面对众多的自媒体的原创，我们借鉴RSS阅读器的基本原理。使用python爬虫。

一般在开发大规模爬虫系统时可能会面临以下挑战：

网页抓取

最简单的抓取就是使用 HTTPClient 或者 fetch 或者 request 这样的 HTTP 客户端。现在随着单页应用这样富客户端应用的流行，我们可以使用 Selenium、PhantomJS 这样的 Headless Brwoser 来动态执行脚本进行渲染。

网页解析

对于网页内容的抽取与解析是个很麻烦的问题，DOM4j、Cherrio、beautifulsoup 这些为我们提供了基本的解析功能。笔者也尝试过构建全配置型的爬虫，类似于 [Web-Scraper](https://link.juejin.im/?target=http%3A%2F%2Flink.zhihu.com%2F%3Ftarget%3Dhttps%253A%2F%2Fchrome.google.com%2Fwebstore%2Fdetail%2Fweb-scraper%2Fjnhgnonknehpejjnehehllkliplmbmhn%2Frelated)，然而还是输给了复杂多变，多层嵌套的 iFrame 页面。这里笔者秉持代码即配置的理念，对于使用配置来声明的内建复杂度比较低，但是对于那些业务复杂度较高的网页，整体复杂度会以几何倍数增长。而使用代码来声明其内建复杂度与门槛相对较高，但是能较好地处理业务复杂度较高的网页。笔者在构思未来的交互式爬虫生成界面时，也是希望借鉴 FaaS 的思路，直接使用代码声明整个解析流程，而不是使用配置。

反爬虫对抗

类似于淘宝这样的主流网站基本上都有反爬虫机制，它们会对于请求频次、请求地址、请求行为与目标的连贯性等多个维度进行分析，从而判断请求者是爬虫还是真实用户。我们常见的方式就是使用多 IP 或者多代理来避免同一源的频繁请求，或者可以借鉴 GAN 或者增强学习的思路，让爬虫自动地针对目标网站的反爬虫策略进行自我升级与改造。另一个常见的反爬虫方式就是验证码，从最初的混淆图片到现在常见的拖动式验证码都是不小的障碍，我们可以使用图片中文字提取、模拟用户行为等方式来尝试绕过。

分布式调度

单机的吞吐量和性能总是有瓶颈的，而分布式爬虫与其他分布式系统一样，需要考虑分布式治理、数据一致性、任务调度等多个方面的问题。笔者个人的感觉是应该将爬虫的工作节点尽可能地无状态化，以 Redis 或者 Consul 这样的能保证高可用性的中心存储存放整个爬虫集群的状态。

