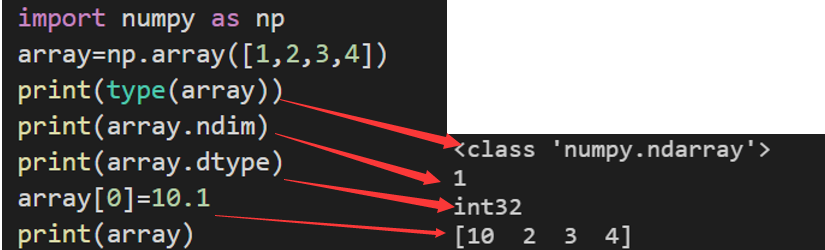
# Numpy笔记

1. **numpy中的ndarray()类型（type）和attributes**

nd表示“n-dimensions”

dtype：array中存储着数据（data），data也是有type的，如32int、8unsigned等，array一开始的dtype取决于存储中的最“牛逼”那个，如果32int和float一起存进了array，则array的dtype是float



numpy对array的数据类型非常严格，比如给定了array的datatype为int，再更改为float会发生截断

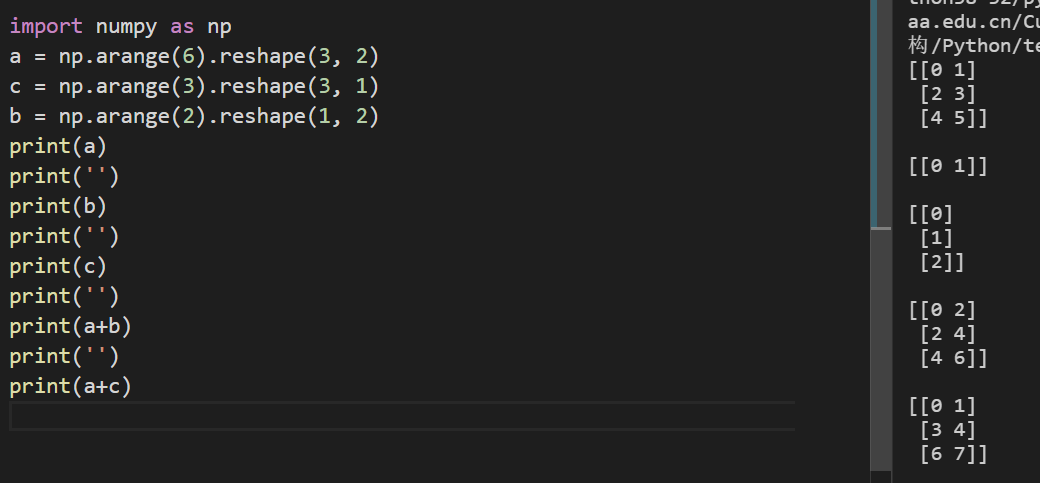
如何创建array？

1. 直接手动输入
2. 利用一些函数的返回值，如np.zeros\_like(array\_a)

给出一个和array\_a形状一样的array，只是用0填满了

1. **numpy中的array的相互操作：**
2. **broadcast**

numpy在对两个数组运算时，先检查是不是compatible，（shapes have to match）依赖右对齐的方式



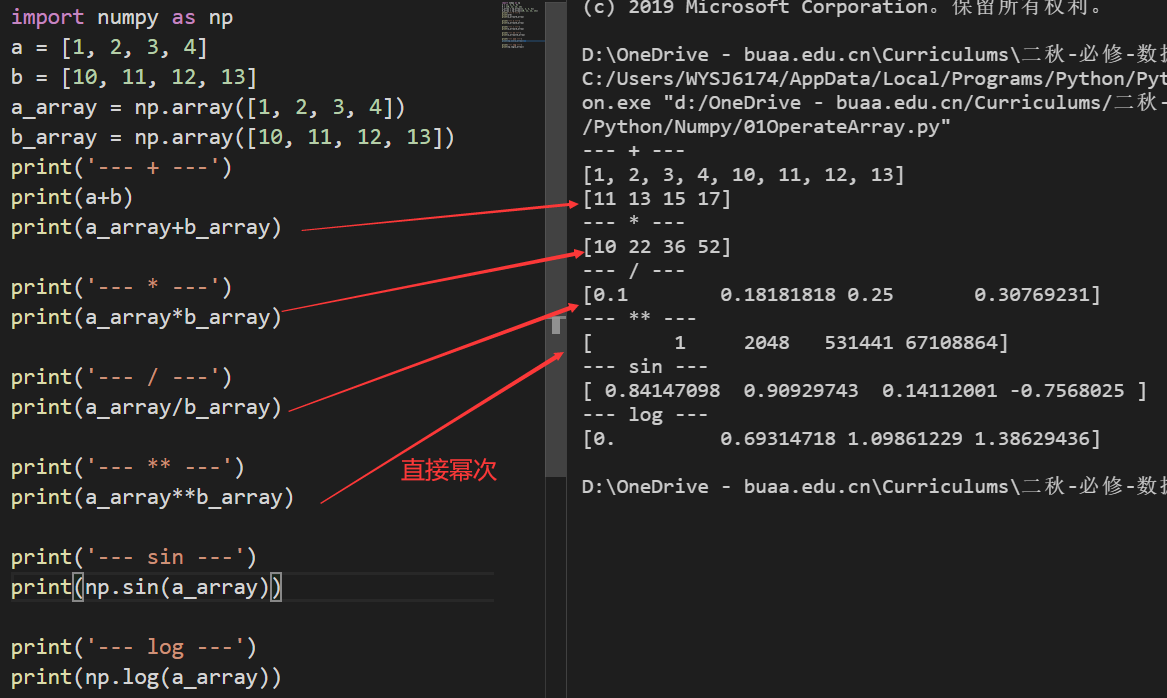
(ab,cd)和(ab,1) (ab,cd)和(1,cd)

