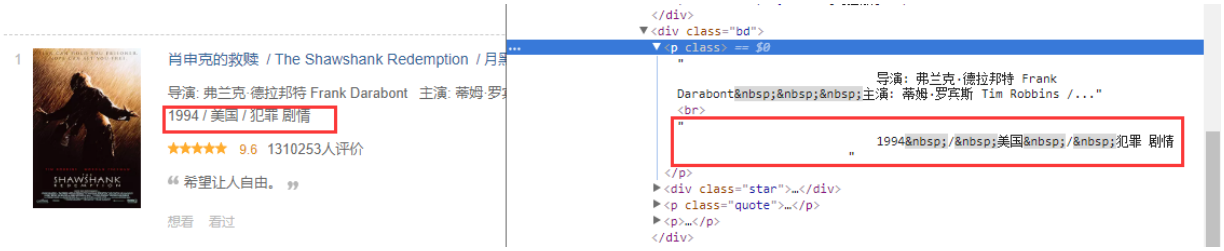
爬虫04-闲谈数据处理

之前我们讨论了一下请求和响应，接下来几天我们都会讨论对数据的处理。

接触了爬虫这个领域，大家肯定都听过正则表达式的鼎鼎大名，不过今天我们暂时不谈正则，我们先来讨论一下数据的简单处理，为之后的正则表达式做准备。

我们用requests.get或requests.post获取到网页的源码，通过BeautifulSoup解析之后，得到的数据还是可能千奇百怪的，可能多了空格，可能有些内容我们不需要等等，所以我们应该对这些数据进行简单的处理。

首先，来看第一种情况，首尾很多空格的情况，下图是豆瓣电影Top250的图，如果我们想要获取电影上映年份/上映地点/电影类别，实际通过BeautifulSoup解析获取到的数据并不是我们想要的结果。




```
movie = '''
                    1994 / 美国 / 犯罪 剧情
                '''
# 我们实际能获取到的数据是这样的，现在我们要去掉首尾多余的空格，可以这么做
movie = movie.strip()
# strip()表示去除首尾的空格，这个对字符串的处理用得非常多
# 得到的结果就是1994 / 美国 / 犯罪 剧情
```

经过strip()处理之后，可以看到得到得结果中仍然还是有空格，这个就可以用replace来替换掉空格。

```
movie = '''
                    1994 / 美国 / 犯罪 剧情
                '''
```

```
movie = movie.strip()
# 得到的结果就是1994 / 美国 / 犯罪 剧情
movie = movie.replace(' ', '')
# 这一行表示将字符串中的空格替换掉，replace第一个参数表示要替换的字符串，第二个参数表示要用什么来替换。
# 得到的结果就是1994/美国/犯罪剧情
```

经过replace()处理之后，其实得到的数据已经可以了，但是，如果我们想分别提取出上映年份/上映地点/电影分类呢。

这个时候就用到我们之前用过的split了。

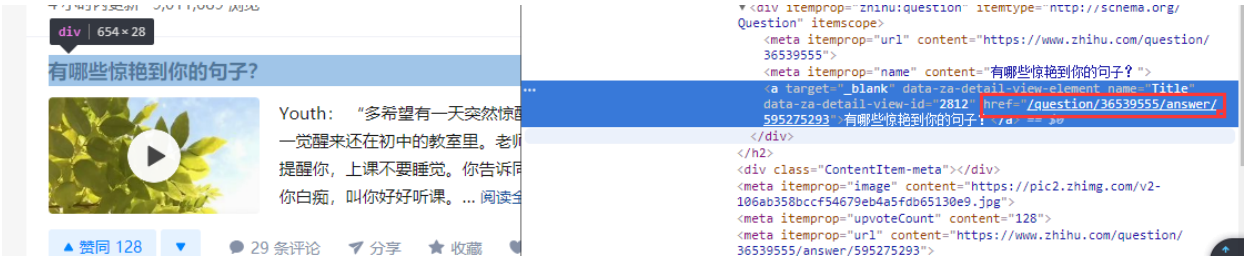
```
movie = '''
                    1994 / 美国 / 犯罪 剧情
                '''
movie = movie.strip()
# 得到得结果就是1994 / 美国 / 犯罪 剧情
movie = movie.replace(' ', '')
# 得到的结果就是1994/美国/犯罪剧情
movie = movie.split('/')
# 这一行表示将字符串通过/进行分割，得到的是一个列表
# 得到的结果是['1994', '美国', '犯罪剧情']
```

