班级：计算机科学与技术21留学生1 学号：2021529620004姓名：LOW REN HONG

实验二 动态规划

**一、实验目的**

1、理解动态规划算法的概念；

2、掌握动态规划算法的基本要素；

3、掌握设计动态规划算法的步骤；

4、通过应用范例学习动态规划算法的设计技巧与策略。

**二、实验内容和要求**

**实验要求**：通过上机实验进行算法实现，保存和打印出程序的运行结果，并结合程序进行分析，上交实验报告和程序文件。

**实验内容**：

**1、最长公共子序列问题**

给定两个序列X={x1,x2,…,xm}和Y={y1,y2,…,yn}，找出X和Y的最长公共子序列。

**2、机器分配**

题目描述：总公司拥有高效设备M台，准备分给下属的N个分公司。各分公司若获得这些设备，可以为国家提供一定的盈利。问：如何分配这M台设备才能使国家得到的盈利最大？求出最大盈利值。其中M≤15，N≤10。分配原则：每个公司有权获得任意数目的设备，但总台数不超过设备数M。

输入

输入数据文件格式为：第一行有两个数，第一个数是分公司数N，第二个数是设备台数M。

接下来是一个N\*M的矩阵，表明了第I个公司分配J台机器的盈利。

输出

输出第一行是总盈利

然后n行每行2个数，第1个数是公司序号，第2个数是该公司分配到的机器数

P.S.要求答案的字典序最小

样例输入

3 3

30 40 50

20 30 50

20 25 30

样例输出

70

1 1

2 1

3 1

**3、潜水员**

题目描述：潜水员为了潜水要使用特殊的装备。他有一个带2种气体的气缸：一个为氧气，一个为氮气。让潜水员下潜的深度需要各种的数量的氧和氮。潜水员有一定数量的气缸。每个气缸都有重量和气体容量。潜水员为了完成他的工作需要特定数量的氧和氮。他完成工作所需气缸的总重的最低限度的是多少？

例如：潜水员有5个气缸。每行三个数字为：氧，氮的（升）量和气缸的重量：

3 36 120

10 25 129

5 50 250

1 45 130

4 20 119

如果潜水员需要5升的氧和60升的氮则总重最小为249（1，2或者4，5号气缸）。

你的任务就是计算潜水员为了完成他的工作需要的气缸的重量的最低值。

输入

第一行有2整数m,n（1<=m<=21,1<=n<=79）。它们表示氧，氮各自需要的量。

第二行为整数k（1<=n<=1000）表示气缸的个数。

此后的k行，每行包括ai，bi，ci（1<=ai<=21，1<=bi<=79，1<=ci<=800）3整数。这些各自是：第i个气缸里的氧和氮的容量及汽缸重量。

样例输入

1 1

1

1 1 1

样例输出

1

1. **程序代码  
     
   1)**def longest\_common\_subsequence(x,y):  
    m=len(x)  
    n=len(y)  
     
    #创建一个二维数组并初始化为0  
    dp=[[0]\*(n+1)for \_ in range (m+1)]  
     
    #使用动态规划来填充数组  
    for i in range(1,m+1):  
    for j in range(1,n+1):  
    if x[i-1]==y[j-1]:  
    dp[i][j]=dp[i-1][j-1]+1  
    else:  
    dp[i][j]=max(dp[i-1][j],dp[i][j-1])  
     
    #构造最长公共子序列  
    lcs=[]  
    i=m  
    j=n  
    while i>0 and j>0:  
    if x[i-1]==y[j-1]:  
    lcs.insert(0,x[i-1])  
    i-=1  
    j-=1  
    elif dp[i-1][j]>dp[i][j-1]:  
    i-=1  
    else:  
    j-=1  
    return ''.join(lcs)  
     
   #让用户输入序列X和Y的值  
   x=input("请输入序列X的值：")  
   y=input("请输入序列Y的值：")  
     
   result= longest\_common\_subsequence(x,y)  
   print("最长公共子序列：",result)

