Институт открытых программ дополнительного образования ВШЭ

Python: Инструментальные средства для автоматизации и анализа данных. Часть 2.

Автор курса: Дмитрий Румянцев

Лекция 11

Введение в DOM. Сбор данных: web-scraping с BeautifulSoup

конспект

Москва 2023 г.

Краткая справка

HTML: HyperText Markup Language – язык гипертекстовой разметки.

Основу HTML составляет понятие так называемого **тега** – маркера, специальным образом выделяющего фрагменты текста, и информирующего веб-браузер, что этот фрагмент имеет определенное отличие от прочего текста.

Тег в HTML представляет из себя специальное служебное слово или, иногда, последовательность из нескольких слов. Тег заключается в угловые скобки.

Большинство тегов являются парами в виде ***открывающего*** и ***закрывающего*** тега. Закрывающий тег представляет из себя тег, у которого на втором месте сразу после первой угловой скобки стоит правый слеш. Тег, который не имеет закрывающей пары, называется монотегом.

Парные теги: <HTML>, <HEAD>, <TITLE>, <SCRIPT>, <STYLE>, <BODY>, <H*xx*>, <P>, <DIV>, <A>, <DIV> и др.

Монотеги: <META>, <LINK>, <IMG>,

Любой тег может иметь вложенные теги и сам, в свою очередь, может быть вложен в другие теги. Такая иерархия обычно определяется парадигмой «родитель–потомок».

Для парсинга наибольшие интерес представляет информация, вложенная в парный тег <BODY>, потому что именно внутри этого тега находится информация, которая показывается в браузере. Информация, расположенная внутри тега <HEAD>, является в основном настроечной.

Любой тег может иметь дополнительные настроечные аргументы.

Наиболее часто используются аргументы: **class**, **id**, **name**.

Значения аргумента class в пределах одного документа может быть одинаковыми у разных тегов (например, у тега DIV).

Значение аргумента id любого тега в рамках документа всегда уникально. Поэтому этот атрибут удобно использовать для поиска элемента.

На основании описания тегов браузер, загружая документ, формирует структуру, под названием Объектная модель документа – Document Object Model

или, сокращенно, DOM. DOM используется для получения элементов документа при парсинге.

Библиотека BS4

Официальная документация:

https://beautiful-soup-4.readthedocs.io/en/latest/

Пакет

```
bs4
```

Класс

```
BeautifulSoup4
```

Подключение

```
from bs4 import BeautifulSoup4
```

Особенностью BeautifulSoup является то, что его ядром является т.н. *парсер*. **Парсер** – это анализатор теста (от англ. parse – анализ, разбор). BeautifulSoup может работать с разными парсерами. По умолчанию будет использоваться собственный парсер: **html.parser**.

Парсер на основании текста HTML документа формирует DOM, которая затем используется для получения элементов.

Вызов:

```
dom = bs4.BeautifulSoup(html, 'html.parser')
```

Здесь: html – строковая переменная, содержащая HTML-текст, 'html.parser' – имя используемого парсера.

Возвращаемое значение: объект для работы с DOM.

Методы:

Получить заголовок документа в виде тега

```
dom.title
```

Получить заголовок документа в виде строкового значения

```
dom.title.string
```

Получить все элементы

```
dom.find_all(name=None, attrs={}, recursive=True, string=None,
                 limit=None, **kwargs)
```

Параметры:

name: фильтр по имени тега. Может быть:

- строкой с именем HTML-тега,
- списком с несколькими именами HTML-тегов,
- объектом регулярного выражения,
- значением bool,
- функцией.

attrs: справочник значений атрибутов вида `{"id":"id1"}`

recursive: при значении True осуществляется рекурсивный поиск всех дочерних элементов с указанными параметрами, в противном случае ищутся только прямые потомки.

string: ищет совпадение в тексте HTML-документа, а не в тегах.

limit: ограничение на поиск результатов (не более целого числа limit).

****kwargs**: фильтр по атрибутам HTML-тега. Принимаемые значения такие же, как у аргумента name.

Возвращает:

Список объектов HTML-документа или пустой список [].

Примеры:

```
dom.find_all(class_='stop23')
```

Ищет все элементы, имеющие атрибут класса *stop23*. Для поиска класса надо использовать слово **class_** с символом подчеркивания на конце.

```
dom.find_all('DIV', class_='stop23')
```

Ищет все теги DIV класса *stop23*.

```
dom.find_all('DIV', limit=3, class_='stop23')
```

Ищет не более 3 тегов DIV класса *stop23*.

Получить элемент name

```
dom.find(name)
```

Получить все элементы name

```
dom.findAll(name)
```

Вызывает find_all(); имеет тот же набор параметров.

Получить следующий элемент

```
dom.findNext
```

Получить элемент-потомок

```
dom.findChild
```

Получить элемент-родитель

```
dom.findParent
```

У полученных при помощи функций семейства findXXX элементов можно выделить атрибуты при помощи

```
element.attrs
```

Список атрибутов возвращается в виде справочника, в котором ключом элемента является имя атрибута.

Значение атрибута можно получить при помощи метода get:

```
element.get('attr_name')
```

Итератор всех элементов DOM

```
dom.recursiveChildGenerator()
```

Пример просмотра в цикле всех элементов DOM:

```
for child in dom.recursiveChildGenerator():
```

Тогда в цикле можно использовать метод child.name для получения имени тега и child.attrs для получения его атрибутов.

Если очередной элемент child не является тегом, child.name возвращает значением None. Таким образом можно проверять полученный элемент – тег это или нет.

Библиотека urllib.request

Официальная документация:

https://docs.python.org/3/library/urllib.request.html

Загрузить HTML кода по URL

```
urlopen(url).read()
```

Возвращает документ. Перед дальнейшим парсингом его надо преобразовать в строковый тип:

```
html_code = str(urlopen(url).read())
```

Иногда может возникнуть необходимость поменять кодировку при преобразовании:

```
html_code = str(urlopen(url).read(), 'utf-8')
```

```
html_code = str(urlopen(url).read(), 'windows-1251')
```