第六关 扮演小红帽：带headers请求

2022年12月8日 14:03

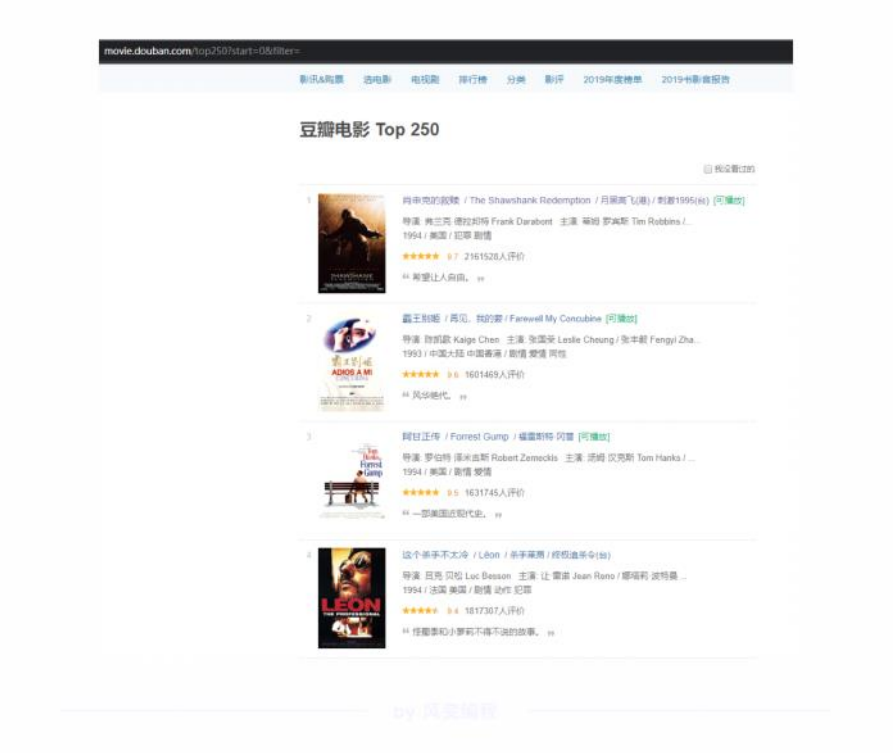
1. 项目代码

展示代码前，我们先要知道需要爬取的网站和信息是什么。

1.1 明确需求

本节课我们需要爬取的目标是豆瓣电影Top250前 100 部电影的信息，这是豆瓣电影 Top250 的网址：https://movie.douban.com/top250?start=0&filter= 。

你可以打开它，简单观察一下。



通过观察网站一页的电影数，可以发现一页只有 25 部电影的信息。

也就是说我们需要爬取网站前4页（100 = 25*4）的电影信息。

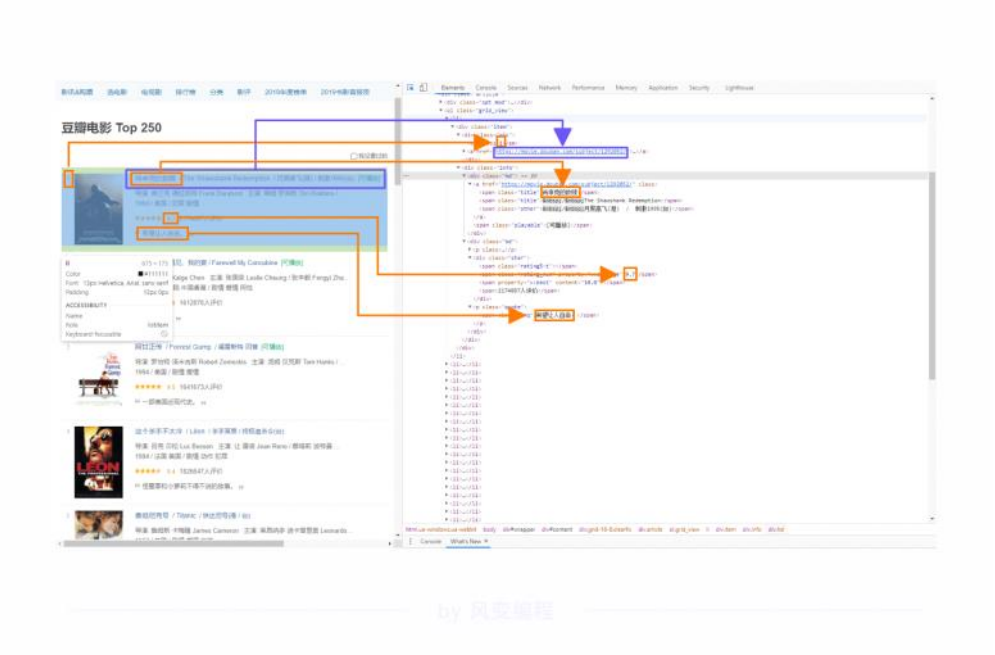
爬取的内容是这4页网页里，每一部电影的序号、电影名、评分、推荐语以及详情链接。

1.2 分析网页

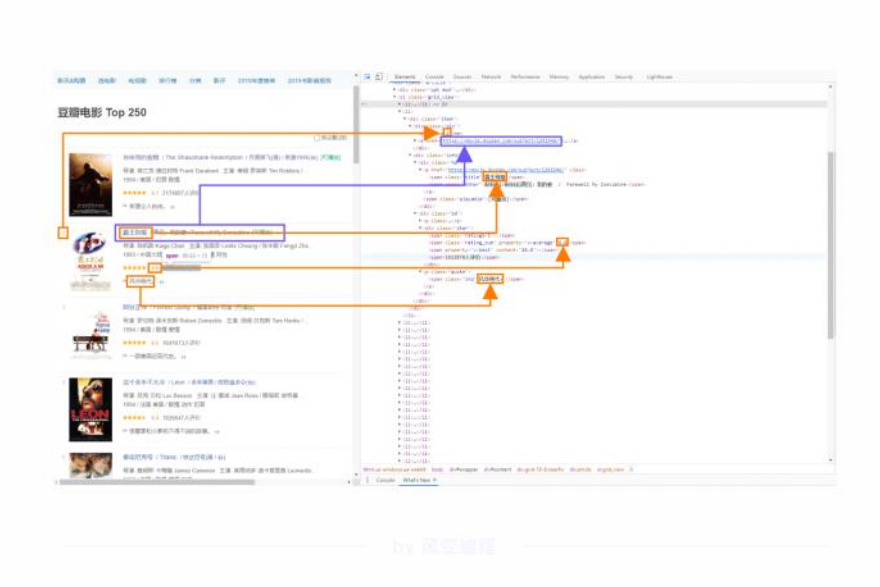
明确了目标后，回到第一页网页，通过快捷键打开网页的开发者工具（Windows 用户可以在浏览器页面下按 Ctrl + Shift + I 键唤出浏览器开发者工具，Mac 用户的快捷键为 command + option + I）。

