计算机技术

关注者 6 被浏览 6,572

Python异常处理中try，except用法?

try:
x = input('number 1:')
y = input('number 2:')
print(x/y)
except (ZeroDivisionError, TypeError, NameError) as e:
print("Error!!!")

运行结果不对

我是python3 求助 谢谢

1 个回答

默认排序

石溪
清华大学 计算机科学与技术硕士

11 人赞同了该回答

我来详细介绍异常编码的语法模式，try/except/else和try/finally。

先重新回顾一下try、except、else、finally几个关键字:

try后面紧跟着缩进的语句代码，代表此语句的主要动作：试着执行的程序代码。

然后是一个或多个**except**分句来识别要捕获的异常，except子句内定义try代码块内引发的异常处理器，

最后是一个可选的**else**分句，提供没发生异常时要执行的语句。

分别讨论下面的几种情形:

如果try代码块语句执行时的确发生了异常，python就跳出try，执行第一个符合引发异常的except子句下面的语句。当except代码块执行结束后，控制权就会到整个try代码块后继续执行。

如果异常发生在try代码块内，没有符合的except子句，异常就会传递到顶层，迫使python终止这个程序并打印默认的出错信息。

如果try首行底下执行的语句没有发生异常，python就会执行else行下的语句，控制权会在整个try语句下继续。

换句话说，except分句会捕获try代码块执行时所发生的异常，而else子句只在try代码块执行时不发生异常才会执行。

except是专注于异常处理器的：捕捉只在相关try代码块中的语句所发生的异常。尽管这样，因为try代码块语句可以调用写在程序其他地方的函数，异常的来源可能在try语句自身之外。

关于except子句的一些说明:

except子句可以用括号列出一组异常[except （e1,e2,e3）]，而如果except子句后没有列出异常名称，即except:时，会捕捉所有的异常类型。

但是，空except也会引发一些设计的问题，尽管方便，也可能捕捉和程序代码无关、意料之外的系统异常，而且可能意外拦截其他处理器的异常。例如，在python中，即使是系统离开调用，也会触发异常，而显然你通常会想让这些事件通过。

相关问题

在使用vector的时候，遇到_DEBUG_ERROR("vector iterators incompatible");? 6 个回答

关于a? b:c这个函数和printf一起用时导致的bug? 10 个回答

C中为什么要有返回值? 11 个回答

为什么在Python定义函数中想使用return语句返回一个list却没有任何返回值? 23 个回答

相关推荐

计算机网络技术基础任务驱动式教程（第 2 版）
36 人读过 阅读

计算机技能操作实训教程
缪建波主编
1 人读过 阅读

编码：隐匿在计算机软硬件背后的语言
10 人读过 阅读

python引入了一个替代方案来解决这个问题，捕获一个名为Exception的异常，几乎与一个空的except：具有相同的效果，但是忽略和系统退出相关的异常。

来看看try/else语句的作用

也许我们无法一眼看出else子句的用途，不过仔细想想，如果没有else，是无法知道控制流程是否通过了try语句，到底是没有异常引发，还是异常发生了且已被处理过了，不使用else的话很难分得清。

再来分析一下try/finally语句

try中包含了finally子句，python一定会在try语句后执行其语句代码块，无论try代码块执行时是否发生异常。

利用这个变体，python可先执行try首行下的语句代码块。接下来发生的事情，取决于代码块中是否发生异常：

如果try代码块运行时没有异常发生，python会跳至执行finally代码块，然后在整个try语句后继续执行下去

是，在出现异常时，仍能利用finally关闭文件和断开服务器连接。

最后我们来看最完整的形式：try/except/else/finally

```
try:
    main-action
except Exception1:
    handler1
except Exception2:
    handler2
...
else:
    else-block
finally:
    finally-block
```

我们从头梳理一遍：

就像往常一样，这个语句中的main-action代码会先执行。如果该程序代码引发异常，那么所有except代码块就会逐一测试，寻找与抛出的异常相符的语句，如果引发的异常是Exception1，就会执行handler1，如果引发的的异常是Exception2，就会执行handler2，以此类推，如果没有引发任何异常，将会执行else-block。而无论之前发生了什么，当main-action代码块完成的时候，而任何引发的异常都已经处理后，finally-block就会执行。事实上，即使异常处理器或者else-block内有错误发生而引发新的异常，finally-block内的程序代码依然会执行。就像之前所说的那样，finally子句并没有终止异常：当finally-block执行的时候，如果异常还存在，就会在finally-block代码块执行后继续传递，而控制权会跳至程序其他地方，如我们的默认的顶层处理器。

最后我们叮嘱一下，try语句必须有一个except或一个finally，else是可选的，但是如果有else，则必须至少有一个except。

关于数据科学更系统、更深入的探讨可进入我们的专栏《Python数据科学之路》：

酱油哥：来吧，一起踏上Python数据科学之路
zhuanlan.zhihu.com

本专栏模仿美剧剧集编排分为五季，第一季：**Python编程语言核心基础**、第二季：**Python数据分析基本工具**、第三季：**Python语言描述的数学基础**、第四季：**机器学习典型算法专题**、第五季：**实战热点深度应用。**

编辑于 2018-06-23

写回答