Python 科学计算基础

第七章 错误处理和文件读写

2025 年 9 月 5 日

目录

错误的分类

程序发生的错误可分为以下三大类：

- 语法错误
- 逻辑错误
- 运行时错误

语法错误和运行时错误都有明确的出错信息，改正这两类错误比较容易。相比之下，改正逻辑错误的难度更大。

语法错误

语法错误是指程序违反了程序设计语言的语法规则。

例如语句

```
if 3>2
    print('3>2')
```

因冒号缺失导致语法解析器 (parser) 报错
"SyntaxError: invalid syntax"。

逻辑错误

逻辑错误是指程序可以正常运行，但结果不正确。

例如程序 7.1 的本意是对从 1 至 10 的所有整数求和，但由于对 range 函数的错误理解，实际是对从 1 至 9 的所有整数求和。

运行时错误

运行时错误也称为异常 (exception), 是指程序在运行过程中发生了意外情形而无法继续运行。

例如语句

```
a = 1/0 + 3
```

在运行过程中报错
"ZeroDivisionError: division by zero" 并终止。

避免发生错误

避免发生错误的基本方法列举如下。

1. 在编写较复杂程序之前应构思一个设计方案，把要完成的任务分解成为一些子任务，各子任务分别由一个模块完成。每个模块内部根据需要再进行功能分解，实现一些类和函数。这样使得整个程序有清晰合理的结构，容易修改和维护。
2. 对于每个类、函数和模块进行充分的测试。
3. 实现某一功能之前，先了解 Python 标准库和扩展库是否已经实现了该功能。如果有，则可以直接利用。这些库由专业软件开发人员实现，在正确性和运行效率上优于自己编写的程序。

查找逻辑错误的方法

对于比较简单的程序，可以通过反复阅读程序和输出中间步骤的变量值查找逻辑错误。

对于比较复杂的程序，上述方法的效率较低，更为有效的方法是设断点调试程序。

设断点调试程序

设断点调试程序所依据的原理是基于命令式编程范式编写的程序的运行过程可以理解为状态转换的过程。

- 状态包括程序中所有变量的值和正在运行的语句编号。
- 每条语句的运行导致某些变量的值发生变化，可以理解为发生了一步状态转换。
- 程序的输出结果是最终状态。从程序开始运行到结束经历了多次状态转换。如果程序结束时的输出结果有错，则错误必定发生在某一次状态转换中。
- 在可能出错的每一条语句之前设断点。程序运行到断点停下以后，单步运行程序以观察每一步状态转换并与预期结果对照，这样最终一定会找到出错的语句。

设断点调试程序

高斯消去法可用来求解线性方程组 $\mathbf{Ax} = \mathbf{b}$。通过一系列的初等行变换，高斯消去法将增广矩阵 $\mathbf{M} = (\mathbf{A}, \mathbf{b})$ 转变成一个上三角矩阵。设矩阵 $\mathbf{M}$ 的行数和列数分别为 m 和 n。

以实现高斯消去法的程序为例说明在 Spyder 中设断点调试程序的方法。程序的设计方案如下：

1. 外循环对第 j 列 $(0 \le j \le n-2)$ 运行，每次循环完成后第 j 列处于主对角线下方的元素变为 0。
   1.1 内循环对第 i 行 $(j+1 \le i \le m-1)$ 运行，将第 j 行乘以 $-a_{ij}/a_{jj}$ 的结果从第 i 行减去，目的是使得 a_{ij} 变为 0。

设断点调试程序

程序 7.2 的运行结果显示它对矩阵 **A** 输出了正确的结果，但对 **B** 输出的结果有错误并且报告在第 7 行出现除数 M[j, j] 为 0 的警告，这个警告提示了出错的可能原因。

为了能设置条件断点使程序在 A[j, j] 为 0 时暂停运行，在第 6 行后面加上语句

```
if abs(M[j, j]) < 1e-10:
    k = 0
```

这里语句 k = 0 不会影响程序的运行结果。

设断点调试程序

由于程序只对 **B** 的输出有错，需要先在第 16 行以 **B** 作为实参调用函数的语句处设置断点。在左边编辑窗口中的某一行设置断点的步骤是：首先用鼠标点击该行使得光标在该行跳动，然后点击 Debug 菜单的菜单项 “Set/Clear breakpoint” 或按下快捷键 F12。此时该行的行号的右边出现了一个红色的圆点，表示已在这一行设置断点。设置断点的操作类似电灯的开关，再进行一次以上操作则会取消断点。

在第 16 行设置断点以后，需要进行调试运行。和正常运行不同，调试运行会在当前设置的所有断点处停下，以便检查程序的中间状态。

设断点调试程序

Debug 菜单提供了多个菜单项用于调试运行。

- Set/Clear breakpoint：设置/清除断点
- Debug: 开始调试运行。调试运行和普通运行的区别在于遇到断点会停止。
- Step: 单步运行。如果当前语句是函数调用语句，不会进入函数内部。
- Step Into: 单步运行。如果当前语句是函数调用语句，会进入函数内部。
- Continue: 继续运行直至遇到下一个断点，然后停止。
- Stop: 结束调试运行。
- Clear breakpoints in all files: 清除所有断点。

