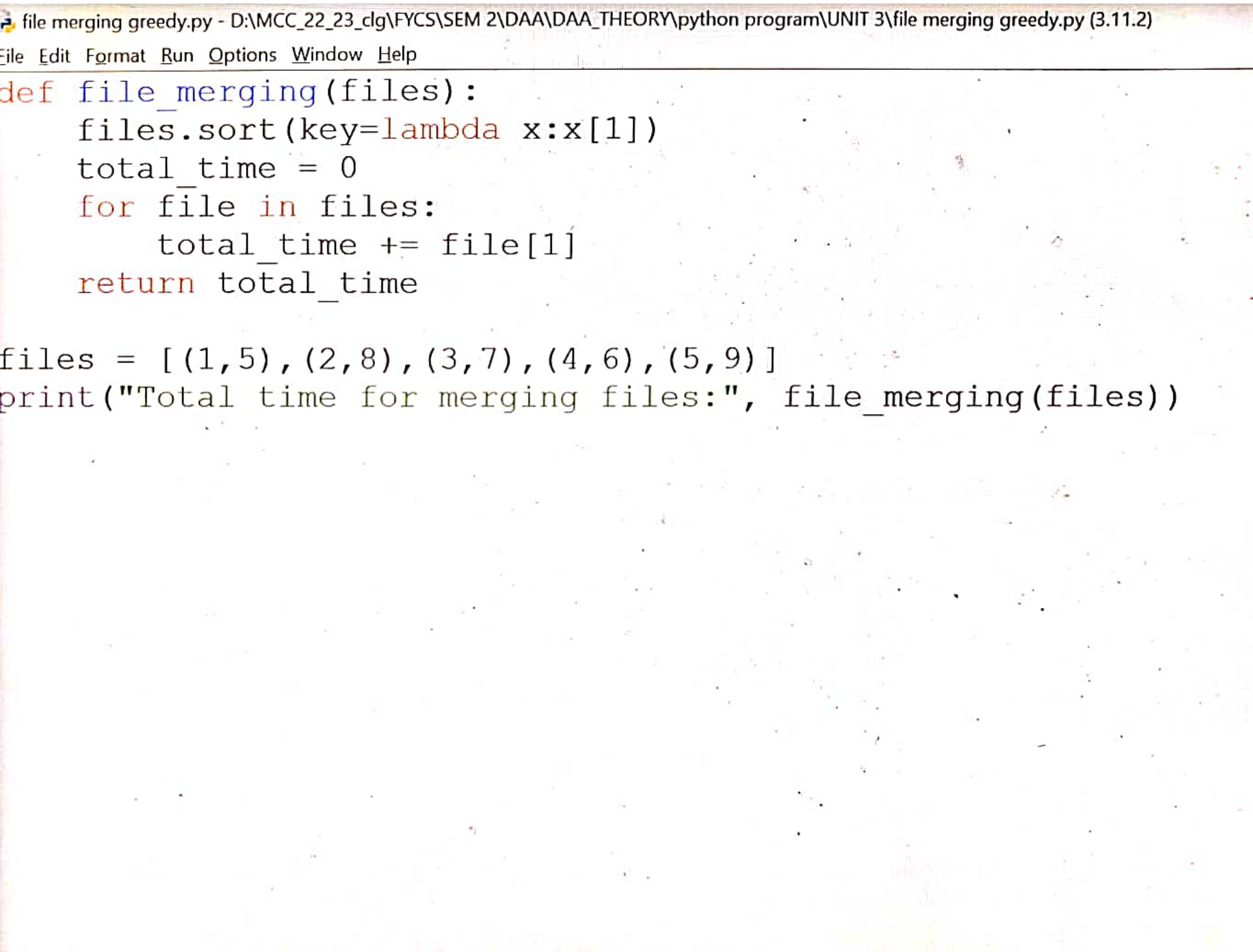
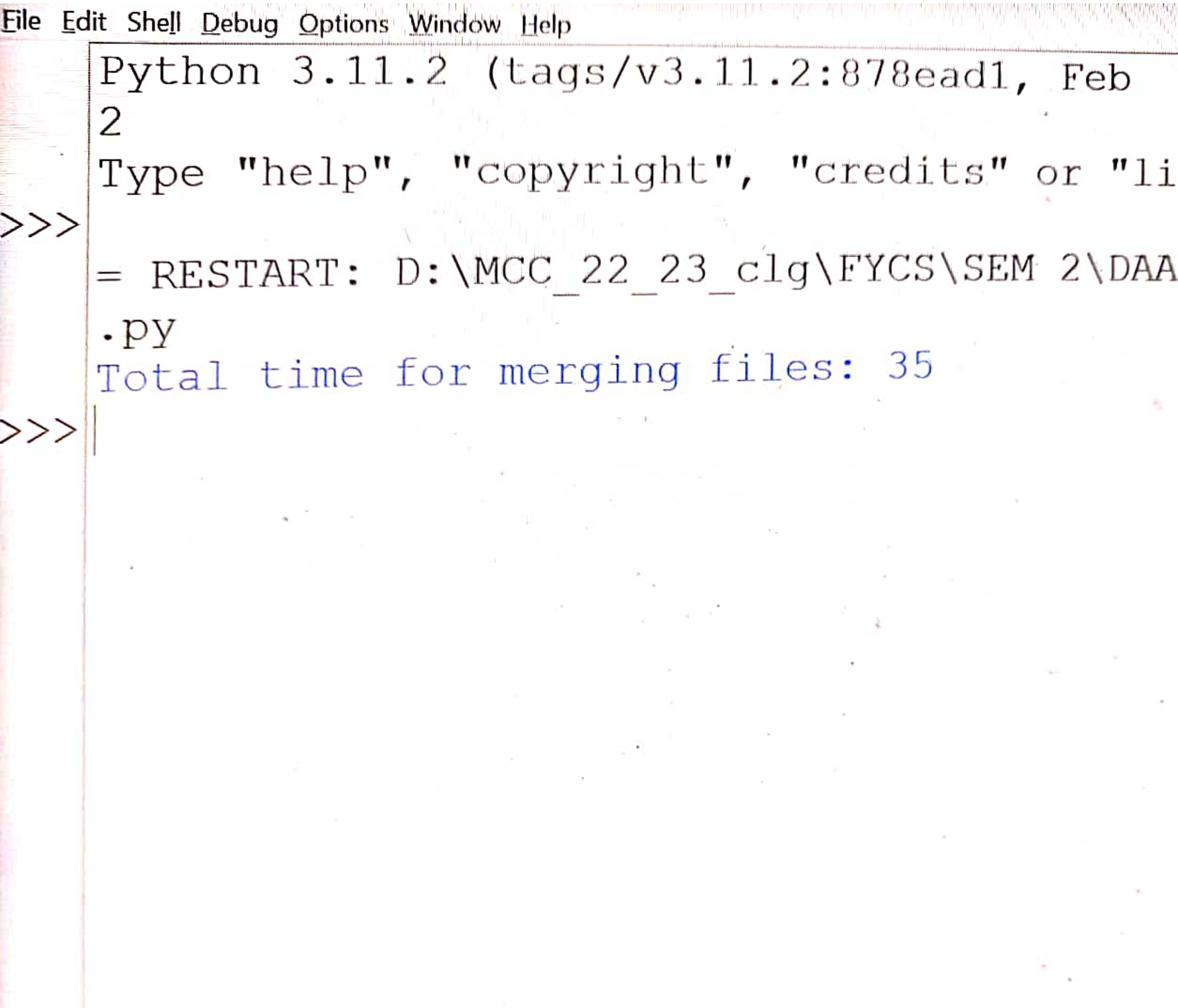
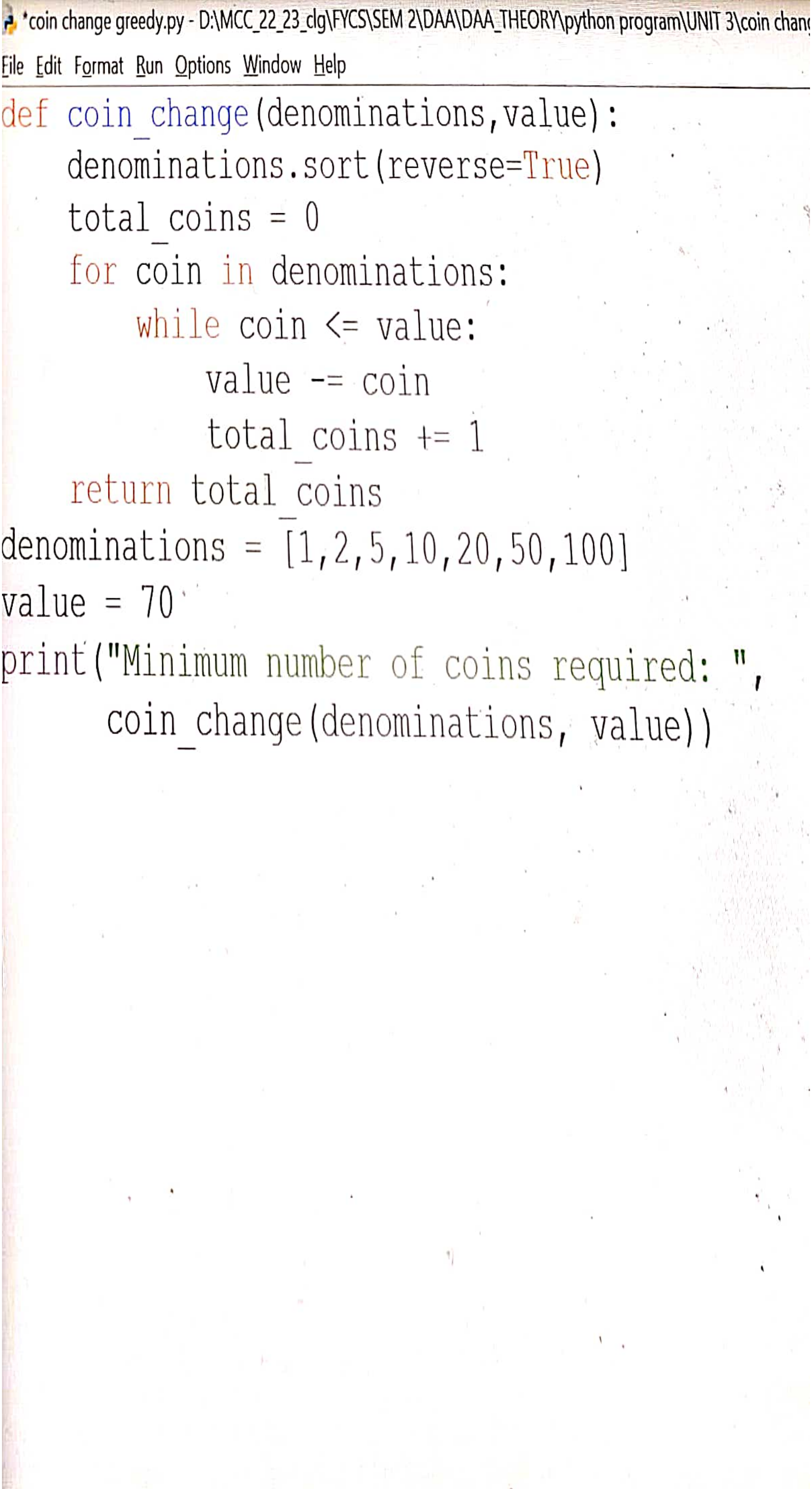

file merging greedy.py - D:\MCC_22_23_clg\FYCS\SEM 2\DAA\DAA_THEORY\python program\UNIT 3\file merging greedy.py (3.11.2)
File Edit Format Run Options Window Help

```python
def file_merging(files):
    files.sort(key=lambda x:x[1])
    total_time = 0
    for file in files:
        total_time += file[1]
    return total_time

files = [(1,5),(2,8),(3,7),(4,6),(5,9)]
