**15. 3Sum**

Given an array nums of *n* integers, are there elements *a*, *b*, *c* in nums such that *a* + *b* + *c* = 0? Find all unique triplets in the array which gives the sum of zero.

**Note:**

The solution set must not contain duplicate triplets.

**Example:**

Given array nums = [-1, 0, 1, 2, -1, -4],

A solution set is:

[

[-1, 0, 1],

[-1, -1, 2]

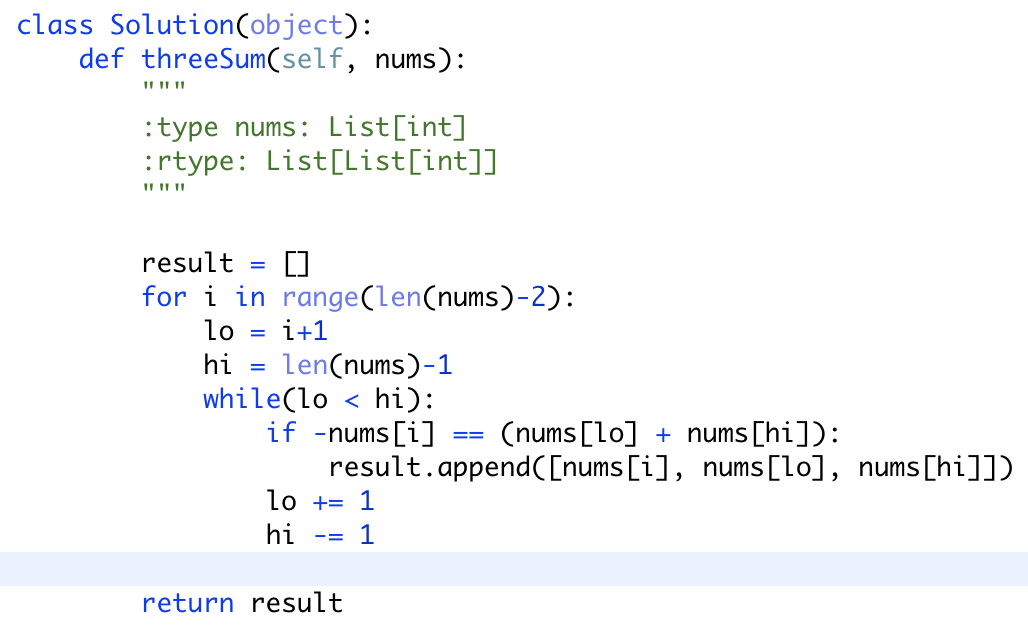
]

