Отчет по проекту Веб скрейпинг и парсинг сайтов

МАХНЕВ КИРИЛЛ

КОМАНДА: IFTISHACKERS

Содержание

Введение

- В нашем современном цифровом мире приобретает все большее значение умение извлекать ценные данные из веб-ресурсов для принятия обоснованных решений. В рамках данного проекта было проведено исследование и разработка скриптов для веб скрейпинга с целью сбора и анализа информации с различных интернет-ресурсов. В данном отчете будут представлены результаты работы, методы, используемые инструменты. Проект нацелен на эффективное получение и использование данных из сети Интернет с целью улучшения бизнес-процессов и принятия обоснованных решений.

Цели и задачи

- Цели проекта:

- 1. Получение актуальных данных с веб-сайтов для использования в аналитике.
- 2. Автоматизация процесса сбора информации с различных онлайн-ресурсов.
- 3. Создание дата-фрейма на основе полученных данных для дальнейшего анализа.

- Задачи проекта:

- 1. Исследование целевых веб-сайтов и определение структуры данных для скрейпинга.
- 2. Выбор подходящих инструментов и технологий для проведения веб-скрейпинга.
- 3. Написание скриптов для сбора, обработки и сохранения данных с веб-страниц.
- 4. Разработка алгоритмов для обхода препятствий, таких как капчи или ограничения скорости запросов.
- 5. Создание удобного интерфейса для визуализации и анализа полученных данных.

Основные понятия

- Парсинг – это процесс синтаксического анализа текста с целью извлечения определенной информации. Скрапинг – это процесс загрузки веб-страницы и попытка извлечения информации из неё, обычно в обход API, ограничений и правил пользования сайтом
- **Зачем нужен веб-скрапинг?**
- Веб-скрапинг может использоваться для различных целей, таких как:
  - сбор данных для анализа (например, статистика посещаемости сайтов)
  - мониторинг цен на товары и услуги
  - создание баз данных контента для исследований
  - автоматизация рутинных задач, связанных с работой в интернете
  - **HTML**
  - HTML (от английского HyperText Markup Language) – это язык гипертекстовой разметки текста. Он нужен, чтобы размещать на веб-странице элементы: текст, картинки, таблицы и видео.
  - Теги (с англ. tag) – это метки, которые классифицируют контент и облегчают его поиск для пользователей.

ТЕГИ ДЛЯ ФОРМАТИРОВАНИЯ ТЕКСТА	
<pre></pre>	Обрамляет предварительно отформатированный текст.
	Создает САМЫЙ БОЛЬШОЙ заголовок
	Создает самый маленький заголовок
	Создает жирный текст
	Создает наклонный текст
<tt></tt>	Создает текст - имитирующий стиль печатной машинки.
<cite></cite>	Используется для цитат, обычно наклонный текст.
	Используется для выделения из текста слова (наклонный или жирный текст)
	Устанавливает размер текста в пределах от 1 до 7.
	Устанавливает цвет текста, используя значение цвета в виде RRGGBB.
ГИПЕРССЫЛКИ	
	Создает гиперссылку на другие документы или часть текущего документа.
	Создает гиперссылку вызова почтовой программы для написания письма по указанному адресу.
	Отмечает часть текста как цель для гипперссылок в документе.
	Создает гиперссылку на часть текущего документа.

Методы и инструменты веб скрейпига

- Библиотека Requests:
- Библиотека Requests для Python позволяет работать с HTTP-запросами любого уровня сложности, используя простой синтаксис. Это помогает не тратить время на написание кода, а быстро взаимодействовать с серверами.
- Библиотека BeautifulSoup4
- BeautifulSoup – это библиотека для парсинга HTML и XML документов в Python. Она предоставляет простой и удобный способ извлекать данные из веб-страниц, а также облегчает работу с этими данными. У библиотеки BeautifulSoup удобный интерфейс для взаимодействия с HTML-кодом, который позволяет легко находить нужные элементы и извлекать из них информацию. Эта библиотека является одной из наиболее популярных и широко используется для работы с парсерами и при анализе данных.

