

day03-课堂笔记

day03-课堂笔记

1. requests模块处理cookie

- 1.1 状态保持的原理（web知识）
- 1.2 在爬虫中为什么要携带cookie
- 1.3 方式一
- 1.4 方式二

2. requests模块设置超时

3. 数据提取的概述

- 3.1 响应数据的分类
- 3.2 常见的数据解析方法

4. json模块的使用

- 4.1 json.loads()
- 4.2 json.dumps()
- 4.3 json.load()
- 4.4 json.dump()

作业：

5. xpath语法的使用

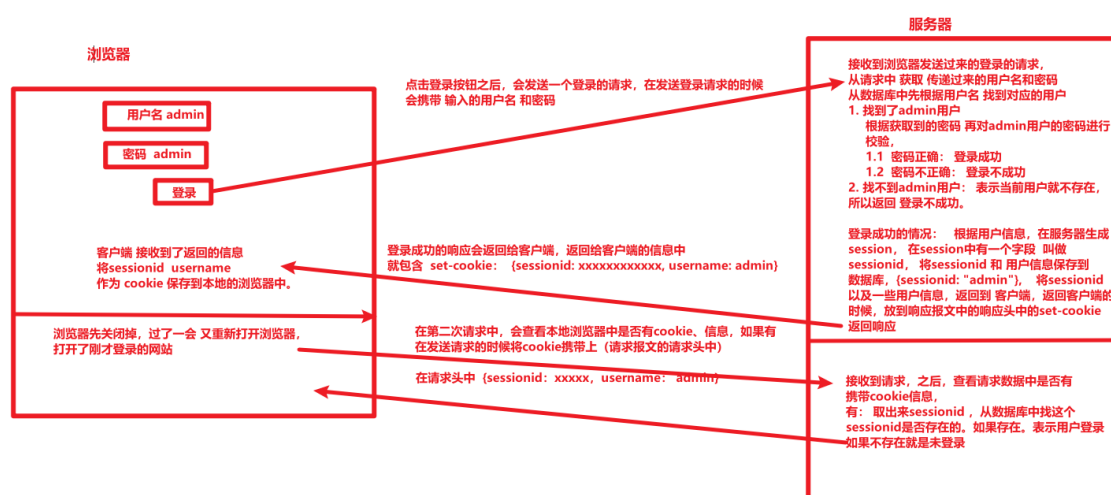
- 5.1 xpath的介绍
- 5.2 xpath_helper插件的安装
- 5.3 基本语法
- 5.4 进阶使用

如果豆瓣不会带cookie

1. requests模块处理cookie

cookie主要是用于在web开发的时候，状态保持（用户的登录状态）。

1.1 状态保持的原理（web知识）



1.2 在爬虫中为什么要携带cookie

- 为了登录，需求：要大家去抓取一下每个人 淘宝的购物记录，获取登录之后才可以看到的用户信息。

1.3 方式一

在请求的时候，如果想要实现状态保持，那么在发送请求的时候，就需要在请求头中携带登录之后的cookie信息，才可以实现登录。

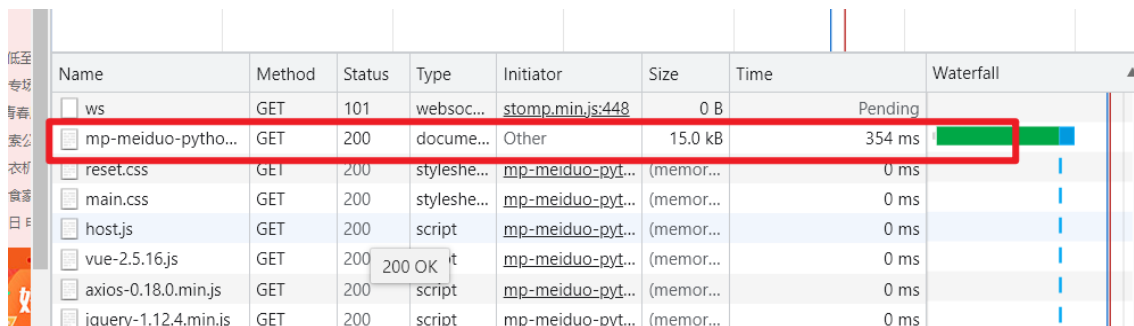
在requests模块中，是可以在发送请求的时候携带请求头的，

```
1 headers = {}
2 response = requests.get(url, headers=headers)
```

