**Как автоматизировать вход в систему с помощью Selenium в Python?**

Автоматизация процесса входа на какой-либо веб-сайт, используя программу, написанную на Python, очень удобна и практична в эксплуатации. Реализовывается это благодаря библиотеке [Selenium WebDriver](https://www.selenium.dev/documentation/en/webdriver/). [Selenium WebDriver](https://www.selenium.dev/documentation/en/webdriver/) — это библиотека для управления браузером, которая поддерживает все основные браузеры и доступна для разных языков программирования, включая Python. И в данной работе я бы хотел использовать ее для автоматизированного входа на GitHub.

Первым делом, что нам необходимо сделать – это установить Selenium для Python, использую команду: pip3 install selenium. После этого, понадобится установить специфичный драйвер для браузера, которым мы хотим управлять. Лично я, буду использовать ChromeDriver, но вы можете использовать любой другой. Далее, необходимо открыть новый скрипт Python, инициализировать WebDriver и ввести свои учетные данные от GitHub:

from selenium import webdriver

from selenium.webdriver.support.ui import WebDriverWait

# Github credentials

username = "username"

password = "password"

# initialize the Chrome driver

driver = webdriver.Chrome("chromedriver")

После того, как мы загрузили и разархивировали драйвер, необходимо будет его поместить в текущий каталог, то есть, скачанный мною **chromedriver.exe** я помещаю в папку с проектом. Именно поэтому, я просто передаю его имя конструктору.

Поскольку мы заинтересованы в автоматизации входа в систему GitHub, то перейдем на страницу входа в Github и посмотрим на неё через кнопку f12 для того, чтобы найти её HTML‑элементы, а именно: идентификатор полей ввода логина и пароля, а также имя кнопки входа, необходимые для получения этих элементов в коде и их запрограммирования. Обратите внимание, что поле ввода имени пользователя/адреса электронной почты имеет id поля login\_field, где поле ввода пароля имеет id password. Также кнопка отправки имеет имя commit. И приведенный ниже код переходит на страницу входа в Github, извлекает эти элементы, заполняет учетные данные и нажимает кнопку:

# перейти на страницу входа в github

driver.get("https://github.com/login")

# найти поле имени пользователя / электронной почты и отправить само имя пользователя в поле ввода

driver.find\_element\_by\_id("login\_field").send\_keys(username)

# найти поле ввода пароля и также вставить пароль

driver.find\_element\_by\_id("password").send\_keys(password)

# нажмите кнопку входа в систему

driver.find\_element\_by\_name("commit").click()

В данном коде, функция find\_element\_by\_id() извлекает HTML‑элемент по его идентификатору, а метод send\_keys() имитирует нажатие клавиш. Приведенный выше код заставит Chrome ввести электронное письмо и пароль, а затем нажать кнопку «Sign in».

Следующее, что нужно сделать, это определить, был ли наш вход в систему успешным. Для этого есть масса способов, но здесь мы попытаемся обнаружить ошибку при входе в систему. С помощью все той же консоли, которая вызывается клавишей f12, мы увидим, что при непрвильном вводе учетных данных, появляется новый элемент div HTML с классом «flash-error», который имеет текст «**Incorrect username or password**» («Неправильное имя пользователя или пароль»). Приведенный ниже код отвечает за ожидание загрузки страницы после входа в систему с помощью WebDriverWait() и проверяет наличие ошибки:

# ждем завершения состояния готовности

WebDriverWait(driver=driver, timeout=10).until(

lambda x: x.execute\_script("return document.readyState === 'complete'")

)

error\_message = "Incorrect username or password."

# получаем ошибки (если есть)

errors = driver.find\_elements\_by\_class\_name("flash-error")

# при необходимости распечатать ошибки

# для e в ошибках:

# print(e.text)

# если мы находим это сообщение об ошибке в составе error, значит вход не выполнен

if any(error\_message in e.text for e in errors):

print("[!] Login failed")

else:

print("[+] Login successful")

Тут мы используем WebDriverWait, чтобы дождаться завершения загрузки документа, метод execute\_script() выполняет Javascript в браузере, код JS возвращает document.readyState === 'complete' возвращает True, если всё хорошо и False в противном случае.

В результате всех этих действий мы получаем очень удобную программу, которая упрощает нам жизнь. Например, мы написали какую-либо программу на Python и хотим ее выгрузить на наш GitHub. И раньше, чтобы это сделать, необходимо было заходить в браузер, переходить на сайт, вводить свои учетные данные и после этого уже загружать свой новый проект в репозиторий. Теперь же, достаточно просто запустить эту программу, и она все основные действия сделает за вас, вам останется лишь загрузить свой проект.