想要完成疫情填报的自动化脚本，（默认是Windows10系统）首先要理清楚操作步骤，大致如下：访问网站→输入账号密码→查询未填报同学的名字。Python爬虫常用的库有requests、Scrapy库，两者都没有处理应对验证码的功能，需要扩展新的库才能识别处理验证码，非常的麻烦（其实是不太会），所以我选择使用selenium库来完成，selenium是一款web自动化测试工具，可以很方便地模拟真实用户对浏览器进行操作，它支持各种主流浏览器：IE、Chrome、Firefox、Safari、Opera等。在使用selenium库之前，要先确保电脑安装了python以及配置了相关环境，具体可以百度，怕麻烦的可以安装Anaconda，安装好后，它会解决大部分的工作（内置python，高度集成python数据科学生态拥有强大的包管理工具-**conda**可用超过600个python数据科学库）有一种拎包入住的感觉。

还需要安装一个代码编辑器，写代码和调试时会方便不少。

参考（https://zhuanlan.zhihu.com/p/30324113）

Selenium的安装

1、下载安装Selenium

由于Anaconda并没有selenium库，需要自己下载安装，按下快捷键win+r健打开运行，输入cmd，打开win命令行，输入pip install selenium 后马上就开始下载安装了，等待片刻后显示了很多行的Requirement already satisfied 说明安装成功了

2、安装浏览器驱动

当selenium升级到3.0之后，对不同的浏览器驱动进行了规范。如果想使用selenium驱动不同的浏览器，必须单独下载并设置不同的浏览器驱动。以Chrome浏览器为例，在chrome浏览器中输入“chrome://version/”来查看chrome版本，访问<http://chromedriver.storage.googleapis.com/index.html>下载对应版本的浏览器驱动，注：把下载好的chromedriver.exe放到python安装路径的scripts目录中即可（安装Anaconda同理），参考文章（<https://zhuanlan.zhihu.com/p/61312218>）

一切配置好后就可以开始写代码了，#!/usr/bin/python是用来说明脚本语言是python的，是要用/usr/bin下面的程序（工具）python，这个解释器，来解释python脚本，来运行python脚本的。

# -\*- coding: utf-8 -\*-是用来指定文件编码为utf-8的（因为网站是这个编码，现在的网页（包括国外）大多采用utf-8的编码，因为utf-8是国际通用的编码格式）

from selenium import webdriver是从selenium中导入webdriver模块

import time是导入时间库（用于让浏览器等待网页的反应时间），import re则是导入正则表达式库（用来规范输出获取的内容，没用上）

driver= webdriver.Chrome()是打开浏览器

1、针对浏览器的主要操作方法：

创建浏览器对象：driver = <http://webdriver.xxx()>

直接将chromedriver.exe放置在python.exe的同目录下，代码中可以直接写为：

1. driver= webdriver.Chrome()
2. 窗口最大化：maximize\_window()
3. 获取浏览器尺寸：get\_window\_size()
4. 设置浏览器尺寸：set\_window\_size()
5. 获取浏览器位置：get\_window\_position()
6. 设置浏览器位置：set\_window\_position(x,y)
7. 关闭当前标签/窗口：close()
8. 关闭所有标签/窗口：quit()

2、Selenium定位元素

因为selenium是模仿真实点击浏览器的行为，所以必须要先定位网页元素，才能进行各种操作。

定位页面元素的8种主要方式

1. id定位：driver.find\_element\_by\_id(value)
2. name属性值定位：driver.find\_element\_by\_name(value)
3. 类名定位： driver.find\_element\_by\_class\_name(value)
4. 标签名定位： driver.find\_element\_by\_tag\_name(value)
5. 链接文本定位：driver.find\_element\_by\_link\_text(value)
6. 部分链接文本：driver.find\_element\_by\_partial\_link\_text(value)
7. xpath路径表达式：driver.find\_element\_by\_xpath(value)
8. css选择器：driver.find\_element\_by\_css\_selector(value)

selenium操作网页

定位元素后，需要对网页进行各种操作，比如点击、刷新、保存等。

点击展开新的页面，点击方法：element.click()

其他主要操作方法：

请求某个url：driver.get(url)

刷新页面操作：refresh()

回退到之前的页面：back()

前进到之后的页面：forward()

获取当前访问页面url：current\_url

获取当前浏览器标题：title

保存图片：get\_screenshot\_as\_png()/get\_screenshot\_as\_file(file)

网页源码：page\_source

通过以上功能就可以找到浏览器中账号密码的位置并进行输入了，而通过获取浏览器的全局代码发现浏览器的验证码图案并不是一张图片（我是一开始通过浏览器开发调试工具发现的，打开开发调试工具的方式为按下F12键），而是实际的文本内容，通过定位验证码的位置并获取text内容就得到了验证码的内容了（这个验证码让人感到非常无语，防不了爬虫，倒是要人浪费时间去输入验证码，不过也省了一大堆识别验证码的麻烦了），然后和输入账号密码同样的步骤输入验证码即可。

关于怎么复制Xpath路径

参考（https://jingyan.baidu.com/article/6dad5075571fdaa123e36ee2.html）

接着通过定位登录按钮的位置并使用.click()功能进行点击，就登录账户了，然后同样的方法定位电力工程学院并点击，就进入了需要进入的系统页面了。

我们班现在都在外面进行实习，所以不需要进行体温填报，而是要填报信息，非常完美的是登录系统后的页面就是我们所需要的信息。

本想着提取网页的信息也会和获取验证码一样的简单，只要获取网页的内容，然后提取一下就完事了，结果发现selenium爬取不到想要的信息，通过查找资料发现是因为网站使用了frame标签，frame标签有framesetframe、iframe三种，frameset跟其他普通标签没有区别，不会影响到正常的定位，而frame与iframe对selenium定位而言是一样的，selenium有一组方法对frame进行操作。

selenium提供了switch\_to.frame(reference)方法来切换frame。

注：reference是传入的参数，用来定位frame，可以传入id、name、index以及selenium的WebElement对象，通常采用id和name就能够解决绝大多数问题。但有时候frame并无这两项属性，则可以用index和WebElement来定位：

