1. Установите браузер Mozilla Firefox.

2. Скачайте драйвер для работы Selenium с Firefox: https://github.com/mozilla/geckodriver/releases/tag/v0.26.0

3. Извлеките из архива exe файл в папку с интерператором python.exe.

4. Создайте новый проект типа «Pure Python», установите пакет selenium.

5. Будем тестировать сайт python.org. Зайдите на этот сайт, нажмите правой кнопкой мыши по строке поиска, выберите пункт "просмотреть код", убедитесь, что у этого элемента name="q".

6. Введите в строку поиска что-нибудь несуществующее, например, «chupakabra», убедитесь в появлении сообщения об отсутствии результатов, затем введите что-нибудь несуществующее, например, «pycon», убедитесь в появлении результатов поиска.

7. Создайте Python unit test для проверки работы Selenium, драйвера и Firefox на проделанном вручную сценарии:

import time

import unittest

from selenium import webdriver

from selenium.webdriver.common.keys import Keys

class PythonOrgSearch(unittest.TestCase):

def setUp(self):

# запуск Firefox при начале каждого теста

self.driver = webdriver.Firefox()

def test\_search\_in\_python\_org(self):

driver = self.driver

# открытие в Firefox страницы http://www.python.org

driver.get("http://www.python.org")

# проверка наличия слова Python в заголовке страницы

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

# ждем 5 секунд

time.sleep(5)

# получение элемента страницы с именем q (строка поиска)

# (откройте вручную в любом браузере сайт http://www.python.org,

# нажмите правой кнопкой мыши по строке поиска,

# выберите пункт "просмотреть код",

# убедитесь, что у этого элемента name="q")

elem = driver.find\_element\_by\_name("q")

# ждем 5 секунд

time.sleep(5)

# набор слова chupakabra в найденном элементе

elem.send\_keys("chupakabra")

# ждем 5 секунд

time.sleep(5)

# нажатие кнопки Enter в найденном элементе

elem.send\_keys(Keys.RETURN)

# ждем 5 секунд

time.sleep(5)

# проверка наличия строки "No results found."

# на странице с результатами поиска

self.assertIn("No results found.", driver.page\_source)

# ждем 5 секунд

time.sleep(5)

# получение элемента страницы с именем q

# на обновленной странице

elem = driver.find\_element\_by\_name("q")

# очищаем строку поиска

elem.clear()

# ждем 5 секунд

time.sleep(5)

# набор слова pycon в найденном элементе

elem.send\_keys("pycon")

# ждем 5 секунд

time.sleep(5)

# нажатие кнопки Enter в найденном элементе

elem.send\_keys(Keys.RETURN)

# ждем 5 секунд

time.sleep(5)

# проверка отсутствия строки "No results found."

# на странице с результатами поиска

self.assertNotIn("No results found.", driver.page\_source)

def tearDown(self):

# закрытие браузера при окончании каждого теста

self.driver.close()

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

unittest.main()

8. Запустите тест, убедитесь в его успешном выполнении. Учитывайте, что в коде есть задержки по 5 секунд для наглядности выполнения каждого шага в тесте.

9. Зайдите на страницу https://www.python.org/psf-landing/, зарегистрируйтесь не через социальные сети (“Sign In” – “Sign Up / Register”), перейдите по ссылке, пришедшей по почте, нажмите кнопку “Confirm”.

10. Войдите на сайт (“Sign In” – “Sign In”), убедитесь в появлении кнопки “Your account”, выйдите с сайта (“Your account” – “Sign Out”, кнопка “ Sign Out ”).

11. Протестируйте вход и выход автоматически с помощью нового теста внутри существующего класса PythonOrgSearch:

def test\_login\_logout(self):

driver = self.driver

# открытие в Firefox страницы http://www.python.org/psf-landing/

# на которой есть кнопка Sign In

driver.get("https://www.python.org/psf-landing/")

# ждем 5 секунд

time.sleep(5)

# поиск ссылки с текстом "Sign In"

elem = driver.find\_element\_by\_link\_text("Sign In")

# нажатие на ссылку

elem.click()

# ждем 5 секунд

time.sleep(5)

# поиск текстового поля для ввода логина по XPath

# (тег input с name='login')

elem = driver.find\_element\_by\_xpath("//input[@name='login']")

