파이썬 쓰레드 문법

파이썬의 쓰레드와 동기화 객체 문법에 대해 정리한다.

함수 -> 쓰레드

threadEx1.py

```
# -*- coding: utf-8 -*-

import threading, time

def testThread(id):
    for i in range(10):
        print 'thread(%s) --> %s' % (id, i)
        time.sleep(0)    # context switching

testThreads = []
for i in range(3):
    th = threading.Thread(target=testThread, args=(i,))
    th.start()
    testThreads.append(th)

for th in testThreads:
    th.join()

print 'end'
```

클래스 -> 쓰레드 (상속 이용)

threadEx2.py

```
# -*- coding: utf-8 -*-

import threading, time

class testThread(threading.Thread):
    def run(self):
        for i in range(10):
            print 'thread(%s) --> %s' % (self.getName(), i)
            time.sleep(0)    # context switching

testThreads = []
for i in range(3):
```

```
    th = testThread()
    th.start()
    testThreads.append(th)

for th in testThreads:
    th.join()

print 'end'
```

Lock

threadLockEx.py

```
# -*- coding: utf-8 -*-

import threading, time

g_count = 0
class testThread(threading.Thread):
    def run(self):
        global g_count
        for i in range(100):
            lock.acquire()
            g_count += 1
            lock.release()
            time.sleep(0)

lock = threading.Lock()
testThreads = []
for i in range(100):
    th = testThread()
    th.start()
    testThreads.append(th)

for th in testThreads:
    th.join()

print g_count
```

Semaphore

threadSemEx.py

```
# -*- coding: utf-8 -*-
```

```
import threading, time

g_count = 0
class testThread(threading.Thread):
    def run(self):
        global g_count
        for i in range(100):
            sem.acquire()
            g_count += 1
            sem.release()
            time.sleep(0)

sem = threading.Semaphore(3)
testThreads = []
for i in range(100):
    th = testThread()
    th.start()
    testThreads.append(th)

for th in testThreads:
    th.join()

print g_count
```

Event

threadEventEx.py

```
# -*- coding: utf-8 -*-

import threading, time

eve = threading.Event()
class ready(threading.Thread):
    def run(self):
        print 'Ready'
        eve.set()

class waitAndStart(threading.Thread):
    def run(self):
        print self.getName(), 'wating...'
        eve.wait()
        print self.getName(), 'start'

thList = []

for i in range(3):
```

```
    thList.append(waitAndStart())
    thList[i].start()

ready().start()

for th in thList:
    th.join()

print 'exit'
```

참고

- [파이썬 기초] 스레드