Багатопоточність

Багатопоточність (або паралельність) - це можливість програми виконувати кілька завдань (потоків) одночасно. Кожен потік представляє собою окремий шлях виконання, який може виконувати свій власний код паралельно з іншими потоками.

У Python існують кілька способів роботи з багатопоточністю:

1. **Модуль `threading`**:
    - Це вбудований модуль Python, який надає можливість створювати та керувати потоками. З допомогою цього модуля можна створювати, запускати, призупиняти та завершувати потоки.

    Приклад:

    ```
    import threading

    def print_numbers():
        for i in range(5):
            print(i)

    thread = threading.Thread(target=print_numbers)
    thread.start()
    ```

2. **Модуль `concurrent.futures`**:
    - Цей модуль надає вищий рівень абстракції над багатопоточністю, дозволяючи вам працювати з потоками та процесами за допомогою спільного інтерфейсу.

    Приклад (використання пулу потоків):

    ```
    from concurrent.futures import ThreadPoolExecutor

    def print_numbers(i):
        print(i)

    with ThreadPoolExecutor(max_workers=3) as executor:
        executor.map(print_numbers, range(5))
    ```

3. **Модуль** `asyncio` (для асинхронного програмування):

   - Дозволяє створювати асинхронні функції та корутини, які виконуються паралельно в одному потоці. Про нього ми поговоримо пізніше

   Приклад:

```
import asyncio

async def print_numbers(i):
    print(i)

asyncio.run(asyncio.gather(*(print_numbers(i) for i in range(5))))
```

Багатопоточність дозволяє програмі виконувати завдання паралельно, що може покращити продуктивність в деяких випадках. Однак слід бути обережним при використанні багатопоточності, оскільки вона може призвести до гонок за ресурсами і інших проблем, які пов'язані з паралельним виконанням коду.

Threading

Модуль `threading` є вбудованим модулем у Python, який надає можливість створювати та керувати потоками в програмі. Потоки представляють собою окремі шляхи виконання, які можуть працювати паралельно, дозволяючи виконувати кілька завдань одночасно.

Основні класи та функції, які надає модуль `threading`, включають:

1. `Thread`: Цей клас дозволяє створювати нові потоки. Він приймає функцію, яку ви хочете виконати у новому потоці.

   Приклад:

```
import threading

def print_numbers():
    for i in range(5):
        print(i)

thread = threading.Thread(target=print_numbers)
thread.start()  # Запускає потік
```

2. `Lock`: Клас `Lock` використовується для синхронізації доступу до спільних ресурсів з декількох потоків.

   Приклад:

```
import threading

shared_variable = 0
lock = threading.Lock()

def increment():
    global shared_variable
    with lock:
        shared_variable += 1

threads = []
for _ in range(5):
    thread = threading.Thread(target=increment)
    threads.append(thread)
    thread.start()

for thread in threads:
    thread.join()

print("Загальна змінна:", shared_variable)
```

3. `Event`: Клас `Event` дозволяє потокам взаємодіяти за допомогою сигналів.

   Приклад:

```
import threading

event = threading.Event()

def wait_for_event():
    print("Чекаю на подію...")
    event.wait()
    print("Подія сталася!")

def set_event():
    print("Подія встановлена.")
    event.set()

thread1 = threading.Thread(target=wait_for_event)
thread2 = threading.Thread(target=set_event)
```

```
thread1.start()
thread2.start()
```

4. `Semaphore`: Клас `Semaphore` дозволяє обмежити кількість потоків, які можуть одночасно виконувати певний блок коду.

   Приклад:

```
import threading

semaphore = threading.Semaphore(value=2)

def limited_function():
    with semaphore:
        print("Початок виконання")
        # Ваш код тут
        print("Кінець виконання")

threads = []
for _ in range(4):
    thread = threading.Thread(target=limited_function)
    threads.append(thread)
    thread.start()

for thread in threads:
    thread.join()
```

Ці приклади демонструють основні концепції роботи з модулем `threading` в Python. Важливо бути уважним, коли використовуєте багатопоточність, оскільки це може призвести до гонок за ресурсами та інших проблем, пов'язаних з паралельним виконанням коду.