从零开始学 Python 网络爬虫

Huan

2019 年 5 月 23 日

0.1 从零开始学 Python 网络爬虫

```
In [1]: ## format (), 格式化函数
        # 在 str.format() 调用时使用关键字参数, 可以通过参数名来引用值
        print('This {food} is {adjective}.'.format(food='spam', adjective='absolutely horrible'))

        print("{:.2f}".format(3.1415926))

        "{} {}".format("hello", "world")    # 不设置指定位置, 按默认顺序

        path = 'https://{}/{}'.format("bd", "com")
        print(path)

        url = ['http://abc/p{}/'.format(number) for number in range(1,10)]
        print(url)
```

```
This spam is absolutely horrible.
3.14
https://bd/com
['http://abc/p1/', 'http://abc/p2/', 'http://abc/p3/', 'http://abc/p4/', 'http://abc/p5/', 'http:/
```

0.1.1 爬虫原理

网络连接需要一次 Requests 请求和服务器端的 Response 回应。爬虫原理:
- 模拟电脑对服务器发起 Requests 请求 - 接收服务器端的 Response 的内容并解析、提取所需信息

常用的两种爬虫的流程: 多页面和跨页面爬虫流程。

0.1.2 爬虫三大库

- **Requests**

  Requests 库的错误和异常:
- ConnectionError: 网络连接错误异常, 如 DNS 查询失败、拒绝连接等

- HTTPError：HTTP 错误异常，比如网页不存在，返回 404
- URLRequired：URL 缺失异常
- TooManyRedirects：超过最大重定向次数，产生重定向异常
- ConnectTimeout：连接远程服务器超时异常
- Timeout：请求 URL 超时，产生超时异常
- **BeautifulSoup**
  在浏览器中可以得到 BeautifulSoup 的 select 方法的路径：右键 > Copy Selector。
  解析器：html.parser、lxml 等，用法:
  - BeautifulSoup(url.text, “html.parser”)
  - BeautifulSoup(url.text, “lxml”)

- **Lxml**

```
In [2]: ## requests
        import requests

        # 加入请求头，伪装成浏览器
        headers = {
            'User-Agent':'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Geck
        }

        def get_links(url):
            wb_data = requests.get(url,headers=headers)
            print(wb_data.status_code)
            try:
                print(wb_data)
                # print(wb_data.text)
            except ConnectonError:
                print('Requests Error')

        # test
        url = "http://www.baidu.com"
        get_links(url)
```

```
200
<Response [200]>
```

```
In [3]: ## BeautifulSoup, select 方法
        from bs4 import BeautifulSoup
        import requests
```

```
        headers = {
            'User-Agent':'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Geck
        }
        url = "http://bj.xiaozhu.com/"
        res = requests.get(url, headers=headers)
        soup = BeautifulSoup(res.text, 'html.parser')
        # 选择房价的元素的路径 (右键 > Copy Selector), 得到房价值
        prices = soup.select('#page_list > ul > li > div.result_btm_con.lodgeunitname > div > span
        for price in prices:
            print(price, "\t", price.get_text())
```

```
<i>488</i>          488
<i>388</i>          388
<i>388</i>          388
<i>458</i>          458
<i>278</i>          278
<i>498</i>          498
<i>419</i>          419
<i>369</i>          369
<i>498</i>          498
<i>528</i>          528
<i>300</i>          300
<i>298</i>          298
<i>609</i>          609
<i>599</i>          599
<i>408</i>          408
<i>598</i>          598
<i>339</i>          339
<i>428</i>          428
<i>278</i>          278
<i>418</i>          418
<i>278</i>          278
<i>400</i>          400
<i>498</i>          498
<i>448</i>          448
```

```
In [4]: ## BeautifulSoup
        # 需要的 url 如下,
        # <a target="_blank" href="http://bj.xiaozhu.com/fangzi/29968007503.html" class="resule_im
        # Element 的 Selector:  page_list > ul > li:nth-child(1) > a
        url = "http://bj.xiaozhu.com/search-duanzufang-p2-0/"
```

```
        res = requests.get(url, headers=headers)
        soup = BeautifulSoup(res.text, "lxml")
        link = soup.select('#page_list > ul > li > a')
        ## 用相同的 select 方法，得到了该级元素的内容，即 <a...</a>
        #+ 然后用 get(element_name) 方法，获得"href" 属性值
        print(link[0], 2*"\n", link[0].get("href"))

        ## 同时，我们可以进一步用相同的方法提取： title、img 等信息
```

```
<a class="resule_img_a" href="http://bj.xiaozhu.com/fangzi/2597552363.html" target="_blank">
<img alt=" 西域 复式 2 人地铁近 做饭 可发票 出差~" class="lodgeunitpic" data-growing-title="2597552
</a>
```

```
 http://bj.xiaozhu.com/fangzi/2597552363.html
```