**2)** def allocate\_equipment(N, M, profits): #使用动态规划求解最大盈利

dp = [[0] \* (M + 1) for \_ in range(N + 1)] #创建一个二维数组，用于保存最大盈利

for i in range(1, N + 1): #使用动态规划填充数组

for j in range(1, M + 1): #遍历每个公司i和机器j的数量

for k in range(j + 1): #在j机器下，对于i公司遍历分配k台机器

if profits[i][k] + dp[i - 1][j - k] > dp[i][j]: #如果分配k台机器的盈利加上前i-1个公司在剩余j-k台机器下的最大盈利大于当前最大盈利，更新最大盈利

dp[i][j] = profits[i][k] + dp[i - 1][j - k]

# 回溯找出分配方案

allocation = []

i = N

j = M

while i > 0 and j > 0: # 从dp[N][M]开始，逆向寻找最大盈利时的分配方案

for k in range(j, 0, -1): # 通过比较profits[i][k] + dp[i-1][j-k]与dp[i][j]的大小，确定分配机器k的数量

if profits[i][k] + dp[i - 1][j - k] == dp[i][j]:

allocation.append((i, k))

i -= 1

j -= k

break

# 反转分配方案并计算总盈利

total\_profit = dp[N][M]

allocation.reverse()

return total\_profit, allocation # 返回总盈利和分配方案的元组

# 读取输入数据

N, M = map(int, input("请输入M和N：").split())

profits = [[0] \* (M + 1) for \_ in range(N + 1)] # 创建一个二维数组profits，用于保存每个公司在不同机器数量下的盈利

for i in range(1, N + 1):

row = list(map(int, input("输入M\*N的矩阵：").split())) # 输入每个公司在不同机器数量下的盈利

for j in range(1, M + 1): # 将盈利值存储在profits数组中

profits[i][j] = row[j - 1]

# 调用函数计算最大盈利和分配方案

max\_profit, allocation = allocate\_equipment(N, M, profits)

# 输出最大盈利

print(max\_profit)

# 输出分配方案

for company, machines in allocation:

print(company, machines)

**3)**def minimum\_weight(m, n, k, cylinders):

# 创建一个二维数组dp，用于记录达到特定氧和氮量时的最小重量

INF = float('inf')

dp = [[INF] \* (n + 1) for \_ in range(m + 1)] # 创建一个二维数组dp

dp[0][0] = 0 #初始值为正无穷

# 动态规划求解最小重量

for c in cylinders:

oxygen, nitrogen, weight = c

for i in range(m, -1, -1):

for j in range(n, -1, -1): # 更新dp[i][j]，取当前值和通过加入当前气缸后的重量的较小值

dp[i][j] = min(dp[i][j], dp[max(0, i - oxygen)][max(0, j - nitrogen)] + weight)

return dp[m][n] # 返回达到目标氧量m和氮量n时的最小重量

# 读取输入数据

m, n = map(int, input("氧，氮各自的需要：").split())

k = int(input("气缸："))

cylinders = []

for \_ in range(k): # 输入每个气缸的氧量、氮量和重量，存储在cylinders列表中

cylinder = list(map(int, input("气缸里的氧气和氮气的容量和气缸的重量：").split()))

cylinders.append(cylinder)

