实验3静态网页爬取实训

(源自:https://biglab.site)

(版本:Ver1.2-20230828)

理论：正则表达式

任务描述

通过解析网页可以获取网页包含的数据信息，如文本、图片、视频等，这需要爬虫具备定位网页中信息的位置并解析网页内容的功能。

本任务分别通过Xpath、Beautiful Soup库和正则表达式解析3.1.3小节中通过Requests库获取的网页“http://www.tipdm.com”的网页内容，即获取其中的元素及相关信息。

使用正则表达式匹配字符串。

使用正则表达式查找网页中的标题内容。

正则表达式介绍

- 在编写处理网页文本的程序时，经常会有查找符合某些复杂规则的字符串的需求，而正则表达式正好能满足这一点。
- 正则表达式（Regular Expression，RE），又称为正规表示法或常规表示法，常用于检索、替换符合某个模式的文本。其主要思想为，首先设置一些特殊的字及字符组合，然后通过组合的“规则字符串”来对表达式进行过滤，从而获取或匹配需要的特定内容。
- 正则表达式具有灵活、逻辑性和功能性非常强的特点，能迅速地通过表达式，从字符串中找到所需信息，但对于刚接触的人来说，比较晦涩难懂。

正则表达式模块

正则表达式常见符号

元字符：正则表达式常见符号及其描述如下表

常见符号	描述	示例
literal	匹配文本字符串的字面值literal	foo
re1|re2	匹配正则表达式re1或re2	foo|bar
.	匹配任何字符（除了\n之外）	b.b
^	匹配字符串起始部分	^Dear
$	匹配字符串终止部分	/bin/*sh$
*	匹配0次或者多次前面出现的正则表达式	[A-Za-z0-9]*
+	匹配1次或者多次前面出现的正则表达式	[a-z]+.com
?	匹配0次或者1次前面出现的正则表达式	goo?
{*N*}	匹配*N*次前面出现的正则表达式	[0-9]{3}
{*M*,*N*}	匹配*M* ~ *N*次前面出现的正则表达式	[0-9]{5,9}

常见符号	描述	示例
[...]	匹配来自字符集的任意单一字符	[aeiou]
[..*x*-*y*..]	匹配*x*～*y*范围中的任意单一字符	[0-9], [A-Za-z]
[^...]	不匹配此字符集中出现的任何一个字符，包括某一范围的字符（如果在此字符集中出现）	[^aeiou],A-Za-z0-9
(*|+|?|{})?	用于匹配上面频繁出现/重复出现符号的非贪婪版本（*、+、?、{}）	.*?[a-z]
(...)	匹配封闭的正则表达式，然后另存为子组	([0-9]{3})?,f(oo|u)bar

正则表达式特殊符号

正则表达式特殊符号及其描述如下表

特殊字符	描述	示例
\d	匹配任何十进制数字，与[0-9]一致（\D与\d相反，不匹配任何非数值型的数字）	data\d+.txt
\w	匹配任何字母与数字字符，与[A-Za-z0-9_]相同（\W与之相反）	[A-Za-z_]\w+
\s	匹配任何空白字符，与[\n\t\r\v\f]相同（\S与之相反）	of\sthe
\b	匹配任何单词边界（\B与之相反）	\bThe\b
\N	匹配已保存的子组N（参见表315的符号(...)）	price: \16
\c	逐字匹配任何特殊字符c（即仅按照字面意义匹配，不匹配特殊含义）	.,\,*
\A（\Z）	匹配字符串的起始（结束）	\ADear

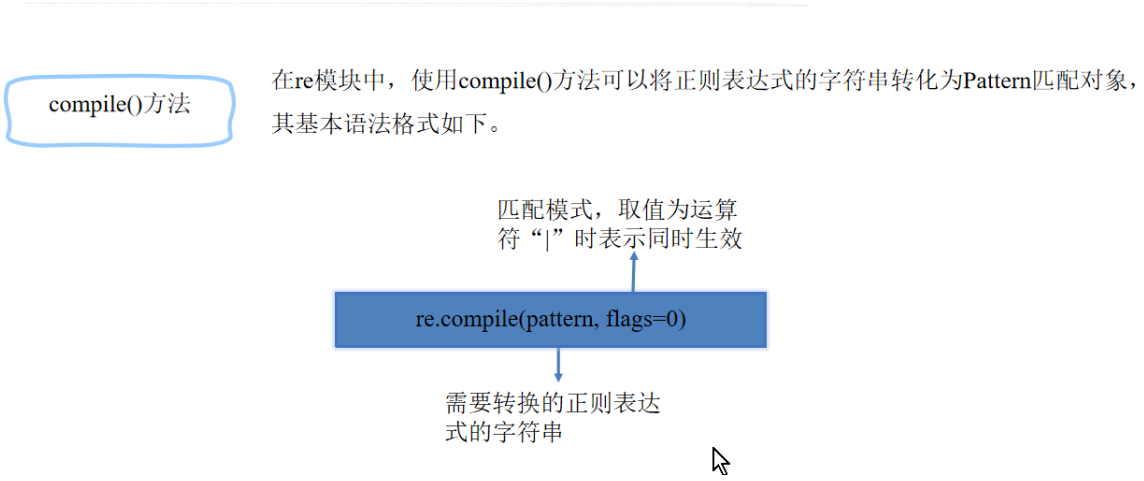
正则表达式常用的方法

使用re模块的步骤为：首先，将正则表达式的字符串编译为Pattern实例；其次，使用Pattern实例处理文本并获得匹配结果（一个Match实例）；最后，使用Match实例获得信息，并进行其他的操作。在re模块中，常用的方法及其说明如下表。

