实 验 报 告

实验 1：HTML 与 Beautiful Soup

517030910374

郭嘉宋

1. 实验准备

(1) 实验环境

本实验采用 Ubuntu 系统下的 python2.7 进行，使用 miniconda 环境，Ubuntu 系统安装在 VMware Workstation 提供的虚拟机中，同时使用 Beautiful Soup 对网页的 html 文档进行解析。

(2) 实验目的

了解 html 的工作原理，学习爬虫的使用方法，编写程序在网页上寻找需要的信息并将其存于本地的文档中方便后续使用。

(3) 实验原理

浏览网页的原理是当在浏览器中输入网址后，浏览器向网站发送 http 请求，网站处理请求后发给浏览器对应的 html 信息，浏览器解析得到。因此可以利用程序得到对应的 html 信息，即爬虫爬下了需要的内容，本实验采用 python 语言编程，利用 Beautiful Soup 讲 html 的纯文本转化为便于程序访问的数据结构，更易于爬取相应内容。

2. 实验过程

(1) 练习 1

1.1 实验步骤

使用 Beautiful soup 中的 findAll 函数找到 tag 为 a 的链接即可，链接名为“ href=… ”的形式，只需要得到 href 值即可，将所有值加入 urlset，最终使用 outputs 函数写为 txt 即可。

```python
import sys
import urllib2
from bs4 import BeautifulSoup

def parseURL(content):
    urlset = set()
    soup = BeautifulSoup(content)
    for url in soup.findAll('a'):
        x = url.get('href', '')
        urlset.add(x)
    return urlset

def write_outputs(urls, filename):
    with open(filename, 'w') as f:
        for url in urls:
            #x=url.get('href','')
            f.write(url)
            f.write('\n')
```

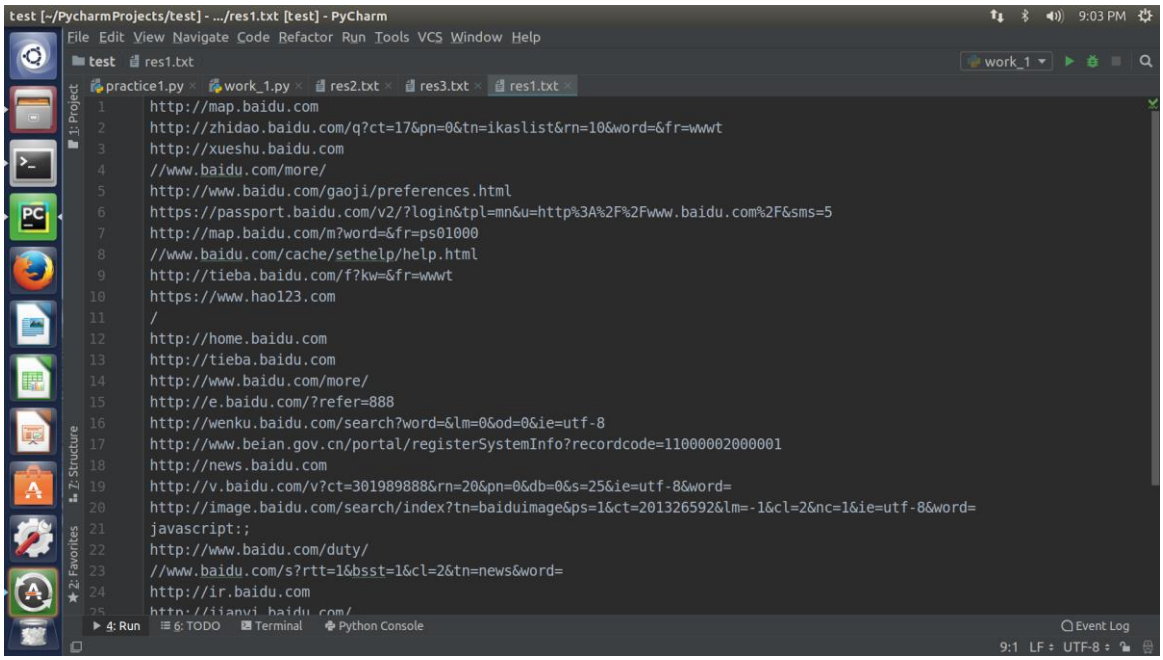
```python
def main():
    url = 'http://www.baidu.com'
    #url = 'http://www.sjtu.edu.cn'
    if len(sys.argv) > 1:
        url = sys.argv[1]
    content = urllib2.urlopen(url).read()
    urls = parseURL(content)
    write_outputs(urls, 'res1.txt')


if __name__ == '__main__':
    main()
```

