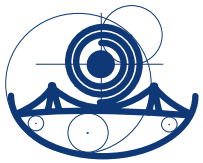
Multi-threading i multi-processing u Pythonu

Rebeka Čorić i Mateja Đumić

SVEUČILIŠTE J.J. STROSSMAYERA U OSIJEKU
ODJEL ZA MATEMATIKU
Trg Ljudevita Gaja 6
31000 Osijek, Hrvatska
http://www.mathos.unios.hr

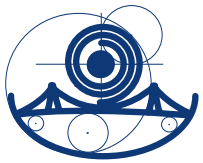
rcoric@mathos.hr, mdjumic@mathos.hr

Kako napraviti thread u Pythonu?



```
import threading

if __name__ == "__main__":
    t = threading.Thread(target=imeFunkcije, args=(
        arg1, arg2,), name="imeThreada")
    t.start()
    t.join()
```

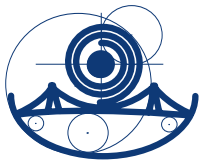
```
import threading

if __name__ == "__main__":
    lock = threading.RLock()
    lock.acquire()
    ...
    lock.release()
    with lock:
        ...
```



Zadatak 1.

a) Napišite klasu **MojaKlasa** koja će kao atribut imati jednu varijablu **x**. Klasa treba imati dvije metode: **increase** i **decrease** koje će uvećavati, odnosno smanjivati vrijednost atributa **x** za 1. U main dijelu programa instancirajte jedan objekt te klase.

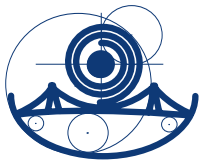
b) Napišite dvije nove funkcije **pozivIncrease** i **pozivDecrease** koje će kao argumente primati jedan integer **brojPonavljanja** te jedan objekt klase **MojaKlasa**. Obje funkcije trebaju **brojPonavljanja** puta pozvati metodu **increase**, odnosno **decrease** na proslijeđenom objektu i nakon toga ispisati trenutno stanje atributa **x** danog objekta.



Zadatak 1. (nastavak)

c) U main dijelu programa napravite dva threada. Prvi thread treba pozvati funkciju **pozivIncrease** 10 puta na objektu koji ste stvorili u a) dijelu zadatka, a drugi thread treba pozvati funkciju **pozivDecrease** 15 puta na istom tom objektu. Pokrenite oba threada i pazite da glavni program ne završi prije nego child threadovi. Na kraju ispišite vrijednost atributa **x** objekta iz a) zadatka.
Napomena: RLock()!

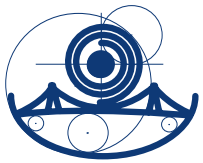
d) Modificirajte main dio programa tako da se threadovi ne pozivaju 10, odnosno 15 puta, nego slučajan broj puta koristeći biblioteku **random**. Ispišite koliko puta se pozvala koja funkcija.



Zadatak 2.

a) Napišite funkciju **isParan** koja će kao argument primiti neki prirodan broj **n** i vratiti **True** ako je broj paran ili **False** ako je broj neparan. Napišite i funkciju **isSavrsen** koja će kao argument primiti neki prirodan broj **n** i vratiti **True** ako je broj savršen ili **False** u suprotnom.

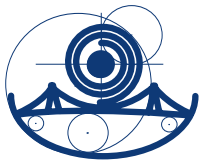
b) U main dijelu programa definirajte listu **parnilliSavrseni** u koju će se spremati brojevi koji su parni ili savršeni i listu **ostali** u koju će se spremati brojevi koji nisu parni ili nisu savršeni.



Zadatak 2. (nastavak)

c) Napišite funkciju **pozivIsParan** koja će kao argument primiti prirodan broj **brojPonavljanja**. Ta funkcija treba **brojPonavljanja** puta slučajno generirati broj između 1 i 1000 (koristiti **random**) i svaki puta provjeriti je li generirani broj paran ili nije koristeći funkciju **isParan**. Ukoliko je broj paran, treba ga dodati u listu **parnilliSavrseni**, a ukoliko nije paran, u listu **ostali**. Pripazite na lock mehanizam prilikom spremanja u listu. Analogno treba napraviti i funkciju **pozivIsSavrsen**.

d) U main dijelu programa napravite dva threada. Prvi thread treba pozvati funkciju **pozivIsParan** 20 puta, a drugi thread treba pozvati funkciju **pozivIsSavrsen** 10 puta. Prvom threadu dajte ime **t1**, a drugom **t2**. Pokrenite oba threada i pazite da glavni program ne završi prije nego child threadovi.



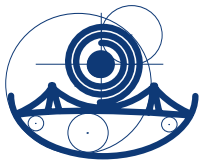
Zadatak 3.

a) Napišite klasu **A** koja će nasljediti threading modul te koja će u run metodi stavljati u red (Queue) brojeve od 1 do 10.

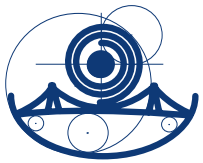
b) Napišite klasu **B** koja će nasljediti threading modul te koja će u run metodi ispisivati brojeve iz reda.

c) U main dijelu programa definirajte red i instancirajte po jedan objekt klase **A** i klase **B**.