接着使用开发者工具中的指针工具，大致查看一下前两部电影中，需爬取的信息所在位置，观察一下其中是否有什么规律。

可以发现第一部电影里序号、电影名、评分、推荐语以及详情链接在class属性值为"item"的标签里。



第二部电影里的序号、电影名、评分、推荐语以及详情链接也在class属性值为"item"的标签里。



两部电影包含爬取信息的标签结构十分相似。

接下来体验一下我写好的代码吧。

1.3 体验代码

方法1

```python
import csv
import requests
from bs4 import BeautifulSoup

# 设置列表，用以存储每部电影的信息
data_list = []
# 设置请求头
headers = {
    'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36
    (KHTML, like Gecko) Chrome/85.0.4183.102 Safari/537.36'
}

# 使用 for 循环遍历取值范围为 0~3 的数据
for page_number in range(4):
    # 设置要请求的网页链接
    url = 'https://movie.douban.com/top250?start={}&filter='.format(page_number * 25)
    # 请求网页
    movies_list_res = requests.get(url, headers=headers)
    # 解析请求到的网页内容
    bs = BeautifulSoup(movies_list_res.text, 'html.parser')
    # 搜索网页中所有包含单部电影全部信息的 Tag
    movies_list = bs.find_all('div', class_='item')
```

```python
    # 使用 for 循环遍历搜索结果
    for movie in movies_list:
        # 提取电影的序号
        movie_num = movie.find('em').text
        # 提取电影名
        movie_name = movie.find('span').text
        # 提取电影的评分
        movie_score = movie.find("span",class_='rating_num').text
        # 提取电影的推荐语
        movie_instruction = movie.find("span",class_='inq').text
        # 提取电影的链接
        movie_link = movie.find('a')['href']
```

```
        # 将信息添加到字典中
        movie_dict = {
            '序号': movie_num,
            '电影名': movie_name,
            '评分': movie_score,
            '推荐语': movie_instruction,
            '链接': movie_link
        }

        # 打印电影的信息
        print(movie_dict)
        # 存储每部电影的信息
        data_list.append(movie_dict)

# 新建 csv 文件，用以存储电影信息
with open('movies.csv', 'w', encoding='utf-8-sig') as f:
    # 将文件对象转换成 DictWriter 对象
    f_csv = csv.DictWriter(f, fieldnames=['序号', '电影名', '评分', '推荐语', '链接'])
    # 写入表头与数据
    f_csv.writeheader()
    f_csv.writerows(data_list)
```

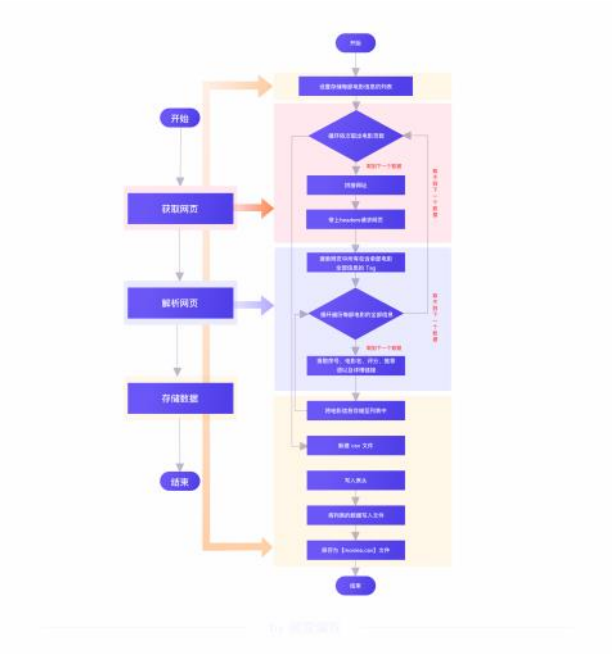
方法2

```
'''https://movie.douban.com/top250?start=0&filter=
爬取的目标是豆瓣电影Top250前 100 部电影的信息
也就是前4页（100 = 25*4）的电影信息
每一部电影的序号、电影名、评分、推荐语以及详情链接
'''

import requests
from bs4 import BeautifulSoup
import csv

movie_list = []  # 存储所有的电影信息

headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36 Edg/108.0.1462.42'}

for number in range(0,76,25) :
    douban_movie_url = f"https://movie.douban.com/top250?start={number}&filter="
    douban_movie_res =requests.get(douban_movie_url, headers=headers)
    douban_movie_bs = BeautifulSoup(douban_movie_res.text, 'html.parser')

    douban_movie_list = douban_movie_bs.find_all('div', class_='item')

    for douban_movie in douban_movie_list:
        movie_number = douban_movie.find('em',class_='').text
        movie_name = douban_movie.find('span', class_='title').text
        movie_rating = douban_movie.find('div', class_='star').find('span',
        class_='rating_num').text
        movie_instruction = douban_movie.find('span', class_='inq').text
        movie_link = douban_movie.find('a')['href']


        movie_dict = {'序号':movie_number, '电影名':movie_name, '评分':movie_rating, '推荐
        语':movie_instruction, '详情链接':movie_link}

        movie_list.append(movie_dict)
    print(movie_list)
```

```
with open("D:\PythonTest\风变python学习资料\Python爬虫\douban_movie_Top100.csv","w",
newline="",encoding="utf-8-sig") as csvfile:
    douban_movie_top100 = csv.DictWriter(csvfile, fieldnames=["序号","电影名","评分","推荐
    语","详情链接"])
    douban_movie_top100.writeheader()
    douban_movie_top100.writerows(movie_list)
    print("保存成功")
```

```
import requests
from bs4 import BeautifulSoup
import csv

movie_list = []  # 存储所有的电影信息

headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/108.0.0.0 Safari/537.36 Edg/108.0.1462.42'}
for number in range(0,76,25) :
    douban_movie_url = f"https://movie.douban.com/top250?start={number}&filter="
    douban_movie_res =requests.get(douban_movie_url, headers=headers)
    douban_movie_bs = BeautifulSoup(douban_movie_res.text, 'html.parser')

    douban_movie_list = douban_movie_bs.find_all('div', class_='item')

    for douban_movie in douban_movie_list:
        movie_number = douban_movie.find('em',class_='').text
        movie_name = douban_movie.find('span', class_='title').text
        movie_rating = douban_movie.find('div', class_='star').find('span',class_='rating_num').text
        movie_instruction = douban_movie.find('span', class_='inq').text
        movie_link = douban_movie.find('a')['href']


        movie_dict = {'序号':movie_number, '电影名':movie_name, '评分':movie_rating, '推荐
语':movie_instruction, '详情链接':movie_link}
        movie_list.append(movie_dict)
    print(movie_list)


with open("D:\PythonTest\风变python学习资料\Python爬虫\douban_movie_Top100.csv","w",newline="",encoding="utf-8-
sig") as csvfile:
    douban_movie_top100 = csv.DictWriter(csvfile, fieldnames=["序号","电影名","评分","推荐语","详情链接"])
    douban_movie_top100.writeheader()
    douban_movie_top100.writerows(movie_list)
    print("保存成功")
```

这是如何实现的呢?
其实爬取的步骤跟你之前学习过的爬虫步骤一样，都是获取网页→解析网页→存储数据三步走。



要实现上面的爬取流程有以下三个难点:
1）如何成功获取 Top250 前 4 页网页信息?
2）如何提取豆瓣 Top250 前 4 页电影的序号、电影名、评分、推荐语以及详情链接?
3）如何存储电影数据?

