```
#通过driver.find_elements_by_tag_name定位页面元素，输入用户名
driver.find_elements_by_tag_name("input")[1].send_keys("123456")
#通过driver.find_elements_by_tag_name定位页面元素，输入密码
```

任务 3.4 鼠标模拟操作方法使用

任务介绍

用Selenium进行自动化，有时候会遇到需要模拟鼠标操作才能进行的情况，比如单击、双击、右击、拖动等。而Selenium提供一个类来处理这类事件——ActionChains。本任务针对鼠标模拟操作进行介绍。

任务目标

掌握鼠标模拟操作的使用方法。

知识储备

ActionChains基本能够满足所有对鼠标操作的需求。使用此类中的方法时，需要先引入此类，引入代码为：

```
selenium.webdriver.common.action_chains.ActionChains(driver)
```

需要了解ActionChains的执行原理，当调用ActionChains( )方法时，不会立即执行，而是会将所有的操作按顺序存放在一个队列里，当调用perform( )方法时，按照队列里面的顺序进行执行。其中调用的perform( )方法必须放在ActionChains方法最后。

这种情况下可以有两种调用方法：

1. 链式写法

```
menu=driver.find_element_by_css_selector(".nav")
hidden_submenu=driver.find_element_by_css_selector(".nav #submenu1")
ActionChains(driver).move_to_element(menu).click(hidden_submenu).perform()
```

2. 分步写法

```
menu=driver.find_element_by_css_selector(".nav")
hidden_submenu=driver.find_element_by_css_selector(".nav #submenu1")
actions=ActionChains(driver)
actions.move_to_element(menu)
actions.click(hidden_submenu)
actions.perform()
```

注意：两种写法本质是一样的，ActionChains都会按照顺序执行所有的操作。

ActionChains方法列表如表2–3–4–1所示。

表 2-3-4-1

方　法	描　述
click(on_element=None)	单击鼠标左键
click_and_hold(on_element=None)	单击鼠标左键，不松开
context_click(on_element=None)	单击鼠标右键
double_click(on_element=None)	双击鼠标左键
drag_and_drop(source, target)	拖动到某个元素然后松开
drag_and_drop_by_offset(source,xoffset,yoffset)	拖动到某个坐标然后松开
key_down(value, element=None)	按下某个键盘上的键
key_up(value, element=None)	松开某个键
move_by_offset(xoffset, yoffset)	鼠标指针从当前位置移动到某个坐标
move_to_element(to_element)	鼠标指针移动到某个元素
move_to_element_with_offset(to_element,xoffset,yoffset)	移动到距某个元素（左上角坐标）多少距离的位置
perform()	执行链中的所有动作
release(on_element=None)	在某个元素位置松开鼠标左键
send_keys(*keys_to_send)	发送某个键到当前焦点的元素
send_keys_to_element(element, *keys_to_send)	发送某个键到指定元素

任务实施

实例：

（1）进入人力资源综合服务系统登录页面；

（2）在用户名输入框中输入信息；

（3）双击信息；

（4）右击信息；

（5）输入密码，然后单击“登录”按钮；

（6）将鼠标指针悬停在系统首页右上角用户名处。

在PyCharm中进行代码编写：

```
import time
from selenium import webdriver
from selenium.webdriver import ActionChains
driver=webdriver.Chrome()
driver.get("http://192.168.X.XXX/suthr/logon")
#进入人力资源综合服务系统登录页面
driver.find_element_by_name("username").send_keys("hrteacher")
#输入用户名
```

```
time.sleep(5)
a=driver.find_element_by_name("username")
ActionChains(driver).double_click(a).perform()
#双击操作
time.sleep(5)
ActionChains(driver).context_click(a).perform()
#右击操作
time.sleep(5)
driver.find_element_by_name("password").send_keys("123456")
#输入密码
time.sleep(5)
b=driver.find_element_by_class_name("uppercase")
ActionChains(driver).click(b).perform()
#单击操作
time.sleep(5)
c=driver.find_element_by_xpath("/html/body/div[1]/div/div[2]/div[2]/ul/li[7]/a/span")
ActionChains(driver).move_to_element(c).perform()
#悬停操作
```

