**Модуль 9: Web-scraping**

**Заняття 1: Beautiful Soup**

**Вступ**

**Парсинг** — це найпоширеніший спосіб отримання даних з Інтернету для різного типу застосунків. Практично нескінченна кількість інформації в мережі пояснює факт існування різноманітних інструментів для її збирання. У процесі скрапінгу комп'ютер відправляє запит, у відповідь на який отримує HTML-документ. Після цього починається етап парсингу. Тут уже можна зосередитись лише на тих даних, які потрібні. Розглянемо парсинг із використанням таких бібліотек, як Beautiful Soup, Ixml та Requests.

**Основи Beautiful Soup**

Встановлення необхідних пакетів:

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-09/module-09-01/bfs#%D0%B2%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F-%D0%BD%D0%B5%D0%BE%D0%B1%D1%85%D1%96%D0%B4%D0%BD%D0%B8%D1%85-%D0%BF%D0%B0%D0%BA%D0%B5%D1%82%D1%96%D0%B2)pip install lxml

pip install requests

pip install beautifulsoup4

**З чого почати скрапінг**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-09/module-09-01/bfs#%D0%B7-%D1%87%D0%BE%D0%B3%D0%BE-%D0%BF%D0%BE%D1%87%D0%B0%D1%82%D0%B8-%D1%81%D0%BA%D1%80%D0%B0%D0%BF%D1%96%D0%BD%D0%B3)Для знайомства з процесом скрапінгу можна скористатися сайтом <https://quotes.toscrape.com/> (https://quotes.toscrape.com/), який, схоже, був створений для цього. З нього можна було б створити, наприклад, сховище імен авторів, тегів або самих цитат. У будь-якому сучасному браузері код сторінки можна переглянути, клацнувши правою кнопкою на сторінці та натиснувши «Перегляд коду сторінки». На екрані буде виведено сиру HTML-розмітку сторінки. Наприклад таку:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

....

</div>

</footer>

</body>

</html>

У цьому прикладі можна побачити, що розмітка включає масу, здавалося б перемішаних даних. Завдання веб-скрапінгу – отримання доступу до тих частин сторінки, які потрібні.

**Автоматизація отримання "сирих" даних**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-09/module-09-01/bfs#%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D0%B7%D0%B0%D1%86%D1%96%D1%8F-%D0%BE%D1%82%D1%80%D0%B8%D0%BC%D0%B0%D0%BD%D0%BD%D1%8F-%D1%81%D0%B8%D1%80%D0%B8%D1%85-%D0%B4%D0%B0%D0%BD%D0%B8%D1%85)

Для початку імпортуємо потрібні пакети, зробимо запит за допомогою **requests**, у відповідь отримаємо HTML сторінки, що шукаємо. Далі перетворимо відповідь від сервера у формат **lxml** і передамо результат на обробку**BeautifulSoup.**

import requests

from bs4 import BeautifulSoup

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

response = requests.get(url)

soup = BeautifulSoup(response.text, 'lxml')

print(soup)

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<title>Quotes to Scrape</title>

<link href="/static/bootstrap.min.css" rel="stylesheet" />

<link href="/static/main.css" rel="stylesheet" />

</head>

<body>

<div class="container">

<div class="row header-box">

<div class="col-md-8">

<h1>

<a href="/" style="text-decoration: none">Quotes to Scrape</a>

...

</h1>

</div>

</div>

</div>

</body>

</html>

У консолі побачите код сторінки сайту без відступів, але розбитий за тегами так, що кожен тег на окремому рядку.

**Lxml** використовувати не обов'язково, **BeautifulSoup** вміє працювати і з "чистим" HTML, але бібліотека Lxml може трохи прискорити обробку.

Далі потрібно у браузері знайти деякий шаблон, за яким ви зможете отримати лише потрібні дані.