在线有价值页面预判

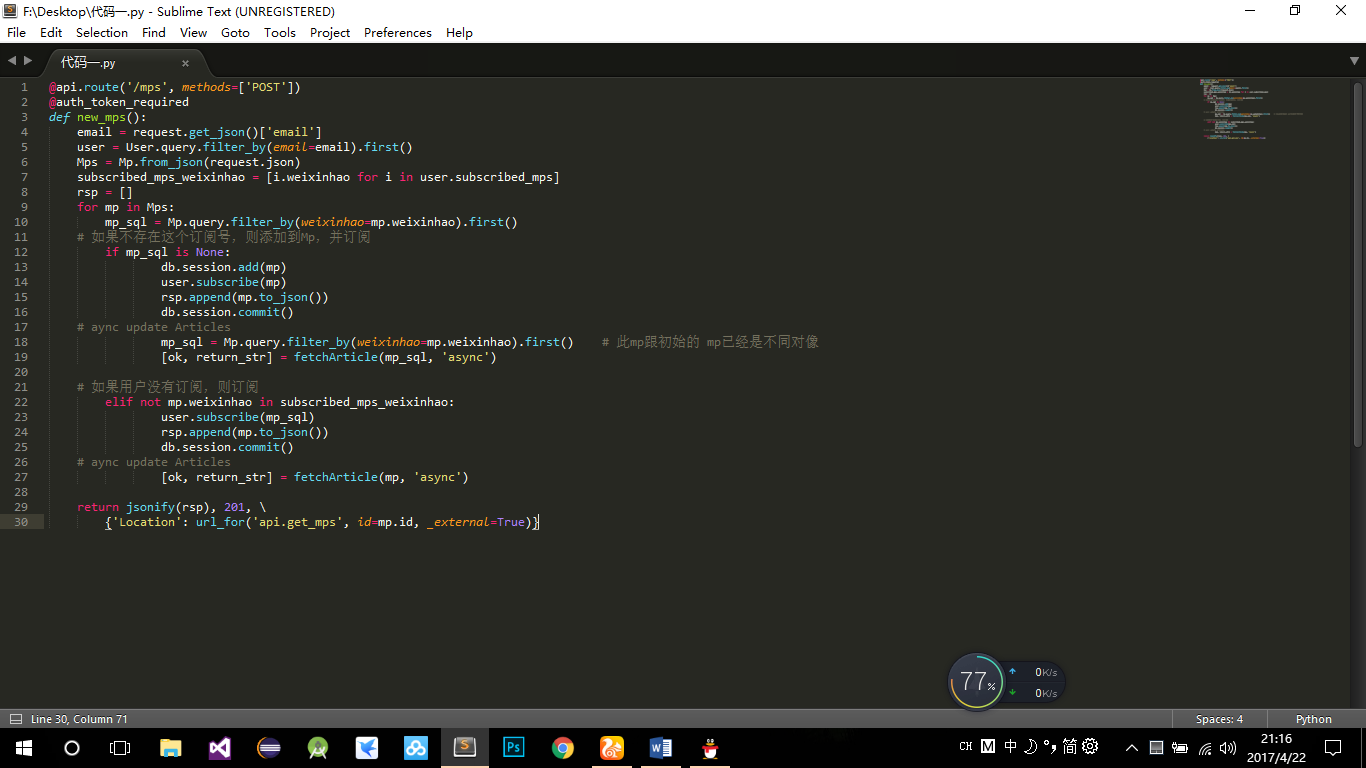
Google 经典的 PageRank 能够基于网络中的连接信息判断某个 URL 的有价值程度，从而优先索引或者抓取有价值的页面。而像[Anthelion](https://link.juejin.im/?target=http%3A%2F%2Flink.zhihu.com%2F%3Ftarget%3Dhttps%253A%2F%2Fgithub.com%2Fyahoo%2Fanthelion) 这样的智能解析工具能够基于之前的页面提取内容的有价值程度来预判某个 URL 是否有抓取的必要。

