Муниципальное общеобразовательное учреждение

Гимназия города Малоярославца

**Проект**

**«Парсеры. Сбор информации с сайта читай-город»**

**Выполнил:**

**Ученик 11 «Б» класса**

**Сидиков Никита**

**Руководитель:**

**Калашников Виктор Сергеевич**

г. Малоярославец

2022

**Содержание**

[Введение 3](#_Toc119567999)

[ГЛАВА I. Парсинг в современном мире. 4](#_Toc119568000)

[1.1. Определение и принцип работы. 4](#_Toc119568001)

[1.2. Законно ли использовать парсинг? 4](#_Toc119568002)

[1.3. Для чего и кем используется парсинг. 5](#_Toc119568003)

[1.4. Классификация парсеров. Парсеры на основе Python и PHP. 6](#_Toc119568004)

[1.5. Парсеры на основе Python и PHP 7](#_Toc119568005)

[ГЛАВА 2. Создание Парсера. 9](#_Toc119568006)

[2.1. Предпосылки создания программы-парсера. Курсы Иннополис 9](#_Toc119568007)

[2.2. Начало кода. Загрузка библиотек. 11](#_Toc119568008)

[2.3. Инициализация драйвера 11](#_Toc119568009)

[2.4. Selenium. Путь до страницы с книгами 12](#_Toc119568010)

[2.5. Поиск информации о книге. 13](#_Toc119568011)

[2.6. Запуск программы 16](#_Toc119568012)

[2.7. Первое включение программы. Потенциальные проблемы. 16](#_Toc119568013)

[2.8. Цельный код программы 18](#_Toc119568014)

[Заключение 19](#_Toc119568015)

# Введение

В современном мире с каждым годом все большую и большую роль играют компьютерные технологии. Компьютер проник во все сферы деятельности человека, начиная с образования и заканчивая созданием новых материй. Причем при поднятии технологии на следующую ступеньку открывается сразу несколько путей усовершенствования. Создаются новые программы, заменяя собой старые, усовершенствуются наиболее используемые. Сейчас компьютерная техника принимает все более разнообразные очертания и способы применения, продолжает внедряться в новые сферы жизни. Благодаря интернету мы можем без особых усилий заказать любую вещь на дом, оплатить коммунальные счета, поддерживать связь с родственниками из других концов Земли.

**Цель:**

Создать программу, способную автоматизировано собирать данные о книгах с сайта Читай-Город и сохранять эту информацию в текстовый файл

**Задачи проекта:**

1. Узнать больше информации о парсинге, видах парсеров
2. Научиться создавать парсер на основе языка программирования Python
3. Написать код, учитывая все нюансы сайта и браузера
4. Написать ошибки, которые могут возникнуть во время работы программы
5. Сделать выводы по окончании работы

**Актуальностью проекта является:**

За последние годы также результатом технического прогресса является переизбыток информации. Порой человек просто не в состоянии систематизировать гигантский объем информации вручную. И мне захотелось узнать, как можно без особого труда собирать определенную информацию с различных источников. Я начал искать, и ответ не заставил себя долго ждать – парсинг, быстро развивающаяся ветвь программирования. Но я понимал, что обучиться самостоятельно будет непросто, поэтому я начал искать курсы, различные видеоролики по обучению парсингу.

Я остановил свой выбор на бесплатном курсе от университета Иннополис. О данном курсе будет сказано в практической части в подглаве 2.1. После прохождения курсов, я освоил основы создания парсеров. На одном проекте я попытаюсь рассказать и показать большинство возможностей парсинга, расскажу про все трудности, с которыми мне пришлось столкнуться на пути создания парсера и в конце подведу итог, способен ли парсер облегчить жизнь обычному человеку.

# ГЛАВА I. Парсинг в современном мире.

## 1.1. Определение и принцип работы.

Парсинг - процесс сбора и систематизации информации, размещенной на определенных сайтах, с помощью специальных программ, автоматизирующих процесс по заданным условиям. Парсить можно данные с сайтов, поисковой выдачи, форумов и социальных сетей, порталов и агрегаторов. Сам же парсер – код-программа, скрипт, с помощью которой и осуществляется сбор и систематизация данной информации. Источником данных может служить текстовое наполнение, HTML-код сайта, заголовки, пункты меню, базы данных и другие элементы.

**Ход работы программы-парсера:**

1. Поиск необходимых источников по заданным параметрам — например, парсеру можно дать список сайтов (ссылок), а он найдёт на них страницы с интересующим для нас содержимым;
2. Извлечение из источников нужную информацию — определенный текст на сайте (заголовки, статьи, текст с жирным шрифтом), ссылки, картинки и т.д.;
3. Конвертация полученных данных в формат, удобный для изучения, а также по необходимости их систематизация в виде таблиц или отчетов для дальнейшего использования;
4. Сохранение информации в нужном виде, по необходимости сохранить в файл.

Объектом парсинга может быть любая грамматически структурированная система: информация, закодированная естественным языком, языком программирования, математическими выражениями и т.д.. Например, если исходный массив данных представляет собой HTML-страницу, парсер может вычленить из кода информацию и перевести ее в текст, понятный для человека.

Смысл парсинга в том, чтобы ускорить рутинную работу. Чтобы собрать и сохранить в таблице список из тысячи статей на сайте, человек потратит часы. Парсер сделает эту работу за несколько минут. Парсер ускоряет работу в сотни раз и допускает меньше ошибок, чем человек.

## 1.2. Законно ли использовать парсинг?

После выяснения, что такое парсинг, может показаться, что это нечто, не соответствующее нормам действующего законодательства. На самом деле это не так. Законом не преследуется парсинг. Более того, в Конституции закреплено «право свободно искать, получать, передавать, производить и распространять информацию любым законным способом». Если информация есть в открытом доступе, не защищенна любыми ограничениями, то ее можно копировать и распространять. Законным также является сбор сведений для сайтов, где не требуется авторизация. Парсеры просто позволяют ускорить процесс и избежать ошибок из-за человеческого фактора. **Но все же есть несколько ограничений, запрещающих использовать парсинг для:**

1. Совершения DDOS-атак (на сайт в результате парсинга данных ложится слишком высокая нагрузка)
2. Использования авторского контента (фотографии, уникальные тексты, подлинность которых заверена у нотариуса и т. п.) с целью плагиата
3. Получения данных для спама, спам-рассылок
4. Кража личных данных пользователей, полученных в ходе парсинга, и их злоумышленное использование.

