**广州大学学生实验报告**

**开课学院及实验室：**网络空间安全学院 **2023年 4 月23 日**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **学院** | 网络空间安全学院 | **年级/专业/班** | 网安211 | **姓名** |  | **学号** |  |
| **实验课程名称** | 密码技术及应用 | | | | | **成绩** |  |
| **实验项目名称** | 序列密码加密实验 | | | | | **指导**  **老师** | 李树栋 |

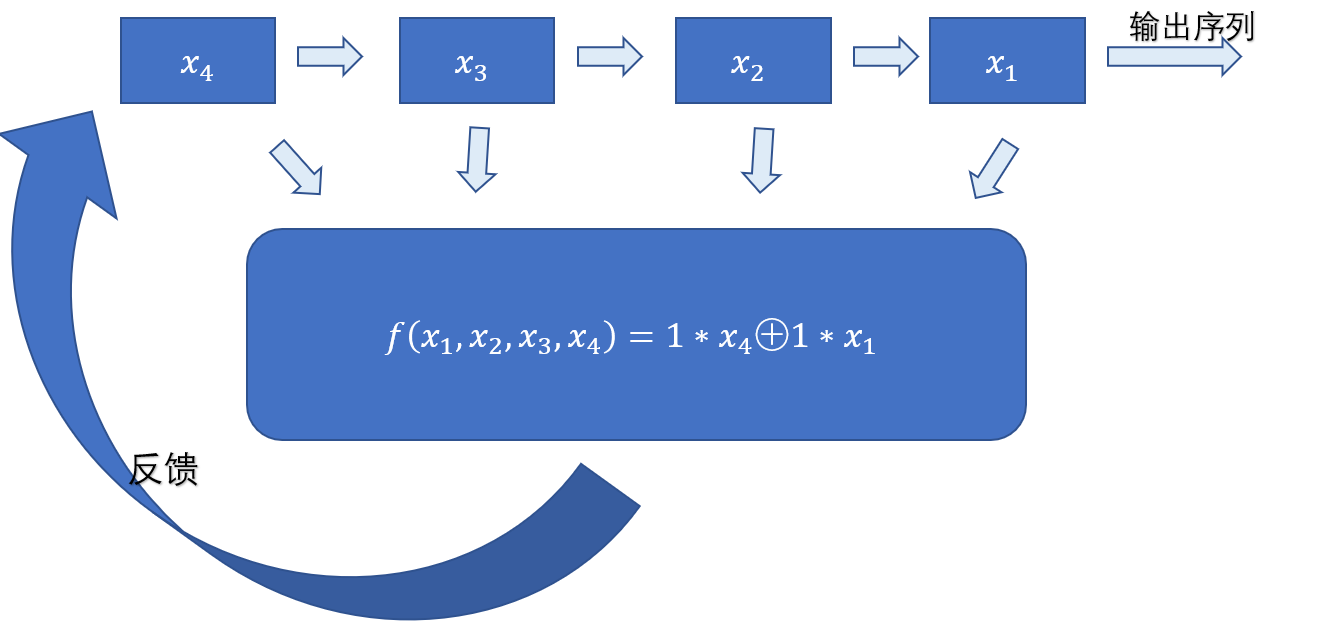
1. 实验目的

（1）了解线性反馈移位寄存器的数学原理及其周期分析；

（2）m序列的线性移位寄存器；

（3）线性移位寄存器的破解。

1. 实验内容
2. 根据图片用代码实现线性移位寄存器，其初始序列为1001。请编写程序实现该寄存器、打印其接下来的二十轮输出序列和状态序列，并给出其周期长度。



1. 现有三位线性移位寄存器，其初始状态为100，明文为010001，密文为100101。请编写程序，构造矩阵并运算求出该线性移位寄存器的系数。
2. 实验过程及结果

3.1 同学们把你们的实验过程及结果写在下面

3.2

# 定义一个函数lfsr，接受初始状态initial\_state和迭代轮数rounds作为参数

def lfsr(initial\_state, rounds):

