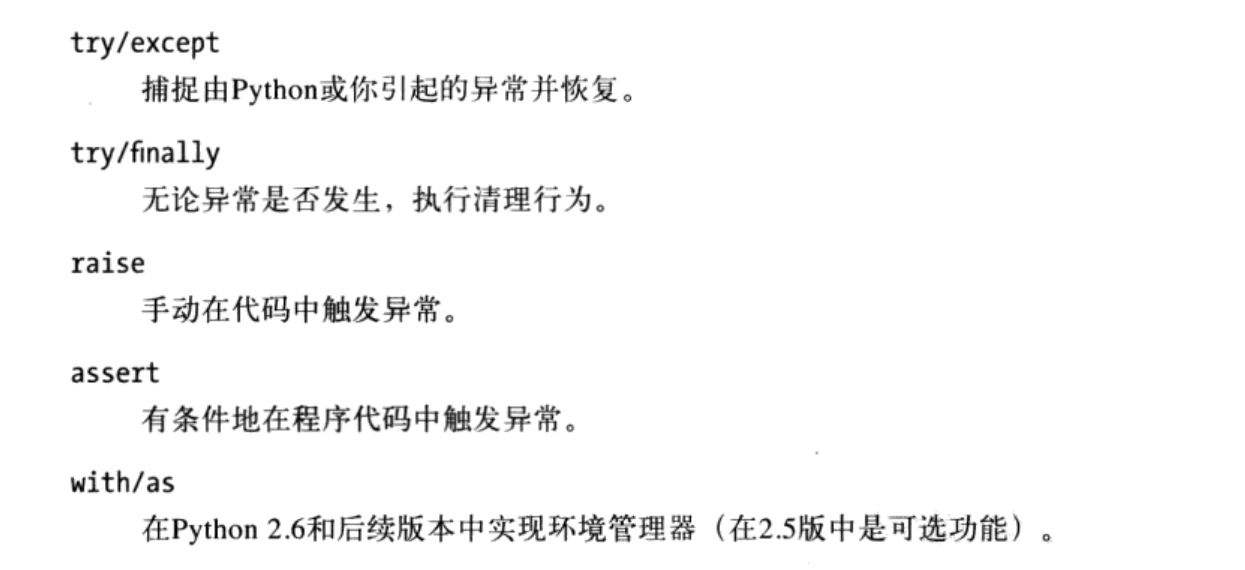
异常基础：

几种异常方式：



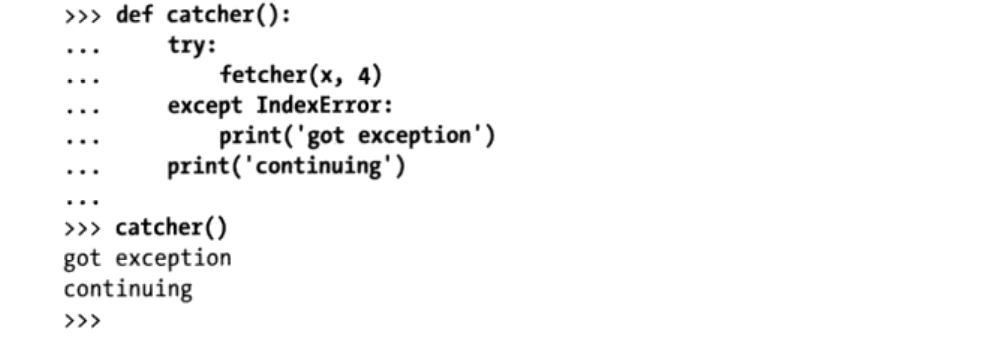
注意这里的名词：异常处理器。Try语句内部的语句块

使用异常的原因：

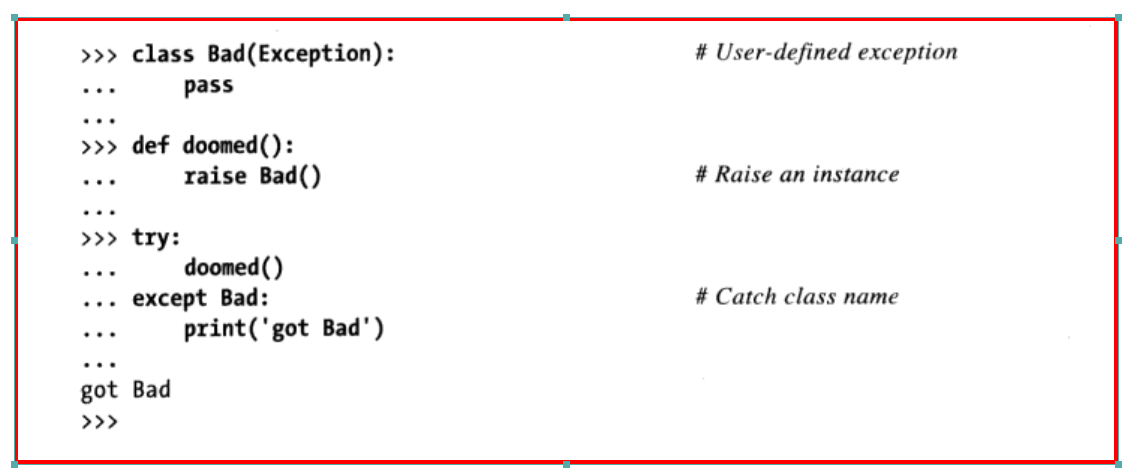
1. 错误处理
2. 终止行为
3. 事件通知
4. 特殊行为处理