这个因数只能是1，不能是其他公因数：



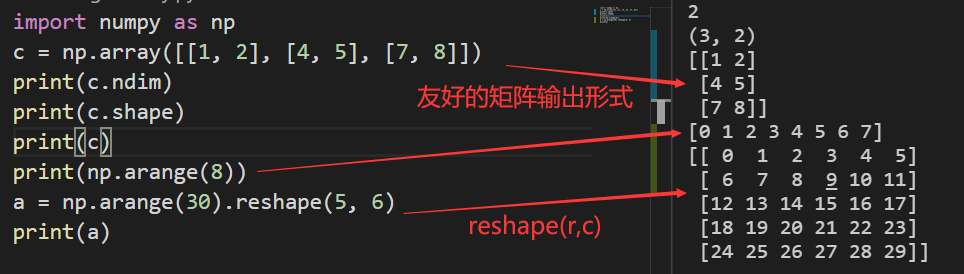
对于维数不同的情况，如(4,5,2)，把一个或多个改成1可以，如(5,1)、(1,2)右对齐即可

1. **operations**



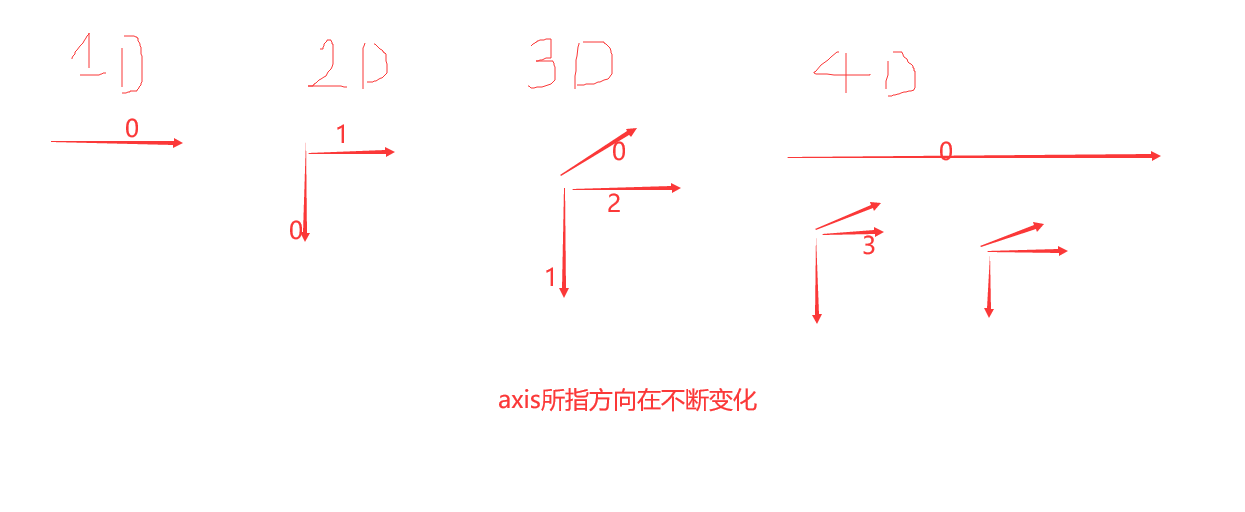
不同于python中直接的extend列表，numpy的加减乘除甚至内置的函数，都是是针对array中的每个元素