## 1.3. Для чего и кем используется парсинг.

Парсинг открывает для пользователя обширный простор для действий. Он помогает автоматизировать процесс анализа данных и снижает нагрузку на сотрудников, перенаправляет их время и силы на решение других задач. Также ускоряет анализ большого объема информации — например, нескольких сотен страниц интернет-магазина, несколько тысяч статей википедии или обширную базу данных. Помогает разработчикам выявлять ошибки на сайте или в любом другом информационном продукте, если в программе заданы настройки на их поиск. Такими сервисами пользуются интернет-маркетологи, вебмастера, SEO-специалисты, сотрудники отделов продаж, аналитики баз данных.

**Примеры использования парсинга:**

1. **В создании сайтов.** Как и языки программирования, языки разметки (например HTML) непонятны компьютеру. Чтобы он смог отобразить HTML-разметку в виде визуально структурированного и понятного интерфейса сайта, парсер браузера анализирует исходный код страницы, вычленяет нужные данные, переводит их в понятный машине формат. Также парсинг позволяет выявить ошибки и недочеты в созданном сайте.
2. **Сбор новостей.**Для упорядоченной подачи новостей сайты-агрегаторы или новостные агентства используют парсеры. Они собирают обновления со всех доступных источников, анализируют их и подают сотрудникам для конечной редактуры и публикации.
3. **Интернет-маркетинг.**В SEO и SMM с помощью парсеров собирают и анализируют данные пользователей, товарные позиции в интернет-магазинах, изменение цен. Эти данные используются для оптимизации сайта, продвижения коммерческих групп в социальных сетях, настройки таргетированной и контекстной рекламы.
4. **Мониторинг цен.** Парсерами можно извлечь расценки товаров на сайтах-конкурентах, чтобы проанализировать текущую ситуацию на рынке и выработать ценовую политику. Также с их помощью можно привести прайс-листы на собственном сайте в соответствие с ценами у поставщиков.

## 1.4. Классификация парсеров. Парсеры на основе Python и PHP.

В настоящее время существует множество способов создания и реализации парсера. Их можно классифицировать по различным критериям.

1. По способу доступа к интерфейсу: облачные решения и программы, которые требуют установки на компьютер.

* Облачные сервисы не требуют установки на ПК. Они реализуется на серверах разработчика, вы получаете только результат парсинга (например, [Xmldatafeed](https://xmldatafeed.com/), [Octoparce](https://www.octoparse.com/), [ParseHub](https://www.parsehub.com/).)
* ПО для программы-парсера нужно устанавливать на компьютер. НА выходе пользователь также получает только результат работы парсера(например, [ParserOK](https://parserok.ru/), [Datacol](http://web-data-extractor.net/))

1. По технологии: парсеры на основе языков программирования (Python, PHP), расширения для браузеров, надстройки в Excel, формулы в Google таблицах.
2. По назначению: мониторинг конкурентов, сбор данных в определенной нише рынка, парсинг товаров и цен для наполнения каталога интернет-магазина, парсеры данных соцсетей, проверка оптимизации своего ресурса.

**Более подробно остановимся на парсерах по технологиям:**

1. Парсеры-расширения для браузеров. (простые программы-расширения, используемые для сбора информации из html-кода сайта, используя язык запроса XPath, и дальнейшей работы с данными)
2. Парсеры сайтов на основе Excel. (В таких программах парсинг с последующей выгрузкой данных в форматы XLS\* и CSV реализован при помощи макросов — специальных команд для автоматизации действий в MS Excel. Пример такой программы — [ParserOK](https://parserok.ru/))
3. Парсеры на основе Python и PHP. (Такие парсеры создают программисты. Без специальных знаний сделать парсер самостоятельно не получится. На сегодня самый популярный язык для создания таких программ Python. Подробнее разберем данный вид парсеров)

## 1.5. Парсеры на основе Python и PHP

Парсинг на основе языка программирования Python, вероятно, является самым популярным способом создания и реализации парсера. Ведь когда программист самостоятельно пишет код, он может оптимизировать программу под определенные ресурсы, статические или динамические сайты, сайт с файлами cookies или API. Существует множество библиотек, способных облегчить процесс парсинга или облегчить код программисту.

В программировании библиотека — это набор или, точнее сказать, предварительно настроенный набор подпрограмм, функций и операций, которые в дальнейшем может использовать ваша программа. Эти элементы часто называют модулями, которые хранятся в объектном формате. Библиотеки очень важны, потому что вы можете загрузить модуль и использовать все, что он предлагает, без явной связи с вашей программой. Они действительно автономны, так что вы можете создавать свои собственные программы с ними, и все же они остаются отделенными от ваших программ.

**Наиболее популярные из них при парсинге:**

* 1. **Beautiful soup**

Библиотека для извлечения данных HTML и XML со статических сайтов или страниц. Работает почти с любыми видами парсеров, дает естественные способы навигации, поиска и извлечения информации из HTML-дерева. Очень полезная библиотека для парсинга сайтов, целью которых является извлечение информации в виде текста или анализа HTML кода. Для упрощения и ускорения работы программы можно использовать нестандартные модули HTML. Наиболее популярным является -lxml-

* 1. **Request**

Библиотека разработана с учетом потребностей современных web-разработчиков и актуальных технологий. Многие операции автоматизированы, а ручные настройки сведены к минимуму. В ее возможно входит, например, управление HTTP-запросами и поддержка постоянного соединения с повторным использованием, автоматическое декодирование контента, загрузка и выгрузка файлов, передача требуемых заголовков на web-ресурсы и многое другое. (HTTP- протокол, способный передавать и получать различные данные. В его основе входит технология «клиент-сервер» Клиент посылает запрос на сервер, сервер ожидает соединения для получения запроса, производит необходимые действия и возвращает обратно сообщение с результатом.) С помощью request вы можете добавлять контент, например заголовки, формы, многокомпонентные файлы и параметры, используя только простые библиотеки Python. Также вы можете получать доступ к таким данным.