参考文献：<https://leetcode.wang/leetCode-15-3Sum.html>

思路：

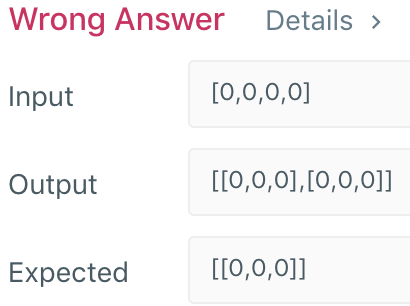
第一步：解决 the sum of zero

满足等式 -x = y + z，x为第一层遍历，y，z为第二次遍历。



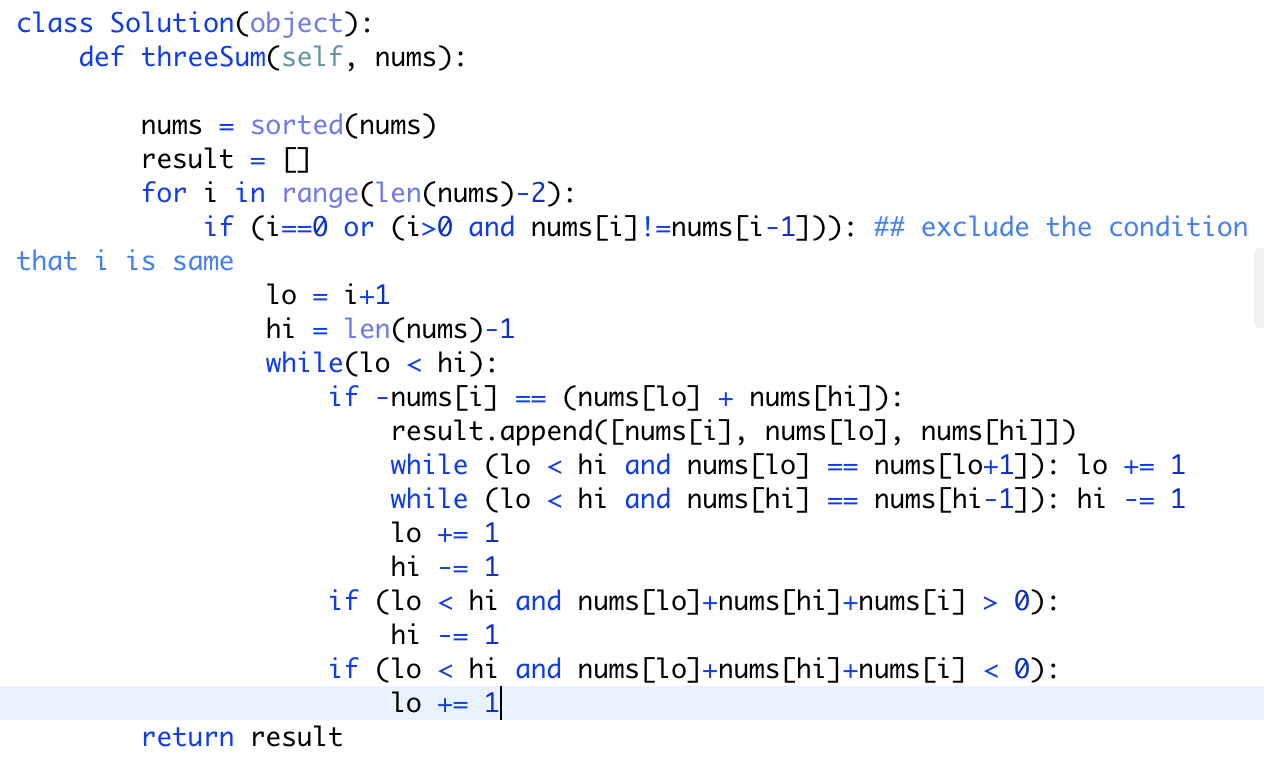
这部分我大致写了一下逻辑，但后面测试发现其实细节还有一些问题，将在后面进行讲解。

第二步：解决The solution set must not contain duplicate triplets



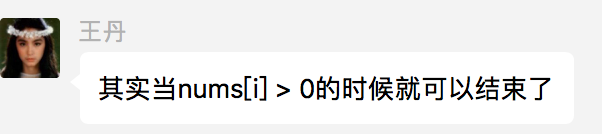
代码的亮点在对代码进行排序后，分别用两个向前和向后的index遍历，检测重复数字。满足等式 -x = y + z，i为第一层遍历，lo，hi为第二次遍历。

1. 当i遍历到重复数字时，一定会把重复的solution加入进答案，所以要用if语句排除这种情况
2. 排序后，相同数字必定会出现在一起，所以遇到重复数字时不断加一就可以跳过。注意圈里的条件不可以省略，那么问题来了，明明前面while语句都判断了这个条件，为什么这里还需要这个条件。
3. 我第一步的代码有问题地方就是每次都会跳过两个数字，不是每个数字组合都遍历到，就是缺了这个代码。



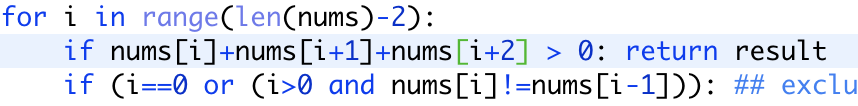


更新1



因为数组从小到大排列，所以当nums[i]>0时，就一定找不到结果了。

更新2:



会更快一点