Python爬虫常用库

一、课前准备

1. 根据课程大纲复习Python基本原理；
2. 根据课程大纲提前预习Python爬虫常用库；

二、课堂主题

本小节主要学习Python爬虫常用库的操作。

三、课堂目标

1. 掌握BeautifulSoup爬虫库的使用。
2. 掌握文件的读取操作。
3. 掌握JSON常用模块。
4. 掌握动态数据的爬虫。

四、知识要点

1、 BeautifulSoup库的基本使用

1.1 BeautifulSoup简介

BeautifulSoup库一个灵活又方便的网页解析库，处理高效，支持多种解析器。

```
pip install bs4
```

中文参考文档

1.2 基本使用

```
from bs4 import BeautifulSoup
html = '''
    <body>
    <header id="header">
    <h3 id="name">小强也可爱</h3>
    <title>标题</title>
      <div class="sns">
        <a href="http://www.kaikeba.com/feed/" target="_blank" rel="nofollow" title="RSS"><i class="fa fa-rss" aria-hidden="true"></i></a>
        <a href="http://kaikeba.com/kaikeba" target="_blank" rel="nofollow" title="Weibo"><i class="fa fa-weibo" aria-hidden="true"></i></a>
                <a href="https://www.kaikeba.com/in/kaikeba" target="_blank" rel="nofollow" title="Linkedin"><i class="fa fa-linkedin" aria-hidden="true"></i></a>
                <a href="mailto:kaikeba@gmail.com" target="_blank" rel="nofollow" title="envelope"><i class="fa fa-envelope" aria-hidden="true"></i></i></a>
          </div>
      <div class="nav">
           <ul><li class="current-menu-item"><a href="http://www.kaikeba.com/">hello</a></li>
                <li><a href="http://www.kaikeba.com/about-me/">word</a></li>
                <li><a href="http://www.kaikeba.com/post-search/">nihao</a></li>
                <li><a href="http://www.kaikeba.com/wp-login.php">kkb</a></li>
            </ul>
      </div>
    </header>
    </body>
'''
soup = BeautifulSoup(html,'lxml')
# 格式化输出 soup 对象的内容
print(soup.prettify())

# 根据标签名获取整个标签(但是拿出的是第一个)
print(soup.li)

# 获取标签的名字
print(soup.title.name)

# 获取标签中的文本
print(soup.title.string)

# 获取标签title的父标标签
```

```python
print(soup.title.parent.name)

# 获取li标签的子标签
print(soup.li.contents)

# 获取便签的属性值的两种方式
print(soup.li["class"])
print(soup.li.attrs['class'])

# 使用select，css选择器
print(soup.select('li'))
# 类名前加.，id名前加#
print(soup.select('.current-menu-item'))

# 获取内容
print(soup.select('.current-menu-item')[0].get_text())
# 获取属性值
print(soup.select('.current-menu-item')[0].attrs['class'])

# 获取li标签下面的子标签
print(soup.select('li > a')[1].get_text())

# 使用find和findall进行查找
print(soup.find('li',attrs={'class':'current-menu-item'}))
print(soup.find_all('li',attrs={"class":"current-menu-item"})[0])
```

解析器	使用方法	优势	劣势
Python标准库	`BeautifulSoup(markup, "html.parser")`	• Python的内置标准库 • 执行速度适中 • 文档容错能力强	• Python 2.7.3 or 3.2.2)前的版本中文档容错能力差
lxml HTML 解析器	`BeautifulSoup(markup, "lxml")`	• 速度快 • 文档容错能力强	• 需要安装C语言库
lxml XML 解析器	`BeautifulSoup(markup, ["lxml", "xml"])` `BeautifulSoup(markup, "xml")`	• 速度快 • 唯一支持XML的解析器	• 需要安装C语言库
html5lib	`BeautifulSoup(markup, "html5lib")`	• 最好的容错性 • 以浏览器的方式解析文档 • 生成HTML5格式的文档	• 速度慢 • 不依赖外部扩展

