北京郵電大學

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

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

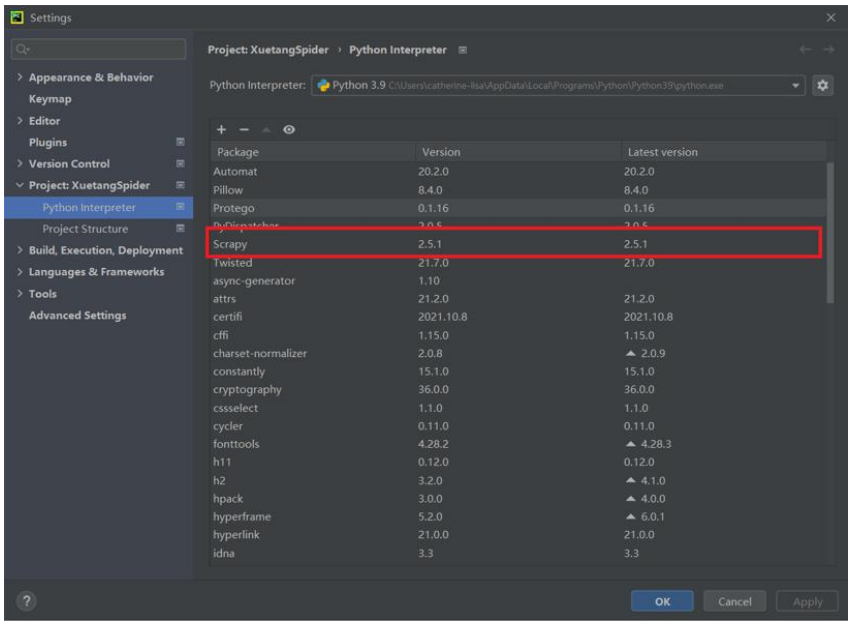
要求将开课院校的学校名称和对应的课程数量，保存到一个 json 文件中。

二、 实验准备

1、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：

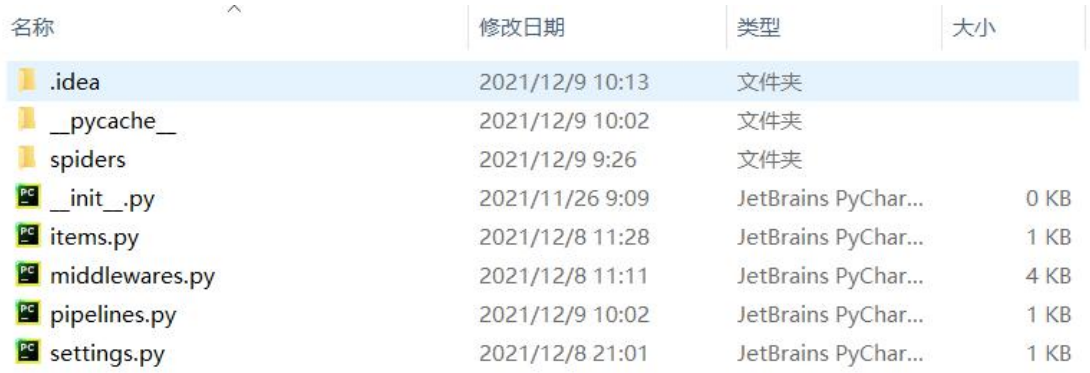


2、项目构建

在项目目录下输入 Scrapy 命令，初始化一个项目：

```
PS D:\PythonCode> scrapy startproject Spider
```

生成项目目录如下：



名称	修改日期	类型	大小
.idea	2021/12/9 10:13	文件夹	
__pycache__	2021/12/9 10:02	文件夹	
spiders	2021/12/9 9:26	文件夹	
__init__.py	2021/11/26 9:09	JetBrains PyChar...	0 KB
items.py	2021/12/8 11:28	JetBrains PyChar...	1 KB
middlewares.py	2021/12/8 11:11	JetBrains PyChar...	4 KB
pipelines.py	2021/12/9 10:02	JetBrains PyChar...	1 KB
settings.py	2021/12/8 21:01	JetBrains PyChar...	1 KB

