**数算cheetsheet**

**勿忘：0.一定要记得考虑一些边界情况！！**比如只有1个路灯（那样的话距离序列就不存在），或者只有0个元素；以及一定要验证**第一个格子是否是陷阱**~（重点检查：首步、尾步、0个元素）

Vip：debug许久的时候，检查**是否拼写有误or字母写错**~

1. 如果TLE，可以**把list改成set**，这样添加、查询（判断在不在内）。移除均为O(1)，可大幅减少time！也可以**用dict记录数据**~（可以创建字典来**记录每个数和他对应的index**！！）

（p.s. 1.set只有add；2.set无序，不可查询一个元素的位置；3.set内只能有**int，float，str，tuple**，不能有set，list，dict！）

2.系统判断两个列表是否相同的时候会考虑元素顺序！（元组同理）

3.ord将一个字符转换为他的Unicode 码点（整数），chr反之

4.None的N要大写！！一定**不要忘记移动curr！！**

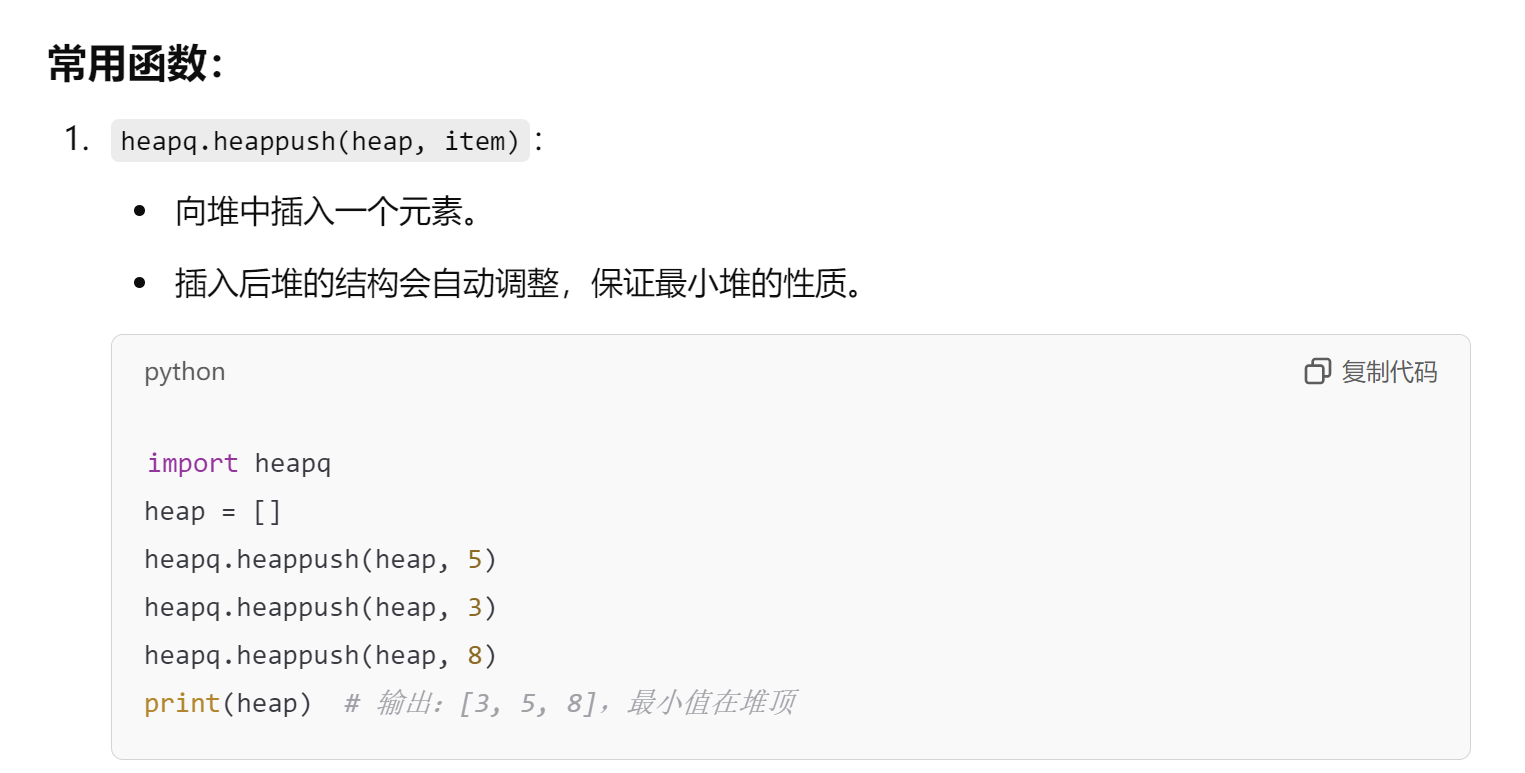
5.输出列表时不要输出deque（输出形式不一样）！

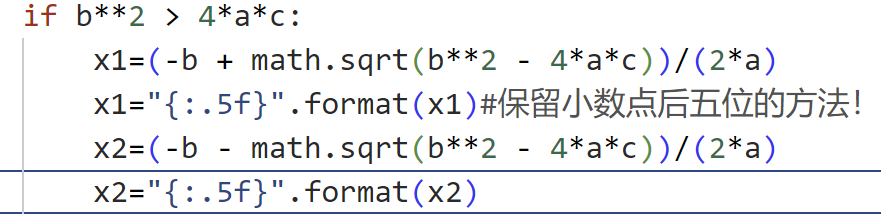
6.要小心是否有多组输入！！

7.一定要**先添加、再移除**！！（remove永远在最后！）

**import heapq（函数：heappush/heappop/heapify）**

**heapify类似于sort，是一个就地操作，不会返回任何东西！**

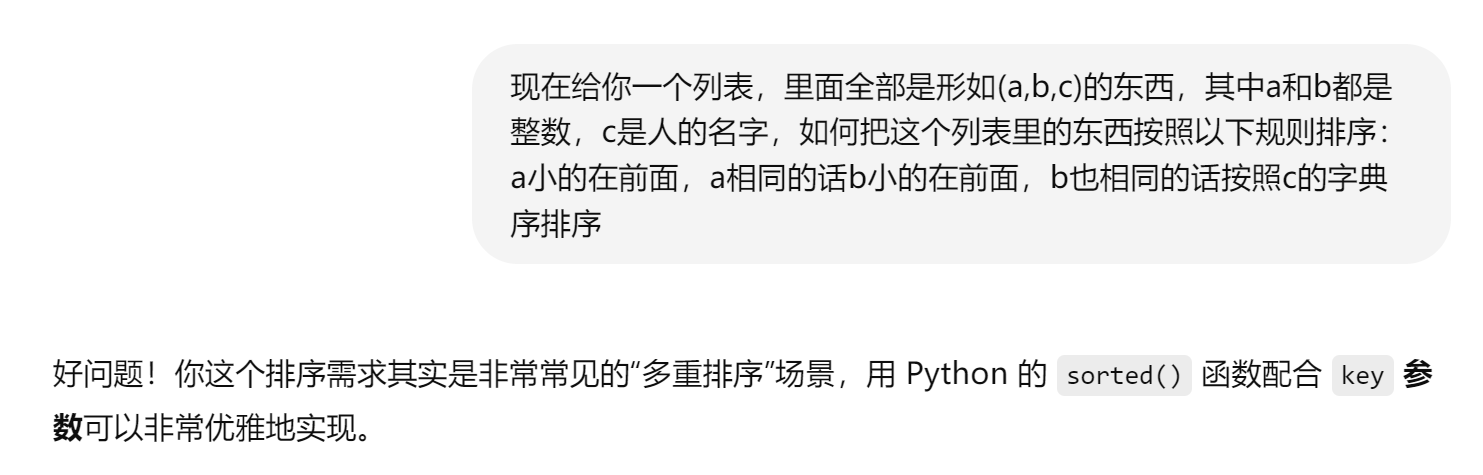