**Парсинг HTML, пошук на сторінці за тегом, за класом, за ID**

HTML-документ зберігає багато інформації, але, завдяки Beautiful Soup, простіше знаходити потрібні дані. Часом для цього потрібно лише один рядок коду. Спробуємо знайти всі теги span з класом text. Це, в свою чергу, поверне всі теги. Коли потрібно знайти кілька однакових тегів, варто використати функцію **find\_all().**

import requests

from bs4 import BeautifulSoup

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

response = requests.get(url)

soup = BeautifulSoup(response.text, 'lxml')

quotes = soup.find\_all('span', class\_='text')

print(quotes)

[<span class="text" itemprop="text">“The world as we have created it is a process of our thinking. It cannot be changed without changing our thinking.”</span>, <span class="text" itemprop="text">“It is our choices, Harry, that show what we truly are, far more than our abilities.”</span>, <span class="text" itemprop="text">“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.”</span>, <span class="text" itemprop="text">“The person, be it gentleman or lady, who has not pleasure in a good novel, must be intolerably stupid.”</span>, <span class="text" itemprop="text">“Imperfection is beauty, madness is genius and it's better to be absolutely ridiculous than absolutely boring.”</span>, <span class="text" itemprop="text">“Try not to become a man of success. Rather become a man of value.”</span>, <span class="text" itemprop="text">“It is better to be hated for what you are than to be loved for what you are not.”</span>, <span class="text" itemprop="text">“I have not failed. I've just found 10,000 ways that won't work.”</span>, <span class="text" itemprop="text">“A woman is like a tea bag; you never know how strong it is until it's in hot water.”</span>, <span class="text" itemprop="text">“A day without sunshine is like, you know, night.”</span>]

Розмітка, що повертається, — це не зовсім те, що потрібно. Для отримання лише даних - цитат у цьому випадку, можна використовувати властивість **.text.**

import requests

from bs4 import BeautifulSoup

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

response = requests.get(url)

soup = BeautifulSoup(response.text, 'lxml')

quotes = soup.find\_all('span', class\_='text')

for quote in quotes:

print(quote.text)

“The world as we have created it is a process of our thinking. It cannot be changed without changing our thinking.”

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

“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.”

“The person, be it gentleman or lady, who has not pleasure in a good novel, must be intolerably stupid.”

“Imperfection is beauty, madness is genius and it's better to be absolutely ridiculous than absolutely boring.”

“Try not to become a man of success. Rather become a man of value.”

“It is better to be hated for what you are than to be loved for what you are not.”

“I have not failed. I've just found 10,000 ways that won't work.”

“A woman is like a tea bag; you never know how strong it is until it's in hot water.”

“A day without sunshine is like, you know, night.”

Для пошуку та виведення всіх авторів працюємо за тим самим принципом — спершу потрібно вручну вивчити сторінку. Можна звернути увагу на те, що кожен автор взятий в тег <small> з класом **author**. Далі використовуємо функцію **find\_all()** та зберігаємо результат у змінній **authors.**

import requests

from bs4 import BeautifulSoup

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

response = requests.get(url)

soup = BeautifulSoup(response.text, 'lxml')

quotes = soup.find\_all('small', class\_='author')

for quote in quotes:

print(quote.text)

Albert Einstein

J.K. Rowling

Albert Einstein

Jane Austen

Marilyn Monroe

Albert Einstein

André Gide

Thomas A. Edison

Eleanor Roosevelt

Steve Martin

Додамо код отримання всіх тегів для кожної цитати. Спочатку потрібно отримати кожен зовнішній блок кожної колекції тегів. Якщо цей перший крок не виконати, теги можна буде отримати, але асоціювати їх із конкретною цитатою — ні.

Коли блок отримано, можна опускатись нижче за допомогою функції **find\_all** для отриманої підмножини. А далі буде потрібно додати внутрішній цикл для завершення процесу.

import requests

from bs4 import BeautifulSoup

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

response = requests.get(url)

soup = BeautifulSoup(response.text, 'lxml')

quotes = soup.find\_all('span', class\_='text')

authors = soup.find\_all('small', class\_='author')

tags = soup.find\_all('div', class\_='tags')

for i in range(0, len(quotes)):

print(quotes[i].text)

print('--' + authors[i].text)

tagsforquote = tags[i].find\_all('a', class\_='tag')

for tagforquote in tagsforquote:

print(tagforquote.text)

break

“The world as we have created it is a process of our thinking. It cannot be changed without changing our thinking.”

--Albert Einstein

change

deep-thoughts

thinking

world

У цьому прикладі ми вийшли з циклу обробки цитат після першої цитати, використовуючи **break**. Якщо потрібно отримати всі цитати з цієї сторінки, закоментуйте **break.**

INFO

Ще один хороший ресурс для вивчення скрапінгу - http://scrapingclub.com. Там є безліч посібників щодо використання іншого, просунутішого інструменту — Scrapy.

