1. **问题描述**
2. 爬取学堂在线的计算机类课程页面内容

https://www.xuetangx.com/search?query=&org=&classify=1&type=&status=&page=1

要求将课程名称、老师、所属学校和选课人数信息，保存到一个csv文件中。

1. 爬取链家官网二手房的数据

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

要求爬取北京市东城、西城、海淀和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json文件中。

1. **实验环境**

Microsoft Windows 10 版本18363

PyCharm 2020.2.1 (Community Edition)

Python 3.8(Scrapy 2.4.0+selenium 3.141.0+chromedriver 85.0.4183.87)

1. **实验内容**
2. 爬取学堂在线的计算机类课程页面内容

class XuetangItem(scrapy.Item):

course = scrapy.Field()

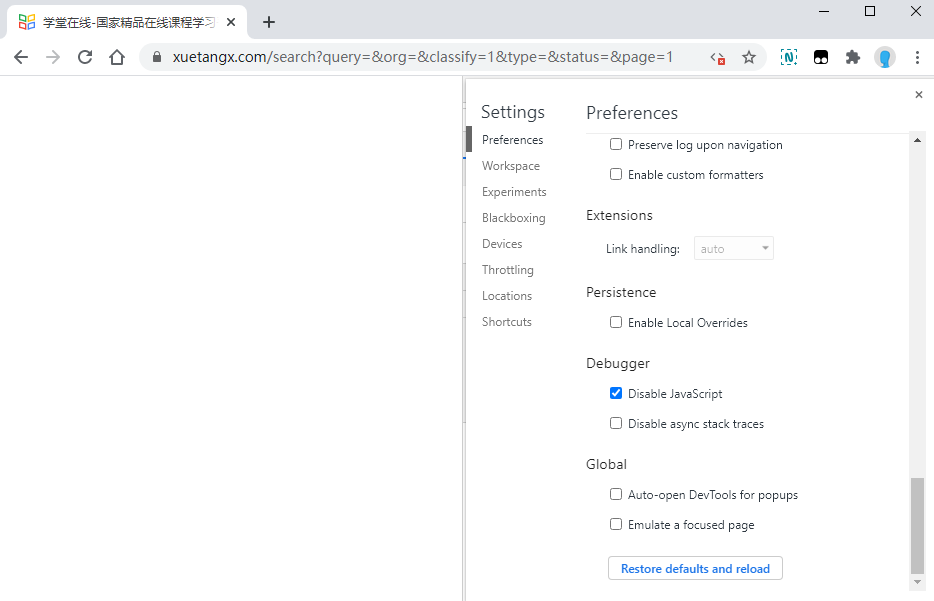
teacher = scrapy.Field()

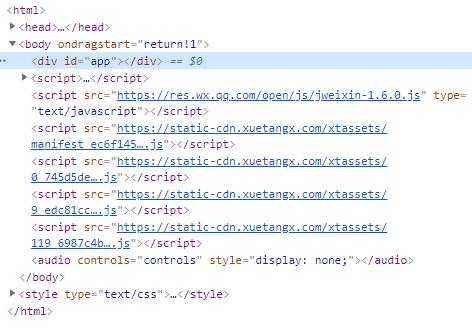
university = scrapy.Field()

num\_stu = scrapy.Field()

pass

↑分析实验要求，在items.py中添加课程名称、老师、所属学校、选课人数信息项。





↑分析页面，关闭Javascript时，发现页面无法正常加载，且源代码中不包含条目信息，说明搜索条目信息需要执行JS代码后才能获取，此时考虑结合Selenium启用DownloaderMiddleware将加载后的页面返回进行分析。

DOWNLOADER\_MIDDLEWARES = {

'xuetang.middlewares.XuetangDownloaderMiddleware': 543,

}

↑在settings.py中启用如上语句

↑主要修改middlewares.py的selenium\_request与process\_request函数

class XuetangDownloaderMiddleware:

def selenium\_request(self, url):

js = ""

chrome\_options = webdriver.ChromeOptions()

chrome\_options.add\_argument("--headless")

chrome\_options.add\_experimental\_option("prefs", {"profile.managed\_default\_content\_settings.images": 2})

driver = webdriver.Chrome(chrome\_options=chrome\_options,

executable\_path=r"C:\Users\(略)\PycharmProjects\pythonProject\venv\Scripts\chromedriver")

driver.get(url)

driver.execute\_script(js)

content = driver.page\_source.encode('utf-8')

# driver.quit()

driver.close()

