

Programowanie 1R – Ćwiczenia 14

Programowanie współbieżne (wielowątkowe i wieloprosesowe)

Mikołaj Marcinkowski

9 czerwca 2026

Zadania wzorowane na Serii 14 zadań autorstwa Bartłomieja Zglinickiego (<https://www.fuw.edu.pl/~bzglinicki/teaching/plr/zadania/seria-14/>), udostępnionej na licencji CC BY 4.0..

UWAGA: Jupyter (pliki `.ipynb`) wykorzystuje pod spodem metody współbieżne i przez to może źle działać, gdy próbujemy w nim wykonywać własny współbieżny kod. Zwłaszcza biblioteka `multiprocessing` lubi sprawiać problemy i po prostu nie działać. Dlatego też polecam wykonać dzisiejsze ćwiczenia jako skrypty `.py`.

Zadanie 1. Wątki i procesy

Wykorzystując bibliotekę `threading`, w szczególności klasę `Thread`, napisz funkcję `wykonaj_wielowatkowo(f, args, N)`, która tworzy N wątków, po czym za pomocą każdego wywołuje funkcję `f` na argumentach `args`.

Wykorzystując bibliotekę `multiprocessing`, w szczególności klasę `Process`, napisz funkcję `wykonaj_wieloprosesowo(f, args, N)`, która tworzy N procesów, po czym za pomocą każdego wywołuje funkcję `f` na argumentach `args`.

Mając te dwie funkcje, porównaj ich zachowanie w następujących przypadkach:

- a) *Problem obciążający procesor*: napisz funkcję, której wykonanie silnie obciąża procesor, np. rekurencyjny wzór na ciąg Fibonacciego (bez optymalizacji). Następnie wywołaj ją 10 razy
 1. pętlą `for` (jednowątkowo),
 2. wielowątkowo (funkcją `wykonaj_wielowatkowo`),
 3. wieloprosesowo (funkcją `wykonaj_wieloprosesowo`).

Porównaj czasy wykonania wszystkich metod.

- b) *Czekanie na plik/stronę internetową*: pod adresem `https://www.fuw.edu.pl/~mmarcinkowski/dydaktyka/powolna_strona.php` jest długo ładująca się strona internetowa. Python potrafi wczytać stronę internetową za pomocą funkcji `urlopen(adres_URL)` z wbudowanej biblioteki `urllib.request`. Na obiekcie zwróconym przez `urlopen` można wywołać metodę `.read().decode('utf-8')`, która zwraca treść strony jako string z kodem HTML.

Wczytaj tę stronę 10 razy

1. pętlą `for` (jednowątkowo),
2. wielowątkowo (funkcją `wykonaj_wielowatkowo`),
3. wieloprocusowo (funkcją `wykonaj_wieloprocusowo`).

Porównaj czasy wykonania wszystkich metod.

Zadanie 2. Równoległe obliczenia pulą procesów

Biblioteka `multiprocessing` posiada bardzo przydatne narzędzie zwane pulą procesów: `Pool`. Za jej pomocą można wykonać równoległe jakąś funkcję na wielu wartościach (na przykład odczytanych z listy) i zebrać ich wyniki.

Niech funkcja `f` przyjmuje jeden argument i zwraca jakąś wartość. Można wykonać ją równoległe na elementach listy `lista` kodem:

```
with Pool() as pula_procesow:
    wyniki = pula_procesow.imap(f, lista)
    for w in wyniki:
        # robimy coś z w
```

Metoda `.imap` zachowuje się podobnie, jak funkcja `map` wbudowana w Pythona. `Pool` posiada też inne metody do równoległego wykonywania funkcji, na przykład `.imap_unordered` jest trochę szybsza, ale nie gwarantuje, że kolejność zwróconych wartości będzie taka sama, jak kolejność elementów `lista`.