**Пошук елементів**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-09/module-09-01/bfs_02#%D0%BF%D0%BE%D1%88%D1%83%D0%BA-%D0%B5%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%D1%96%D0%B2)Систематизуємо отриману інформацію з пошуку елементів на вебсторінці.

import requests

from bs4 import BeautifulSoup

soup = BeautifulSoup(page.text, "html.parser")

url = "http://quotes.toscrape.com/"

page = requests.get(url)

Як бачимо, Beautiful Soup дає змогу шукати елементи на сторінці за допомогою різних методів. Наприклад, ви можете використовувати метод find для пошуку першого входження елемента на сторінці:

# знайти перший тег <p> на сторінці

first\_paragraph = soup.find("p")

Всередині змінної first\_paragraph буде наступний html елемент:

<p>

<a href="/login">Login</a>

</p>

Ви також можете використовувати вже знайомий метод find\_all для пошуку всіх входжень елемента на сторінці:

# знайти всі теги <p> на сторінці

all\_paragraphs = soup.find\_all("p")

Всередині змінної all\_paragraphs буде список наступних html-елементів:

[<p>

<a href="/login">Login</a>

</p>,

<p class="text-muted">

Quotes by: <a href="https://www.goodreads.com/quotes">GoodReads.com</a>

</p>,

<p class="copyright">

Made with <span class="sh-red">❤</span> by <a href="https://scrapinghub.com">Scrapinghub</a>

</p>]

**Робота з вмістом елементів**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-09/module-09-01/bfs_02#%D1%80%D0%BE%D0%B1%D0%BE%D1%82%D0%B0-%D0%B7-%D0%B2%D0%BC%D1%96%D1%81%D1%82%D0%BE%D0%BC-%D0%B5%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%D1%96%D0%B2)Beautiful Soup дає змогу отримати доступ до вмісту елементів на сторінці. Крім використання атрибута text, ви можете отримати текст елемента за допомогою методу get\_text:

# отримати текст першого тега <p> на сторінці

first\_paragraph\_text = first\_paragraph.get\_text()

print(first\_paragraph\_text.strip()) # 'Login'

Ви також можете отримати доступ до атрибутів елемента за допомогою звичайного синтаксису Python:

# отримати значення атрибута "href" першого тегу <a> на сторінці

first\_link = soup.find("a")

first\_link\_href = first\_link["href"]

print(first\_link\_href) # '/'

**Навігація документом**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-09/module-09-01/bfs_02#%D0%BD%D0%B0%D0%B2%D1%96%D0%B3%D0%B0%D1%86%D1%96%D1%8F-%D0%B4%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%BE%D0%BC)Beautiful Soup дає змогу виконувати навігацію документом і отримати доступ до батьківських і дочірніх елементів.

**Дочірні елементи**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-09/module-09-01/bfs_02#%D0%B4%D0%BE%D1%87%D1%96%D1%80%D0%BD%D1%96-%D0%B5%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%D0%B8)Щоб отримати всі дочірні елементи першого тегу <body> на сторінці, використовуємо атрибут children.

body\_children = list(first\_paragraph.children)

print(body\_children)

Виведення:

['\n', <a href="/login">Login</a>, '\n']

Ми можемо використовувати ланцюжки викликів методів find.

# знайти перший тег <a> всередині першого тегу <div> на сторінці

first\_div = soup.find("div")

first\_div\_link = first\_div.find("a")

print(first\_div\_link)

Виведення:

<a href="/" style="text-decoration: none">Quotes to Scrape</a>

**Батьківські елементи**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-09/module-09-01/bfs_02#%D0%B1%D0%B0%D1%82%D1%8C%D0%BA%D1%96%D0%B2%D1%81%D1%8C%D0%BA%D1%96-%D0%B5%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%D0%B8)Щоб отримати батьківський елемент першого тегу <p> на сторінці, ми можемо використовувати властивість parent

first\_paragraph\_parent = first\_paragraph.parent

print(first\_paragraph\_parent)

Виведення:

<div class="col-md-4">

<p>

<a href="/login">Login</a>

</p>

</div>

Також можна використовувати методи find\_parent і find\_parents для пошуку батьківських елементів:

container = soup.find("div", attrs={"class": "quote"}).find\_parent("div", class\_="col-md-8")

print(container)

Повне виведення ми пропустимо через його великий обсяг:

<div class="col-md-8">

<div class="quote" itemscope="" itemtype="http://schema.org/CreativeWork">

<span class="text" itemprop="text">“The world as we have created it is a process of our thinking. It cannot be changed without changing our thinking.”</span>

<span>by <small class="author" itemprop="author">Albert Einstein</small>

<a href="/author/Albert-Einstein">(about)</a>

</span>

...