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
```

3、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。

三、 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。

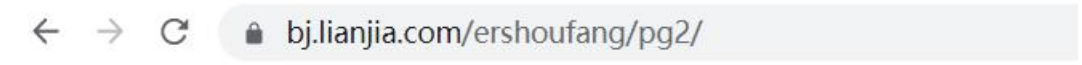
allow_domains = [] 是搜索的域名范围，也就是爬虫的约束区域，规定爬虫只爬取这个域名下的网页，本题中为链家域名“bj.lianjia.com”。

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。

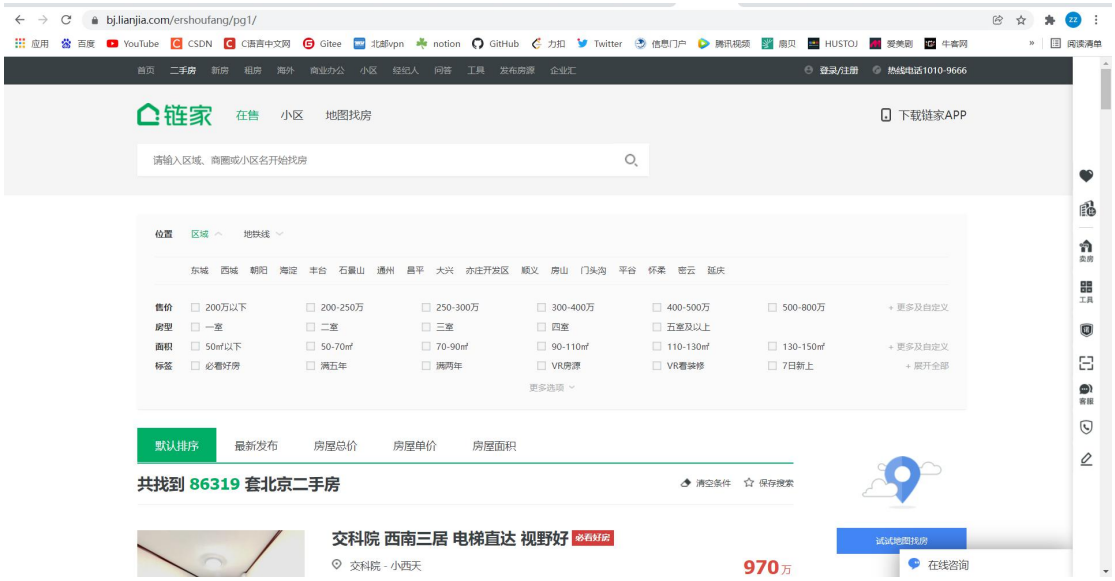
```
class LianjiaSpider(scrapy.Spider):
    name = 'lianjia'
    allowed_domains = ['bj.lianjia.com']
    start_urls = ['https://bj.lianjia.com/ershoufang/dongcheng/',
                  'https://bj.lianjia.com/ershoufang/xicheng/',
                  'https://bj.lianjia.com/ershoufang/chaoyang/',
                  '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：

