# Numpy科学计算库

Numpy是Python科学计算库的基础，主要包括：

* 强大的N维数组对象和向量运算
* 一些复杂的功能
* 与C／C++和Fortran代码的集成
* 实用的线形代数运算、傅立叶变换、随机数生产等

Numpy提供了一个简洁的C语言接口，可以非常方便地使用C语言编写的代码操作Numpy数组对象，反过来使用C语言生成的数组也可以轻松地转化成Numpy数组对象；这一特性使得Python轻松的与其他编程语言C/C++黏结在一起。Numpy除了用作科学计算，也可以用作多维度普通数据的容器；并且可以定义任何类型的数据，使得Numpy高效、无缝地与各种类型的数据分析库连接起来。

## 1 Numpy基础

Numpy的主要对象是一个多维度的、均匀的多维数组。Numpy提供了各种函数方法可以非常方便灵活的操作数组，熟练掌握数组的基本概念是使用数组这种数据结构的基本要求。

### 1.1 数组对象介绍

多维数组类似于表格，表格内的数据是同一类型，一般是数字，是一个高效的大量数据的容器，如图1所示。数组可以像普通的数字那样进行简单的算术运算，如图2和图3所示；也可以通过下标对数组内的数据进行索引取值。

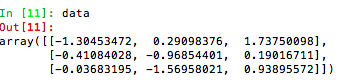


图1 Numpy数组

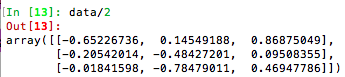


图2 数组除法运算

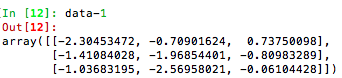


图3 数组减法运算

在Numpy中方向称作轴，轴的数目称作维。例如，3D空间内的一点坐标是[1,1,0],在Numpy中是一个一维数组，因为它只有一个轴。再如：

array([1,1,1],

[0.0.0])

这个数组有两个方向，有两个轴，所以是一个二维的数组。

一个N维数组是一个普通的均质数据的容器，就是说容器内的数据必须是同一数据类型。N维数组在Numpy中是一个ndarray类，也可以叫作array。不要把Numpy中的array和Python标准库中的array.array混淆，标准库中的array只处理一维数组，且只提供少量的函数。ndarray比较重要的属性包括有：

**1．ndarray.ndim**

数组的维数，如图4所示。

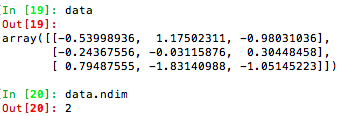


图4 数组的维度

**2．ndarray.shape**

数组各个轴的长度。返回一个元组，表示数组在各个方向上的长度。例如，一个n行m列的矩阵，shape值会是（n，m）。返回元组的长度等于数组的维度，如图5所示。

/Users/zxy/Desktop/屏幕快照 2017-06-25 下午7.57.53.png

图5 数组的形状

**3．ndarray.size**

数组元素的总个数个数，这个值等于shape元组内各个元素的乘积，如图6所示。

/Users/zxy/Desktop/屏幕快照 2017-06-25 下午8.00.54.png

图6 数组的大小

**4．ndarry.dtype**

数组内元素的数据类型。可以使用Python的标准数据类型，Numpy也提供了常用的数据类型。

**5．ndarry.itemsize**

表示数组内元素占计算机的内存的大小。例如，一个float64类型的数据在计算机内8比特的内存空间。

**6．ndaary.data**

数组在内存中的地址，一般通过索引操作元素，所以很少使用到这个属性。

### 1.2 生成数组

生成数组的方式多种多样，比较常用的是使用Python列表或元组或其他有序数据结构作为array函数的参数来生成数组。Numpy会根据元素来推测合适的数组数据类型，如图7所示。

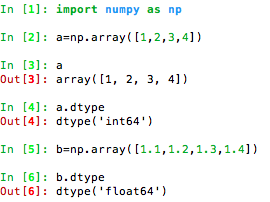


图7 数组的数据类型

初学者在使用array生成数组时，直接在括号内输入一系列数字，而不是使用列表等作为输入参数，这是比较常见的错误，如图8所示。

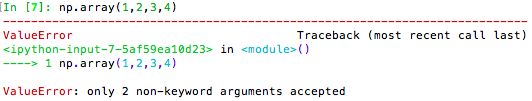


图8 错误的生产数组

array会把序列的序列转换成二维数组，序列内嵌套序列的序列会生成三维数组，依次类推，如图9所示。

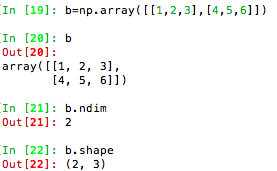


图9 生成二维、三维数组

也可以在array内显式的声明数组的数据类型，如图10所示。

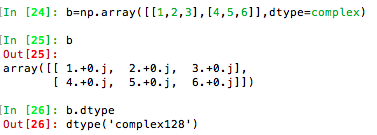


图10 声明数组的数据类型

实际中经常是已知数组的大小，但是数组内元素的值不知道。因此Numpy提供了其他生产数组的方法，数组内的数据只是用来占位的，后面有需要可以更新。

例如，zeros生成全为0的数组，ones生成全为1的数组，empty生成随机数组。在默认情况下，使用这些函数生产的数组数据类型是float64。zeros的输入参数如果是一个整数，会生成一维数组；如果是一个序列，会生成多维数组，如图11所示。

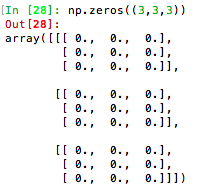


图11 使用zeros生成数组

ones的使用方法和zeros类似，如图12所示。

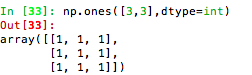


图12 使用ones生成数组

empty会根据系统的状态，随机生成一些占位的元素，如图13所示

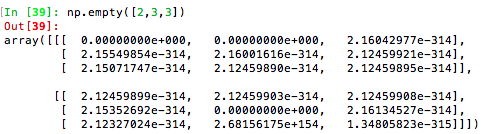


图13 使用empty生成数组

注意：empty生成的数完全是随机的，不要试图假设生成的数组元素的值。

Numpy提供了一个与Python内建函数range类似的arange函数，输入参数包括初始值、终止值、间隔值，可以生成一维的有序数组。

如果只输入一个参数，默认的初始值是0，间隔值为1，如图14所示

屏幕快照%202017-06-27%20下午9.53.06.png

图14 arange函数的用法

如果提供两个参数，第一个参数作为初始值，第二个参数作为终止值，默认间隔值为1，如图15所示。

屏幕快照%202017-06-27%20下午9.55.29.png

图15 arange函数的用法

如果提供三个参数，三个输入值依次对应初始值、终止值、间隔值，如图16所示。

屏幕快照%202017-06-27%20下午9.57.13.png

图16 arange函数的用法

注意：三种情况下，输出数组的元素都没有包括终止值。

当arrange用于浮点数的时候，由于浮点数有限的精度，很难预测所生成数组的大小。Numpy提供了linspace函数，该函数可以指定生成数组的大小，在指定的区间内生成均匀的数组。第一个参数为开始值，第二个参数为终止值，第三个参数为数组的大小，默认值是50，如图17所示。