方法名称	方法名称
compile()	将正则表达式的字符串转化为Pattern匹配对象
match()	将输入的字符串从头开始对输入的正则表达式进行匹配，如果遇到无法匹配的字符或到达字符串末尾，那么立即返回None，否则获取匹配结果
search()	将输入的整个字符串进行扫描，对输入的正则表达式进行匹配，并获取匹配结果，如果没有匹配结果，那么输出None
split()	以能够匹配的字符串作为分隔符，将字符串分割后返回一个列表
findall()	搜索整个字符串，返回一个包含全部能匹配子串的列表
finditer()	与findall()方法的作用类似，以迭代器的形式返回结果
sub()	使用指定内容替换字符串中匹配的每一个子串内容

compile()方法

功能：将正则表达式的字符串转化为Pattern匹配对象

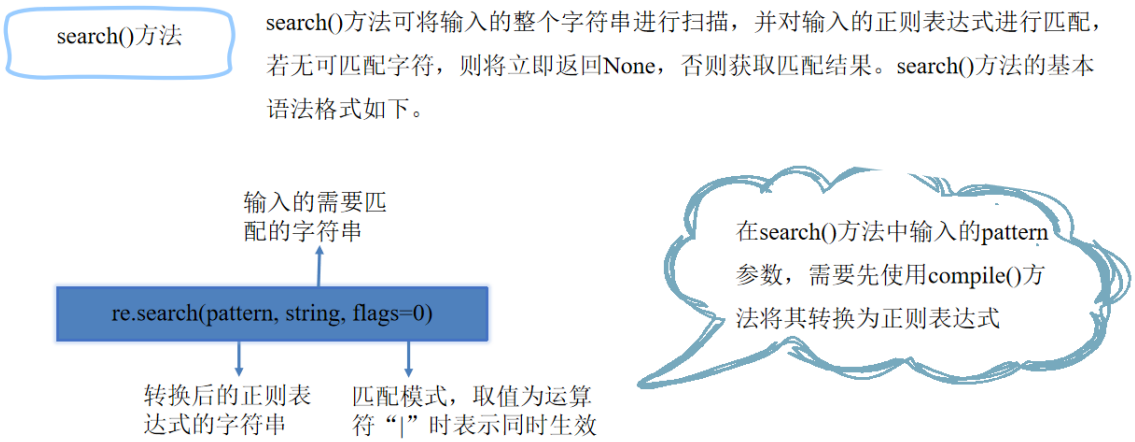


flag参数的可选值如下表。

可选值	说明
re.I	忽略大小写
re.M	多行模式，改变“^”和“$”的行为
re.S	将“.”修改为任意匹配模式，改变“.”的行为
re.L	使预定字符类\w\W\b\B\s\S，取决于当前区域设定
re.U	使预定字符类\w\W\b\B\s\S\d\D，取决于Unicode定义的字符属性
re.X	详细模式，该模式下正则表达式可为多行，忽略空白字符并可加入注释

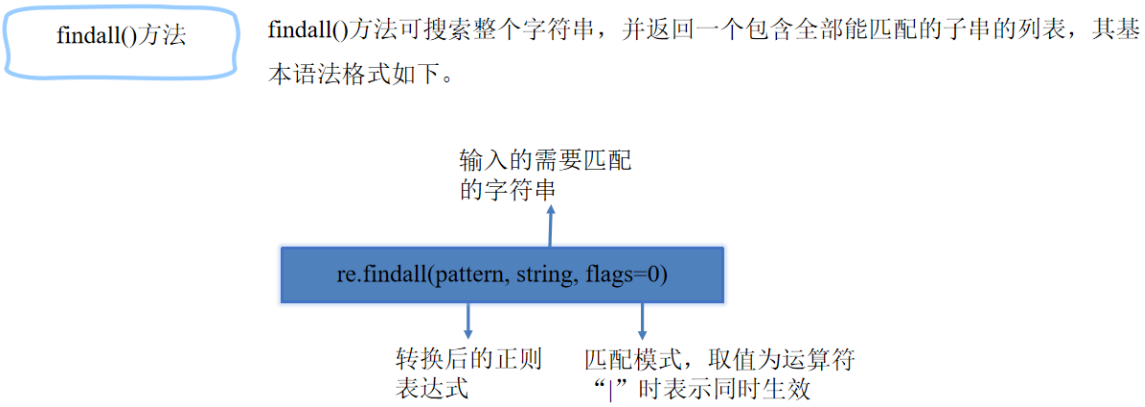
search()方法

功能：search()方法可将输入的整个字符串进行扫描，并对输入的正则表达式进行匹配，若无可匹配字符，则将立即返回None，否则获取匹配结果. 注意:**如果string中存在多个pattern子串，只返回第一个.**



findall()方法

功能：findall()方法可搜索整个字符串，并返回一个包含**全部能匹配**的子串的列表



课堂互动1

分别使用re模块中的search()方法和findall()方法查找网页内容中的title内容，即获取网页中的标题内容.

```
import requests
import re

headers = {
    "user-agent": "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.125 Safari/537.36"
}
url = "http://www.baidu.com/s?wd=李某某"

res = requests.get(url=url, headers=headers)
res.encoding = 'utf-8'
html_data = res.text
# pat = re.compile(r'\d+')   # 转换用于匹配数字的正则表达式

pat_href = re.compile('href="([^"]*?)"') #匹配链接地址
```

```
15
16  pat_myname = re.compile('<em>李某某</em>([^"|^<]*?)"') #按名字匹配
17
18  result = re.search(pat_href, html_data)
19  print(result.group(0))
20
21  result1 = re.findall(pat_myname, html_data)
22  print(result1)
23
24  result2 = re.findall(pat_href, html_data)
25  print(result2)
26
27
```

实验1爬取蓝桥全站课程标题

本实验首先对 requests 库下面的高频方法 get 进行分析讲解，之后将重点介绍 re 模块与正则表达式之间的密切联系，基础知识具备之后，我们将对实验楼全站课程标题内容进行爬取与存储。

