第九章　异常

9.1.1 认识异常

异常就是一个事件，该事件会在程序执行过程中有语法等错误的时候发生，异常会影响程序的正常执行。

通常在Python无法正常处理程序时就会发生一个异常，程序会终止执行。

当Python程序发生异常时，我们需要检测捕获处理它，否则程序会终止执行。

Python中的异常处理机制是面向对象的，常见异常都封装成了异常类，部分常见异常如下表所示。

Python标准异常(部分)

异常名称	描述
BaseException	所有异常的基类
Exception	常规错误的基类
StopIteration	迭代器没有更多的值
StandardError	所有的内建标准异常的基类
ArithmeticError	所有数值计算错误的基类
FloatingPointError	浮点计算错误
OverflowError	数值运算超出最大限制
ZeroDivisionError	除(或取模)零 (所有数据类型)
AttributeError	对象没有这个属性
EOFError	没有内建输入,到达EOF 标记
IOError	输入/输出操作失败
OSError	操作系统错误
FileNotFoundError	文件未找到错误（IOError的子类)
PermissionError	权限错误（IOError的子类)

异常名称	描述
ImportError	导入模块/对象失败
LookupError	无效数据查询的基类
IndexError	序列中没有此索引(index)
KeyError	映射中没有这个键
MemoryError	内存溢出错误(对于Python 解释器不是致命的)
NameError	未声明/初始化对象 (没有属性)
UnboundLocalError	访问未初始化的本地变量
ReferenceError	弱引用(Weak reference)试图访问已经垃圾回收了的对象
RuntimeError	一般的运行时错误
SyntaxError	Python 语法错误
IndentationError	缩进错误
SystemError	一般的解释器系统错误
TypeError	对类型无效的操作
ValueError	传入无效的参数

9.1.2 处理异常

如何检测并处理异常呢?

try/except语句用来检测try语句块中的错误，从而让except语句捕获异常信息并处理。任何出现在try语句范围内的异常都可以被检测到。若不想在异常发生时中断程序，则应在try里检测它，并在except中捕获和处理它。

有4种模式的try语句： **try —except语句、try—except—finally语句、try—except—else语句、try (with)—except语句。**

(1) try — except语句

```
try:
    [语句块]
except 异常类型 [as 异常新名称]:
   出现异常(exception)后的处理代码
```

9.1.2 处理异常

示例

未进行异常处理：

```
import math

x = float(input('请输入一个数：'))
y = math.fabs(x)
print(x, y)
```

```
请输入一个数：abc
Traceback (most recent call last):
  File "C:/Users/ruanz/PycharmProjects/new/ch9/demo_09_01_01.py", line 9, in
<module>
    x = float(input('请输入一个数：'))
ValueError: could not convert string to float: 'abc'
```

9.1.2 处理异常

添加异常处理：

```
import math

while True:
    try:
        x = float(input('请输入一个数：'))
        break
    except Exception as ex:
        print('输入格式有误！', ex)

y = math.fabs(x)
print(x, y)
```

Exception 替换为ValueError，则异常类型更具体。

```
请输入一个数：abc
输入格式有误！ could not convert string to float: 'abc'
请输入一个数：123a
输入格式有误！ could not convert string to float: '123a'
请输入一个数：-5.8
-5.8 5.8
```

9.1.2 处理异常

一个try语句还可以和多个except语句搭配，对感兴趣的异常进行检测处理。

如果try语句包含的异常没有出现在后面跟着的except语句中的时候，则程序直接报错输出异常的类型。

```
z = 0
while True:
    try:
        x = float(input('请输入一个被除数：'))
        y = float(input('请输入一个除数：'))
        z = x / y
        break
    except ValueError as ex1:
        print('输入格式有误！', ex1)
    except ZeroDivisionError as ex2:
        print('ZeroDivisionError！', ex2)

print('{0}/{1} = {2:.4f}'.format(x, y, z))
```

```
请输入一个被除数：abc
输入格式有误！ could not convert string to float: 'abc'
请输入一个被除数：10
请输入一个除数：0
ZeroDivisionError！ float division by zero
请输入一个被除数：10
请输入一个除数：3
10.0/3.0 = 3.3333
```

9.1.2 处理异常

如果当我们不确定在try语句块中会出现哪一种异常的时候，我们可以在except后面不跟具体的异常类型。

上面这种处理方式不推荐采用，因为这样做会掩藏程序员未想到的所有未曾处理过的错误。

有一点一定要注意：try语句检测范围一旦出现了异常，剩下的其它语句将不会被执行，程序立即跳转到except执行异常处理程序，其它程序代码块就不再执行了。

9.1.2 处理异常

另外，如果要对多个异常进行统一的处理，采用如下的语法格式：

```
try:
    [语句块]
except (Exception1，Exception2，Exception3，…):
    出现异常(exception)后的处理代码
```

