1.raise抛出异常

'''

Python中，用异常对象表示异常情况

'''

#x=1/0

#raise Exception("first exception msg")

#raise ArithmeticError("一个和数值有关的异常")

from boto.codedeploy.exceptions import InvalidRoleException

#raise InvalidRoleException(2,"和Role有关的异常")

'''

python内置异常类

Exception 所有异常类的基类

AttributeError 属性引用或赋值失败时抛出的异常

OSError 当操作系统无法执行任务时抛出的异常

keyError 当使用序列中不存在的键值时抛出的异常

IndexError 当使用序列中不存在的索引时抛出的异常

NameError 在找不到名字或变量时抛出异常

SyntaxError 在代码微错误形式时触发

TypeError 类型错误抛出的异常

ValueError 在使用不合适的值时抛出的异常

ZeroDivisionError 除零异常

'''

dict={}

#print(dict['a'])

list=[]

print(list[2])

2.自定义异常

class MyException(Exception):#所有的异常都继承自Exception

pass

n=10

if n<20:

raise MyException("hello")

3.try\_except捕捉异常

'''

try:

statement1

statement2

...

except:

statement1

statement2

...

'''

x=None

while True:

try:

if x==None:

x=int(input("input 分子:"))

y=int(input("请输入分母:"))

print('x/y={}'.format(x/y))

break;

except:#遇到异常直接跳到except代码块

print("分母不能为0，请重新输入。")

finally:#无论是否异常都会执行

print("finally.")

4.捕捉多个异常

'''

一个try, 多个异常

try:

...

except 异常类1:

...

except 异常类2:

...

except 异常n:

...

'''

#操作数或计算结果为负数抛出的异常

class NegativeException(Exception):

pass

#操作数为0时抛出的异常

class ZeroException(Exception):

pass

class SpecialCalc:

def add(self,x,y):

if x<0 or y<0:

raise NegativeException("x或y小于0")

return x+y

def sub(self,x,y):

if x-y<0:

raise NegativeException("x-y<0")

return x-y

def mul(self,x,y):

if x==0 or y==0:

raise ZeroException("x或y等于0")

return x\*y

def div(self,x,y):

return x/y

while True:

try:

myCalc=SpecialCalc()

expr=input("输入要计算的表达式，例如add(1,2):")

if expr==":exit":

break;

result = eval('myCalc.{}'.format(expr))

print("计算结果:{}".format(result))

except NegativeException:

print("NegativeException,负数异常")

except ZeroException:

print("操作数为0异常")

except ZeroDivisionError:

print("分母为0")

except:

print("其他异常")

5.一个代码块处理多个异常

class CustomException1(Exception):

pass

class CustomException2(Exception):

pass

class CustomException3(Exception):

pass

import random

def raiseException():

n = random.randint(1,3)

if n==1:

raise CustomException1

elif n==2:

raise CustomException2

else:

raise CustomException3

try:

raiseException()

except(CustomException1,CustomException2,CustomException3):

print("\*\*\*\*\*\*处理异常程序\*\*\*\*\*\*")

6.捕捉异常对象

'''

try：

...

except 异常类 as e

except(异常类1，异常类2,...)

'''

#操作数或计算结果为负数抛出的异常

class NegativeException(Exception):

pass

#操作数为0时抛出的异常

class ZeroException(Exception):

pass

class SpecialCalc:

def add(self,x,y):

if x<0 or y<0:

raise NegativeException("x或y小于0")

return x+y

def sub(self,x,y):

if x-y<0:

raise NegativeException("x-y<0")

return x-y

def mul(self,x,y):

if x==0 or y==0:

raise ZeroException("x或y等于0")

return x\*y

def div(self,x,y):

return x/y

while True:

try:

myCalc=SpecialCalc()

expr=input("输入要计算的表达式，例如add(1,2):")

if expr==":exit":

break;

result = eval('myCalc.{}'.format(expr))

print("计算结果:{}".format(result))

except(NegativeException,ZeroException,ZeroDivisionError) as e:

print(e)

7.捕捉异常中的else用法

while True:

try:

x=int(input("输入分子:"))

y = int(input("输入分母"))

value = x/y

print("x/y=",value)

except Exception as e:

print(e)

else:

print("ok")

break

8.finally用法

#不是必须存在，存在时可以不要except；没有finally，必须有except

#finally一定会执行

'''

1.在try中的语句正常执行完后，会执行finally中的代码；

2.在try中抛出异常，跳到except执行后，会执行finally子句

3.当try字节跳出时(break,return)，也会执行finally子句；

4.当在finally释放资源，最好用try/except语句，防止抛出异常

'''

def fun1():

try:

print("try in fun1 ok")

finally:

print("fun1 finally")

fun1()

def fun2():

try:

raise Exception

except:

print("fun2抛出异常")

finally:

print("fun2 finally")

fun2()

def fun3():

try:

return 20

finally:

print("fun3 finally")

fun3()

def fun4():

try:

x=1/0

except ZeroDivisionError as e:

print(e)

finally:

print("fun4 finally")

try:

del x

except Exception as e:

print(e)

fun4()

9.异常跟踪：

def fun1():

raise Exception("fun1 raise exception")

def fun2():

fun1()

def fun3():

fun2()

fun3()

C:\ProgramData\Anaconda3\python.exe F:/development/py/li\_ning/ch9/demo9\_跟踪异常.py

Traceback (most recent call last):

File "F:/development/py/li\_ning/ch9/demo9\_跟踪异常.py", line 28, in <module>

fun3()

File "F:/development/py/li\_ning/ch9/demo9\_跟踪异常.py", line 26, in fun3

fun2()

File "F:/development/py/li\_ning/ch9/demo9\_跟踪异常.py", line 24, in fun2

fun1()

File "F:/development/py/li\_ning/ch9/demo9\_跟踪异常.py", line 22, in fun1

raise Exception("fun1 raise exception")

Exception: fun1 raise exception

Process finished with exit code 1