第十关 指挥浏览器：自动爬取网页数据

2022年12月28日 8:43

```
# 自动登录网站
from selenium import webdriver
# 导入by方法
from selenium.webdriver.common.by import By
import pytesseract
from selenium.webdriver.edge.options import Options
import time
# 获取手机号码
phone_number = input('请输入手机号码：')
# 输入密码
password = input('输入密码')
# 设置静默模式
#options = Options()
#options.headless = True
# 初始化浏览器
driver = webdriver.Edge()
driver.get('https://music.facode.cn/index.php/Home/Index/login.html')


# 定位手机号码标签，并输入手机号码
phone_tag = driver.find_element(By.NAME, 'phone')
phone_tag.send_keys(phone_number)
# 一次性定位密码标签，并输入密码
pass_tag = driver.find_element(By.NAME, 'pass')
pass_tag.send_keys(password)
# 获取验证码图片数字
graph = driver.find_element(By.ID, 'graph_img')
# 截取图片内容，这里的内容是一组二进制字节
graph_bytes = graph.screenshot_as_png
# 为了获取验证码图片里的数字，必须保存刚才获取的内容已图片格式,这里wb 表示已二进制写入
with open('D:\PythonTest\风变python学习资料\Python爬虫\验证码.png','wb') as f:
    f.write(graph_bytes)
# 获取刚才保存的验证码数字
verify_number = pytesseract.image_to_string('D:\PythonTest\风变python学习资料\Python爬虫\验证码.png')
# 定位验证码 并输入
verify_tag = driver.find_element(By.NAME,'verify')
verify_tag.send_keys(verify_number.strip())  # 传入的数字也要去掉换行符，否则识别不了
# 定位登录按钮，并点击
log_button = driver.find_element(By.CLASS_NAME,'lowin-btn')
log_button.click()
# 获取登录网站的cookies
cookies_list = driver.get_cookies()
# 创建储存 cookie 的空字符
cookies = ''.join(
    f"{cookie['name']}={cookie['value']};" for cookie in cookies_list
)
# 设置请求头
header = {'Cookie':cookies}
# 打印请求头
print(header)

# 窗口停留2秒
time.sleep(2)
# 关闭浏览器
driver.quit()
```

```python
# 自动登录网站

from selenium import webdriver
# 导入by方法
from selenium.webdriver.common.by import By

import pytesseract

from selenium.webdriver.edge.options import Options

import time
```

导入需要的库

```python
# 获取手机号码
phone_number = input('请输入手机号码：')
# 输入密码
password = input('输入密码')
# 设置静默模式
#options = Options()
#options.headless = True

# 初始化浏览器
driver = webdriver.Edge()
driver.get('https://music.facode.cn/index.php/Home/Index/login.html')
```

模拟登录网页，实例化webdriver

```python
# 定位手机号码标签，并输入手机号码
phone_tag = driver.find_element(By.NAME, 'phone')

phone_tag.send_keys(phone_number)

# 一次性定位密码标签，并输入密码
pass_tag = driver.find_element(By.NAME, 'pass')
pass_tag.send_keys(password)
```

查找元素，通过By后面的方法Name，定位名字的方法，在通过send_keys传入

```python
# 获取验证码图片数字
graph = driver.find_element(By.ID, 'graph_img')
# 截取图片内容，这里的内容是一组二进制字节
graph_bytes = graph.screenshot_as_png
# 为了获取验证码图片里的数字，必须保存刚才获取的内容已图片格式，这里wb 表示已二进制写入
with open('D:\PythonTest\风变python学习资料\Python爬虫\验证码.png','wb') as f:
    f.write(graph_bytes)

# 获取刚才保存的验证码数字
verify_number = pytesseract.image_to_string('D:\PythonTest\风变python学习资料\Python爬虫\验证码.png')

# 定位验证码 并输入
verify_tag = driver.find_element(By.NAME,'verify')
verify_tag.send_keys(verify_number.strip())  # 传入的数字也要去掉换行符，否则识别不了
```

通过By后面ID方法定位验证码图片，并截图，截图已二进制方式保存。

保存图片

提取图片中的数字

定位验证码输入框，将数字传入

```python
# 定位登录按钮，并点击
log_button = driver.find_element(By.CLASS_NAME,'lowin-btn')
log_button.click()
```


```python
# 获取登录网站的cookies
cookies_list = driver.get_cookies()

# 创建储存 cookie 的空字符
cookies = ''.join(
    f"{cookie['name']}={cookie['value']};" for cookie in cookies_list
)

# 设置请求头
header = {'Cookie':cookies}
# 打印请求头
print(header)


# 窗口停留2秒
time.sleep(2)

# 关闭浏览器
driver.quit()
```

