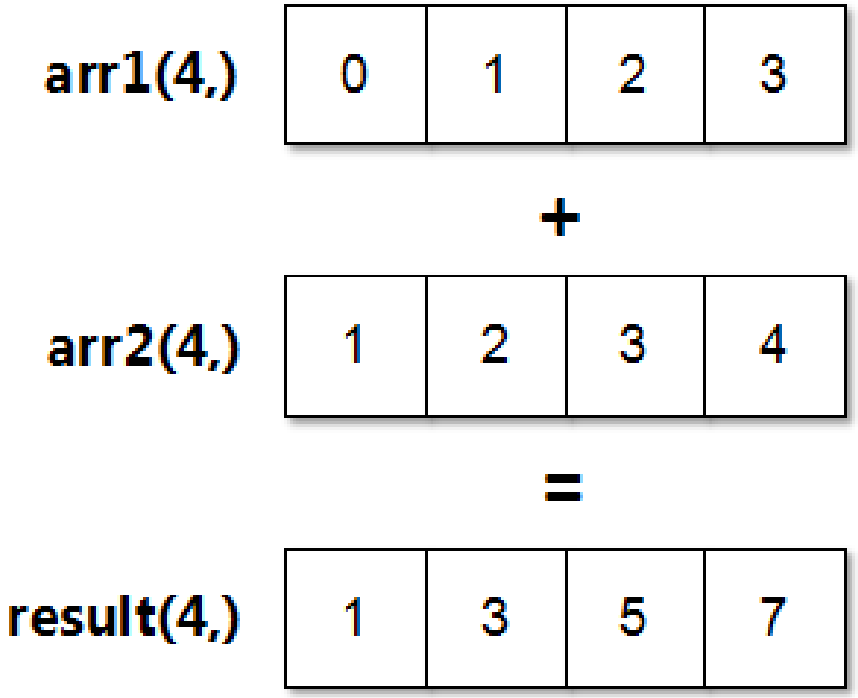
矢量化运算

数组运算可以分为以下三种：



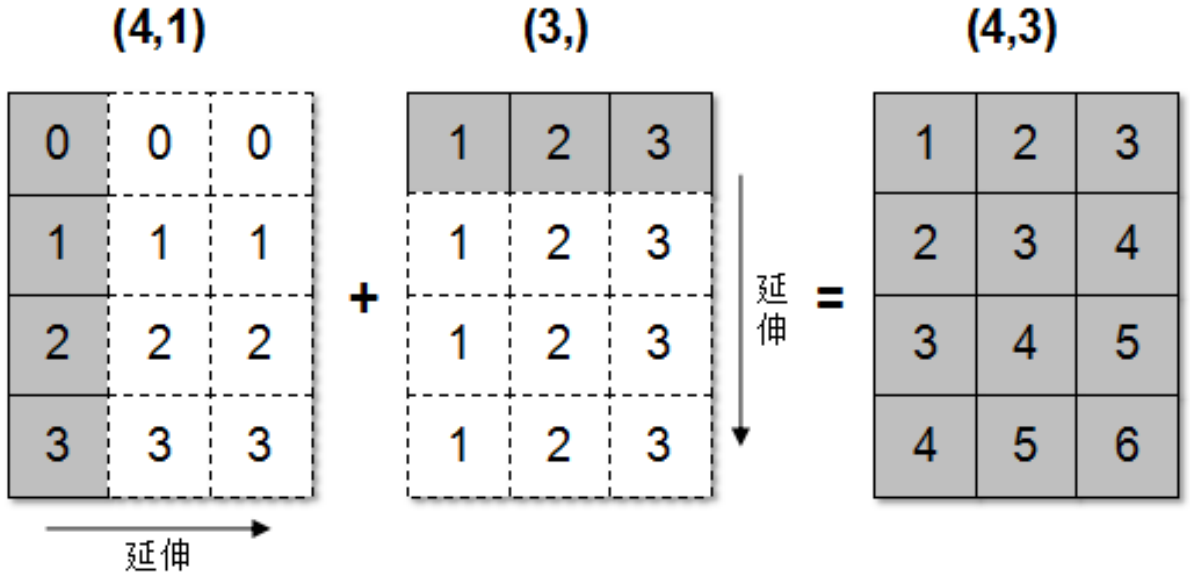
矢量化运算

形状相等的数组之间的任何算术运算都会应用到元素级，即只用于位置相同的元素之间，所得的运算结果组成一个新的数组。



数组广播

当形状不相等的数组执行算术计算的时候，就会出现广播机制，该机制会对数组进行扩展，使数组的shape属性值一样，这样就可以进行矢量化运算了。



数组广播

广播机制需要满足如下任意一个条件即可：

（1）两个数组的某一维度等长。

（2）其中一个数组为一维数组。

广播机制需要扩展维度小的数组，使得它与维度最大的数组的shape值相同，以便使用元素级函数或者运算符进行运算。

数组与标量间的运算

标量运算会产生一个与数组具有相同行和列的新矩阵，其原始矩阵的每个元素都被相加、相减、相乘或者相除。

1	2	3
4	5	6

+

10	10	10
10	10	10

=

11	12	13
14	15	16

过渡页

01 认识NumPy数组对象

02 创建NumPy数组

03 ndarray对象的数据类型

04 数组运算

05 ndarray的索引和切片

06 数组的转置和轴对称

整数索引和切片的基本使用

对于一维数组来说，从表面上来看，它使用索引和切片的方式，与Python列表的功能相差不大。

```
arr = np.arange(8)
# 获取索引为5的元素
arr[5]
# 获取索引为3~5的元素，但不包括5
arr[3:5]
```

5

array([3, 4])

整数索引和切片的基本使用

对于多维数组来说，索引和切片的使用方式与列表就大不一样了，比如二维数组的索引方式如下：

	第0列	第1列	第2列
第0行	0,0	0,1	0,2
第1行	1,0	1,1	1,2
第2行	2,0	2,1	2,2

整数索引和切片的基本使用

在二维数组中，每个索引位置上的元素不再是一个标量了，而是一个一维数组。

```
arr2d = np.array([[1, 2, 3],
                  [4, 5, 6],
                  [7, 8, 9]])
# 获取索引为1的元素
arr2d[1]
```

array([4, 5, 6])

整数索引和切片的基本使用

如果想获取二维数组的单个元素，则需要通过形如“arr[x，y]”的索引来实现，其中x表示行号，y表示列号。

```
# 获取位于第0行第1列的元素
arr2d[0, 1]
```

