异常处理

1.异常简介

```
print('-----test--1---')
open('123.txt','r')
print('-----test--2---')
```

1.1、异常定义

异常即是一个事件，该事件会在程序执行过程中发生，影响了程序的正常执行。
一般情况下，在Python无法正常处理程序时就会发生一个异常。

异常是Python对象，表示一个错误。当Python脚本发生异常时我们需要捕获处理它，否则程序会终止执行。

1.2、异常案例

1）捕获异常 try...except...

```
try:
    #print(num)
    f2 = open('D:/5.txt', 'r')

except FileNotFoundError:
    print('异常发生了...%s'%FileNotFoundError)
```

2）捕获多个异常

```
try:
    a = input()
    print(a)
    f2 = open('D:/5.txt', 'r')
#使用元组 捕获多个异常
except(FileNotFoundError,NameError):
    print('异常发生了')
```

3)捕获所有异常，得到错误信息

```
try:
    a = input()
    print(int(a)/0)
    f2 = open('D:/5.txt', 'r')
#使捕获所有异常，得到错误信息
except Exception as result:
    print('捕获了异常:%s' % result)
```

1.3、else

如果没有捕获到异常，那么就执行 else中的事情 。

```
try:
    a = input()
    print(int(a)/6)
    #f2 = open('D:/5.txt', 'r')
#使捕获所有异常，得到错误信息
except Exception as result:
    print('捕获了异常:%s' % result)
else:
    print('没有捕获到')
```

1.4、try...finally...

在程序中，一段代码必须要执行，无论异常是否产生都要执行，此时就使用finally.比如文件 关闭，释放锁，

把数据库连接返还给连接池等。

```
try:
    f = open('D:/a.txt', 'r')
    while True:
        #读取一行代码
        content = f.readline()
        if len(content) == 0 :
            break
        #打印内容
        print(content)
except Exception as result:
    print('捕获了异常:%s' % result)
finally:
    f.close()
    print('关闭文件')
```

1.5、使用raise抛出异常

```
class Test(object):
    def start(self):
        try:
            content = input('请输入内容')
            if len(content) < 3 :
                #抛出python自带异常
                raise Exception('<3了')
        except Exception as info:
            print('异常产生了%s' % info)

t = Test()
t.start()
```

2.模块

2.1、定义自己的模块

在Python中，每个Python文件都可以作为一个模块，模块的名字就是文件的名字。比如有这样一个文件test.py，在test.py中定义了函数add。

便于团队分工协作，每人负责一个模块

大大提高了代码的可维护性，编写代码不必从0开始。平时我们在编写程序的时候，也经常引用其他模块，

包括python内置的模块和来自第三方的模块。

标准库	说明
builtins	内建函数默认加载
os	操作系统接口
sys	Python自身的运行环境
functools	常用的工具
json	编码和解码 JSON 对象
logging	记录日志，调试
multiprocessing	多进程
threading	多线程
copy	拷贝
time	时间
datetime	日期和时间
calendar	日历
hashlib	加密算法
random	生成随机数
re	字符串正则匹配
socket	标准的 BSD Sockets API
shutil	文件和目录管理
glob	基于文件通配符搜索

2.2、常用标准库

1）md5加密

Md5是让大容量信息在用数字签名软件签署私人密匙前被"压缩"成一种保密的格式（就是把一个任意长度的字节串变换成一定长的大整数）。不管是MD2、MD4还是MD5，它们都需要获得一个随机长度的信息并产生一个**128位**的信息摘要。

破解md5网站：http://www.cmd5.com/

```
import hashlib
m = hashlib.md5()
print(m)
str = '123'
m.update(str.encode()) #更新哈希对象以字符
print(m.hexdigest())  #返回16进制数字字符串，共计32位
```

2）常用扩展库

扩展库	说明
requests	使用的是 urllib3，继承了urllib2的所有特性
urllib	基于http的高层库
scrapy	爬虫
beautifulsoup4	HTML/XML的解析器
celery	分布式任务调度模块
redis	缓存
Pillow(PIL)	图像处理
xlsxwriter	仅写excle功能,支持xlsx
xlwt	仅写excle功能,支持xls ,2013或更早版office
xlrd	仅读excle功能
elasticsearch	全文搜索引擎
pymysql	数据库连接库
mongoengine/pymongo	Mongodb python接口
matplotlib	画图
numpy/scipy	科学计算
django/tornado/flask	web框架
xmltodict	xml 转 dict
SimpleHTTPServer	简单地HTTP Server,不使用Web框架
gevent	基于协程的Python网络库
fabric	系统管理
pandas	数据处理库
scikit-learn	机器学习库

