# 1.基础

NumPy的主要对象是同构的多维数组。它是一个元素表（通常是数字），都是相同的类型，由正整数元组索引。在NumPy维度中称为轴。轴数是等级。

例如，3D空间[1,2,1]中的点的坐标是等级1的数组，因为它具有一个轴。该轴的长度为3.在下图所示的示例中，数组具有等级2（它是2维的）。第一维（轴）的长度为2，第二维的长度为3。

[[ 1., 0., 0.],

[ 0., 1., 2.]]

NumPy的数组类称为ndarray。 它的别名也叫数组。请注意，numpy.array与标准Python库类array.array不同，python库中它只处理一维数组并提供较少的功能。ndarray对象的更重要的属性是：

* ndarray.ndim

数组的轴数（维度）。 在Python世界中，维度的数量称为排名。

* ndarray.shape

数组的大小。 这是一个整数元组，表示每个维度中数组的大小。对于具有n行和m列 的矩阵，形状将为（n，m）。 因此，形状元组的长度是等级或维数ndim。

* ndarray.size

数组的元素总数。 这等于形状(shape)元素的乘积。

* ndarray.dtype

描述数组中元素类型的对象。 可以使用标准Python类型创建或指定dtype。 此外， NumPy还提供自己的类型。 例如numpy.int32，numpy.int16和numpy.float64。

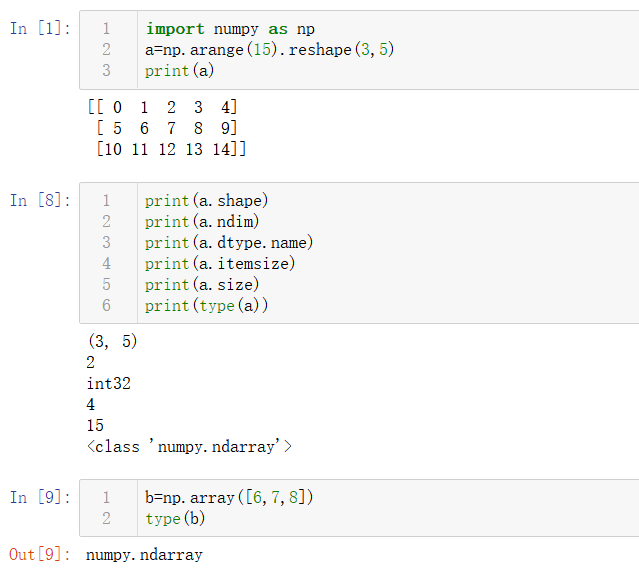
* ndarray.itemsize

数组中每个元素的大小（以字节为单位）。例如，float64类型的元素数组具有itemsize 8 （= 64/8），而complex32类型之一具有itemsize 4（= 32/8）。 它相当于 ndarray.dtype.itemsize。

* ndarray.data

包含数组实际元素的缓冲区。 通常，我们不需要使用此属性，因为我们将使用索引工 具访问数组中的元素。

## 1.1.一个示例



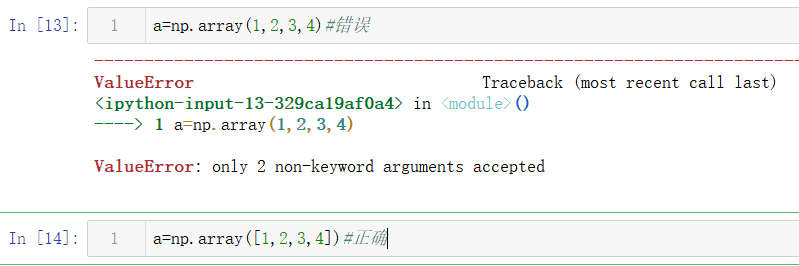
## 1.2.创建数组(array)