默认的异常处理：将异常一层一层的向上进行传递，最后输出打印错误信息。同时打印出错的堆栈消息。

举例，格式：



用户定义异常:异常的定义使用类的模式进行定义，一般最原始的基类为exception

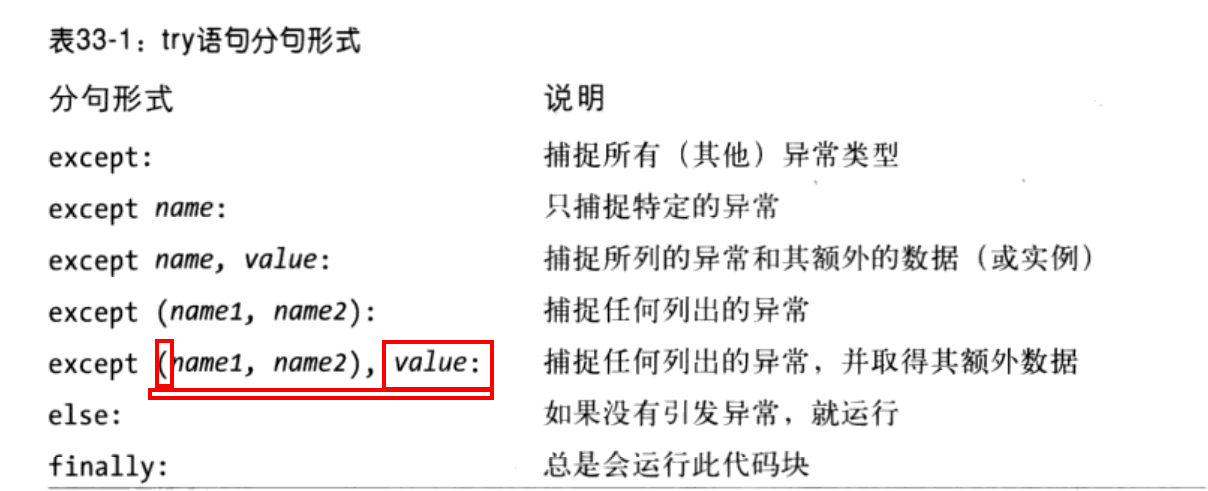


Try/finally组合：

情况一：在try的代码块里面发异常，则代码执行finally，且代码执行到finally这里就断了。

