实战|手把手教你用Python爬虫(附详细源码)

原创 我是J哥 [菜J学Python](javascript:void(0);) 2020-05-18 23:22

收录于合集

#Python实战73个

#爬虫30个

大家好，我是J哥，专注原创，致力于用浅显易懂的语言分享爬虫、数据分析及可视化等干货，希望人人都能学到新知识。最近J哥做了个爬虫小项目，感觉还挺适合新手入门的，于是迫不及待想分享给大家。

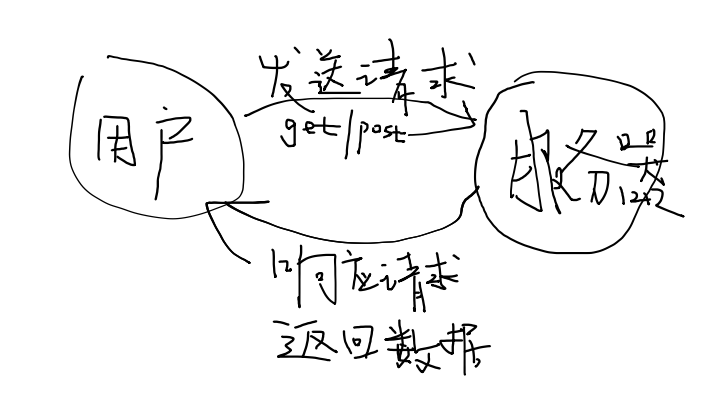
**什么是爬虫？**

实践来源于理论，做爬虫前肯定要先了解相关的规则和原理，要知道互联网可不是法外之地，你一顿爬虫骚操作搞不好哪天就...

首先，咱先看下爬虫的定义：网络爬虫（又称为网页蜘蛛，网络机器人，在FOAF社区中间，更经常的称为网页追逐者），是一种按照一定的规则，自动地抓取万维网信息的程序或者脚本。一句话概括就是网上信息搬运工。

我们再来看下爬虫应该遵循的规则：robots协议是一种存放于网站根目录下的ASCII编码的文本文件，它通常告诉网络搜索引擎的漫游器（又称网络蜘蛛），此网站中的哪些内容是不应被搜索引擎的漫游器获取的，哪些是可以被漫游器获取的。一句话概括就是告诉你哪些东西能爬哪些不能爬。

了解了定义和规则，最后就是熟悉爬虫的基本原理了，很简单，作为一名灵魂画手，我画个示意图给你看下就明白了。



(⊙o⊙)…尴尬，鼠标写字咋这么丑，都不好意思说自己学过书法，好一个脸字打得呱呱响。

**项目背景**

理论部分差不多讲完了，有些小朋友估计要嫌我啰嗦了，那就不废话，直接讲实操部分。本次爬虫小项目是应朋友需求，爬取中国木材价格指数网中的红木价格数据，方便撰写红木研究报告。网站长这样：



所需字段已用红框标记，数据量粗略看了下，1751页共5万多条记录，如果你妄想复制粘贴的话，都不知道粘到猴年马月了。而python只要运行几分钟就能把所有数据保存到你的excel里，是不是很舒服？