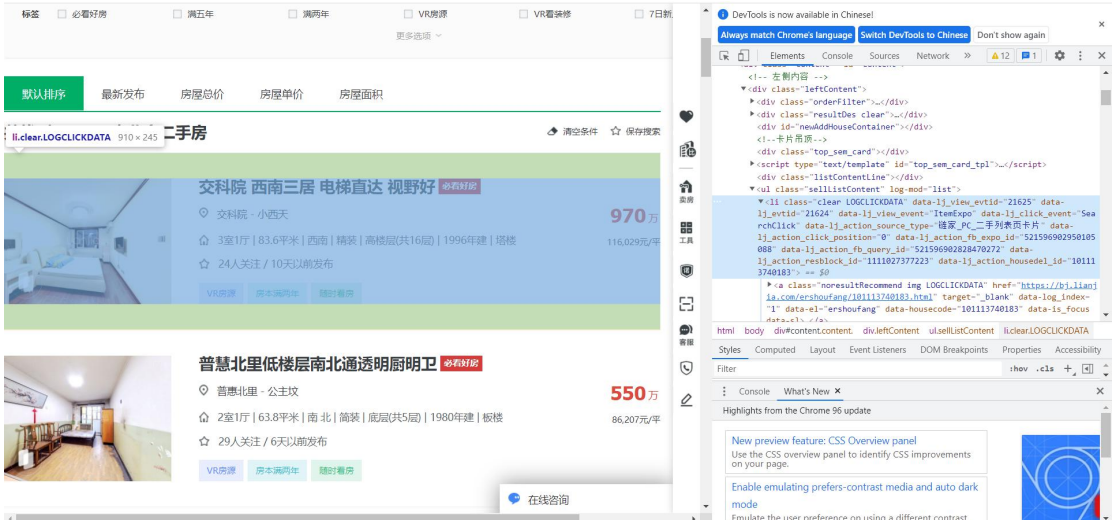


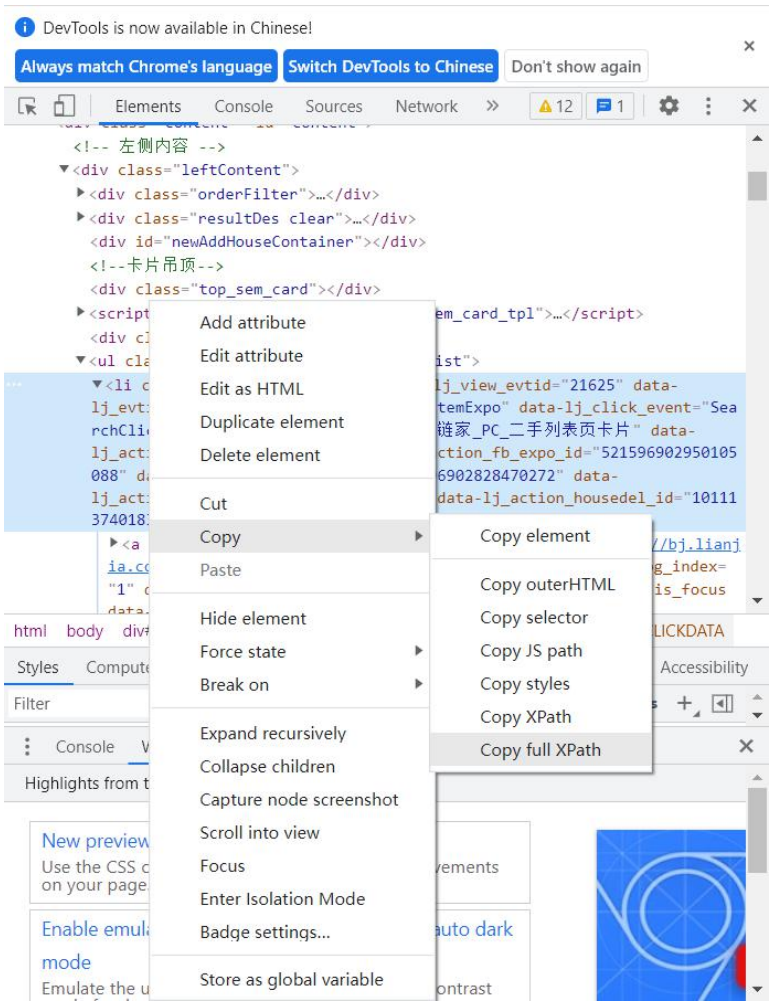
故可以通过在每个区域后面添加 pg1, pg2, …, pg5，分别爬取 5 页的数据。此处编写了一个 start_requests 函数代替原本框架的 start_requests 函数，并在该函数中给 start_urls 中每一个 url 分别加上 pg1, pg2, …, pg5，并返回保存给 start_urls，并对每一个 url 提交一个 Request 爬虫。

```
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()
```

4、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。

	A	B	C	D	E	F	G
295	黄寺大街	980万	94.69平米	103,496元	dongcheng		
296							
297	琉璃井东	485万	50.71平米	95,642元/	dongcheng		
298							
299	安定路	850万	72.64平米	117,016元	dongcheng		
300							
301	新景家园	970万	91.32平米	106,220元	dongcheng		
302							
303	交科院	970万	83.6平米	116,029元	haidian		
304							
305	普惠北里	550万	63.8平米	86,207元/	haidian		
306							
307	翠微东里	730万	53.6平米	136,195元	haidian		
308							
309	会城门小	415万	41.5平米	100,000元	haidian		
310							
311	缘溪堂	4200万	418.05平	100,467元	haidian		
312							
313	普惠北里	595万	60.9平米	97,702元/	haidian		
314							
315	铭科苑	680万	114.24平	59,524元/	haidian		
316							
317	远大园六	1620万	156.47平	103,535元	haidian		
318							
319	信悦华庭	1288万	147.87平	87,104元/	haidian		
320							
321	诚品建筑	1500万	122.85平	122,101元	haidian		
322							
323	碧李里	406万	41.59平米	97,620元/	haidian		

