



**Python 爬虫实验**

**姓名： 江姝潼**

**班级： 2019211314**

**学号： 2019211653**

**学院： 计算机学院**

**专业： 网络工程**

目录

[一、 实验要求 4](#_Toc90031185)

[二、 实验准备 4](#_Toc90031186)

[1、 安装scrapy库 4](#_Toc90031187)

[2、 项目构建 5](#_Toc90031188)

[3、 Scrapy库执行流程 5](#_Toc90031189)

[三、 爬取链家 6](#_Toc90031190)

[1、 Item.py 6](#_Toc90031191)

[2、 spider.py 6](#_Toc90031192)

[3、 pipelines.py 9](#_Toc90031193)

[4、 settings.py 10](#_Toc90031194)

[5、运行结果 10](#_Toc90031195)

[四、 爬取学堂在线 15](#_Toc90031196)

[1、 实验准备 15](#_Toc90031197)

[2、 Item.py 18](#_Toc90031198)

[3、 spider.py 18](#_Toc90031199)

[4、pipelines.py 21](#_Toc90031200)

[5、settings.py 22](#_Toc90031201)

[6、运行结果 22](#_Toc90031202)

[五、 实验感想 23](#_Toc90031203)

[六、 附录：代码 24](#_Toc90031204)

[1、爬取链家代码 24](#_Toc90031205)

[2、爬取学堂在线代码 26](#_Toc90031206)

# 实验要求

1.爬取北京链家官网二手房数据

https://bj.lianjia.com/ershoufang/

要求爬取东城、西城、海淀、朝阳四个城区的数据（每个区爬取5页），将楼盘名称、总 价、平米数、单价，保存到csv文件中

2.爬取学堂在线的合作院校页面内容

https://www.xuetangx.com/university/all

要求将开课院校的学校名称和对应的课程数量，保存到一个json文件中。

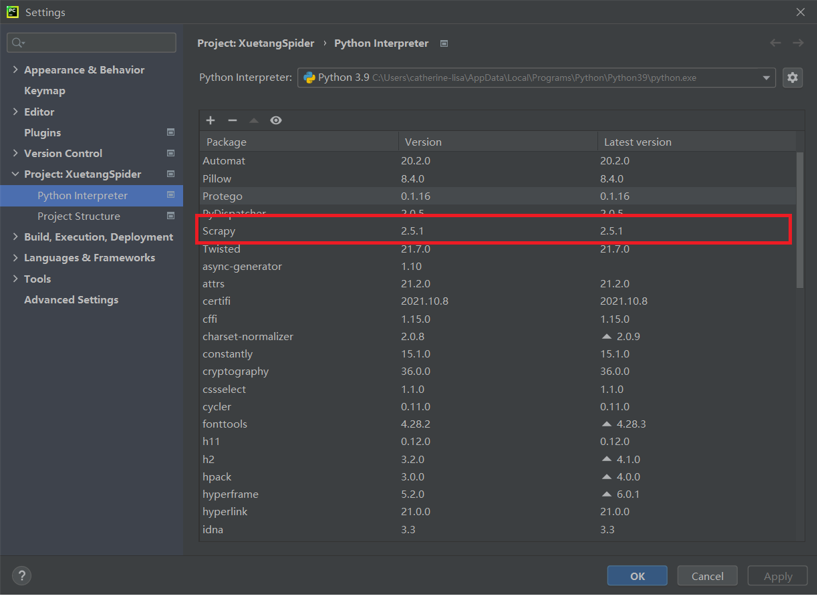
# 实验准备

## 安装scrapy库

在终端中输入以下命令安装Scrapy库：



或者可以在Pycharm的Setting中查看已安装库，可以看到，Scrapy库已成功安装：

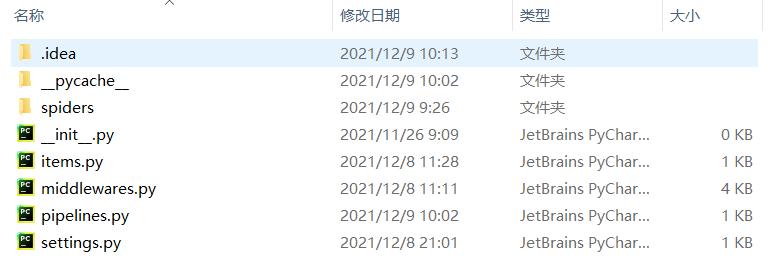


## 项目构建

在项目目录下输入Scrapy命令，初始化一个项目：



生成项目目录如下：



目录下各项目作用如下：

Spider 项目文件夹

├── Spider 用来装载项目文件的目录

│ ├── items.py 定义要抓取的数据结构

│ ├── middlewares.py 中间件，用来设置一些处理规则

│ ├── pipelines.py 管道文件，处理抓取的数据

│ ├── settings.py 全局配置文件

│ └── spiders 用来装载爬虫文件的目录

└── scrapy.cfg # 项目基本配置文件

## Scrapy库执行流程

第一步：由“引擎”向爬虫文件索要第一个待爬取的 URL，并将其交给调度器加入 URL 队列当中。

第二步：调度器处理完请求后， 将第一个 URL 出队列返回给引擎；引擎经由下载器中间件将该 URL 交给下载器去下载 response 对象。

第三步：下载器得到响应对象后，将响应结果交给引擎，引擎收到后，经由蜘蛛中间件将响应结果交给爬虫文件。

第四步：爬虫文件对响应结果进行处理、分析，并提取出所需要的数据。

第五步：最后，提取的数据会交给管道文件去存数据库，同时将需要继续跟进的二级页面 URL 交给调度器去入队列。

上述操作一直循环到没有要爬取的 URL 为止。

# 爬取链家

## 1、 Item.py

首先在Item.py中定义一个LianjiaItem类，用来存储爬取的数据，根据需求定义了楼盘名称、总价、平米数、平方单价、所在地区等几个变量。

import scrapy  
  
class LianjiaItem(scrapy.Item):  
 name = scrapy.Field() #楼盘名称  
 price = scrapy.Field() #总价  
 area = scrapy.Field() #平米数  
 unit\_price = scrapy.Field() #单价  
 place = scrapy.Field() #地点  
 pass

## 2、 spider.py

下面编写爬虫过程的主要代码：

首先定义爬虫得三个强制属性：

name = "" ：这个爬虫的识别名称，必须唯一。

allow\_domains = [] 是搜索的域名范围，也就是爬虫的约束区域，规定爬虫只爬取这个域名下的网页，本题中为链家域名“bj.lianjia.com”。