在上述语法中，多个异常之间用逗号“，”隔开。

9.1.2 处理异常

(2) try – except – finally语句：

try:
    [语句块]
except 异常类型 [as 异常新名称]:
    出现异常(exception)后的处理代码
finally:
    无论如何都会被执行的代码

```
import math

while True:
    try:
        x = float(input('请输入一个数：'))
        break
    except ValueError as ex:
        print('输入格式有误！', ex)
    finally:
        print("-"*10)  # 分隔线

y = math.fabs(x)
print(x, y)
```

```
请输入一个数：abc
输入格式有误！  could not convert string to float: 'abc'
----------
请输入一个数：-5.8
----------
-5.8 5.8
```

9.1.2 处理异常

(3) try – except - else语句：

```
try:
    [语句块]
except except 异常类型 [as 异常新名称]:
    出现异常(exception)后的处理代码
else:
    没有异常后被执行的代码
```

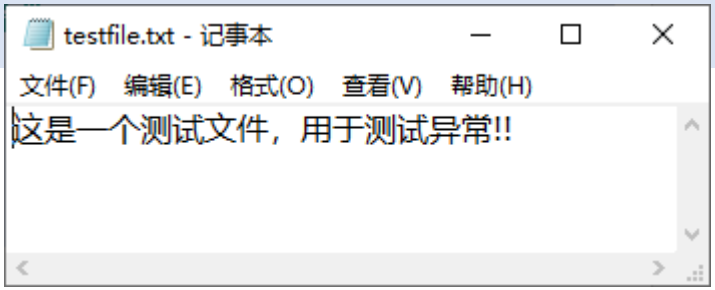
如果一旦检测到try语句块中有任何异常，程序就会根据异常类型跳转到except处执行对应异常类型的处理代码，最后终止程序的执行；

如果在try语句块中没有检测到任何异常，程序在执行完try语句块里的代码后，跳转到else处执行里面的代码。

9.1.2 处理异常

例如：

```
try:
    fh = open("testfile.txt", "w")
    fh.write("这是一个测试文件，用于测试异常!!")
except IOError as ex:
    print("Error: ", ex)
else:
    print("内容写入文件成功")
    fh.close()
```



运行结果：

```
内容写入文件成功
```

将testfile.txt文件属性修改为“只读” 后，再次运行程序，结果如下：

```
Error:  [Errno 13] Permission denied: 'testfile.txt'
```

9.1.2 处理异常

(4) try (with)—except语句：

```
try:
    with <语句> as name:
        [语句块]
except except 异常类型 [as 异常新名称]:
    出现异常(exception)后的处理代码
```

在语法中可见，with语句出现在try语句块中，一般情况下就不用再写finally语句块了。**使用with语句的最大好处是减少代码量**，比如当我们对文件操作时忘记了关闭文件操作，则with语句会自动执行关闭文件操作。

```
try:
    with open("testfile2.txt", "w") as fh:
        fh.write("这是一个测试文件，用于测试异常!!")
        print("内容写入文件成功")
except IOError as ex:
    print("Error: ", ex)
```

9.1.3 处理异常嵌套

异常处理可以进行嵌套。

```
path = r"H:\testfile.txt"    # 文件路径，假定不存在H盘
try:
    with open(path, "w") as fh:
        try:
            fh.write("这是一个测试文件，用于测试异常!!")
        except Exception as ex1:
            print("Error: 文件写入错误！\n", ex1)
except (FileNotFoundError, PermissionError) as ex2:
    print(ex2)
else:
    print("内容写入文件成功")
```

运行结果：

```
[Errno 2] No such file or directory: 'H:\\testfile.txt'
```

将文件路径改为path = **r"testfile.txt"**后，再次运行程序，结果如下：

```
[Errno 13] Permission denied: 'testfile.txt'
```

第九章　异常

Python中的异常处理流程是：

当程序运行try语句块检测到异常时，立即终止有异常的语句的执行，跳转到匹配该异常的except子句执行异常处理代码，异常处理完毕后，如果有finally语句就执行该语句块中的代码，最后终止整个程序的执行，如果没有finally语句就直接终止整个程序的执行。

如果检测到异常，但没有该异常匹配的except子句，分两种情形：如果有finally语句就执行该语句块中的代码，最后终止整个程序的执行；如果没有finally语句就直接终止整个程序的执行。

如果在try语句块中没有检测到异常，程序执行完try语句块后，如果有else语句块就执行里面的内容最后控制流就通过整个try语句，没有else语句控制流就直接通过整个try语句。

第九章　异常

自定义一个简单的异常类，如下所示：

```
class MyError(Exception):
    pass   # 空操作（什么也不做），是一个占位标记，这里表示类没有方法
```

自定义异常类应该是典型的继承自Exception类，通过直接或间接的方式。

使用raise语句**主动抛出自定义的异常**，**从而产生了一个异常。**

```
raise MyException(defineexceptname)
```

