ASSIGNMENT -3

QUE-11

import math

from functools import reduce

def gcd(a, b):

return math.gcd(a, b)

def lcm(a, b):

return a \* b // gcd(a, b)

def gcd\_of\_list(numbers):

return reduce(gcd, numbers)

def lcm\_of\_list(numbers):

return reduce(lcm, numbers)

# Example usage:

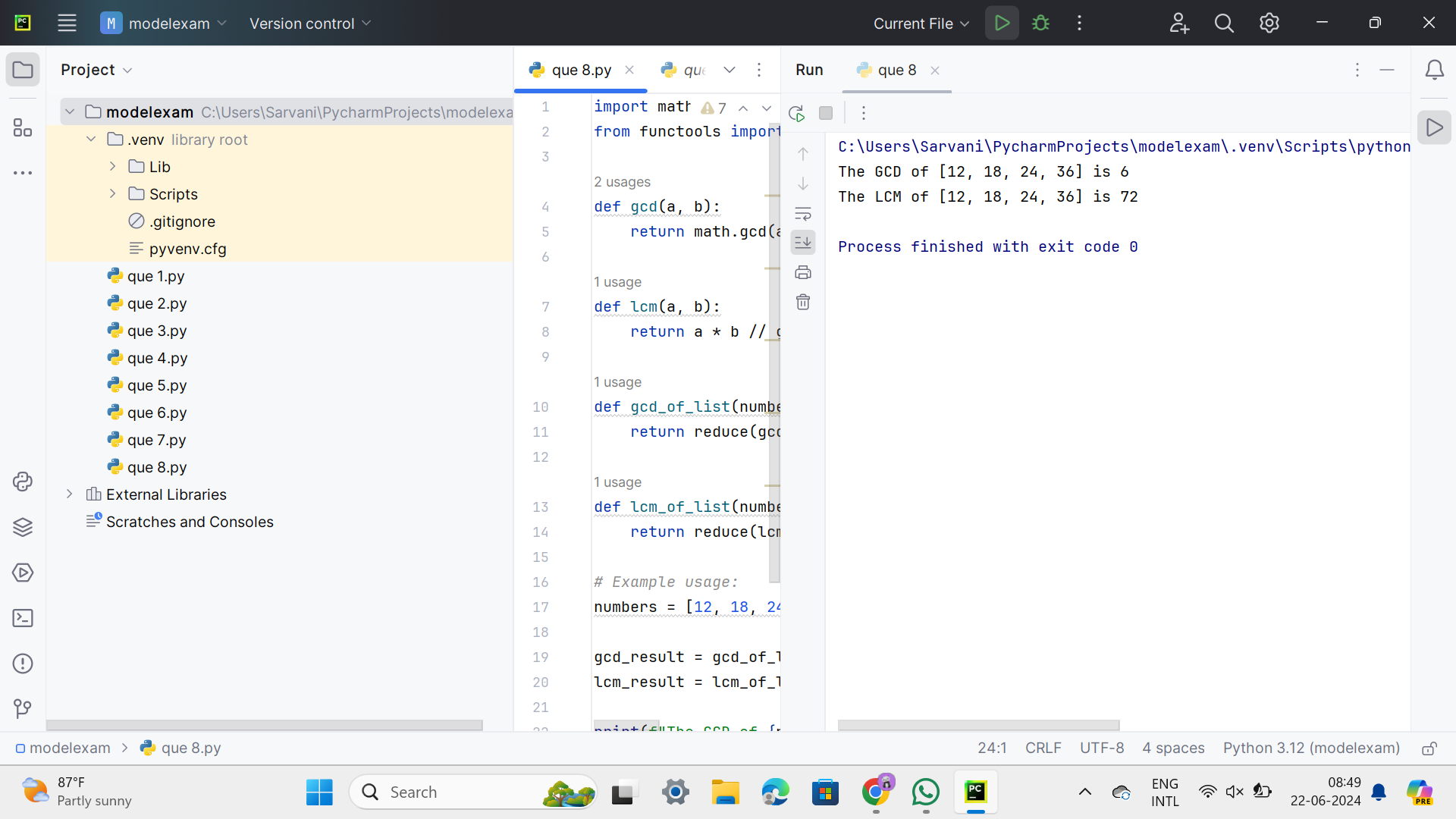
numbers = [12, 18, 24, 36]

gcd\_result = gcd\_of\_list(numbers)

lcm\_result = lcm\_of\_list(numbers)

print(f"The GCD of {numbers} is {gcd\_result}")

print(f"The LCM of {numbers} is {lcm\_result}")



