Experiment 1

517030910356 Zhihui Xie

September 18, 2018

Contents

1 Experiment Preparation

1.1 Environment

The experiment is based on Python 2.7 with Linux (running on Ubuntu with the help of VMware workstation).

1.2 Purpose

Given that this experiment are designed for further study of Web Crawler, we are supposed to master the basic structure of HTML, and get started with HTML parser.

From a higher perspective, the experiment aims to push us to try new things, and strengthens the ability of self-study.

1.3 Principles

1.3.1 What happens after we input the website?

As we know, the entire Intenet is organized with tons of small nets, and based on some internet protocols, we can share massive information on the Internet.

When we click the "Access" button, our browser will send the HTML request to the website, which includes the information of Cookie and User-agent. And then the website will respond, returning the HTML text. Finally, the browser will render the page.

1.3.2 The basic structure of HTML

As the name suggests, HTML(HyperText Markup Language) uses different markup tags to organize the page. These tags come in pairs and play different roles. And the content between a pair is called an HTML element, which is usually attached with some attributions.

1.3.3 BeautifulSoup

To deconstruct the HTML page, we use the module BeautifulSoup as the resolver.

The module transforms a complex HTML document into a complex tree of Python objects. There are four kinds of objects we are going to deal with: Tag, NavigableString, BeautifulSoup, and Commment.

2 Experiment Process

2.1 Exercise 1

2.1.1 Main procedure

In this task, I only need to complete the function 'parseURL' and the main, which accepts an URL, and returns a set of all the URLs of hyperlinks.

parseURL I first define the set 'urlset', and convert the URL into an BeautifulSoup object.

```
urlset = set()
soup = BeautifulSoup(content, features='html.parser')
```

And then, to filter out some noise, such as 'javascript:;', I use the function 'compile' in the module 're' to define a pattern, which represents the URLs like '//www.baidu.com/duty/' or 'https://www.hao123.com'.

```
pattern = re.compile(('^\w*:*//'))
```

Finally, I search all the tags 'a' in the 'soup' using the function 'findAll', and extract the attribution 'href'.

```
for link in soup.findAll('a', {'href': pattern}):
    url = link.get('href', '')
    url = urljoin(sys.argv[1], url)
    urlset.add(url)
```

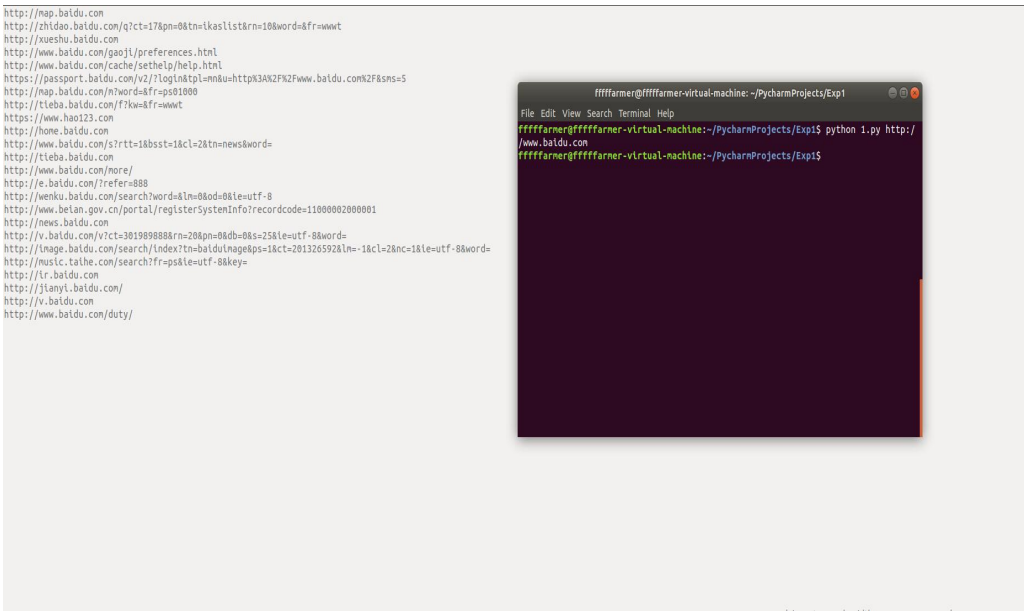
It's worth mentioning that, since the Transfer Protocol is missing, I use the function 'urljoin' to complete it. The parameter is the input URL.