**Сусідні елементи**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-09/module-09-01/bfs_02#%D1%81%D1%83%D1%81%D1%96%D0%B4%D0%BD%D1%96-%D0%B5%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%D0%B8)Ви можете отримати доступ до сусідніх елементів за допомогою атрибутів next\_sibling та previous\_sibling.

Наприклад, щоб отримати наступний сусідній елемент першого тегу <span> з класом "tag-item" на сторінці:

next\_sibling = soup.find("span", attrs={"class": "tag-item"}).find\_next\_sibling("span")

print(next\_sibling)

Виведення:

<span class="tag-item">

<a class="tag" href="/tag/inspirational/" style="font-size: 26px">inspirational</a>

</span>

Щоб отримати попередній сусідній елемент першого тегу <span> з класом "tag-item" на сторінці:

previous\_sibling = next\_sibling.find\_previous\_sibling("span")

print(previous\_sibling)

Виведення:

<span class="tag-item">

<a class="tag" href="/tag/love/" style="font-size: 28px">love</a>

</span>

**Пошук за CSS-селекторами**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-09/module-09-01/bfs_02#%D0%BF%D0%BE%D1%88%D1%83%D0%BA-%D0%B7%D0%B0-css-%D1%81%D0%B5%D0%BB%D0%B5%D0%BA%D1%82%D0%BE%D1%80%D0%B0%D0%BC%D0%B8)**Прості селектори**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-09/module-09-01/bfs_02#%D0%BF%D1%80%D0%BE%D1%81%D1%82%D1%96-%D1%81%D0%B5%D0%BB%D0%B5%D0%BA%D1%82%D0%BE%D1%80%D0%B8)Прості селектори шукають елементи за ім'ям тегу, класом або ідентифікатором, за допомогою методу select. Метод select дозволяє шукати елементи на основі [**CSS-селекторів**](https://developer.mozilla.org/ru/docs/Web/CSS/CSS_Selectors) (https://developer.mozilla.org/ru/docs/Web/CSS/CSS\_Selectors)**.** Він приймає рядок із CSS-селектором і повертає всі елементи, що відповідають цьому селектору.

Знайдемо всі теги <p> на сторінці

p = soup.select("p")

print(p)

Виведення:

[<p>

<a href="/login">Login</a>

</p>, <p class="text-muted">

Quotes by: <a href="https://www.goodreads.com/quotes">GoodReads.com</a>

</p>, <p class="copyright">

Made with <span class="sh-red">❤</span> by <a href="https://scrapinghub.com">Scrapinghub</a>

</p>]

Знайдемо всі елементи з класом "text"

text = soup.select(".text")

print(text)

Виведення:

[<span class="text" itemprop="text">“The world as we have created it is a process of our thinking. It cannot be changed without changing our thinking.”</span>,

<span class="text"

itemprop="text">“It is our choices, Harry, that show what we truly are, far more than our abilities.”</span>,

<span class="text" itemprop="text">“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.”</span>,

<span class="text" itemprop="text">“The person, be it gentleman or lady, who has not pleasure in a good novel, must be intolerably stupid.”</span>,

<span class="text" itemprop="text">“Imperfection is beauty, madness is genius and it's better to be absolutely ridiculous than absolutely boring.”</span>,

<span class="text" itemprop="text">“Try not to become a man of success. Rather become a man of value.”</span>, <span

class="text"

itemprop="text">“It is better to be hated for what you are than to be loved for what you are not.”</span>, <span

class="text" itemprop="text">“I have not failed. I've just found 10,000 ways that won't work.”</span>, <span

class="text" itemprop="text">“A woman is like a tea bag; you never know how strong it is until it's in hot water.”</span>,

<span class="text" itemprop="text">“A day without sunshine is like, you know, night.”</span>]

Знайдемо всі елементи з ідентифікатором "header". Ідентифікатор - це спеціальний атрибут тегу id.

header = soup.select("#header")

print(header)

Як бачимо, таких елементів не існує та отримуємо порожній список.