情况二：在try的代码块这里面没有发生异常，代码执行到finally这里，然后代码接着try语句继续执行。

Try/exception/else语句，else在没有发生异常的时候执行。



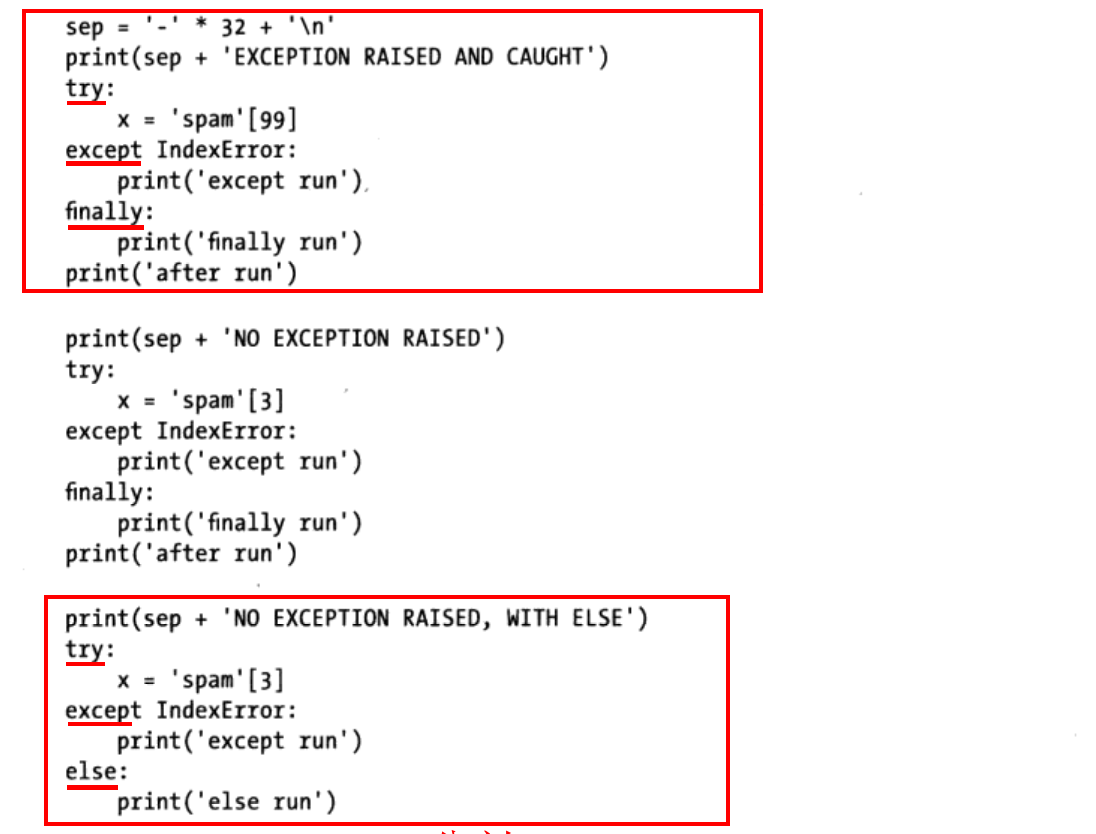
捕获其他所有异常的方式：

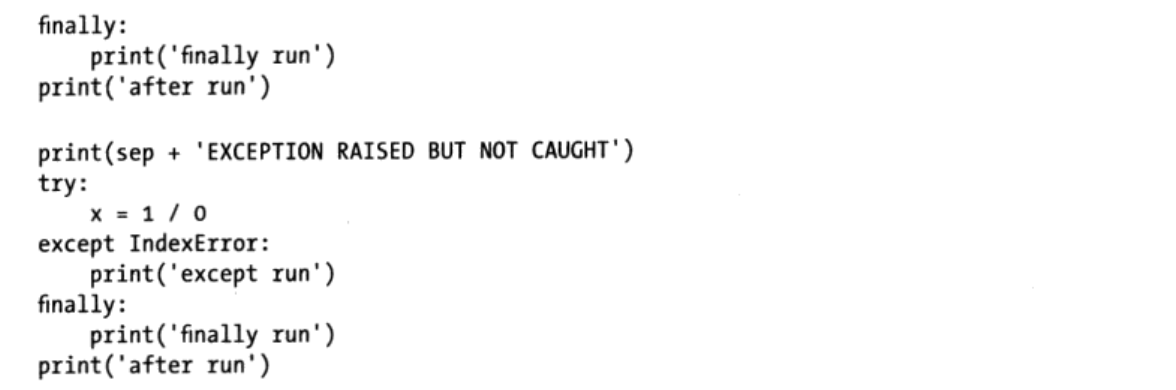
方式一：except:

方式二：except：Exception

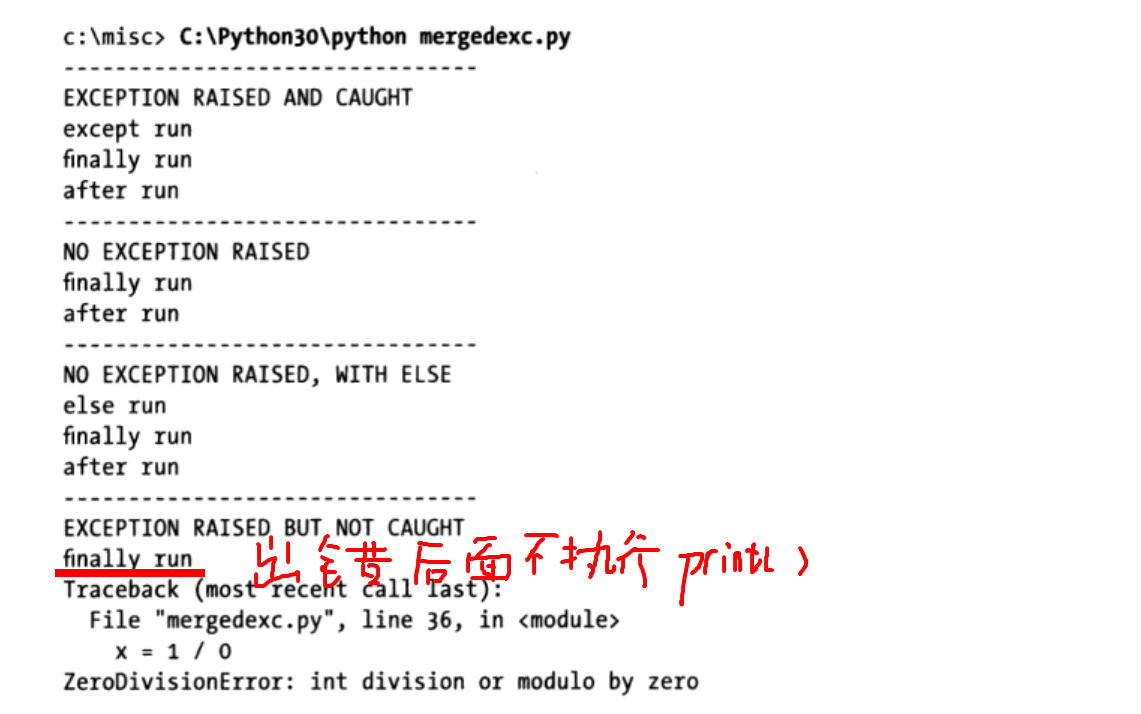
这种捕获所有异常的方式不好的原因：除了会捕获一些未知的异常外，还会捕获如系统异常等情况，这样在分辨具体出错的异常的时候就很麻烦，公司禁止使用这种异常捕获方式