知识点

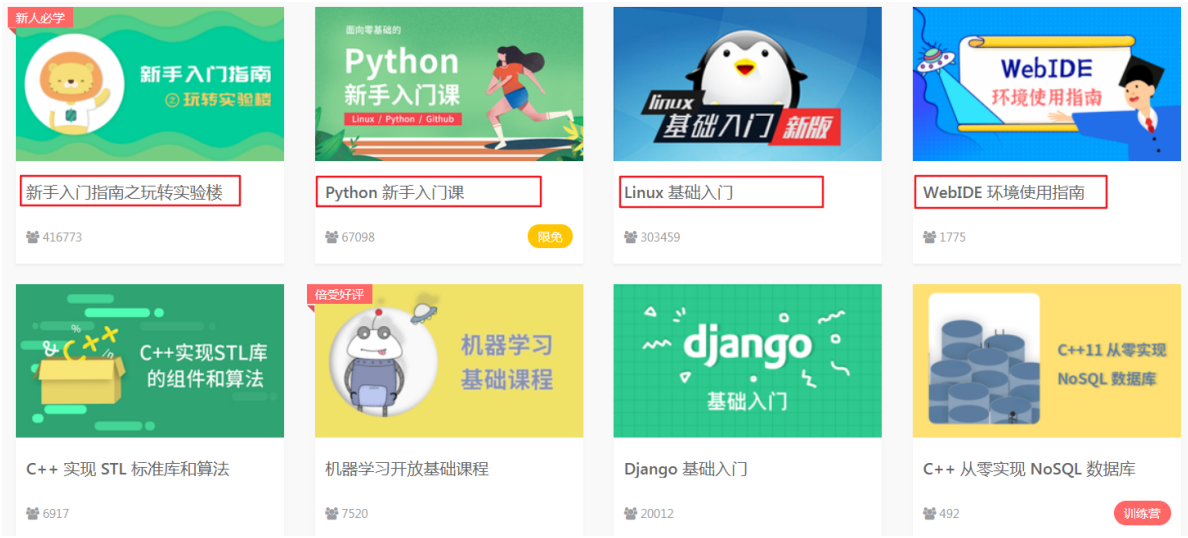
- requests 库 get 方法详解
- re 模块与正则表达式
- re 模块解析实验楼课程页
- 爬取数据存储

Request方法详解

requests 库 get 方法详解

get 方法常用参数介绍

本实验首先从 `requests` 库的 `get` 方法开始，逐步为大家介绍爬虫的最基本知识与原理，实验的最终目标将获取 蓝桥云课课程页 页面所有的课程标题，也就是下图红框所示区域。为了避免目标网站更新导致课程内容失效，我们在本节中使用 `https://labfile.oss.aliyuncs.com/courses/3086/lanqiao.html` 进行实验。



`requests.get` 方法是一种发起网络请求的方式，如果你对 HTML 和 JS 语言有所了解，这部分内容应该非常熟悉，和 `get` 方法类似的还有 `post`、 `put`、 `delete`、 `head`、 `options`，本系列实验中主要应用 `get` 与 `post`，其余内容大家可以自行扩展学习。

对于一个方法而言，最重要的就是它的参数，在 `get` 方法中，主要参数如下：

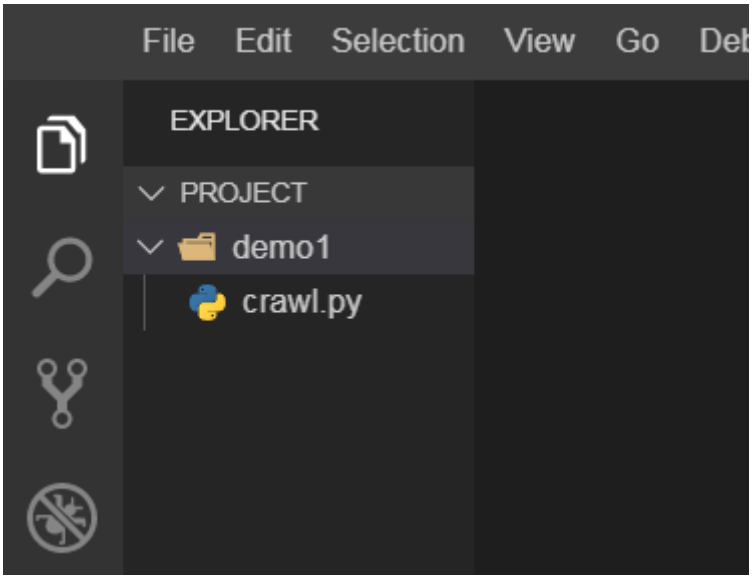
- url， 请求地址，必填项
- headers，请求头，非必填
- params，请求参数，非必填
- proxies，代理 IP，非必填
- verify，SSL 验证，非必填，主要用在访问 https 协议的网站
- timeout，延迟限制，非必填
- cookies，网页 cookies，非必填

以上参数中只有 url 参数是必填项，所以对于访问一个网站网址，你可以直接采用下面的代码进行首次尝试。如果不能获取到数据，在考虑增加其它参数，常见的操作是增加请求头，增加代理，增加 cookies 等。

比如在本实验中，对于 蓝桥云课课程页 可以先做初步尝试，首先创建所需目录和文件（上节实验已经创建）：

```
cd ~/../project
mkdir demo1
cd demo1
touch crawl.py
```

目前的目录结构如下图所示：



然后在 `crawl.py` 文件中编写如下代码：

```
# 导入库
import requests

url = "https://labfile.oss.aliyuncs.com/courses/3086/lanqiao.html"
# 发送数据请求
res = requests.get(url)
print(res)
```

进入 demo1 文件夹，通过 python 命令对 python 文件进行编译，命令如下：

```
python3 crawl.py
```

如果运行代码之后，输出下图所示内容，表示已经请求到数据：

get 核心参数 headers

增加请求头最大的价值是可以让爬虫抓取数据时更加像浏览器在进行访问，针对此，首先增加的是 UA 参数，全称叫做 `user-agent` 。

第一讲提及过一个名词叫做反爬，就是被爬取的网站会采取一些措施，防止自己的数据被爬虫获取，而最常见的措施之一，就是检测请求头中的 UA 参数。UA 参数是一个特殊字符串，简单说，它会携带用户使用的操作系统版本、CPU 类型、浏览器版本、浏览器语言等等内容提供给服务器，服务器依据这些内容判断该请求是否是正常请求，甚至可以通过 UA 去检测一个请求是否在同一个时间内高频率的访问。如果出现一个相同的 UA 高频率的访问网站，系统会自动判断该请求用户为机器人，阻止继续请求。

