NumPy 迭代数组 NumPy 位运算

Numpy 数组操作

Numpy 中包含了一些函数用于处理数组，大概可分为以下几类：

- 修改数组形状
- 翻转数组
- 修改数组维度
- 连接数组
- 分割数组
- 数组元素的添加与删除

修改数组形状

函数	描述
reshape	不改变数据的条件下修改形状
flat	数组元素迭代器
flatten	返回一份数组拷贝，对拷贝所做的修改不会影响原始数组
ravel	返回展开数组

numpy.reshape

numpy.reshape 函数可以在不改变数据的条件下修改形状，格式如下： numpy.reshape(arr, newshape, order='C')

- arr：要修改形状的数组
- newshape：整数或者整数数组，新的形状应当兼容原有形状
- order：'C' -- 按行，'F' -- 按列，'A' -- 原顺序，'k' -- 元素在内存中的出现顺序。

实例

```
import numpy as np
a = np.arange(8)
print ('原始数组: ')
print (a)
print ('\n')
b = a.reshape(4,2)
print ('修改后的数组: ')
print (b)
```

输出结果如下：

```
原始数组:
[0 1 2 3 4 5 6 7]

修改后的数组:
[[0 1]
 [2 3]
 [4 5]
 [6 7]]
```

numpy.ndarray.flat

numpy.ndarray.flat 是一个数组元素迭代器，实例如下:

实例

```
import numpy as np
a = np.arange(9).reshape(3,3)
print ('原始数组: ')
for row in a:
print (row)
#对数组中每个元素都进行处理，可以使用flat属性，该属性是一个数组元素迭代器:
print ('迭代后的数组: ')
for element in a.flat:
print (element)
```

输出结果如下：

```
原始数组:
[0 1 2]
[3 4 5]
[6 7 8]
迭代后的数组:
0
1
2
3
4
5
6
7
8
```

numpy.ndarray.flatten

numpy.ndarray.flatten 返回一份数组拷贝，对拷贝所做的修改不会影响原始数组，格式如下：

```
ndarray.flatten(order='C')
```

参数说明：

- order：'C' -- 按行，'F' -- 按列，'A' -- 原顺序，'K' -- 元素在内存中的出现顺序。

实例

```
import numpy as np
a = np.arange(8).reshape(2,4)
print ('原数组：')
print (a)
print ('\n')
# 默认按行
print ('展开的数组：')
print (a.flatten())
print ('\n')
print ('以 F 风格顺序展开的数组：')
print (a.flatten(order = 'F'))
```

输出结果如下：

```
原数组：
[[0 1 2 3]
 [4 5 6 7]]


展开的数组：
[0 1 2 3 4 5 6 7]


以 F 风格顺序展开的数组：
[0 4 1 5 2 6 3 7]
```

numpy.ravel

numpy.ravel() 展平的数组元素，顺序通常是"C风格"，返回的是数组视图（view，有点类似 C/C++引用reference的意味），修改会影响原始数组。

该函数接收两个参数：

```
numpy.ravel(a, order='C')
```

参数说明：

- order：'C' -- 按行，'F' -- 按列，'A' -- 原顺序，'K' -- 元素在内存中的出现顺序。

实例

```
import numpy as np
a = np.arange(8).reshape(2,4)
print ('原数组：')
print (a)
print ('\n')
print ('调用 ravel 函数之后：')
```

```
print (a.ravel())
print ('\n')
print ('以 F 风格顺序调用 ravel 函数之后：')
print (a.ravel(order = 'F'))
```

输出结果如下：

```
原数组:
[[0 1 2 3]
 [4 5 6 7]]


调用 ravel 函数之后:
[0 1 2 3 4 5 6 7]


以 F 风格顺序调用 ravel 函数之后:
[0 4 1 5 2 6 3 7]
```

翻转数组

函数	描述
transpose	对换数组的维度
ndarray.T	和 self.transpose() 相同
rollaxis	向后滚动指定的轴
swapaxes	对换数组的两个轴

numpy.transpose

numpy.transpose 函数用于对换数组的维度，格式如下：

```
numpy.transpose(arr, axes)
```

参数说明:

- arr：要操作的数组
- axes：整数列表，对应维度，通常所有维度都会对换。

实例

```
import numpy as np
a = np.arange(12).reshape(3,4)
print ('原数组：')
print (a )
```

```
print ('\n')
print ('对换数组：')
print (np.transpose(a))
```

