Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «ИНЖЕНЕРНЫЙ БИЗНЕС И МЕНЕДЖМЕНТ»

КАФЕДРА «ПРОМЫШЛЕННАЯ ЛОГИСТИКА» (ИБМ-3)

Отчёт

По домашнему заданию

По дисциплине «Парадигмы и конструкции языков программирования»

Студент ИБМ3-34Б:

Нургалиева А.А.

Преподаватель:

Гапанюк Ю.Е.

2024 г.

Веб-скрейпинг на Python

Введение

Веб-скрейпинг – это процесс автоматического извлечения данных с веб-страниц. Он играет важную роль в анализе данных, исследовательской деятельности, создании контента и других областях. Python, благодаря своей простоте и богатой экосистеме библиотек, является одним из наиболее популярных языков для веб-скрейпинга.

Основные концепции веб-скрейпинга

HTML: Язык разметки гипертекста, используемый для создания веб-страниц. Веб-скрейпинг, по сути, это парсинг HTML-кода для извлечения нужной информации.

HTTP-запросы: Для получения HTML-кода страницы необходимо отправить HTTP-запрос на сервер. Библиотека requests в Python упрощает этот процесс.

Парсинг: Процесс анализа HTML-кода для извлечения данных. Для этого используются парсеры, такие как BeautifulSoup4.

Селекторы: Специальные выражения для поиска элементов в HTML-документе. Они позволяют точно определить, какие данные нужно извлечь.

XPath: Язык для навигации по XML-документам, который также может использоваться для парсинга HTML.

Библиотеки Python для веб-скрейпинга

- requests: Основная библиотека для отправки HTTP-запросов.
- BeautifulSoup4: Популярная библиотека для парсинга HTML и XML.
- Scrapy: Мощный фреймворк для больших проектов веб-скрейпинга.
- Selenium: Используется для работы с динамическими веб-страницами, которые требуют выполнения JavaScript.

Процесс веб-скрейпинга

1. Определение цели: Четко сформулировать, какие данные необходимо извлечь.
2. Анализ структуры веб-страницы: Изучить HTML-код страницы, чтобы понять, как организованы данные.
3. Выбор библиотеки: Выбрать подходящую библиотеку в зависимости от сложности задачи и объема данных.
4. Написание кода: Написать код для отправки HTTP-запроса, парсинга HTML и извлечения данных.
5. Сохранение данных: Сохранить извлеченные данные в удобном формате (CSV, JSON, база данных).

Практический пример с использованием BeautifulSoup4

```
import requests
from bs4 import BeautifulSoup

url = 'https://example.com'
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')

# Находим все заголовки h1
headings = soup.find_all('h1')
for heading in headings:
  print(heading.text)
```

Применение веб-скрейпинга

– Анализ рынка: Сбор данных о ценах, продуктах, конкурентах.
– Мониторинг социальных сетей: Анализ настроений пользователей, отслеживание трендов.
– Научные исследования: Сбор данных для анализа.
– Создание контента: Генерация идей для статей, блогов.
– Разработка веб-приложений: Автоматизация рутинных задач.

– Этические соображения
– Robots.txt: Уважение правил сайта.
– Частота запросов: Не перегружать сервер.
– Личные данные: Не собирать конфиденциальную информацию.
– Авторские права: Соблюдение авторских прав на контент.

Сложности веб-скрейпинга

– Динамический контент: Веб-страницы, которые генерируются JavaScript.
– CAPTCHA: Защита от автоматизированного доступа к сайту.
– Блокировка IP-адресов: Предотвращение чрезмерного использования сайта.

Заключение

Веб-скрейпинг — это мощный инструмент для извлечения данных из интернета. Python, благодаря своей простоте и богатой экосистеме, является идеальным языком для реализации проектов веб-скрейпинга. Однако, при использовании веб-скрейпинга необходимо соблюдать этические нормы и уважать правила сайтов.