****

****

**保留小数点后若干位的方法！**

**-------------------------------------------**

**如何对多个元素的元组排序？**

****



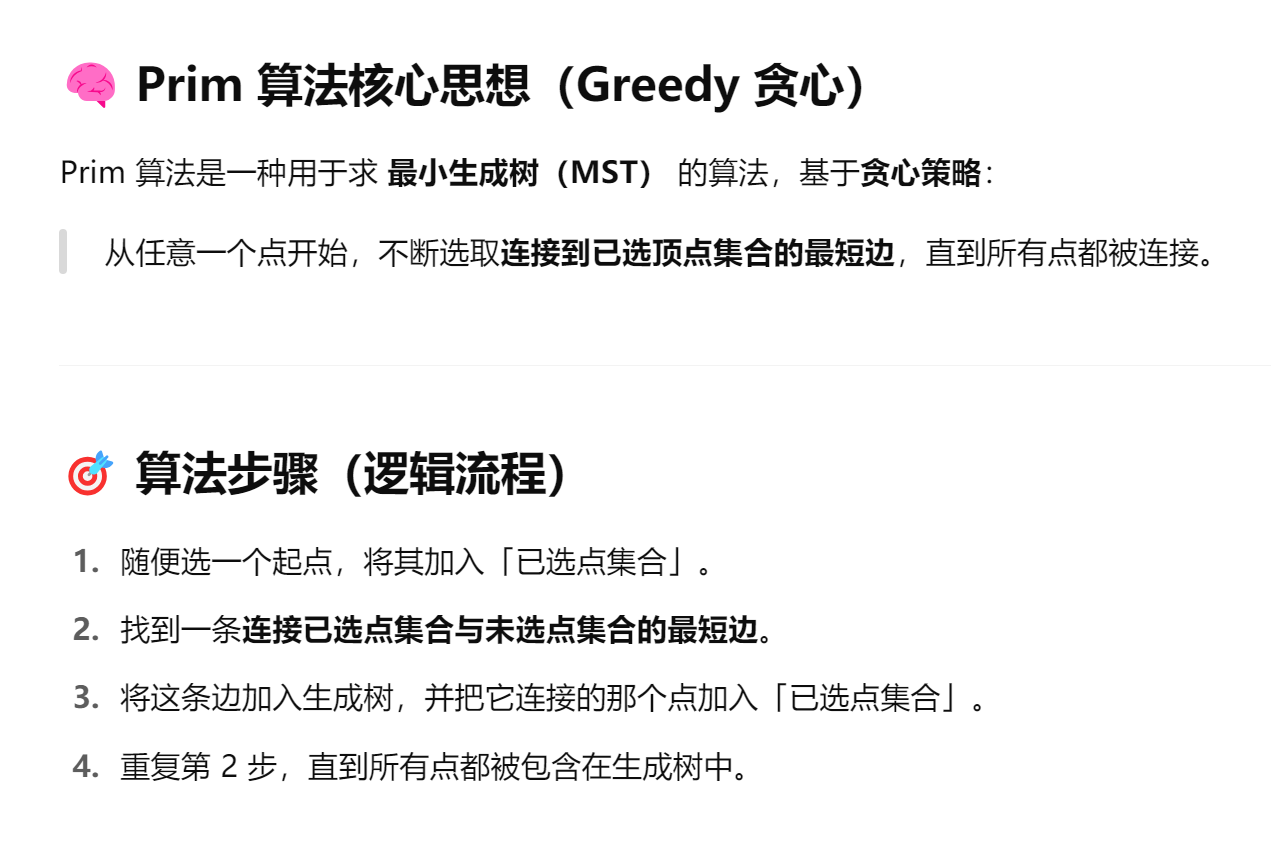
**重要：不要直接对字典的values排序！这样会损失key的对应关系!**

**MST：用最小的代价，连接所有点！**

****

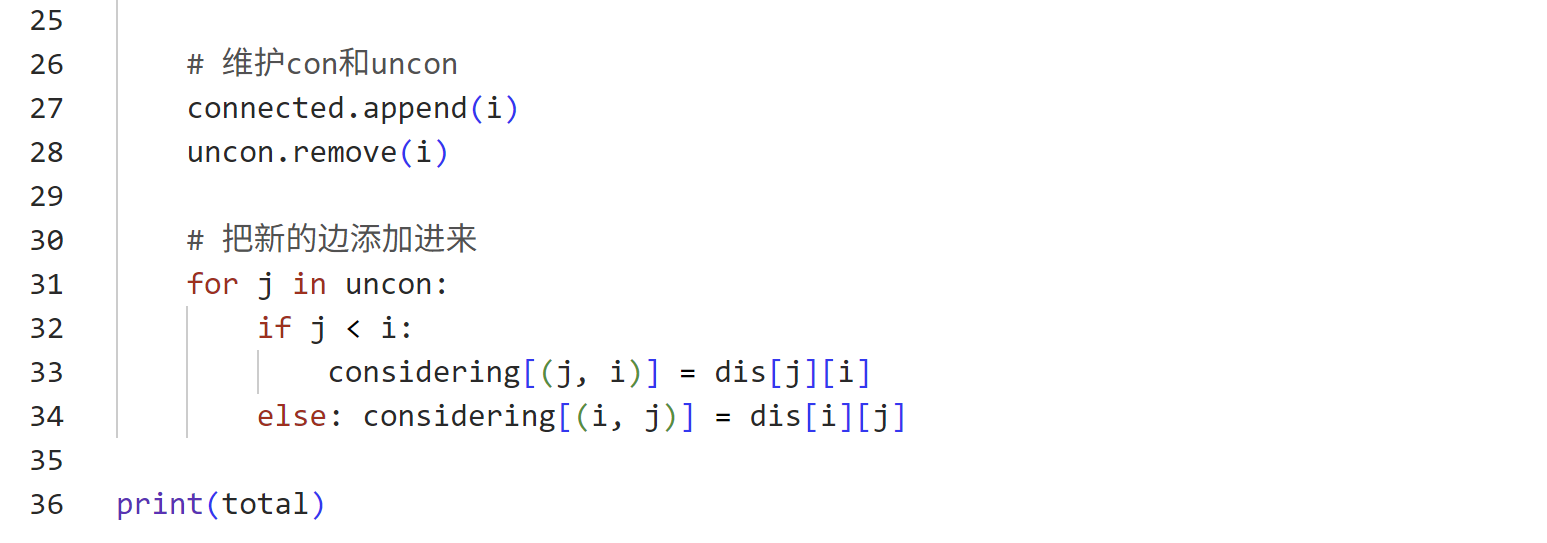
**Prim算法（适用于找最小生成树）：**

**每次都找那个未被连接的、且连接代价最小的点加入树！**

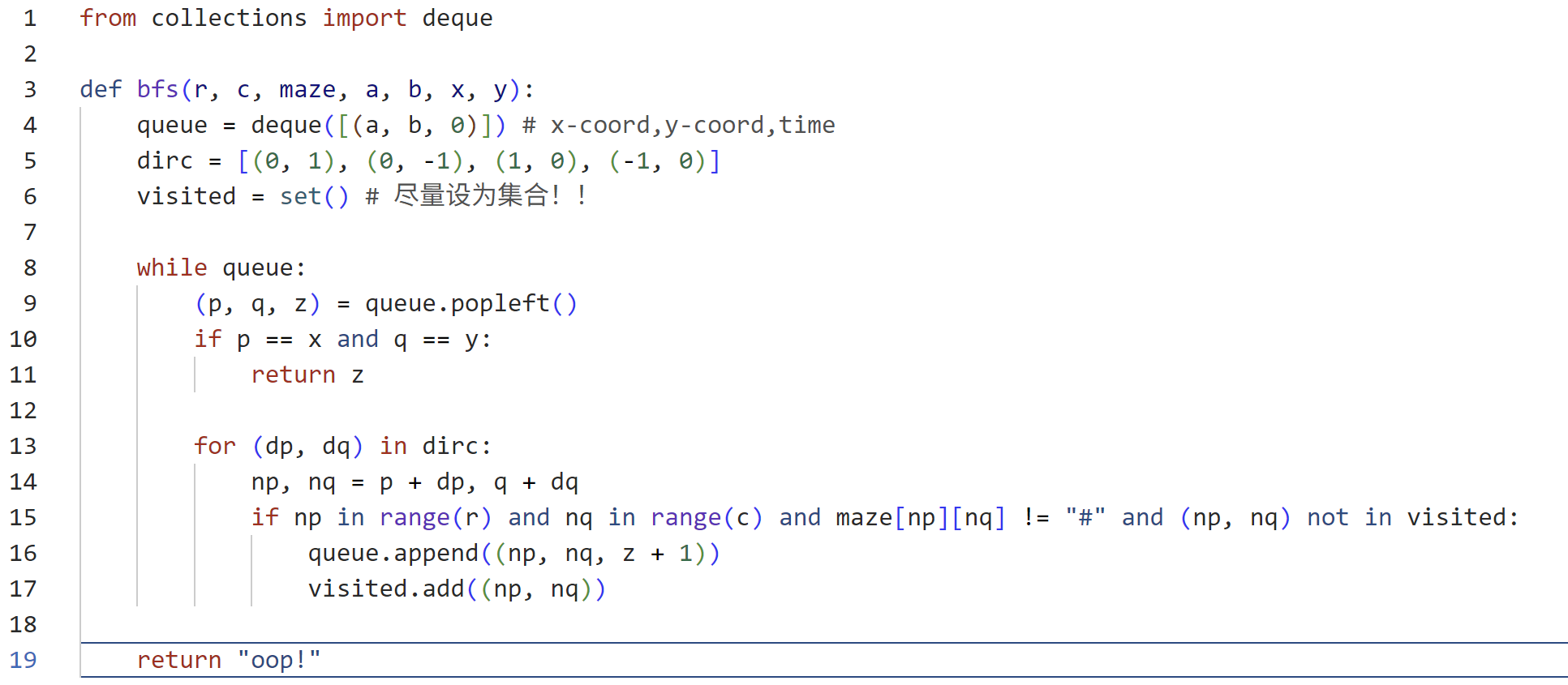
****

**Prim标准代码：**

****

****

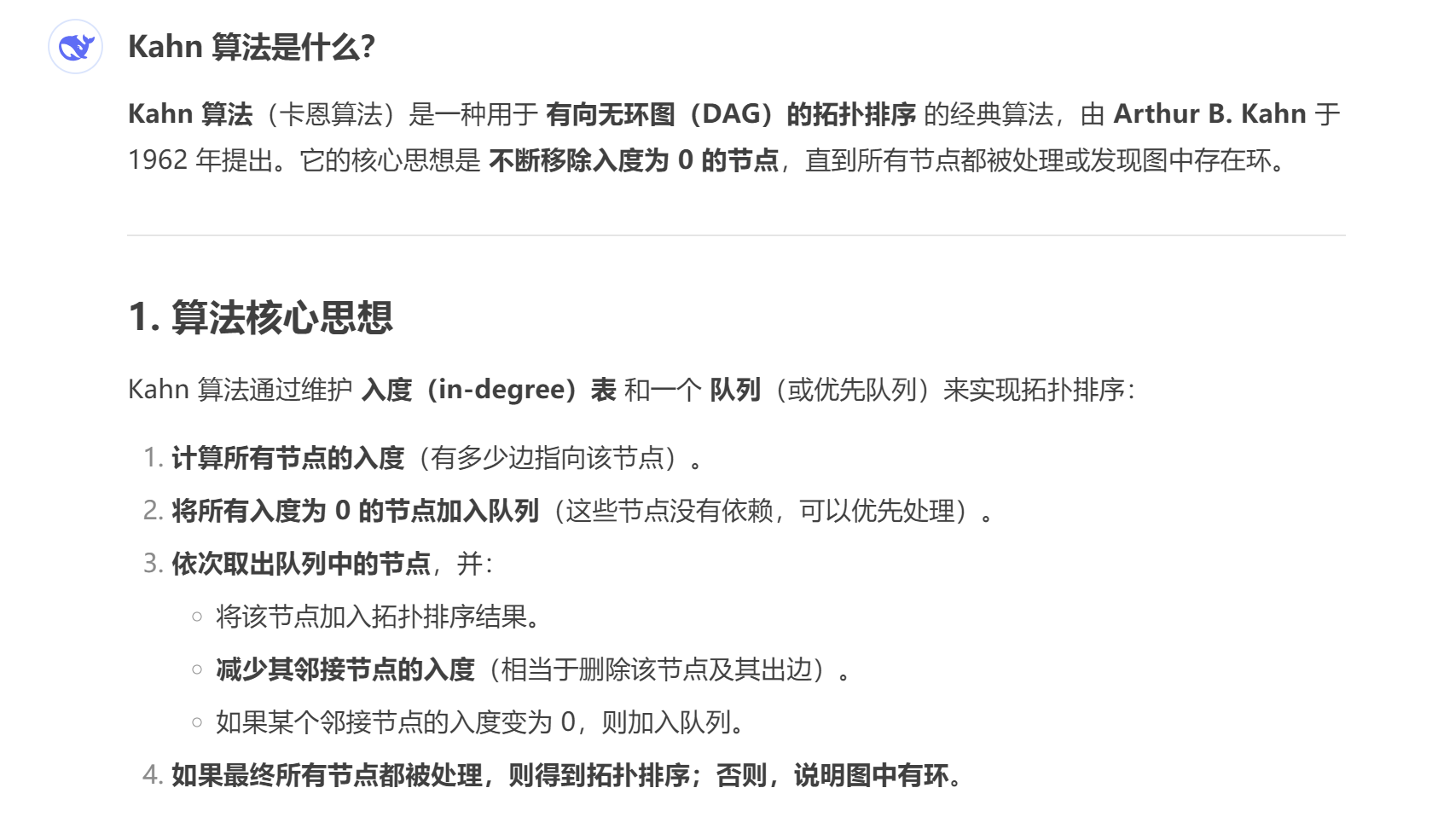
