Other Thread Components

The threading module includes support for other items too. For example, you can create a **Semaphore** which is one of the oldest synchronization primitives in computer science. Basically, a Semaphore manages an internal counter which will be decremented whenever you call **acquire** on it and **incremented when you call**release*. *The counter is designed in such a way that it cannot go below zero. So if you happen to callacquire** when it's zero, then it will block.

Another useful tool that's included is the **Event**. It will allow you to communicate between threads using signals. We will be looking at an example that uses an Event in the next section.

Finally in Python 3.2, the **Barrier** object was added. The Barrier is a primitive that basically manages a thread pool wherein the threads have to wait for each other. To pass the barrier, the thread needs to call the **wait()** method which will block until all the threads have made the call. Then it will release all the threads simultaneously.