Wątki i komunikacja między nimi w języku Python

Czyli kolejny poziom abstrakcji.

Michał Mazurek
mazurek.michal@gmail.com

http://www.ifresearch.pl

January 14, 2008

Python?
A co to?

Python

Interpretowany, interaktywny język programowania stworzony przez Guido van Rossuma w 1990. Python posiada w pełni dynamiczny system typów i automatyczne zarządzanie pamięcią, jest zatem podobny do takich języków, jak Tcl, Perl, Scheme czy Ruby.[a]

Więcej informacji w Google. :)

[a] źródło http://pl.wikipedia.org/wiki/Wątek_(informatyka)

Spis treści.

Jeden wątek, a kilka wątków.

W programie jednowątkowym liczymy w tym samym czasie jedną rzecz, w programie wielowątkowym kilka rzeczy możemy policzyć w tym samym momencie.[1]

[1] Tak naprawde to nie jest nigdy ten sam moment, ale różnice czasowe są takie małe, że w sumie nie ma różnicy, dla nas. Dla procesora to dużo czasu i pewnie dla czegoś jeszcze też, w sumie dla całego wszechświata nasza cywilizacja to jest moment...

Zastosowanie

Przecież nie będziemy tylko liczyć!

- Serwery socketowe
- Różnego rodzaju playery
- Systemy operacyjne
- Wszystko dookoła[2]

[2] Tak naprawdę to wszystko jest wielowątkowe, niby pisząc to nawet nie robie tylko tego, używam jabbera, słucham czegoś. **Zawsze** wszystko działa w wątkach

Moduł *threading*

threading

High-levelowy interfejs do obsługi wątków, zbudowany na module *thread*.[a]

[a] Nie będe tu przytaczał funkcji itd. polecam dokumentacje python'a

threading.Thread

Prosta obsługa wątków.

```
1   from threading import Thread
2   class MyThread(Thread):
3
4       def __init__(self, times):
5           try:
6               self.times = int(times)
7           except ValueError:
8               self.times = 0
9           self.alive = True
10          super(MyThread, self).__init__()
11
12      def run(self):
13
14          for x in range(0, self.times):
15              print x
16
17
```

Ale co z wątkiem który ma działać zawsze.

Implementacja metody *stop()*

```
14          def run(self):
15              import time
16              while self.alive:
17                  print "smth"
18                  time.sleep(1)
19
20          def stop(self):
21              self.alive = False
```

Ok więc aplikacja się wiesza...

To zróbmy coś z tym? Moduł *signal*

```
1   import signal
2   import time
3   def signal_handler(signum, frame):
4       print "Caught signal! %s" % str(signum)
5
6   signal.signal(signal.SIGINT, signal_handler)
7   signal.signal(signal.SIGTERM, signal_handler)
8
9   while True:
10      print "test"
11      time.sleep(1)
```

Przykład
threadtest1.py 1

```
import sys
import time
import signal
from threading import Thread

class MyThread(Thread):

    def __init__(self, times):
        try:
            self.times = int(times)
        except ValueError:
            self.times = 0
        self.alive = True
        #self.file = open("thread1.out", "w")
        super(MyThread, self).__init__()

    def run(self):
        while self.alive:
            self.times -= 1
            print str(self), self.times
            time.sleep(1)

    def stop(self):
        self.alive = False

threads = []
```

Przykład
threadtest1.py 2

```
def signal_handler(signum, frame):
    print "Caught signal! %s" % str(signum)
    for th in threads:
        print "Closing %s" % str(th)
        th.stop()
    sys.exit(0)

# we're catching the sigterm
signal.signal(signal.SIGINT, signal_handler)
signal.signal(signal.SIGTERM, signal_handler)

th = MyThread(10)
th.start()

threads.append(th)

for x in range(0, 10):
    print x
    time.sleep(1)

print "Now let's stop..."
th.stop()
```

Motyw message board'u.

