Shift+tab ：格式化代码

1.解析url规律

https://search.51job.com/list/170300,000000,0000,00,9,99,python,

2,1.html?lang=c&stype=&postchannel=0000&workyear=99&cotype=99&

degreefrom=99&jobterm=99&companysize=99&providesalary=99&

lonlat=0%2C0&radius=-1&ord\_field=0&confirmdate=9&fromType=&

dibiaoid=0&address=&line=&specialarea=00&from=&welfare=

2.url发送请求 -> 服务器响应：网页源代码 ->抓包 ->read().decode() 解码读取

3.正则表达式

爬取人人车车辆信息

多线程/多进程 -->提高代码的执行效率；在爬虫中就是提高爬取效率；

之前糗百/51job是同步执行（按照先后顺序执行）

1. 职位搜索页：输入查询：python；分别查询三个不同城市，摘取url进行分析；

北京：

<https://search.51job.com/list/000000%252C00,000000,0000,00,9,99,python,2,1.html?lang=c&stype=1&postchannel=0000&workyear=99&cotype=99&degreefrom=99&jobterm=99&companysize=99&lonlat=0%2C0&radius=-1&ord_field=0&confirmdate=9&fromType=&dibiaoid=0&address=&line=&specialarea=00&from=&welfare>=

上海：

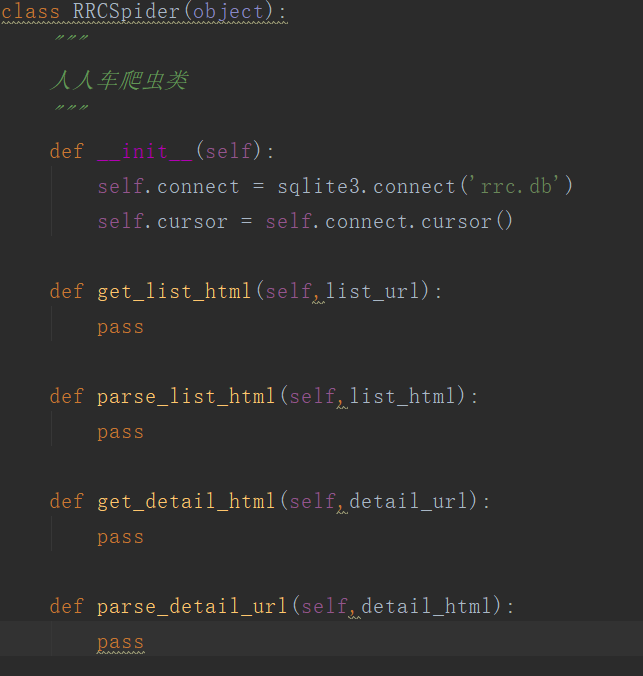
<https://search.51job.com/list/000000%252C00,000000,0000,00,9,99,python,2,1.html?lang=c&stype=1&postchannel=0000&workyear=99&cotype=99&degreefrom=99&jobterm=99&companysize=99&lonlat=0%2C0&radius=-1&ord_field=0&confirmdate=9&fromType=&dibiaoid=0&address=&line=&specialarea=00&from=&welfare>=

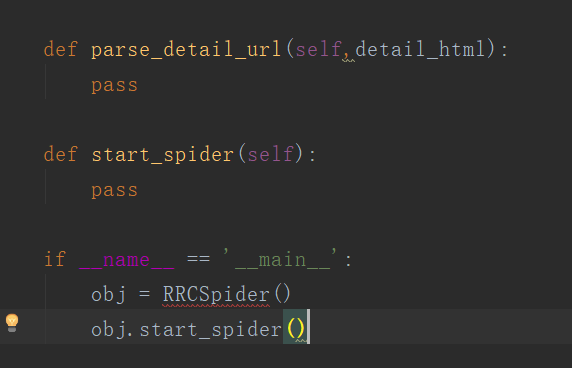
郑州：

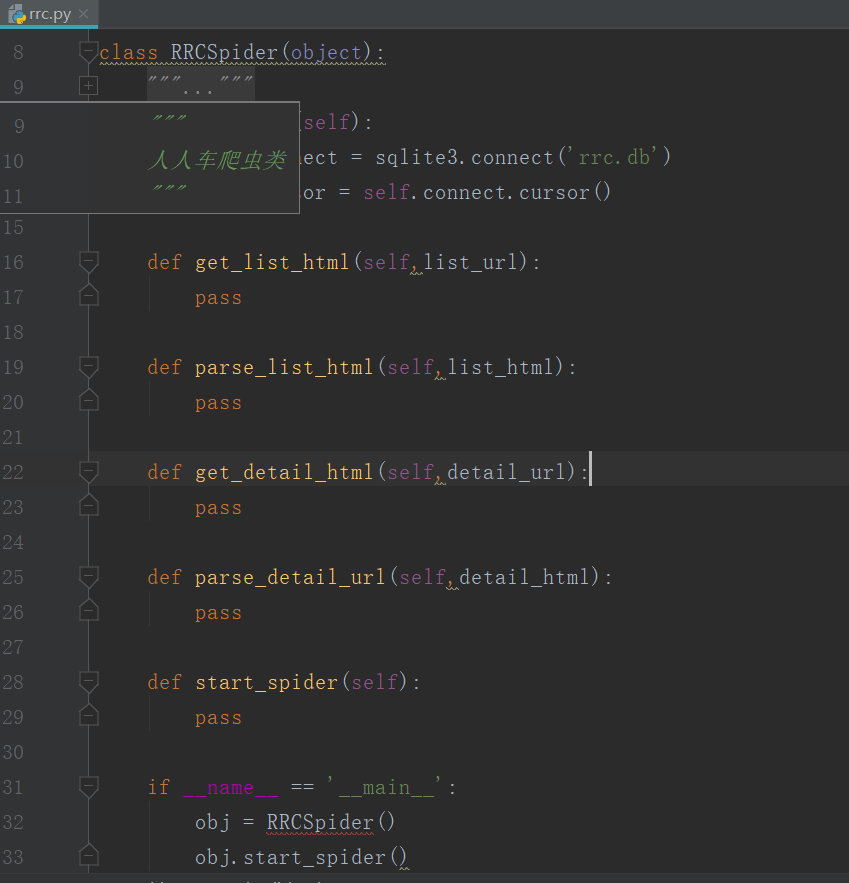
<https://search.51job.com/list/000000%252C00,000000,0000,00,9,99,python,2,1.html?lang=c&stype=1&postchannel=0000&workyear=99&cotype=99&degreefrom=99&jobterm=99&companysize=99&lonlat=0%2C0&radius=-1&ord_field=0&confirmdate=9&fromType=&dibiaoid=0&address=&line=&specialarea=00&from=&welfare>=