# return None

return content

def process\_request(self, request, spider):

content = self.selenium\_request(request.url)

if content.strip() != '':

return HtmlResponse(request.url, encoding='utf-8', body=content, request=request)

return None

↑在/spiders文件目录下新建爬虫文件并构造主体内容。由于在原网页中下一页按钮仍未javascript实现的button，不易获取地址，因而考虑效率以及为实现方便，将所需爬取的共37页直接添加在start\_urls中。

class xtzxSpider(scrapy.Spider):

name = "xtzx"

allowed\_domains = ["xuetangx.com/search"]

start\_urls = []

for i in range(1, 38):

url = 'https://www.xuetangx.com/search?query=&org=&classify=1&type=&status=&page=' + str(i)

start\_urls.append(url)

def parse(self, response):

item = XuetangItem()

for each in response.xpath('//div[@class="resultListCon"]'):

item['course'] = "".join(each.xpath('div[2]//p[1]//span//text()').extract())

item['teacher'] = "".join(each.xpath('div[2]//p[2]//span[1]//span[1]//text()').extract())

item['university'] = "".join(each.xpath('div[2]//p[2]//span[2]//span[1]//text()').extract())

item['num\_stu'] = each.xpath('div[2]//p[2]//span[3]//text()').extract()[-1].replace(' ', '').replace('\n', '')

if item['course']:

yield item

在项目目录下执行scrapy crawl xtzx -o result.csv将内容保存在result.csv文件中即可。

1. 爬取链家官网二手房的数据

class LianjiaItem(scrapy.Item):

district = scrapy.Field()

name = scrapy.Field()

pricetotal = scrapy.Field()

priceunit = scrapy.Field()

area = scrapy.Field()

pass

↑分析实验要求，在items.py中添加区属、楼盘名称、总价、平米数、单价信息项。

同爬取学堂在线内容时的分析方法，在禁用Javascript后发现页面的主要信息均可正常加载。

class ljSpider(scrapy.Spider):

name = "lj"

allowed\_domains = ["lianjia.com/ershoufang"]

start\_urls = []

for i in range(1, 6):

url = 'https://bj.lianjia.com/ershoufang/dongcheng/pg' + str(i)

start\_urls.append(url)

url = 'https://bj.lianjia.com/ershoufang/xicheng/pg' + str(i)

start\_urls.append(url)

url = 'https://bj.lianjia.com/ershoufang/chaoyang/pg' + str(i)

start\_urls.append(url)

url = 'https://bj.lianjia.com/ershoufang/haidian/pg' + str(i)

start\_urls.append(url)

def parse(self, response):

item = LianjiaItem()

district = ''

if 'dongcheng' in response.request.url:

district = '东城区'

elif 'xicheng' in response.request.url:

district = '西城区'

elif 'chaoyang' in response.request.url:

district = '朝阳区'

elif 'haidian' in response.request.url:

district = '海淀区'

for each in response.xpath('//div[@class="info clear"]'):

item['district'] = district

item['name'] = "".join(each.xpath('div[@class="title"]//a//text()').extract())

item['pricetotal'] = "".join(each.xpath('div[@class="priceInfo"]//div[@class="totalPrice"]//text()').extract())

item['priceunit'] = "".join(each.xpath('div[@class="priceInfo"]//div[@class="unitPrice"]//span//text()').extract())[2:]

item['area'] = "".join(each.xpath('//div[@class="houseInfo"]//text()').extract()).partition('|')[2].partition('|')[0].replace(' ', '')

if item['name']:

yield item

↑静态网页直接配置/spiders目录下的爬虫文件即可。

class LianjiaPipeline:

# 构造方法（初始化对象时执行的方法）

def \_\_init\_\_(self):

# 必须使用 w+ 模式打开文件，以便后续进行 读写操作（w+模式，意味既可读，亦可写）

# 注意：此处打开文件使用的不是 python 的 open 方法，而是 codecs 中的 open 方法

self.json\_file = codecs.open('result.json', 'w+', encoding='UTF-8')

# 爬虫开始时执行的方法

def open\_spider(self, spider):

# 在爬虫开始时，首先写入一个 '[' 符号，构造一个 json 数组

# 为使得 Json 文件具有更高的易读性，我们辅助输出了 '\n'（换行符）

self.json\_file.write('[\n')

# 爬虫 pipeline 接收到 Scrapy 引擎发来的 item 数据时，执行的方法

def process\_item(self, item, spider):

# 将 item 转换为 字典类型，并编码为 json 字符串，写入文件

# 为使得 Json 文件具有更高的易读性，我们辅助输出了 '\t'（制表符） 与 '\n'（换行符）

item\_json = json.dumps(dict(item), ensure\_ascii=False)

self.json\_file.write('\t' + item\_json + ',\n')

return item

# 爬虫结束时执行的方法

def close\_spider(self, spider):

# 在结束后，需要对 process\_item 最后一次执行输出的 “逗号” 去除

# 当前文件指针处于文件尾，我们需要首先使用 SEEK 方法，定位文件尾前的两个字符（一个','(逗号), 一个'\n'(换行符)）的位置

self.json\_file.seek(-2, os.SEEK\_END)

# 使用 truncate() 方法，将后面的数据清空

self.json\_file.truncate()

# 重新输出'\n'，并输入']'，与 open\_spider(self, spider) 时输出的 '['，构成一个完整的数组格式

self.json\_file.write('\n]')

# 关闭文件

self.json\_file.close()