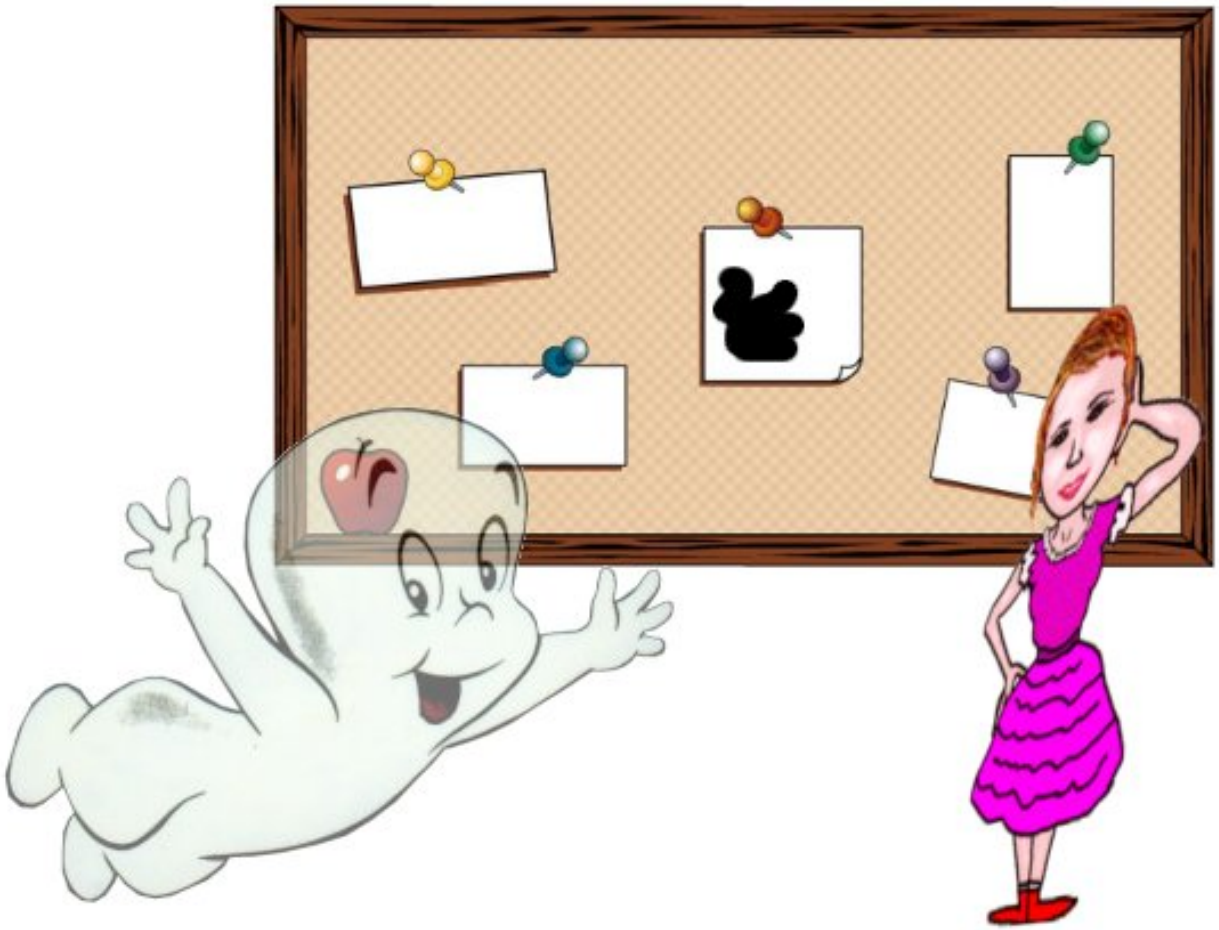
Co to message board?

Message Board

Wyobraź sobie tablicę z informacjami, na której pojawiają sie informacje, nie znani są nadawcy i odbiorcy, ważna jest tylko informacja

Przyklad message board'u - 1.

Przyklad message board'u - 2.



Przyklad message board'u - 3.

Implementacja

Moduł A

```
1   import modulec
2
3   modulec.flag = True
```

Moduł B

```
1   import modulec
2
3   if modulec.flag:
4       print "Hello World!"
```

Moduł C

```
1   flag = False
```

Implementacja z threadami

threadmodulec.py

```
from threading import Event

event = Event()
flag = "Value"
die = False
```

Implementacja z threadami

threadmodulea.py

```
from threading import Thread
import time

import threadmodulec

class ModuleA(Thread):

    def __init__(self):
        self.alive = True
        super(ModuleA, self).__init__()

    def run(self):
        for x in range(0, 10):
            threadmodulec.event.set()
            threadmodulec.flag = x
            if threadmodulec.die or not self.alive:
                print "Killing ModuleA"
                break
            time.sleep(2)


    def stop(self):
        self.alive = False

th = ModuleA()
start = th.start
```

Implementacja z threadami

threadmoduleb.py

```
import time
from threading import Thread

import threadmodulec

class ModuleB(Thread):

    def __init__(self):
        self.alive = True
        super(ModuleB, self).__init__()

    def run(self):

        while True:
            threadmodulec.event.wait(5)
            if threadmodulec.event.isSet():
                print threadmodulec.flag
                threadmodulec.event.clear()
            else:
                print "I've waited 5 secs..."
            if threadmodulec.die or not self.alive:
                print "Killing ModuleB"
                break

    def stop(self):
        self.alive = False

th = ModuleB()
start = th.start
```

Implementacja z threadami

threadmodulemain.py

```
import signal
import time
import sys

from threadmodulea import start as modulea_start
from threadmoduleb import start as moduleb_start
import threadmodulec


def signal_handler(signum, frame):
    print "Caught signal! %s" % str(signum)
    #setting 'die' for true
    threadmodulec.die = True
    #sys.exit(0)

# we're catching the sigterm
signal.signal(signal.SIGINT, signal_handler)
signal.signal(signal.SIGTERM, signal_handler)

#lets start the threads
modulea_start()
moduleb_start()

while not threadmodulec.die:
    time.sleep(1)
    print "Main iterated..."

print "Exiting main thread"
```

Linki

Źródła do pobrania

`http://www.michalmazurek.pl/zimowisko/zimowisko.tar.gz`

Python

`http://www.python.org`

Dokumentacja modułu threading

`http://docs.python.org/lib/module-threading.html`