2

整数索引和切片的基本使用

多维数组的切片是沿着行或列的方向选取元素的，我们可以传入一个切片，也可以传入多个切片，还可以将切片与整数索引混合使用。

使用一个切片示例：

```
arr2d[:2]
```

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

整数索引和切片的基本使用

使用两个切片示例：

```
arr2d[0:2, 0:2]
```

```
array([[1, 2],
       [4, 5]])
```

切片与整数索引混合使用的示例：

```
arr2d[1, :2]
```

```
array([[1, 2],
       [4, 5]])
```

花式（数组）索引的基本使用

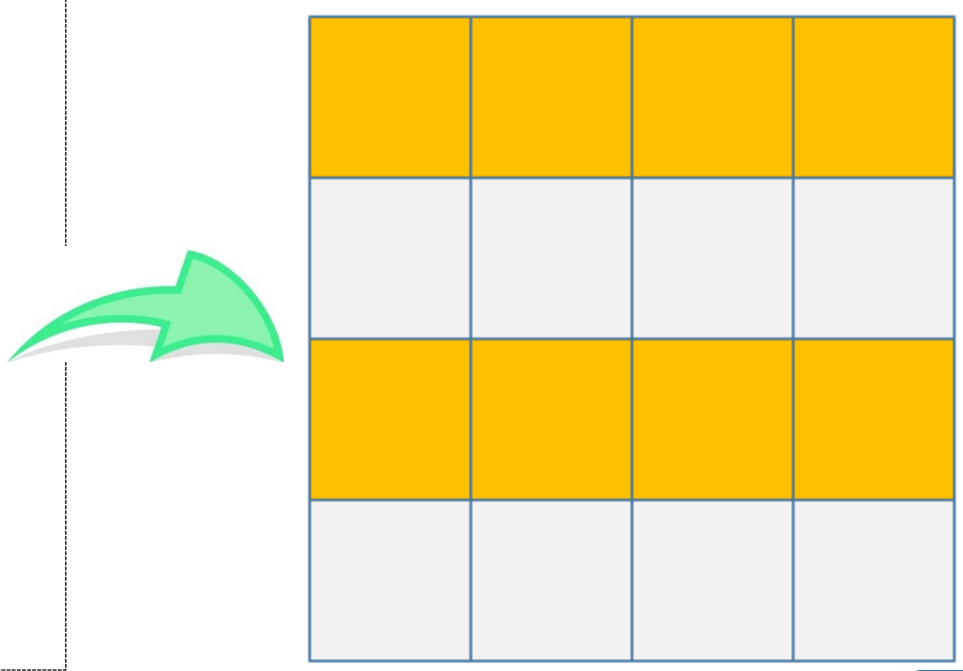
花式索引是NumPy的一个术语，是指用整数数组或列表进行索引，然后再将数组或列表中的每个元素作为下标进行取值。

- 当使用一个数组或列表作为索引时，如果使用索引要操作的对象是一维数组，则获取的结果是对应下标的元素。

花式（数组）索引的基本使用

如果要操作的对象是一个二维数组，则获取的结果就是对应下标的一行数据。

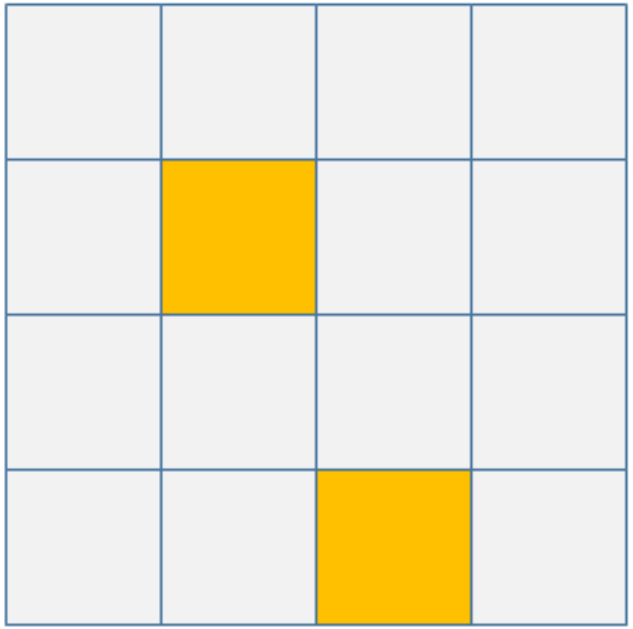
```
# 创建一个二维数组
demo_arr = np.empty((4, 4))
for i in range(4):
    demo_arr[i] = np.arange(i, i + 4)
# 获取索引为[0,2]的元素
demo_arr[[0, 2]]
```