有几种方法可以创建数组。

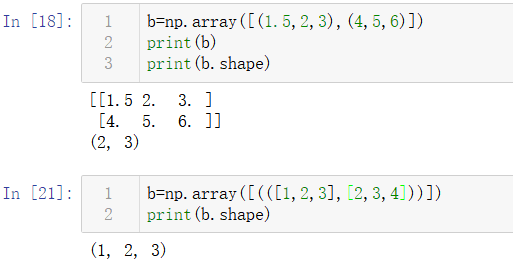
例如，您可以使用数组函数从常规Python列表或元组创建数组。 结果数组的类型是从序列中元素的类型推导出来的。

|  |
| --- |
| a=np.array([2,3,4])  print(a)  print(a.dtype)  b=np.array([1.2,3.5,5.1])  print(b.dtype)  输出:  [2 3 4]  int32  float64 |

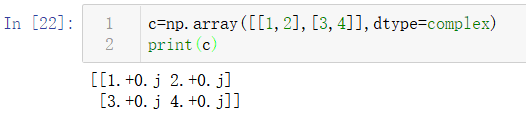
频繁的错误在于调用具有多个数字参数的数组，而不是提供单个数字列表作为参数。



array函数将序列的序列转换为二维数组，将序列的序列的序列转换为三维数组，等等。

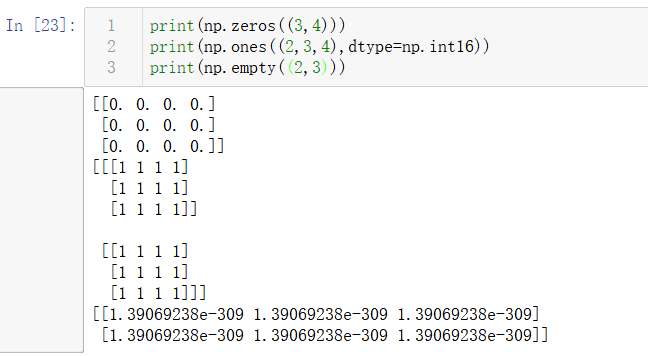


也可以在创建时显式指定数组的类型：

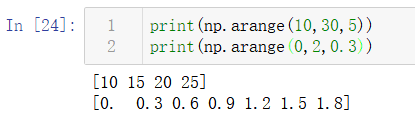


通常，数组的元素最初是未知的，但其大小是已知的。 因此，NumPy提供了几个函数来创建具有初始占位符内容的数组。 这些最小化了增长数组的必要性，这是一项昂贵的操作。

函数zeros创建一个充满0的数组，函数ones创建一个充满1的数组，函数empty创建一个空数组，其初始内容是随机的，取决于内存的状态。 默认情况下，创建的数组的dtype是float64。



为了创建数字序列，NumPy提供了一个类似于返回数组而不是列表的range的函数。



当arange与浮点参数一起使用时，由于有限的浮点精度，通常无法预测获得的元素数。出于这个原因，通常最好使用函数linspace接收我们想要的元素数作为参数：

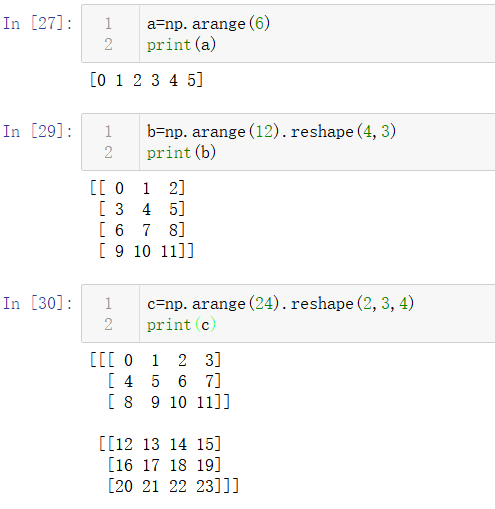
|  |
| --- |
| from numpy import pi  print(np.linspace(0,2,9))  x=np.linspace(0,2\*pi,100)  f=np.sin(x)  print(f) |

