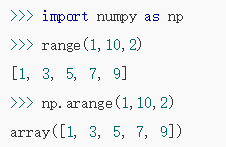
**1. Two sum问题（题号1，easy）**

**（1）range & arange**

range(start, end, step)，返回一个list对象，起始值为start，终止值为end，但不含终止值，步长为step。只能创建int型list。

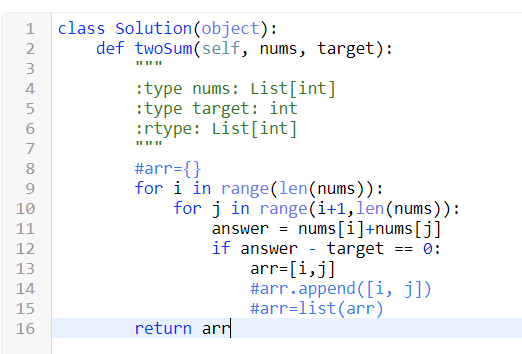
arange(start, end, step)，与range()类似，但是返回一个array对象。需要引入import numpy as np，并且arange可以使用float型数据。



**（2）我的方案**

思路：用每一个数字和其后边所有数字以此相加，找到目标值，返回此时的索引号

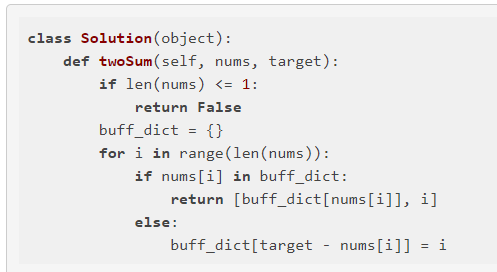
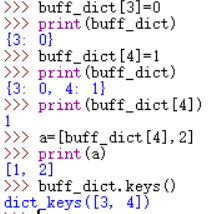
考虑两点：一是不能和自身相加，因此j+1，二是可能有多种结果，所以append;



遗留问题：若用append结果是[[]]形式，怎么把二维数组转一维，但若不考虑重复，可以按上述来写；

运行时间：32ms

**（3）题中方案：**

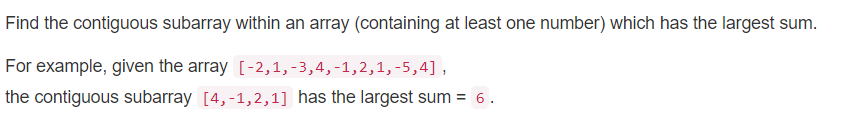
时间：26ms

Nums=[3,2,4], target = 6;

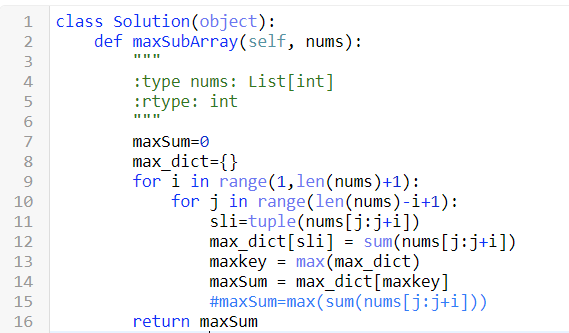
理解：在一开始啥都没有时候创建一个字典，就是说我第一个数和目标差3，那么我就把差的3和我本来3的索引（0）做成一个字典，当后边出现3时，字典就可以查到这个3，然后告诉他前边有个索引是0的3在等它，然后返回这两个3的索引；以此类推，2进来时，告诉他他和目标差4，那么当后边遇到4时，就告诉他前边有个2在等它，把他俩的坐标返回即可，这种方法的好处是不用逐个尝试，前边有记录谁缺什么

**2. Maximum subarray问题（题号53，easy**）

问题：求连续的最大和：

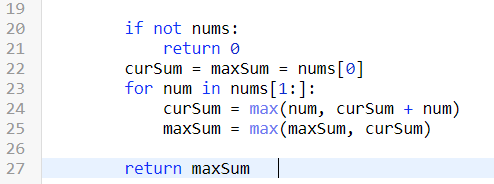


1. 初始思路：创建一个字典，使连续子列表作为键，其和作为值，max这个字典，即可得到其最大值对应的键，但是行不通，因为列表是不可哈希的，因此不可以作为键，大多数的python对象都可以为键，但是列表和字典都是不可哈希的，不能为键，因此可以改为元组