3. selenium 库

selenium 库是一个第三方库，可以自动化地对浏览器进行操控。

比如，它可以代替人工找 bug 的操作，实现自动化测试。同时，它也是绕开反爬虫的重要方式，例如我们今天使

用它来获取 cookie。

由于是第三方库，若要在本地编译器中使用该库，需要执行以下指令进行安装：

1）**Windows 系统**：pip install selenium

2）**Mac 系统**：pip3 install selenium

而 selenium 控制浏览器，需要借助库中的模块 webdriver。

3.1 webdriver 模块

webdriver 模块提供了自动化控制 Web 浏览器的调用接口，可通过浏览器的驱动程序来控制浏览器。

这里的**驱动程序**就像字面意思，是我们用于驱动浏览器工作的程序。

所以，为了操作浏览器，我们还得下载浏览器的相关驱动程序。比如我们课程学习所使用的浏览器为谷歌浏览

器，那么就需要下载 Chrome 驱动程序。

课程中已完成相应的配置，但为了让你在本地编译器上也能成功执行代码，我为你准备好了驱动程序的配置流

程，你可以根据文档里面的流程进行驱动的下载配置：Chrome 浏览器驱动配置流程。

```python
# 练习 打开百度 查看源码
from selenium import webdriver
# 不使用静默模式
# 初始化浏览器
browser = webdriver.Edge()
browser.get('https://www.baidu.com')
print(browser.page_source)
browser.quit()
```

```
Output exceeds the size limit. Open the full output data in a text editor
<html><head><script type="text/javascript" charset="utf-8"
src="https://dss0.bdstatic.com/5aV1bjqh_Q23odCf/static/superman/js/components/advert-
064271ed9b.js"></script><script type="text/javascript" charset="utf-8"
src="https://dss0.bdstatic.com/5aV1bjqh_Q23odCf/static/superman/js/components/qrcode-
0e4b67354f.js"></script><script type="text/javascript" charset="utf-8"
src="https://dss0.bdstatic.com/5aV1bjqh_Q23odCf/static/superman/js/super_load-86e18c5005.js">
</script><script type="text/javascript" charset="utf-8"
src="https://dss0.bdstatic.com/5aV1bjqh_Q23odCf/static/superman/js/components/tips-
e2ceadd14d.js"></script><meta http-equiv="Content-Type" content="text/html;charset=utf-8"><me
http-equiv="X-UA-Compatible" content="IE=edge,chrome=1"><meta content="always" name="referrer
<meta name="theme-color" content="#ffffff"><meta name="description" content="全球领先的中文搜索
擎、致力于让网民更便捷地获取信息，找到所求。百度超过千亿的中文网页数据库，可以瞬间找到相关的搜索结果。
```

Selenium用法 主要就4步：

- 1.创建WebDriver对象；
- 2.用WebDriver对象的**get方法**打开网址；
- 3.用WebDriver对象的**find_element方法**去找到网页上你需要的元素，返回WebElement对象（找到对象是重点）；
- 4.对WebElement对象进行操作，一般就是获取对象的内容、填写入内容或者click操作。

这篇文章主要针对第3步，列举一些查找元素的方法。（文章代码中使用selenium4版本）

在浏览器中查找网页元素，直接按F12，打开开发者模式，在【元素】（Elements）选项中就可以查看网页源码。

查找网页元素时，有两个情况：

- webdriver.find_element：只查找1个元素（返回WebElement对象）；
- webdriver.find_elements：查找多个元素（返回一个List的WebElement对象）

1.简单的直接查找

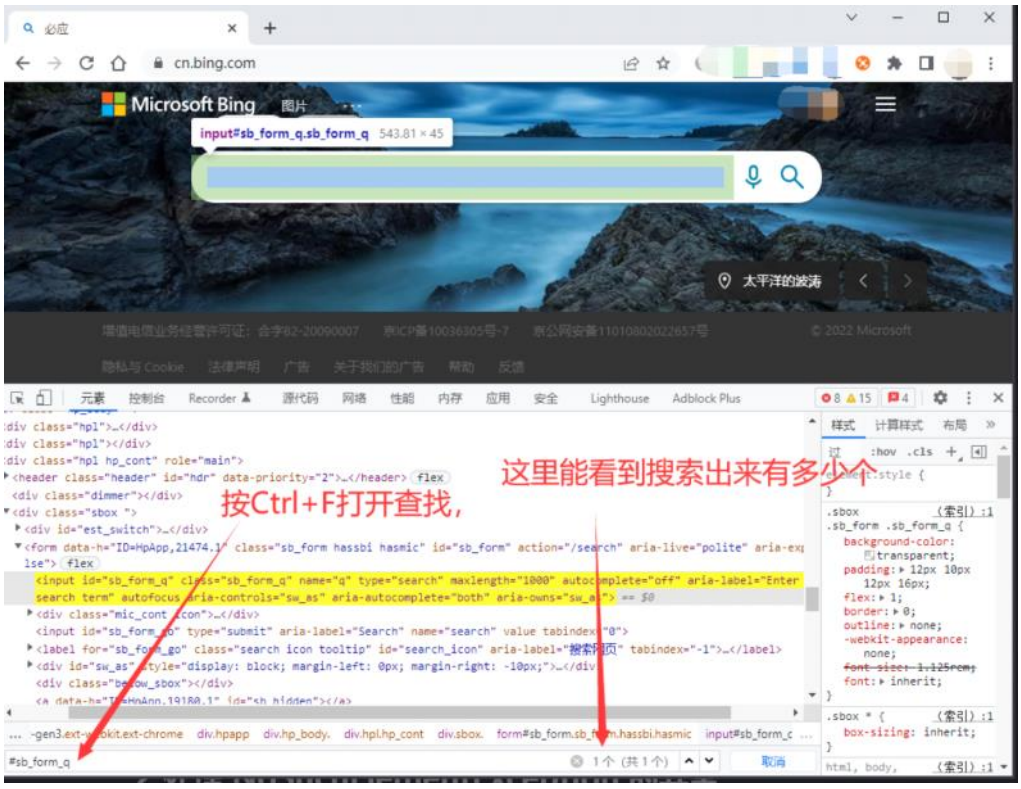
- By.ID：通过HTML标签的id属性（id属性在HTML文档中具体唯一性，所以有id优先通过这个查找）
- By.CLASS_NAME：通过HTML标签的class属性（class属性并没有唯一性，如果有多个元素的话，只会返回第1个；当然有些class在站点中唯一的话，也是可以用来查找的）
- By.TAG_NAME：通过HTML标签（多个元素的话，也是返回第1个）

对应代码为：

```
from selenium.webdriver.common.by import By
# 初始化代码 ...
wd = webdriver.Chrome(options=option)
wd.get('https://cn.bing.com/')
# ...

e1 = wd.find_element(By.ID, 'sb_form_q') # 搜索框
e2 = wd.find_element(By.CLASS_NAME, 'sb_form_q')
e3 = wd.find_element(By.TAG_NAME, 'input')
```

