1. Wstęp do programowania wielowątkowego w Pythonie

Programowanie wielowątkowe to technika, która pozwala na wykonywanie równoległych operacji w programie. Python oferuje kilka modułów do obsługi wielowątkowości, takich jak thread, threading i concurrent.futures.

2. Moduły thread, threading oraz concurrent.futures

Moduł thread

Moduł thread to stary moduł Pythona obsługujący niskopoziomowe programowanie wielowątkowe. Choć jest teraz przestarzały i zastąpiony przez moduł threading, nadal można go spotkać w niektórych starszych kodach. Poniżej znajduje się przykład użycia tego modułu.

```
import thread

def print_numbers():
    for i in range(10):
        print(i)

def print_letters():
    for letter in 'abcdefghij':
        print(letter)

# Start two threads
thread.start_new_thread(print_numbers, ())
thread.start_new_thread(print_letters, ())
```

Moduł threading

Moduł **threading** jest nowocześniejszym interfejsem do obsługi wątków w Pythonie. Jest bardziej rozbudowany i oferuje większą kontrolę nad wątkami. Przykład użycia poniżej:

```
import threading

def print_numbers():
    for i in range(10):
        print(i)

def print_letters():
    for letter in 'abcdefghij':
        print(letter)

# Create two threads
t1 = threading.Thread(target=print_numbers)
t2 = threading.Thread(target=print_letters)

# Start the threads
t1.start()
t2.start()

# Wait for both threads to finish
t1.join()
t2.join()
```

Moduł concurrent.futures

Moduł concurrent.futures jest jeszcze bardziej zaawansowany i oferuje takie funkcje jak puli wątków oraz przyszłości (futures), które są obiektami reprezentującymi wyniki operacji, które mogą nie być jeszcze dostępne. Przykład użycia:

```
from concurrent.futures import ThreadPoolExecutor

def print_numbers():
    for i in range(10):
        print(i)

def print_letters():
    for letter in 'abcdefghij':
        print(letter)

# Create a ThreadPoolExecutor
with ThreadPoolExecutor(max_workers=2) as executor:
    # Start two threads
    executor.submit(print_numbers)
    executor.submit(print_letters)
```

3. Synchronizacja wątków

Gdy pracujemy z wieloma wątkami, często pojawia się problem synchronizacji. Na przykład, dwa wątki mogą próbować jednocześnie zmodyfikować ten sam zasób, co może prowadzić do niespójnych wyników. Python oferuje kilka narzędzi do synchronizacji wątków, takich jak blokady (**Locks**), semafory (**Semaphores**) czy zmienne warunkowe (**Condition**).

```
import threading

# Create a Lock
lock = threading.Lock()

def print_numbers():
    with lock:
        for i in range(10):
            print(i)

def print_letters():
    with lock:
        for letter in 'abcdefghij':
            print(letter)

# Start two threads
t1 = threading.Thread(target=print_numbers)
t2 = threading.Thread(target=print_letters)
t1.start()
t2.start()
```