通过strip和replace和split就能得到我们想要的结果了，这三种字符串的处理方法会在处理数据中经常使用。

然后我们来看下第二种情况，是关于URL拼接问题的，假设我们现在要爬取知乎首页的超链接，然后顺着爬取到的超链接一直深入爬下去，这样就能爬取整个知乎了。

下图是知乎首页其中一个问题的URL，可以发现这个URL并不完整，需要拼接上当前的域名才能构成一个完整的URL。

我们当然可以用https://www.zhihu.com和这个URL拼接起来，但是这样又有一个问题，我们是想要获取到知乎的所有URL，有的URL是带了https://www.zhihu.com的，这样再拼接一下，就容易出错，所以我们需要预先处理一下。这就需要用到startswith方法了。




```
origin_url = 'https://www.zhihu.com'
url_list = ['/question/36539555/answer/595275293', '/question/308663552/answer/577063117', 'https://www.zhihu.com/special/20743868']

for i in range(len(url_list)):
 if not url_list[i].startswith('http'):
 # 这一行表示如果url_list[i]不是以http开头的话，那么就执行if内部的语句
 url_list[i] = origin_url + url_list[i]

print(url_list)
# 得到的最终结果是['https://www.zhihu.com/question/36539555/answer/595275293', 'https://www.zhihu.com/question/308663552/answer/577063117', 'https://www.zhihu.com/special/20743868']
```

通过startswith提前判断一下，就能够得到我们想要的结果了。当然有startswith，自然而然会有endswith，同样举个例子来看。

```
url_list = ['https://pic2.zhimg.com/50/v2-5502c54842dceeb2e8901e884407a7f_fhd.jpg', 'https://www.zhihu.com/special/20743868']
for url in url_list:
 if url.endswith('jpg'):
 # 这一行表示如果url是以jpg结尾的话，就执行if内部的语句
 url_list.remove(url)
print(url_list)
# 最终结果是['https://www.zhihu.com/special/20743868']
```

通过startswith和endswith可以用来过滤我们不想要的字符串，并对其进行操作。

最后，我们再来谈一个join方法，这个方法是用来拼接一个序列（列表/元组等）的值的，将一个序列转换一个字符串。

下图是豆瓣中一本书的详情页，每本书都有他的标签，有的时候为了方便存储，我们需要将这些标签连起来组成一个字符串，我们就可以用join来操作了。


```
# 上图我们用爬虫去爬取书籍的标签的话，得到的是tag这样的一个列表
tag = ['文学', '短篇小说', '小说', '先锋文学']
tag = '-'.join(tag)
# 这一行表示用'-'符号将tag这个列表中的每个值连接起来，得到的是一个字符串
# 最终结果是'文学-短篇小说-小说-先锋文学'
```

好了，今天的分享本来就到这里结束了，但还是忍不住要插一个列表的去重，因为真的经常会用到。

有这样一个需求，一本书总共有600000个英文单词，保存在了一个列表中，现在想要统计如果想要阅读这本书，需要多少得词汇量，那么我们面对得问题就是去重。

我们直接看代码：

```
# 实现功能：将列表中相同的元素去重，统计书籍词汇量

content = ['Whatever', 'is', 'worth', 'doing', 'is', 'worth',  'doing', 'well']
new_content = set(content)
# 这一步是将列表转换成集合，就去重成功了，因为集合内的元素是不能重复的，但它是无序的
new_content = list(new_content)
# 这一步是将上一步得到的集合转换成一个列表，这样就得到了最终结果列表了
print(new_content)
# 得到的结果是['worth', 'Whatever', 'is', 'doing', 'well']
```

```
print(len(new_content))
# 得到的结果是5，说明这本书的词汇量是5个
```

至此，我们总共讨论了字符串的6种常用方法，分别是strip、replace、split、startswith、endswith、join，以及列表的去重。

明天我们开始来讨论正则表达式，大家加油。