【Python】No.8网络爬虫基础-爬取课表为例

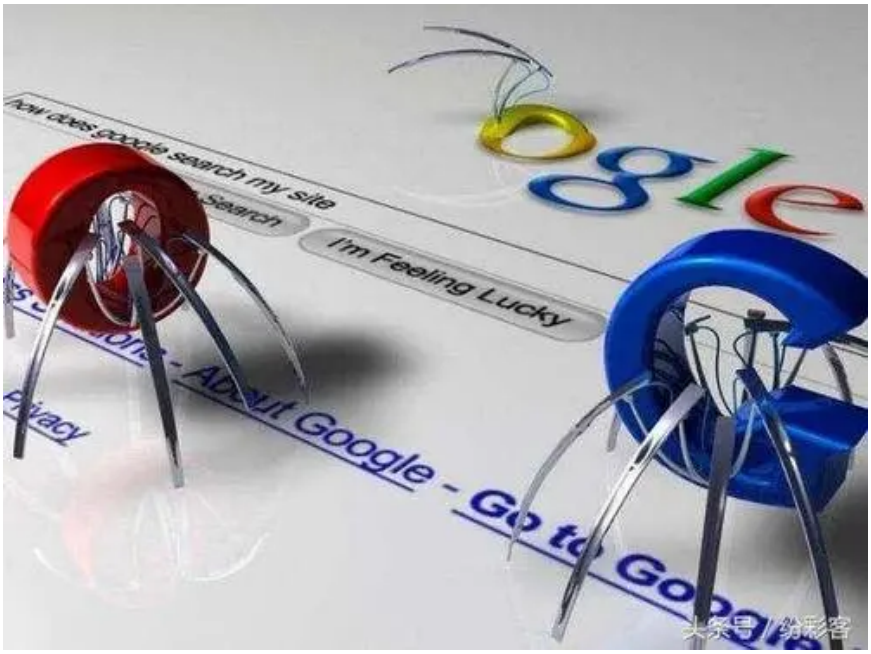
NAU Analysts 2020-05-01 23:29

以下文章来源于Financial Workshop ，作者潘徐智

Financial Workshop

Python 爬虫简介

现如今是个数据为王的时代，互联网存储着大量的信息。作为没有掌握一定的爬虫技术的网民，我们可能都是通过使用浏览器来访问互联网上的内容。但是如果我们想要批量得到一些散布在互联网各个角落的某方面消息，人为的去获取这些信息可能非常麻烦，费事费力，所以本次编者想和大家分享一个系列：如何运用python去爬取一些数据，并将其转换成价值数据，进行线性回顾分析，具有一定的经济学意义。



网络爬虫就是一种按照一定的规则，自动地抓取万维网信息的程序或者脚本。
我们通过python程序，模拟浏览器向服务器发送请求、获取信息、分析信息并存储我们所需要的价值数据。

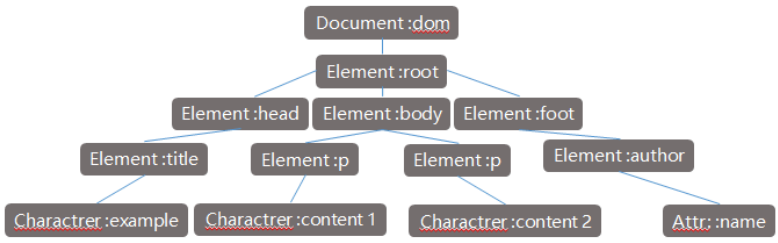
以百度新闻为例，发布者每天发布的新闻内容不同但是排版格式相同，为了方便每天重复排版的工作，发布者会用一些手段给这些内容进行自动的“化妆”，爬虫就是顺着他们化妆的方法来爬取到我们需要的内容。

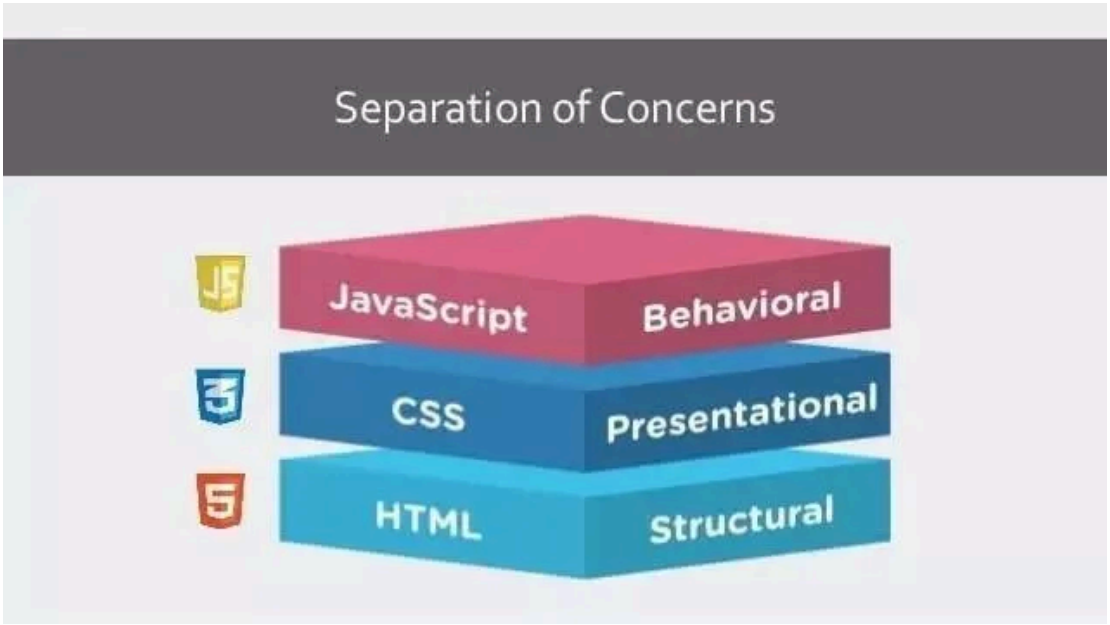
我们爬取的对象就是HTML超文本标记语言，它是最基础的网页语言，而不是编程语言。

