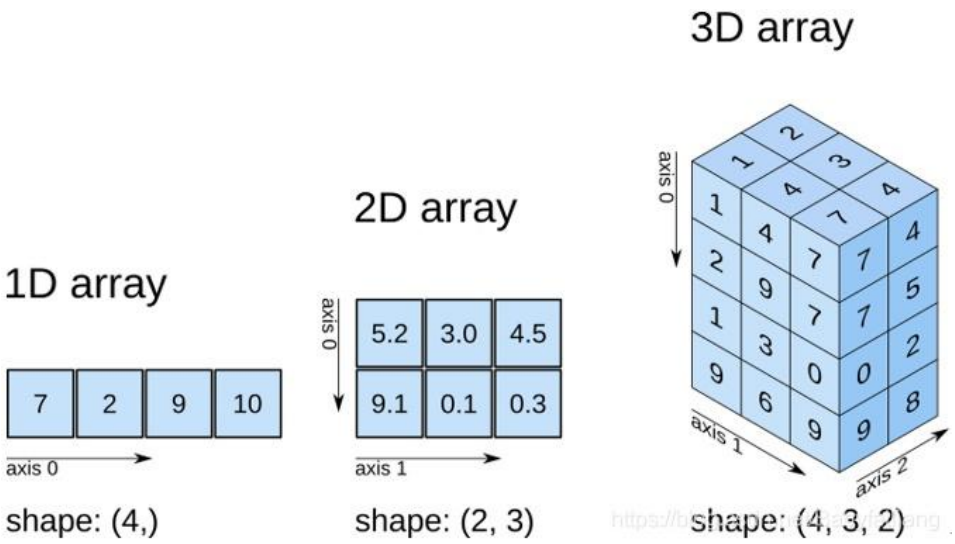
NumPy中的维度(dimension)、轴(axis)、秩(rank)的含义

转载自https://blog.csdn.net/Babyfatliang/article/details/87721282



NumPy 的时候，其中最重要的就是学习它的 ndarray 对象，它是多维度的同数据类型的数组。这个和Python自带的列表有较大的区别，列表中的
可以不相同的，如一个列表中，它可以包含数字、字符、字符串等，而在数组中，它的数据类型是相同的，如都是整型或者浮点型。

为什么Python中已经有了列表之后，在NumPy中还要引进一个 数组 对象呢？有以下三点可以作为参考，但在本文中不做具体描述：

1. 数组对象可以去掉元素间运算所需的循环，使一维向量更像单个数据
2. 设置专门的数组对象，经过优化，可以提升这类应用的运算速度
3. 数组对象采用相同的数据类型，有助于节省运算和存储空间

NumPy中有几个概念比较绕，对于我来说比较难理解，因此以此文作为记录。它们分别是： 维度 、轴、秩。

对于维度的介绍，官网是这么写的“ In NumPy dimensions are called axes”，即维度称为轴。为了更直观的理解，可以将其与现实世界联系起来，比
即二维的世界中，我们描述一个点的时候，通常使用 x 轴、y 轴，这样就能确定一个点的具体位置了。因此，这里的两个维度，也就跟两个轴对应了
是立体的三维世界中，我们就会多出一个z轴，以此更加准确的来反映点的位置。所以，我么可以把以上的维度和轴进行等价。

什么是秩(rank)？它是指轴的数量，或者维度的数量，是一个标量

在下面的例子中，有一个数组 [1,2,1]， 它的维度是1，也就是有一个轴，这个轴的长度是3，而它的秩也为1。这些信息，都可以通过NumPy提供的
获得。

> ndarray.ndim
> the number of axes (dimensions) of the array
> 秩，数组轴的数量，或者维度的数量
>
> ndarray.shape
> the dimensions of the array. This is a tuple of integers indicating the size of the array in each dimension. For a matrix with n rows and m colum
> will be (n,m). The length of the shape tuple is therefore the number of axes, ndim.
> 数组的维度。它的返回值是一个元组，这个元组描述了每个维度中数组的大小。相对于一个矩阵来说，shape表示的就是n行m列。这个元组的长
> 于轴/维度的个数，即秩的值

依旧以下图为例，我们已经知道数组a的维度数是1。查看它的shape，返回值为元组(3,) ，从这里也可以反映出这个数组的基本形状，即它是1维的，
空间中，拥有3个数据。

```
In [1]: import numpy as np

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

In [3]: print(a)
[1 2 1]

In [4]: print(a.ndim)
1

In [5]: print(a.shape)
(3,)

In [6]: a
Out[6]: array([1, 2, 1])
```

现在我们升级为一个2维的数组。如下图中的数组b，从属性 ndim 中可以知道，它的秩为2，即轴的个数是2，或者维度的数量是2（行和列两个维度中反映出来的是，它是2行3列的一个矩阵。