NOTE：刚开始实验时 url.get('href','') 写在了注释的位置，会出现乱码，原因发现 findAll 函数运行后得到的类似一个 list，将 list 加入 urlset 后再查找会出现错误，因此更改到了如图正确的位置。

1.2 实验结果

运行了如图所示程序后，会得到 res1.txt 文档，文档内容即为所有链接。



NOTE：结果中会有'/'这个链接，经过查找资料，发现'/'是相对路径的根目录表示方法。还有'javascript:;'这个地址，经过查找资料，发现它其实是执行一段空白 JAVASCRIPT 语句，是为了防止链接跳转。

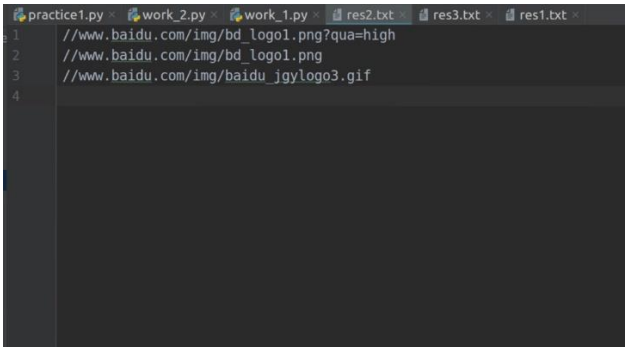
(2) 练习 2

2.1 实验步骤

练习 2 与练习 1 思路相同，只需要将查找内容从链接换为图片即可，较容易。

```
def parselMG(content):
    imgset=set()
    soup = BeautifulSoup(content)
    for url in soup.findAll('img'):
        x = url.get('src','')
        imgset.add(x)
    return imgset
```

2.2 实验结果



res.txt 中的文档内容即为图片的地址。

(3) 练习 3

3.1 实验步骤

首先我先用 chrome 浏览器对网页进行了分析，理清网页的结构，发现在 body->div#content->div#content-block clearfix->div#content-left 中有每一张图的 id content 和地址内容，因此定向爬虫即可。

Id 获取直接搜索以“qiushi_tag”开头的 tag 即可：

```
list_id=[]
for url_1 in soup.findAll('div',{'id':re.compile('^qiushi_tag')}):

    t=url_1.get('id','')
    tmp=t.split('_')[-1]
    list_id.append(tmp)
```

```
list_content=[]
for url_2 in soup.findAll('div',{'class','content'}):
    for url_2_1 in url_2.findAll('span'):
#findAll can deal with the element( or the definite html string, not a list)
        tmp2=url_2_1.get_text()
        list_content.append(tmp2)
```

NOTE：在寻找 content 时，内容位于两个 span tag 之间，查找资料后运用 get_text() 语法解决，同时搜寻 span tag 时，在 class#content 下寻找，在运用 findAll 函数时只可对外层 url_2 中的元素进行，不可对 url_2 直接 findAll，前期一直遇到类型 bug，url_2 类似 list，只有元素可以 findAll.

```
url_original = 'http://www.qiushibaike.com/pic'
list_url=[]
for url_3 in soup.findAll('div',{'class','thumb'}):
    for url_3_1 in url_3.findAll('img'):
        tmp3=url_3_1.get('src','')
        tmp3 = urlparse.urljoin(url_original, tmp3)
        list_url.append(tmp3)
```

寻找图片链接时将相对地址用 urljoin 改为绝对地址即可。

在寻找下一页链接时，经过网页源码比对，发现在 class#pagination 中的 li tag 下含有第 1,2,3 页的地址，因此第 2 个 li tag 即为我们要找的下一页链接， 运用循环控制即可找到。最后将所有元素按照规定格式插入字典即可，最后写入 res3.txt。