[]

**Комбіновані селектори**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-09/module-09-01/bfs_02#%D0%BA%D0%BE%D0%BC%D0%B1%D1%96%D0%BD%D0%BE%D0%B2%D0%B0%D0%BD%D1%96-%D1%81%D0%B5%D0%BB%D0%B5%D0%BA%D1%82%D0%BE%D1%80%D0%B8)Комбіновані селектори шукають елементи, що відповідають кільком умовам.

Наприклад, знайдемо всі елементи <a> всередині тегу <div> з класом "container":

a = soup.select("div.container a")

print(a)

Виведення:

[<a href="/" style="text-decoration: none">Quotes to Scrape</a>, <a href="/login">Login</a>, <a

href="/author/Albert-Einstein">(about)</a>, <a class="tag" href="/tag/change/page/1/">change</a>, <a class="tag"

href="/tag/deep-thoughts/page/1/">deep-thoughts</a>,

<a class="tag" href="/tag/thinking/page/1/">thinking</a>, <a class="tag" href="/tag/world/page/1/">world</a>, <a

href="/author/J-K-Rowling">(about)</a>, <a class="tag" href="/tag/abilities/page/1/">abilities</a>, <a

class="tag" href="/tag/choices/page/1/">choices</a>, <a href="/author/Albert-Einstein">(about)</a>, <a

class="tag" href="/tag/inspirational/page/1/">inspirational</a>, <a class="tag" href="/tag/life/page/1/">life</a>, <a

class="tag" href="/tag/live/page/1/">live</a>, <a class="tag" href="/tag/miracle/page/1/">miracle</a>, <a

class="tag" href="/tag/miracles/page/1/">miracles</a>, <a href="/author/Jane-Austen">(about)</a>, <a class="tag" href="/tag/aliteracy/page/1/">aliteracy</a>,

<a class="tag" href="/tag/books/page/1/">books</a>, <a class="tag" href="/tag/classic/page/1/">classic</a>, <a

class="tag" href="/tag/humor/page/1/">humor</a>, <a href="/author/Marilyn-Monroe">(about)</a>, <a class="tag"

href="/tag/be-yourself/page/1/">be-yourself</a>,

<a class="tag" href="/tag/inspirational/page/1/">inspirational</a>, <a href="/author/Albert-Einstein">(about)</a>, <a

class="tag" href="/tag/adulthood/page/1/">adulthood</a>, <a class="tag" href="/tag/success/page/1/">success</a>,

<a class="tag" href="/tag/value/page/1/">value</a>, <a href="/author/Andre-Gide">(about)</a>, <a class="tag"

href="/tag/life/page/1/">life</a>,

<a class="tag" href="/tag/love/page/1/">love</a>, <a href="/author/Thomas-A-Edison">(about)</a>, <a class="tag"

href="/tag/edison/page/1/">edison</a>,

<a class="tag" href="/tag/failure/page/1/">failure</a>, <a class="tag"

href="/tag/inspirational/page/1/">inspirational</a>, <a

class="tag" href="/tag/paraphrased/page/1/">paraphrased</a>, <a href="/author/Eleanor-Roosevelt">(about)</a>, <a

class="tag" href="/tag/misattributed-eleanor-roosevelt/page/1/">misattributed-eleanor-roosevelt</a>, <a

href="/author/Steve-Martin">(about)</a>, <a class="tag" href="/tag/humor/page/1/">humor</a>, <a class="tag" href="/tag/obvious/page/1/">obvious</a>,

<a class="tag" href="/tag/simile/page/1/">simile</a>, <a href="/page/2/">Next <span aria-hidden="true">→</span></a>, <a

class="tag" href="/tag/love/" style="font-size: 28px">love</a>, <a class="tag" href="/tag/inspirational/"

style="font-size: 26px">inspirational</a>, <a

class="tag" href="/tag/life/" style="font-size: 26px">life</a>, <a class="tag" href="/tag/humor/"

style="font-size: 24px">humor</a>, <a

class="tag" href="/tag/books/" style="font-size: 22px">books</a>, <a class="tag" href="/tag/reading/"

style="font-size: 14px">reading</a>, <a

class="tag" href="/tag/friendship/" style="font-size: 10px">friendship</a>, <a class="tag" href="/tag/friends/"

style="font-size: 8px">friends</a>,

<a class="tag" href="/tag/truth/" style="font-size: 8px">truth</a>, <a class="tag" href="/tag/simile/"

style="font-size: 6px">simile</a>, <a

href="https://www.goodreads.com/quotes">GoodReads.com</a>, <a href="https://scrapinghub.com">Scrapinghub</a>]