Исследование целевых веб сайтов

- Для успешной реализации проекта по веб скрайпингу и парсингу было проведено исследование различных веб-ресурсов с целью выбора оптимального для скрапинга. Одним из ключевых сайтов, который был выбран в качестве объекта скрапинга, стал Avito.
- Почему Avito был выбран в качестве сайта для скрапинга:
- - Широкий ассортимент данных: Avito предоставляет широкий спектр информации о товарах и услугах, что позволяет получить разнообразные данные для анализа.
- - Популярность и актуальность: Avito является одним из самых популярных онлайн-площадок для объявлений, что обеспечивает доступ к актуальным данным.
- - Структурированность данных: Сайт Avito хорошо структурирован, что упрощает процесс парсинга и извлечения требуемой информации.
- Выбор Avito в качестве целевого сайта для скрапинга обусловлен возможностью получения высококачественных данных, актуальности информации и удобством работы с веб-страницами данного ресурса.

Разработка скриптов для сбора данных

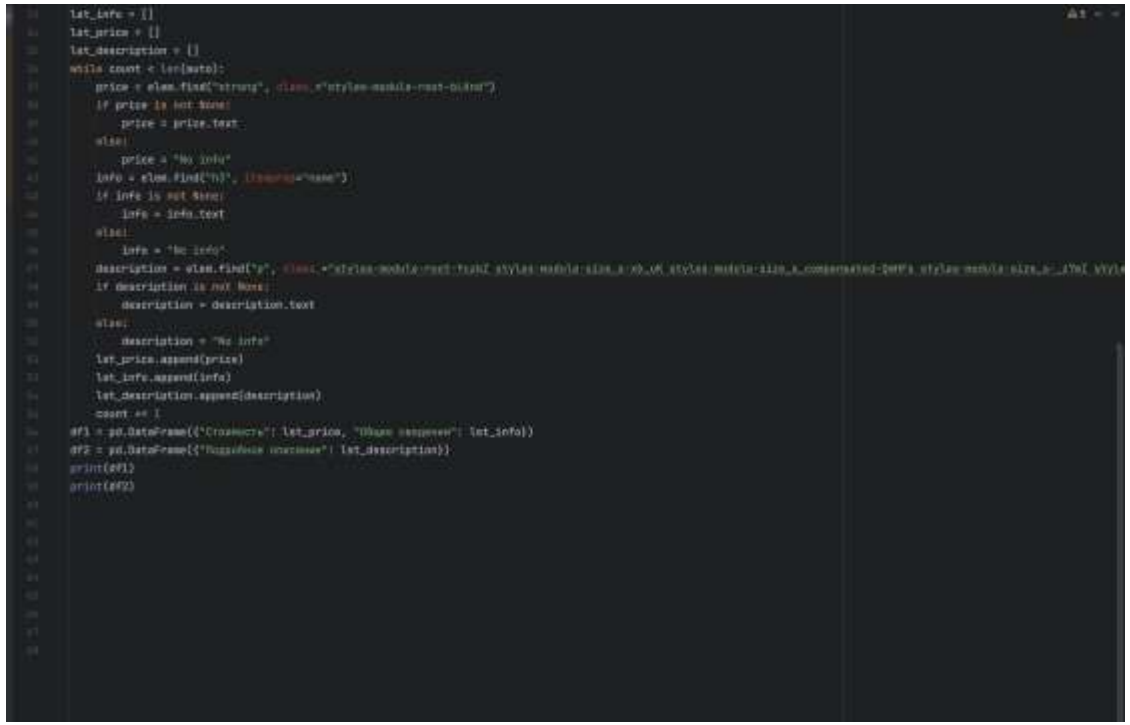
- Принцип работы кода:
- Код осуществляет скрапинг данных с страниц Avito, используя прокси и случайные задержки во избежание блокировки. Каждая страница парсится с помощью BeautifulSoup, чтобы извлечь информацию о цене, названии и описании объявлений об автомобилях. Полученные данные структурируются в DataFrame для последующего анализа или использования.