图片

**项目实战**

工具：PyCharm

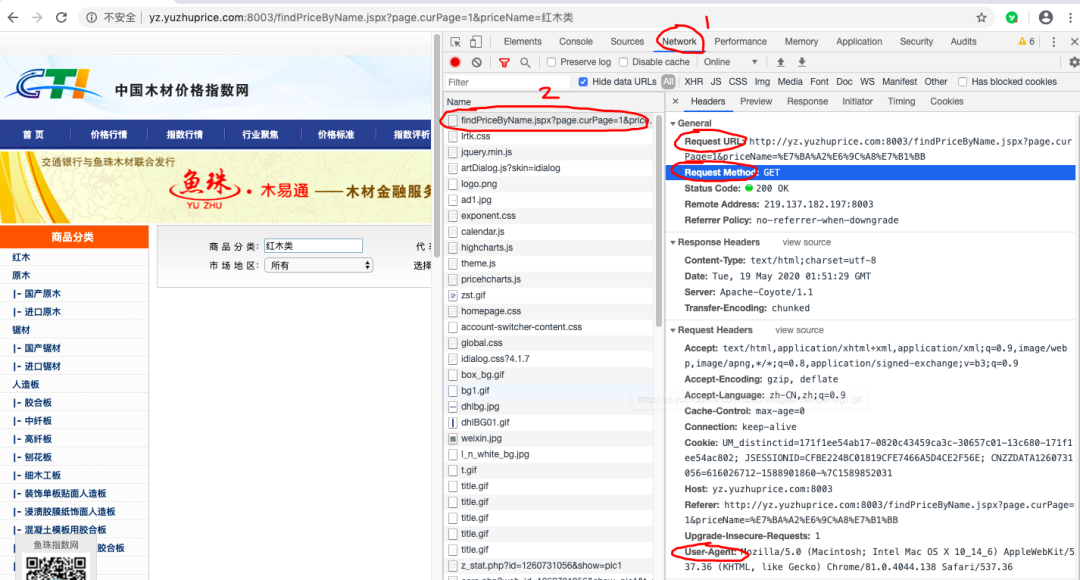
Python版本：Python 3.7

浏览器：Chrome (推荐)

对于第一次写爬虫的朋友可能觉得很麻烦，咱不慌，由浅入深，先爬一页数据试试嘛。

一爬取一页

首先，我们需要简单分析下网页结构，鼠标右键点击检查，然后点击Network，刷新网页，继续点击Name列表中的第一个。我们发现此网站的请求方式为GET，请求头Headers反映用户电脑系统、浏览器版本等信息。



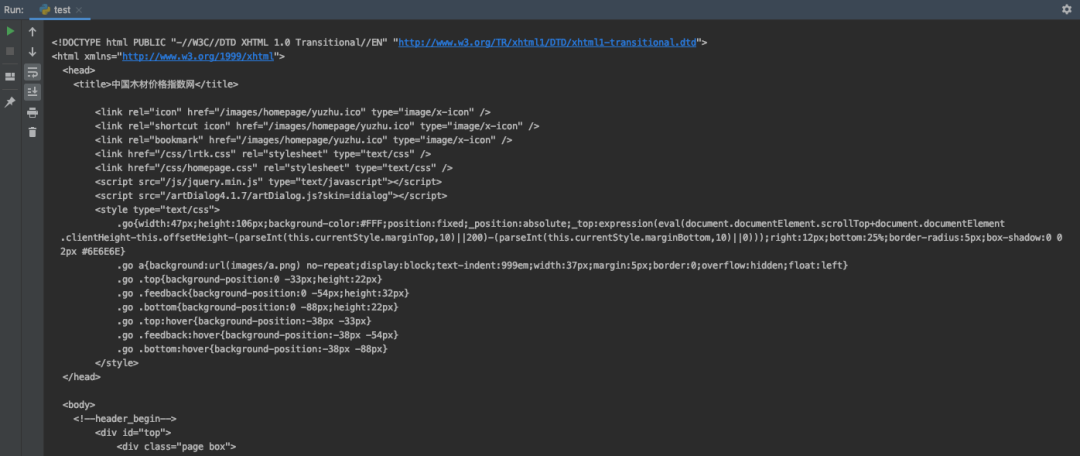
接着，把爬虫所需的库都pip安装一下并导入，所有库的功能都有注释。

import csv  #用于把爬取的数据存储为csv格式，可以excel直接打开的  
import time  #用于对请求加延时，爬取速度太快容易被反爬  
from time import sleep #同上  
import random  #用于对延时设置随机数，尽量模拟人的行为  
import requests  #用于向网站发送请求  
from lxml import etree    #lxml为第三方网页解析库，强大且速度快

