第一次实验报告

学号：518030910308
姓名：刘文轩

一、实验准备

1、实验环境介绍

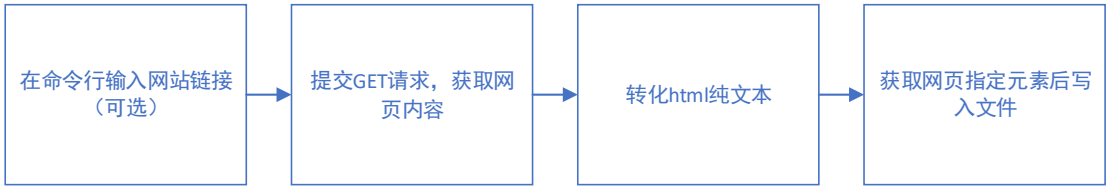
操作系统：Windows 10
语言：Python 3
IDE：Pycharm 2018.2.4

2、实验目的

2.1 学会在给定任意网页内容的前提下，返回指定元素的内容（如超链接的 URL，图片的地址、相应文本描述等）
2.2 学会将 PARSER 获得的内容存入文件
2.3 学会将相对地址改写为绝对地址
2.4 学会使用命令行参数输入需要爬取的网页链接以及相关的参数

3、实验思路

3.1 使用 requests 库，并使用 requests.get() 方法向网站提交 GET 请求，获得网站内容
3.2 使用 bs4 库，通过里面的 Beautifulsoup 把 html 纯文本转化为便于程序访问的数据结构
3.3 使用 Beautifulsoup 里面的 findAll()和 get() 来找到我们想要获得的网页元素
3.4 使用 sys.argv 在命令行输入程序参数



二、实验过程

1、练习一

1.1 网站链接与内容获取

我们使用 requests 库，通过 requests.get()构造一个向服务器请求资源的 Request 对象，并将返回的对象赋给 r，这是一个包含服务器资源的 Response 对象。

但是网络连接有风险，异常处理很重要。

在此使用 r.raise_for_status()，它在方法内部判断 r.status_code 是否等于 200，如果不是 200，就会产生异常 requests.HTTPError，有了它，不需要增加额外的 if 语句，该语句便于利用 try-except 进行异常处理。

为了使内容编码方式正确，我们使用语句 r.encoding = r.apparent_encoding，其中 r.encoding 是从 HTTP header 中猜测的响应内容编码方式，r.apparent_encoding 是从内容中分析出的响应内容编码方式，我们可以将其理解为一种备选编码方式。

然后使用 content = r.content 就可以获得网站的内容，便于我们进行下一步的处理。

这一段的代码如下：（完整的含 try-except 的代码见 1.4）

```
r = requests.get(url,timeout=30)
r.raise_for_status()
r.encoding = r.apparent_encoding
content = r.content
```

1.2 parseURL 函数设计

我们使用 from bs4 import BeautifulSoup 来完成对于 Beautiful Soup 库的引用。

在调用 BeautifulSoup 来进行对于网站内容的解析时，可以在参数里加入 features="html.parser"来使用 bs4 的 HTML 解析器。将解析后的对象赋给 soup。

我们要返回网页中所有超链接的 URL（不包括图片地址），同时题目要求链接地址只需要考虑形如<a href="..">这样的形式，我们在此使用 soup.findAll()方法，指定 findAll()的 name 参数等于'a'来返回一个列表类型，储存所有查找到的超链接的结果。

对于返回的列表中的每一个元素，我们通过.get('href','')来获得每一个超链接的 URL，同时在没有超链接的情况下返回一个空字符串避免引起程序异常，将获得的结果添加到 urlset 中，最后返回这个集合。

同时爬到的链接需要过滤，如果只是一个如同"/"之类的字符，又或者是例如"javascript:addBookmark2()"的部分，都需要过滤掉。

接下来，我们将所有 http 或者是 https 开头的绝对路径的 url 直接保存。但是对于相对路径，我们先通过 urllib.parse.urljoin()将其改写为绝对路径，再存入集合。还有一类网站，直接以//www.开头，我们将其补充完整，在它的前面补充上 http 后存入集合。