1. index从0开始，传入整型参数即判定为用index定位，传入str参数则判定为用id/name定位
2. WebElement对象，即用find\_element系列方法所取得的对象，我们可以用tag\_name、xpath等来定位frame对象

我们的网站就属于第二种情况，然后我用了老本行通过xpath来定位进入了，用于存储已经完成填报信息的frame。

参考（<https://blog.csdn.net/huilan_same/article/details/52200586>）

接着就可以按照获取验证码的方法，我们用同样的方法获取了已经完成填报同学的信息了，我把这些信息赋值给了一个叫name的变量，而这次是用find\_elements方法而不是用find\_element方法是因为find\_element 定位的是一个元素webelement。find\_elements定位的是一组元素list【webelement1，webelement2.....】，如一个表格的所有内容（以行为单位）

但是我们的最终目的是要发现还未填报同学的名字。我的想法是通过列表的遍历找到未填报同学的名字。我们先自己制作一个带有全班姓名并分隔开的列表leng，然后通过打印发现name中的元素是一个很长的字符文本，字符文本中的元素通过一个空格符号分隔开的，于是定义了空列表lists，我们遍历name这个文本，把每次遍历到的元素添加到lists列表中，并用str.split方法将lists中的元素全部分隔开，这下我们的工作就差不多完成了。

我们接着创建了空列表list3和list4，用于存储完成填报同学的信息和未完成填报同学的信息。我们首先遍历列表leng（所有人的名字）和列表lists（完成填报同学的所有信息，包括姓名，电话等很多信息），如果计算机找到它们的相同元素，就会把相同的元素添加到list3中，遍历完成后，我们就得到了已经完成填报同学的名字，于是我们故技重施遍历leng和list3找出它们中不相同的元素，也就是未填报同学的名字，将它们添加到列表list4中，我们就得到了未填报同学的名字了。

最后我们打印出列表list4，就可以清楚的知道还有哪些同学没有完成填报了，接着我们使用driver.quit() 功能关闭浏览器，做一个不给电脑内存添麻烦的好主人。

文章至此，也接近尾声了，其实这个代码还有很多需要优化的地方，比如没有进行模块化编程，最后的输出结果并不好看，应该用正则表达式规范一下输出格式，不过人总是慢慢进步的，写这篇文章也是为了总结实习中做的事，为了超越自己，不断进步升级。

代码下载地址（<https://github.com/clh1751764916/anglechen.git>）

代码文本如下：

1. #!/usr/bin/python
2. # -\*- coding: UTF-8 -\*-
3. from selenium import webdriver
4. from selenium.webdriver import Chrome
5. import time
6. import re

9. driver= webdriver.Chrome()
10. #访问网站
11. driver.get('http://xg.fjsdxy.com/SPCP/Sys/UserLogin.aspx')
12. print(driver.title)  打印网页标签（学生健康状态填报平台）
13. time.sleep(0.5)  #等待0.5s
14. #定位用户名  "//input[@id='UserName']"
15. UserName\_el=driver.find\_element\_by\_xpath("//\*[@id='UserName']")
16. #清空表内元素
17. UserName\_el.clear()
18. #send\_keys()在元素上进行输入
19. UserName\_el.send\_keys('输入账号')
20. time.sleep(0.5)
21. #定位密码，输入密码//\*[@id="Password"]
22. Password\_el=driver.find\_element\_by\_xpath("//\*[@id='Password']")
23. #清空表内元素
24. Password\_el.clear()
25. Password\_el.send\_keys('输入密码')
26. time.sleep(0.5)
28. #获取全局代码html\_text=driver.page\_source
29. #打印print(html\_text)
31. YZM=driver.find\_element\_by\_xpath("//\*[@id='code-box']").text
32. #定位验证码框，输入验证码"//\*[@id='codeInput']"
33. yzm\_el=driver.find\_element\_by\_xpath("//\*[@id='codeInput']")
34. #清空表内元素
35. yzm\_el.clear()
36. yzm\_el.send\_keys(YZM)
37. time.sleep(0.5)

40. #driver.find\_element\_by\_id() id是唯一的，可以直接定位,queryBtn是登录按钮的id
41. driver.find\_element\_by\_id('queryBtn').click()

44. #定位电力工程学院，//\*[@id="XYQX"]/a[2]/strong/font
45. DLGCXY\_el=driver.find\_element\_by\_xpath('//\*[@id="XYQX"]/a[2]/strong/font').click()
46. time.sleep(5)

49. # 进入frame
50. driver.switch\_to.frame(driver.find\_element\_by\_xpath('//\*[@id="MainSpace"]/frame[2]'))
51. name=driver.find\_elements\_by\_xpath('//\*[@id="GridView1"]/tbody')
52. #print(name)
53. leng=["小明","小红",…]  #班级所有人的名字，做成一个列表
54. lists=[]
55. **for** i in name:
56. a=i.text
57. lists.append(a)
58. lists=str(lists).split(' ')
59. #lists=str(lists).replace(' ',',')
61. list3=[]
62. list4=[]
63. #遍历列表找相同
64. **for** i in leng:
65. **for** j in lists:
66. **if** i ==j:
67. list3.append(i)
69. #找不同
70. **for** b in(leng):
71. **if** b not in list3:
72. list4.append(b)

75. print(list4)

78. driver.quit()