**我是如何零基础开始能写Python爬虫的**

刚开始接触爬虫的时候，简直惊为天人，十几行代码，就可以将无数网页的信息全部获取下来，自动选取网页元素，自动整理成结构化的文件。

利用这些数据，可以做很多领域的分析、市场调研，获得很多有价值的信息。这种技能不为我所用实在可惜，于是果断开始学习。

**并非开始都是最容易的**

刚开始对爬虫不是很了解，又没有任何的计算机、编程基础，确实有点懵逼。从哪里开始，哪些是最开始应该学的，哪些应该等到有一定基础之后再学，也没个清晰的概念。

因为是 Python 爬虫嘛，Python 就是必备的咯，那先从 Python 开始吧。于是看了一些教程和书籍，了解基本的数据结构，然后是列表、字典、元组，各种函数和控制语句（条件语句、循环语句）。

学了一段时间，才发现自己还没接触到真正的爬虫呢，而且纯理论学习很快就忘了，回去复习又太浪费时间，简直不要太绝望。把 Python 的基础知识过了一遍之后，我竟然还没装一个可以敲代码的IDE，想想就哭笑不得。

**开始直接上手**

转机出现在看过一篇爬虫的技术文章后，清晰的思路和通俗易懂的语言让我觉得，这才是我想学的爬虫。于是决定先配一个环境，试试看爬虫到底是怎么玩的。（当然你可以理解为这是浮躁，但确实每个小白都想去做直观、有反馈的事情）

因为怕出错，装了比较保险的 Anaconda，用自带的 Jupyter Notebook 作为IDE来写代码。看到很多人说因为配置环境出各种BUG，简直庆幸。很多时候打败你的，并不是事情本身，说的就是爬虫配置环境这事儿。

遇到的另一个问题是，Python 的爬虫可以用很多包或者框架来实现，应该选哪一种呢？我的原则就是是简单好用，写的代码少，对于一个小白来说，性能、效率什么的，统统被我 pass 了。于是开始接触 urllib、美丽汤（BeautifulSoup），因为听别人说很简单。

我上手的第一个案例是爬取豆瓣的电影，无数人都推荐把豆瓣作为新手上路的实例，因为页面简单且反爬虫不严。照着一些爬取豆瓣电影的入门级例子开始看，从这些例子里面，了解了一点点爬虫的基本原理：下载页面、解析页面、定位并抽取数据。

当然并没有去系统看 urllib 和 BeautifulSoup 了，我需要把眼前实例中的问题解决，比如下载、解析页面，基本都是固定的语句，直接用就行，我就先不去学习原理了。



用 urllib 下载和解析页面的固定句式

当然 BeautifulSoup 中的基本方法是不能忽略的，但也无非是 find、get\_text() 之类，信息量很小。就这样，通过别人的思路和自己查找美丽汤的用法，完成了豆瓣电影的基本信息爬取。



