

# 目录

前言	1.1
Python爬虫简介	1.2
裸写Python爬虫代码	1.3
用Python库写爬虫代码	1.4
用Python框架写爬虫代码	1.5
举例	1.6
抓取百度热榜	1.6.1
用Chrome分析逻辑	1.6.1.1
用Python实现爬虫逻辑	1.6.1.2
附录	1.7
参考资料	1.7.1

# 如何用Python写爬虫

- 最新版本: v1.0
- 更新时间: 20200730

## 简介

总结如何用Python去写爬虫，包括如何裸写爬虫代码，如何用Python库去写爬虫，如何用Python爬虫框架写爬虫，并给出实例详细解释具体的操作过程，比如抓取百度首页中百度热榜标题列表。

## 源码+浏览+下载

本书的各种源码、在线浏览地址、多种格式文件下载如下：

### Gitbook源码

- [crifan/use\\_python\\_write\\_spider: 如何用Python写爬虫](#)

### 如何使用此Gitbook源码去生成发布为电子书

详见：[crifan/gitbook\\_template: demo how to use crifan gitbook template and demo](#)

### 在线浏览

- [如何用Python写爬虫 book.crifan.com](#)
- [如何用Python写爬虫 crifan.github.io](#)

### 离线下载阅读

- [如何用Python写爬虫 PDF](#)
- [如何用Python写爬虫 ePUB](#)
- [如何用Python写爬虫 Mobi](#)

### 版权说明

此电子书教程的全部内容，如无特别说明，均为本人原创和整理。其中部分内容参考自网络，均已备注了出处。如有发现侵犯您版权，请通过邮箱联系我 `admin 艾特 crifan.com`，我会尽快删除。谢谢合作。

## 鸣谢

感谢我的老婆陈雪的包容理解和悉心照料，才使得我 crifan 有更多精力去专注技术专研和整理归纳出这些电子书和技术教程，特此鸣谢。

crifan.com，使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,  
powered by Gitbook最后更新：2020-07-30 21:59:28

## Python爬虫简介

爬取你要的数据：爬虫技术中已经解释了爬虫的核心步骤了和相关涉及内容，也提到了很多语言都可以实现爬虫，都能爬取到你要的数据。

不过不同语言有自己的侧重点，而其中爬虫领域，最方便的要数**Python**。Python在爬虫领域，有很多的现成的库和框架可供使用，便于快速高效的实现爬虫的功能。

## 用Python写爬虫的不同方式

正如爬取你要的数据：爬虫技术中所整理的，用Python去写爬虫，也有三种方式：

- 裸写Python爬虫代码
  - 下载
    - python的内置http网络库
      - [urllib](#)
      - [crifanLibPython](#)中的[getUrlRespHtml](#)
    - 提取
      - [re](#)模块
        - [Python中的正则表达式：re模块详解](#)
    - 保存
      - [txt](#)
      - [csv / excel](#)
        - [Python心得：操作CSV和Excel](#)
  - 用各种**Python库**组合去写爬虫代码
    - 下载
      - 选择第三方的、更强大的、更好用的网络库
        - [Python心得：http网络库](#)
          - [Requests](#)
          - [aiohttp](#)
      - 提取
        - [BeautifulSoup](#)
          - [Python专题教程：BeautifulSoup详解](#)
            - v3 -> Python2
            - v4 -> Python3
          - [PyQuery](#)
            - [Python心得：HTML解析库PyQuery](#)
          - [lxml](#)
            - [【记录】Python中尝试用lxml去解析html – 在路上](#)
          - 等等
        - 保存

- [csv / excel](#)
- [PyMySQL](#)
  - 主流关系数据库: MySQL
- [PyMongo](#)
  - 主流文档型数据库: MongoDB
- 等等
- 用爬虫框架去写爬虫代码
  - 常见Python爬虫框架
    - [PySpider](#)
      - Python爬虫框架: PySpider
    - [Scrapy](#)
      - 主流Python爬虫框架: Scrapy
    - 其他相关
      - [【整理】pyspider vs scrapy](#)

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,  
powered by Gitbook最后更新: 2020-07-16 17:52:22

# 裸写Python爬虫代码

TODO:

用python内置urllib去裸写代码，去下载，再用re正则去提取，汽车之家车型车系数据。

## 旧教程

之前已写过一些相关教程，供参考：

- 详解抓取网站，模拟登陆，抓取动态网页的原理和实现（Python, C#等）
  - 【教程】模拟登陆网站之Python版（内含两种版本的完整的可运行的代码） – 在路上
- Python专题教程：抓取网站，模拟登陆，抓取动态网页

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,  
powered by Gitbook最后更新：2020-07-16 17:45:05

# 用Python库写爬虫代码

TODO:

用python的第三方http库，比如requests，去下载，再去用BeautifulSoup去提取，汽车之家车型车系数据。

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,  
powered by Gitbook最后更新：2020-07-16 17:37:14

## 用Python框架写爬虫代码

### 用Python爬虫框架PySpider去爬取汽车之家的车型车系数据

此处举例说明，用PySpider这个Python爬虫框架去爬取汽车之家的车型车系数据

详细过程参见：

[【已解决】写Python爬虫爬取汽车之家品牌车系车型数据 – 在路上](#)

期间包括：

- [【记录】Mac中安装和运行pyspider](#)
- [【已解决】pyspider中如何写规则去提取网页内容](#)
- [【已解决】pyspider中如何加载汽车之家页面中的更多内容](#)
- [【已解决】PySpider如何把json结果数据保存到csv或excel文件中](#)
- [【已解决】PySpider中如何清空之前运行的数据和正在运行的任务](#)

TODO：

把 `autohomeCarData` 代码上传到GitHub，并在此贴出地址

而关于PySpider更多的介绍，详见：

[Python爬虫框架：PySpider](#)

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,  
powered by Gitbook最后更新：2020-07-16 17:37:14

## 举例

下面通过举例例子来说明，去实现同一个爬虫目标，三种不同方式抓包是什么样的。

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,  
powered by Gitbook最后更新： 2020-07-30 20:54:32

## 抓取百度热榜

具体详见：

- 【记录】演示如何实现简单爬虫：用Python提取百度首页中百度热榜内容列表
- 【已解决】用Python代码获取到百度首页源码并提取保存百度热榜内容列表
- 【已解决】Mac中用Chrome开发者工具分析百度首页的百度热榜内容加载逻辑

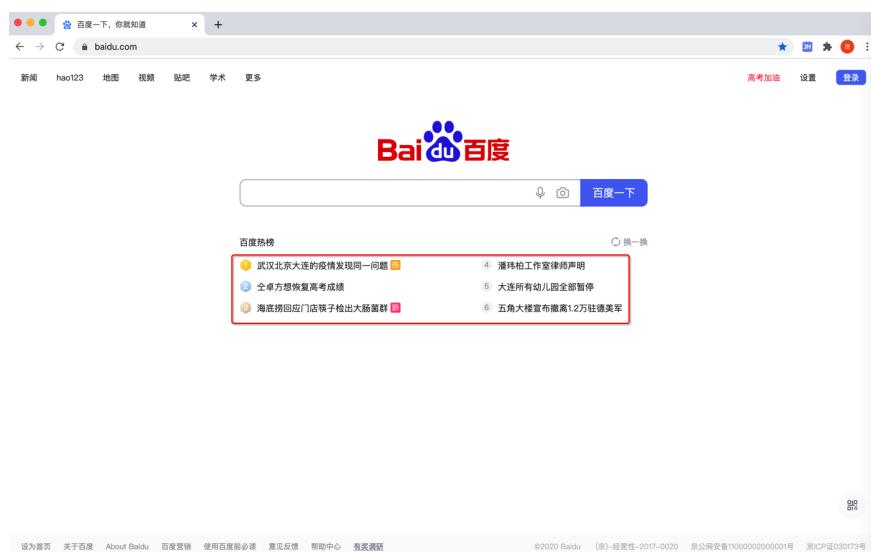
## 目标

爬取百度首页

百度一下，你就知道

<https://www.baidu.com/>

中的 百度热榜的内容的标题的列表：



希望输出的内容：

一个字符串列表：

- 武汉北京大连的疫情发现同一问题
- ...
- 五角大楼宣布撤离1.2万驻德美军

保存格式，暂定为csv文件。

## 先了解基础逻辑

入手之前，先要了解清楚：

- 写爬虫的思路
  - 先去（用工具）分析流程
    - 此处：用Chrome中开发者工具去分析
      - 用Chrome的开发者工具分析百度首页的内容加载的流程
  - 再去用代码实现逻辑
    - 此处：用Python代码实现
    - 要做的事情可以分成3个步骤
      - Download=下载：html网页源码
      - 期间可能涉及
        - 多次利用Chrome的开发者工具去调试页面内容加载逻辑
      - Parse=分析：分析html中源码中我们要的内容的提取规则是什么
        - 需要事先
          - 分析要抓取的内容，所对应的规则
          - 然后用代码实现规则，提取内容
      - Save=保存：把抓取到的内容保存出来

crifan.com，使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,  
powered by Gitbook最后更新：2020-07-30 21:48:42