Рабочий код

```python
import requests
import pandas as pd
from bs4 import BeautifulSoup
import time
import random

auto = []
prox = {"https": "http://87.247.186.60:3080", "http": "http://50.222.245.58:80"}
head = {"Accept": "*/*",
        "User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/124.0.0.0 Safari/537.36"}
url = "https://www.avito.ru/moskva/avtomobili/bmw-X5g0AgICAUTgtgIk1yg?cd=1&p=2&radius=3000&searchRadius=3000&s="
page = 1
while page < 30:
    get = requests.get(url + str(page), headers=head, proxies=prox)
    soup = BeautifulSoup(get.text, "lxml")
    # Обход попукни для парсера (Этот код пропускает любую ссылку с атрибутом style=display: none и атрибута style, что является общей характеристикой ссылок, которые
    for link in soup.select('a'):
        if 'display' in link.get('style', default: '') and 'none' in link['style']:
            continue
    auto_info = soup.find_all("div", class_="iva-item-body-KlUuy")
    if auto_info is not []:
        auto.extend(auto_info)
        # Задержка запроса по времени между запросами чтобы уменьшить нагрузку на сайт
        time_delay = random.random()
        scaled_time_delay = 1 + (time_delay * (0 - 1))
        time.sleep(scaled_time_delay)
    else:
        print("Больше данных нет")
        break
    page += 1
print(auto)
count = 0
lst_info = []
lst_price = []
lst_description = []
while count < len(auto):
    price = elem.find("strong", class_="styles-module-root-bLKnd")
    if price is not None:
        price = price.text
    else:
```

Рабочий код



```
lst_info = []
lst_price = []
lst_description = []
while count < len(auto):
    price = elem.find("strong", class_="styles-module-root-bi8td")
    if price is not None:
        price = price.text
    else:
        price = "No info"
    info = elem.find('h3', itemprop="name")
    if info is not None:
        info = info.text
    else:
        info = "No info"
    description = elem.find("p", class_="styles-module-root-TsahI styles-module-size_s-xb_uK styles-module-size_s-compensated-2eWFk styles-module-size_s-_iYhI yYyl
    if description is not None:
        description = description.text
    else:
        description = "No info"
    lst_price.append(price)
    lst_info.append(info)
    lst_description.append(description)
    count += 1
df1 = pd.DataFrame({"Стоимость": lst_price, "Общее описание": lst_info})
df2 = pd.DataFrame({"Подробное описание": lst_description})
print(df1)
print(df2)
```

Рабочий код

	Стоимость	Общие сведения
0	7 420 000 ₽	BMW X7 3.0 AT, 2021, 31 000 км
1	3 750 000 ₽	BMW 4 серия 2.0 AT, 2017, 95 000 км
2	1 750 000 ₽	BMW X6 M 4.4 AT, 2011, 171 000 км
3	13 500 000 ₽	BMW X7 3.0 AT, 2022, 22 000 км
4	2 430 000 ₽	BMW X1 1.5 AMT, 2020, 118 485 км
5	10 500 000 ₽	BMW X7 3.0 AT, 2021, 64 000 км
6	7 830 000 ₽	BMW X5 3.0 AT, 2019, 50 000 км
7	8 150 000 ₽	BMW X7 3.0 AT, 2019, 48 000 км
8	8 590 000 ₽	BMW X3 2.0 AT, 2024
9	10 430 000 ₽	BMW 5 серия 2.0 AT, 2023
10	от 6 390 000 ₽	BMW 3 серия 2.0 AT, 2024
11	6 750 000 ₽	BMW X7 3.0 AT, 2019, 82 000 км
12	4 580 000 ₽	BMW X3 3.0 AT, 2020, 126 000 км
13	1 750 000 ₽	BMW X1 2.0 AT, 2014, 162 800 км
14	3 750 000 ₽	BMW X6 3.0 AT, 2012, 210 829 км
15	5 700 000 ₽	BMW X3 2.0 AT, 2021, 57 000 км
16	2 300 000 ₽	BMW 5 серия 2.0 AT, 2012, 175 000 км
17	2 650 000 ₽	BMW X2 2.0 AT, 2019, 165 000 км
18	1 345 000 ₽	BMW X1 2.0 AT, 2012, 214 484 км
19	4 650 000 ₽	BMW X3 2.0 AT, 2018, 48 000 км
20	3 990 000 ₽	BMW X6 3.0 AT, 2015, 167 100 км
21	4 800 000 ₽	BMW 6 серия GT 2.0 AT, 2021, 49 000 км
22	6 950 000 ₽	BMW X5 3.0 AT, 2019, 112 648 км