用 BeautifulSoup 爬取豆瓣电影详情

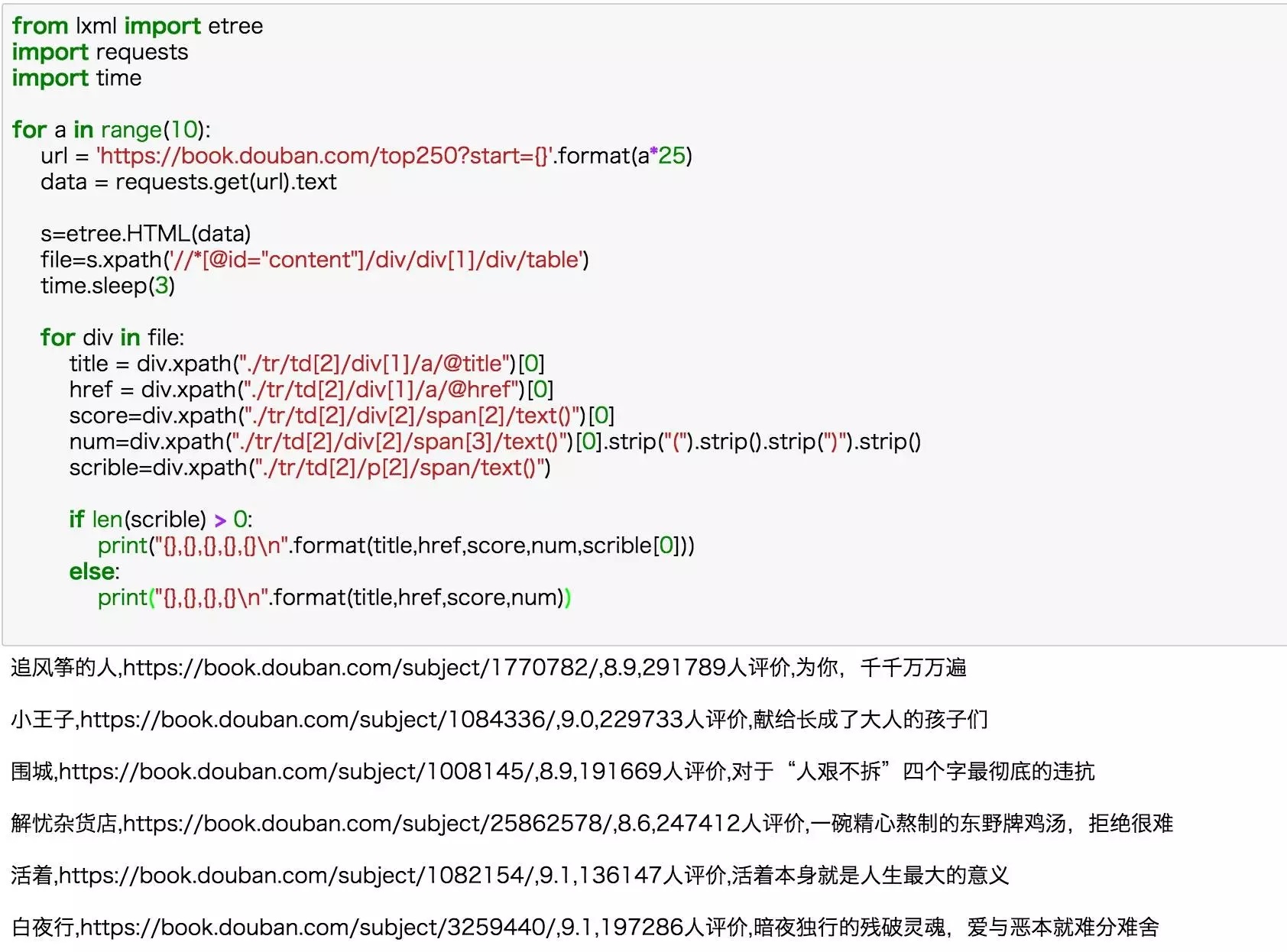
**爬虫渐入佳境**

有了一些套路和形式，就会有目标，可以接着往下学了。还是豆瓣，自己去摸索爬取更多的信息，爬取多部电影，多个页面。这个时候就发现基础不足了，比如爬取多个元素、翻页、处理多种情况等涉及的语句控制，又比如提取内容时涉及到的字符串、列表、字典的处理，还远远不够。

再回去补充 Python 的基础知识，就很有针对性，而且能马上能用于解决问题，也就理解得更深刻。这样直到把豆瓣的TOP250图书和电影爬下来，基本算是了解了一个爬虫的基本过程了。

BeautifulSoup 还算不错，但需要花一些时间去了解一些网页的基本知识，否则一些元素的定位和选取还是会头疼。

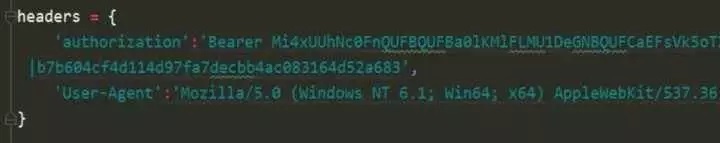
后来认识到 xpath 之后相见恨晚，这才是入门必备利器啊，直接Chrome复制就可以了，指哪打哪。即便是要自己写 xpath，以w3school上几页的 xpath 教程，一个小时也可以搞定了。requests 貌似也比 urllib 更好用，但摸索总归是试错的过程，试错成本就是时间。