错误：理解问题：输出最大值即可，不需要子列表，因此14行又取了一遍字典里的值，但是结果仍有问题，此时结果为后六个数之和5，i,j有问题

1. 题中方案：

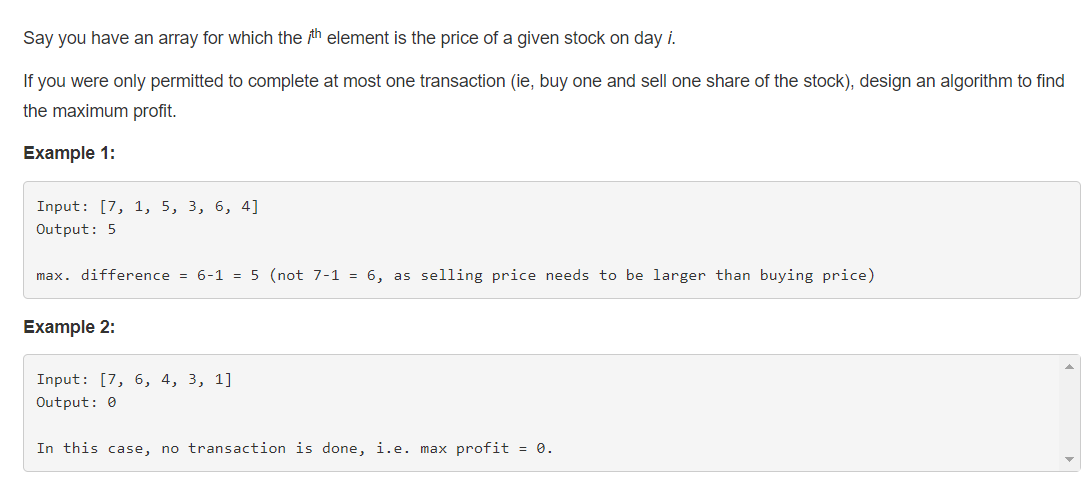


这个思路就是相当于把前两个数先相加，结果与第二个数比，若比它小，则相当于拖后腿，放弃第一个数，从第二个数开始往后加，若比第三个数大，则保留这两个的结果，接着往后加第四个数，结果若比第四个数小则放弃前三个，从第四个数开始加，也就是说从前往后依次相加，每次加两个数，把和与后一个数相比，若大则留作为新的起点，若小则弃，并使后一个数变成新的起点，再将新的数与后一个相加，同理，同时把新的数与和的最大值保存下来，以确定到哪里停止，直至加到最后一个数。

这个方法的优点也在于不用逐个相加，而是先分析数据间的关系，始终排除前边拖后腿的数，并在收集过程中的最值，以便取出，不能不取最值一直加到最后，那样和值有可能变小。

**3. Best Time to Buy and Sell Stock问题（题号121，easy）**

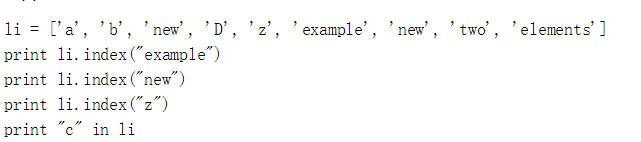
题目：给定一组股票的价格，只有一次交易（一次买入和卖出），设法使获利最大，注意买入一定在卖出之前

