Python3 通过两个标准库 \_thread 和 threading 提供对线程的支持。

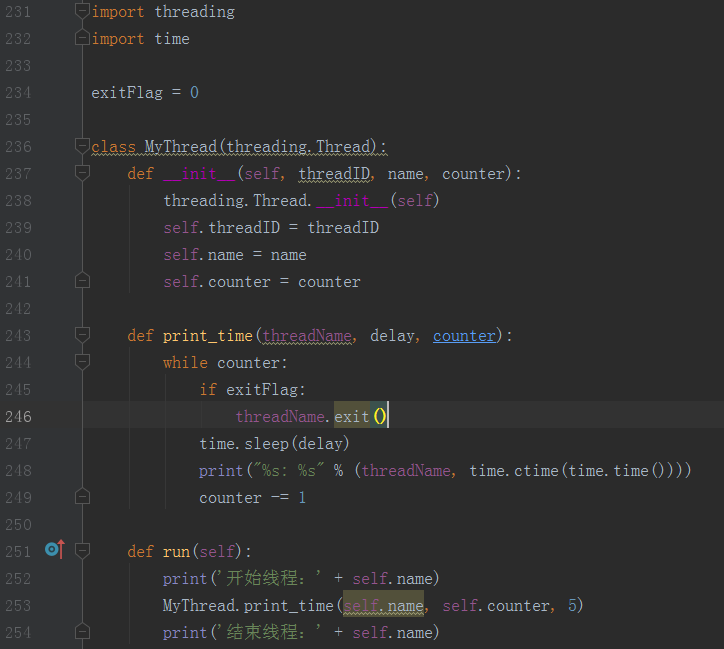
\_thread 提供了低级别的、原始的线程以及一个简单的锁，它相比于 threading 模块的功能还是比较有限的。

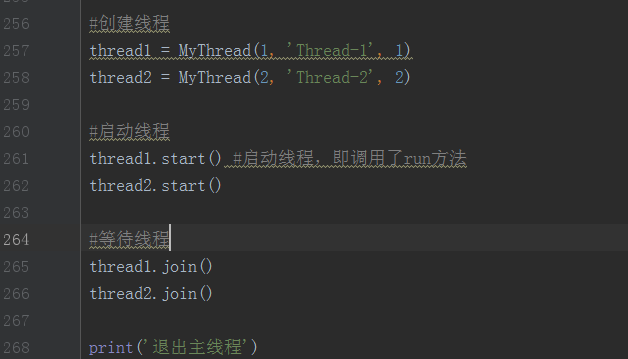
1. threading 模块除了包含 \_thread 模块中的所有方法外，还提供的其他方法：
2. threading.currentThread(): 返回当前的线程变量。
3. threading.enumerate(): 返回一个包含正在运行的线程的list。正在运行指线程启动后、结束前，不包括启动前和终止后的线程。

threading.activeCount(): 返回正在运行的线程数量，与len(threading.enumerate())有相同的结果。

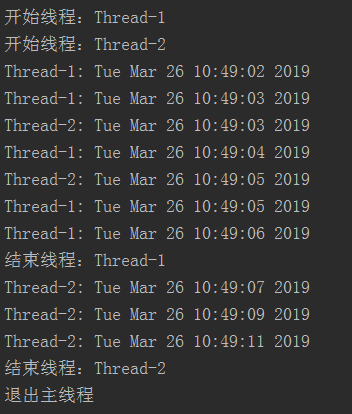
除了使用方法外，线程模块同样提供了Thread类来处理线程，Thread类提供了以下方法:

1. **run():** 用以表示线程活动的方法。
2. **start():**启动线程活动。
3. **join([time]):** 等待至线程中止。这阻塞调用线程直至线程的join() 方法被调用中止-正常退出或者抛出未处理的异常-或者是可选的超时发生。
4. **isAlive():** 返回线程是否活动的。
5. **getName():** 返回线程名。
6. **setName():** 设置线程名。





运行结果：



如果多个线程共同对某个数据修改，则可能出现不可预料的结果，为了保证数据的正确性，需要对多个线程进行同步。

使用 Thread 对象的 Lock 和 Rlock 可以实现简单的线程同步，这两个对象都有 acquire 方法和 release 方法，对于那些需要每次只允许一个线程操作的数据，可以将其操作放到 acquire 和 release 方法之间。如下：



