

Talk for the Boston Python Interest Group

- 1 # Synchronization in Python
- 2 # Allen B. Downey
- 3 # Olin College of Engineering
- 4 # July 14, 2005

Thread Idiom #1

```
1 import threading
2
3 class HelloThread(threading.Thread):
4     def run(self):
5         print 'hello'
6
7 ht = HelloThread()
8 ht.start()
```

Thread Idiom #2

```
1 import threading
2
3 def entry(name):
4     print 'hello,', name
5
6 t = threading.Thread(target=entry, args=['world'])
7 t.start()
```

Thread wrapper class

```
1 class Thread(threading.Thread):
2     def __init__(self, target, *args):
3         threading.Thread.__init__(self, target=target, args=args)
4         self.start()
5
6 def entry(name):
7     print 'hello,', name
8
9 t = Thread(entry, 'world')
```

Example using thread wrapper

```
1 def child_code(n=10):  
2     for i in range(n):  
3         print i  
4         time.sleep(1)  
5  
6 children = [Thread(child_code) for i in range(3)]  
7 for child in children:  
8     child.join()
```

Shared namespace idiom

```
1 class Shared:
2     def __init__(self):
3         self.counter = 0
4
5     def child_code(shared):
6         while True:
7             shared.counter += 1
8
9 shared = Shared()
10 children = [Thread(child_code, shared) for i in range(2)]
11 for child in children:
12     child.join()
```

Thread A

```
1 t1 = shared.counter  
2 t2 = t1 + 1  
3 shared.counter = t2
```

Thread B

```
1 t3 = shared.counter  
2 t4 = t3 + 1  
3 shared.counter = t4
```

Protecting a shared variable with a Lock

```
1 class Shared:
2     def __init__(self):
3         self.counter = 0
4         self.lock = threading.Lock()
5
6 def child_code(shared):
7     while True:
8         shared.lock.acquire()
9         shared.counter += 1
10        shared.lock.release()
```

Using a Semaphore as a Lock

```
1 class Shared:
2     def __init__(self):
3         self.counter = 0
4         self.lock = threading.Semaphore()
5
6 def child_code(shared):
7     while True:
8         shared.lock.acquire()
9         shared.counter += 1
10        shared.lock.release()
```

mutex.py

```
1 # mutual exclusion
2 mutex = Semaphore(1)
3 counter = 0
4
5 ## Thread code
6
7 mutex.wait()
8 # critical section
9 counter += 1
10 mutex.signal()
```

signal.py

```
1 # signaling example
2 initComplete = Semaphore(0)
3
4 ## Thread A
5 # perform initialization
6 initComplete.signal()
7 # proceed
8
9 ## Thread B
10 initComplete.wait()
11 initComplete.signal()
12 # proceed
```

rendez.py

```
1 # Rendezvous
2 Aarrived = Semaphore(0)
3 Barrived = Semaphore(0)
4
5 ## Thread A
6 Aarrived.signal()
7 Barrived.wait()
8 # proceed
9
10 ## Thread B
11 Barrived.signal()
12 Aarrived.wait()
13 # proceed
```

barrier.py

```
1 # simple barrier
2
3 ## initialization
4 count = 0
5 mutex = Semaphore(1)
6 barrier = Semaphore(0)
7
8
9 ## Thread code
10 # rendezvous
11
12 mutex.wait()
13     count = count + 1
14     if count == num_threads(): barrier.signal()
15 mutex.signal()
16
17 barrier.wait()
18 barrier.signal()
19
20 # critical point
```

```
1 leaders = followers = 0
2 mutex = Semaphore(1)
3 leaderQueue = Semaphore(0)
4 followerQueue = Semaphore(0)
```

Leader

```
1 mutex.wait()
2 if followers > 0:
3     followers--
4     followerQueue.signal()
5     mutex.signal()
6 else:
7     leaders++
8     mutex.signal()
9     leaderQueue.wait()
10
11 # dance!
```

Follower

```
1 mutex.wait()
2 if leaders > 0:
3     leaders--
4     leaderQueue.signal()
5     mutex.signal()
6 else:
7     followers++
8     mutex.signal()
9     followerQueue.wait()
10
11 # dance!
```

coke.py

```
1 # producer-consumer
2 mutex = Semaphore(1)
3 cokes = Semaphore(0)
4 spaces = Semaphore(3)
5
6 ## Thread
7 cokes.wait()
8 mutex.wait()
9     # get a coke
10 mutex.signal()
11 spaces.signal()
12 # drink the coke
13
14 ## Thread
15 # bring a coke to the machine
16 spaces.wait()
17 mutex.wait()
18     # put a coke in the machine
19 mutex.signal()
20 cokes.signal()
```

readwrite.py

```
1 # readers-writers
2 readers = 0
3 mutex = Semaphore(1)
4 roomEmpty = Semaphore(1)
5
6 ## Thread
7 # writers
8 roomEmpty.wait()
9 # go ahead and write
10 roomEmpty.signal()
11
12 ## Thread
13 #readers
14 mutex.wait()
15     readers += 1
16     if readers == 1: roomEmpty.wait()
17 mutex.signal()
18 # critical section for readers
19 mutex.wait()
20     readers -= 1
21     if readers == 0: roomEmpty.signal()
22 mutex.signal()
```