newdata

	A	B	C	D	E	F	G	H
595	郦城三区	1599万	142.26平米	112,400元	haidian			
596								
597	北新家园	490万	70.2平米	69,801元/	haidian			
598								
599	榆苑公寓	1200万	117.41平米	102,206元	haidian			
600								
601	远大园四[	985万	119.01平米	82,767元/	haidian			
602								
603	新街口西!	1280万	105.14平米	121,743元	xicheng			
604								
605	荣丰2008	473万	27.86平米	169,778元	xicheng			
606								
607	羊肉胡同:	1120万	81.7平米	137,087元	xicheng			
608								
609	冠英园小[	1995万	152.36平米	130,940元	xicheng			
610								
611	北礼士路	790万	63.5平米	124,410元	xicheng			
612								
613	新街口西!	1005万	84.86平米	118,431元	xicheng			
614								
615	新街口西!	1180万	84.59平米	139,497元	xicheng			
616								
617	北营房西!	750万	57.5平米	130,435元	xicheng			
618								
619	三里河南:	789万	54.9平米	143,716元	xicheng			
620								
621	南菜园街:	780万	77.78平米	100,283元	xicheng			
622								
623	大堰地	995万	69.4平米	141,921元	xicheng			

newdata

	A	B	C	D	E	F
895	新外大街	935万	66.2平米	141,239元	xicheng	
896						
897	南菜园街	460万	40.37平米	113,946元	xicheng	
898						
899	远见国际	477万	47.71平米	99,980元/	xicheng	
900						
901	团结大院	688万	59.2平米	116,217元	xicheng	
902						
903	康城别墅	1450万	342.72平	42,309元/	chaoyang	
904						
905	望京西园	1095万	148.98平	73,500元/	chaoyang	
906						
907	首开畅心	310万	71.84平米	43,152元/	chaoyang	
908						
909	北辰福第	415万	86.36平米	48,055元/	chaoyang	
910						
911	芍药居北	1166万	130.26平	89,514元/	chaoyang	
912						
913	管庄西里	349万	71.32平米	48,935元/	chaoyang	
914						
915	望京新城	660万	89.48平米	73,760元/	chaoyang	
916						
917	朝阳新城	545万	107.07平	50,902元/	chaoyang	
918						
919	山水蓝维	1130万	150.3平米	75,183元/	chaoyang	
920						
921	和乔丽致	649万	79.18平米	81,966元/	chaoyang	
922						
923	南沙滩小	426万	52.9平米	82,420元/	chaoyang	