    # 将初始状态赋值给状态变量state

    state = initial\_state

    # 初始化一个空的列表，用于存储输出序列

    output = []

    # 迭代轮数次，生成LFSR序列

    for \_ in range(rounds):

        # 计算反馈位，即第0位和第3位异或的结果

        feedback = state[0] ^ state[3]

        # 将反馈位添加到状态序列的末尾，并删除状态序列的第一个元素

        state = state[1:] + [feedback]

        # 将反馈位添加到输出序列中

        output.append(feedback)

    # 返回输出序列和最终状态序列

    return output, state

# 定义一个函数find\_cycle，接受初始状态initial\_state作为参数

def find\_cycle(initial\_state):

    # 将初始状态赋值给状态变量state

    state = initial\_state

    # 初始化周期长度为0

    cycle = 0

    # 循环计算LFSR序列，直到状态序列与初始状态序列相等

    while True:

        # 计算反馈位，即第0位和第3位异或的结果

        feedback = state[0] ^ state[3]

        # 将反馈位添加到状态序列的末尾，并删除状态序列的第一个元素

        state = state[1:] + [feedback]

        # 增加周期长度

        cycle += 1

        # 如果状态序列与初始状态序列相等，退出循环

        if state == initial\_state:

            break

    # 返回周期长度

    return cycle

# 定义初始状态和迭代轮数

initial\_state = [1, 0, 0, 1]

rounds = 20

# 调用函数lfsr，生成LFSR序列并打印输出序列和最终状态序列

output, final\_state = lfsr(initial\_state, rounds)

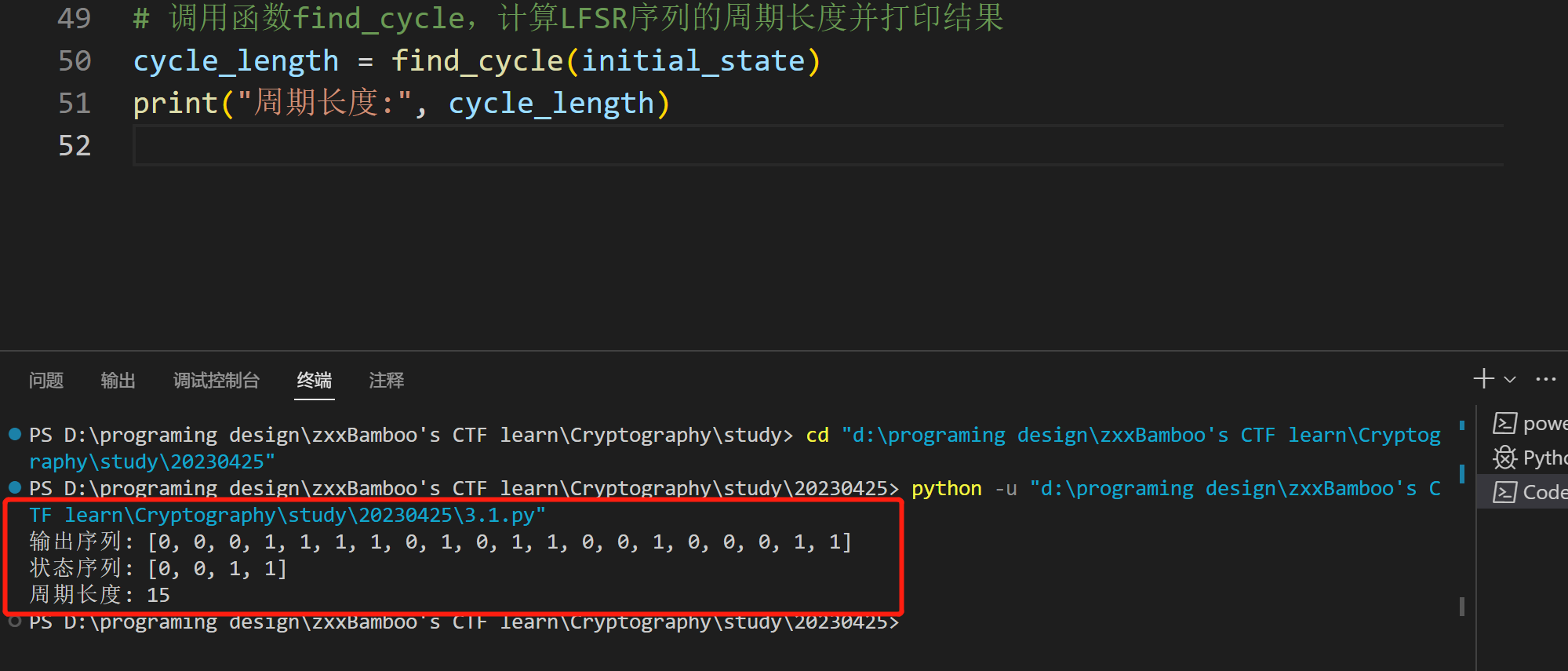
print("输出序列:", output)