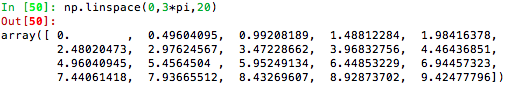


图17 linspace函数生产数组

Numpy提供了rand函数用于生成数组值在（0,1）区间上均匀分布的数组。输入参数是数组在各个维度上的长度，如图18所示。

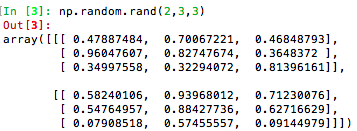


图18 rand函数生产数组

注意：以上方法生产的数组，默认的元素的数据类型都是float64。

常用的生产数组的方法，如表1所示。

表1 生成数组的方法

|  |  |
| --- | --- |
| array | 把输入值(例如，元组、列表等)转换成数组，默认情况会复制输入值 |
| zeros | 生产全为0的数组 |
| ones | 生产全为1的数组 |
| zeros\_like | 返回一个与输入数组形状和数据类型但数组值全为0的数组 |
| ones\_like | 返回一个与输入数组形状和数据类型但数组值全为1的数组 |
| arange | 生成一维数组序列 |
| linspace | 生产指定大小的一维数组 |
| numpy.random.rand | 生产指定大小在元素在[0,1)区间上均匀分布的数组 |
| numpy.random.randn | 生产服从标准分布的数组 |

### 1.3 数组对象数据类型

Python只定义了一类特殊的数据类型，例如，只有一种整型数、一种浮点数；在使用Python编写程序的时候，不需要像C／C++那样声明变量的数据类型，使得编写程序更加容易。但是，对于科学计算而言，对数据类型的控制有时是必需的。

不同的数据类型，存储方式会有所不同。Numpy提供了新的24种基本的数值数据类型，大多数的命名方式都是基于C语言数据类型的命名方式。首先是数据的类型，然后是一个整数，表示其占用的内存位数，例如，int8，表示占8位或一比特的存储空间的整型数。表2列出一些常用的数值数据类型。

表2 常用的Numpy数值数据类型

|  |  |
| --- | --- |
| 类别 | 描述 |
| bool\_ | 与Python bool\_值兼容 |
| bool18 | 8位存储空间 |
| byte | 与C的char类型兼容 |
| short | 与C的short兼容 |
| intc | 与C的int兼容 |
| int\_ | 与Python的int兼容 |
| lonlong | 与C的 long long 兼容 |
| int8 | 8位存储空间 |
| int16 | 16位存储空间 |
| int32 | 32位存储空间 |
| int64 | 64位存储空间 |
| single | 与C的 float兼容 |
| double | 与C的double兼容 |
| float\_ | 与Python的 float兼容 |
| float16 | 16位存储空间 |
| float32 | 32位存储空间 |
| float64 | 64位存储空间 |
| float106 | 106位存储空间，不与所有系统兼容 |
| float128 | 128位存储空间，不与所有系统兼容 |

表2中某种数据类型与C兼容的意思就是说在Numpy数据的存储方式和解释方法与C语言操作这种类型的数据是一模一样，可以使用C语言编写的程序处理来Numpy的这类数据类型。

注意：没必要记住表内的各种数据类型，在常规的数据处理分析的时候，只需要关心数据属于哪一个大类，例如，整型数、浮点数、字符串、布尔数等。

可以使用ndarray的astype方法，转换数组的数据类型，如图110所示。

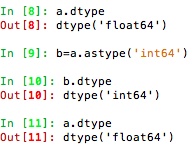


图19 数组的数据类型

这里需要说明一下，astype首先复制原数组，然后再转换数据的数据类型，并返回新的数组，所以原数组的数据类型并没有发生变化，如图20所示。

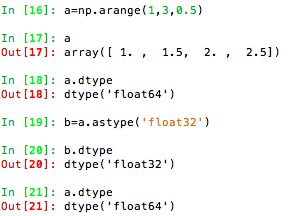


图20 astype转换数据类型

把浮点点数转换成整型数，会截断小数点后面的数字，失去精度，如图21所示。

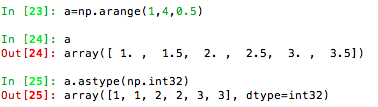


图21 astype转换数据类型

也可以把字符串转换成数字，前提是字符串本身是表示数字的。否则会引发TypeError异常，如图22所示。

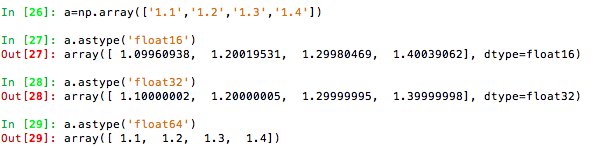


图22 astype数据类型转换

注意：使用不同类型的浮点数，转换结果的精度是不一样的；浮点数类型占用的位数多，精度高。

### 1.4 打印数组

Numpy以列表嵌套的形式打印出数组，如图23所示，但有自动分层的特点：

* 最后一个轴，从左向右
* 倒数第二个轴，从上到下
* 剩下的从上到下，并用空行隔开

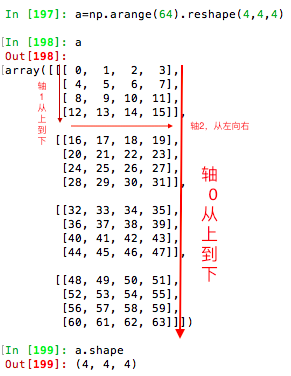


图23 数组的结构

例如，一维数组，以行的形式输出，二维数组以矩阵的形式输出,三维数组以嵌套矩阵的形式打印，如图24所示。

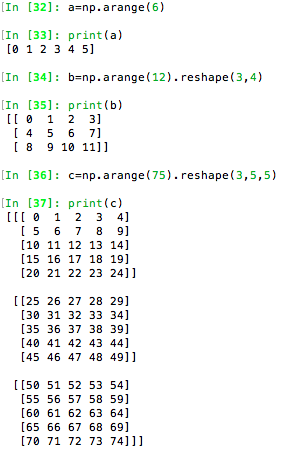


图24 打印不同维度的数组

这里使用到了reshape方法，后面会介绍到如果数组太长，Numpy会自动跳过中间的部分，只打印四周的元素。

一维数组的输出结果，如图25所示。

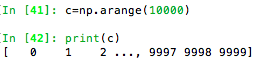


图25 打印一维数组

二维数组的输出结果，如图26所示。

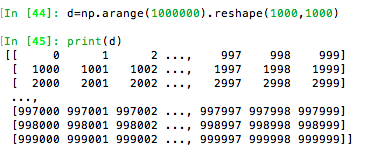


图26 打印二维数组

## 2 数组的基本操作

数组的操作非常灵活，允许数组与数组之间的运算，如图27~29所示；以及数组与某一个数字之间的运算，如图30和图31所示。进行算术运算的时候，Numpy会创建一个新的数组，并把结果写入这个新的数组中。

