**异常的抛错机制：**

1、如果在运行时发生异常，解释器会首先看当前的执行环境（函数或者类）有没有捕获这个异常。

2、要是在当前函数里没有找到的话，它会将异常传递给上层的调用函数，看看那里有没有处理。

3、如果在最外层（全局“main”）还是没有找到的话，解释器就会退出，同时打印出traceback信息在标准错误里。

**注： 当发生异常的时候：**

1. 代码终止，退出
2. 异常信息会打印在标准错误里

**需要捕获异常的场景：**

1. 允许代码发生某种类型的异常不需要终止代码，可以捕获异常，继续往下走
2. 代码发生异常的时候，需要将异常信息汇报给上层系统

例如部署代码脚本，部署机器上百个，不可能通过登录机器去查找报错

**捕获异常5种方式：**

1. try: … expect: …. 捕获所有的异常信息，然后处理。打印异常信息可以：print traceback.format\_exc()
2. try: … expect Exception, ex: …. 捕获异常，并得到异常信息，只有异常信息，没有traceback信息、也没有错误类型。
3. try: … expect 特定的异常： print 捕获特定的异常信息，其他形式的异常不捕获。
4. 不处理异常， try...finally...子句用来表达这样的情况：

finally: 我们不管程序捕捉到的是什么错误，无论错误是不是发生，这些代码“必须”运行，比如文件关闭，释放锁(例如zookeeper锁)，把数据库连接连接返还给连接池等。

5） try: xxx except:xxx finally:xxx

**关于捕获异常信息：**

并不是自己代码越多捕获异常信息就越好，python本身是有异常机制，正常情况下我们并不捕获异常信息就行，这样代码抛异常了，我们会直观地看到。

捕获异常信息另一层面其实就是隐层异常信息，当然我们可以把它打印出来，做对应的处理，但是这不会终止程序，程序进行下去有出错的可能。

参考捕获异常的场景

**常见python异常类型：**

|  |  |
| --- | --- |
| 异常 | 描述 |
| NameError | 使用一个未定义的对象 |
| ZeroDivisionError | 除数为0 |
| SyntaxError | 语法错误 |
| IndexError | 下标索引超出序列边界 |
| KeyError | 试图访问字典里不存在的键 |
| IOError | 输入输出错误（比如你要读的文件不存在） |
| AttributeError | 试图访问一个对象没有的属性 |
| ValueError | 传给函数的参数类型不正确，比如给int()函数传入字符串形 |
| TypeError | 传入对象的类型不符 |
| ImportError | 无法引入模块或者包，基本是路径的问题 |
| IndentationError | 语法错误，例如代码没有对齐 |
| KeyboardInterrupt | Ctrl + C 终止异常 |
| invalid syntax | 语法不对： print ‘fa’: |

**我自己想定义异常类型怎么破？**

class **CommonException**(Exception):

def **\_\_init\_\_**(*self*, value):

super(CommonException, *self*).\_\_init\_\_(

*"Common Exception. Error Message %s "* % (value))

*self*.value = value

def **\_\_str\_\_**(*self*):

return repr(*self*.value)

只需把**CommonException 换成自己定义的类名即可**

**场景举例：**

**在web编程的时候，出现以下情况，我要将详细结果汇报给调用者**

1. **传入参数不合法（传入参数不够、参数value不合法）**
2. **例如 调用外部系统接口出错等**
3. **自己代码报错 500错误**

**我自己想抛出异常怎么破?**

raise：

例如 ： raise **CommonException** (‘this is a TypeError Exception’)