QUE-12

def decimal\_to\_binary(n):

return bin(n).replace("0b", "")

def decimal\_to\_octal(n):

return oct(n).replace("0o", "")

# Example usage:

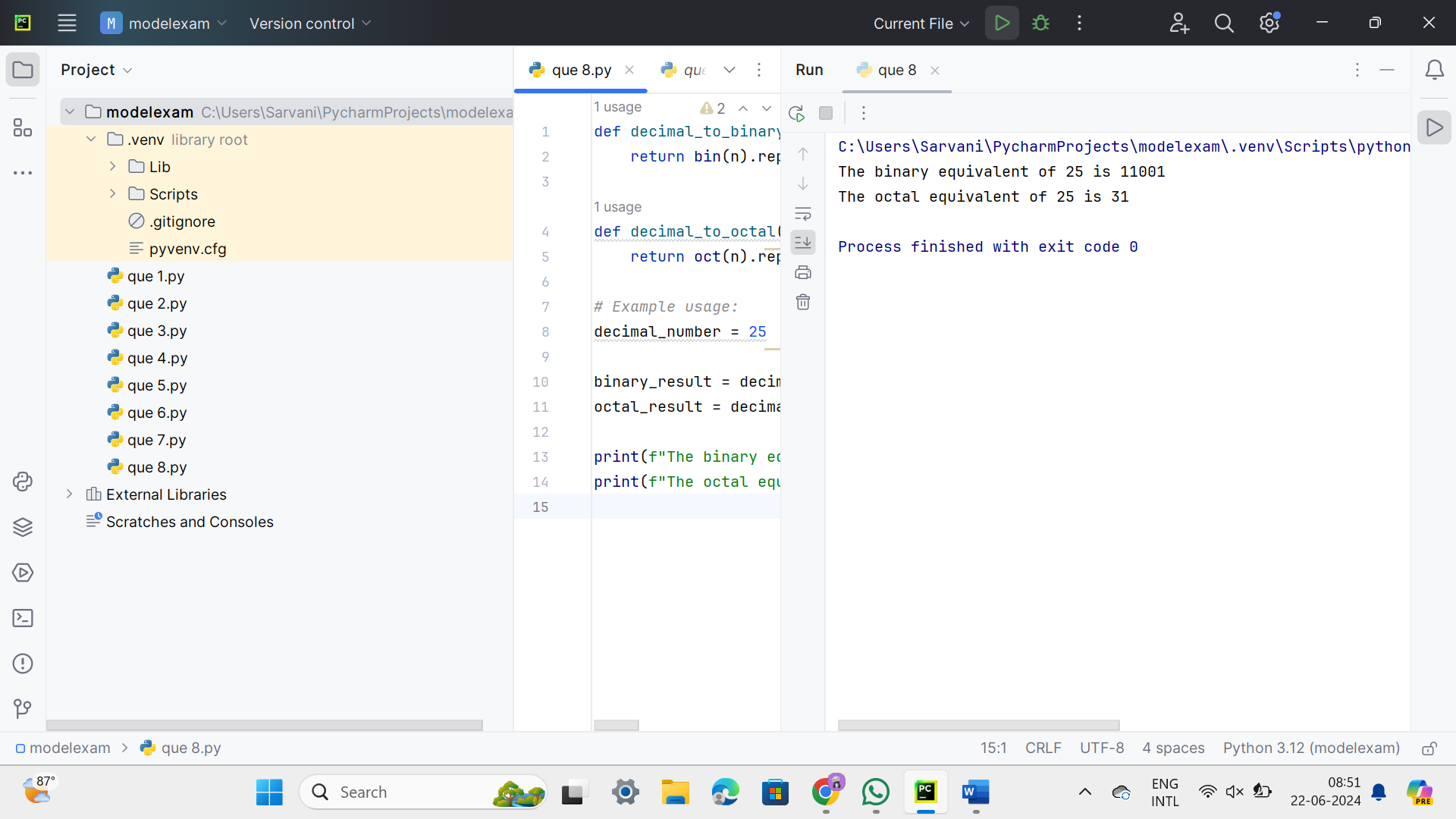
decimal\_number = 25

binary\_result = decimal\_to\_binary(decimal\_number)

octal\_result = decimal\_to\_octal(decimal\_number)

print(f"The binary equivalent of {decimal\_number} is {binary\_result}")

print(f"The octal equivalent of {decimal\_number} is {octal\_result}")



QUE-13

def print\_pattern(n):