输出结果如下：

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


对换数组：
[[ 0  4  8]
 [ 1  5  9]
 [ 2  6 10]
 [ 3  7 11]]
```

numpy.ndarray.T 类似 numpy.transpose：

实例

```
import numpy as np
a = np.arange(12).reshape(3,4)
print ('原数组：')
print (a)
print ('\n')
print ('转置数组：')
print (a.T)
```

输出结果如下：

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


转置数组：
[[ 0  4  8]
 [ 1  5  9]
 [ 2  6 10]
 [ 3  7 11]]
```

numpy.rollaxis

numpy.rollaxis 函数向后滚动特定的轴到一个特定位置，格式如下：

```
numpy.rollaxis(arr, axis, start)
```

参数说明：

- `arr`：数组
- `axis`：要向后滚动的轴，其它轴的相对位置不会改变
- `start`：默认为零，表示完整的滚动。会滚动到特定位置。

实例

```
import numpy as np
# 创建了三维的 ndarray
a = np.arange(8).reshape(2,2,2)
print ('原数组：')
print (a)
print ('\n')
# 将轴 2 滚动到轴 0（宽度到深度）
print ('调用 rollaxis 函数：')
print (np.rollaxis(a,2))
# 将轴 0 滚动到轴 1：（宽度到高度）
print ('\n')
print ('调用 rollaxis 函数：')
print (np.rollaxis(a,2,1))
```

输出结果如下：

```
原数组：
[[[0 1]
  [2 3]]

 [[4 5]
  [6 7]]]


调用 rollaxis 函数：
[[[0 2]
  [4 6]]

 [[1 3]
  [5 7]]]


调用 rollaxis 函数：
[[[0 2]
  [1 3]]

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

numpy.swapaxes

numpy.swapaxes 函数用于交换数组的两个轴，格式如下：

```
numpy.swapaxes(arr, axis1, axis2)
```

- arr：输入的数组
- axis1：对应第一个轴的整数
- axis2：对应第二个轴的整数

实例

```
import numpy as np
# 创建了三维的 ndarray
a = np.arange(8).reshape(2,2,2)
print ('原数组：')
print (a)
print ('\n')
# 现在交换轴 0（深度方向）到轴 2（宽度方向）
print ('调用 swapaxes 函数后的数组：')
print (np.swapaxes(a, 2, 0))
```

输出结果如下：

```
原数组：
[[[0 1]
  [2 3]]

 [[4 5]
  [6 7]]]


调用 swapaxes 函数后的数组：
[[[0 4]
  [2 6]]

 [[1 5]
  [3 7]]]
```

修改数组维度

维度	描述
broadcast	产生模仿广播的对象
broadcast_to	将数组广播到新形状

维度	描述
expand_dims	扩展数组的形状
squeeze	从数组的形状中删除一维条目

numpy.broadcast

numpy.broadcast 用于模仿广播的对象，它返回一个对象，该对象封装了将一个数组广播到另一个数组的结果。

该函数使用两个数组作为输入参数，如下实例：

实例

```
import numpy as np
x = np.array([[1], [2], [3]])
y = np.array([4, 5, 6])
# 对 y 广播 x
b = np.broadcast(x,y)
# 它拥有 iterator 属性，基于自身组件的迭代器元组
print ('对 y 广播 x: ')
r,c = b.iters
# Python3.x 为 next(context) ，Python2.x 为 context.next()
print (next(r), next(c))
print (next(r), next(c))
print ('\n')
# shape 属性返回广播对象的形状
print ('广播对象的形状: ')
print (b.shape)
print ('\n')
# 手动使用 broadcast 将 x 与 y 相加
b = np.broadcast(x,y)
c = np.empty(b.shape)
print ('手动使用 broadcast 将 x 与 y 相加: ')
print (c.shape)
print ('\n')
c.flat = [u + v for (u,v) in b]
print ('调用 flat 函数: ')
print (c)
print ('\n')
# 获得了和 NumPy 内建的广播支持相同的结果
print ('x 与 y 的和: ')
print (x + y)
```

输出结果为：

```
对 y 广播 x:
1 4
1 5


广播对象的形状:
(3, 3)
```

```
手动使用 broadcast 将 x 与 y 相加:
(3, 3)