main Here I also use the module 'sys' to get the input URL. And 'urllib' helps me read the URL directly and transforms it into a string.

```
def main():
    url = sys.argv[1]
    content = urllib2.urlopen(url).read()
    print type(content)
    urls = parseURL(content)
    write_outputs(urls, 'res1.txt')
```

2.1.2 Result

We can now directly input the URL in the console and then the result URL will be stored in res1.txt.



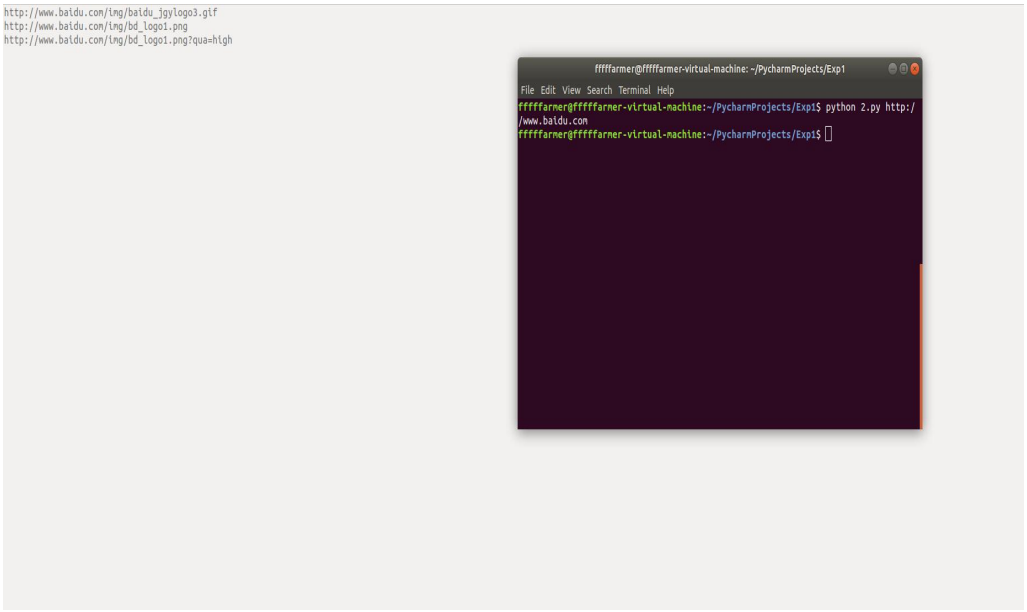
2.2 Exercise 2

2.2.1 Main procedure

Since the procedure in this task is very similar to the last one, I'm not going to discuss it in detail.

2.2.2 Result

We can now directly input the URL in the console and then the all the images will be stored in res2.txt.



2.3 Exercise 3

2.3.1 Main procedure

This time, the task is more tricky. We need to go deeper into the HTML page.

parseQiushibaikePic As usual, I first do some initialization.

```
        docs = {}
        nextPage = ''
        soup = BeautifulSoup(content, features='html.parser')
```

Then, I search for tags whose attribution 'id' begins with 'qiushi_tag'. And I extract the id, as well as the content and the url by using BeautifulSoup method. A BeautifulSoup tag can have different attributions, and we only need to operate just like dealing with a dictionary.

```
        for i in soup.findAll('div', {'id': re.compile('^qiushi_tag')}):
            id = i['id'].split('_')[-1]
            id = id.encode('utf-8')

            imgurl = urljoin(sys.argv[1], i.img['src'])

            content = i.find('div', {'class': 'content'}).span.string
```

```
            content = unicode(content)
            content = content.encode('utf-8')

            tag = {}
            tag['content'] = content
            tag['imgurl'] = imgurl

            docs[id] = tag
```

