|  |  |  |
| --- | --- | --- |
| **得分** | **教师签名** | **批改日期** |
|  |  |  |

课程编号： IB01017

****

**深圳技术大学实验报告**

**课程名称： Python 语言程序设计**

**实验名称： 生产者和消费者**

**班 级： 计算机科学与技术2班**

**指导教师： 柯笑**

**报 告 人： 黄荣权 学号： 202002020213**

**合 作 者： 组号：**

**实验地点：**

**实验时间： 2022 年 06 月 09 日 星期 四**

**提交时间： 2022/06/09**

## 实验十二 生产者和消费者

1. 实验学时

2学时

### 实验目的

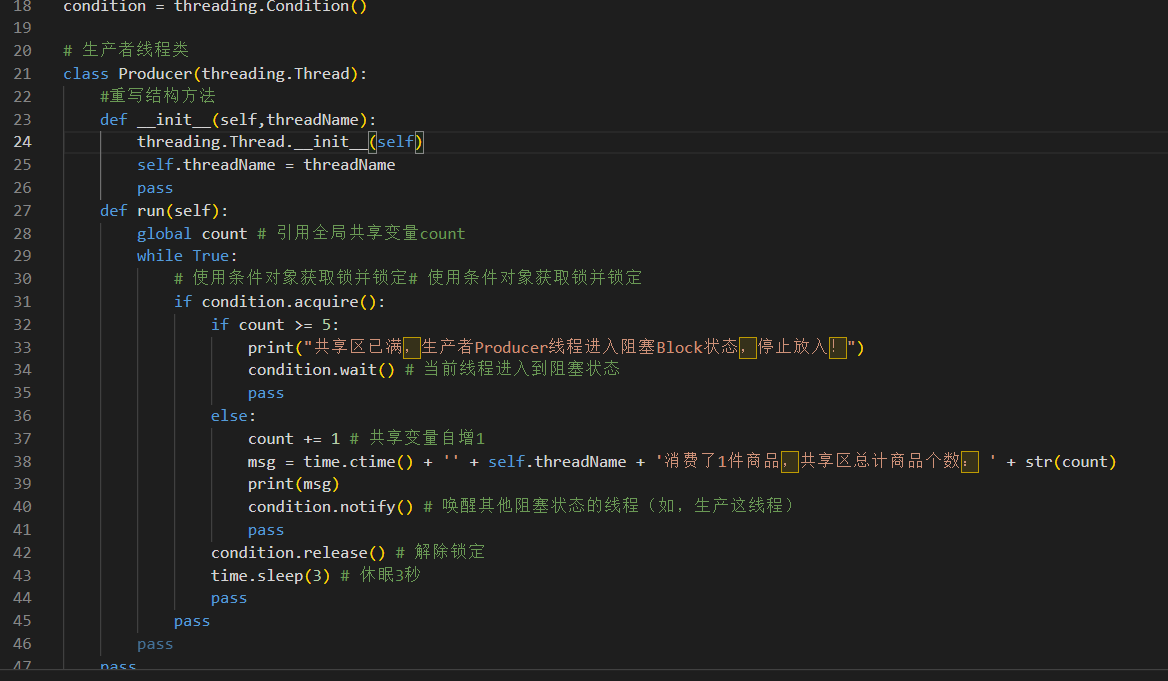
### 了解生产者和消费者问题

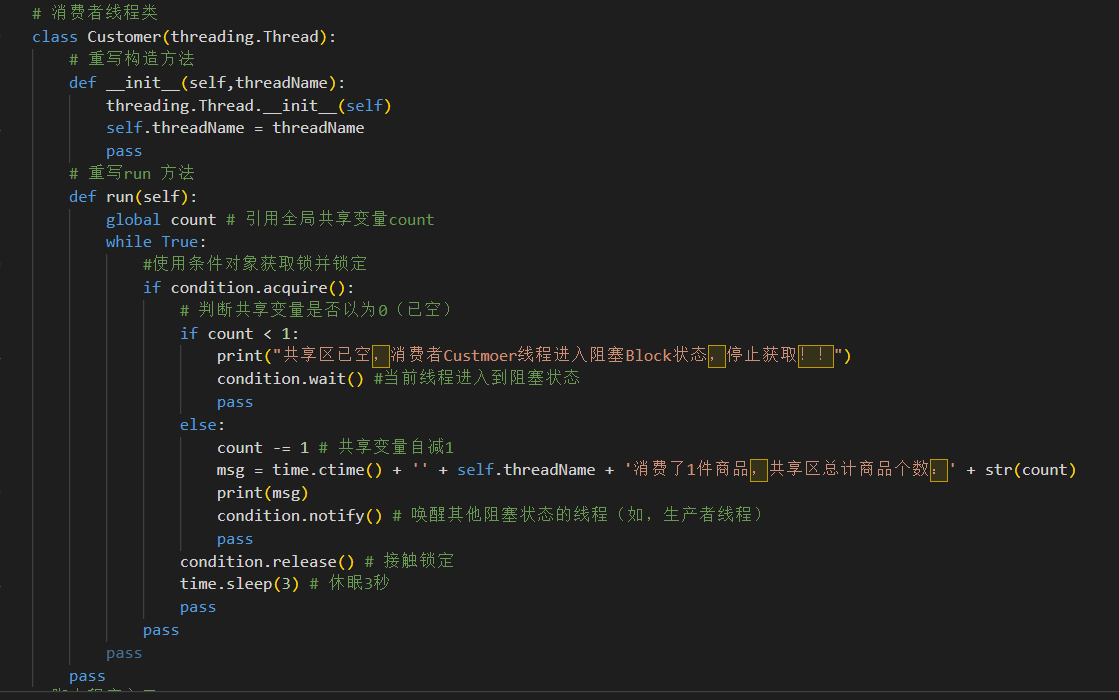
### 掌握python标准库threading创建线程，并实现线程之间的同步。