**Атрибути**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-09/module-09-01/bfs_02#%D0%B0%D1%82%D1%80%D0%B8%D0%B1%D1%83%D1%82%D0%B8)Можна шукати елементи за значенням атрибутів. Знайдемо всі елементи, у яких атрибут href починається з "https://"

href = soup.select("[href^='https://']")

print(href)

Виведення:

[<a href="https://www.goodreads.com/quotes">GoodReads.com</a>, <a href="https://scrapinghub.com">Scrapinghub</a>]

Знайдемо всі елементи, у яких атрибут class містить слово "text":

ctext = soup.select("[class\*='text']")

print(ctext)

Виведення:

[<span class="text" itemprop="text">“The world as we have created it is a process of our thinking. It cannot be changed without changing our thinking.”</span>,

<span class="text"

itemprop="text">“It is our choices, Harry, that show what we truly are, far more than our abilities.”</span>,

<span class="text" itemprop="text">“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.”</span>,

<span class="text" itemprop="text">“The person, be it gentleman or lady, who has not pleasure in a good novel, must be intolerably stupid.”</span>,

<span class="text" itemprop="text">“Imperfection is beauty, madness is genius and it's better to be absolutely ridiculous than absolutely boring.”</span>,

<span class="text" itemprop="text">“Try not to become a man of success. Rather become a man of value.”</span>, <span

class="text"

itemprop="text">“It is better to be hated for what you are than to be loved for what you are not.”</span>, <span

class="text" itemprop="text">“I have not failed. I've just found 10,000 ways that won't work.”</span>, <span

class="text" itemprop="text">“A woman is like a tea bag; you never know how strong it is until it's in hot water.”</span>,

<span class="text" itemprop="text">“A day without sunshine is like, you know, night.”</span>, <p class="text-muted">

Quotes by: <a href="https://www.goodreads.com/quotes">GoodReads.com</a>

</p>]

**Підсумок**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-09/module-09-01/bfs_02#%D0%BF%D1%96%D0%B4%D1%81%D1%83%D0%BC%D0%BE%D0%BA)Вибір між методом select і методом find залежить від ваших уподобань, рівня знань CSS-селекторів і вимог до пошуку елементів. Якщо вам зручніше працювати з CSS-селекторами та вам потрібно виконати складні запити, то select буде кращим вибором. Якщо вам потрібно просто знайти перший відповідний елемент, то find може бути зручнішим варіантом.

**Практичний приклад**

**Підготовка проекту**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-09/module-09-01/project_01#%D0%BF%D1%96%D0%B4%D0%B3%D0%BE%D1%82%D0%BE%D0%B2%D0%BA%D0%B0-%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D1%83)У практичному прикладі ми будемо парсити сайт <http://books.toscrape.com/>, який і був створений спеціально для цього.

Насамперед створимо віртуальне оточення[**poetry**](https://python-poetry.org/docs/) (https://python-poetry.org/docs/)**.** Простіше це зробити через IDE **PyCharm.**

Далі встановимо необхідний пакет [Beautiful Soup](https://www.crummy.com/software/BeautifulSoup/) (<https://www.crummy.com/> software/BeautifulSoup/):

poetry add beautifulsoup4

Для **HTTP** запитів будемо використовувати пакет [requests](https://requests.readthedocs.io/en/master/) (https://requests.readthedocs.io/en/latest/).

poetry add requests

А для збереження результатів у базі даних використовуємо [SQLAlchemy](https://www.sqlalchemy.org/) (https://www.sqlalchemy.org/).