在F12打开的开发者工具中，按 Ctrl+F键，打开搜索。
如果是id，搜索id名前面加上#号，直接就能搜索id。如图所示：



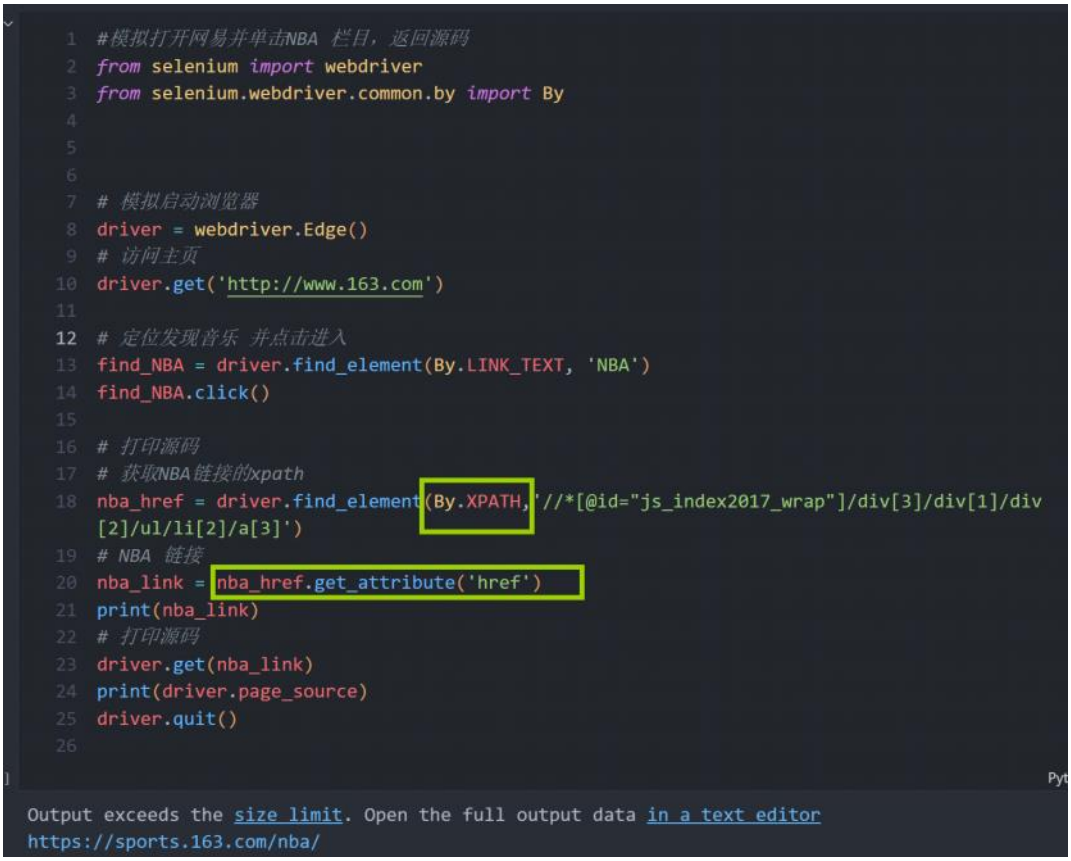
可以用 WebElement.send_keys('搜索内容')测试是否找到了这个元素，或者直接print一下。
如果找不到元素，或者有这个元素、网站响应速度比代码执行速度慢点，find_element 方法就会抛出 selenium.common.exceptions.NoSuchElementException 异常。

4.Xpath选择器（全能查找）

用css查找基本上很方便、强大了，只是有些场景用 css 查找元素会比较麻烦，而xpath比较方便。
另外 Xpath 还有其他领域会使用到，比如爬虫框架 Scrapy，手机App框架 Appium。

用Xpath的时候，代码就用By.XPATH，类似这样： `wd.find_element(By.XPATH,'/html/body/div')`

等回头研究用爬虫的时候，用的例子多一些，再单独写一篇。此篇略。



```python
#模拟打开网易并单击NBA 栏目，返回源码
from selenium import webdriver
from selenium.webdriver.common.by import By



# 模拟启动浏览器
driver = webdriver.Edge()
# 访问主页
driver.get('http://www.163.com')

# 定位发现音乐 并点击进入
find_NBA = driver.find_element(By.LINK_TEXT, 'NBA')
find_NBA.click()

# 打印源码
# 获取NBA链接的xpath
nba_href = driver.find_element(By.XPATH,'//*[@id="js_index2017_wrap"]/div[3]/div[1]/div[2]/ul/li[2]/a[3]')
# NBA 链接
nba_link = nba_href.get_attribute('href')
print(nba_link)
# 打印源码
driver.get(nba_link)
print(driver.page_source)
driver.quit()
```

```
Output exceeds the size limit. Open the full output data in a text editor
https://sports.163.com/nba/
```


```python
#模拟打开网易并单击NBA 栏目，返回源码
from selenium import webdriver
from selenium.webdriver.common.by import By


# 模拟启动浏览器
driver = webdriver.Edge()
# 访问主页
driver.get('http://www.163.com')
# 定位发现音乐 并点击进入
find_NBA = driver.find_element(By.LINK_TEXT, 'NBA')
find_NBA.click()
# 打印源码
# 获取NBA链接的xpath
nba_href = driver.find_element(By.XPATH,'//*[@id="js_index2017_wrap"]/div[3]/div[1]/div[2]/ul/li[2]/a[3]')
# NBA 链接
nba_link = nba_href.get_attribute('href')
print(nba_link)
# 打印源码
driver.get(nba_link)
print(driver.page_source)
driver.quit()
```

二、查找多个元素（find_elements方法）

比如有时候一个class_name会有很多地方在用，就可以查多个元素。

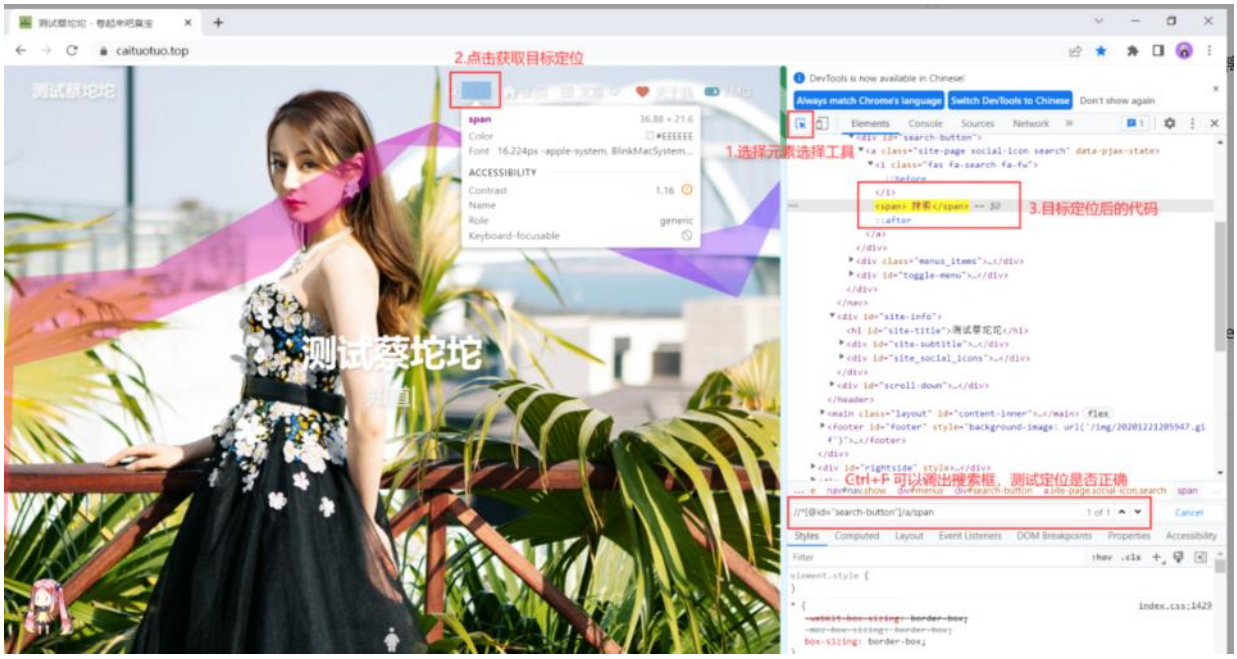
find_elements方法返回是一个List的列表，所以用for循环一下就能得到每一个。

```
elements = wd.find_elements(By.CLASS_NAME, 'sb_form_q')
for element in elements:
    print(element.text)
```