1.列表页 --> 详情页 -->

1. 导包 - 爬虫类 - 在构造函数中建立连接以避免重复建立连接 - 定义4个函数（获取列表页、解析列表页、获取详情页、解析详情页）；



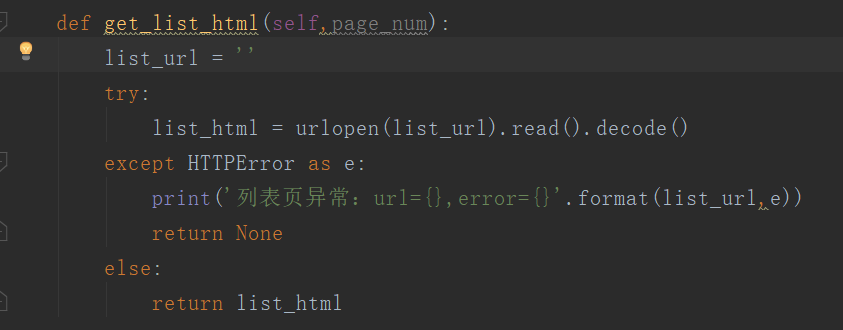
1. 新增爬虫启动入口函数，取代之前的 if \_\_name\_\_ ==’\_\_main\_\_’:函数入口模块；
2. 
3. 整体框架



1.实现第一个函数：

url 拼接：

将get\_list\_url的参数 list\_url 更改为 page\_num ；



2.实现第二个函数（正则表达式）：

4.实现第四个函数（正则表达式）：ctrl+f ：搜索标签和属性 确定唯一性；

函数实现完毕；知识完成了队第一页的爬取；

5.实现for循环爬取；

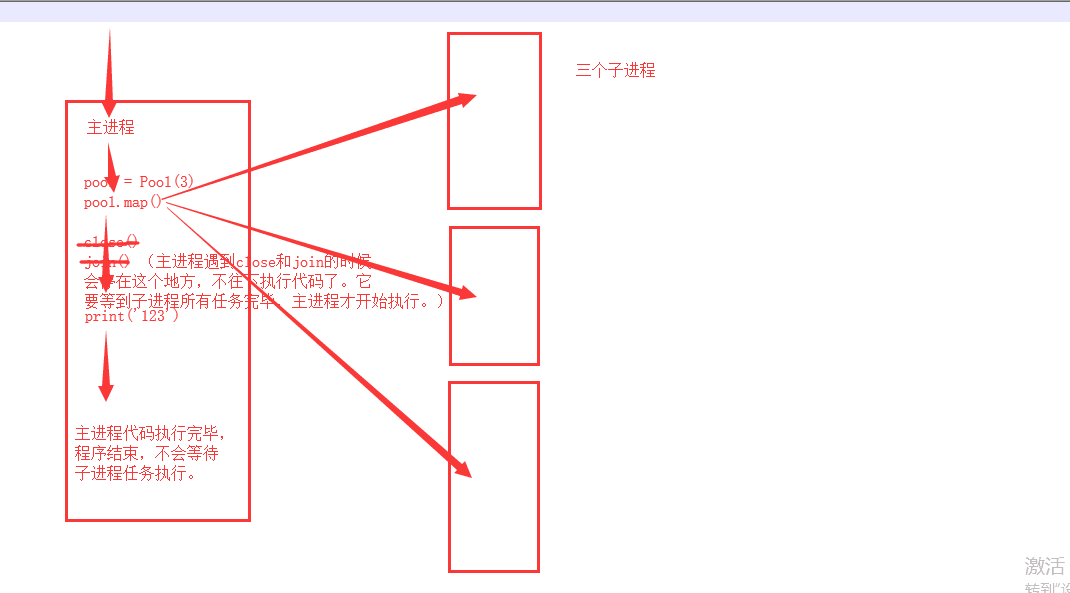
Main 函数中实现；在start中增加参数 num;