* 1. **Selenium WebDriver**

Данная библиотека предназначена для управление браузером с помощью драйвера. WebDriver представляет собой набор драйверов для различных браузеров и клиентские библиотеки на разных языках программирования для управления драйвером. Драйвер на браузер обычно устанавливается отдельно от библиотеки. По сути, использование парсера с таким веб-драйвером сводится к созданию бота, который выполняет всю работу на браузере автоматизировано. Чаще всего Selenium используют для сбора информации c web-сайтов, где присутствует ожидание загрузки страницы. Частным случаем является пример, когда обычный парсинг, используя BS4, невозможен из-за перенаправления или ожидания страниц (мы можем увидеть сообщения, такие как: «Подождите, страница загружается» или «Перенаправляем вас на другую страницу») Понятно, что такие страницы не является окончательной целью парсинга, однако обойти в большинстве не представляется возможным. Поэтому Selenium WebDriver позволяет в таких случаях «ожидать», как ожидал бы человек, пока на странице, к примеру, не появится элемент с необходимым именем. Еще одним плюсом является то, что Selenium демонстрирует действия веб-драйвера на странице, что помогает в случае ошибки быстро определить область, где была допущена ошибка в коде.

С использованием различных библиотек программистам стало намного легче писать коды любой сложности и вариативности. Также написанные программы через длительное время после создания будут легко читаемы для других программистов, ознакомленных с принципом работы библиотеки, используемой при написании кода.

Данные 3 библиотеки (request, selenium и beautiful soup) являются основополагающими при создании парсера любой сложности. Также эти библиотеки легли в основу создания моей собственной программы, речь о которой пойдет во второй части проекта

# ГЛАВА 2. Создание Парсера.

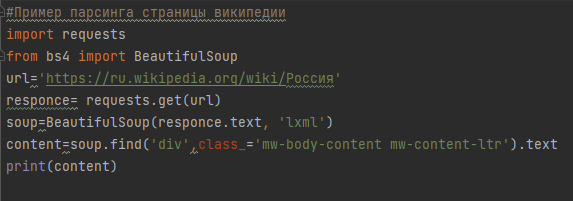
## 2.1. Предпосылки создания программы-парсера. Курсы Иннополис

С октября 2021 года меня все чаще и чаще посещала идея создания своего проекта на основе программирования. Я хотел написать свой код, который будет интересен мне как и в написании, так и в работе в дальнейшем. Изучив множество разных курсов в интернете, я нашел то, что действительно могло бы мне помочь – курсы по программированию от университета Иннополис.

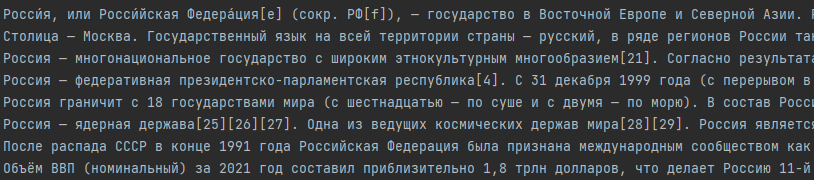
Данный проект для учащихся 8 – 11 классов представляет на выбор несколько направлений по программированию на Python, например, создание парсеров, создание telegram-ботов и графическое моделирование. Курс проводится в течение 5 месяцев (с января по май) и помогает познакомиться с объектно-ориентированным программированием и Python для создания дальнейших проектов. Мой выбор остановился на создании парсеров.  
Обучение проходило в онлайн-формате 2 раза в неделю с домашними заданиями и общением с наставником. За все 5 месяцев прохождения курса я узнал много нового о программировании на Python, узнал много тонкостей создания парсеров, познакомился с интересными людьми. По окончании курса я получил сертификат об успешной прохождении курса. И теперь я был полностью готов создать свой первый проект.



Прежде всего я решил выбрать сайт, с которого буду «собирать» информацию. Довольно скучно собирать информацию со статических сайтов, то есть с сайтов, где вся информация «вшита» в HTML-код сайта и для ее парсинга нужно всего лишь знать основы парсинга и основы библиотеки requests и BS4. Одним из примеров сайта со статическим кодом является Википедия. Очень просто парсить страницы Википедии, т.к. вся информация встраивается в код и при многоразовом открытии сайта она не меняется и сразу доступна потенциальному читателю. Поэтому программа для парсинга информации с любой страницы Википедии может занимать не больше 7 строк! А на выходе мы получаем всю информацию (к примеру статью), находящуюся на странице. Исходя из этого, Википедия не вызывается никакого интереса к парсеру, использую данный сайт можно только тренироваться создавать свои первые парсеры. В примере представлен парсер статьи «Россия» из Википедии:



Элемент вывода программы:

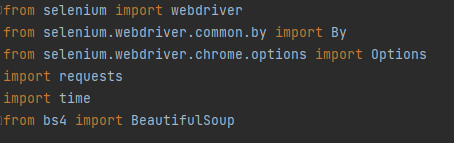


Я решил выбрать что-то посложнее – читай-город. Данный сайт не требует у пользователя принятия файлов-cookies, что значительно упрощает создание парсера, есть механизмы «скрытия» данных от парсера, скрипты, но с помощью библиотеки seleniumих легко обойти. В общем, сайт показался мне довольно интересным вариантом для моего будущего проекта, поэтому я остановился на нем.

Программу я решил разделить на несколько частей: загрузка библиотек, инициализация драйвера, главный механизм (selenium), парсер отдельного элемента и сохранение его в файл, запуск программы. Остановимся на каждом подробнее.

## 2.2. Начало кода. Загрузка библиотек.

Я решил использовать самые базовые и распространенные библиотеки для создания парсера: request, bs4, selenium и библиотеку time.