Try/exception/else/finally语句合并举例





上面语句的执行结果：注意在最后一个出现错误，执行finally，之后就在那里一直执行finally。

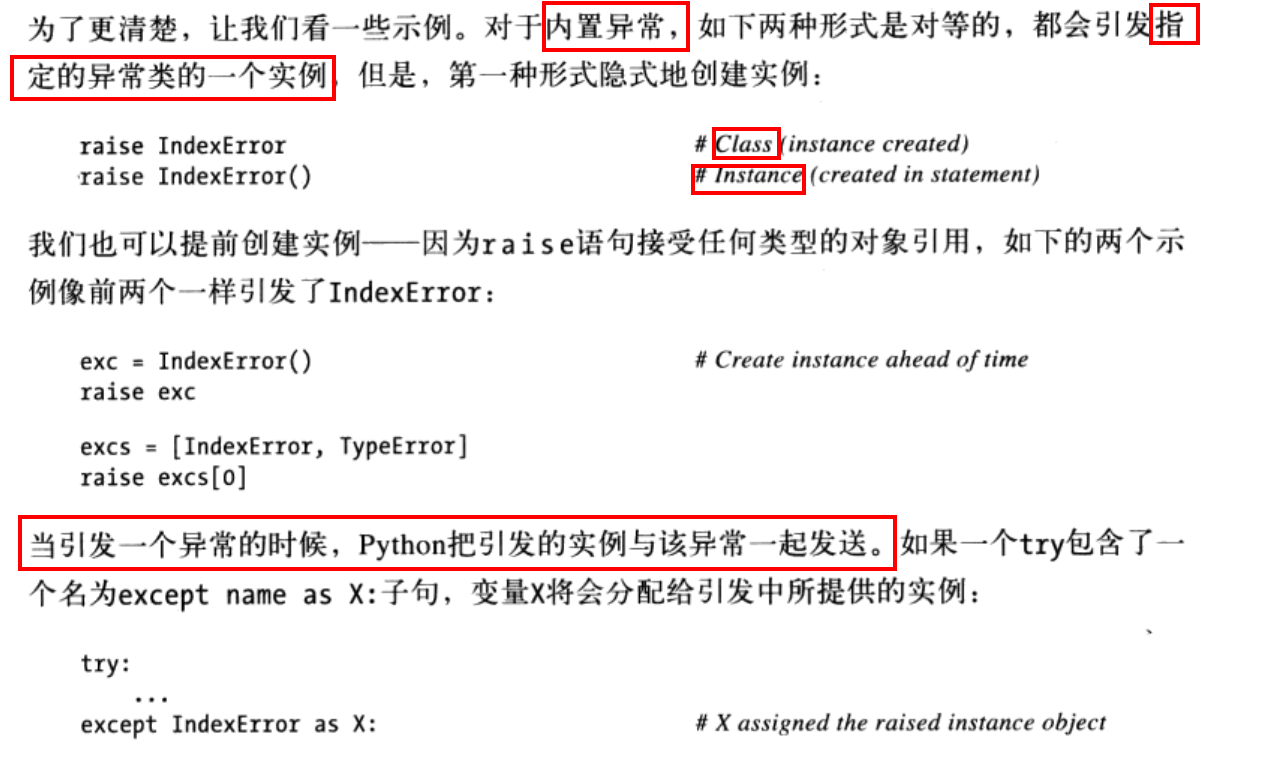


Raise语句进行异常的触发：

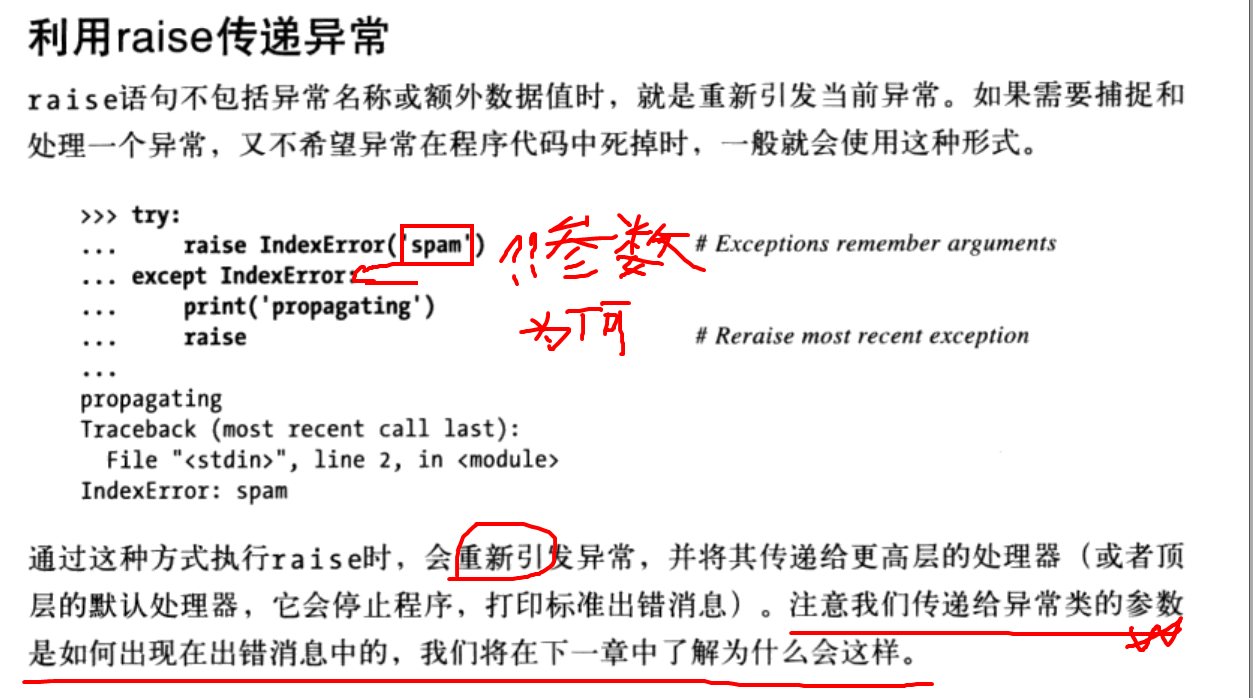
触发内置异常的两种写法：

方式一：raise class，这会在触发异常的时候隐形的创建一个异常实例

方式二：raise instance。 这会直接触发异常实例。



这里抛出一个疑问：为什么会这样呢



Assert语句，这个语句通常用来判断程序的运行状态