页面内容提取与储存

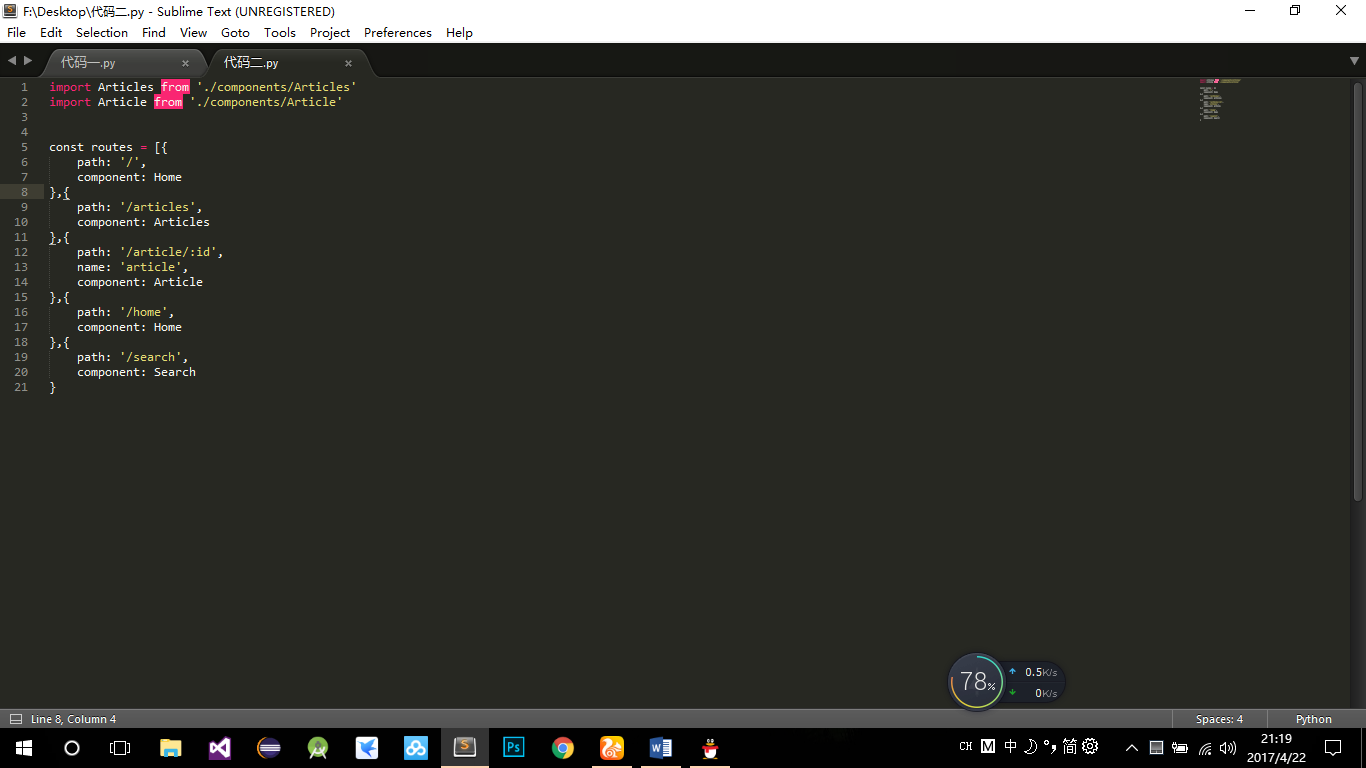
对于网页中的结构化或者非结构化的内容实体提取是自然语言处理中的常见任务之一，而自动从海量数据中提取出有意义的内容也涉及到机器学习、大数据处理等多个领域的知识。我们可以使用 Hadoop MapReduce、Spark、Flink 等离线或者流式计算引擎来处理海量数据，使用词嵌入、主题模型、LSTM 等等机器学习技术来分析文本，可以使用 HBase、ElasticSearch 来存储或者对文本建立索引。