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

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

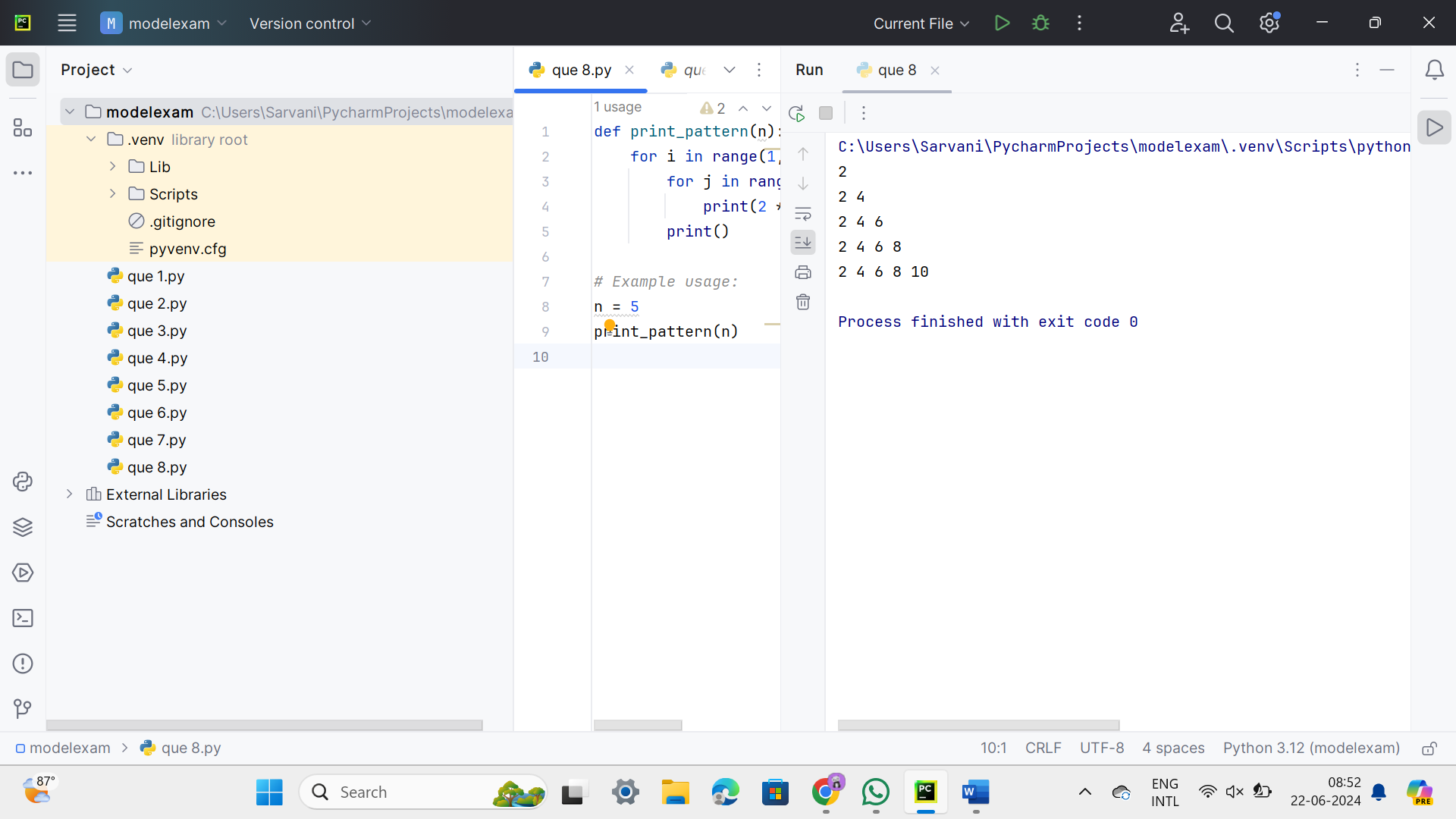
print(2 \* j, end=" ")

print()

# Example usage:

n = 5

print\_pattern(n)