poetry add sqlalchemy

**Повний код**

[​](https://textbook.edu.goit.global/python-web-textbook/uk/docs/module-09/module-09-01/project_01#%D0%BF%D0%BE%D0%B2%D0%BD%D0%B8%D0%B9-%D0%BA%D0%BE%D0%B4)Наведемо відразу повний код нашого проекту, а потім розберемо, як він працює. Файл main.py

import requests

from bs4 import BeautifulSoup

from sqlalchemy.engine import create\_engine

from sqlalchemy.orm import sessionmaker

from model import Book, Base

def parse\_data():

rate\_to\_number = {

'One': 1,

'Two': 2,

'Three': 3,

'Four': 4,

'Five': 5

}

url = 'http://books.toscrape.com/'

store\_ = []

html\_doc = requests.get(url)

if html\_doc.status\_code == 200:

soup = BeautifulSoup(html\_doc.content, 'html.parser')

books = soup.select('section')[0].find\_all('article', attrs={'class': 'product\_pod'})

for book in books:

img\_url = f"{url}{book.find('img')['src']}"

rating = rate\_to\_number.get(book.find('p', attrs={'class': 'star-rating'})['class'][1])

title = book.find('h3').find('a')['title']

price = float(book.find('p', attrs={'class': 'price\_color'}).text[1:])

store\_.append({

'img\_url': img\_url,

'rating': rating,

'title': title,

'price': price

})

return store\_

if \_\_name\_\_ == '\_\_main\_\_':

store = parse\_data()

engine = create\_engine("sqlite:///:memory:")

Base.metadata.create\_all(engine)

Base.metadata.bind = engine

Session = sessionmaker(bind=engine)

session = Session()

for el in store:

book = Book(img\_url=el.get('img\_url'), rating=el.get('rating'), title=el.get('title'), price=el.get('price'))

session.add(book)

session.commit()

books = session.query(Book).all()

for b in books:

print(vars(b))

session.close()

Далі ми розберемо функціонування нашого прикладу.

**Розбір моделей**

Спочатку ми імпортуємо необхідні модулі та модель для збереження даних.

Файл для моделі даних model.py має наступну структуру:

from datetime import datetime

from sqlalchemy import Column, Integer, String, Float

from sqlalchemy.ext.declarative import declarative\_base

from sqlalchemy.sql.sqltypes import DateTime

Base = declarative\_base()

class Book(Base):

\_\_tablename\_\_ = "books"

id = Column(Integer, primary\_key=True, autoincrement=True)

img\_url = Column(String(250), nullable=False)

rating = Column(Integer, nullable=False)

title = Column(String(150), nullable=False, unique=True)

price = Column(Float, nullable=False)

created = Column(DateTime, default=datetime.now())

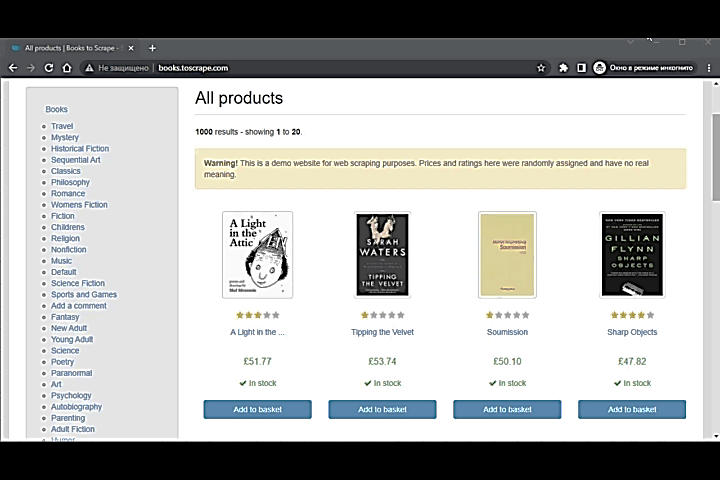
Чим зумовлена саме така структура зберігання даних?

Спочатку подивимося на картку книги.



Ми бачимо зображення - обкладинку картки, назву книги, її рейтинг та ціну.

Далі проаналізуємо **HTML** код картки книги. Клікнемо по зображенню з книгою правою кнопкою миші та викличемо контекстне меню. У нас відкриється інспектор **Chrome** для сторінки і ми можемо переглянути структуру коду картки



Код картки наступний.

<

article class="product\_pod">

<div class="image\_container">

<a href="catalogue/a-light-in-the-attic\_1000/index.html"

><img

src="media/cache/2c/da/2cdad67c44b002e7ead0cc35693c0e8b.jpg"

alt="A Light in the Attic"

class="thumbnail"

/></a>

</div>

<p class="star-rating Three">

<i class="icon-star"></i>

<i class="icon-star"></i>

<i class="icon-star"></i>

<i class="icon-star"></i>

<i class="icon-star"></i>

</p>

<h3>

<a

href="catalogue/a-light-in-the-attic\_1000/index.html"

title="A Light in the Attic"

>A Light in the ...</a

>

</h3>

<div class="product\_price">

<p class="price\_color">£51.77</p>

<p class="instock availability">

<i class="icon-ok"></i>

In stock

</p>

<form>

<button

type="submit"

class="btn btn-primary btn-block"

data-loading-text="Adding..."