Selenium八大元素定位

所谓八大元素定位方式就是
id、name、class_name、tag_name、link_text、partial_link_text、xpath、css_selector。

在介绍定位方式之前先来说一下定位工具，以Chrome浏览器为例，使用F12或右键检查进入开发者工具



ID

通过元素的id属性定位，一般情况下id在当前页面中是唯一的。使用id选择器的前提条件是元素必须要有id属性。由于id值一般是唯一的，因此当元素存在id属性值时，优先使用id方式定位元素。

例如：下面的这个input标签的id属性值为kw

例如：下面的这个input标签的id属性值为kw

```
<input type="text" class="s_ipt" name="wd" id="kw" maxlength="100" autocomplete="off">
```

语法：

登录后复制

```
driver.find_element(By.ID, "id属性值")
```

举栗：

登录后复制

```
# author: 测试蔡坨坨
# datetime: 2022/10/22 19:08
# function: id定位

import time
from selenium import webdriver
from selenium.webdriver.common.by import By

driver = webdriver.Chrome()
driver.get("https://www.baidu.com")
driver.find_element(By.ID, "kw").send_keys("测试蔡坨坨")
driver.find_element(By.ID, "su").click()
time.sleep(3)
driver.quit()
```

NAME

通过元素的name属性来定位。name定位方式使用的前提条件是元素必须有name属性。由于元素的name属性值可能存在重复，所以必须确定其能够代表目标元素唯一性后，方可使用。

当页面内有多个元素的特征值相同时，定位元素的方法执行时只会默认获取第一个符合要求的特征对应的元素。

例如：下面的这个input标签的name属性值为wd

```
<input type="text" class="s_ipt" name="wd" id="kw" maxlength="100" autocomplete="off">
```

语法：

```
driver.find_element(By.NAME, "name属性值")
```

举栗：

登录后复制

```
# author: 测试蔡坨坨
# datetime: 2022/10/22 19:23
# function: name定位

import time
from selenium import webdriver
from selenium.webdriver.common.by import By

driver = webdriver.Chrome()
driver.get("https://www.baidu.com")
driver.find_element(By.NAME, "wd").send_keys("测试蔡坨坨")
driver.find_element(By.ID, "su").click()
time.sleep(3)
driver.quit()
```

CLASS_NAME

通过元素的class属性来定位，class属性一般为多个值。使用class定位方式的前提条件是元素必须要有class属性。

虽然方法名是class_name，但是我们要找的是class属性。

例如：下面这个input标签的class属性值为but1

登录后复制

```
<input class="but1" type="text" name="key" placeholder="请输入你要查找的关键字" value="">
```

语法：

```
driver.find_element(By.CLASS_NAME, "class属性值")
```

举栗：

```
# author: 测试蔡坨坨
# datetime: 2022/10/22 19:31
# function: class定位

import time
from selenium import webdriver
from selenium.webdriver.common.by import By

driver = webdriver.Chrome()
# 打开电商网站
driver.get("http://127.0.0.1")
driver.maximize_window()

# 搜索框中输入 鞋子
driver.find_element(By.CLASS_NAME, "but1").send_keys("鞋子")
# 点击搜索
driver.find_element(By.CLASS_NAME, "but2").click()
```

注意：如果class name是一个复合类（存在多个属性值，每个属性值以空格隔开），则只能使用其中的任意一个属性值进行定位，但是不建议这么做，因为可能会定位到多个元素。

例如：下面这个标签的class属性值为bg s_btn btn_h btnhover

```
<input class="bg s_btn btn_h btnhover" type="text" name="key">
```

则只能使用复合类的任意一个单词去定位：

```
driver.find_element(By.CLASS_NAME,"bg") # 正确示范

driver.find_element(By.CLASS_NAME,"bg s_btn btn_h btnhover") # 错误示范 NoSuchElementException
```

TAG_NAME

通过元素的标签名称来定位，例如input标签、button标签、a标签等。

由于存在大量标签，并且重复性高，因此必须确定其能够代表目标元素唯一性后，方可使用。如果页面中存在多个相同标签，默认返回第一个标签元素。一般情况下标签重复性过高，要精确定位，都不会选择tag_name定位方式。

语法：

```
driver.find_element(By.TAG_NAME, "标签名称")
```

举栗：

登录后复制

```
driver.find_element(By.TAG_NAME, "input")
```

LINK_TEXT

定位超链接标签。只能使用精准匹配（即a标签的全部文本内容），该方法只针对超链接元素（a 标签），并且需要输入超链接的全部文本信息。

例如：下面这个a标签的全部文本内容为联系客服

```
<a href="http://XXX">联系客服</a>
```

语法：

```
driver.find_element(By.LINK_TEXT, "a标签的全部文本内容")
```

举栗：

登录后复制

```
# author: 测试蔡坨坨
# datetime: 2022/10/22 20:27
# function: link_text定位

import time
from selenium import webdriver
from selenium.webdriver.common.by import By

driver = webdriver.Chrome()
driver.get("http://127.0.0.1")
driver.maximize_window()

# 点击联系客服
driver.find_element(By.LINK_TEXT, "联系客服").click()
```

