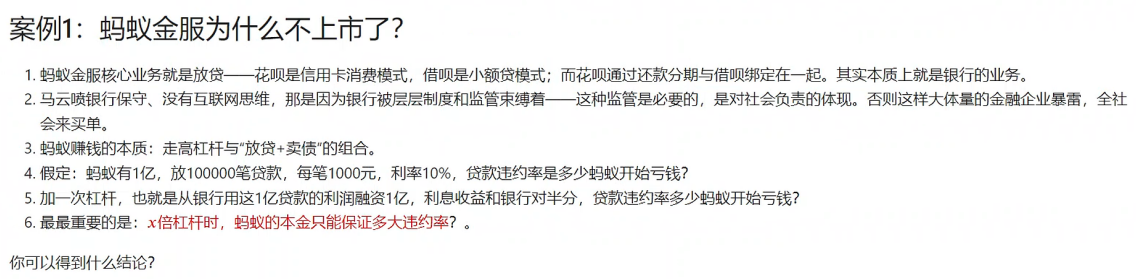
《Python 程序设计高阶》
(2023-2024 学年第 1 学期)
作业 1

学号:2021329600006　姓名: 陈昊天　班级: 计算机科学与技术 21(4) 班
学号:2021329621213　姓名: 陈佳伟　班级: 计算机科学与技术 21(3) 班
学号:2021329621257　姓名: 冯佳钧　班级: 计算机科学与技术 21(4) 班

§1　题目 1



案例1：蚂蚁金服为什么不上市了？

1. 蚂蚁金服核心业务就是放贷——花呗是信用卡消费模式，借呗是小额贷模式；而花呗通过还款分期与借呗绑定在一起。其实本质上就是银行的业务。
2. 马云喷银行保守、没有互联网思维，那是因为银行被层层制度和监管束缚着——这种监管是必要的，是对社会负责的体现。否则这样大体量的金融企业暴雷，全社会来买单。
3. 蚂蚁赚钱的本质：走高杠杆与"放贷+卖债"的组合。
4. 假定：蚂蚁有1亿，放100000笔贷款，每笔1000元，利率10%，贷款违约率是多少蚂蚁开始亏钱？
5. 加一次杠杆，也就是从银行用这1亿贷款的利润融资1亿，利息收益和银行对半分，贷款违约率多少蚂蚁开始亏钱？
6. 最最重要的是：x倍杠杆时，蚂蚁的本金只能保证多大违约率？。

你可以得到什么结论？

1. 根据案例 1 提供的内容，请你完成案例的第 4-6 题；
2. 然后根据以上的结论，你可以得到什么结论？

§1.1　解答

第 4 题

当违约的贷款金额达到总金额减去违约金额的 10% 时，蚂蚁开始亏钱

```python
p = 10**8
i = 1
while 1000 * i < (p - 1000 * i) * 0.1 and i <= 10**5:
    i += 1
print(f"违约数：{i}，违约率：{i / (10 ** 5)}")
# 违约数：9091，违约率：0.09091
```

第 5 题

当违约的贷款金额加上违约金额的 5%（因为和银行对半分，所以是 5%）达到总金额减去违约金额的 5% 时，蚂蚁开始亏钱

```python
# r=int(input('杠杆轮次:'))
r = 1
p = (10**8) * r
i = 1
while 1000 * i + 1000 * i * 0.05 < (p - 1000 * i) * 0.05 * r:  # 坏账大于蚂蚁收入
```

```
    i += 1
print(f"违约数: {i}, 违约率: {i/((10**5)*r)}")
# 违约数: 4546, 违约率: 0.04546
```

第 6 题

```
# r=int(input('杠杆轮次:'))
r = 10
p = (10**8) * r
i = 1
while 1000 * i + 1000 * i * 0.05 < (p - 1000 * i) * 0.05 * r:  # 坏账大于蚂蚁收入
    i += 1
print(f"违约数: {i}, 违约率: {i/((10**5)*r)}")
# 违约数: 322581, 违约率: 0.322581
```

§1.2 结论

杠杆倍数越大，蚂蚁能承受的违约率越小；蚂蚁将一部分利润和风险转嫁给银行和借贷者；蚂蚁可以用小规模的本金撬动大规模的社会资金 [1]。

§2 题目 2

CPU 乱序执行

统筹方法在 CPU 指令执行上的表现，就是乱序执行。所谓乱序执行，就是在后面的指令可能比前面的指令先执行。现在，请你设计一个程序，能够证明 CPU 执行代码时，是乱序执行的。

