西安电子科技大学 XIDIAN UNIVERSITY | 软件学院 School of Software

并 行 计 算

课程实验报告

实验名称： Python 并发编程

任课教师： 徐悦甦老师

课程班级： 15 级-云计算方向

学号姓名： 15130120141-孙 晖

提交日期： 2018 年 6 月 27 日

目录

软件学院本科生《并行计算》

课程实验报告

一、实验名称

Python 并发编程（共两道题）

第一题：四个窗口同时出售 30 张电影票（用 Python）

第二题：张三李四分别在柜台和 ATM 上取同一个账户的钱（用 python）

二、实验日期

2018 年 06 月 27 日　软件学院实验室 G346

三、实验学生

15130120141 孙晖

四、实验目的

本次实验通过模拟电影票订票以及对同一个账户取款的模拟，学习 python 并发与并行程序设计，在上一次实验的基础上（上一次实验是为了掌握 python 的基本语法）。这一次实验是通过模拟电影票订票和账户取款来学习 python 的多线程，老师在课堂上将了两种方法，但是更加建议使用第二种方法，因为第二种方法更加直观，通过重写 run 方法，更加美观易懂。

五、实验内容

本次实验要求学生独自完成，不存在组队的情况。

[1] 第一题：

实验第一题要求用 Python 语言模拟四个售票窗口同时出售 30 张电影票。要求实现不能出现卖出同一张票的情况。要求创建一个 BoxOffice 类来实现，BoxOffice 类要继承 threading.Thread()。

[2] 第二题：

实验第二题要求用 Python 语言模拟两个人通过同一个账户取钱的情况，其中张三在柜台取钱，李四在 ATM 取钱。要求，创建一个 Bank 类；创建一个张三类代表在柜台取钱；创建一个李四类代表在 ATM 上取钱；创建主方法的调用类，也就是

if __name__="__name__"

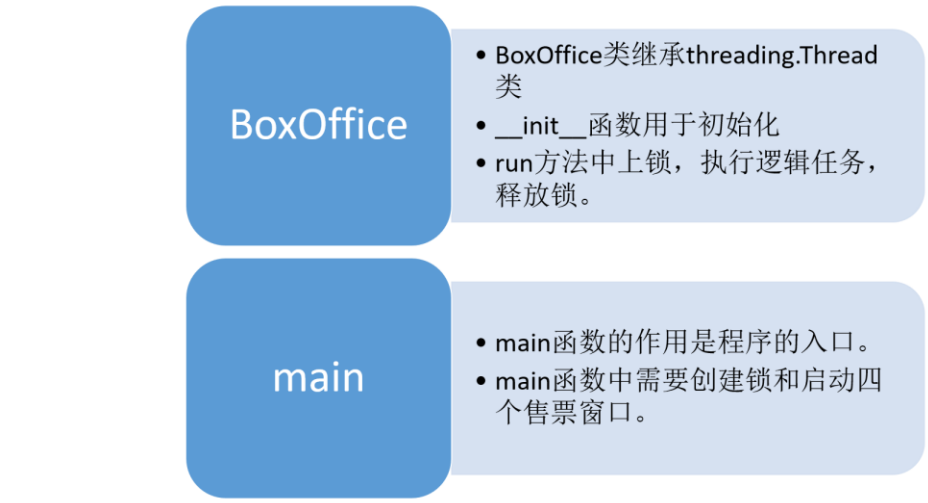
六、程序思路、结构

[1] 第一题：

面对题目的第一反应应该是用什么方法实现 java 多线程，老师在课堂上讲了两种方法，一种是通过继承 threading.Thread 类，通过重写 run 方法实现想要实现的功能。另一种方法是通过把要执行的函数作为参数传递给 threading.Thread()。依据老师 PPT 中给出的参考思路，要使用第一种方法来创建线程，也就是通过继承 threading.Thread 的方法。

程序思路：创建 BoxOffice 类去继承 threading.Thread，重写 run 方法，代码块里实现想要实现的功能，在这一题目中就是上锁，然后执行代码，然后解锁。main 方法里创建所，然后 start 四个窗口线程。

结构：



[2] 第二题：

程序思路：先按照老师的提示创建 Bank 类，ZhangSan 类和 LiSi 类， Bank 类里应该有余额，（这个余额的功能是通过 python 的__init__实现的）这样才能模拟两人在同一个账户上取钱的各种情况，存在都可以取钱的情况，或者只有其中一个人能取钱，因为另一个人取完钱后，该客户想取钱的数额大于余额，这个时候就取不了钱了。老师在课堂上给出了好几种处理多线程的机制包括信号量，加锁等等。这里我才用在 withdraw 方法和 deposit 里加锁的机制，实现两人在不同机器上存取款不会冲突。