构造请求url，添加头部信息headers即复制前文标记的User-Agent，通过requests.get方法向服务器发送请求，返回html文本。添加headers目的在于告诉服务器，你是真实的人在访问其网站。如果你不添加headers直接访服务器，会在对方服务器显示python在访问，那么，你很可能会被反爬，常见的反爬就是封你ip。

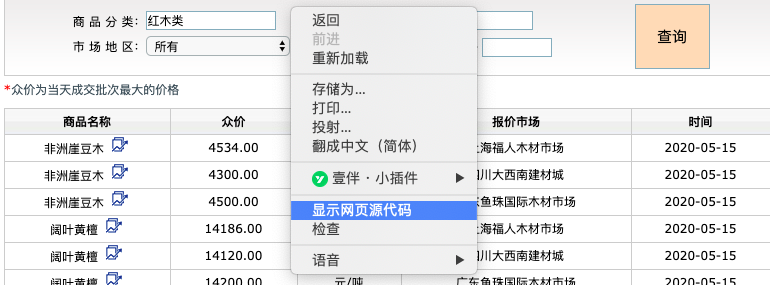
url = 'http://yz.yuzhuprice.com:8003/findPriceByName.jspx?page.curPage=1&priceName=%E7%BA%A2%E6%9C%A8%E7%B1%BB'  
headers = {  
    'User-Agent': "Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_14\_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.129 Safari/537.36",  
}  
response = requests.get(url, headers=headers, timeout=10)  
html = response.text    
print(html)

我们运行下以上代码，看下效果：



看到这个，第一次接触爬虫的朋友可能会有点懵。

其实这就是网页源代码，咱们右键打开下源代码看一哈。



长这样：



而我们需要提取的数据，就潜藏在这网页源代码中，我们要用lxml库中的etree方法解析下网页。

parse = etree.HTML(html)  #解析网页

解析完之后，就可以开开心心的提取我们所需要的数据了。方法很多，比如xpath、select、beautiful soup，还有最难的re(正则表达式)。本文爬取的数据结构较为简单，就直接用xpath玩一下吧。