start\_urls = () ：为要爬取的URL列表。爬虫从这里开始抓取数据，这里定义了包括东城、西城、朝阳、海淀这四个区域的房源的网页。

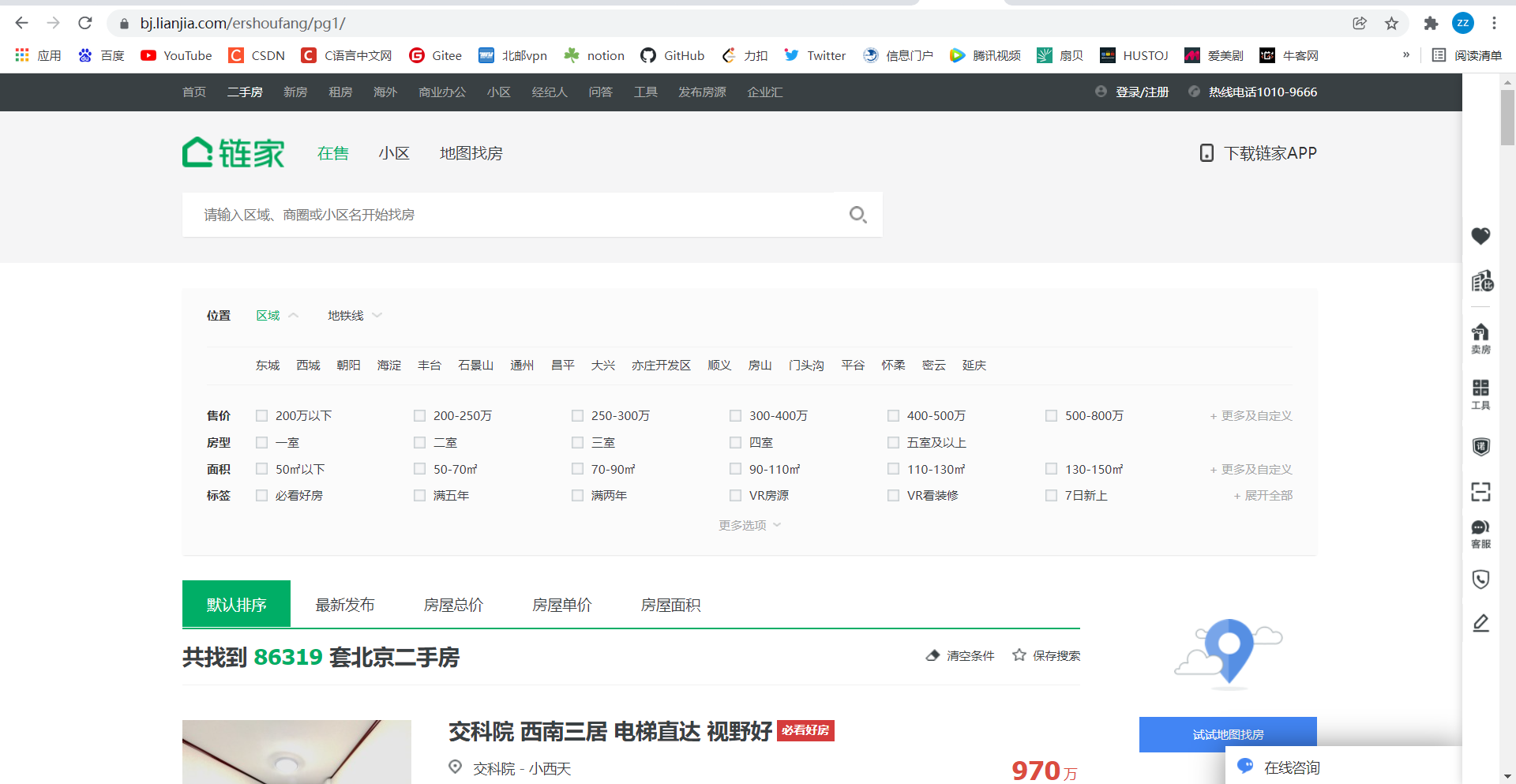
class LianjiaSpider(scrapy.Spider):  
 name = 'lianjia'  
 allowed\_domains = ['bj.lianjia.com']  
 start\_urls = ['https://bj.lianjia.com/ershoufang/dongcheng/',  
 'https://bj.lianjia.com/ershoufang/xicheng/',  
 'https://bj.lianjia.com/ershoufang/chaoyang/',  
 'https://bj.lianjia.com/ershoufang/haidian/']