此方式下数据库关闭需放在最后；

以上为同步循环爬取；

多进程 for 循环：

在main 中实现



**Json类型网页**

1. Json类型网页 ：该网页提取数据不需要正则表达式，只需要解析json 即可；
2. 智联招聘：sou.zhaopin.com 以json数据返回内容；
3. 分析网站是否有接口返回json数据，若有，使用json.loads() 对字符串进行解析，获取数据；

爬取步骤:

1. 智联招聘官网

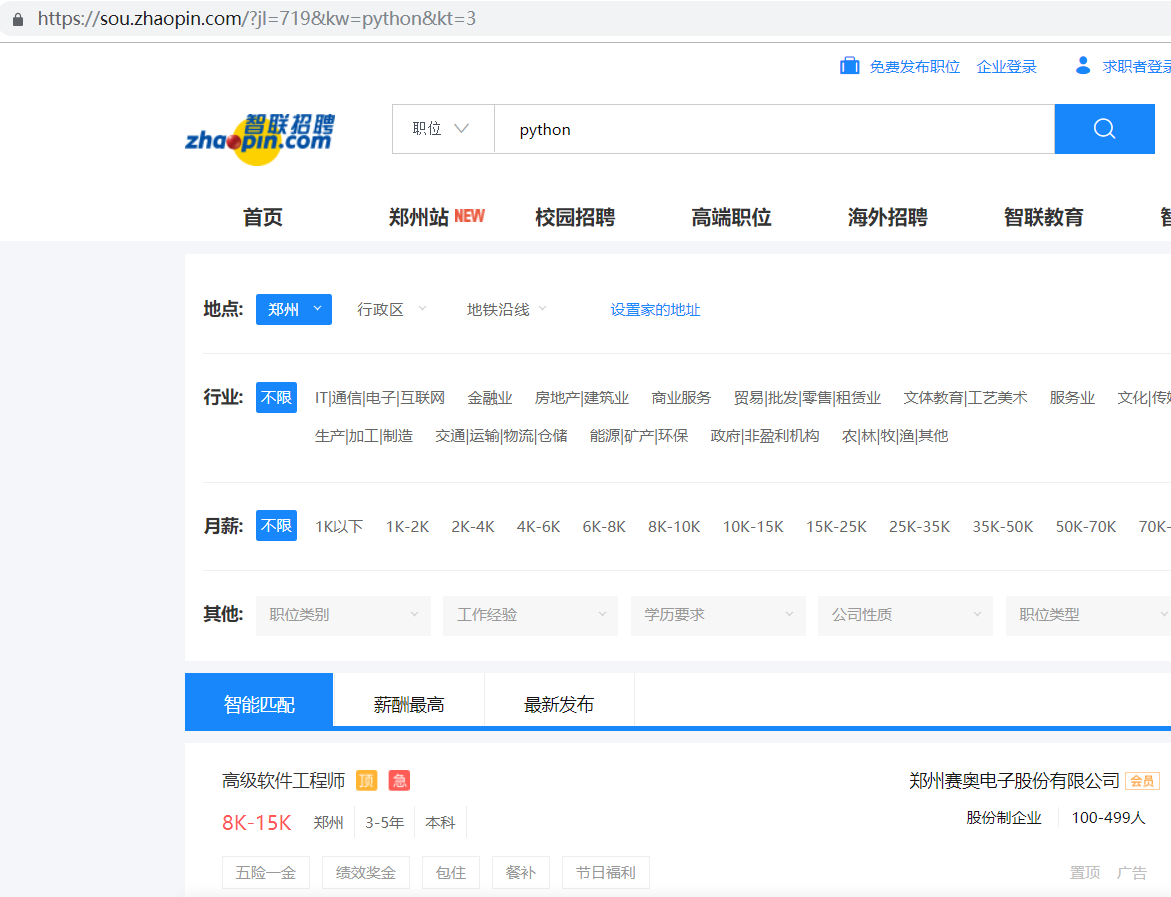


1. 点击老用户登陆 🡪 点击智联招聘logo

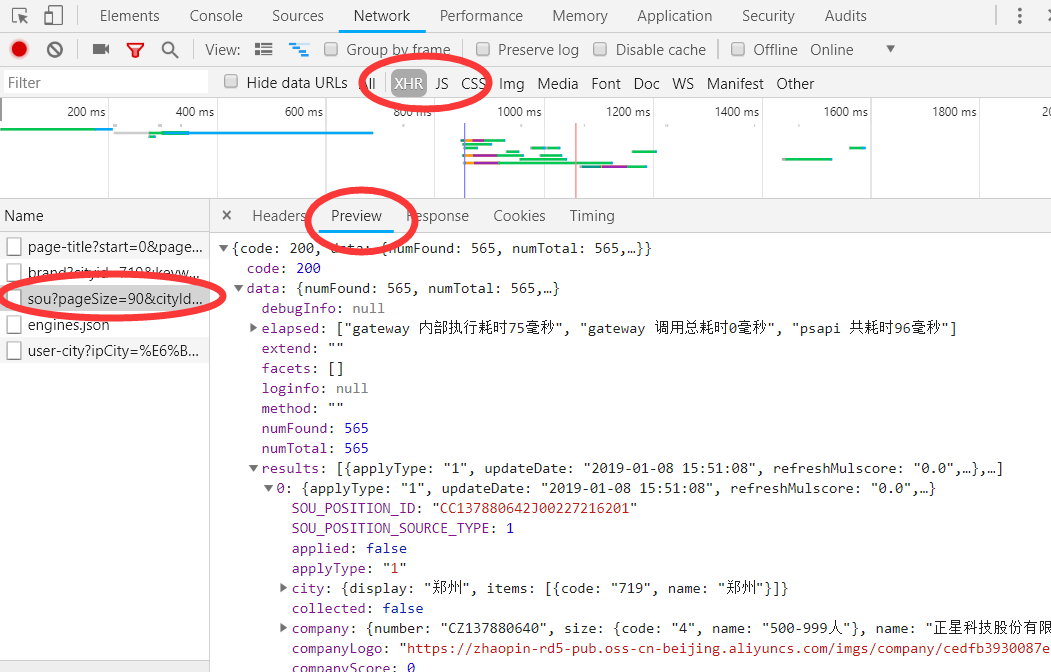
🡪 进入网站搜索页（https://www.zhaopin.com/）