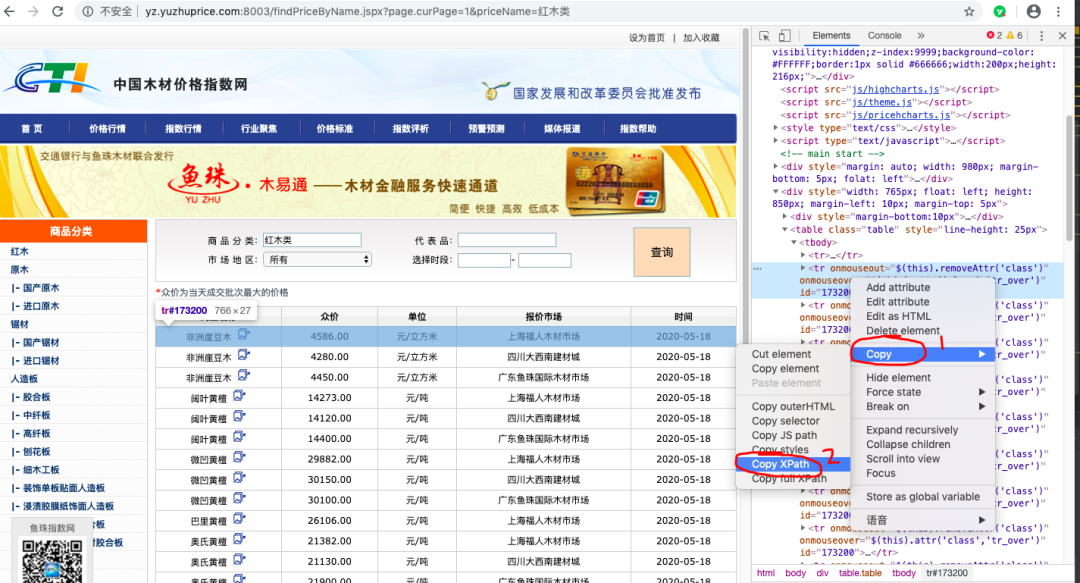


我们发现，每一行数据对应源码里的一个id=173200的tr，那就先把这些tr都提取下来。

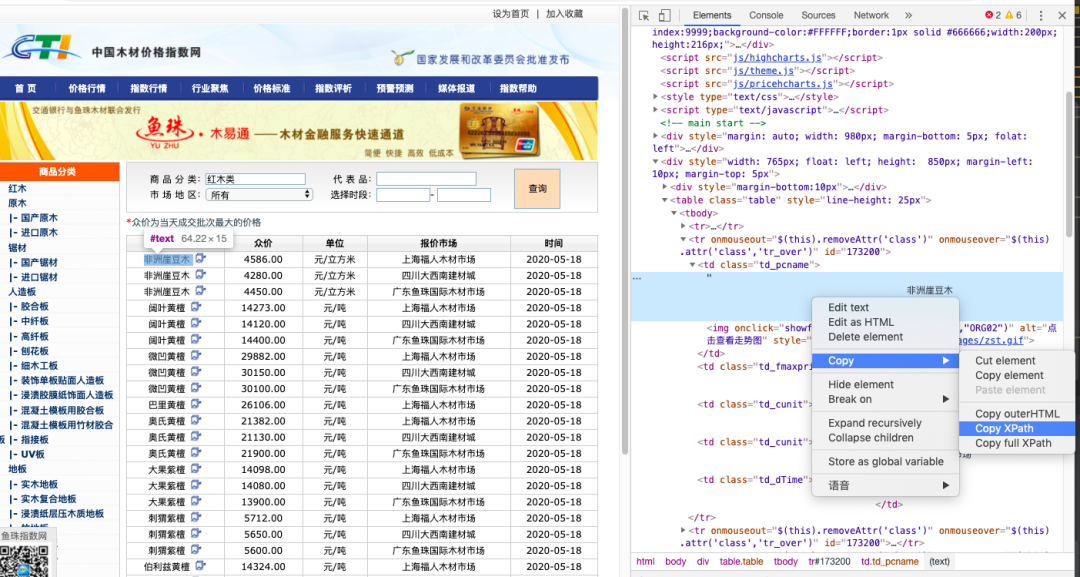
all\_tr = parse.xpath('//\*[@id="173200"]')

有些小伙伴不会写xpath。

那就找个简单办法，直接copy所需的xpath。



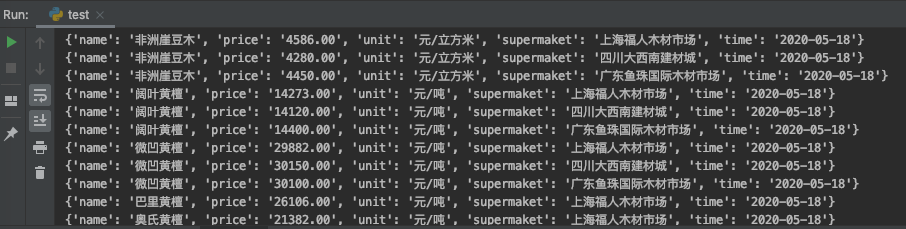
所有tr都提取下来了，接下来就得依次从tr里面提取具体字段了。比如提取商品名称字段，点开第一个tr，选中商品，copy其xpath。其他字段同理。



