УДК 004.852

Ханенко Олександр Андрійович
студент
Національний технічний університет України «Київський політехнічний інститут»
Ханенко Александр Андреевич
студент
Национальный технический университет Украины «Киевский политехнический институт »
Khanenko O.
student
National Technical University of Ukraine "Kyiv Polytechnic Institute"

ОГЛЯД МОЖЛИВОСТЕЙ PYTHON ЯК ЗАСОБУ ДЛЯ СТВОРЕННЯ ВЕБ-ПАРСЕРА

ОБЗОР ВОЗМОЖНОСТЕЙ PYTHON КАК СРЕДСТВА ДЛЯ СОЗДАНИЯ ВЕБ-ПАРСЕРА

REVIEW CAPABILITIES PYTHON AS A MEANS TO CREATE WEB PARSER

Анотація. *Мета даної роботи – розглянути можливості мови програмування Python для створення веб-парсера. Дана робота зосереджується на виборі оптимального модуля для реалізації програмного продукту.*

Ключові слова: *парсинг, Python, модуль, URL, HTML, LXML, запит, синтаксичний аналіз, веб-сайт.*

Аннотация. *Цель данной работы – рассмотреть возможности языка программирования Python для создания веб-парсера. Данная работа сосредотачивается на выборе оптимального модуля для реализации программного продукта.*

Ключевые слова: *парсинг, Python, модуль, URL, HTML, LXML, запрос, синтаксический анализ, веб-сайт.*

Summary. *The aim of this work is to consider the capabilities Python programming language for creating web parser. This work focuses on the selection of optimal module for software implementation.*

Key words: *parsing, Python, module, URL, HTML, LXML, request, syntactic analysis, website.*

Вступ

Python — високорівнева мова програмування загального призначення, орієнтована на підвищення продуктивності розробника і читання коду. Синтаксис ядра Python мінімалістичний. У той же час стандартна бібліотека включає великий обсяг корисних функцій.

Основні архітектурні риси — динамічна типізація, автоматичне керування пам'яттю, повна інтроспекція, механізм обробки виключень, підтримка багатопоточних обчислень і зручні високорівневі структури даних. Код в Python організовується у функції та класи, які можуть об'єднуватися в модулі (вони в свою чергу можуть бути об'єднані в пакети).

Як наслідок, на цій мові створено немало зарекомендованих бібліотек, серед них і ті, які прекрасно справляються з парсингом веб-сторінок.

Етапи парсинга

Парсинг html-сторінки вдає із себе процес, який можна розбити на три етапи:

1. Отримання початкового коду веб-сторінки — скачати програмний код тієї сторінки сайту, з якої необхідно витягти інформацію.
2. Синтаксичний аналіз html-коду.
3. Фіксація результату (експорт даних).

Завантаження веб-сторінок

Для завантаження веб-сторінок в Python доступні бібліотеки urllib, urllib2, які виконують URL запити, але пропонують різні функціональні можливості:

- urllib2 може приймати об'єкт Request, щоб встановити заголовки для запиту URL, urllib приймає тільки URL.

- urllib надає метод urlencode, який використовується для генерації рядків запиту GET, urllib2 не має такої функції. Це одна з причин того, чому urllib часто використовується разом з urllib2.

При завантаженні веб-сторінок може виникнути багато проблем, що призупинять виконання програми або повернуть некоректні дані. Основні з них наведені нижче:

- Сервер не відповідає.
- Веб-сайт блокує агента користувача робота.
- Доступ до сайту дозволений лише в окремих країнах.
- Веб-сторінки з динамічним контентом.
- Взаємодія з формами.
- Обхід CAPTCHA.

Всі ці проблеми можливо обійти, використовуючи готові модулі в Python та написавши декілька функцій.

У разі помилки 5xx коли сервер не відповідає, потрібно реалізувати функцію, яка буде повторно звертатися до сервера через певний проміжок часу. Потрібно використовувати ідентифіковану назву агента користувача, для того, щоб веб-сайт не розпізнав програми-робота, передавши у функцію `urllib2.Request()` установку `headers = {'User-agent': 'wswp'}`.

Деякі веб-сайти блокують користувачів, які знаходяться за межами даної країни, в такому разі отримати доступ можна через проксі-сервер. Реалізувати підтримку проксі можливо з urllib2.