print("Total time for merging files:", file_merging(files))
```

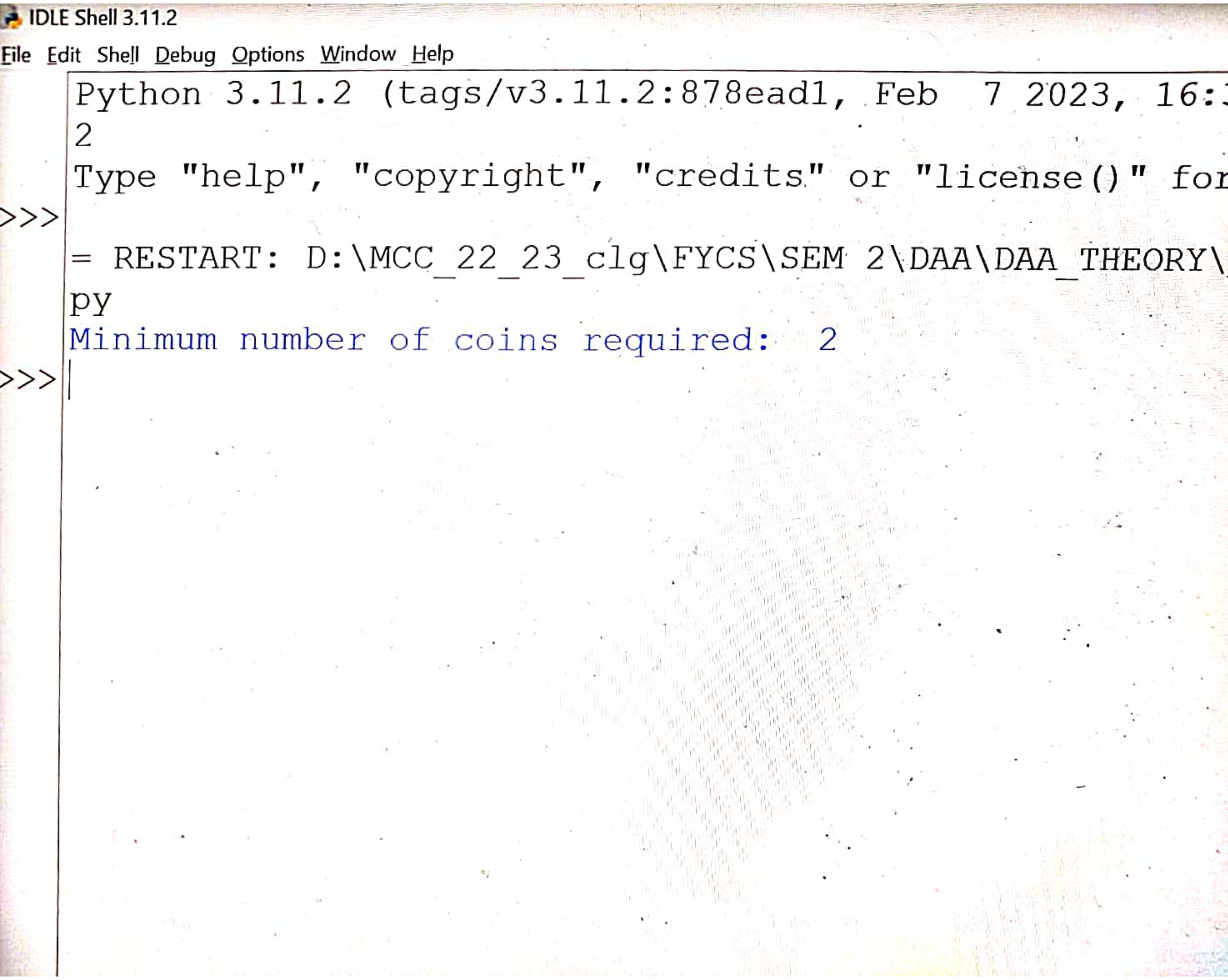


```
Python 3.11.2 (tags/v3.11.2:878ead1, Feb
2
Type "help", "copyright", "credits" or "li
>>>
= RESTART: D:\MCC_22_23_clg\FYCS\SEM 2\DAA
.py
Total time for merging files: 35
>>>
```

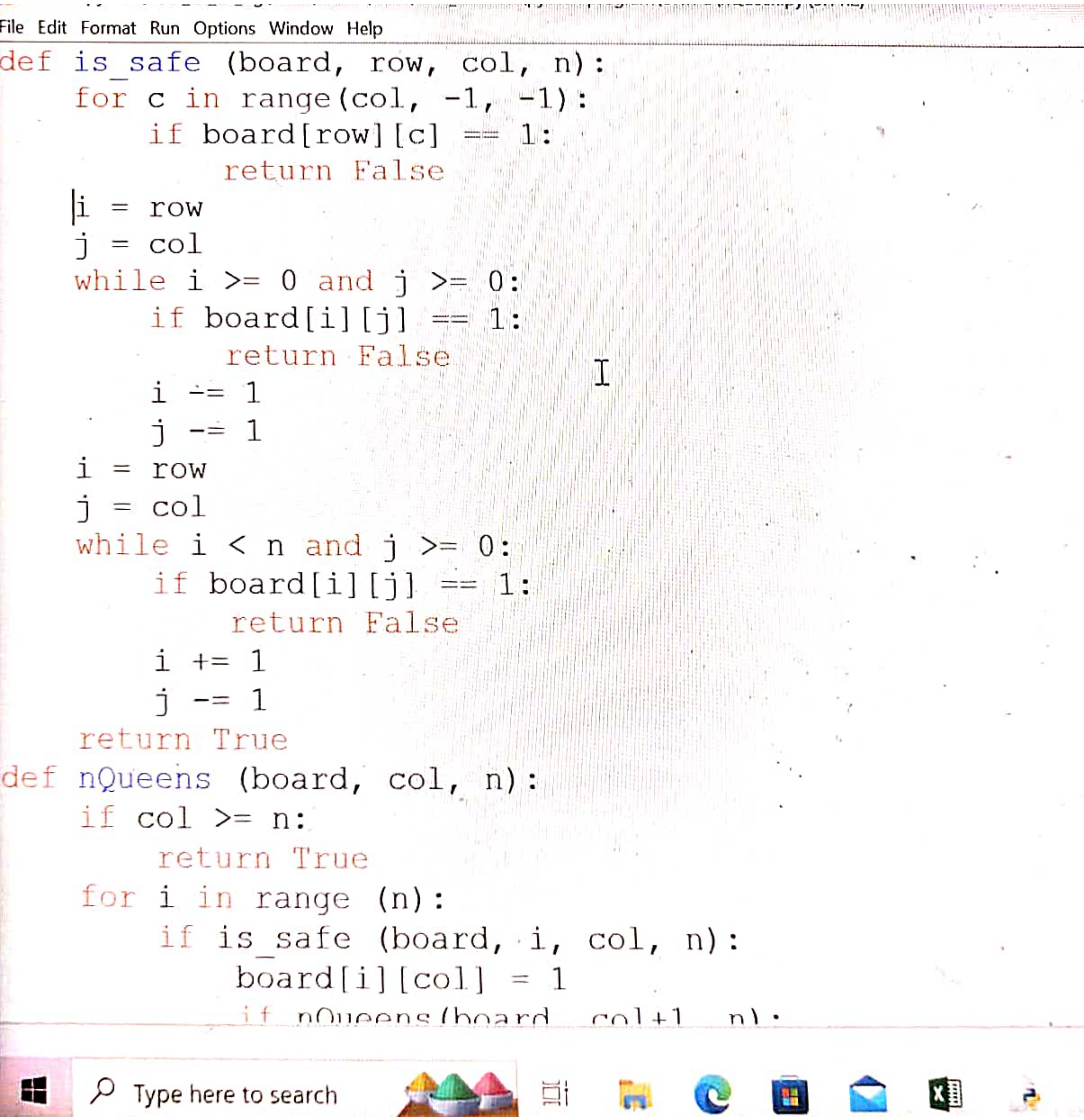


*coin change greedy.py - D:\MCC_22_23_clg\FYCS\SEM 2\DAA\DAA_THEORY\python program\UNIT 3\coin chan

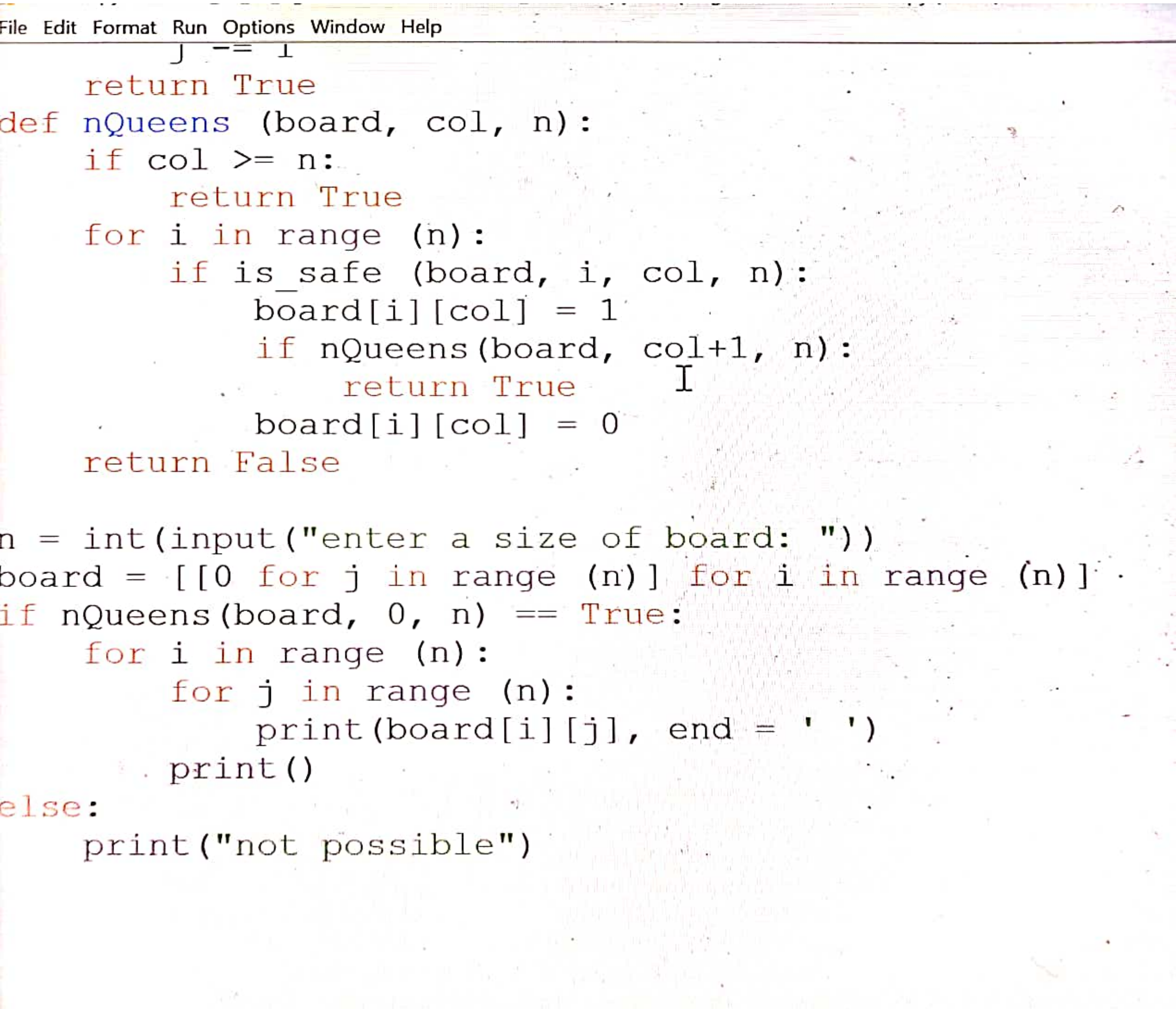