花式（数组）索引的基本使用

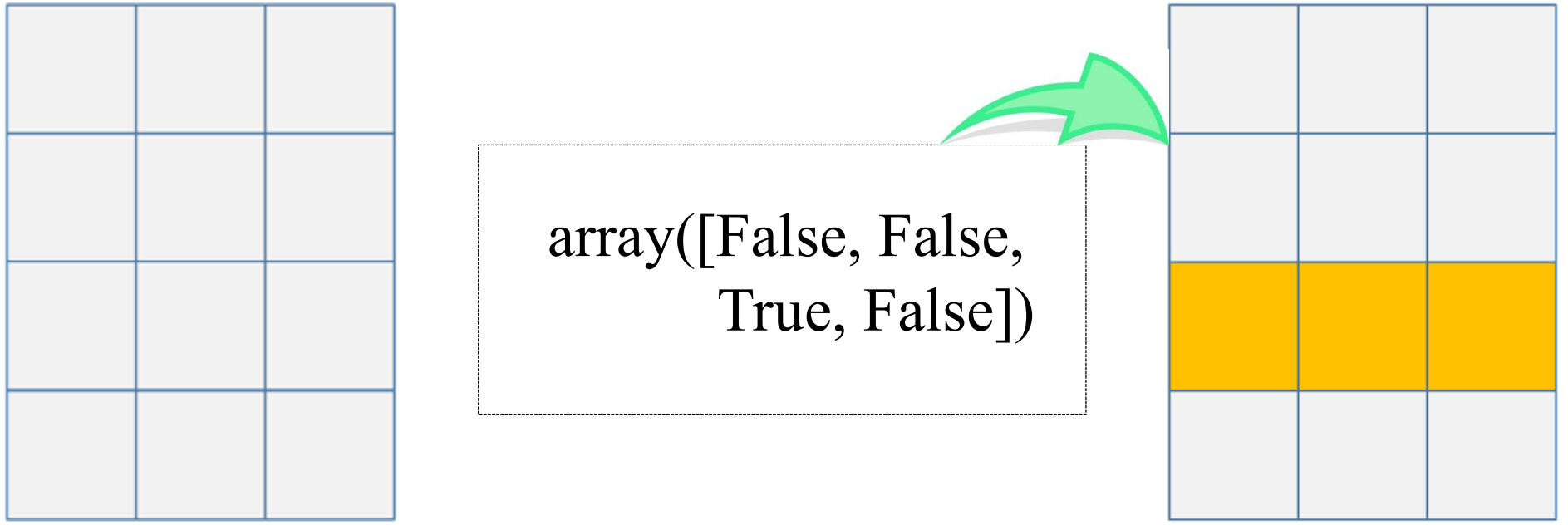
如果用两个花式索引操作数组，则会将第1个作为行索引，第2个作为列索引，以二维数组索引的方式选取其对应位置的元素。

```
# 获取索引为(1,1)和(3,2)的元素
demo_arr[[1, 3], [1, 2]]
```



布尔型索引的基本使用

布尔型索引指的是将一个布尔数组作为数组索引，返回的数据是布尔数组中True对应位置的值。



过渡页

01 认识NumPy数组对象

02 创建NumPy数组

03 ndarray对象的数据类型

04 数组运算

05 ndarray的索引和切片

06 数组的转置和轴对称

数组的转置和轴对称

数组的转置指的是将数组中的每个元素按照一定的规则进行位置变换。

NumPy提供了两种实现方式：

- T属性
- transpose()方法

数组的转置和轴对称

简单的转置可以使用T属性，它其实就是进行轴对换而已。

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

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

数组的转置和轴对称

当使用transpose()方法对数组的shape进行调换时，需要以元组的形式传入shape的编号，比如(1,0,2)。

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

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

数组的转置和轴对称

如果我们不输入任何参数，直接调用transpose()方法，则其执行的效果就是将数组进行转置，作用等价于transpose(2,1,0)。

```
In [36]: arr.transpose()
Out[36]:
array([[[ 0,  8],
        [ 4, 12]],

       [[ 1,  9],
        [ 5, 13]],

       [[ 2, 10],
        [ 6, 14]],

       [[ 3, 11],
        [ 7, 15]]])
```

```
In [38]: arr.transpose(2,1,0)
Out[38]:
array([[[ 0,  8],
        [ 4, 12]],

       [[ 1,  9],
        [ 5, 13]],

       [[ 2, 10],
        [ 6, 14]],

       [[ 3, 11],
        [ 7, 15]]])
```

