

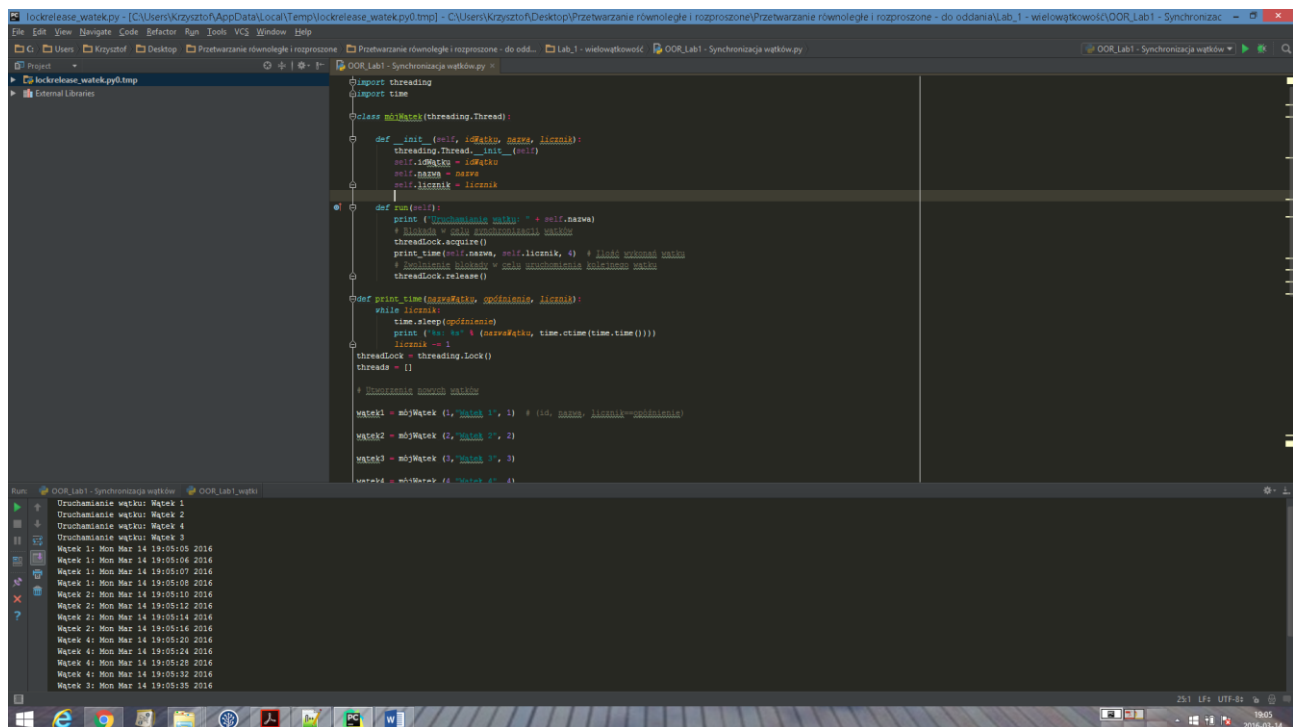
OOB: Programowanie równoległe i rozproszone

Laboratorium 1:

Za pomocą wybranego przez siebie języka programowania zademonstruj tworzenie i synchronizowanie wątków.

Zaprezentuj użycie metod `run()` i `join()` związanych z wątkami.