3）第三方库

获取微信好友男女性别比例

```
import itchat
from echarts import Echart, Legend, Pie
# 先登录
itchat.login()
# 获取好友列表
friends = itchat.get_friends(update=True)[0:]
# 初始化计数器，有男有女，当然，有些人是不填的
male = female = other = 0
# 遍历这个列表，列表里第一位是自己，所以从"自己"之后开始计算
# 1表示男性，2女性
for i in friends[1:]:
    sex = i["Sex"]
    if sex == 1:
        male += 1
    elif sex == 2:
        female += 1
    else:
        other += 1

# 总数算上，好计算比例啊～
total = len(friends[1:])
chart = Echart(u'%s的微信好友性别比例' % (friends[0]['NickName']), 'from WeChat')
chart.use(Pie('WeChat',
              [{'value': male, 'name': u'男性 %.2f%%' % (float(male) / total *
100)},
               {'value': female, 'name': u'女性 %.2f%%' % (float(female) / total
* 100)},
               {'value': other, 'name': u'其他 %.2f%%' % (float(other) / total *
100)}],
              radius=["50%", "70%"]))
chart.use(Legend(["male", "female", "other"]))
del chart.json["xAxis"]
del chart.json["yAxis"]
chart.plot()
```

3.爬虫

3.1、爬虫定义

就是按照一定的规则，自动的从网络中抓取信息的程序或者脚本。万维网就像一个巨大的蜘蛛网，我们的爬虫就是上面的一个蜘蛛，不断的去抓取我们需要的信息。

爬虫3要素：**抓取、分析、存储**

3.2、基础的抓取操作

```
import urllib.request
response = urllib.request.urlopen('http://www.python.org')
#print(response.read().decode('utf-8'))
#利用type()方法输出Response的类型
print(type(response))
print(response.status)#得到返回结果的状态
print(response.getheaders())#获取headers中的值
#该类型主要包含的方法：read()、readinfo()
```

3.3、爬虫的基本原理

网页请求的过程分为两个环节：

1. Request （请求）：每一个展示在用户面前的网页都必须经过这一步，也就是向服务器发送访问请求。
2. Response（响应）：服务器在接收到用户的请求后，会验证请求的有效性，然后向用户（客户端）发送响应的内容，客户端接收服务器响应的内容，将内容展示出来，就是我们所熟悉的网页请求，如图 8 所示。

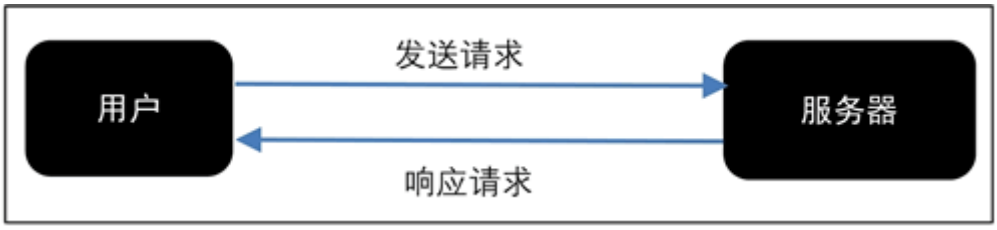

图 8 Response相应

网页请求的方式也分为两种： get请求 、post请求

3.4、使用Beautiful Soup 解析网页

通过 requests 库已经可以抓到网页源码，接下来要从源码中找到并提取数据。Beautiful Soup 是 python

的一个库，其最主要的功能是从网页中抓取数据。Beautiful Soup 目前已经被移植到 bs4 库中，也就是说在导 入 Beautiful Soup 时需要先安装 bs4 库。

```
import requests
import re
from bs4 import    BeautifulSoup
url='http://www.cntour.cn/'
strhtml=requests.get(url)
soup=BeautifulSoup(strhtml.text,'html.parser', from_encoding='utf-8')
data =
soup.select('#main>div>div.mtop.firstMod.clearfix>div.centerBox>ul.newsList>li>a
')
#将数据
print(data)
for item in data:
    #明确提取的数据为标题和链接
    result={
        #提取标签正文
        'title': item.get_text(),
        #提取标签中的href属性
        'link': item.get('href'),
        #提取链接中的id
        'id':re.findall('\d+', item.get('href'))
    }
print(result)
```