确定好参数后，下面来编写相关的操作函数：

由于题目要求需要爬取5页的数据，选择下一页，可以看到网址为：



把后缀改为pg1，可以看到又返回了第一页：

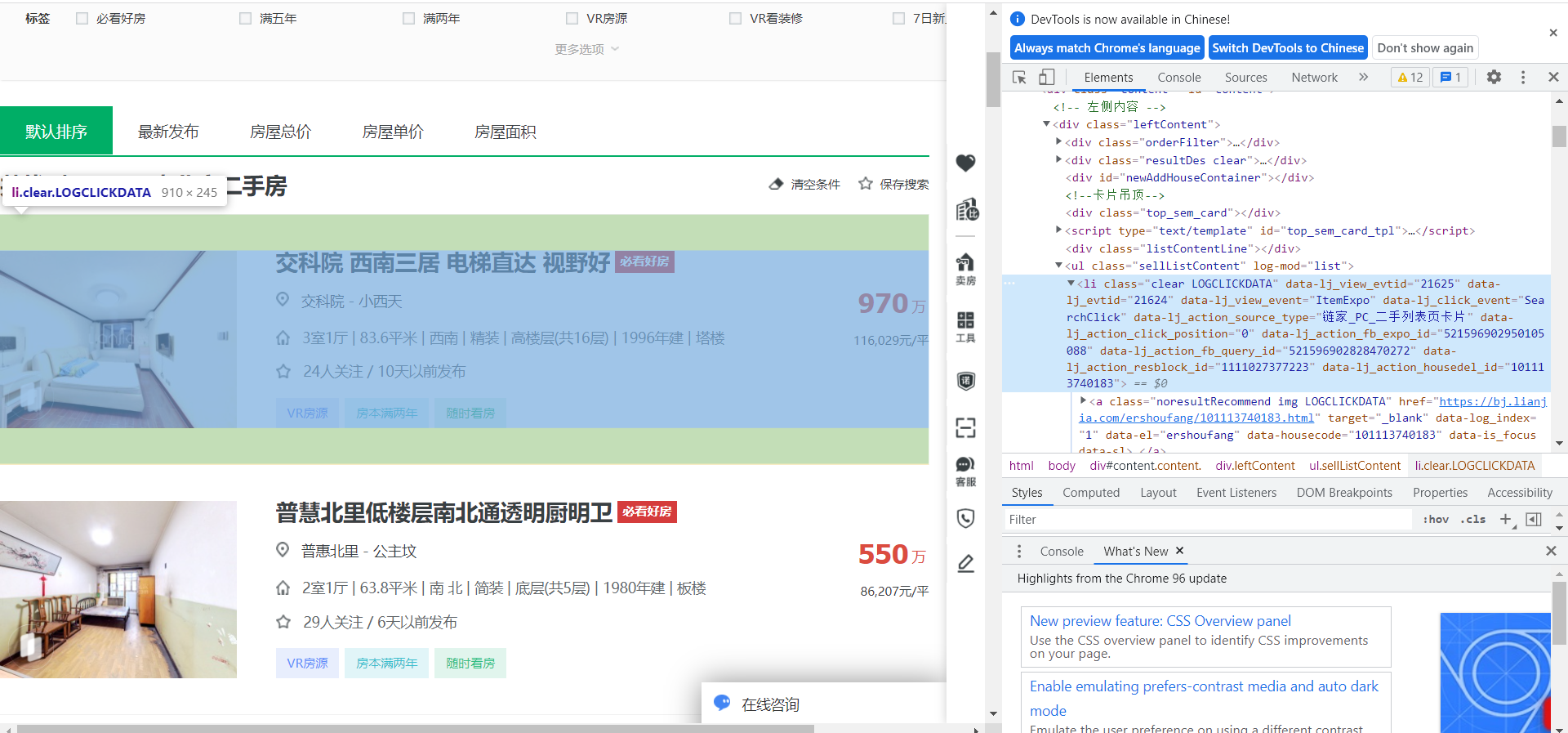


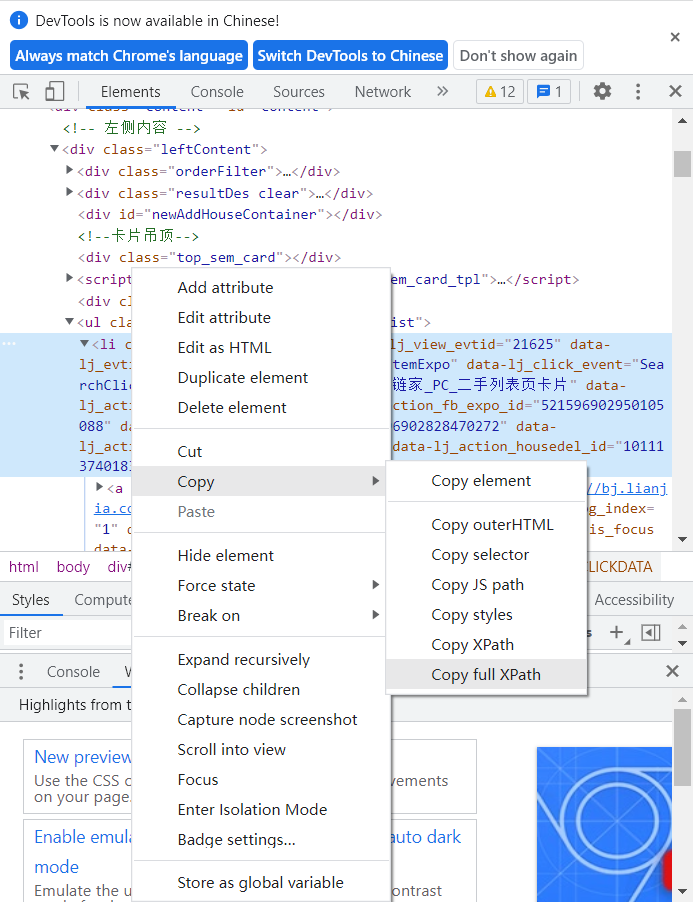
故可以通过在每个区域后面添加pg1,pg2,…,pg5，分别爬取5页的数据。此处编写了一个start\_requests函数代替原本框架的start\_requests函数，并在该函数中给start\_urls中每一个url分别加上pg1,pg2,…,pg5，并返回保存给start\_urls，并对每一个url提交一个Request爬虫。

def start\_requests(self):  
 basic\_urls = self.start\_urls  
 self.start\_urls = []  
 for urls in basic\_urls:  
 for i in range(1, 6):  
 new\_url = urls + "pg" + str(i) + '/'  
 self.start\_urls.append(new\_url)  
 for url in self.start\_urls:  
 yield Request(url, dont\_filter=True)

下面编写对爬取的response进行信息获取操作，即编写parse函数。

先在初始网页中找到要爬取的数据的位置，并对应地复制它的xpath：





以同样的方法找到楼盘名称、总价、平米数、平方单价所在的位置，并把这些信息保存在之前定义的LianjiaItem类中，下一步提交给pipeline处理。

def parse(self, response):  
 item = LianjiaItem()  
 for each in response.xpath("/html/body/div[4]/div[1]/ul/li/div[1]"):  
 item['name'] = each.xpath("div[2]/div/a[1]/text()").get().strip()  
 item['price'] = each.xpath("div[6]/div[1]/span/text()").get().strip() + each.xpath("div[6]/div[1]/i[2]/text()").get().strip()  
 item['area'] = each.xpath("div[3]/div/text()").get().split('|')[1].strip()  
 item['unit\_price'] = each.xpath("div[6]/div[2]/span/text()").get().strip()  
 item['place'] = response.url.split('/')[4]  
 yield (item)

## 3、 pipelines.py

在pipelines.py文件中定义对数据的处理方法，此处定义open\_spider、process\_item和close\_spider三个函数。

在本实验中，需要将爬取的数据存在csv文件中，故在open\_spider函数中打开创建newdata.csv文件，如果打开失败则报错。在该函数中还定义了一个字典dictionary，用来分别存储四个区域爬取的数据。

在process\_item函数中，把每次获取的数据item按照字典分别放在对应的区中，全部处理完后至close\_spider函数中统一输出，并关闭文件。这样处理的原因是，一开始爬取一行输出一行时，发现输出的数据并不按照爬取的数据排列，出现了各个区之间的数据交错排列的情况，结果不直观，用存在字典里这种方式，对数据做了初步的处理，让结果更为清晰直观。

Pipelines.py的具体代码如下：

import csv  
  
class LianjiaPipeline:  
 def open\_spider(self, spider):  
 try:  
 self.file = open('newdata.csv', 'w', encoding='utf\_8\_sig')  
 self.result = csv.writer(self.file)  
 self.result.writerow(['area', 'name', 'price', 'unit\_price', 'place'])  
 self.dictionary = {'dongcheng':[], 'haidian':[], 'xicheng':[], 'chaoyang':[]}  
 print("open file")  
 except Exception as err:  
 print(err)  
 def process\_item(self, item, spider):  
 dict\_item = dict(item)  
 place = dict\_item['place']  
 self.dictionary[place].append(dict\_item)  
 # self.result.writerow(dict\_item.values())  
 return item  
 def close\_spider(self, spider):  
 for i in self.dictionary:  
 for j in self.dictionary[i]:  
 self.result.writerow(j.values())  
 self.file.close()

## settings.py

以上内容基本完成了数据的爬取，故可以不再设置中间件，仅在settings.py文件中启用上面编写的LianjiaPipeline：

BOT\_NAME = 'Spider'  
  
SPIDER\_MODULES = ['Spider.spiders']  
NEWSPIDER\_MODULE = 'Spider.spiders'  
  