1.3 BeautifulSoup练习

```python
soup = BeautifulSoup(result_str,'lxml')
# 获取文本
texts = [i.get_text() for i in soup.find_all('a',attrs = {'class':"j_th_tit"})]
# 获取连接
lins = ['https://tieba.baidu.com{}'.format(i.attrs['href']) for i in
soup.find_all('a',attrs = {'class':"j_th_tit"})]
```

2、文件基础操作

文件包括 文本文件和二进制文件（声音，图像，视频）。 从存储方式来说，文件在磁盘上的存储方式都是二进制形式，所以，文本文件其实也应该算二进制文件。先从他们的区别来说，虽然都是二进制文件，但是二进制代表的意思不一样。打个比方，一个人，我们可以叫他的大名，以叫他的小名，但其实都是代表这个人。二进制读写是将内存里面的数据直接读写入文本中，而文本呢，则是将数据先转换成了字符串，再写入到文本中。

2.1、读取文件

要以**读取文件**的模式打开一个文件对象，使用Python内置的`open()`函数，传入文件名和标示符：

```
>>> f = open('/Users/michael/test.txt', 'r')
```

标示符'r'表示读，这样，我们就成功地打开了一个文件。

如果文件**不存在**，`open()`函数就会抛出一个`IOError`的错误，并且给出错误码和详细的信息告诉你文件不存在：

```
>>> f=open('/Users/michael/notfound.txt', 'r')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
FileNotFoundError: [Errno 2] No such file or directory: '/Users/michael/notfound.txt'
```

如果文件打开**成功**，接下来，调用`read()`方法可以一次读取文件的全部内容，Python把内容读到内存，用一个`str`对象表示：

```
>>> f.read()
'Hello, world!'
```

最后一步是调用`close()`方法关闭文件。文件使用完毕后必须关闭，因为文件对象会占用操作系统的资源，并且操作系统同一时间能打开的文件数量也是有限的：

```
>>> f.close()
```

由于文件读写时都有可能产生`IOError`，一旦出错，后面的`f.close()`就不会调用。所以，为了保证无论是否出错都能正确地关闭文件，我们可以使用`try ... finally`来实现：

```
try:
    f = open('/path/to/file', 'r')
    print(f.read())
finally:
    if f:
        f.close()
```

但是每次都这么写实在太繁琐，所以，Python引入了`with`语句来自动帮我们调用`close()`方法：

```
with open('/path/to/file', 'r') as f:
    print(f.read())
```

这和前面的`try ... finally`是一样的，但是代码更佳简洁，并且不必调用`f.close()`方法。

调用`read()`会一次性读取文件的**全部**内容，如果文件有10G，内存就爆了，所以，要保险起见，可以反复调用`read(size)`方法，每次最多读取**size**个字节的内容。另外，调用`readline()`可以每次读**取一行**内容，调用`readlines()`一次读取所有内容并按行返回`list`。因此，要根据需要决定怎么调用。

如果**文件很小**，`read()`一次性读取最方便；如果不能确定文件大小，反复调用`read(size)`比较保险；如果是配置文件，调用`readlines()`**最方便**：

```
for line in f.readlines():
    print(line.strip()) # 把末尾的'\n'删掉
```