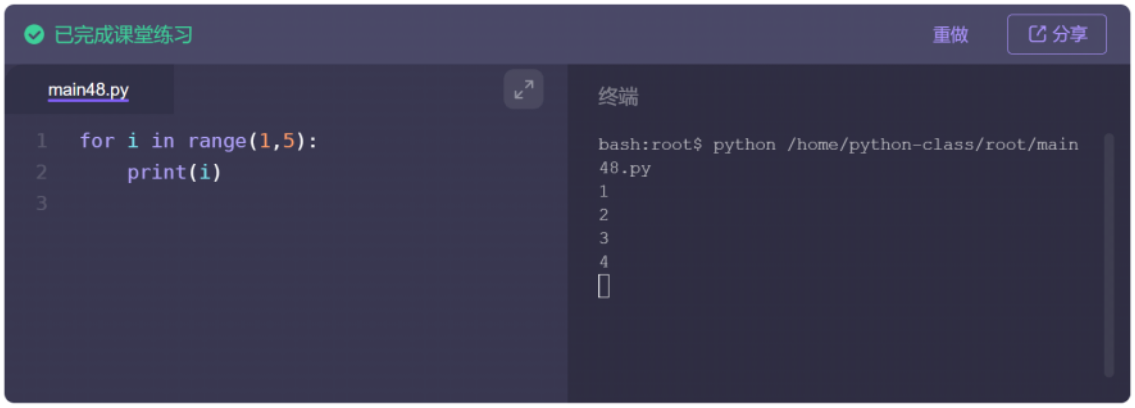
2. 课前复习

在正式讲解代码之前，我们先对之前的部分知识进行复习。

【第一题】range()

```
代码内容:
    for i in [1,2,3,4]:
        print(i)
```

将上面的代码用 range() 改写。



【第二题】find() 与 find_all() 的区别

单选题

以下对 find() 与 find_all() 说法错误的是?

A.find() 方法的返回值是一个Tag对象。

B.find_all() 方法返回的是一个可迭代对象，该对象的结构类似于 Python 中的列表。

C.Tag.find("a")跟Tag.find_all("a")[0]结果相同。

D.Tag对象在使用 find_all() 之后可以直接继续使用 find() 进一步提取。

回答正确

D 错误，find_all() 方法返回的不是 Tag 对象，不能直接使用 find() 或者 find_all() 方法。

【第三题】标签提取

单选题

Tag = Tag.find("div") 的返回结果是
** 变成一道光 **
下面哪个代码可以进一步提取 a 标签里的 class 属性值?

A.Tag.find("a")["class"]

B.Tag["class"]

C.Tag.find("a")["href"]

D.Tag.find("a", class="post-title")["class"]

回答正确

B、方法提取的是 **div** 标签里的 **class** 属性值，错误；
C、标签提取的是 **a** 标签里 **href** 的属性值，错误；
D、 **class** 是python关键字，所以需要在使用find()方法传入 **class** 属性值时需要用 **class_** ，错误。

3. 获取网页

3.1 网页规律分析

现在正式开始我们的爬虫之路。第一步，先查看豆瓣电影 Top250 的 Robots 协议。

```
User-agent: *
Disallow: /subject_search
Disallow: /amazon_search
Disallow: /search
Disallow: /group/search
Disallow: /event/search
Disallow: /celebrities/search
Disallow: /location/drama/search
Disallow: /forum/
Disallow: /new_subject
Disallow: /service/iframe
Disallow: /j/
Disallow: /link2/
Disallow: /recommend/
Disallow: /doubanapp/card
Disallow: /update/topic/
Disallow: /share/
Allow: /ads.txt
Sitemap: https://www.douban.com/sitemap_index.xml
Sitemap: https://www.douban.com/sitemap_updated_index.xml
# Crawl-delay: 5

User-agent: Wandoujia Spider
Disallow: /

User-agent: Mediapartners-Google
Disallow: /subject_search
Disallow: /amazon_search
Disallow: /search
Disallow: /group/search
Disallow: /event/search
Disallow: /celebrities/search
Disallow: /location/drama/search
Disallow: /j/
```

并没有看到Disallow: /Top250，这说明可以对这个网页进行爬取。
继续，获得豆瓣电影 Top250 前 4 页的网址，需要将前4页的网址一个个复制下来爬取吗?
当然不用，学会 Python 的我们要抵制这种重复性劳动。
跟闪光读书网站一样，一个网站有多页需要爬取时，我们可以先找找这些网址名称的规律。
仔细观察豆瓣电影 Top250 前 4 页的网址。

可以发现每一页网址中只有start=后面的数字发生改变，而且很有规律，都是 25 的倍数，分别是 25x0，25x1，25x2 …。
这样我们可以使用循环配合字符串拼接，来实现网址中间局部字符串的修改。
用+进行字符串拼接的代码我已经写好了，直接运行看看。

方法1

```
for page_number in range(4):
    print("https://movie.douban.com/top250?start="+str(25*page_number)+"&filter=")
```

```
https://movie.douban.com/top250?start=0&filter=
https://movie.douban.com/top250?start=25&filter=
https://movie.douban.com/top250?start=50&filter=
https://movie.douban.com/top250?start=75&filter=
```

方法2

```
for number in range(0,76,25):
    douban_movie_url = f"https://movie.douban.com/top250?start={number}&filter="
    print(douban_movie_url)
```

25是步长值

```
https://movie.douban.com/top250?start=0&filter=
https://movie.douban.com/top250?start=25&filter=
https://movie.douban.com/top250?start=50&filter=
https://movie.douban.com/top250?start=75&filter=
```

有了网址，我们就开始用Requests库来发起请求吧。
还记得Requests库如何发起请求的吗？我们先用闪光读书网来熟悉一下这个过程。请你补全下方代码中的第 2、6 行。

```
# 导入requests模块

# 输入网址
url = "https://wp.forchange.cn"
# 发起请求网页
res =
print(res.status_code)
```

```python
# 导入requests模块
import requests
# 输入网址
url = "https://wp.forchange.cn"
# 发起请求网页
res = requests.get(url)
print(res.status_code)
```

```
200
```

现在继续来获取豆瓣电影 Top250 前 4 页的响应状态码，请你补充下方代码的第 7、9 行。

```python
import requests
# 使用 for 循环遍历取值范围为 0~3 的数据
for page_number in range(4):
    # 设置要请求的网页链接
    url = "https://movie.douban.com/top250?start={}&filter=".format(25*page_number)
    #请求网页
    res = requests.get()
    # 打印网页返回的状态码
    print()
```

```python
import requests
# 使用 for 循环遍历取值范围为 0~3 的数据
for page_number in range(4):
    # 设置要请求的网页链接
    url = "https://movie.douban.com/top250?start={}&filter=".format(25*page_number)
    #请求网页
    res = requests.get(url)
    # 打印网页返回的状态码
    print(res.status_code)
```

```
418
418
418
418
```

网页返回的状态码是 418，看来是遇到了问题?
返回码 4xx 时说明我们的GET请求被拒绝了。
这时可以用电脑的浏览器，尝试打开一下豆瓣电影 Top250 网页。