The search for the nextPage url is similar. To find the url, I first search for the hyperlink tag 'a' whose href starts with '/pic/page/'. And then, I take the url whose class is 'next'.

```
        for i in soup.findAll('a', {'href': re.compile('^/pic/page/')}):
            spanClass = i.span['class'][0].encode('utf-8')
            if spanClass == 'next':
                nextPage = urljoin(sys.argv[1], i['href'])
```

main To get access to the website 'Qiushibaike', I set up the 'User-Agent' with the function in the module 'urllib'.

```
        def main():
            req = urllib2.Request(sys.argv[1], None, {'User-agent': 'Custom User Agent'}
                                                                    )
            content = urllib2.urlopen((req)).read()
            content = content.replace('<br/>', '')
            info = parseQiushibaikePic(content)

            write_outputs(info, 'res3.txt')
```

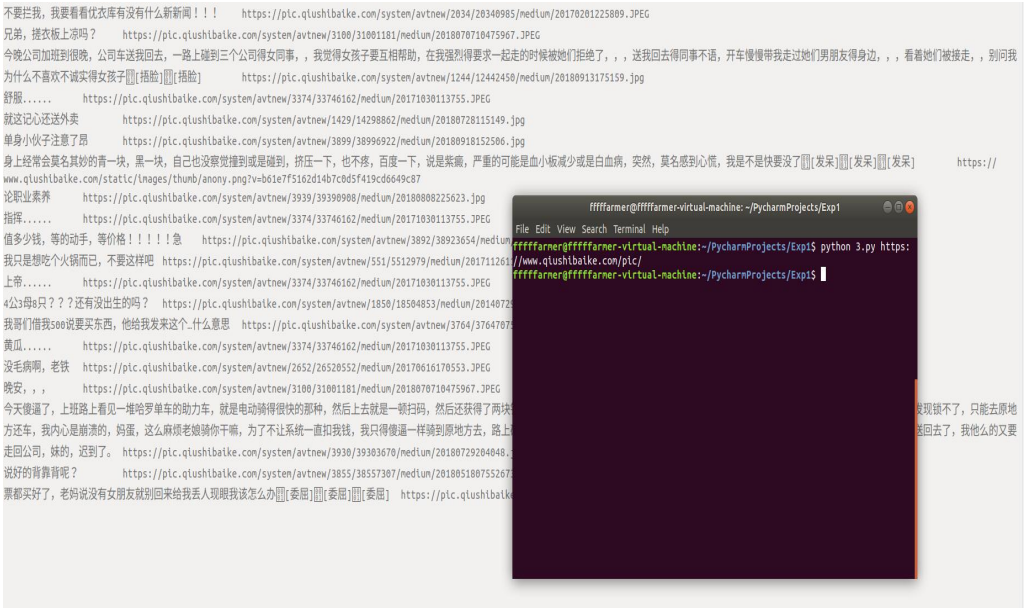
write_outputs I also rewrite the function 'write_outputs' to meet the need of writing out.

```
        def write_outputs(info, filename):
            with open(filename, 'w') as f:

                for i in info[0].values():
                    f.write(i['content'])
                    f.write('\t')
                    f.write(i['imgurl'])
                    f.write('\n')
```

2.3.2 Result

We can now directly input the URL in the console and then the all the images and the corresponding contents will be stored in res3.txt.



3 Experiment Thoughts

3.1 The importance of data processing

It is far from completing the task when we finish crawling. There are massive data waiting for processing.

I find it kind of tricky espercially when I deal with the encoding 'utf-8'. I need to do some transformation so that the content can be stored in a right form.

So we really need to pay attention to those raw data.

3.2 information Retrieval

I also find that when we crawl some websites, we will use different methods to search for the information we want. And it's really important to improve our arithmetic espercially when the scale is large.