Zaprezentuj użycie metod `acquire()` i `release()` związanych z blokadami (Lock).



```
lockrelease_watek.py - [C:\Users\Krzysztof\AppData\Local\Temp\lockrelease_watek.py0.tmp] - C:\Users\Krzysztof\Desktop\Przetwarzanie równoległe i rozproszone\Przetwarzanie równoległe i rozproszone - do oddaniaLab_1 - wielowątkowość\OOB_Lab1 - Synchronizacja
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Project: lockrelease_watek.py0.tmp
External Libraries:
OOB_Lab1 - Synchronizacja wątków.py
import threading
import time

class mójWątek(threading.Thread):
    def __init__(self, idWątku, nazwa, licznik):
        threading.Thread.__init__(self)
        self.idWątku = idWątku
        self.nazwa = nazwa
        self.licznik = licznik

    def run(self):
        print(f"Uruchamianie wątku: {self.nazwa}")
        # blokada = mójProstokątnyLock
        threadlock.acquire()
        print_time(self.nazwa, self.licznik, 4) # ilość wykonań wątku
        # wykonanie blokad = mójProstokątnyLock.release()
        threadlock.release()

    def print_time(nazwaWątku, wykonanie, licznik):
        while licznik:
            time.sleep(0.001)
            print(f"{nazwaWątku} {licznik} {nazwaWątku} {time.time() - time.time(0)}")
            licznik -= 1
        threadlock = threading.Lock()
        threads = []

        # Uruchomienie nowych wątków
        wątek1 = mójWątek(1, "Wątek 1", 1) # (id, nazwa, licznik=wykonanie)
        wątek2 = mójWątek(2, "Wątek 2", 2)
        wątek3 = mójWątek(3, "Wątek 3", 3)
        threads = [wątek1, wątek2, wątek3]
```

Sum: OOB_Lab1 - Synchronizacja wątków.py OOB_Lab1 - Synchronizacja wątków.py

Uruchamianie wątku: Wątek 1
Uruchamianie wątku: Wątek 2
Uruchamianie wątku: Wątek 4
Uruchamianie wątku: Wątek 3
Wątek 1: Mon Mar 14 19:05:05 2016
Wątek 1: Mon Mar 14 19:05:06 2016
Wątek 1: Mon Mar 14 19:05:07 2016
Wątek 1: Mon Mar 14 19:05:08 2016
Wątek 2: Mon Mar 14 19:05:10 2016
Wątek 2: Mon Mar 14 19:05:12 2016
Wątek 2: Mon Mar 14 19:05:14 2016
Wątek 2: Mon Mar 14 19:05:16 2016
Wątek 4: Mon Mar 14 19:05:20 2016
Wątek 4: Mon Mar 14 19:05:24 2016
Wątek 4: Mon Mar 14 19:05:28 2016
Wątek 4: Mon Mar 14 19:05:32 2016
Wątek 3: Mon Mar 14 19:05:35 2016

Przykład ilustrujący uruchamianie wielu wątków i równoległe ich wykonywanie.

Czas wykonywania kolejnych wątków jest różny, dlatego o kolejności zakończenia nie decyduje kolejność uruchomienia, lecz czas wykonywania danego wątku (dlatego, że wątki są wykonywane niezależnie i szybciej skończą swoje działanie wątki o krótszym czasie wykonywania).