# ввод логина

elem.send\_keys("**ВАШ ЛОГИН**")

# ждем 5 секунд

time.sleep(5)

# поиск текстового поля для ввода пароля по XPath

# (тег input с name='password')

elem = driver.find\_element\_by\_xpath("//input[@name='password']")

# ввод логина

elem.send\_keys("**ВАШ ПАРОЛЬ**")

# ждем 5 секунд

time.sleep(5)

# жмем ввод для отправки формы

elem.send\_keys(Keys.RETURN)

# ждем 5 секунд

time.sleep(5)

# проверка наличия на странице строки "Your account"

# после входа

self.assertIn("Your account", driver.page\_source)

# ждем 5 секунд

time.sleep(5)

# вывод кода страницы для отладки, потом можно будет убрать

print(driver.page\_source)

# поиск ссылки с текстом "Sign out"

elem = driver.find\_element\_by\_link\_text("Sign out")

# нажатие на ссылку

elem.click()

# ждем 5 секунд

time.sleep(5)

# поиск кнопки на форме в главной области страницы

# по CSS-селектору

elem = driver.find\_element\_by\_css\_selector(

'div.container section.main-content form button'

) # нажатие на кнопку

elem.click()

# ждем 5 секунд

time.sleep(5)

# проверка отсутствия на странице строки "Your account"

# после выхода

self.assertNotIn("Your account", driver.page\_source)

12. Запустите тест, убедитесь в возникновении ошибки при обнаружении ссылки "Sign out", нажмите на окно «Run», нажмите Ctrl + F для поиска, введите "Sign out", убедитесь, что в верстке страницы такая ссылка есть, но у ее родительского элемента некорректные атрибуты:

<li class="tier-2 element-5" **"=""** role="treeitem&quot;">

13. Таким образом, наш тест обнаружил проблему. Для продолжения тестирования до ее исправления заменим поиск и нажатие ссылки Sign out на переход по соответствующему адресу:

# заменить

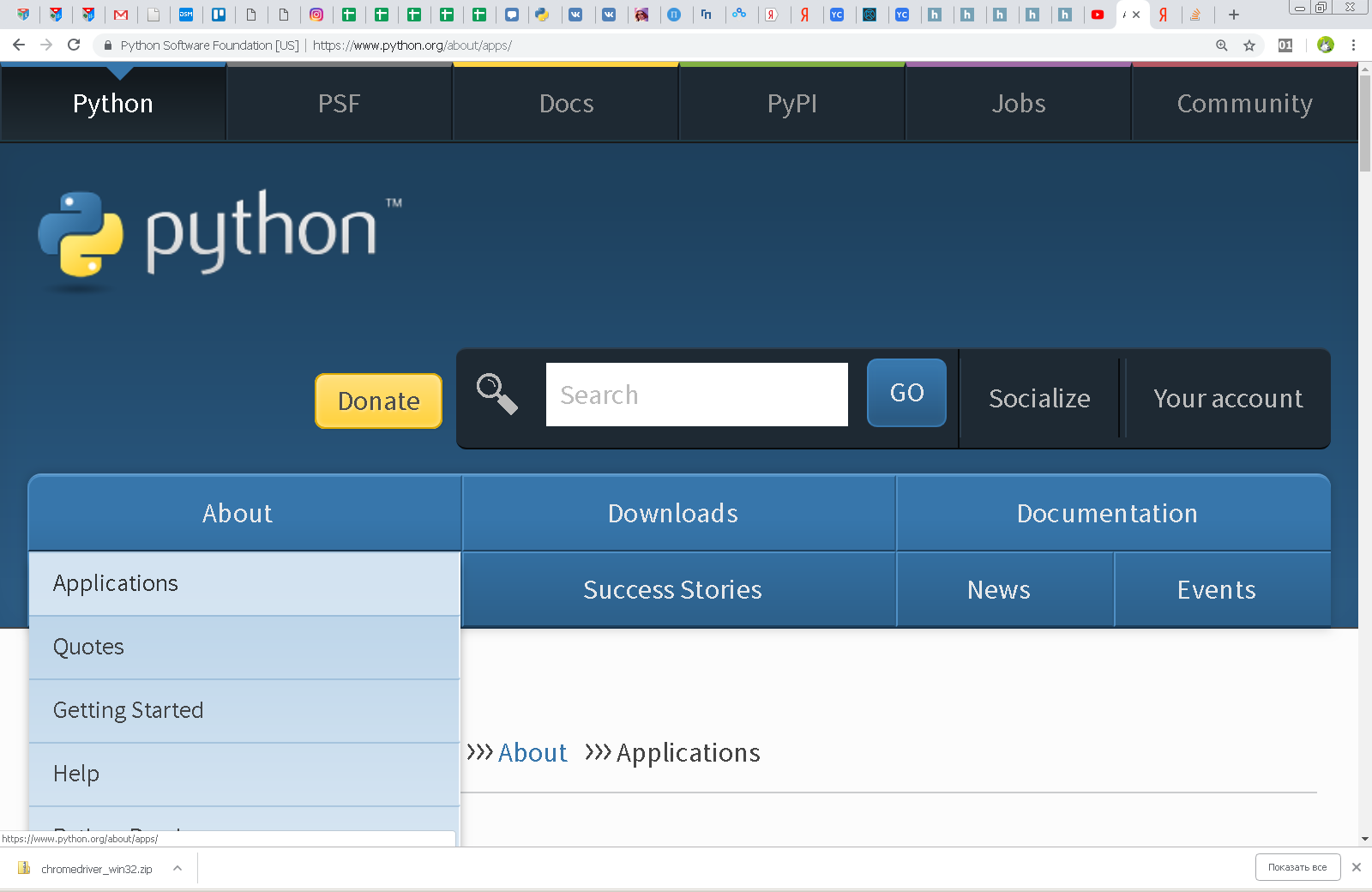
# elem = driver.find\_element\_by\_link\_text("Sign out")

