Департамент образования и науки города Москвы

Юго-восточный административный округ

Государственное бюджетное общеобразовательное учреждение

города Москвы «Школа № 1547»

Проект

«Создание скрипта для сбора данных с маркетплейса»

Выполнил:

Аврамов Вячеслав Сергеевич

ученик 11 «Т»

ГБОУ школа №1547

**Москва, 2022г.**

Оглавление

[Введение 3](#_Toc101388157)

[Актуальность 3](#_Toc101388158)

[Цель работы 3](#_Toc101388159)

[Задачи работы 3](#_Toc101388160)

[Процесс создания программы 4](#_Toc101388161)

[Изучение теоретической части 4](#_Toc101388162)

[Разработка алгоритма скрипта и его написание 7](#_Toc101388163)

[Итог 9](#_Toc101388164)

[Перспективы развития 9](#_Toc101388165)

# Введение

## Актуальность

Последние годы маркетплейсы демонстрируют бурный рост. Крупные и мелкие торговые компании пользуются ими для реализации своих товаров конечному потребителю. Все заинтересованы в продаже максимального количества своей продукции. Для обеспечения этого необходимо определить конкурентную цену и оптимальное качество товара. Разработка ценового предложения требует актуальной информации по всей номенклатуре в выбранной нише, т. е. мониторинга текущей ситуации. Именно для этого и необходима автоматическая система сбора информации со страниц маркетплейса, т. к. ручной сбор является очень затратным по времени и трудоресурсам.

## Цель работы

Создание скрипта, который собирает информацию с маркетплейса по нужной категории товаров.

## Задачи работы

1. Выбор языка программирования
2. Изучение предыдущих решений проблемы
3. Выбор маркетплейса
4. Изучение логики сайта маркетплейса
5. Разработка логики скрипта
6. Написание скрипта

# Процесс создания программы

Для реализации поставленной цели работа была разделена на две основные части:

* Изучение теории для формирования базовых понятий и алгоритма решения;
* Разработка программы и ее тестирование.

## Изучение теоретической части

Парсер — это специальная программа, онлайн-сервис или скрипт, собирающие данные с нужных сайтов, аккаунтов социальных сетей и других интернет-площадок, а затем преобразующие их в нужный вид.

С английского языка слово «скрипт» переводится как сценарий, из чего уже можно сделать определенные выводы. Это набор команд, то есть строк кода, которые вкупе выполняют конкретную задачу. Для ее выполнения и создаются скрипты.

1. Выбор языка

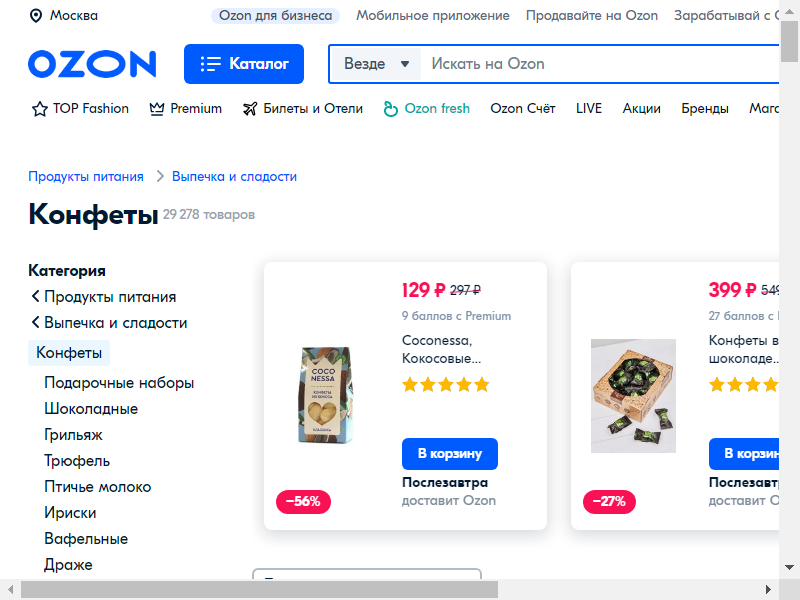
Язык программирования — [формальный язык](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA), предназначенный для записи [компьютерных программ](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0). Язык программирования определяет набор [лексических](https://ru.wikipedia.org/wiki/%D0%9B%D0%B5%D0%BA%D1%81%D0%B8%D0%BA%D0%B0), [синтаксических](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) и [семантических](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%BC%D0%B0%D0%BD%D1%82%D0%B8%D0%BA%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) правил, определяющих внешний вид программы и действия, которые выполнит исполнитель под её управлением.

Для парсинга используются PHP, Perl, Java и Python. Так как я уже до этого имел опыт работы с последним и из-за его простоты, решено было остановиться именно на нем.

