Сбор и разметка данных

Урок 7. Selenium

Задание

1. Выберите веб-сайт, который содержит информацию, представляющую интерес для извлечения данных. Это может быть новостной сайт, платформа для электронной коммерции или любой другой сайт, который позволяет осуществлять скрейпинг (убедитесь в соблюдении условий обслуживания сайта).
2. Используя Selenium, напишите сценарий для автоматизации процесса перехода на нужную страницу сайта.
3. Определите элементы HTML, содержащие информацию, которую вы хотите извлечь (например, заголовки статей, названия продуктов, цены и т.д.).
4. Используйте BeautifulSoup для парсинга содержимого HTML и извлечения нужной информации из идентифицированных элементов.
5. Обработайте любые ошибки или исключения, которые могут возникнуть в процессе скрейпинга.
6. Протестируйте свой скрипт на различных сценариях, чтобы убедиться, что он точно извлекает нужные данные.
7. Предоставьте ваш Python-скрипт вместе с кратким отчетом (не более 1 страницы), который включает следующее: URL сайта. Укажите URL сайта,

который вы выбрали для анализа. Описание. Предоставьте краткое описание информации, которую вы хотели извлечь из сайта. Подход. Объясните подход, который вы использовали для навигации по сайту, определения соответствующих элементов и извлечения нужных данных. Трудности. Опишите все проблемы и препятствия, с которыми вы столкнулись в ходе реализации проекта, и как вы их преодолели. Результаты. Включите образец извлеченных данных в выбранном вами структурированном формате (например, CSV или JSON). Примечание: Обязательно соблюдайте условия обслуживания сайта и избегайте чрезмерного скрейпинга, который может нарушить нормальную работу сайта.

Решение

1. Сайт

Сразу заинтересовал сайт auchan.ru, т.к. на семинаре не смогли его отрыть с помощью selenium.

В результате? получилось "готовое решение" для зав.складами, снабжением, столовыми, ресторанами и для домохозяек.

Решение оформлено в виде Python-пакета, со следующей структурой:

- Основной стартовый модуль

1. main.py

- Внутренние модули пакета auchan

2. scraper
3. selen
4. db
5. tokens

Для реализации решения понадобятся следующие модули:

- undetected_chromedriver
- requests
- selenium
- BeautifulSoup

- json, csv, pymongo
- random, os, time

Установить все необходимые библиотеки можно следующим образом:

```
pip install undetected_chromedriver selenium beautifulsoup4 requests pymongo
```

Решение основано на браузере Chromium и драйвере Chromedriver. Их можно скачать и установить на 64-разрядный Linux следующими командами:

```
wget https://storage.googleapis.com/chrome-for-testing-
public/123.0.6312.86/linux64/chrome-linux64.zip
wget https://storage.googleapis.com/chrome-for-testing-
public/123.0.6312.86/linux64/chromedriver-linux64.zip
unzip *.zip
mv chrome-linux64 /opt/chrome-linux64
mv chromedriver-linux64 /opt/chrome-linux64
```

Далее скачиваем архив данного решения в удобное место и распаковываем его:

```
wget https://github.com/allseenn/api/raw/main/07.Tasks/snab.zip
unzip auchan.zip
```

Перейдя в распакованную директорию, запускаем основной стартовый модуль main.py:

```
cd snab
python main.py
```

Программа запросит ввести поисковое слово (товар):

```
Введите ключевое слово для поиска товаров: молоко
```

После ввода, программа приступить к поиску, скрейпинку, парсингу и скачиванию фотографий всех найденных позиций товара.

2. Selenium

Для того, чтобы сайт Ашана не замечал обращения к нему с помощью selenium, драйвер нужно инициализировать с помощью модуля undetected_chromedriver и передать в него набор опций, подробнее см. модуль selen.py

Столкнулся с трудностями скроллинга страницы, ни одна команда запуска скрипта прокрутки не работала.

Вышел из положения следующим образом:

- Осуществляю двойной клик по якорю. Якорем выступает слово "Уточнить" на верхней части страницы. По его атрибутам и тегу получается активировать нужный фрейм. Но, и в этом фрейме с найденными товарами, скроллинг не работает с помощью скрипта.
- Прокрутка вниз успешно работает с помощью имитации нажатия клавиш Page-Down.

Следующей проблемой служил сомоподгружаемый список товаров. С одной стороны, не используется пагинация, все товары расположены на одной странице. Но, с другой, в самом низу, некчему зацепиться в виде якоря. Т.к. при поиске подвал сайта пропадает и нижняя часть полностью генерируется JS-скриптами.

Выход из ситуации, опять нашел, на верхней части сайта. При выдаче результатов поиска по товарам, на верху выдается общее количество найденных позиций. Это количество запоминаю в переменной. Каждый раз, после очередной прокрутки, весь товар сравнивается с общим количеством. В случае совпадения всего загруженного товара и общего количества, цикл заканчивается и начинается парсинг.

3. Элементы для парсинга

Благодаря тому, что все данные находятся на одной странице, их легко парсить и нет проблем с пагинацией.

Структура html кода правильная. Каждая позиция заключена в конкретный div с логически понятным классом.

На сайте Ашана явно не предполагают о возможности преодоления барьера описанного ранее. Поэтому, классы и другие атрибуты тегов не генерируются автоматически, а соответствуют классическому англоязычному описанию.

4. BeautifulSoup

В соответствии с заданием, парсинг был реализован с помощью BeautifulSoup.

В отличии от сайта wildberries, на котором постоянно приходилось ловить исключения, в связи с неполной информацией, либо разными форматами. Французике веб-мастера проработали все до мелочей. При многократных поисках, в том числе объемных запросов, такие как рыба, молоко или кофе, ни разу не выбрасывались исключения.

5. Ошибки

Все же, возникали проблемы, например с яйцом. В результатах поиска 113 наименований яиц, а по факту 110, возможно это связанно с недавними событиями с этим продуктом и организован скрытый резерв. Но, именно тут, и не где иначе, такая ситуация. Решил ее просто, с помощью тайм-аута. Если, количество загруженных товаров не дошло, до указанной на верху цифры общего количества, то процесс завершится сам, через десяток секунд, как только перестанут прибавляться новые товары.

6. Данные

Как было упомянуто ранее, поиск был разнообразный и протестирован не только на продуктах питания. Искал бытовую технику, смартфоны, принтеры и т.д. Результат всегда на выходе такой какой ожидался.

Вся полученная информация передается между модулями программы в виде словаря со следующими полями:

- _id - идентификатор позиции
- name - наименование позиции
- price - цена позиции
- image - ссылка на изображение на локальном диске
- link - ссылка на позицию на сайте Ашана

7. Вывод

Результаты поиска сохраняются в двух форматах json и csv. Также, существует возможность экспорта в базу MongoDB, для этого достаточно задать нужные параметры в модуле tokens.py (пароли, токены и др.) и раскомментировать одну строку в стартовом файле main.py.

CSV-файл построил таким образом, что разделитель, кодировку и другие параметры можно задавать через соответствующую функцию в основном модуле main.py. По умолчанию, разделителем выступает табуляция и файл имеет расширение md. Что обеспечивает хорошую читаемость, особенно на гитхабе. И самое главное, обеспечена кликабельность: по слову Photo открывается с локального хранилища фотка товара, а при клике по Link, осуществляется переход непосредственно в интернет магазин Ашана к описанию конкретного товара.

Примеры запросов на два известных продукта:

1. Сгущенка

- CSV-файл
- JSON-файл

2. Тушенка

- CSV-файл
- JSON-файл