**BFS标准代码（queue/dirc/visited!）**

****

**Kahn算法：用于有向图拓扑排序、检测是否有环**

**逻辑：每次把入度为0的点加入队列，将其加入排序结果**

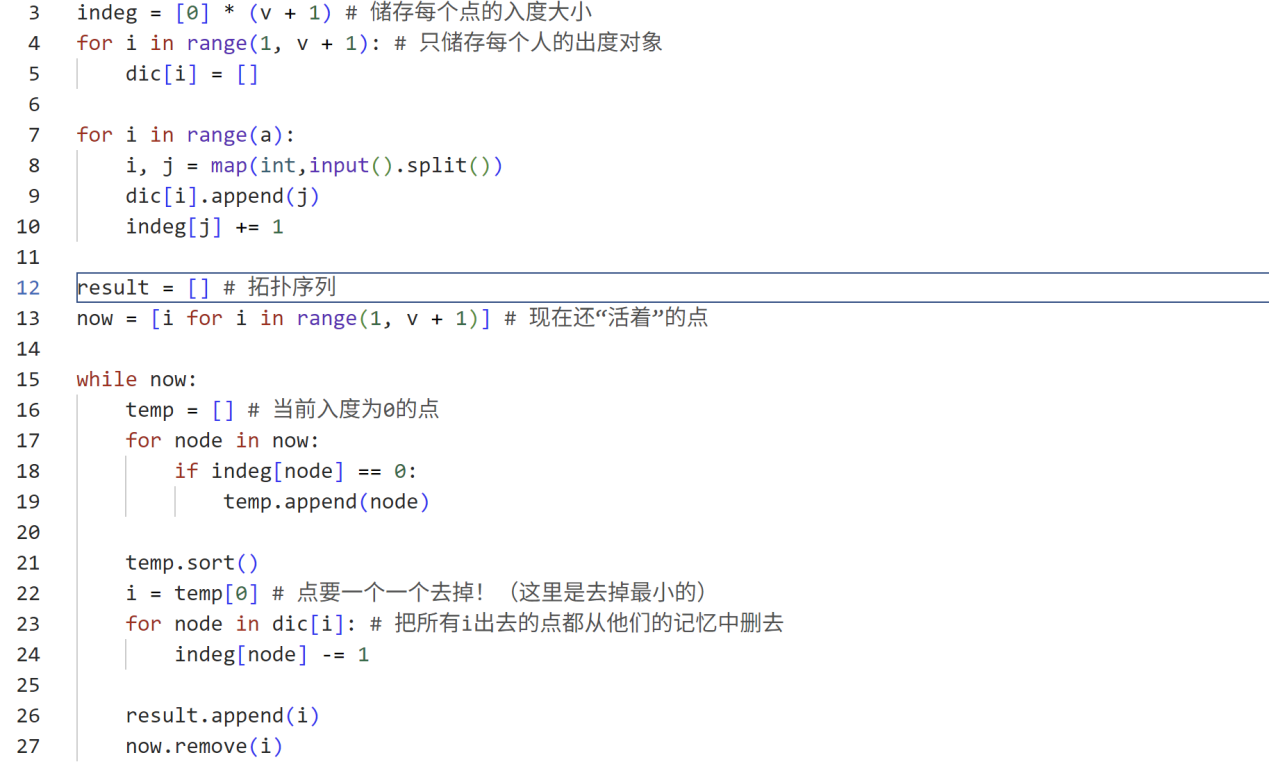
**（最好一个一个点扔掉！！）**

****

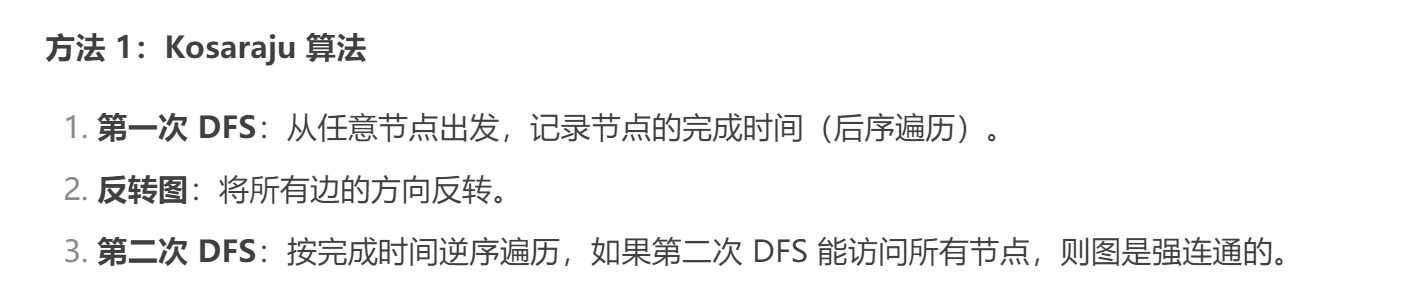
由数学归纳法易得，一个无环的图一定永远有入度为0的点；

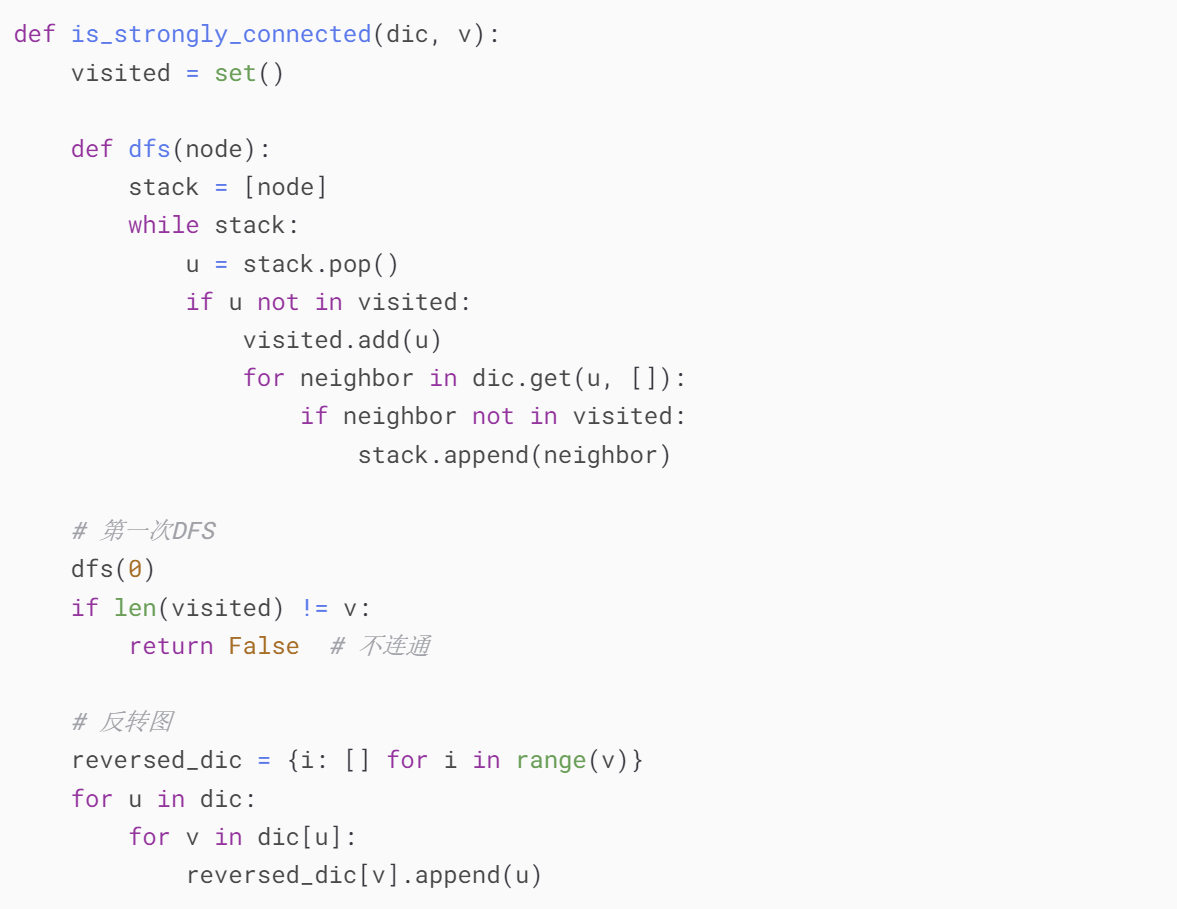
**因此，如果最终结果没有包含所有点，则说明有环！！**

**Kahn标准代码：**



有向图检测环如上，但检测是否连通较为复杂，需要两次dfs：







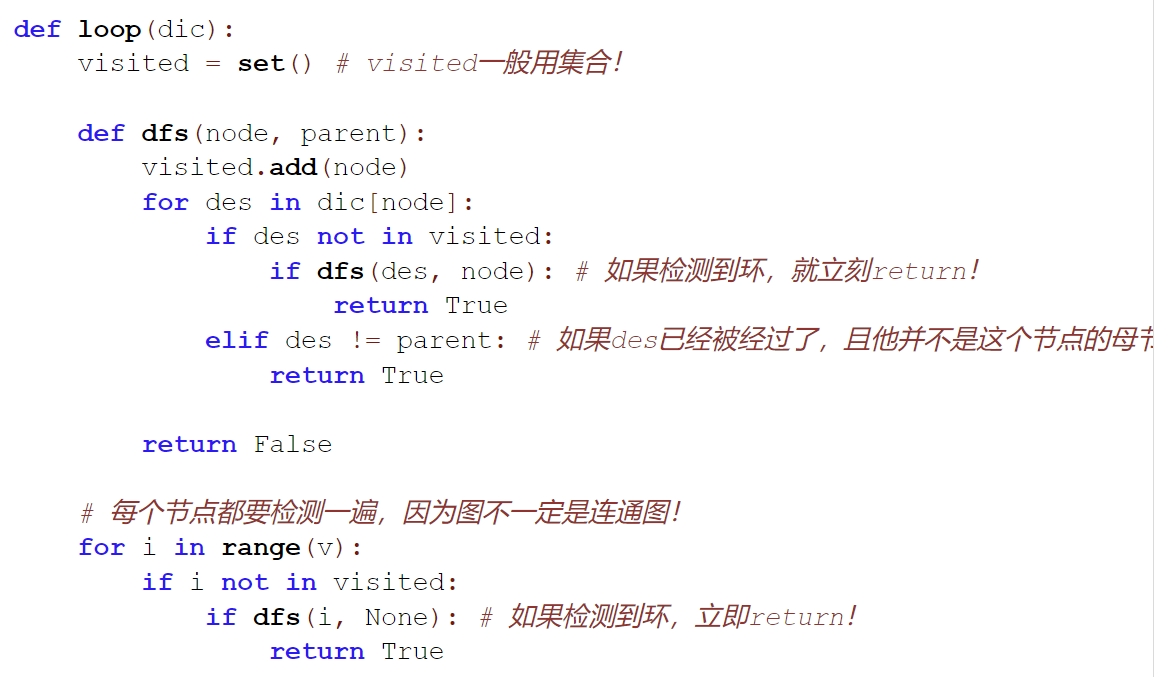