Было принято решение не загружать всю библиотеку selenium, т.к. она довольно обширная и имеет множество возможностей, которые не пригодятся мне во время написания программы. Я загрузил только самые необходимые разделы: 1) webdriver (драйвер браузера; не имеющая собственного интерфейса программная библиотека, которая позволяет **различным другим программам взаимодействовать с браузером, управлять его поведением, получать от браузера какие-то данные и заставлять браузер выполнять какие-то команды)**; 2) By(механизм, необходимый для поиска элементов на динамической странице); 3)Options(Опции настройки веб-драйвера)

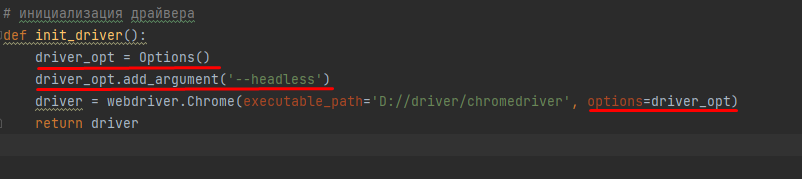
Библиотека requests предназначалась выгрузки информации о каждой книге (получает HTML страницу книги используя url-адрес)

Библиотека bs4 необходима для поиска необходимой информации с выгруженной HTML страницы по тэгам (h1-заголовки, p-абзацы, ul-списки, href-гиперсслыки).

Библиотека time использовалась для того, чтобы запускать на драйвере режим ожидания для того, чтобы страница полностью прогрузилась и была доступна для дальнейшей обработки.

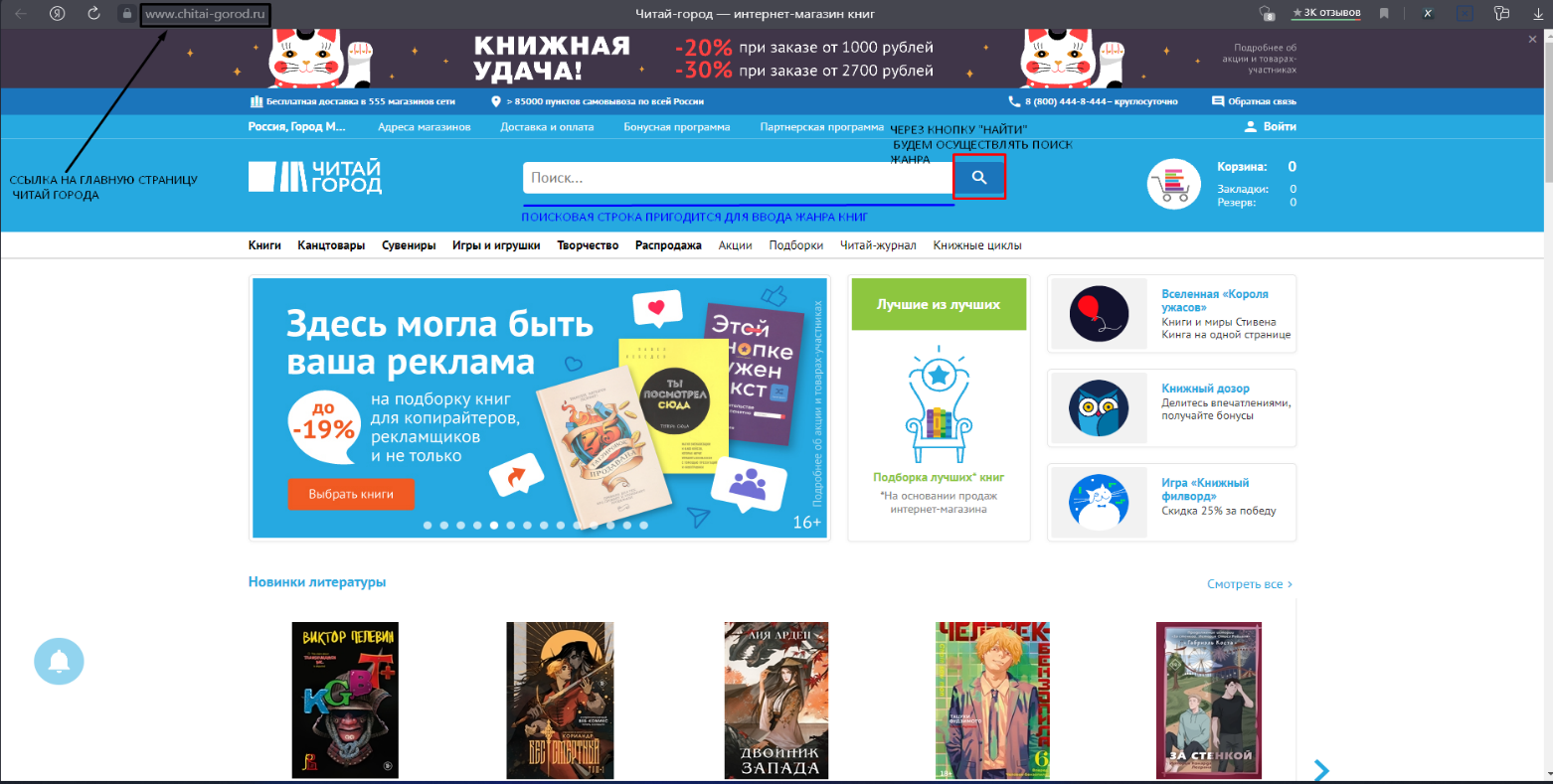
## 2.3. Инициализация драйвера

Для запуска веб-драйвера я создал функцию, для которой я создал переменную driver, чтобы в будущем мог быстро обращаться непосредственно к драйверу. Сам драйвер необходимо скачать с интернета и в программе указать путь к данному файлу для того, чтобы парсер работал стабильно. Со временем драйвер приходится обновлять из-за обновления версии браузера, иначе он просто не запустится. Настройки драйвера (Options) заключаются лишь в том, чтобы драйвер мог работать в закрытом окне и тем самым не нагружая систему. Если данной функции нет, то драйвер работает в открытом окне, мы видим весь функционал драйвера, но при этом идет нагрузка на систему, как от обычного браузера. Строки с настройками драйвера подчеркнуты красной чертой.