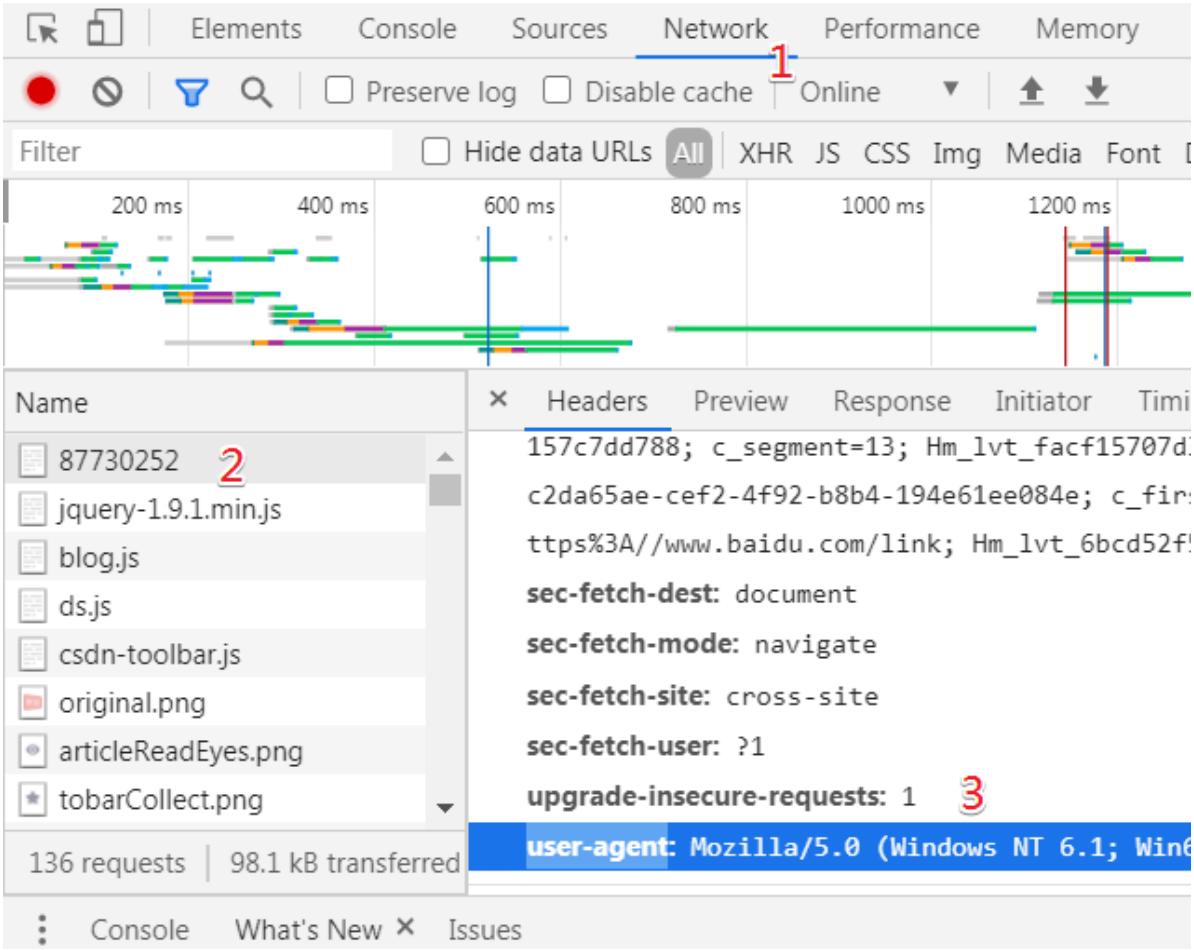
当然 UA 参数携带这些信息还是有其它好处的，很多网站会根据 UA 携带的操作系统和浏览器版本，匹配响应的网站，这样我们可以轻易伪造出各种 UA 参数，例如可以通过 UA 伪造成 Android 手机访问一个网站，让服务器返回手机 Web 站点，很多时候，手机 Web 站点是爬取数据的重要突破点。

UA 查看方式非常简单，使用谷歌或火狐浏览器，打开开发者工具。

开发者工具打开方式有下面三种办法：

1. 打开谷歌浏览器之后按键盘上的 F12 。
2. 通过谷歌浏览器右上角 `...` 进入设置，在弹出的菜单选择**更多工具**，在选择开发者工具也可打开。
3. 通过快捷键 Ctrl + Shift + I 也可打开。

在开发者工具中切换到 NetWork 面板，在下图所示区域找到你的浏览器 UA。



headers 接收的是一个字典类型的变量（Python 的一种变量类型），参照代码如下：

```
# 导入模块
import requests

# 请求头设置
headers = {
    "user-agent": "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.125 Safari/537.36"
}
# 请求地址
url = "https://labfile.oss.aliyuncs.com/courses/3086/lanqiao.html"
# 发起请求，并返回数据
res = requests.get(url=url, headers=headers)
print(res)
```

上述 headers 部分代码可以使用你自己检索到的 UA。

注意上述代码，还有一个小细节，`get` 方法中的参数，都增加了参数名，也就是 `url=url` 前面的 url 表示的是参数名，后面的 url 表示的变量。

headers 字典中还可以增加其它内容，例如增加 `Host`，`Referer`，`Accept` 等内容。在后续实验中，将逐步涉及，这些参数的核心目的是尽量让爬虫程序看起来像一个正常的访问请求，让网站将正确的数据返回给我们。

proxies 参数用于设置请求代理 IP，如果大量的访问一个网站，有时电脑的 IP 会被封禁掉，此时该参数就会起到作用，可以在每次访问的时候都通过一个代理 IP 进行请求，防止服务器封杀 IP 操作。

