三种网络爬虫：

1. 网站地图爬虫
2. ID遍历爬虫
3. 链接爬虫

自我感觉（说不定是我的错觉）：示例网站感觉打开得有点慢，使得python脚本运行得也慢，如果仅仅是为了做试验的话，抽取局部会更好。

## 网站地图爬虫

首先，得科普一下网站地图和robots.txt的基本概念。网站地图，又称站点地图，它就是一个页面，上面放置了网站上需要搜索引擎抓取的所有页面的[链接](https://baike.baidu.com/item/%E9%93%BE%E6%8E%A5" \t "https://baike.baidu.com/item/%E7%BD%91%E7%AB%99%E5%9C%B0%E5%9B%BE/_blank)。robots.txt是robots协议，网站通过Robots协议告诉搜索引擎哪些页面可以抓取，哪些页面不能抓取。 sitemap\_crawler.py代码被改写过，使用了BeautifulSoup。

## ID遍历爬虫

这种方法怎么看都有点取巧的意思，但是也是一种便捷的方式。个人认为了解了解就行。

## 链接爬虫

相对于前两种爬虫，这种应该是更加地通用，但前两种最小化了需要下载的网页数量。作为新手，首先要对代码中用到的python模块有大致的了解，还是一个一个函数来看，了解会更深。

1、Throttle类记录了每个域名上次访问的时间，如果当前时间距离上次访问时间小于指定延时，则执行睡眠操作。我们可以在每次下载之前调用Throttle对爬虫进行限速。

Throttle类的wait()：

|  |
| --- |
| domain = urllib.parse.urlparse(url).netloc  last\_accessed = self.domains.get(domain)  if self.delay > 0 and last\_accessed is not None:  sleep\_secs = self.delay - (datetime.now() - last\_accessed).seconds  if sleep\_secs > 0:  time.sleep(sleep\_secs)  self.domains[domain] = datetime.now() |

urllib.parse.urlparse是url解析，将url拆分为六个组件。这对应于URL的一般结构：scheme://netloc/path;parameters?query#fragment。举个例子：

|  |
| --- |
| urlparse('http://www.cwi.nl:80/%7Eguido/Python.html')  >>>ParseResult(scheme='http',netloc='www.cwi.nl:80',path='/%7Eguido/Python.html', params='', query='', fragment='') |

1. download方法

这个方法其实和前面的download方法差别并不大，只是换了urllib.request.build\_opener，这个类从目前的使用只是为了支持代理。

1. normalize方法

urllib.parse.urldefrag(url)。如果url包含片段标识符，则返回没有片段标识符的url的修改版本，片段标识符作为单独的字符串。如果url中没有片段标识符，则返回未修改的url和空字符串。

片段标识符究竟是什么东西（不了解html一定不知道）？



urllib.parse.urljoin通过将“基本网址”（基本）与另一个网址（url）组合，构造完整（“绝对”）网址。非正式地，这使用基本URL的组件，特别是寻址方案，网络位置和（部分）路径，以在相对URL中提供缺失的组件。简单来说就是，将两个参数拼在一起。

1. same\_domain方法就不需解析了
2. get\_robots方法

urllib.robotparser.RobotFileParser此类提供了读取，解析和回答有关网址的robots.txt文件的问题的方法。