既然在发送请求的时候需要在请求头中携带cookie，那么我们就可以将登录之后的cookie放到headers字典中就可以。

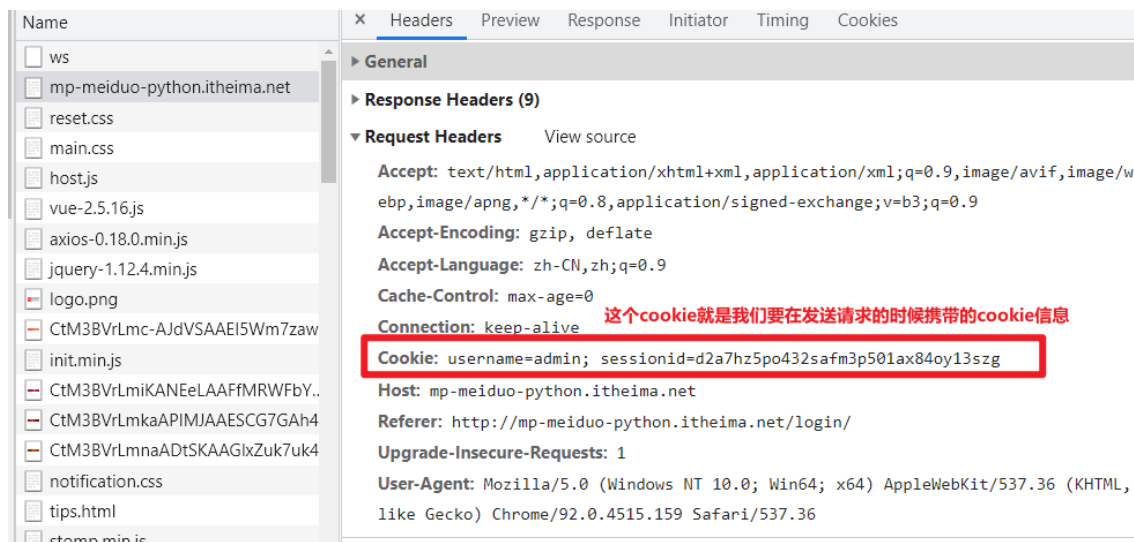
实现步骤：

- 打开浏览器，在浏览器中去登录要抓取的目标网站。
- 浏览器中，右键-- 检查--network，刷新页面，找到当前请求的首页的url地址



Name	Method	Status	Type	Initiator	Size	Time	Waterfall
ws	GET	101	websoc...	stomp.min.js:448	0 B	Pending	
mp-meiduo-python.itheima.net	GET	200	docume...	Other	15.0 kB	354 ms	
reset.css	GET	200	styleshe...	mp-meiduo-pyt...	(memor...	0 ms	
main.css	GET	200	styleshe...	mp-meiduo-pyt...	(memor...	0 ms	
host.js	GET	200	script	mp-meiduo-pyt...	(memor...	0 ms	
vue-2.5.16.js	GET	200	script	mp-meiduo-pyt...	(memor...	0 ms	
axios-0.18.0.min.js	GET	200	script	mp-meiduo-pyt...	(memor...	0 ms	
jquery-1.12.4.min.js	GET	200	script	mp-meiduo-pyt...	(memor...	0 ms	

- 打开这个之后，找到 request headers



Name	Headers	Preview	Response	Initiator	Timing	Cookies
mp-meiduo-python.itheima.net	<p>General</p> <p>Response Headers (9)</p> <p>Request Headers View source</p> <p>Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9</p> <p>Accept-Encoding: gzip, deflate</p> <p>Accept-Language: zh-CN,zh;q=0.9</p> <p>Cache-Control: max-age=0</p> <p>Connection: keep-alive</p> <p>Cookie: username=admin; sessionId=d2a7hz5po432safm3p501ax84oy13szzg</p> <p>Host: mp-meiduo-python.itheima.net</p> <p>Referer: http://mp-meiduo-python.itheima.net/login/</p> <p>Upgrade-Insecure-Requests: 1</p> <p>User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.159 Safari/537.36</p>					

- 拿到这个cookie之后，将这个cookie进行复制，放到代码中，将复制来的cookie改成一个字典，headers字典。
- 可以在发送请求的时候将headers字典传递给headers参数。

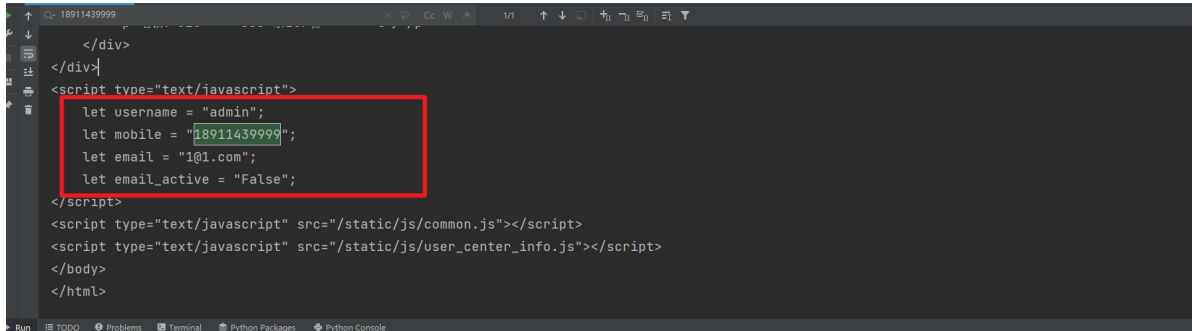
完整代码实现

```
1 import requests
2
3 # 1. 准备url地址 个人中心的url地址，如果不登录是访问不到这个页面的，
4 # 如果直接访问 返回的就是 登录页面
5 url = "http://mp-meiduo-python.itheima.net/info/"
6
7 # 准备好登录的时候要携带的 cookie的请求头字典
```

```

8 headers = {
9     "Cookie": "username=admin; sessionid=eho9j4fys8h148f19b0pw0jff37k7boy"
10 }
11
12 # 2. 发送请求 获取响应 要进行状态保持，这个时候 发送请求的时候将 准备好的headers字典携带上
13 response = requests.get(url, headers=headers)
14
15 print(response.content.decode())

```



1.4 方式二

方式一中，我们在携带cookie的时候，使用的 请求头来携带的cookie，

我们也可以去使用一个参数， cookies参数来携带cookie

语法：

```

1 cookies = {}
2
3 response = requests.get(url, cookies=cookies)
4
5
6 下面的是从浏览器中复制来的cookie
7 "username=admin; sessionid=eho9j4fys8h148f19b0pw0jff37k7boy"
8
9 准备cookie的时候，需要的是一个 字典类型，但是呢从浏览器复制过来的cookie数据是一个字符串，
  想办法将cookie字符串 转换为 cookie字典，