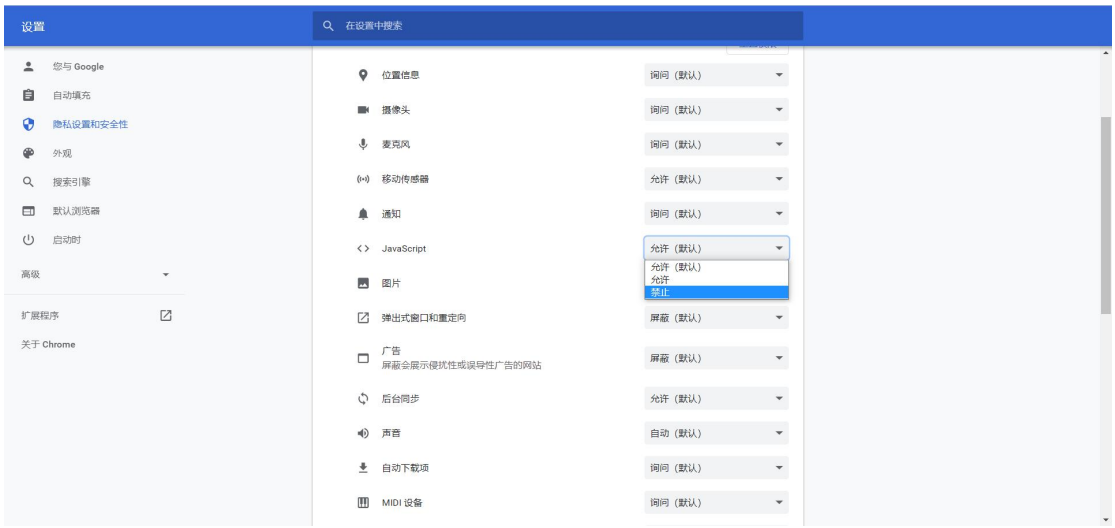
newdata

四、 爬取学堂在线

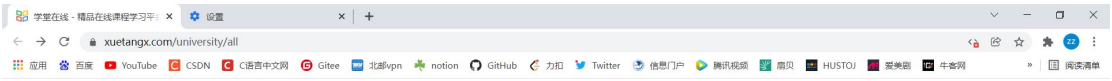
1、 实验准备

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。

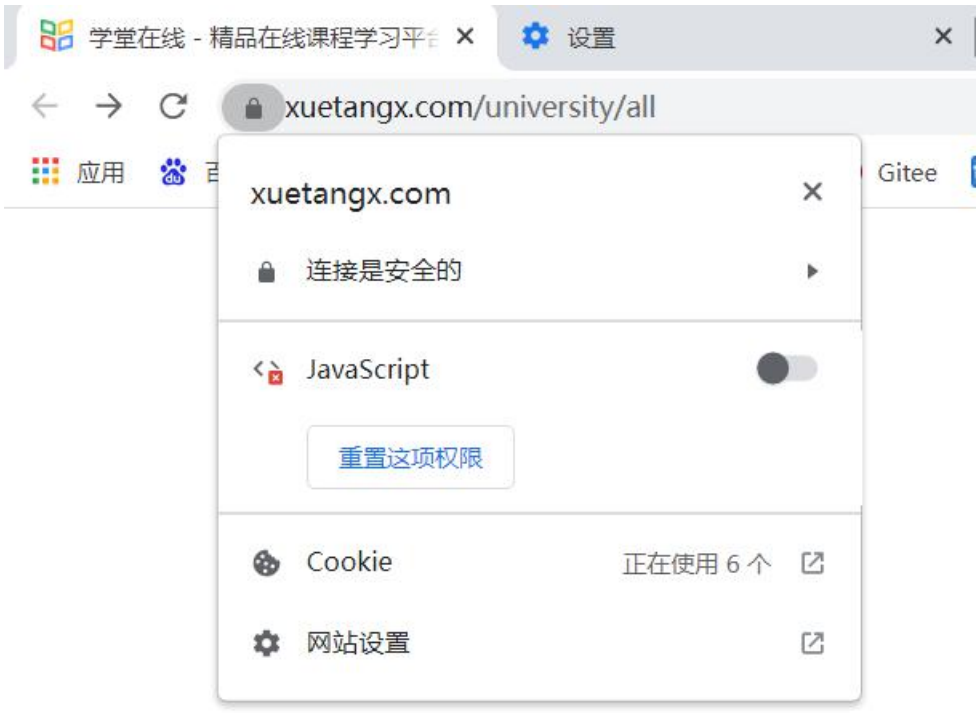
于是我做了以下尝试，我在 chrome 中关闭了 js：



然后再重新加载学堂在线网页，加载如下：

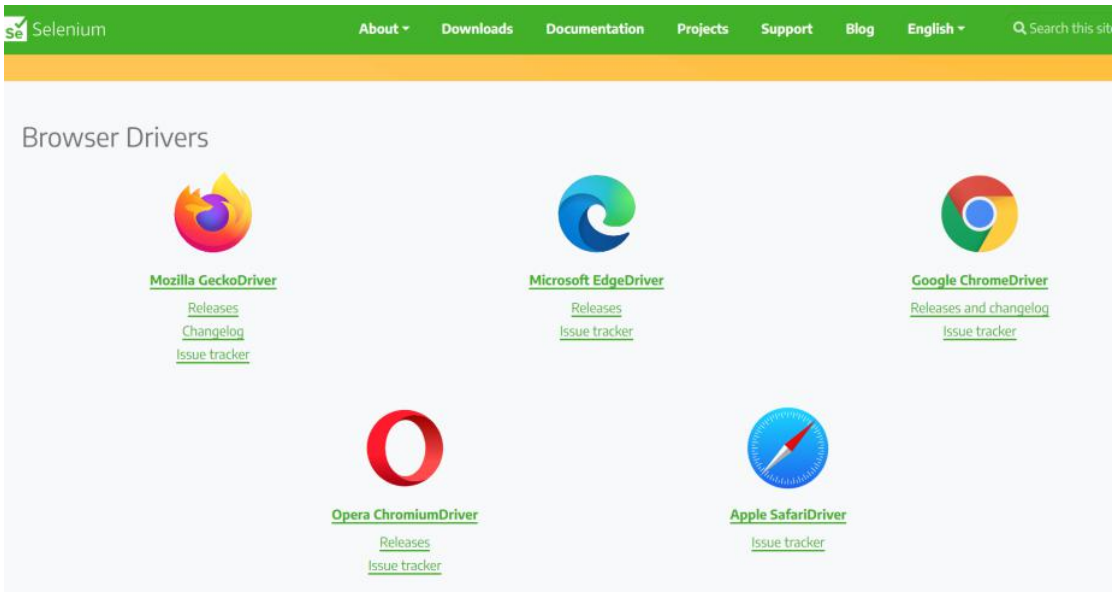


可以看到，整个网页是空的，没有显示任何内容。打开 javascript 再重新加载网页，又全部恢复正常了。



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。