	Подробное описание
0	Автомобиль пригнан из Америки, с минимальными ...
1	Продам нереально крутой авто.\nПродажа только ...
2	Продаю свой автомобиль. В отличном состоянии, ...
3	Продаю X7 в идеальнейшем состоянии. Самая полн...
4	100% гарантия юридической чистоты.\nБез скрыти...
5	Продаю в отличном состоянии свой x7 был на кам...
6	Модель x5 xdrive 30d Sport:\nМашина из Германи...
7	Дилерский автомобиль. Куплен в Рольф Премиум Х...
8	No info
9	No info
10	No info
11	Продаю отличный авто Bmw X-7, был привезен из ...
12	Bmw X3 g01 полный привод 3.0 tdi. Чистый 2020 ...
13	В продаже Bmw X1 xdrive20d!\n2014 год рейстали...
14	Bmw X6 M 50 D 3.0 AT (580 л. С).\n\nM Perform...
15	X3 20D в цвете black sapphire metallic.\nM-Spo...
16	Продам Bmw 525D Xdrive (218 л/с полный привод)...
17	В продаже X2 с ориг пробегом, пригнан из Герма...
18	Продается официальным дилером Omoda, Jaecoo и ...
19	Не пригнан. Куплен в Дилерском центре Краснода...
20	Продается Bmw X6.\nГод выпуска – 2015, из сало...
21	Продаю свой автомобиль Bmw 6 серии 2021 года в...
22	Bmw X5 3.0D G05 Белый перламутр.\n\nM собствен...
23	В продаже bmw 118i 2021 года выпуска. Я явлюс...

Разбор кода

- 1. Импортирование необходимых библиотек:
- 2. Определение списка auto, который будет содержать информацию об автомобилях.
- 3. Создание словаря prox с прокси-серверами для отправки запросов.
- 4. Создание словаря head с заголовками HTTP, включая user-agent.
- 5. Установка URL, который будет использоваться для парсинга объявлений на сайте Avito. Параметры URL включают страницу и радиус поиска.
- 6. Начало цикла while, который будет проходить по нескольким страницам объявлений на сайте Avito.
- 7. Отправка GET запроса к URL страницы с помощью requests.get() и передача заголовков и прокси.
- 8. Парсинг HTML страницы с помощью BeautifulSoup и поиск информации об объявлениях.
- 9. Получение информации об автомобилях и добавление её в список auto.
- 10. Добавление случайной задержки между запросами с помощью time.sleep().
- 11. Проверка наличия данных об автомобилях на странице. Если данных нет, происходит остановка цикла.
- 12. Увеличение номера страницы для следующего запроса.
- 13. Создание списков lst_info, lst_price и lst_description для хранения данных из объявлений.
- 14. Цикл while для обработки каждого элемента из списка auto.
- 15. Для каждого элемента из списка auto происходит поиск информации о цене, общих сведениях и подробном описании.
- 16. Если информация есть, она добавляется в соответствующие списки, иначе добавляется "No info".
- 17. Создание двух DataFrame из списков lst_info, lst_price и lst_description с использованием библиотеки pandas.
- 18. Вывод на экран двух DataFrame с информацией об объявлениях.
- Этот код работает следующим образом: он отправляет запросы на сайт Avito, получает информацию об автомобилях с нескольких страниц, парсит данные и сохраняет их в таблицы, используя библиотеку pandas.

Обход препятствий и обеспечение безопасности

- Навигация по сайту может быть сложной из-за таких препятствий, как honeypot (ловушки для хакеров). Honeypot — это скрытые ссылки, специально разработанные для того, чтобы оставаться незамеченными обычными пользователями, однако быть обнаруженными парсерами и ботами. Эти ссылки часто скрываются с помощью HTML-элементов, для которых установлено значение hidden или none, или маскируются под кнопки, чей цвет соответствует фону страницы. Основная цель внедрения “медовых точек” – выявление и занесение ботов в черный список.
- Простой фрагмент кода, который можно использовать, чтобы попытаться избежать ловушек в Python:

```
for link in soup.select('a'):
    if 'display' in link.get( key: 'style', default: '') and 'none' in link['style']:
        continue
auto_info = soup.find_all( name: "div", class_="iva-item-body-KLUuy")
if auto_info is not []:
    auto.extend(auto_info)
```