```
def coin_change(denominations,value):
    denominations.sort(reverse=True)
    total_coins = 0
    for coin in denominations:
        while coin <= value:
            value -= coin
            total_coins += 1
    return total_coins
denominations = [1,2,5,10,20,50,100]
value = 70
print("Minimum number of coins required: ",
      coin_change(denominations, value))
```



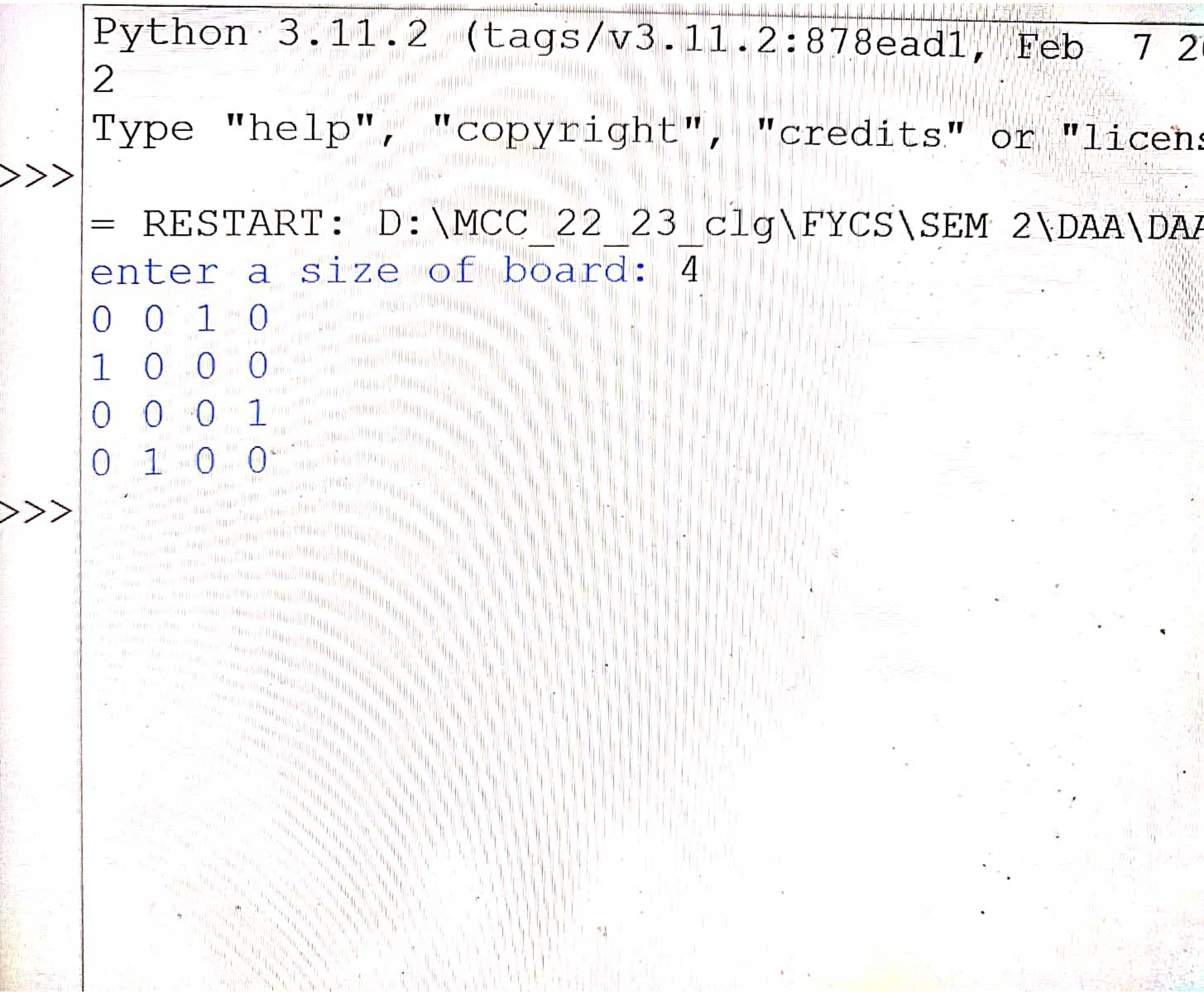
```
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023, 16:3