PARTIAL_LINK_TEXT

定位超链接标签，与LINK_TEXT不同的是它可以使用精准或模糊匹配，也就是a标签的部分文本内容，如果使用模糊匹配最好使用能代表唯一的关键词，如果有多个元素，默认返回第一个。

例如：下面这个a标签的全部文本内容为“联系客服”，模糊匹配就可以使用a标签的部分文本内容，比如联系、客服、联、服.....

```
<a href="http://XXX">联系客服</a>
```

语法：

```
driver.find_element(By.PARTIAL_LINK_TEXT, "a标签的部分文本内容")
```

举栗：

```
# author: 测试蔡坨坨
# datetime: 2022/10/22 20:34
# function: partial_link_text定位器

import time
from selenium import webdriver
from selenium.webdriver.common.by import By

driver = webdriver.Chrome()
driver.get("http://127.0.0.1")
driver.maximize_window()
# 点击联系客服
driver.find_element(By.PARTIAL_LINK_TEXT, "联系").click()
```

XPATH

定义

XML Path Language 的简称，用于解析XML和HTML。（不仅可以解析XML还可以解析HTML，因为HTML与XML是非常相像的，XML多用于传输和存储数据，侧重于数据，HTML多用于显示数据并关注数据的外观）

Xpath策略有多种，无论使用哪一种策略，定位的方法都是同一个，不同策略只决定方法的参数的写法。

Xpath不仅可以用于Selenium，还适用于Appium，是一个万能的定位方式。

Xpath有一个缺点，就是速度比较慢，比CSS_SELECT要慢很多，因为Xpath是从头到尾一点一点去遍历。

绝对路径

从最外层元素到指定元素之间所有经过元素层级的路径，绝对路径是以/html根节点开始，使用 来分割元素层级的语法，比如：/html/body/div[2]/div/div[2]/div[1]/form/input[1]（因为会有多个div标签，所以用索引的方式定位div[2]，且XPath的下标是从1开始的，例如：/bookstore/bool[1]表示选取属于bookstore子元素的第一个book元素，除了用数字索引外，还可以用last()、position()函数来表达索引，例如：/bookstore/book[last()]表示选取属于bookstore子元素的最后一个book元素，/bookstore/book[last()-1]表示选取属于bookstore子元素的倒数第二个book元素，/bookstore/book[position()<3]表示选取最前面的两个属于bookstore元素的子元素的book元素）

由于绝对路径对页面结构要求比较严格，因此不建议使用绝对路径。

语法：

```
driver.find_element(By.XPATH, "/html开头的绝对路径")
```

举栗：

登录后复制

```
# author: 测试蔡坨坨
# datetime: 2022/10/23 11:13
# function: xpath绝对路径

import time
from selenium import webdriver
from selenium.webdriver.common.by import By

driver = webdriver.Chrome()
# 打开电商网站
driver.get("http://127.0.0.1")
driver.maximize_window()

# 绝对路径
# 搜索框输入 阿迪达斯
# XPath的下标是从1开始的
driver.find_element(By.XPATH, "/html/body/div[2]/div/div[2]/div[1]/form/input[1]").send_keys("阿迪达斯")
# 点击搜索
driver.find_element(By.XPATH, "/html/body/div[2]/div/div[2]/div[1]/form/input[2]").click()

driver.quit()
```

相对路径

匹配任意层级的元素，不限制元素的位置，相对路径是以 开始，后面跟元素名称，不知元素名称时可以使用 * 号代替，在实际应用中推荐使用相对路径。

语法：

```
driver.find_element(By.XPATH, "//input")

driver.find_element(By.XPATH, "//*")
```

举栗：

登录后复制

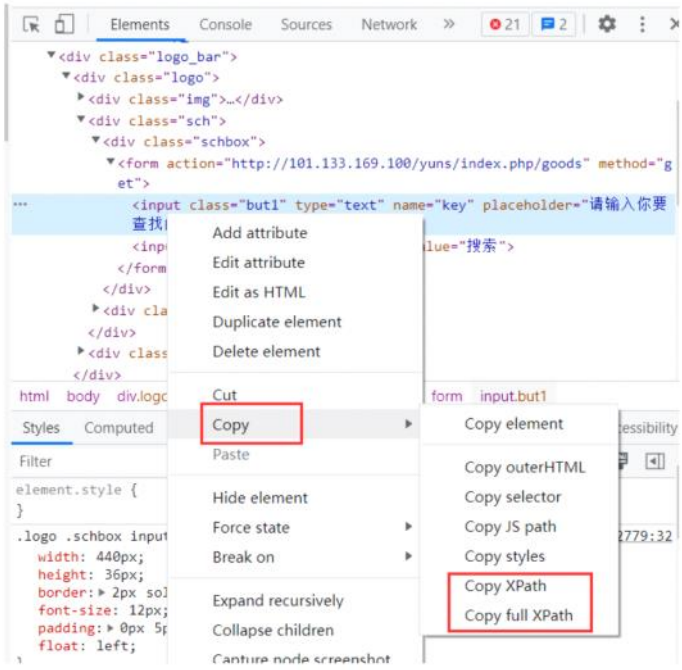
```
# author: 测试蔡坨坨
# datetime: 2022/10/23 12:35
# function: xpath相对路径

import time
from selenium import webdriver
from selenium.webdriver.common.by import By

driver = webdriver.Chrome()
# 打开电商网站
driver.get("http://127.0.0.1")
driver.maximize_window()

# 相对路径
# XPath相对路径以 // 开头
# 搜索框输入 鞋子
driver.find_element(By.XPATH, "//input[@class='but1']").send_keys("鞋子")
# 点击搜索按钮
driver.find_element(By.XPATH, "//*[@class='but2']").click()
```

使用浏览器开发者工具直接复制xpath路径值（偷懒的方法，不推荐在学习的时候使用）：



通过元素属性定位

单个属性

使用目标元素的任意一个属性和属性值（需保证唯一性）。

注意：

使用 XPath 策略，建议先在浏览器开发者工具中根据策略语法，组装策略值，测试验证后再放入代码中使用。

目标元素的有些属性和属性值可能存在多个相同特征的元素，需注意唯一性。

语法：

登录后复制

```
driver.find_element(By.XPATH, "//标签名[@属性='属性值']")

driver.find_element(By.XPATH, "//*[@属性='属性值']")
```