Wykorzystując pulę procesów powtórz eksperyment Monte Carlo zadania 5 z 12 serii zadań, w którym szacowaliśmy wartość liczby π . Porównaj czas obliczeń jednowątkowych z obliczeń pulą.

Przydatne: Biblioteka `tqdm` (do samodzielnego zainstalowania) dodaje łatwe w użyciu paski postępu do tego typu problemów. Pokazują też od razu czas wykonania, szacowany czas do końca i szybkość.

Zadanie 3. Kolejka zadań do wykonania

Innym przydatnym narzędziem jest kolejka `Queue` z wbudowanej biblioteki `queue`. Jest to struktura danych pozwalająca na łatwą komunikację między wątkami.

W kolejce można umieszczać elementy metodą `kolejka.put(zadanie)`. Metoda `kolejka.get()` zwraca *najstarszy* (najwcześniej umieszczony) element kolejki i go z niej usuwa. Kolejki są kompatybilne z wątkami – jeśli kilka wątków spróbuje jednocześnie wyjąć element metodą `kolejka.get()`, każdy dostanie inny i dokładnie jeden. Jeżeli kolejka jest pusta, wątek wywołujący `kolejka.get()` będzie czekał, aż w kolejce pojawi się nowy element.

Kolejki są pomyślane do zlecania zadań grupie wątków. Dlatego też, gdy umieszczamy w kolejce nowe zadanie metodą `kolejka.put()`, zwiększa wewnętrzny licznik oczekujących zadań o jeden. Wątki mogą zasignalizować, że skończyły wykonywać dane zadanie, wywołując metodę `kolejka.task_done()`. Pomniejsza to ten licznik o jeden. Specjalnie pomyślana metoda `kolejka.join()` czeka, aż ten licznik spadnie do zera (podobnie jak `watek.join()` czeka, aż wątek skończy pracę).

Wykorzystując kolejkę, napisz program równolegle tworzący wykresy i zapisujący je do plików. Kolejka powinna otrzymywać zadania w postaci krotki `(x, y, nazwa)`. Kilka (np. 3) wątki powinny czytywać zadania z kolejki, tworzyć wykresy $y(x)$ i zapisywać do pliku `nazwa`. Każdy wątek powinien mieć indywidualny numer czy nazwę i wypisywać informację, który zrobił który wykres.

Uwaga: Matplotlib nie współpracuje z wielowątkowymi programami w trybie okienkowym. Jeśli na początku programu wywołamy `matplotlib.use('agg')`, informujemy bibliotekę, że nie zamierzamy otwierać okienek, tylko zapisywać rysunki do plików. Wtedy nie koliduje ona z wielowątkowością.

Przydatne: Na potrzeby programów wieloprocesowych istnieje analogiczna klasa `multiprocessing.Queue`.

Zadanie 4. Webscraping

Webscraping polega na analizowaniu zawartości strony internetowej patrząc na jej kod HTML, w szczególności szukając linków do innych stron i scrapując je.

Napisz funkcję, która przyjmuje adres URL oraz maksymalną liczbę razy, ile ma szukać nowych linków. Dla każdej zbadanej strony niech wypisze listę znalezionych linków. Każdą nową stronę powinna badać w nowym wątku.

Linki można znaleźć na różne sposoby. Można na przykład przeszukiwać kod strony linijka po linijce wyrażeniem regularnym (regex). Przykładowe wyrażenie regularne znajdujące linki to:

```
r'<a.*href=\"([^\"]*)\".*?>'
```

Innym sposobem jest wykorzystać gotową bibliotekę do analizowania kodu HTML. Przykładowo popularna biblioteka **Beautiful Soup** (do samodzielnego zainstalowania). Tego typu problem jest opisany we wstępie do jej dokumentacji.

Linki na stronach są bardzo często względne. Żeby dostać link bezwzględny (oczekiwany przez `urlopen`), można wykorzystać funkcję `urllib.parse.urljoin`.