百度新闻模板化的新闻格式

网页结构

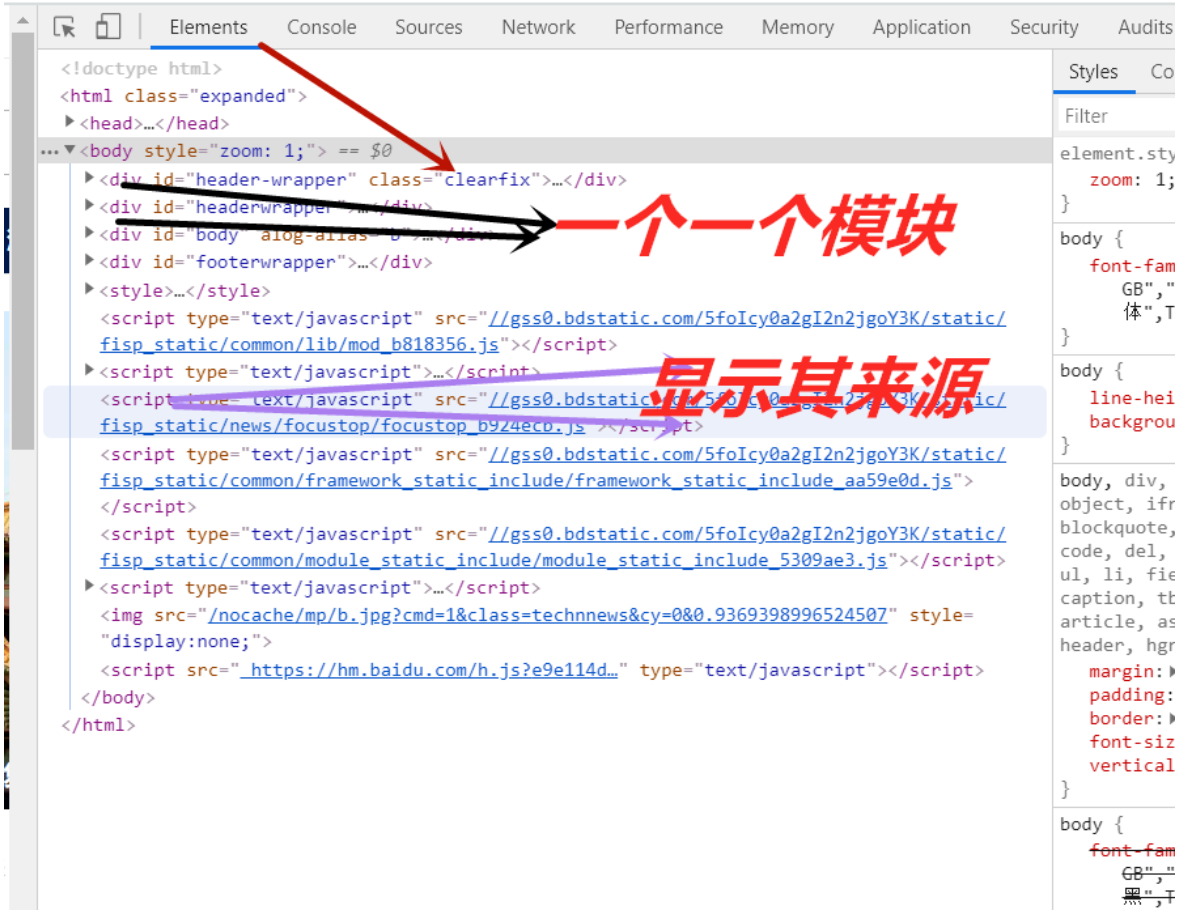
```
<?xml version="1.0"?>
  <root>
    <head>
      <title>example</title>
    </head>
    <body>
      <p>content 1</p>
      <p>content 2</p>
    </body>
    <foot>
      <author name="lisi"/>
    </foot>
  </root>
```





网页结构也是一种很有层次的结构，一种树形结构。访问网页远非我们输入地址后看到网页那么简单。需要我们在浏览器按下“F12”，或者右键网页，选择”检查“，就可以看到网页背后的代码。

以谷歌的Chrome浏览器为例，在百度新闻的网站，按下”F12“，会出现一个浏览器的检查窗口。默认的Elements窗口为当前界面的HTML代码。



网页就是按照这种源码进行排版，点击上面的代码，就可以看见这些代码所对应的内容在何处，它们以一个一个div下面又会嵌套许多小的list来存储内容。

HTML的结构

```
<!DOCTYPE HTML>
<!--用来指定当前页面所遵循的html的版本-->
<HTML>
<meta http-equiv = "Content-type" content="text/html;charset=gbk"/>
   <!--其中charset规定浏览器用什么编码解析当前页面-->
   <title>指定网页的标题</title>
   <HEAD>头部用来存放html页面的基本属性信息，优先被加载</HEAD>
   <BODY>体部分用来存放页面数据，是可见的页面内容</BODY>
</HTML>
```

网页的修饰方式与选择器 -CSS与选择器

```
<p>p标签
   <div class="h1" id="s1"> 新年好</div>
</p>
```

标签名选择器	标签名{}	div{color:red;}
类选择器	.类名{}	.h1{color:red;}
id选择器	#id值	#s1{color:red;}
扩展选择器之：后代选择器		p div{color:red;}

处理都是为了对其网页进行化妆，比如说第一行标签名的选择，red表明颜色主要用红色，div{}是其命名格式。不同的选择器应用不一样，有自己的优缺点。这里就不一一说明，可以参考如下网站。

菜鸟教程：https://www.runoob.com/css/css-id-class.html

这些选择器往往不断发展，就有许多拓展选择器。

拓展选择器

(1)后代选择器
选择父元素中的后代元素
父元素选择器，后代元素选择器
p{color:#00ff00}
p b{color:#ff0000;}
p标签刘德华段落样式