结构：

七、程序代码

[1] 第一题代码：

```
# -*- coding: utf-8 -*-
import threading
import time
import random

class BoxOffice(threading.Thread):
    #继承这个类

    '''售票窗口，初始化，为了模拟人是不停地来，
    这里其实可以直接把等待人数设置成 30，但是还是不要这样了。'''
    def __init__(self, wait_num=10, index=0):
        super().__init__()
        self.wait_num = wait_num
        self.setName('窗口' + str(index))

    #重写 run 方法，while 的条件就是如果没票或者没人两者任意一个满足就不卖了
```

```
    def run(self):
        global counter, mutex

        while counter and self.wait_num:
            time.sleep(random.randrange(0,1))
            mutex.acquire()
            #这里的作用就是上锁了
            if counter == 0:
                mutex.release()
            #这里的作用是解锁
                print(self.getName(), '：票已售完')
                break

            counter = counter - 1
            print('%s：剩余 %s 张票'%(self.getName(), counter))
            #其实用 format 也 ok
            mutex.release()
            self.wait_num -= 1
            self.wait_num += random.randrange(0,2)


if __name__ == '__main__':
    # 剩余车票数
    counter = 30

    # 创建锁
    mutex = threading.Lock()

    windows = []
    for i in range(4):
```

```
    #这个线程等待时间不能设置太小啊
        wait_num = random.randrange(0,10)
        windows.append(BoxOffice(wait_num, i+1))

    # start
    for w in windows:
        w.start()
```

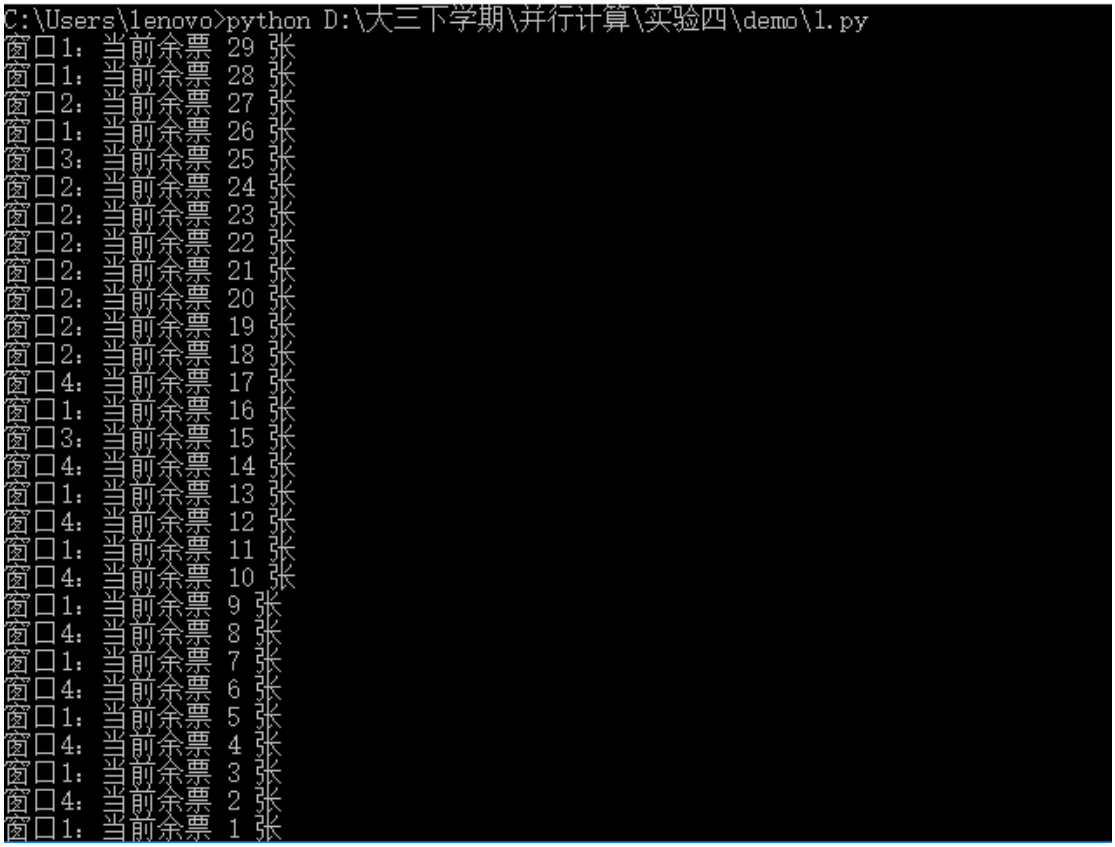
[2] 第二题代码：

```
# -*- coding: utf-8 -*-
import threading,time
#tab 和空格之间的排版问题导致出错真的是一件很难查找出来的问题

class Bank(threading.Thread):
    def __init__(self,money):
        threading.Thread.__init__(self)
        self.money=money

    def withdraw(self,num):
        if lock.acquire():
            temp=self.money
            temp -= num
            self.money = temp
            lock.release()

    def deposit(self,num):
        self.money += num

    def get_money(self):
        return self.money
```

