## 一、爬取数据：

### 1.1 **scrapy爬虫库简介：**

Scrapy，Python开发的一个快速、高层次的屏幕抓取和web抓取框架，用于抓取web站点并从页面中提取结构化的数据。Scrapy用途广泛，可以用于数据挖掘、监测和自动化测试.

其最初是为了页面抓取 (更确切来说, 网络抓取 )所设计的， 后台也应用在获取API所返回的数据(例如 Amazon Associates Web Services ) 或者通用的网络爬虫.

Scrapy吸引人的地方在于它是一个框架，任何人都可以根据需求方便的修改。它也提供了多种类型爬虫的基类，如BaseSpider、sitemap爬虫等，最新版本又提供了web2.0爬虫的支持.

### 1.2 **技术实现**：

#### 1.2.1搭建框架结构

首先搭建scrapy库项目，运行如下代码生成代码框架：

scrapy startproject crawler

cmdline.execute('scrapy crawl jd'.split())

Python

spiders文件夹中的py文件即为主要爬虫代码，当获取网页请求后，在该文件的主类中重写parse方法，即请求解析代码。

spiders上级目录下的items文件即为获取到的字段，也就是爬取的属性值，示例如下：

class CrawljingdongItem(scrapy.Item):

id=scrapy.Field()

comment=scrapy.Field()

productName=scrapy.Field()

storeName=scrapy.Field()

address=scrapy.Field()

price=scrapy.Field()

UserComments=scrapy.Field()

