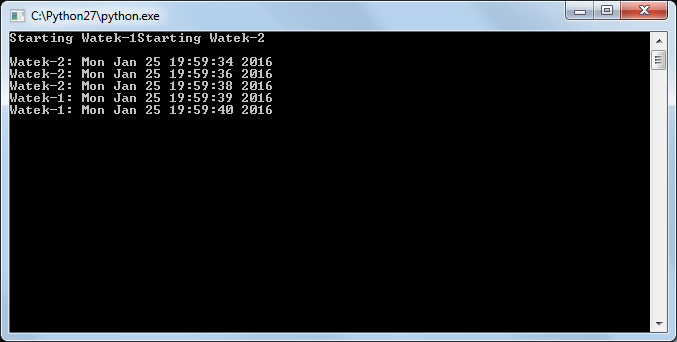
Łukasz Fidurski 125 NCI-A data 25.01.2016

Przedmiot : Programowanie równoległe i rozproszone

Laboratorium numer 1

Wielowątkowość

1. Celem laboratorium było stworzenie i synchronizowanie wątków
2. Wynik:



import threading

import time

class myThread(threading.Thread):

def \_\_init\_\_(self, threadID, name, counter):

threading.Thread.\_\_init\_\_(self)

self.threadID = threadID

self.name = name

self.counter = counter

def run(self): #uruchomienie

print "Starting " + self.name

threadLock.acquire()

print\_time(self.name, self.counter, 3)

threadLock.release()

def print\_time(threadName, delay, counter):

while counter:

time.sleep(delay)

print "%s: %s" % (threadName, time.ctime(time.time()))

counter -= 1

threadLock = threading.Lock()

threads = []

thread1 = myThread (1,"Watek-1", 1)

thread2 = myThread (2,"Watek-2", 2)

thread1.start()

thread2.start()

threads.append(thread1)

threads.append(thread2)

for t in threads:

t.join()

print "Watek"

Jak widać na powyższym zdjęciu wątki były tworzone i synchronizowane

Metoda "run" nadpisuje tą z klasy "Thread", dzięki czemu przy wywołaniu "start()" na wątku "mythread" wykona ona automatycznie metodę "run". W tej metodzie serwer może wejść w pętlę ciągłego odczytu danych od klienta aż do czasu rozłączenia się (klienta z serwerem).

Metoda join() tworzy string łącząc stringi na liście przekazanej jako argument

Repozytorium: