**6.2.3 当事情开始变得艰难**

据说，肯尼迪总统的父亲约瑟夫·肯尼迪在教导他的孩子们的时候曾说过：当情况开始变得艰难，意志坚定的人将继续前行，虽然他从没调试过软件程序。这一小节将告诉你一些当软件调试出现瓶颈时的实用的小细节。

* 寻找常见的疑点。比如，你是否有以下情况：
* 将参数传递给函数的顺序是错误的
* 拼错了名字，比如：本应是大写字母却写成了小写
* 没有重置变量
* 未检验两个浮点数的值是否相等而不是近似相等（记住浮点数运算和你在学校里学的运算是不同的）
* 当你想要两个对象相等时（比如：id（L1）=id（L2）），未检验它们的值是否相等（比如：通过写表达式来比较两个列表L1==L2）
* 忘记某些内建函数有副作用
* 忘记（）会使对一个函数类型的对象的引用变为调用
* 创建一个无意义的对象名
* 犯一些你典型的错误
* 别再追问自己为什么程序不在做你想让它做的事，而是问问自己为什么程序会是现在这样。后者的答案更容易找出，而且可能是你解决程序问题迈出的第一步。
* 永远记住bug可能不在你能想到的地方，如果想得到，那么你早就解决它了。一种典型的找出bug的方法就是想想bug不可能出现在哪。就像夏洛克福尔摩斯说的那样：“排除掉其它所有因素，剩下的那个就必然是真相”。
* 尝试将这个问题告诉其他人。我们每个人都有自己的盲点，经常你不经意间把问题告诉其他人会发现一些你遗漏的点，比较好的方法是告诉他们你觉得bug不可能出现的地方。
* 不要相信你阅读的所有资料，尤其是那些文献。你的代码可能并不是像书中那样运行的
* 停止调试，写点东西。这可以帮你从不同角度解决问题。
* 放弃调试，明天再尝试继续。如果你已经陷在解决bug的泥潭中，这意味着你可以大大减少花在调试程序的上的时间，而是稍后及时解决bug。

**6.2.4 当你已经找出bug的时候**

当你觉得你已经找到了你的代码中的bug，那种想要开始编程测试修复的欲望便蠢蠢欲动。然而这时候最好先慢下来。记住我们的目的不是在修复一个bug，而是朝着没有bug的程序快速高效的前进。

想想这个bug是不是解释了所有症状的原因，还是它只是冰山一角。如果是后者，那你最好把这个bug和其他变化放在一起考虑。举个例子，假如你发现一个bug会导致一个列表的不稳定突变。你可以局部地避开这个问题（可能你会再复制一个此列表）或者你会用一个数组来替代这个列表（因为数组是不变的），可能会消除别处的相似bug。

在任何修改之前，尝试思考这个改动会产生的后果。这个改动会中断别打代码吗？它会增加程序变得复杂性吗？它会使别的部分的代码变得更简洁吗？

请确认你可以回到修改之前的状态。当你意识到一连串的修改让你比开始时离目标的距离更远了的时候，那简直太令人沮丧了。硬盘的空间一般都是足量的，用它来储存你的程序的各个版本吧。

最后，如果同时出现了多个错误，那就得考虑是否逐个找出错误并修复是更好的方法。想想是否有更好的组织程序的方法或者更简单更正确的算法，这会让你的感觉好得多。

**编程中的感悟**

我目前还是一个编程的菜鸟，只学过C++这一种语言，编程其实是一个相当艰苦的事情，我现在接触过的都是一些最简单的基本程序。对于那些大的工程根本想都不敢想，但是老师在上课的时候传达的一个理念，编程都是一边写一边学的，这给了我极大的信心。

想想当时学习C++的时候，也是从一个什么都不会的菜鸟，慢慢自己摸索出自己的一套门道，慢慢的觉得越来越得心应手。既然这样，我想在python的学习当中，如果能自己主动去钻研、探索，总是能把程序编的越来越好的。