**Chap2：**

1. BeautifulSoup中使用get\_txt方法可以去掉标签；用find、findall可以用来查找标签，使用.children标签可以只找出子标签，而不显示其他的后代标签，.next\_siblings会寻找兄弟标签，但不会显示自身，.parent可以寻找父标签
2. Find或findall()其中可以跟一个匿名函数，匿名函数的输入参数必须是一个标签且结果是布尔类型，最后把评估结果为真的结果保留，为假的剔除

**Chap3：**

1. 利用random.seed()可以生成随机数种子
2. 所有的标题都在h1-span标签里，所有的正文都在div-bodyContent标签里；编辑链接位于caedit-span-a标签里
3. Scrapy简单实用：使用scrapy startproject 项目名来建立一个项目，在items.py文件中定义一个类，每一个item对象代表网站的一个页面，在spiders文件夹里新建py文件，来写相关代码；通过scrapy crawl 条目名称 来调用爬虫，

**Chap4：**

1. API：应用编程接口

**Chap5：**

1、urllib.urlretrieve(url[, filename[, reporthook[, data]]])可以用来下载网页的html，startswith(str)方法用来查找字符串是否以str开头；os.path.dirname(path)可以取得地址；os.makedirs（path）可以创建目录

2、csv格式，建立一个csv文件可以使用如下命令：csvFile = open(r'./test.csv','wt',newline='', encoding='utf-8')；建立一个写对象：writer = csv.writer(csvFile)；写入数据：writer.writerow(csvRow)

**Chap6：**

1. bytes函数的功能将之转换为字节
2. csv.DictReader方法可以将csv文件中的每一行转换为字典对象
3. 读取csv文件，首先打开文件，利用read命令读取，decode进行解码，之后将解码后的数据转换成StringIO对象进行操作
4. StringIO可以实现在内存中缓存数据
5. 解析PDF文件：PDFResourceManager（）创建一个资源管理器对象，用于存储共享资源，如字体或图像；LAParams()设定参数进行分析；TextConverter(rsrcmgr, retstr, laparams=laparams)创建一个pdf解析器对象；process\_pdf（）用来进行PDF解析，getvalue返回内存的所有数据
6. 解析docx:所有的docx文件为了节省空间都进行了压缩，所以需要利用Python自带的库zipfile进行解压，读取docx文件流程如下：先利用urlopen打开网上的文件，利用read进行读取，变为字符串，在利用BytesIO读入内存中，最后利用ZipFile解压缩，decode解码；text读取Unicode数据，content读取二进制数据

**Chap7：**

清洗数据：使数据变成n-gram模型，易于处理，同时清除数据中的凌乱字符，可以利用re模块中的sub函数对某些凌乱字符进行替代；清除数据中的重复字符，首先将n-gram进行字符化，利用Counter函数计算出其频率，用sorted函数按频率进行排序，最终使用OrderedDict函数按顺序固定数据，生成一个有序的字典

**Chap8：自然语言处理**

dict.items()返回可遍历的元组

1. 利用语句元素的相关性，生成马尔科夫链：建立一个二维的字典，通过迭代，将马尔科夫链以字典元素的形式给出，具体形式如下：{‘a’:{‘b’:1,’c’:2,’d’:3……}}
2. 广度优先搜索算法：先搜索与初始页面连接的所有链接，如果链接不包含目标页面，再进行第二层的搜索，直至达到了搜索深度。
3. 利用nltk做统计分析：1）创建FreqDist频率分布对象可以查看单词的使用频率，例如使用使用most\_common方法可以查看出现频率最高的某几个词2）利用ngrams函数可以将文本分成任意规模的n-gram序列3）pos\_tag可以用来识别词性