# 用Chrome分析逻辑

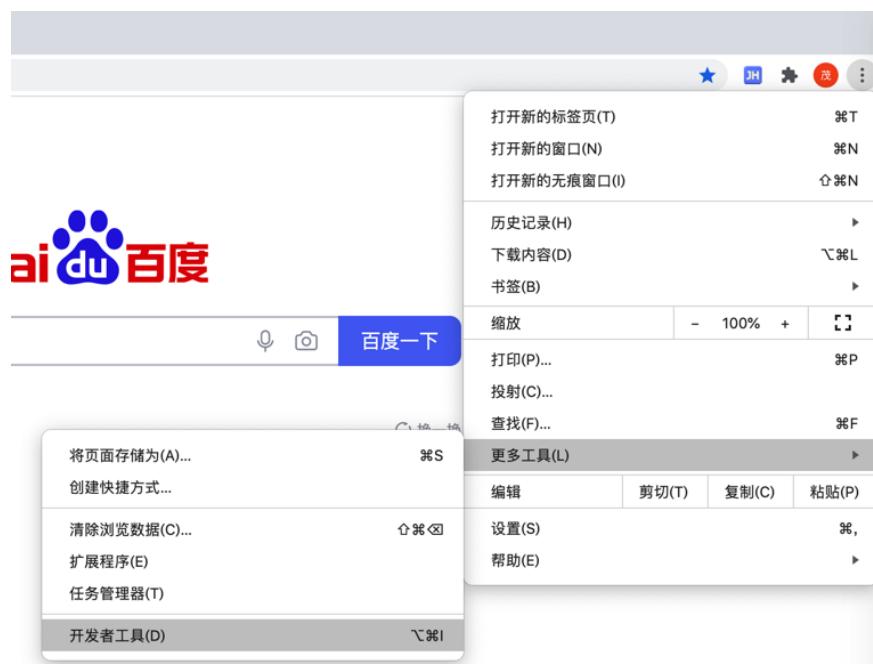
此处去用 Chrome 的 开发者工具 去分析百度首页中加载出百度热榜中的内容的基本逻辑。

先去 Mac 中打开 Chrome 中的 开发者工具 = Developer Tools :

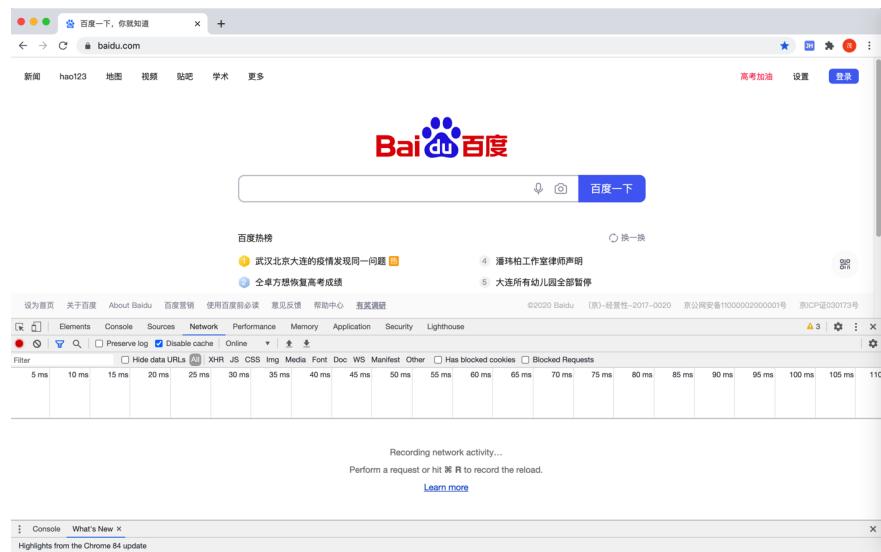
直接用快捷键 (Mac中是) : Option+Command+I

或:

更多 -> 更多工具 -> 开发者工具



打开后效果:



## 用Chrome分析逻辑

先了解一下简单但常用的功能

比如最常见的 查看元素：

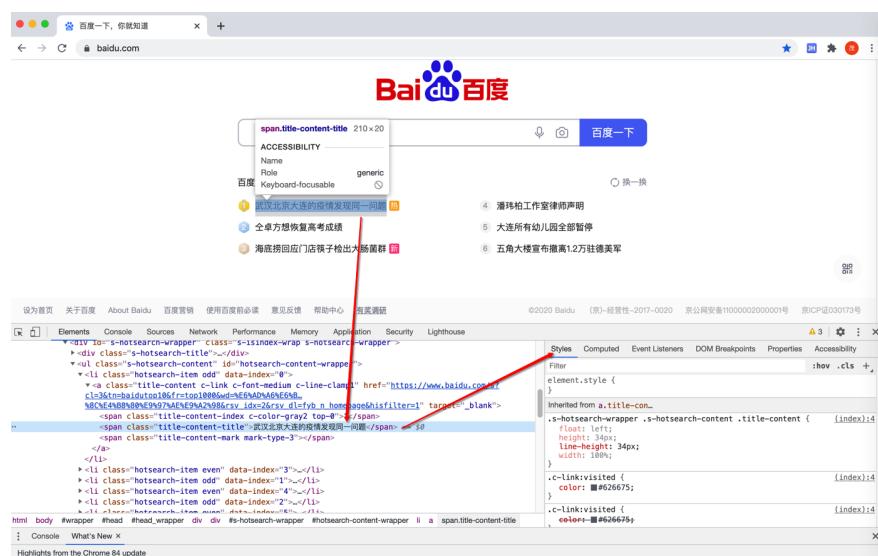
右键 -> 检查



即可看到：

坐标的Elements，表示 显示html网页源码

以及 右边是 css的Styles部分



此处，我们目的是：分析 百度热榜中的内容列表 是如何加载出来的

所以先去根据列表中第一个元素的内容：

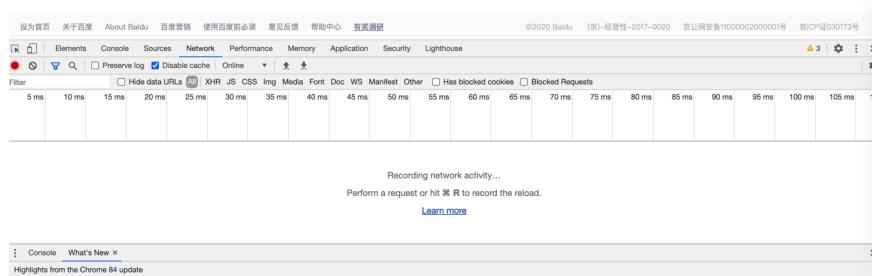
武汉北京大连的疫情发现同一问题

## 用Chrome分析逻辑

去尝试搜索Network部分中的内容（请求或返回的响应中）能否搜索到

步骤：

切换到 Network 一栏：



可见，此处是空的

原因是，在打开开发者工具之前我们就已经加载完毕页面，所以开发者工具没有记录到内容。

先去使得工具能看到当前网页加载所有内容的过程和请求的列表，则思路就是：重新加载网页

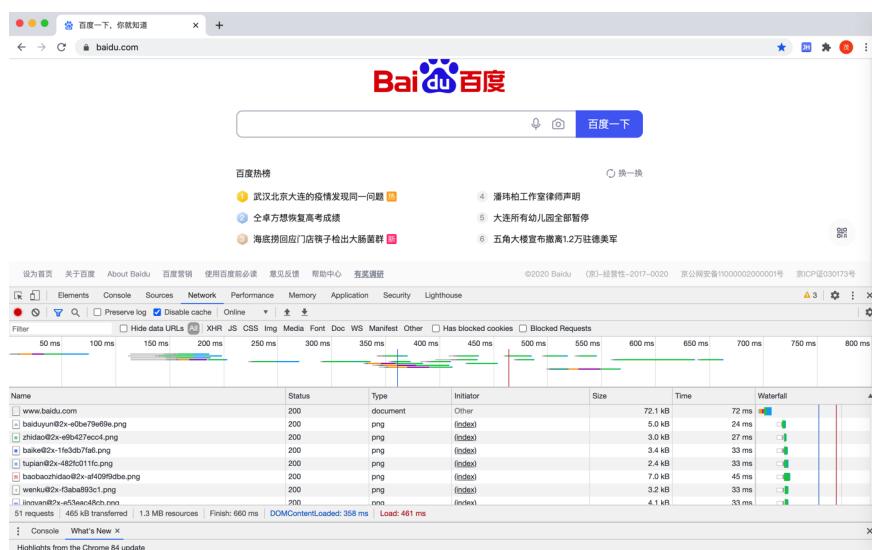
其中也可以看到也有对应提示：

Recording network activity ...