0.1.3 实践 Task：爬取酷狗 Top500 的数据

方法是：requests+BeautifulSoup

- url：https://www.kugou.com/yy/rank/home/1-8888.html?from=rank
- 代码：kugou.py, 用 Python3 运行（修复了原代码书中的一个 bug）
- 思路：(1) 观察翻页的各页 url 主入口如何获取；(2) 分别在各页爬取

0.1.4 正则表达式：Python re 模块

- search()
- sub()
- findall()

可以用正则表达式直接解析返回的 html 文件，得到有用的信息。

```
In [5]: import re
        ## re.search()
        a = "one1two2three3"
        info = re.search('\D+', a)
        print(info, "\n", info.group(), "\n")

        ## re.sub()
        new_info = re.sub('\d+', ' ', a)
        print(new_info)

        ## re.findall()
```

```
        infos = re.findall('\D+', a)
        print("\n", infos)

        a= "<a>123</a><a>456</a><a>789</a>"
        ## 边界匹配，括号里的内容作为返回结果
        infos = re.findall('<a>(.*?)</a>', a)
        print("\n", infos)
```

```
<_sre.SRE_Match object; span=(0, 3), match='one'>
 one

one two three

 ['one', 'two', 'three']

 ['123', '456', '789']
```

0.1.5 实践 Task：爬取《斗破苍穹》全文小说

方法是：requests+re

- url: http://www.doupoxs.com/doupocangqiong/
- 代码：doupo_xiaoshuo.py, 用 Python3 运行
- `content.decode('utf-8')`
- `re.S`，re 修饰符，匹配包含换行在内的所有字符
- `time.sleep(1)`，防止请求频率过快导致爬虫失败
  注：**最好加入一个 try/except 判断，如果请求因过快被拒绝，则重新连接，保证数据的完整性**。

0.1.6 实践 Task：爬取糗事百科的段子

方法是：requests+re

- url: https://www.qiushibaike.com/text/
- 代码：qiushibaike.py, 用 Python3 运行
- P.S. 对段子中的`"</br>"` 字符串，需要替换删除

0.1.7 Lxml 库 + Xpath 语法

lxml 库是用来解析 XML 和 HTML 文件的一个 Python 库。
Xpath 是一门在 XML 文档中查找信息的语言，同时支持 HTML 文档。
可参考：Xpath-菜鸟教程
在爬虫实战中，Xpath 路径可以通过浏览器得到：右键 > Copy Xpath。

表达式	描述/结果
/	从根节点选取
//	从匹配的当前节点选取
..	选取当前节点的父节点
@	选取属性
-	-
/cnode	选取根元素 cnode
cnode/node	选取属于 cnode 的子元素的所有 node 元素
//cnode	选取所有 cnode 子元素，不管它们在文档中的位置
cnode//node	选取属于 cnode 的所有 node 子元素，不管它们位于 cnode 之下的什么位置
//@attribute	选取名为 attribute 的所有属性
//li[@attr]	选取所有拥有名为 attr 属性的 li 元素
//li[@attr=" red"]	选取所有 attr 属性值为 red 的 li 元素
/text()	获取标签中的文本