ITEM\_PIPELINES = {  
 'Spider.pipelines.LianjiaPipeline': 300,  
}

## 5、运行结果

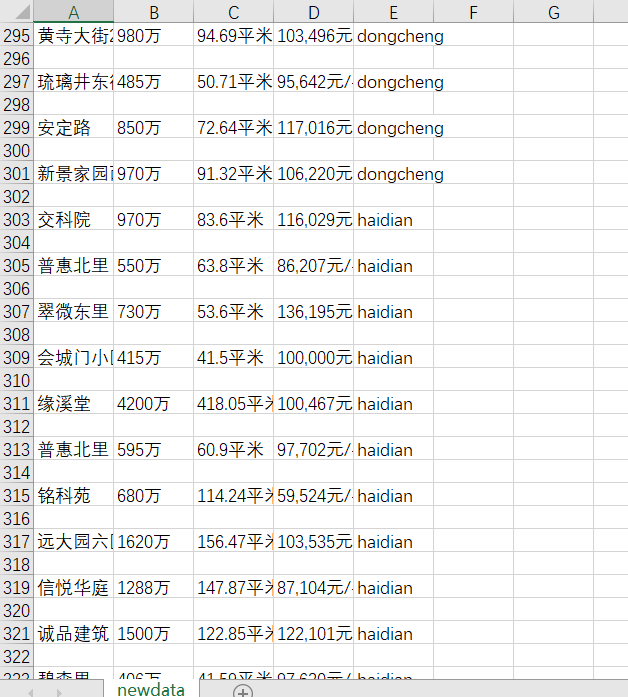
在项目目录下输入scrapy crawl lianjia，运行爬虫。

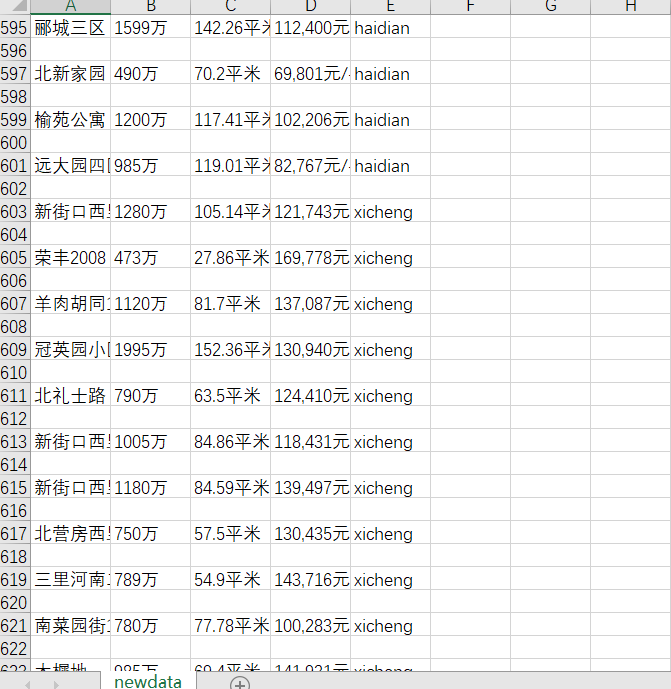
运行结束后，增加了一个newdata.csv文件，为爬取的数据。

爬取的前50条记录如下：

area,name,price,unit\_price,place  
  
广渠家园,695万,62.34平米,"111,486元/平",dongcheng  
  
本家润园一期,1850万,154.09平米,"120,060元/平",dongcheng  
  
东四北大街,675万,61.7平米,"109,401元/平",dongcheng  
  
豆瓣胡同,935万,71.44平米,"130,880元/平",dongcheng  
  
海运仓小区,650万,47.87平米,"135,785元/平",dongcheng  
  
国瑞城中区,633万,58.67平米,"107,892元/平",dongcheng  
  
西营房,496万,36.31平米,"136,602元/平",dongcheng  
  
小黄庄一区,732万,68.39平米,"107,034元/平",dongcheng  
  
西革新里110号院,376万,44.44平米,"84,609元/平",dongcheng  
  
建予园,758万,105.73平米,"71,693元/平",dongcheng  
  
金鱼池西区,670万,61.48平米,"108,979元/平",dongcheng  
  
安馨园,2750万,311.87平米,"88,178元/平",dongcheng  
  
新景家园东区,880万,79.93平米,"110,097元/平",dongcheng  
  
安外花园,679万,60.76平米,"111,752元/平",dongcheng  
  
景泰西里西区,705万,88.23平米,"79,905元/平",dongcheng  
  
东花市北里中区,1100万,139.04平米,"79,114元/平",dongcheng  
  
中海紫御公馆,1028万,90.08平米,"114,121元/平",dongcheng  
  
东花市北里东区,870万,85.43平米,"101,838元/平",dongcheng  
  
美术馆后街,685万,60.3平米,"113,599元/平",dongcheng  
  
朝阳门南小街,930万,80.32平米,"115,787元/平",dongcheng  
  
新景家园西区,688万,58.45平米,"117,708元/平",dongcheng  
  
和平里中街3号院,1130万,95.27平米,"118,611元/平",dongcheng  
  
北官厅胡同甲二号院,710万,57.12平米,"124,300元/平",dongcheng  
  
沙子口路70号,331万,38.28平米,"86,469元/平",dongcheng  
  
交道口北三条,660万,56.15平米,"117,543元/平",dongcheng  
  