设断点调试程序

点击 Debug 菜单的菜单项 “Debug” 使程序开始调试运行。此时第 16 行行号的右边出现一个蓝色箭头，表示已在这一行停下。点击右上角窗口下边界处的 “Variable explorer” 可使右上角窗口自动显示所有变量的值。

下一步需要在第 8 行设置断点。为了使程序继续运行直至下一个断点 (即第 8 行)，点击 Debug 菜单的菜单项 “Continue”。程序停下后，在控制台输入 “M, i, j”，结果显示 M[1, 1] 的值为 0，接下来在运行第 9 行时会发生除数 A[j, j] 为 0 的情形。

设断点调试程序

为了避免出现某行在主对角线上的元素为 0 的情形，可以将该行与其他行进行交换，使得主对象线上的元素的绝对值尽可能大。已经找到出错原因后，为了终止调试运行，点击 Debug 菜单的菜单项 “Stop” 或使用快捷键 Ctrl+Shift+F12。

程序 7.4 实现了以上修改。第 6 行在位于第 j 列的第 j 行及以下的元素中找到绝对值最大的元素所在行的索引值 p，这个索引值是相对于 j 的。第 7 行判断如果 p 大于 0，则说明绝对值最大的元素不在第 j 行，需要在第 8 行交换第 j 行和第 p+j 行。第 9 行判断若 M[j, j] 为 0 则退出外循环，否则运行内循环。

异常处理

若程序中某一语句块在运行过程中可能发生运行时错误，可以利用 if 语句对每一种出错情形进行判断和处理。当出错情形较多时，这些 if 语句导致程序结构不清晰并难于理解。

异常处理是比 if 语句更好的错误处理方式，体现在将程序的主线和错误处理分离。

异常处理

异常处理为每种出错的情形定义一种异常，然后将可能出错的语句置于 try 语句块中。如果这些语句在运行时出错，运行时系统会抛出异常，导致程序跳转到对应这种异常的 except 语句块中处理异常。else 语句块是可选的，包含不发生任何异常时必须运行的语句，必须位于所有 except 语句块之后。
例如程序 7.5 要求用户在命令行输入两个整数作为参数，然后计算它们的最大公约数。

sys.argv 是一个列表，存储了用户在命令行输入的所有字符串。索引值为 0 的字符串是程序的名称，其余字符串是用户输入的参数。

异常处理

如果用户输入的参数个数少于两个，或者某个参数不是整数，都会导致错误。

- 如果输入的参数个数少于两个，则读取 sys.argv[2] 导致 IndexError，第 13 行至第 14 行的 except 语句块对这种异常进行处理，即输出具体的出错信息。
- 如果某个参数不是整数，则 int 函数报错 ValueError，第 15 行至第 16 行的 except 语句块对这种异常进行处理。
- 如果用户输入了至少两个参数，并且前两个都是整数，则程序不会发生异常，第 12 行运行完成以后跳转到第 17 行至第 19 行的 else 语句块。else 语句块调用 gcd 函数计算前两个整数的最大公约数，然后输出。

异常处理

Python 标准库定义了很多内置异常类，它们都是 Exception 类的直接或间接子类，构成一个继承层次结构。这些异常类针对的错误类型包括算术运算、断言、输入输出和操作系统等。except 语句块中声明某一个异常类时，可以处理对应于该异常类或其子类的运行时错误。

用户在程序中可以使用这些内置异常类，也可以根据需要自定义异常类，自定义的异常类以 Exception 作为父类。例如程序 7.7 的第 1 行至第 7 行针对用户输入的整数为负数的出错情形定义了异常类 InputRangeError。

异常处理

InputRangeError 类只有一个属性，即出错信息。gcd 函数在第 10 行判断用户输入的两个整数中是否存在负数，若是则在第 11 行抛出 InputRangeError 异常，因为这种情形下 while 循环不会终止。异常导致 gcd 函数返回，该异常对象被第 29 行的 except 语句块捕获，然后在第 30 行输出出错信息。如果用户输入了至少两个参数，并且前两个都是正整数，则程序不会发生异常。

第 31 行至第 32 行的 finally 语句块是可选的。finally 语句块必须位于所有其他语句块之后。无论是否发生异常，finally 语句块都会运行，通常用于回收系统资源等善后工作。

文件

如果程序需要输入大量数据，应从文件中读取。如果程序需要输出大量数据，应写入文件中。

文件可分为两类：文本文件和二进制文件。

- 文本文件存储采用特定编码方式 (例如 UTF-8、GBK 等) 编码的文字信息，以字符作为基本组成单位，可在文本编辑器中显示内容。
- 二进制文件存储图片、视频、音频、可执行程序或其他格式的数据，以字节作为基本组成单位，在文本编辑器中显示为乱码。

