14.9 捕获异常后抛出另外的异常¶

问题¶

你想捕获一个异常后抛出另外一个不同的异常，同时还得在异常回溯中保留两个异常的信息。

解决方案¶

为了链接异常，使用 `raise from` 语句来代替简单的 `raise` 语句。 它会让你同时保留两个异常的信息。例如：

```
>>> def example():
...     try:
...             int('N/A')
...     except ValueError as e:
...             raise RuntimeError('A parsing error occurred') from e
...
>>> example()
Traceback (most recent call last):
  File "<stdin>", line 3, in example
ValueError: invalid literal for int() with base 10: 'N/A'
```

上面的异常是下面的异常产生的直接原因：

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 5, in example
RuntimeError: A parsing error occurred
>>>
```

在回溯中可以看到，两个异常都被捕获。 要想捕获这样的异常，你可以使用一个简单的 `except` 语句。 不过，你还可以通过查看异常对象的 `__cause__` 属性来跟踪异常链。例如：

```
try:
    example()
except RuntimeError as e:
    print("It didn't work:", e)

    if e.__cause__:
        print('Cause:', e.__cause__)
```

当在 `except` 块中又有另外的异常被抛出时会导致一个隐藏的异常链的出现。例如：

```
>>> def example2():
...     try:
...             int('N/A')
...     except ValueError as e:
...             print("Couldn't parse:", err)
...
>>>
>>> example2()
Traceback (most recent call last):
  File "<stdin>", line 3, in example2
ValueError: invalid literal for int() with base 10: 'N/A'
```

在处理上述异常的时候，另外一个异常发生了：

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 5, in example2
NameError: global name 'err' is not defined
>>>
```

这个例子中，你同时获得了两个异常的信息，但是对异常的解释不同。 这时候，`NameError` 异常被作为程序最终异常被抛出，而不是位于解析异常的直接回应中。

如果，你想忽略掉异常链，可使用 `raise from None`：

```
>>> def example3():
...     try:
...             int('N/A')
...     except ValueError:
...             raise RuntimeError('A parsing error occurred') from None
...
>>>
example3()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 5, in example3
RuntimeError: A parsing error occurred
>>>
```

讨论¶

在设计代码时，在另外一个 `except` 代码块中使用 `raise` 语句的时候你要特别小心了。 大多数情况下，这种 `raise` 语句都应该被改成 `raise from` 语句。也就是说你应该使用下面这种形式：

```
try:
   ...
except SomeException as e:
   raise DifferentException() from e
```

这样做的原因是你应该显示的将原因链接起来。 也就是说，`DifferentException` 是直接从 `SomeException` 衍生而来。这种关系可以从回溯结果中看出来。

如果你像下面这样写代码，你仍然会得到一个链接异常， 不过这个并没有很清晰的说明这个异常链到底是内部异常还是某个未知的编程错误。

```
try:
   ...
except SomeException:
   raise DifferentException()
```

当你使用 `raise from` 语句的话，就很清楚的表明抛出的是第二个异常。

最后一个例子中隐藏异常链信息。 尽管隐藏异常链信息不利于回溯，同时它也丢失了很多有用的调试信息。 不过万事皆平等，有时候只保留适当的信息也是很有用的。