领行国际,536万,58.06平米,"92,319元/平",dongcheng  
  
富贵园三区,728万,76.46平米,"95,214元/平",dongcheng  
  
忠实里,720万,82.2平米,"87,592元/平",dongcheng  
  
龙潭北里,630万,65.37平米,"96,375元/平",dongcheng  
  
天坛东里中区,430万,45.59平米,"94,319元/平",dongcheng  
  
中海紫御公馆,1200万,91.44平米,"131,234元/平",dongcheng  
  
阳光都市,2200万,240.55平米,"91,458元/平",dongcheng  
  
国瑞城中区,1699万,157.42平米,"107,928元/平",dongcheng  
  
东花市北里东区,890万,75.16平米,"118,415元/平",dongcheng  
  
板厂南里,590万,50.04平米,"117,906元/平",dongcheng  
  
海运仓小区,1070万,88.93平米,"120,320元/平",dongcheng  
  
幸福北里,752万,70.46平米,"106,728元/平",dongcheng  
  
保利蔷薇,1130万,92.34平米,"122,374元/平",dongcheng  
  
潘家坡胡同4号院,725万,66.45平米,"109,105元/平",dongcheng  
  
小黄庄一区,765万,72.44平米,"105,605元/平",dongcheng  
  
竹杆胡同,810万,68.25平米,"118,682元/平",dongcheng  
  
朝阳门内大街,924万,117平米,"78,975元/平",dongcheng  
  
金桥国际,619万,63.35平米,"97,712元/平",dongcheng  
  
竹杆胡同,799万,71.96平米,"111,034元/平",dongcheng  
  
保利蔷薇,716万,59.25平米,"120,844元/平",dongcheng  
  
百荣嘉园,371万,43.76平米,"84,781元/平",dongcheng  
  
民旺园,808万,70.71平米,"114,270元/平",dongcheng  
  
广渠门内大街,655万,63.93平米,"102,456元/平",dongcheng  
  
沙滩后街55号院,780万,56.91平米,"137,059元/平",dongcheng  
  
新奥洋房,549万,62.42平米,"87,953元/平",dongcheng

因为爬取的数据是按照区域排列的，所以前面的都是东城区的数据。将该数据用excel文件打开，可以看到爬取的数据是依照东城、海淀、西城、朝阳的顺序排列的。





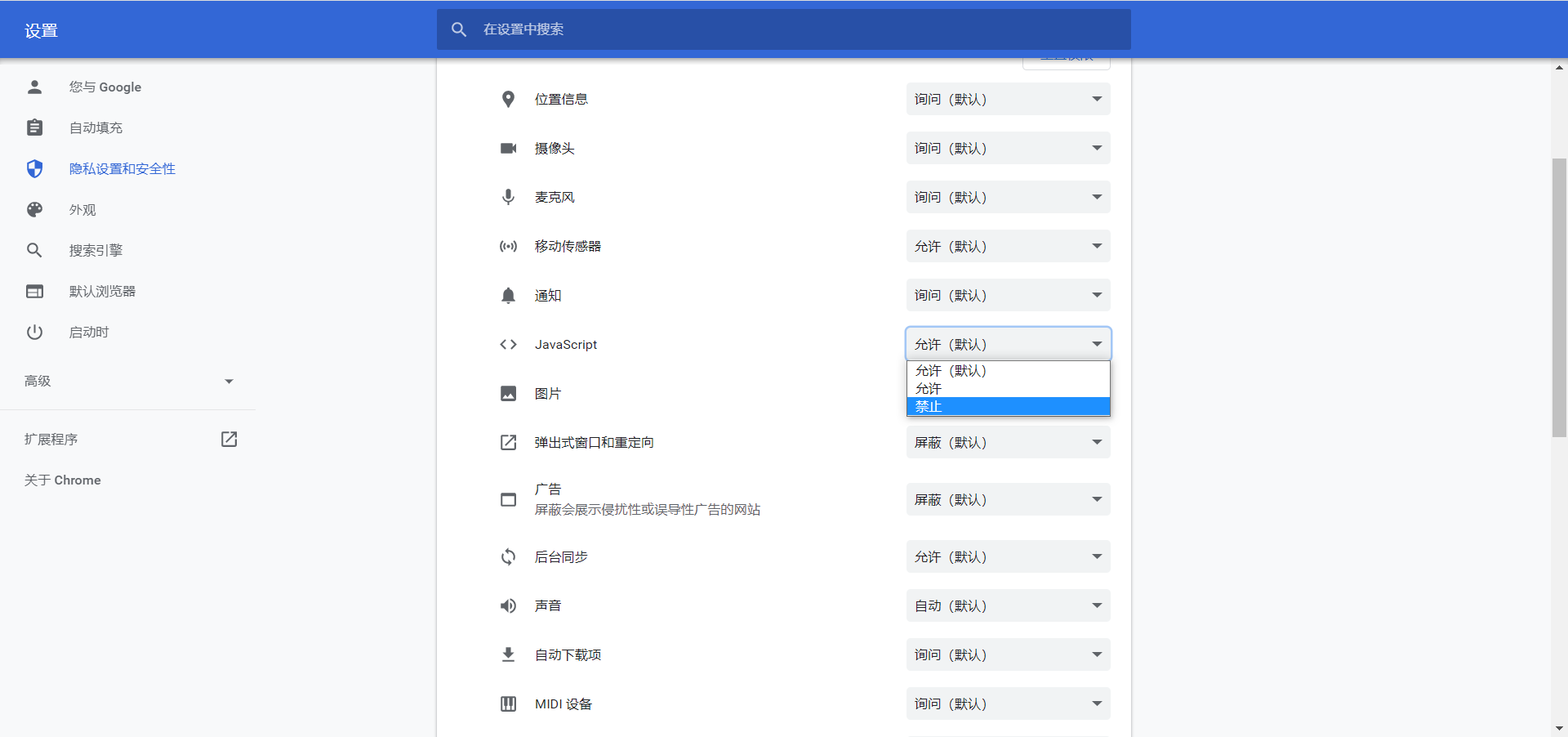


# 爬取学堂在线

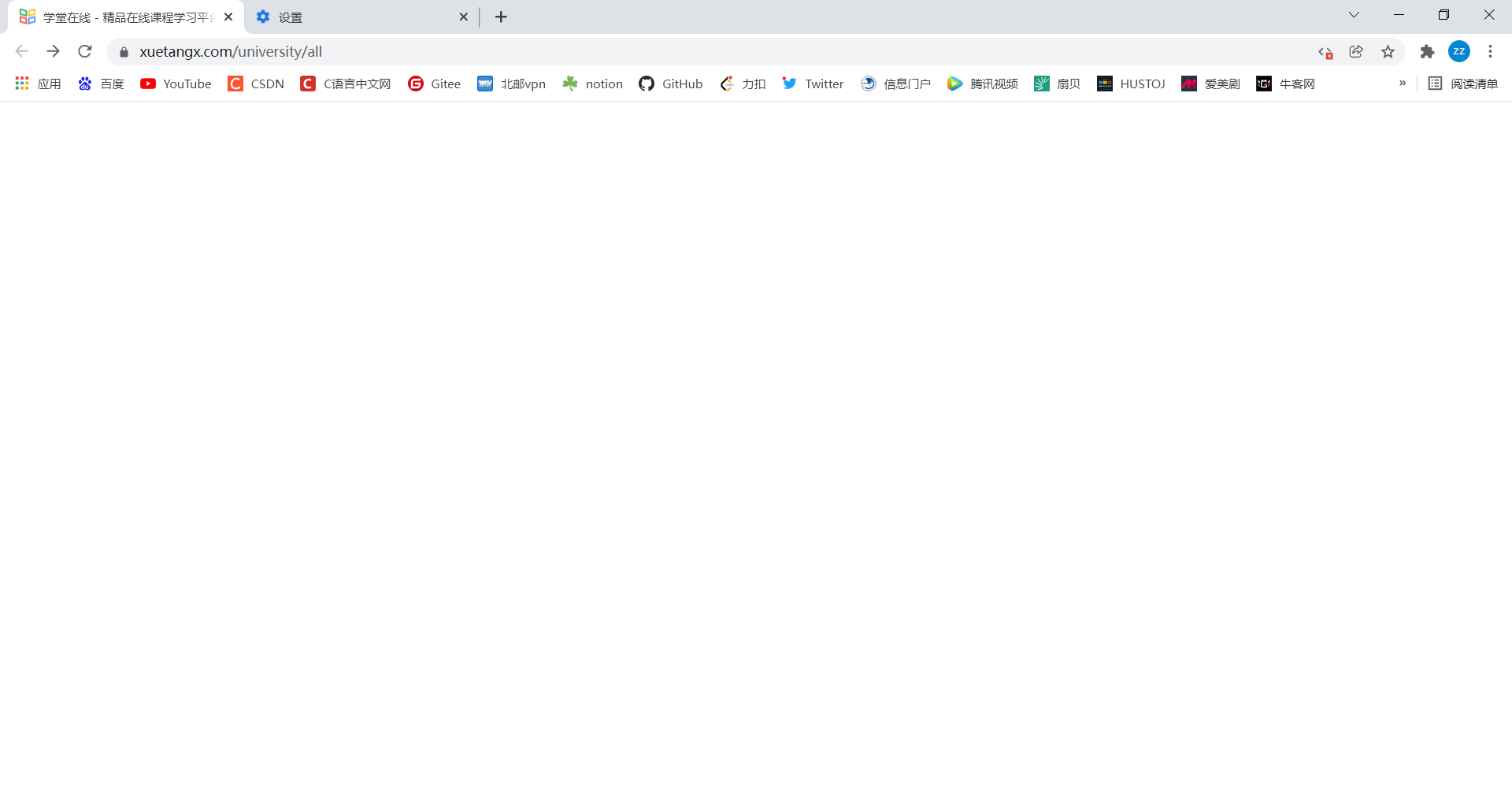
## 1、 实验准备

学堂在线网页和链家不一样的是，它是一个动态网页，但是“动态“具体体现在哪里，一开始我并不是非常理解。但是按照之前的方式爬取，发现爬得的结果是空。

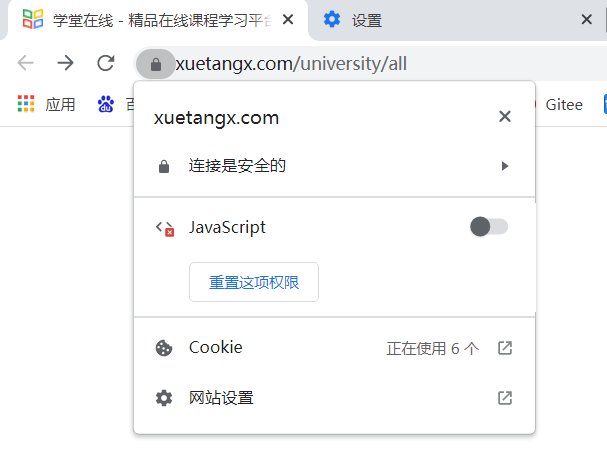