`get` 方法在后续的课程中是出现频率最高的方法，大家可以提前将所有的参数都预实验一遍，熟悉代码写法

课堂互动1

练习1：参考上述源码，将下面的源码存在crawl_myname.py，然后执行，将源码和运行结果截图到随堂练习中。

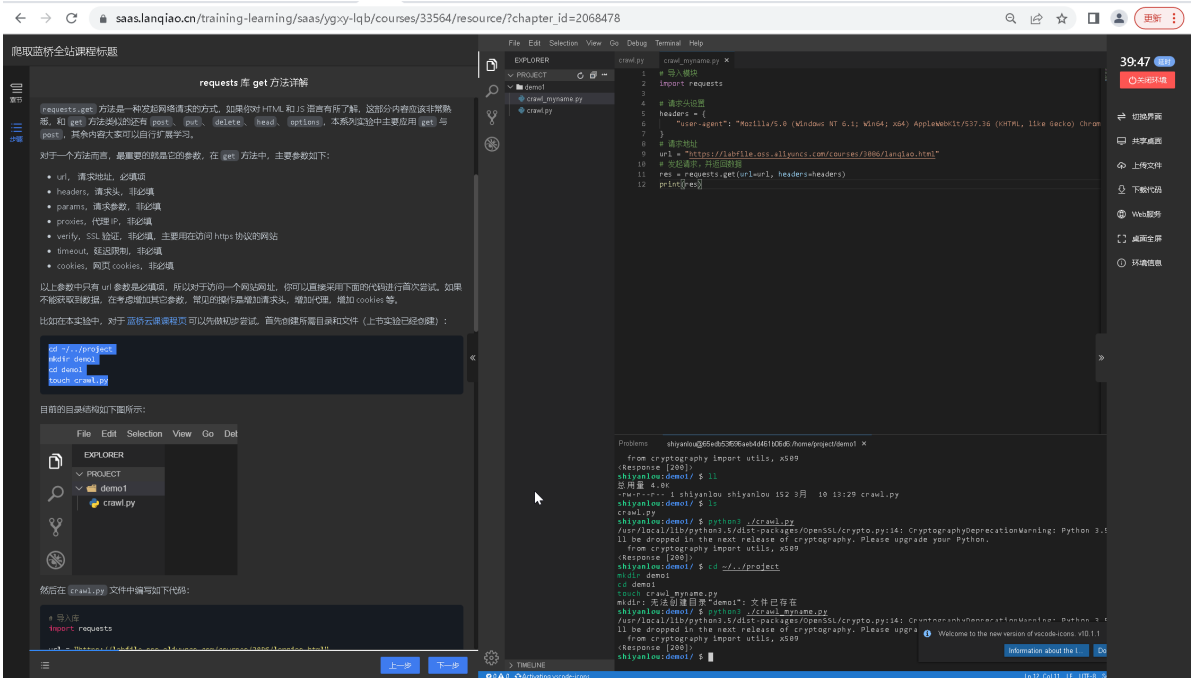
```
cd ~/../project
mkdir demo1
cd demo1
touch crawl_myname.py
```

复制下方源码到crawl_myname.py

```
# 导入模块
import requests

# 请求头设置
headers = {
    "user-agent": "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.125 Safari/537.36"
}
# 请求地址
url = "https://labfile.oss.aliyuncs.com/courses/3086/lanqiao.html"
# 发起请求，并返回数据
res = requests.get(url=url, headers=headers)
print(res)
```

```
python3 crawl_myname.py
```

截图参考：



正则表达式

re 模块与正则表达式

```
# 通过 search 匹配数据
result = re.search("<title>(.*)</title>", html)
print(result)
print(result.group(1))
```

上述代码的一些重要知识点如下：

- `re.search` 方法第一个参数也是正则表达式，第二个参数是待匹配的字符串，在本代码中是模拟的一段 HTML 代码。
- 正则表达式中出现了 `.`，`*`，`()`，注意都是英文输入法状态输入的，`.` 表示匹配任意字符，`*` 表示匹配前面的字符零次或者多次，括号的用途是为了后面的提取。
- 最后一行代码打出了网页的标题，使用的代码是 `result.group(1)` 表示获取正则表达式中第一个括号的内容，你可以尝试将数字换成 2，看一下报错内容。

通过命令行运行之后，效果如下图所示：

```
shiyanlou:demo1/ $ python3 crawl.py
<_sre.SRE_Match object; span=(137, 167), match='<title>实验楼Python爬虫实战课程</title>'>
实验楼Python爬虫实战课程
shiyanlou:demo1/ $
```

接下来我们在获取一下 `h2` 标签内的文字，代码如下：

```
# 只展示不同部分
result = re.search("<h2>(.*)</h2>", html)
print(result)
print(result.group(1))
```

我只将正则表达式部分的 title 切换成 h2 即可匹配成功。

下面我们在提高一下难度，通过 HTML 标签我们将 h2 的颜色设置为红色，代码修改如下（只展示差异部分）：

```
<h2 style="color:red;">梦想橡皮擦老师的课程</h2>
```

这时，我们使用刚才的正则表达式就无法获取到文字内容了，因为 `<h2>(.*)</h2>` 无法正确匹配到上述 HTML 代码，下面做出相应的调整，注意这里依旧采用 `.` 与 `*` 的配合即可。

```
# 注意在h2的后面存在一个空格，之后是.*表示匹配任意字符，在之后是一个>，用于和后面的内容做一下区分。
result = re.search("<h2 .*>(.*)</h2>", html)
print(result)
print(result.group(1))
```

上述代码大家需要加深理解，由于篇幅的关系，没有办法为大家详细的把正则表达式全部内容都展开，所以在有限的案例中一定要找到正则表达式与 re 模块配合使用的感觉。

re 模块解析实验楼课程页

re 模块解析实验楼课程页

通过 request 获取待解析网页

对于任何一个爬虫程序来说，第一步都是获取待解析的内容，本实验要获取的网页地址为[实验楼课程页](https://labfile.oss.aliyuncs.com/courses/3086/lanqiao.html)。

我们在前文中已经把要使用的技术做好相应的铺垫，使用如下代码获取即可。

```
import requests

headers = {
    "user-agent": "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.125 Safari/537.36"
}
url = "https://labfile.oss.aliyuncs.com/courses/3086/lanqiao.html"
res = requests.get(url=url, headers=headers)
res.encoding = 'utf-8'
print(res.text)
```

代码说明与结果展示：

- 注意最后一行代码打印的内容为 `res.text` 它表示打印网页返回的文档信息。

代码运行之后，你会看到如下**一堆**内容的输出，不用担心，稍后就会解析清楚。

这里打印出的内容，就是访问网页返回的网页源码，而通过 re 模块配合正则表达，就可以实现对它的解析，例如，把网页的标题匹配出来。

```
# 导入模块
import requests
import re

headers = {
    "user-agent": "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.125 Safari/537.36"
}
url = "https://labfile.oss.aliyuncs.com/courses/3086/lanqiao.html"
# 发送数据请求
res = requests.get(url=url, headers=headers)
res.encoding = 'utf-8'
html_data = res.text
# 通过正则匹配数据
```

```
result = re.search("<title>(.*)</title>", html_data)
print(result.group(1))
```