Python – высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ.

1. Выбор и изучение структуры сайта

Озон – старейший российский универсальный интернет-магазин; по данным исследовательского агентства Data Insight, второй по обороту онлайн-магазин России. После недолгих раздумий выбор пал именно на него.



Скриншот, сделанный скриптом.

После его изучения стало понятно, что мне недостает знаний по механизмам работы сайтов. Проблема в том, что простейшие сайты построены просто с помощью html.

HTML - Стандартизированный язык разметки документов для просмотра веб-страниц в браузере.

Однако сайт использовал динамическое подгрузку страницы. Это происходит с помощью скрипта, который заложен в самой странице. Потому при прямом запросе страницы с помощью requests получалась лишь ссылка на скрипт, без необходимой для сбора информации.

Это осложняло выбор библиотек, ведь теперь нужно было не просто получить html-файл и распарсить его, а загрузить заранее страницу и с нее уже собирать информацию.

1. Просмотр решений, реализованных ранее.

В интернете я нашел несколько платных решений. Первые были комплексными (включали в себя парсер для нескольких сайтов), а соответственно дорогими. Вторые обозначались как парсер именно с Озона. Невысокая цена была их преимуществом. Одна лишь проблема, все решения были нерабочими, что зачастую указывалось даже самими авторами, похоже по причине их устаревания.

Из бесплатных решений с открытым кодом было найдено несколько. Рабочих вариантов не было, но удалось подчерпнуть важное: библиотеки для использования и алгоритм.

1. Выбор и изучение библиотек

Для сбора данных требовалось заходить на страницу сайта, собирать с нее информацию, а затем собирать ее в файле на компьютере. Выбор пал на библиотеки:

- Selenium – Это инструмент для автоматизации действий веб-браузера. В большинстве случаев используется для тестирования Web-приложений, но этим не ограничивается. В частности, он может быть использован для решения рутинных задач администрирования сайта или регулярного получения данных из различных источников. Работает он за счет эмулирования действий пользователя в специальном браузере. Хоть загрузка страницы довольно долгий процесс, зато это позволило избежать самую главную проблему – динамическое создание страницы.

- Pandas – Программная библиотека на языке Python для обработки и анализа данных. Работа pandas с данными строится поверх библиотеки NumPy, являющейся инструментом более низкого уровня. Предоставляет специальные структуры данных и операции для манипулирования числовыми таблицами и временными рядами. В нашем случае она будет отвечать за перенос данных в файл.

- NumPy — это open-source модуль для python, который предоставляет общие математические и числовые операции в виде пре-скомпилированных, быстрых функций.

- Time – модуль для работы со временем. Используется для замера времени работы скрипта.

Рассматривались также библиотеки BS4 и requests в качестве замены Selenium, но так как страница сайта генерируется с помощью скрипта, было невозможно их использование. Но есть вариант для развития парсера с помощью добавления этих библиотек, а еще многопоточности, что позволило бы, чуть замедлив, увеличить количество собираемой информации о каждом товаре.

## Разработка алгоритма скрипта и его написание

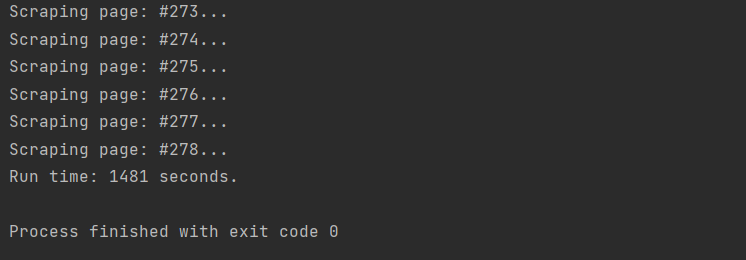
Идеально было бы переходить на страницу каждого товара и оттуда собирать всю информацию. Для этого, зайдя на главную страницу, скрипт собирал ссылки на товары в массив и последовательно по ним переходил.

Реализовав данный вариант, появились большие проблемы – появлялась капча.

Капча - Компьютерный тест, используемый для того, чтобы определить, кем является пользователь системы: человеком или компьютером. Ее почти невозможно пройти с тех малых мощностей и возможностей, что были в моем распоряжении, а потому был лишь один способ – избегать ее появления.

В данном случае капча появлялась из-за того, что скрипт очень быстро переходил со страницы на страницу. Из-за этого на 30-40 товарах уже вылезала капча, а в данном случае это 1-2 страница.

Самым очевидным решением было добавление задержки. При 1 секунде капча не пропадала, то же и при 2. Так проверяя поочередно, я дошел до задержки в 5 секунд, что было невероятно долго. Более того, капча не пропадала даже в этом случае, она лишь появлялась позже. Потому мне нужно было иное решение.