于是我做了以下尝试，我在chrome中关闭了js：



然后再重新加载学堂在线网页，加载如下：

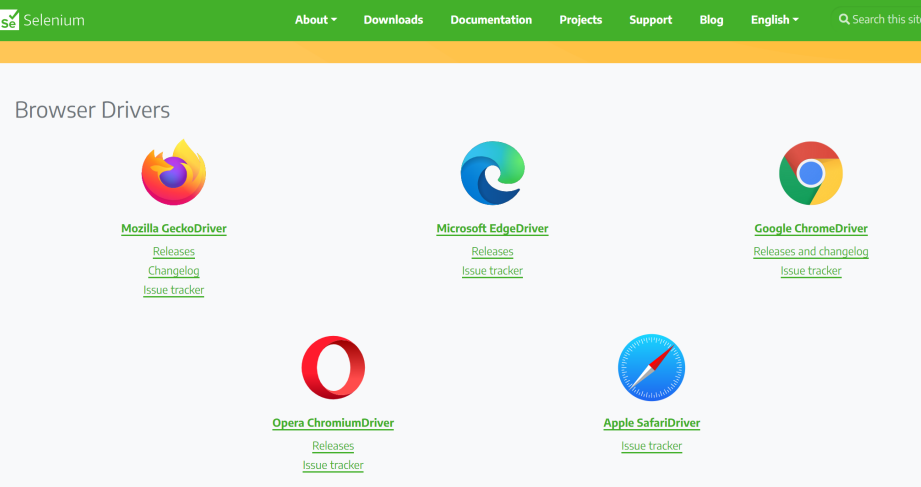


可以看到，整个网页是空的，没有显示任何内容。打开javascript再重新加载网页，又全部恢复正常了。



通过以上操作可以知道，学堂在线网页的主要内容是通过js加载的，而不是在原始的html文件里。需要通过Selenium来模拟操作打开网页。

为了保证Selenium正确运行，需要安装webdriver:

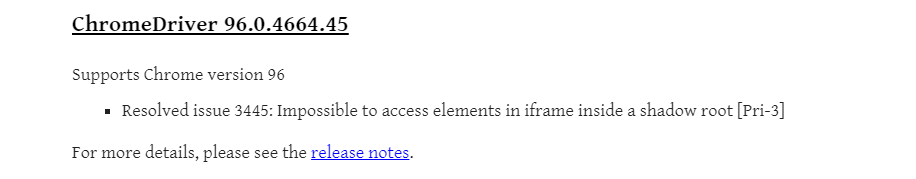


我使用的是Chrome浏览器，故需要安装ChromeDriver。

Webdriver和版本必须一致，所以在安装前先查看我自己的Chrome浏览器版本：



故安装对应的版本：



至此，就可以开始动态网页的爬虫了。

## 2、 Item.py

和爬取链家类似，首先在Item.py中定义一个XuetangSpiderItem类，用来存储爬取的数据，包括学校名、课程数两个变量。

import scrapy  
  
class XuetangspiderItem(scrapy.Item):  
 # define the fields for your item here like:  
 # name = scrapy.Field()  
 school = scrapy.Field()  
 classnum = scrapy.Field()  
 pass

## 3、 spider.py

首先定义爬虫得三个强制属性：

name = "" ：定义爬虫名称为Xuetang。

allow\_domains = [] 是搜索的域名范围，也就是爬虫的约束区域，本题中为学堂域名“www.xuetangx.com”。

start\_urls = () ：为要爬取的URL列表。本题中只有一个“https://www.xuetangx.com/university/all”。

class XuetangSpider(scrapy.Spider):

name = 'xuetang'

allowed\_domains = ['www.xuetangx.com']

start\_urls = ['https://www.xuetangx.com/university/all']

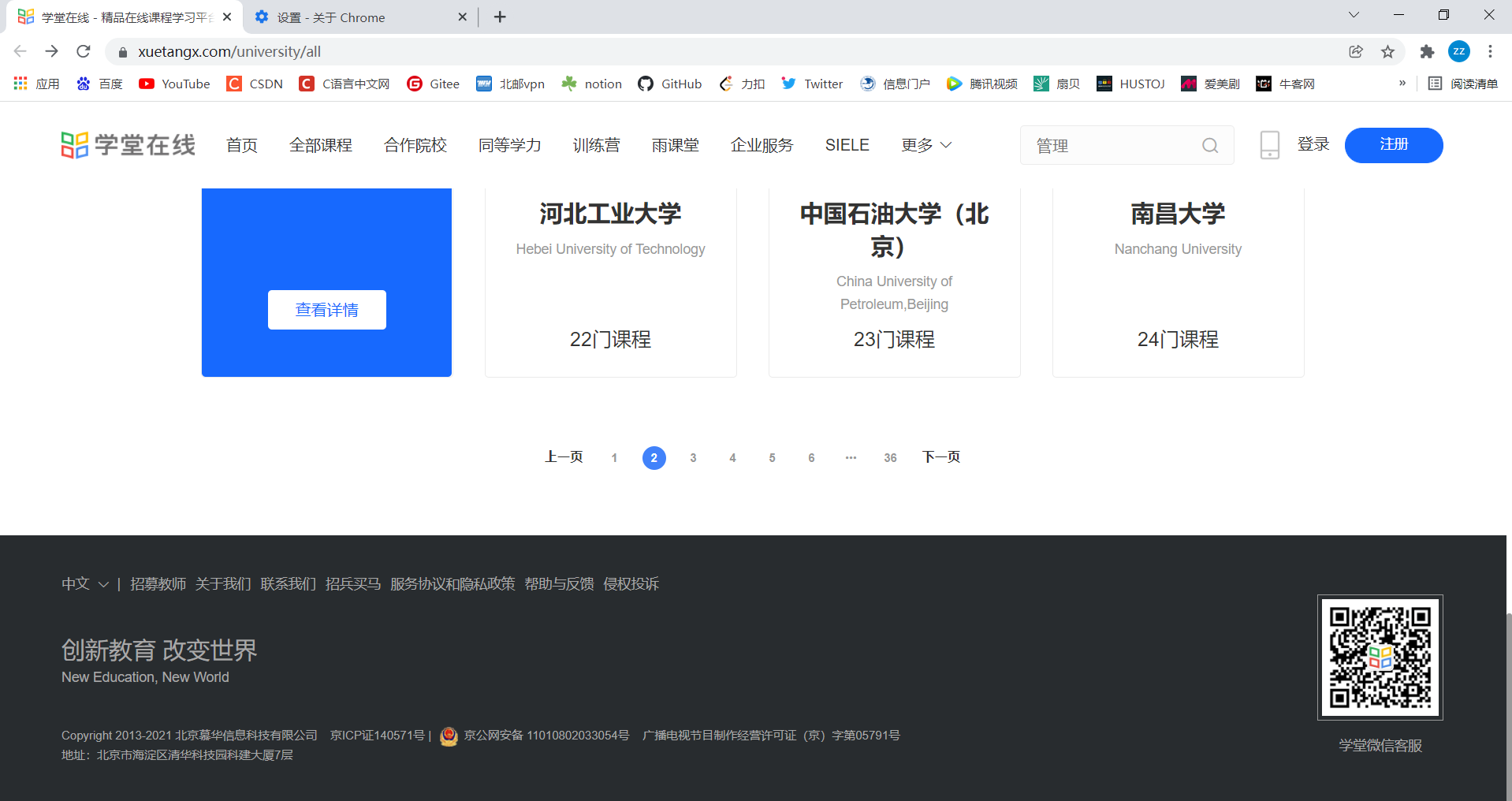
确定好参数后，下面来编写操作函数parse：

这里定义了option来设置打开浏览器的选项，为其增加了--headless 属性让在爬取时浏览器不弹出，然后用 driver 请求 request.url，获取网页信息。

def parse(self, response):  
 item = XuetangspiderItem()  
 option = webdriver.ChromeOptions()  
 option.add\_argument('--headless')  
 driver = webdriver.Chrome(chrome\_options=option)  
 driver.get(response.url)  
 driver.implicitly\_wait(5)