requests+xpath 爬取豆瓣TOP250图书信息

**跟反爬虫杠上了**

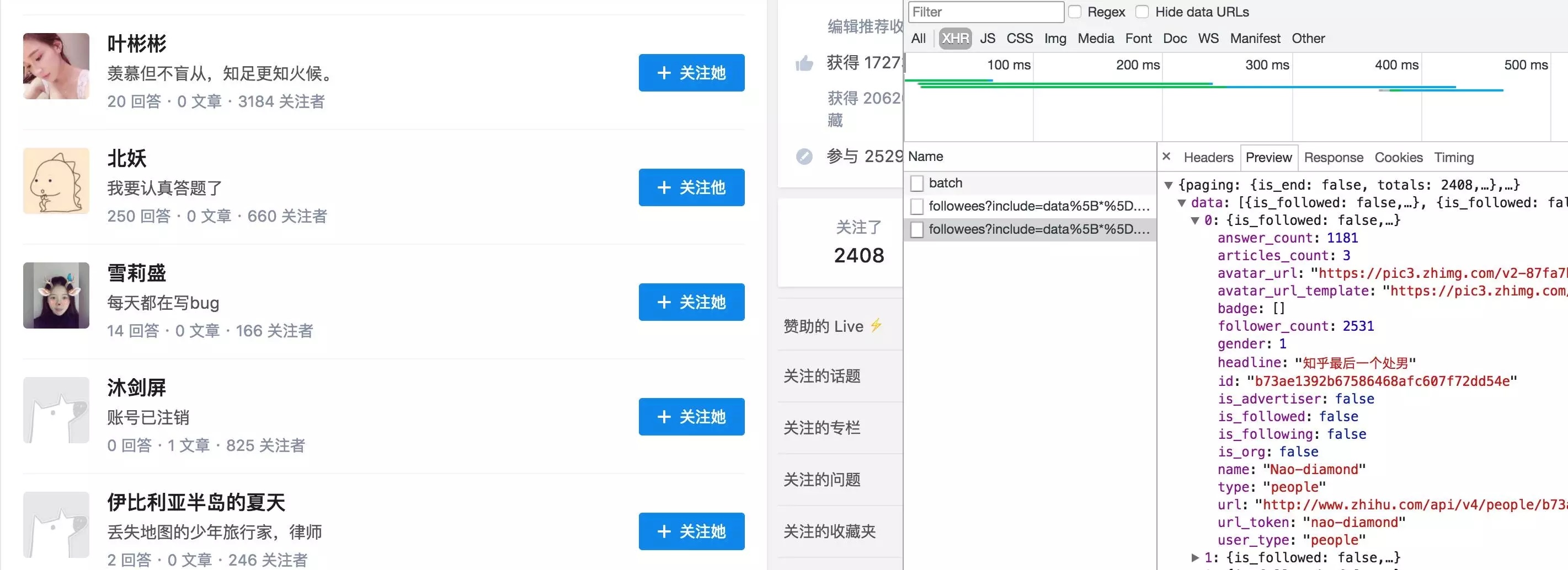
通过 requests+xpath，我可以去爬取很多网站网站了，后来自己练习了小猪的租房信息和当当的图书数据。爬拉勾的时候就发现问题了，首先是自己的请求根本不会返回信息，原来要将自己的爬虫伪装成浏览器，终于知道别人代码中那一坨 headers 信息是干啥的了??。



在爬虫中添加 headers 信息，伪装成真实用户

接着是各种定位不到元素，然后知道了这是异步加载，数据根本不在网页源代码中，需要通过抓包来获取网页信息。于是在各种 JS、XHR的文件中 preview，寻找包含数据的链接。

当然知乎还好，本身加载的文件不多，找到了 json 文件直接获取对应的数据。（这里要安利一个chrome插件：jsonview，让小白轻松看懂 json 文件）



浏览器抓取 JavaScript 加载的数据

在这里就对反爬虫有了认识，当然这还是最基本的，更严格的IP限制、验证码、文字加密等等，可能还会遇到很多难题。但是目前的问题能够解决不就很好了吗，逐个击破才能更高效地学习。