接下来是无向图检测是否连通、是否有环：

1. 是否连通：每一轮把**新加进来的连通的点的所有的还未被连通的邻居都加进来**；
2. 是否有环：用dfs记录每个点和他的parent，**每次递归时遍历一个点的所有邻居**，如果这个领居不在visit里就接着递归，如果**在visit里面就检查这个人是不是他的parent；如果不是则说明有环！**（注意要对每个点都dfs，因为不一定连通）

**检测是否连通：**

****

**检测是否有环：**



**建立树：（1.建立 2.直接 3.通用）**







**并查集：find**（仅适用于**无向图！）**

让树变得更扁平，加快后续查找速度

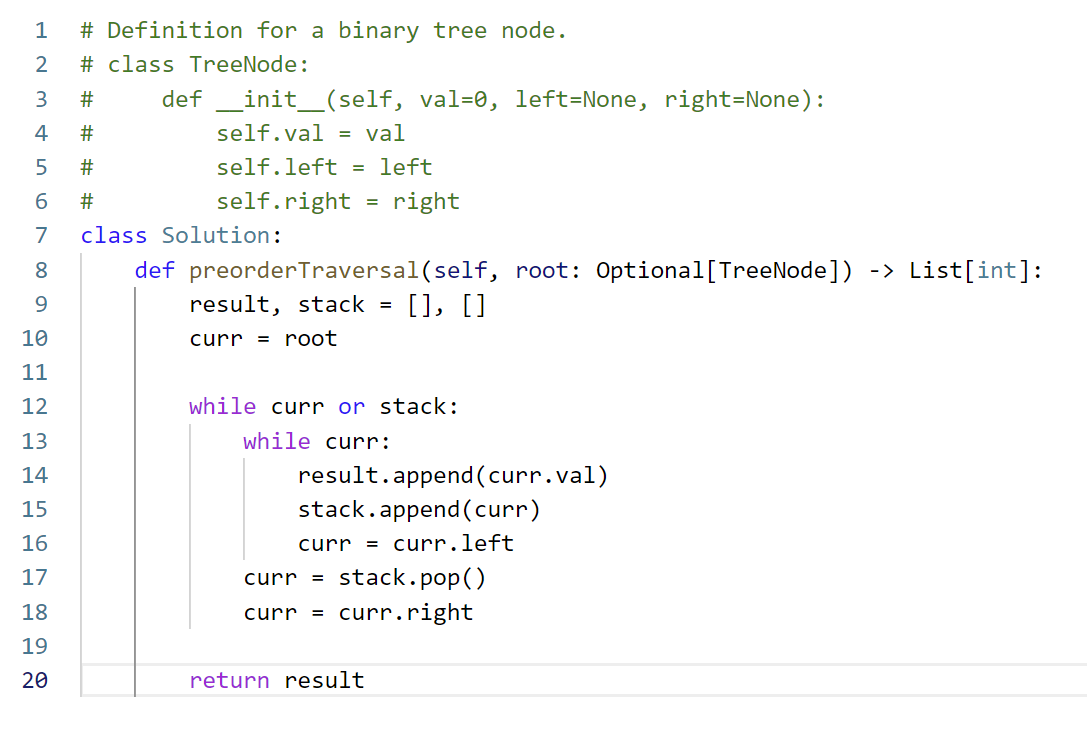
**（一直向前递归，直到找到一个自己连接自己的，然后把递归过程中所有点都连接到这个“根”）**