比如：下面这个input标签的placeholder属性的属性值为“请输入你要查找的关键字”

```
<input class="but1" type="text" name="key" placeholder="请输入你要查找的关键字">
```

举栗：

登录后复制

```
# author: 测试蔡坨坨
# datetime: 2022/10/23 17:27
# function: 单个属性

import time
from selenium import webdriver
from selenium.webdriver.common.by import By

driver = webdriver.Chrome()
driver.get("http://127.0.0.1")
driver.maximize_window()

# 通过单个属性匹配
driver.find_element(By.XPATH, "//input[@placeholder='请输入你要查找的关键字']").send_keys("测试蔡坨坨")
```

4.4.1 pytesseract 模块

pytesseract 模块是 Python 的第三方 OCR (Optical Character Recognition) 工具，可用于识别、读取图像中的内容。

由于也是第三方模块，在本地编译器中使用该模块，同样需要执行以下指令进行安装：

1）**Windows 系统**：pip install pytesseract

2）**Mac 系统**：pip3 install pytesseract

安装完该模块后不能直接使用，还需要本地下载 Tesseract-OCR，才能用以识别图片中的数字。

课程中已经完成相应的配置，但如果你想在本地使用 pytesseract，可以根据此文档进行 Tesseract-OCR 的下载配

置：Tesseract 的安装流程。windows10 - Windows安装Tesseract-OCR 4.00并配置环境变量 - 全栈开发之路 - SegmentFault 思否

现在我们来体验一下 pytesseract 识别图片内容的功能：

```
1  # 识别图片文字
2  # 注意识别的内容可能里面存在换行或者空格。
3
4  import pytesseract
5
6  number = pytesseract.image_to_string('D:\PythonTest\风变python学习资料\Python爬虫\圆周率.
   png')
7  print(number)
8
```

```
3.1415926535897932
```

```
1  # 识别图片文字
2  # 注意识别的内容可能里面存在换行或者空格。
3
4  import pytesseract
5
6  number = pytesseract.image_to_string('D:\PythonTest\风变python学习资料\Python爬虫\圆周率.
   png')
7  print([number])
```

```
['3.1415926535897932\n']
```

```
1  # 识别图片文字
2  # 注意识别的内容可能里面存在换行或者空格。
3
4  import pytesseract
5
6  number = pytesseract.image_to_string('D:\PythonTest\风变python学习资料\Python爬虫\圆周率.
   png')
7  print([number.strip()])
```

```
['3.1415926535897932']
```

具体详细使用方法可以参考：(4条消息) Python OCR工具pytesseract详解_测试开发小记的博客-CSDN博客_pytesseract

3、识别图片中的文字

image_to_string()用来识别图片中的文字，最简单的用法传入2个入参，一个是图片的文件名称，一个是识别所用的语言包类型，比如要识别下图中的文字，这是一段从pdf文件中截屏的片段，文件名为bookseg.png，语言包选择chi_sim：

```
1  import pytesseract
2  text = pytesseract.image_to_string('D:\PythonTest\风变python学习资料\Python爬虫\识别文字.
   png',lang='chi_sim')
3  print(text)
```

```
0.6s
```

```
引言

数字图像处理方法的重要性源于两个主要应用领域：改善图示信息以便人们解释；为存储、传
输和表示而对图像数据进行处理，以便于机器自动理解。本章有几个主要目的：(D) 定义我们称之为
图像处理领域的范围；(2)从历史观点回顾图像处理的起源；(3) 通过考察一些主要的应用领域，给出
图像处理技术状况的概念；(4) 简要讨论数字图像处理中所用的主要方法；(5) 概述通用目的的典型图
像处理系统的组成；〈6) 列出公开发表的数字图像处理领域的一些图书和文献。
```

4、获取图片中文字的详细信息

image_to_data()用来获取识别出来的文字的详细信息，包含识别到的文本内容，可信度，位置等：

最后一列是识别出来的文本内容，往前一列是识别出来的可信度，再往前4列是在图片中的位置，包含left，top，width，height等4个要素。

注意image_to_data()返回的是str类型的数据，如果要使用其中的conf可信度，left，top等位置信息，还需要经过提取、转换才能得到。

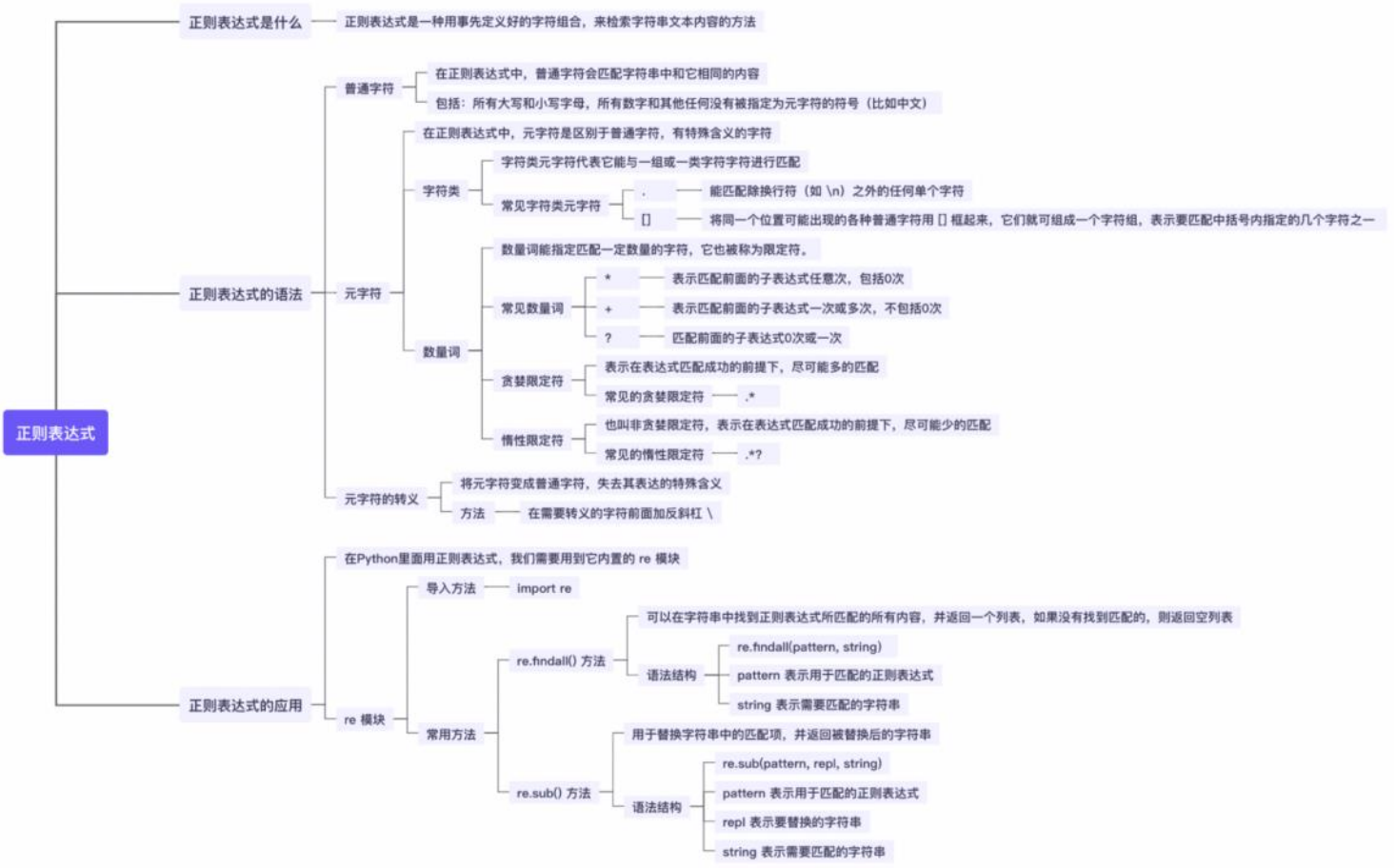
```python
import pytesseract
text = pytesseract.image_to_data('D:\PythonTest\风变python学习资料\Python爬虫\识别文字.png', lang='chi_sim')
print(text)
```

