一、 selenium 库

Selenium 库是 Python 的自动化测试工具 ，它支持多种浏览器包括 Chrome、Mozilla Firefox、PhantomJS 等。在爬虫中主要解决因为利用 JavaScript 渲染操作不能利用 Requests 库获取网页内容的问题。利用 JavaScript 渲染的网站有很多典型的如淘宝、今日头条、一点资讯、快递 100 等。

作用：利用 selenium 调用浏览器模拟真人自动访问目标网站。（通过 Python 程序控制浏览器）

1、搭建 selenium 环境

- 安装 selenium 库，方式与一般第三方库一致（pip install selenium）

  ```
  import selenium
  help(selenium)        #查看版本
  ```

- 安装 chrome 浏览器，并在“设置”-“关于”中，查询 chrome 的详细版本。（**不推荐**用该方法，一旦进入则自动升级，会导致与驱动不匹配）

  在 chrome 地址栏输入： **chrome://version/** **（推荐）**

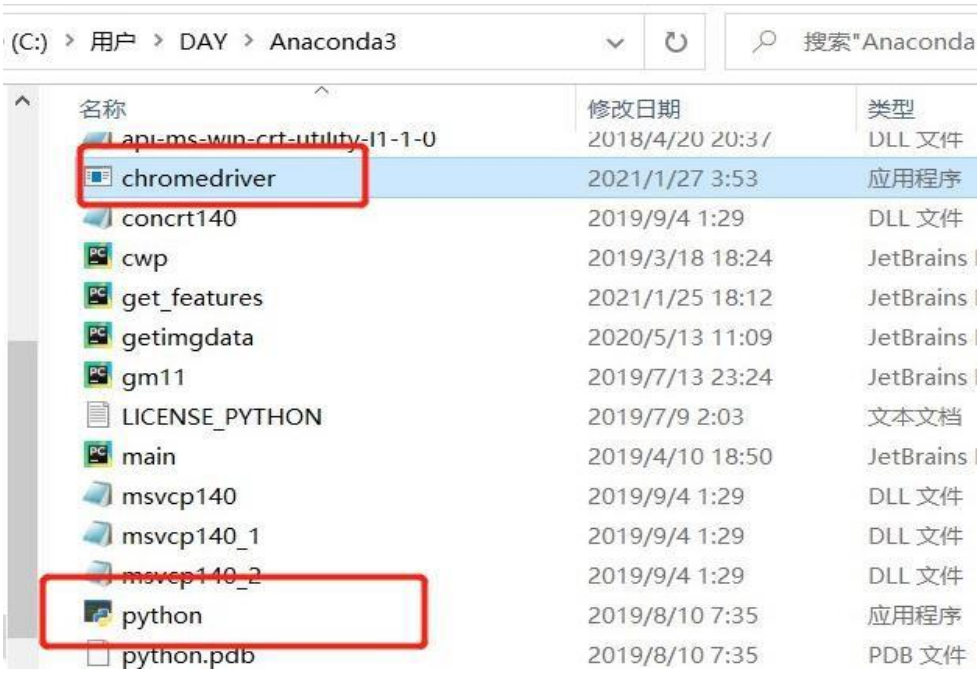
- 关闭 chrome 的自动更新。

- 下载 selenium webdriver 插件，注意版本必须和 chrome 浏览器版本完全一致。

  镜像：http://npm.taobao.org/mirrors/chromedriver/

  http://chromedriver.storage.googleapis.com/index.html

- 将插件移动至系统环境变量中，建议放置于 Python 解释器所在路径（已添加至系统环境变量中）



2、获取网页数据，写入文件：

```
# selenium 库安装完毕后，查看是否安装成功
from selenium import webdriver
from lxml import etree

driver = webdriver.Chrome()       # 打开 chrome 浏览器
driver.get('https://www.ptpress.com.cn/') # 访问指定页面
#-------------------------------------------------------------
driver.page_source   #  网页内容，str

with open('d:/aa/temp/renyou.txt','w',encoding='utf-8') as f:
    f.write(driver.page_source)   #将 driver.page_source 写入文件
# pagesource.txt 中含有网页中能看见的所有元素
# 也不一定，淘宝就不是含有所有数据，因为淘宝主页也是动态加载，滚动才会出新数据
#-------------------------------------------------------------
```