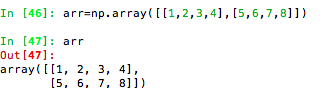


图27 生成arr数组

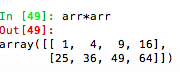
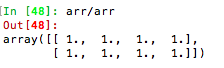
 

图28 数组间的乘法运算 图29 数组间的除法运算

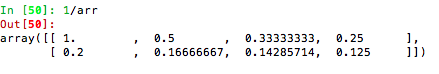


图30 数组与数间的除法运算

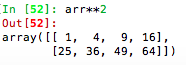


图31 数组与数间的乘法运算

有的运算符号+=、／=等。这类运算符号，并不会创建一个新的数组，而是直接对原来数组的元素值进行修改，如图32所示。

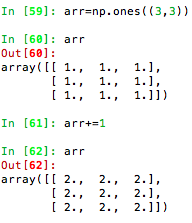


图32 数组+=运算符

这一类运算符，初学者容易误用，例如，a数组的数据类型是整型数，b数组的数据的数据类型是浮点数。假设a、b数组相加，并把值赋给a数组,这时会引发异常，如图33所示。

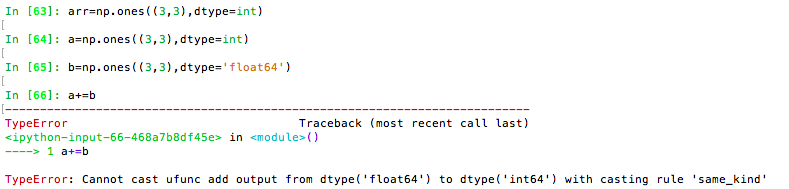


图33 两个不同类型的数组相加

现在，假设a、b数组相加，并把值赋给b数组，看下会是什么结果，如图34所示。

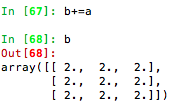


图34 两个不同类型的数组相加

为什么第二种，会得到正确的结果呢？

因为a、b数组相加，Numpy首先把a数组的元素的数据类型转换成b数组相同的数据类型。这里有一个转换原则，就是向精度更高的数据类型转换。a、b相加后，得到的数组数据类型是浮点数。而a数组的数据类型是整型数，Numpy不会把精度高的数转换成精度低的数，所以会引发异常；b数组的数据类型本身就是浮点数，所以可以把运算结果，赋给b数组。不同大小的数组之间也可以进行运算，后面会介绍。

## 3 基本的分片和索引操作

数组可以进行索引和分片操作，这可以非常方便地提取数组的子集或是某一个元素。一维数组可以进行索引、分片、迭代等操作,和对列表的操作方式相似，如图35所示。

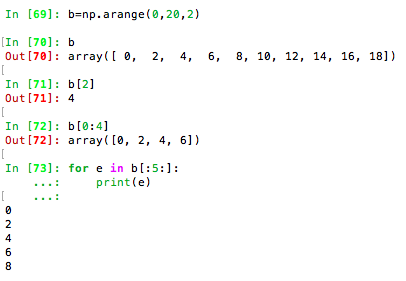


图35 一维数组的基本操作

数组经过分片后，分片得到的数组仍然指向原数组，并没有生成一个新的数组。

注意：数组分片和列表分片的区别

例如，分片出来的数组重新赋值，原数组的值也会改变，如图36所示。之所以不直接生成一个新的数组主要原因是为了节约内存资源；可以想象一下，对于一个很大的数据集，如果每次切片操作都生成一个新的数组，对内存的消耗有多严重。

屏幕快照%202017-06-29%20下午3.31.50.png

图36 数组的分片赋值操作

注意：这里是以2为间隔进行切片操作。

多维数组的索引、切片操作更加丰富，难度也更大。如果要取出多维数组内的某一个元素，和一维数组的语法有差别，如图37所示。二维数组可以看作一张表，行、列的起始位置都是从0开始，如图38所示。

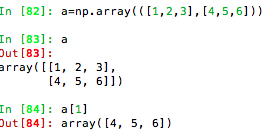


图37 二维数组取值操作



图38 二维数组

在二维数组中，要对某一个元素进行索引，需要在两个轴向上定位元素，如图39所示。

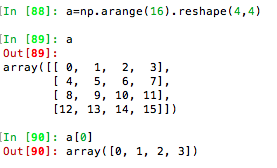


图39 二维数组的索引操作

例如，现在需要取出15这个元素，有两种方式，如图40所示

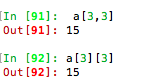


图40 二维数组的取值操作

第一种索引方式更加简洁。

三维数组类似。要取出某一个元素，必须在三个方向上定位;如果只在一个方向上索引，会得到一个二维数组，如图41所示。

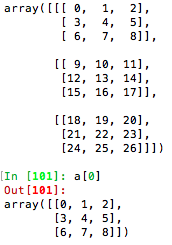


图41 三维数组的取值操作

这里得到的是一个3\*3的二维数组。可以把一个数赋给取出的这个二维数组，也可以把一个形状一样的二维数组赋值给它，如图42所示。

首先把复制 a[0] 的原始值并保存到old\_value这个变量中，并把10赋值给a[0]

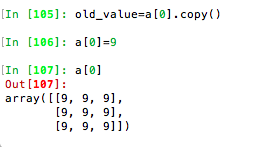


图42 赋值操作

再把old\_value保存的二维数组值赋给a[0]，如图43所示。

和一维数组一样，多维数组也可以进行切片操作，如图44所示。

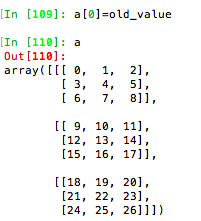
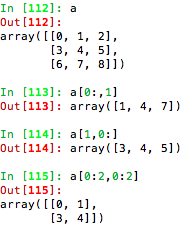
 

图43 赋值操作 图44 多维数组切片

从上面这几个切片操作可以看出来，对多维数组进行切片操作，需要分别定义各个轴上的切片范围。

注意：方括号内从左向右依次表示0轴、1轴依次类推，参考图38。

如果要切整个轴，可以只使用一个冒号，如图45所示。

屏幕快照%202017-06-29%20下午4.53.38.png

图45 使用冒号进行切片操作

除了以上的索引方式，还可以使用负值切片，如图46。

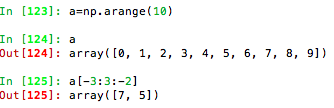


图46 负值切片

基本语法是 a[i:J:k]，如果i、j于0那么i、j就被解释成n+I、n+j。

注意：n表示进行切片操作的这个轴的长度

k小于0表示由大到小，反向切片。再来看一下上面这个例子，i<0,那么i就被解释7，k=-2,会依次取出索引值为7、5的元素。所以结果切片结果是：

array([7,5])

Newaxis对象可以在切片操作中，新建一个长度为1的轴，如图47所示。

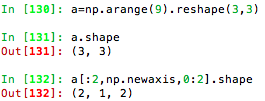


图47 Newaxis对象

## 4 高级索引

当方括号内的索引方式不再是之前介绍的简单序列的时候，会触发高级索引；高级索引包括整数索引和布尔索引。

注意：与常规索引切片不同的是高级索引会复制原数据，并返回一个新的数组。

### 4.1 整数索引

对于一维数组而言，使用数组进行索引，返回的结果，和索引数组的形状一样;不同的地方是被索引数组的相关的值会替代索引数组对应位置的值。下面索引数组的形状是2\*2，返回的结果也是2\*2的数组；由于被索引数组在1、2、3、4的位置上的值也正好是1、2、3、4所以返回的结果和索引数组一模一样，如图48所示。