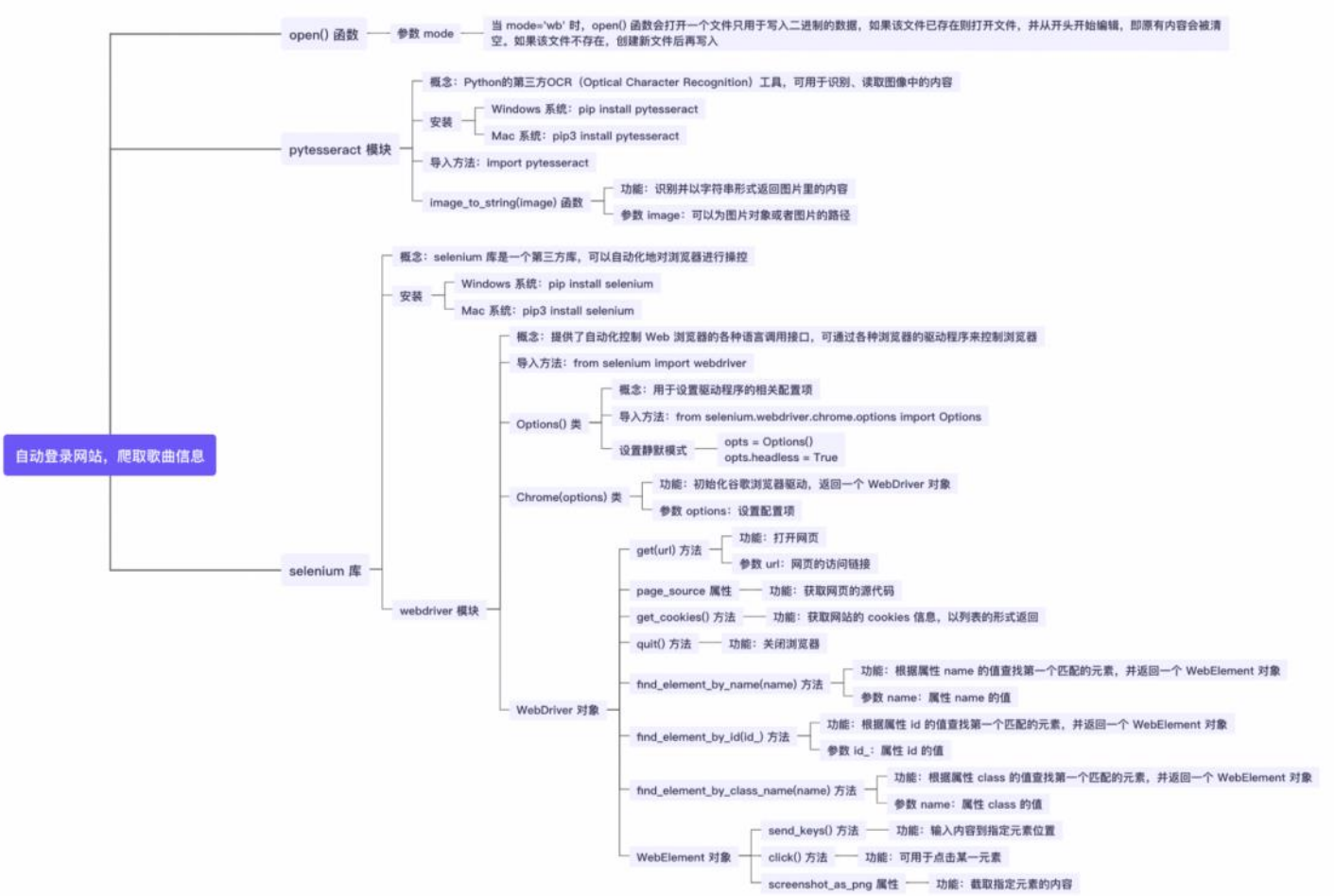
[5] ✓ 0.6s Python

Output exceeds the size limit. Open the full output data in a text editor

level	page_num	block_num	par_num	line_num	word_num	left	top	width	height	conf	text
1	1	0	0	0	0	0	0	1319	341	-1	
2	1	1	0	0	0	37	33	1240	278	-1	
3	1	1	1	0	0	41	33	74	37	-1	
4	1	1	1	1	0	41	33	74	37	-1	
5	1	1	1	1	1	41	33	74	37	96.128098	引
言											
3	1	1	2	0	0	37	111	1240	200	-1	
4	1	1	2	1	0	97	111	1180	28	-1	
5	1	1	2	1	1	97	112	63	26	96.479385	数
字											
5	1	1	2	1	2	164	111	28	28	93.296310	图
5	1	1	2	1	3	197	111	12	27	92.702560	像
5	1	1	2	1	4	225	112	13	25	93.217659	处
5	1	1	2	1	5	252	113	16	24	93.294319	理
5	1	1	2	1	6	280	112	45	26	96.948677	方
法											
5	1	1	2	1	7	337	112	17	26	96.979988	的