代码运行结果：

```
shiyanlou:demo1/ $ python3 crawl.py
精选项目课程_IT热门课程_蓝桥课程 - 蓝桥
shiyanlou:demo1/ $
```

这里就是上文的一个整体结合了，它的逻辑如下：

1. 通过 requests 库获取网页源码。
2. 通过 re 模块使用正则表达式解析网页。
3. 输出我们得到的数据。

解析所有的课程标题

网页标题已经解析出来，继续增加难度，将网页中所有的课程标题解析出来，这里需要在学习 re 模块下的一个方法。

`re.findall` 该方法看到名字就能猜出它的含义，匹配所有的数据。它的标准格式如下：

```
a = re.findall("匹配规则", "待匹配字符串")
```

方法先放在这个位置等待使用，下面你要用人眼看的方式去找到待匹配的数据所在位置。你可以使用谷歌或者火狐浏览器的开发者工具进行分析，如下图所示：

也可以在浏览器右键查看源码，通过搜索的方式找到数据所在区域，如下图所示：

不管通过何种方式，都要了解到数据所在区域的 HTML 代码格式，只有看到，才能爬取到。（注意这句话的潜在价值，爬虫的真谛。）

接下来就可以进行实际的编码了。

HTML 代码结构：

```
<h6 title="新手入门指南之玩转实验楼" class="course-name" data-v-585c8a00>
  ..........
</h6>
```

正则表达式（可以进行比对着些，也可以通过搜索引擎搜索【在线正则表达式匹配】使用工具完成）。

```
<h6 title="(.*)" class="course-name"
```

Python 爬虫代码如下：

```
import requests
import re

headers = {
    "user-agent": "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.125 Safari/537.36"
}
url = "https://labfile.oss.aliyuncs.com/courses/3086/lanqiao.html"
res = requests.get(url=url, headers=headers)
res.encoding = 'utf-8'
html_data = res.text
results = re.findall('<h6 title="(.*)" class="course-name"', html_data)
print(results)
```

代码说明：

- findall 方法部分，注意匹配规则中如果出现了双引号，为了方式代码报错，需要在最外层使用单引号进行区分。

运行结果：

```
shiyanlou:demo1/ $ python3 crawl.py
['新手入门指南之玩转实验楼', 'Python 新手入门课', 'Linux 基础入门', 'WebIDE 环境使用指南', 'C++ 实现 STL 标准库和算法', '
门', 'C++ 从零实现 NoSQL 数据库', '知识图谱构建射雕三部曲人物关系', 'Ansible 基础入门', 'JavaScript 基础入门', 'Python 异
'PyTorch 入门与实战', 'TensorFlow 2 深度学习入门与实践', 'Spring Cloud 与 Docker 实战', '自然语言处理基础入门', 'C++ 实现
'Flask 实现一个问答社区']
shiyanlou:demo1/ $
```

代码运行完毕，成功得到本页所有课程标题。

课堂互动2

练习1：参考上述源码，将下面的源码存在crawl_myname.py，然后分别在蓝桥云课和pycharm上执行，将源码和运行结果截图到随堂练习中。

```
cd ~/../project
mkdir demo1
cd demo1
touch crawl_re_myname.py
```

