> 同济大学数据科学导论实验报告03

《万维网运行原理分析》

> - 学院：电子信息工程学院

> - 专业：数据科学与大数据专业
> - 学号：1851804
> - 姓名：苗成林
> - 指导教师：郭玉臣
> - 时间：2021.10.21

实验报告正文

实验目的及要求

1. 掌握单页面数据采集方法
2. 掌握 requests 库用法
3. 掌握正则表达式用法

requests介绍

requests是一个很实用的Python HTTP客户端库，编写爬虫和测试服务器响应数据时经常会用到，Requests是Python语言的第三方的库，专门用于发送HTTP请求。

Requests响应

r.status_code 响应状态码
r.heards 响应头
r.cookies 响应cookies
r.text 响应文本

r. encoding 当前编码
r. content 以字节形式（二进制）返回

正则表达式

概念介绍

正则表达式（Regular Expression）是用于描述一组字符串特征的模式，用来匹配特定的字符串。通过特殊字符+普通字符来进行模式描述，从而达到文本匹配目的工具。

正则表达式目前被集成到了各种文本编辑器/文本处理工具当中

应用场景

（1）验证：表单提交时，进行用户名密码的验证。

（2）查找：从大量信息中快速提取指定内容，在一批url中，查找指定url。

（3）替换：将指定格式的文本进行正则匹配查找，找到之后进行特定替换。

实验内容

通过rquests库请求得到豆瓣top250网页源代码，并通过正则表达式匹配得到对应信息-书名，作者信息，评分以及简介。网站的URL为’ https://book.douban.com/top250?start=0’ ,但我们拉到底部发现250本读书的信息被分成了10页，这就需要我们首先对URL的规律进行分析得到所有页面的URL信息传递给get（）方法中请求源代码。点击到第2页发现URL为’ https://book.douban.com/top250?start=25’ ，第三页为’ https://book.douban.com/top250?start=50’ ，最后一页为’ https://book.douban.com/top250?start=225’ ，我们发现URL为每增加一页将最后的数字一次增加25，所以可以用一个方法进行URL的生成，代码如下：

```
def get_allpage():
    urllist=[]#用于存储生成的所有URL
    baseurl='https://book.douban.com/top250?start='
    for i in range(10):
        allurl=baseurl+str(i)
        urllist.append(allurl)
    return urllist
```

这样我们就获取全部10页的URL信息。接下来利用requests获取网页源代码，代码如下：

```
import requests
def get_page(url):
    try:
        headers={'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36(KHTML, like Gecko) Chrome/64.0.3282.140 Safari/537.36
Edge/17.17134'}
        res=requests.get(url,headers=headers)
        if res.status_code==200:
            return res.text
        else:
            return None
    except RequestException:
        print("请求错误")
```

因为豆瓣网没有发现什么反爬虫措施，所以我们只需要给get（）传递headers以及URL参数即可。下面就是利用正则表达式匹配我们所需要的信息，这里还是先啰嗦几句关于正则表达式的使用吧。正则表达式式用来处理字符串的强大工具，可以利用它实现字符串的检索替换以及匹配验证。但初看正则表达式只会感觉它是一串乱七八糟的字符，当然这是一开始不了解它的语法结构的原因，比如’\d’是匹配数字，‘a-z’是匹配小写字母，下面是一些通用的匹配规则：

```
\w-匹配数字字母和下划线
\W-与\w相反
\s-匹配任意空白字符
\S-匹配非空字符
\d-匹配数字
^-匹配一行字符串的开头
$-匹配一行字符串的结尾
.-匹配任意非换行符外的字符
*和±匹配0个或多个表达式
？-匹配0个或多个前面的正则表达式定义片段，非贪婪模式
（）-匹配括号内的表达式，也表示一个组
```

python的re库提供了整个正则表达式的实现，下面的代码具体介绍一些实例：

match()方法

这是re库中最基础的一个匹配方法，向它传入正则表达式以及原字符串就可以从头检测正则表达式是否匹配字符串：

```
#匹配QQ邮箱地址
strr="this is my email address 1400000076@qq.com"
restr="^this\s\w{2}\s\w{2}\w{5}\s\w{6}\s(\d{10})@\w{2}.\w{3}"
resu=re.match(restr,strr)
if resu:
    print(len(strr))
    print(resu.group())#匹配内容
    print(resu.group(1))#匹配结果
    print(resu.span())#匹配范围
```

