

**并 行 计 算**

课程实验报告

实验名称： Python并发编程

任课教师： 徐悦甡老师

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

[**一、** **实验名称** 3](#_Toc514339557)

[**二、** **实验日期** 3](#_Toc514339558)

[**三、** **实验学生** 3](#_Toc514339559)

[**四、** **实验目的** 3](#_Toc514339560)

[**五、** **实验内容** 3](#_Toc514339561)

[**[1]** **第一题：** 3](#_Toc514339562)

[**[2]** **第二题：** 3](#_Toc514339563)

[**六、** **程序思路、结构** 3](#_Toc514339564)

[**[1]** **第一题：** 4](#_Toc514339565)

[**[2]** **第二题：** 4](#_Toc514339566)

[**七、** **程序代码** 5](#_Toc514339567)

[**[1]** **第一题代码：** 5](#_Toc514339568)

[**[2]** **}第二题代码：** 7](#_Toc514339569)

[**八、** **实验结果** 9](#_Toc514339570)

[**[1]** **第一题结果：** 9](#_Toc514339571)

[**[2]** **第二题结果：** 10](#_Toc514339572)

[**九、** **总结建议** 10](#_Toc514339573)

**软件学院本科生《并行计算》**

**课程实验报告**

1. **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）

1. **实验日期**

2018年06月27日 软件学院实验室G346

1. **实验学生**

15130120141 孙晖

1. **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。

1. **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()。

1. **第二题：**

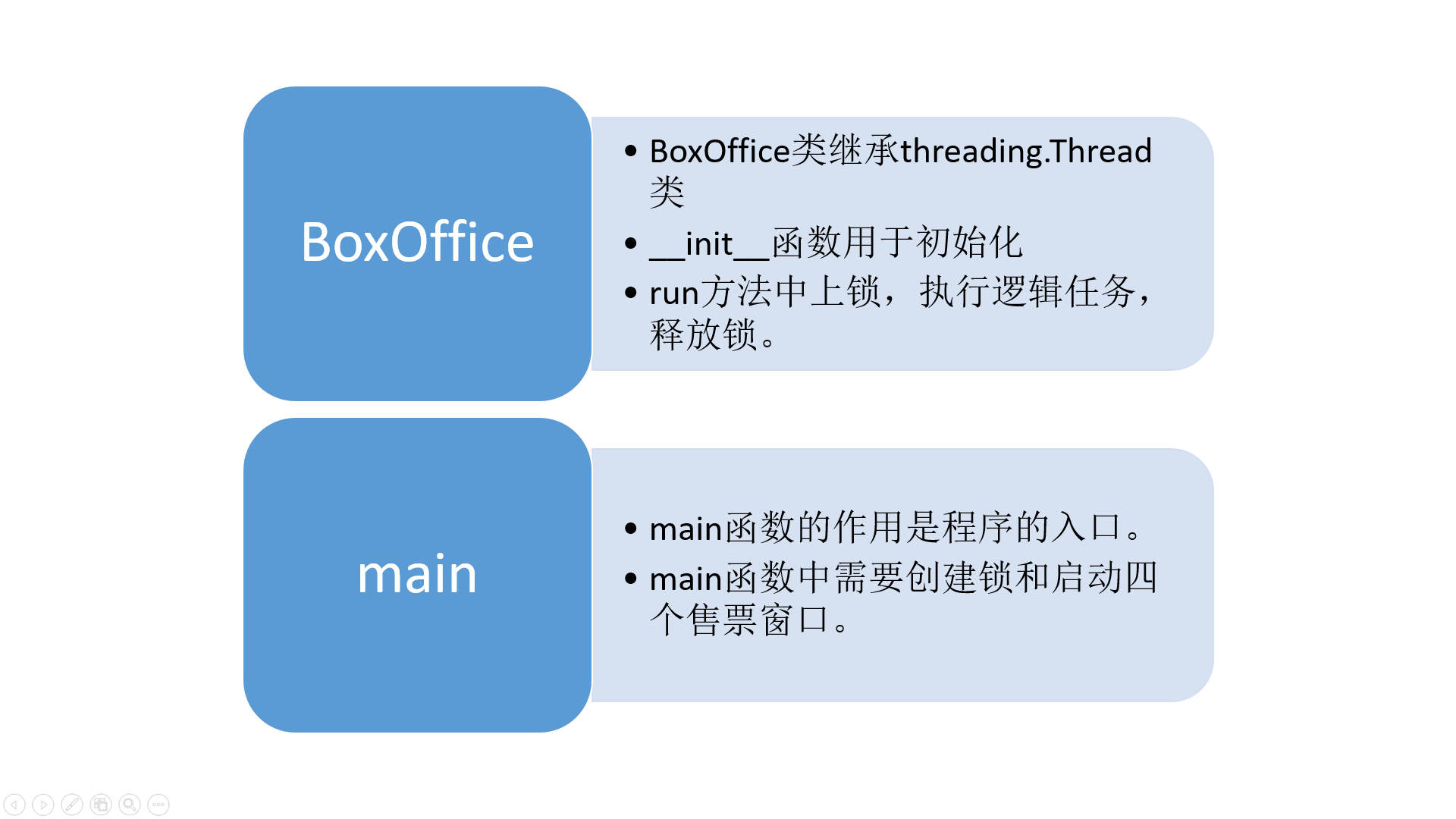
实验第二题要求用Python语言模拟两个人通过同一个账户取钱的情况，其中张三在柜台取钱，李四在ATM取钱。要求，创建一个Bank类；创建一个张三类代表在柜台取钱；创建一个李四类代表在ATM上取钱；创建主方法的调用类，也就是