(2)子元素选择器
选择父元素中的子元素
父元素选择器>子元素选择器{}
h1>strong{color:red;}

(3)分组选择器
将多个选择器进行同一个css模式的操作
选择器1，选择器2,...{}
p,div{color:#ffffff}

(4)相邻兄弟选择器
选择器选择到相邻的兄弟元素，同级
选择器+兄弟名{}，只能通过大哥找小弟
div1+div2{margin-top:50px}

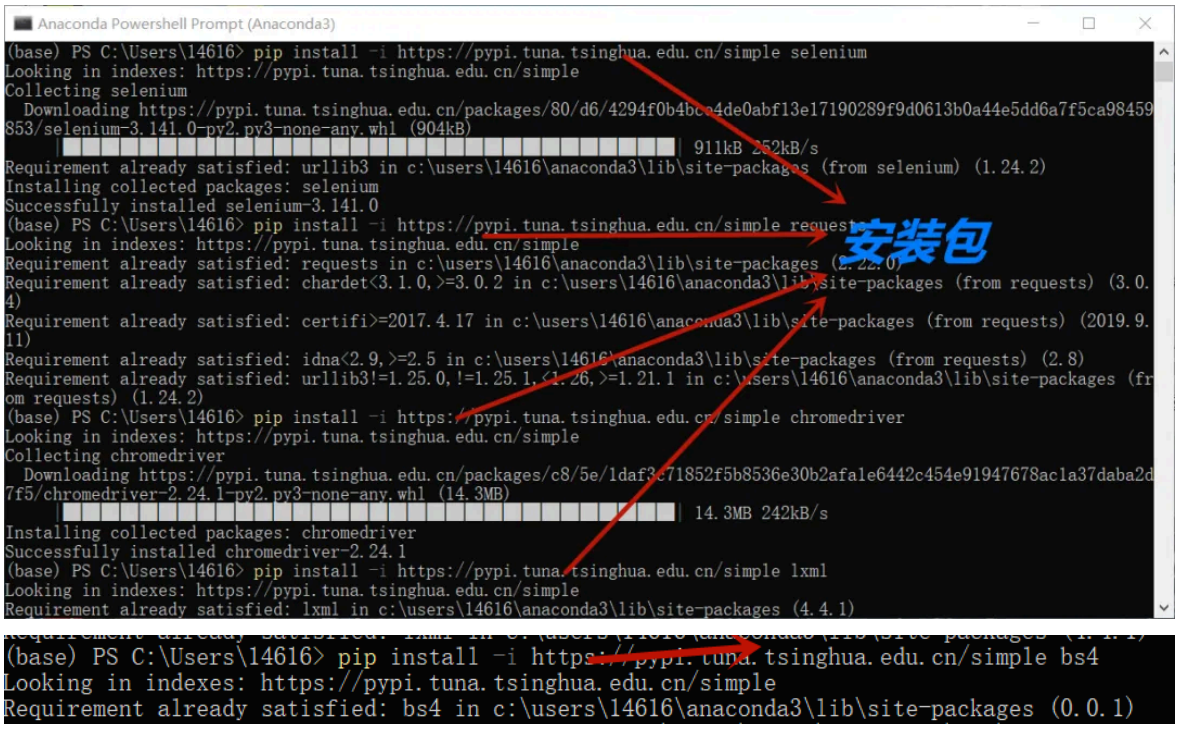
(5)属性选择器
选择具有指定属性，或指定属性的值等于指定值的选择器
选择器[属性名]{}
选择器[属性名='属性值']{}

如果希望把包含属性(title)的所有元素变红，可以写作
*[title]{color:red;}
与上面类似，可以只对有href属性的锚(a元素)应用样式
a[href]{color:red;}
为了将同时有href和title属性的html超链接文本设置为红色，可以这样写：
a[href][title]{color:red}
假设只希望选择moons属性值为1那些planet元素:
planet[moons="1"]{color:red;}

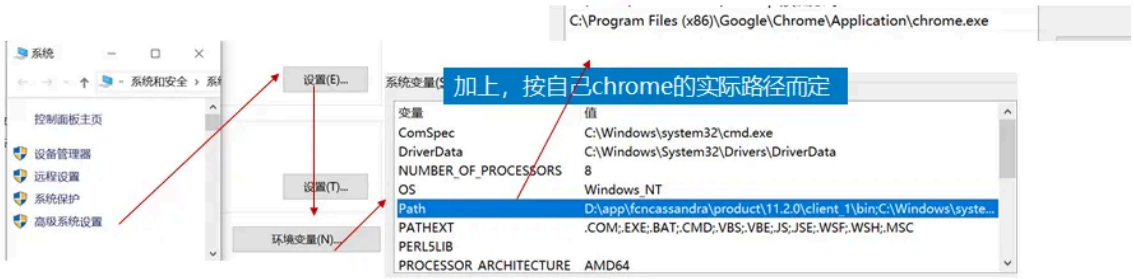
(6)伪元素选择器
其实就在html中预先定义好的一些选择器，称为伪元素，是因为css的术语
:link未点击的状态
:visted被点击过的装填
:hover鼠标移动到元素之上但是仍然未点击的状态
:active被鼠标点击着的状态
a:visited{ color:red; }
结合用:
.td1:hover{backgroud-color:$ff0000;}

通过这些选择器可以爬取到我们所能看到的内容，没有显示的内容我们可能无法看到，对我们的以后的分析打下基础。当我们爬虫学到高阶的内容，我们大家的抢课可以进行自动化操作。但是经常的ip地址不断访问，就会被管理员发现，就会禁止你这个ip地址，但是我们也可以应用ip代理，去爬取消息。总之就是这些网页与爬取者不断"斗争"（京东在这方面的限制没有淘宝rigid，我们可以在京东上做一些试验）。

Python爬虫所需要安装的包



对于网络自动化的爬虫，让机器自己模拟去爬虫，需要下载Chrome，并利用Chrome的工具ChromeDriver。



下面为大家分享一下如何下载ChromeDriver并加入环境变量中。

下载网址：http://npm.taobao.org/mirrors/chromedriver/

首先打开该网址，一定选择与自己实际Chrome浏览器版本一样的ChromeDriver下载，先可以通过如下步骤查看自己的Chrome版本。

然后，下载完之后，对压缩文件进行解压，得到chromedriver.exe，然后打开此电脑中的高级设置，继续点开环境变量，复制chromedriver.exe在电脑中的地址，将其加入到环境变量中，确定即可。

提示：要清楚chromedriver.exe所在位置，后期爬虫需要运用到该地址。

库的基础介绍：

- requests:用于向服务器发送请求并获取数据
- bs4：用于分析html数据
- pandas：用于分析数据

除此之外，我们平常爬虫可能还用到的库

- squlite:轻量级数据库
- re：用于进行正则表达式匹配

感兴趣的同学还可以继续去学习它们。

爬虫引入

导入必要的库

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.wait import WebDriverWait
from bs4 import BeautifulSoup   #分析html数据格式
import requests             #向服务器发送请求并获取数据
import pandas as pd       #便于时候的数据分析
```

requests的使用

```
url="https://www.baidu.com/" #给定网址
response=requests.get(url) #使用请求并获取数据
```

上述代码我们是以程序的方式访问了www.baidu.com百度网页，其中"requests.get（网页地址)"就是以get的方式去访问网页。我们已经使用"response=requests.get（url)"将获得的信息传入到response中。我们输出response，得到的不是网站的代码，而是响应码。