2.2、文件的打开方式

模式	描述
r	以只读方式打开文件。文件的指针将会放在文件的开头。这是默认模式。
rb	以二进制格式打开一个文件用于只读。文件指针将会放在文件的开头。
r+	打开一个文件用于读写。文件指针将会放在文件的开头。
rb+	以二进制格式打开一个文件用于读写。文件指针将会放在文件的开头。
w	打开一个文件只用于写入。如果该文件已存在则打开文件，并从开头开始编辑，即原有内容会被删除。如果该文件不存在，创建新文件。
wb	以二进制格式打开一个文件只用于写入。如果该文件已存在则打开文件，并从开头开始编辑，即原有内容会被删除。如果该文件不存在，创建新文件。
w+	打开一个文件用于读写。如果该文件已存在则打开文件，并从开头开始编辑，即原有内容会被删除。如果该文件不存在，创建新文件。
wb+	以二进制格式打开一个文件用于读写。如果该文件已存在则打开文件，并从开头开始编辑，即原有内容会被删除。如果该文件不存在，创建新文件。
a	打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾。也就是说，新的内容将会被写入到已有内容之后。如果该文件不存在，创建新文件进行写入。
ab	以二进制格式打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾。也就是说，新的内容将会被写入到已有内容之后。如果该文件不存在，创建新文件进行写入。
a+	打开一个文件用于读写。如果该文件已存在，文件指针将会放在文件的结尾。文件打开时会是追加模式。如果该文件不存在，创建新文件用于读写。
ab+	以二进制格式打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾。如果该文件不存在，创建新文件用于读写。

2.3、字符编码

要读取非UTF-8编码的文本文件，需要给`open()`函数传入`encoding`参数，例如，读取GBK编码的文件：

```
>>> f = open('/Users/michael/gbk.txt', 'r', encoding='gbk')
>>> f.read()
'测试'
```

遇到有些编码不规范的文件，你可能会遇到`UnicodeDecodeError`，因为在文本文件中可能夹杂了一些非法编码的字符。遇到这种情况，`open()`函数还接收一个`errors`参数，表示如果遇到编码错误后如何处理。最简单的方式是直接忽略：

```
>>> f = open('/Users/michael/gbk.txt', 'r', encoding='gbk', errors='ignore')
```

2.4、写文件

写文件和**读文件**是一样的，唯一区别是调用`open()`函数时，传入**标识符**`'w'`或者`'wb'`表示写文本文件或写二进制文件：

```
>>> f = open('/Users/michael/test.txt', 'w')
>>> f.write('Hello, world!')
>>> f.close()
```

你可以反复调用`write()`来写入文件，但是**务必**要调用`f.close()`来**关闭**文件。当我们写文件时，操作系统往往不会立刻把数据写入磁盘，而是放到内存缓存起来，空闲的时候再慢慢写入。只有调用`close()`方法时，操作系统才保证把没有写入的数据全部**写入磁盘**。忘记调用`close()`的**后果**是数据可能只写了一部分到磁盘，剩下的丢失了。所以，还是用`with`语句来得保险：

```
with open('/Users/michael/test.txt', 'w') as f:
    f.write('Hello, world!')
```

要写入特定编码的文本文件，请给`open()`函数传入`encoding`参数，将字符串自动转换成**指定编码**。

注意：以`'w'`模式写入文件时，如果文件已存在，会直接**覆盖**（相当于删掉后新写入一个文件）。如果我们希望追加到文件末尾怎么办？可以传入`'a'`以**追加**（append）模式写入。