2
Type "help", "copyright", "credits" or "license()" for
>>>
= RESTART: D:\MCC_22_23_clg\FYCS\SEM 2\DAA\DAA_THEORY\
py
Minimum number of coins required:  2
>>>
```



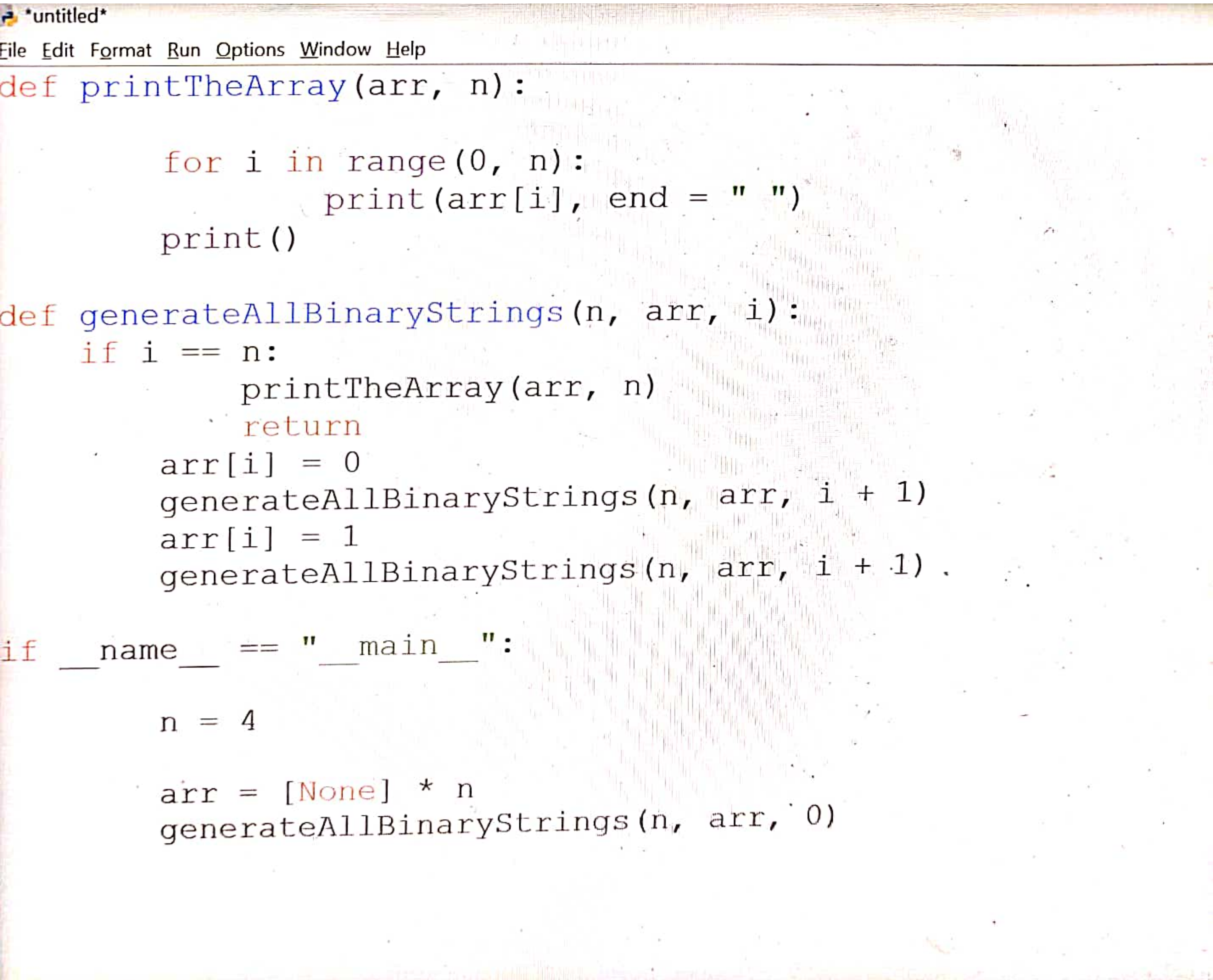
```
def is_safe (board, row, col, n):
    for c in range(col, -1, -1):
        if board[row][c] == 1:
            return False
    i = row
    j = col
    while i >= 0 and j >= 0:
        if board[i][j] == 1:
            return False
        i -= 1