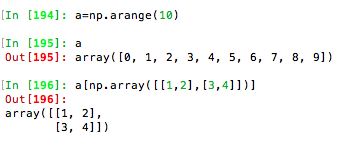


图48 整数索引

每一个整数数组代表那一个维度上的索引值。

假设现在有一个4\*的二维数组，现在需要选出其对角线上的元素，如图49所示。首先选出各行，行号包括[0,1,2,3]；然后再从各行里面来选择相应的对角线上元素的列号，也正好是[0,1,2,3]。

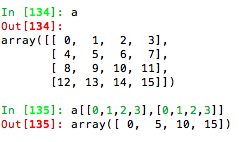


图49 使用整数索引取出数组对角线上的元素

可以在索引数组内重新排列各行的行号，来定义索引的顺序，如图50所示。也可以使用负值来索引数组，如图51所示。

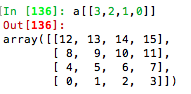
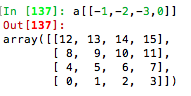
 

图50 整数索引 图51 负值整数索引

### 4.2 布尔索引

布尔索引是另外一种高级索引方式。

当索引的方式是布尔数组的时候，会触发布尔索引。首先使用random下的rand函数生成一个10\*4的均匀分布的二维数组；然后定义一个长度为10的一维数组，数组元素是一些人名，每行数据和人名一一对应，如图52所示。

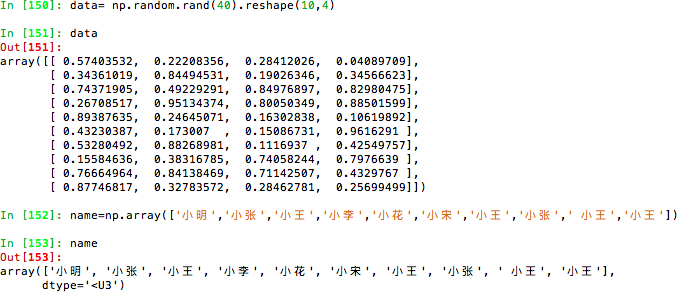


图52 生成数组数据

假设要选出所有与小王相关的数据，如图53所示。

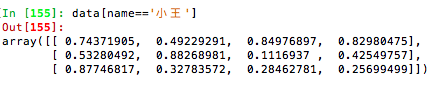


图53 布尔索引

name==’小王’

会生成一个布尔数组，凡是name这个数组里面元素值是‘小王’的，布尔值为True， 其余的则为False，如图54所示。

屏幕快照%202017-06-30%20下午4.07.22.png

图54 ==运算符

在布尔索引的同时也可以进行切片操作，如图55所示。

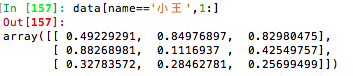


图55 布尔索引和切片操作

还可以和之前介绍的整数索引结合起来一起使用，如图56所示。

屏幕快照%202017-06-30%20下午4.13.08.png

图56 布尔索引和整数索引

可以使用！=(不等于)取出除了小王以为的其他数据，如图57所示。

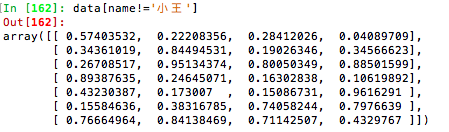


图57 布尔索引

还可以使用逻辑运算符号，&(且),|(或则)，~(非)连接多个逻辑表达式，进行更加复杂的布尔索引操作。

例如，上面使用！=比较运算符，选出了不属于小王的数据，也可以使用逻辑运算符得到相同的结果，如图58所示。

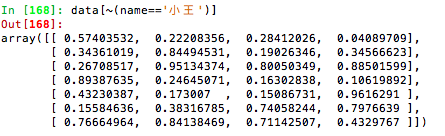


图58 布尔索引

要取出属于小王和小张的数据，如图59所示。

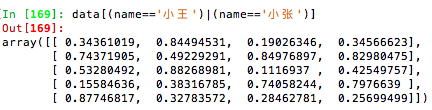


图59 布尔索引

### 4.3 布尔索引的简单应用

在进行数据分析的时候，经常会剔除掉数据集内的缺省值。定义一个带有nan( Numpy中代表无的意思)的二维数组，并使用isnan函数筛选出值为nan的数组元素，并且剔除掉它们，如图60所示。

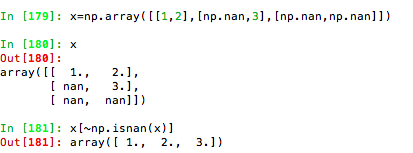


图60 处理带缺省值的二维数组

还可以对数组内的某些特殊元素进行修改，例如，要对数组内元素值小于1的元素加上一个值，如图61所示。

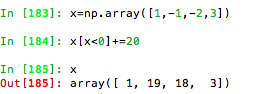


图61 增加数组值

obj.nonzero 可以用来生成数组元素值非0的整数数组索引，如图62所示。

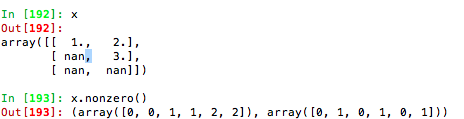


图62 nonzero的使用

布尔索引数组必须要和被索引数组对应轴上的长度一样，否则会引发异常。因为布尔索引数组只有True和False两个值，如果和要索引轴上的长度不一样的话，Numpy不知道要对哪个位置的元素进行操作。

## 5 改变数组的形状

数组的形状可以被改变，使得元素重新排列，但是并不改变元素本身。例如，之前使用的reshape，如图63所示，输入参数是表示数组形状的序列。

在reshape的操作中，如果某一个方向上的大小为-1，那么会自动根据其他方向上的长度来计算其长度，如图64所示。

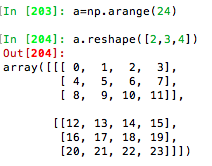
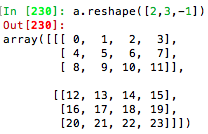
 