在爬取网页数据的时候可以发现，一页的数据量时非常少的，现在想实现爬取所有页的数据。

打开第二页，发现网址的url并没有发生改变，用上一题链家修改url的方式失效了：



遇到这种情况时，可以使用selenium的操作浏览器的功能，通过搜索classname找到翻下一页的按钮并点击，一共有36页，故要模拟35次点击操作。

for i in range(1, 36):  
 response = HtmlResponse(url=driver.current\_url, body=driver.page\_source, encoding='utf-8')  
 for each in response.xpath("/html/body/div[1]/div/div[2]/div[1]/div[2]/\*"):  
 item['school'] = each.xpath("./div[1]/p[1]/text()").get().strip()  
 item['classnum'] = each.xpath("./p/text()").get().strip()  
 yield (item)  
 but = driver.find\_element(By.CLASS\_NAME, "btn-next")  
 but.click()  
 time.sleep(2)

在实际中运行这个程序中，发现只能爬取两页的数据，然后就报错显示找不到要点击的按钮。为了找到原因，注释掉—headless属性，操作浏览器打开，可以看到，爬取第二页的时候还没有完全加载出来，就又执行了点击下一页按钮的操作，操作失败退出了浏览器。为了解决该问题，设置了一个time.sleep，让每次切换页面时都等待两秒，给页面有充足的加载时间。再次运行，发现爬取数据变得正常了，但对应的爬取速度由于等待也变慢了。

完整spider.py代码如下：

import scrapy  
from XuetangSpider.items import XuetangspiderItem  
from scrapy.http import HtmlResponse  
from selenium import webdriver  
from selenium.webdriver.common.by import By  
from scrapy.http import HtmlResponse  
import time  
  
class XuetangSpider(scrapy.Spider):  
 name = 'xuetang'  
 allowed\_domains = ['www.xuetangx.com']  
 start\_urls = ['https://www.xuetangx.com/university/all']  
  
 def parse(self, response):  
 item = XuetangspiderItem()  
 option = webdriver.ChromeOptions()  
 option.add\_argument('--headless')  
 driver = webdriver.Chrome(chrome\_options=option)  
 driver.get(response.url)  
 driver.implicitly\_wait(5)  
 for i in range(1, 36):  
 response = HtmlResponse(url=driver.current\_url, body=driver.page\_source, encoding='utf-8')  
 for each in response.xpath("/html/body/div[1]/div/div[2]/div[1]/div[2]/\*"):  
 item['school'] = each.xpath("./div[1]/p[1]/text()").get().strip()  
 item['classnum'] = each.xpath("./p/text()").get().strip()  
 yield (item)  
 but = driver.find\_element(By.CLASS\_NAME, "btn-next")  
 but.click()  
 time.sleep(2)

## 4、pipelines.py

在pipelines.py文件中定义对数据的处理方法，此处定义open\_spider、process\_item和close\_spider三个函数。

在本实验中，需要将爬取的数据存在json文件中，故在open\_spider函数中打开创建data.json文件，如果打开失败则报错。在process\_item函数中，使用json.dumps函数将Python对象转化为JSON字符串，对每个item输出一行，全部输出完毕后关闭文件。

完整代码如下：

import json  
  
class XuetangspiderPipeline:  
 def open\_spider(self, spider):  
 try:  
 self.file = open('data.json', 'w', encoding='utf-8')  
 print("open file")  
 except Exception as err:  
 print(err)  
  
 def process\_item(self, item, spider):  
 dict\_item = dict(item)  
 json\_str = json.dumps(dict\_item, ensure\_ascii=False) + "\n"  
 self.file.write(json\_str)  
 return item  
  
 def close\_spider(self, spider):  
 self.file.close()

## 5、settings.py

在 settings 中启用 XuetangspiderPipeline：

BOT\_NAME = 'XuetangSpider'  
  
SPIDER\_MODULES = ['XuetangSpider.spiders']  
NEWSPIDER\_MODULE = 'XuetangSpider.spiders'  
  
ITEM\_PIPELINES = {  
 'XuetangSpider.pipelines.XuetangspiderPipeline' : 300  
}  
  
  
# Obey robots.txt rules  
ROBOTSTXT\_OBEY = True

## 6、运行结果

输入“scrapy crawl xuetang”运行爬虫，待运行完毕后，出现了一个data.json文件，打开可得运行的结果如下：

{"school": "清华大学", "classnum": "425门课程"}  
{"school": "北京大学", "classnum": "24门课程"}  
{"school": "北京师范大学", "classnum": "62门课程"}  
{"school": "国防科技大学", "classnum": "15门课程"}  
{"school": "西安交通大学", "classnum": "102门课程"}  
{"school": "哈尔滨工业大学", "classnum": "38门课程"}  
{"school": "华南理工大学", "classnum": "48门课程"}  
{"school": "南开大学", "classnum": "41门课程"}  
{"school": "复旦大学", "classnum": "9门课程"}  
{"school": "南京大学", "classnum": "9门课程"}  
{"school": "中国科学技术大学", "classnum": "7门课程"}  
{"school": "圣彼得堡国立大学", "classnum": "15门课程"}  
{"school": "重庆大学", "classnum": "47门课程"}  
{"school": "暨南大学", "classnum": "75门课程"}  
{"school": "东北大学", "classnum": "34门课程"}  
{"school": "中南大学", "classnum": "41门课程"}  
{"school": "中国农业大学", "classnum": "36门课程"}  
{"school": "云南大学", "classnum": "39门课程"}  
{"school": "山东大学", "classnum": "69门课程"}  
{"school": "西北工业大学", "classnum": "28门课程"}  
{"school": "四川大学", "classnum": "34门课程"}  
{"school": "大连理工大学", "classnum": "14门课程"}  
{"school": "SDGAcademy", "classnum": "13门课程"}  
{"school": "湖南大学", "classnum": "17门课程"}  
{"school": "天津大学", "classnum": "16门课程"}  
{"school": "武汉大学", "classnum": "9门课程"}  
{"school": "上海交通大学", "classnum": "4门课程"}  
{"school": "浙江大学", "classnum": "5门课程"}  
{"school": "北京体育大学", "classnum": "29门课程"}  
{"school": "河北工业大学", "classnum": "22门课程"}  
{"school": "中国石油大学（北京）", "classnum": "23门课程"}  
{"school": "南昌大学", "classnum": "24门课程"}  
{"school": "北京理工大学", "classnum": "51门课程"}  
{"school": "中国传媒大学", "classnum": "23门课程"}  
{"school": "北京交通大学", "classnum": "37门课程"}  
{"school": "宁夏大学", "classnum": "14门课程"}  
{"school": "郑州大学", "classnum": "21门课程"}  
{"school": "北京林业大学", "classnum": "11门课程"}  
{"school": "大连海事大学", "classnum": "18门课程"}  
{"school": "中央民族大学", "classnum": "18门课程"}  
{"school": "华北电力大学", "classnum": "33门课程"}  
{"school": "天津医科大学", "classnum": "6门课程"}  
{"school": "武汉理工大学", "classnum": "14门课程"}  
{"school": "中南财经政法大学", "classnum": "8门课程"}  
{"school": "苏州大学", "classnum": "11门课程"}  
{"school": "台湾交通大学", "classnum": "1门课程"}  
{"school": "国际关系学院", "classnum": "6门课程"}  
{"school": "东南大学", "classnum": "5门课程"}  
{"school": "青海大学", "classnum": "7门课程"}  
{"school": "辽宁对外经贸学院", "classnum": "10门课程"}

# 实验感想

本次实验的收获是非常丰富的，也是我的第一次爬虫尝试，看到能在很短的时间爬取那么多数据，不禁感叹python功能的强大。两次爬虫的内容不尽相同，动态和静态网页的差距造成了需要用不同的方式去处理。

本次实验使用的是scrapy框架，这要求我只需要在已有的框架自定义爬取的数据就可以了，操作非常地简单方便。这次实验同时也让我了解了scrapy框架的运行机制，并用了selenium库模拟人的操作进行处理。在爬虫处理的过程中，也让我对网页的运行流程有了更深刻的认识，为什么动态网页用直接爬取html的方式无法实现、网页加载流程是怎样的，这些知识渗透在爬虫的过程中，也对为什么要执行那些操作给出了解释。

这次实验使用了两种方式csv和json进行输出。两者输出也各自的特点，例如csv是可以用excel打开的，看起来更加简单直观；而json则方便后续的数据导入处理，有很高的应用价值。

# 附录：代码

## 1、爬取链家代码

Items.py

import scrapy  
  
class LianjiaItem(scrapy.Item):  
 name = scrapy.Field() #楼盘名称  
 price = scrapy.Field() #总价  
 area = scrapy.Field() #平米数  
 unit\_price = scrapy.Field() #单价  
 place = scrapy.Field() #地点  
 pass

Spider.py

import scrapy  
from Spider.items import LianjiaItem  
from scrapy.http import Request  
  
class LianjiaSpider(scrapy.Spider):  
 name = 'lianjia'  
 allowed\_domains = ['bj.lianjia.com']  
 start\_urls = ['https://bj.lianjia.com/ershoufang/dongcheng/',  
 'https://bj.lianjia.com/ershoufang/xicheng/',  
 'https://bj.lianjia.com/ershoufang/chaoyang/',  
 'https://bj.lianjia.com/ershoufang/haidian/']  
  
 def start\_requests(self):  
 basic\_urls = self.start\_urls  
 self.start\_urls = []  
 for urls in basic\_urls:  
 for i in range(1, 6):  
 new\_url = urls + "pg" + str(i) + '/'  
 self.start\_urls.append(new\_url)  
 for url in self.start\_urls:  
 yield Request(url, dont\_filter=True)  
  
 def parse(self, response):  
 item = LianjiaItem()  
 for each in response.xpath("/html/body/div[4]/div[1]/ul/li/div[1]"):  
 item['name'] = each.xpath("div[2]/div/a[1]/text()").get().strip()  
 item['price'] = each.xpath("div[6]/div[1]/span/text()").get().strip() + each.xpath("div[6]/div[1]/i[2]/text()").get().strip()  
 item['area'] = each.xpath("div[3]/div/text()").get().split('|')[1].strip()  
 item['unit\_price'] = each.xpath("div[6]/div[2]/span/text()").get().strip()  
 item['place'] = response.url.split('/')[4]  
 yield (item)

pipelines.py

import csv  
  
class LianjiaPipeline:  
 def open\_spider(self, spider):  
 try:  
 self.file = open('newdata.csv', 'w', encoding='utf\_8\_sig')  
 self.result = csv.writer(self.file)  
 self.result.writerow(['area', 'name', 'price', 'unit\_price', 'place'])  
 self.dictionary = {'dongcheng':[], 'haidian':[], 'xicheng':[], 'chaoyang':[]}  
 print("open file")  
 except Exception as err:  
 print(err)  
 def process\_item(self, item, spider):  
 dict\_item = dict(item)  
 place = dict\_item['place']  
 self.dictionary[place].append(dict\_item)  
 # self.result.writerow(dict\_item.values())  
 return item  
 def close\_spider(self, spider):  
 for i in self.dictionary:  
 for j in self.dictionary[i]:  
 self.result.writerow(j.values())  
 self.file.close()

Settings.py

BOT\_NAME = 'Spider'  
  
SPIDER\_MODULES = ['Spider.spiders']  
NEWSPIDER\_MODULE = 'Spider.spiders'  
  
ITEM\_PIPELINES = {  
 'Spider.pipelines.LianjiaPipeline': 300,  
}

## 2、爬取学堂在线代码

Items.py

import scrapy

class XuetangspiderItem(scrapy.Item):

# define the fields for your item here like:

# name = scrapy.Field()

school = scrapy.Field()

classnum = scrapy.Field()

pass

spider.py

import scrapy  
from XuetangSpider.items import XuetangspiderItem  
from scrapy.http import HtmlResponse  
from selenium import webdriver  
from selenium.webdriver.common.by import By  
from scrapy.http import HtmlResponse  
import time  
  
class XuetangSpider(scrapy.Spider):  
 name = 'xuetang'  
 allowed\_domains = ['www.xuetangx.com']  
 start\_urls = ['https://www.xuetangx.com/university/all']  
  
 def parse(self, response):  
 item = XuetangspiderItem()  
 option = webdriver.ChromeOptions()  
 option.add\_argument('--headless')  
 driver = webdriver.Chrome(chrome\_options=option)  
 driver.get(response.url)  
 driver.implicitly\_wait(5)  
 for i in range(1, 36):  
 response = HtmlResponse(url=driver.current\_url, body=driver.page\_source, encoding='utf-8')  
 for each in response.xpath("/html/body/div[1]/div/div[2]/div[1]/div[2]/\*"):  
 item['school'] = each.xpath("./div[1]/p[1]/text()").get().strip()  
 item['classnum'] = each.xpath("./p/text()").get().strip()  
 yield (item)  
 but = driver.find\_element(By.CLASS\_NAME, "btn-next")  
 but.click()  
 time.sleep(2)

pipelines.py

import json  
  
class XuetangspiderPipeline:  
 def open\_spider(self, spider):  
 try:  
 self.file = open('data.json', 'w', encoding='utf-8')  
 print("open file")  
 except Exception as err:  
 print(err)  
  
 def process\_item(self, item, spider):  
 dict\_item = dict(item)  
 json\_str = json.dumps(dict\_item, ensure\_ascii=False) + "\n"  
 self.file.write(json\_str)  
 return item  
  
 def close\_spider(self, spider):  
 self.file.close()

Settings.py

BOT\_NAME = 'XuetangSpider'  
  
SPIDER\_MODULES = ['XuetangSpider.spiders']  
NEWSPIDER\_MODULE = 'XuetangSpider.spiders'  
'''  
DOWNLOADER\_MIDDLEWARES = {  
 'XuetangSpider.middlewares.XuetangspiderDownloaderMiddleware' : 500  
}  
'''  
ITEM\_PIPELINES = {  
 'XuetangSpider.pipelines.XuetangspiderPipeline' : 300  
}