笔记

> 版权：智泊AI
>
> 作者：Jeff

一、错误和异常

1.概念

> 两种容易辨认的错误
>
> 语法错误：一些关于语法的错误【缩进】 Error
>
> 异常：代码完全正确，但是，程序运行之后，会报出 的错误 Exception
>
> exception/error
>
> 代码演示：

```
# 常见的错误
# print(num)   # NameError: name 'num' is not defined

list1 = [12,34,23,45]
print(list1[0])
print(list1[1])
print(list1[2])
print(list1[3])
print(list1[4])   # IndexError: list index out of range

print("over")
```

> 异常特点：当程序在执行的过程中遇到异常，程序将会终止在出现异常的代码处，代码不会继续向下执行
>
> 解决问题：越过异常，保证后面的代码继续执行【实质：将异常暂时屏蔽起来，目的是为了让后面的代码的执行不受影响】

2.常见的异常

> NameError:变量未被定义
>
> TypeError:类型错误
>
> IndexError:索引异常
>
> keyError:
>
> ValueError:
>
> AttributeError:属性异常
>
> ImportError:导入模块的时候路径异常

> SyntaxError:代码不能编译
>
> UnboundLocalError:试图访问一个还未被设置的局部变量

3.异常处理方式【掌握】

> 捕获异常
>
> 抛出异常

3.1捕获异常

try-except:

```
# 第一种方式:try-except
try:# 尝试执行一段代码
    num = 12 / 0
    print(num)
except:# 如果程序报错了,就执行这里面的代码
    pass
    # print("报错了")
    # 在真实的项目中,通常会在这里记录错误日志,方便后续排查,进行业务的更改

print("hello world")
```

try-except:

```
# 第二种方式:(常用)
try:
    num = 12 / 0
    print(num)
except Exception as e:
    print(e,type(e))  #division by zero <class 'ZeroDivisionError'>
print("haha")
```

try-except-else

> 语法:
>
> try:
>
> 可能存在异常的代码
>
> except 错误表示码 as 变量:
>
> 语句1
>
> except 错误表示码 as 变量:
>
> 语句2
>
> 。。。
>
> else:

语句n

说明：

a.try-except-else的用法类似于if-elif-else

b.else可有可无，根据具体的需求决定

c.try后面的代码块被称为监测区域【检测其中的代码是否存在异常】

d.工作原理：首先执行try中的语句，如果try中的语句没有异常，则直接跳过所有的except语句，执行else；如果try中的语句有异常，则去except分支中进行匹配错误码，如果匹配到了，则执行except后面的语句；如果没有except匹配，则异常仍然没有被拦截【屏蔽】

代码演示：

```
# 第三种方式：try-except-else   except和else只会执行一个区间
try:
  num = 24 / 1
  print(num)
except Exception as e:
  print("报错了,走这里")
else:
  print("不报错,走这里")

print("lele")
```

try-except-finally

语法：

try:

可能存在异常的代码

except 错误表示码 as 变量：

语句1

except 错误表示码 as 变量：

语句2

。。。

finally:

语句n

说明:不管try中的语句是否存在异常，不管异常是否匹配到了except语句，finally语句都会被执行

作用：表示定义清理行为，表示无论什么情况下都需要进行的操作

代码演示：

```
# 第四种方式：try-except-finally   不管上面的程序有没有报错,都会执行finally
try:
  num = 24 / 0
  print(num)
except Exception as e:
  print("报错了,走这里")
finally:
  print("不管你上面对与错,都要来我这报道!")

print("嘿嘿!")
```

3.2抛出异常

触发异常我们可以使用raise语句自己触发异常

当然，我们手动让程序引发异常，很多时候并不是为了让其崩溃。事实上，raise 语句引发的异常通常用 try except（else finally）异常处理结构来捕获并进行处理。

> raise抛出一个指定的异常对象
>
> 语法：raise 异常对象 或者 raise
>
> 说明：异常对象通过错误表示码创建，一般来说错误表示码越准确越好
>
> 代码演示：

```
# 我们自己可以使用raise语句自己触发异常
try:
  num = input("请输入一个数字:")
  # 判断用户输入的是否是数字:
  if num.isdigit():
      print(num)
  else:
      raise ValueError("num必须是数字")
except Exception as e:
  print('引发异常',repr(e))
```

4.assert断言

> 对某个问题做一个预测，如果预测成功，则获取结果；如果预测失败，则打印预测的信息
>
> 代码演示：

```
# 断言  assert
def test(a):
  assert a != 0  # 我断定a不等于0,当a = 0 时,导致的程序错误就叫做断言错误.
  print(12 / a)
test(0)
```