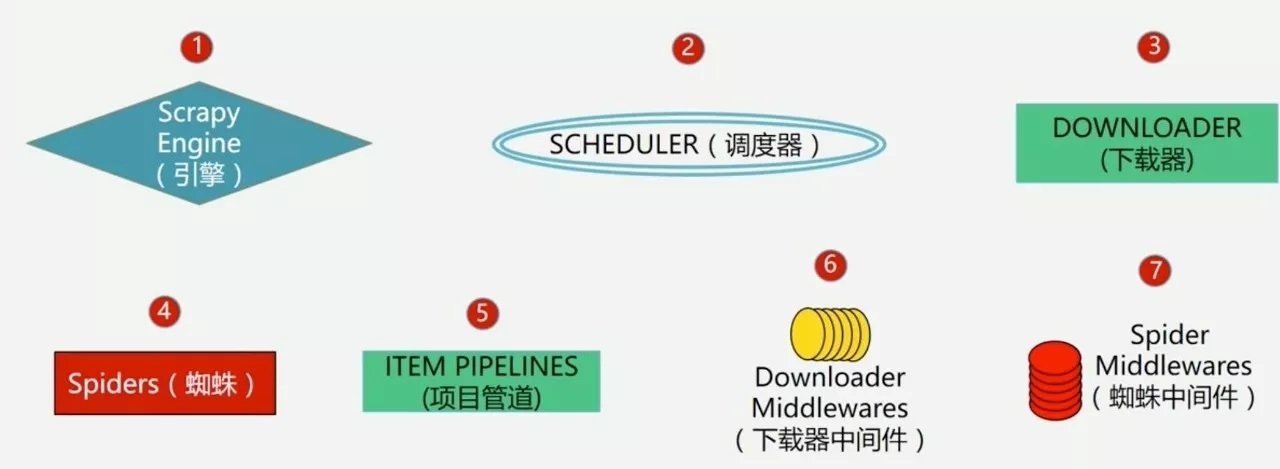
比如后来在爬其他网站的时候就被封了IP，简单的可以通过 time.sleep() 控制爬取频率的方法解决，限制比较严格或者需要保证爬取速度，就要用代理IP来解决。

当然，后来也试了一下 Selenium，这个就真的是按照真实的用户浏览行为（点击、搜索、翻页）来实现爬虫，所以对于那些反爬虫特别厉害的网站，又没有办法解决，Selenium 是一个超级好用的东东，虽然速度稍微慢点。

**尝试强大的 Scrapy 框架**

有了 requests+xpath 和抓包大法，就可以做很多事情了，豆瓣各分类下的电影，58同城、知乎、拉勾这些网站基本都没问题。不过，当爬取的数据量级很大，而且需要灵活地处理各个模块的话，会显得很力不从心。

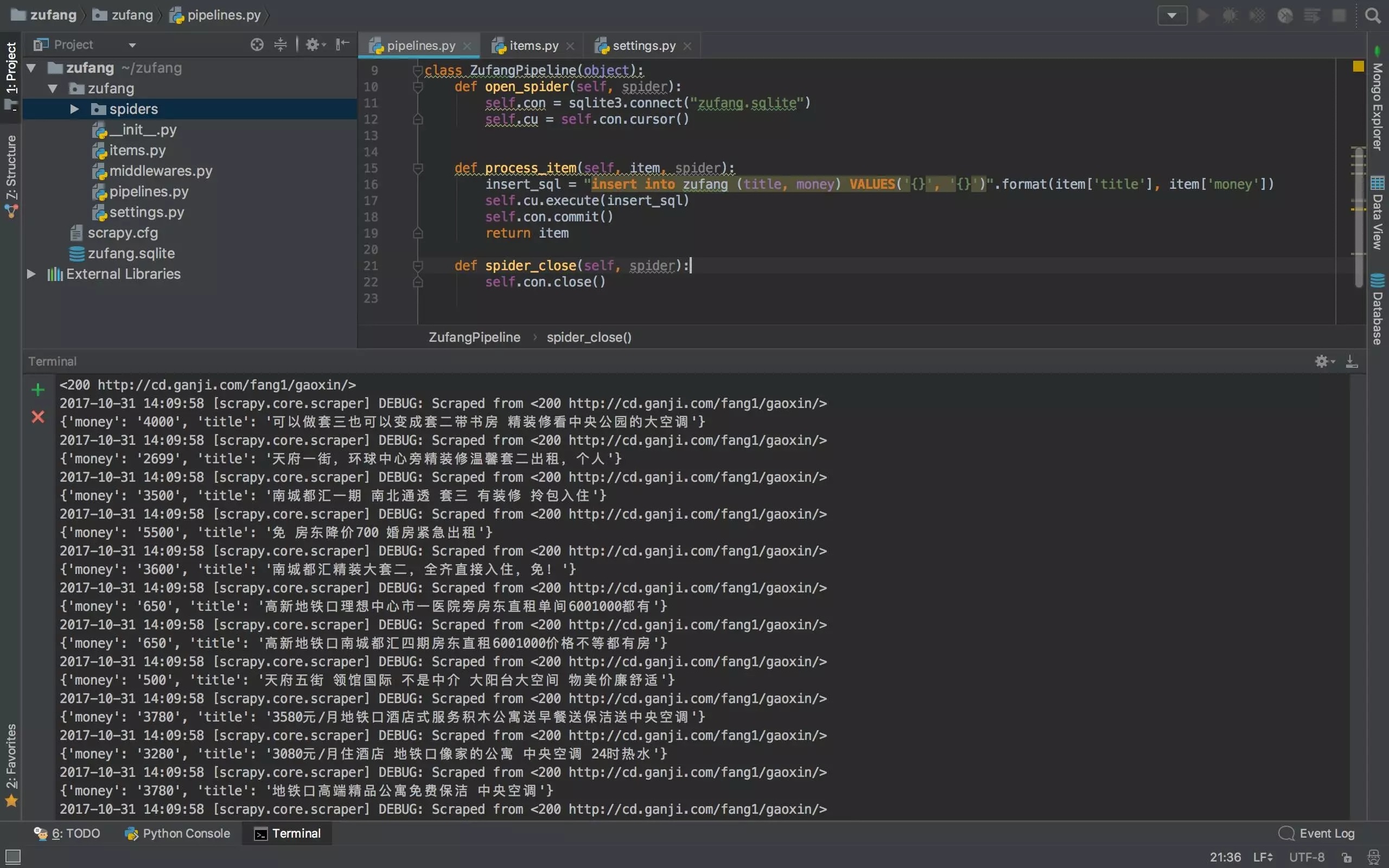
于是了解到强大的 Scrapy 框架，它不仅能便捷地构建 Request，还有强大的 Selector 能够方便地解析 Response，然而最让人惊喜的还是它超高的性能，可以将爬虫工程化、模块化。