## 1.3 打印数组(Arrays)

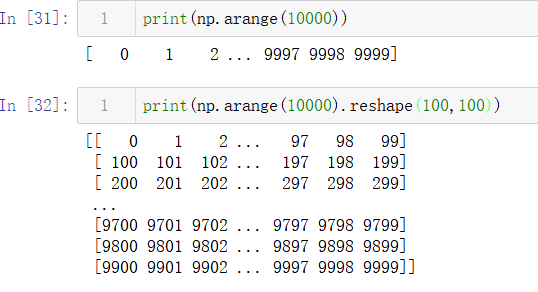
当您打印数组时，NumPy以与嵌套列表类似的方式显示它，但具有以下布局：

* 最后一个轴从左到右打印，
* 倒数第二个从上到下打印，
* 其余部分也从上到下打印，每个切片用空行分隔。

然后将一维数组打印为行，将二维数据打印为矩阵，将三维数据打印为矩阵列表。



如果数组太大而无法打印，NumPy会自动跳过数组的中心部分并仅打印角落：

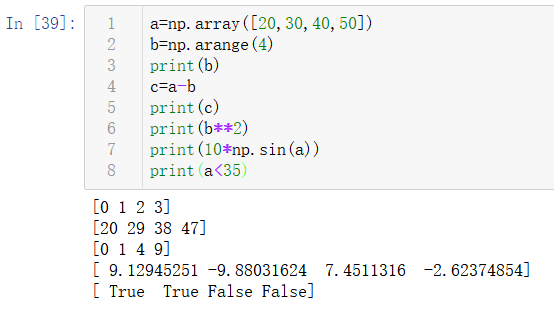


要禁用此行为并强制NumPy打印整个阵列，可以使用set\_printoptions更改打印选项。

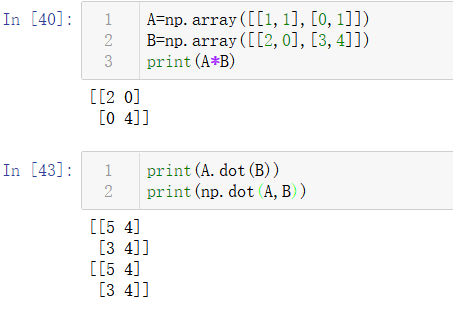
np.set\_printoptions(threshold=np.nan)

