Boo generator method benchmark

This is a comparative generator method benchmark in C# 2.0, Python, and Boo. It implements the simple weightless cooperating threads as described in Charming Python: Implementing "weightless threads" with Python generators. The times to complete the benchmark on a 1.7GHz Pentium M IBM T41p running Windows XP Professional SP1:

Language / runtime	Normalized time	Actual time (secs)
Visual C# 2005 Express beta / .Net 2.0 beta	0.42	5.76
boo 0.4.3 / .Net 1.1	1	13.72
ActivePython 2.3.2	7.88	108.05

Boo once again trounces Python.

The benchmark code is reproduced below for boo, C#, and Python. Run the benchmark with the command line "threads".

boo version

```
/*
 *  boo version - generator method benchmark
by Bill Wood
 */

import System.Collections

class Threads:
    static x = 0
    static active = true
    static TOTALSWITCHES = 10**8
    static NUMTHREADS = TOTALSWITCHES/100
    static threads = array(IEnumerator,
NUMTHREADS + 1)
```

```
    def looper():
        while 1:
            x += 1
            yield null

    static def quitter():
        for n in
range(TOTALSWITCHES/NUMTHREADS):
            x += 1
            yield null
        active = false

    static def scheduler():
        while active:
            for thread in threads:
                thread.MoveNext()
    static def main():
        start = date.Now
        for i in range(NUMTHREADS):
            threads[i] =
Threads().looper().GetEnumerator()
        threads[NUMTHREADS] =
quitter().GetEnumerator()
        print("Boo time to initialize:
$(date.Now - start)")
        start2 = date.Now
        scheduler()
        print("Boo time to run:
$(date.Now - start2)")
        print("Boo total time:
$(date.Now - start)")
```

```
        print("TOTAL SWITCHES:
$TOTALSWITCHES")
        print("TOTAL THREADS:  $NUMTHREADS")
        if x != TOTALSWITCHES + NUMTHREADS +
TOTALSWITCHES/NUMTHREADS:
            print("Bad checksum: $x")
```

```
Threads.main()
```

C# version

```
/*
 * C# 2.0 version - generator method
benchmark by Bill Wood
 */
#region Using directives

using System;
using System.Collections.Generic;
using System.Text;

#endregion

namespace Threads {
    class Threads {
        static int x = 0;
        static bool active = true;
        const int TOTALSWITCHES = 100000000,
NUMTHREADS = TOTALSWITCHES/100;
        static IEnumerator<int>[] threads =
new IEnumerator<int>[NUMTHREADS + 1];

// Property version
        IEnumerable<int> looper {
            get {
                while (true) {
```

```
                    yield return ++x;
                    }
                }
            }

// Function version
        static IEnumerable<int> quitter() {
            for (int i = 0; i <
TOTALSWITCHES / NUMTHREADS; i++)
                yield return ++x;
            active = false;
            }

        static void scheduler() {
            while (active)
                foreach (IEnumerator<int>
thread in threads) {
                    thread.MoveNext();
                    }
            }

        static void Main(string[] args) {
            DateTime start = DateTime.Now;
            for (int i = 0; i < NUMTHREADS;
i++) {
                threads[i] = (new
Threads()).looper.GetEnumerator();
                }
            threads[NUMTHREADS] =
quitter().GetEnumerator();
```

```
            Console.WriteLine("C# time to
initialize: " + (DateTime.Now - start));
            DateTime start2 = DateTime.Now;
            scheduler();
            Console.WriteLine("C# time to
run:       " + (DateTime.Now - start2));
            Console.WriteLine("C# total
time:        " + (DateTime.Now - start));
            Console.WriteLine("Total
Switches: {0}", TOTALSWITCHES);
            Console.WriteLine("Total
Threads:  " + NUMTHREADS);
            if (x != TOTALSWITCHES +
NUMTHREADS + TOTALSWITCHES / NUMTHREADS)
                Console.WriteLine("Bad
checksum: " + x);
            }
```

```
        }
    }
```

Python version

```
"""
 Python version - generator method benchmark
by Bill Wood

"""

from __future__ import generators
import sys, time

threads = []
TOTALSWITCHES = 10**8
NUMTHREADS    = TOTALSWITCHES/100
x = 0

def null_factory():
    def looper():
        global x
        while 1:
            x += 1
            yield None
    return looper()

def quitter():
    global x
    for n in
```

```
xrange(TOTALSWITCHES/NUMTHREADS):
        x += 1
        yield None

def scheduler():
    global threads
    try:
        while 1:
            for thread in threads:
thread.next()
    except StopIteration:
        pass

if __name__ == "__main__":
    starttime = time.clock()
    for i in range(NUMTHREADS):
        threads.append(null_factory())
    threads.append(quitter())
    print "Python initialization time:
", time.clock()-starttime
    starttime2 = time.clock()
    scheduler()
    print "Python time to run:    ",
time.clock()-starttime2
    print "Python total time:     ",
time.clock()-starttime
    print "TOTAL SWITCHES:", TOTALSWITCHES
    print "TOTAL THREADS: ", NUMTHREADS
    if x != TOTALSWITCHES + NUMTHREADS +
```

```
TOTALSWITCHES/NUMTHREADS:
        print "Bad checksum: ", x
```