ITEM\_PIPELINES = {

'lianjia.pipelines.LianjiaPipeline': 300,

}

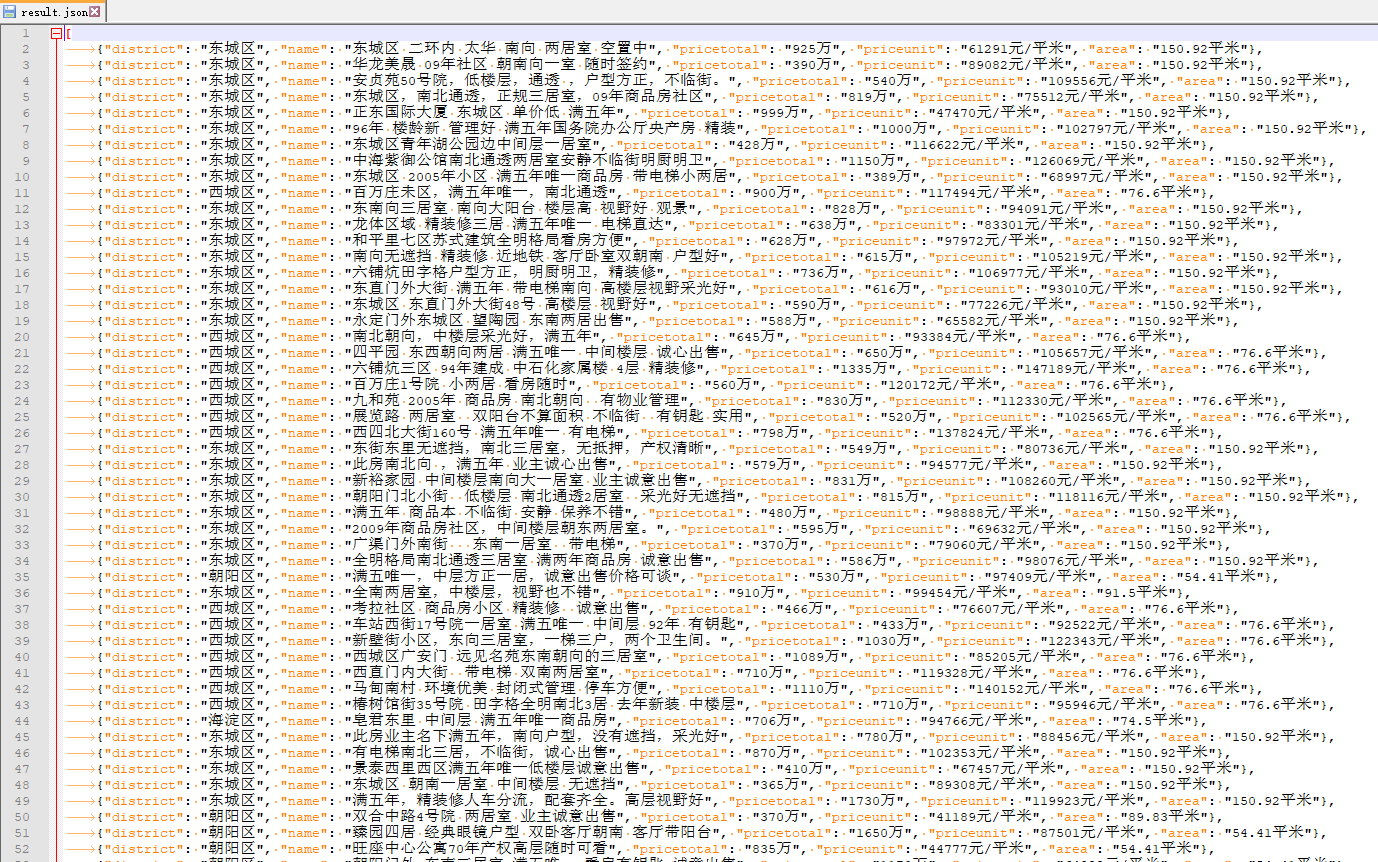
↑由于输出格式为json文件，因而需要在settings.py中启用pipelines，并在pipelines.py中配置如上写入方法。

在项目目录下执行scrapy crawl lj即可，内容通过pipeline保存在result.json文件中。

1. **实验结果**
2. 爬取学堂在线的计算机类课程页面内容



1. 爬取链家官网二手房的数据



1. **心得与体会**

在本次实验中，我收获了许多。由于之前对前端略有了解，因而在数据通过xpath的定位和获取上没有花费太多精力。但是在处理学堂在线动态页面上花费了较长时间寻找解决办法，了解到了selenium、splash等库，以及通过分析js链接获取真实数据接口位置等方法。最终选择了较为简单、常用的selenium搭配chrome无头浏览器进行获取。另一方面，我深刻感受到了python作为当今的热门编程语言，其社区内容的丰富性，例如本实验中的Scrpay与Selenium，将一些棘手的问题进行了封装抽象，使得爬虫只需简单配置及连接内容即可轻松实现。这在我今后使用python处理问题时有启发作用，以及在今后程序设计中对问题的抽象有了更多的认识。非常感谢老师的实验设计！