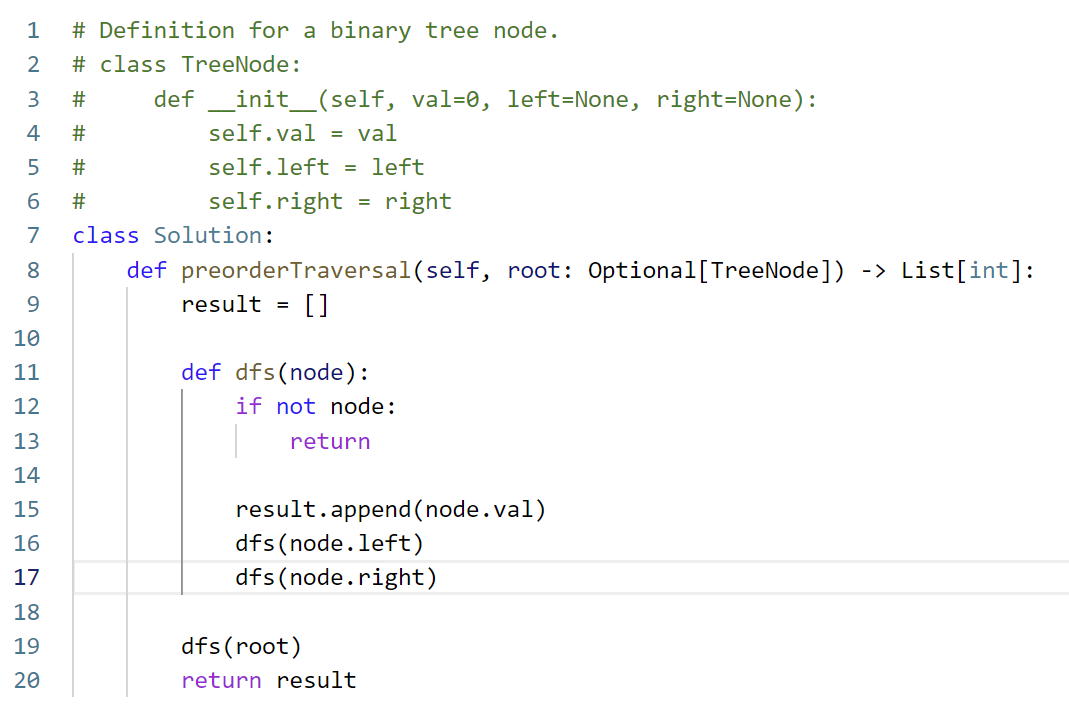
应用：



**前序遍历（迭代）：**

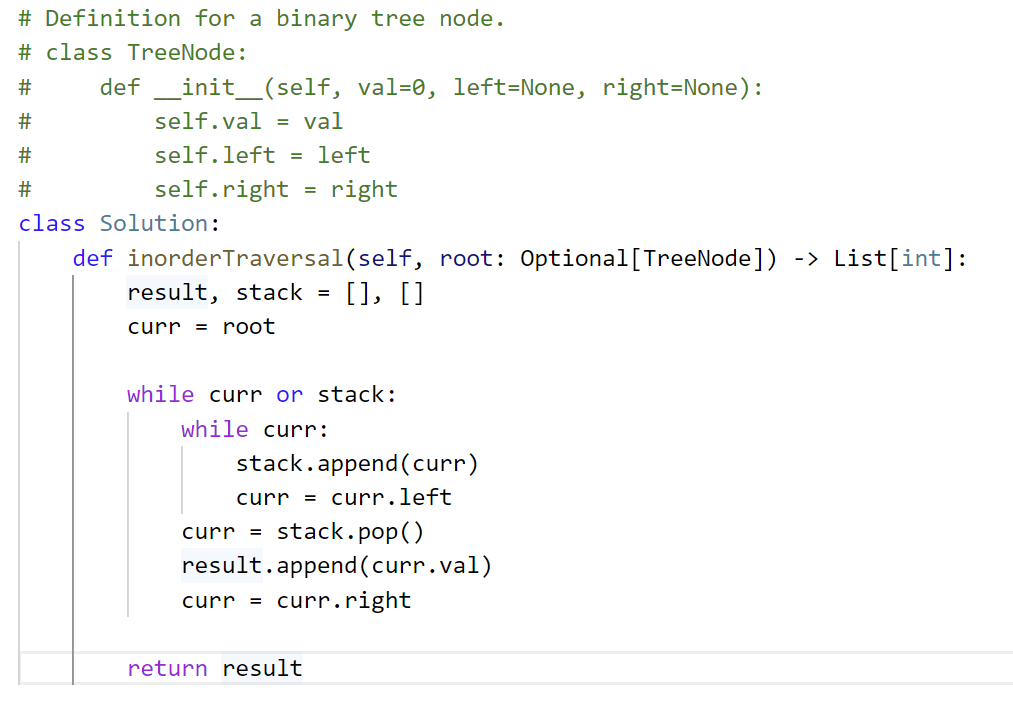
****

**前序遍历（递归）：**

****

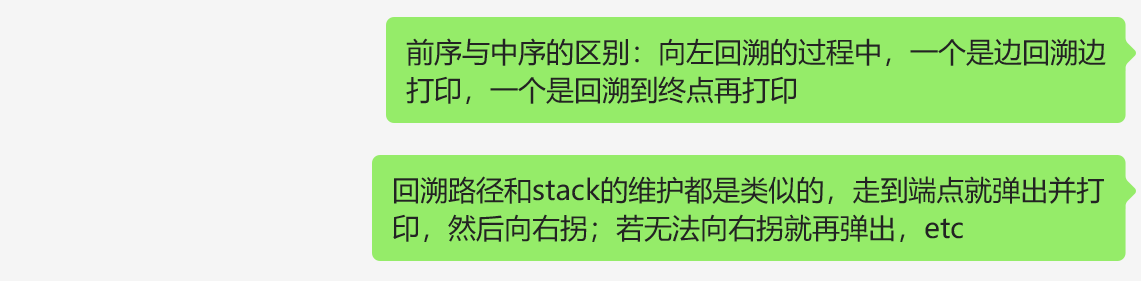
**前序本质：一直向左走，走到哪打印到哪；走到尽头了，拐向右边（如果右边没有，就回溯到stack上一个，继续试图拐向右边）**