        j -= 1
    i = row
    j = col
    while i < n and j >= 0:
        if board[i][j] == 1:
            return False
        i += 1
        j -= 1
    return True
def nQueens (board, col, n):
    if col >= n:
        return True
    for i in range (n):
        if is_safe (board, i, col, n):
            board[i][col] = 1
            if nQueens(board, col+1, n):
```



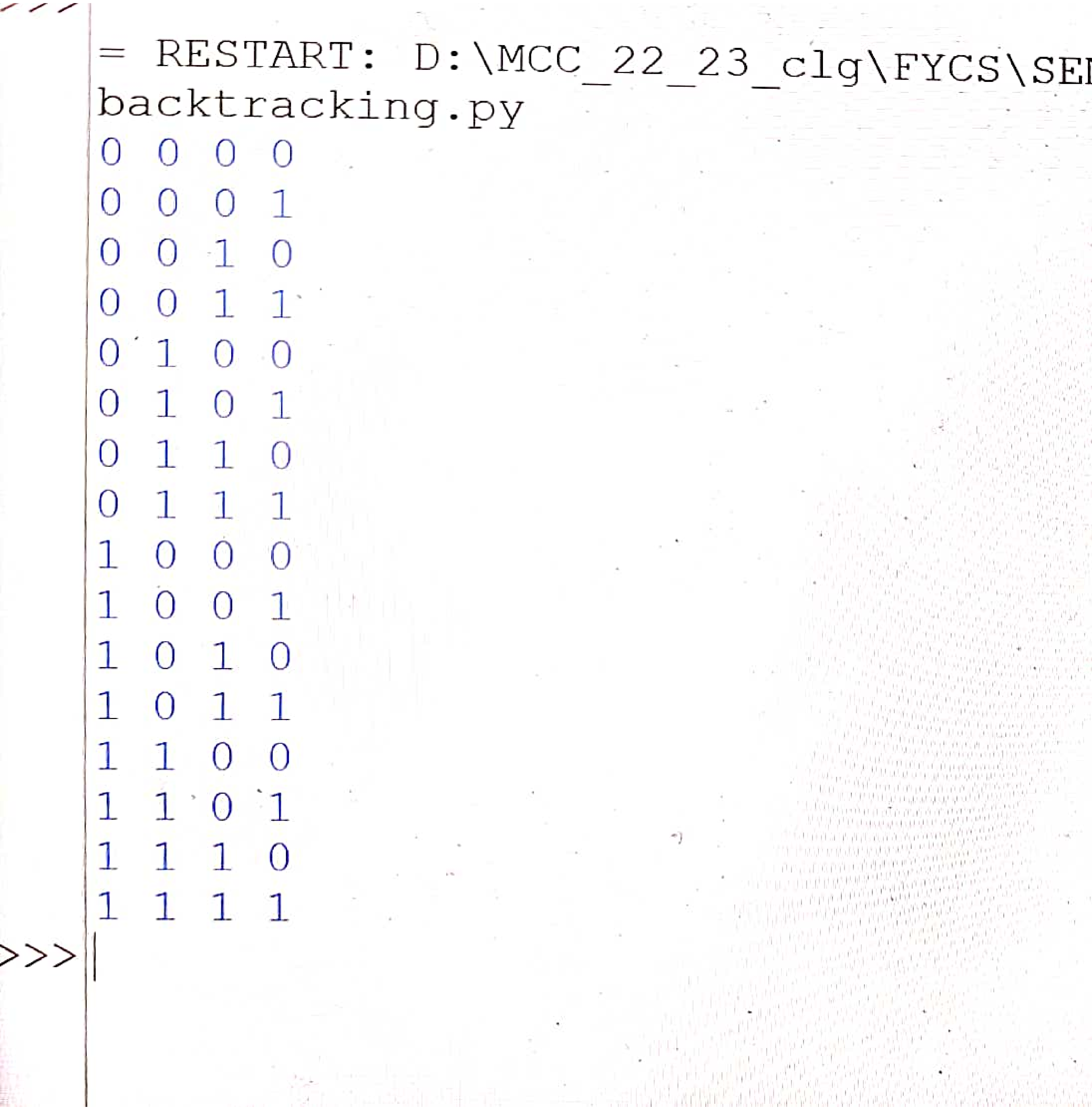
```
        j -= 1
    return True
def nQueens (board, col, n):
