异常

1. try..except

我们**尝试**读取用户的一段输入。按 **Ctrl-d**，看一下会发生什么。

```
>>> s = raw_input('Enter something --> ')
Enter something --> Traceback (most recent call last):
  File "<stdin>", line 1, in ?
EOFError
```

Python 引发了一个称为 `EOFError` 的错误，这个错误基本上意味着它发现一个不期望的 文件尾 （由 **Ctrl-d** 表示）
接下来，我们将学习如何处理这样的错误。

我们可以使用 `try..except` 语句来处理异常。我们把通常的语句放在 try-块中，而把我们的错误处理语句放在 except-块中。

```
#!/usr/bin/python
# Filename: try_except.py

import sys

try:
    s = raw_input('Enter something --&gt; ')
except EOFError:
    print '\nWhy did you do an EOF on me?'
    sys.exit() # exit the program
except:
    print '\nSome error/exception occurred.'
    # here, we are not exiting the program

print 'Done'
```

输出

```
$ python try_except.py
Enter something --&gt;
Why did you do an EOF on me?

$ python try_except.py
Enter something --&gt; Python is exceptional!
Done
```

它如何工作

我们把所有可能引发错误的语句放在 `try` 块中，然后在 `except` 从句/块中处理所有的错误和异常。`except` 从句可以专门处理单一的错误或异常，或者一组包括在圆括号内的错误/异常。如果没有给出错误或异常的名称，它会处理所有的 错误和异常。对于每个 `try` 从句，至少都有一个相关联的 `except` 从句。
如果某个错误或异常没有被处理，默认的 Python 处理器就会被调用。它会终止程序的运行，并且打印一个消息，我们已经看到了这样的处理。

你还可以让 `try..catch` 块关联上一个 `else` 从句。当没有异常发生的时候，`else` 从句将被执行。
我们还可以得到异常对象，从而获取更多有个这个异常的信息。这会在下一个例子中说明。

2. 引发异常

你可以使用 `raise` 语句 引发 异常。你还得指明错误/异常的名称和伴随异常触发的 异常对象。你可以引发的错误或异常应该分别是一个 `Error` 或 `Exception` 类的直接或间接导出类。

```
#!/usr/bin/python
# Filename: raising.py

class ShortInputException(Exception):
    '''A user-defined exception class.'''
    def __init__(self, length, atleast):
        Exception.__init__(self)
        self.length = length
        self.atleast = atleast

try:
    s = raw_input('Enter something --&gt; ')
    if len(s) &lt; 3:
        raise ShortInputException(len(s), 3)
    # Other work can continue as usual here
except EOFError:
    print '\nWhy did you do an EOF on me?'
except ShortInputException, x:
    print 'ShortInputException: The input was of length %d, \
          was expecting at least %d' % (x.length, x.atleast)
else:
    print 'No exception was raised.'
```

输出

```
$ python raising.py
```

```
Enter something --&gt;
Why did you do an EOF on me?

$ python raising.py
Enter something --&gt; ab
ShortInputException: The input was of length 2, was expecting at least 3

$ python raising.py
Enter something --&gt; abc
No exception was raised.
```

它如何工作

这里，我们创建了我们自己的异常类型，其实我们可以使用任何预定义的异常/错误。这个新的异常类型是 `ShortInputException` 类。它有两个域——`length` 是给定输入的长度，`atleast` 则是程序期望的最小长度。
在 `except` 从句中，我们提供了错误类和用来表示错误/异常对象的变量。这与函数调用中的形参和实参概念类似。在这个特别的 `except` 从句中，我们使用异常对象的 `length` 和 `atleast` 域来为用户打印一个恰当的消息。

3. try..finally

假如你在读一个文件的时候，希望在无论异常发生与否的情况下都关闭文件，该怎么做呢？这可以使用 `finally` 块来完成。注意，在一个 `try` 块下，你可以同时使用 `except` 从句和 `finally` 块。如果你要同时使用它们的话，需要把一个嵌入另外一个。

```
#!/usr/bin/python
# Filename: finally.py

import time

try:
    f = file('poem.txt')
    while True: # our usual file-reading idiom
        line = f.readline()
        if len(line) == 0:
            break
        time.sleep(2)
        print line,
finally:
    f.close()
    print 'Cleaning up...closed the file'
```

输出

```
$ python finally.py
Programming is fun
When the work is done
Cleaning up...closed the file
Traceback (most recent call last):
  File "finally.py", line 12, in ?
    time.sleep(2)
KeyboardInterrupt
```

它如何工作

我们进行通常的读文件工作，但是我有意在每打印一行之前用 `time.sleep` 方法暂停 2 秒钟。这样做的原因是让程序运行得慢一些（Python 由于其本质通常运行得很快）。在程序运行的时候，按 **Ctrl-c** 中断/取消程序。
我们可以观察到 `KeyboardInterrupt` 异常被触发，程序退出。但是在程序退出之前，finally 从句仍然被执行，把文件关闭