## 1.4 基本操作

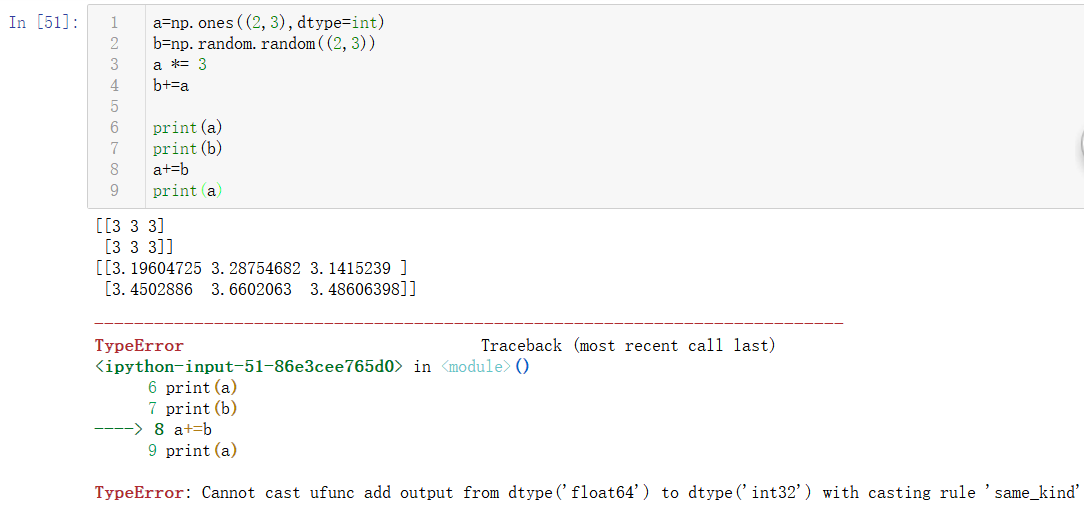
数组上的算术运算符应用于元素。 创建一个新数组并填充结果。



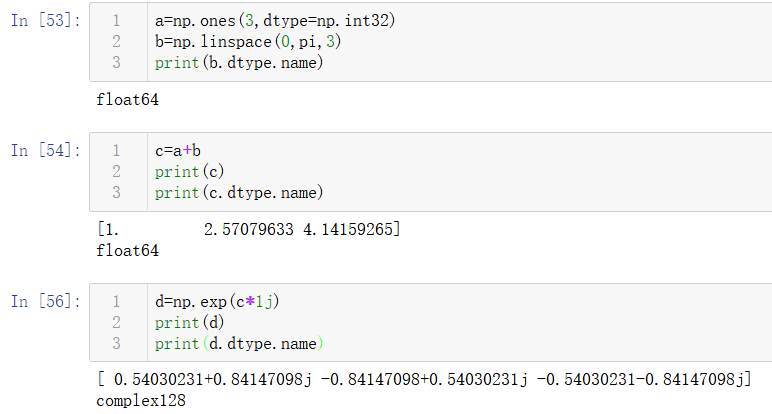
与许多矩阵语言不同，产品运算符\*在NumPy数组中以元素方式运行。 可以使用dot函数或方法执行矩阵乘积：



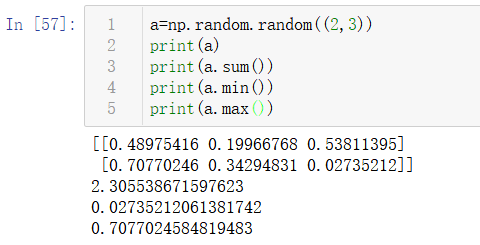
某些操作（例如+ =和\* =）用于修改现有阵列而不是创建新阵列。



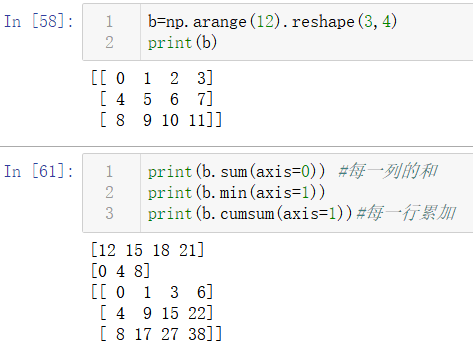
当使用不同类型的数组进行操作时，结果数组的类型对应于更一般或更精确的数组（称为向上转换的行为）。



许多一元操作，例如计算数组中所有元素的总和，都是作为ndarray类的方法实现的。

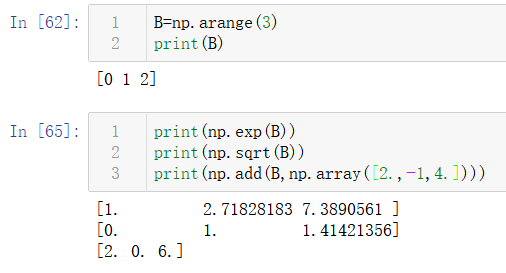


默认情况下，这些操作适用于数组，就像它是一个数字列表一样，无论其形状如何。 但是，通过指定axis参数，您可以沿数组的指定轴应用操作：



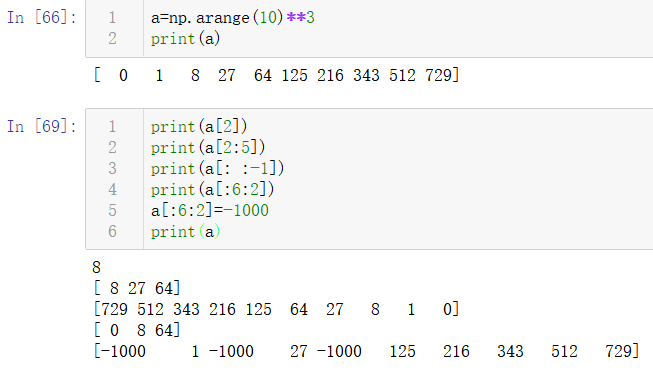
## 1.5 通用函数

NumPy提供熟悉的数学函数，例如sin，cos和exp。 在NumPy中，这些被称为“通用函数”（ufunc）。 在NumPy中，这些函数在数组上以元素方式运行，产生一个数组作为输出。