比较三种方法：re/BeautifulSoup/Lxml

- 代码：compare.py
- 速度：Lxml ≈ 正则 > BeautifulSoup
- 难度：Lxml ≈ BeautifulSoup < 正则

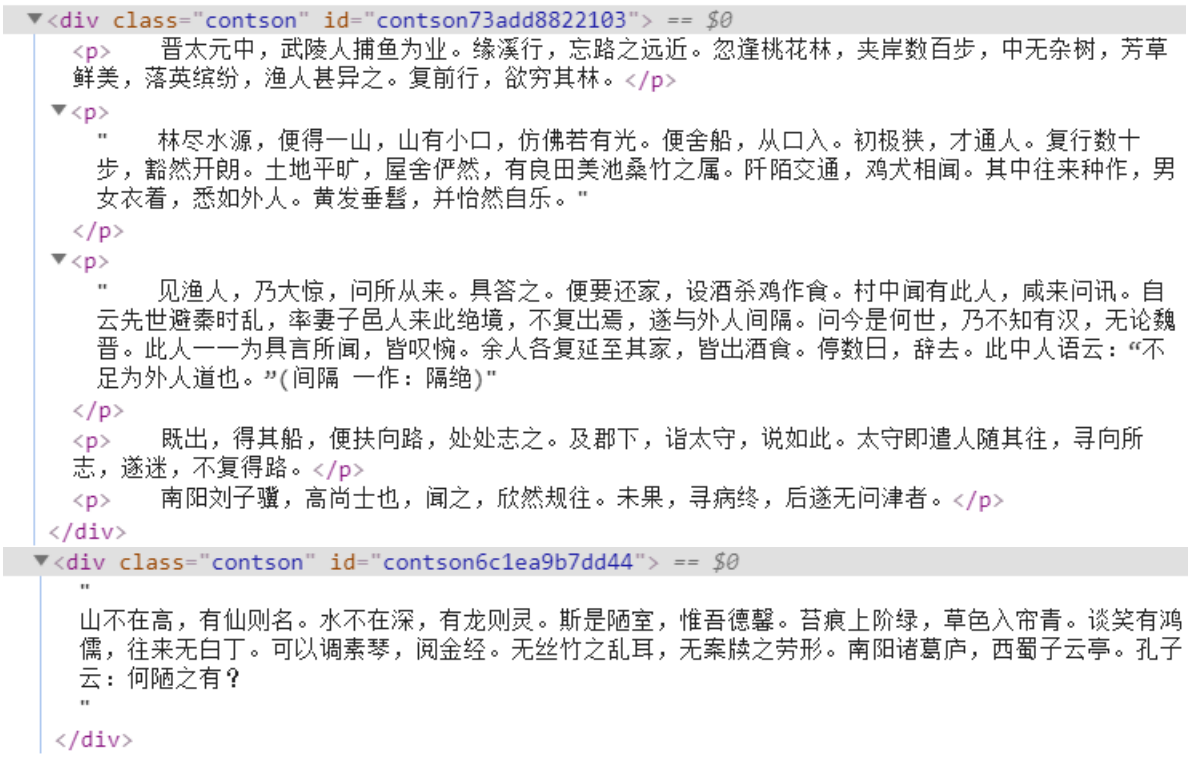
所以当爬取的数据量大，且需要快速实现时，**选择 Lxml 是最佳选择，因为速度快，实现简单**。

```
In [6]: ## Xpath, 获取糗事百科/文字中的一个页面下的所有 user name
        import requests
        # lxml 库
        from lxml import etree
        headers = {
            'User-Agent':'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Geck
        }
        url = 'http://www.qiushibaike.com/text/'
        res = requests.get(url,headers=headers)
        ## etree.HTML
        selector = etree.HTML(res.text)
        usernames_path = selector.xpath('//div[@class="article block untagged mb15 typs_long" or
                                        @class="article block untagged mb15 typs_hot" or  \
                                        @class="article block untagged mb15 typs_old"]')
        ## 以上复杂的形式，可以用 starts-with 简化、替代
        usernames_path = selector.xpath('//div[starts-with(@class, "article block untagged mb15")]
```

```
        for username_path in usernames_path:
            if username_path.xpath('div[1]/a[2]/h2'):
                ## 因为返回的是 list，所以取唯一的一个元素即可， [0]
                username = username_path.xpath('div[1]/a[2]/h2/text()')[0].strip()
                print(username)
            else:  # 匿名用户没有以上节点
                pass
```