MyException——自定义异常的类型

defineexceptname——自定义异常的说明。

也可以抛出标准异常对象，例如：raise Exception("除数为零异常！")

同时，我们也可以结合try—except主动抛出自定义的异常，如下例代码所示：

```
class MyError(Exception):
    pass


pwd = input('请输入密码：')
try:
    if pwd!='abc':
        raise MyError('测试自定义的MyError异常：密码错误')
except MyError as ex:
    print(ex)
```

运行结果：

输入：123
输出：测试自定义的MyError异常：密码错误

第九章　异常

断言语句 assert 用于检测某个条件表达式是否为真。如果表达式为False，则会抛出一个AssertionError异常。可以在脚本执行时加上python -O 参数来跳过assert检测。

assert expression1 ["," expression2]

- 如果只接一个表达式，则相当于：

```
if __debug__:
    if not expression1: raise AssertionError
```

- 如果接两个表达式，则相当于：

```
if __debug__:
    if not expression1: raise AssertionError(expression2)
```

其中：

- __debug__：如果程序运行时不带-O参数，则为True；反之则为False。
- AssertionError：是一个继承Exception类的异常类

assert 语句经常用于参数被使用前的检查操作，如果检查未通过则直接抛出异常及早发现错误，避免明显错误的参数还被往后传递。

```
int_var = int(input("please enter a positive number:"))
# 如果输入的数值不大于0，断言失败，抛出异常
assert int_var > 0

int_var = int(input("please enter a positive number:"))
# 如果输入的数值不大于0，断言失败，抛出异常
assert int_var > 0, "输入有误，要求输入正数！"
```

assert 语句经常用于参数被使用前的检查操作，如果检查未通过则直接抛出异常及早发现错误，避免明显错误的参数还被往后传递。

```
int_var = int(input("please enter a positive number:"))
# 如果输入的数值不大于0，断言失败，抛出异常
assert int_var > 0
print(int_var)
```

```
please enter a positive number:>? 5
5
```

```
please enter a positive number:>? -5
Traceback (most recent call last):
  File "<input>", line 3, in <module>
AssertionError
```

assert 语句经常用于参数被使用前的检查操作，如果检查未通过则直接抛出异常及早发现错误，避免明显错误的参数还被往后传递。

```
int_var = int(input("please enter a positive number:"))
# 如果输入的数值不大于0，断言失败，抛出异常
assert int_var > 0, "输入有误，要求输入正数！"
print(int_var)
```

```
please enter a positive number:>? 5
5
```

```
please enter a positive number:>? -5
Traceback (most recent call last):
  File "<input>", line 3, in <module>
AssertionError: 输入有误，要求输入正数！
```

由于assert本质上还是raise，所以一样可以使用try-except捕获，而不是说断言错误程序就一定会终止。

```
try:
    int_var = int(input("please enter a positive number:"))
    # 如果输入的数值不大于0，断言失败，抛出异常
    assert int_var > 0
except:
    print(f"sorry, please enter a positive number")
print(f"what you enter is: {int_var}")
```

```
please enter a positive number:>? 5
what you enter is: 5
```

```
please enter a positive number:>? -5
sorry, please enter a positive number
what you enter is: -5
```

由于assert本质上还是raise，所以一样可以使用try-except捕获，而不是说断言错误程序就一定会终止。

```
while True:
    try:
        int_var = int(input("please enter a positive number:"))
        assert int_var > 0, 'The number must be positive. Please try again.'
        break
    except Exception as e:
        print(f"Exception: {e}")
print(f"what you enter is: {int_var}")
```

```
please enter a positive number:>? a
Exception: invalid literal for int() with base 10: 'a'
please enter a positive number:>? -5
Exception: The number must be positive. Please try again.
please enter a positive number:>? 5
what you enter is: 5
```

第九章　异常

9.5.1 利用try-except处理除数为零的异常

9.5.2 自定义异常的使用

9.5.3 raise关键字的使用

9.5.4 内置异常处理语句的使用

第九章　异常

在Python当中，若程序在运行时出错，系统会自动地在出错的地方生成一个异常对象，而后系统会在出错的地方向后寻找是否有对这个异常对象处理的代码，如果没有，系统会将这个异常对象抛给其调用函数，这样层层抛出，如果在程序主函数中仍然没有对这个异常对象处理的代码，系统会将整个程序终止，并将错误的信息输出。

第九章　异常

习题：

1. Python异常处理结构有哪几种形式?
2. 异常和错误是同一概念吗？为什么?
3. 定义一个函数isTriangle(a, b, c)，用于判断三个数是否构成三角形，要求如果参数为负数则抛出异常，并进行必要的测试。(提示：函数中使用raise或assert抛出异常，测试代码中使用try...except捕获并处理异常)

感谢聆听