数组的转置和轴对称

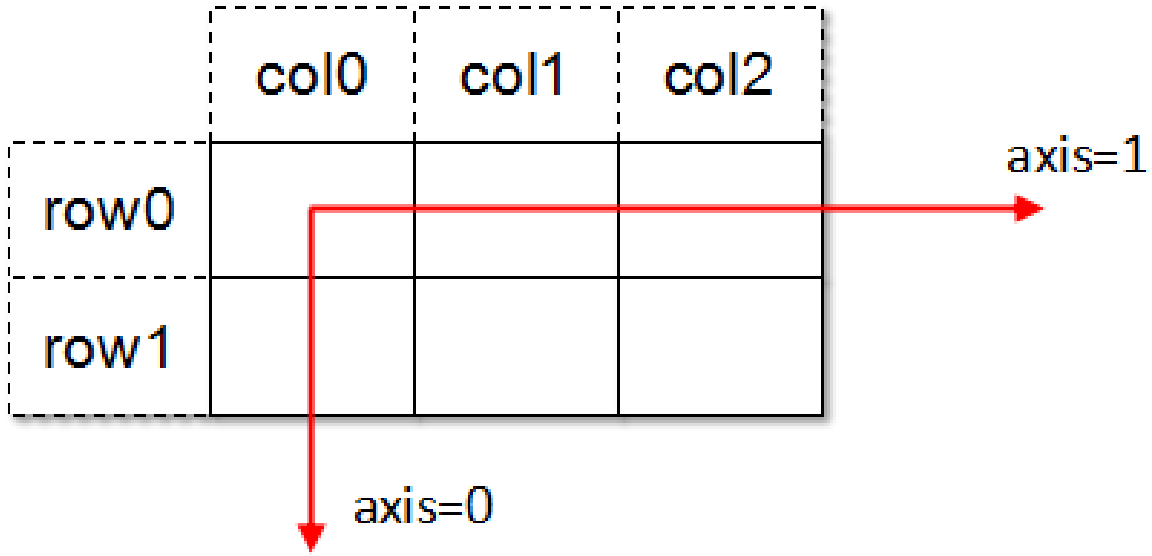
有时可能只需要转换其中的两个轴，这时可以使用swapaxes()方法实现，该方法需要接受一对轴编号，比如(1,0)。

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

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

多学一招

高维数据执行某些操作（如转置）时，需要指定维度编号，这个编号是从0开始的，然后依次递增1。其中，位于纵向的轴（y轴）的编号为0，位于横向的轴（x轴）的编号为1，以此类推。



过渡页

07 NumPy通用函数

08 利用NumPy数组进行数据处理

09 线性代数模块

10 随机数模块

NumPy通用函数

通用函数（ufunc）是一种针对ndarray中的数据执行元素级运算的函数，函数返回的是一个新的数组。

- 我们将ufunc中接收一个数组参数的函数称为**一元通用函数**，接受两个数组参数的则称为**二元通用函数**。

NumPy通用函数

常见的一元通用函数如下表：

函数	描述
abs、fabs	计算整数、浮点数或复数的绝对值
sqrt	计算各元素的平方根
square	计算各元素的平方
exp	计算各元素的指数 e^x
log、log10、log2、log1p	分别为自然对数（底数为 e），底数为 10 的 log，底数为 2 的 log，log(1+x)
sign	计算各元素的正负号：1（正数）、0（零）、-1（负数）
ceil	计算各元素的 ceilling 值，即大于或者等于该值的最小整数