```
'''
响应状态码
200：成功
301：跳转
403：无权限访问
404：文件不存在
502：服务器错误
'''
response
```

```
<Response [200]>
```

响应码状态表示我们之前requests的结果。常见的有200，代表成功；403，无权限访问；404文件不存在；502，服务器访问错误。相信大家之前访问某些国外网址的时候可能经常遇到这些错误。

想要看到我们之前使用的"requests.get(url)"得到的网页内容，我们需要先进行"response.encoding='utf-8'",该步骤是将得到的网页内容进行utf-8编码，否则我们看不到网页中的中文。

```
response.encoding='utf-8'
response.text
```

```
'<!DOCTYPE html>\r\n<!--STATUS OK--><html> <head><meta http-equiv=content-type content=text/html;charset=utf-8><meta http-equiv=X-UA-Compatible content=IE=Edge><meta content=always name=referrer><link rel=stylesheet type=text/css href=https://ss1.bdstatic.com/5eN1bjq8AAUYm2zgoY3K/r/www/cache/bdorz/baidu.min.css><title>百度一下，你就知道</title></head> <body link=#0000cc> <div id=wrapper> <div id=head> <div class=head_wrapper> <div class=s_form> <div class=s_form_wrapper> <div id=lg> <img hidefocus=true src=//www.baidu.com/img/bd_logo1.png width=270 height=129> </div> <form id=form name=f action=//www.baidu.com/s class=fm> <input type=hidden name=bdorz_come value=1> <input type=hidden name=ie value=utf-8> <input type=hidden name=f value=8> <input type=hidden name=rsv_bp value=1> <input type=hidden name=rsv_idx value=1> <input type=hidden name=tn value=baidu><span class="bg s_ipt_wr"><input id=kw name=wd class=s_ipt value maxlength=255 autocomplete=off autofocus=autofocus></span><span class="bg s_btn_wr"><input type=submit id=su value=百度一下 class="bg s_btn" autofocus></span> </form> </div> </div> <div id=u1> <a href=http://news.baidu.com name=tj_trnews class=mnav>新闻</a> <a href=https://www.hao123.com name=tj_trhao123 class=mnav>hao123</a> <a href=http://map.baidu.com name=tj_trmap class=mnav>地图</a> <a href=http://v.baidu.com name=tj_trvideo class=mnav>视频</a> <a href=http://tieba.baidu.com name=tj_trtieba class=mnav>贴吧</a> <noscript> <a href=http://www.baidu.com/bdorz/login.gif?login&amp;tpl=mn&amp;u=http%3A%2F%2Fwww.baidu.com%2f%3fbdorz_come%3d1 name=tj_login class=lb>登录</a> </noscript> <script>document.write(\'<a href="http://www.baidu.com/bdorz/login.gif?login&tpl=mn&u=\'+ encodeURIComponent(window.location.href+ (window.location.search === "" ? "?" : "&")+ "bdorz_come=1")+ \'" name="tj_login" class="lb">登录</a>\');\r\n                </script> <a href=//www.baidu.com/more/ name=tj_briicon class=bri style="display: block;">更多产品</a> </div> </div> </div> <div id=ftCon> <div id=ftConw> <p id=lh> <a href=http://home.baidu.com>关于百度</a> <a href=http://ir.baidu.com>About Baidu</a> </p> <p id=cp>&copy;2017 Baidu <a href=http://www.baidu.com/duty/>使用百度前必读</a>  <a href=http://jianyi.baidu.com/ class=cp-feedback>意见反馈</a> 京ICP证030173号  <img src=//www.baidu.com/img/gs.gif> </p> </div> </div> </div> </body> </html>\r\n'
```

这样操作之后我们就可以看到网页的代码了。

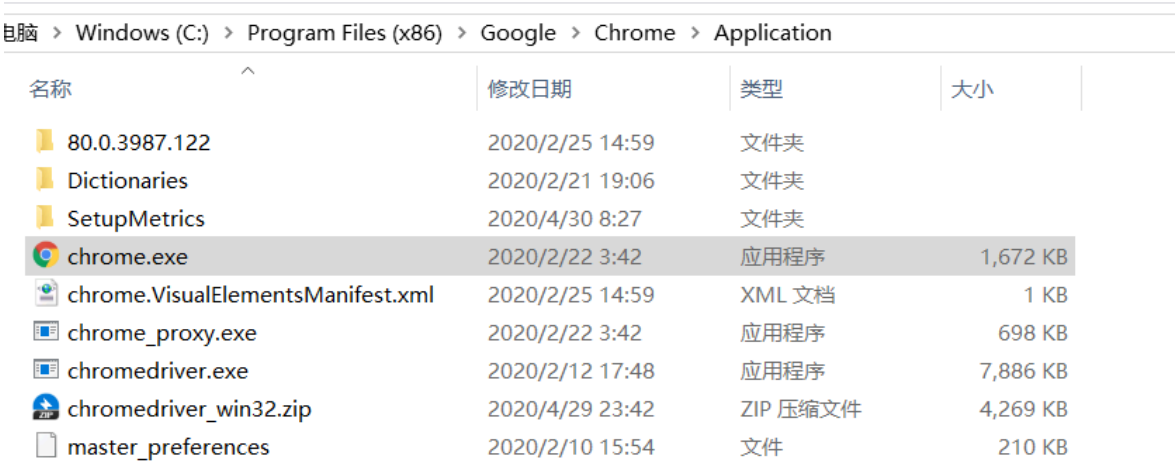
相信大家对爬虫有了简单的了解，下面我们结合实例和大家操作一遍，我们就以大家的问题入手-南审教务在线的课表爬取。

爬虫教务实战

访问网站