这里实现匹配字符串中的数字内容，即QQ号，正则表达式开头的^是字符串开头这里是this，接下来\s匹配空格，\w{2}匹配2个字母，依次匹配字母及空格，匹配的数字使用（\d{10}）表示匹配连续出现的10个数字，接下来匹配.以及com$结尾，match（）返回bool值，利用group(1)函数即可输出想要的带的数字结果。

search()方法

如果不想从头开始搜索，可以使用search()方法，正则表达式就可以不用以this开头，只需在匹配的数字之前就可以。通用匹配：上面我们使用了特定的匹配字母，空格以及数字的正则表达式去匹配，这样不免有些繁琐，其实可以使用通用的匹配表达式来进行匹配，分析上面列出的匹配规则可以发现，’ . ‘可以匹配出换行符外的任意字符，而’ ‘表示匹配前面的字符无限次，也就是说’ .'可以匹配任意字符任意次，所以上面的表达式其实可以修改为：

```
'^this.*(\d+).*com$
```

这样使用匹配后，我们其实会发现一个奇怪的问题，匹配的结果只得到了’ 6 ‘这一个数字，这是怎么回事？因为我们要获取的是中间的数字，所以两侧的均使用通用匹配符来来实现，但数字之前这样使用会出现一个问题，在这种情况下，之前的通用匹配符会尽可能多的匹配，而只需要给匹配内容留下一个数字就满足要求。这里涉及的是正则语法中的贪婪与非贪婪模式，因为上面我们匹配数字时没有指定数字数量，所以贪婪模式下通

用匹配符就尽可能多的匹配字符，解决的办法也很简单，只需要在通用匹配符后加上？即可实现非贪婪模式。
转义匹配：前一段说到’.'可以匹配任意字符，那么如果目标字符串就包含’．‘怎么处理呢？这里只需要使用转义匹配方法，及在要匹配的’.'字符前加上\即可实现转义，实例如下：

```
#转义匹配
conn="(百度)www.baidu.com"
restr3="\(百度\)www\.baidu\.com"
resu3=re.match(restr3,conn)
print(resu3)
print(resu3.group(1))
```

答应匹配结果，发现成功匹配了原字符串。

findall()方法

前面的match(),search()方法当匹配到第一个内容后就返回，但是如果想要匹配整个字符串内容，就需要使用findall方法了，该方法会搜索这个字符串，然后返回所有目标字符串。
接下来我们抓取一段网站源代码试着去写出匹配我们我要得到信息的正则字符串。
打开’ https://book.douban.com/top250?start=0’ ,打开检查元素，选择对应位置选取html代码，

我们先选区网页排名第一的图示，点击追风筝的人书名，找到对应的html代码
分析发现，这个信息是在标签div，pl2下面，以结尾，匹配内容为title内的内容“追风筝的人”，利用之前的只是，很很轻松就可以写出对应的正则表达式为：href. ?title="(.?)". ?,这样就匹配到了书名信息，再接再励，分析其他信息的html，可以写出整个正则表达式为：'href.?title="(. ?)".?. ?pl.?>(. ?)
.?rating_nums. ?>(.?). ?inq.?>(.*?)’ ，由此，我们就可以解析前面得到的整个网页的html，得到书名，作者等信息，代码如下：

```
import re
def jx_html(text):
    restr=re.compile('href.*?title="(.*?)".*?</a>.*?pl.*?>(.*?)</p>.*?rating_nums.*?>(.*?)</span>.*?inq.*?>(.*?)</span>',re.S)#compile方法用于将正则表达式编译成正则对象以复用，re.S用于匹配换行字符串
    result=re.findall(restr,text)
    for ree in result:
        yield{'bookname':ree[0],'author':ree[1],'score':ree[2],'expl':ree[3]}#利用迭代器返回获取信息的字典对象
```

最后我们需要将得到的解析信息保存为.txt文件,代码如下：

```
def save_fil(content):
    with open ('book250.txt','a',encoding='utf-8')as f:
        f.write(json.dumps(content,ensure_ascii=False)+'\n')#利用json库的dumps（）方法实现字典的序列化
```