运行结果是自动打开 chrome，和人邮主页，并在 chrome 中显示：chrome 正受到自动测试软件的控制

接下来，以湖北第二师范学院主页为例，介绍 selenium 功能

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from lxml import etree

driver = webdriver.Chrome()       # 打开 chrome 浏览器
driver.get('https://www.hue.edu.cn/main.htm') #访问湖北第二师范学院主页
```

二、 定位网页中各元素

1、find_element 方法和 find_elements 方法

通过 webdriver 调用这两种方法搭配 BY 模块进行定位可以解决大部分的定位需求。

1)、find_element 方法和 find_elements 方法的区别

find_element 只返回第一个满足条件的标签。

find_elements 返回的是一个列表,是所有满足条件的标签。

2)、find_element 方法和 find_elements 方法搭配 BY 模块使用

引入 By 模块： **from selenium.webdriver.common.by import By**

格式： **find_element(By 模块,'属性值'),**

find_elements(By 模块,'属性值')

selenium By 八种方式定位元素	
1.标签的 id 属性值进行定位使用:	By.ID
2.通过标签的 name 属性值进行定位使用:	By.NAME
3.通过标签的 class 属性值进行定位使用:	By.CLASS_NAME
4.通过标签的文本值进行定位使用:	By.LINK_TEXT
5.通过标签的部分文本值进行定位使用:	By.PARTIAL_LINK_TEXT
6.通过标签的名称进行定位使用:	By.TAG_NAME
7.通过标签的 xpath 路径进行定位使用:	By.XPATH
8.通过标签的 css 选择器进行定位使用:	By.CSS_SELECTOR

【例】找到主页中的“信息公开”，并点击该标签。

```
driver.find_element(By.LINK_TEXT,"信息公开").click()
```

2. 获取元素属性

获取元素文本值：get_attribute('textContent')
获取元素属性值：get_attribute('属性名称')

（有时.text 也可以获取文本，但文本若被隐藏，则不能用此方法）

3. 元素交互操作

元素交互操作是指向页面搜索框发送文本，点击等。

判断元素是否显示： is_displayed()

向元素发送关键字： send_keys('关键字')

点击元素： click() 点击

浏览器中当前窗口： driver.current_window_handle

浏览器中所有窗口： driver.window_handles

切换至最新的窗口：driver.switch_to.window(driver.window_handles[-1])

以下这些都是窗口的句柄：

driver.window_handles[-1]

driver.window_handles[0]

window.open() 新开选项卡　　driver.close()　关闭当前选项卡

三、 等待（重点）

网页中很多内容是通过 Ajax 加载，需要一定时间才能加载出来，但 selenium 默认的是打开网页后变开始定位元素，这样容易出现找不到元素这样的问题，不过 selenium 提供了两种等待元素出现的方式：显示等待和隐式等待。（隐式暂时不管，它需要等待页面所有元素全部加载，图片视频加载很慢，导致爬取效率低；显示等待只需要等待指定元素加载即可）。除此之外，系统还提供了一种：“强制等待”机制。

1. 强制等待：sleep(0.5)，表示进程挂起 0.5 秒的时间

```
import time + time.sleep(2)
```

或

```
from time import sleep + sleep(2)
```

2. 显示等待：WebDriverWait

WebDriverWait 一般搭配 until()和 until_not()使用，能够根据条件灵活等待，主要过程是：程序每隔 0.5 秒（默认）检查，如果条件成立，则正常返回；否则继续等待，直到超时设置的最长时间，就会抛出 TimeoutException 异常。使用格式如下：

```
WebDriverWait(driver,timeout).until(method,message='')
WebDriverWait(driver,timeout).until_not(method,message='')
```

例如：

```
wait=WebDriverWait(driver,10)    # 定义一个等待，语句较长通常分开定义
bookelem = wait.until( EC.visibility_of_all_elements_located (
            ( By.XPATH,'//*[@id="newBook"]/div[2]/div[2]//p') )    )
#等待，直到本页面所有满足路径的 p 结点全部可见，bookelem 是一个元素列表
booklist=[i.text for i in bookelem]    # 获取每个 p 结点的文本
```

method 表示显示等待过程中，我们期待出现的条件，需要导入一个 expected_conditions 类才能使用。

```
from selenium.webdriver.support import expected_conditions as EC
```

由此，如果需要在获取数据时，用到显示等待机制，则共需要引入 4 个模块：

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
```

method 有数十种，最常见的是以下几个：

① 判断某个元素是否被加到 dom 树中，并不代表该元素一定可见

```
EC.presence_of_element_located(…)
```

② 判断某个元素是否可见，可见代表元素非隐藏，并且元素的宽和高都不等于 0，即不仅加载到 dom 树中，也必须是可见的。

```
EC.visibility_of_element_located(…)
```

③ 判断某个元素中是否可见并且是 enable 的，这样才是 clickable。

```
EC.element_to_be_clickable(…)  确认元素是否是可点击的
```

以上，返回值都是单个元素。

```
EC.presence_of_all_elements_located( (By.XPATH, '…') )
EC.visibility_of_all_elements_located( (By.XPATH, '…') )
```

以上返回值为元素列表。