3、 JSON

JSON是一种轻量级的数据交换格式，它使得人们很容易进行阅读和编写。同时也方便了机器进行解析和生成，适用于进行数据交互的场景，比如网站前台与后台之间的数据交互。

```
▼{message: "success", data: [{title: "献血小哥走后捐献器官挽救三人生命；留下妻子孩儿家徒四壁...", gallary_flag: 2,…},…]}
  ▼data: [{title: "献血小哥走后捐献器官挽救三人生命；留下妻子孩儿家徒四壁...", gallary_flag: 2,…},…]
    ▼0: {title: "献血小哥走后捐献器官挽救三人生命；留下妻子孩儿家徒四壁...", gallary_flag: 2,…}
       article_url: "/group/6695997778155274756/"
       cover_image_url: "//p1.pstatp.com/list/300x170/pgc-image/3879b72efcb74400affa7ca89a0338db"
       gallary_flag: 2
       gallery_image_count: 17
      ▶image_list: [{url: "//p1.pstatp.com/list/364x360/pgc-image/3879b72efcb74400affa7ca89a0338db"},…]
       title: "献血小哥走后捐献器官挽救三人生命；留下妻子孩儿家徒四壁..."
    ▼1: {title: "广州白天鹅宾馆：中国内地首家五星级酒店", gallary_flag: 1,…}
       article_url: "/group/6674456951658643980/"
       cover_image_url: "//p1.pstatp.com/list/300x170/pgc-image/6b05e44d8ea1474e9843b9848f4f1a83"
       gallary_flag: 1
       gallery_image_count: 4
      ▶image_list: [{url: "//p1.pstatp.com/list/640x360/pgc-image/6b05e44d8ea1474e9843b9848f4f1a83"},…]
       title: "广州白天鹅宾馆：中国内地首家五星级酒店"
    ▶2: {title: "演员陈欣予，新版《倚天屠龙记》张无忌妈妈殷素素，颜值在线！", gallary_flag: 2,…}
    ▶3: {title: "国产运-20“鲲鹏”大型运输机高清图赏", gallary_flag: 1,…}
    ▶4: {title: "俄罗斯女孩在中国留学三年，称这五点让自己印象最深刻", gallary_flag: 1,…}
    ▶5: {title: "德国隐世湖心小岛，岛上有棵569岁的玫瑰树，茜茜公主常在此玩耍", gallary_flag: 1,…}
    ▶6: {title: "你知道我国各朝的武士铠甲都是什么样的吗？现在带大家了解一下", gallary_flag: 3,…}
    ▶7: {title: "吴哥窟：恢宏精美的庙宇历尽风霜，仍令后世游人惊为神迹", gallary_flag: 1,…}
    message: "success"
```


3.1. JSON语法规则

- 数据在名称/值对中
- 数据由逗号分隔
- 花括号保存字典对象
- 方括号保存数组

3.2 JSON 值

JSON 值可以是:

- 数字（整数或浮点数）
- 字符串（在双引号中）
- 逻辑值（true 或 false）
- 数组（在方括号中）
- 对象（在花括号中）
- null

3.3 JSON模块

通过Python的json模块，可以将字符串形式的json数据转化为字典，也可以将Python中的字典数据转化为字符串形式的json数据。

3.3.1 dumps

dumps只完成了序列化为str。

```
# dumps 将“obj” 数据类型 转换为 JSON格式的字符串
dict1 = {
    'Code': 200,
    'Count': 657,
    'Posts': [
                {
                    'Id': 0,
                    'PostId': "1123178321664806912",
                    'RecruitPostId': 49691
```

```
                },
                {
                    'Id': 0,
                    'PostId': "1123178321664806912",
                    'RecruitPostId': 49691
                }
            ]
        }
json_dict = json.dumps(dict1)
print(json_dict)
print(type(json_dict))
```

3.3.2 loads

loads 将包含str类型的JSON文档反序列化为一个python对象

```
dic  = json.loads('{"name":"Tom", "age":23}')
print(dic)
print(type(dic))
```

4、动态数据获取

```
import requests
import json
import time
now_time = int(time.time())
jiang_13 = now_time*1000
base_url = 'https://careers.tencent.com/tencentcareer/api/post/Query?timestamp={}&keyword=Python&pageIndex=1&pageSize=10&language=zh-cn&area=cn'
headers = {
        "User-Agent":"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.121 Safari/537.36"
    }
response = requests.get(url=base_url.format(jiang_13),headers=headers)
# print(response.content.decode('utf-8'))

# 将数据转化成Python对象
content_dict = json.loads(response.content.decode('utf-8'))

posts_list = content_dict['Data']['Posts']
# print(posts_list)
for value_dict in posts_list:

    # 招聘的海报的名字
    RecruitPostName = value_dict['RecruitPostName']
    # 职责
    Responsibility = value_dict['Responsibility']
    # 最后更新时间
    LastUpdateTime = value_dict['LastUpdateTime']
    print(RecruitPostName,Responsibility,LastUpdateTime)
```

五、总结

1. 掌握网页的基本构成及解析方式。
2. 掌握使用XPath爬取网页数据。