爬虫10-正则表达式+requests实现原生爬虫

来使用正则表达式和requests实现原生的爬虫，不使用BeautifulSoup或者Xpath了。

我们爬取的目标网站是豆瓣电影Top250，获取的内容有电影名称、上映时间、上映地点、电影分类、电影评分。

接下来，我们将分步实现这个功能。

第一步，获取第一页的网页源码并进行预处理：

```
import requests

headers = {
  'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.139 Safari/537.36'
}

url = 'https://movie.douban.com/top250'
response = requests.get(url)
content = response.text.replace(' ', '').replace('\n', '')
 # 这一行表示将网页源码中的空格和换行替换掉，不然有可能会影响到我们的正则匹配，因为网页源码中太多换行和空格了。

```

第二步，从预处理好的网页中提取我们需要的标签，为之后的内容提取做准备：

```
import re

# 第一步：提取出网页源码中class="grid_view"的div下的所有<li>标签，因为里面包含了所有的电影信息

all_tags = re.findall('.*grid_view.*?(<li>.*</li>)', content)[0]

# 这一行中第一个.*表示grid_view前可以是除换行符以外的任意字符，这样就定位到了class="grid_view"这个div标签
```

```
# 第二个.*?表示grid_view与<li>之间可以是除换行符意外的任意字符，同时采用非贪婪匹配，遇见第一个<li>就终止了，否则会一直匹配下去，导致我们最终得到的<li>标签只有最后一个，而不是所有的<li>标签。
# <li>.*</li>表示一个组，以<li>开头，以</li>结尾，中间采用贪婪模式匹配，尽可能多地匹配， 注意这里要用贪婪模式，否则用非贪婪的话，遇见第一个</li>就终止了，这样就只匹配到了第一个<li>标签

 # 第二步：从所有的<li>标签中提取出每个<li>标签：
 movie_tags = re.findall('<li>.*?</li>', all_tags)

 # 这一行中.*?就表示非贪婪模式，尽可能少地匹配，这样才能获取到所有的<li>标签列表，否则获取到的还是原来的字符串。
```

第三步，从所有的<li>标签中提取出具体的内容：

```
import re

# 第一步：提取出电影名称
for movie_tag in movie_tags:
  movie_name = re.findall('.*?class="title">(.*?)<', movie_tag)[0]
  # 这里的.*?和之前的意思是一样的，帮助我们快速定位到class="title"标签
  # 中间的.*?表示非贪婪匹配，匹配>和后面遇到的第一个<之间的内容，电影名称就藏在里面

# 第二步：提取出上映时间、上映地点、电影分类
  other = re.findall('.*?class="bd".*?<br>(.*?)</p>.*', movie_tag)[0]
  # 前面的.*?和之前一样，帮助我们快速定位到class=bd标签
  # 之后的.*?<br>是非贪婪模式，遇见第一个<br>就终止
  # <br>(.*?)</p>表示以非贪婪模式提取出<br>与</p>之间的内容，包含了上映时间、上映地点、电影分类。
  # other的内容是'1994 / 美国 / 犯罪剧情'

  # 从other中提取上映时间
  release_date = int(re.findall('\d+', other)[0])

  # 从other中提取上映地点
  release_place = re.findall('.*?/(.*?)/', other)[0].replace(' ', '')
  # 这一行的第一个.*?表示以非贪婪的模式匹配到第一个/，第二个.*?也是表示以非贪婪的模式匹配/到下一个/之间的内容

  # 从other中提取电影分类
  release_category = re.findall('.*/(.*)', other)[0].replace(' ', '')
  # 这一行的.*表示以贪婪的模式匹配到最后一个/，最后一个/后面的内容就是电影分类
```

```
26
27  # 第三步：提取出电影评分
28   movie_rate = float(re.findall('.*?rating_num.*?>(.*?)<.*?', movie_tag)
[0])
```