调用 flat 函数:
[[5. 6. 7.]
 [6. 7. 8.]
 [7. 8. 9.]]


x 与 y 的和:
[[5 6 7]
 [6 7 8]
 [7 8 9]]
```

numpy.broadcast_to

numpy.broadcast_to 函数将数组广播到新形状。它在原始数组上返回只读视图。 它通常不连续。 如果新形状不符合 NumPy 的广播规则，该函数可能会抛出ValueError。

```
numpy.broadcast_to(array, shape, subok)
```

实例

```
import numpy as np
a = np.arange(4).reshape(1,4)
print ('原数组: ')
print (a)
print ('\n')
print ('调用 broadcast_to 函数之后: ')
print (np.broadcast_to(a,(4,4)))
```

输出结果为：

```
原数组:
[[0 1 2 3]]


调用 broadcast_to 函数之后:
[[0 1 2 3]
 [0 1 2 3]
 [0 1 2 3]
 [0 1 2 3]]
```

numpy.expand_dims

numpy.expand_dims 函数通过在指定位置插入新的轴来扩展数组形状，函数格式如下:

```
numpy.expand_dims(arr, axis)
```

参数说明：

- arr：输入数组
- axis：新轴插入的位置

实例

```
import numpy as np
x = np.array(([1,2],[3,4]))
print ('数组 x: ')
print (x)
print ('\n')
y = np.expand_dims(x, axis = 0)
print ('数组 y: ')
print (y)
print ('\n')
print ('数组 x 和 y 的形状: ')
print (x.shape, y.shape)
print ('\n')
# 在位置 1 插入轴
y = np.expand_dims(x, axis = 1)
print ('在位置 1 插入轴之后的数组 y: ')
print (y)
print ('\n')
print ('x.ndim 和 y.ndim: ')
print (x.ndim,y.ndim)
print ('\n')
print ('x.shape 和 y.shape: ')
print (x.shape, y.shape)
```

输出结果为：

```
数组 x:
[[1 2]
 [3 4]]


数组 y:
[[[1 2]
  [3 4]]]


数组 x 和 y 的形状:
(2, 2) (1, 2, 2)
```

```
在位置 1 插入轴之后的数组 y:
[[[1 2]]

 [[3 4]]]


x.ndim 和 y.ndim:
2 3


x.shape 和 y.shape:
(2, 2) (2, 1, 2)
```

numpy.squeeze

numpy.squeeze 函数从给定数组的形状中删除一维的条目，函数格式如下：

```
numpy.squeeze(arr, axis)
```

参数说明：

- arr：输入数组
- axis：整数或整数元组，用于选择形状中一维条目的子集

实例

```
import numpy as np
x = np.arange(9).reshape(1,3,3)
print ('数组 x: ')
print (x)
print ('\n')
y = np.squeeze(x)
print ('数组 y: ')
print (y)
print ('\n')
print ('数组 x 和 y 的形状: ')
print (x.shape, y.shape)
```

输出结果为：

```
数组 x:
[[[0 1 2]
  [3 4 5]
  [6 7 8]]


数组 y:
```

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


 数组 x 和 y 的形状:
 (1, 3, 3) (3, 3)
```

连接数组

函数	描述
concatenate	连接沿现有轴的数组序列
stack	沿着新的轴加入一系列数组。
hstack	水平堆叠序列中的数组（列方向）
vstack	竖直堆叠序列中的数组（行方向）

numpy.concatenate

numpy.concatenate 函数用于沿指定轴连接相同形状的两个或多个数组，格式如下：

```
numpy.concatenate((a1, a2, ...), axis)
```

参数说明：

- a1, a2, ...：相同类型的数组
- axis：沿着它连接数组的轴，默认为 0

实例

```
import numpy as np
a = np.array([[1,2],[3,4]])
print ('第一个数组: ')
print (a)
print ('\n')
b = np.array([[5,6],[7,8]])
print ('第二个数组: ')
print (b)
print ('\n')
# 两个数组的维度相同
print ('沿轴 0 连接两个数组: ')
print (np.concatenate((a,b)))
print ('\n')
print ('沿轴 1 连接两个数组: ')
print (np.concatenate((a,b),axis = 1))
```

