第八章 异常处理

8.1 异常的概念

异常：在程序运行时产生的例外、违例情况被称为异常。如果不能在异常发生时及时妥善地处理它们，程序将崩溃，无法继续运行下去。

在Python中，异常是以对象的形式实现的。 BaseException类是所有异常类的基类，而其子类Exception类则是除了SystemExit、GeneratorExit和KeybaordInterrupt三个系统级异常之外所有内置异常类和用户自定义异常类的基类。

8.2 异常的抛出

程序在运行过程中出现错误而无法正常运行时，会陷入异常。此外，Python也为用户提供了raise关键字以人为地抛出指定类型的异常。

使用raise语句手动抛出异常在程序调试、自定义异常等场景下有诸多应用。注意，Python不会自动引发自定义异常，这要求程序开发者为自定义的异常编写合理的异常抛出代码 。

8.2 异常的抛出

异常名称	描述	异常名称	描述
Exception	普通错误的基类	AttributeError	对象没有这个属性
IOError	输入/输出操作失败	IndexError	序列中没有此索引
KeyError	映射中没有这个键	NameError	未声明/初始化对象
SyntaxError	Python语法错误	SystemError	一般解释器系统错误
ValueError	传入无效参数	ZeroDivisionError	除0异常
ImportError	导入模块异常	TypeError	类型异常
ReferenceError	引用不存在对象异常	AssertionError	assert语句触发的异常

表1常见异常列表

8.3 异常的捕获

当异常发生时，就需要捕获并处理相应的异常。try...except语句是捕获处理异常的常用语句之一，其语法如下：

- try:
- 	<语句>
- except <异常类型>:
- 	<语句>

其中，except子句可以有多个，当try后的语句执行时发生异常，Python就跳过try代码段余下的部分，执行第一个匹配该异常的except子句，异常处理完毕，控制流就通过整个try...except语句（除非在处理异常时又引发新的异常）。

except后面可以放置多个异常类型（以逗号分割）以表明若多个异常中至少发生一个，则执行该部分异常处理代码，若不放置任何异常类型，则代表可匹配所有的异常类型。

8.4 异常的处理

Python还提供了else和finally两个子句，以用于try…except异常处理语句。其语法如下：

```
try:
    可能抛出异常的代码段
except (Exception1, Exception2, ...) as e:
    若发生以上多个异常中的一个，则执行这块代码
        e可以获取解释器传递而来的错误信息
        except可以写多个
else:
    若没有异常，则执行这块代码
finally:
    无论异常是否发生均执行该块代码
```

8.5 自定义异常

Python如同很多高级程序设计语言一样允许用户自定义异常类型，用于描述Python异常体系中没有涉及的异常情况。通过前面的学习，可知除3个系统级异常外，其他异常类型均是Exception子类；而定义一个自定义异常也十分简单，只需要定义一个继承了Exception类的派生类即可。Python不会自动为用户抛出或处理任何自定义异常，因而用户需要使用raise语句在合理的场合手工触发异常。

在使用自定义异常类型时，经常需要在捕获异常的同时获取该异常的实例（例如，上例中的e），以获取存储在异常实例中的数据，这只需要在异常类型后放置一个实例名即可。

8.6 使用断言

在程序调试过程中，用户经常希望知道某个条件在运行时是否为真（例如，储蓄账户余额始终为正），并在条件不成立时提示编码者错误出现的位置。Python中提供了断言assert语句，以检测某个表达式是否为真，当表达式不成立时，会引发AssertionError异常。

同时，还可以通过assert语句传递提示信息给AsserttionError异常，以提示编码者错误发生的部位和可能的原因。