

# Парсинг HTML. BeautifulSoup.

Урок 2



# План курса



Инструменты разметки наборов данных.



# Что будет на уроке сегодня

- ★ Как работает веб-скрейпинг?
- 🖈 Законен ли веб-скрейпинг?
- 🖈 Введение в Beautiful Soup.
- 🖈 Парсинг HTML с помощью Beautiful Soup
- 🖈 Скрейпинг веб-страницы





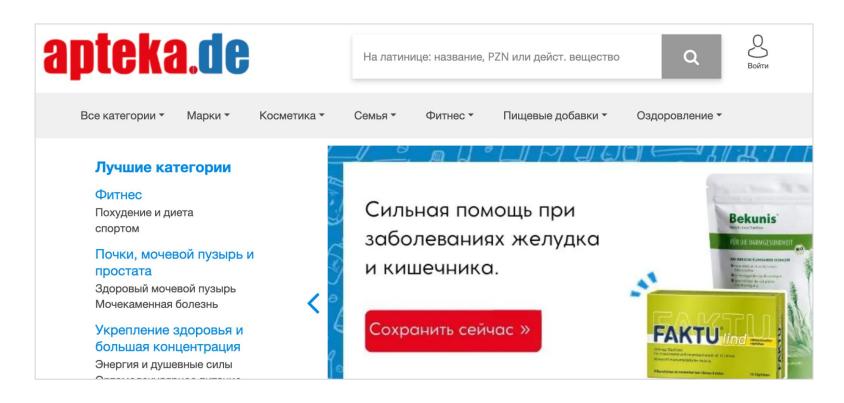


#### Вопрос

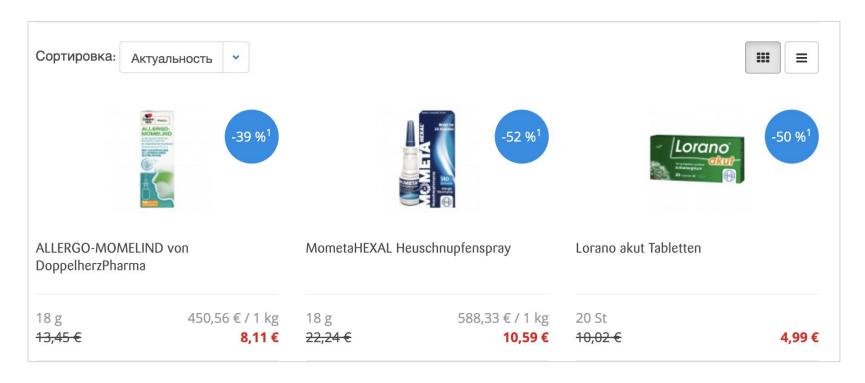
# Какую информацию для анализа можно получить с веб-страниц?