代码如下：

```
def parseURL(content,url):
    urlset = set()
    soup = BeautifulSoup(content,features="html.parser")
    for i in soup.findAll('a'):
        link = i.get('href','')
        if len(link) <= 1:
            continue
        match = re.match(r'^javascript.*',link)
        if match:
            continue
        match2 = re.match(r'^https?.*',link)
        if match2:
            urlset.add(match2.group(0))
            continue
        match3 = re.match(r'^/?\w.*',link)
        if match3:
            link = urllib.parse.urljoin(url, match3.group(0))
        match4 = re.match(r'^//w{3}.*',link)
        if match4:
            link = urllib.parse.urljoin('http:', match4.group(0))
        urlset.add(link)
    return urlset
```

1.3 write_outputs 函数设计

文件的写入与保存已经在之前的程序中有所涉及，我们只需要使用 with open(filename, 'w') as f:，然后再使用 f.write()就可以写入我们需要的内容了，具体代码如下：

```python
def write_outputs(urls, filename):
    with open(filename, 'w') as f:
        for url in urls:
            f.write(url)
            f.write('\n')
```

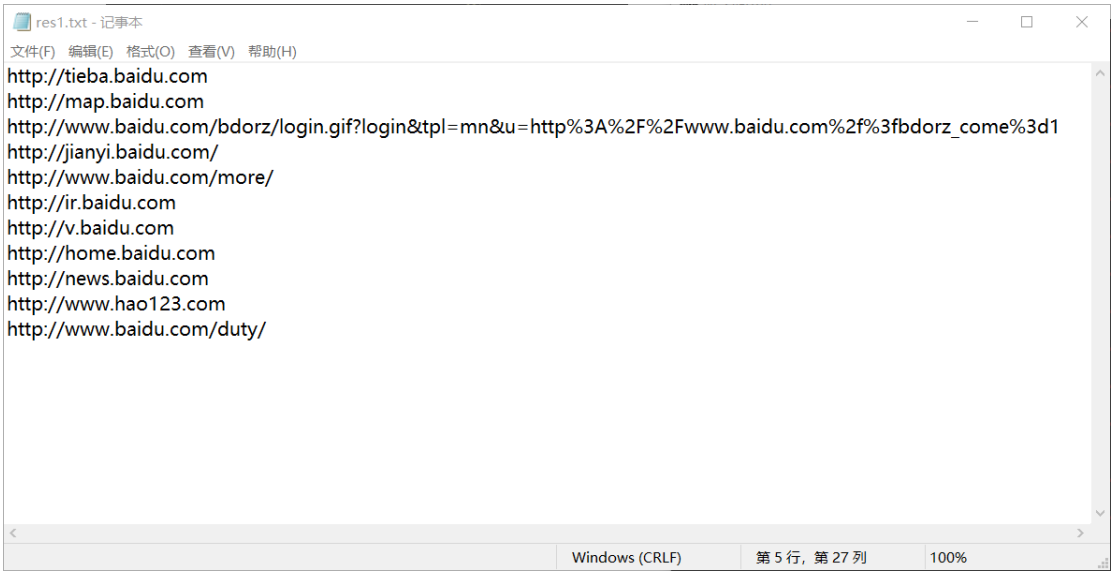
1.4 命令行界面操控

为了便于在命令行界面中操控，我们使用 sys 库，sys.argv 获取运行此 python 文件时的命令行参数，且以 list 形式存储参数，只需要将 main 函数代码修改如下即可：

```python
def main():
    url = 'http://www.baidu.com'
    # url = 'http://www.sjtu.edu.cn'
    if len(sys.argv) > 1:
        url = sys.argv[1]
    try:
        r = requests.get(url,timeout=30)
        r.raise_for_status()
        r.encoding = r.apparent_encoding
        content = r.content
        urls = parseURL(content,url)
        write_outputs(urls, 'res1.txt')
    except:
        print("产生异常")
```