Kako napraviti proces u Pythonu?



```
import multiprocessing

if __name__ == "__main__":
    p =multiprocessing.Process(target=imeFunkcije,
        args=(args1,args2,), name="imeProcesa")
    p.start()
    p.join()

import multiprocessing

def imeFunkcije(lock)
    lock.acquire()
    ...
    lock.release()

if __name__ == "__main__":
    lock = multiprocessing.Lock()
    p =multiprocessing.Process(target=imeFunkcije,
        args=(lock,), name="imeProcesa")
    p.start()
    p.join()
```

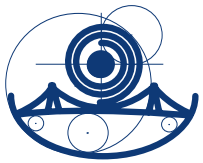


Zadatak 4.

a) Napišite klasu **A** koja će nasljediti multiprocessing.Process te koja će u run metodi stavljati u red (Queue) brojeve od 1 do 10.

b) Napišite klasu **B** koja će nasljediti multiprocessing.Process te koja će u run metodi ispisivati brojeve iz reda.

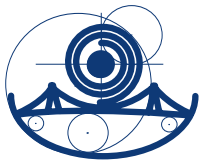
c) U main dijelu programa definirajte red i instancirajte po jedan objekt klase **A** i klase **B**.



Zadatak 5.

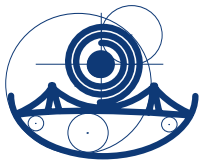
a) Napišite funkciju **red** koja će u red (queue) stavljati brojeve od 1 do 20, a koji predstavljaju kupce koji čekaju u redu za blagajnu.

b) Napišite funkciju **blagajna** koja će brinuti o posluživanju kupaca koji čekaju u redu, odnosno koja će poslužiti sljedećeg kupca u redu.



Zadatak 5. (nastavak)

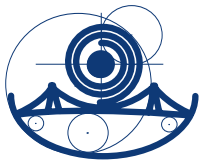
c) U main dijelu programa definirajte red u koji će se stavljati kupci te odredite broj blagajni koji je jednak broju jezgri računala umanjen za 1. Zatim kreirajte proces s ciljnom funkcijom **red**, a za svaku blagajnu definirajte poseban proces čija ciljna funkcija je funkcija **blagajna**. Pokrenite sve procese.



Zadatak 6.

Definirajte klasu **mojThread** koja će nasljeđivati klasu Thread iz modula threading i imati **__init__**, **run** i **getResult** metode.
Metoda **__init__** kao argumente treba imati naziv funkcije koja joj se može proslijediti, listu argumenata i izborni argument **ime** koji se postavlja na prazan string ako ništa nije proslijeđeno.
Metoda **run** treba ispisati "Thread (ime) započeo u (vrijeme početka).", zatim pozvati funkciju proslijeđenu **__init__** metodi sa argumentima koji su proslijeđeni **__init__** metodi te spremiti rezultat u varijablu **self.rez** koju će metoda **getResult** vratiti kada bude pozvana. Na kraju treba ispisati "Thread (ime) završio u (vrijeme završetka)".
Napišite i funkciju koja prima integer **n** i ispisuje je li broj **n** djeljiv sa 5 ili ne.
U main dijelu programa napravite dva threada **t1** i **t2**, koji će kao argumente primiti tu funkciju i pokrenite ih.

Zadatak 7.

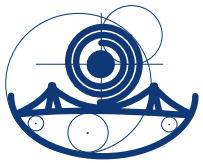


Treba napraviti automat s čokoladicama iz kojeg se mogu kupovati čokoladice ili se mogu nadopunjavati, koristeći **BoundedSemaphore**.

a) Napišite funkciju **nadopuni()** koja će zaključati thread i pokušati nadopuniti čokoladice. Ako je nadopunjavanje uspješno, treba ispisati 'OK', ako je nadopunjavanje neuspješno (tj. ako je automat pun), treba ispisati 'puno, preskačem'. Na kraju treba otključati thread.

b) Napišite funkciju **kupi()** koja će zaključati thread i pokušati kupiti čokoladicu. Ako uspije, treba ispisati 'OK', u suprotnom, treba ispisati 'prazno, preskačem'.

c) Definirajte funkciju **dobavljac()** koja će kao argument primati neki prirodan broj **n** te **n** puta pozvati funkciju **nadopuni** i otići u sleep na slučajan broj sekundi u rasponu od 0 do 3 poslije svakog poziva funkcije. Isto tako, definirajte i funkciju **kupac()** koja će kao argument primiti neki prirodan broj **m** te **m** puta pozvati funkciju **kupi** i otići u sleep na slučajan broj sekundi u rasponu od 0 do 3 poslije svakog poziva funkcije.



Zadatak 7. (nastavak)

d) U main dijelu programa treba ispisati ’Počinje u: ’ i pomoću **ctime()** ispisati vrijeme i ispod toga ’Automat s čokoladicama (pun sa (broj) čokoladica)!’. Zatim treba izabrati slučajan broj petlji između 2 i 6 koristeći biblioteku **random**. Nakon toga treba napraviti 2 threada **t1** i **t2**.
U **t1** treba proslijediti funkciju **kupac** i kao argument proslijediti slučajan broj između odabranog broja petlji i (odabranBrojPetlji + maksimalanBrojCokoladica+2).
U **t2** treba proslijediti funkciju **dobavljac** i kao argument odabran broj petlji.
Pokrenite threadove, pozovite metodu **join** da main thread ne završi prije t1 i t2 te na kraju ispišite ’Gotovo u: ’ i pomoću **ctime()** ispišite vrijeme.