>

Add to basket

</button>

</form>

</div>

</article>

Поверхневий аналіз говорить, що обкладинка книги знаходиться в тезі img, а назва книги в полі title у тезі a. Також ми бачимо рейтинг книги в класі class="star-rating Three" тегу p та її ціну в тезі p з класом price\_color.

**Отримуємо дані для книги**

Для цього у нас є функція parse\_data.

Визначаємо url = 'http://books.toscrape.com/' для запиту. Змінну store\_, де зберігатимемо отримані дані. І виконуємо запит до сайту html\_doc=requests.get(url).

Якщо результат вдалий і код відповіді 200, починаємо витягувати дані з HTML коду.

Виконуємо парсинг даних з HTML коду.

soup = BeautifulSoup(html\_doc.content, 'html.parser')

Знаходимо всі книги наступним запитом.

books = soup.select('section')[0].find\_all('article', attrs={'class': 'product\_pod'})

А далі в циклі із картки кожної книги починаємо діставати інформацію.

Обкладинку книги дістаємо з атрибуту src тегу img.

img\_url = f"{url}{book.find('img')['src']}"

Варто зауважити, що шлях до обкладинки книги всередині тегу src - відносний media/cache/2c/da/2cdad67c44b002e7ead0cc35693c0e8b.jpg. Щоб він був функціональним, необхідно перетворити його на абсолютний шлях. Для цього потрібно підставити на початку шляху значення змінної url.

Рейтинг книги знаходиться як назва класу у тезі p.

<p class="star-rating Three">

<i class="icon-star"></i>

<i class="icon-star"></i>

<i class="icon-star"></i>

<i class="icon-star"></i>

<i class="icon-star"></i>

</p>

Але зберігати в базі даних (БД) краще числове значення рейтингу. Для цього ми ввели словник

rate\_to\_number = {

'One': 1,

'Two': 2,

'Three': 3,

'Four': 4,

'Five': 5

}

Який і буде перетворювати ім'я класу в число

rating = rate\_to\_number.get(book.find('p', attrs={'class': 'star-rating'})['class'][1])

Варто зазначити, що вираз book.find('p', attrs={'class': 'star-rating'})['class'] поверне нам список ['star-rating', 'Three'] , де нам потрібний лише другий клас елемента.

Назву книги ми отримуємо з атрибуту title тегу a.

title = book.find('h3').find('a')['title']

Оскільки всередині картки книги два теги a, ми конкретизуємо пошук.

І останнім ми отримуємо ціну книги

price = float(book.find('p', attrs={'class': 'price\_color'}).text.removeprefix('£'))

Видаляємо префікс £ з ціни і приводимо її до числа.

Останній етап - поміщаємо словник із даними книги у список store\_.

store\_.append({

'img\_url': img\_url,

'rating': rating,

'title': title,

'price': price

})

**Поміщаємо дані в БД**

Після виконання всіх дій ми отримуємо список store з даними про книги. Поміщаємо ці дані в БД.

if \_\_name\_\_ == '\_\_main\_\_':

store = parse\_data()

engine = create\_engine("sqlite:///my\_books.db")

Base.metadata.create\_all(engine)

Base.metadata.bind = engine

Session = sessionmaker(bind=engine)

session = Session()

for el in store:

book = Book(img\_url=el.get('img\_url'), rating=el.get('rating'), title=el.get('title'), price=el.get('price'))

session.add(book)

session.commit()

books = session.query(Book).all()

for b in books:

print(vars(b))

session.close()

Якщо ви виконали все без помилок, то отримаєте **SQLite** БД у пам'яті з даними про книги.

Якщо виконати запит до таблиці books у БД, ми отримаємо список книг.

SELECT \* FROM books LIMIT 5

Результат запиту виглядає так:

У цьому практичному прикладі ми виконали парсинг HTML сторінки. Отримали дані з HTML коду та помістили їх у базу даних.