QUE-14

**def remove\_duplicates(arr):**

**if not arr:**

**return []**

**# Initialize variables**

**result = [arr[0]]**

**# Traverse the sorted array and remove duplicates**

**for i in range(1, len(arr)):**

**if arr[i] != arr[i - 1]:**

**result.append(arr[i])**

**return result**

**# Sample Input**

**array = [15, 14, 25, 14, 32, 14, 31]**

**# Sorting the array**

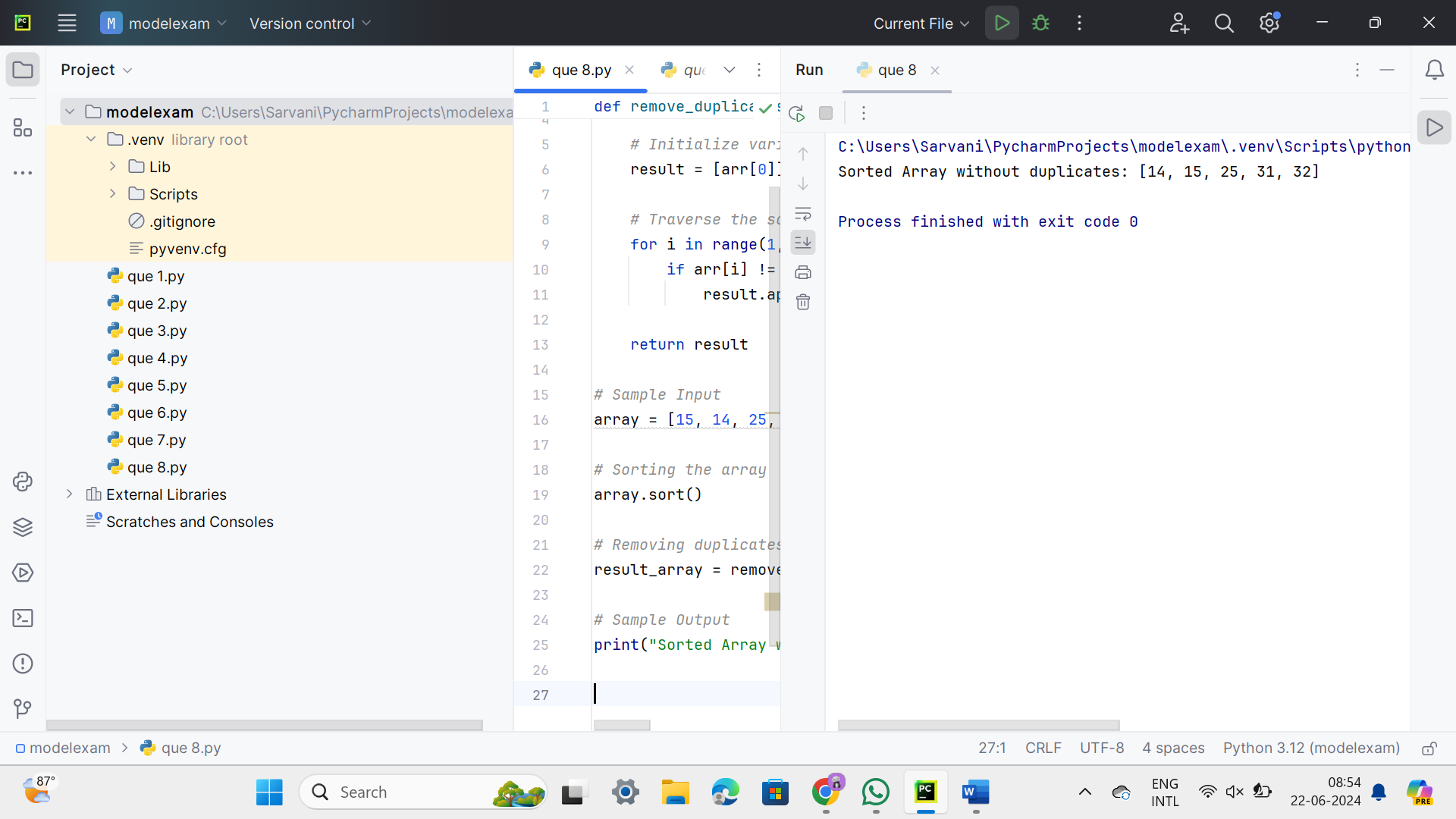
**array.sort()**

**# Removing duplicates**

**result\_array = remove\_duplicates(array)**

**# Sample Output**

**print("Sorted Array without duplicates:", result\_array)**



QUE-15

import string

def count\_special\_characters(statement):

special\_characters = set(string.punctuation)

count = 0

for char in statement:

if char in special\_characters:

count += 1

return count

# Sample Input

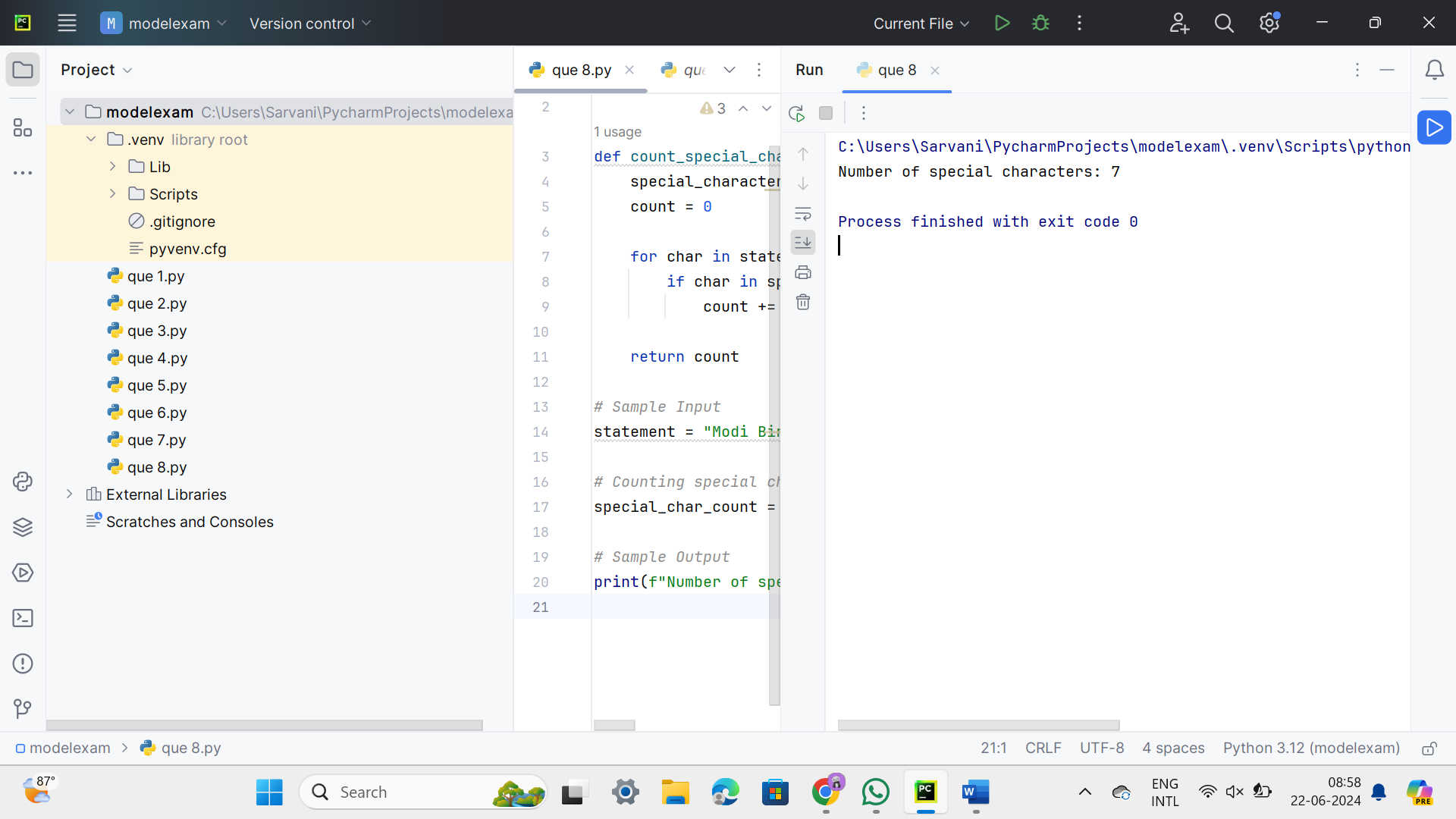
statement = "Modi Birthday @ September 17, #&$% is the wishes code for him."

# Counting special characters

special\_char\_count = count\_special\_characters(statement)

# Sample Output

print(f"Number of special characters: {special\_char\_count}")



QUE-16

def is\_prime(n):

if n <= 1:

return False

if n <= 3:

return True

if n % 2 == 0 or n % 3 == 0:

return False

i = 5

while i \* i <= n:

if n % i == 0 or n % (i + 2) == 0:

return False

i += 6

return True

def identify\_number(n):

if n <= 1:

return "Neither prime nor composite"

elif is\_prime(n):

return "Prime number"

else:

return "Composite number"