print("状态序列:", final\_state)

# 调用函数find\_cycle，计算LFSR序列的周期长度并打印结果

cycle\_length = find\_cycle(initial\_state)

print("周期长度:", cycle\_length)



3.3

# 导入NumPy库

import numpy as np

# 定义一个函数find\_coefficients，接受明文plaintext、密文ciphertext和初始状态initial\_state作为参数

def find\_coefficients(plaintext, ciphertext, initial\_state):

    # 将初始状态赋值给状态变量state，并获取状态长度n

    state = initial\_state

    # n = len(initial\_state)

    # 初始化两个空的列表，用于存储矩阵A和向量B的元素

    A = []

    B = []

    # 遍历明文和密文的每个字符，生成矩阵A和向量B

    for p, c in zip(plaintext, ciphertext):

        # 将当前状态添加到矩阵A的末尾

        A.append(state)

        # 将密文和明文的异或结果添加到向量B中

        B.append(c ^ p)

        # 更新状态，将当前明文添加到状态序列的末尾，并删除状态序列的第一个元素

        state = state[1:] + [p]

    # 将列表A和B转换为NumPy数组

    A = np.array(A)

    B = np.array(B)

    coefficients, \_, \_, \_ = np.linalg.lstsq(A, B, rcond=None)

    return coefficients.round().astype(int)

# 定义初始状态、明文和密文

initial\_state = [1, 0, 0]