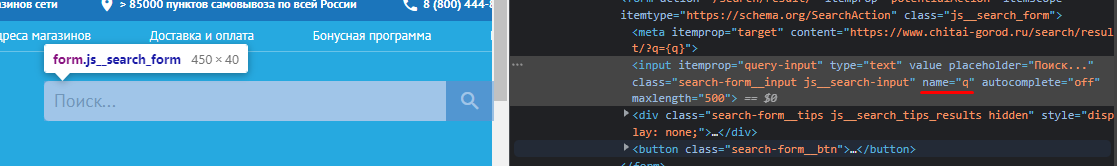
## 2.4. Selenium. Путь до страницы с книгами

Когда драйвер готов к использованию, можно приступить к основной части кода. Для начала проанализируем страницу читай города:

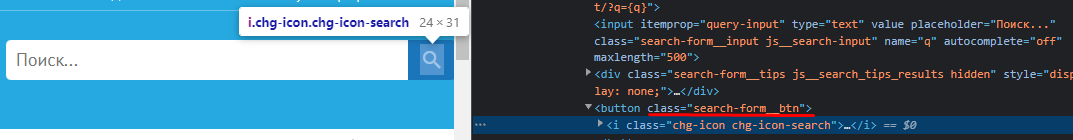


На главной странице нас интересует только 3 параметра: поисковая строка, кнопка «Найти» и ссылка на главную страницу. Поисковую строку используем для ввода жанра книги, кнопку «Найти» для поиска жанра, а ссылку передаем нашему веб драйверу, тем самым передаем драйверу стартовую страницу.

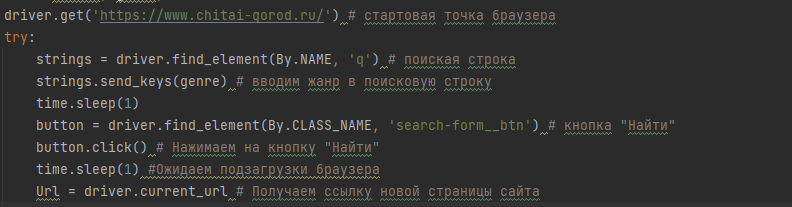
Чтобы вводить в поисковую строку необходимое для нас значение нам нужно через консоль разработчика найти, какому элементу соответствует поисковая строка. Для этого нужно исследовать элемент поисковой строки. Результат подчеркнут красной чертой:



Аналогичным образом осуществляем поиск кнопки «Найти». Результат подчеркнут красной линией:

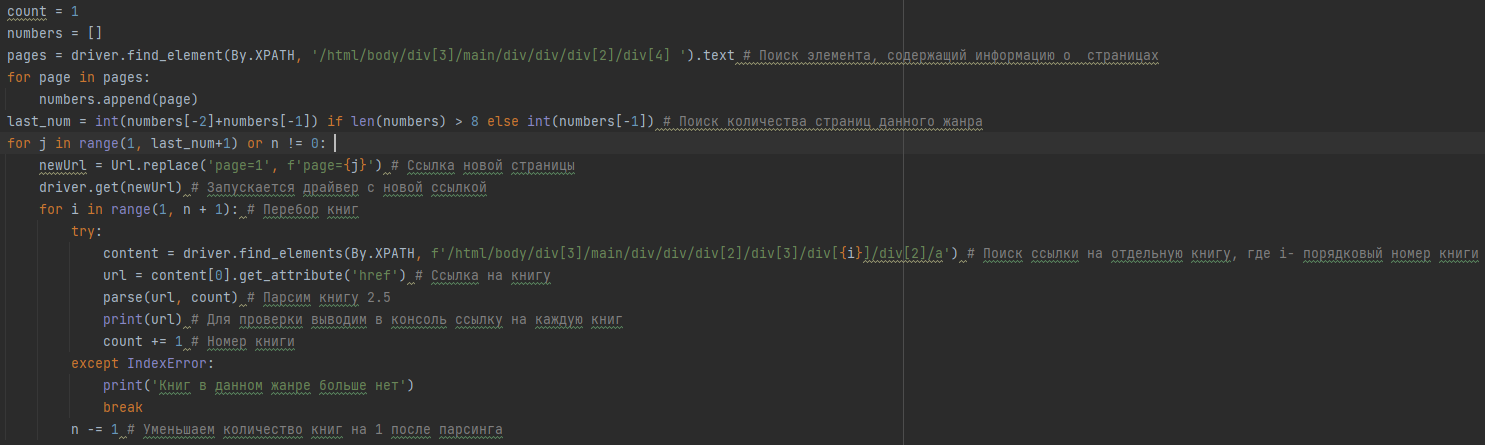


Итак, мы нашли все элементы, необходимые для веб-браузера, чтобы он смог перейти с главной страницы на страницу необходимого нам жанра. Пишем код:



Теперь нам нужно найти какое количество страниц занимает жанр. Это нам необходимо для того, чтобы, если нам необходимо найти книг больше, чем находится на 1 странице, программа могла самостоятельно переключать страницу сайта без ошибок. Например, если пользователь желает получить данные по 30 книгам, а на одной странице количество книг ограниченно 15, то программа откроет следующую страницу и оттуда спарсит недостающее количество книг. Также, в крайнем случае, если введено количество книг больше, чем доступно на сайте, программа выдаст сообщение «Книг в данном жанре больше нет» и досрочно завершит программу.

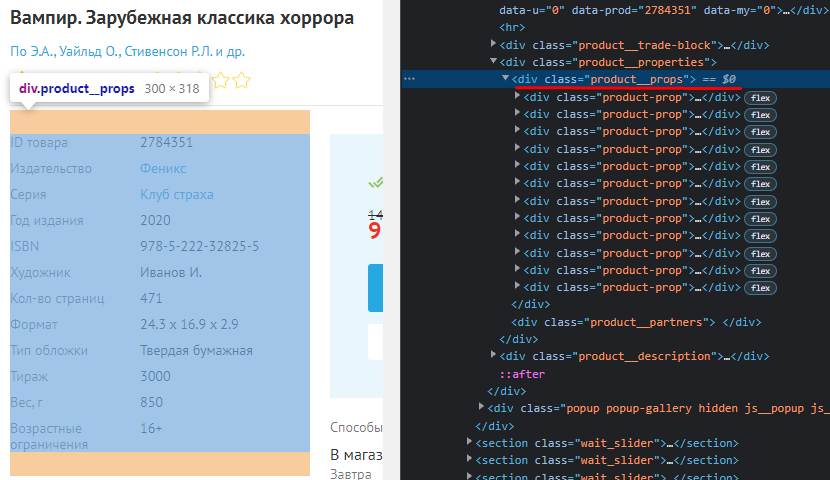
Код:



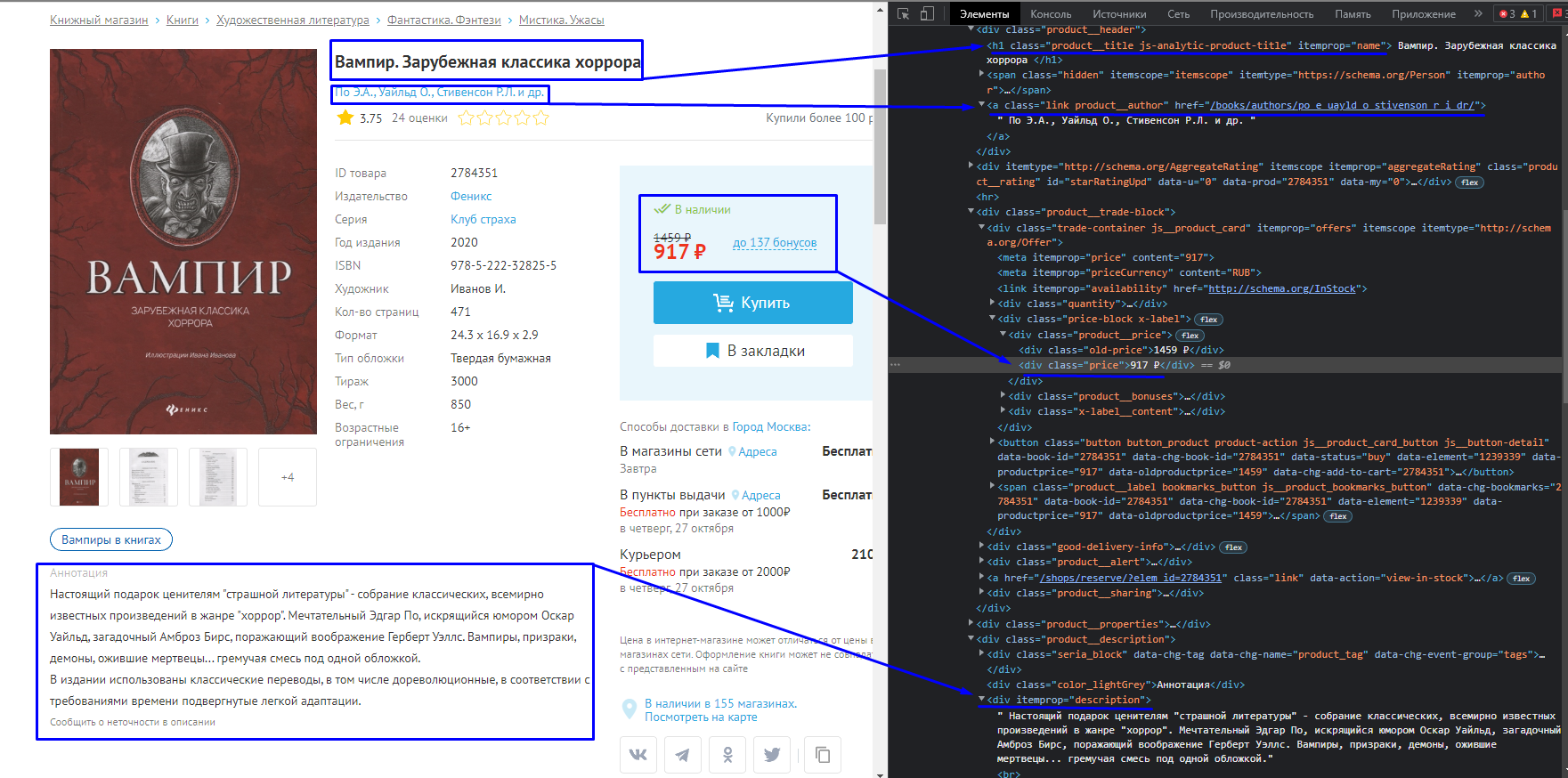
Больше selenium нам не понадобиться, парсить информацию о каждой книге мы будем с помощью requests и bs4.

## 2.5. Поиск информации о книге.

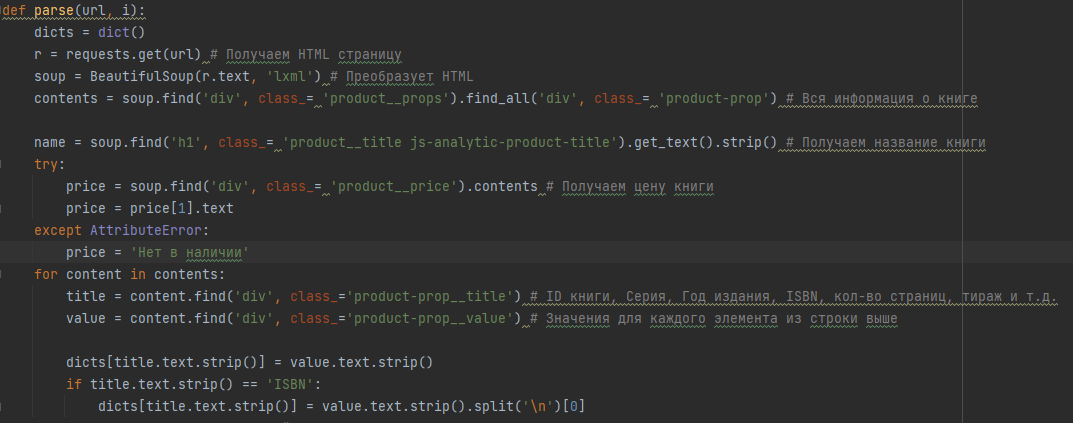
Теперь, когда мы имеем ссылку на книгу, мы можем спарсить страницу книги с помощью requests. Первым делом мы должны получить всю информацию о книге: название, цену, автора, ID, год издания, издательство и т.д. Для этого мы должны найти элемент с названием класса product-prop, так как именно в этом классе находится большинство интересующей нас информации.