NumPy通用函数

常见的一元通用函数如下表：

函数	描述
floor	计算各元素的 floor 值，即小于等于该值的最大整数
rint	将各元素四舍五入到最接近的整数
modf	将数组的小数和整数部分以两个独立数组的形式返回
isnan	返回一个表示“哪些值是 NaN”的布尔型数组
isfinite、isinf	分别返回表示“哪些元素是有穷的”或“哪些元素是无穷”的布尔型数组
sin、sinh、cos、cosh、tan、tanh	普通型和双曲型三角函数
arcos、arccosh、arcsin	反三角函数

NumPy通用函数

常见的二元通用函数如下表：

函数	描述
add	将数组中对应的元素相加
subtract	从第一个数组中减去第二个数组中的元素
multiply	数组元素相乘
divide，floor_divide	除法或向下整除法（舍去余数）
maximum、fmax	元素级的最大值计算
minimum、fmin	元素级的最小值计算
mod	元素级的求模计算
copysign	将第二个数组中的值的符号赋值给第一个数组中的值
greater、greater_equal、less、less_equal、equal、not_equal、logical_and、logical_or、logical_xor	执行元素级的比较运算，最终产生布尔型数组，相当于运算符>、≥、<、≤、==、!=

过渡页

07 NumPy通用函数

08 利用NumPy数组进行数据处理

09 线性代数模块

10 随机数模块

将条件逻辑转为数组运算

NumPy的where()函数是三元表达式x if condition else y的矢量化版本。

```
arr_x = np.array([1, 5, 7])
arr_y = np.array([2, 6, 8])
arr_con = np.array([True, False, True])
result = np.where(arr_con, arr_x, arr_y)
```

array([1, 6, 7])

数组统计运算

通过NumPy库中的相关方法，我们可以很方便地运用Python进行数组的统计汇总。

方法	描述
sum	对数组中全部或某个轴向的元素求和
mean	算术平均值
min	计算数组中的最小值
max	计算数组中的最大值
argmin	表示最小值的索引
argmax	表示最大值的索引
cumsum	所有元素的累计和
cumprod	所有元素的累计积

数组排序

如果希望对NumPy数组中的元素进行排序，可以通过sort()方法实现。

```
arr = np.array([[6, 2, 7],
                [3, 6, 2],
                [4, 3, 2]])
arr.sort()
```

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

数组排序

如果希望对任何一个轴上的元素进行排序，则需要将轴的编号作为sort()方法的参数传入。

```
arr = np.array([[6, 2, 7],
                [3, 6, 2],
                [4, 3, 2]])
# 沿着编号为0的轴对元素排序
arr.sort(0)
```

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

检索数组元素

all()函数用于判断整个数组中的元素的值是否全部满足条件，如果满足条件返回True，否则返回False。

```
arr = np.array([[1, -2, -7],
                [-3, 6, 2],
                [-4, 3, 2]])
# arr的所有元素是否都大于0
np.all(arr > 0)
```

False

检索数组元素

any()函数用于判断整个数组中的元素至少有一个满足条件就返回True，否则就返回False。

```
arr = np.array([[1, -2, -7],
                [-3, 6, 2],
                [-4, 3, 2]])
# arr的所有元素是否有一个大于0
np.any(arr > 0)
```

True

唯一化及其他集合逻辑

针对一维数组，NumPy提供了unique()函数来找出数组中的唯一值，并返回排序后的结果。

```
arr = np.array([12, 11, 34, 23, 12, 8, 11])
np.unique(arr)
```

array([8, 11, 12, 23, 34])

唯一化及其他集合逻辑

in1d()函数用于判断数组中的元素是否在另一个数组中存在，该函数返回的是一个布尔型的数组。

```
arr = np.array([12, 11, 34, 23, 12, 8, 11])
np.in1d(arr, [11, 12])
```