小小獠
阳光倾城暖谁心
孤犬落寞吠
未识已终
lonely 仓鼠
乌漆嘛黑的夜
义行天下 7
鹰嫂
不喜欢洗锅
千年瀑布
小呆妹!
偷🄵
夕冬温存
手倦🄵书
冬天的铁门很甜
偷🄵
-小门神 ~
吃个榴莲压压惊
黄山小妖
12312a
饮最烈的酒、艹最爱…
又一盏素酒
秀的拽，你不配
惊鸿一剑
歌薇洛洗护潮州妹

用 string(.) 获取标签套标签的文本内容 文言文-古诗文网中的多段长文和只有一段的文言文的节点设置不同。
此时，为了把 div 节点下的所有文本都获取，可以用 string(.) 解决该问题。




```
In [7]: ## 函数，代码复用
        def stringall(url, name):
            res = requests.get(url, headers=headers)
            selector = etree.HTML(res.text)
            guwen_path = selector.xpath('//div[@class="contson"]')[0]
            ## string(.)
            guwen = guwen_path.xpath('string(.)').strip("\n")
            info = "******" + name + "******"
            print(info)
            print(guwen)
            print()

        ## 桃花源记
        url = "https://so.gushiwen.org/shiwenv_73add8822103.aspx"
        stringall(url, "桃花源记")

        ## 陋室铭
        url = "https://so.gushiwen.org/shiwenv_6c1ea9b7dd44.aspx"
        stringall(url, "陋室铭")
```

****** 桃花源记 ******
晋太元中，武陵人捕鱼为业。缘溪行，忘路之远近。忽逢桃花林，夹岸数百步，中无杂树，芳草鲜美，落英缤纷，

林尽水源，便得一山，山有小口，仿佛若有光。便舍船，从口入。初极狭，才通人。复行数十步，豁然开朗。土地
见渔人，乃大惊，问所从来。具答之。便要还家，设酒杀鸡作食。村中闻有此人，咸来问讯。自云先世避秦时乱，
既出，得其船，便扶向路，处处志之。及郡下，诣太守，说如此。太守即遣人随其往，寻向所志，遂迷，不复得路
南阳刘子骥，高尚士也，闻之，欣然规往。未果，寻病终，后遂无问津者。

****** 陋室铭 ******
山不在高，有仙则名。水不在深，有龙则灵。斯是陋室，惟吾德馨。苔痕上阶绿，草色入帘青。谈笑有鸿儒，往来无白

0.1.8 实践 Task：爬取豆瓣图书 Top250

方法是：Requests + Lxml

- 将结果存储为结构化的 csv 文件格式
- 代码：doubanbook.py

0.1.9 使用 API 爬取数据

- 返回的一般是 JSON 格式的数据，需要用 JSON 解析库解析 JSON 数据
- 可以用 Requests 库发起请求（GET 或者 POST）。但一般来说，开放的 API 有一些限制：需要验证、或者有调用次数的限制、或者需要 apikey 才有调用权限。
- **API 爬虫很常见，比如开源的聊天机器人、天气、地图等等**

In [8]:
```
## json
import json
jsonstring = '{"name":"xiaoming", "gender":"man"}'
json_data = json.loads(jsonstring)
print(json_data["name"])
```

```
xiaoming
```