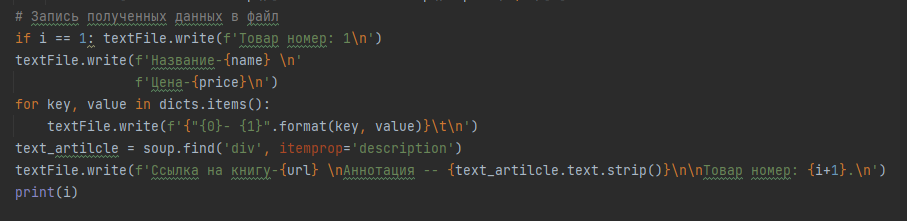
После этого аналогичным образом находим названия классов для названия книги, цены (если парсер не сможет найти цену, то будет выведено, что товара нет в наличии), имя автора, и аннотации (краткая презентация книги)



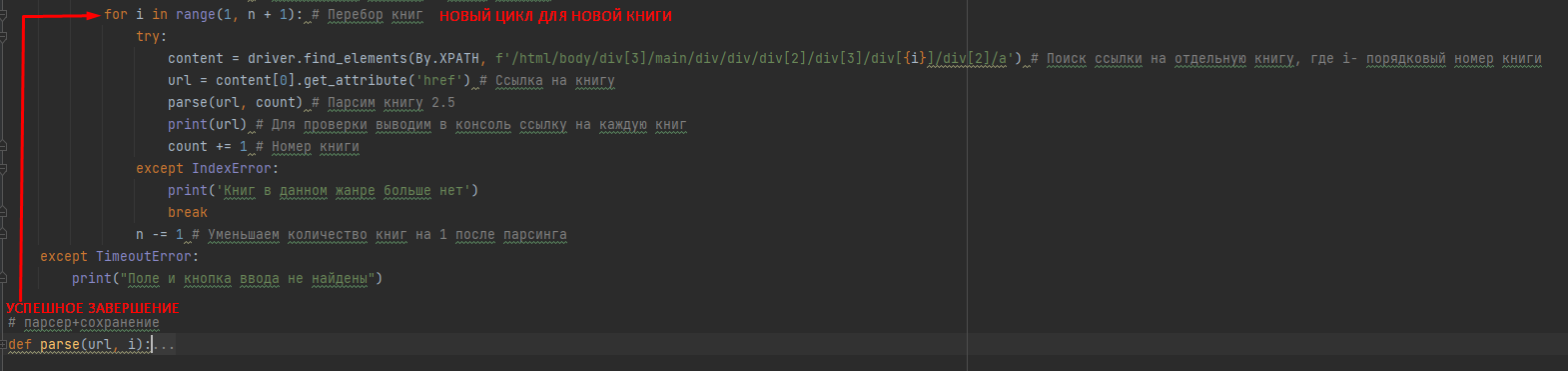
Данные наименования классов будут соответствовать всем книгам на сайте, т.к. все новые книги загружаются на сайт по старому шаблону, а значит, все тэги, XPath, имена классов, div-ов одной книги будут соответствовать наименованиям любой другой книги, что значительно упрощает работу парсера. Код полученный после преобразования алгоритмов и внесения данных:



Записывать данные в файл код будет записывать сразу в функции парсера. Запись производится стандартным образом на файл с расширением .txt. Если до запуска программы данного файла не было, программа самостоятельно создаст файл с именем «book.txt». Также на этом этапе в консоль выводится порядковый номер книги, которая подвергалась парсингу:



После успешного окончания парсинга функция parse закрывается и в функции selen запускается новый цикл для парсинга следующей книги:



Программа будет работать до тех пор, пока не спарсит все количество книг, после чего автоматически закроется.

## 2.6. Запуск программы

В самом начале мы должны запустить драйвер, используя функцию init\_driver, о которой рассказывалось в пункте 2.3. Создадим для этого переменную, чтобы позже передать драйвер в функцию selen (пункт 2.4). Также в функцию selen передадим 2 значения – genre (название жанра, книги которой мы хотим найти) и count (количество книг, которое нам нужно собрать). И открываем пустой текстовый файл с кодировкой UTF-8 для записи книг. В конце программы выключаем driver, что означает, что программа закончила свою работу.



## 2.7. Первое включение программы. Потенциальные проблемы.

Видеозапись с полной работой программы будет приложена вместе со всей работой. Среднее время работы программы варьируется от 30 сек до нескольких минут в зависимости от количества книг, которые необходимо собрать. Программа работает структурировано, без погрешностей и ошибок.

**Проблемы, которые могут возникнуть до или вовремя работы программы:**

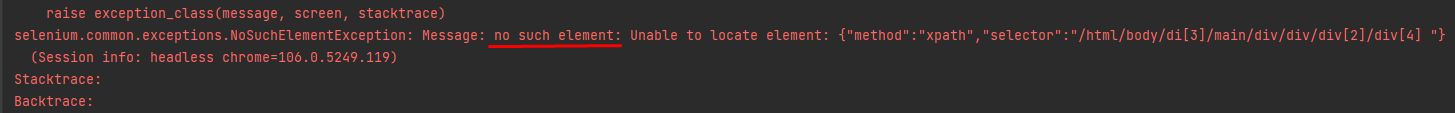
1. Версия веб-драйвера устарела. Необходимо установить последнюю версию с официального сайта веб-драйвера (<https://chromedriver.chromium.org/>) Ошибка легко устраняется

Ошибка:



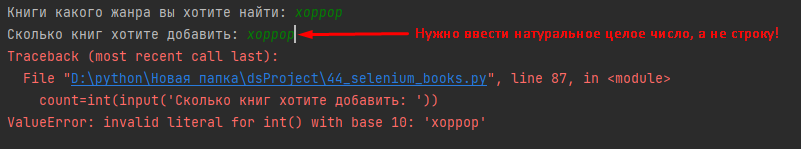
В файле прикреплена последняя версия веб-драйвера

1. Программа не нашла элемент



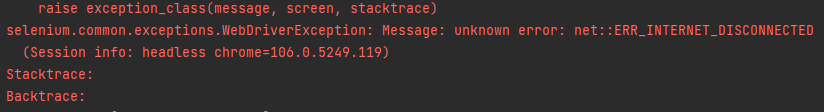
Данная ошибка появляется в случае, если сайт сменил некоторые элементы HTML страницы, поэтому программа не может найти нужный для дальнейшей работы, или в программе изначально была допущена ошибка в указании элемента. Ошибка легко устраняется

1. Ошибка неправильного ввода. Данная ошибка может произойти, если при вопросе «Сколько книг хотите добавить» ввели нечисловое значение (слово, символ, дробное число и т.д.).