输出结果为：

```
第一个数组:
[[1 2]
 [3 4]]


第二个数组:
[[5 6]
 [7 8]]


沿轴 0 连接两个数组:
[[1 2]
 [3 4]
 [5 6]
 [7 8]]


沿轴 1 连接两个数组:
[[1 2 5 6]
 [3 4 7 8]]
```

numpy.stack

numpy.stack 函数用于沿新轴连接数组序列，格式如下：

```
numpy.stack(arrays, axis)
```

参数说明：

- arrays相同形状的数组序列
- axis：返回数组中的轴，输入数组沿着它来堆叠

实例

```
import numpy as np
a = np.array([[1,2],[3,4]])
print ('第一个数组: ')
print (a)
print ('\n')
b = np.array([[5,6],[7,8]])
print ('第二个数组: ')
print (b)
print ('\n')
print ('沿轴 0 堆叠两个数组: ')
print (np.stack((a,b),0))
print ('\n')
```

```
print ('沿轴 1 堆叠两个数组：')
print (np.stack((a,b),1))
```

输出结果如下：

```
第一个数组：
[[1 2]
 [3 4]]


第二个数组：
[[5 6]
 [7 8]]


沿轴 0 堆叠两个数组：
[[[1 2]
  [3 4]]

 [[5 6]
  [7 8]]]


沿轴 1 堆叠两个数组：
[[[1 2]
  [5 6]]

 [[3 4]
  [7 8]]]
```

numpy.hstack

numpy.hstack 是 numpy.stack 函数的变体，它通过水平堆叠来生成数组。

实例

```
import numpy as np
a = np.array([[1,2],[3,4]])
print ('第一个数组：')
print (a)
print ('\n')
b = np.array([[5,6],[7,8]])
print ('第二个数组：')
print (b)
print ('\n')
print ('水平堆叠：')
c = np.hstack((a,b))
print (c)
print ('\n')
```

输出结果如下：

```
第一个数组:
[[1 2]
 [3 4]]


第二个数组:
[[5 6]
 [7 8]]


水平堆叠:
[[1 2 5 6]
 [3 4 7 8]]
```

numpy.vstack

numpy.vstack 是 numpy.stack 函数的变体，它通过垂直堆叠来生成数组。

实例

```
import numpy as np
a = np.array([[1,2],[3,4]])
print ('第一个数组: ')
print (a)
print ('\n')
b = np.array([[5,6],[7,8]])
print ('第二个数组: ')
print (b)
print ('\n')
print ('竖直堆叠: ')
c = np.vstack((a,b))
print (c)
```

输出结果为：

```
第一个数组:
[[1 2]
 [3 4]]


第二个数组:
[[5 6]
 [7 8]]


竖直堆叠:
[[1 2]
 [3 4]
 [5 6]
 [7 8]]
```

分割数组

函数	数组及操作
split	将一个数组分割为多个子数组
hsplit	将一个数组水平分割为多个子数组（按列）
vsplit	将一个数组垂直分割为多个子数组（按行）

numpy.split

numpy.split 函数沿特定的轴将数组分割为子数组，格式如下：

```
numpy.split(ary, indices_or_sections, axis)
```

参数说明：

- ary：被分割的数组
- indices_or_sections：果是一个整数，就用该数平均切分，如果是一个数组，为沿轴切分的位置（左开右闭）
- axis：沿着哪个维度进行切向，默认为0，横向切分。为1时，纵向切分

实例

```
import numpy as np
a = np.arange(9)
print ('第一个数组：')
print (a)
print ('\n')
print ('将数组分为三个大小相等的子数组：')
b = np.split(a,3)
print (b)
print ('\n')
print ('将数组在一维数组中表明的位置分割：')
b = np.split(a,[4,7])
print (b)
```

输出结果为：

```
第一个数组：
[0 1 2 3 4 5 6 7 8]


将数组分为三个大小相等的子数组：
[array([0, 1, 2]), array([3, 4, 5]), array([6, 7, 8])]
```

```
将数组在一维数组中表明的位置分割:
[array([0, 1, 2, 3]), array([4, 5, 6]), array([7, 8])]
```

numpy.hsplit

numpy.hsplit 函数用于水平分割数组，通过指定要返回的相同形状的数组数量来拆分原数组。

实例

```
import numpy as np
harr = np.floor(10 * np.random.random((2, 6)))
print ('原array: ')
print(harr)
print ('拆分后: ')
print(np.hsplit(harr, 3))
```