Scrapy 框架的基本组件

学会 Scrapy，自己去尝试搭建了简单的爬虫框架，在做大规模数据爬去的时候能够结构化、工程化地思考大规模的爬取问题，这使我可以从爬虫工程的维度去思考问题。

当然 Scrapy 本身的 selector 、中间件、spider 等会比较难理解，还是建议结合具体的例子，参考别人的代码，去理解其中实现的过程，这样能够更好地理解。



用 Scrapy 爬取了大量租房信息

**本地文件搞不动了，上数据库**

爬回来大量的数据之后就发现，本地的文件存起来非常不方便，即便存下来了，打开大文件电脑会卡得很严重。怎么办呢？果断上数据库啊，于是开始入坑 MongoDB。结构化、非结构化的数据都能够存储，安装好 PyMongo，就可以方便地在 Python 中操作数据库了。

MongoDB 本身安装会比较麻烦，如果自己一个人去折腾，很有可能会陷入困境。刚开始安装的时候也是出现各种BUG，幸得大神小X指点，解决了很多问题。

当然对于爬虫这一块，并不需要多么高深的数据库技术，主要是数据的入库和提取，顺带掌握了基本的插入、删除等操作。总之，能够满足高效地提取爬下来的数据就OK了。



爬取拉勾招聘数据并用 MongoDB 存储

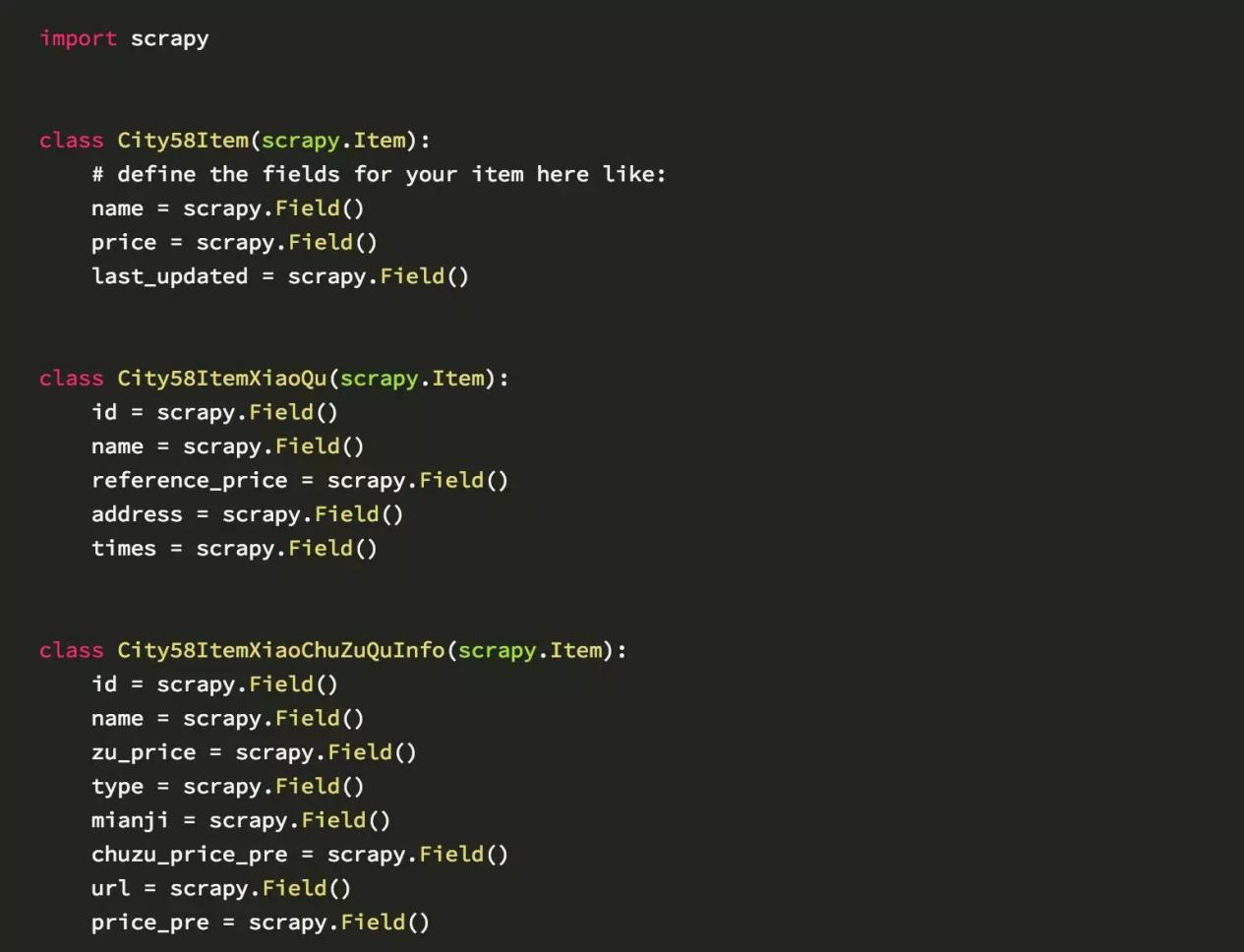
**传说中的分布式爬虫**

这个时候，基本上很大一部分的网页都能爬了，瓶颈就集中到爬取大规模数据的效率。因为学了 Scrapy，于是自然地接触到一个很厉害的名字：分布式爬虫。

分布式这个东西，一听不明觉厉，感觉很恐怖，但其实就是利用多线程的原理让多个爬虫同时工作，除了前面学过的 Scrapy 和 MongoDB，好像还需要了解 Redis。