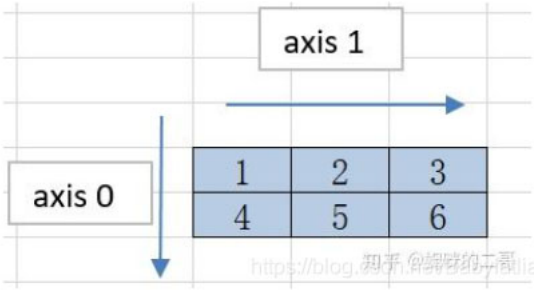
当前轴的数量已经上升为2，那么NumPy中是怎么定义这个轴的方向的呢？在二维中，轴0表示了数组的行，轴1表示了数组的列，见下方的示意图。列中轴0的长度是2，轴1的长度是3。

如果我们把轴0上的数进行相加，可以得到一个一维数组，值为[5,7,9]，数组的元素个数为3。

如果我们把轴1上的数进行相加，也可以得到一个一维数组，但值为[6, 15]，数组的元素个数为2。

所以在不同轴上进行数据操作会得到不同的值，数组中的值是由沿轴方向上的数据相加所得。

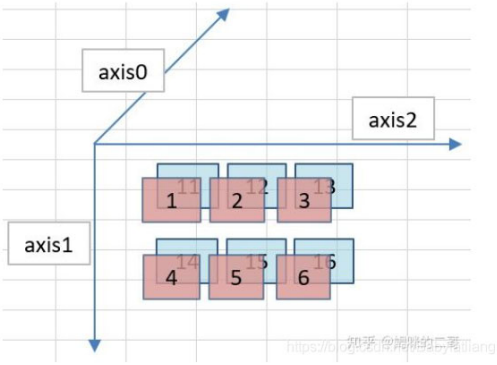
```
In [11]: b = np.array([[1,2,3],
    ...:               [4,5,6]])
    ...:

In [12]: b
Out[12]:
array([[1, 2, 3],
       [4, 5, 6]])

In [13]: b.ndim
Out[13]: 2

In [14]: b.shape
Out[14]: (2, 3)

In [15]: b.sum(axis=0)
Out[15]: array([5, 7, 9])

In [16]: b.sum(axis=1)
Out[16]: array([ 6, 15])
```



现在，我们再升级一个维度，使其成为一个3维数组，如下所示。秩的值已经变为3，它的shape反映出它的轴0长度为2，轴1长度为2，轴2的长度为

轴0-轴2，他们具体是怎么表现的呢？参见下面的图示，从图中可以看到，NumPy对于轴的编号由外向内，从行到列。

```
In [24]: c = np.array([[[1,2,3],
    ...:                [4,5,6]],
    ...:               [[11,12,13],
    ...:                [14,15,16]]])
    ...:

In [25]: c
Out[25]:
array([[[ 1,  2,  3],
        [ 4,  5,  6]],

       [[11, 12, 13],
        [14, 15, 16]]])

In [26]: c.ndim
Out[26]: 3

In [27]: c.shape
Out[27]: (2, 2, 3)
```

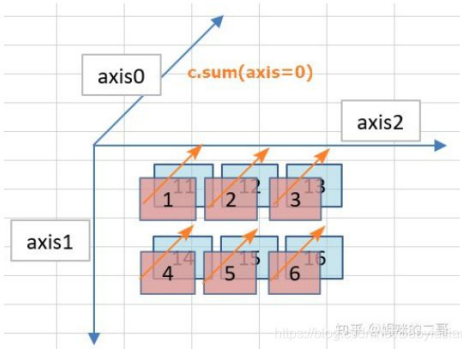


我们通过求和运算来验证上面轴方向的猜测是否正确。

先计算轴0上的数值，从示意图中看更像是，从表面到内部的一次“叠加”操作。这样计算后的一个结果，应该是一个2行3列的一个数组，形状如下：

[12, 14, 16]

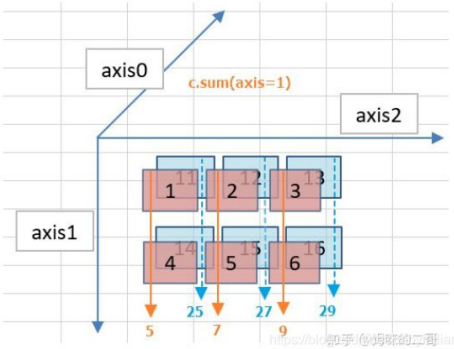
[18, 20, 22]



再计算轴1上的数值，从示意图中可以看出，这像是从上到下的一次“叠加”操作，计算后的值也是一个2行3列的新数组，形状如下：

[5, 7, 9]