```
1 browser = webdriver.Chrome(r'C:\Program Files (x86)\Google\Chrome\Application\chromedriver') #声明浏览器对象
2 browser.get('http://jwc.nau.edu.cn/coursearrangeInfosearch.aspx')
```

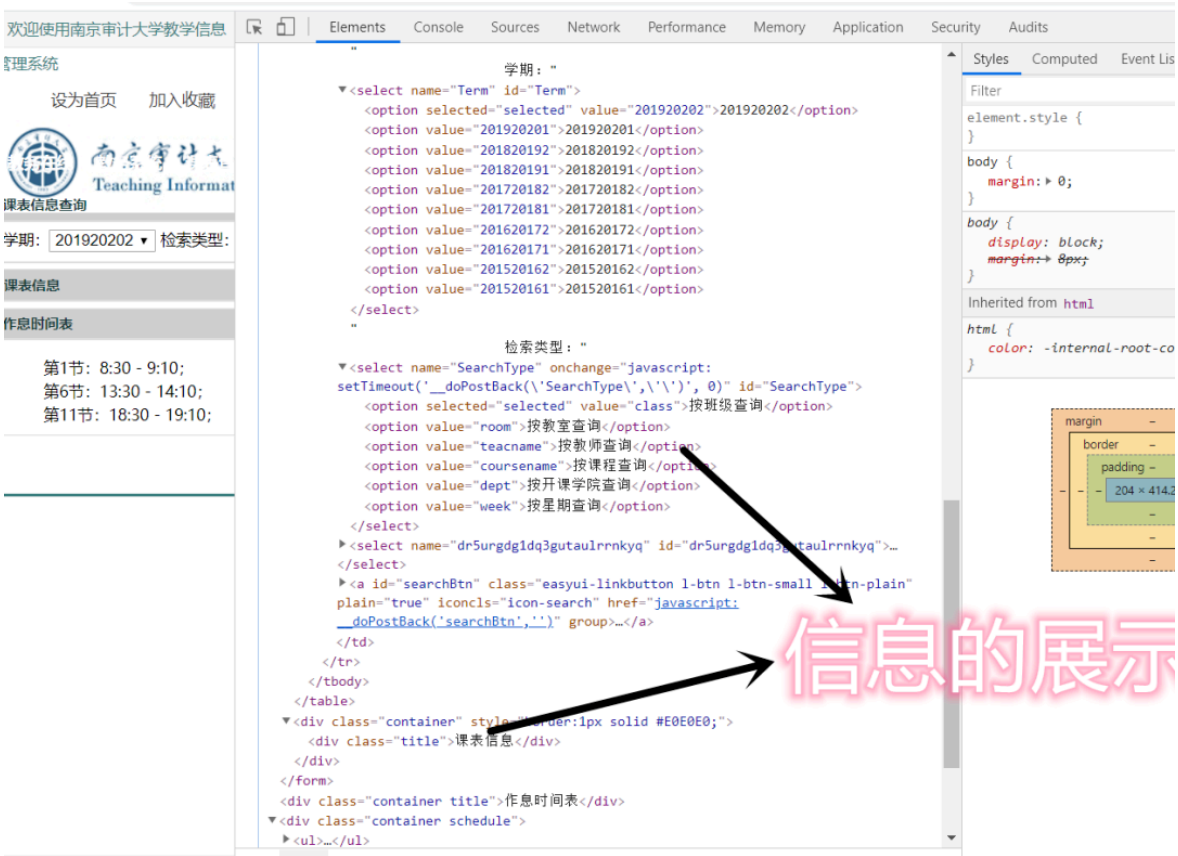
首先我们找到我们前期下载的chrome driver的文件夹所在位置（一定要将其解压，将解压中的exe文件放在chrome文件下，小编一开始就因为这个忽略而报错，希望大家不要再踩这个坑了），copy一下所在位置，显示如下：



大家运行一下这个代码就可以自动进入教务在线的课表，需要注意的是这个符号"\"需要在前面加入一个r。

获取班级所有的内容

小编在此先进行简单的扩展一下，这个有关于我们如何获取这样的课表信息。我们选择不同的班级，就相当于不同的值，然后从前端传输到后端，运用if语句，反馈给我们这样的信息。所以说我们想获取这样的值，要先获得这些班级的列表，浏览器才能获得这些班级课表的内容。其实这些班级的信息已经在前端显示，我们只要按下拉框，就可以找到所有班级的信息，虽然我们肉眼可能无法看见，其实它的本质已经存储在这个界面了。我们可以按下F12查看源代码来验证一下。



我们可以发现这些信息存放在body下，选择select name就可以找到这些信息（这些内容已经加载到前端了）。

所以我们首先要定义一个select name，因为select name后面的值是随机的，需要人工导入，比较麻烦，但是也有一些其它选择器可以自动处理，这里小编就分享一个爬虫比较进阶的内容。我们需要选择上文中我们的其它选择器中的一个叫兄弟选择器的方法，去准确地定位我们所需要爬取的内容。具体用法如下：

复制一下这个id到'#Term+select=select'中。运行该代码，可以得到信息，这个代码是如何具体运行呢？我们发现网页代码的临近id=Term，这样我们就可以通过连续的两次select找到我们所需要的信息（由于小编发现学校这个教务系统有发爬虫措施，有些元素动态变化，但是兄弟选择器定位他的标签，爬虫技术很好的人，只要能正常打开网页，就能爬取到所想看到的东西）。

但是这个展示和我们的网页展示不一样，这就是我们前面所讲的html的结构。这个内容不能直接用，所以我们要将其转换成列表的形式，这就运用我们前面所学的python容器list列表，这个列表我们首先要进行分割，按照\n--表示换行符，运用要前面同学所分享的split。

```
1  class_name = [i[1:] for i in ls.text.split('\n')]

1  len(class_name)
```

浏览器自动选取并点击

下面我们就要运用浏览器自动地点"查询"这个键，但是这个键在哪里？我们就又要按下F12找到span，它是一个事件，触发浏览器运用的一个发动器，我们要找到这个出发span的代码，我们可以发现href=javascript，（javascript是一个前端语言，它可以触发运行，我们大概率就可以将查询定位到这个标签。

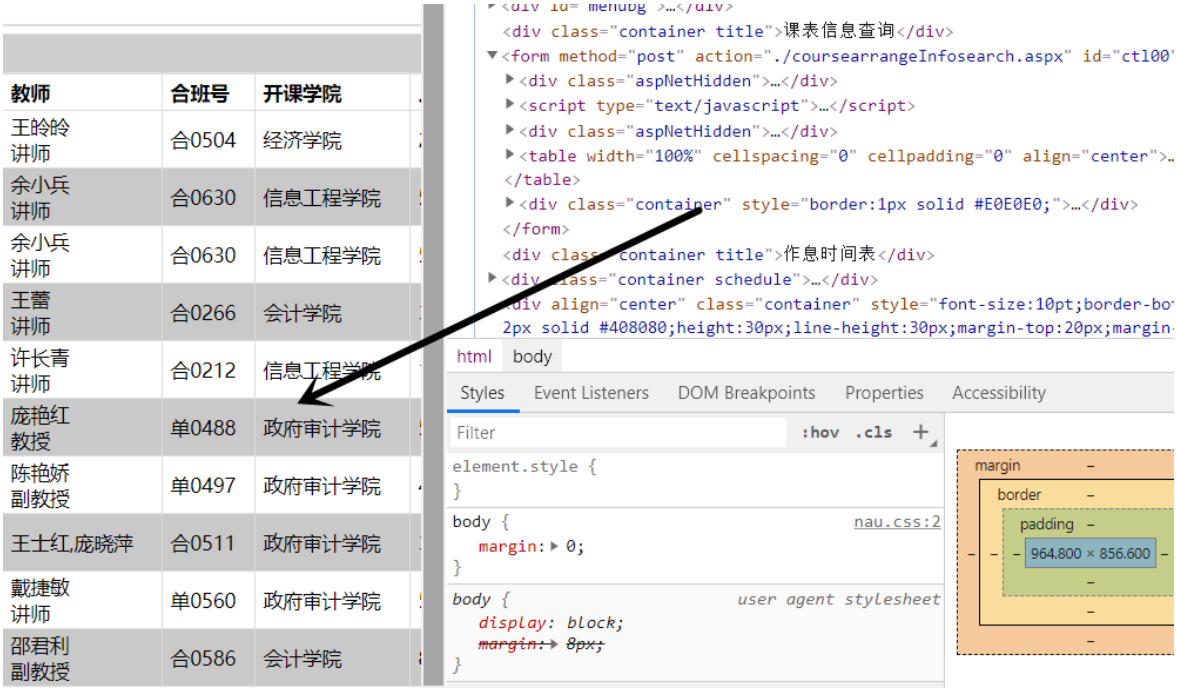
我们发下这个span也有一个id"searchBth"，将其复制。

