Создадим папку, где будут храниться наши виртуальные окружения, и перейдем в неё:

mkdir environments

cd environments

Создадим виртуальное окружение:

python -m venv selenium\_env

Запустим созданный для нас приложением venv файл activate.bat, чтобы активировать окружение:

selenium\_env\Scripts\activate.bat

Если окружение активировано, то мы увидим в начале командной строки терминала название окружения в круглых скобках. Теперь мы можем устанавливать нужные нам пакеты и запускать скрипты для тестов, которые мы напишем:



Если понадобится выйти из нашего окружения, достаточно выполнить команду deactivate:



В нашем виртуальном окружении установим библиотеку Selenium. Проверьте, что вы указали для установки именно версию 3.14.0. Проект Selenium развивается очень быстро и в других версиях могут быть расхождения в наборе методов Selenium и некоторые примеры у вас могут не работать:

pip install selenium==3.14.0

Проверим, что библиотека действительно установлена:

pip list

**Установка драйвера для браузера: Windows**

Для установки откройте сайт <https://sites.google.com/chromium.org/driver/> (старая версия сайта <https://sites.google.com/a/chromium.org/chromedriver/downloads>) и скачайте ту версию ChromeDriver, которая соответствует версии вашего браузера. Чтобы узнать версию браузера, откройте новое окно в Chrome, в поисковой строке наберите: chrome://version/ и нажмите Enter. В верхней строчке вы увидите информацию про версию браузера.

* Скачайте с сайта <https://sites.google.com/chromium.org/driver/> (старая версия сайта <https://sites.google.com/a/chromium.org/chromedriver/downloads>) драйвер для вашей версии браузера. Разархивируйте скачанный файл.
* Создайте на диске C: папку chromedriver и положите разархивированный ранее файл chromedriver.exe в папку C:\chromedriver.
* Добавьте в системную переменную PATH папку C:\chromedriver. Как это сделать в разных версиях Windows, описано здесь: <https://www.computerhope.com/issues/ch000549.htm>.

**Пример: как добавить путь в системную переменную PATH на Windows10**

1. Откройте настройки системы.

2. В настройках откройте вкладку About, затем System info:



3. Выберите Advanced system settings:



4. Выберите Environment Variables:



5. Кликните два раза на строчке Path в System variables:



6. Нажмите кнопку New. Введите в новую строку путь к ChromeDriver — C:\chromedriver. Нажмите Enter. У вас должна появится строка с указанным путем:



7. Если у вас была открыта командная строка Windows, не забудьте ее закрыть. Затем откройте новую командную строку, чтобы изменения переменной окружения стали доступны. Активируйте снова виртуальное окружение selenium\_env, которое мы создали в предыдущих шагах.

Давайте убедимся в том, что вебдрайвер установлен правильно.

Для начала проверим содержимое переменной path, для этого наберем в командной строке **Path:**



Ура, там есть папка с chromedriver! Попробуем вызвать его напрямую из командной строки:



Магия переменной path: хотя программа chromedriver находится где-то в другом каталоге, мы можем напрямую открывать её, используя имя chromedriver. Чтобы завершить процесс в консоли, нажмите Ctrl+C.

Знак на этом этапе, что пошло что-то не так:



В таком случае попробуйте перезапустить консоль, перезапустить компьютер, перепроверить и добавить заново по инструкциям папку с chromedriver в переменную path.

Установить PyCharm

В каждый новый проект устанавливать библиотеку selenium

Поиск

по id: #id, [id="..."]

по tag: h1

по значению атрибута:[...="..."]

по name: [name="..."]

по class: [class="....."], .classname

ПРИМЕР(class="lead text-muted" - Вариант [class="lead"] не найдет нам этот элемент, так как он ищет по точному совпадению. Чтобы найти элемент, нам нужно будет написать [class="lead text-muted"], порядок классов при этом важен. [class="text-muted lead"] — уже не найдет искомый элемент.

Вариант .lead при этом позволит найти данный элемент, так как он ищет простое вхождение класса в элемент. Для уточнения селектора можно задать также оба класса, для этого нужно добавить второй класс к строке поиска без пробела и предварить его точкой: .lead.text-muted. Порядок классов в отличие от первого способа здесь не важен — .text-muted.lead так же найдет нужный элемент. Рекомендуем пользоваться вторым способом поиска классов, так как он является более гибким.

составныe CSS-селекторы:

ПРИМЕР(<div id="posts" class="post-list">

<div id="post1" class="item">

<div class="title">Как я провел лето</div>

<img src="./images/summer.png">

</div>

<div id="post2" class="item">

<div class="title second">Ходили купаться</div>

<img src="./images/bad\_dog.jpg">

</div>

<div id="post3" class="item">

<div class="title">С друзьями</div>

<img src="./images/friends.jpg">

</div>

</div>)

Использование потомков

Попробуем найти элемент с текстом "Ходили купаться". Для решения этой задачи мы можем взять элемент, стоящий выше в иерархии нужного нам элемента, и написать следующий селектор:

#post2 .title

Здесь символ # означает, что надо искать элемент с id post2, пробел - что также нужно найти элемент-потомок, а ., что элемент-потомок должен иметь класс со значением title.

