**Тема занятия №37: Парсинг данных. Библиотеки - beatifulsoup4 и selenium**

**1. Использование библиотеки selenium**

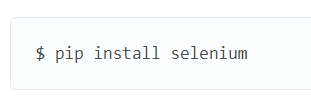
Python – один из самых популярных языков для веб-автоматизации с Selenium, поскольку в нем есть упрощенный синтаксис, который позволяет выполнять больше задач за меньшее количество строк кода! Таким образом, Python и Selenium создают идеальную комбинацию для автоматизированного тестирования в вебе.

Начнем с того, что Selenium – это открытый фреймворк, который в основном используется для автоматизации взаимодействия с веб-элементами в AUT (Application Under Test). Наряду с Python, Selenium также поддерживает Java, C#, JavaScript, Ruby и PHP. Однако, по своему опыту скажу, что Python – лучший вариант для веб-автоматизации в комбинации с Selenium.

Все больше разработчиков, как и я в свое время, осваивают Python, который считается третьим по популярности языком, если верить опросу Stack Overflow Developer Survey 2021. Поэтому шпаргалка по Selenium с Python может служить для ознакомления с полезным API для автоматизации веб-сайтов (или веб-приложений).

Если в Selenium вы новичок и вам интересно, что же это такое, мы рекомендуем ознакомиться с нашим руководством What is Selenium?

В этой статье мы разберемся с модулем в Python, который предоставляет нам функционал Selenium WebDriver. Для установки пакета Selenium WebDriver мы воспользуемся Python Package Index (PyPI). Выполните в терминале следующую команду, чтобы установить Selenium для Python:



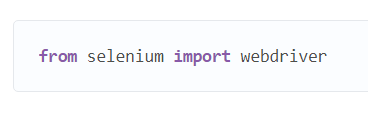
Если вы хотите запускать свои сценарии тестирования онлайн с Selenium Grid, воспользуйтесь LambdaTest для автоматизации тестирования.

Эта библиотека Python оборачивает Selenium WebDriver и предоставляет методы для автоматизации целого ряда задач, таких как заполнение форм, вход на сайт, нажатие на кнопки и многое другое. Кроме того, вы можете ознакомиться с руководством по Selenium с Python, в котором подробно рассматриваются неотъемлемые аспекты Selenium Python с точки зрения автоматизации веб-тестирования.

Команды, которые вы увидите в шпаргалке, можно использовать в качестве удобного ресурса для всех, кто играется с Selenium на Python для автоматизации веб-приложений. Если вам нужно краткое описание работы Selenium в Python, ознакомьтесь с руководством, в котором подробно рассматривается архитектура Selenium WebDriver и освещаются неотъемлемые аспекты, связанные с Selenium WebDriver в Python. Итак, давайте начнем нашу шпаргалку!

Импортируем библиотеку Selenium

Перед тем, как начать пользоваться командами Selenium, вам нужно импортировать пакет Selenium WebDriver.



Инициализация дайвера в Python

После загрузки соответствующего драйвера браузера вам необходимо запустить Selenium WebDriver и драйвер браузера (например, ChromeDriver для Chrome, GeckoDriver для Firefox и т.д.).

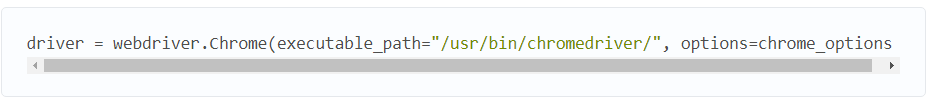


Если местоположения драйвера браузера нет в переменной PATH (или если его нет в System Path), нужно добавить следующие аргументы:

executable\_path: Путь к вашему веб-драйверу Selenium (бинарный файл)