    if col >= n:
        return True
    for i in range (n):
        if is_safe (board, i, col, n):
            board[i][col] = 1
            if nQueens(board, col+1, n):
                return True
            board[i][col] = 0
    return False

n = int(input("enter a size of board: "))
board = [[0 for j in range (n)] for i in range (n)]
if nQueens(board, 0, n) == True:
    for i in range (n):
        for j in range (n):
            print(board[i][j], end = ' ')
        print()
else:
    print("not possible")
```



```
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 20
2
Type "help", "copyright", "credits" or "licens
>>>
= RESTART: D:\MCC_22_23_clg\FYCS\SEM 2\DAA\DAA
enter a size of board: 4
0 0 1 0
1 0 0 0
0 0 0 1
0 1 0 0
>>>
```



```python
def printTheArray(arr, n):

        for i in range(0, n):
                print(arr[i], end = " ")
        print()

def generateAllBinaryStrings(n, arr, i):
    if i == n:
            printTheArray(arr, n)
            return
        arr[i] = 0
        generateAllBinaryStrings(n, arr, i + 1)
        arr[i] = 1
        generateAllBinaryStrings(n, arr, i + 1)

if __name__ == "__main__":

        n = 4

        arr = [None] * n
        generateAllBinaryStrings(n, arr, 0)
```



```
= RESTART: D:\MCC_22_23_clg\FYCS\SE
backtracking.py
0 0 0 0
0 0 0 1
0 0 1 0
0 0 1 1
0 1 0 0
0 1 0 1
0 1 1 0
0 1 1 1
1 0 0 0
1 0 0 1
1 0 1 0
1 0 1 1
1 1 0 0
1 1 0 1
1 1 1 0
1 1 1 1
>>> 
```