**中序遍历（迭代）：**

****

**中序遍历（递归）：**

****



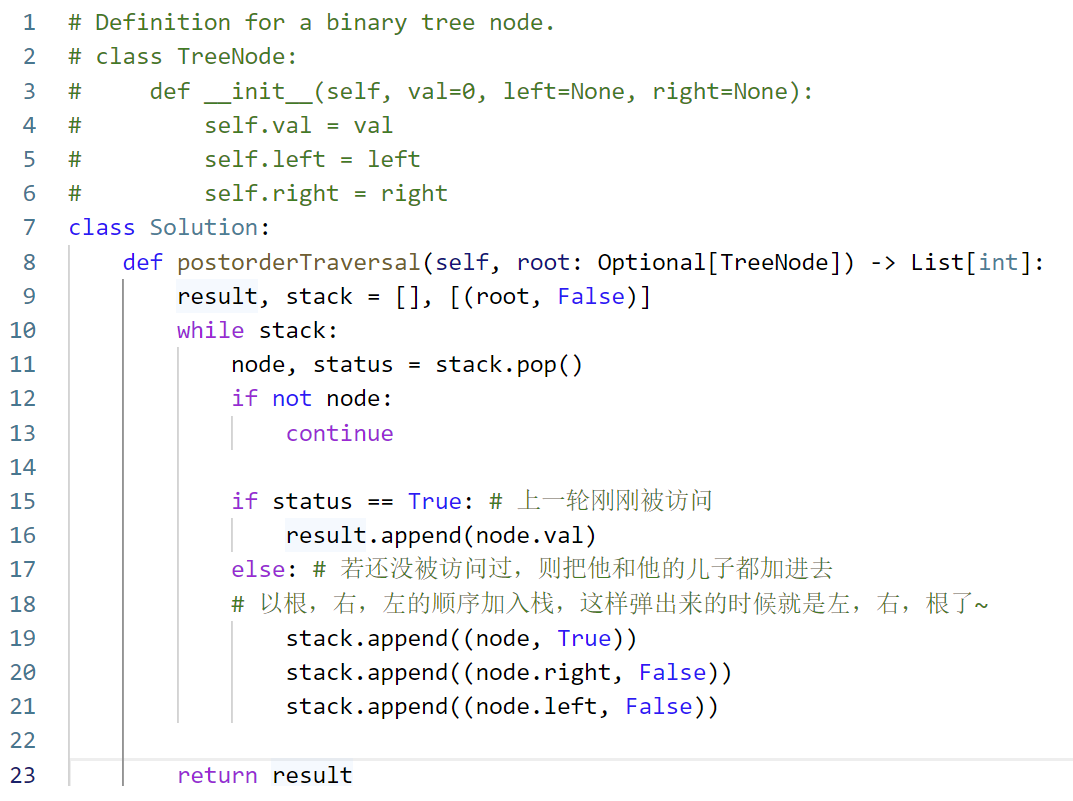
**中序本质：一直向左走，走到尽头再打印；**

**走到尽头了，拐向右边（如果右边没有，就回溯到自己的上一个并打印他，接着继续试图拐向右边）**

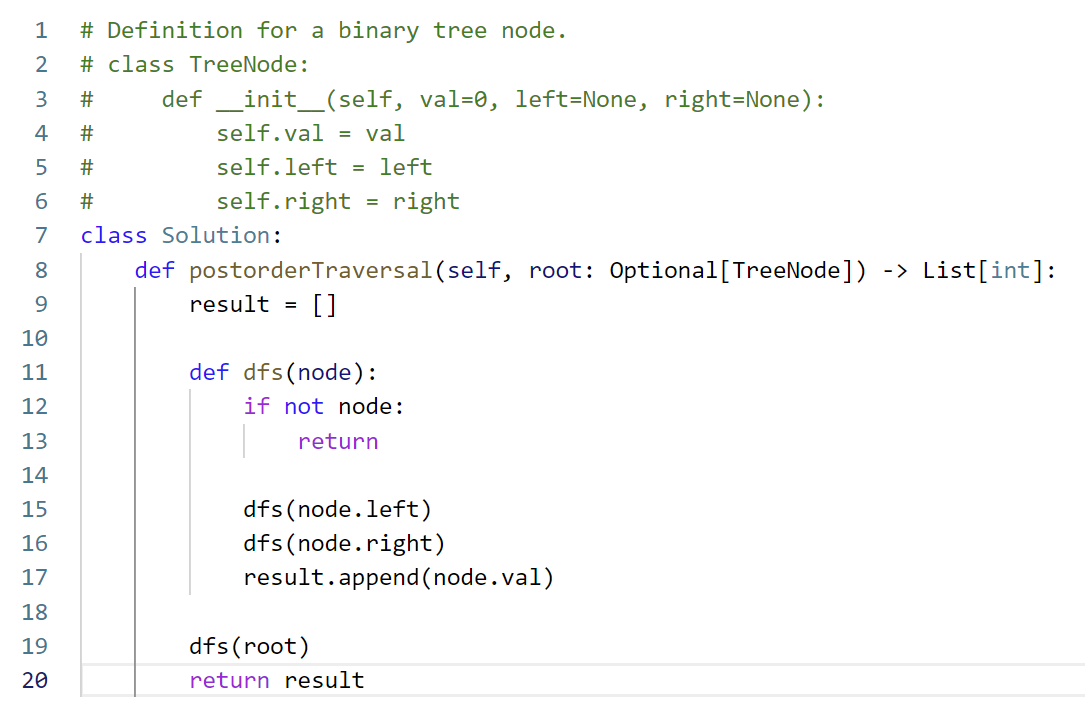
**后序（迭代法一：将前序完全反转过来。向右回溯，这样result=根，右，左；最后将result也反转即可）**