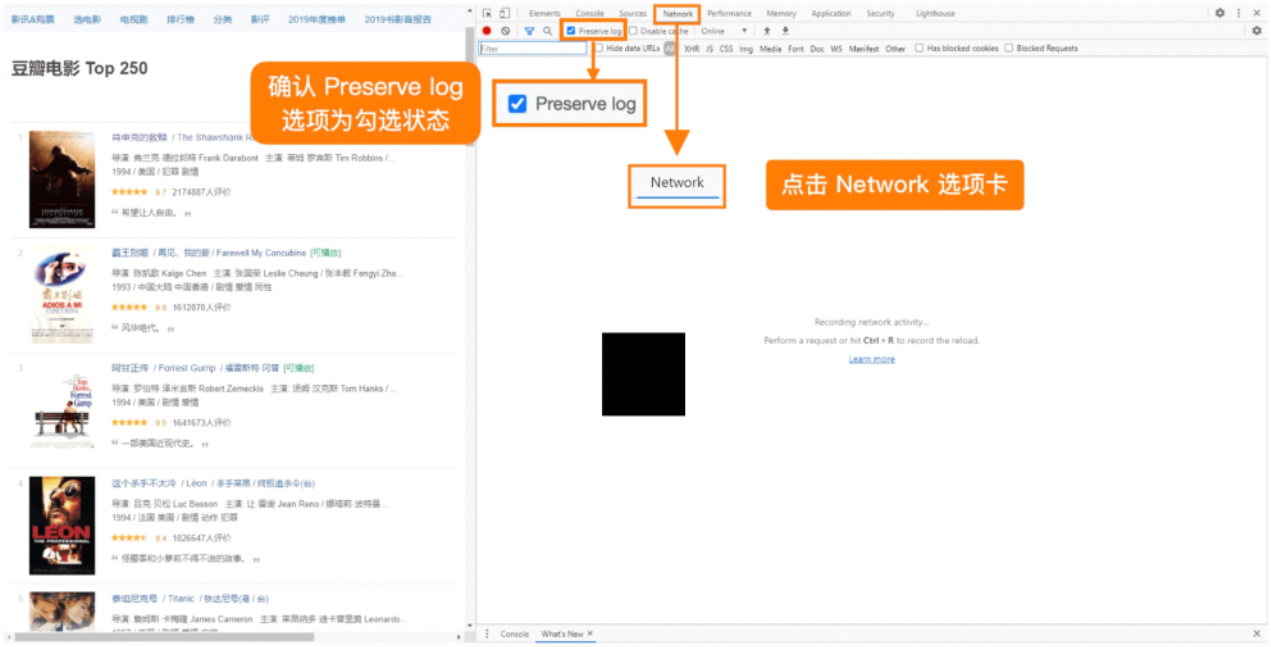
返回码 4xx 时说明我们的GET请求被拒绝了。
这时可以用电脑的浏览器，尝试打开一下豆瓣电影 Top250 网页。
如果本地也无法打开，那么基本上是你本身的网络受到限制。毕竟爬虫需要依赖一个正常的网络环境。
大部分情况下，本地是可以正常浏览豆瓣电影 Top250 的，这里出现 418 的状态码是因为我们被豆瓣电影的反爬程序识破了。

豆瓣网站识别出发送请求的，并不是浏览器，而是我们的程序代码，因此拒绝了我们。
小红帽的故事，相信大部分人都听说过 —— 大灰狼先吃掉了外婆，然后伪装成外婆吃掉了小红帽，最后勇敢的猎人识破了大灰狼，将外婆与小红帽救了出来。
如果外婆能一早认出大灰狼，后面一连串的事故就不会发生。于是，被猎人救出后的外婆引以为戒，提高了自己的防范意识，之后，别的大灰狼再敲门就会被外婆识破。

怎么办?
要给爬虫程序加一些“伪装”，嗯…要怎么加?
不妨给 Python 代码（大灰狼）披上浏览器（小红帽）的大衣，让网页（外婆）给你开门。
在互联网世界中，网络请求会将浏览器信息储存在请求头（Request Header）当中。
只要我们将浏览器信息复制下来，在爬虫程序只要在发起请求时，设置好与请求头对应的参数，即可成功伪装成浏览器。
下面我们就来看看如何获取请求头信息。

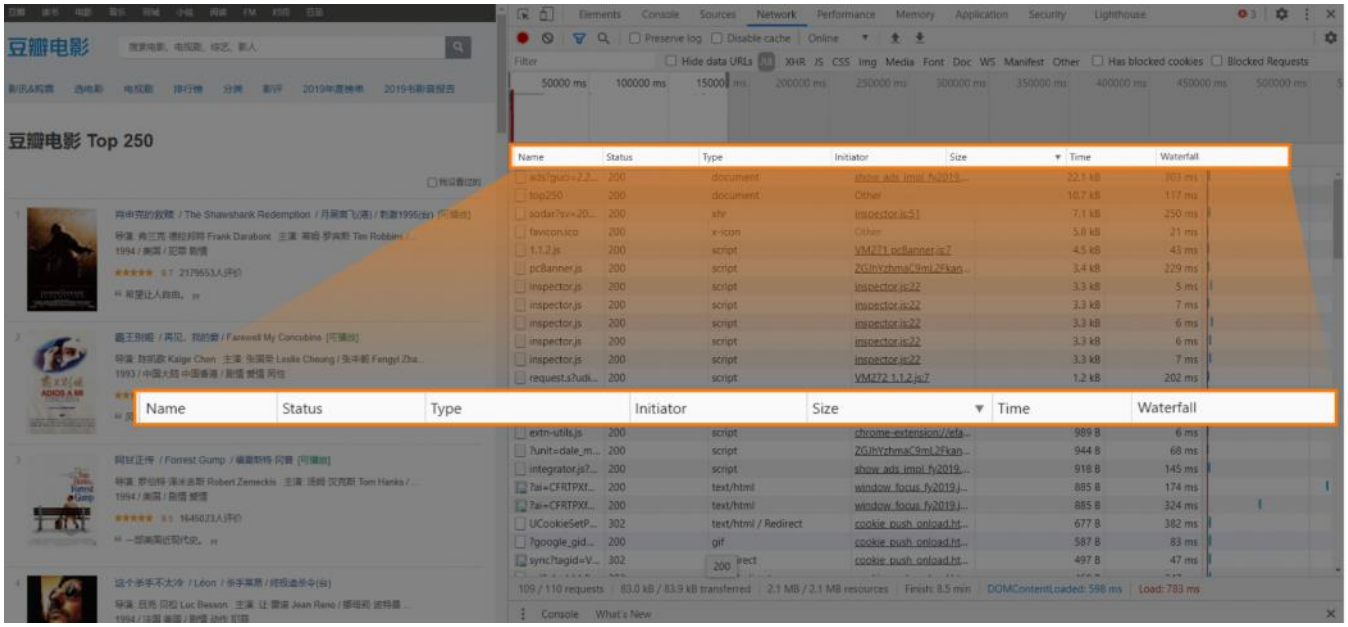
3.2 获取请求头

打开网页的开发者工具（Windows 用户可以在浏览器页面下按 Ctrl + Shift + I 键唤出浏览器开发者工具，Mac 用户的快捷键为 command + option + I）



然后点击Network选项卡，并确认Preserve log选项为勾选状态。这个选项卡可以显示网页中所有加载的信息，包括图片、文本、视频……
如果界面没显示全的话可以将Network界面拉伸到合适位置。
拉伸到适合我们观察的范围后，我们需要在浏览器里刷新一下网页，或者按照图片上的【提示】在当前界面按下对应按键，让网页重新加载内容，这样就可以看到整个网页的全部加载信息了。

拉伸到适合我们观察的范围后，我们需要在浏览器里刷新一下网页，或者按照图片上的【提示】在当前界面按下对应按键，让网页重新加载内容，这样就可以看到整个网页的全部加载信息了。
对于这个界面，重点可以看这两部分。
这一部分区域是Network本身对于网页加载信息格式的筛选功能　。
这些选项将加载信息进行了分类。