输出结果为：

```
原array:
[[4. 7. 6. 3. 2. 6.]
 [6. 3. 6. 7. 9. 7.]]
拆分后:
[array([[4., 7.],
       [6., 3.]]), array([[6., 3.],
       [6., 7.]]), array([[2., 6.],
       [9., 7.]])]
```

numpy.vsplit

numpy.vsplit 沿着垂直轴分割，其分割方式与hsplit用法相同。

实例

```
import numpy as np
a = np.arange(16).reshape(4,4)
print ('第一个数组: ')
print (a)
print ('\n')
print ('竖直分割: ')
b = np.vsplit(a,2)
print (b)
```

输出结果为：

```
第一个数组:
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]
```

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

数组元素的添加与删除

函数	元素及描述
resize	返回指定形状的新数组
append	将值添加到数组末尾
insert	沿指定轴将值插入到指定下标之前
delete	删掉某个轴的子数组，并返回删除后的新数组
unique	查找数组内的唯一元素

numpy.resize

numpy.resize 函数返回指定大小的新数组。

如果新数组大小大于原始大小，则包含原始数组中的元素的副本。

```
numpy.resize(arr, shape)
```

参数说明：

- arr：要修改大小的数组
- shape：返回数组的新形状

实例

```
import numpy as np
a = np.array([[1,2,3],[4,5,6]])
print ('第一个数组：')
print (a)
print ('\n')
print ('第一个数组的形状：')
print (a.shape)
print ('\n')
b = np.resize(a, (3,2))
print ('第二个数组：')
print (b)
print ('\n')
print ('第二个数组的形状：')
print (b.shape)
print ('\n')
# 要注意 a 的第一行在 b 中重复出现，因为尺寸变大了
```

```
print ('修改第二个数组的大小: ')
b = np.resize(a,(3,3))
print (b)
```

输出结果为：

```
第一个数组:
[[1 2 3]
 [4 5 6]]


第一个数组的形状:
(2, 3)


第二个数组:
[[1 2]
 [3 4]
 [5 6]]


第二个数组的形状:
(3, 2)


修改第二个数组的大小:
[[1 2 3]
 [4 5 6]
 [1 2 3]]
```

numpy.append

numpy.append 函数在数组的末尾添加值。 追加操作会分配整个数组，并把原来的数组复制到新数组中。 此外，输入数组的维度必须匹配否则将生成ValueError。

append 函数返回的始终是一个一维数组。

```
numpy.append(arr, values, axis=None)
```

参数说明：

- `arr`：输入数组
- `values`：要向`arr`添加的值，需要和`arr`形状相同（除了要添加的轴）
- `axis`：默认为 None。当axis无定义时，是横向加成，返回总是为一维数组！当axis有定义的时候，分别为0和1的时候。当axis有定义的时候，分别为0和1的时候（列数要相同）。当axis为1时，数组是加在右边（行数要相同）。

实例

```
import numpy as np
a = np.array([[1,2,3],[4,5,6]])
print ('第一个数组：')
print (a)
print ('\n')
print ('向数组添加元素：')
print (np.append(a, [7,8,9]))
print ('\n')
print ('沿轴 0 添加元素：')
print (np.append(a, [[7,8,9]],axis = 0))
print ('\n')
print ('沿轴 1 添加元素：')
print (np.append(a, [[5,5,5],[7,8,9]],axis = 1))
```

输出结果为：

```
第一个数组：
[[1 2 3]
 [4 5 6]]


向数组添加元素：
[1 2 3 4 5 6 7 8 9]


沿轴 0 添加元素：
[[1 2 3]
 [4 5 6]
 [7 8 9]]


沿轴 1 添加元素：
[[1 2 3 5 5 5]
 [4 5 6 7 8 9]]
```

numpy.insert

numpy.insert 函数在给定索引之前，沿给定轴在输入数组中插入值。

如果值的类型转换为要插入，则它与输入数组不同。 插入没有原地的，函数会返回一个新数组。 此外，如果未提供轴，则输入数组会被展开。

```
numpy.insert(arr, obj, values, axis)
```