图63 reshape的使用 图64 数组的reshape方法

在线性代数中，经常会对矩阵进行转置，数组也有一个特殊的T属性，做类似矩阵的转置操作，如图65所示。

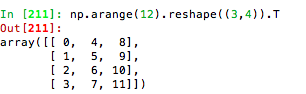


图65 数组的转置操作

对于维度大于2的数组，可以使用transpose函数来对数组进行重新排列。通过排列各个轴的轴号，实现数组结构的改变，如图66所示。

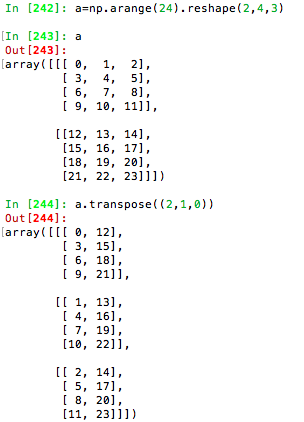


图66 数组的transpose方法

还可以使用ravel，改变数组的形状；逐行读取元素并返回一个一维数组，如图67所示。

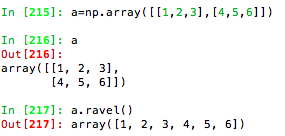


图67 数组的ravel方法

还可以使用resize改变数组的形状，与reshape不同的是，它是直接对原数组进行修改，如图68所示。

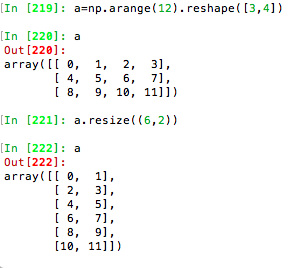


图68 数组的resize方法

## 6 组装、分割数组

几个数组可以在不同的方向上，把它们组装起来，例如，两个二维数组，可以在垂向上把它们组装起来，如图69所示。

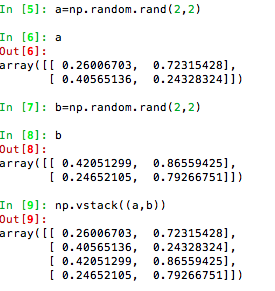


图69 数组垂向上的组装

也可以在横向上把两个数组组装起来，如图70所示。

屏幕快照%202017-07-01%20下午5.11.02.png

图70 数组横向上的组装

反过来也可以对数组进行切割操作，例如，在横向上切割；通过在第2和第4列对数组进行切割，得到三个数组，三个数组在横轴上的长度分别为2、2、4，如图71所示。

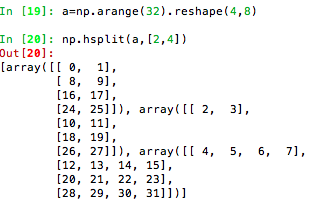


图71 数组的切割操作

## 7 数组的基本函数

Numpy提供了一些基本的函数，可以非常方便地对数组内的元素进行运算,运算结果一般也是数组。例如，指数exp、sin、add等函数，如图72所示。

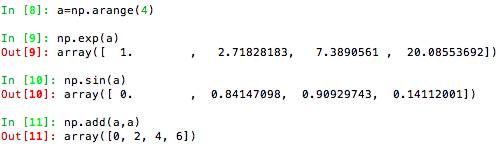


图72 数组的基本函数

还有一些比较常用的函数argmax，返回某一个轴上的最大值的位置。如果没有定义轴的方向，数组自动扩展成为一维数组，然后返回最大元素的索引值，如图73所示。

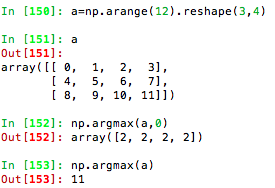


图73 数组的argmax方法

在线形代数的运算中，经常进行矩阵乘法运算；Numpy提供的dot函数，可以对两个形状匹配的二维数组进行矩阵乘法运算，如图74所示。

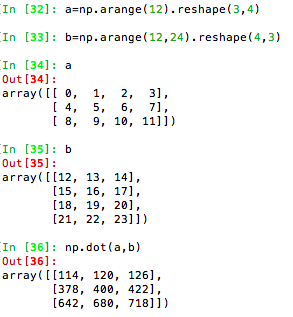


图74 矩阵的乘法运算

表3列出了一些常用的数组函数，包括数组的算术运算、三角函数、比较运算、浮点函数4类。

表3 数组的常用函数

|  |  |
| --- | --- |
| add | 两个数组元素相加 |
| subtract | 两个数组元素相减 |
| multiply | 两个数组元素相乘 |
| divide | 两个数组元素相除 |
| power | 第一个数组的元素作为底，第二个数组元素作为指数，进行幂指数运算 |
| mod、remainder | 两个数组元素相除并返回结果的余数 |
| absolute、fabs | 返回数组元素的绝对值 |
| exp、exp2 | 返回输入数组以e和2为底的指数结果 |
| log、log2、log10 | 返回自然对数，或者以2为底的对数或以10为底的对数 |
| sqrt | 返回数组元素的平方根 |
| square | 返回数组元素的平方 |
| sin、cos、tan | 返回数组元素的正弦值、余弦值或正切值 |
| arcsin、arccos、arctan | 返回数组元素的反正弦值、反余弦值或反正切值 |
| greater(x1,x2) | 如果x1>x2，则返回布尔真值 |
| greater\_equal(x1,x2) | 如果x1>=x2，则返回布尔真值 |
| less(x1,x2) | 如果x1<x2，则返回布尔真值 |
| less\_equal(x1,x2) | 如果x1<=x2，则返回布尔真值 |
| not\_equal | x1不等于x2,则返回布尔真值 |
| equal | x1等于x2,则返回布尔真值 |
| floor | 向下取整运算 |
| ceil | 向上取整运算 |
| trunc | 截掉小数点后面的数 |

## 8 复制和指代

前面介绍了众多与数组相关的函数，有的函数是先对原数组复制过后，再进行操作；有的是直接对原数组进行修改，然后返回一个对原数组的指代(view)。这个地方，往往使初学者感到迷惑。一般有三种情况：完全不复制、返回指代或深度复制。

1. **完全不复制**

赋值操作，属于完全不复制数组的一类操作，如图75所示。

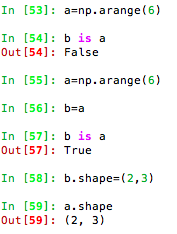


图75 不复制数组的操作

1. **指代**

不同的变量可以指向同一个数组。数组的view方法，可以生成一个数组对象，指代同一个数组，如图76所示。切片操作也是返回一个对原数组的指代，如图77所示。

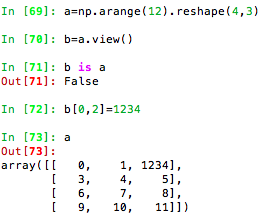
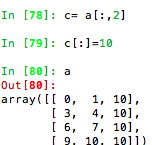
 

图76 返回数组指代的操作 图77 返回数组指代的操作

完全不复制和返回一个指代二者的共同点是都没有复制原数组。

1. **深度复制**

copy方法复制原数组及里面的原数组，通俗一点来说的话，就是原数组和新的数组没有任何关系，任何对新数组的操作，不会影响到原数组，如图78所示。

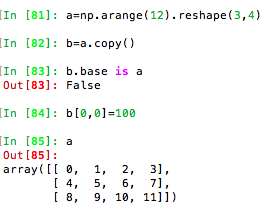


图78 深度复制数组的操作

## 10 线性代数

Numpy可用于计算矩阵相乘、分解矩阵、求解线性方程等线性代数问题。之前已经介绍过可以使用dot函数求解矩阵相乘.使用Matlab的用户，需要注意一下，不能使用\*这个运算直接计算两个矩阵的乘积；因为\*在Numpy中是使得两个矩阵的元素相乘，如图79所示。