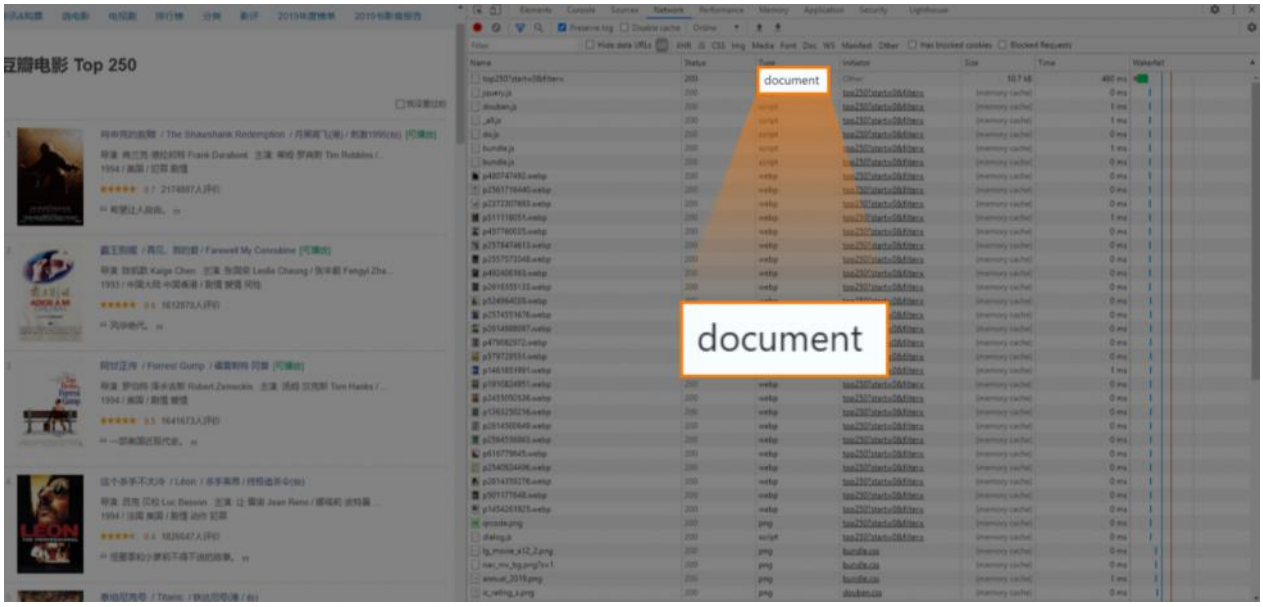


ALL	查看全部
XHR	一般是通过动态请求返回的文件
Doc	Document，第1个请求一般在这里
Img	仅查看图片
Media	仅查看媒体文件
Other	其他
JS和CSS	前端代码，负责发起请求和页面实现
Font	字体
WS和Manifest	网络编程相关知识，无需了解

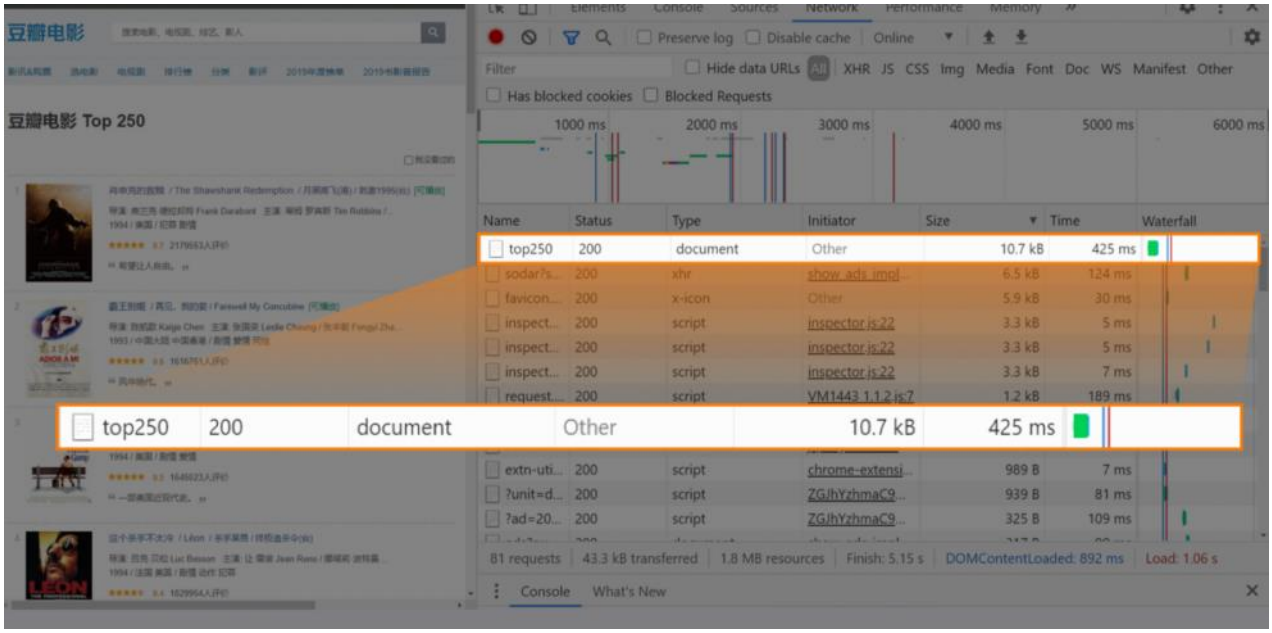
至于这部分则代表着加载文件的一些信息 。
这些信息包括：

Name	请求名称。一般会以 URL 的最后一部分内容当做名称
Status	响应的状态码
Type	请求的文档类型
Initiator	请求源。用来标记请求是由哪里发起
Size	从服务器下载的文件和请求的资源大小
Time	发起请求到获取响应所用的总时间
Waterfall	网络请求的可视化瀑布流