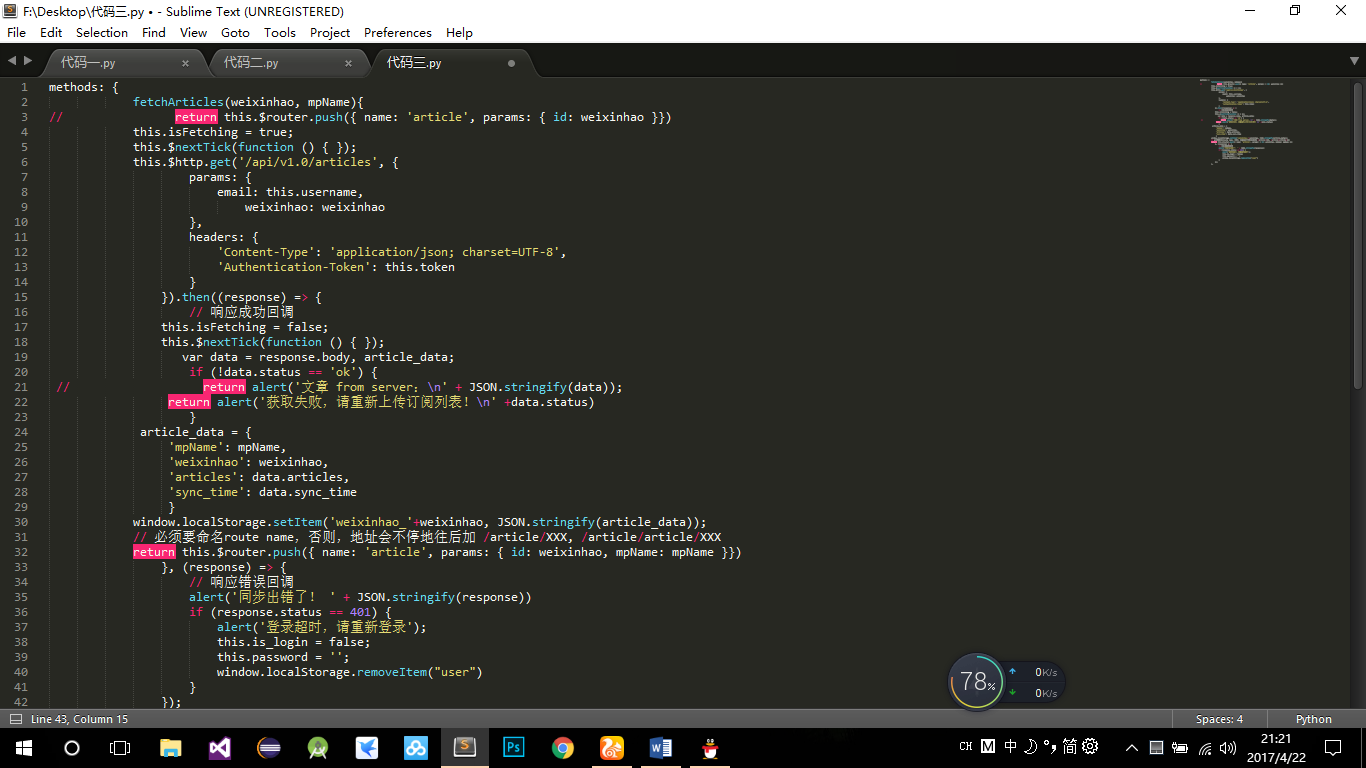
**云库需要完成功能有：（代码详见附录）**

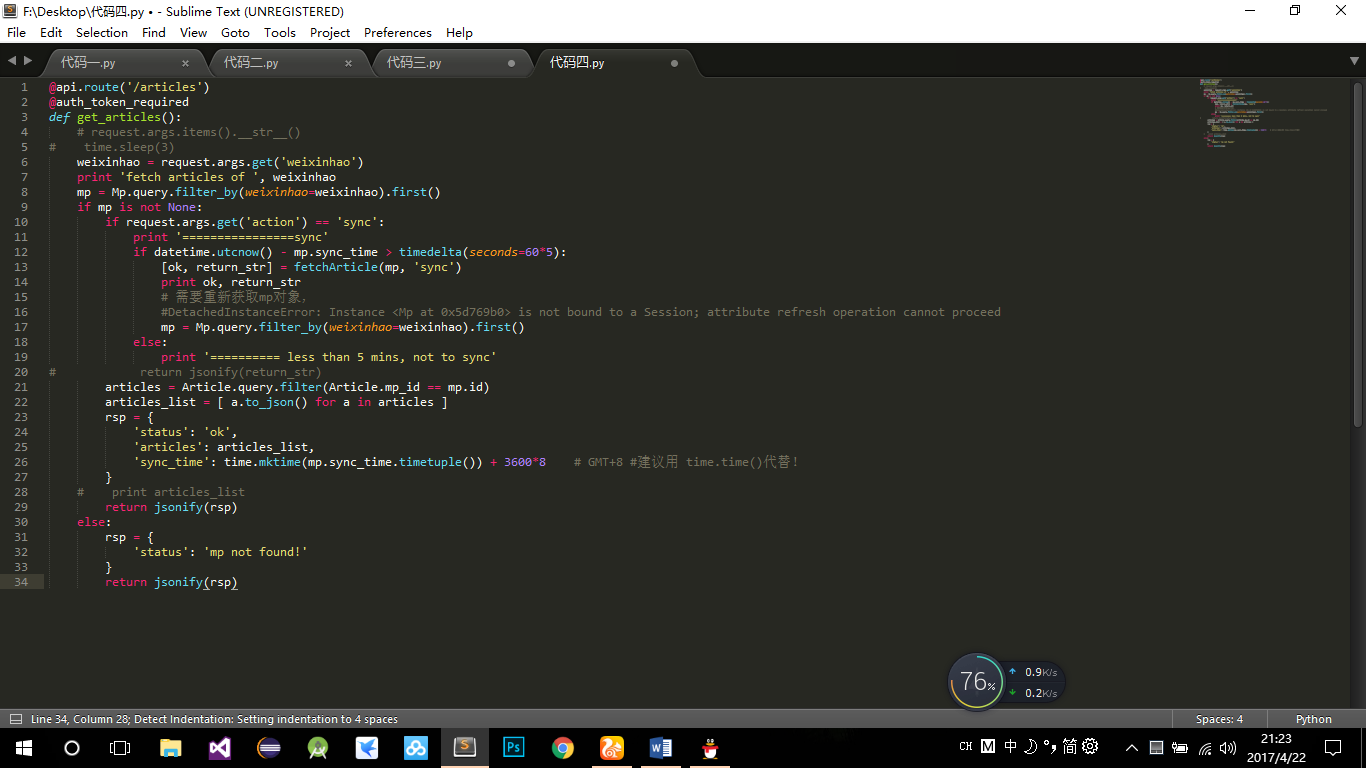