if \_\_name\_\_="\_\_name\_\_"

1. **程序思路、结构**
2. **第一题：**

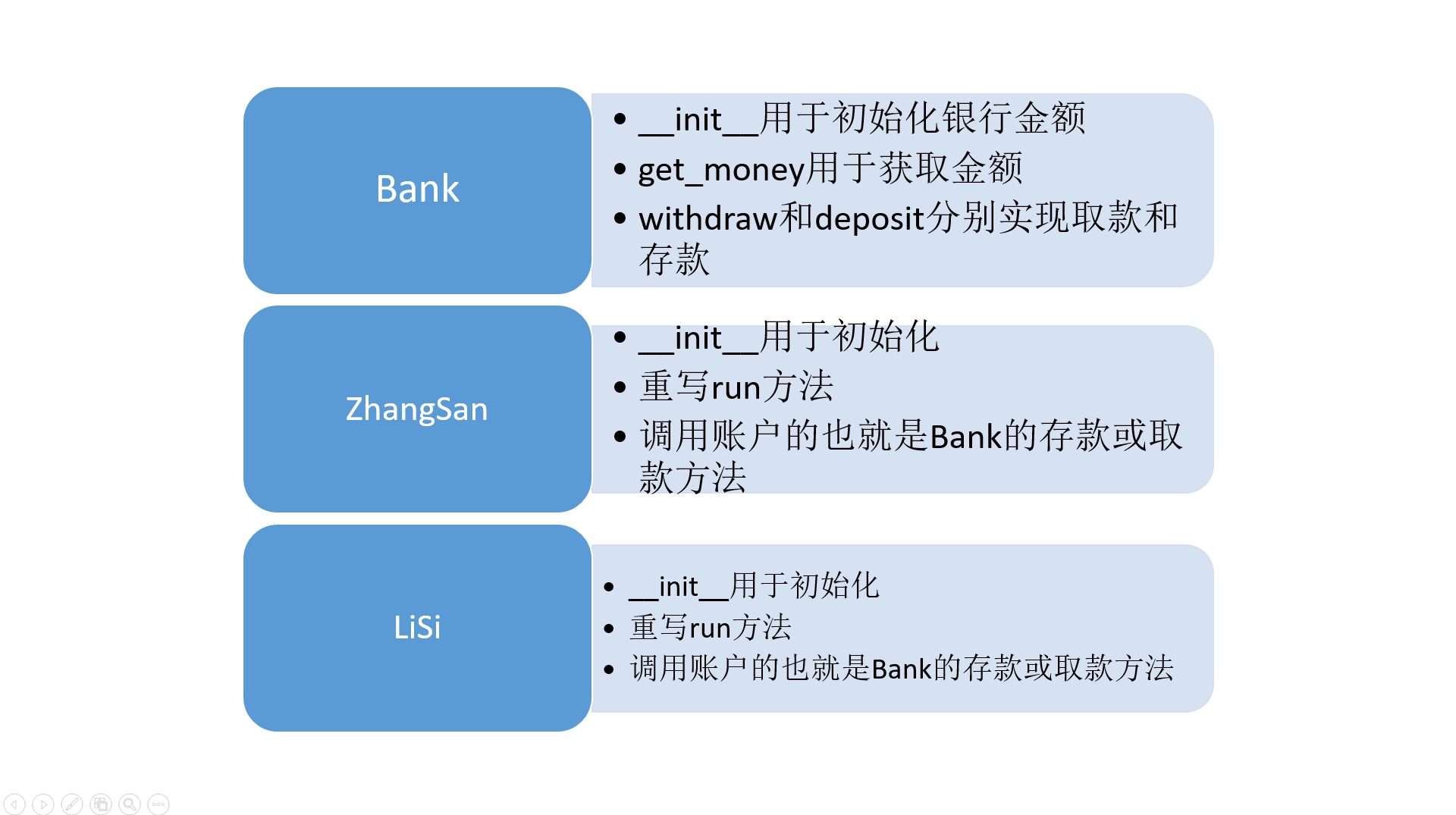
面对题目的第一反应应该是用什么方法实现java多线程，老师在课堂上讲了两种方法，一种是通过继承threading.Thread类，通过重写run方法实现想要实现的功能。另一种方法是通过把要执行的函数作为参数传递给threading.Thread()。依据老师PPT中给出的参考思路，要使用第一种方法来创建线程，也就是通过继承threading.Thread的方法。