现在，我们重点找的是最先出现的document格式的文件，这一般就是我们看到的网页界面。

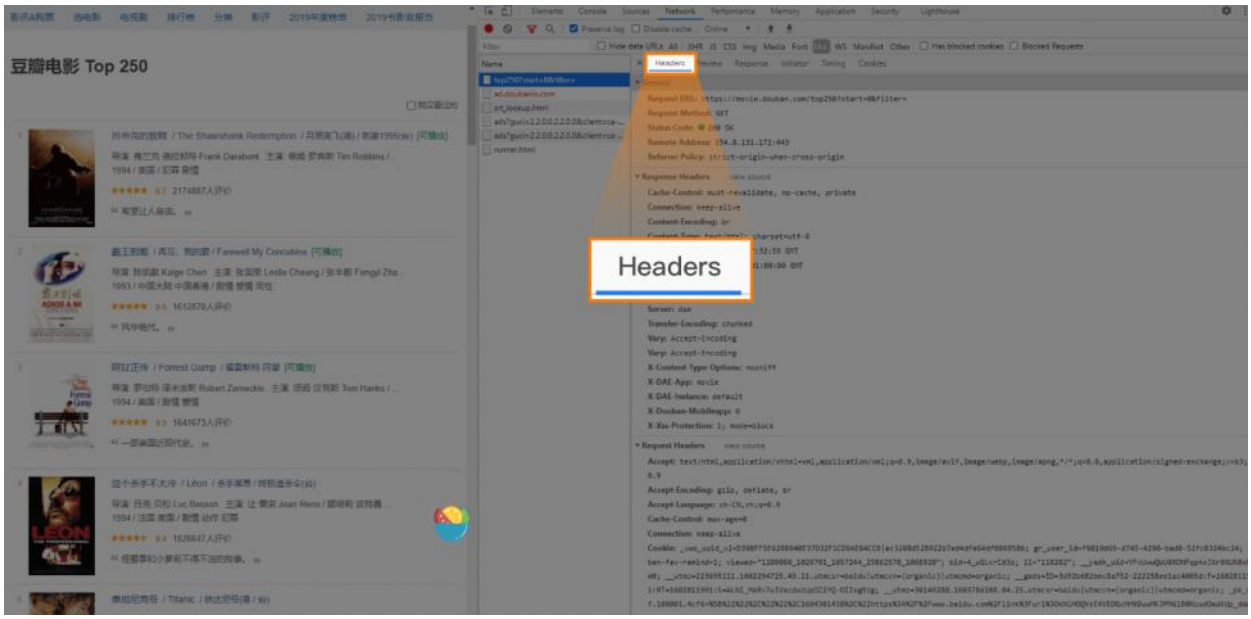


当然，也可以通过筛选功能选择Doc格式的文件，这样在网页加载信息太多时，可以查找更加方便。



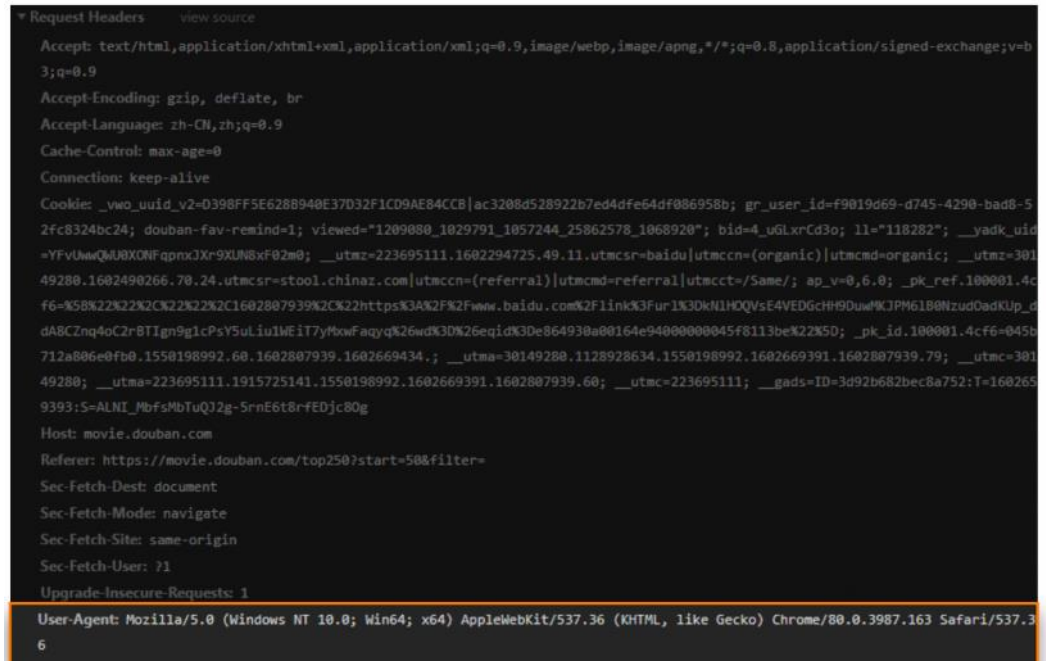
继续，点击这个document文件的name，再点击它的preview选项卡，查看是否含有我们想要爬取的信息。

通过观察可以判断我们找对了文件，再切换到Headers选项卡



暂时先忽略其他信息，重点看Request Headers。

这代表着"请求头"，也就是浏览器证明自己身份的信息。
这些就像小红帽的衣服和其他小装饰品，可以帮助我们进行伪装。
而其中最重要的衣服就是User-Agent这一项，它包括了请求网页时使用的操作系统、浏览器及版本等信息。

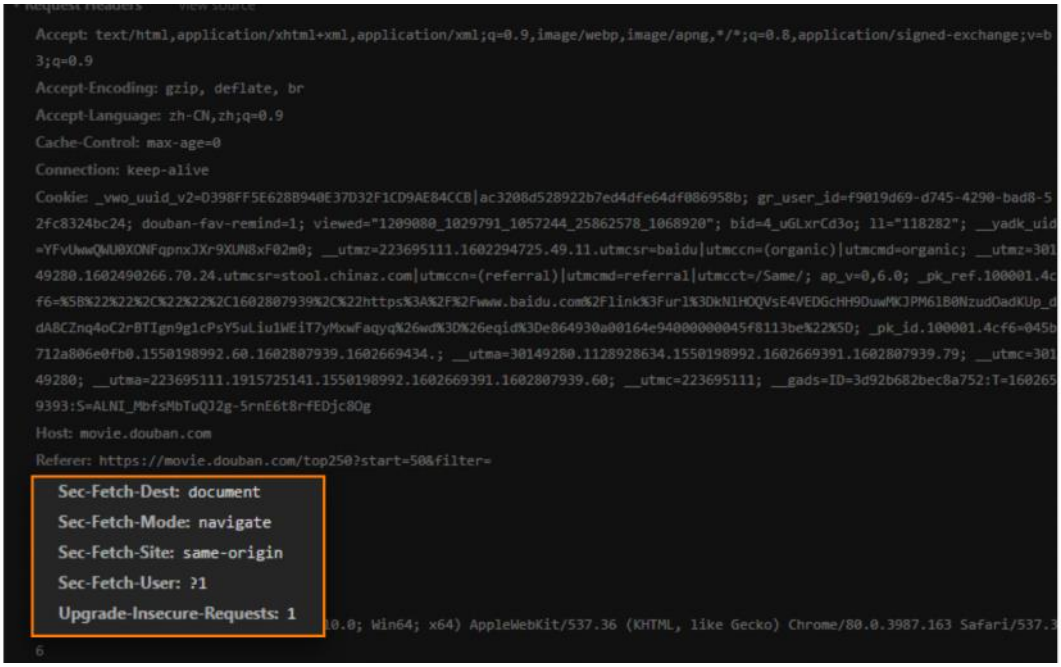


而 Headers 选项卡的其他信息中，有些是各个Headers里都有的参数。
像General中包含了这个请求的网址、响应状态码、请求方式。

在Request Headers中，除了User-Agent，还有Accept、Host、Cookie 等信息。

Accept	指定客户端可以接受哪些类型的信息
Accept-Language	指定客户端可以接受的语言类型
Accept-Encoding	指定客户端可以接受的内容编码
Host	指定请求资源的主机 IP
Cookie	网站为了辨别用户进行会话跟踪而存储在用户本地的数据
Refer	标识请求是由哪个页面发出，服务器可以拿到信息并做相应处理
User-Agent	用以识别客户使用的操作系统及版本、浏览器及版本等信息。加上此信息可以伪装为浏览器，不加可能会被识别为爬虫
Content-Type	表示具体请求中的媒体类型信息

