Select value from a dropdown

**from** **selenium.webdriver.support.ui** **import** Select

select = Select(driver.find\_element\_by\_name('name'))

select.select\_by\_index(index)

select.select\_by\_visible\_text("text")  
select.select\_by\_value(value)

Add hover effect

**from** **selenium.webdriver** **import** ActionChains

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()

Drag & Drop

**from** **selenium.webdriver** **import** ActionChains

element = driver.find\_element\_by\_name("source")

target = driver.find\_element\_by\_name("target")

action\_chains = ActionChains(driver)  
action\_chains.drag\_and\_drop(element, target).perform()

Hover effect & drop down sample code

**import** time

**from** **selenium** **import** webdriver

**from** **selenium.webdriver** **import** ActionChains

driver = webdriver.Chrome()

driver.maximize\_window()

driver.get("https://opensource-demo.orangehrmlive.com/")

user\_name = driver.find\_element\_by\_id("txtUsername")

user\_name.send\_keys("admin")

password = driver.find\_element\_by\_id("txtPassword")

password.send\_keys("admin123")

driver.find\_element\_by\_id("btnLogin").click()

time.sleep(3)

admin\_menu = driver.find\_element\_by\_xpath("//\*[@id='menu\_admin\_viewAdminModule']/b")

submenu = driver.find\_element\_by\_xpath("//\*[@id='menu\_admin\_UserManagement']")

inner\_submenu = driver.find\_element\_by\_id("menu\_admin\_viewSystemUsers")

ActionChains(driver).move\_to\_element(admin\_menu).move\_to\_element(submenu).click(inner\_submenu).perform()

time.sleep(5)

print driver.title

driver.find\_element\_by\_id("btnAdd").click()

role\_dropdown = Select(driver.find\_element\_by\_id("systemUser\_userType"))

role\_dropdown.select\_by\_index(0)

time.sleep(5)

status\_dropdown = Select(driver.find\_element\_by\_id("systemUser\_status"))

# status\_dropdown.select\_by\_value("0")

status\_dropdown.select\_by\_visible\_text("Disabled")

time.sleep(5)

driver.quit()

Unit test framework in Python

The unittest unit testing framework was originally inspired by JUnit and has a similar flavor as major unit testing frameworks in other languages. It supports test automation, sharing of setup and shutdown code for tests, aggregation of tests into collections, and independence of the tests from the reporting framework.

The unittest module provides a rich set of tools for constructing and running tests. This section demonstrates that a small subset of the tools suffice to meet the needs of most users.

Sample unit test code

**Import** unittest

**from** **selenium** **import** webdriver

**class TestUbuntuHomepage(unittest.TestCase):**

**def setUp(self):**

**self.browser = webdriver.Firefox()**

**self.browser.maximize\_window()**

**def testTitle(self):**

**self.browser.get('http://www.ubuntu.com/')**

**print "Page Title: ", self.browser.title**

**self.assertIn('Ubuntu', self.browser.title)**

**def tearDown(self):**

**self.browser.quit()**

**if \_\_name\_\_ == '\_\_main\_\_':**

**unittest.main()**

Code explanation

A testcase is created by subclassing unittest.TestCase. The test is defined with methods whose names start with the letter test. This naming convention informs the test runner about which methods represent tests.

The crux of each test is a call to assertIn() to check for an expected result; assertTrue() or assertFalse() to verify a condition; or assertRaises() to verify that a specific exception gets raised.

The setUp() and tearDown() methods allow you to define instructions that will be executed before and after each test method.