动态网页爬取
静态网页和动态网页
静态网页
特点
通过 HTML 文档所在的请求能获取所有内容
举例
闪光读书网的书籍评论
豆瓣电影 Top250 的电影信息
动态网页
特点
通过 HTML 文档所在的请求不能获取所有内容
举例
在某搜索引擎输入不同的关键词，会出现不同的搜索提示
在某博热门资讯页向下滚动页面，可以不断地刷出新的内容
AJAX
定义
AJAX 是一套综合了多项技术的网页开发技术
优势
在不刷新页面的情况下，通过 XHR 对象向服务器请求数据，然后插入到页面中进行呈现
XHR
概念
XHR 对象是 AJAX 技术的核心
查看 xhr 请求
步骤
1）右键【检查】打开开发者工具；
2）点击 Network 选项卡；
3）点击 XHR 筛选 xhr 请求；
4）点击【清空】按钮，然后刷新页面；
5）打开 xhr 请求，查看 Headers、Preview、Response
数据结构
一般为 JSON 格式文本
JSON
定义
JSON 是一种基于文本的数据交换方式
语法规则
JSON 由字典结构和列表结构自由组合而成，可以进行多次嵌套
字典结构：{key1: value1, key2: value2, key3: value3...}
列表结构：[元素 1, 元素 2, 元素 3...]
Response.json() 方法
将响应的 JSON 格式文本转换为字典


正则表达式
正则表达式是什么
正则表达式是一种用事先定义好的字符组合，来检索字符串文本内容的方法
正则表达式的语法
普通字符
在正则表达式中，普通字符会匹配字符串中和它相同的内容
包括：所有大写和小写字母，所有数字和其他任何没有被指定为元字符的符号（比如中文）
元字符
在正则表达式中，元字符是区别于普通字符，有特殊含义的字符
字符类
字符类元字符代表它能与一组或一类字符字符进行匹配
常见字符类元字符
.
能匹配除换行符（如 \n）之外的任何单个字符
[]
将同一个位置可能出现的各种普通字符用 [] 框起来，它们就可组成一个字符组，表示要匹配中括号内指定的几个字符之一
数量词
数量词能指定匹配一定数量的字符，它也被称为限定符。
常见数量词
*
表示匹配前面的子表达式任意次，包括0次
+
表示匹配前面的子表达式一次或多次，不包括0次
?
匹配前面的子表达式0次或一次
贪婪限定符
表示在表达式匹配成功的前提下，尽可能多的匹配
常见的贪婪限定符
.*
惰性限定符
也叫非贪婪限定符，表示在表达式匹配成功的前提下，尽可能少的匹配
常见的惰性限定符
.*?
元字符的转义
将元字符变成普通字符，失去其表达的特殊含义
方法
在需要转义的字符前面加反斜杠 \
正则表达式的应用
在Python里面用正则表达式，我们需要用到它内置的 re 模块
re 模块
导入方法
import re
常用方法
re.findall() 方法
可以在字符串中找到正则表达式所匹配的所有内容，并返回一个列表，如果没有找到匹配的，则返回空列表
语法结构
re.findall(pattern, string)
pattern 表示用于匹配的正则表达式
string 表示需要匹配的字符串
re.sub() 方法
用于替换字符串中的匹配项，并返回被替换后的字符串
语法结构
re.sub(pattern, repl, string)
pattern 表示用于匹配的正则表达式
repl 表示要替换的字符串
string 表示需要匹配的字符串


自动登录网站，爬取歌曲信息
open() 函数
参数 mode
当 mode='wb' 时，open() 函数会打开一个文件只用于写入二进制的数据，如果该文件已存在则打开文件，并从开头开始编辑，即原有内容会被清空。如果该文件不存在，创建新文件后再写入
pytesseract 模块
概念：Python的第三方OCR（Optical Character Recognition）工具，可用于识别、读取图像中的内容
安装
Windows 系统：pip install pytesseract
Mac 系统：pip3 install pytesseract
导入方法：import pytesseract
image_to_string(image) 函数
功能：识别并以字符串形式返回图片里的内容
参数 image：可以为图片对象或者图片的路径
selenium 库
概念：selenium 库是一个第三方库，可以自动化地对浏览器进行操控
安装
Windows 系统：pip install selenium
Mac 系统：pip3 install selenium
webdriver 模块
概念：提供了自动化控制 Web 浏览器的各种语言调用接口，可通过各种浏览器的驱动程序来控制浏览器
导入方法：from selenium import webdriver
Options() 类
概念：用于设置驱动程序的相关配置项
导入方法：from selenium.webdriver.chrome.options import Options
设置静默模式
opts = Options()
opts.headless = True
Chrome(options) 类
功能：初始化谷歌浏览器驱动，返回一个 WebDriver 对象
参数 options：设置配置项
WebDriver 对象
get(url) 方法
功能：打开网页
参数 url：网页的访问链接
page_source 属性
功能：获取网页的源代码
get_cookies() 方法
功能：获取网站的 cookies 信息，以列表的形式返回
quit() 方法
功能：关闭浏览器
find_element_by_name(name) 方法
功能：根据属性 name 的值查找第一个匹配的元素，并返回一个 WebElement 对象
参数 name：属性 name 的值
find_element_by_id(id_) 方法
功能：根据属性 id 的值查找第一个匹配的元素，并返回一个 WebElement 对象
参数 id_：属性 id 的值
find_element_by_class_name(name) 方法
功能：根据属性 class 的值查找第一个匹配的元素，并返回一个 WebElement 对象
参数 name：属性 class 的值
WebElement 对象
send_keys() 方法
功能：输入内容到指定元素位置
click() 方法
功能：可用于点击某一元素
screenshot_as_png 属性
功能：截取指定元素的内容