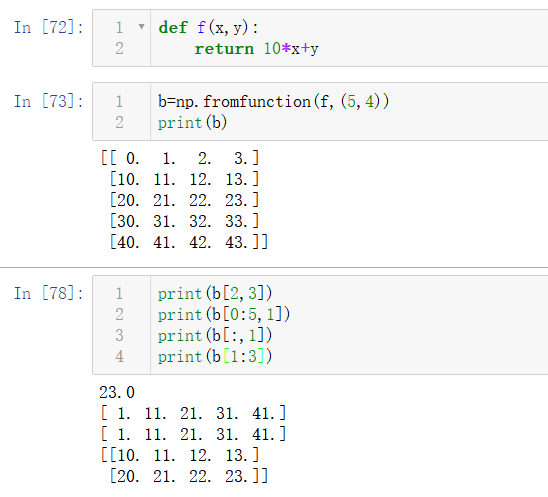


## 1.6 Indexing, Slicing and Iterating

一维数组可以被索引，切片和迭代，就像列表和其他Python序列一样

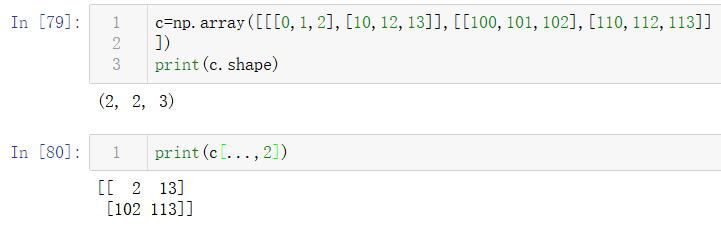


多维数组每个轴可以有一个索引。 这些索引以逗号分隔的元组给出：

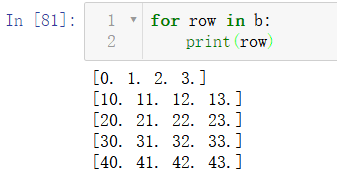


点（...）表示生成完整索引元组所需的冒号。 例如，如果x是秩5数组（即，它具有5个轴），那么

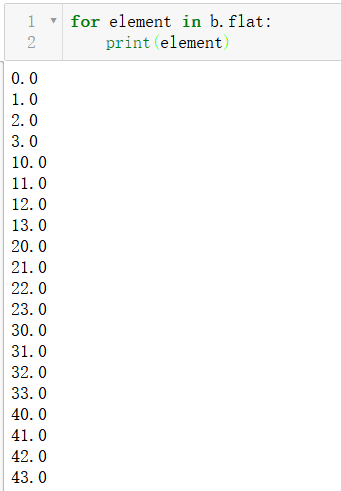
* x[1,2,...] 相当于 x[1,2,:,:,:],
* x[...,3] 相当于 x[:,:,:,:,3]
* x[4,...,5,:]相当于 x[4,:,:,5,:]