In [9]:
```
# Python2 会有中文编码问题，打印 json 字符串，输出的中文就成了 unicode 码
import json
d = {'name': '张三', 'age': '1'}
# Python3: {'name': ' 张三', 'age': '1'}
# Python2: {'age': '1', 'name': '\xe5\xbc\xa0\xe4\xb8\x89'}
print(d)
# Python2: jd = json.dumps(d, ensure_ascii=False, encoding='utf-8'))
jd = json.dumps(d, ensure_ascii=False)
print(jd)
```

```
{'name': ' 张三', 'age': '1'}
{"name": " 张三", "age": "1"}
```

0.1.10 数据库存储

- MongoDB 数据库
  MongoDB 是一种非关系型数据库 (NoSQL)。NoSQL 可以解决大规模数据集合多重数据种类的问题。
  NoSQL 分为 4 大类：键值存储数据库 Redis、列存储数据库 Hbase、文档型数据库 MongoDB、图形数据库 Graph。
  数据库和集合，类似于 Excel 文件和其中的表格，一个数据库可以有多个集合。
  - 安装 MongoDB 数据库：sudo apt install mongodb-clients, sudo apt install mongodb-server
  - 配置数据库：mongod -dbpath path
  - 启动：mongo，使用 MongoDB 期间，启动服务的窗口不能关闭
  - Python 安装 pymong 库：pip3 install pymongo
  - 导出数据：mongoexport -d mydb -c test –csv -f name,gender,age -o test.csv
- MySQL 最流行的开源的关系型数据库

```
In [10]: import pymongo
         client = pymongo.MongoClient('localhost', 27017)  # 连接数据库
         mydb = client['mydb']  # 新建 mydb 数据库
         test = mydb["test"]  # 新建 test 数据集合
         # 删除一个集合中的所有文档
         result = test.delete_many({})
         test.insert_one({"name":"Ming", "gender":"male", "age":"30"})
         ## 在 MongoDB 设置的路径下，打开命令行窗口，运行以下命令导出数据
         # -d: 数据库， -c: 数据集合， -f: 导出的字段
         # mongoexport -d mydb -c test --csv -f name,gender,age -o test.csv

Out[10]: <pymongo.results.InsertOneResult at 0x7ff8b87fad48>
```

0.1.11 实践 Task：爬取豆瓣音乐 Top250、豆瓣电影 Top250

方法是：Requests + Lxml + re + **MongoDB**

- 代码：doubanmusic.py
- 修复了一个 bug，网页代码有变化
- 导出结果：mongoexport -d mydb -c musictop –csv -f name,author,style,time,publisher,score -o douban_music.csv

0.1.12 多进程爬虫

可以使用 Python 的 multiprocessing 库的进程池方法进行多进程爬虫。

0.1.13 实践 Task：爬取 58 同城二手市场的商品信息

方法是：Requests + Lxml + **MongoDB + Muitiprocessing/Pool**

- 代码：58project/main.py
- 可以参考的一个思想，也是在实践中常常遇到的问题：如果由于某些原因，爬虫程序中断，如何**不手动更改却可以实现断点续爬**？
  代码中的实现方法是一个参考。

```
In [11]: '''
         # 导入 multiprocessing 库的 Pool 模块

         # 创建进程池，processes 参数为设置进程的个数
         pool = Pool(processes=4)

         # 用 map() 函数进行进程运行
         # func 参数为需运行的函数，在爬虫实战中，为爬虫函数
         # iterable 为迭代参数，在爬虫实战中，可为多个 URL 列表进行迭代。
         pool.map(func,urls)
         '''
         from multiprocessing import Pool
```

0.1.14 异步加载

异步加载技术（AJAX），即异步 JavaScript 和 XML，是指一种创建交互式网页应用的网页开发技术。通过在后台与服务器进行少量数据交换，AJAX 可以使网页实现异步更新。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。

- 通过下滑进行浏览，并没有分页的信息，而是一直浏览下去，而网址信息并没有改变，可以判断该网页使用了异步加载技术
- 也可以通过查看数据是否在**网页源代码**中来判断网页是否采用了异步加载技术。下滑后的信息并不在网页源代码中，以此判断使用了异步加载技术

对于异步加载网页，之前的方法是无法获取到页面信息的，那么如何爬取异步加载的网页数据呢?