以下要注意几点，tr={key1 : value1, key2 : value2 }是python的字典数据类型（你也可以根据自己兴趣或需要存为列表或元组类型）。''.join是指把获取到的列表转为字符串。./是指继承前面的//\*[@id="173200"]，strip()表示对提取的数据进行简单的格式清洗。

for tr in all\_tr:  
    tr = {  
        'name': ''.join(tr.xpath('./td[1]/text()')).strip(),  
        'price': ''.join(tr.xpath('./td[2]/text()')).strip(),  
        'unit': ''.join(tr.xpath('./td[3]/text()')).strip(),  
        'supermaket': ''.join(tr.xpath('./td[4]/text()')).strip(),  
        'time': ''.join(tr.xpath('./td[5]/text()')).strip()  
    }

咱们打印一下print(tr)，看下效果。



但还没完，数据有了，咱们还得保存csv格式到本地，这一步比较简单，直接贴代码。

    with open('wood.csv', 'a', encoding='utf\_8\_sig', newline='') as fp:  
        # 'a'为追加模式（添加）  
        # utf\_8\_sig格式导出csv不乱码  
        fieldnames = ['name', 'price', 'unit', 'supermaket', 'time']  
        writer = csv.DictWriter(fp, fieldnames)  
        writer.writerow(tr)

打开下刚生成的wood.csv，长这样：



二爬取多页

别开心的太早，你还仅仅是爬了一页数据，人家复制粘贴都比你快。咱们的志向可不在这，在诗和远方，哦不，是秒速爬海量数据。

那么，怎么才能爬取多页数据呢？没错，for循环。

我们再回过头来分析下url：

http://yz.yuzhuprice.com:8003/findPriceByName.jspx?page.curPage=1&priceName=%E7%BA%A2%E6%9C%A8%E7%B1%BB

我们把里面的page.curPage改成2试试，如下：



你也许发现玄机，只要改变page.curPage就可以实现翻页。OK，那我们直接在url前面加个循环就好啦。format(x)是一种格式化字符串的函数，可以接受不限个数的参数。

for x in range(1,3):  
    url = 'http://yz.yuzhuprice.com:8003/findPriceByName.jspx?page.curPage={}&priceName=%E7%BA%A2%E6%9C%A8%E7%B1%BB'.format(x)

至此，你只要改变range想爬多少页就爬多少页，开不开心？意不意外？

三完善爬虫

如果仅仅按照以上代码爬虫，很有可能爬了十几页程序就崩了。我就多次遇到过中途报错，导致爬虫失败的情况。好不容易写出的爬虫，怎么说崩就崩呢。

报错原因就很多了，玩爬虫的都知道，调试bug是很麻烦的，需要不断试错。这个爬虫的主要bug是TimeoutError。因此，我们需要进一步完善代码。

首先，要将以上代码封装成函数，因为不用函数有以下缺点：

1、复杂度增大

2、组织结构不够清晰

3、可读性差

4、代码冗余

5、可扩展性差

其次，在可能出现报错的地方都加上异常处理。即try......except。

完善之后，截取部分，如下图。限于篇幅，我就不贴所有代码了，需要完整代码的小伙伴关注公众号，后台回复wood即可免费获取啦。



结 语

自此，红木数据爬虫代码写完啦，数据爬取下来后，就可以进行可视化的分析了，比如可以看下每年不同市场红木的价格走势，同一市场不同红木的价格走势，或者还可以建立起红木的价格指数。可视化的内容后续我会重点讲解，感兴趣的朋友可以留意下哦~

当然，此爬虫也还有很大的完善空间，比如加入多线程、scrapy框架爬取速度会更快。另外，引入随机的headers和代理ip，可以很好的规避一些反爬，这些在很多相对复杂点的爬虫里是必须引入的内容。

到这，本文终于结束啦，然鹅，由于红木研究还需要其他相关数据。作为金融狗，自然而然想到打开wind康一康，如下：



看完之后，我的状态是这样的：



*参考链接：*

*Xpath:https://www.w3school.com.cn/xpath/index.asp;*

*Functions:https://www.runoob.com/python/python-functions.html;*

*Exception:https://www.runoob.com/python/python-exceptions.html*

