**Модуль 9: Web-scraping**

**Заняття 2: Фреймворк Scrapy**

**Фреймворк Scrapy**

Для швидкого збору даних з однієї сторінки **BeautifulSoup** підходить на 100%. Але якщо вам потрібно зібрати дані на кількох сторінках, пройтися сторінками з пагінацією, то таке завдання значно ускладнює скрипт. Адже на кожній сторінці вам потрібно буде знайти ще й потрібне посилання, і потім зробити запит на це посилання. Крім того, менеджмент аутентифікації та/або авторизації знаходиться в зоні відповідальності розробника.

Останній недолік зв'язки **BeautifulSoup** та **requests**— це складність налаштування "ощадливого" [скрапінгу](https://en.wikipedia.org/wiki/Web_scraping) (https://en.wikipedia.org/wiki/Web\_scraping). Коли потрібно зібрати дійсно багато даних із тисяч або десятків тисяч сторінок, ваш скрипт створить занадто велике та небажане навантаження на сервер. В результаті і, швидше за все, всі запити до цього сервера з вашої адреси будуть заблоковані. Щоб уникнути цього, потрібно робити запити з деяким інтервалом, немов імітуючи поведінку реального користувача. Реалізувати цей механізм може бути складним завданням.

Вирішення зазначених недоліків надає асинхронний фреймворк збору даних [**Scrapy**](https://scrapy.org/) (<https://www.edu.goit.global/uk/learn/15249151/10926565/10926654/> training?blockI d=20077140)**.**

Основні можливості **Scrapy**:

* асинхронність (значне зростання продуктивності);
* автоматична робота із заголовками запитів та куками;
* "ощадливий" скрапінг;
* імітація браузера зазначеної версії у заголовках запиту;
* інструменти для побудови складних багаторівневих схем обробки зібраних даних (очищення, валідація, форматування, збереження в базу даних тощо);
* власний інструмент тестування.

Scrapy надає потужний фреймворк для витягування, обробки та зберігання даних.

Встановлення:

Scrapy використовує Spiders - набір автономних сканерів з певним набором інструкцій. За допомогою фреймворку легко розробити навіть великі проекти для скрапінгу так, щоб інші розробники могли використовувати цей код.

Встановлення:

pip install Scrapy

або

poetry add Scrapy

Scrapy надає оболонку веб-сканера Scrapy Shell, яку розробники можуть використовувати для перевірки своїх припущень щодо поведінки сайту. Щоб зайти в консоль **Scrapy**, виконайте у консолі: scrapy shell. Далі необхідно запустити парсер на сторінці за допомогою команди **fetch** в оболонці. Перед цим краще зберегти адресу сторінки, яку будемо парсити в змінну: url = 'https://quotes.toscrape.com/'.

Виконавши **fetch(url)**,

>>> url = 'https://quotes.toscrape.com/'

>>> fetch(url)

в консолі буде щось подібне:

2021-07-24 07:40:13 [scrapy.core.engine] INFO: Spider opened

2021-07-24 07:40:14 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://quotes.toscrape.com/> (referer: None)

Парсер повертає response (відповідь), яку можна подивитися за допомогою команди view(response). А сторінка відкриється в браузері за замовчуванням. За допомогою команди print(response.text) можна переглянути сирий HTML.

**Використання XPath для витягування**

XPath — це мова запитів для вибору вузлів у документах типу XML.

Орієнтуватися за документом можна за допомогою XPath. Scrapy використовує цю мову для роботи з об'єктами документа HTML.

Краще ознайомитися з детальною документацією щодо використання XML парсера в Scrapy [на сторінці документації](https://docs.scrapy.org/en/latest/topics/selectors.html?highlight=xpath) (<https://docs.scrapy.org/en/latest/topics/> selectors.html?highlight=xpath).

Щоб отримати список всіх авторів цитат на сторінці, можна виконати:

>>> fetch(url)

2022-10-02 00:33:26 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://quotes.toscrape.com/> (referer: None)

>>> response

<200 https://quotes.toscrape.com/>

>>> response.xpath("/html//small").extract()

Результатом буде список:

['<small class="author" itemprop="author">Albert Einstein</small>',

'<small class="author" itemprop="author">J.K. Rowling</small>',

'<small class="author" itemprop="author">Albert Einstein</small>',

'<small class="author" itemprop="author">Jane Austen</small>',

'<small class="author" itemprop="author">Marilyn Monroe</small>',

'<small class="author" itemprop="author">Albert Einstein</small>',

'<small class="author" itemprop="author">André Gide</small>',

'<small class="author" itemprop="author">Thomas A. Edison</small>',

'<small class="author" itemprop="author">Eleanor Roosevelt</small>',

'<small class="author" itemprop="author">Steve Martin</small>']

Щоб отримати тільки текст з тегів, потрібно у вираз **Xpath** додати /text():

response.xpath("/html//small[@class='author']/text()").extract()

У результуючому списку будуть лише імена авторів:

['Albert Einstein',

'J.K. Rowling',

'Albert Einstein',

'Jane Austen',

'Marilyn Monroe',

'Albert Einstein',

'André Gide',

'Thomas A. Edison',

'Eleanor Roosevelt',

'Steve Martin']

**Створення проекту Scrapy**

Щоб створити проект **Scrapy**, виконайте: scrapy startproject test\_spyder. В результаті структура проекту буде приблизно наступна:

.