```

```

1 _octo=GH1.1.1065531107.1630458486;
2 tz=Asia%2FShanghai;
3 _device_id=0868823d691ece302cba3c1fe80d7adc;
4 has_recent_activity=1;
5 user_session=1ANoHoT-pIlfveH1vnp2me313mGNYPgfp2Mjk9-276JKoXz;
6 __Host-user_session_same_site=1ANoHoT-pIlfveH1vnp2me313mGNYPgfp2Mjk9-
  276JKoXz; tz=Asia%2FShanghai;
  color_mode=%7B%22color_mode%22%3A%22auto%22%2C%22light_theme%22%3A%7B%22name%
  22%3A%22light%22%2C%22color_mode%22%3A%22light%22%7D%2C%22dark_theme%22%3A%7B
  %22name%22%3A%22dark%22%2C%22color_mode%22%3A%22dark%22%7D%7D; logged_in=yes;
  dotcom_user=ZuoAndroid;
  _gh_sess=djsg03%2BDZxNwtjQSEN%2BKPHYkDIZemty3JHQqyKbpTQXRmGcam2l8mQXFC0iQjkeb
  3tyDhQkhWke3GG3C9JpA7s3WLW4ekx4MN7FAS8h7TnATYx%2Bvfu8xtHpKZW6yqK%2FXzS5RQHewL
  PcvGynMLbrbvg6mKRMZZ%2B89kFdkeFmLZXQVWSiAaGJpoEXWHTJdx9AQhVzEqmQSUSd9J%2FjqT
  MDUO%2Fj8700Tx8gRxEL4X%2BCJwXGmOCFGko5xSVovy0sAWMX0Gltwgsyqwk9808MoXY%2BPPsZg
  xp9TrqFSKMWIXjaDwc%2BBvcuMqYuPHxtPW8emYATrSK%2BwFRU5fopP2QVkuGcfDXyFxyxEK4Ibx
  wlcZJOS0zdX9QmZjhtREZBHy4X4LnUjBJ7i6WHVWF1hk8ZZN05mdso8QHJVJvZKCFiFe9rBnryNqi
  GE3rj%2FavmbdFCRk5T107ai3i9cR5nMGISuLxFM4HREw0z2fiHbe8q%2Be%2FtbZgUxjITDMA1uY
  RWIg%2FER2F8%2Fqbtjje17p9occ9YECzmfW%2F4Q%2BaS%2BFPh9kw0SEfpneiddIuwK1ji%2BYS
  JwCBt3fkbZ2z1056yq5vYZxtibENRASCfpuEYjBKM7SGO6IIFsYS60LNxSPA1mwR9MaxreSN5JjWY
  zbmc1d%2BY8kyEdTAYN99leqiDTnDRrSonXnX1JBQAE9KmNY%2B1M43ecfv8Mg598IHSHEQWCJK
  4owF4d2glYEukV%2BgMhuagSn3xtg4rUPaS0%2BjBL%2B1hpe7u0eq3wwLgxiZhD4wGYrxzGgYabJ
  D5CwJ8W2Lgsd8Q1rUcrfQv168rY6unHxYjYJONJwxnLtrNILM352CEyG1n6o2gYJTJC08e6E7Djac
  mJQCEILq7j8vgVrtsXEPNSbFU9XD6z24cZ3nSwXcKc%3D--zkMFRPbsdlpcGGxe--
  w5xo4iq3IUYyToTH8jTigA%3D%3D

```

如何将cookie字符串转换为cookie字典

```

1 cookie_str = "_octo=GH1.1.1065531107.1630458486; tz=Asia%2FShanghai;
  _device_id=0868823d691ece302cba3c1fe80d7adc; has_recent_activity=1;
  user_session=1ANoHoT-pIlfveH1vnp2me313mGNYPgfp2Mjk9-276JKoXz; __Host-
  user_session_same_site=1ANoHoT-pIlfveH1vnp2me313mGNYPgfp2Mjk9-276JKoXz;
  tz=Asia%2FShanghai;
  color_mode=%7B%22color_mode%22%3A%22auto%22%2C%22light_theme%22%3A%7B%22name
  %22%3A%22light%22%2C%22color_mode%22%3A%22light%22%7D%2C%22dark_theme%22%3A%
  7B%22name%22%3A%22dark%22%2C%22color_mode%22%3A%22dark%22%7D%7D;
  logged_in=yes; dotcom_user=ZuoAndroid;
  _gh_sess=djsg03%2BDZxNwtjQSEN%2BKPHYkDIZemty3JHQqyKbpTQXRmGcam2l8mQXFC0iQjeb
  3tyDhQkhWke3GG3C9JpA7s3WLW4ekx4MN7FAS8h7TnATYx%2Bvfu8xtHpKZW6yqK%2FXzS5RQHe
  wLPcvGynMLbrbvg6mKRMZZ%2B89kFdkeFmLZXQVWSiAaGJpoEXWHTJdx9AQhVzEqmQSUSd9J%2F
  jqTMDUO%2Fj8700Tx8gRxEL4X%2BCJwXGmOCFGko5xSVovy0sAWMX0Gltwgsyqwk9808MoXY%2BP
  PsZgxp9TrqFSKMWIXjaDwc%2BBvcuMqYuPHxtPW8emYATrSK%2BwFRU5fopP2QVkuGcfDXyFxyxE
  K4IbxwlcZJOS0zdX9QmZjhtREZBHy4X4LnUjBJ7i6WHVWF1hk8ZZN05mdso8QHJVJvZKCFiFe9rB
  nryNqiGE3rj%2FavmbdFCRk5T107ai3i9cR5nMGISuLxFM4HREw0z2fiHbe8q%2Be%2FtbZgUxjI
  TDMA1uYRWIg%2FER2F8%2Fqbtjje17p9occ9YECzmfW%2F4Q%2BaS%2BFPh9kw0SEfpneiddIuwK
  1ji%2BYSJwCBt3fkbZ2z1056yq5vYZxtibENRASCfpuEYjBKM7SGO6IIFsYS60LNxSPA1mwR9Max
  reSN5JjWYzbmc1d%2BY8kyEdTAYN99leqiDTnDRrSonXnX1JBQAE9KmNY%2B1M43ecfv8Mg598I
  HSHEQWCJK4owF4d2glYEukV%2BgMhuagSn3xtg4rUPaS0%2BjBL%2B1hpe7u0eq3wwLgxiZhD4w
  GYrxzGgYabJD5CwJ8W2Lgsd8Q1rUcrfQv168rY6unHxYjYJONJwxnLtrNILM352CEyG1n6o2gYJT
  JC08e6E7DjaCmJQCEILq7j8vgVrtsXEPNSbFU9XD6z24cZ3nSwXcKc%3D--zkMFRPbsdlpcGGxe-
  -w5xo4iq3IUYyToTH8jTigA%3D%3D"
2
3 # cookie字符串中， 每一个分号+空格 分隔就是一条cookie，在上述的cookie字符串中， 实际上存
  储的是两条cookie
4 # 在cookie字符串中有几条cookie，我们转换后的字典中就有几个键值对。
5 # a=b    key:value