**Chap8：穿越表单，解决网站的登录的问题**

1. 简单的网站登录：首先需要找到实际表单提交的页面，之后根据需要post的信息建立相应的字典，即找到想要提交的字段名称以及表单的action属性，之后利用requests模块中的函数进行登录。
2. 当表单需要提交图像时，跟简单登录类似，不过字典的值改为需要提交的打开的图像
3. 需要保持登录状态：这个问题出现的原因是，当登录之后，相关信息会保存在cookie中，网站依据cookie来判断是否处于登录状态，如果不及时关注cookie信息，再次打开页面会需要重新登录。解决这个问题可以使用session对象，session会持续跟踪会话信息
4. 解决http基本认证问题：利用auth模块。利用HTTPBasicAuth生成一个特殊的对象，用来作为auth参数提交

**Chap10：采集客户端语言javascript**

1. javascript简介：所有变量都用var关键词定义，并可以把函数当做变量使用
2. 常用的javascript库：1、jquery，可以动态的创建html内容，只有在javascript代码执行之后才会显示2、google analytics 3、google地图
3. 解决使用Ajax和DHTML技术的页面采集问题：两种思路，一种是直接从javascript中采集，另一种是利用第三方库运行javascript，直接采集浏览器的页面
4. 使用selenium时，需要建立driver来运行javascript，通过函数webdriver.Chrome(executable\_path=安装路径)来实现，然后get函数来建立页面，通过类似于find\_element\_by\_id()、find\_element\_by\_css\_selsector(‘#content’)、find\_element\_by\_tag\_name(‘div’)等函数来查找元素;为了等待页面完全加载，引入了WebDriverWait，如WebDriverWait (driver, 10)来控制隐式等待的最大时间，定位器是By对象，通过expected\_conditions来使用，当元素被触发的期望条件满足时启动。使用示例如下：element = WebDriverWait(driver, 10).until(expected\_conditions.presence\_of\_element\_located((By.ID, 'loadedButton')))

**Chap11：处理图片与文字**

1、处理文字图片：使用tesseract，使用命令示例：tesseract text.tif textoutput | cat textoutput.txt，因为tesseract是外部程序，所以需要调用，故使用subprocess包，通过命令subprocess.call([‘tesseract’,newFilePath,’output’])调用并保存结果。

2、当从网站上抓取图片进行处理时，可以利用selenium进行操作，利用driver.find\_element\_by\_id().click()来点击特定的按钮，判断某个按钮是否可以点击时，可以通过按钮的属性进行判断，如while 'pointer' in driver.find\_element\_by\_id('sitbReaderRightPageTurner').get\_attribute('style'):

3、对于登录需要验证码来说，有必要对tesseract进行必要的训练

**Chap12：使爬虫更像人类**

1. 通过修改请求头，可以避免爬虫被识别出来，其中，最重要的是user-agent
2. 为了阻止爬虫，网站有时候会设置蜜罐陷阱，将某些字段隐藏起来作为陷阱。为了避免这点，可以使用.isdisplayed()函数来查看有没有显示在页面上 ，如果没有显示，就要小心

**Chap13：写单元测试**

1. 单元测试的要求：每个单元测试测试一个零件功能的一方面；每个单元测试可以独立运行；每个单元测试至少包括一个断言；单元测试与生产代码分开
2. 单元测试使用的工具：unittest模块，需要导入并继承unittest.TestCase类，为每个单元测试的开始和结束提供setUp和tearDown函数，提供不同类型的断言，将所有以test\_开头的函数当做单元测试运行，忽略不带有test\_开头的函数
3. 对selenium进行单元测试，只需要断言即可，如：assert “Monty Python” in driver.title
4. 与网站进行交互：点击动作.click()，点击并发送字段.click().send\_keys，使用动作链，示例：

actions = ActionChains(driver).click(firstnameField).send\_keys("Ryan")\

.click(lastnameField).send\_keys("Mitchell").send\_keys(Keys.RETURN)

actions.perform()

鼠标拖放示例：

Actions = ActionChains(driver)

Actions.drag\_and\_drop(element, target).perform()

截屏命令driver.get\_screenshot\_as\_file(文件保存目录)