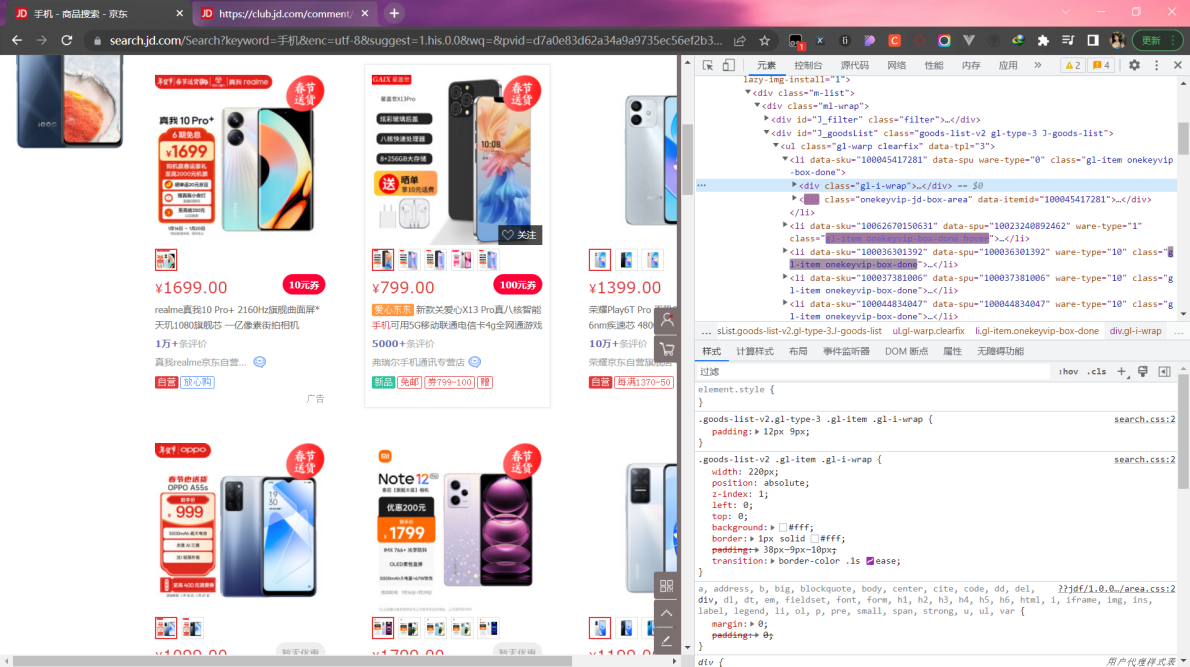
pass

Python

#### 1.2.2 分析网页结构

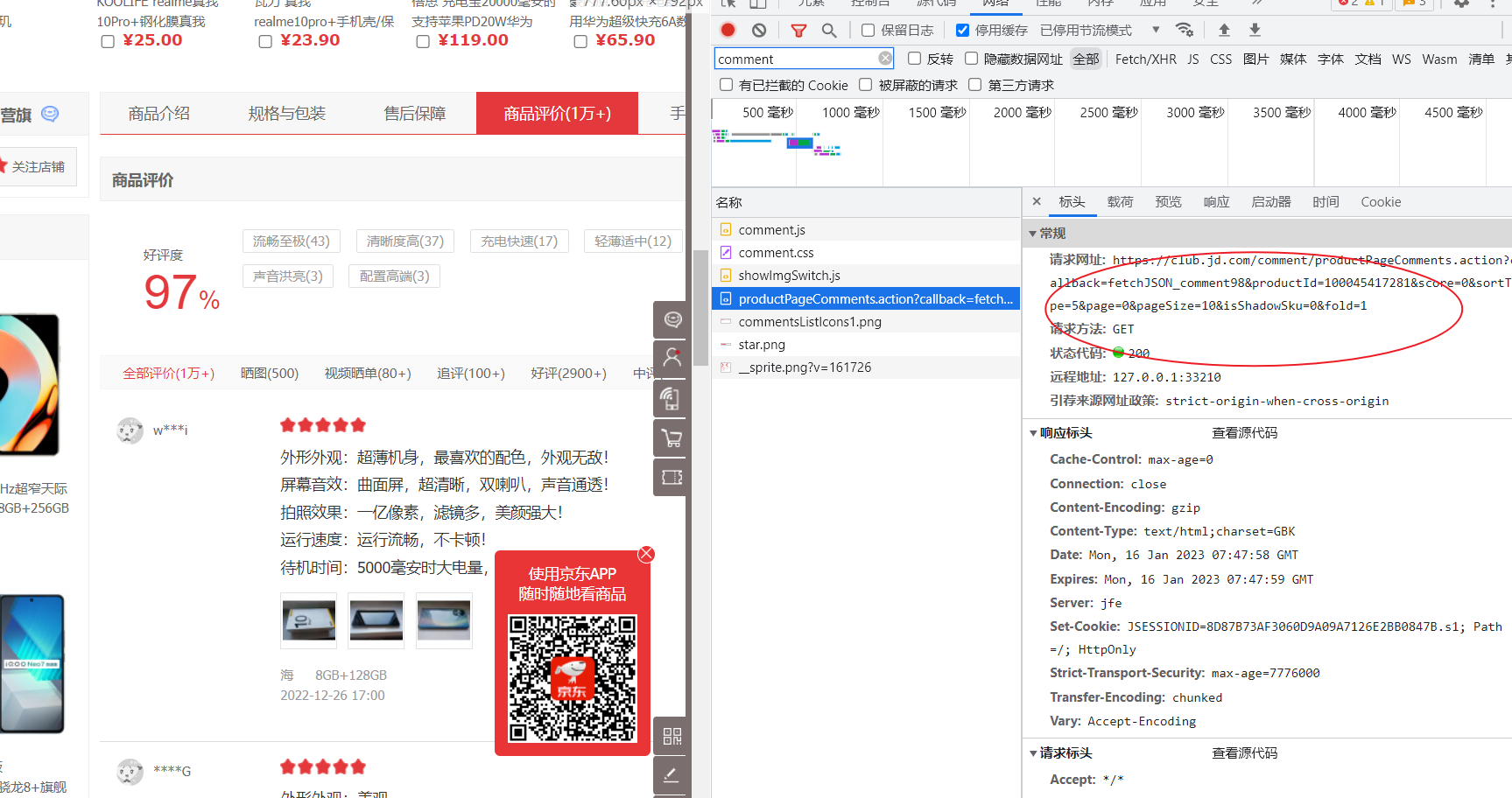
在重写parse请求时，首先需要对网页进行结构分析，这里以京东商品页面进行展示。