### 理解缓冲区的重要性。

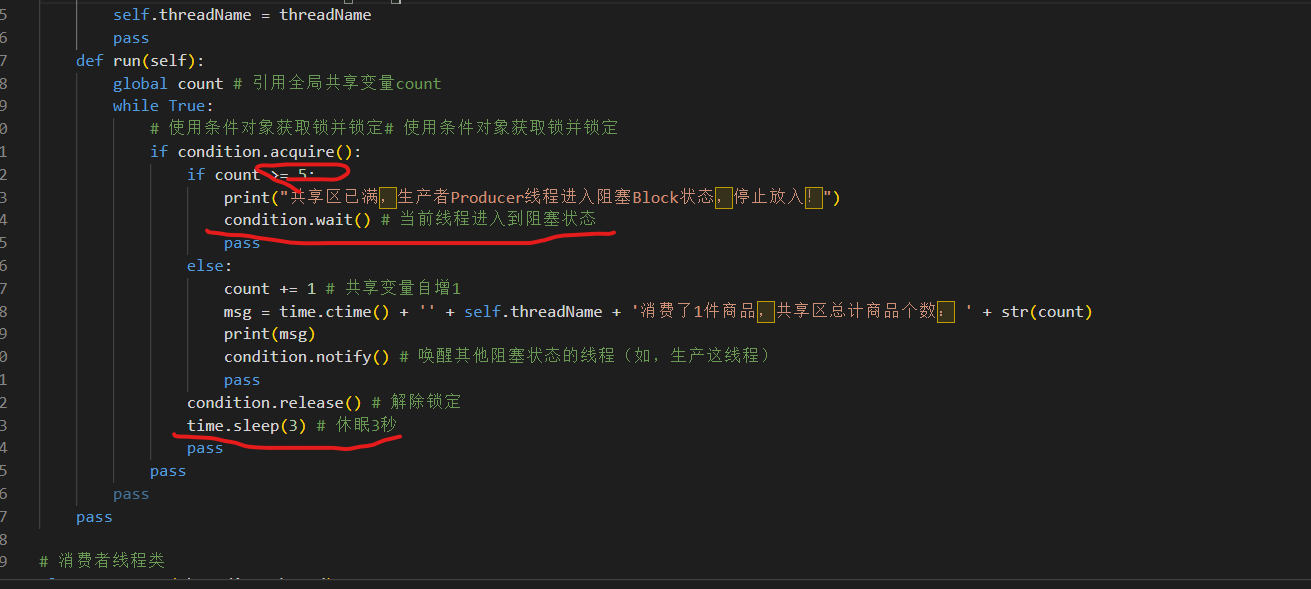
### 三、实验内容

### （一）编写程序，创建生产者线程和消费者线程以及大小为5的缓冲区。

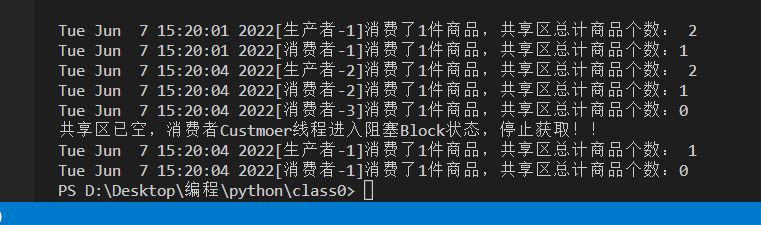




### 生产者每隔1～3秒就产生一个数字放入缓冲区，如果缓冲区已满，则等待；消费者每隔1～3秒就从缓冲区里面取出一个生产日期较早的数字进行消费，如果缓冲区已空则等待。



### 运行时观察缓冲区内的数字变化以及生产者线程和消费者线程之间的同步。



### 四、思考题

（一）阐述一下锁在程序中的作用。

使数据更加安全，解决多线程间的数据完整性和状态同步

（二）python一共有几种实现线程同步的方式？他们分别如何实现的。

Lock()

这是最普通的锁，拥有两种状态：上锁和未上锁。

缺点：是最低级的同步基础组件。从实践角度，会导致严重的死锁。因此，还需要使用其他方法来确保对共享内存的同步访问避免竞态条件。

RLock()

这是对 Lock() 的升级，在类外面能实现线程安全的访问，同时又使用类里面相同的方法。它避免了不同线程在对同一资源要求上锁时，产生的死锁。但要求上锁和解锁成对出现。

Semaphore()

semaphore 的基本实现是基于锁，它在锁的基础上增加了计数器的设置。

缺点：等待与信号操作要在原子块中执行。

Condition()

condition 基本实现也是基于锁，标识了一个线程等待特定的条件，另一个线程通知它条件已经发生。在接口上，除了上锁(acquire)和解锁(release)，还增加了等待(wait)和通知(notify)。

Event()

event 和 semaphore 类似，它通过 bool 类型来区别信号，它另外还增加了等待接口。

# 五、实验结论或体会

通过这次实验，我了解了生产者和消费者的问题，理解了缓冲区的重要性，掌握了使用python标准库threading创建线程，并实现线程之间的同步。

|  |
| --- |
| **指导教师批阅意见：** |
| **成绩评定：**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **操作及记录**  （50分） | **实验总结**  （20分） | **思考题**  （10分） | **报告整体印象**  （20分） | **总分** | |  |  |  |  |  | |

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后10日内。