任务 3.5 键盘模拟操作方法使用

任务介绍

用Selenium进行自动化，有时候会遇到需要用到模拟键盘操作的情况，而Selenium提供了一个类来处理这类事件——Keys。本任务针对键盘模拟操作进行介绍。

任务目标

掌握键盘模拟操作的使用方法。

知识储备

Keys基本能够满足对键盘基本操作的需求。模拟键盘的操作需要先引入键盘模块，引入脚本为：

```
from selenium.webdriver.common.keys import Keys
```

在使用Keys方法时，可以分为两大类：

（1）先使用ActionChains类将鼠标移动到需要进行键盘操作的位置，然后进行键盘操作，如以下代码：

```
ActionChains(driver).send_keys(Keys.TAB).send_keys(Keys.ENTER).perform()
#对定位到的元素进行回车操作
```

（2）先通过元素定位方式进行元素定位，然后通过send_keys()进行键盘操作，如以下代码：

```
driver.find_element_by_id("kw").send_keys(Keys.ENTER)
#模拟Enter键操作回车按钮
```

Keys方法列表如表2-3-5-1所示。

表 2-3-5-1

方　法	描　述	方　法	描　述
回车键	Keys.ENTER	剪切 (Ctrl+X)	Keys.CONTROL,'x'
删除键	Keys.BACK_SPACE	粘贴 (Ctrl+V)	Keys.CONTROL,'v'
空格键	Keys.SPACE	【F1】键	send_keys(Keys.F1)
【Tab】键	Keys.TAB	【F2】键	send_keys(Keys.F2)
回退键	Keys.ESCAPE	【F3】键	send_keys(Keys.F3)
刷新键	Keys.F5	【F4】键	send_keys(Keys.F4)
【Shift】键	Keys.SHIFT	【F5】键	send_keys(Keys.F5)
【Esc】键	Keys.ESCAPE	【F6】键	send_keys(Keys.F6)
上键	Keys.ARROW_UP	【F7】键	send_keys(Keys.F7)
下键	Keys.ARROW_DOWN	【F8】键	send_keys(Keys.F8)
左键	Keys.ARROW_LEFT	【F9】键	send_keys(Keys.F9)
右键	Keys.ARROW_RIGHT	【F10】键	send_keys(Keys.F10)
【=】键	EQUALS	【F11】键	send_keys(Keys.F11)
全选 (Ctrl+A)	Keys.CONTROL,'a'	【F12】键	send_keys(Keys.F12)
复制 (Ctrl+C)	Keys.CONTROL,'c'	—	—

任务实施

实例：

（1）进入人力资源综合服务系统登录页面；

（2）输入用户名和密码，单击“登录”按钮；

（3）单击系统页面中的“培训进修”按钮；

（4）在培训内容输入框中输入信息dd；

（5）全选信息；

（6）复制信息；

（7）粘贴信息两次；

（8）通过鼠标的形式单击“查询”按钮。

在PyCharm中进行代码编写：

```
    import time
    from selenium import webdriver
    from selenium.webdriver import ActionChains
    from selenium.webdriver.common.keys import Keys
    driver=webdriver.Chrome()
    driver.get("http://192.168.X.XXX/suthr/logon")
    #进入人力资源综合服务系统登录页面
    driver.find_element_by_name("username").send_keys("hrteacher")
    #输入用户名
    driver.find_element_by_name("password").send_keys("123456")
    #输入密码
    driver.find_element_by_class_name("uppercase").click()
    #单击“登录”按钮
    driver.find_element_by_link_text("培训进修").click()
    #单击“培训进修”按钮
    driver.find_element_by_id("content").send_keys("dd")
    #在培训内容输入框输入信息
    time.sleep(3)
    driver.find_element_by_id("content").send_keys(Keys.CONTROL,'a')
    time.sleep(3)
    #全选信息
    driver.find_element_by_id("content").send_keys(Keys.CONTROL,'c')
    time.sleep(3)
    #复制信息
    driver.find_element_by_id("content").send_keys(Keys.CONTROL,'v')
    time.sleep(3)
    #粘贴信息
    driver.find_element_by_id("content").send_keys(Keys.CONTROL,'v')
    time.sleep(3)
    #粘贴信息
    ActionChains(driver).send_keys(Keys.TAB).send_keys(Keys.TAB).send_keys(Keys.
TAB).send_keys(Keys.ENTER).perform()
    #单击“查询”按钮
```

任务 3.6 时间等待处理方法使用

任务介绍

测试过程中，会发现脚本执行的时候展现出来的效果都是很快结束，此时可以增加一个等待时间来观察执行效果。这种等待时间只是为了便于观察，这种情况下是否包含等待时间不会影响执行结果，但是有一种情况会直接影响执行结果。在打开一个网站的时候，由于环境的因素导致页面没有加载完成，