Kod programu obsługującego wiele wątków w języku Python:

```
# -*- coding: utf-8 -*-
import time
import datetime
import threading

def licz(x):
    time.sleep(x)
    return x * x

# dodajemy stempelek czasowy do komunikatu

def log(message):
    now = datetime.datetime.now().strftime("%H:%M:%S")
    print ("%s %s" % (now, message))

class UruchomWatek(threading.Thread):
    def __init__(self, value):
        threading.Thread.__init__(self)
        self.value = value
    def run(self):
        result = licz(self.value)
        log("Watek z parametrem %s -> wykonuje sie %s sekund" % (self.value,
result))

def main():

    # Każde zadanie podniesienia parametru wątku do potęgi 2 zostanie wykonane w
    # osobnym wątku i będzie się wykonywać przez
    # liczbę sekund będącą wynikiem obliczeń dzięki czemu wątki o niższym parametrze
    # skończą działanie szybciej (a nie w kolejności uruchomienia)

    log("Wątek główny (wykona się najszybciej, bo nie ma opóźnienia)")

    UruchomWatek(3).start()      # Wątek z parametrem: 3

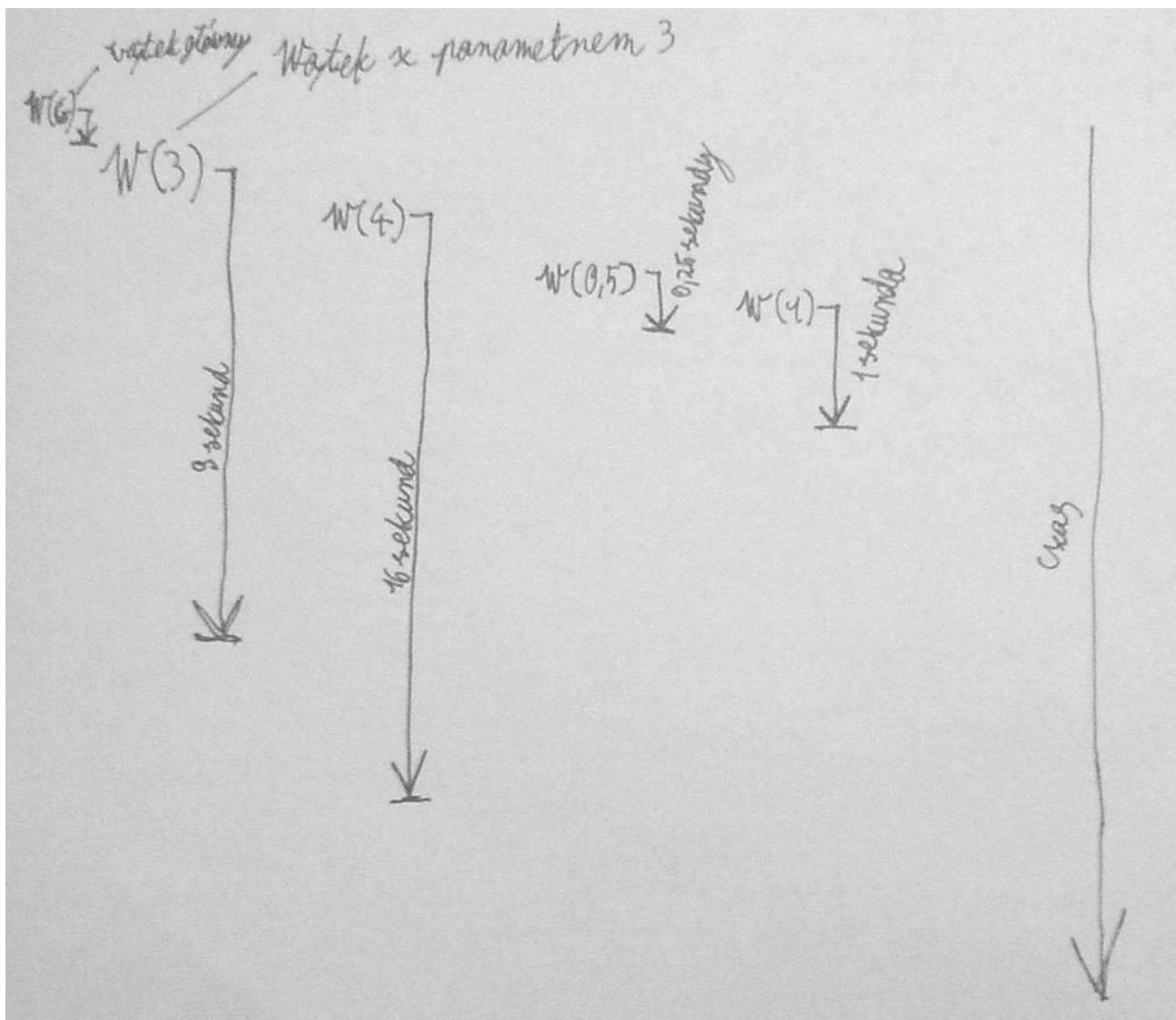
    UruchomWatek(4).start()      # Wątek z parametrem: 4

    UruchomWatek(0.5).start()    # Wątek z parametrem: 0.5

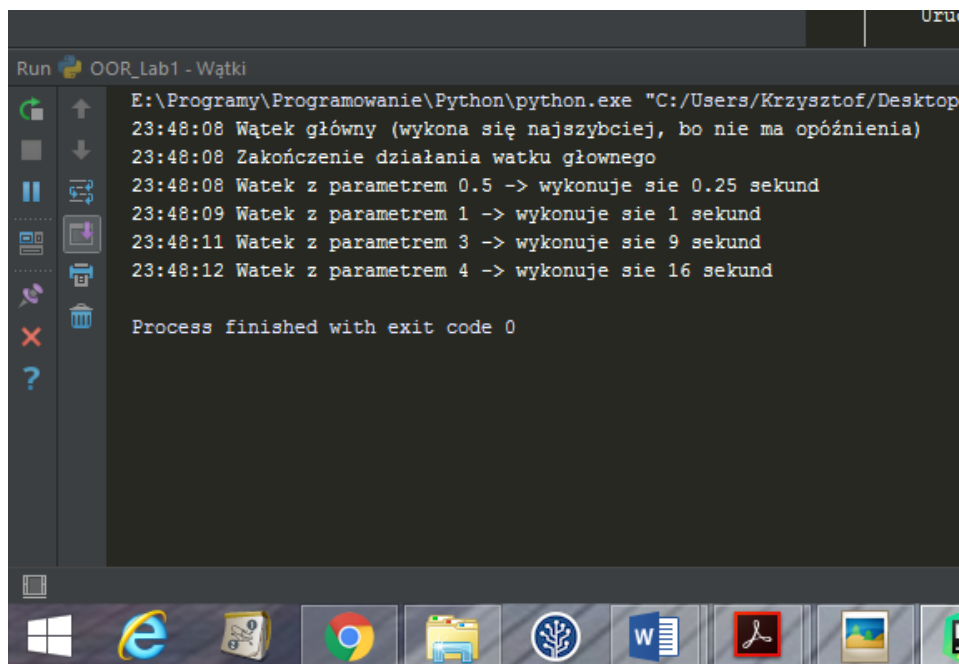
    UruchomWatek(1).start()      # Wątek z parametrem: 1

    log("Zakończenie działania wątku głównego")

if __name__ == "__main__":
    main()
```



