爬虫基础

Urllib版本

https://www.cnblogs.com/zhaof/p/6910871.html

1登录网页

1. **from** urllib.request **import** urlopen
3. # if has Chinese, apply decode()
4. html = urlopen(
5. "https://morvanzhou.github.io/static/scraping/basic-structure.html"
6. ).read().decode('utf-8')
7. **print**(html)

request:

https://blog.csdn.net/yaoyefengchen/article/details/79167421

2匹配信息re,bs4

Re

案例一：找title

1. **import** re
2. res = re.findall(r"<title>(.+?)</title>", html)
3. **print**("\nPage title is: ", res[0])

案例二：找所有的链接

1. res = re.findall(r'href="(.\*?)"', html)
2. **print**("\nAll links: ", res)
3. # All links:
4. ['https://morvanzhou.github.io/static/img/description/tab\_icon.png',
5. 'https://morvanzhou.github.io/',
6. 'https://morvanzhou.github.io/tutorials/scraping']

Bs4:

爬网页的流程：

1. 选着要爬的网址 (url)
2. 使用 python 登录上这个网址 (urlopen等)
3. 读取网页信息 (read() 出来)
4. **将读取的信息放入 BeautifulSoup**
5. **使用 BeautifulSoup 选取 tag 信息等 (代替正则表达式)**

常规读取网页：

1. **from** bs4 **import** BeautifulSoup
2. **from** urllib.request **import** urlopen
4. # if has Chinese, apply decode()
5. html = urlopen("https://morvanzhou.github.io/static/scraping/basic-structure.html").read().decode('utf-8')
6. **print**(html)

读取这个网页信息, 我们将要加载进 BeautifulSoup, 以 lxml 的这种形式加载，然后 soup 里面就有着这个 HTML 的所有信息. 如果你要输出 <h1> 标题, 可以就直接 soup.h1.

1. soup = BeautifulSoup(html, features='lxml')
2. **print**(soup.h1)
4. """
5. <h1>爬虫测试1</h1>
6. """
8. **print**('\n', soup.p)
10. """
11. <p>
12. 这是一个在 <a href="https://morvanzhou.github.io/">莫烦Python</a>
13. <a href="https://morvanzhou.github.io/tutorials/scraping">爬虫教程</a> 中的简单测试.
14. </p>

如果网页中有过个同样的 tag, 比如链接 <a>, 我们可以使用 find\_all() 来找到所有的选项. 因为我们真正的 link 不是在 <a> 中间 </a>, 而是在 <a href="link"> 里面, 也可以看做是 <a> 的一个属性. 我们能用像 Python 字典的形式, 用 key 来读取 l["href"]

1. """
2. <a href="https://morvanzhou.github.io/tutorials/scraping">爬虫教程</a>
3. """
5. all\_href = soup.find\_all('a')
6. all\_href = [l['href'] **for** l **in** all\_href]
7. **print**('\n', all\_href)
9. # ['https://morvanzhou.github.io/', 'https://morvanzhou.github.io/tutorials/scraping']

按css的class匹配

我要找所有 class=month 的信息. 并打印出它们的 tag 内文字

1. soup = BeautifulSoup(html, features='lxml')
3. # use class to narrow search
4. month = soup.find\_all('li', {"class": "month"})
5. **for** m **in** month:
6. **print**(m.get\_text())
8. """
9. 一月
10. 二月
11. 三月
12. 四月
13. 五月
14. """

找到 class=jan 的信息. 然后在 <ul> 下面继续找 <ul> 内部的 <li> 信息. 这样一层层嵌套的信息, 非常容易找到.

1. jan = soup.find('ul', {"class": 'jan'})
2. d\_jan = jan.find\_all('li')              # use jan as a parent
3. **for** d **in** d\_jan:
4. **print**(d.get\_text())
6. """
7. 一月一号
8. 一月二号
9. 一月三号
10. """

Bs4结合正则表达式

我们发现, 如果是图片, 它们都藏在这样一个 tag 中:

1. <td>
2. <img src="https://morvanzhou.github.io/static/img/course\_cover/tf.jpg">
3. </td>

所以, 我们可以用 soup 将这些 <img> tag 全部找出来, 但是每一个 img 的链接(src)都可能不同. 或者每一个图片有的可能是 jpg 有的是 png, 如果我们只想挑选 jpg 形式的图片, 我们就可以用这样一个正则 r'.\*?\.jpg' 来选取. 把正则的 compile 形式放到 BeautifulSoup 的功能中, 就能选到符合要求的图片链接了.