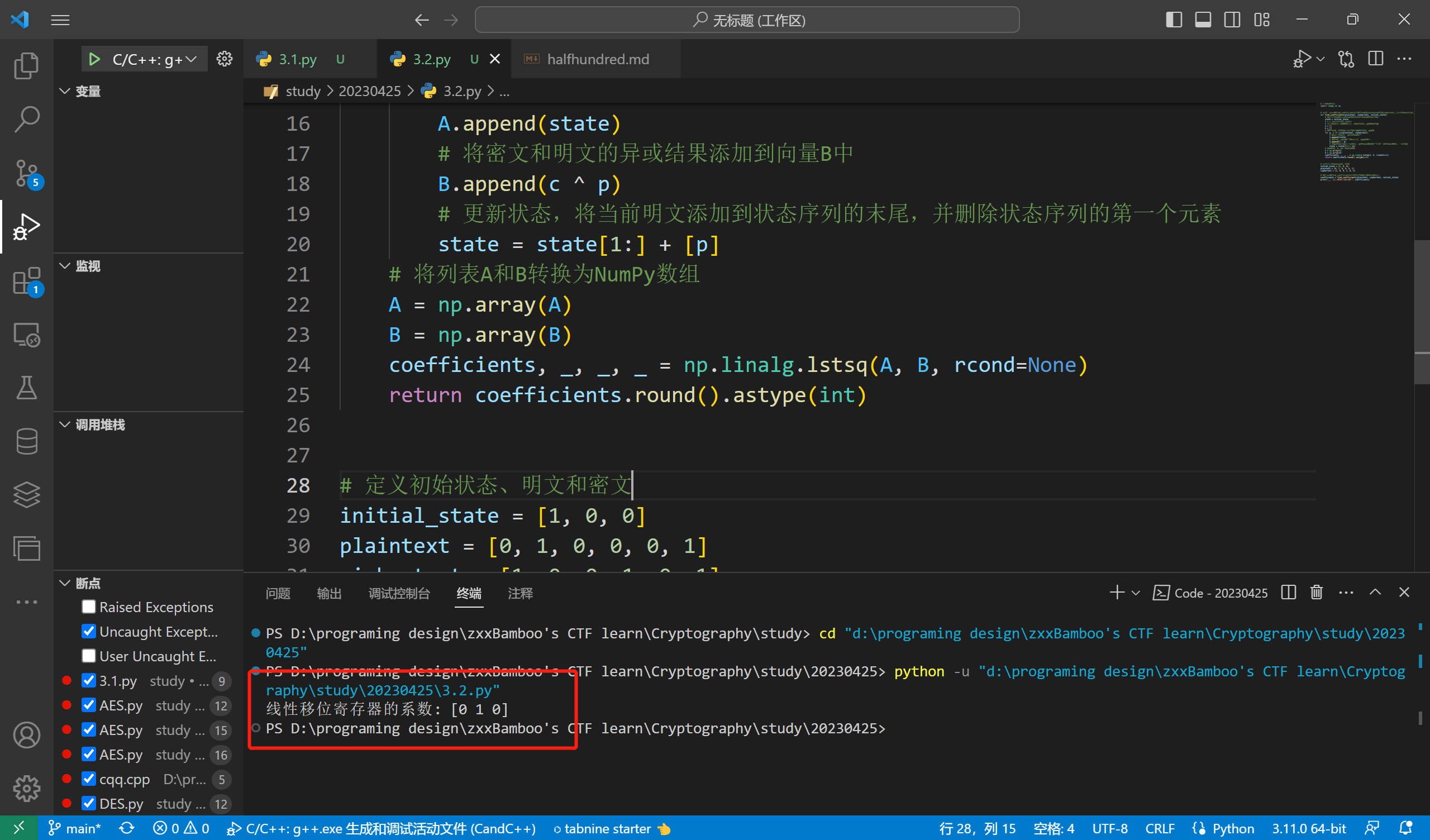
plaintext = [0, 1, 0, 0, 0, 1]

ciphertext = [1, 0, 0, 1, 0, 1]

# 调用函数find\_coefficients，破解LFSR的系数并打印结果

coefficients = find\_coefficients(plaintext, ciphertext, initial\_state)

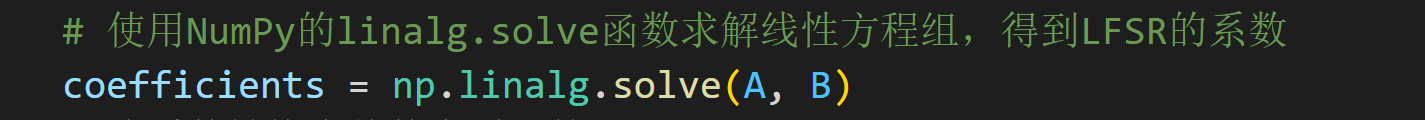
print("线性移位寄存器的系数:", coefficients)



