Re: need some help with threading module...

Source: http://coding.derkeiler.com/Archive/Python/comp.lang.python/2004−12/4860.html

---

From: chahnaz.ourzikene (*chahnaz.ourzikene_at_wanadoo.fr*)
Date: 12/30/04

Date: Thu, 30 Dec 2004 11:44:50 +0100

Hi,

I fixed the code, it runs under Linux but not under windows 0_o ??! i guess windows and Linux do not handle threads the same way.

However, i don't have the result i excpect.

Please have a look here :

```
## In this little program, i'm trying to find a way to yield data from a
thread within another
## thread, where both threads are NON−COOPERATIVE.
## To do so, i imagined an application launching two threads : a subject (to
yield data from)
## and a controller that will try to yield data from the subject. The
controller has a link
## with the subject, so he can call his methods and ask him to return the
desired data.
##
## The problem is that it appears that on windows systems, this program
idles and crashes after
## some seconds. On linux systems, this program runs but not the way i
figured : the two threads
## seem not to run in parallel...
##
## Look at the controller code : his job is to call the subject's methode
"count()" and compare
## the returned data to a threshold he's exepcting to reach. If the count
method of the subject
## returns a number less than threshold, than the controller loop again (in
a recursive function)
## until the subject's count method returns a number greater or equal to
threshold or the recursive funtion
## runs "limit" times ( to avoid unlimited recursive calls ).
##
## Now look at the subject code : his only job is to increment a counter in
a recursive methode.
## This counter is retrieved within the controller code via the count()
```

methode of the subject.
When that counter reaches "threshold" (defined in the controller), the controller returns.
##
Execute the following code to see the result. You'll find someting like this under linux :
##
######> controller waiting... 0 loops
######> controller waiting... 1 loops
######> controller waiting... 2 loops
######> controller waiting... 3 loops
######> controller waiting... 4 loops
######> controller waiting... 5 loops
######> controller waiting... 6 loops
######> controller waiting... 7 loops
######> controller waiting... 8 loops
######> controller waiting... 9 loops
######> controller waiting... 10 loops
######> controller waiting... 11 loops
######> controller waiting... 12 loops
######> controller waiting... 13 loops
######> controller waiting... 14 loops
######> controller waiting... 15 loops
######> controller waiting... 16 loops
######> controller waiting... 17 loops
######> controller waiting... 18 loops
######> controller waiting... 19 loops
######> controller waiting... 20 loops
######> controller waiting... 21 loops
######> controller waiting... 22 loops
######> controller waiting... 23 loops
######> controller waiting... 24 loops
######> controller waiting... 25 loops
######> controller waiting... 26 loops
######> controller waiting... 27 loops
######> controller waiting... 28 loops
######> controller waiting... 29 loops
######> controller waiting... 30 loops
######> controller waiting... 31 loops
######> controller waiting... 32 loops
######> controller waiting... 33 loops
######> controller waiting... 34 loops
######> controller waiting... 35 loops
######> controller waiting... 36 loops
######> controller waiting... 37 loops
######> controller waiting... 38 loops
######> controller waiting... 39 loops
######> controller waiting... 40 loops
######> controller waiting... 41 loops
######> controller waiting... 42 loops
######> controller waiting... 43 loops

######> controller waiting... 44 loops
######> controller waiting... 45 loops
######> controller waiting... 46 loops
######> controller waiting... 47 loops
######> controller waiting... 48 loops
######> controller waiting... 49 loops
controller : limit of recursive loops reached :
threshold 20 never reached
Subject : the counter is now 0
Subject : the counter is now 1
Subject : the counter is now 2
Subject : the counter is now 3
Subject : the counter is now 4
Subject : the counter is now 5
Subject : the counter is now 6
Subject : the counter is now 7
Subject : the counter is now 8
Subject : the counter is now 9
Subject : the counter is now 10
Subject : the counter is now 11
Subject : the counter is now 12
Subject : the counter is now 13
Subject : the counter is now 14
Subject : the counter is now 15
Subject : the counter is now 16
Subject : the counter is now 17
Subject : the counter is now 18
Subject : the counter is now 19
Subject : the counter is now 20
Subject : the counter is now 21
Subject : the counter is now 22
Subject : the counter is now 23
Subject : the counter is now 24
Subject : the counter is now 25
Subject : the counter is now 26
Subject : the counter is now 27
Subject : the counter is now 28
Subject : the counter is now 29
Subject : the counter is now 30
Subject : the counter is now 31
Subject : the counter is now 32
Subject : the counter is now 33
Subject : the counter is now 34
Subject : the counter is now 35
Subject : the counter is now 36
Subject : the counter is now 37
Subject : the counter is now 38
Subject : the counter is now 39

from threading import Thread
####################################

```
class Subject(Thread):

    def __init__(self,loops = 100) :
        Thread.__init__(self)
        self.counter = 0
        self.recu_loops = 0
        self.loops=loops

    def run(self):
        self.doRecursiveStuff(self.loops)

    def incrementCounter(self,n=1) :
        self.counter = self.counter + n

    def doIteratingStuff(self, ntimes = 300) :
        for i in xrange(ntimes) :
            self.incrementCounter()
            print "Subject : the counter is now", self.counter

    def doRecursiveStuff(self,ntime = 80) :
        if self.recu_loops < ntime :
            print "Subject : the counter is now", self.counter
            self.incrementCounter()
            self.recu_loops = self.recu_loops + 1
            self.doRecursiveStuff(ntime)
        return

    def count(self):
        return self.counter
####################################
class Controller(Thread):

    def __init__(self, objectToControl, limit=100, threshold=20) :
        Thread.__init__(self)
        self.controlled = objectToControl
        self.recursive_turns = 0
        self.limit= limit
        self.threshold = threshold

    def run(self):
        self.stopCounting()

    def stopCounting(self):
        if self.recursive_turns < self.limit :
            print "######> controller waiting... ", self.recursive_turns,
"loops"
            self.recursive_turns = self.recursive_turns + 1
            if self.controlled.count() >= self.threshold :
                print "controller : threshold", self.threshold, "reached"
                return
            self.stopCounting()
```

```
        else :
            print "controller : limit of recursive loops reached :"
            print "threshold",self.threshold,"never reached"
####################################
class Application (object) :

    def __init__(self) :
        pass

    def launch(self) :
        self.sub = Subject(loops=40)
        self.control = Controller(self.sub,limit=50,threshold=20)
        self.control.start()
        self.sub.start()
####################################
a = Application()
a.launch()
```