УДК 004.432.2+004.4'282

МУЛЬТИПОТОКОВІСТЬ У PYTHON ЯК ІНСТРУМЕНТ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ПРОГРАМ

Людмила Глинчук

MULTITHREADING IN PYTHON AS A TOOL FOR INCREASING PROGRAM EFFICIENCY

Liudmyla Hlynchuk

Волинський національний університет імені Лесі Українки, просп. Волі, 13, Луцьк, 43025, Україна

***Abstract**. This paper explores the efficiency of multithreading in Python for I/O-bound tasks. A comparison between sequential and concurrent implementations is provided using real examples. The results demonstrate significant performance gains with native threading usage.*

Зі зростанням складності програмного забезпечення та необхідністю швидкого виконання завдань, програмісти все частіше звертаються до мультипотокового програмування. Це важливо у задачах, що вимагають паралельної обробки великої кількості даних або взаємодії з кількома зовнішніми джерелами, такими як бази даних, вебсервіси, файли тощо. Python, завдяки своїй простоті й великій кількості бібліотек, є однією з найпопулярніших мов для розробки багатозадачних систем. Мультипотоковість у Python дозволяє реалізовувати конкурентні програми навіть у складних середовищах [1], [2].

Метою дослідження є вивчення можливостей та обмежень мультипотокового програмування в Python, а також оцінка його ефективності на прикладі різних типів задач. Зокрема, ми зосередимось на задачах, пов'язаних із затримками вводу/виводу та задачах, що вимагають значного навантаження на процесор.

Мультипотоковість у Python виявилася ефективним інструментом для розв’язання задач, що включають очікування вводу/виводу, таких як мережеві запити або операції з файлами. Наприклад, у задачі масового завантаження вебсторінок за допомогою модуля threading, можна значно зменшити час виконання за рахунок паралельного завантаження кількох сторінок. Такий підхід дозволяє досягти значної економії часу, оскільки поки один потік чекає відповіді від сервера, інші потоки можуть обробляти інші запити [3].

Код для послідовного виконання, рисунок 1:

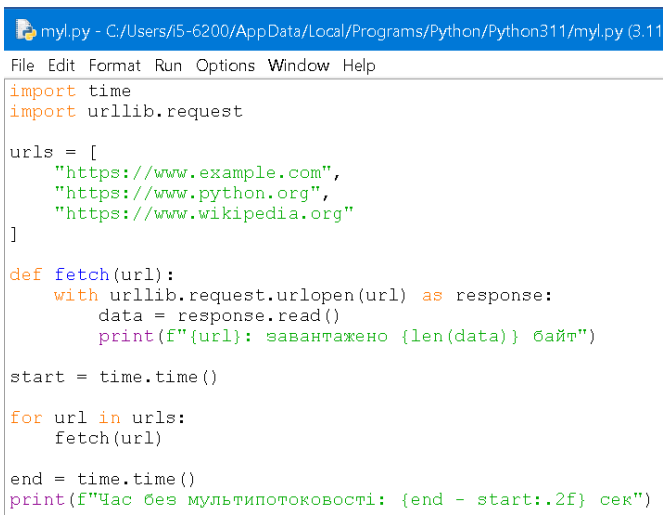

Рисунок 1. Послідовно завантажуємо вебсторінки

Результат роботи програми послідовного виконання: рисунок 2.

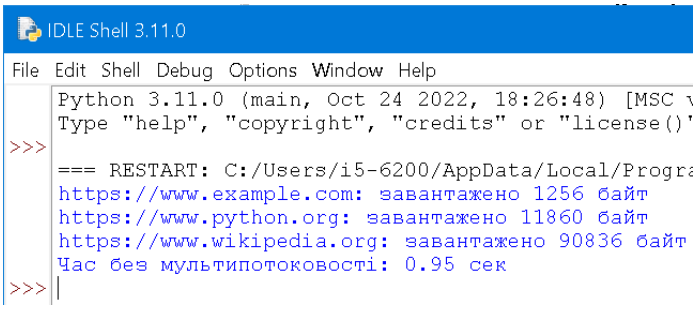

Рисунок 2. Час виконання послідовно (без потоків)

Код для паралельного виконання, тобто в потоках: рисунок 3.

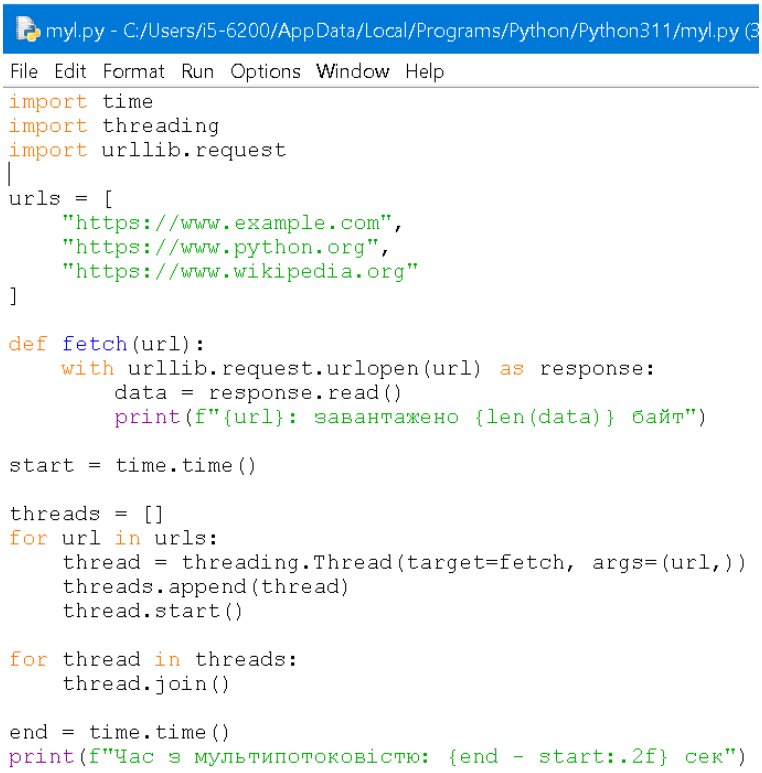

Рисунок 3. Паралельно завантажуємо вебсторінки (мультипотоковість)

Результат роботи програми паралельного виконання: рисунок 4.

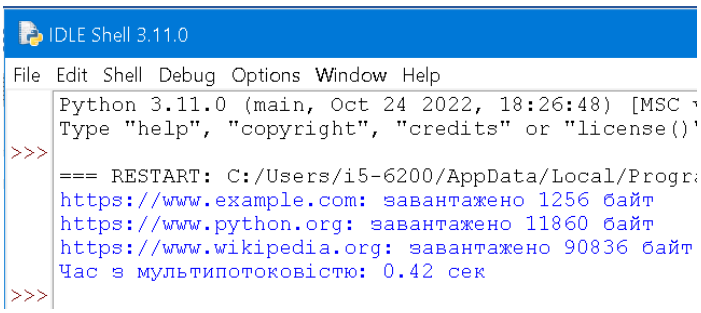

Рисунок 4. Час виконання (мультипотоковість)

Отже, порівнюючи дані, що видали програми, можна зробити наступний висновок: час виконання з мультипотоковістю майже у 2 рази менший ніж при послідовному виконанні.

Ще одним прикладом є задачі з обробки великих обсягів файлів. Використовуючи мультипотоковість, можна одночасно копіювати кілька файлів або обробляти їх паралельно, що дозволяє зменшити час на виконання операцій вводу/виводу. Це особливо корисно в застосунках, які працюють з великою кількістю файлів або даних, наприклад, в системах архівування або резервного копіювання [3].

Проте, при розв'язуванні задач, що вимагають великих обчислювальних ресурсів, мультипотоковість в Python обмежена через наявність глобального блокувальника інтерпретатора (GIL). Це означає, що тільки один потік може виконувати Python-код одночасно, що значно знижує ефективність у задачах, де необхідна інтенсивна обробка даних на процесорі. Для таких випадків більш підходить використання мультипроцесорності, що дозволяє ефективно використовувати всі ядра процесора. Наприклад, обчислення факторіалів чи інших важких математичних задач краще реалізовувати за допомогою multiprocessing [4, 5].

Комбінація потоків і процесів особливо ефективна в складних системах, де є потреба в одночасному обслуговуванні великої кількості підключень і виконанні важких обчислень. У такому випадку потоки можуть обробляти запити користувачів, а важчі задачі (наприклад, аналітика даних, шифрування, перевірка підписів) виконуються в окремих процесах. Такий підхід широко використовується в системах моніторингу, великих розподілених системах, а також в області обробки даних у реальному часі [3], [5].

У представленому дослідженні здійснено порівняльний аналіз ефективності виконання задач введення/виведення у Python з використанням мультипотокового підходу. Новизна полягає у практичному застосуванні вбудованих засобів стандартної бібліотеки Python (зокрема модуля threading) без використання сторонніх бібліотек, що дозволяє легко відтворити експерименти в онлайн-середовищах. Вперше в тезах такого спрямування наведено реальні приклади завантаження вебсторінок із замірами часу та прямим порівнянням продуктивності між послідовною та паралельною реалізаціями. Це сприяє кращому розумінню принципів конкурентного програмування студентами та дослідниками, які не мають глибокого досвіду в системному програмуванні.

Отже, мультипотоковість у Python є потужним інструментом для підвищення ефективності програм, особливо в задачах, пов'язаних з обробкою вводу/виводу. Однак для обчислювально важких завдань, де потрібне повне використання ресурсів процесора, більш ефективним буде використання мультипроцесорності. Тому важливо вміти вибирати правильний інструмент для кожного типу задачі. У випадку високопродуктивних вебсистем, обробки даних і моніторингу, мультипотоковість в Python дозволяє значно покращити швидкодію та зменшити час реакції системи.

Бібліографія

1. Погорілий С. Д., Семьонов Б. О. Дослідження паралельних алгоритмів мовою Python з використанням різних платформ [Електронний ресурс] / С. Д. Погорілий, Б. О. Семьонов // Національний університет «Києво-Могилянська академія». – Режим доступу: https://ekmair.ukma.edu.ua/bitstream/handle/123456789/12539/Pohorilyi_Doslidzhennia_paralelnykh.pdf
2. DevZone. Вступ до асинхронного програмування на Python [Електронний ресурс]. – Режим доступу: https://devzone.org.ua/post/vstup-do-asynkhronnoho-prohramuvannia-na-python
3. Hillel IT School. Асинхронний Python: різні форми конкурентності [Електронний ресурс]. – Режим доступу: https://blog.ithillel.ua/articles/async-python
4. JavaRush. Python Core – Паралельні алгоритми та їх складність [Електронний ресурс]. – Режим доступу: https://javarush.com/ua/quests/lectures/ua.javarush.python.core.lecture.level20.lecture05
5. Python Software Foundation. Паралельне виконання – Python 3.13.3 документація [Електронний ресурс]. – Режим доступу: https://docs.python.org/uk/3.13/library/concurrency.html