**Chap14：为了避免ip的封杀，使用远程采集**

1. 为了避免个人信息被泄露，需要对ip进行加密，可以利用tor来实现。当使用socket通信时，可以通过命令：

socks.set\_default\_proxy(socks.SOCKS5, "localhost",9150) socket.socket = socks.socksocket

来使用tor隐藏ip

如果使用selenium时，只需要增加service\_args参数设置代理参数，如：

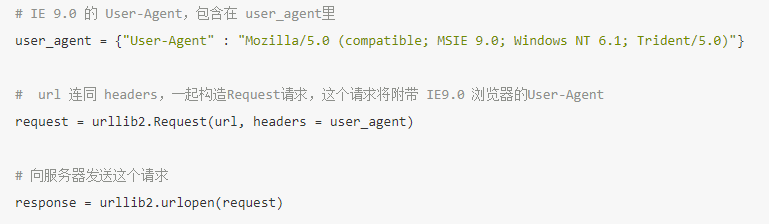
service\_args = ['--proxy=localhost:9150', '--proxy-type=socks5',]

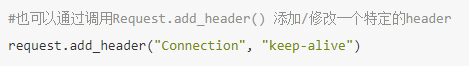
driver = webdriver.Chrome(executable\_path=r'C:\ProgramFiles (x86)\Google\Chrome\Application\chromedriver.exe',service\_args=service\_args)

爬虫学习

Day1：

1. 给http request添加特定的header





编码工作使用urllib的urlencode()函数，帮我们将key:value这样的键值对，转换成"key=value"这样的字符串，解码工作可以使用urllib的unquote()函数。（ 注意，不是urllib2.urlencode())

1. 有些网页内容使用AJAX请求加载，这种数据无法直接对网页url进行获取。但是只要记住，AJAX请求一般返回给网页的是JSON文件，只要对AJAX请求地址进行POST或GET，就能返回JSON数据了
2. 处理ssl证书

首先需要导入ssl处理模块，然后设置忽略未经核实的ssl证书，之后在打开链接时添加相应参数即可，如下所示：



1. 如果想要使用代理，则需要自定义handler处理器，简单定义过程如下：



通过使用proxyHandler来设置代理服务器，使用时将上文所述HTTPHandler替换，并添加相应服务器即可

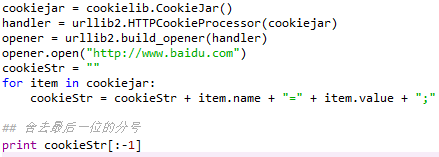
HTTPPasswordMgrWithDefaultRealm()类将创建一个密码管理对象，用来保存 HTTP 请求相关的用户名和密码，主要应用两个场景：验证代理授权的用户名和密码 (ProxyBasicAuthHandler())；验证Web客户端的的用户名和密码 (HTTPBasicAuthHandler())





1. 使用cookie，其格式如下：Set－Cookie: NAME=VALUE；Expires=DATE；Path=PATH；Domain=DOMAIN\_NAME；SECURE

使用cookielib库来存储cookies，主要用到cookiejar，如果需要和本地文件交互，则需要MozillaCookjar（）或者LWPCookieJar（），使用示例如下：



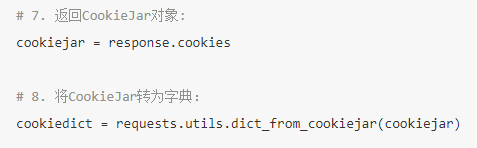


1. 进阶，使用Requests，基本命令如下：









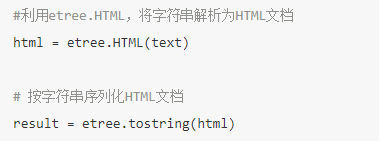
使用cookie，可以用session命令发送请求

想要跳过ssl证书验证，只需要：



Day2：页面解析

1. 使用lxml：





Day3：动态网页，包括利用selenium等进行页面的操作，之后了解scracy框架。项目被收集后，会使用pipeline进行操作，包括检查跟查重等。为了启动这个组件，将之添加到setting中。