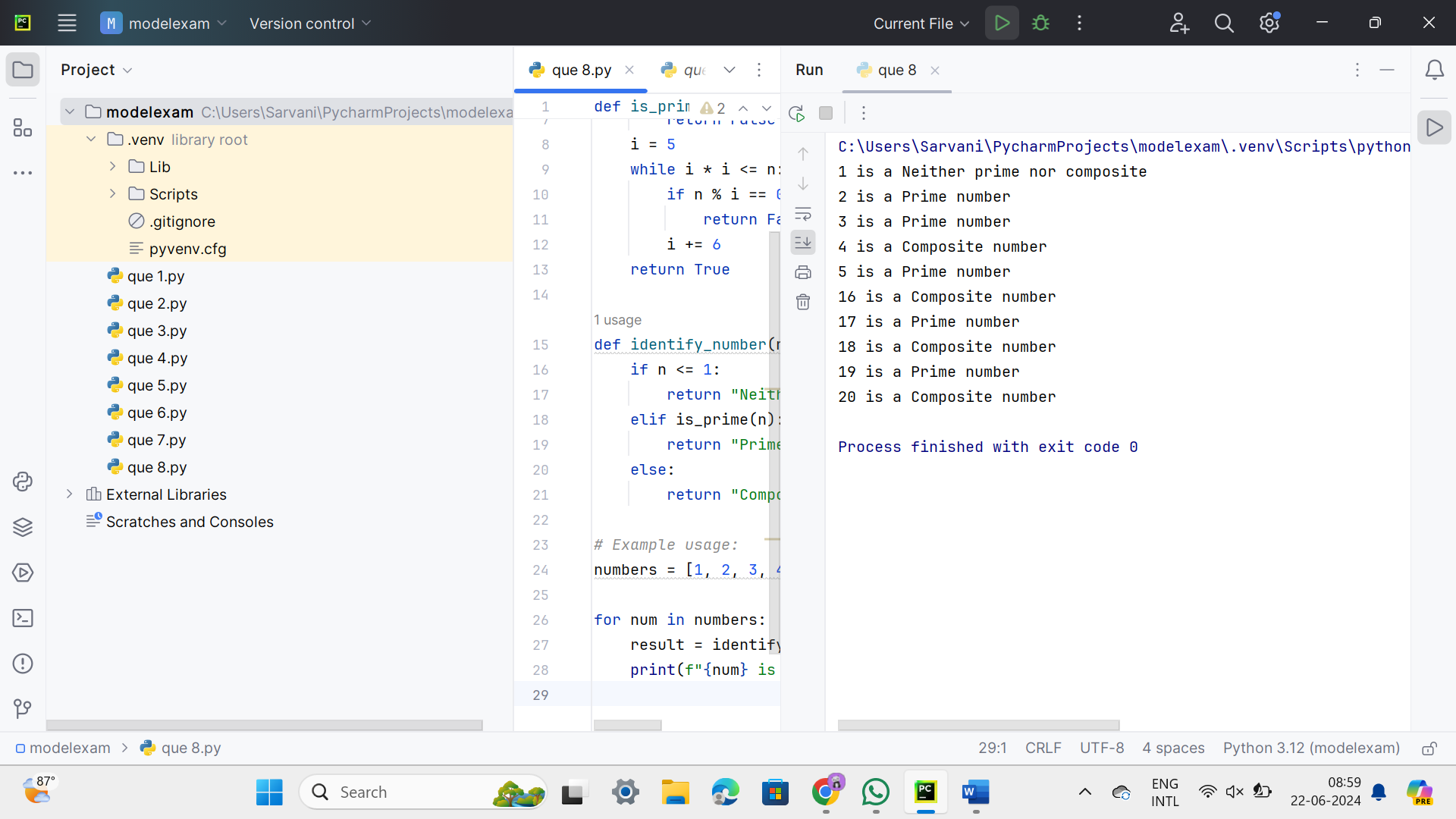
# Example usage:

numbers = [1, 2, 3, 4, 5, 16, 17, 18, 19, 20]

for num in numbers:

result = identify\_number(num)

print(f"{num} is a {result}")



QUE-17

def generate\_sequence(M, N, K):

if K == 0:

return "Step size K cannot be zero"

if (M < N and K < 0) or (M > N and K > 0):

return "Step size K is not valid for the given range"

sequence = []

if K > 0:

current = M

while current <= N:

sequence.append(current)

current += K

else: # K < 0

current = M

while current >= N:

sequence.append(current)

current += K

return sequence

# Example usage:

test\_cases = [

(50, 100, 7),

(15, 5, 2),

(25, 50, 4),

(15, 100, -2),

(0, 0, 2),

(200, 200, 50)

]

for i, (M, N, K) in enumerate(test\_cases):

result = generate\_sequence(M, N, K)

print(f"Test case {i + 1}: M = {M}, N = {N}, K = {K}")