****

**后序（迭代法二：通用模板。用visited表示每个点有没有被访问过；注意要把点和状态存储在一起，否则无法一一对应起来）**

****

**后序（递归）：**

****

**from collections import Counter**

**（统计str/list中每个元素出现的次数，并输出一个字典）**

****

**from itertools import permutations**



（perms生成的是元组）



**格式：对象，长度（长度可省略，但对象必须以变量名的形式出现！）**



**（会自动按字典序生成）**

**注意：不能print（perms（x））！**

****

**因为perms只会返回一个迭代器！！**

**import sys**

**sys.setrecursionlimit(10\*\*5)** #设置递归深度限制（默认1000）

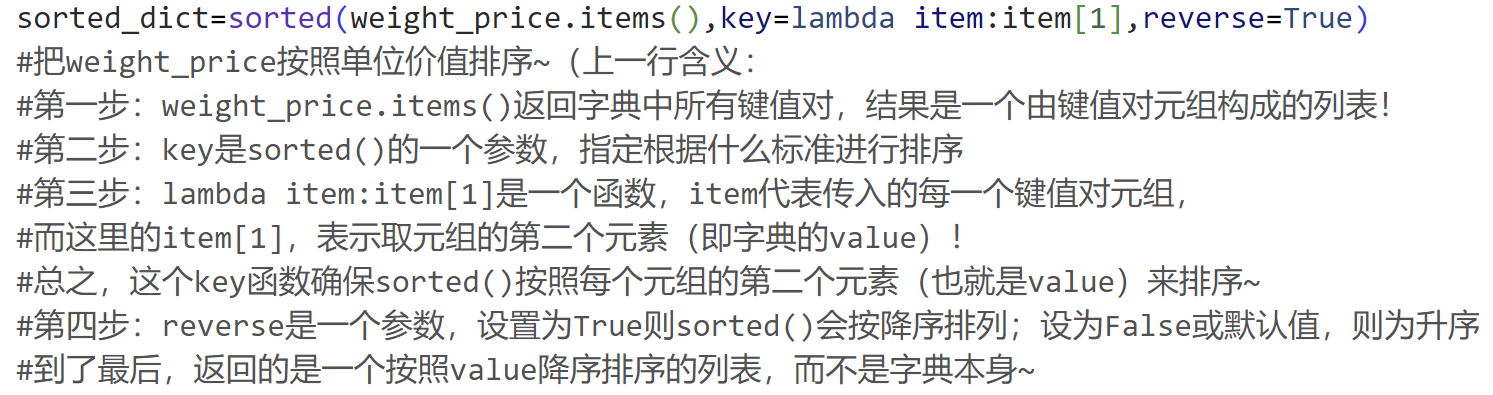
**lines=list(sys.stdin.read().split())** #一次性读取所有数据，并按空格划分开，合并为一个列表（本地测试需要ctrl+Z终止输入）

**import copy**

**copy.deepcopy（X）**#浅拷贝（copy.copy(X)）只复制了第一层的引用，如果修改了第二层，则原列表也会被修改！

而深拷贝复制了整个对象及**其嵌套**的所有部分，完全不共享内存！

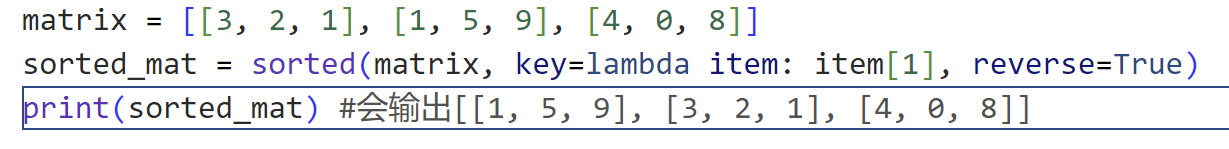
**琐碎**

****

**#对字典中的value进行排序**

（对第k个位置的元素排序，写item[k-1]即可）

这个方法也适用于对矩阵（列表套列表）里的某个固定位置的元素进行排序：

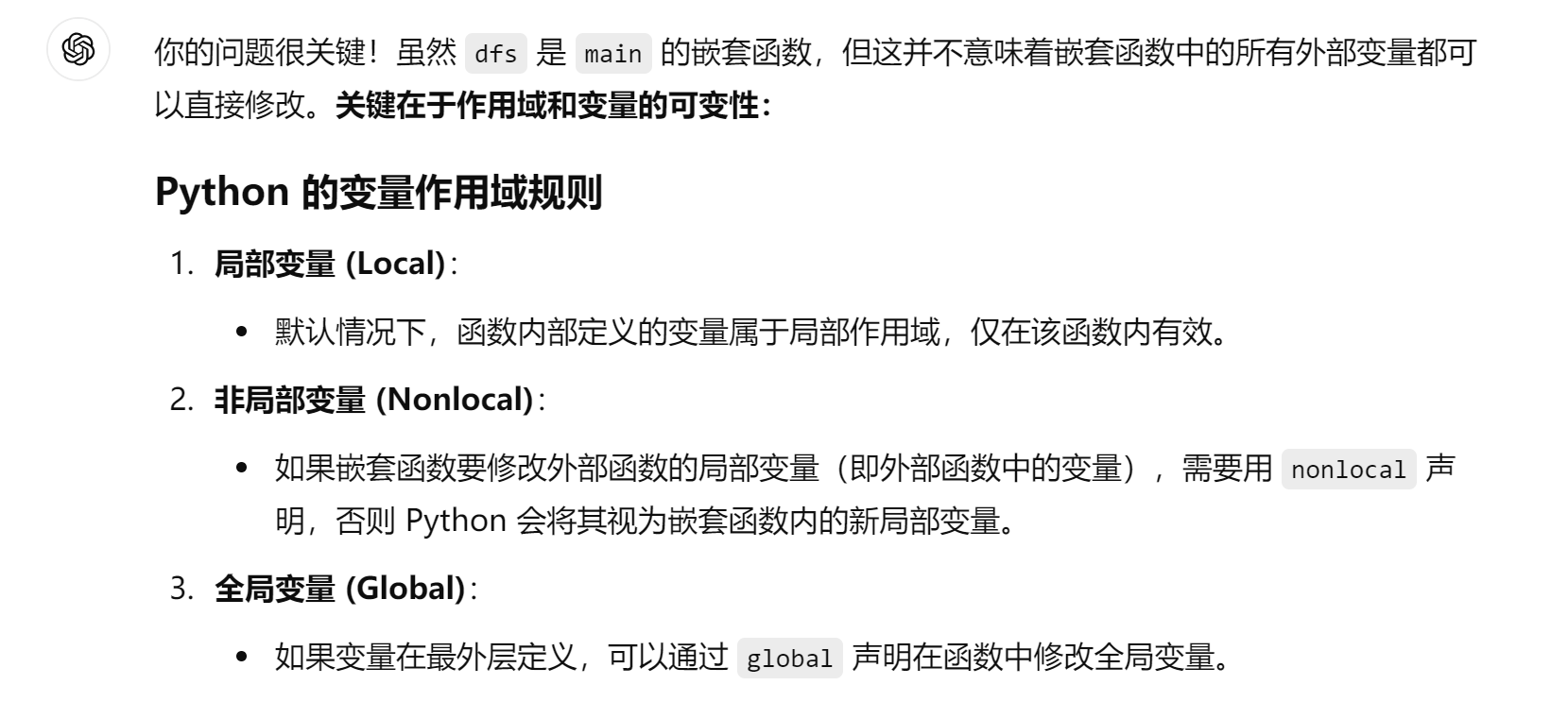


例如，这里就是对矩阵每行的第二个数倒序排列~

大函数里的小函数属于大函数的嵌套函数，可以直接访问大函数内的任何变量！**但是不能修改！！**

**Nonlocal：**解禁**不可修改但可引用**的**外层函数**的局部变量

**Global：引用并修改全局变量**（有global便可对全局变量修改，否则只会视为函数里的局部变量，对外界没有任何影响）

****



**永远正确的判断一个数是否平方数的方法：**