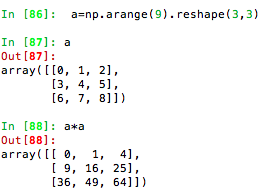


图79 数组相乘

numpy.linalg 有矩阵计算经常会用到的功能，例如，inv函数，求解矩阵的反矩阵，如图80所示。首先导入这个模块：

import numpy.linalg as LA

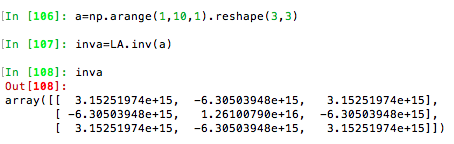


图80 求解矩阵的反矩阵

可以使用solve函数求解下面这个线性方程组，如图81所示。

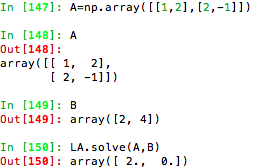


图81 求解线性方程组

读者可以分别把代入线性方程检验，看结果是否正确。

使用eig函数求解一个平方矩阵的特征值、和特征向量，如图82所示。

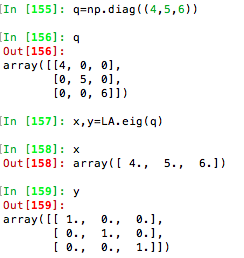


图82 求解矩阵的特征值和特征向量

这里q是一个对角线上元素为4、5、6的对角矩阵；使用eig求解该矩阵后，同时返回该矩阵的特征值和对应的特征向量。感兴趣的读者，可以自己检验下结果的正确性。

## 10 使用数组来处理数据

使用Numpy数组处理数据的时候，可以避免重复的写循环语句；使用Numpy表达式可以节约大量的编程时间，把更多的精力投入到数据处理分析本身上，而不需要去关注烦琐的细节问题。

首先使用Numpy的mgrid函数，创建一个多维的网格；mgird的输入参数是与所建的网格相关的序列值，返回值是与网格坐标相关的两个数组；两个数组的组合代表了序列值可能生成的所有坐标对，如图83所示。

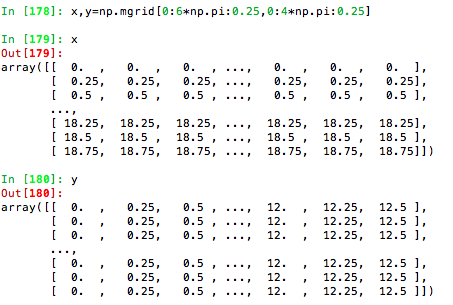


图83 使用mgrid创建坐标位置

然后使用之前介绍的Numpy基本函数，根据x、y坐标计算Z值，如图84所示。

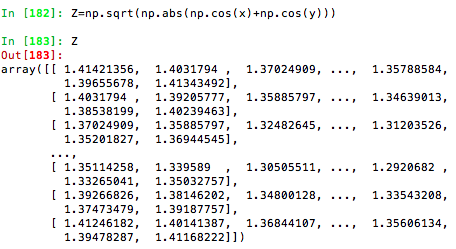


图84 计算x、y的函数值

最后使用matplolib绘制Z=f(x,y)的关系，如图85和图86所示。

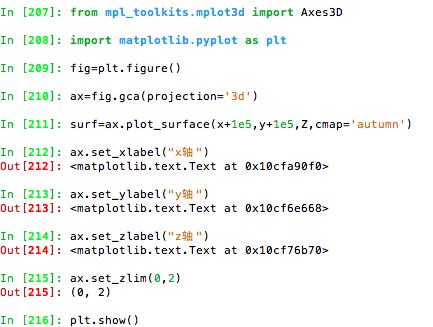


图85 绘制3D图

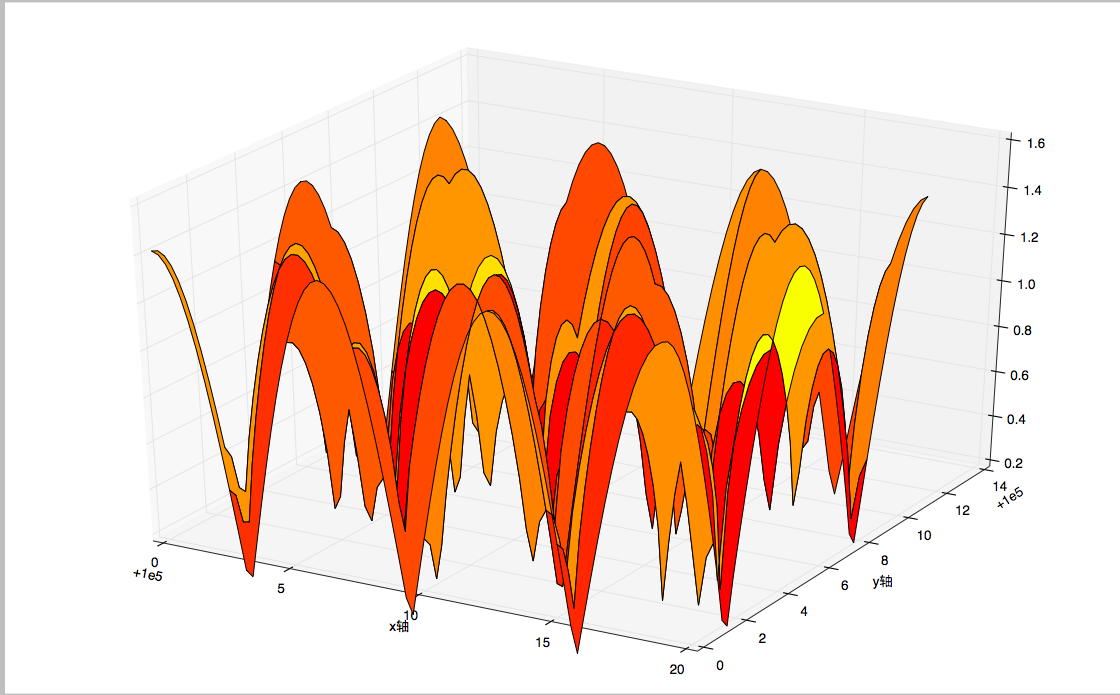


图86 Z=f(x,y) 的函数关系图

注意：现在不熟悉matplotlib也没关系，后面会详细介绍到。

## 11 Numpy的where函数和统计函数

筛选满足数组元素是数据处理经常会遇到到的任务，Numpy为了避免用户重复写代码，提供了where函数，使得用户可以非常简洁的筛选数据。数据的统计分析离不开类似于求和、计算方差等基本函数，通过对这些函数基本函数的掌握，用户可以去实现更加复杂的逻辑与应用。

### 11.1 where函数

where函数的语法是where(condition,[x,y])。使用这个函数可以方便地进行数组在一定条件下的筛选。例如，现在有三个数组a、b、c，其中c是bool数组。现在要在bool数值元素为True时，选择a对应位置元素到d数组;否则，选择b对应位置的元素到d，如图87所示。