1. **高维度的array**



axis的概念：

在索引一个array时，



最后一个axis一定是沿着横轴的，也就是说，axis=-1，在维度改变的情况下，是不会变的。

1. **index**
2. **直接定位元素**

a[index1,index2,…]，如同坐标系一般，但是从0开始的。和list嵌套list的定位是不一样的——list[index1][index2]…

1. slice获取元素，将给出一个视图（view），

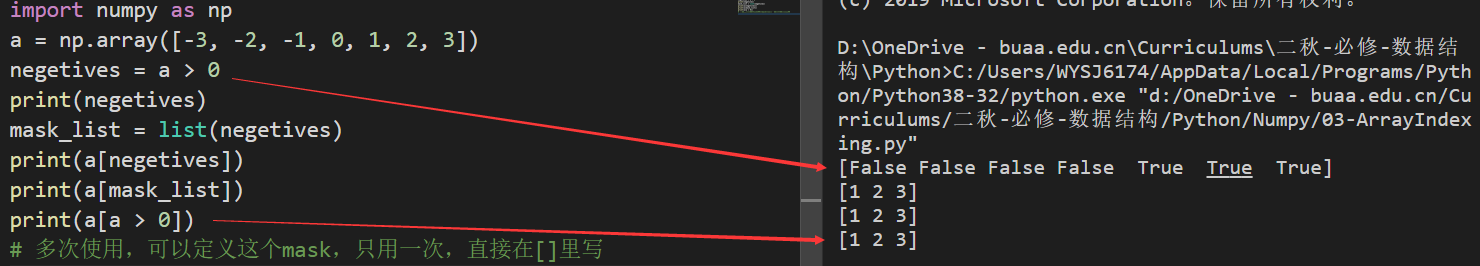
在view中更改，会导致整个array变化

主要说明二维的slice，以为例

a([start:end:step,start:end:step])

slice几次就有几个维度的返回

实际上取得范围是start~end-1，end的索引是不会在切片里的，到end-1

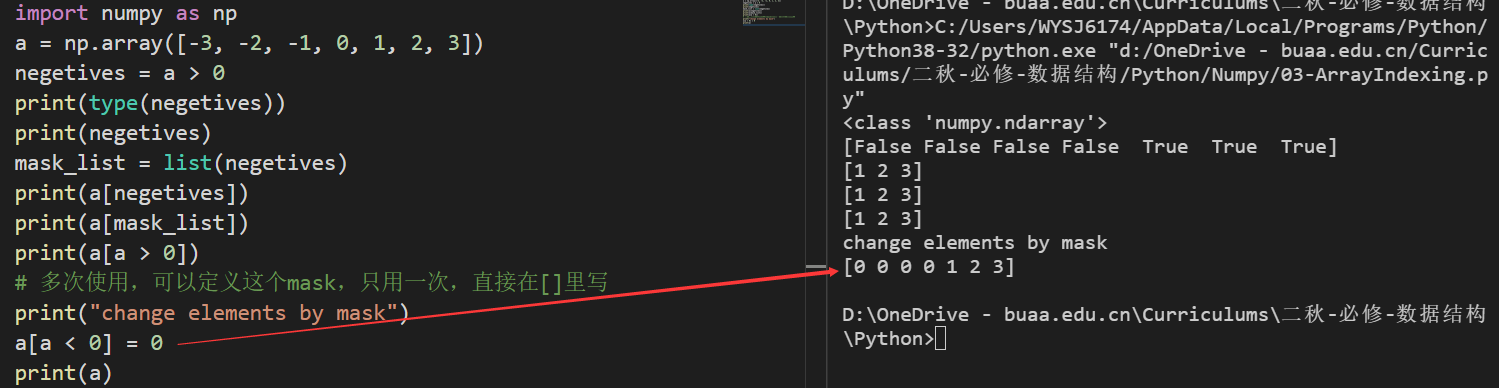
1. mask——fancy indexing，会生成一个copy，在mask中的更改，不会作用到原来的array里
2. 使用mask作为索引，好像array被一张纸给遮住了，然后在一些允许的地方剪了洞，这样，只有被剪掉的地方可以看得见。

