**IN eclipse, python codes can be commented by ctrl + 3 (comment) and ctrl +shift + 3(uncomment)**

**A complete list of key bindings in eclipse ide gets displayed when u type Ctrl+shift+L**

**Add comment block – Ctrl +4**

**Remove comment block – Ctrl + 5**

browser.implicitly\_wait(10) #wait 10 seconds when doing a find\_element before carrying on

find\_element\_by\_id

find\_element\_by\_name

find\_element\_by\_xpath

find\_element\_by\_link\_text

find\_element\_by\_partial\_link\_text

find\_element\_by\_tag\_name

find\_element\_by\_class\_name

find\_element\_by\_css\_selector

f I were you I would try to find the Python equivalents of findElements() and isDisplayed() that are available in the Java bindings.

For example, I would do something similar to this:

// ...  
myElementList = driver.findElements(By.Id("fancybox-close"))  
if (myElementList.isEmpty()) {  
    // The element doesn't exist. findElements, in plural, returns a list of the matching elements, or an empty list if no one is found  
else {  
    // We know it exists, now we need to know if it's displayed (visible) or not  
    if (myElementList[0].isDisplayed()) {  
        // This means the element is visible  
    else {  
        // ...  
    }  
  
}

isTextPresent equivalent way of asserting in webdriver

There's no isTextPresent in Selenium 2 WebDriver.

What is the correct way to assert the existence of some text on a page with WebDriver?

**AssertEquals(driver.getPageSource().contains(text), true);**

**AssertEquals(getMyWebElement().getText(), "Expected text");**

Alerts and prompts

Alert alert = driver.switch\_to\_alert()

Alert.accept()

Example

Python Webdriver Script:

from selenium import webdriver  
  
browser = webdriver.Firefox()  
browser.get("http://sandbox.dev/alert.html")  
alert = browser.switch\_to\_alert()  
alert.accept()  
browser.close()

other one is alert.dismiss()

If you are not sure there will be an alert then you need to catch the error with something like this.

from selenium import webdriver  
  
browser = webdriver.Firefox()  
browser.get("http://sandbox.dev/no-alert.html")  
  
try:  
    alert = browser.switch\_to\_alert()  
    alert.accept()  
except:  
    print "no alert to accept"  
browser.close()

If you need to check the text of the alert, you can get the text of the alert by accessing the text attribute of the alert object:

from selenium import webdriver  
  
browser = webdriver.Firefox()  
browser.get("http://sandbox.dev/alert.html")  
  
try:  
    alert = browser.switch\_to\_alert()  
    **print alert.text**  
    alert.accept()  
except:  
    print "no alert to accept"  
browser.close()

first driver focus has to be  in image then call getAttribute()  
  
try this,  
  
driver.findElement(By.xpath("//div[@class='type4gallery']/div[2]/ul/li/a/img")).getAttribute("src");

**Webdriver Pop-up window handling**

driver.switch\_to\_window(window name)

if the name is unknown use

driver.get\_window\_handles()

to get the list of known windows. You may pass the handle to driver.switch\_to\_window()

**In python:**

if you are using python and webdriver   
    def close\_all\_popups(driver):  
        driver.window\_handles  
        for h in driver.window\_handles[1:]:  
            driver.switch\_to\_window(h)  
            driver.close()  
       driver.switch\_to\_window(driver.window\_handles[0])  
 it will close except parent window. dri

driver.switch\_to\_window("windowName")

All calls to driver will now be interpreted as being directed to the particular window. But how do you know the window’s name? Take a look at the javascript or link that opened it:

<a href="somewhere.html" target="windowName">Click here to open a new window</a>

Alternatively, you can pass a “window handle” to the “switchTo().window()” method. Knowing this, it’s possible to iterate over every open window like so:

for handle in driver.window\_handles:

driver.switch\_to\_window(handle)

**How to switch control to pop-up window ?**

If you want to do any operations in pop-up window you need to switch the control to pop-up window then do all your operations in that and finally close the pop-up window and again select the default (main ) window.  
  
  
here is WebDriver logic to select Pop-up window  
  
  
**1 . Pop-up window has name/id**  
  
     driver.switchTo().window("<window name>");  
  
  
**2. Pop-up window doesn't have name / you don't want to hard code the window name then go for below logic.**

* before opening pop-up get the main window handle.

             String mainWindowHandle=driver.getWindowHandle();

* open the pop-up (click on element which causes open a new window)

             webElement.click();

* try to get all available open window handles with below command. (the below command returns all window handles as Set)

            Set s = driver.getWindowHandles();

* from that above set try get newly opened window and switch the control to that (pop-up window handle), as we already know the mainWindowHandle.

                        Set s = driver.getWindowHandles();

Iterator ite = s.iterator();

while(ite.hasNext())

                       {

String popupHandle=ite.next().toString();

if(!popupHandle.contains(mainWindowHandle))

{

driver.switchTo().window(popupHandle);

}

}

* Now control is in pop-up window, do all your operations in pop-up and close the pop-up window.
* Select the default window again.

                 driver.switchTo().window( mainWindowHandle );

**To capture Screen-shot in webdriver**

Use driver.save\_screenshot('/path/to/file') or driver.get\_screenshot\_as\_file('/path/to/file'):

**Assert text on webpage**

driver.find\_element\_by\_tag\_name("body")).Text.Contains(textToFind);

*self*.assertIn(*"classifieds"*,driver.find\_element\_by\_id(*"welcome\_text"*).text)

**Ex 1**

*self*.assertIn(*"Python"*, driver.title)

ex 2  
element = browser.find\_element\_by\_tag\_name('h1')  
assert element.text == 'Example Domains'

**Ex 3**

element = driver.find\_element\_by\_tag\_name(*"h1"*)

print element.text

# mytext = element.text

if (*"Example Domains"* in element.text):

print *"The searched text is present"*

else:

print *"Searched text not present"*

*self*.assertEqual(driver.title, *"Selenium: Beginners Guide"*)

WebDriver doesn't have verify/assert methods per say. Assertions are performed in the test itself. If you take a look at Corey's answer, he performs an "assert" on an element's text. If the element's text is not 'Example Domains' an AssertionError will be raised, effectively failing your test. But what about a verify? Well as mentioned, WebDriver doesn't have one. But you could still perform something equivalent by doing a comparison

if element.text != u'Example Domains':  
    print "Verify Failed: element text is not %r" % element.text

Selecting element from drop down list (now not just select command), e.g.:

|  |  |
| --- | --- |
| 1 | el = driver.find\_element\_by\_id('id\_line') |
| 2 | for option in el.find\_elements\_by\_tag\_name('option'): | |

|  |  |  |
| --- | --- | --- |
| 3 | if option.text == "line to select": | |
| 4 | option.click() |

|  |  |
| --- | --- |
| 5 | break |

# handles1 = driver.window\_handles

# print handles1

# driver.find\_element\_by\_link\_text("this link").click()

# handles2 = driver.window\_handles

# print handles2

#

# driver.switch\_to\_window(handles2[1])

# #time.sleep(5)

# self.driver.implicitly\_wait(5)

# page\_source = driver.page\_source

# print page\_source

# self.assertIn("This is an example of a popup window.", page\_source)

# driver.switch\_to\_window(handles2[0])

**To find value of attribute**

IWebElement element = \_driver.FindElementById("InputField");  
string value = element.GetAttribute("value");

In python eachinput.**get\_attribute**("value")