day08

python内置方法 isinstance()/issubclass() 对对象和类进行判断

1.isinstance(obj,cls)

#判断对象obj是否由类cls创建 如果为真则返回True 如果为假则返回Flase， 如果当前类为派生类（子类）那么对于基类（父类）也是同样适用的

例：

*#-\*- coding:utf-8 -\*-  
#/usr/bin/env python  
#定义类cls***class** cls(object):  
 **pass***#定义类sub 继承类cls***class** sub(cls):  
 **pass**obj = sub()  
*#查看对象是否属于类cls（基类）***print 'obj is cls object? %s'**%isinstance(obj,cls)  
*#查看对象是否属于类sub（派生类）***print 'obj is sub object? %s'**%isinstance(obj,sub)  
*#查看对象是否属于int类***print 'obj is int object? %s'**%isinstance(obj,int)

输出结果：

obj is cls object? True

obj is sub object? True

obj is int object? False

#判断sub是否为super的派生类（子类） 如果为真则返回Ture 如果为假则返回Flase

2.issubclass(sub,super)

例：

*#-\*- coding:utf-8 -\*-  
#/usr/bin/env python  
#定义类cls***class** cls(object):  
 **pass***#定义类sub 继承类cls***class** sub(cls):  
 **pass***#查看sub是否是cls的派生类* **print** issubclass(sub,cls)

*#查看sub是否是int的派生类*  
**print** issubclass(sub,int)

输出结果：

True

False

python异常处理 try…exect

作用：

处理程序异常、抛出异常、友好输出异常

1. 如何捕获异常、处理异常

**try**:  
 **pass  
except** ValueError:  
 **pass  
except** Exception:  
 **pass  
else**:  
 **pass  
finally**:  
 **pass**

try

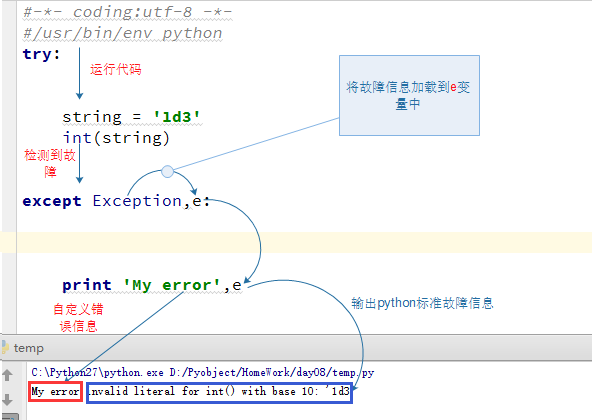
*代码块检测*

*#检测到相应故障进行处理 其中下面的e（标准故障信息）可以指定也可以不指定*

except ErrorType，e：

*对故障进行处理*

具体如下图所示:



1. 异常的种类

上面的故障处理模块只检测了Exception异常类，其实这个故障是所有异常的基类（父类） 无论什么异常都可以用他进行处理，不过在程序中要处理指定异常那么就必须使用相关的派生类（子类）比如：

IndexError 下标错误

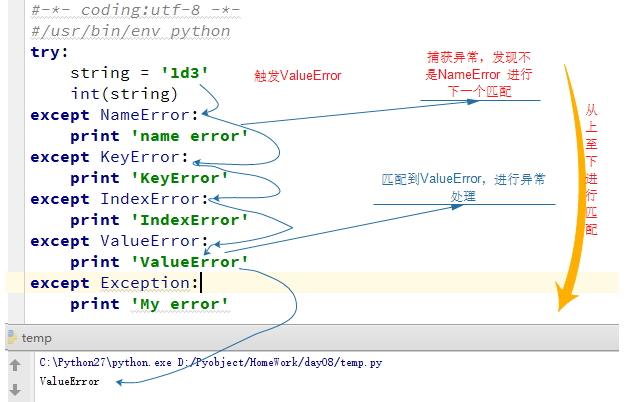
KeyError key错误

NameError 名称错误

ValueError 值错误

…

那么如果我定义了NameError 和 ValueError Exception等等这些异常处理标签如果我有一个ValueError的异常该如何进行处理呢？请看下图