# elem.click()

# на

driver.get("https://www.python.org/accounts/logout/")

14. Протестируем отображение «хлебных крошек» на страницах из выпадающего списка About – строчки, отображающей путь от корня сайта до текущей страницы через иерархию промежуточных пунктов:



15. Допишем новый тест для хлебных крошек в существующий класс PythonOrgSearch:

def test\_about\_breadcrumbs(self):

driver = self.driver

# открытие в Firefox страницы http://www.python.org

driver.get("http://www.python.org")

# получаем список ссылок в меню About по CSS-селектору

elems = driver.find\_elements\_by\_css\_selector('#about ul li a')

# перебираем полученные подпункты меню,

# выписываем названия и ссылки в отдельные списки

# потому что при переходе по ссылкам на другие страницы

# связь со списком подпунктов будет потеряна

href\_list = []

name\_list = []

for e in elems:

href\_list.append(e.get\_attribute("href"))

name\_list.append(e.get\_attribute('innerHTML'))

# перебираем полученные ссылки

for i in range(len(href\_list)):

# переходим по ссылке

driver.get(

href\_list[i]

)

# получаем строчку хлебных крошек

elem = driver.find\_element\_by\_css\_selector('.breadcrumbs')

# проверка наличия в хлебных крошках ссылки на пункт About

self.assertIn("About", elem.get\_attribute('innerHTML'))

# проверка наличия в хлебных крошках

# наличия названия текущего пункта

self.assertIn(

# название текущего пункта

name\_list[i],

# строчка хлебных крошек

elem.get\_attribute('innerHTML')

)

# ждем 5 секунд

time.sleep(3)

16. Запустите тест, убедитесь, что последний пункт Python Brochure не загружается. Проблема выявлена, для продолжения теста до ее устранения будем выполнять цикл по всем пунктам, кроме этого последнего – уменьшим число итераций во втором цикле:

# заменить

# for i in range(len(href\_list)):

# на

for i in range(len(href\_list)-1):

17. Установите браузер Google Chrome.

18. Скачайте драйвер для работы Selenium с Chrome: https://chromedriver.storage.googleapis.com/index.html?path=79.0.3945.36/

19. Извлеките из архива exe файл в папку с интерператором python.exe.

20. Создайте в проекте копию созданного ранее файла с тестами, поменяйте в копии Firefox на Chrome:

# заменить

# self.driver = webdriver. Firefox()

# на

self.driver = webdriver.Chrome()

21. Запустите тесты в Chrome, убедитесь в их успешном выполнении.