1. 上传订阅列表时，Python抓取公众号主页上的文章列表

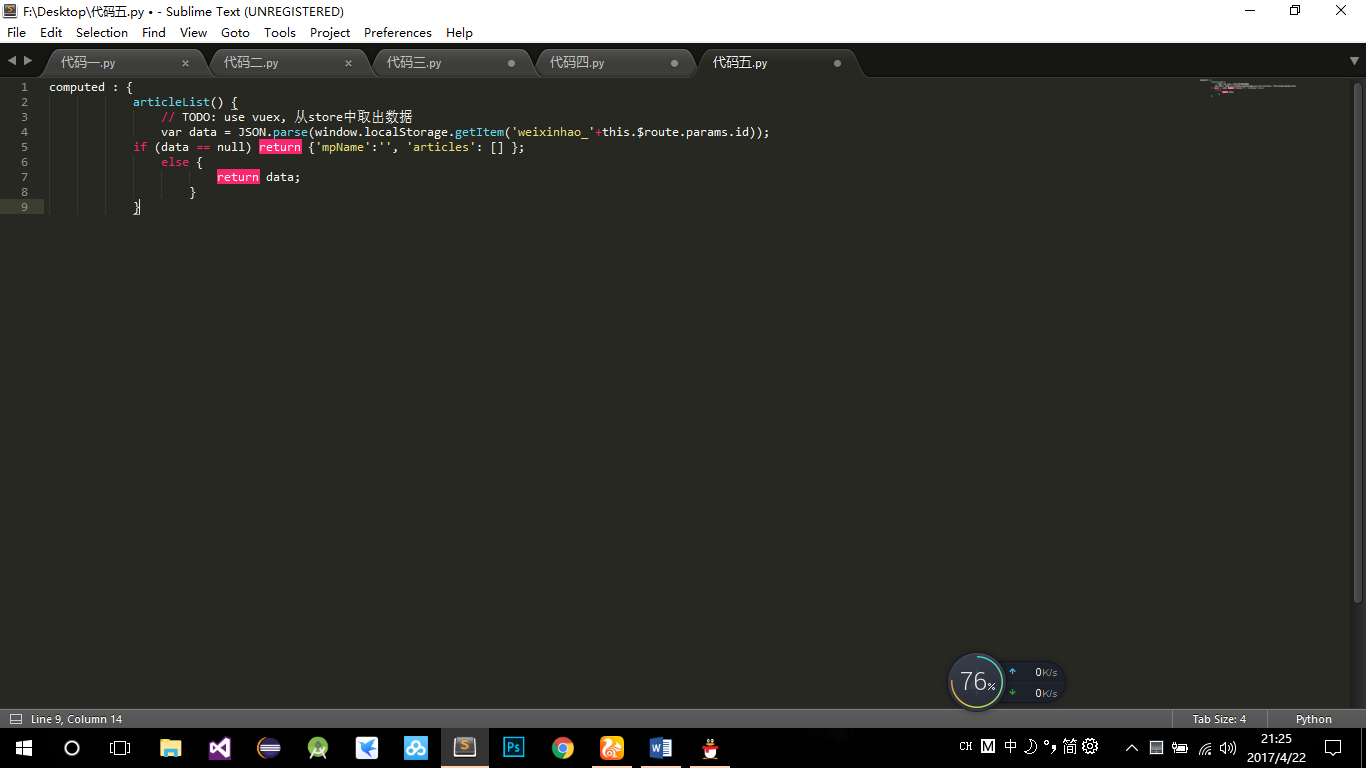
通过添加：Flask异步调用函数fetchArticle(mp, 'async')，来爬虫抓取公众号主页上的文章（代码1）