Элемент .title называется потомком (англ. descendant) элемента #post2. Потомок может находиться на любом уровне вложенности, все элементы с селектором .title также являются и потомками элемента #posts, хотя и расположены от него на два уровня ниже. #posts .title найдет все 3 элемента с классом title.

!Внимание. Символ пробела " " является значащим в CSS-селекторах. Это важный символ, который разделяет описание предка и потомка. Если бы мы записали селектор #post2.title без пробела, то в данном примере не было найдено ни одного элемента. Такая запись означала бы, что мы хотим найти элемент, который одновременно содержит id "post2" и класс "title". Таким образом #post2 .title и #post2.title — это разные селекторы.

Использование дочерних элементов

Другой способ найти этот элемент:

#post2 > div.title

Здесь мы указали еще тег элемента div и уточнили, что нужно взять элемент с тегом и классом: div.title, который находится строго на один уровень иерархии ниже чем элемент #post2. Для этого используется символ >.

Элемент #post2 в этом случае называется родителем (англ. parent) для элемента div.title, а элемент div.title называется дочерним элементом (англ. child) для элемента #post2. Если символа > нет, то будет выполнен поиск всех элементов div.title на любом уровне ниже первого элемента.

!Внимание. В данном случае символы пробела вокруг символа ">" не несут важного значения в отличие от предыдущего примера, и могут быть опущены. Запись #post2>div.title аналогична записи #post2 > div.title.

Использование порядкового номера дочернего элемента

Еще один способ найти этот элемент:

#posts > .item:nth-child(2) > .title

Псевдо-класс :nth-child(2) — позволяет найти второй по порядку элемент среди дочерних элементов для #posts. Затем с помощью конструкции > .title мы указываем, что нам нужен элемент .title, родителем которого является найденный ранее элемент .item.

Использование нескольких классов

Также мы можем использовать сразу несколько классов элемента, чтобы его найти. Для этого классы записываются подряд через точку: .title.second

By.ID – поиск по уникальному атрибуту id элемента;

By.CSS\_SELECTOR – поиск элементов с помощью правил на основе CSS;

By.XPATH – поиск элементов с помощью языка запросов XPath;

By.NAME – поиск по атрибуту name элемента;

By.TAG\_NAME – поиск по названию тега;

By.CLASS\_NAME – поиск по атрибуту class элемента;

By.LINK\_TEXT – поиск ссылки с указанным текстом. Текст ссылки должен быть точным совпадением;

By.PARTIAL\_LINK\_TEXT – поиск ссылки по частичному совпадению текста.

Важно. Вы можете столкнуться с ситуацией, когда на странице будет несколько элементов, подходящих под заданные вами параметры поиска. В этом случае WebDriver вернет вам только первый элемент, который встретит во время поиска по HTML. Если вам нужен не первый, а второй или следующие элементы, вам нужно либо задать более точный селектор для поиска, либо использовать методы find\_elements\_by, которые мы рассмотрим чуть позже.

browser.close() закрывает текущее окно браузера

browser.quit() закрывает все окна, вкладки, и процессы вебдрайвера, запущенные во время тестовой сессии

GIT

Установка имени и электронной почты

git config --global user.name "Your Name"

git config --global user.email "your\_email@whatever.com"

Параметры установки окончаний строк

Для пользователей Windows:

ВЫПОЛНИТЬ:

git config --global core.autocrlf true

git config --global core.safecrlf warn

Установка отображения unicode

git config --global core.quotepath off

Команды в окружении

git init (начинает следить)

git add (добавляет либо файл git add hello.html либо все файлы git add -A)

git status (проверка состояния)

Предположим, что вы отредактировали три файла (a.html, b.html, и c.html). Теперь вы хотите закоммитить все изменения, при этом чтобы изменения в a.html и b.html были одним коммитом, в то время как изменения в c.html логически не связаны с первыми двумя файлами и должны идти отдельным коммитом.

В теории, вы можете сделать следующее:

git add a.html

git add b.html

git commit -m "Changes for a and b"

git add c.html

git commit -m "Unrelated change to c"

Разделяя индексацию и коммит, вы имеете возможность с легкостью настроить, что идет в какой коммит.

Для Windows:

> selenium\_env\Scripts\activate.bat

(selenium\_env) С:\Users\user\environments> pip install pytest==5.1.1

Фиксируем пакеты в requirements.txt

Количество пакетов в нашем проекте растет, а мы тем временем все дальше уходим от учебных кусочков скриптов в сторону настоящего тестового проекта, поэтому в этом шаге давайте зафиксируем все пакеты, которые мы используем. Это стандартная практика, которая позволяет быстро переключаться в свежее виртуальное окружение, а также работать нескольким людям над одним проектом, получая одинаковые результаты.

Откройте терминал, перейдите в директорию, в которой вы работаете с автотестами, и активируйте виртуальное окружение.

После чего выполните в терминале команду:

pip freeze > requirements.txt

Эта команда сохранит все версии пакетов в специальный файл requirements.txt.

Как их оттуда достать? Попробуйте создать новое виртуальное окружение (если нужно, вернитесь в модуль 1 за инструкциями) и активировать. После чего выполните команду:

pip install -r requirements.txt

В свежем окружении все пакеты установлены одной командой!

pip3 install requests