```

```

6
7 # 字典: {key: value}
8
9 # 1. 先对 cookie字符串 进行 按照 分号+空格 的方式进行 分隔 split("") 分隔完毕之
  后 列表
10 cookie_list = cookie_str.split("; ")
11
12 # 2. 最终需要的是 key:value 的格式, 并不是 key=value
13 # 遍历列表, 取出列表中的每一个元素, 每一个元素就是一条cookie, 对应的就是字典中的一个键值对
  _octo=GH1.1.1065531107.1630458486
14 # 再次根据 = 进行一次分隔 [_octo, GH1.1.1065531107.1630458486]
15 # 根据=分隔完毕之后, 拿到的列表中有两个元素, {_octo:GH1.1.1065531107.1630458486}
16 cookie_dict = {}
17 for cookie in cookie_list:
18     # 当前遍历出来的cookie 值得格式 _octo=GH1.1.1065531107.1630458486
19     c_list = cookie.split("=")
20     key = c_list[0]
21     value = c_list[1]
22     # 字典中的赋值, 增加键值对
23     cookie_dict[key] = value
24
25 print(cookie_dict)

```

完整代码实现

```

1 import requests
2
3 # 1. 准备url地址
4 url = "https://github.com/ZuoAndroid"
5
6 # 准备cookie字典
7 cookie_str = "_octo=GH1.1.1065531107.1630458486; tz=Asia%2FShanghai;
  _device_id=0868823d691ece302cba3c1fe80d7adc; has_recent_activity=1;
  user_session=1ANoHoT-pIlfveH1vnp2me313mGNYPgfp2MJk9-276JKoXz; __Host-
  user_session_same_site=1ANoHoT-pIlfveH1vnp2me313mGNYPgfp2MJk9-276JKoXz;
  tz=Asia%2FShanghai;
  color_mode=%7B%22color_mode%22%3A%22auto%22%2C%22light_theme%22%3A%7B%22name
  %22%3A%22light%22%2C%22color_mode%22%3A%22light%22%7D%22dark_theme%22%3A%
  7B%22name%22%3A%22dark%22%2C%22color_mode%22%3A%22dark%22%7D%7D;
  logged_in=yes; dotcom_user=ZuoAndroid;
  _gh_sess=djsg03%2BDZxNwtjqSEN%2BKPPhyKdIZemty3JHQqyKbpTQXRmGcam2l8mQXFC0iQjek
  b3tyDhQkhWke3GG3C9Jpa7s3WLW4eKx4MN7Fas8h7TnATYx%2Bvfu8xtHpKZW6yqK%2FXzS5RQHe
  wLPcVGynMLbRbvg6mKRMZZ%2B89kFdKeFmLZXQVWSiAaGJpoEXWHTJdxb9AQhvzEqmQSUSd9J%2F
  jqTMDUO%2Fj87OOTx8gRxEL4X%2BCJwXGmOCFGko5xSVoVy0sAWMX0GltwgSyqwk9808MoXY%2BP
  PsZgxp9TrqFSKMWIXjaDwc%2BBvcuMqYuPHxtPW8emYATrSK%2BWfRU5fopP2QvkugcfdXYfxyE
  K4IbxwlCZJos0zdx9QmzjhtREZBHY4X4LnUjBJ7i6WHVWF1hk8ZZN05mdso8QHJVJvZKCFiFe9rB
  nryNqiGE3rj%2FavmbdFCrk5T107ai3i9cR5nMGIsuLxFM4HREw0z2fiHbe8q%2Be%2FtbzgUxjI
  TDMA1uYRWig%2FER2F8%2Fqbtjje17p9occ9YECzmfw%2F4Q%2Bas%2BFPh9kw0SEfpneiddIuwK
  1ji%2BYSJwCbT3fkbZ2z1056yq5vYZxtiBENRASCfpUEYjBKM7SGO6IIFsYS60LNXSPA1mwR9Max
  reSN5JjWYzbmc1d%2BY8KyEdTAYN99leqkiDTnDRrSonXnX1JBQaE9KmNY%2B1M43ecfV8Mg598I
  HSHSEQwcJK4owF4d2g1YEukv%2BgMhUagSn3xtg4rUPaS0%2BjBL%2B1hpE7u0eq3wwLgxiZhD4w
  GYrxzGgYabJD5CwJ8W2Lgsd8Q1rUCrfQv168rY6unHXyjYJONJwxnLtrnILM352CEYG1n6o2gYJT
  JCO8e6E7DjaCmJQCEILq7j8vgVrtsXEPnsbFU9XD6z24cZ3nSwXcKc%3D--zkMFRPbsdlpgCGxe-
  -w5xo4iQ3IUYyToTH8jTigA%3D%3D"
8 # 将cookie字符串转换为cookie字典
9

```

```

10 # cookie_dict = {cookie.split('=')[0]: cookie.split('=')[1] for cookie in
    cookie_str.split('; ')}
11
12 # 先根据 分号+空格进行分隔
13 cookie_list = cookie_str.split('; ')
14 # 准备一个空字典 用于存储cookie信息
15 cookie_dict = {}
16 # 遍历上述得到的列表
17 for cookie in cookie_list:
18     # 再根据 等号 进行分隔
19     c_list = cookie.split('=')
20     key = c_list[0]
21     value = c_list[1]
22     # 将键值对赋值到字典中
23     cookie_dict[key] = value
24
25 # 2, 发送请求 获取响应
26 response = requests.get(url, cookies=cookie_dict)
27
28 # 将响应的内容保存到一个html文件中
29 with open("github.html", "w", encoding='utf-8') as f:
30     f.write(response.content.decode())