Scrapy 用于做基本的页面爬取，MongoDB 用于存储爬取的数据，Redis 则用来存储要爬取的网页队列，也就是任务队列。

分布式这东西看起来很吓人，但其实分解开来，循序渐进地学习，也不过如此。



分布式爬58同城：定义项目内容部分

零基础学习爬虫，坑确实比较多，总结如下：

1.环境配置，各种安装包、环境变量，对小白太不友好；

2.缺少合理的学习路径，上来 Python、HTML 各种学，极其容易放弃；

3.Python有很多包、框架可以选择，但小白不知道哪个更友好；

4.遇到问题甚至不知道如何描述，更不用说去寻找解决办法；

5.网上的资料非常零散，而且对小白不友好，很多看起来云里雾里；

6.有些东西看似懂了，但结果自己写代码还是很困难；

……………………

所以跟我一样，很多人爬坑最大的体会是：尽量不要系统地去啃一些东西，找一个实际的项目（从豆瓣这种简单的入手），直接开始就好。

因为爬虫这种技术，既不需要你系统地精通一门语言，也不需要多么高深的数据库技术，从实际的项目中去学习这些零散的知识点，你能保证每次学到的都是最需要的那部分。

当然麻烦的是，在具体的问题中，如何找到具体需要的那部分学习资源、如何筛选和甄别，遇到困难时如何有效解决，是很多初学者面临的大问题。

不过不用担心，我们准备了一门非常系统的爬虫课程，除了为你提供一条清晰、无痛的学习路径，我们甄选了最实用的学习资源以及庞大的主流爬虫案例库。即便是零基础，经过短时间的学习，也能很好地掌握爬虫这个技能，获取你想得到的数据。