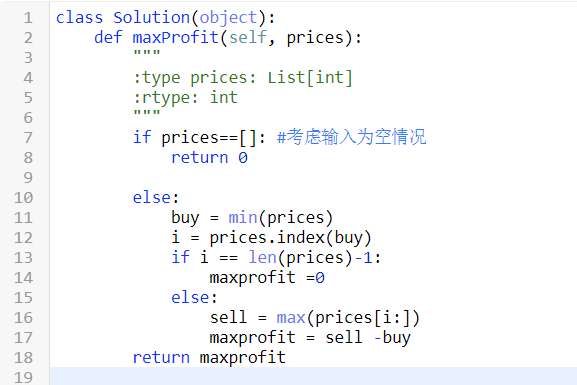


1. 我的方案：

思路：先找最小值，然后返回他的索引，然后找它之后的最大值，二者相减，若之后无最大值（或者说无值），返回为0，

返回数组的索引号用的是index

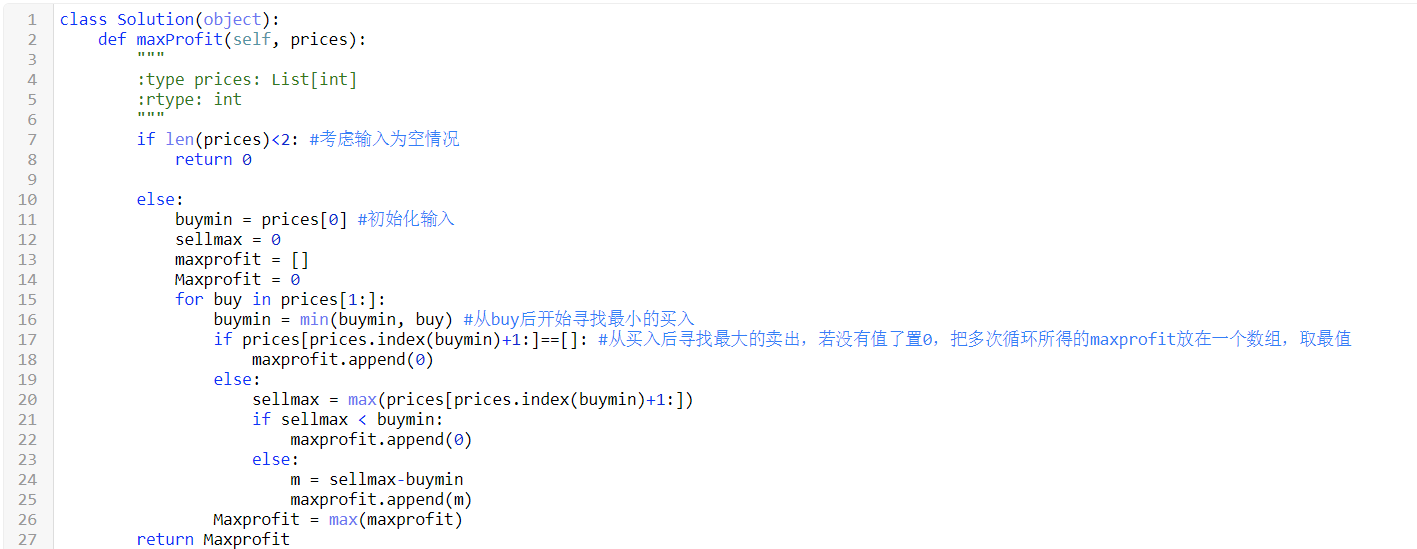




这个思路**有误**，因为不应该找最小值，而是只要买了之后有更高的价格，均可以卖出。如输入[2,4,1]，最小值1后边没有值了，但是可以2买入，4卖出

**因此需要改**：

思路：在循环内，先确定一个买入的最小值，然后从其后找最大值，若没有值了返回为0，若有最大值则计算最大利润 ；在循环外，往后继续找最小值，若找到新的最小值，再从其后找最大值，将循环中所得的最大值不断补充到利润数组里，最后取出数组中的最大值即可