Kod wynikowy:



```
E:\Programy\Programowanie\Python\python.exe "C:/Users/Krzysztof/Desktop
23:48:08 Wątek główny (wykona się najszybciej, bo nie ma opóźnienia)
23:48:08 Zakończenie działania wątku głównego
23:48:08 Wątek z parametrem 0.5 -> wykonuje się 0.25 sekund
23:48:09 Wątek z parametrem 1 -> wykonuje się 1 sekund
23:48:11 Wątek z parametrem 3 -> wykonuje się 9 sekund
23:48:12 Wątek z parametrem 4 -> wykonuje się 16 sekund

Process finished with exit code 0
```

Kod programu obsługującego wiele wątków, z synchronizacją wątków w języku Python:

```
import threading
import time

class mójWątek(threading.Thread):

    def __init__(self, idWątku, nazwa, licznik):
        threading.Thread.__init__(self)
        self.idWątku = idWątku
        self.nazwa = nazwa
        self.licznik = licznik

    def run(self):
        print ("Uruchamianie wątku: " + self.nazwa)
        # Blokada w celu synchronizacji wątków
        threadLock.acquire()
        print_time(self.nazwa, self.licznik, 4) # Ilość wykonań wątku
        # Zwolnienie blokady w celu uruchomienia kolejnego wątku
        threadLock.release()

def print_time(nazwaWątku, opóźnienie, licznik):
    while licznik:
        time.sleep(opóźnienie)
        print ("%s: %s" % (nazwaWątku, time.ctime(time.time())))
        licznik -= 1

threadLock = threading.Lock()
threads = []

# Utworzenie nowych wątków

watek1 = mójWątek (1,"Wątek 1", 1) # (id, nazwa, licznik==opóźnienie)
watek2 = mójWątek (2,"Wątek 2", 2)
watek3 = mójWątek (3,"Wątek 3", 3)
watek4 = mójWątek (4,"Wątek 4", 4)

# Uruchomienie nowych wątków

watek1.start()
watek2.start()

watek4.start() # Wątek 4 zostanie uruchomiony przed wątkiem 3
watek3.start()

# Dodawanie wątków do listy wątków

threads.append(watek1)
threads.append(watek2)
threads.append(watek4)
threads.append(watek3)

# Oczekiwanie na zakończenie wszystkich wątków

for t in threads:
    t.join()

print ("Kończenie wątku głównego")
```