1. 点击右侧导航栏某一公众号，左侧显示它所包含的文章列表

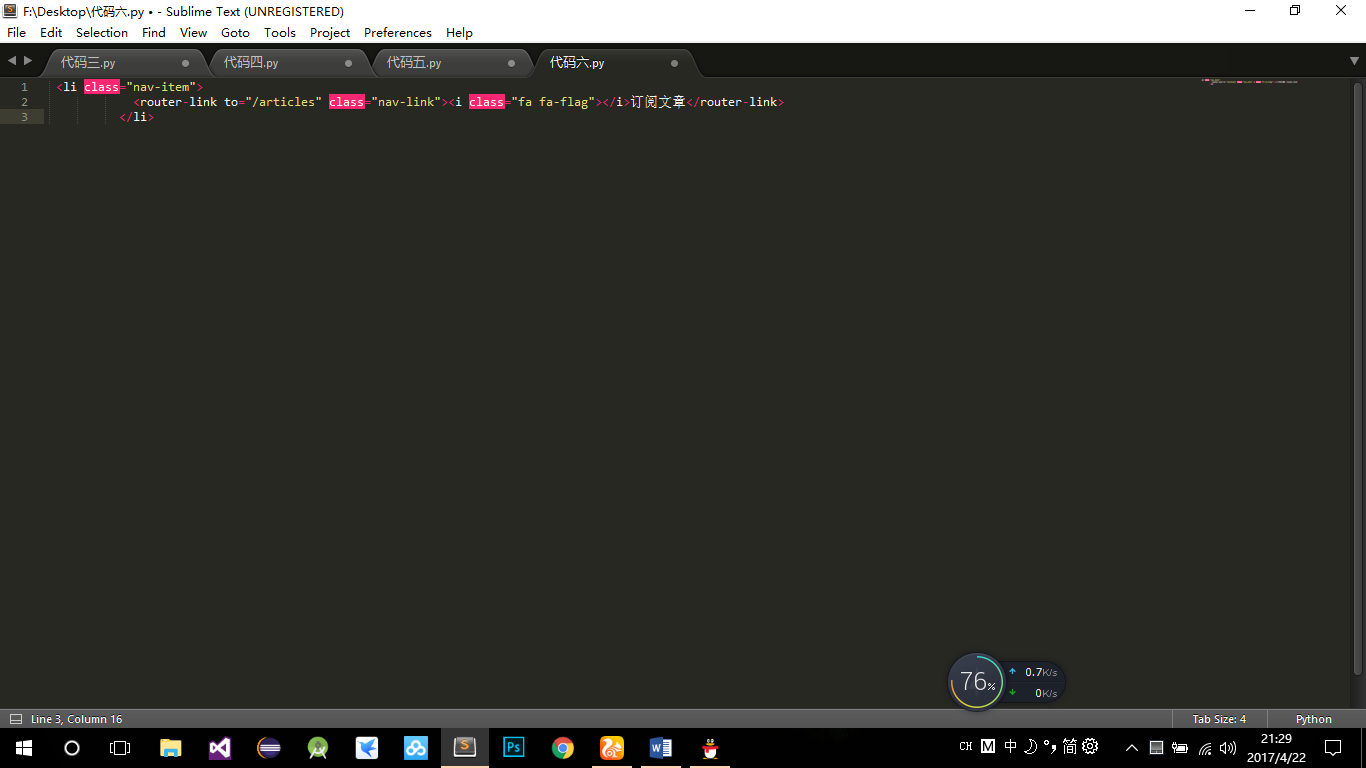
这里需要在vue-router里，添加新的路由，以显示公众号文章列表。  
注意：新的路由path: '/article/:id'是动态的，可以匹配任意公众号文章的视图，比如/article/weixinhao1, /article/weixinhao2...  
另外，取了个别名：name: 'article'，在编程式路由跳转时会用到。（代码2）