options: Параметры, касающиеся выполнения веб-браузеров

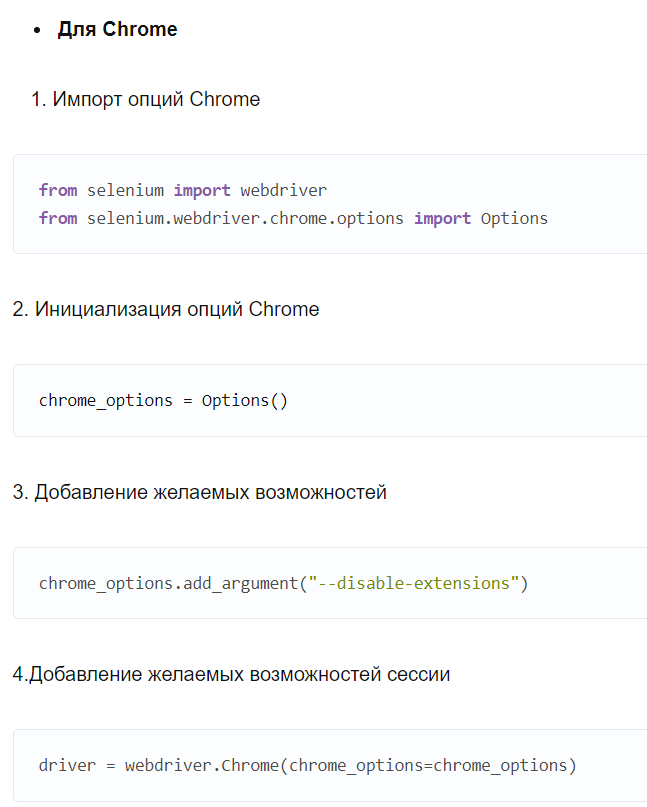
Пример:



Настройка параметров Selenium WebDriver

Класс Options в Selenium обычно используется в сочетании с желаемыми возможностями кастомизации Selenium WebDriver.

Так вы можете выполнять различные операции, такие как открытие браузера (Chrome, Firefox, Safari, IE, Edge и т.д.) в режиме увеличения, включение и отключение расширений браузера, отключение режима GPU, отключение всплывающих окон и многое другое. Поэтому важно хорошо разобраться в этом разделе шпаргалки по Selenium в Python, поскольку так вы сможете решить проблемы автоматизации, связанные с изменением свойств браузера, о которых мы говорили ранее.



Поиск элемента

В основном локаторы в Selenium (https://www.lambdatest.com/learning-hub/selenium-locators) используются для определения местоположения веб-элементов из DOM. Соответствующие взаимодействия (или действия) в дальнейшем выполняются относительно найденных веб-элементов. Несколько популярных локаторов в Selenium - ID, Name, Link Text, Partial Link Text, CSS Selectors, XPath, TagName и т.д.

Поиск элемента по атрибуту ID

В этом методе поиск элемента в DOM производится по ID. ID уникален для каждого элемента на странице. Таким образом, с помощью ID можно однозначно идентифицировать элемент. Например, ниже показано использование атрибута ID для поиска веб-элементов на странице входа в систему LambdaTest:

Так вы можете воспользоваться атрибутом ID в Selenium:



Поиск элемента по классу CSS

Элементы в HTML DOM еще можно найти по имени класса, который хранится в атрибуте класса HTML-тега. У класса может быть несколько экземпляров, но метод вернет первый элемент с соответствующим классом.



Поиск элемента по имени

Веб-элементы, такие как открывающий тег, имеют связанный с ними атрибут Name. В Selenium есть метод поиска веб-элементов с помощью атрибута NAME. Если у нескольких элементов одинаковое имя, он вернет первое совпадение.

Вот HTML-код, в котором есть элемент ввода имени Name.



Ниже показано использование метода Selenium Python для определения местоположения веб-элемента с помощью атрибута NAME:



Поиск элемента по тегу

Этот метод используется для поиска и выбора веб-элементов по HTML-тегу. Метод find\_element\_by\_tag\_name() используется для поиска таких тегов, как H1, DIV, INPUT и т.д. Если тег встречается несколько раз, он возвращает первый совпадающий.