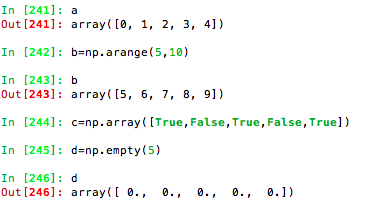


图87 生产数组

常规的做法可能是使用for循环来筛选相应的元素，如图88所示：

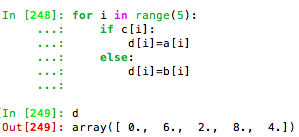


图88 筛选元素

可以看出这种方法还是比较麻烦的，可以使用where函数简洁的得到一样的结果，如图89所示。

屏幕快照%202017-07-02%20下午7.17.37.png

图89 where函数

where的第二个和第三个参数也可以不是数组，如图90所示。

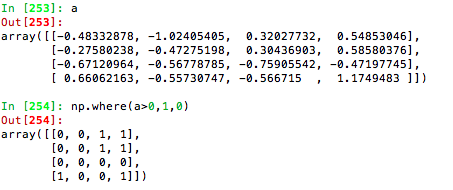


图90 where函数

这里的操作是，凡是数组a内大于0的值，设定1，小于或等于0的时候，设定为0。

### 11.2 统计函数

统计函数是Numpy提供的数据分析的基础函数，在处理数据的时候会经常用到。

sum函数计算整个数组元素的和，或则在某一个轴向上元素的和，如图91和图92所示。

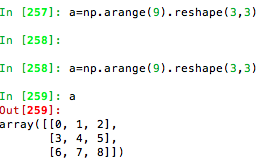


图91 生成数组a

屏幕快照%202017-07-02%20下午7.33.06.png 屏幕快照%202017-07-02%20下午7.33.27.png

图92 sum函数

mean函数计算整个数组元素的平均值，或则某一个轴向上的平均值，如图93所示。

屏幕快照%202017-07-02%20下午7.36.52.png 屏幕快照%202017-07-02%20下午7.36.58.png

图93 mean函数

min、max函数返回数组元素的最大值、最小值或某一个方向元素的最大、最小值，如图94和图95所示。

屏幕快照%202017-07-02%20下午7.40.43.png 屏幕快照%202017-07-02%20下午7.42.01.png

图94 min函数

屏幕快照%202017-07-02%20下午7.42.14.png 屏幕快照%202017-07-02%20下午7.41.41.png

图95 max函数

计算数组元素的标准差、方差，如图96所示。

屏幕快照%202017-07-02%20下午7.45.00.png 屏幕快照%202017-07-02%20下午7.45.07.png

图96 std、var函数

最大最小元素的索引值，如图97所示。

屏幕快照%202017-07-02%20下午7.50.21.png 屏幕快照%202017-07-02%20下午7.50.41.png

图97 argmin、argmax函数

计算数组元素的累和，例如，分别计算三列元素的累积和，如图98所示。

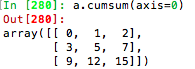


图98 cumsum函数

## 12 输入与输出

Numpy数组可以写入文件中，并保存下来；反过来也可以从文件中读取数据，生成Numpy数组。常用的文件格式包括：二进制文件和普通文本文件。

### 12.1 二进制文件

Numpy提供了load、save、savez、savez\_compressed这四个函数来处理二进制文件。其中save函数默认情况下以.npy为扩展名的二进制文件保存一个Numpy数组，如图99所示。

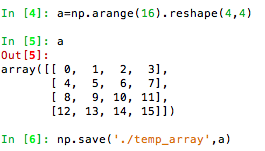


图99 save函数

使用load函数导入保存的.npy文件，如图100所示。

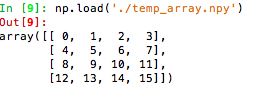


图100 load函数

也可以使用savez函数将多个数组以非压缩的方式写入同一个二进制文件中，如图101所示。

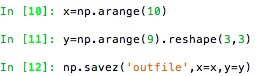


图101 savez函数

如果不对写入文件的数组命名，Numpy会以array\_0、array\_1…对写入的数组命名。同样可以使用load函数读取文件内的数组，并返回字典形式的数组,如图102所示。

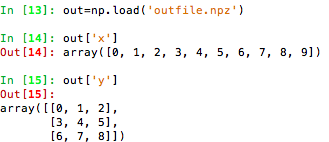


图102 写入多个数组

如果不知道数组名字，可以使用files属性查看文件内保存的所有数组名。使用savez\_compressed函数可以把多个数组以压缩的格式保存到二进制文件中。

### 12.2 文本文件

操作文本文件是数据处理会经常遇到的任务，下面介绍下savetxt函数和loadtxt函数。

和save函数相比，savetxt函数可以通过dilimiter参数定义数组元素在文本文件中的分隔符号。例如，使用“，”来分隔数组元素在文本文件中的保存方式，如图103所示。

屏幕快照%202017-07-10%20下午10

图103 savetxt函数

使用文本编辑器打开out.txt，可以看到用逗号分隔开的数字(CSV格式的文本)

注意：fmt参数可以用来定义元素写入文件的格式

0,1,2,3

4,5,6,7

8,10,10,11

12,13,14,15

使用loadtxt函数读取文本文件内的数组，如图104所示。

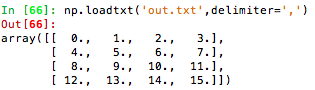


图104 loadtxt函数

## 13 生成随机数

Numpy.random模块下提供了一些常用的随机生成函数，可以生成不同概率分布的随机数，例如，之前使用过的rand函数，生成[0,1]区间上满足均匀分布的随机数的数组，如图所示105所示。

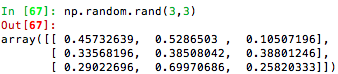


图105 rand函数

表4列出了一些常见的随机函数。

表4 随机函数

|  |  |
| --- | --- |
| rand(d0,d1,…dn) | 返回均匀分布的随机数 |
| randn(d0,d1,…,dn) | 返回标准分布的随机数 |
| randint(low,[,high,size,dtype]) | 返回给定开区间上的随机整数 |
| random\_intergers | 返回给定闭区间上的随机整数 |
| random | 返回[0,1]区间上的浮点数 |
| choice | 返回给定的一维数组的随机数 |
| Bytes | 返回随机的位置 |

下面介绍一个使用randint随机函数来模拟掷骰子的例子。

假设有两颗骰子，每次试验同时掷下两颗骰子，并记录下两颗骰子的总和。重复上面这个试验10000次，并分析点数和在[2,12]区间上的分布情况。使用randint函数生成试验所需要的结果，如图106所示。

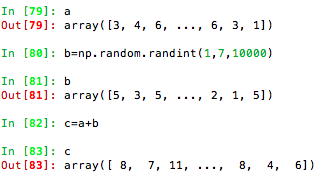
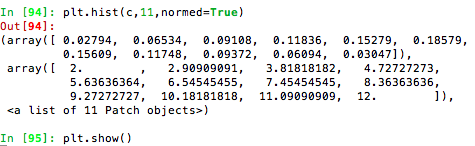