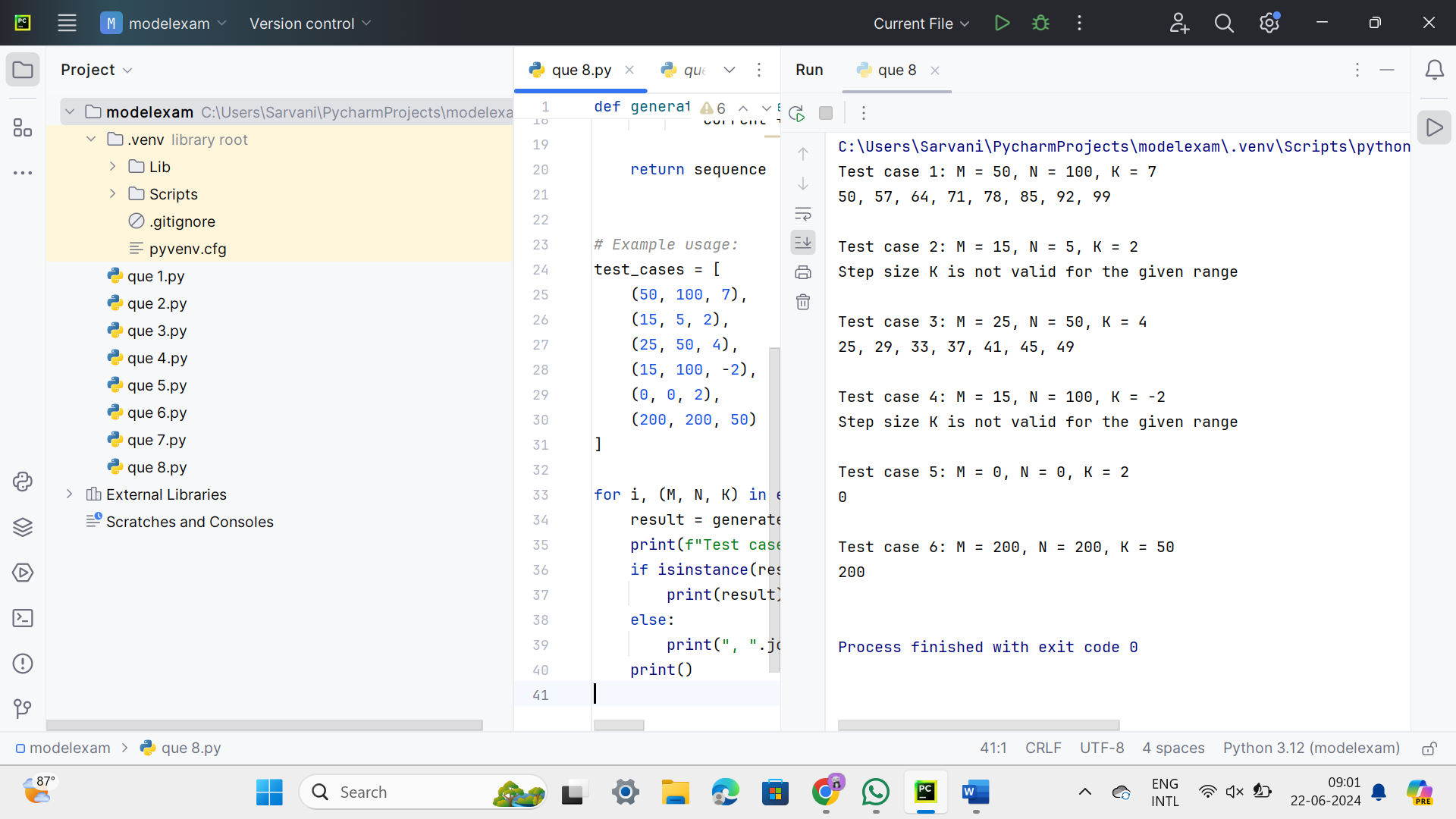
if isinstance(result, str):

print(result)

else:

print(", ".join(map(str, result)))

print()



QUE-18

def matrix\_addition(mat1, mat2):

if len(mat1) != len(mat2) or len(mat1[0]) != len(mat2[0]):

raise ValueError("Matrices must have the same dimensions for addition.")

rows = len(mat1)

cols = len(mat1[0])

# Initialize the result matrix with zeros

result = [[0 for \_ in range(cols)] for \_ in range(rows)]

# Perform element-wise addition

for i in range(rows):

for j in range(cols):

result[i][j] = mat1[i][j] + mat2[i][j]

return result

def print\_matrix(matrix):

for row in matrix:

print(" ".join(map(str, row)))

# Sample Input

mat1 = [

[1, 2],

[5, 3]

]

mat2 = [

[2, 3],

[4, 1]

]