打开和关闭文件

读写文件之前，先要使用"f = open(filename, mode)" 语句打开文件名为 filename 的文件并创建文件对象 f。其中 mode 是打开方式，可以是'r'(读)、'w'(写) 或'a'(追加)。mode 中如果有 'b' 表示以二进制方式打开。

读写一个文件 f 完成以后，需要使用"f.close() " 语句关闭文件。

使用"with open(filename, mode) as f:" 语句块打开的文件 f 会在语句块运行结束时自动关闭。

读写文本文件

从普通文本文件读取数据的基本方法是分析文件中的数据格式，采用合适的方法提取有效数据。

文件 rainfall.dat 记录了合肥市每月的平均降水量。文件中的有效数据在第 2 行至第 13 行，每行的格式是“月份名称+空格+降水量”。

程序 7.10 从文件 rainfall.dat 读取数据，然后计算每月平均降水量的最大值、最小值和平均值并写入文件 rainfall_stat.dat 中。

读写 CSV 文件

CSV 是一种简单的电子表格文件格式，其中的数据值之间用逗号分隔。办公软件 (如 Excel 和 LibreOffice Calc 等) 可以读入 CSV 文件并显示为电子表格。Python 标准库的 csv 模块可将 CSV 文件中的数据读入一个嵌套列表中，也可以将一个嵌套列表写入 CSV 文件中。

程序 7.12 从文件 scores.csv 中读取四位学生在三个科目上的考试成绩数据并存入嵌套列表 table 中，然后计算每位学生的总分和每个科目的平均成绩，再将这些计算结果添加到 table 中，最后将 table 写入文件 scores2.csv。

读写 JSON 文件

JSON 是 “JavaScript Object Notation” 的缩写，是一种常用的应用程序间数据交换格式。Python 标准库的 json 模块可将结构化数据 (字典、列表或它们的组合) 转换成为一个 JSON 格式的字符串并写入一个文件中，也可以从一个 JSON 文件中读取结构化数据。

程序 7.15 将一组通讯录数据 contacts 以 JSON 格式写入文件 contacts.json。

读写 pickle 文件

pickle 是一种 Python 定义的数据格式。Python 标准库的 pickle 模块可将结构化数据 (字典、列表或它们的组合以及类的对象) 转换成为一个字节流并写入一个二进制文件中，也可以从一个 pickle 文件中读取结构化数据。JSON 格式适用于使用多种程序设计语言编写的程序之间的数据交换，而 pickle 格式只适用于使用 Python 语言编写的程序之间的数据交换。

程序 7.18 将一组通讯录数据 contacts 以 pickle 格式写入文件 contacts.pickle。

读写 NumPy 数组的文件

np.savetxt 函数可以把一个 NumPy 数组保存为一个文本文件。
np.loadtxt 函数可以从一个文本文件中读入一个数组。

np.save 函数可以把一个数组保存为一个后缀为“npy”的二进制文件。np.load 函数可以从一个后缀为“npy”的二进制文件中读入一个数组。

np.savez 函数可以把多个数组保存为一个后缀为“npz”的二进制文件。np.savez_compressed 可以把多个数组保存为一个后缀为“npz”的压缩二进制文件。

程序 7.20

实验 7：错误处理和文件读写

本实验的目的是掌握以下内容：程序调试，异常处理和文件读写。

在 Blackboard 系统提交一个文本文件 (txt 后缀)，文件中记录每道题的源程序和运行结果。

1. 程序调试

设断点单步运行本章的程序程序 7.4、程序 7.10 和程序 7.12，观察变量的值。(本题无需提交)

2. 异常处理

编写程序读入用户在命令行输入的三个 float 类型的数值，判断它们是否能构成一个三角形的三条边。若可以，则使用以下公式计算并输出三角形的面积。公式中的 a, b, c 为三角形的三条边的长度。

$$area = \sqrt{s(s-a)(s-b)(s-c)},\ s = \frac{a+b+c}{2}$$

程序应处理以下类型的错误并输出出错信息：

- 用户在命令行输入的参数小于三个；
- 用户在命令行输入的三个参数不全是 float 类型；
- 用户在命令行输入的三个 float 类型的数值不能构成一个三角形的三条边，需要自定义异常类 InvalidTriangleError。

3. 文件读写

编写程序读入一个存储了程序 7.20 的文本文件，从中提取用户输入的代码然后输出到一个文本文件中。输出的文件的内容应为：

```
import numpy as np; a = np.arange(1, 16, 2)**2; a
b = a.reshape(2, 4); b
np.savetxt('D:/Python/dat/b.txt', b)
c = np.loadtxt('D:/Python/dat/b.txt'); c
np.save('D:/Python/dat/b.npy', b)
c = np.load('D:/Python/dat/b.npy'); c
np.savez('D:/Python/dat/ab.npz', a, b)
......
```