```
ls.send_keys(i)   #定义关键词（比如说定义我们CFA专业
btn = browser.find_element_by_id('searchBtn')  #将id放入该函数中
btn.send_keys(Keys.ENTER)    #表示浏览器的正常操作"enter"
```

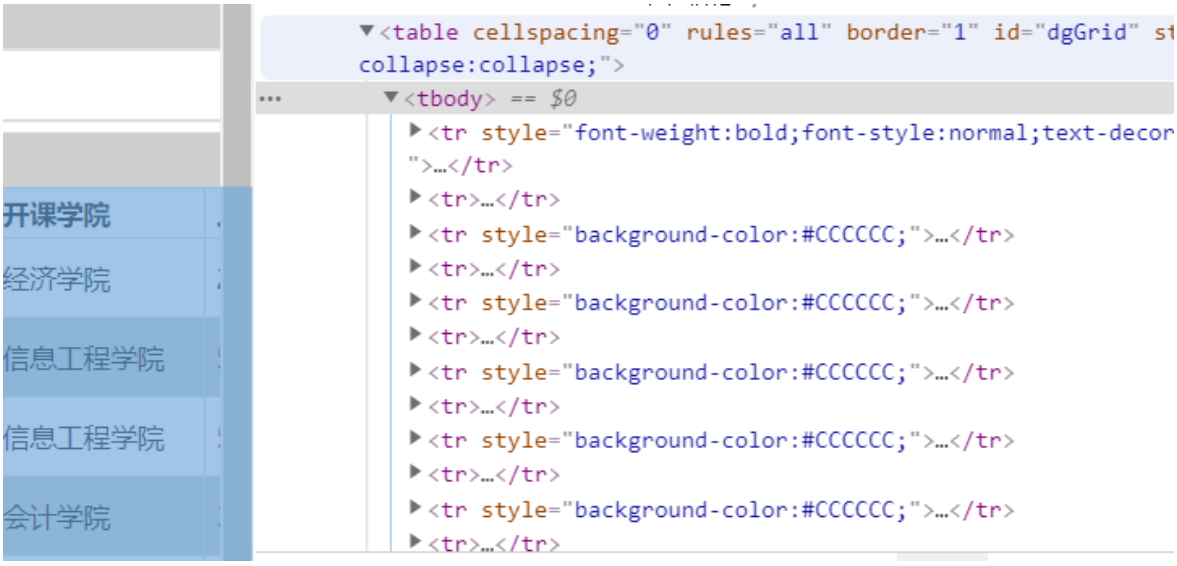
但是运行的发现有错误，keys没有定义，所以我们运用我们前面的包：from selenium.webdriver.common.keys import Keys，这样操作一下，我们就可以得到我们所

需要的班级内容。下面我们就要将这些内容收集起来，这时候就要我们对列表的结构有一定的认识，梳理一下。

获取指定班级的课表



我们继续F12找到课表内容所在的位置，发现有一个container标签，然后然后再往下找到一个table标签，打开有一个tbody就可以找到我们所需要的内容，tbody下面又有一个tr，点击可以发现，它是一行一行的，上面的background-color就是给其上色（前面的css文件谈过类似内容），下一层结构还有一个td，点击发现，它才是我们真正多需要获得的内容。



每一个tr里面的排版是一样的，一个是班级，然后是课程，学分，上课地址。这就需要我们自己定义顺序，具体操作如下：

```
soup = BeautifulSoup(browser.page_source, "html.parser")
```

我们使用soup将html与其剥离开来（前文已经引用了from bs4 import BeautifulSoup 作为解析文件）接着是由soup.select选择我们所需要的内容：

elements = soup.select('.container tr')[1:-1]

定义获取网页内容的函数

```python
def getContents(elements):
    page = {}   #定义了page字典
    ind = []    #定义了一些空列表，这些空列表就是网页的内容
    cla = []
    wee = []
    nam = []
    sco = []
    roo = []
    tea = []
    mer = []
    dep = []
    cou = []
    for i in range(len(elements)):    #运用了一个for循环，这个lens就是传入soup.container的内容
        ind.append(elements[i].select('td')[0].text) #利用append将其所有的信息追加到列表中去
        cla.append(elements[i].select('td')[1].text)
        wee.append(elements[i].select('td')[2].text)
        nam.append(elements[i].select('td')[3].text)
        sco.append(elements[i].select('td')[4].text)
        roo.append(elements[i].select('td')[5].text)
        tea.append(elements[i].select('td')[6].text)
        #ran.append(elements[i].select('td')[7].text)
        mer.append(elements[i].select('td')[7].text)
        dep.append(elements[i].select('td')[8].text)
        cou.append(elements[i].select('td')[9].text)
    page['序号'] = ind #规整地存到字典中去
    page['班级'] = cla
    page['课程安排'] = wee
    page['课程名'] = nam
    page['学分'] = sco
    page['教室'] = roo
    page['老师'] = tea
    page['合班'] = mer
    page['学院'] = dep
    page['学生人数'] = cou
    return pd.DataFrame(page) #返回到dataframe中
```

严正申明:

爬虫本身会对服务器产生很大的压力，大量的数据爬取导致服务器崩溃导致，导致一共网站的公司的利益受损，但是少量数据不影响服务器正常运行的情况下。所以在此为了演示的需求，我们仅仅爬取2017级学生的课表。