图106 生成试验所需要的结果

使用matplotlib绘制,两次掷骰子点数和在[2,12]区间上的概率分布直方图，如图107所示。



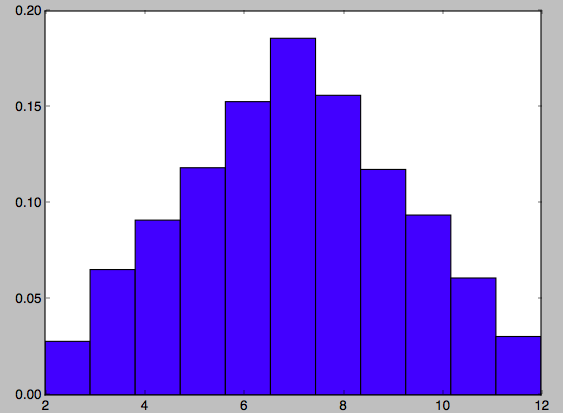


图107 试验结果的概率分布直方图

## 14 数组的排序和查找

排序和查找是数据分析和处理经常会遇到的任务。

### 14.1 排序

Numpy提供了sort函数，复制数组并返回排序结果，如图108所示。

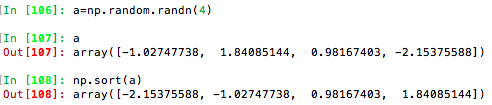


图108 sort函数

如果是多维数组，可以设置axis参数定义需要排序的轴，sort函数会对这个轴向上的所有元素排序，如图109所示。

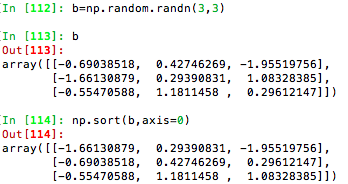


图109 sort函数

sort函数的keyword参数可以定义使用排序的关键字。现在有小明、小张、小王、小李4个同学的身高和体重数据，如图110所示。

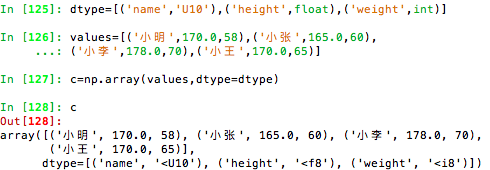


图110 四个同学的身高和体重数据

首先以身高这个字段排序，如果身高一样，再以体重这个字段对这几人排序,排序结果如图111所示。

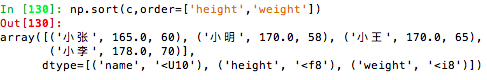


图111 排序结果

数组对象也有一个sort方法，与Numpy的顶层的sort函数的区别于ndarray.sort是对数组进行原地排序。

除了sort一类的直接排序函数以外，Numpy还提供了间接排序函数，返回排序结果的索引值，如图112所示。

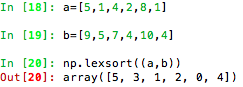


图112 lexsort函数

这里使用lexsort函数对a、b两列数排序，lexsort函数的输入参数可以是一个k\*n行的数组也可以k\*n的有序数字，本例输入的是2个长度为5的列表，返回排序结果的索引值。读者可能会对这个排序方法感到迷惑，下面讲下为什么排序结果第一个索引值是5。

该函数默认的是从最后一行、倒数第二行依次进行排序。例如，本例首先对b列表排序，该行最小的数是4，索引值是3和5；然后再根据a列表里的数字排序，索引位置3和5对应的数是2和1，所以索引位置5应该排在3的前面。

### 14.2 查找

searchsorted是查找数组元素会经常用到的函数。查找一个合适的位置将元素插入到已排好序1-D数组中，并且保持顺序不被打乱，如图113所示。

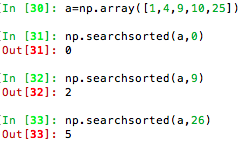


图113 searchsorted函数

除了可以输入某个数以外，还可以输入一个序列，如图114所示

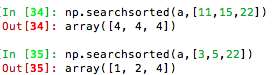


图114 searchsorted函数

注意：这里返回的是一个数组，对应着序列中数字的合适位置。

searchsorted 还有一个side参数，默认情况下，该参数的默认值是“left”,假设数组有多个位置与查找的数字匹配，会返回第一个位置；如果设置为“right”，会返回最后一个位置，如图115所示。

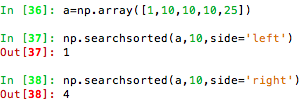


图115 searchsorted函数

extract函数可以用来提取满足一定条件的元素，例如，提取出0、1…16中不能被4整除的数，如图116所示。

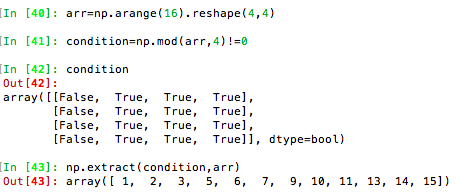


图116 extract函数

扩充转换(Broadcasting)

Numpy的基础函数都是对输入数组的单个元素进行运算后，输出结果数组；标准的扩宽转换的目的是，即使各个输入数组的形状不完全一样，也能进行运算。扩宽转换属于Numpy的高级内容，即使是经验丰富的使用者也会感到迷惑不已。扩充转换遵循下面这四条原则：

* 除维度最大的输入数组以外，其余的所有输入数组的维度向输入数组中维度最大的数组靠齐，并把那个维度考虑为1。
* 输出数组在各个维度上的长度和所有输入数组在那个维度上的最大长度一样
* 如果输入数组在某个维度上为1或和输出数组在那个维度上的长度一样，会被用于实际的计算过程
* 如果输入数组在某一个维度上的长度为1，那么这个维度上的第一个元素会被用于后续在该维度上的计算

最简单的例子就是一个数组和一个纯数间的运算，如图117所示。

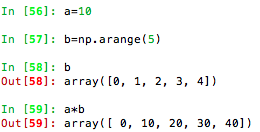


图117 数组和纯数间的运算

这里可以把a的形状看作(5, )。例如，一个4\*3的数组和一个(3, ) 的数组相加，如图118所示。

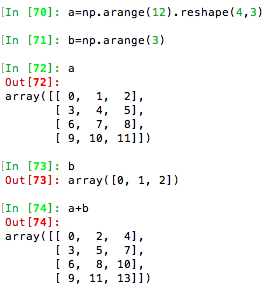


图118 数组间的加法

b数组扩充后和a数组的形状一样，然后再进行加法运算，如图119所示。

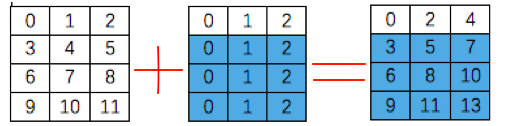


图119 对b数组的行进行扩充

如果b 是一个(4,1)的数组，运算结果如图120所示。

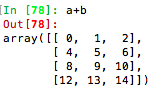
 

图120 数组间的加法

这里b数组的扩充方式和之前有所不同，是对其列进行扩充，如图121所示。

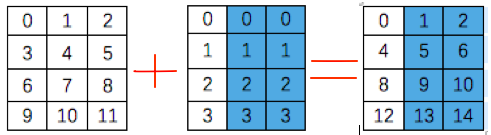


图121 对b数组的行进行扩充

三维甚至更高维度的数组的扩充转换，更加复杂，这里就不介绍了，有兴趣的读者可以自己去探索。