此时去定位元素无法找到元素，这种情况下增加等待时间就会显得万分重要。本任务针对时间等待处理进行介绍。

任务目标

掌握时间等待处理的使用方法。

知识储备

Selenium主要提供WebDriverWait和Implicit Wait两种模式的等待，但是Python time模块也提供一种非智能的sleep( )等待，使用这个设置以后必须强制等待设置的时间，只有等待时间结束以后才会继续执行。这种模式一般会用于观察执行效果的时候，而WebDriverWait和Implicit Wait这两种时间等待属于智能等待。

1. 显式等待 WebDriverWait( )

显式等待指等待页面加载完成，找到某个条件发生后再继续执行后续代码，如果超过设置时间检测不到则抛出异常。使用WebDriverWait首先需要导入此模块。代码如下：

```
from selenium.webdriver.support.ui import WebDriverWait
```

使用格式：

```
WebDriverWait(driver,timeout,poll_frequency=0.5,ignored_exceptions=None)
```

- driver：WebDriver的驱动程序（Ie、Firefox、Chrome）；
- timeout：最长超时时间，默认以秒（s）为单位；
- poll_frequency：休眠时间的间隔（步长）时间，默认为0.5 s；
- ignored_exceptions：超时后的异常信息，默认情况下抛出NoSuchElementException异常。

例如：

```
element=WebDriverWait(driver,10).until(lambda x: x.find_element_by_id("someId"))
is_disappeared=WebDriverWait(driver,30,1,(ElementNotVisibleException)).until_
not(lambda x: x.find_element_by_id("someId").is_displayed())
```

注意：until是固定格式，可以理解为直到元素定位到为止；lambda x:x是一个匿名函数构建的方法，这里不太好理解，可以理解为固定格式，最后接定位方法。

WebDriverWait( )一般由unit( )或until_not( )方法配合使用。

- until(method, message=' ')调用该方法提供的驱动程序作为一个参数，直到返回值不为False；
- until_not(method, message=' ')调用该方法提供的驱动程序作为一个参数，直到返回值为False。

2. 强制等待 sleep( )

设置等待最简单的方法就是强制等待，也就是sleep( )方法。它可以让程序暂停运行一定时间，时间过后继续运行。其缺点是不智能，如果设置的时间太短，元素还没有加载出来，则照样会报错；如果设置的时间太长，则会浪费时间。如果代码量过大，多个强制等待会影响整体的运行速度。

使用强制等待sleep()，需要导入sleep。代码如下：

```
from time import sleep
```

使用格式：

```
sleep()
```

3. 隐性等待 implicitly_wait()

隐性等待就是设置一个等待时间范围，这个等待时间是不固定的，最长的等待时间是设置的最大值。隐性等待也称智能等待，是Selenium提供的一个超时等待。它等待一个元素被发现，或一个命令完成，如果超出设置时间则抛出异常。

使用格式：

```
driver.implicitly_wait()
```

任务实施

实例1：

（1）进入人力资源综合服务系统登录页面；

（2）输入用户名和密码；

（3）单击“登录”按钮，设置时间等待。

在PyCharm中进行代码编写：

```
from selenium import webdriver
from selenium.webdriver.support.wait import WebDriverWait
driver=webdriver.Chrome()
driver.get("http://192.168.X.XXX/suthr/logon")
#进入人力资源综合服务系统登录页面
driver.find_element_by_name("username").send_keys("hrteacher")
#输入用户名
driver.find_element_by_name("password").send_keys("123456")
#输入密码
element=WebDriverWait(driver,10).until(lambda x:x.find_element_by_id("loginBtn"))
#定位“登录”按钮并设置时间等待
element.click()
#单击“登录”按钮
```

实例2：

（1）进入人力资源综合服务系统登录页面；

（2）输入用户名和密码，设置时间等待；

（3）单击“登录”按钮。

在PyCharm中进行代码编写：

```
from time import sleep
from selenium import webdriver
driver=webdriver.Chrome()
driver.get("http://192.168.X.XXX/suthr/logon")
#进入人力资源综合服务系统登录页面
driver.find_element_by_name("username").send_keys("hrteacher")
#输入用户名
```

```
driver.find_element_by_name("password").send_keys("123456")
#输入密码
sleep(5)
#时间等待
driver.find_element_by_class_name("uppercase").click()
#单击“登录”按钮
```