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：

1. **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：

1. **程序代码**
2. **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()

1. **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)

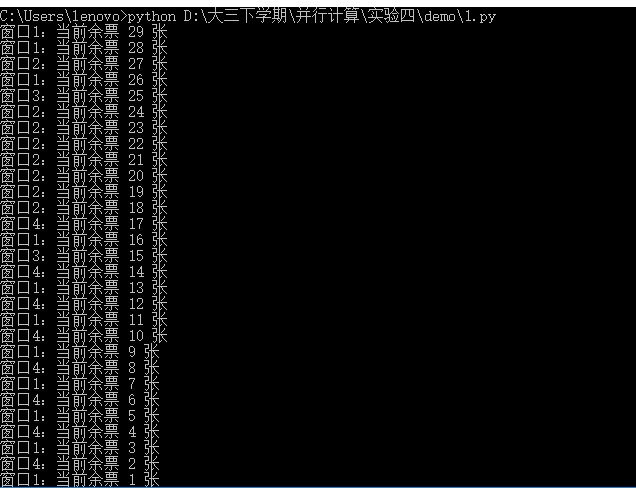
lisi = LiSi(account)

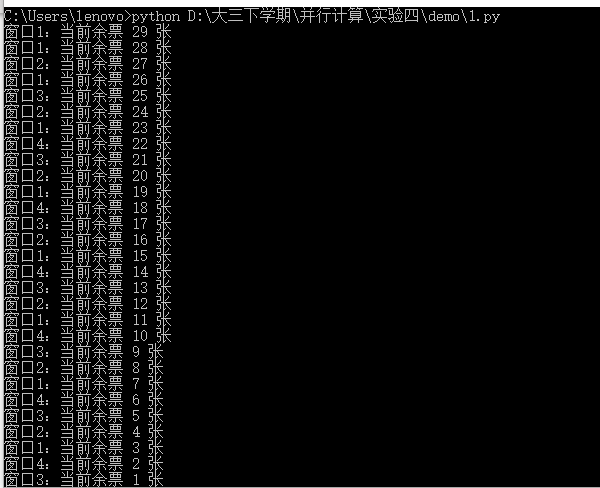
zhangsan.start()

lisi.start()

print("剩余%s元"%(account.get\_money()))

1. **实验结果**
2. **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：

****

1. **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。

1. **总结建议**
2. 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。
3. 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。
4. 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。