```python
filter_name = []
for i in class_name:
    if i[:4]=='2017':
        filter_name.append(i)
```

通过for循环和if语句，以及利用append函数，我们爬取到2017级的课表。

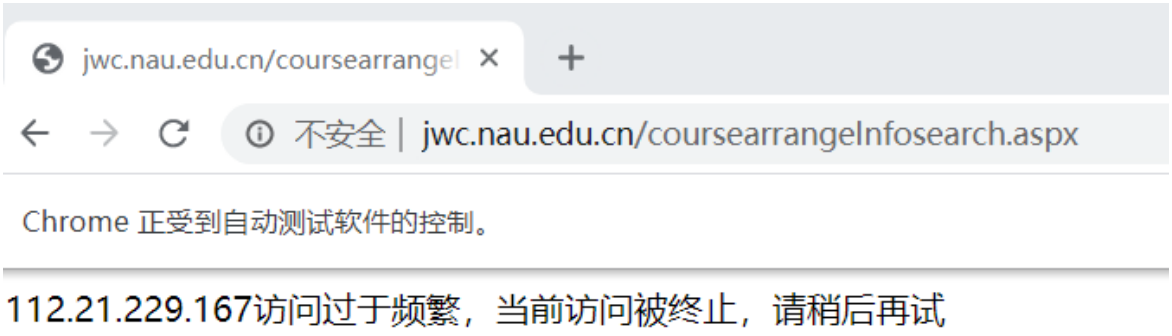
内容合并

下面我们对前面的内容进行整整合，将爬取来的数据进行处理。具体操作如下：

```python
for i in filter_name:
#     browser.get('http://jwc.nau.edu.cn/coursearrangeInfosearch.aspx')
    ls = browser.find_elements_by_css_selector('#Term+select+select')[0]
    ls.send_keys(i)
    btn = browser.find_element_by_id('searchBtn')
    btn.send_keys(Keys.ENTER)
    soup = BeautifulSoup(browser.page_source, "html.parser")
    elements = soup.select('.container tr')[1:-1]
    if flag:            #做一个判断的合并
        total = getContents(elements) #将网页内容复制给total变量
        flag = False
    else:
        total = pd.concat([total, getContents(elements)]) #将这些很多地dataframe合并在一起
    print(i)
    del filter_name[filter_name.index(i)]
```

该代码del的应用可以解决我们爬取过程中被截断的问题，主要被截断，我们就可以发现我们还有哪些数据没有爬取到，及其自动化。

执行该代码的时候，发现浏览器在自动地爬取。（需要一定的时间）整个爬取就完成了，但是中间过程可能被教务的老师所阻拦（反爬虫）。



班级	序号	课程安排	课程名	学分	教室	老师	合班	学院	学生人数
2017级财管2班	24	24	24	24	24	24	24	24	24
2017级财管1班	24	24	24	24	24	24	24	24	24
2017级IAEP1班	20	20	20	20	20	20	20	20	20
2017级IAEP2班	20	20	20	20	20	20	20	20	20
2017级信管2班	14	14	14	14	14	14	14	14	14

在这里我们只截取一些例子作为展示（小编也很害怕教务老师生气）。

这在以后的进阶内容会继续分享（我们也可以换ip地址去进行爬取，一旦报错，就可以获取数据）魔高一尺，道高一丈。

数据简单分析

爬取完这些数据，我们就可以对这些数据进行简单的数据分析（比如说那个班课最多，哪个老师的课做多。这里小编只展现一些数据处理的方法了（爬取的过程ip地址被教务封了，由此可以看出教务的反爬虫系统做的还是非常好的）。

```
total.groupby(by='老师').count().sort_values(by='序号',ascending = False).head(20)
```

这里我们运用了groupby函数首先将老师聚合，并count（）计数，然后sort值进行排序，看那些老师的课比较多，assending=False采用降序排列，但是由于数据并没有爬取全，所以这些的排序并不完全准确。

部分结果展示

老师课时数目

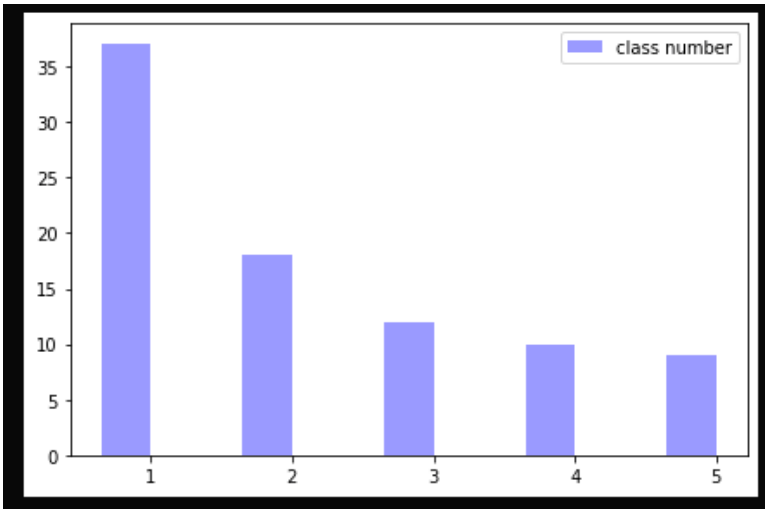
老师	序号	班级	课程安排	课程名	学分	教室	合班	学院	学生人数
讲师	37	37	37	37	37	37	37	37	37
教授	18	18	18	18	18	18	18	18	18
讲师	12	12	12	12	12	12	12	12	12
讲师	10	10	10	10	10	10	10	10	10
讲师	9	9	9	9	9	9	9	9	9
讲师	7	7	7	7	7	7	7	7	7
教授	6	6	6	6	6	6	6	6	6
教授	6	6	6	6	6	6	6	6	6
李丹	6	6	6	6	6	6	6	6	6
教授	6	6	6	6	6	6	6	6	6
讲师	6	6	6	6	6	6	6	6	6
教授	6	6	6	6	6	6	6	6	6
教授	6	6	6	6	6	6	6	6	6
教授	5	5	5	5	5	5	5	5	5
讲师	5	5	5	5	5	5	5	5	5
讲师	5	5	5	5	5	5	5	5	5
讲师	5	5	5	5	5	5	5	5	5
讲师	5	5	5	5	5	5	5	5	5
讲师	4	4	4	4	4	4	4	4	4
教授	4	4	4	4	4	4	4	4	4

数据展示