而有些则是网站自己定义的参数



Request Headers中这些信息加上后，可以更加逼真地模拟成真实浏览器。当然，在大部分情况下，不加上也不会被识破。
至于如何将这个信息写进我们的 Python 代码呢，requests.get()函数里，本身就提供了一个 headers参数让我们去传入。
将网页中Request Headers里的信息用一个字典保存起来，然后再将这个字典传递给 get() 函数里的headers参数，就可以让 get() 函数伪装成“小红
帽”的样子去拜访“外婆”。

```python
import requests

# 设置要请求的网页链接
url = 'https://movie.douban.com/top250?start=0&filter='

# 设置请求头
headers = {
    'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko)
     Chrome/85.0.4183.102 Safari/537.36'
}

# 请求网页
res = requests.get(url, headers=headers)

# 打印网页状态码
print(res.status_code)
```

```
200
```

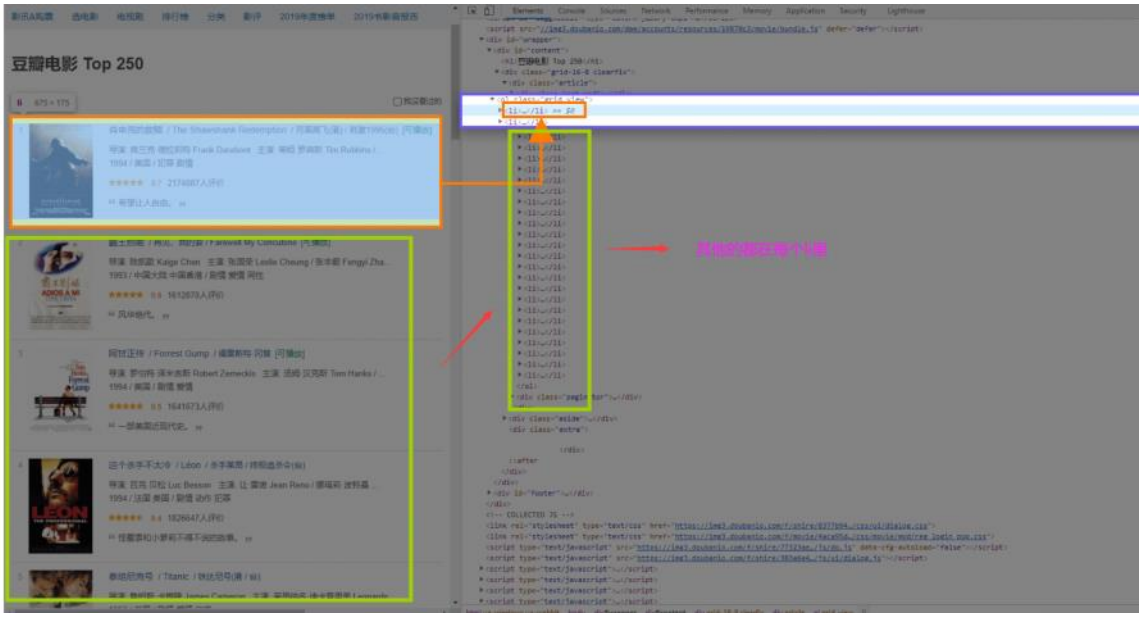
再次回顾一下我们查看 headers 的步骤。

4. 解析网页

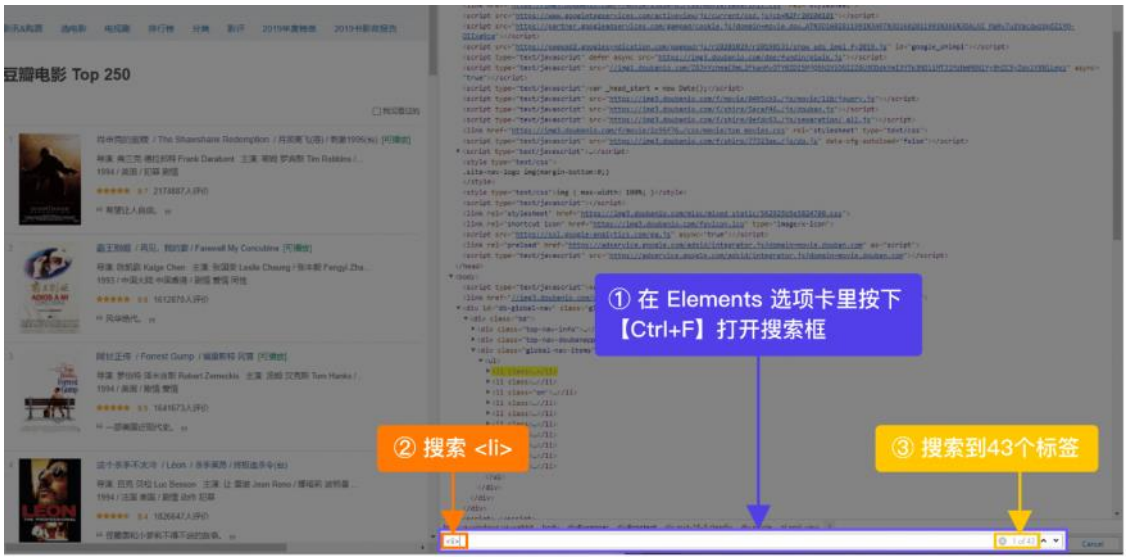
4.1 定位网页数据

在解析网页这一步，有点像是找规律。比如之前闪光图书时每一本图书的全部信息，都在有规律的Tag对象里。而豆瓣电影中，每一部电影包含的全部信息基本也都遵循着规律，在一个 Tag 对象中。

你可以自己用指针工具在网站上移动，看一看对应的标签有什么规律。
发现了吗，下面是我找到的规律，电影1：



可以发现，每一部电影的所有信息都在一个li标签里。
那么我们是否可以通过bs.find_all("li")来提取呢?
li作为一个没有特定属性的标签，很容易有一些其他无关的li标签混入其中。
所以我们需要在Elements选项卡里按下Ctrl+F打开搜索框，在里面搜索<li>，查看结果是不是只有25个。



这种情况我们需要在li标签里另外找一个有属性的标签来精确定位到每部电影。

就决定是图片中 class 属性值为item的div标签了，为了保证严谨性，我们还是再到搜索框中搜索一下。
直接搜索item可能会被网页中，一些属性值里包含 item 的标签干扰。

我们可以给 item 加上引号，直接搜索"item"，这样可以筛选出无关的标签。

4.2 提取网页数据

```
import requests
from bs4 import BeautifulSoup
import csv

movie_list=[]#存储所有的电影信息

headers={'User-Agent':'Mozilla/5.0(WindowsNT10.0;Win64;x64)AppleWebKit/537.36(KHTML,likeGecko)
Chrome/108.0.0.0Safari/537.36Edg/108.0.1462.42'}
for number in range(0,76,25):
douban_movie_url=f"https://movie.douban.com/top250?start={number}&filter="
douban_movie_res=requests.get(douban_movie_url,headers=headers)
douban_movie_bs=BeautifulSoup(douban_movie_res.text,'html.parser')

douban_movie_list=douban_movie_bs.find_all('div',class_='item')

for douban_movie in douban_movie_list:
movie_number=douban_movie.find('em',class_='').text
movie_name=douban_movie.find('span',class_='title').text
movie_rating=douban_movie.find('div',class_='star').find('span',class_='rating_num').text
movie_instruction=douban_movie.find('span',class_='inq').text
movie_link=douban_movie.find('a')['href']


movie_dict={'序号':movie_number,'电影名':movie_name,'评分':movie_rating,'推荐语':movie_instruction,'详情链接':movie_link}
movie_list.append(movie_dict)
print(movie_list)
```

```python
import requests
from bs4 import BeautifulSoup
import csv

movie_list = []  # 存储所有的电影信息

headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
 Chrome/108.0.0.0 Safari/537.36 Edg/108.0.1462.42'}
for number in range(0,76,25) :
    douban_movie_url = f"https://movie.douban.com/top250?start={number}&filter="
    douban_movie_res =requests.get(douban_movie_url, headers=headers)
    douban_movie_bs = BeautifulSoup(douban_movie_res.text, 'html.parser')

    douban_movie_list = douban_movie_bs.find_all('div', class_='item')

    for douban_movie in douban_movie_list:
        movie_number = douban_movie.find('em',class_='').text
        movie_name = douban_movie.find('span', class_='title').text
        movie_rating = douban_movie.find('div', class_='star').find('span',class_='rating_num').text
        movie_instruction = douban_movie.find('span', class_='inq').text
        movie_link = douban_movie.find('a')['href']
```

```python
        movie_dict = {'序号':movie_number, '电影名':movie_name, '评分':movie_rating,
         '推荐语':movie_instruction, '详情链接':movie_link}
        movie_list.append(movie_dict)
    print(movie_list)
```