Проблема решается перезапуском программы и вводом корректного значения. Ошибка легко устраняется.

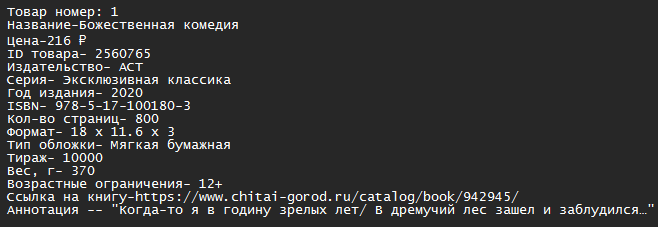
1. Отсутствует подключение к сети интернет



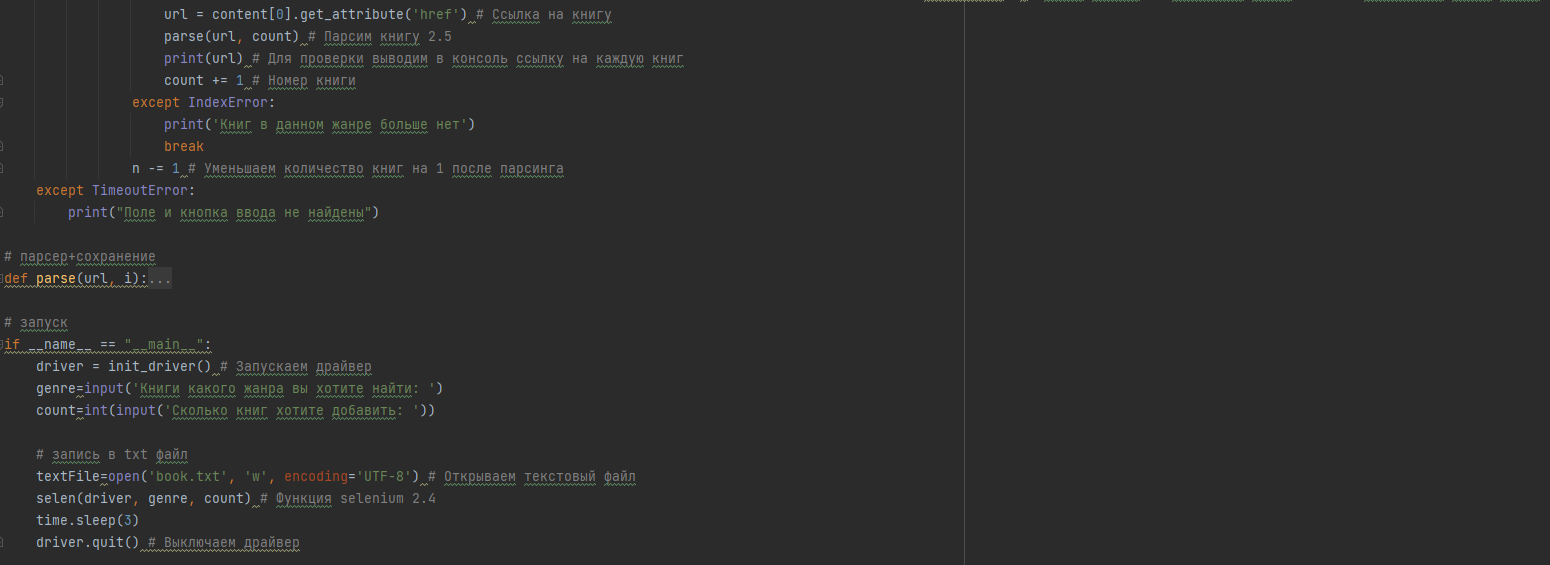
Веб-драйвер работает только при стабильном подключение к интернету, поэтому программа не работает без подключения к интернету. Необходимо провести диагностику сети.

1. Из-за медленного интернет соединения или из-за большой нагрузки программа может длительное время загружать данные в файл. Методы решения: перезапустить программу, переустановить подключение к сети или просто подождать

Если после запуска программы не выдало ошибки, то после окончания программы продукт ее работы будет находится в файле «book.txt» Пример продукта программы, на примере 1 книги жанра фэнтези:



## 2.8. Цельный код программы



# Заключение

Данная работа была посвящена знакомству с отдельной ветвью программирования – парсингом. Данный вид программирования имеет много подводных камней, начиная от большого количества средств разработки и заканчивая особенностями разным сайтов для парсинга. Самым интересным конечно же является парсинг на Python. Благодаря обучению на курсе я в полной мере освоил все основы для создания собственного парсинга, после чего успешно смог создать свой стабильно работающий парсер. Конечно, было множество проблем с созданием парсера (трудности в выборе сайта, его нестабильность, сложность в составлении алгоритма, который будет не только работать, но также будет понятным для чтения, проблемы с оптимизацией программы и т.д.), но, благодаря помощи моим наставникам, я смог довести код до распознаваемого и читаемого состояния.

Код демонстрирует большой спектр возможностей парсинга: от выборочного сохранения информации до использования веб-браузера. За все время изучения области парсинга, множества попыток создания первого парсера и, наконец, создания проекта я изучил очень интересную область программирования, имеющая популярность и спрос у программистов.

Я планирую и дальше развиваться в эту область программирования, потому что она многогранна, имеет большое количество ответвлений и направлений.

Подводя итог, могу с уверенностью сказать, что работа успешно завершена.