1.5 代码运行结果

在命令行中运行这个 py 文件，并且输入网站为 www.baidu.com，爬取的链接结果如图所示。



2、练习二

2.1 代码改动

对于练习二，相对于练习一而言，我们只是将获取网站中的网页链接变成了获取网页中所有的图片地址，整体而言改动不大，我们将需要修改的部分代码作一说明。

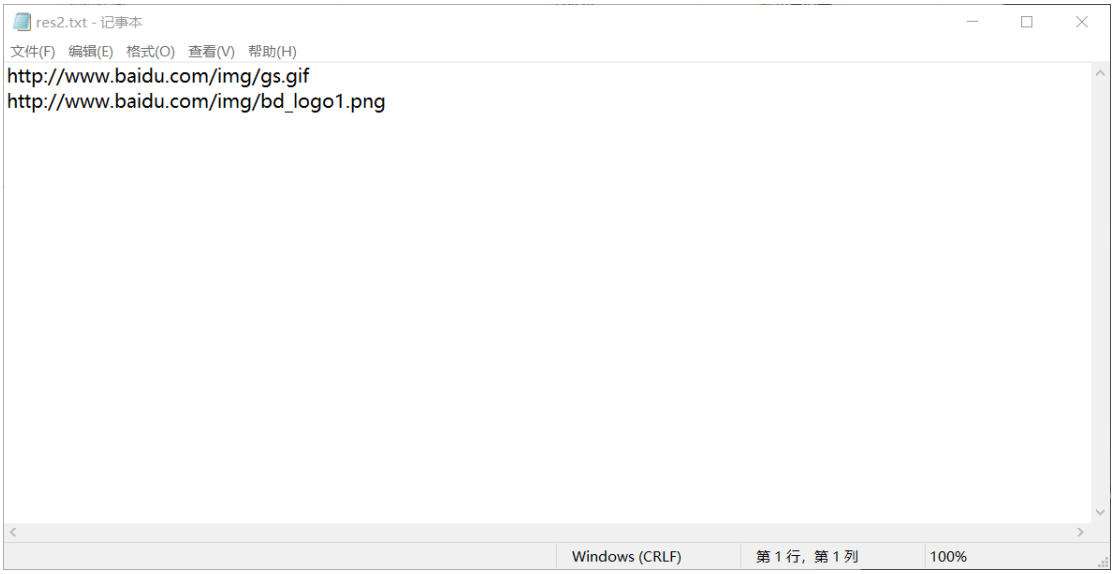
在我们使用了 soup = BeautifulSoup(content,features="html.parser")获得了 soup 后，接下来由于只需要获取图片链接，我们将内容获取开始部分的代码修改为：

```
for i in soup.findAll('img'):
    link = i.get('src','')
```

再将保存文件的文件名修改为 res2.txt 即可。

2.2 代码运行结果

在命令行中运行这个 py 文件，并且输入网站为 www.baidu.com，爬取的图片地址如图所示。



3、练习三

3.1 网站链接与内容获取

这一块的内容与练习一二中的部分没有很大的区别，需要注意的是，有一些网站更愿意让用户通过浏览器正常的访问，对于例如 pycharm 之类的 IDE，它返回的结果会和正常的浏览器浏览到的结果不相同，甚至有的网站只允许浏览器来访问，所以我们在这里要把自己“伪装”成一个用浏览器浏览的用户，我们可以在使用 requests.get()时传入参数 headers，令其等于键值对：{'user-agent': 'Mozilla/5.0'}，来模拟用户通过浏览器浏览网站时的样子。

我们将 main 函数的代码修改如下：

```
def main():
    url = 'http://daily.zhihu.com'
    if len(sys.argv) > 1:
        url = sys.argv[1]
    try:
        kv = {'user-agent': 'Mozilla/5.0'}
        r = requests.get(url, timeout=30, headers=kv)
        r.raise_for_status()
        r.encoding = r.apparent_encoding
        content = r.content
        urls = parseIMG(content)
        write_outputs(urls, 'res3.txt')
    except:
        print("产生异常")
```

3.2 parseIMG 函数设计

因为在本题中，对于每一个日报中的看点，我们要分别保存其的图片地址、标题内容、网站地址，这很适合通过字典类型来完成它们一一对应的关系，因此我们最后将所有字典存入一个 list 中，并将其作为函数返回值。

同时观察网页的代码可以知道，这个网站的看点链接保存方式都很统一，都是以/story 的相对路径开头，因此我们可以先通过正则表达式找到所有以/story 开头的链接，找到这内部包含的图片地址，标题内容，再通过 urllib.parse.urljoin()将相对路径改写为绝对路径。

代码如下：

```
def parseIMG(content):
    urlset = []
    soup = BeautifulSoup(content,features="html.parser")
    for i in soup.findAll('a',{'href':re.compile('^/story')}):
        dic = dict()
        pic = i.contents[0]
        picurl = pic.get('src', '')
        dic['picurl'] = picurl
        dic['content'] = i.contents[1].string
        link = i.get('href','')
        url = urllib.parse.urljoin('http://daily.zhihu.com/', link)
        dic['url'] = url
        urlset.append(dic)
    return urlset
```