第四步，将我们提取的内容组成一个元组，并添加到一个列表中：

```
1  movie_informations = []
2  movie_informations.append((movie_name, release_date, release_place, relea
se_category, movie_rate))
3
```

第五步：获取要爬取的所有URL，保存在列表中：

```
1  url_list = []
2
3  for i in range(10):
4   url = 'https://movie.douban.com/top250?start=' + str(25*i) + '&filter='
5   url_list.append(url)
6
```

第六步：遍历整个url_list，对每个url进行数据提取：

```
import re
import requests

url_list = []

headers = {
'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/66.0.3359.139 Safari/537.36'
}

for i in range(10):
url = 'https://movie.douban.com/top250?start=' + str(25*i) + '&filter='
url_list.append(url)

for url in url_list:
response = requests.get(url, headers=headers)
content = response.text.replace(' ', '').replace('\n', '')
all_tags = re.findall('.*grid_view.*?(<li>.*</li>)', content, flags=re.S)[0]
movie_tags = re.findall('<li>.*?</li>', all_tags)

```

```
    for movie_tag in movie_tags:
movie_name = re.findall('.*?class="title">(.*?)<', movie_tag)[0]
other = re.findall('.*?class="bd".*?<br>(.*?)</p>.*', movie_tag)[0]
release_date = int(re.findall('\d+', other)[0])
release_place = re.findall('.*?/(.*?)/', other)[0].replace(' ', '')
release_category = re.findall('.*/(.*)', other)[0].replace(' ', '')
movie_rate = float(re.findall('.*?rating_num.*?>(.*?)<.*?', movie_tag)[0])
movie_informations.append((movie_name, release_date, release_place, release_category,
    movie_ra
```

第七步：打印出提取结果：

```
for index, movie_information in enumerate(movie_informations):
 # 遍历enumerate(序列)与直接遍历一个序列得到的结果区别在于，enumerate多了一个值，就是index，序列的下标，从0开始。
 movie_name, release_date, release_place, release_category, movie_rate = movie_information
 # 这一行为序列的解包，将序列中的每个值拆出来分别赋予=前面的变量
 print(index+1, movie_name, release_date, release_place, release_category, movie_rate)

以上，就完成了所有的代码了，所有的代码整合一下，就是下面的完整代码：

# 实现功能：利用正则表达式提取电影名称/上映时间/上映地点/电影分类/电影评分
import re
import requests

url_list = []
movie_informations = []

headers = {
 'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.139 Safari/537.36'
}

for i in range(10):
 url = 'https://movie.douban.com/top250?start=' + str(25*i) + '&filter='
 url_list.append(url)

for url in url_list:
 response = requests.get(url, headers=headers)
```

```
    content = response.text.replace(' ', '').replace('\n', '')
    all_tags = re.findall('.*grid_view.*?(<li>.*</li>)', content, flags=re.S)[0]
    movie_tags = re.findall('<li>.*?</li>', all_tags)
    for movie_tag in movie_tags:
    movie_name = re.findall('.*?class="title">(.*?)<', movie_tag)[0]
    other = re.findall('.*?class="bd".*?<br>(.*?)</p>.*', movie_tag)[0]
    release_date = int(re.findall('\d+', other)[0])
    release_place = re.findall('.*?/(.*?)/', other)[0].replace(' ', '')
    release_category = re.findall('.*/(.*)', other)[0].replace(' ', '')
    movie_rate = float(re.findall('.*?rating_num.*?>(.*?)<.*?', movie_tag)[0])
    movie_informations.append((movie_name, release_date, release_place, release_category, movie_rate))

for index, movie_information in enumerate(movie_informations):
    movie_name, release_date, release_place, release_category, movie_rate = movie_information
    # 这一行为序列的解包
    print(index+1, movie_name, release_date, release_place, release_category, movie_rate)

```

这一行为序列的解包，将序列中的每个值拆出来分别赋予=前面的变量

对于我们学到的知识，应该多实践，尤其是编程，一定要多写代码，大家加油。