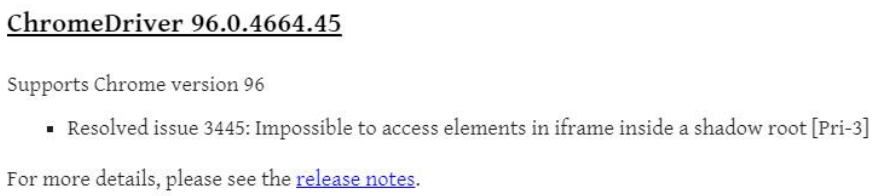
为了保证 Selenium 正确运行，需要安装 webdriver:



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']
```

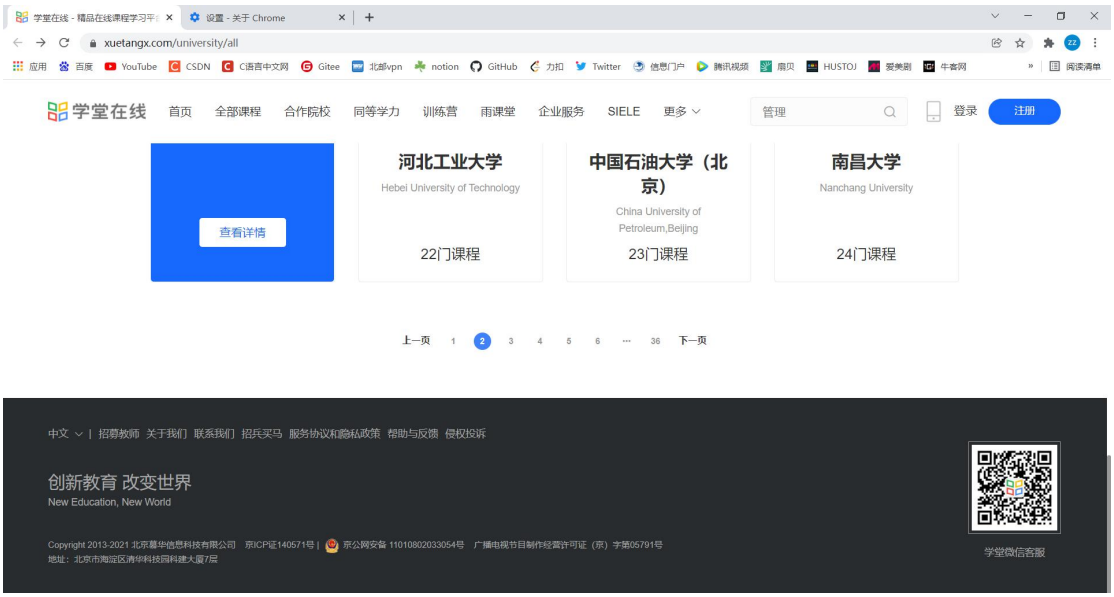
确定好参数后，下面来编写操作函数 parse：

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"}
```


五、 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。

六、 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
}
```