异常

常见异常举例：

```
"""
假设电脑是家，我们要去找碗
绝对路径：家：\厨房\餐具柜\第二层
假设我现在在客厅
相对路径：我在的客厅的旁边\厨房\餐具柜\第二层

一般在项目中，用相对路径比较多
"""
# 绝对路径：指明了文件在电脑的哪个磁盘哪个具体目录
# D:\pythonWork\pyVIP19\12、文件操作\月光--胡彦斌.mp3
# 相对路径：相对于当前代码文件的路径，比如在代码中写“月光--胡彦斌.mp3”
# python会从当前代码文件的附近（同级目录下）去找这首歌
# FileNotFoundError  文件找不到错误
with open("月光--胡彦斌.mp3",'rb')as f:
    f.read()

# NameError   名称错误
# Python只能认识特定的数据类型：整数、小数、复数、布尔、字符串、列表、元祖、集合、字典
print(a)    # python可能会认为这是一个变量

# SyntaxError 语法错误
if 1>0
     print("1>0")
"""
当我们通过代码编辑器写好了代码之后，我们的代码要通过Python解释器进行编译和执行。
编译 ：  将Python代码解析为电脑能理解的字节码
执行 ：  让电脑去执行他能够看懂的解析好的代码数据

轻度语法错误：代码能够正常的编译通过，然后执行，在执行过程中抛出语法错误

重度语法错误：代码直接在编译的时候就不通过，直接报错
"""
# TypeError 类型错误
print('a' / 2)
```

异常处理：

```
# 通用异常处理
try:
    # try里面的代码是可能会发生异常的代码
    print('a' / 2)   # TypeError 类型错误
except :       # !!!except后面不跟任何内容直接写：表示捕获所有的异常
    # except里面是如果异常发生了之后 处理这个异常的解决方案
    a = 10
    print(a / 2)

# 指定捕获某个异常情况
```

```
try:
    print('a' / 2)
except TypeError as t :       # 只捕获TypeError异常
    print("刚刚发生了一个错误，错误信息是：",t)
    a = 10
    print(a / 2)

# 指定捕获多个异常情况
try:
    print(b)    # 可能会发生异常的代码
except (TypeError,NameError) as t :       # 只捕获TypeError,NameError异常
    print("刚刚发生了一个错误，错误信息是：",t)
    print('b')

# 捕获所有继承 Exception 的普通异常
try:
    print(b)    # 可能会发生异常的代码
except Exception as t :       # 捕获所有继承 Exception 的普通异常
    print("刚刚发生了一个错误，错误信息是：",t)
    print('b')

# 异常捕获的使用场景：
# 1、当我们预测某段代码可能会出现错误的时候提前准备好解决方案并进行异常捕获操作。
# 2、有一个bug短时间内没法解决，但是需要保证程序能够正常秩序，因此用异常捕获跳过这段错误代码。

# 如果异常没有出现的处理
try:
    print("hello")   # 假设 代码没有异常
except Exception as e :
    print("出现异常：",e)
else:
    print("没有出现异常，这是没有异常的处理代码")

# 查看异常情况与非异常情况的案例
try:
    #这段代码可能会有问题
    x = input("异常情况请按1,正常情况请按0>>")
    if x == '1':
        print('a' / 2)   # 异常代码
    elif x == "0":
        print("今天天气好热哦！")      # 正常代码
except Exception as e:     # 这里是只要出现异常就会进入的代码
    print("报错信息：",e)
else:          # 如果没有异常会进入的代码
    print("----------------哈哈哈哈哈没有报错吧！")
finally:       # 不管有没有错误，都会去执行的代码（finally是在异常捕捉操作之后但是还没结束之前执行）
    print("hello大家好我是finally")
```

异常具有传递性：

```
def func01():
```

```
    print("这是func01开始")
    try :
        num = 1 / 0      # ZeroDivisionError 除数为0异常
    except Exception as e:
        print(e)
    # 零除：在数学中，将一个数除以零的操作，通常被认为是未定义的，因为它没有唯一的结果。
    print("这是func01结束")

def func02():
    print("这是func02开始")
    func01()    # 调用 会报错的函数
    # try:
    #     func01()
    # except Exception as e:
    #     print(e)
    print("这是func02结束")

def main():
    func02()  # 在主函数中调用func02()

main()
# try:
#     main()
# except Exception as e:
#     print(e)
```

自定义异常类+`raise`抛出异常

```
# 自定义异常类,!!!!一定要继承异常父类
class MyError(Exception):
    def __init__(self,msg):
        self.msg = msg
    def __str__(self):
        return self.msg

# raise 用于抛出异常的代码
# raise MyError("用户名为空异常！")      # 实例化一个异常，并抛出

# 登录功能 --自定义异常的用法举例
# 登录的时候必须用户名要填写
userName = input("请输入用户名：")
if userName=='' or userName==None:  # 如果用户名为空
    raise MyError("用户名为空！")
else:
    userPass = input("请输入密码：")
    if len(userPass)==8:
        print("登录成功！")
        return None
    else:
        raise MyError("登录密码不对！")
```

递归捕捉异常

```python
# 自定义异常类,!!!!一定要继承异常父类
class MyError(Exception):
    def __init__(self,msg):
        self.msg = msg
    def __str__(self):
        return self.msg
# 递归捕捉异常函数
def login():   # 函数自己调用自己
    try:
        userName = input("请输入用户名: ")
        if userName=='' or userName==None:  # 如果用户名为空
            raise MyError("用户名为空! ")
        else:
            userPass = input("请输入密码: ")
            if len(userPass)==8:
                print("登录成功! ")
                return None
            else:
                raise MyError("登录密码不对! ")
    except Exception as e:
        print(e)
        login()

login()
```