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

唯一化及其他集合逻辑

NumPy提供的有关集合的函数还有很多，常见的函数如下表所示。

函数	描述
unique(x)	计算 x 中的唯一元素，并返回有序结果
intersect1d(x,y)	计算 x 和 y 中的公共元素，并返回有序结果
union1d(x,y)	计算 x 和 y 的并集，并返回有序结果
in1d(x,y)	得到一个表示“x 的元素是否包含 y”的布尔型数组
setdiff1d(x,y)	集合的差，即元素在 x 中且不在 y 中
setxor1d(x,y)	集合的对称差，即存在于一个数组中但不同时存在于两个数组中的元素

过渡页

07 NumPy通用函数

08 利用NumPy数组进行数据处理

09 线性代数模块

10 随机数模块

线性代数模块

numpy.linalg模块中有一组标准的矩阵分解运算以及诸如逆和行列式之类的东西。

例如，矩阵相乘，如果我们通过“*”对两个数组相乘的话，得到的是一个元素级的积，而不是一个矩阵点积。

线性代数模块

NumPy中提供了一个用于矩阵乘法的dot()方法。

```
arr_x = np.array([[1, 2, 3], [4, 5, 6]])
arr_y = np.array([[1, 2], [3, 4], [5, 6]])
# 等价于np.dot(arr_x, arr_y)
arr_x.dot(arr_y)
```

```
array([[22, 28],
       [49, 64]])
```

线性代数模块

矩阵点积的条件是矩阵A的列数等于矩阵B的行数，假设A为 m*p的矩阵，B为 p*n 的矩阵，那么矩阵A与B的乘积就是一个 m*n 的矩阵C，其中矩阵C的第i行第j列的元素可以表示为：

$$(A,B)_{ij} = \sum_{k=1}^{p} a_{ik} b_{kj} = a_{i1} b_{1j} + a_{i2} b_{2j} + \cdots + a_{ip} b_{pj}$$

线性代数模块

除此之外，linalg模块中还提供了其他很多有用的函数。

函数	描述
dot	矩阵乘法
diag	以一维数组的形式放回方阵的对角线，或将一维数组转为方阵
trace	计算对角线元素和
det	计算矩阵的行列式
eig	计算方阵的特征值和特征向量
inv	计算方阵的逆
qr	计算 qr 分解
svd	计算奇异值（SVD）
solve	解线性方程组 Ax=b，其中 A 是一个方阵
lstsq	计算 Ax=b 的最小二乘解

过渡页

07 NumPy通用函数

08 利用NumPy数组进行数据处理

09 线性代数模块

10 随机数模块

随机数模块

与Python的random模块相比，NumPy的random模块功能更多，它增加了一些可以高效生成多种概率分布的样本值的函数。

```
# 随机生成一个二维数组
np.random.rand(3, 3)
```

rand()函数隶属于numpy.random模块，它的作用是随机生成N维浮点数组。

随机数模块

除此之外，random模块中还包括了可以生成服从多种概率分布随机数的其它函数。

函数	描述
seed	生成随机数的种子
rand	产生均匀分布的样本值
randint	从给定的上下限范围内随机选取整数
normal	产生正态分布的样本值
beta	产生 Beta 分布的样本值
uniform	产生在[0,1]中的均匀分布的样本值

随机数模块

seed( )函数可以保证生成的随机数具有可预测性，也就是说产生的随机数相同。

```
numpy.random.seed(seed=None)
```

上述函数中只有一个seed参数，用于指定随机数生成时所用算法开始的整数值。

随机数模块

当调用seed()函数时，如果传递给**seed参数的值相同**，则每次**生成的随机数都是一样的**。

当传递的参数值不同或者不传递参数时，则seed()函数的作用跟rand()函数相同，即多次生成随机数且每次生成的随机数都不同。

本章小结

- 本章主要针对科学计算库NumPy进行了介绍，包括ndarry数组对象的属性和数据类型、数组的运算、索引和切片操作、数组的转置和轴对称、NumPy通用函数、线性代数模块、随机数模块以及使用数组进行数据处理的相关操作。

- 通过本章的学习，希望大家能熟练使用NumPy包，为后面章节的学习奠定基础。