Perform a request or hit Command+R to record the reload

所以去 重新刷新页面

快捷键：Command+R



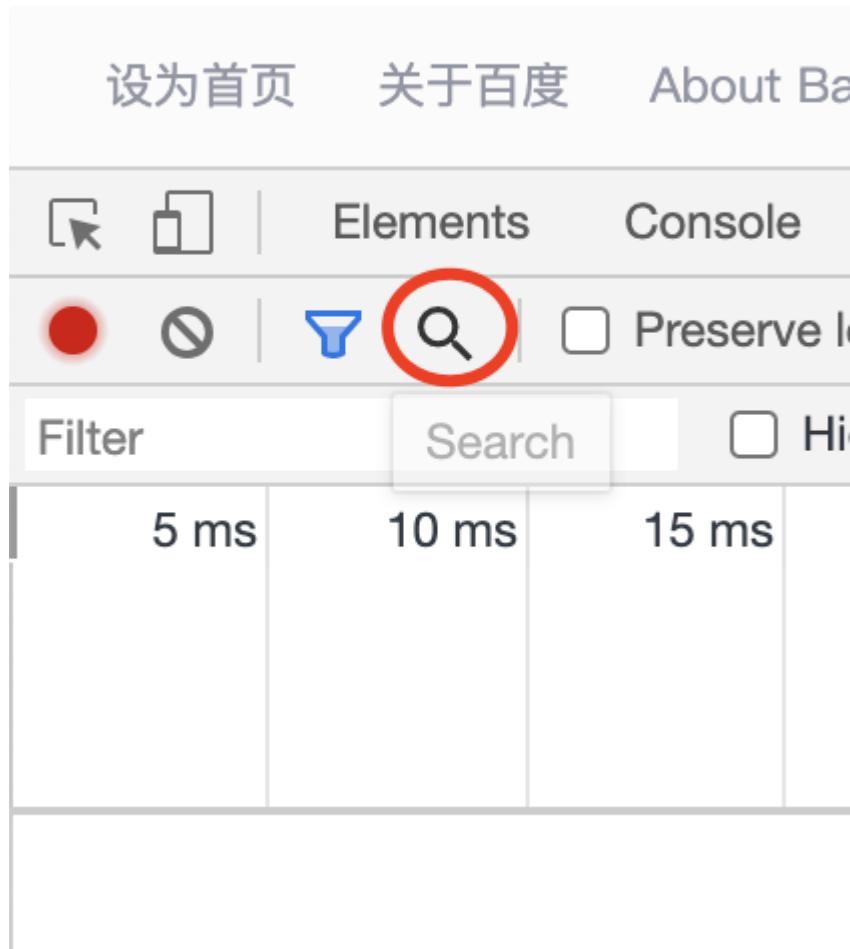
就可以看到网页内容加载的细节和具体每个请求和详情了。

接下来再去找我们的要的内容。

先打开搜索：

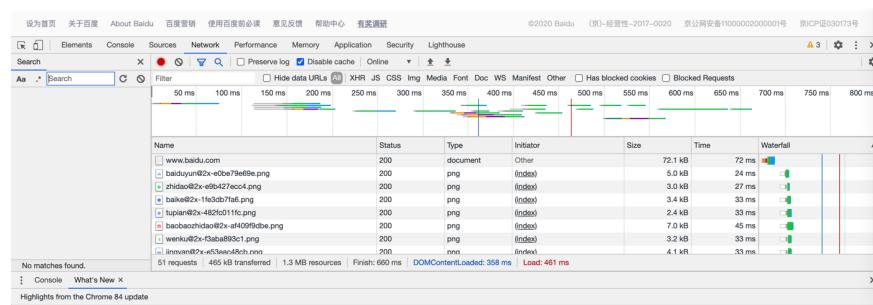
用Chrome分析逻辑

点击 搜索 按钮:



或快捷键: Command+F

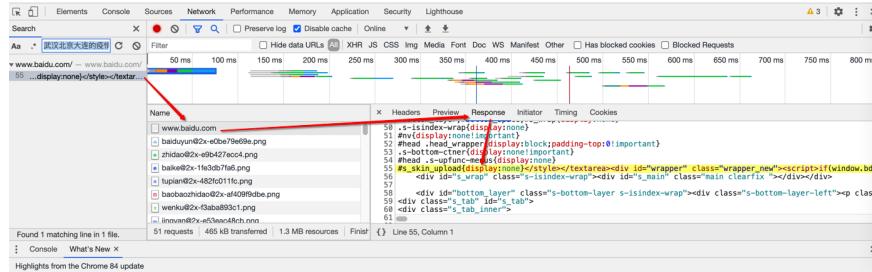
即可打开搜索界面:



输入要搜的内容: 武汉北京大连的疫情发现同一问题 , 并回车触发搜索

此处可以搜到一条记录, 点击会跳转过去:

## 用Chrome分析逻辑



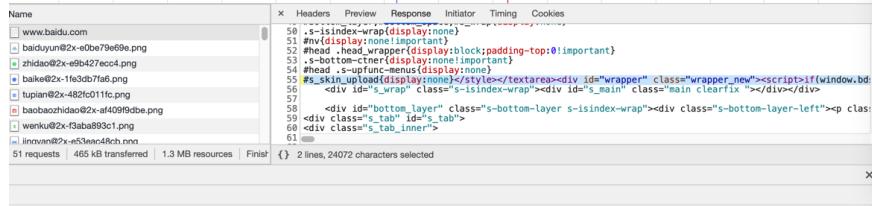
The screenshot shows the Network tab in the Chrome DevTools. A request for 'www.baidu.com' is selected. The 'Response' column displays the HTML source code of the page. A red arrow points to the first few lines of the response, specifically the doctype declaration and the opening `<html>` tag.

此处可以看出是：

[www.baidu.com](http://www.baidu.com)

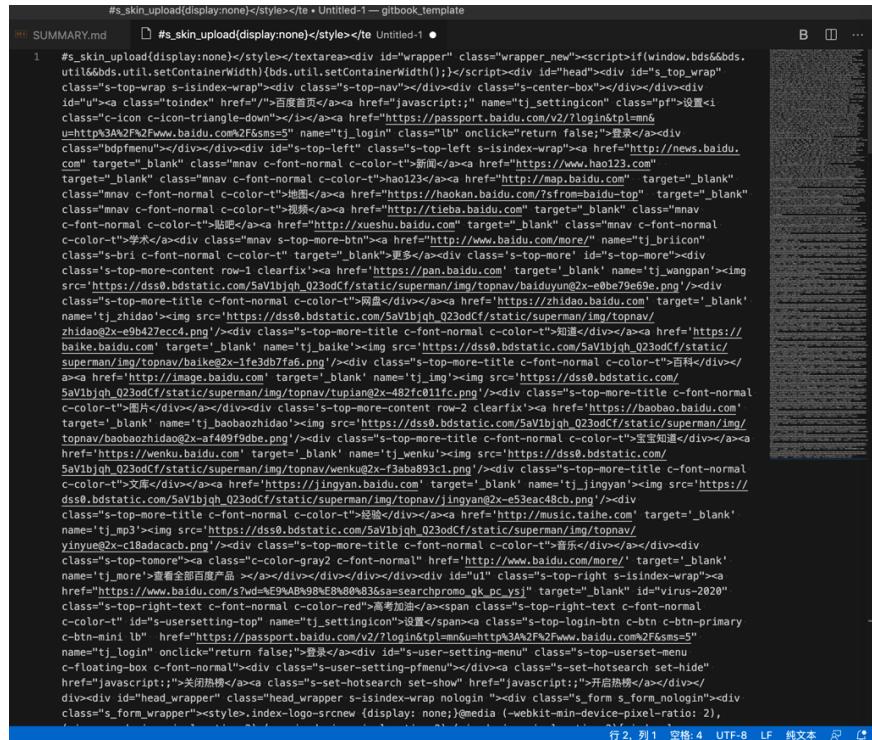
的这条记录中的 Response 部分返回的 html 源码中包含了我们要搜索的内容

双击选中并复制该行：



The screenshot shows the Network tab in the Chrome DevTools. The same request for 'www.baidu.com' is selected. The entire HTML source code is now highlighted in blue, indicating it is selected for copying.