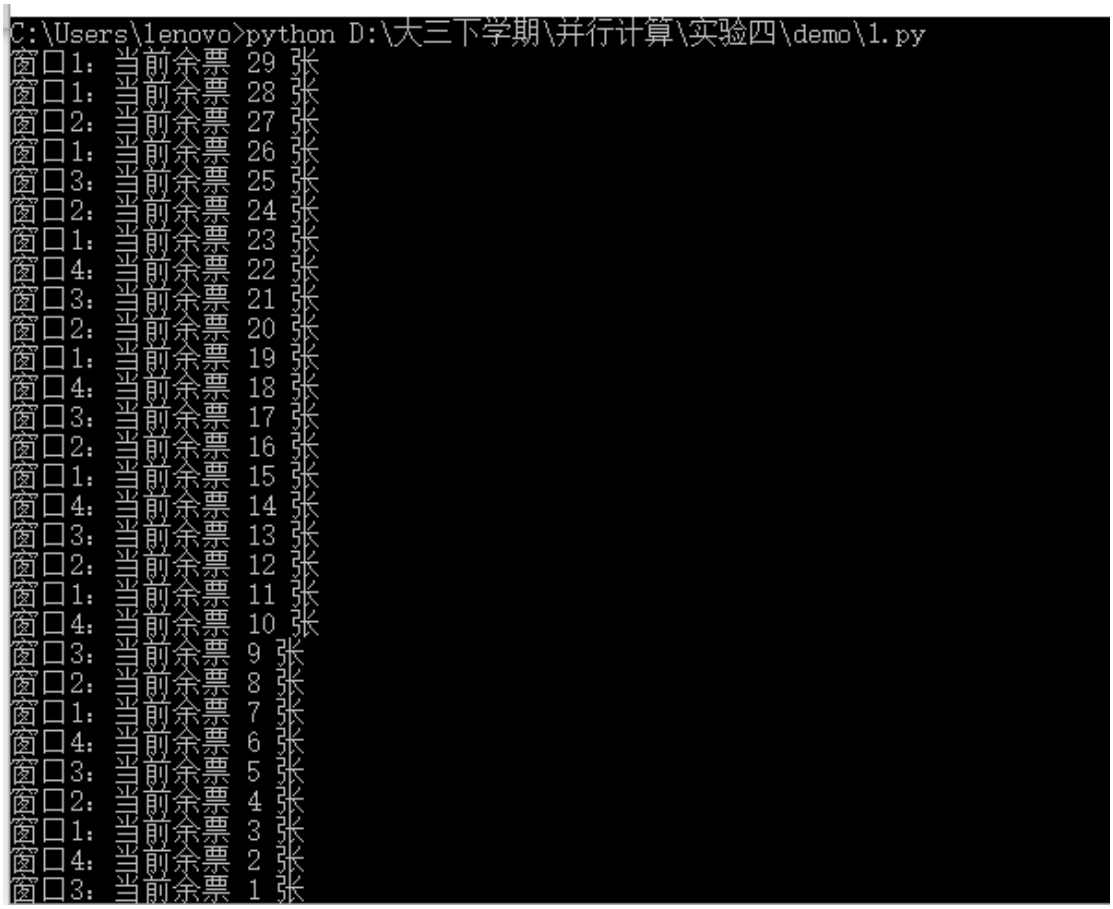
```
class ZhangSan(threading.Thread):
    def __init__(self,account):
        threading.Thread.__init__(self)
        self.account = account

    def run(self):
        self.account.withdraw(100)
        print('张三取了 100 元')


class LiSi(threading.Thread):
    def __init__(self,account):
        threading.Thread.__init__(self)
        self.account = account

    def run(self):
        self.account.withdraw(100)
        print('李四取了 100 元')

if __name__ == '__main__':
    lock=threading.Lock()
    account=Bank(1000)
    #存了 1000
    zhangsan = ZhangSan(account)
    lisi = LiSi(account)

    zhangsan.start()
    lisi.start()

    print("剩余%s 元"%(account.get_money()))
```

八、实验结果

[1] 第一题结果：



因为我是用 random 模拟的购票人数是不停地添加的，所以每一次运行的结果，窗口售票的顺序都是不同的，和下图对比可见：



[2] **第二题结果：**

第二题因为在题目中我定义的是 ZhangSan 先 start，LiSi 后 start，所以结果如图所示。

九、总结建议

[1] 本次代码是在 notepad++上写的，一个非常困扰人的问题就是缩进问题，总是在运行的时候提醒 tab 或者空格混用，主要不是出现在程序里，程序直接 Enter 一下然后 tab 一下就给处理好了，主要是备注部分总是没有一个固定式的缩进，以后备注要牢记用 tab 缩进。

[2] 个人感觉 java 线程要比 python 更加易懂，或者说于我个人而言，java 的多线程要比 python 的多线程简单一些。

[3] 感觉相比于 java 实现多线程的那样手动一次次调用直到票的结束，这一次我在网上发现有人模拟人数不停的购票更加的现实和有趣，我的意思是，购票人数不是一开始规定好的，而是票数是规定好的，然后人数是一点点用 random 添加进来的，每次添加一两个人来到窗口前买票更加的逼近于现实的情况。