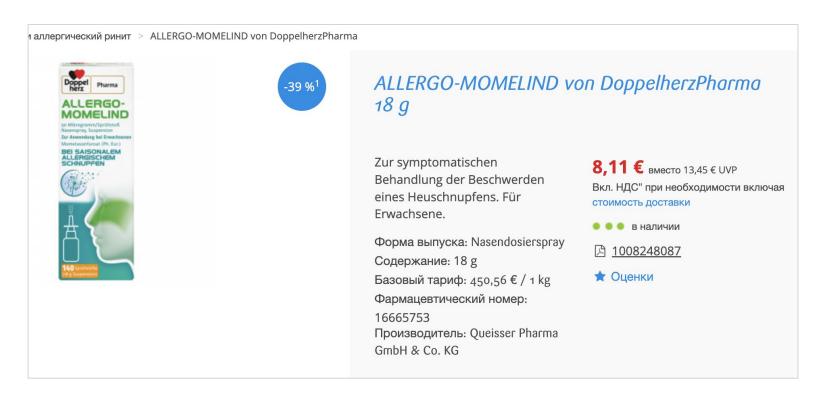
# Пример сайта, содержащего информацию

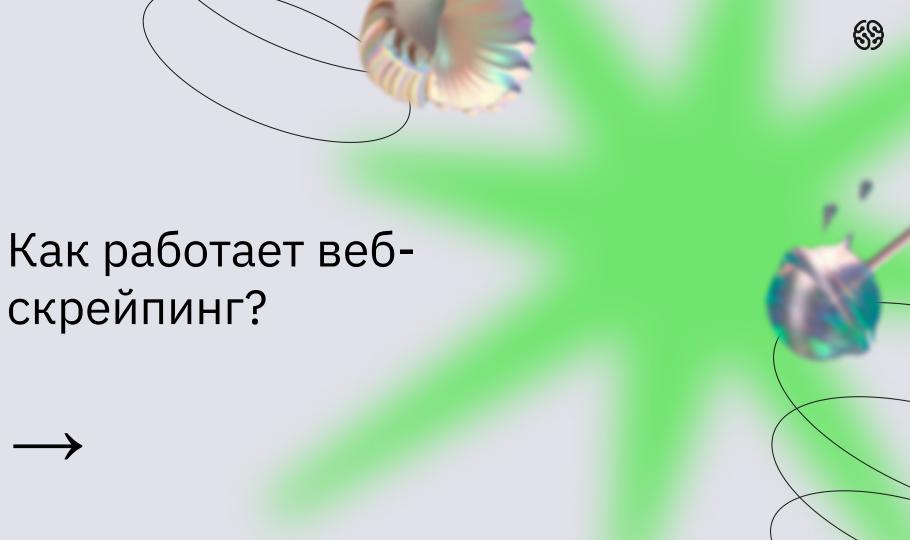


# Пример сайта, содержащего информацию



# Пример сайта, содержащего информацию







# Последовательность операций при веб-скрейпинге

- 🢡 Запросить содержимое (HTML-код) определенного URL с сервера
- 🢡 Загрузить полученное содержимое
- 🢡 Определить элементы страницы, содержащие нужную нам информацию
- 💡 Извлечь и при необходимости выполнить парсинг элементов страницы в датасет



# Правовые нормы, касающиеся веб-скрейпинга

- 1. Правила о гражданско-правовой ответственности и о причинении вреда имуществу.
- 2. Уголовная ответственность за преступления в сфере компьютерной информации.
- 3. Нормы договорного права.
- 4. Право интеллектуальной собственности.
- 5. Правила о персональных данных.



#### Выдержка из Пользовательского соглашения Авито

Без согласия Компании запрещено использовать технические средства для взаимодействия с сервисом в обход обычного порядка использования баз данных и программ для ЭВМ. В том числе запрещено использовать автоматизированные скрипты для сбора информации на Авито, а также для автоматической регистрации профилей.



# Пример файла www.example.com/robots.txt

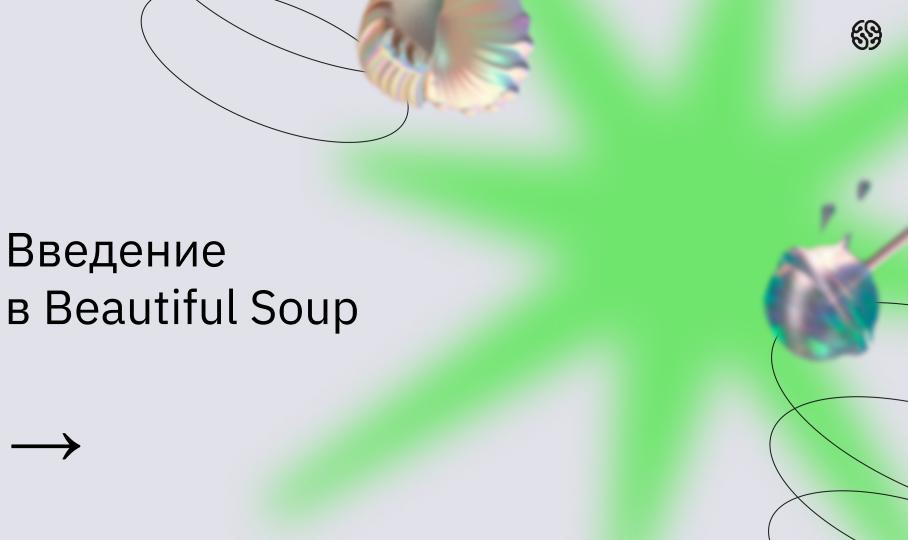
```
User-agent: Googlebot/
Disallow: /nogooglebot/

User-agent: *
Allow: /
Sitemap: http://www.example.com/sitemap.xml
```



# Правила скрейпинга

- 💡 никогда не делайте скрейп чаще, чем это необходимо.
- рассмотрите возможность кэширования содержимого, которое вы скрейпируете, чтобы оно загружалось только один раз.
- встраивайте в код паузы (для этого есть функция time.sleep()), чтобы не перегружать сервер слишком большим количеством запросов.





# Особенности Beautiful Soup

- 🥊 Парсинг документов HTML и XML.
- P Навигация по парсинг-дереву (parse tree).
- 🦞 Доступ к тегам и атрибутам.
- 💡 Изменение HTML-кода.
- 🦿 Поддержка Unicode.





# **Parsing tree**

— это структура данных, которая представляет синтаксическую структуру строки текста в соответствии с правилами формальной грамматики.





# **Parsing tree**

```
<!DOCTYPE html>
<html>
<head>
  <title>Example</title>
</head>
<body>
  <h1>Welcome to my website!</h1>
  This is a paragraph of text.
</body>
</html>
```



# Parsing tree

```
from bs4 import BeautifulSoup
import requests
url = "https://example.com"
response = requests.get(url)
html = response.content
soup = BeautifulSoup(html, 'html.parser')
print(soup.title.string)
```



### Методы Beautiful Soup

```
find()
a = soup.find('a')
find_all()
Доступ к атрибутам.
link = soup.find('a')
href = link['href']
get()
link = soup.find('a')
href = link.get('href', None)
string и text
link = soup.find('a')
text = link.text
```



#### Методы Beautiful Soup

```
contents
html = soup.find('html')
children = html.contents
descendants
html = soup.find('html')
descendants = html.descendants
parent и parents
link = soup.find('a')
href = link['href']
next_sibling и previous_sibling
find_all() и find()
div = soup.find('div')
links = div.find all('a')
```



```
<div>
    This is a <b>bold</b> statement.

        Item 1
        Item 2

        Item 2

        </div>
```