首先打开京东界面，得到网址，填入starturls中。然后使用xpath进行HTML的解析，此时可以获取静态页面中的所有内容



于是使用data = ans\_html.xpath("//\*[@class='gl-warp clearfix']/li") 可以得到所有的列表标签，可以看到，里面存放了所有的单位商品信息。随后使用for循环遍历每个列表，对商品信息进行单独提取，并使用item类进行存储。

这里需要注意的是，对于评论等数据是无法直接从HTML中提取出来的，因为这些数据通常是基于ajax技术进行异步传输，即滑动时会进行加载（动态加载），所以必须找到保存评论信息的json文件。通过网页检查器，可以发现文件位置，根据该URL使用request库进行请求即可。



主要逻辑代码完成后，下一步会进入spiders上级目录下的pipelines文件中进行数据的存储级操作，这里使用mysql数据库进行数据的存储。

## 数据保存：

首先需要导入pymsql库，其次进行主要连接属性的配置

def dbHandle(self):

conn = pymysql.connect(

host="localhost",

user="root",

passwd="135157",

charset="utf8",

use\_unicode=False

)

return conn

Python

然后根据sql语法，使用pymysql的execute方法将查询语句传入到sql中进行查询，可以看到数据存储如下：



下一步使用pandas库的read\_sql\_query方法从mysql数据库中读取数据，同时进行分析以及处理。

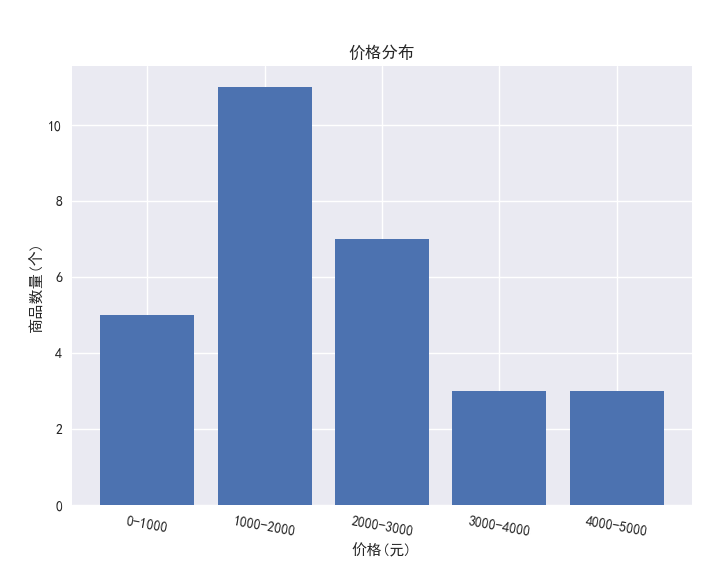
## 三、数据读取以及分析：

将数据读取后得到一个DataFrame对象，然后分别进行数据处理，最终得到 商品价格区间的划分，不同价格区间范围内的商品数量，用户的评论集合，商品名称的集合等数据信息，进行下一步数据可视化。

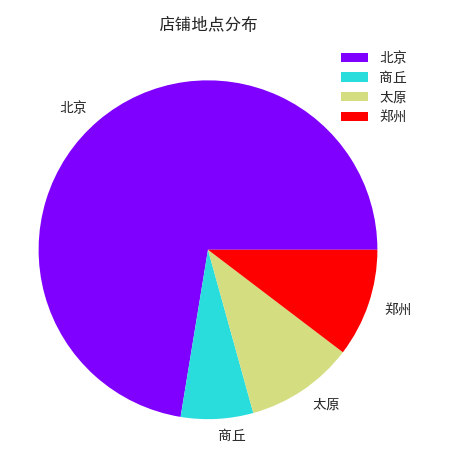
## 四、数据可视化：

首先使用matplotlib的plot方法进行相关的操作。

对商品价格区间内的商品数量使用条形图进行可视化：



对店铺地址进行统计并使用饼图进行可视化：



将所有评论连接起来成一段文章，使用jieba中文分词库进行关键词提取，然后拼接起来调用WordCloud对象的wc.generate方法生成词云：



将所有商品名称连接起来成一段文章，使用jieba中文分词库进行关键词提取，然后拼接起来调用WordCloud对象的wc.generate方法生成词云：