└── test\_spyder

├── scrapy.cfg

└── test\_spyder

├── \_\_init\_\_.py

├── items.py

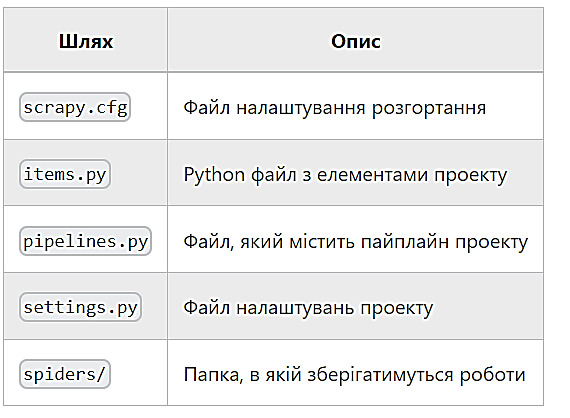
├── middlewares.py

├── pipelines.py

├── settings.py

└── spiders

└── \_\_init\_\_.py



Після створення проекту потрібно перейти в нову папку та написати наступну команду:

scrapy genspider authors quotes.toscrape.com

Команда створить шаблон павука в spyders/authors.py.

**spyders/authors.py**

import scrapy

class AuthorsSpider(scrapy.Spider):

name = 'authors'

allowed\_domains = ['quotes.toscrape.com']

start\_urls = ['http://quotes.toscrape.com/']

def parse(self, response):

pass

* name - це ім'я робота. Вдалі та правильно підібрані імена дозволять простіше відстежувати всіх наявних роботів. Вони повинні бути унікальними, адже саме вони використовуються для запуску командою scrapy crawl name\_of\_spider.
* allowed\_domains (опціонально) – список дозволених для парсингу доменів. Запит до URL-адрес, не зазначених у цьому списку, не буде виконано. Він повинен включати лише домен сайту (наприклад, aliexpress.com), а не цілу URL, вказану в start\_urls, інакше виникнуть помилки.
* start\_urls - запит до згаданих URL-адрес. З них робот почне здійснювати пошук, якщо конкретна URL-адреса не вказана. Першими завантаженими сторінками будуть ті, що вказані тут. Наступні запити будуть генеруватися послідовно з даних, збережених у початкових URL-адресах.
* parse – ця функція викликається, коли парсинг URL успішно виконано. Її ще називають функцією зворотного виклику. Response (використовується в оболонці Scrapy) повертається як результат парсингу, передається цій функції, а в ній знаходиться код для витягування.

Метод parse повинен повертати:

* або об'єкт Request, який є новим запитом і результат якого потрапить на обробку в parse знову;
* або об'єкт Item, який піде в обробку та зберігання;
* або словник, який веде себе аналогічно **Item.**

Scrapy використовує Tornado фреймворк і всі методи, які виконуються асинхронно, повинні бути генераторами, parse один з таких методів. Scrapy за типом результату, що повертається, визначає, що з ним потрібно зробити.

Щоб parse був генератором, потрібно повертати результати з нього, використовуючи оператор yield.

**Парсинг HTML за допомогою XPath**

Використовуючи браузер, можна визначити, що всі блоки коду з цитатами знаходяться всередині тегу div з класом quote. Тоді пройдемо в циклі по всіх блоках на сторінці, що відповідає цьому виразу Xpath:

for quote in response.xpath("/html//div[@class='quote']"):

...

Всередині кожного блоку ключові слова знаходяться всередині тегу посилання а, який сам знаходиться всередині тегу div з класом tags. Тоді ключові слова можна отримати, використовуючи Xpath:

for quote in response.xpath("/html//div[@class='quote']"):

keywords = quote.xpath("div[@class='tags']/a/text()").extract()

Автори цитат знаходяться всередині тегу small, який, у свою чергу, знаходиться всередині тегу span, а текст цитати — всередині тегу span з класом text. Повернімо словник із зібраною інформацією

import scrapy

class AuthorsSpider(scrapy.Spider):

name = 'authors'

allowed\_domains = ['quotes.toscrape.com']

start\_urls = ['http://quotes.toscrape.com/']

def parse(self, response):

for quote in response.xpath("/html//div[@class='quote']"):

yield {

"keywords": quote.xpath("div[@class='tags']/a/text()").extract(),

"author": quote.xpath("span/small/text()").extract(),

"quote": quote.xpath("span[@class='text']/text()").get()

}

Результат можна побачити в консолі, для цього запустимо павука командою: scrapy crawl authors.