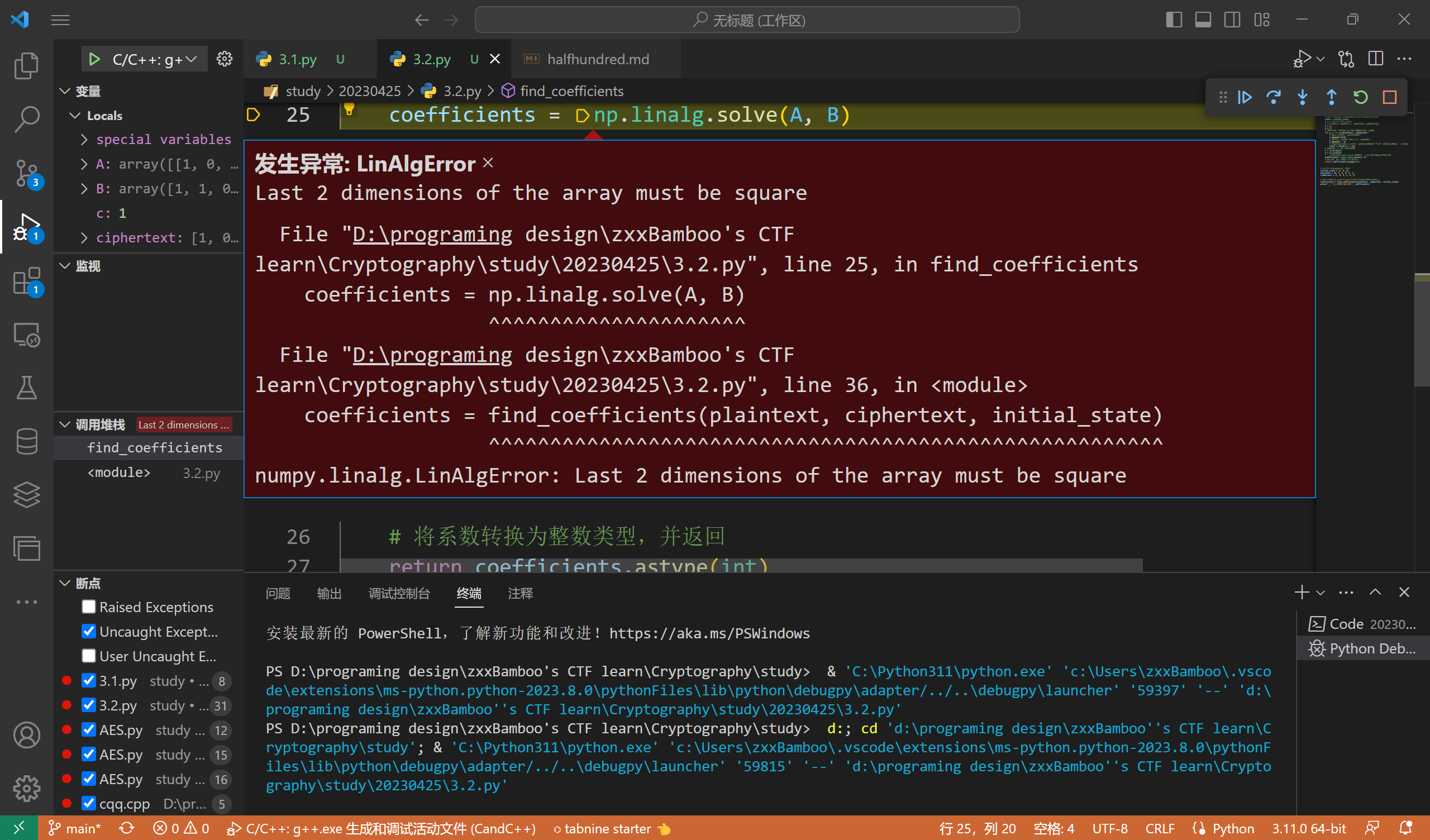
1. 实验总结

4.1 同学们把你们的实验过程及结果写在下面

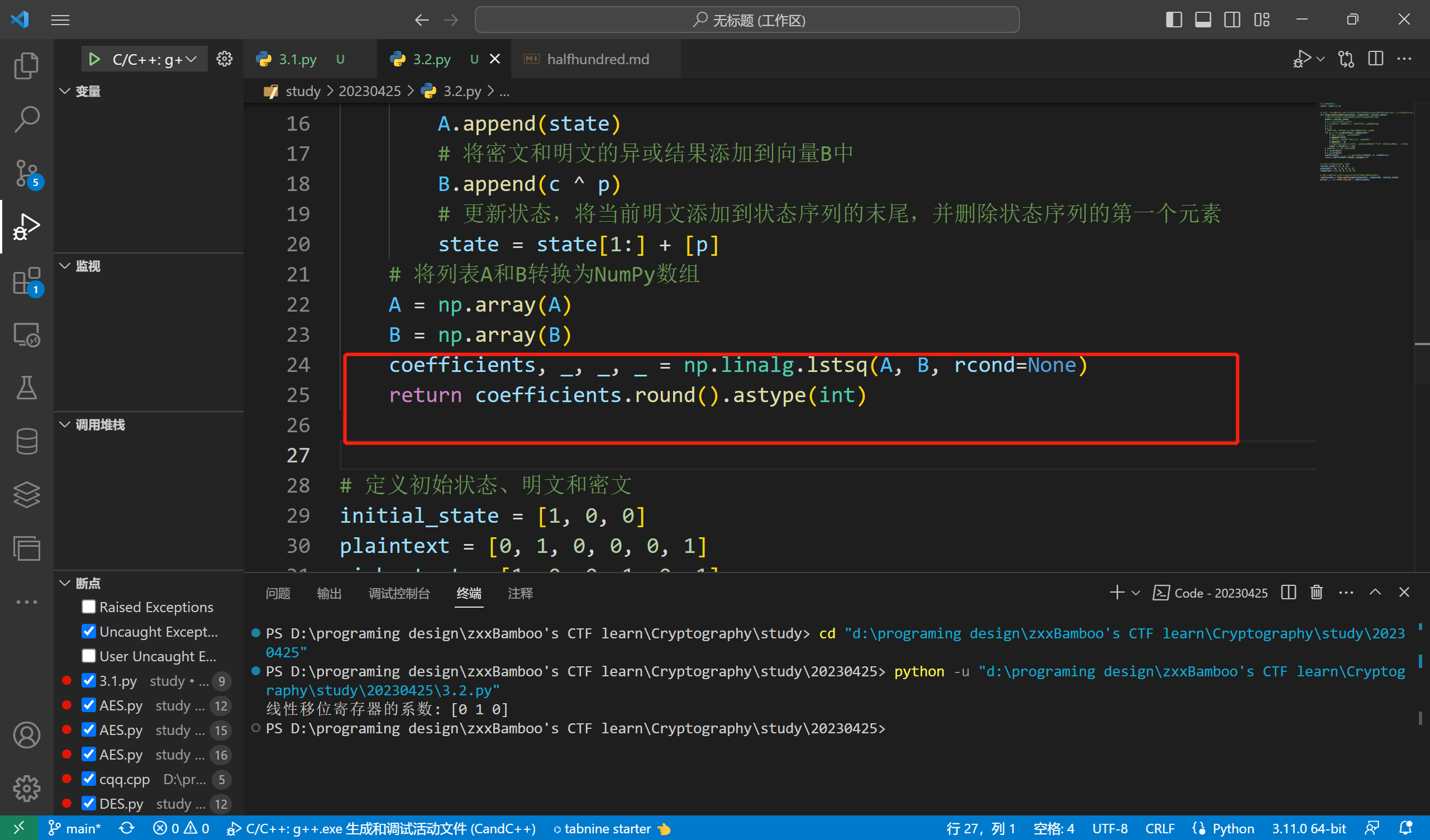
第二个实验时：



报错，



因为不能直接使用numpy.linalg.solve来求解这个问题，因为A矩阵不是方阵。应该使用numpy.linalg.lstsq来求解最小二乘问题。



使用numpy.linalg.lstsq求解最小二乘问题，并使用coefficients.round().astype(int)将系数四舍五入并转换为整数。