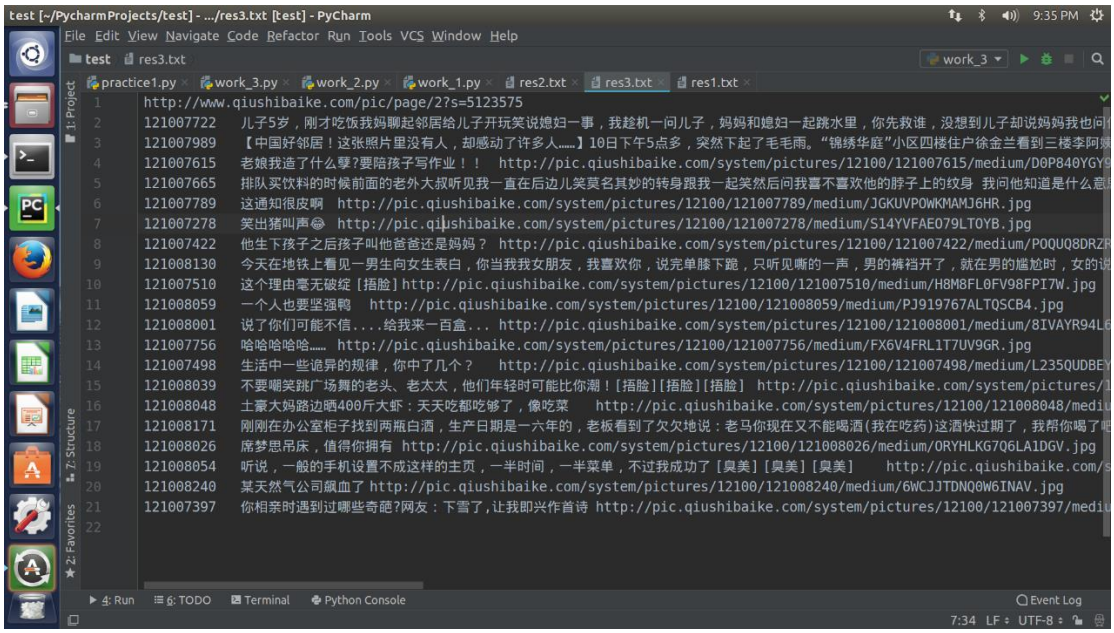
```
    url_next=soup.findAll('ul',{'class','pagination'})
    # the last "pagination class" contains the next page
    flag=1
    # url_next is a list, you must take its elements to use findAll
    for i in url_next:
        for use in i.findAll('li'):
            if flag==2:
                key=use
                break
            flag += 1
    for tmp in key.findAll('a'):
        key_url=tmp.get('href','')

    url_original = 'http://www.qiushibaike.com/pic'
    nextPage=urlparse.urljoin(url_original,key_url)
    nextpage=str(nextPage)
```

NOTE：在写入文字时中文会面临无法写入的情况，因此需要在开头载入宏包的时候用 UTF8 重新加载，如图：

```
import sys
reload(sys)
sys.setdefaultencoding('UTF-8')
import re
import urlparse
import urllib2
from bs4 import BeautifulSoup
```

3.2 实验结果



最上一行为下一页地址，之后每行依次是图片 id 文字内容和图片地址。3 个 txt 文档建议使用 notepad++打开，用记事本打开格式不会对齐，不美观。

3. 实验总结

此次试验是我第一次接触爬虫，对爬虫有了初步的了解，在编程过程中也遇到了很多困难和 bug，在上述“2.实验步骤”中，都用加粗的 **NOTE** 加以了说明。此次实验让我认识了 beautiful soup 这个极为实用的 python 工具，同时给我的最大印象就是，在爬虫之前一定要搞清楚网页的信息结构，定向地寻找需要的信息，而且网页的各个 tag 往往都有规律，可以通过类比的方式寻找不同类型的信息，父子节点搞清楚搜索起来就变得简单了。最后希望在之后的课程学习中对爬虫有着更深的了解，继续努力

o(*≧▽≦)ツ！

517030910374
郭嘉宋