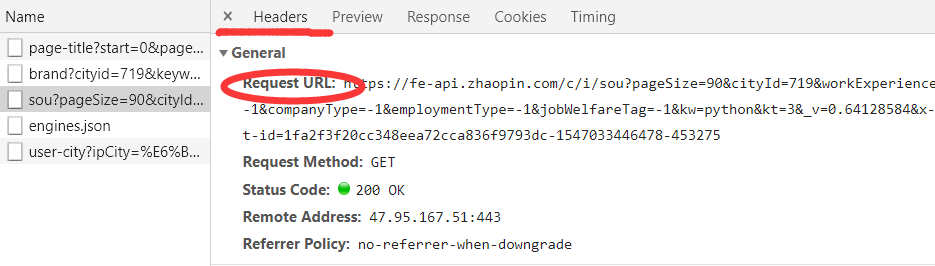
1. 搜索 python 获取职位列表页



1. Fn + f12 打开 “检查”；🡪 network 🡪 包类型：xhr 🡪 刷新页面，抓包 🡪 找到包含所需信息的包



1. 点击 Headers 获取该包的url



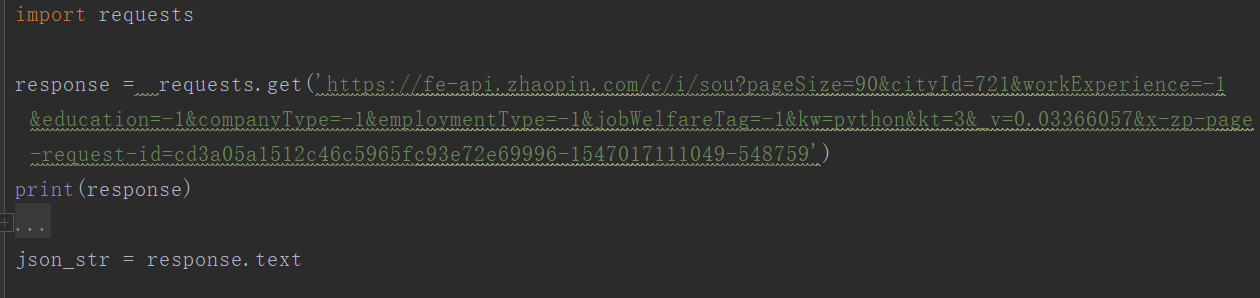
该url信息：



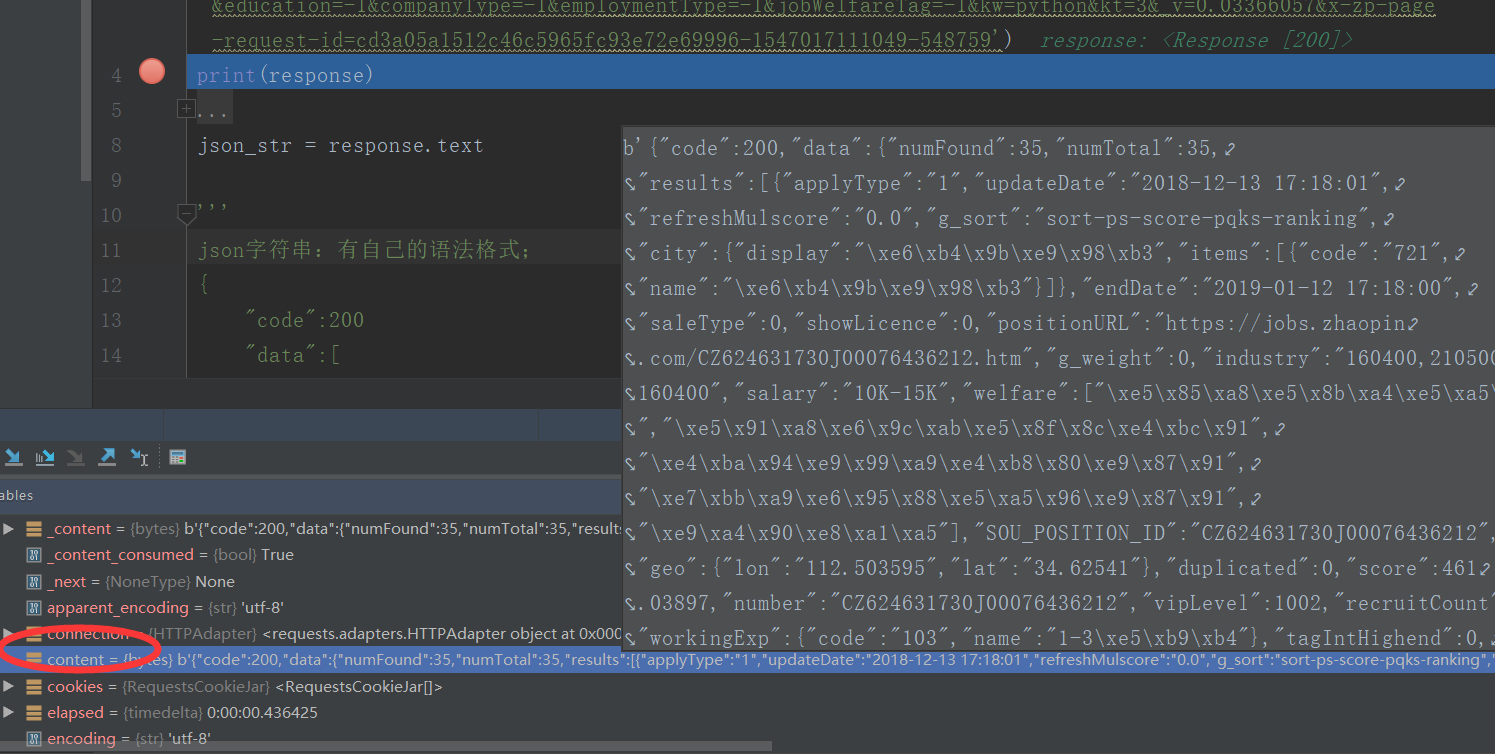
6.使用requests库 获取数据 ，并转换为文本数据。

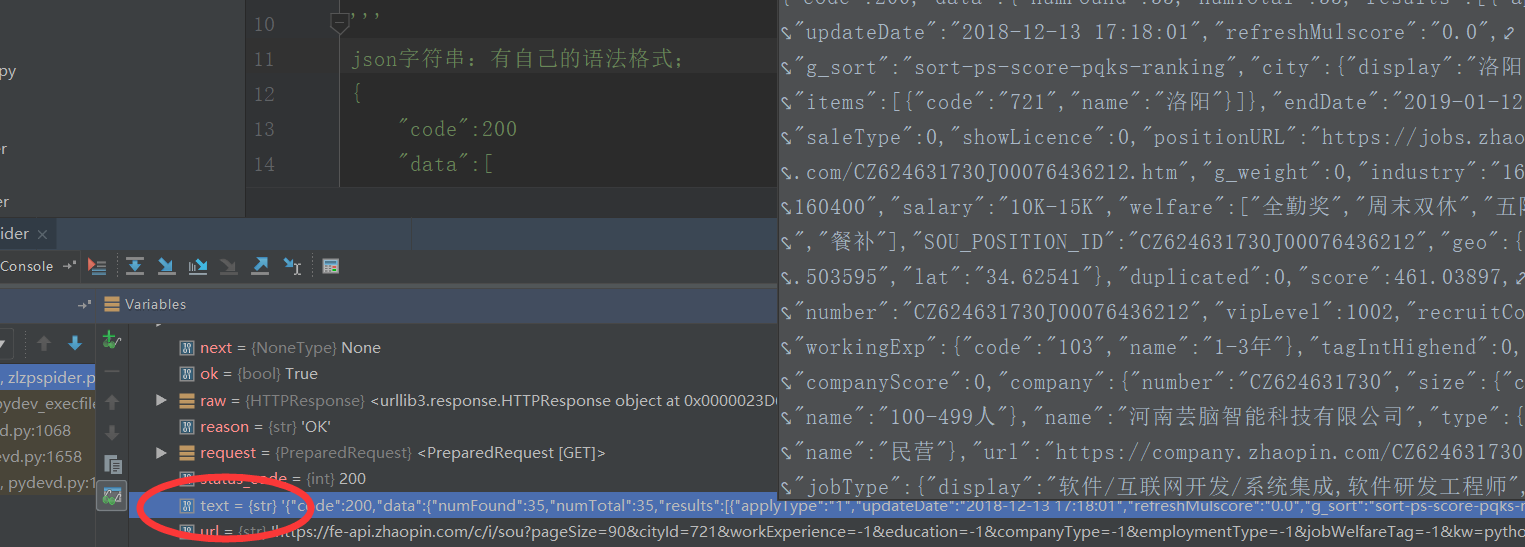
response = requests.get(url)：作用：发送网络请求，返回响应数据

response.text：获取response中的数据信息，返回的是一个 unicode 型的文本数据



7.debug模式下可以看到 response的内容：





Content变量中为响应包的二进制文件；

Text中为解码后的文件。

8.分析json 数据可以发现所需的数据存放在层层嵌套的字典中，所以使用json库来获取数据。

(1)json.dumps()函数是将一个Python数据类型列表进行json格式的编码（可以这么理解，json.dumps()函数是将字典转化为字符串）  
　(2)json.loads()函数是将json格式数据转换为字典（可以这么理解，json.loads()函数是将字符串转化为字典）

CSDN最新文章爬取：C:\Users\WJY\AppData\Local\Temp\%W@GJ$ACOF(TYDYECOKVDYB.pnghttps://www.csdn.net/nav/newarticles