```python
from pylab import mpl
import matplotlib.pyplot as plt
import numpy as np
plt.figure(3)
x_index = np.arange(5)
x_data = ('1', '2', '3', '4', '5')
y_data = (37, 18, 12, 10, 9)
bar_width = 0.35
rects = plt.bar(x_index, y_data, width=bar_width, alpha=0.4, color='b', label='class number' )
plt.xticks(x_index + bar_width/2, x_data)
plt.legend()
plt.tight_layout()
plt.show()
```

这些代码前面的徐博凡同学已经讲解，这里就不过多讲解了，大家可以查看第二期python画图的内容，还可以画出许多有意思的图。

输出结果



我们用maplotlib展示一些数据，为了隐私问题，我们将一些老师定义为“1，2，3，4，5”，可以看出每个老师每学期的课时数目，在未来，我们可以将这些图表与班级平均成绩联系起来，就可以看出老师上课课时数和成绩是否有一定的反向关系。

根据已有的数据我们可以得出许多结果（由于隐私问题，这里就不展示老师的姓名了），由于爬取的中途过程中，我们被拦截了，所以这里只是部分数据，不代表所有。在这我们可以想想这些老师上课的地点和学生的成绩，或者上课的人数和未来期末考试成绩（爬取期末考试成绩）有什么关系等等，这都是可以提高教育质量的一种方法。

也可以有些线性关系。我们下次继续分享有些线性回归的内容，将其与爬虫结合(爬取一些具有相关性的内容）发掘更大的价值，对未来进行一定的预测，归因分析等。

爬虫的法律问题

《关于办理非法利用信息网络、帮助信息网络犯罪活动等刑事案件适用法律若干问题的解释》（法释〔2019〕15号）
第四条拒不履行信息网络安全管理义务，致使用户信息泄露，具有下列情形之一的，应当认定为刑法第二百八十六条之一第一款第二项规定的“造成严重后果”：
（一）致使泄露行踪轨迹信息、通信内容、征信信息、财产信息五百条以上的；
（二）致使泄露住宿信息、通信记录、健康生理信息、交易信息等其他可能影响人身、财产安全的用户信息五千条以上的；
（三）致使泄露第一项、第二项规定以外的用户信息五万条以上的；

严正原则：

1.不爬正常访问不到的内容

2.不超量爬取对服务器造成压力

3.不对别人的隐私造成不良影响

4.不对网站维护方产生伤害（爬虫就是模拟多个用户去访问网站，并不真正是网站维护方的真正用户，所以对服务器产生压力，对网站维护方的盈利并没有太大帮助

整体代码

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.wait import WebDriverWait
from bs4 import BeautifulSoup   #分析html数据格式
import requests             #向服务器发送请求并获取数据
import pandas as pd       #便于时候的数据分析
def getContents(elements):
    page = {}   #定义了page字典
    ind = []    #定义了一些空列表，这些空列表就是网页的内容
    cla = []
    wee = []
    nam = []
    sco = []
    roo = []
```

```
    tea = []
    mer = []
    dep = []
    cou = []
    for i in range(len(elements)):    #运用了一个for循环，这个lens就是传入soup.c
        ind.append(elements[i].select('td')[0].text) #利用append将其所有的信息这
        cla.append(elements[i].select('td')[1].text)
        wee.append(elements[i].select('td')[2].text)
        nam.append(elements[i].select('td')[3].text)
        sco.append(elements[i].select('td')[4].text)
        roo.append(elements[i].select('td')[5].text)
        tea.append(elements[i].select('td')[6].text)
        #ran.append(elements[i].select('td')[7].text)
        mer.append(elements[i].select('td')[7].text)
        dep.append(elements[i].select('td')[8].text)
        cou.append(elements[i].select('td')[9].text)
    page['序号'] = ind #规整地存到字典中去
    page['班级'] = cla
    page['课程安排'] = wee
    page['课程名'] = nam
    page['学分'] = sco
    page['教室'] = roo
    page['老师'] = tea
    page['合班'] = mer
    page['学院'] = dep
    page['学生人数'] = cou
    return pd.DataFrame(page) #返回到dataframe中
browser = webdriver.Chrome(r'C:\Program Files (x86)\Google\Chrome\Applicatior
browser.get('http://jwc.nau.edu.cn/coursearrangeInfosearch.aspx')
ls=browser.find_elements_by_css_selector('#Term+select+select')[0]
class_name = [i[1:] for i in ls.text.split('\n')]
class_name = [i[1:] for i in ls.text.split('\n')]
filter_name = []
for i in class_name:
    if i[:4]=='2017':
        filter_name.append(i)
for i in filter_name:
#     browser.get('http://jwc.nau.edu.cn/coursearrangeInfosearch.aspx')
    ls = browser.find_elements_by_css_selector('#Term+select+select')[0]
    ls.send_keys(i)
    btn = browser.find_element_by_id('searchBtn')
    btn.send_keys(Keys.ENTER)
    soup = BeautifulSoup(browser.page_source, "html.parser")
    elements = soup.select('.container tr')[1:-1]
    if flag:            #做一个判断的合并
        total = getContents(elements) #将网页内容复制给tatal变量
```

```
        flag = False
    else:
        total = pd.concat([total, getContents(elements)]) #将这些很多地datafra
    print(i)
    del filter_name[filter_name.index(i)]
total.groupby('老师').count().sort_values(by='序号', ascending=False)
total.to_csv('总课表.csv',encoding='gbk')
total = pd.read_csv('总课表.csv', encoding='gbk').iloc[:,1:]
total.groupby(by='老师').count().sort_values(by='序号',ascending = False).hea
```

本次的分享到此结束！希望大家有所收获，欢迎大家后期的关注！

往期回顾：

【Python】 No.7 机器学习基础
【Python】No.6 通过Tushare获取金融数据
【Python】No.5 Python在股票分析中的综合应用——金叉死叉交易策略与可视化
【Python】No.4 与三剑客无关的Python基础内容
【Python】No.3 Pandas的入门级教程
【Python】No.2 Numpy和Matplotlib初体验
【Python】No.1 Python安装部署和conda管理

本期撰稿人：潘徐智
南京审计大学2018级CFA2班
邮箱：1461669129@qq.com