异常会从上到下一条条匹配异常处理标签直到匹配到为止 ExceptError在正常情况下放在异常处理程序的最末端 来处理未知异常，如果有些异常需要特殊操作为，放在“except ExceptError”的上面。

1. 自定义异常

了解自定义异常之前 首先需要理解两个概念：

1. 类中的方法\_\_str\_\_(self) 执行类的时候会返回其中的内容 用于输出异常信息

例如：

*#定义Alan\_class类***class** Alan\_class(Exception):

*#定义\_\_str\_\_方法*  
 **def** \_\_str\_\_(self):

*#指定输出信息*  
 **return 'Alan Error'**

*#执行类*Alan\_class **print** Alan\_class()

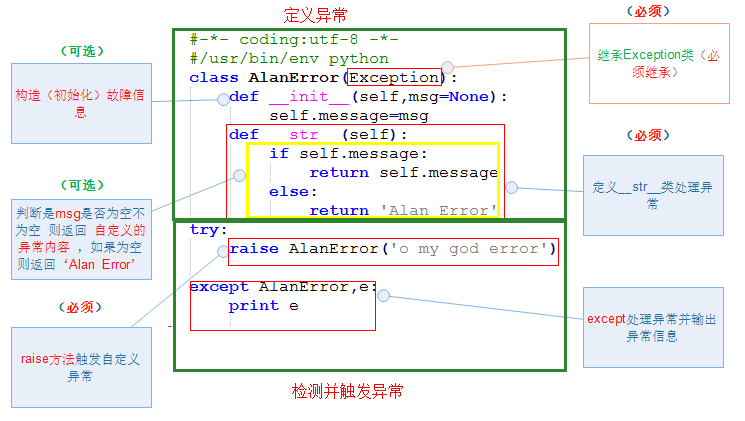
输出结果为：

Alan Error

1. raise 抛出异常

比如 执行 raise ValueError() 那么就会抛出一个 ValueError值错误的异常

了解完上面的两个知识点下面开始了解如何进行定义自己的异常



通过上图我们简单的梳理下：

第一步：定义异常

class 异常名称 （Exception）：

def \_\_init\_\_(self,msg)

self.message=msg

def \_\_str\_\_(self): //定义\_\_str\_\_函数处理异常

判断是否输入了自定义的异常信息

if 输入了：

return ‘自定义异常信息’

else：

rerun ‘默认异常信息’

第二步:抛出并处理异常

raise 异常名称（‘自定义异常信息’） //触发异常

except 异常名称，异常信息

print 异常信息

1. assert 断言

**try**:  
 **assert 'Lisa'** == **'Alan'  
except** Exception:  
 **print 'name not same'**

执行结果：

name not same

反射：

#模块module中是否包含func方法 包含返回True不包含返回Flash

hasattr(module,func)

#获取module模块中的func方法

getattr(module,func)

#在module中设置一个name = value的选项（设置内存中的name）

setattr(module,name,value)

#删除module中删除name（删除内存中的name）

delattr(module,name)

反射操作类、对象

动态模块导入

单例模式：

单例模式 就是单个实例（内存中只有一个实例）

*#-\*- coding:utf-8 -\*-  
#/usr/bin/env python***class** Alan(object):  
 \_\_static\_class = None  
 **def** test(self):  
 **print 'test'** @classmethod  
 **def** search(cls):  
 **if** cls.\_\_static\_class:  
 **return** cls.\_\_static\_class  
 **else**:  
 \_\_static\_class = Alan.test()  
 **return** cls.\_\_static\_class  
obj = Alan()  
obj2 = Alan()  
**print** id(obj.search)  
**print** id(obj2.search)

执行结果：

40727248

40727248

socket 套接字

什么是socket

socket 别名 网络套接字，指网络通信句柄 其实就是一堆网络信息（ip+协议+端口） 建立起的链接称之为socket

SocketServer

必须定义一个继承SocketServer.BaseRequestHandler的类，必须定义一个handle方法

class MyServer():

def handle():

pass

server = SocketServer.ThreadingTCPServer((‘host’,port),MyServer)

server.server\_forever()