实例3：

（1）进入人力资源综合服务系统登录页面，设置时间等待；

（2）输入用户名和密码；

（3）单击“登录”按钮。

在PyCharm中进行代码编写：

```
from selenium import webdriver
driver=webdriver.Chrome()
driver.get("http://192.168.X.XXX/suthr/logon")
#进入人力资源综合服务系统登录页面
driver.implicitly_wait(30)
#时间等待
driver.find_element_by_name("username").send_keys("hrteacher")
#输入用户名
driver.find_element_by_name("password").send_keys("123456")
#输入密码
driver.find_element_by_class_name("uppercase").click()
#单击“登录”按钮
```

任务 3.7 窗口切换方法使用

任务介绍

有些页面的链接打开后会重新打开一个窗口，想在新页面上操作就得先切换窗口。获取窗口的唯一标识用句柄表示，所以只需要切换句柄，就能在多个页面上灵活操作。本任务针对窗口切换进行介绍。

任务目标

掌握窗口切换的使用方法。

视频
窗口切换

知识储备

以人力资源综合服务系统为例，单击门户首页，在门户首页中单击“论坛”按钮，发现右侧多出一个窗口标签，如图2-3-7-1所示。

图 2-3-7-1

从图2–3–7–1可以看出，当需要单击新弹出窗口标签页中的某个元素时，如果人为操作，可以通过眼睛看，识别不同的窗口单击切换。但是脚本不知道用户要操作哪个窗口，所以需要先进行窗口切换，定位到此标签页中，然后再进行操作，在这里用driver.switch_to.window( )方法进行窗口之间的任意切换。

元素有属性，浏览器的窗口也有属性，浏览器窗口的属性可以用脚本（handle）来识别。

对于初学者来说，可以将窗口切换分为四步进行。

（1）获取第一个窗口的名字，代码为：

```
print(driver.current_window_handle)
```

（2）获取所有窗口的名字，代码为：

```
print(driver.window_handles)
```

（3）获取到第二个窗口的名字，代码为：

```
print(driver.window_handles[1])
```

（4）进行窗口切换，代码为：

```
driver.switch_to.window(driver.window_handles[1])
```

任务实施

实例：

（1）进入人力资源综合服务系统登录页面；

（2）输入用户名和密码，单击“登录”按钮；

（3）单击页面上面的“人资工作台”按钮；

（4）单击页面左侧的“论坛后台管理”按钮；

（5）单击“举报处理”按钮；

（6）单击“回帖举报”按钮；

（7）单击回帖举报页面中的“举报流水”按钮；

（8）关闭新弹出的标签页。

在PyCharm中进行代码编写：

```
import time
from selenium import webdriver
driver=webdriver.Chrome()
driver.get("http://192.168.X.XXX/suthr/logon")
#进入人力资源综合服务系统登录页面
driver.implicitly_wait(30)
driver.find_element_by_name("username").send_keys("hrteacher")
#输入用户名
driver.find_element_by_name("password").send_keys("123456")
#输入密码
driver.find_element_by_class_name("uppercase").click()
#单击“登录”按钮
driver.find_element_by_link_text("人资工作台").click()
#单击“人资工作台”按钮
driver.find_element_by_link_text("论坛后台管理").click()
#单击“论坛后台管理”按钮
driver.find_element_by_link_text("举报处理").click()
#单击“举报处理”按钮
driver.find_element_by_link_text("回帖举报").click()
#单击“回帖举报”按钮
driver.find_element_by_xpath("/html/body/div[4]/div[2]/div/div[2]/div/div/div[2]/
div[2]/table/tbody/tr[1]/td[10]/a[1]").click()
#单击“举报流水”按钮
print(driver.current_window_handle)
#获取第一个窗口的名字
print(driver.window_handles)
#获取所有窗口的名字
print(driver.window_handles[1])
#获取到第二个窗口的名字,Window_handle[1]用到数组的原理
driver.switch_to.window(driver.window_handles[1])
#进行窗口切换
time.sleep(5)
driver.close()
```

任务 3.8 页面元素属性删除方法使用

任务介绍

在操作页面时，经常会遇到单击某个超链接、弹出新的窗口，当需要对这个新弹出窗口中的某个元素进行单击操作时，可以使用窗口切换进行解决。如果不想进行窗口切换，还想单击新弹出窗口中的元