没有下一页的页面 基本上都是json

刚开始10条用正则表达式

剩余用json

今日头条：

下拉加载型

天气、电影 、csdn 练习；

**百度天气查询**

1. 先定义主函数接口：IP-api和 weather-api；
2. 定义获取位置函数：请求ip-api 获取位置；
3. 定义获取天气函数（含参：城市名）：请求weather-api 获取天气；将文本格式转字典后，进行字典的嵌套取值操作，获取相应数据；
4. 定义查询函数获取输入（城市名）；
5. 另外：两个API接口为网上搜索获取；

**爬取csdn文章并保存到 mongoDB 数据库**

Network – all – article包 – 查看html数据

XHR – 查看Json数据

首次刷新页面后共返回21条数据：

页面前11条是html数据

后10条是自动请求下拉接口获取json数据—请求下拉接口获取；

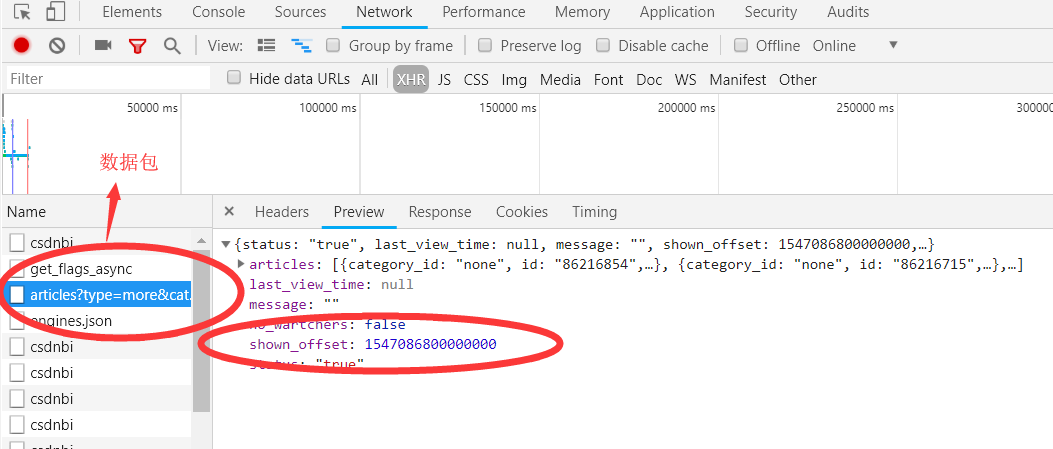
以上21条数据已被页面解析，可以用正则表达式获取数据；

之后下拉页面会不断请求接口获取数据，每次10条；

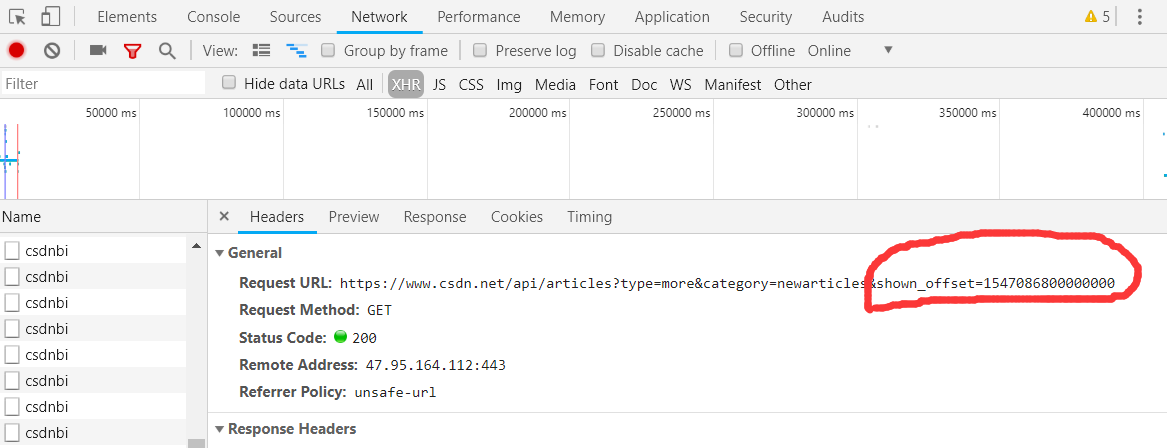
可以解析json包直接获取，比正则表达式方便；

规律：对接口数据分析：

一次接口数据：



下一次接口数据：



接口请求规律：

第一次刷新页面自动请求下拉接口；该接口会返回一个shown\_offset，下拉页面后，不断请求接口，这个shown\_offset的值会作为下一个接口请求的参数，所以这个参数的值可以从上一个数据接着返回新的数据；

也就是说：

第一次请求 <https://www.csdn.net/nav/newarticles> 地址时，返回的网页源代码默认展示11条数据，同时又进行了一次下拉接口的请求，接口又加载了10条数据，在返回的网页源代码中的 ul 标签中有一个属性 shown\_offset，这个属性值就是做为第一次请求接口的参数使用。