粘贴出来，放到编辑器中去研究内容，比如放到VSCode中：



The screenshot shows a VSCode editor window with a file named 'SUMMARY.md'. The content of the file is the copied HTML source code from the previous step. The code includes various `<a href=>` and `<img src=>` tags pointing to different parts of the Baidu homepage.

找到对应的内容所在位置：

搜 武汉北京大连的疫情发现同一问题，找到2处：

# 用Chrome分析逻辑

links, `<ul>` lists, and `<div>` containers. A specific `<ul>` element is highlighted in yellow, and its attributes are displayed in the right-hand panel."/>

The screenshot shows the source code of a search result page from Baidu. The code is heavily annotated with numerous `<a href="#">` tags pointing to various Baidu pages. Several `<ul>` elements are present, some of which are highlighted in yellow, indicating they are part of the current analysis or selection. The right-hand panel of the developer tools displays detailed information about the selected `<ul>` element, including its ID (`hotsearch-content-wrapper`) and class (`s-hotsearch-content`). This visual evidence supports the claim that the `ul` element is a key component of the search results.

从经验来看，对于是要显示的html内容来说，第一条更像是我们要的

且再仔细看源码发现，前面有 百度热榜 的字样以及前面是 `ul` 的列表：

This screenshot shows the developer tools in Chrome with the page source of a search result from Baidu. A specific `<ul>` element is highlighted in yellow. The right-hand panel provides a detailed view of this `<ul>` element's structure, including its ID (`hotsearch-content-wrapper`) and class (`s-hotsearch-content`). This visual confirmation further supports the analysis that the `ul` element is the primary container for the search results.

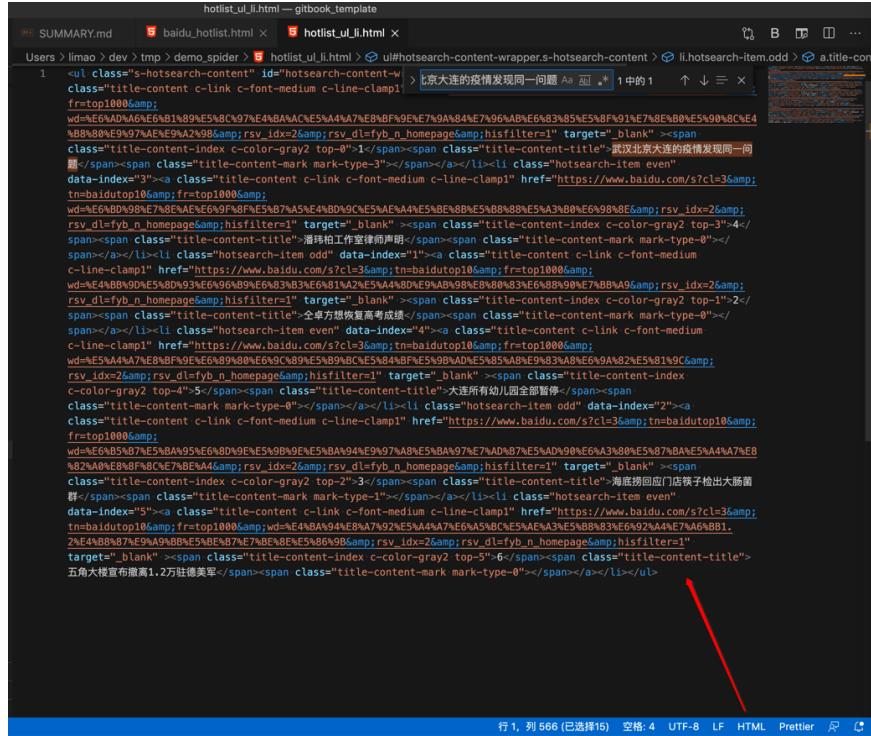
更验证了之前的推断。

把ul这部分html源码拷贝出来：

```
<ul class="s-hotsearch-content" id="hotsearch-content-wrap
```

且为了便于研究，再单独存到另外一个文件，且设置为HTML格式，使得语法高亮，便于阅读：

## 用Chrome分析逻辑



```
hotlist_u_lj.html — gitbook_template
SUMMARY.md baidu_hotlist.html hotlist_u_lj.html #hote... li.hotsearch-item.odd > a.title-cont...
Users > limao > dev > tmp > demo_spider > hotlist_u_lj.html > #hote... li.hotsearch-item.odd > a.title-cont...
1  <ul class="s-hotsearch-content" id="hotsearch-content-w...> 大连的疫情发现同一问题 Aa 1 中的 1 ↑ ↓ ⌂ ...
  class="title-content c-link c-font-medium c-line-clamp1 fr-top1000&...
  wd=%E6%AD%A6%E6%B1%89%E5%8C%97%E4%BA%AC%E5%A4%A7%E7%9A%84%E7%96%AB%E6%83%85%E5%8F%91%E7%8E%...
  %B8%80%E9%97%AE%9A%98&... rsv_idx=2&... rsv_dl=fyb_n_homepage&... hisfilter=1 target=_blank><span class="title-content-index c-color-gray2 top-0"></span><span class="title-content-mark mark-type-0"></span></a><li class="hotsearch-item even" data-index="3"><a class="title-content c-link c-font-medium c-line-clamp1" href="https://www.baidu.com/s?cl=3&...
  tn=baidutop10&... fr=top1000&...
  wd=%E6%BD%90%7BEAE%6%F08F%5%7A5%4%BD%9CE5%AE%4%BD%9CE5%BE%8B%85%88%5%6%98%8&... rsv_idx=2&...
  rsv_dl=fyb_n_homepage&... hisfilter=1 target=_blank><span class="title-content-index c-color-gray2 top-3"></span><span class="title-content-title">潘玮柏工作室律师声明</span><span class="title-content-mark mark-type-0"></span></a><li class="hotsearch-item odd" data-index="4"><a class="title-content c-link c-font-medium c-line-clamp1" href="https://www.baidu.com/s?cl=3&...
  tn=baidutop10&... fr=top1000&...
  wd=%5%A4%7E8%BF%9E%6%89%88%9E%9C%89%84%BF%ES%9%AD%ES%85%8%9E%98%8%8E%9A%81%9C&... rsv_idx=2&...
  rsv_dl=fyb_n_homepage&... hisfilter=1 target=_blank><span class="title-content-index c-color-gray2 top-4"></span><span class="title-content-title">大连所有幼儿院全部暂停</span><span class="title-content-mark mark-type-0"></span></a><li class="hotsearch-item odd" data-index="2"><a class="title-content c-link c-font-medium c-line-clamp1" href="https://www.baidu.com/s?cl=3&...
  tn=baidutop10&... fr=top1000&...
  wd=%E6%85%87%5%8A%95%6%8D%9E%5%9B%9E%5%8A%94%EP%97%AB%ES%BA%97%7E%AD%0%7E%5%AD%0%6%...
  %82%8%0%8B%8%7C%7B%8A%4&... rsv_idx=2&... rsv_dl=fyb_n_homepage&... hisfilter=1 target=_blank><span class="title-content-index c-color-gray2 top-2"></span><span class="title-content-title">海底捞回应门店孩子检出大肠菌群</span><span class="title-content-mark mark-type-0"></span></a><li class="hotsearch-item even" data-index="5"><a class="title-content c-link c-font-medium c-line-clamp1" href="https://www.baidu.com/s?cl=3&...
  tn=baidutop10&... fr=top1000&...
  wd=%E4%BA%94%80%92%5%AA%7E%6%85%8C%5%AE%AA%3%5%88%83%6%0%2%4%4%7%A6%8B1...
  %2%4%88%8%7%9A%9B%5%8E%7%BE%8E%5%86%9B&... rsv_idx=2&... rsv_dl=fyb_n_homepage&... hisfilter=1 target=_blank><span class="title-content-index c-color-gray2 top-5"></span><span class="title-content-title">五角大楼宣布撤离1.2万驻德美军</span><span class="title-content-mark mark-type-0"></span></a></li>
```