```

2. requests模块设置超时

打开一个网址，由于现在网速不是很好，打开一个网址的时候，慢，（转圈圈），

爬虫中请求url地址的是好，可能现在网络也不是很好，等待了一段时间之后，加载不成功，不想等待了，设置一个超时时间，

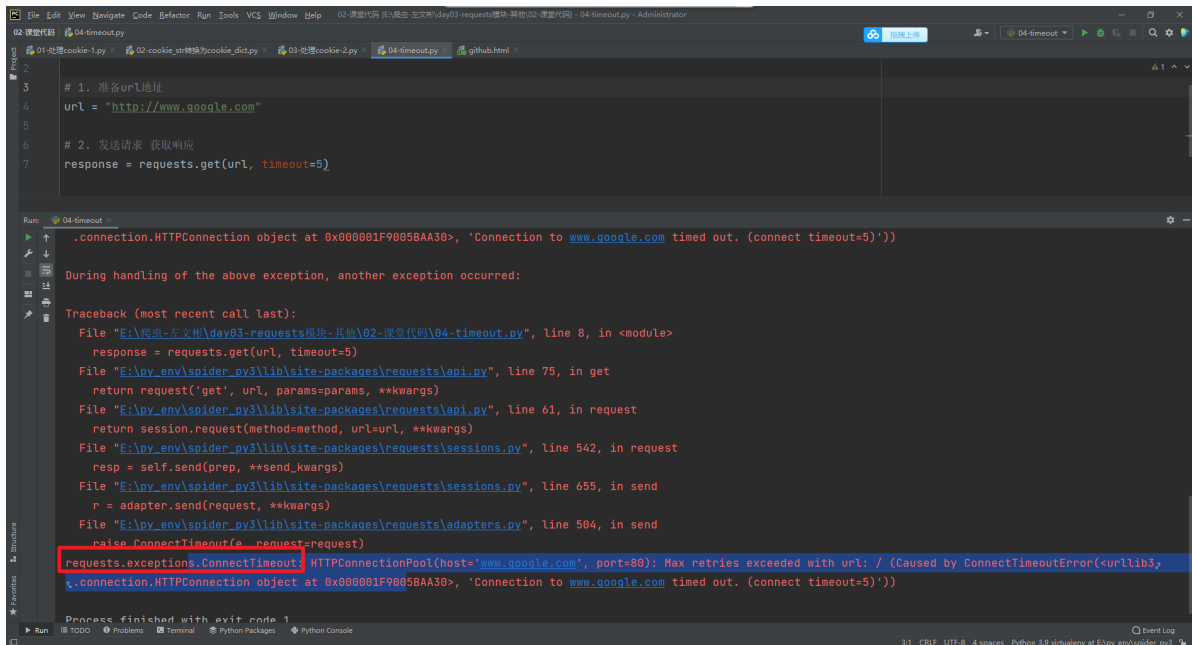
超时时间，等待了一定的时间，如果超出了等待的时间还没有请求成功，程序会抛出异常。

语法：

```

1 response = requests.get(url, timeout=秒数)
2
3 当程序请求一个网站的时候，如果超过了设置的timeout秒数，还没有请求成功，这个时候程序会抛出异常。

```



完整代码

```
1 import requests
2
3 # 1. 准备url地址
4 url_list = ["http://www.google.com", "http://www.baidu.com"]
5
6 # 2. 发送请求 获取响应
7 for url in url_list:
8     # 由于设置了timeout参数，可能其中某一个url地址在请求的时候会超时，程序报错
9     try:
10         response = requests.get(url, timeout=5)
11     except Exception as e:
12         print(f'请求 {url} 的时候 网络超时，请求不成功....')
13     else:
14         print(f"请求 {url} 请求成功，响应状态码是： ", response.status_code)
```

3. 数据提取的概述

3.1 响应数据的分类

响应数据是什么？

响应数据就是我们通过requests模块，发送请求之后，获取到的response响应对象，再使用 `response.text` 或者是 `response.content.decode()` 获取到的内容就是响应的数据。

分类

- 结构化数据： json 或者是 xml
- 非结构化数据： html

数据

json： 就是 js 对象的 字符串表现形式，和Python中的字典特别类似。

- 在json中 引号必须使用双引号
- 在json中的 布尔类型， `true false`，在Python中的 布尔类型 `True False`

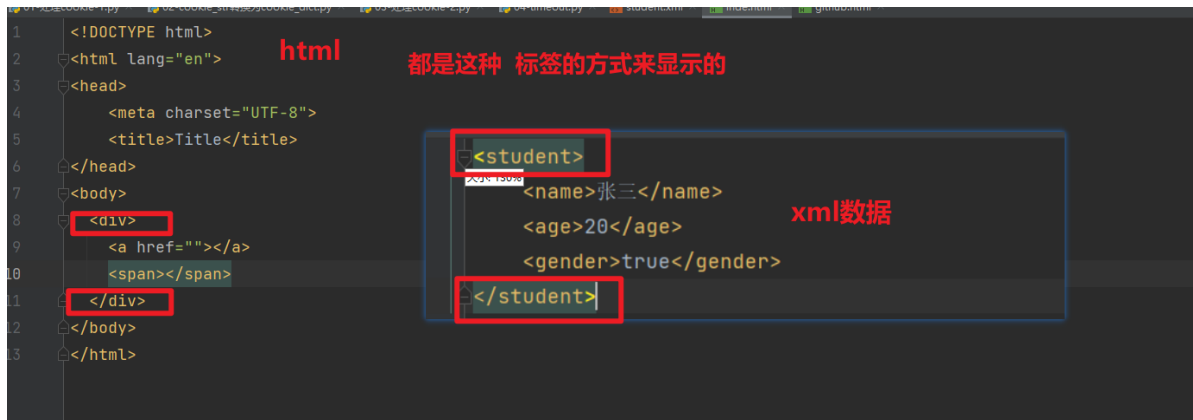
```
1 {
2     "name": "张三",
3     "age": 20,
4     "gender": true
5 }
```

xml: 和html长的特别相似,

```
1 <student>
2     <name>张三</name>
3     <age>20</age>
4     <gender>true</gender>
5 </student>
```

html和xml的区别

- html: 超文本标记语言, 作用: 展示页面, 展示数据
- xml: 可扩展标记语言, 作用: 用于存储和传输数据



3.2 常见的数据解析方法

结构化数据

- json数据: json模块, re正则表达式, jsonpath
- xml: lxml模块+xpath语法, re正则表达式

非结构化数据

- html: lxml+xpath语法, re正则表达式, bs4 (BeautifulSoup4模块), pyquery

4. json模块的使用

4.1 json.loads()