我们在导航栏的每个公众号上，添加@click="fetchArticles(mp.weixinhao, mp.mpName)"，来触发获取文章的ajax请求。传给服务器的参数是：weixinhao、headers:token。获取到的数据，存入LocalStorage中。（代码三）

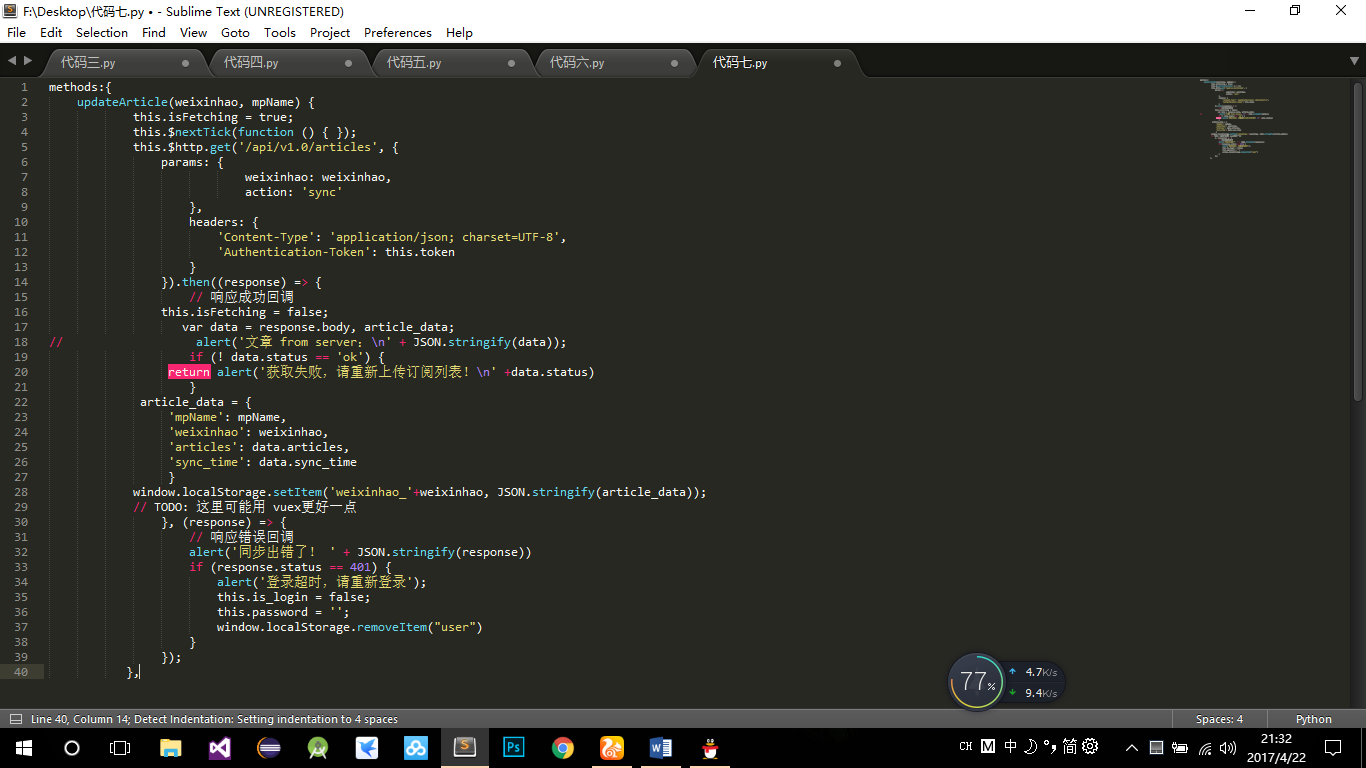
当然，服务器端需要对这个Ajax请求作出响应。检查这个公众号，如果不存在，则需要重新上传订阅列表。如果存在，则查询服务器端对应的文章集合，和上次同步文章列表的时间。这个动作，不会重新去sogou.com抓取新的文章（代码四）