参数说明：

- arr：输入数组
- obj：在其之前插入值的索引

- values：要插入的值
- axis：沿着它插入的轴，如果未提供，则输入数组会被展开

实例

```
import numpy as np
a = np.array([[1,2],[3,4],[5,6]])
print ('第一个数组：')
print (a)
print ('\n')
print ('未传递 Axis 参数。 在插入之前输入数组会被展开。')
print (np.insert(a,3,[11,12]))
print ('\n')
print ('传递了 Axis 参数。 会广播值数组来配输入数组。')
print ('沿轴 0 广播：')
print (np.insert(a,1,[11],axis = 0))
print ('\n')
print ('沿轴 1 广播：')
print (np.insert(a,1,11,axis = 1))
```

输出结果如下：

```
第一个数组：
[[1 2]
 [3 4]
 [5 6]]


未传递 Axis 参数。 在插入之前输入数组会被展开。
[ 1  2  3 11 12  4  5  6]


传递了 Axis 参数。 会广播值数组来配输入数组。
沿轴 0 广播：
[[ 1  2]
 [11 11]
 [ 3  4]
 [ 5  6]]


沿轴 1 广播：
[[ 1 11  2]
 [ 3 11  4]
 [ 5 11  6]]
```

numpy.delete

numpy.delete 函数返回从输入数组中删除指定子数组的新数组。 与 insert() 函数的情况一样，如果未提供轴参数，则输入数组将展开。

```
Numpy.delete(arr, obj, axis)
```

参数说明：

- arr：输入数组
- obj：可以被切片，整数或者整数数组，表明要从输入数组删除的子数组
- axis：沿着它删除给定子数组的轴，如果未提供，则输入数组会被展开

实例

```
import numpy as np
a = np.arange(12).reshape(3,4)
print ('第一个数组：')
print (a)
print ('\n')
print ('未传递 Axis 参数。 在插入之前输入数组会被展开。')
print (np.delete(a,5))
print ('\n')
print ('删除第二列：')
print (np.delete(a,1,axis = 1))
print ('\n')
print ('包含从数组中删除的替代值的切片：')
a = np.array([1,2,3,4,5,6,7,8,9,10])
print (np.delete(a, np.s_[::2]))
```

输出结果为：

```
第一个数组：
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]


未传递 Axis 参数。 在插入之前输入数组会被展开。
[ 0  1  2  3  4  6  7  8  9 10 11]


删除第二列：
[[ 0  2  3]
 [ 4  6  7]
 [ 8 10 11]]


包含从数组中删除的替代值的切片：
[ 2  4  6  8 10]
```

numpy.unique

numpy.unique 函数用于去除数组中的重复元素。

```
numpy.unique(arr, return_index, return_inverse, return_counts)
```

- arr：输入数组，如果不是一维数组则会展开
- return_index：如果为true，返回新列表元素在旧列表中的位置（下标），并以列表形式储
- return_inverse：如果为true，返回旧列表元素在新列表中的位置（下标），并以列表形式储
- return_counts：如果为true，返回去重数组中的元素在原数组中的出现次数

实例

```
import numpy as np
a = np.array([5,2,6,2,7,5,6,8,2,9])
print ('第一个数组：')
print (a)
print ('\n')
print ('第一个数组的去重值：')
u = np.unique(a)
print (u)
print ('\n')
print ('去重数组的索引数组：')
u,indices = np.unique(a, return_index = True)
print (indices)
print ('\n')
print ('我们可以看到每个和原数组下标对应的数值：')
print (a)
print ('\n')
print ('去重数组的下标：')
u,indices = np.unique(a,return_inverse = True)
print (u)
print ('\n')
print ('下标为：')
print (indices)
print ('\n')
print ('使用下标重构原数组：')
print (u[indices])
print ('\n')
print ('返回去重元素的重复数量：')
u,indices = np.unique(a,return_counts = True)
print (u)
print (indices)
```

输出结果为：

```
第一个数组：
[5 2 6 2 7 5 6 8 2 9]


第一个数组的去重值：
```

```
[2 5 6 7 8 9]
```

去重数组的索引数组：

```
[1 0 2 4 7 9]
```

我们可以看到每个和原数组下标对应的数值：

```
[5 2 6 2 7 5 6 8 2 9]
```

去重数组的下标：

```
[2 5 6 7 8 9]
```

下标为：

```
[1 0 2 0 3 1 2 4 0 5]
```

使用下标重构原数组：

```
[5 2 6 2 7 5 6 8 2 9]
```

返回去重元素的重复数量：

```
[2 5 6 7 8 9]
[3 2 2 1 1 1]
```

NumPy 迭代数组 NumPy 位运算

点我分享笔记