返回一个dtype为Boolean的array

高维度的mask给出的是一维的，因为没有自动fill的功能：

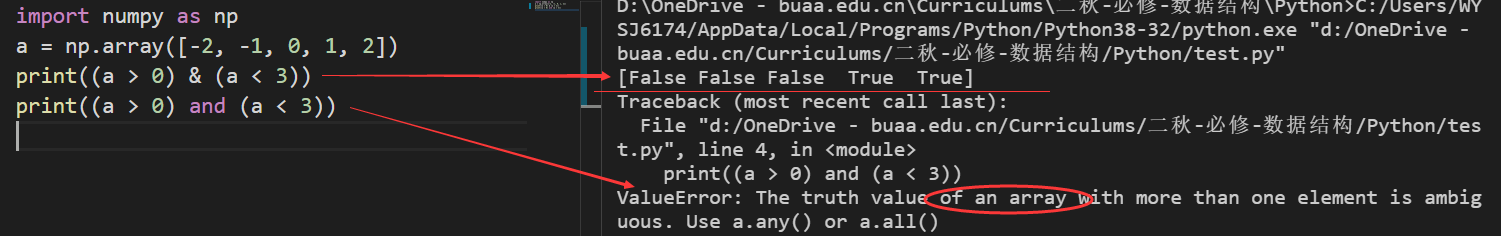


1. 使用mask可以对“露出来”的元素进行更改



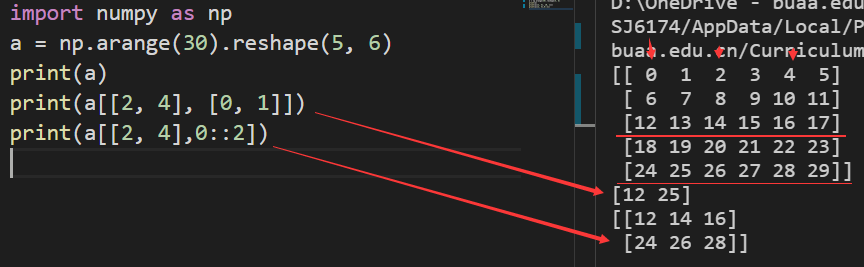
1. 多个条件附加的mask

要用&进行位运算，而不是and关键字

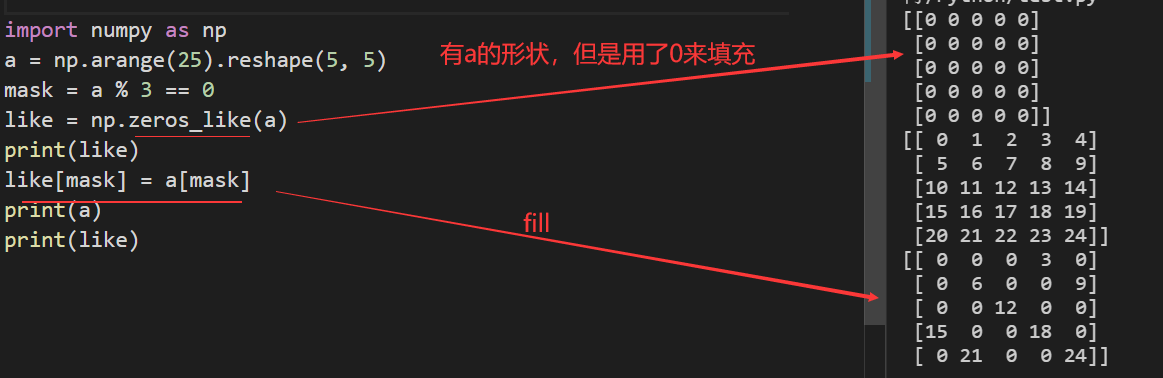


用and关键字话，python试图计算这个多元素的array的真值，而(a > 0)返回的不是真值

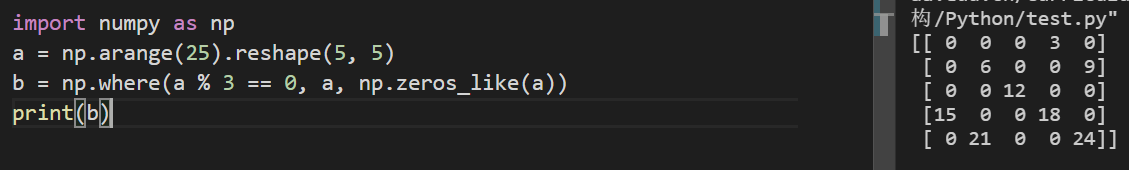
1. 综合使用



1. fill一个array



还可以用where



1. **数组方法**

**注意axis的关键字，有则collapse，如二维的a，a.sum(axis=0)，是在沿着纵向算sum，a.mean(axis=0)是在沿着总向算平均。**