再来整理一遍我们解析数据的步骤与思路。

1）熟练地使用开发者工具的指针工具，可以很方便地帮助我们定位数据。

2）用指针工具定位到各个数据所在位置后，查看它们的规律。

3）想要提取的标签如果具有属性，可以使用 Tag.find(HTML元素名，HTML属性名='')来提取；没有属性的话，可以在这个标签附近找到一个有属性的标签，然后再进行 find() 提取。

通过上述步骤将信息爬取下来后，就走到我们爬虫的最后一步——存储数据。

5. 存储数据

从之前代码的运行结果可以看到，我们是将爬取到的数据都储存到了新建的 csv 文件中。

所以我们可以使用 csv 模块的 **DictWriter()** 类，接下来我们简单回顾一下这个知识点。

1）调用 csv 模块中类 DictWriter 的语法为：**csv.DictWriter(f, fieldnames)**。语法中的参数 f 是 open() 函数打开的文件对象；参数 fieldnames 用来设置文件的表头；

2）执行 **csv.DictWriter(f, fieldnames)** 后会得到一个 DictWriter 对象；

3）得到的 DictWriter 对象可以调用 **writeheader()** 方法，将 fieldnames 写入 csv 的第一行；

4）最后，调用 **writerows()** 方法将多个字典写进 csv 文件中。

参照上面的步骤，本关代码中关于存储数据的部分就写好了。

（提示一下，使用 Windows 系统的同学在本地练习时，可以给open()函数多加上newline=''的参数，避免空行。）

```python
import csv

# 设置列表，用以在解析网页时存储每部电影的信息
data_list = []
# 新建 csv 文件，用以存储电影信息
with open('movies.csv', 'w', encoding='utf-8-sig') as f:
    # 将文件对象转换成 DictWriter 对象
    f_csv = csv.DictWriter(f, fieldnames=['序号', '电影名', '评分', '推荐语', '链接'])
    # 写入表头与数据
    f_csv.writeheader()
    f_csv.writerows(data_list)
```

6. 程序实现与总结

我们先回顾一下这个任务实现的功能有哪些：

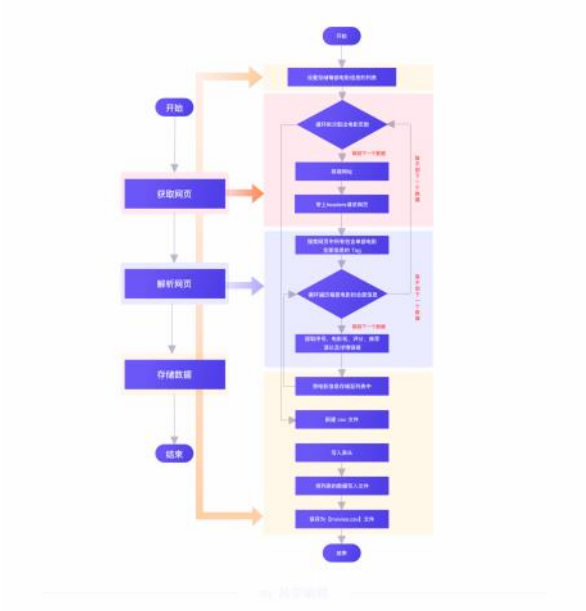
1）通过循环获取网站前 4 页的网页链接；

2）带着 headers 请求网页；

3）解析网页，提取网站前 100 部电影的序号、电影名、评分、推荐语以及详情链接；

4）将爬取到的信息写进 csv 文件中。

我先把网站爬取的流程图展示给你看：



实现项目我们需要运用以下几个知识点

一、获取网页

1.找网页规律；
2.使用 for 循环语句获得网站前4页的网页链接；
3.使用 Network 选项卡查找Headers信息；
4.使用 requests.get() 函数带着 Headers 请求网页。
二、解析网页
1.使用 BeautifulSoup 解析网页；
2.使用 BeautifulSoup 对象调用 find_all() 方法定位包含单部电影全部信息的标签；
3.使用 Tag.text 提取序号、电影名、评分、推荐语；
4.使用 Tag['属性名'] 提取电影详情链接。
三、存储数据
1.使用 with open() as … 创建要写入内容的 csv 文件；
2.使用 csv.DictWriter() 将文件对象转换为 DictWriter 对象；
3.参数 fieldnames 用来设置 csv 文件的表头；
4.使用 writeheader() 写入表头；
5.使用 writerows() 将内容写入 csv 文件。
好了，复习了一遍知识点，下面请你根据右侧代码区的代码注释，复原我们这个任务的代码吧。

```
importcsv
importrequests
frombs4importBeautifulSoup

#设置列表，用以存储每部电影的信息
data_list=[]
#设置请求头
headers={
'User-Agent':'Mozilla/5.0(Macintosh;IntelMacOSX10_14_6)AppleWebKit/537.36(KHTML,likeGecko)Chrome/85.0.4183.102Safari/537.36'
}

#使用for循环遍历取值范围为0~3的数据

#设置要请求的网页链接
url=
#请求网页
movies_list_res=
#解析请求到的网页内容
bs=
#搜索网页中所有包含电影信息的Tag
movies_list=

#使用for循环遍历搜索结果
formovieinmovies_list:
#提取电影的序号
movie_num=
#提取电影名
movie_name=
#提取电影的评分
movie_score=
#提取电影的推荐语
movie_instruction=
#提取电影的链接
movie_link=

#将信息添加到字典中
movie_dict={
'序号':movie_num,
'电影名':movie_name,
'评分':movie_score,
'推荐语':movie_instruction,
'链接':movie_link
}

#打印电影的信息
print(movie_dict)
#存储每部电影的信息
data_list.append(movie_dict)

#新建csv文件，用以存储电影信息
withopen('movies.csv','w',encoding='utf-8-sig')asf:
#将文件对象转换成DictWriter对象
f_csv=csv.DictWriter(f,fieldnames=['序号','电影名','评分','推荐语','链接'])
#写入表头与数据
f_csv.writeheader()
f_csv.writerows(data_list)
```

6.2 知识归纳与总结

本节课新增的知识点主要有两点：
一、Network选项卡的简单使用
1.打开开发者工具（快捷键 + 菜单选择）；
2.点击Network选项卡；

3. 刷新网页，找到目标请求链接；
4. 查看 Request Headers 部分。

二、带着headers请求网页

1. 语法：requests.get(url, headers)。

以下是我们知识点的总结图：