创建两个线程t1和t2，每个线程都修改了全局变量x和y的值，并打印出修改后的结果。由于 CPU 的乱序执行特性，每次运行程序时可能会产生不同的输出 [2]。

```
(base) nanmener@Haotians-MacBook-Pro ~ % /Applications/Anaconda3/anaconda3/bin/python /Users/nanmener/Github/python/main.py
Thread 1: x = 1
Thread 2: y = 2
Thread 2: x = 2
Thread 1: y = 1
Final result: x = 2 y = 1
```

```
(base) nanmener@Haotians-MacBook-Pro ~ % /Applications/Anaconda3/anaconda3/bin/python /Users/nanmener/Github/python/main.py
Thread 1: x = 1
Thread 1: y = 1
Thread 2: y = 2
Thread 2: x = 2
Final result: x = 2 y = 1
```

```
import threading
x,y = 0,0
def thread_1():
    global x, y
    x = 1
    print("Thread 1: x =", x)
    y = 1
    print("Thread 1: y =", y)
```

```
def thread_2():
    global x, y
    y = 2
    print("Thread 2: y =", y)
    x = 2
    print("Thread 2: x =", x)
# 创建两个线程
t1 = threading.Thread(target=thread_1)
t2 = threading.Thread(target=thread_2)
# 启动线程
t1.start()
t2.start()
# 等待线程结束
t1.join()
t2.join()
# 打印最终结果
print("Final result: x =", x, "y =", y)
```

§3 总结

§3.1 分工情况

以下说明小组的分工情况，每个人完成了什么功能，完成的情况如何

陈昊天

负责报告的整合撰写；

负责题目 2-CPU 乱序执行的研究和解答

陈佳伟

负责题目 1-第 4 题、第 5 题的研究和解答

冯佳钧

负责题目 1-第 6 题的研究和解答

§3.2 解决方案评价

以下说明小组对解决方案的评价，是否完成了题目的要求，怎么证明完成了题目的要求，好的地方在哪里，哪里有不足。

3.2.1 题目 1

基本完成题目要求；

主要问题在于杠杆倍数和轮数的定义，即“一倍”“一次”“一轮”杠杆的区别。按照理解，一倍杠杆即本金和借款相等，那么第 5 题的“加一次杠杆”的意思就是加 0.1 倍杠杆（因为借款 1 千万，本金 1 亿）

还有问题在于第 5 题一亿贷款的利润是否要减去坏账的亏损。如果是，那么就融资不到一亿

优点：使用到本节循环的知识；代码结构简单易懂；

缺点：第 6 题在杠杆倍率为 0 时的利率应该是 10%，即第 4 题的特例，在代码中没有体现

3.2.2 题目 2

基本完成题目要求，实现演示乱序执行的效果；

优点：

使用两个线程模拟 CPU 乱序执行的效果，代码逻辑清晰

缺点：

(1) 需要多次尝试观察结果

(2) 多线程乱序执行的现象并不能直接证明 CPU 的乱序执行。在多线程程序中，由于线程之间的竞争和调度的不确定性，线程的执行顺序可能会发生变化，导致最终结果的顺序不确定。

然而，现代 CPU 内部的乱序执行是一种与多线程乱序执行不同的概念。CPU 的乱序执行是指 CPU 在执行指令时，根据指令之间的依赖关系和可执行性进行重排序，以提高指令级并行度和执行效率。这种乱序执行是在单个线程的上下文中进行的，而不是多个线程之间的竞争。

要证明 CPU 的乱序执行，需要使用更加深入的底层技术和工具 [3]。

参考文献

[1] 徐仁平. 网络小额贷款法律监管问题研究——以“蚂蚁金服小额贷款”为例. *Advances in Social Sciences*, 11:1593, 2022.

[2] Quan Nguyen. *Mastering Concurrency in Python: Create faster programs using concurrency, asynchronous, multithreading, and parallel programming*. Packt Publishing Ltd, 2018.

[3] Wikipedia contributors. Out-of-order execution — Wikipedia, the free encyclopedia, 2023. [Online; accessed 24-October-2023].