然后去研究代码，稍微懂点html的，即可了解其基本逻辑：

此处第一个li的元素：武汉北京大连的疫情发现同一问题

对应源码就是：

```
<li class="hotsearch-item odd" data-index="0"><a class="titl...
```

其后的其他几个热榜节点的格式也是类似的，只不过是

- hotsearch-item 是 even
- data-index=1

等等细节不同：

```
<li class="hotsearch-item even" data-index="3"><a class="titl...
```

所以，对于上述内容，此处研究出来的逻辑是：

如果能正常获取到

<https://www.baidu.com/>

的html源码，且内容已加载完毕的情况下

则直接去使用此简化规则去匹配内容：

```
<span class="title-content-title">xxx</span>
```

## 用Chrome分析逻辑

其中xxx是中文字串，是此处希望找到的内容的标题。

说明：自己要确保此规则不会和导致误判，多了或少了，即匹配出其他的额外的不想要的内容，或漏了某些想要的内容。

当误判时，就需要加上其他限定条件，比如此处的：

```
父级节点是：li中 class="hotsearch-item even" 或 "hotsearch-item odd"
```

对于上述简化规则，再去用代码实现，提取要的内容：

(1) Python中re正则

```
contentTitleP = '<span\s+class="title-content-title">( ?P<content>.*? )</span>'
```

(2) Python的用于解析html的第三方库BeautifulSoup

```
allTitleSoupList = soup.find_all("span", attrs={"class": "title-content-title"})
```

至此，要抓取的内容的提取规则，已分析完毕。

接下来，就是回头再去确保，可以正常获取到

<https://www.baidu.com/>

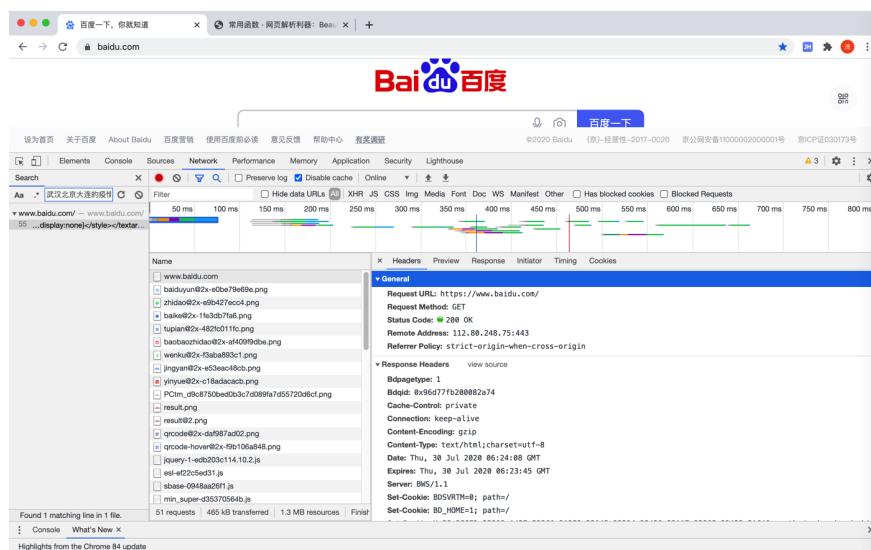
的源码，即可。

对此，往往不太容易一次性就很轻松的获取各个网站的网页源码。

所以一般的逻辑是：直接去写代码，然后出现问题，变调试，变优化代码，直到最终获取到源码

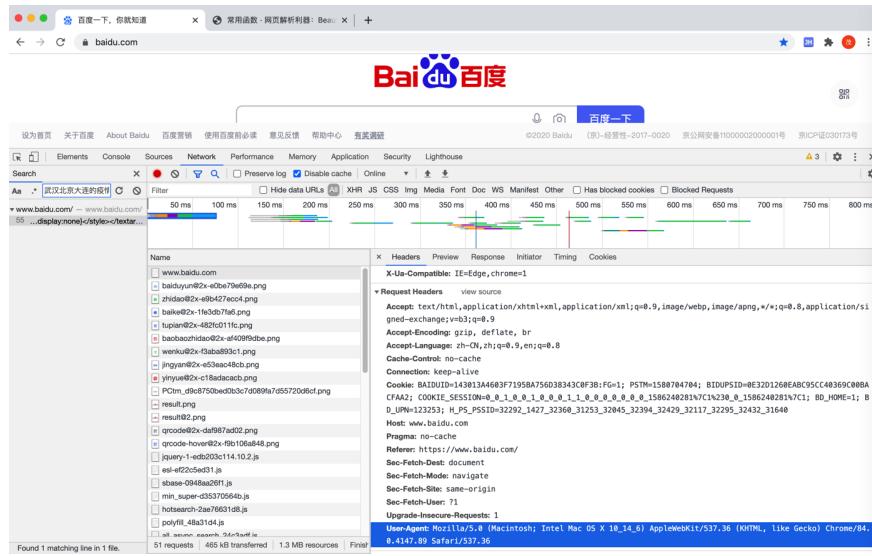
期间继续调试逻辑：

切换到Headers界面：



## 用Chrome分析逻辑

找到 Request Headers 中的 User-Agent 部分：



拷贝出来是：

```
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6)
```

用于后续代码中使用。

crifan.com, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved,  
powered by Gitbook最后更新：2020-07-30 21:26:02

# 用Python实现爬虫逻辑

通过前面用Chrome的开发者工具分析完逻辑后，再去用Python代码实现爬取的全部逻辑：

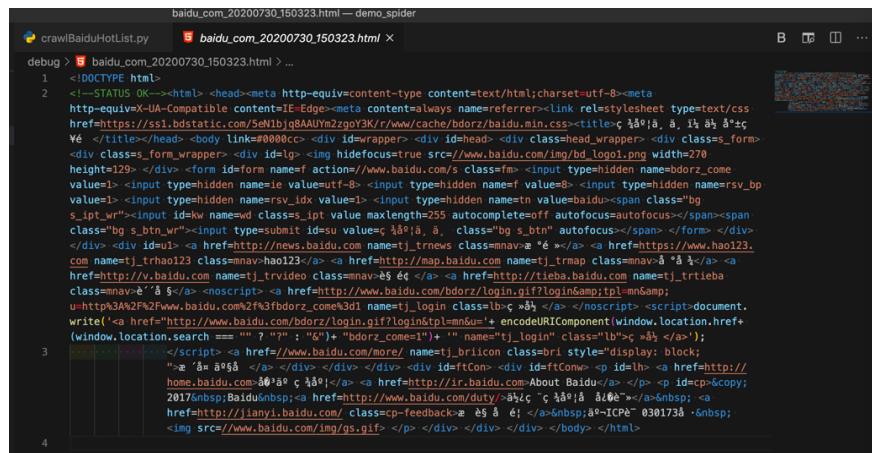
此处记录完整过程。

此处先写出核心代码：

```
import requests

baiduResp = requests.get("https://www.baidu.com/")
baiduHtml = baiduResp.text
curDatetimeStr = getCurDatetimeStr()
baiduHtmlFilename = "baidu_com_%s.html" % curDatetimeStr
baiduHtmlFullPath = os.path.join(DebugRoot, baiduHtmlFilename)
saveTextToFile(baiduHtmlFullPath, baiduHtml)
```

调试返回的html源码是：



The screenshot shows a browser window with the URL `baidu_com_20200730_150323.html`. The page content is the raw HTML source code of a Baidu search results page. The code is heavily encoded with Chinese characters and includes various meta tags, form fields, and links. Notably, it lacks the structured search results we expected, such as the '百度热榜' section.