Виведення буде великим, та й не надто інформативним. Тому краще зберігати результати скрапінгу у файл.

Збережемо результат збору в табличний документ CSV. Для цього у settings.py додамо два налаштування:

FEED\_FORMAT="csv"

FEED\_URI="result.csv"

Тепер результати скрапінгу зберігатимуться у файлі result.csv.

keywords,author,quote

"change,deep-thoughts,thinking,world",Albert Einstein,“The world as we have created it is a process of our thinking. It cannot be changed without changing our thinking.”

"abilities,choices",J.K. Rowling,"“It is our choices, Harry, that show what we truly are, far more than our abilities.”"

"inspirational,life,live,miracle,miracles",Albert Einstein,“There are only two ways to live your life. One is as though nothing is a miracle. The other is as though everything is a miracle.”

"aliteracy,books,classic,humor",Jane Austen,"“The person, be it gentleman or lady, who has not pleasure in a good novel, must be intolerably stupid.”"

"be-yourself,inspirational",Marilyn Monroe,"“Imperfection is beauty, madness is genius and it's better to be absolutely ridiculous than absolutely boring.”"

"adulthood,success,value",Albert Einstein,“Try not to become a man of success. Rather become a man of value.”

"life,love",André Gide,“It is better to be hated for what you are than to be loved for what you are not.”

"edison,failure,inspirational,paraphrased",Thomas A. Edison,"“I have not failed. I've just found 10,000 ways that won't work.”"

misattributed-eleanor-roosevelt,Eleanor Roosevelt,“A woman is like a tea bag; you never know how strong it is until it's in hot water.”

"humor,obvious,simile",Steve Martin,"“A day without sunshine is like, you know, night.”"

**Пагінація**

Метод parse може повертати Request об'єкт і тоді **Scrapy** зробить запит і обробить його, передавши відповідь у parse. Подивившись код сторінки, можна визначити, що посилання на наступну сторінку знаходиться під тегом li з класом next. Всі посилання поміщаються в атрибут href тегу а. Тоді вираз для отримання посилання на наступну сторінку буде таким: response.xpath("//li[@class='next']/a/@href").get().

Це буде не повне посилання, а лише відносний шлях, повний шлях ми отримаємо, склавши домен і відносний шлях self.start\_urls[0] + next\_link.

import scrapy

class AuthorsSpider(scrapy.Spider):

name = 'authors'

allowed\_domains = ['quotes.toscrape.com']

start\_urls = ['http://quotes.toscrape.com']

def parse(self, response):

for quote in response.xpath("/html//div[@class='quote']"):

yield {

"keywords": quote.xpath("div[@class='tags']/a/text()").extract(),

"author": quote.xpath("span/small/text()").extract(),

"quote": quote.xpath("span[@class='text']/text()").get()

}

next\_link = response.xpath("//li[@class='next']/a/@href").get()

if next\_link:

yield scrapy.Request(url=self.start\_urls[0] + next\_link)

Коли павук дійде до останньої сторінки пагінації, не буде кнопки next, і наступний запит не буде згенерований.

Після виконання команди scrapy crawl authors у файл result.csv додадуться результати нового скрапінгу з пагінацією.

Тут потрібно уточнити кілька моментів щод0о живого прикладу. Структура проекту збігається, але прибрано повернення результату у файл result.csv.

Оскільки розмір прикладу обмежений 1024 Мб і кожне виконання прикладу збільшуватиме файл result.csv. Друге, оскільки приклад може виконуватися тільки запуском файлу main.py в системі replit, то запуск краулера виконаний у вигляді скрипту, а не запуском команди scrapy crawl authors.

**main.py**

import scrapy

from scrapy.crawler import CrawlerProcess

class QuotesSpider(scrapy.Spider):

name = 'authors'

allowed\_domains = ['quotes.toscrape.com']

start\_urls = ['http://quotes.toscrape.com/']

def parse(self, response):

for quote in response.xpath("/html//div[@class='quote']"):

yield {

"keywords": quote.xpath("div[@class='tags']/a/text()").extract(),

"author": quote.xpath("span/small/text()").extract(),

"quote": quote.xpath("span[@class='text']/text()").get()

}

next\_link = response.xpath("//li[@class='next']/a/@href").get()

if next\_link:

yield scrapy.Request(url=self.start\_urls[0] + next\_link)

# run spider

process = CrawlerProcess()

process.crawl(QuotesSpider)

process.start()

INFO

Так, якби ми хотіли через скрипт main.py зберігати результат у файл result.csv, то просто додайте атрибут custom\_settings до класу QuotesSpider:

class QuotesSpider(scrapy.Spider):

name = 'authors'

custom\_settings = {"FEED\_FORMAT": "csv", "FEED\_URI": "result.csv"}

...