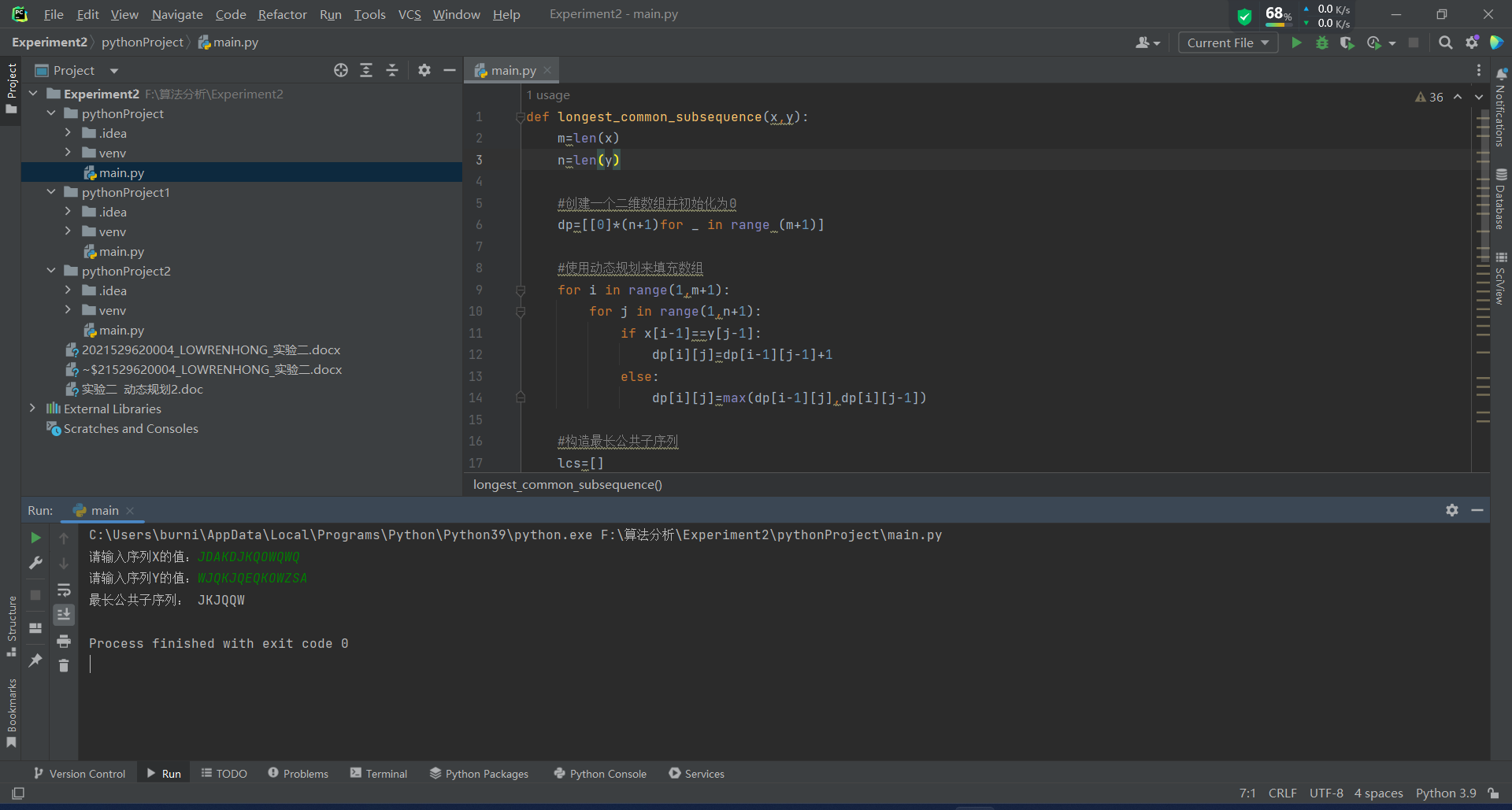
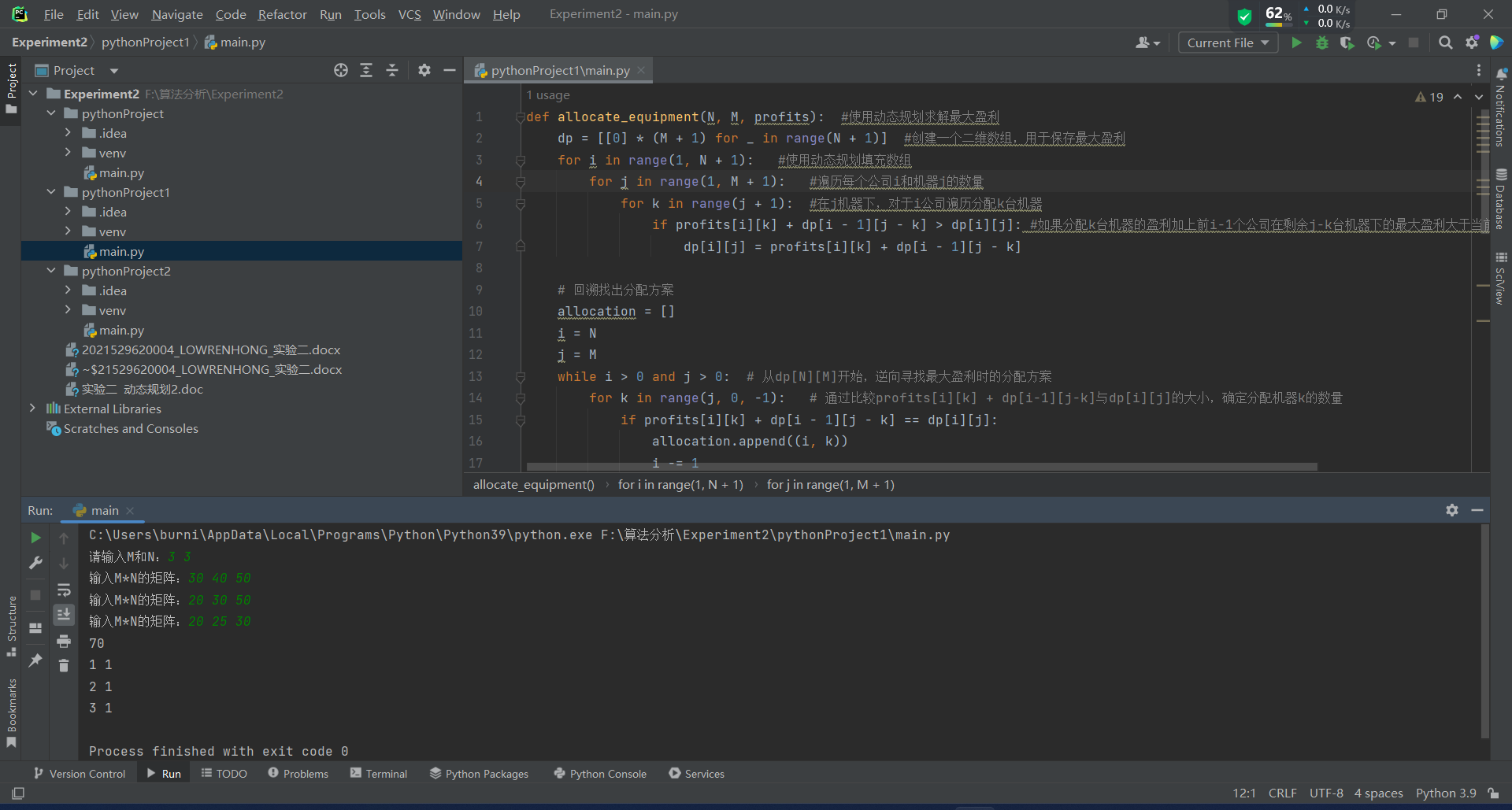
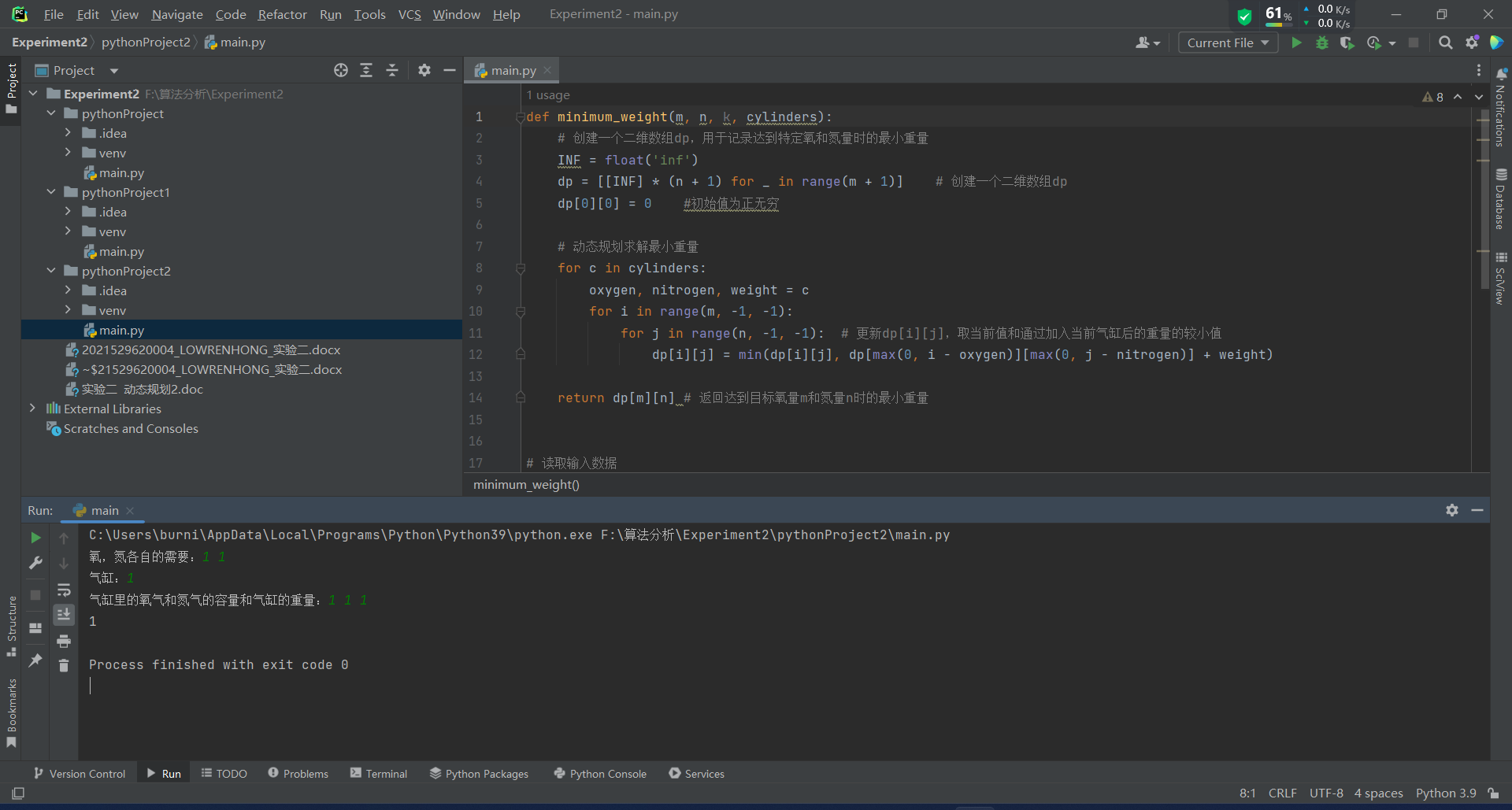
# 调用函数计算最小重量

min\_weight = minimum\_weight(m, n, k, cylinders)

# 输出最小重量

print(min\_weight)

**四、结果运行与分析**

**五、心得与体会**本次实验，我学会了动态规划的概念，解决具有重叠子问题的最优子结构特性的问题算法分析。通过实验的编程代码，加深了对这些概念的理解，并通过实现代码加深了对算法的理解以及掌握。动态规划的基本思想是将原问题分解为若干个子问题，先解决子问题并将其结果存储起来，然后利用子问题的结果推导出原问题的解。通过构建一个表格或数组来存储子问题的解，可以避免重复计算，并且在推导原问题解时可以直接利用已经求解过的子问题的解，从而减少时间复杂度。