Zaawansowane techniki programowania w języku Python

Lista zadań – Multiprocessing

Zadanie 1 – Liczba dostępnych procesorów

Zapoznaj się z dokumentacją modułu `multiprocessing` dostępną na https://docs.python.org/. Napisz program, który po uruchomieniu wyświetli użytkownikowi na ekranie liczbę dostępnych procesorów (CPU).

Zadanie 2 – PID

Zapoznaj się z dokumentacją modułu `os` dostępną na https://docs.python.org/. Napisz program, który:

1. wypisze na ekran swój systemowy identyfikator procesu (PID),
2. utworzy dwa procesy, gdzie każdy z nich powinien wypisac swój PID.

Wskazówki:

- skorzystaj z klasy `Process` pochodzącej z modułu `multiprocessig`.

Zadanie 3 – Queue

Zapoznaj się z dokumentacją klasy `Queue` z modułu `multiprocessing` dostępną na https://docs.python.org/. Napisz program, w którym:

1. utworzysz nowy proces potomny, którego zadaniem będzie:
   a) pobieranie kolejnego elementu z kolejki,
   b) wypisanie go na ekranie,
   c) jeśli elementem tym jest napis `'exit'` proces potomny powinien zakończyć działanie,

Notatki:

.....

.....

d) powrót do punktu *a)*;

2. po utworzeniu procesu potomnego, proces rodzica powinien:

   a) odczekać 1 sekundę,

   b) wczytać od użytkownika napis (używając funkcji `input()`),

   c) umieścić go w kolejce,

   d) gdy użytkownik poda napis `'exit'`, zostanie on umieszczony w kolejce do przeanalizowania przez proces potomny, po czym program powinien zakończyć działanie,

   e) powrócić do punktu *a)*.

Przykładowa sesja z działania programu:

```
Podaj napis: hello
Proces potomny: hello
Podaj napis: world
Proces potomny: world
Podaj napis: exit
Proces potomny: exit
```

Wskazówki:

- skorzystaj z klasy `Process` pochodzącej z modułu `multiprocessig`,
- pamiętaj o poprawnym zakończeniu programu (wywołanie funkcji `join()`),
- możesz użyć funkcji `sleep()` z modułu `time`.

Zadanie 4 – Potęga

Napisz program, który wczyta od użytkownika jedną linię zawierającą liczby oddzielone spacją. Następnie w różnych procesach podniesie je do kwadratu i wypisze wszystkie na ekranie w jednej linii oddzielone znakiem spacji.

Wskazówki:

Notatki:

.....

.....

- możesz skorzystać z klasy `Pool()` znajdującej się w module `multiprocessing`,
- możesz skorzystać z klasy `ProcessPoolExecutor()` znajdującej się w module `concurrent.futures`.

Zadanie 5 – Liczby pierwsze

Napisz program, który dla podanych przez użytkownika liczb, sprawdzi w różnych procesach czy poszczególne liczby są pierwsze.

Zadanie 6 – Suma kontrolna

Napisz program, w którym wygenerujesz 200 identyfikatorów UUID4 zapisanych w postaci napisu szesnastkowego. Następnie, za pomocą wielowątkowości, policzysz sumę MD5 każdego z napisów. Po zakończeniu obliczeń wypisz na ekranie parę identyfikator - suma kontrolna dla każdego identyfikatora.

Przykładowy wynik działania aplikacji:

```
ef0a2ffdf9304cd1bc7a49eaf5117df2 - 3384f58fc2bb4b7ede60db9f06130f8b
7cd48ce7494142cca23af64164c5a3c7 - 2c4948ae96bf5a314f1a9c20aaf8544b
<część tekstu wycięta ze względu na długi wynik>
e0be94164ff745328a3b7f93ed14aa41 - b2b79997ed1fb254b574933b99141448
d5d02e4794ed42fc8126ade4fa78256a - 9a39af4ebd38d721210f58df3f016885
```

Wskazówki:

- identyfikatory UUID w postaci napisów możesz przechowywać w liście;
- do wygenerowania identyfikatorów UUID4 w postaci napisu szesnastkowego użyj modułu `uuid`, funkcji `uuid4()` i pola `hex` z obiektu wynikowego typu `UUID;`
- do obliczenia sumy MD5 użyj modułu `hashlib`, funkcji `md5()` i `hexdigest();`
- funkcja `md5()` w przyjmuje argument typu `bytes`, do zamiany napisu (typ `str`) na ciąg bajtów (typ `bytes`) użyj metody `encode()`, którą wywołasz na obiekcie napisu;

Notatki:

.....

.....

- możesz skorzystać z generatora `zip()` do wypisywania par uuid – suma kontrolna.

Notatki:

.....

.....