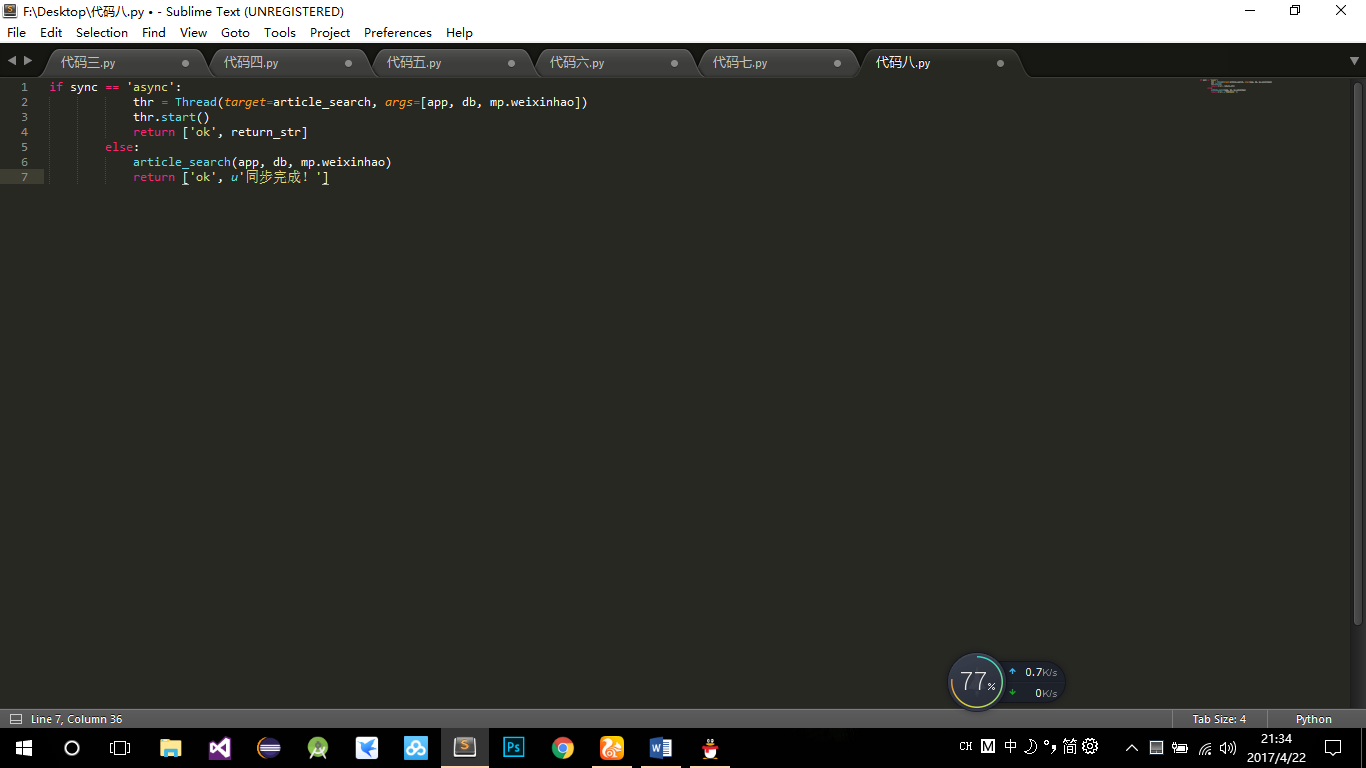
接下来就可以显示公众号文章了articleList用计算属性，读取LocalStorage中的值。（代码五）

1. 点击顶部菜单（订阅文章），左侧显示所有公众号的文章列表，按更新时间排列

如果想查看所有的公众号的文章列表，则先在顶部菜单条上添加路由（代码六）

1. 左侧显示某一公众号文章列表时，点击更新，可以检查是否有最新发表的公众号文章

公众号文章，第一次是在上传订阅列表时更新的。后续再次更新的话，可以由用户来触发。  
我们带入action: 'sync'参数，通知服务器，同步更新就行，不需要异步，本地LocalStorage里，已经有历史数据。（代码七）

爬虫函数，看到参数是sync的话，就不再使用Thead异步抓取了，而是同步等待爬虫结果。客户会看到动态的同步图标，直到更新完毕。（代码八）