# lab2实验报告—Crawler

黄霄童 520030910229

## 练习1

### 问题

注册药智网账号（https://www.yaozh.com/)并填写自己的个人信息，接着仿照第9页PPT的做法，利用cookie登陆自己的个人主页（https://www.yaozh.com/member/basicinfo/），并定位打印出自己个人主页中的真实姓名，用户名，性别，出生年月和简介。

### 实现细节介绍

practice1.py

- 登录药智网

  在药智网上注册成功后，将用户名和密码放入data中，将data编码格式更改为"UTF-8"格式，然后采取如下方式登录：

```
request = urllib.request.Request(sign_in_url, data=postdata,headers =
dict(Referer = sign_in_url))
```

- 解析网页

  获取登录成功的cookie值之后，根据cookie值访问https://www.yaozh.com/member/basicinfo。 解析该网站发现，需要爬取的信息都存储在

```
<div class="U_myinfo clearfix">
```

  将class名为"U_myinfo clearfix"的__子节点__存储在my_info变量中

```
my_info=soup.find_all(attrs={'class': 'U_myinfo clearfix'})[0].contents
```

  真实姓名，用户名，性别（男生1女孩2），出生年月和简介分别在该标签的contents[3], contents[5], contents[7], contents[9], contents[11]中"value"下， 并且相应的标签名存在

  中

```
for i in range(3,10,2):
      # 访问属性value
      print('{0}
```

```
{1}'.format(my_info[i].dt.string,my_info[i].input.attrs['value']))
print('简介: %s'%my_info[11].textarea.text)
```

- 输出结果

```
真实姓名:  黄霄童
用户名:  startosphere
性别:  1
出生年月:  2002-05-12
简介：今天打工不狠，明天地位不稳。
```

# 练习2

## 问题

修改crawler_sample.py中的union_bfs函数，完成BFS搜索

## 实现细节介绍

> practice2&3.py

为去重使用了set()函数，然后采取切片插入的方式将b插在a前

```
def union_bfs(a, b):
    a[0:0]=list(set(b))
```

# 练习3

## 问题

graph结构与crawler_sample.py中g的结构相同。 完成后运行graph, crawled = crawl('A', 'bfs') 查看graph 中的图结构，以及crawled中的爬取结果顺序。

## 实现细节介绍

> practice2&3.py

在`crawler()函数`，在每一次循环中加入如下代码

```
if outlinks!=[]:graph[page]=outlinks
```

若网页中包含的超链接为空则不需要记录入图中，否则就在图中存储下对应page所爬取到的链接。

为方便对照结果，在最后返回的时，采用了如下代码：

```
    return dict(sorted(graph.items(),key=lambda k:k[0])), crawled
```

将graph按照键值排序，以对照dfs bfs的结果是否相同。

输出结果

```
graph_dfs: {'A': ['B', 'C', 'D'], 'B': ['E', 'F'], 'D': ['G', 'H'], 'E': ['I',
'J'], 'G': ['K', 'L']}
crawled_dfs: ['A', 'D', 'H', 'G', 'L', 'K', 'C', 'B', 'F', 'E', 'J', 'I']
graph_bfs: {'A': ['B', 'C', 'D'], 'B': ['E', 'F'], 'D': ['G', 'H'], 'E': ['I',
'J'], 'G': ['K', 'L']}
crawled_bfs: ['A', 'D', 'C', 'B', 'G', 'H', 'F', 'E', 'L', 'K', 'J', 'I']
```

# 练习4

## 问题

将练习23中修改的部分加入crawler.py。

## 实现细节介绍

> practice3.py

- get_page()函数
    1. 功能：对于传递的网址page返回爬取的内容，若访问失败则返回空列表
    2. 实现：采取`urllib.request.urlopen()`函数登录网页，并设置最大登录时间timeout=3

    ```
    response=urllib.request.urlopen(req,timeout=3)
    content=response.read()
    ```

    3. 鲁棒性: 采用try...catch机制防止登录发生错误，判断登录状态码是否为200, 若为200才返回爬取到的内容

    ```
    try:
      ....
      if response.getcode() == 200:
          return content
      else:
          return None
    except:
      return None
    ```

- get_all_links()函数
    1. 功能：对于给予的网页内容返回里面的所有网址的绝对地址

2. 实现： 采用BeautifulSoup中的findALL爬取超链接

```
soup=BeautifulSoup(content,features='html.parser')
a_labels=soup.findAll('a',{'href' : re.compile('^http|^/')})
# 正则表达式筛选href中的url而非JS代码或者#锚点操作
```

对于每一个链接如果是相对地址，采用如下方法转换成绝对地址：

```
if link[0:4]!='http':link=urllib.parse.urljoin(page,link)
```

最后返回一个列表，其中每个元素为一个字符串，表示爬取到的网址。

- crawl()函数 除了作业中自带的代码，在dfs or bfs 时，添加了对于网址中不含超链接的情况进行特判；并在每次爬取深度+1时，max_page-=1,若max_page=0则停止爬取。

```
max_page-=1
if max_page==0:break # 达到最大深度
```

其他函数同练习1、2、3。