- 想要抓取这些通过异步加载方法的网页数据，需要了解网页是如何加载这些数据的，该过程就叫做逆向工程
- **浏览器的 Network 选项卡**可以查看网页加载过程中的所有文件信息，通过对这些文件的查看和筛选，找出需抓取数据的加载文件，以此来设计爬虫代码
- 分页文件大部分在 Network 选项卡中的 XHR 选项。通过查看加载时请求的 URL，找到规律且简化的 URL，然后用之前的爬虫方法

0.1.15 实践 Task：爬取简书用户动态信息

- url：http://www.jianshu.com/u/9104ebf5e177
- 方法是：Requests + Lxml + MongoDB
- 代码：jianshu_timeline.py
- 网页域名构造已经发生变化，没有了复杂的动态 id，直接可以得到 page 信息
  `https://www.jianshu.com/u/9104ebf5e177?order_by=shared_at&page=8`

0.1.16 表单交互与模拟登录

Requests 库的 POST 方法

Cookie 模拟登录

- Cookie 指网站为了辨别用户身份、进行 session 跟踪而储存在用户本地终端上的数据。
- 可以提交 Cookie 模拟登录新浪微博、豆瓣等网站。

```
In [12]: ## 书中的豆瓣登录方法，但现在这种方法已经行不通，改为 Cookie 模拟登录了
         #+ Requests 的 POST 方法
         import requests
         url = 'https://www.douban.com/accounts/login'
         params = {
             'username':'**',
             'password':'**'
         }
         html = requests.post(url, data=params)

         ## Cookie 登录豆瓣，Requests 的 GET 方法
         import requests
         url = 'https://www.douban.com/'
         headers = {
             "Cookie":'**'
         }
         html = requests.get(url, headers=headers)
         #print(html.text)
```

0.1.17 实践 Task：爬取拉勾网招聘信息

- url：http://www.lagoucom/
- 方法是：Requests + 异步加载 + 表单交互 + MongoDB

- **※ 推荐这个任务，看似不同寻常，但并没有相信中那么复杂**。掌握这个任务，可以应对很多相似的问题。
  - 异步加载 + 提交表单，获得翻页后的 Response
  - 异步加载的 URL
  - 提交表单，用 POST 直接获取 Response，json 格式数据，直接解析就可以得到有效信息
    当 JSON 格式很复杂时，可通过 “Preview” 标签来观察。招聘信息在 content-positionResult-result 中。
- 代码示例如下所示，完整版代码：lagou.py

In [13]:
```
import requests
import json
import time
import pymongo
## MongoDB
client = pymongo.MongoClient('localhost', 27017)
mydb = client['mydb']
lagou = mydb['lagou']

headers = {
    'Cookie':'**',
    'User-Agent':'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gec
    'Connection':'keep-alive'
}

def get_page(url,params):
    ## 异步加载 + 提交表单，获得翻页后的 Response
    html = requests.post(url, data=params, headers=headers)
    ## json
    json_data = json.loads(html.text)
    total_Count = json_data['content']['positionResult']['totalCount']
    ## 获取总页数
    page_number = int(total_Count/15) if int(total_Count/15)<30 else 30
    get_info(url,page_number)

def get_info(url,page):
    for pn in range(1,page+1):
        # 表单
        params = {
            'first': 'true',
```

```
            'pn': str(pn),
            'kd': 'Python'
        }
        try:
            ## 异步加载 + 提交表单，获得翻页后的 Response
            ## 用 POST 直接获取 Response，json 格式数据，直接解析就可以得到有效信息
            html = requests.post(url,data=params,headers=headers)
            json_data = json.loads(html.text)
            results = json_data['content']['positionResult']['result']
            ## json 解析
            for result in results:  ## 省略了很多 infos 的元素
                infos = {
                    'businessZones':result['businessZones'],
                    'city':result['city'],
                }
                lagou.insert_one(infos)
                time.sleep(2)
        ## try/except
        except requests.exceptions.ConnectionError:
            pass

if __name__ == '__main__':
    # 异步加载的 URL
    url = 'https://www.lagou.com/jobs/positionAjax.json'
    params = {
        'first': 'true',
        'pn': '1',
        'kd': 'Python'
    }
    # 为了不报错注释，实际运行不注释
    # get_page(url,params)
```