作用: 将json字符串转换为 Python类型 (列表 或者是 字典)

语法:

```
1 import json
2 json.loads(json字符串)
```



```

1  import json
2
3  # 准备一个json字符串
4  json_str = """
5  {
6      "name": "张三",
7      "age": 20,
8      "gender": true
9  }
10 """
11
12 print(json_str, type(json_str))
13
14 # 如何将上述的字符串转换为Python中的字典
15 stu_dict = json.loads(json_str)
16
17 print(stu_dict, type(stu_dict))
18
19

```

为什么要使用该方法，原因就是我们在书写爬虫的时候，发送请求，获取到的响应数据，如果是一个json字符串，如果直接去解析json字符串比较复杂。

我们可以将json转换为Python的字典或者是列表，再根据字典的key去取值就比较简单方便了。

4.2 json.dumps()

作用：可以将 Python类型（列表 字典） 转换为 json

语法：

```

1  import json
2
3  json.dumps(python类型)

```

```

1  import json
2
3  # 准备一个字典
4  stu_dict = {
5      "name": "蔡同学",
6      "age": 23,
7      "gender": True,
8      "like": ["唱", "跳", "rap", "打篮球"]
9  }
10
11 print(stu_dict, type(stu_dict))
12
13 # 现在将上述的学生的字典 转换为 json 字符串
14 # 默认情况下使用dumps方法的时候，如果数据中有中文，在转换时候将中文转换为 ascii编码格式
15 # 不想让其转换为 ASCII 字符。ensure_ascii=False
16 json_str = json.dumps(stu_dict, ensure_ascii=False)
17 print(json_str, type(json_str))
18
19
20 result = str(stu_dict)
21 print(result, type(result))

```

```
06-json.dumps.py
4 stu_dict = {
5     "name": "蔡同学",
6     "age": 23,
7     "gender": True,
8     "like": ["唱", "跳", "rap", "打篮球"]
9 }
10
11 print(stu_dict, type(stu_dict))
12
13 # 现在将上述的学生的字典 转换为 json 字符串
14 # 默认情况下使用dumps方法的时候, 如果数据中有中文, 在转换的时候将中文转换为 ascii编码格式
15 # 不让他转换为 ASCII 字符 ensure_ascii=False
16 json_str = json.dumps(stu_dict, ensure_ascii=False)
17 print(json_str, type(json_str))
18
19
20 result = str(stu_dict)
21 print(result, type(result))

Run: 06-json.dumps
E:\py_env\spider_py3\Scripts\python.exe E:/爬虫-左文彬/day03-requests模块-其他/02-课堂代码/06-json.dumps.py
{'name': '蔡同学', 'age': 23, 'gender': True, 'like': ['唱', '跳', 'rap', '打篮球']} <class 'dict'>
{"name": "蔡同学", "age": 23, "gender": true, "like": ["唱", "跳", "rap", "打篮球"]} <class 'str'>
{"name": '蔡同学', 'age': 23, 'gender': True, 'like': ['唱', '跳', 'rap', '打篮球']} <class 'str'>
Process finished with exit code 0
```

json中的布尔值是 true 和false

json中的引号必须是双引号

4.3 json.load()

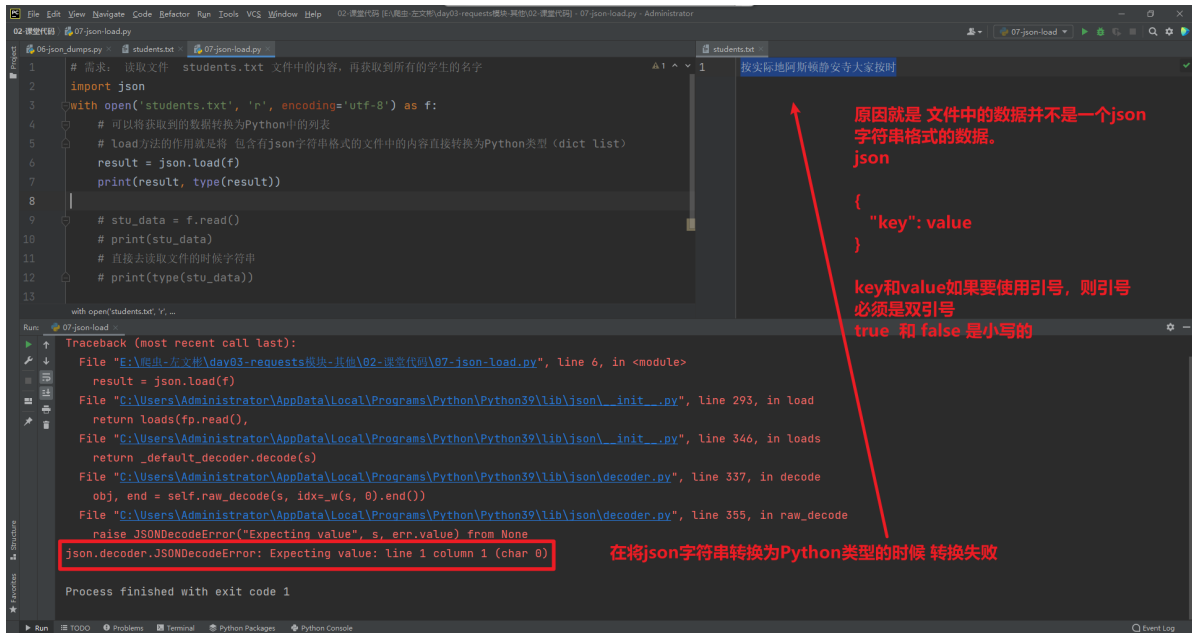
load: 就是将 包含json字符串的文件中的数据 直接转换为Python中的列表或者是字典

```
1 import json
2 # f标示的就是 要读取数据的文件对象
3 json.load(f)
```

```
1 # 需求: 读取文件 students.txt 文件中的内容, 再获取到所有的学生的名字
2 import json
3 with open('students.txt', 'r', encoding='utf-8') as f:
4     # 可以将获取到的数据转换为Python中的列表
5     # load方法的作用就是将 包含有json字符串格式的文件中的内容直接转换为Python类型 (dict
6     result = json.load(f)
7     print(result, type(result))
8
9     # stu_data = f.read()
10    # print(stu_data)
11    # 直接去读取文件的时候字符串
12    # print(type(stu_data))
```

student.txt

```
1 [
2     {
3         "name": "张三",
4         "age": 20
5     },
6     {
7         "name": "李四",
8         "age": 21
9     },
10    {
11        "name": "王五",
12        "age": 22
13    }
14 ]
```