模拟浏览器行为：

1. 刷新浏览器和自动请求一次接口：
2. 定义爬虫类CSDNSpider；
3. 定义请求页面地址函数 get\_article\_html()；

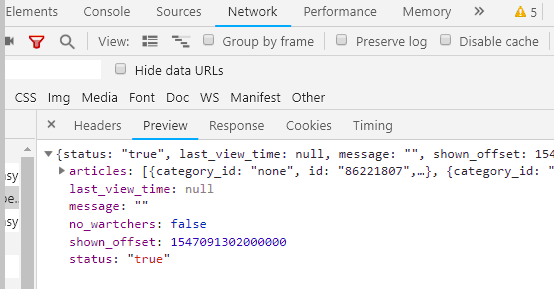
模拟浏览器首次加载的整个过程：

目的：请求article\_url 🡪 提取url标签的shown-offset属性🡪 同时利用正则表达式将html 中的数据提取出来 🡪 利用shown-offset 发起第一次接口请求；

1. 定义请求接口函数（封装成函数一边循环调用）get\_json\_data(shown-offset);

拼接请求地址：api\_url = 接口地址 + shown-offset值

获取json包（嵌套的字典）：



网页中21条 + 首次调用接口10条 = 31条数据；

从字典中取出所需数据，完成模拟浏览器的首次加载。

1. 由shown-offset值循环调用接口，实现不断加载和爬取数据。
2. 导入pymongo库，实现数据库 mongoDB 连接，优化代码；

**今日头条**

--街拍

1.导入相关包

2.创建爬虫类、\_\_init\_\_方法；在该方法中粘贴CSDN中的数据库连接语句；

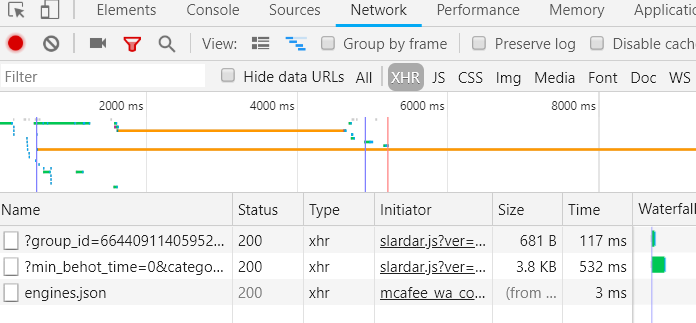
3.分析网页：

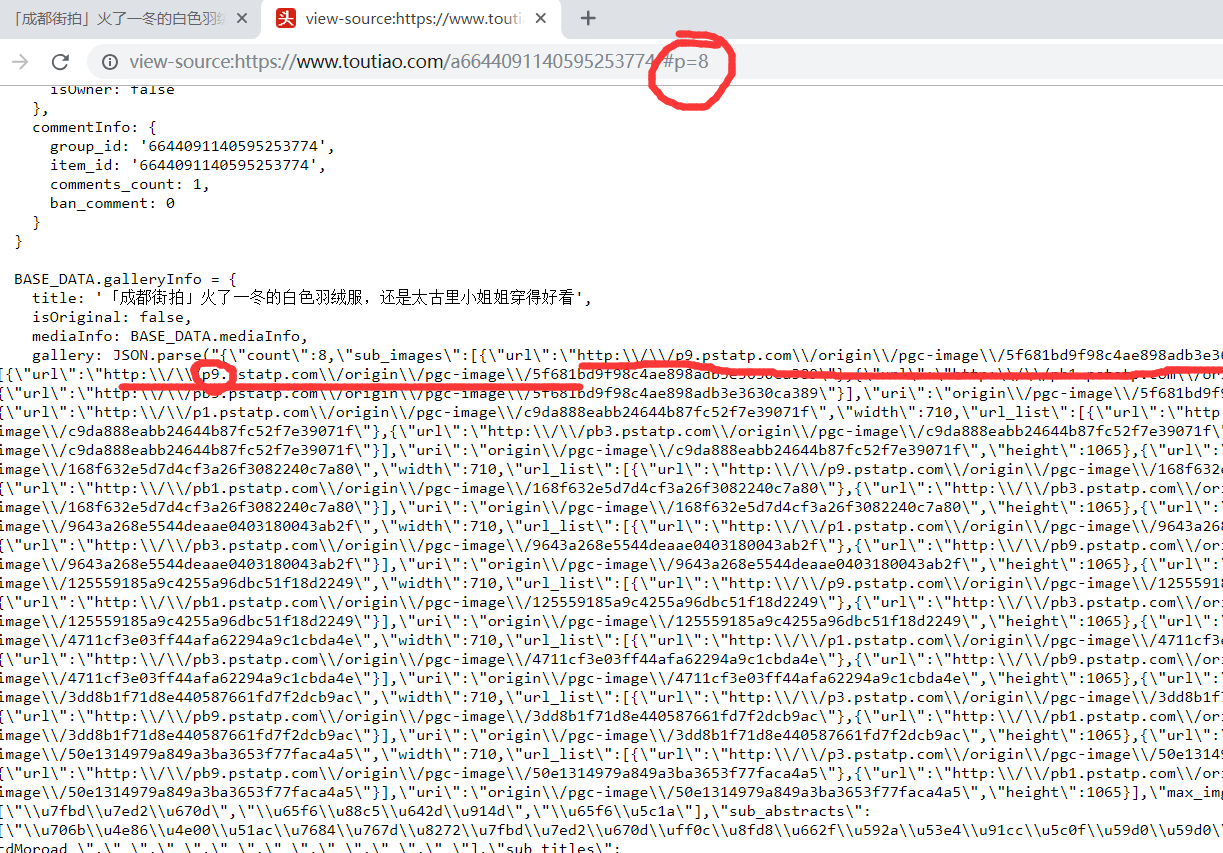
列表页 –> 检查 🡪 XHR包 🡪 下拉加载页面 🡪在头文件中截取三次加载的地址，分析对比三个url 找出变化量 offset🡪 筛选出只包含组图的（不含文章）的包进行爬取 🡪将抓取的包文件转换为字典 🡪 在嵌套的字典中取数据