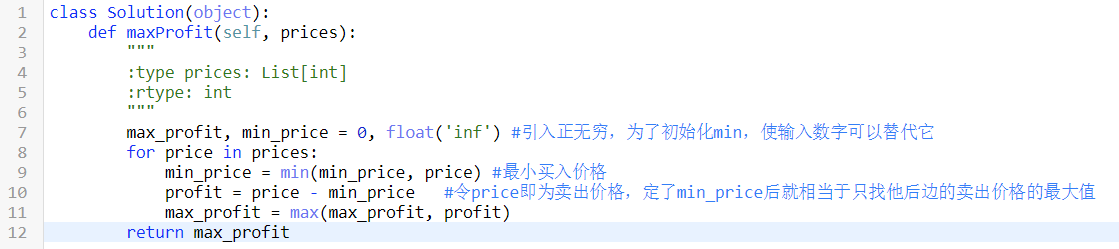


问题：超时26ms

1. 更改版：

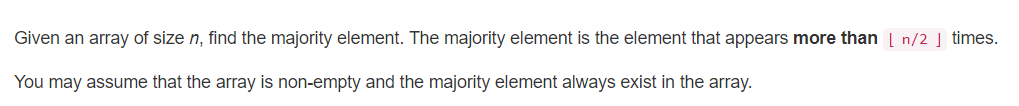
思路：前一部分与我思路一样，先给定一个最小值，但是巧妙之处在于他用了一个正无穷也就是inf来初始化最小值，这样第一个输入很容易就把它代替了，然后用prices数组的每一个数以此作为卖出价格，这样的好处在于不用考虑初始数据是不是空，或者是不是只有一个，买入和卖出的初值都是有的，并且如这种情况相减能保证为0；之后在循环中用price来做卖出值，然后及时取最大更新利润，就不用利润数组的出现了

与我的主要区别在于第10行的加入，用price当卖出，当最小买入定了的时候相当于用price卖出来以此与之作差，因此相当于自动把握了时间顺序去其后找卖出，同时还会更新新的最小值，使步骤简化了许多，值得学习。

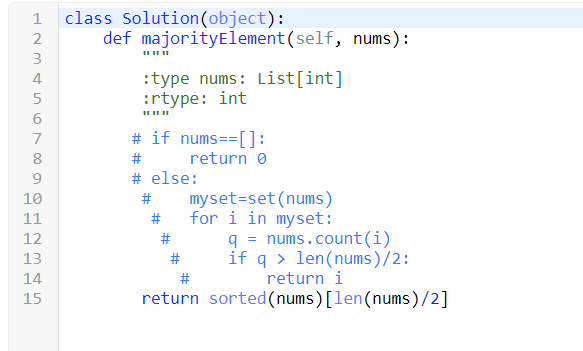


**4. Majority Element（题号169，easy）**

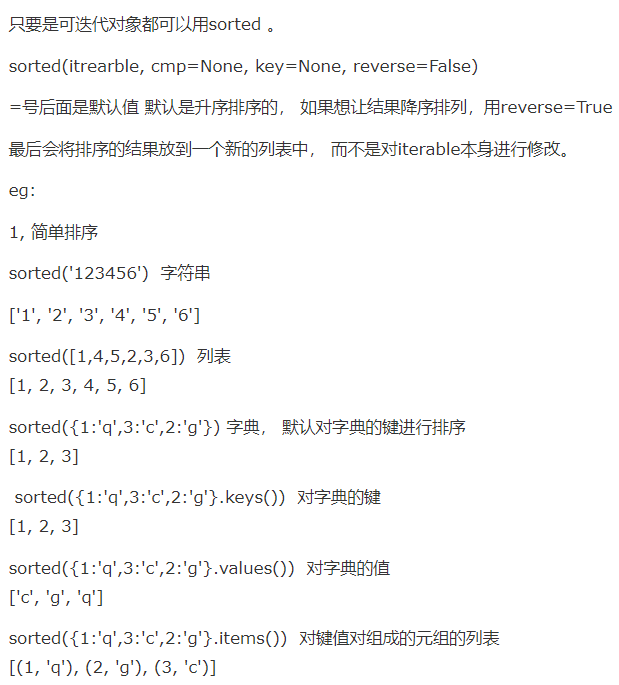
找出数组中出现数目大于一半的元素

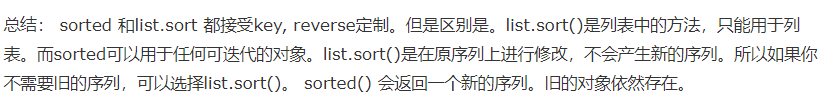


1. 我的思路：用set把非重复元素找出来，再用count逐个计数，选出大于半数的元素
2. 别人思路：用sorted把数组排序，输出其中位数（排序后的中位数一定过半了）