Обход препятствий и обеспечение безопасности

- Задержка по времени для снижения нагрузки на сайт

```
        # сделаем задержку по времени между запросами чтобы уменьшить нагрузку на сайт
        time_delay = random.random()
        scaled_time_delay = 1 + (time_delay * (9 - 5))
        time.sleep(scaled_time_delay)
    else:
        print("Больше данных нет")
        break
    file_in.write(res)
```

Тестирование и оптимизация скриптов

- Тестирование программы производилось с помощью парсинга первой страницы сайта и записи ее содержимого в файл. Далее производилась работа с содержимым этого файла и отладка скрипта именно с этим набором входных данных.

Тестирование и оптимизация скриптов

```
import ...

# Получение данных с первой страницы сайта и сохранение их в файл
#auto = []
#prox = {"https": "http://178.218.44.79:3128", "http": "http://72.10.164.178:33135"}
#head = {"Accept": "*/*",
 #       "User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/124.0.0.0 Safari/537.36"}
#url = "https://www.avito.ru/moskva/avtomobili/bmw-A5gBAgICAUTgtg3klyg?cd=1&p=2&radius=3000&searchRadius=3000&s=1"
#get = requests.get(url, headers=head, proxies=prox)
#res = get.text
#with open("/Users/user/Desktop/book3.html", "w") as file_in:
  #file_in.write(res)
auto = []
lst_info = []
lst_price = []
lst_description = []
with open("/Users/user/Desktop/book3.html", "r") as file_read:
    stran = file_read.read()
soup = BeautifulSoup(stran, features="lxml")
auto_info = soup.find_all(name="div", class_="iva-item-body-KLUuy")
auto.extend(auto_info)
for elem in auto:
    price = elem.find("strong", class_="styles-module-root-bLKnd")
    if price is not None:
        price = price.text
    else:
        price = "No info"
    info = elem.find("h3", itemprop="name")
    if info is not None:
        info = info.text
    else:
        info = "No info"
    description = elem.find("p", class_="styles-module-root-YczkZ styles-module-size_s-xb_uK styles-module-size_s_compensated-QmHFs styles-module-size_s-_z7mI styl
    if description is not None:
        description = description.text
    else:
        description = "No info"
    lst_price.append(price)
    lst_info.append(info)
    lst_description.append(description)
df1 = pandas.DataFrame({"Стоимость": lst_price, "Общие сведения": lst_info})
df2 = pandas.DataFrame({"Подробное описание": lst_description})
```

Результаты

- Проект веб-скрапинга сайта Avito был успешно завершен. Были разработаны скрипты для автоматического сбора данных о объявлениях, таких как цена, модель автомобиля, год выпуска, пробег и дополнительная информация. Собранные данные были обработаны и организованы для анализа. Результатом работы стала подробная база данных с информацией о различных автомобилях, представленных на Avito. Полученные данные позволили провести анализ рынка автомобилей, выявить тенденции и сделать выводы о предложениях на платформе. В целом, проект по веб-скрапингу сайта Avito был полезным для извлечения ценной информации и исследования рынка автомобилей.

Рекомендации по дальнейшему развитию проекта

- 1. Улучшить точность скрапинга: Необходимо постоянно мониторить и обновлять скрипты скрапинга, чтобы обеспечить высокую точность сбора данных.
- 2. Добавить функциональность: Рассмотрите возможность добавления функций, таких как автоматическое обновление данных, уведомления о новых объявлениях или фильтрация информации для более удобного анализа.
- 3. Исследование конкурентов: Проводить анализ других площадок с объявлениями о продаже автомобилей для сравнения предложений и цен, что поможет определить конкурентное положение.
- 4. Визуализация данных: Рассмотреть создание дашбордов или отчетов на основе собранных данных для более наглядного и удобного представления информации.
- 5. Расширение сферы применения: Подумать о возможностях использования аналитических данных не только для внутренних целей, но и для предоставления информации сторонним пользователям или компаниям по запросу.