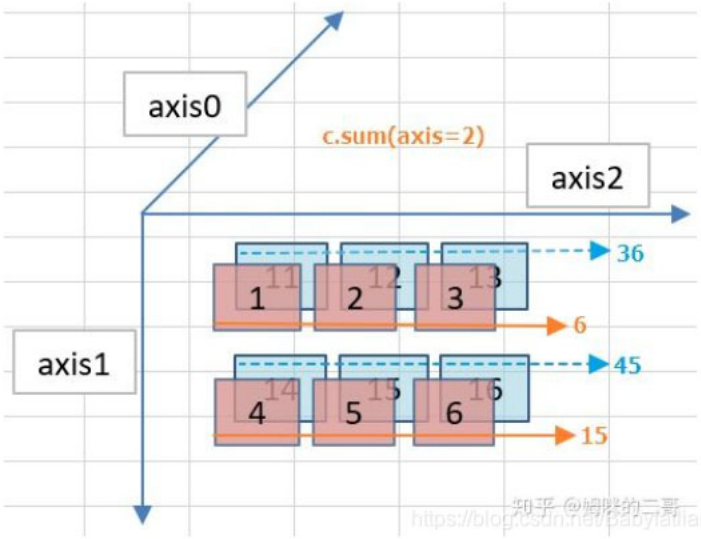
[25, 27, 29]



继续计算轴2上的数值，这次像是一次“右移叠加”的操作，形成的结果是一个2行2列的新数组，形状如下：

[6, 15]

[36, 45]

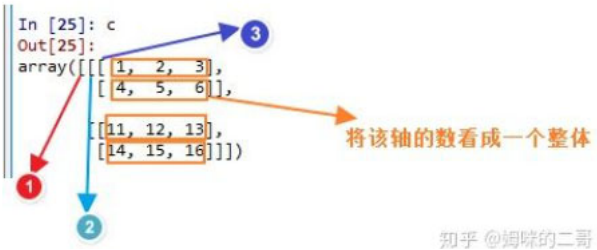


以下是运行结果，和我们的计算结果一致

```
In [28]: c.sum(axis=0)
Out[28]:
array([[12, 14, 16],
       [18, 20, 22]])

In [29]: c.sum(axis=1)
Out[29]:
array([[ 5,  7,  9],
       [25, 27, 29]])

In [30]: c.sum(axis=2)
Out[30]:
array([[ 6, 15],
       [36, 45]])
```

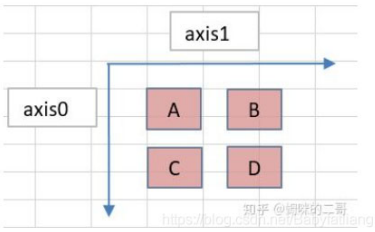
对于3维及更高维度的数组，理解及计算起来比较复杂。因此，我们也可以采用降维的方法来进行计算，如果维度降到二维，那么就非常便于我们的的说就是“替换、降维打击”。

依旧以上面的三维数组c为例，我们只关心最外层的两个维度，将最内层维度的数据看成一个整体。




```
In [34]: A = np.array([1,2,3])

In [35]: B = np.array([4,5,6])

In [36]: C = np.array([11,12,13])

In [37]: D = np.array([14,15,16])

In [40]: new_c = np.array([[A,
    ...:                    B],
    ...:                   [C,
    ...:                    D]])
    ...:

In [41]: new_c
Out[41]:
array([[[ 1,  2,  3],
        [ 4,  5,  6]],

       [[11, 12, 13],
        [14, 15, 16]]])
```

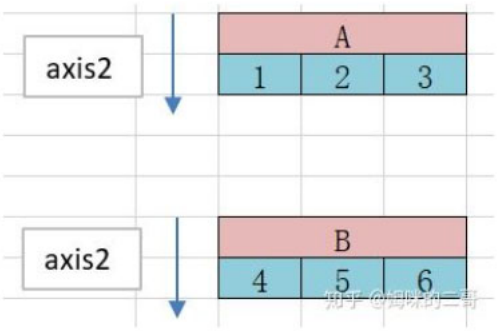
转化后的形状就如下图所示，这时我们计算轴0就等于计算 A+C，B+D；计算轴1就等于计算 A + B，C + D



运行结果如下所示

```
In [45]: new_c.sum(axis=0)
Out[45]:
array([[12, 14, 16],
       [18, 20, 22]])

In [46]: A + C
Out[46]: array([12, 14, 16])

In [47]: B + D
Out[47]: array([18, 20, 22])

In [48]: new_c.sum(axis=1)
Out[48]:
array([[ 5,  7,  9],
       [25, 27, 29]])

In [49]: A + B
Out[49]: array([5, 7, 9])

In [50]: C + D
Out[50]: array([25, 27, 29])
```

轴2要怎么计算呢？由于我们上面已经将最内层的数组看成了一个整体，如图所示。所以，我们计算轴2，就等同于对数组 A、B、C、D进行分别求和




```
In [51]: sum(A)
Out[51]: 6

In [52]: sum(B)
Out[52]: 15

In [53]: sum(C)
Out[53]: 36

In [54]: sum(D)
Out[54]: 45

In [55]: new_c.sum(axis=2)
Out[55]:
array([[ 6, 15],
       [36, 45]])
```

参考：

1， https://deepage.net/features/numpy-axis.html

2，https://flat2010.github.io/2017/05/31/Numpy%E6%95%B0%E7%BB%84%E8%A7%A3%E6%83%91/