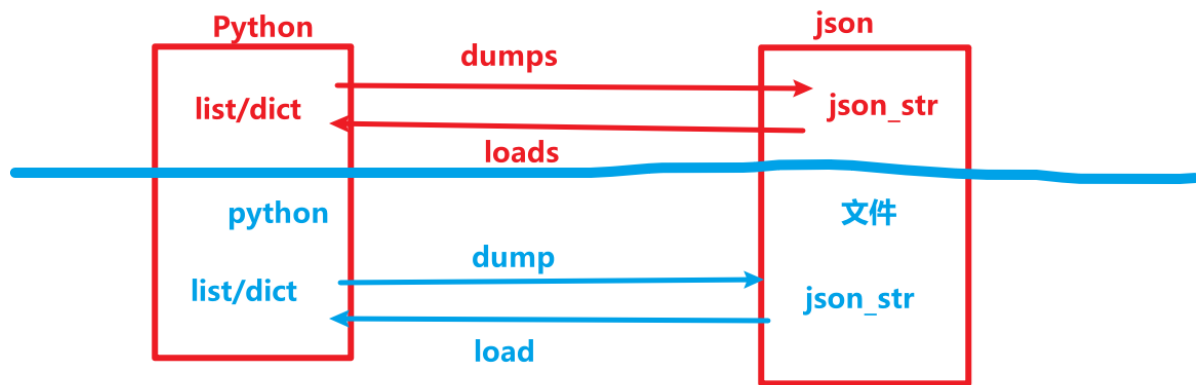


4.4 json.dump()

作用： 可以将Python的列表或者是字典直接写入到json文件中

```
1 import json
2
3 json.dump(f, Python字典或者是列表)
```

```
1 import json
2
3 # 定义了一个列表
4 my_list = [
5     {
6         "name": "张三",
7         "age": 20
8     },
9     {
10        "name": "李四",
11        "age": 21
12    },
13    {
14        "name": "王五",
15        "age": 22
16    }
17 ]
18
19 # 需求: 要讲上述的列表 写入到一个文件中 stu.txt
20 with open('stu.json', 'w', encoding='utf-8') as f:
21     # w 模式只能直接写入 字符串类型的数据, 此时写入的数据my_list 是一个列表类型。
22     # f.write(str(my_list))
23     json.dump(my_list, f, ensure_ascii=False, indent=2)
```



作业:

抓取网址数据: https://movie.douban.com/j/search_subjects?type=tv&tag=%E7%83%AD%E9%97%A8&sort=recommend&page_limit=20&page_start=0

这个url地址返回的数据是一个json数据

要求是

- 对这个url地址发送请求
- 拿到响应数据, json字符串
- 将所有的电视剧的名字获取到, 并且要将这些电视剧的名字写入到一个列表中

1. 准备url地址
2. 发送请求获取响应
3. 获取到响应数据之后, 获取所有电视剧的名字
4. 将所有的电视剧的名字放入到列表
`tv_list = ["扫黑风暴", "云南虫谷", "觉醒年代", "山海情"]`
5. 还需要将 所有 的电视数据 保存到一个 json文件中 `tv.json`

5. xpath语法的使用

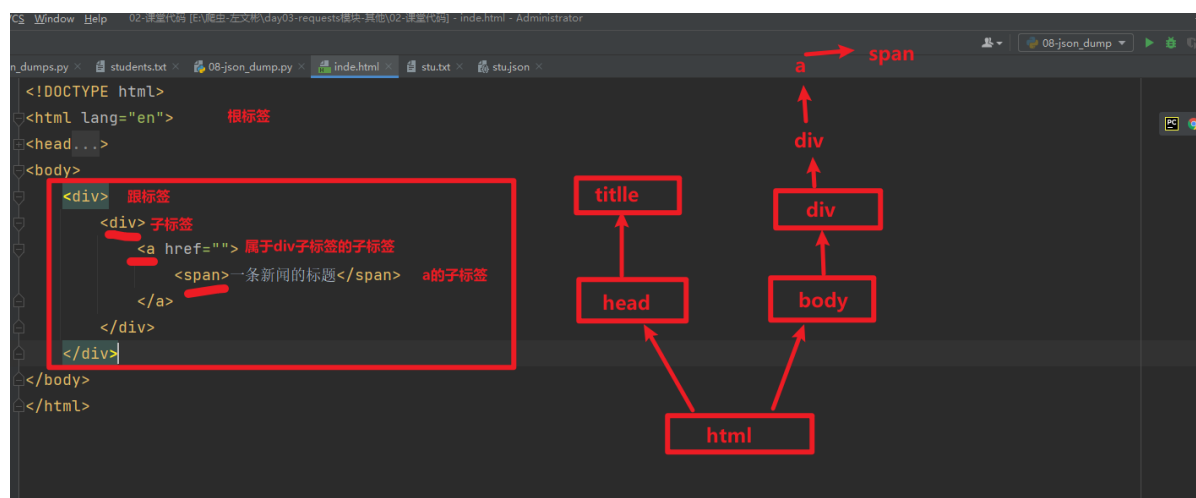
5.1 xpath的介绍

xpath : xml path language 可以在 xml 或者是 html中, 根据标签的路径来定位标签获取标签内容的语法。

标签的路径: 类似于 磁盘的盘符 c ---aa ---bb --cc -- xxx.txt

标签 节点 元素: 这三个内容指的就是 html中的标签。

标签和标签之间的关系:





