**Python计算与编程简介**

**6.2.3 当进展遇到困难时**

据说，总统肯尼迪之父乔瑟夫.P.肯尼迪曾教导他的孩子们：“当事情变得艰难，坚强者才能前行。”然而，他从来没有对任何一个程序进行过调试。本小节包含了一些在调试过程中遇到困难时实用的建议。

下面列举一些常见的问题，例如：

• 传递函数的参数顺序错误；

• 拼写错名称，比如，在你应当输入大写字母时键入了小写字母；

• 没能重新初始化变量；

• 在两个浮点数比较中，使用相等（==）来代替近似相等（注意此处的浮点数与你在学校中所学的并不相同）；

• 使用值相等（例如使用L1==L2来比较两个列表）判断对象相等（例如id(L1)==id(L2)）；

• 忘记某些内建函数有副作用；

• 忘记（）会将函数类型对象的引用转换为函数调用；

• 不慎创建了一个别名；

• 或者其他你常犯的错误。

*不要再反复问自己为什么程序没有像你想象的一样运行*，你应该问自己程序为什么会这样运行。这个问题将会更容易解答，并且可能会成为找到方法修正你程序的第一步。

*记住错误可能并不在你想像的位置。*如果错误在那，你早就应该发现了它。一个确定错误所在位置的可行方法是想想最不可能出错的地方。正如夏洛克•福尔摩斯所说：“排除所有其它因素，剩下的那个一定是真相。”

*试着将问题解释给他人听。*我们都有各自的盲点，但在向他人解释问题时可以让你发现你遗漏的地方。可以试试解释为什么错误不可能出现在某些位置。

*不要相信任何你看到的。*尤其不要相信文档，代码可能不会像注释期望的一样运行。

*中止调试并开始写文档。*这将帮助你在另一个不同的角度靠近问题。

*暂时放下，明天再试。*这可能意味着错误会比预想的解决时间要晚，但你可能会因此花更少的时间去找到它。因此用时间换效率是可行的。（同学们，这是一个先做练习题的绝佳理由！）

**6.2.4 当你找到那个错误时**

当你认为在你的代码中找到了一个错误时，开始编写并修复这个错误的诱惑几乎是无法抵挡的。然而，最好能够先缓一缓行动。记住任务不是仅仅修复一个错误，而是快速且高效地完成没有错误的程序。

问问你自己，这个错误能否解释所有你观察到的问题，或者它仅仅是冰山一角。如果是后者，那么最好将这个错误和其它错误放在一起考虑。假设，你发现产生错误的原因是不小心修改了某个列表。你可以在本地修复这个错误（比如拷贝这个列表），或者你可以用元组来代替列表（因为元组是不可改变的），这样可能会排除掉代码中其余类似的错误。

在进行任何改动之前，尝试并理解预期的“修复”可能带来的后果。它会破坏其它地方吗？它会导致过高的复杂度吗？它有重构其它部分的可能性吗？

确保你可以回到你原来的位置。没有什么比进行一系列修改之后，发现自己离目标的距离越来越远，并且没有方法返回改动前的状态更加令人沮丧了。磁盘空间通常很充裕，记得利用它来保存你程序的旧版本。

最后，如果有许多无法解释的错误，也许你应该考虑是否要逐个发现并修复错误。或许你应该考虑是否有更好的方法组织你的程序，或者是否有更容易正确实现的代码。