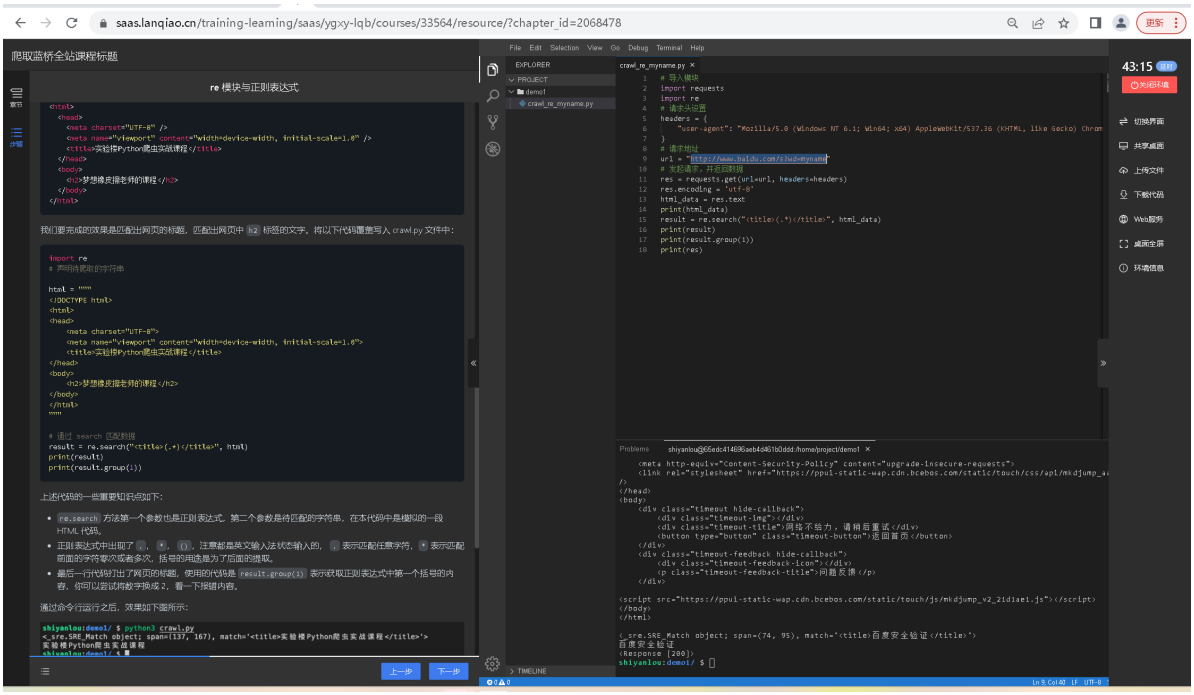
复制下方源码到crawl_re_myname.py

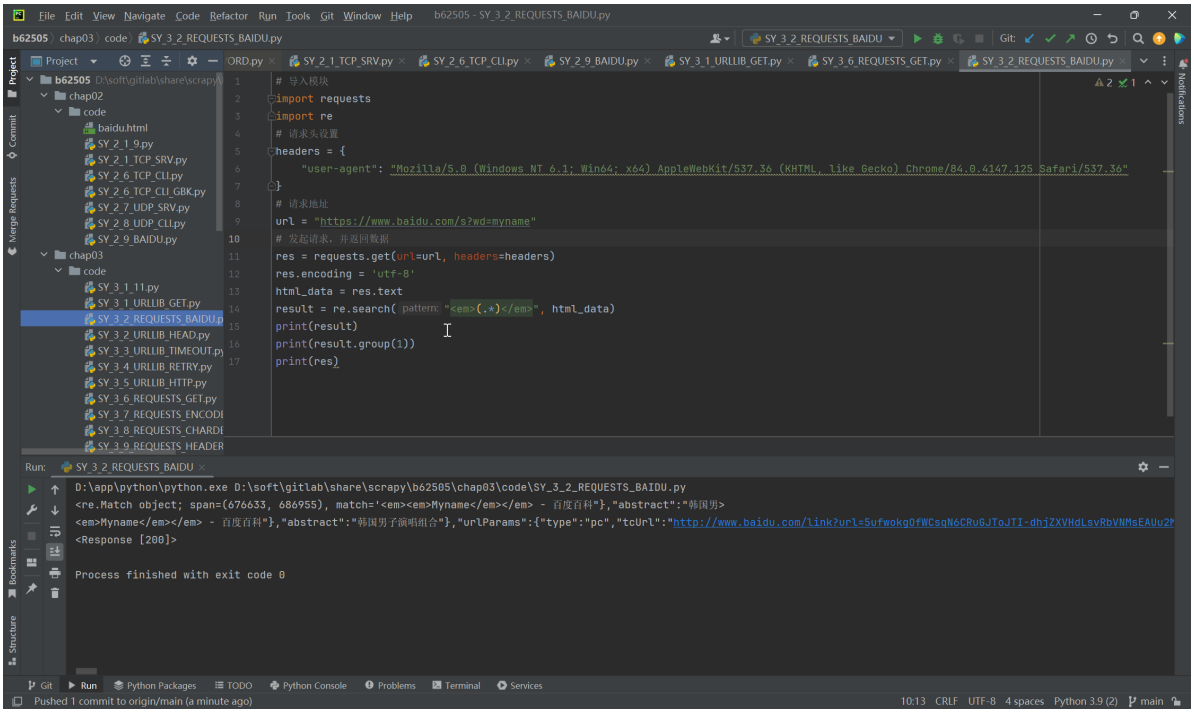
```
# 导入模块
import requests
import re
# 请求头设置
headers = {
```

```
6  "user-agent": "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36
   (KHTML, like Gecko) Chrome/84.0.4147.125 Safari/537.36"
7  }
8  # 请求地址
9  url = "http://www.baidu.com/s?wd=李某某"
10 # 发起请求，并返回数据
11 res = requests.get(url=url, headers=headers)
12 res.encoding = 'utf-8'
13 html_data = res.text
14 print(html_data)
15 result = re.search("<em>(.*)</em>", html_data)
16 print(result)
17 print(result.group(1))
18 print(res)
```

```
1  python3 crawl_re_myname.py
```

截图参考：





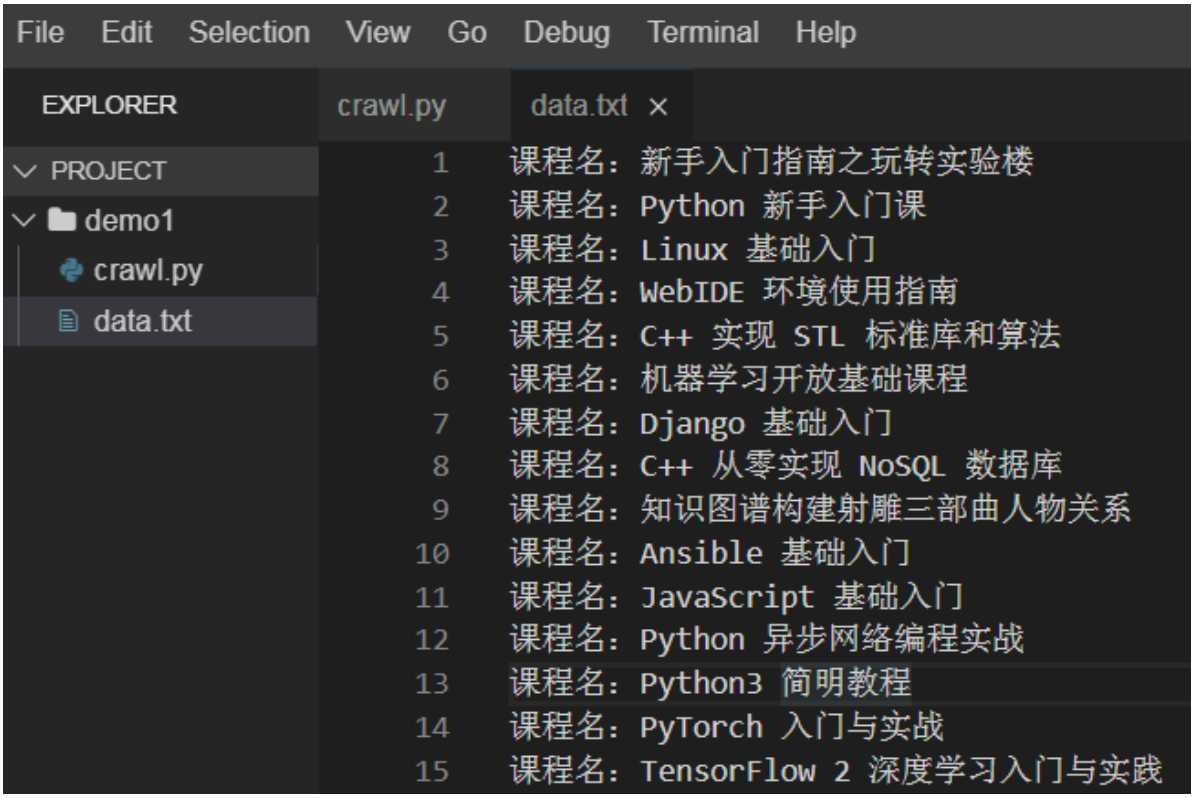
爬取数据存储

格式化后数据存储

下面就是本实验的最后一个步骤了，将爬取到的数据存储到文件中，在本实验中你可以将获取到的数据进行一些简单的格式化操作，再存入文件中，具体代码如下：

```
import requests
import re

headers = {
    "user-agent": "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.125 Safari/537.36"
}
url = "https://labfile.oss.aliyuncs.com/courses/3086/lanqiao.html"
res = requests.get(url=url, headers=headers)
res.encoding = 'utf-8'
html_data = res.text
# 匹配数据，获取所有的课程标题
results = re.findall('<h6 title="(.*)" class="course-name"', html_data)

for item in results:
    # 格式化字符串
    new_str = "课程名：{item}\n".format(item=item)
    # 将格式化之后的数据写入文件
    with open("./data.txt", "a", encoding="utf-8") as f:
        f.write(new_str)
```