0.1.18 Selenium 模拟浏览器

Selenium 是一个用于 Web 应用程序测试的工具。Selenium 直接运行在浏览器中，就像真正的用户在操作一样。

由于这个性质，Selenium 也是一个强大的网络数据采集工具，其可以让浏览器自动加载页面，使用了异步加载技术的网页也可获取数据。- pip3 install selenium - Selenium 自己不带浏览器，需要配合第三方浏览器来使用。常用的浏览器：Firefox、Chrome、PhantomJS - PhantomJS 是无界面的，开销小，速度快 - Selenium 和 PhantomJS 的配合使用可以完全模拟用户在浏览器上的所有操作，包括

输入框内容填写、单击、截屏、下滑等各种操作。对于需要登录的网站，用户可以不需要通过构造表单或提交 cookie 信息来登录网站。

一般不到万不得已（异步加载 + 表达交互依然无法解决问题），基本上不会使用 Selenium 的方式。Learn by doing，遇到具体问题再来看语法。

```
In [14]: ## Selenimu 模拟浏览器，登录豆瓣
         from selenium import webdriver
         ## 当使用 Firfox 或者 Chrome 浏览器时，需要下载 geckodriver
         #+ 并添加到 PATH 或者 mv geckodrive /usr/local/bin/
         driver = webdriver.Firefox()
         driver.get('https://www.douban.com/accounts/login')
         driver.implicitly_wait(10)
         driver.find_element_by_class_name('account-tab-account').click()
         driver.find_element_by_id('username').clear()
         driver.find_element_by_id('username').send_keys('123456')
         driver.find_element_by_id('password').clear()
         driver.find_element_by_id('password').send_keys('abc123')
         driver.implicitly_wait(2)
         driver.find_element_by_class_name('account-form-field-submit ').click()
         #print(driver.page_source)
         driver.implicitly_wait(10)
         driver.close()
```

0.1.19 Scrapy 爬虫框架

非常好用的爬虫框架，但此书的讲解和例子不够好，不再列出笔记。
后续通过其他资料学习和给出实践。

0.2 小结

0.2.1 爬虫原理

结合实例，了解搭建爬虫的逻辑和思路，分析和拆解问题。

0.2.2 爬虫三大库

- Requests，GET 和 POST 请求
- Lxml，Xpath 方法
- BeautifulSoup，select 方法

0.2.3 Python 模块

- re
- json
- time、csv、try/except、pymongo、multiprocessing
- Selenium

0.2.4 除了三大库，还有与爬虫相关的方方面面、细节 & 技巧

- 数据库 MongoDB
- 多进程/多线程
- 异常判断
- 断点自动续爬
- **异步加载**
- **表单交互**
- **Cookie 模拟登录**
- Selenium 模拟浏览器
- Scrapy 爬虫框架

0.2.5 爬虫实例汇总

- 爬取糗事百科的段子
- 爬取《斗破苍穹》全文小说
- 爬取酷狗 Top500 的数据
- 爬取豆瓣图书 Top250、豆瓣音乐 Top250、豆瓣电影 Top250
- 使用 API 爬取数据：百度地图、高德地图
- 爬取 58 同城二手市场的商品信息
- 爬取简书用户动态信息
- 爬取拉勾网招聘信息

0.2.6 爬虫常见方法

- Requests + Lxml/re + MongoDB + Muitiprocessing/Pool
- Requests + 异步加载 + 表单交互 + MongoDB

```
In [15]: ## 与爬虫 or 文本处理有关的 Python 模块/库
         import requests
         import re
         from lxml import etree
         from bs4 import BeautifulSoup
         import json
         import time
```

```
import csv
import pymongo
from multiprocessing import Pool
from selenium import webdriver
```