很明显：没有包含我们希望的 百度热榜 的内容，且连其他的中文，比如 百度一下 之类的字眼都看不到

那么根据经验，需要加其他参数，甚至额外逻辑，才可能获取完整的html代码

而最先要去加上的，就是 User-Agent

先去回去用 Chrome的开发者工具，看看当前的User-Agent是啥，找到值。

再去把User-Agent部分，加到requests中：

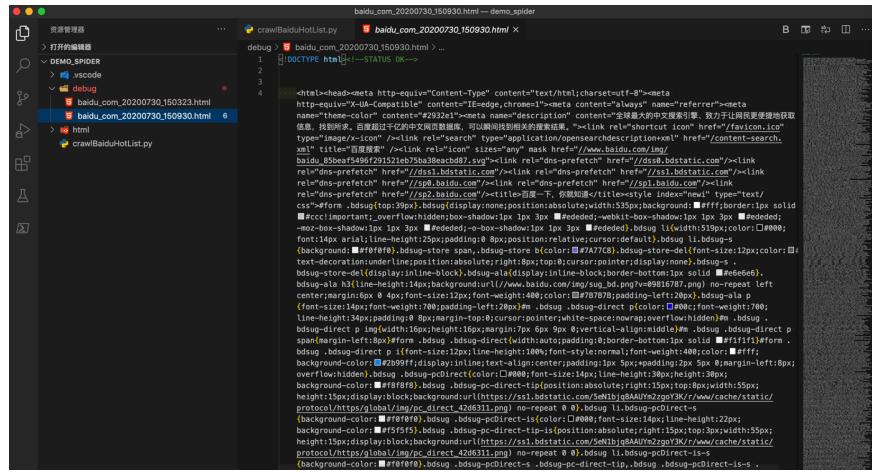
## 用Chrome分析逻辑

```
UserAgent_Chrome_Mac = "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.106 Safari/537.36"

curHeaders = {
    "User-Agent": UserAgent_Chrome_Mac,
}

baiduResp = requests.get("https://www.baidu.com/", headers=curHeaders)
```

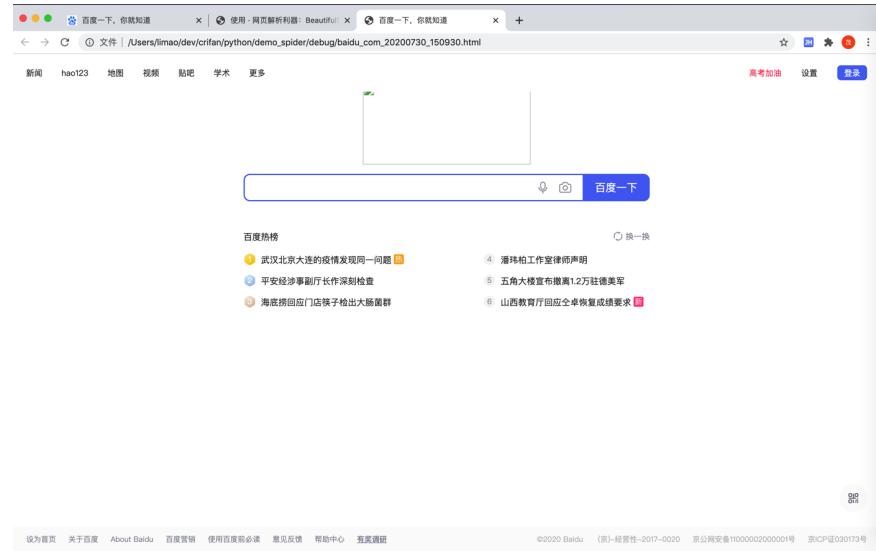
再去试试，此处我们很幸运，立刻就可以返回，大量的内容：



```
baidu.com_20200730_150930.html > curl -H 'User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.106 Safari/537.36' https://www.baidu.com/
```

看起来就是正确的，估计包含我们要找到的 百度热榜 的内容了。

另外顺带，直接用浏览器打开此处抓取到的本地的离线的html，看看效果是什么样的：

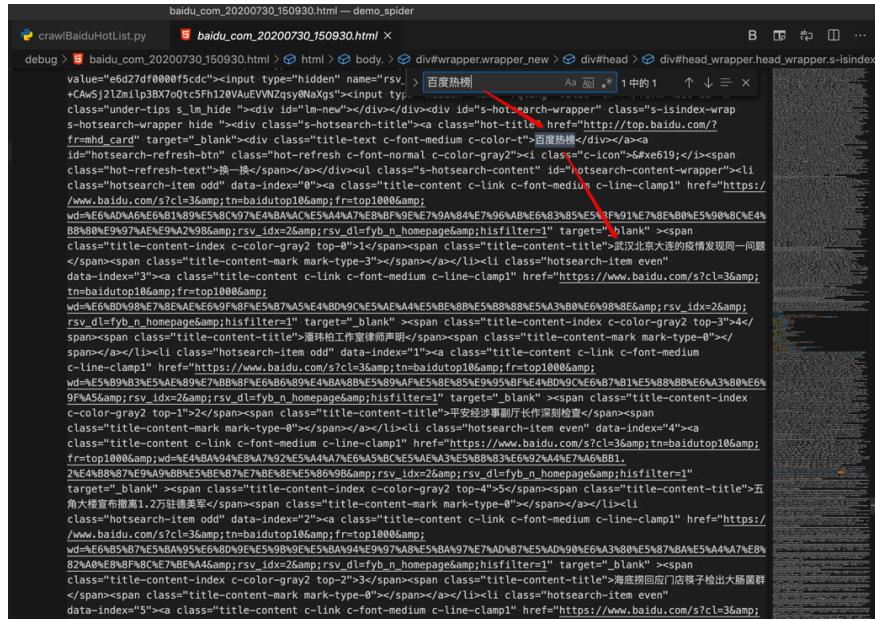


可见（由于本身页面简单不复杂），除了首页logo外，页面效果和浏览器打开的基本一致。

也验证了前面的推测，确认就是完整的源码了。

去搜索 百度热榜

## 用Chrome分析逻辑



```
baidu_com_20200730_150930.html — demo_spider
debug > baidu_com_20200730_150930.html > html > body > div#wrapper.wrapper_new > div#head > div#head_wrapper.head_wrapper.s-isindex
values="e6d27df0000f5cdc"><input type="hidden" name="rsv_t" value="百度热搜" /> 1 中的 1 ↑ ↓ ⌂
+ChWsjzLmipBX0tCfh120AuEVNzqsyNaKg><input type="hidden" name="rsv_t" value="百度热搜" />
class="under-tips s_lm_hide"><div id="lm-new"></div></div><div id="s-hot-search-wrapper" class="s-isindex-wrap">
<s-hotsearch-wrapper hide="0" data-index="0"><a class="hot-title" href="http://top.baidu.com/?fr=mhd_card" target="_blank">百度热搜</a>
<div class="hot-refresh-text">换一换</div><ul class="s-hotsearch-content" id="hotsearch-content-wrapper"><li
class="hotsearch-item odd" data-index="0"><a class="title-content c-link c-font-medium c-color-gray2" href="https://www.baidu.com/s?cl=3&tn=baidutop10&amp;fr=top100&amp;wd=%E6%AD%A6%E6%91%89%E5%BC%97%E4%BA%AC%E5%A4%A7%E9%84%E7%9A%84%E5%83%85%ES%3F%91%E7%8E%80%E5%99%8C%24%88%80%96%97%AE%E9%94%29&amp;rsv_idx=2&amp;rsv_dlf=yb_n_homepage&hisfilter=1" target="_blank">武汉北京大连的疫情发现同一问题</a><span class="title-content-index c-color-gray2 top-0"><span><span class="title-content-mark mark-type-3"></span></span></span></span></li><li class="hotsearch-item even"
data-index="3"><a class="title-content c-link c-font-medium c-line-clamp1" href="https://www.baidu.com/s?cl=3&tn=baidutop10&amp;fr=top100&amp;wd=%E6%BD%99%7BEAE%E6%9F%88%E5%B7%A5%4E%BD%9C%E5%AE%4E%5B%8B%E5%80%8E%88%E5%A3%80%E6%98%8E&amp;rsv_idx=1&fyb_n_homepage&amp;hisfilter=1" target="_blank">换一换</a><span class="title-content-index c-color-gray2 top-3"><span><span class="title-content-mark mark-type-0"></span></span></span></span></li><li class="hotsearch-item odd" data-index="1"><a class="title-content c-link c-font-medium c-line-clamp1" href="https://www.baidu.com/s?cl=3&tn=baidutop10&amp;fr=top100&amp;wd=%E5%89%83%5AE89%7BB8%8F%6B%86%89%4FAE%8E%85%9AE%4E%4AB%0C%6E6B%7A1%E5%88%8B%6A3%80%6E9%FA5&amp;rsv_idx=2&amp;rsv_dlf=yb_n_homepage&amp;hisfilter=1" target="_blank">潘伟柏工作律师声明</a><span class="title-content-mark mark-type-0"></span></span></span></span></li><li class="hotsearch-item even" data-index="4"><a class="title-content c-link c-font-medium c-line-clamp1" href="https://www.baidu.com/s?cl=3&tn=baidutop10&amp;fr=top100&amp;wd=%E4%BD%A9%94%EB%A7%92%EA%4A%7E%EA%9B%C5%AA%3%E5%88%80%3%F6%92%AA%4E%7%A6%BB1%2E4%88%87%9A%8B%5BE%BF%7BE8%E5%86%9B&amp;rsv_idx=2&amp;rsv_dlf=yb_n_homepage&amp;hisfilter=1" target="_blank">五角大楼宣布撤离1.2万驻德美军</a><span class="title-content-mark mark-type-0"></span></span></span></span></li><li class="hotsearch-item odd" data-index="2"><a class="title-content c-link c-font-medium c-line-clamp1" href="https://www.baidu.com/s?cl=3&tn=baidutop10&amp;fr=top100&amp;wd=%E6%85%8B%5BA%9E%5B%8D%9E%5B%8A%94%E9%7A%BE%5B%8A%97%E7%AD%87%E6%A3%80%E5%87%BA%5A%47%E8%82%A0%8B%8F%8C%7BE%AA&amp;rsv_idx=2&amp;rsv_dlf=yb_n_homepage&amp;hisfilter=1" target="_blank">平安经济副厅长作深刻检查</a><span class="title-content-index c-color-gray2 top-2"><span><span class="title-content-mark mark-type-0"></span></span></span></span></li><li class="hotsearch-item even" data-index="5"><a class="title-content c-link c-font-medium c-line-clamp1" href="https://www.baidu.com/s?cl=3&tn=baidutop10&amp;fr=top100&amp;wd=%E5%89%83%5AE89%7BB8%8F%6B%86%89%4FAE%8E%85%9AE%4E%4AB%0C%6E6B%7A1%E5%88%8B%6A3%80%6E9%FA5&amp;rsv_idx=2&amp;rsv_dlf=yb_n_homepage&amp;hisfilter=1" target="_blank">和平经济副厅长作深刻检查</a><span class="title-content-mark mark-type-0"></span></span></span></span></li>
```