1. 定义请求列表页的函数：get\_list\_json(offset);拼接url；发起请求获取json字符串（response.text）
2. 定义解析列表页json数据函数：parse\_list\_json(json\_str);将字符串转化为字典，取数据（参照preview中的数据结构取）；
3. 用判断语句去除字典中含有cell\_type , single\_model (文章)的数据；
4. 从字典中提取数据项article\_url：单组图片的地址；

将所有单个article\_url整合成为一个列表 urls;

1. 定义get\_detail\_page(detail\_urls);接受url列表：urls；
2. 选一组图片，监听json数据包，查找规律，目标：实现通过一次请求，获取一组图片数据；

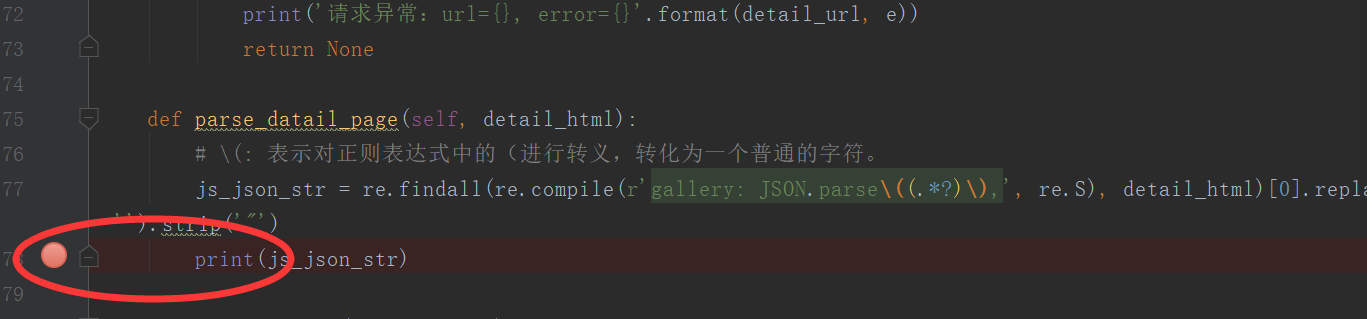
若找不到，再查看网页源代码：



1. 定义parse\_detail\_page(detail\_html)函数，用正则表达式提取详情页源码中的js图片地址（上图红线部分）；需要提取的部分：gallery: JSON.parse(“………”)括号中间的部分；
2. 定义start\_spider()函数；
3. 整理json数据js\_json\_str存入数据库；使用json在线编辑工具<http://www.bejson.com/jsoneditoronline/>；

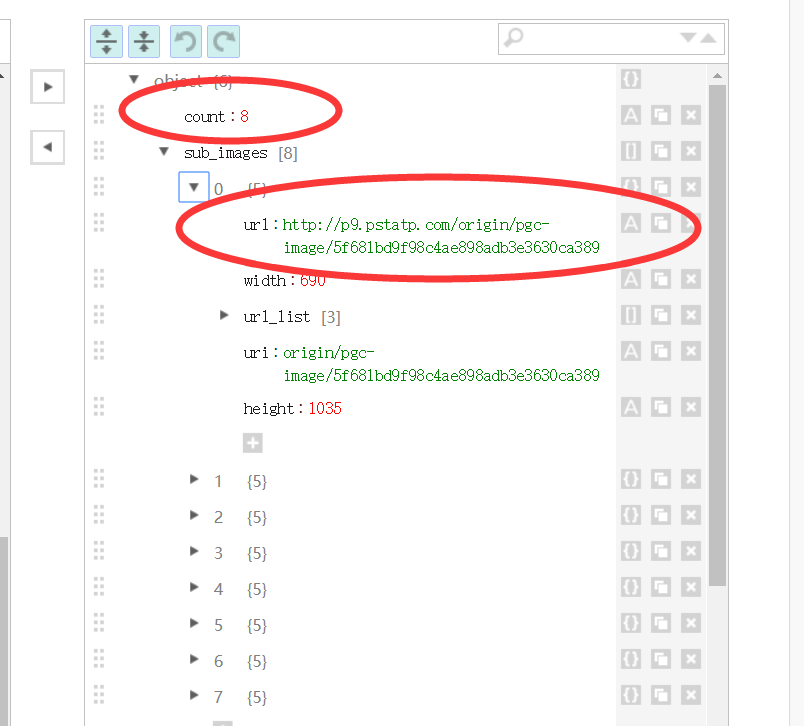
此处获取的时一组8张图片数据





Debug获取js\_json\_str变量的数据，复制值到json在线编辑器中：





将字符串转为字典，从字典中取出每个图片的url地址

1. 定义save\_dict\_to\_db(dic)函数，以便将数据保存进数据库；
2. 定义download\_image(ima\_url),向图片url发送请求获取资源；（图片属于二进制资源：数据的传输是以二进制流的方式进行，故用response.content）;
3. 将二进制图片数据content写入本地数据库；
4. 将单进程改为多进程；