对多维数组进行迭代是针对第一个轴完成的：



但是，如果想要对数组中的每个元素执行操作，可以使用flat属性作为数组所有元素的迭代器：

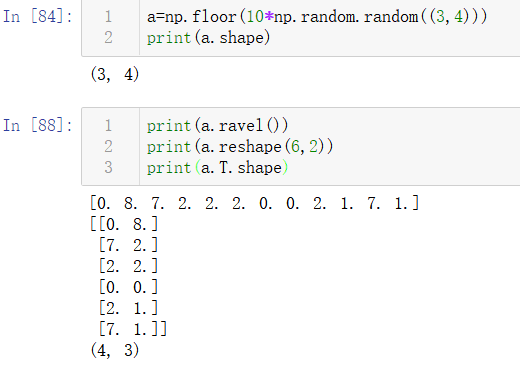


# 2.形状(shape)操作

## 2.1 更改数组的形状

数组的形状由沿每个轴的元素数量给出：

可以使用各种命令更改阵列的形状。 请注意，以下三个命令都返回已修改的数组，但不更改原始数组：



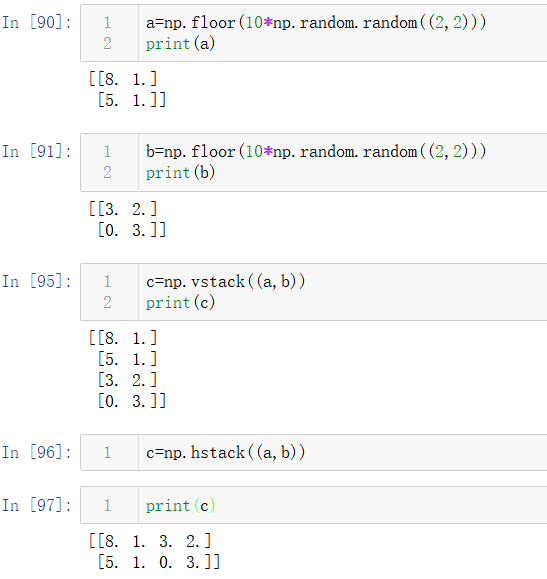
由ravel（）产生的数组中元素的顺序通常是“C风格”，也就是说，最右边的索引“变化最快”，因此[0,0]之后的元素是[0,1]。如果将数组重新整形为其他形状，则该数组将被视为“C风格”。NumPy通常会创建按此顺序存储的数组，因此ravel（）通常不需要复制其参数，但如果数组是通过获取另一个数组的切片或使用异常选项创建的，则可能需要复制它。函数ravel（）和reshape（）也可以使用可选参数指示使用FORTRAN样式的数组，其中最左边的索引变化最快。

reshape函数返回其参数的修改形状，而ndarray.resize方法修改数组本身：

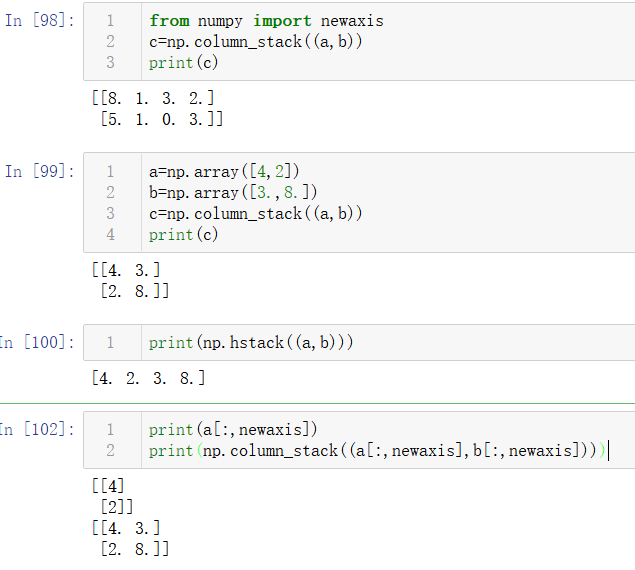
如果在重新整形操作中将维度指定为-1，则会自动计算其他维度：

## 2.2 堆叠不同的数组

几个数组可以沿不同的轴堆叠在一起：



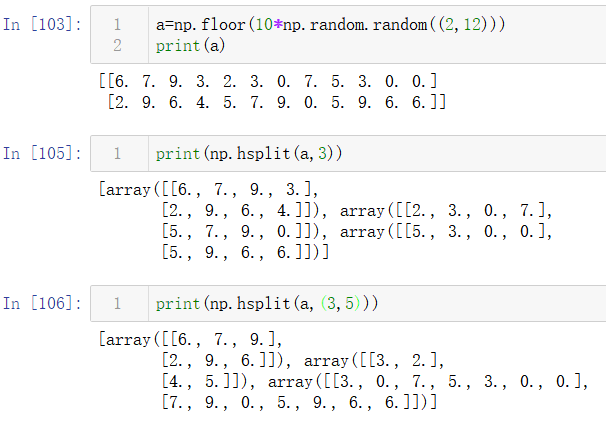
函数column\_stack将1D数组作为列堆叠到2D数组中。 它相当于仅针对2D数组的hstack：



另一方面，函数row\_stack等效于任何输入数组的vstack。 通常，对于具有两个以上维度的数组，hstack堆叠沿着它们的第二个轴，vstack堆叠沿着它们的第一个轴，并且concatenate允许可选参数给出连接应该发生的轴的数量。

## 2.3 将数组划分为几个更小的数组

使用hsplit，您可以沿着水平轴分割数组，方法是指定要返回的同形数组的数量，或者通过指定应该进行划分的列：



vsplit沿垂直轴分割，array\_split允许指定要分割的轴。