Lab 5 – Web Scraping w Pythonie

👤 Autor

Tomasz Królikowski, numer albumu: 153790

Zadanie znajduje się w repozytorium GIT pod adresem:

https://github.com/krolikowski80/studia_WSB/tree/main/Python/intro/zad_5

1. Dziedzina: Web Scraping

Porównanie dwóch popularnych bibliotek do scrapowania danych z internetu: - BeautifulSoup - Scrapy

2. Biblioteki

🔷 BeautifulSoup

- Do prostych zadań, parsowania HTML
- Instalacja: `pip install beautifulsoup4 lxml requests`
- Plusy: prosta, szybka
- Minusy: ręczne przechodzenie po stronach

🔷 Scrapy

- Framework do dużych projektów
- Instalacja: `pip install scrapy`
- Plusy: automatyzacja, eksport danych
- Minusy: wymaga struktury projektu

3. Przykłady (katalog `examples/`)

📘 bs4_example_1.py

Pobiera wszystkie nagłówki `<h2>` ze strony Wikipedia – Web Scraping
Użycie biblioteki `requests` + `BeautifulSoup`.
Działanie: - Pobranie strony HTML - Parsowanie HTML - Wyszukiwanie tagów `<h2>` - Wypisanie ich zawartości tekstowej

📘 bs4_example_2.py

Pobiera wszystkie linki (`<a href=...>`) z tej samej strony Wikipedii
Użycie `BeautifulSoup` do przeszukiwania atrybutów `href`.
Wynik to lista pierwszych 10 linków ze strony.

🕷️ scrapy_example/

Projekt Scrapy z pełną strukturą katalogów.
Zawiera Spidera `quotes_spider.py`, który: - Wchodzi na stronę http://quotes.toscrape.com - Pobiera cytaty i autorów - Przechodzi na kolejne strony (rekurencyjnie) - Eksportuje dane do `quotes.json`

Uruchamiany przez terminal:

```
cd examples/scrapy_example
scrapy crawl quotes -O quotes.json
```

🧪 scrapy_example_advanced.py

Uruchamia Spidera Scrapy bezpośrednio z poziomu Pythona – bez użycia komendy `scrapy crawl`.
Używa `CrawlerProcess` z `scrapy.crawler`.
To sposób na integrację spiderów z innymi aplikacjami lub pipeline'ami.

4. Wnioski

- BeautifulSoup – szybkie skrypty
- Scrapy – duże crawlery

5. Linki

- Dokumentacja BeautifulSoup: https://www.crummy.com/software/BeautifulSoup/bs4/doc/
- Dokumentacja Scrapy: https://docs.scrapy.org/en/latest/