Так было решено остановиться на парсинге только наименований, цен и количества отзывов, как условный показатель популярности продукта, которые можно было собрать с главной страницы. То есть не надо было переходить на страницы непосредственно товаров. Без задержки обойтись не получилось (2 секунды), но скрипт начал отрабатывать полный цикл в почти 300 страниц. 

Скрипт выполнил свою работу чуть меньше, чем за 25 минут.

В ходе разработки добавилась возможность использовать headless режим браузера. До этого вход в этом режиме сразу же сигнализировал сайту, что работает скрипт, а потому выдавало капчу.

В чем же плюсы headless режима:

У него нет реальной отрисовки содержимого, то есть он все отрисовывает в памяти.

За счет этого он потребляет меньше памяти, потому что не нужно отрисовывать картинки или гигабайтные PNG, которые люди пытаются при помощи бомбы положить в бэкенд.

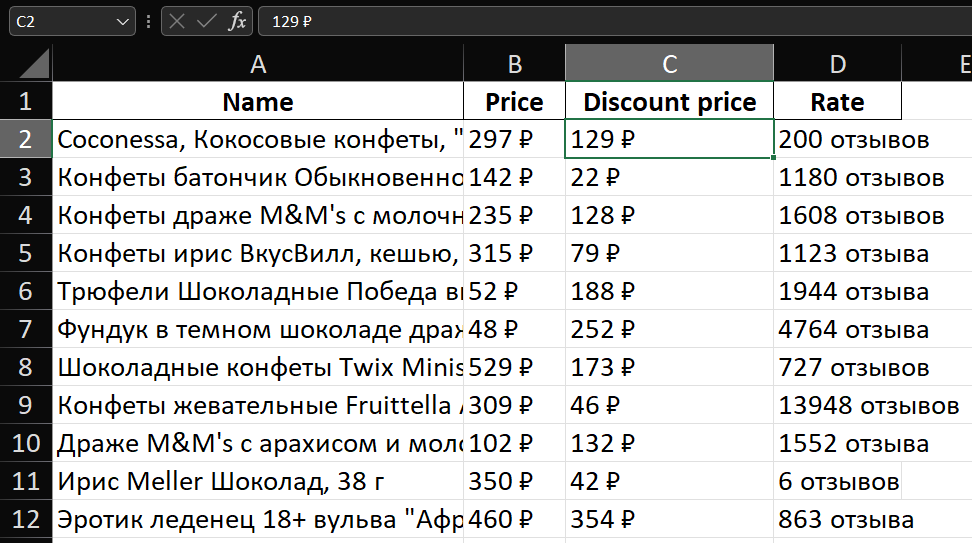
Он работает быстрее, потому что ему ничего не нужно отрисовывать на реальном экране.

Имеет программный интерфейс для управления. Вы спросите — у него же нет интерфейса, кнопочек, окошек? Как же им управлять? Поэтому конечно же он имеет интерфейс для управления.

За счет него удалось увеличить скорость до 15 минут.

Изображение выглядит как текст

Автоматически созданное описание

Выгрузка в файл происходила по окончанию сбора самой информации скриптом.

Лишь после написания самого парсера появилась идея о том, что с помощью библиотеки requests можно получать html-страницу, собирая ссылки на товары с главной со страницы. Этот алгоритм позволил бы не только собирать больше информации, но и добавить многопоточность парсеру, хоть и в ущерб времени. Но это было труднореализуемо, т. к. страница товара тоже формируется с помощью скрипта, что вводило в исступление при изучении. От этой идеи было тоже решено отказаться.

# Итог

По итогам работы был разработан скрипт, который собирает информацию по определенной категории на маркетплейсе Ozon. Так он собрал за 25 минут информацию по категории Конфеты (278 страниц, 36 наименований на каждой).

# Перспективы развития

Пока скрипт собирает лишь поверхностную информацию, так что добавление сбора всей информации по наименованию. Добавление многопоточности парсеру. Доработка устойчивости скрипта к пользовательский ошибкам.

# Список литературы

1. Хабрахабр - русскоязычный веб-сайт в формате системы тематических коллективных блогов с элементами новостного сайта, созданный для публикации новостей, аналитических статей, мыслей, связанных с информационными технологиями, бизнесом и интернетом: <https://habr.com/ru/>
2. Ozon: <https://www.ozon.ru/>
3. Википедия – универсальная интернет-энциклопедия: <https://ru.wikipedia.org/>
4. Документация Selenium: <https://www.selenium.dev/selenium/docs/api/py/index.html>
5. Документация Pandas: <https://pandas.pydata.org/>