代码运行完毕，会在当前文件目录生成一个 `data.txt` 的文件，文件内容如图：



上述代码中，with 上一实验已经介绍过，是 Python 中一种语法结构，with 后面的表达式 `open('./data.txt')` 返回是一个 file 类型的变量，后面用 as 可以理解成给起了一个别名叫做 `f`，其它名称都可以。在 with 语句块中就可以使用这个变量操作文件，执行 with 这个结构之后， `f` 会自动关闭。

with 代码如果使用一般写法，类似下述代码：

```
file = open("data.txt")
try:
    data = file.write("abc")
finally:
    file.close()
```

示例中出现代码段 `open('./data.txt','a',encoding="utf-8")`，里面的各参数含义如下：

- `'./data.txt'` 存储的文件名，其中 `./` 表示当前目录。
- `a` 打开文件的模式：只读 `r`，写入 `w`，追加 `a` 等，具体可以参考：学习地址。
- `encoding="utf-8"` 文件编码。

课堂互动3

练习1：参考上述源码，将下面的源码存在crawl_save_myname.py，然后分别在蓝桥云课和pycharm上执行，将源码和运行结果截图到随堂练习中。

```
cd ~/../project
mkdir demo1
cd demo1
touch crawl_save_myname.py
```

复制下方源码到crawl_save_myname.py

```
import requests
```

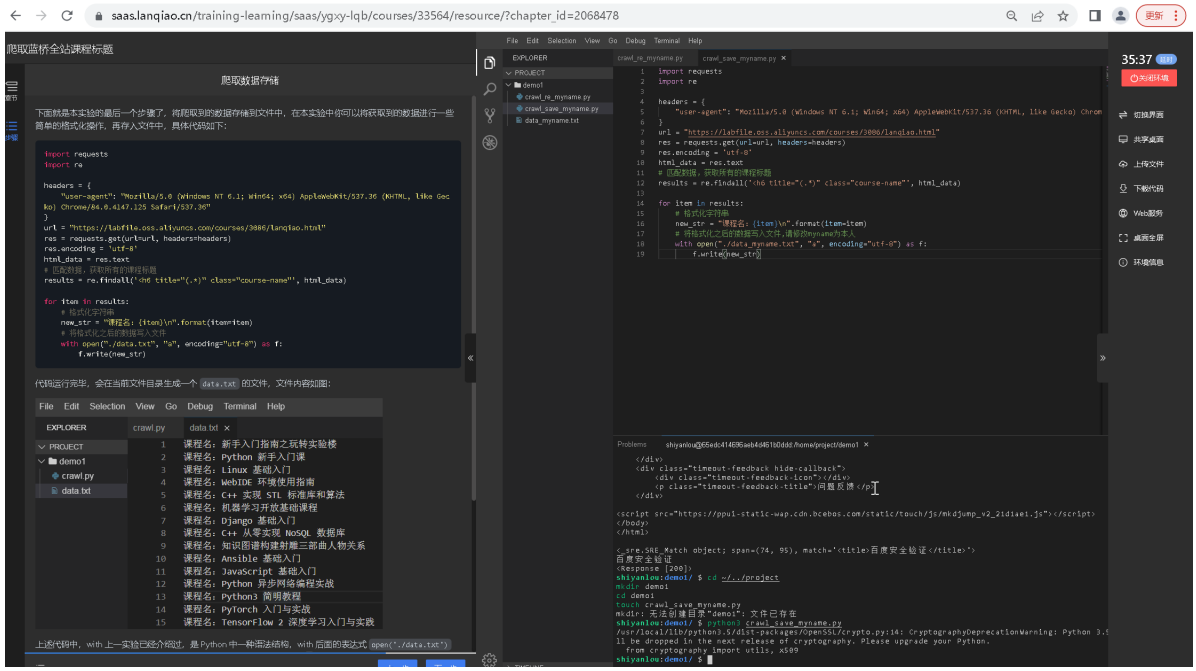
```
import re

headers = {
    "user-agent": "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.125 Safari/537.36"
}
url = "https://labfile.oss.aliyuncs.com/courses/3086/lanqiao.html"
res = requests.get(url=url, headers=headers)
res.encoding = 'utf-8'
html_data = res.text
# 匹配数据，获取所有的课程标题
results = re.findall('<h6 title="(.*)" class="course-name"', html_data)

with open("./data_myname.txt", "w", encoding="utf-8") as f:
    f.write("")

for item in results:
    # 格式化字符串
    new_str = "课程名：{item}\n".format(item=item)
    # 将格式化之后的数据写入文件
    with open("./data_myname.txt", "a", encoding="utf-8") as f:
        f.write(new_str)

```

源码链接：http://home.hddly.cn:8093/biglab-share/scrapy/-/blob/main/b00113scrapy/ch03_re/crawl_save_limm.py?ref_type=heads

运行

```
python3 crawl_save_myname.py
```

运行截图参考：



实验总结

本实验主要实现一个爬虫程序的完整闭环：获取数据，解析数据，存储数据。任何爬虫都可以按照这三个步骤进行操作。对于每个步骤的不同实现，是后续实验中我们要学习的重点内容。

本实验中未展开部分，大家可以在课下检索学习资料进行相应的补全，爬虫开发者需要的技能点比较丰富，这也是爬虫程序编写的魅力所在

源码下载与上传

视频学习