的确可以找到我们要的内容。

## 更多情况下获取完整的全部源码要难很多

此处只加了 User-Agent 就可以返回所需全部的完整的页面源码，是很幸运的。

因为随着web技术发展，反扒技术进步，稍微有点点技术含量的公司所做的web页面，尤其是页面逻辑复杂的，涉及到多个页面的

想要获取完整页面源码，往往都需要加上其他更多参数，才（可）能获取到期望的返回结果。

而关于 其他更多参数，常见的一些有：

- 简单的
  - Accept
  - Accept-Encoding
  - Accept-Language
  - Host
  - Referer
- 复杂的
  - Cookie
    - 很多值，很难获取到（搞懂生成的逻辑）

所以可以接着，去用之前分析出的规则，去解析内容了：

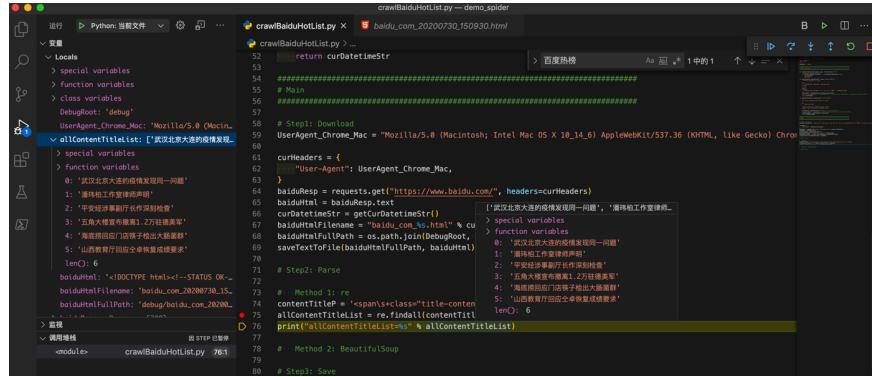
先去用正则re

核心代码：

用Chrome分析逻辑

```
# Step2: Parse
# Method 1: re
contentTitleP = '<span\s+class="title-content-title">(?P<content>.*?</span>)'
allContentTitleList = re.findall(contentTitleP, baiduHtml)
print("allContentTitleList=%s" % allContentTitleList)
```

即可匹配出要的热榜标题列表：

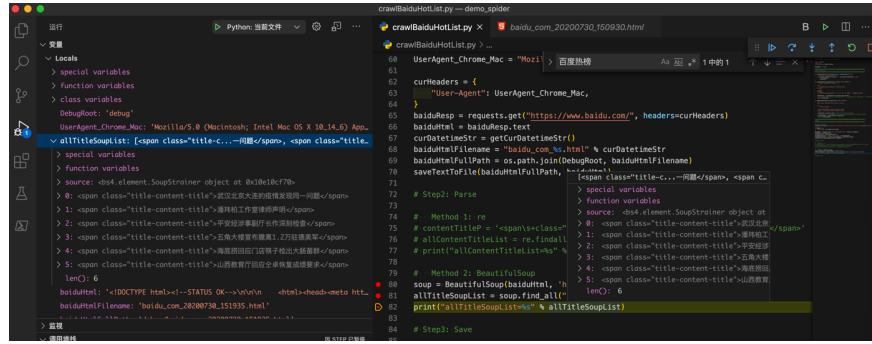


```
# Step2: Parse
# Method 1: re
contentTitleP = '<span\s+class="title-content-title">(?P<content>.*?</span>)'
allContentTitleList = re.findall(contentTitleP, baiduHtml)
print("allContentTitleList=%s" % allContentTitleList)
```

另外，再去换用第三方HTML解析库 BeautifulSoup 试试：

```
# Method 2: BeautifulSoup
soup = BeautifulSoup(baiduHtml, 'html.parser')
allTitleSoupList = soup.find_all("span", attrs={"class": "title-content-title"})
print("allTitleSoupList=%s" % allTitleSoupList)
```

也是可以找到要的内容的：



```
# Step2: Parse
# Method 2: BeautifulSoup
soup = BeautifulSoup(baiduHtml, 'html.parser')
allTitleSoupList = soup.find_all("span", attrs={"class": "title-content-title"})
print("allTitleSoupList=%s" % allTitleSoupList)
```

再去加上代码，把 soup 的 string 保存出来：

```
allContentTitleList = []
for eachTitleSoup in allTitleSoupList:
    titleStr = eachTitleSoup.string
    allContentTitleList.append(titleStr)
print("allContentTitleList=%s" % allContentTitleList)
```

就是我们要的列表了：

用Chrome分析逻辑

```
allContentTitleList=['武汉北京大连的疫情发现同一问题', '潘玮柏工作
```

至此下载和提取都完成了

接着去保存内容，如前面假设，比如保存到csv文件中

```
def saveToCsvByDictList(csvDictList, outputPath):
    # generate csv headers from dict list
    firstItemDict = csvDictList[0]
    csvHeaders = list(firstItemDict.keys())
    with codecs.open(outputPath, "w", "UTF-8") as outC:
        csvDictWriter = csv.DictWriter(outCsvFp, fieldnames=csvHeaders)

        # write header by inner function from fieldnames
        csvDictWriter.writeheader()

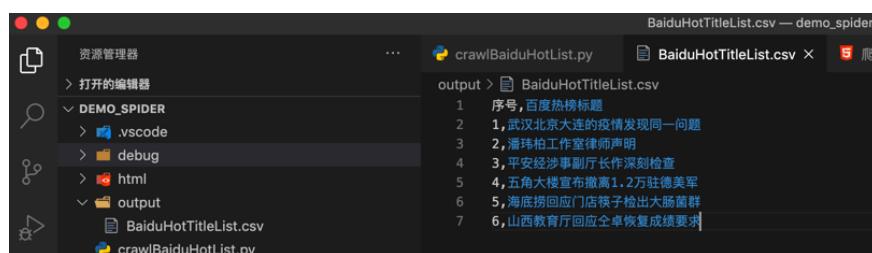
        for eachRowDict in csvDictList:
            csvDictWriter.writerow(eachRowDict)

    # save to csv
OutputCsvHeader = ["序号", "百度热榜标题"]
OutputCsvFilename = "BaiduHotTitleList.csv"
OutputCsvFullPath = os.path.join(OutputRoot, OutputCsvFilename)

outputCsvDictList = []
for curIdx, eachTitle in enumerate(allContentTitleList):
    curNum = curIdx + 1
    csvDict = {
        "序号": curNum,
        "百度热榜标题": eachTitle
    }
    outputCsvDictList.append(csvDict)

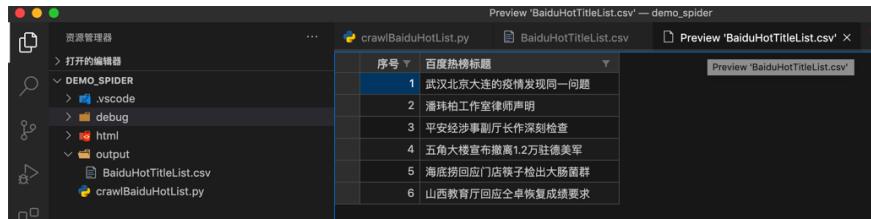
saveToCsvByDictList(outputCsvDictList, OutputCsvFullPath)
```