5.2 xpath_helper插件的安装

准备好 插件的 安装包,

名称	修改日期	类型	大小
 XPath-Helper_v2.0.2.zip	2021/9/1 7:48	WinRAR ZIP 压缩...	247 KB

将压缩包进行解压

名称	修改日期	类型	大小
 XPath-Helper_v2.0.2	2021/9/1 17:01	文件夹	
 XPath-Helper_v2.0.2.zip	2021/9/1 7:48	WinRAR ZIP 压缩...	247 KB

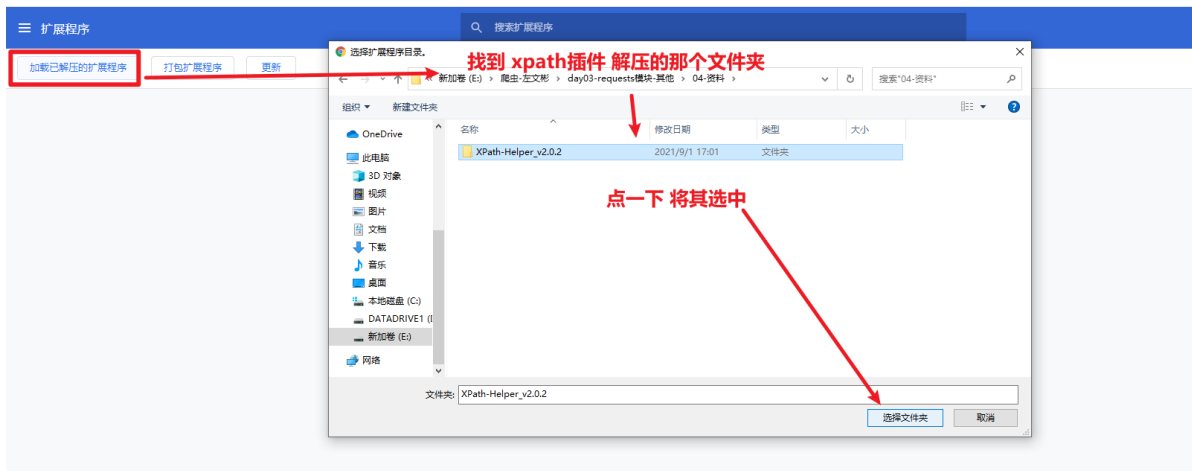
打开自己的谷歌浏览器: 右上角 找到三个小点



将开发者模式打开

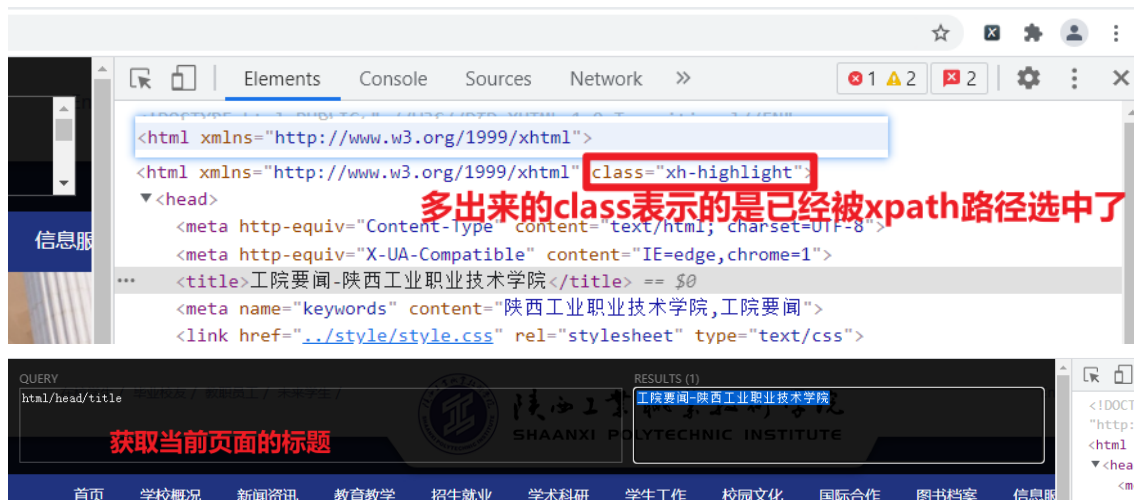


将开发者模式的按钮 打开



5.3 基本语法

- `/` : 第一个含义表示的是 根标签, 第二个含义表示的是标签和标签之间的过渡。



- `.` : 表示的就是 当前标签 (路径)
- `..` : 表示的就是当前标签的父级



- `//`: 表示的是跨标签去定位, 但是有一个缺点, 范围太大, `//a` 就标示的将当前页面中所有的a标签选中。
- `text()`: 获取标签中的文本内容。



- `@属性名`: 根据属性的名字, 去获取该属性对应的值。

5.4 进阶使用

上面学到的基本语法, 在使用 `//` 语法的时候, 范围是整个html中所有的都获取到, 如果我们只想获取到特定的一些标签, 在定位标签的时候加条件。

类似于: `select * from student; //标签`

`select * from student where age = 18;` 在获取数据的时候我只想获取到符合我条件的数据。

- `//标签名[@属性名="属性值"]` 根据标签的属性的值作为条件, 获取到满足条件的标签。



- 根据下标来获取,

- 在xpath中下标是从 1 开始的



- 在xpath中 最后一个元素的下标 `last()`, 倒数第二个就是 `last() - 1`
- `//标签名[text()='文本值']` 根据 标签中的文本内容 作为条件去获取 标签, 一般用于获取下一页的时候去使用。

如果豆瓣不会带cookie

https://api.jinse.com/v6/www/information/list?catelogue_key=%E7%8B%AC%E5%AE%B6&limit=23&information_id=1969948&flag=down&version=9.9.9&source=www

所有的 title 获取到 放到以个列表

再讲 所有的数据保存到 jinse.json