Kod wynikowy:

```
lockrelease_watek.py - [C:\Users\Krzysztof\AppData\Local\Temp\lockrelease_watek.py0.tmp] - C:\Users\Krzysztof\Desktop\Przetwarzanie równoległe i rozproszone
File Edit View Navigate Code Refactor Run Tools VCS Window Help

C:\Users\Krzysztof\Desktop\Przetwarzanie równoległe i rozproszone\Przetwarzanie równoległe i rozproszone - do odd... Lab_1 - wielowątkowość OOR_Lab1

Project
lockrelease_watek.py0.tmp
External Libraries

OOO_Lab1 - Synchronizacja wątków.py x OOR_Lab1 - Wątki.py x

import threading
import time

class mójWątek(threading.Thread):
    def __init__(self, idWątku, nazwa, licznik):
        threading.Thread.__init__(self)
        self.idWątku = idWątku
        self.nazwa = nazwa
        self.licznik = licznik

    def run(self):
        print ("Uruchamianie wątku: " + self.nazwa)
        # Blokada w celu synchronizacji wątków
        threadLock.acquire()
        print_time(self.nazwa, self.licznik, 4) # ilość wykonań wątku
        # Zwolnienie blokady w celu uruchomienia kolejnego wątku
        threadLock.release()

def print_time(nazwaWątku, opóźnienie, licznik):
    while licznik:
        time.sleep(opóźnienie)
        print ("%s: %s" % (nazwaWątku, time.ctime(time.time())))
        licznik -= 1
    threadLock = threading.Lock()
    threads = []

    # Utworzenie nowych wątków
    threads = [mójWątek(i, "Wątek %d", i) for i in range(1, 5)]

Run: OOR_Lab1 - Synchronizacja wątków OOR_Lab1 - Wątki

Uruchamianie wątku: Wątek 1
Uruchamianie wątku: Wątek 2
Uruchamianie wątku: Wątek 4
Uruchamianie wątku: Wątek 3
Wątek 1: Mon Mar 14 19:24:34 2016
Wątek 1: Mon Mar 14 19:24:35 2016
Wątek 1: Mon Mar 14 19:24:36 2016
Wątek 1: Mon Mar 14 19:24:37 2016
Wątek 2: Mon Mar 14 19:24:39 2016
Wątek 2: Mon Mar 14 19:24:41 2016
Wątek 2: Mon Mar 14 19:24:43 2016
Wątek 2: Mon Mar 14 19:24:45 2016
Wątek 4: Mon Mar 14 19:24:49 2016
Wątek 4: Mon Mar 14 19:24:53 2016
Wątek 4: Mon Mar 14 19:24:57 2016
Wątek 4: Mon Mar 14 19:25:01 2016
Wątek 3: Mon Mar 14 19:25:04 2016
Wątek 3: Mon Mar 14 19:25:07 2016
Wątek 3: Mon Mar 14 19:25:10 2016
Wątek 3: Mon Mar 14 19:25:13 2016
Kończenie wątku głównego

Process finished with exit code 0
```

Synchronizacja wątków jest niezbędna, gdy operacje są od siebie zależne i muszą być wykonywane jedna po drugiej (do wykonania akcji 2, wymagane jest zakończenie akcji 1).