即可保存出我们要的csv文件：



以及，用VSCode中csv插件去以列表方式查看的效果：

## 用Chrome分析逻辑



和Mac中的预览效果：



至此，最终完整代码是：

```
# Function: Demo how use Python crawl baidu.com 百度热榜
# Author: Crifan
# Update: 20200730

import os
import codecs
from datetime import datetime, timedelta

import requests
import re
from bs4 import BeautifulSoup

import csv

DebugRoot = "debug"
OutputRoot = "output"

#####
# Utils Functions
#####

def createFolder(folderFullPath):
    """
        create folder, even if already existed
        Note: for Python 3.2+
    """
    os.makedirs(folderFullPath, exist_ok=True)

def saveTextToFile(fullFilename, text, fileEncoding="utf-8"):
    """
        save text content into file
    """
    with codecs.open(fullFilename, 'w', encoding=fileEncoding) as fp:
        fp.write(text)
        fp.close()

def datetimeToStr(inputDatetime, format="%Y%m%d_%H%M%S"):
    """
        Convert datetime to string
    """

    Args:
        inputDatetime (datetime): datetime value
    Returns:
        str
    Raises:
    Examples:
        datetime.datetime(2020, 4, 21, 15, 44, 13, 2000) ->
    """
    datetimeStr = inputDatetime.strftime(format)
    # print("inputDatetime=%s -> datetimeStr=%s" % (inputDatetime, datetimeStr))
    return datetimeStr

def getCurDatetimeStr(outputFormat="%Y%m%d_%H%M%S"):
```

用Chrome分析逻辑

```
"""
    get current datetime then format to string

    eg:
        20171111_220722

    :param outputFormat: datetime output format
    :return: current datetime formatted string
"""

curDatetime = datetime.now() # 2017-11-11 22:07:22.705
# curDatetimeStr = curDatetime.strftime(format=outputFormat)
curDatetimeStr = datetimeToStr(curDatetime)
return curDatetimeStr

def saveToCsvByDictList(csvDictList, outputPath):
    # generate csv headers from dict list
    firstItemDict = csvDictList[0]
    csvHeaders = list(firstItemDict.keys())
    with codecs.open(outputPath, "w", "UTF-8") as outCsvFp:
        csvDictWriter = csv.DictWriter(outCsvFp, fieldnames=csvHeaders)

        # write header by inner function from fieldnames
        csvDictWriter.writeheader()

        for eachRowDict in csvDictList:
            csvDictWriter.writerow(eachRowDict)

def saveToCsvByHeaderAndList(csvHeaderList, csvRowListList):
    with codecs.open(outputPath, "w", "UTF-8") as outCsvFp:
        csvWriter = csv.writer(outCsvFp)

        # write header from list
        csvWriter.writerow(csvHeaderList)

        # type1: write each row
        # for eachRowList in csvRowListList:
        #     csvWriter.writerow(eachRowList)

        # type2: write all rows
        csvWriter.writerows(csvRowListList)

#####
# Main
#####

createFolder(DebugRoot)
createFolder(OutputRoot)

# Step1: Download
UserAgent_Chrome_Mac = "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/61.0.3163.100 Safari/537.36"
```

用Chrome分析逻辑

```
curHeaders = {
    "User-Agent": UserAgent_Chrome_Mac,
}
baiduResp = requests.get("https://www.baidu.com/", headers=curHeaders)
baiduHtml = baiduResp.text
curDatetimeStr = getCurDatetimeStr()
baiduHtmlFilename = "baidu_com_%s.html" % curDatetimeStr
baiduHtmlFullPath = os.path.join(DebugRoot, baiduHtmlFilename)
saveTextToFile(baiduHtmlFullPath, baiduHtml)

# Step2: Parse

# Method 1: re
# contentTitleP = '<span\s+class="title-content-title">(.*?)</span>'
# allContentTitleList = re.findall(contentTitleP, baiduHtml)
# print("allContentTitleList=%s" % allContentTitleList)

# Method 2: BeautifulSoup
soup = BeautifulSoup(baiduHtml, 'html.parser')
allTitleSoupList = soup.find_all("span", attrs={"class": "title-content-title"})
print("allTitleSoupList=%s" % allTitleSoupList)
allContentTitleList = []
for eachTitleSoup in allTitleSoupList:
    titleStr = eachTitleSoup.string
    allContentTitleList.append(titleStr)
print("allContentTitleList=%s" % allContentTitleList)

# Step3: Save

# save to csv
OutputCsvHeader = ["序号", "百度热榜标题"]
OutputCsvFilename = "BaiduHotTitleList.csv"
OutputCsvFullPath = os.path.join(OutputRoot, OutputCsvFilename)

outputCsvDictList = []
for curIdx, eachTitle in enumerate(allContentTitleList):
    curNum = curIdx + 1
    csvDict = {
        "序号": curNum,
        "百度热榜标题": eachTitle
    }
    outputCsvDictList.append(csvDict)

saveToCsvByDictList(outputCsvDictList, OutputCsvFullPath)
print("Completed save data to %s" % OutputCsvFullPath)
```

即可实现完整的爬虫功能：

- 下载百度首页源码
- 提取所需的百度热榜的标题内容

- 保存内容为csv格式

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,  
powered by Gitbook最后更新: 2020-07-30 21:47:15

## 附录

下面列出相关参考资料。

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,  
powered by Gitbook最后更新: 2020-07-16 17:37:14

## 参考资料

- 爬取你要的数据：爬虫技术
- crifanLibPython
- getUrlRespHtml
- Python中的正则表达式：re模块详解
- Python心得：操作CSV和Excel
- Python心得：http网络库
- Python专题教程：BeautifulSoup详解
- Python心得：HTML解析库PyQuery
- 【记录】Python中尝试用lxml去解析html – 在路上
- 主流关系数据库：MySQL
- 主流文档型数据库：MongoDB
- Python爬虫框架：PySpider
- 主流Python爬虫框架：Scrapy
- 【整理】pyspider vs scrapy
- 【教程】模拟登陆网站 之 Python版（内含两种版本的完整的可运行的代码） – 在路上
- Python专题教程：抓取网站，模拟登陆，抓取动态网页
- 【整理】各种浏览器中的开发人员工具Developer Tools：IE9的F12，Chrome的Ctrl+Shift+J，Firefox的Firebug
- 【总结】浏览器中的开发人员工具（IE9的F12和Chrome的Ctrl+Shift+I） -网页分析的利器
- 【教程】如何利用IE9的F12去分析网站登陆过程中的复杂的（参数，cookie等）值（的来源）
- 【教程】手把手教你如何利用工具(IE9的F12)去分析模拟登陆网站(百度首页)的内部逻辑过程
- app抓包利器：Charles
- 【已解决】写Python爬虫爬取汽车之家品牌车系车型数据 – 在路上
- 【记录】Mac中安装和运行pyspider
- 【已解决】pyspider中如何写规则去提取网页内容
- 【已解决】pyspider中如何加载汽车之家页面中的更多内容
- 【已解决】PySpider如何把json结果数据保存到csv或excel文件中
- 【已解决】PySpider中如何清空之前运行的数据和正在运行的任务
- 【已解决】Python中实现带Cookie的Http的Post请求 – 在路上
- 【已解决】Python中如何获得访问网页所返回的cookie – 在路上
- Requests
- re
- aiohttp
- PyMySQL
- PyMongo

- [urllib](#)
- [BeautifulSoup](#)
- [PyQuery](#)
- [lxml](#)
- [PySpider](#)
- [Scrapy](#)
- [Chrome 开发者工具 | Tools for Web Developers](#)
- [rmax/scrapy-redis: Redis-based components for Scrapy.](#)
- [grangier/python-goose: Html Content / Article Extractor, web scrapping lib in Python](#)
- [Bloom Filters by Example](#)
- [Bloom Filters by Example 中文](#)
- [Scrapy入门教程 — Scrapy 0.24.6 文档](#)
- [Scrapy爬虫框架教程（一）-- Scrapy入门](#)
- 

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,  
powered by Gitbook最后更新: 2020-07-16 17:47:06