Існують сторінки з динамічним контентом, коли JavaScript динамічно генерує контент веб-сторінки, це може варіюватися від простих форм до SPA (Single Page Application). Для вирішення цієї проблеми можна використати Selenium WebDriver, — це програмна бібліотека для управління браузерами. За допомогою Selenium можна з легкістю сфільтрувати дані чи виконати пошук по заданих критеріях, загрузити сторінку і дальше аналізувати лише корисні нам дані.

З формами, що відправляють дані до сервера шляхом GET і POST запитів, легко справляється такий модуль як Mechanize. Поля форм є легко доступними, непотрібно управляти куками, Mechanize виконає це самостійно.

Якщо CAPTCHA (рис. 1) прості їх можна легко обійти за допомогою таких модулів як Pillow і Tesseract. В Pillow реалізований клас Image, в якому доступні методи для обробки зображень. Отже перше, що потрібно це забрати фон із зображення. А потім, підключивши модуль Tesseract, за допомогою `pytesseract.image_to_string(img)` перетворити зображення в строку.

У випадку складних CAPTCHA можем скористатись спеціальними онлайн сервісами, що розпізнають CAPTCHA зображення, де працюють реальні люди. Зареєструвавшись, отримуємо особистий ключ до їхньої API, пишем програму, яка буде відправляти дані на сервер та отримувати їх.

Рисунок 1

Аналіз та обробка даних

Три різні підходи в Python для реалізації синтаксичного аналізу:

1) регулярні вирази;
2) BeautifulSoup модуль;
3) lxml модуль.

Регулярні вирази доволі швидко працюють, але їх важко побудувати так як вони нечитабельні. Крім того, якщо розробники допишуть атрибути до тегів, то регулярні вирази прийдеться переписувати.

BeautifulSoup модуль доволі простий в реалізації і зрозумілий, в змозі правильно інтерпретувати відсутні лапки атрибутів і закриваючі теги, а також додати <html> і <body> теги, щоб сформувати повний HTML-документ. Мінус в тому, що він дуже повільно обробляє інформацію.

LXML є Python обгорткою поверх libxml2 XML-бібліотеки, написаної на C, яка допомагає зробити це швидше, ніж Beautiful Soup. Переваги lxml:

- Швидка і гнучка бібліотека для обробки XML/HTML на Python.
- Надає доступ до декларативного синтаксису XPath (рис. 2), зберігаючи при цьому загальну функціональність, доступну в Python
- Простота в написанні коду.
- Висока продуктивність при обробці дуже великих обсягів XML/HTML-даних.

Lxml.etree включає в себе клас ElementTree, що надає метод xpath() для підтримки синтаксису Xpath, а також розширені функції.

Висновок

Для коректної роботи програми-парсера рекомендовано написати універсальні методи для завантаження та обробки HTML-документа. Особливо в функції

Рисунок 2

завантаження врахувати всі можливі перешкоди, що можуть виникнути при спробі отримати веб-сторінку, більш детально ознайомитись з існуючими наведеними вище модулями. Для обробки веб-сторінки найкраще використовувати lxml в Python так як вона одна з найшивдших бібліотек, яка справляється з аналізом даних. В ній реалізовано безліч різноманітних методів, а також вона дозволяє використовувати специфікацію Document Object Model XPath, синтаксис якої досить простий і доступний.

Література

1. Grune D. Parsing Techniques — A Practical Guide / D. Grune, C. Jacobs. — Chichester: Originally published by Ellis Horwood, 1990. — 320 c.

2. Aho A.V. The theory of parsing, translation, and compiling / A.V. Aho, J.D. Ullman. — USA: Prentice-Hall, 1972. — 121 c.

3. Mitchell R. Web Scraping with Python / Ryan Mitchell. — Boston: O\'Reilly Media, 2015. — 256 c.

4. Lxml — XML and HTML with Python [Електронний ресурс] — Режим доступу до ресурсу: http://lxml.de/. — Дата доступу: 28.05.2016.

5. Beautiful Soup 4 Python [Електронний ресурс] — Режим доступу до ресурсу: http://www.pythonforbeginners.com/beautifulsoup/beautifulsoup-4-python. — Дата доступу: 01.06.2016.