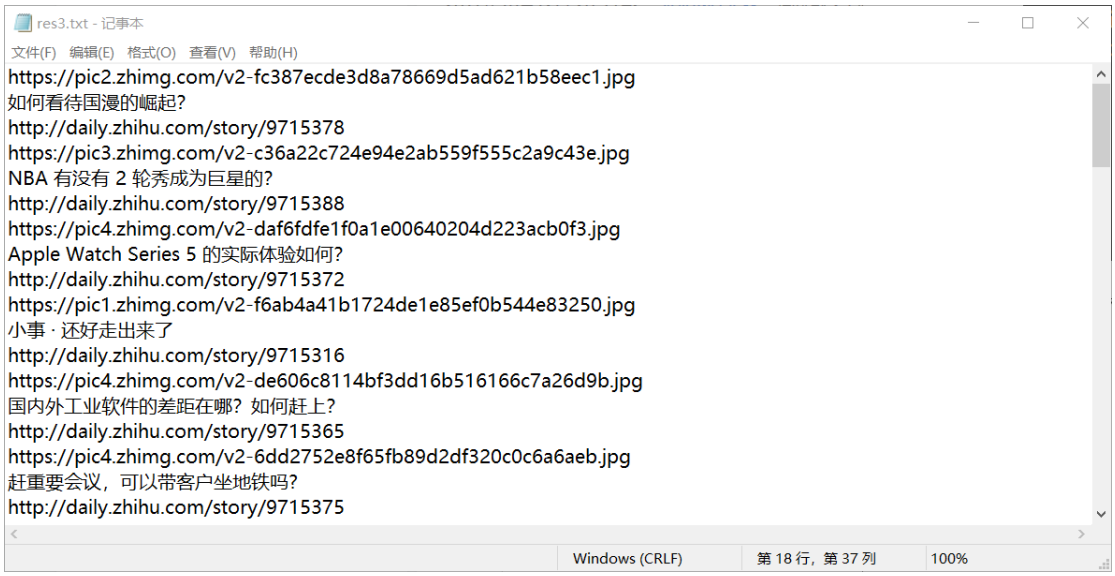
3.3 write_outputs 函数设计

由于传入的 urls 是 list 类型，里面存储的是一个个字典类型，我们将代码稍稍修改：

```
def write_outputs(urls, filename):
    with open(filename, 'w') as f:
        for url in urls:
            f.write(url['picurl'])
            f.write('\n')
            f.write(url['content'])
            f.write('\n')
            f.write(url['url'])
            f.write('\n')
```

3.4 代码运行结果

由于最后获得的内容相当长，因此我们仅仅在此展示我们结果的一部分。



三、实验总结

1、实验概述

本次实验的主要任务，可以总结为在给定网页链接的基础下，获得网页中所有的指定元素，如网站链接、图片地址、文章标题等，三道练习分别从不同元素的角度，让我们熟悉了相关的代码设计。

2、感想总结

在这次的实验中，学会了的东西有很多，其中最重要的就是提高了自己独自处理问题的能力，这在我们将来的学习和工作生活中都是很重要的。而具体细化开来，在本次实验中：

2.1 学会了使用 requests 库连接网站并且获得网站内容

2.2 学会了使用 beautifulsoup 库将网站内容解析为便于程序设计的形式

2.3 学会了正则表达式的使用方法

2.4 学会了在命令行中运行 python 程序，并且输入特定的参数

2.5 学会了通过 urllib.parse 将相对路径改写为绝对路径

3、创新点

首先是对于异常处理，我们不希望当程序出现错误时在用户界面显示一大堆报错相关信息，因此我在程序中使用了 try-except 以及 r.raise_for_status()来完成对程序异常的处理，同

时通过修改 headers 来模拟用户通过浏览器正常浏览网站的情形，还加入了 r.encoding = r.apparent_encoding 来使得页面的解码更加正确。

同时在获取链接时，考虑到了很多不合规的情形，并且通过正则表达式，re.match()这个方法将其一一过滤或者修改，具体的情形已经在上面进行了详细的分析。

4、遇到的问题

其实上面的很多创新点的提出，都是基于我在代码运行中遇到的问题而设计的，如一开始获得的链接中有很多相对路径，又或者是奇怪的单字符，我们可以通过正则表达式将其滤出并加以处理。

又比如在构建 BeautifulSoup 库的对象时，如果不使用参数 features="html.parser"，就会产生 warning，我们只需要在此加入此属性，指定使用 html 解析器即可。