1. soup = BeautifulSoup(html, features='lxml')
3. img\_links = soup.find\_all("img", {"src": re.compile('.\*?\.jpg')})
4. **for** link **in** img\_links:
5. **print**(link['src'])
7. """
8. https://morvanzhou.github.io/static/img/course\_cover/tf.jpg
9. https://morvanzhou.github.io/static/img/course\_cover/rl.jpg
10. https://morvanzhou.github.io/static/img/course\_cover/scraping.jpg

又或者我们发现, 我想选一些课程的链接, 而这些链接都有统一的形式, 就是开头都会有 https://morvan., 那我就将这个定为一个正则的规则, 让 BeautifulSoup 帮我找到符合这个规则的链接.

1. course\_links = soup.find\_all('a', {'href': re.compile('https://morvan.\*')})
2. **for** link **in** course\_links:
3. **print**(link['href'])
5. """
6. https://morvanzhou.github.io/
7. https://morvanzhou.github.io/tutorials/scraping
8. https://morvanzhou.github.io/tutorials/machine-learning/tensorflow/
9. https://morvanzhou.github.io/tutorials/machine-learning/reinforcement-learning/
10. https://morvanzhou.github.io/tutorials/data-manipulation/scraping/
11. """

XPATH：　XPath 是一门在 XML 文档中查找信息的语言。XPath 可用来在 XML 文档中对元素和属性进行遍历。XPath 是 W3C XSLT 标准的主要元素，并且 XQuery 和 XPointer 都构建于 XPath 表达之上

<https://www.cnblogs.com/lei0213/p/7506130.html>

Url.parse:用于解析url中的参数  对url按照一定格式进行拆分或拼接

获取url参数：将url分为6个部分，返回一个包含6个字符串项目的元组：协议、位置、路径、参数、查询、片段。

1. >>> **from** urllib **import** parse
2. >>> url = r'https://docs.python.org/3.5/search.html?q=parse&check\_keywords=yes&area=default'
3. >>> parseResult = parse.urlparse(url)
4. >>> parseResult
5. ParseResult(scheme='https', netloc='docs.python.org', path='/3.5/search.html', params='', query='q=parse&check\_keywords=yes&area=default', fragment='')
6. >>> param\_dict = parse.parse\_qs(parseResult.query)
7. >>> param\_dict
8. {'q': ['parse'], 'check\_keywords': ['yes'], 'area': ['default']}
9. >>> q = param\_dict['q'][0]
10. >>> q
11. 'parse'
12. **from** urllib **import** parse
14. url = "https://docs.python.org/3.5/library/urllib.parse.html?highlight=parse#module-urllib.parse"
15. result = parse.urlparse(url)
16. **print**(result.query)  # 获取返回结果参数内容
17. **print**(parse.parse\_qs(result.query))  # 结果转换成字典
18. **print**(parse.parse\_qsl(result.query))  # 结果转换成列表

JSON：用于字符串和python数据类型间进行转换

json dumps把数据类型转换成字符串 dump把数据类型转换成字符串并存储在文件中  loads把字符串转换成数据类型  load把文件打开从字符串转换成数据类型

1. **import** json
3. test\_dict = {'bigberg': [7600, {1: [['iPhone', 6300], ['Bike', 800], ['shirt', 300]]}]}
4. **print**(test\_dict)
5. **print**(type(test\_dict))
6. #dumps 将数据转换成字符串
7. json\_str = json.dumps(test\_dict)
8. **print**(json\_str)
9. **print**(type(json\_str))

如何下载图片然后保存输出到电脑呢？

1. new\_img\_url = '%s/%02d.%s' % (img\_path, i, img\_format)
2. save\_img\_path = '%s/%02d.%s' % (save\_path, i, img\_format)
3. # 下载图片
4. image = requests.get(new\_img\_url)
5. # 命名并保存图片
6. with open(save\_img\_path, 'wb') as f:
7. f.write(image.content)

Python中os的常用方法：<https://www.cnblogs.com/mufenglin/p/7676160.html>

os.getcwd()函数得到当前工作目录，即当前Python脚本工作的目录路径

os.path.exists()函数用来检验给出的路径是否真地存在

os.path.split(path) ：将path分割成目录和文件名二元组返回。

os.path.join(path,name):连接目录与文件名或目录;使用“\”连接  
os.path.basename(path):返回文件名  
os.path.dirname(path):返回文件路径

爬虫步骤：1网络请求urllib/requests

2解析html（xpath(lxml)/bwautifulsoup4）

3数据存储

入门实践1：美图网站

<https://www.jianshu.com/p/e77245177e34>