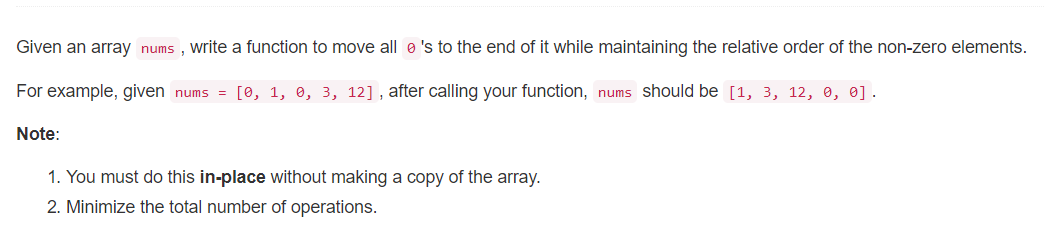


**Sorted()与list.sort()的用法**



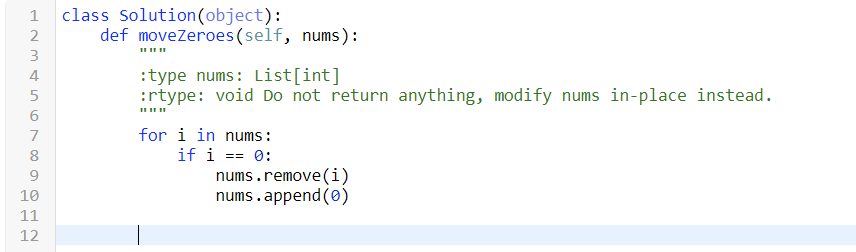


**5. Move Zeros（题号283，easy）**



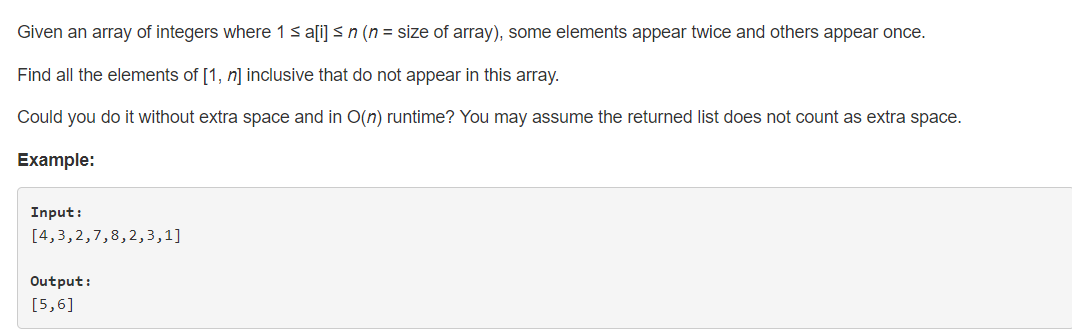
题目：把数组中的0移到后边，非零元素顺序保持不变

思路：逐个检测元素，是0删掉，后边增加0，非零不管

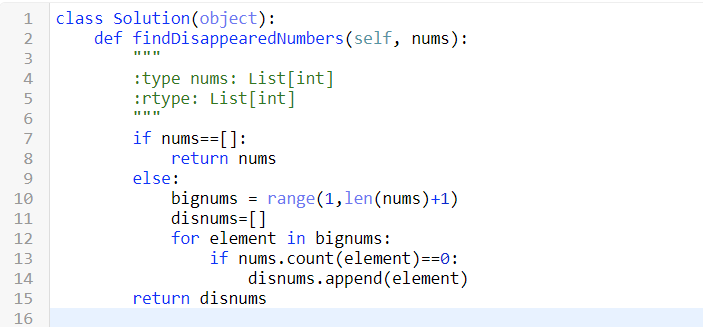


**6. Find All Numbers Disappeared in an Array（题号448，easy）**

题目：找到数组中消失的元素，注意n是数组长度



思路：构建一个大的数组，即1到n的，把原数组当做小的数组，然后在小的数组中计数大数组的每一个元素，要是计数为0，则说明确实，把这个放在缺失元素的数组里



问题：超时

解决的方案：

第一次遍历：用相反数的形式标记出现过的数字（下标）；第二次遍历：找出没有被标记的数字（下标），并返回