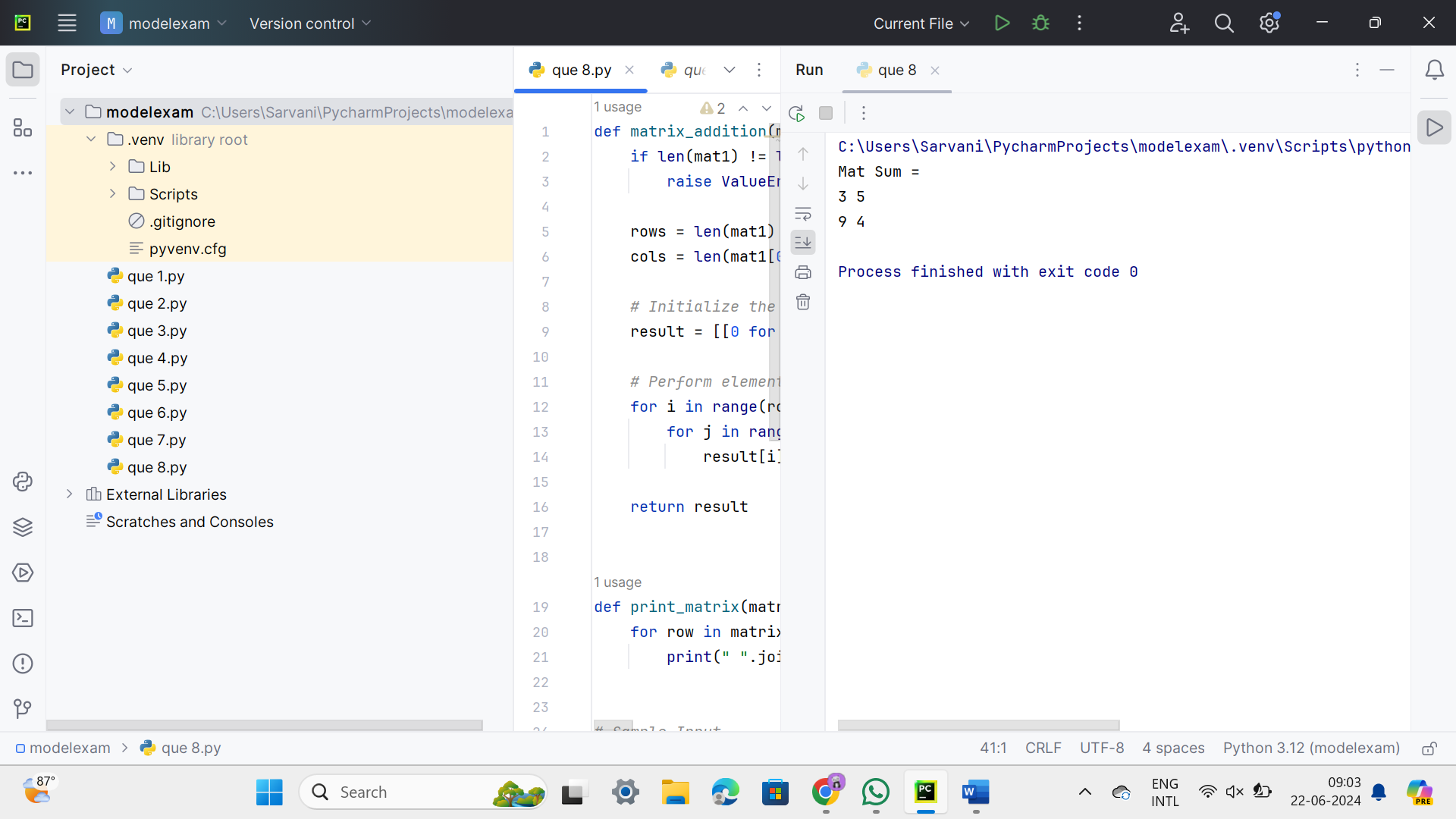
# Performing matrix addition

result\_matrix = matrix\_addition(mat1, mat2)

# Sample Output

print("Mat Sum =")

print\_matrix(result\_matrix)



QUE-19

def sort\_names(names, order):

if order.lower() == 'ascending':

sorted\_names = sorted(names)

elif order.lower() == 'descending':

sorted\_names = sorted(names, reverse=True)

else:

raise ValueError("Order must be 'ascending' or 'descending'")

return sorted\_names

def get\_user\_choice():

choice = input("Enter the sorting order (ascending/descending): ")

return choice

def main():

# Sample list of names

names = [

"Alice", "Bob", "Charlie", "Diana", "Edward", "Fiona"

]

# Get user's choice for sorting order

order = get\_user\_choice()

try:

# Sort names based on user's choice

sorted\_names = sort\_names(names, order)

# Print sorted names

print("Sorted names:")

for name in sorted\_names:

print(name)

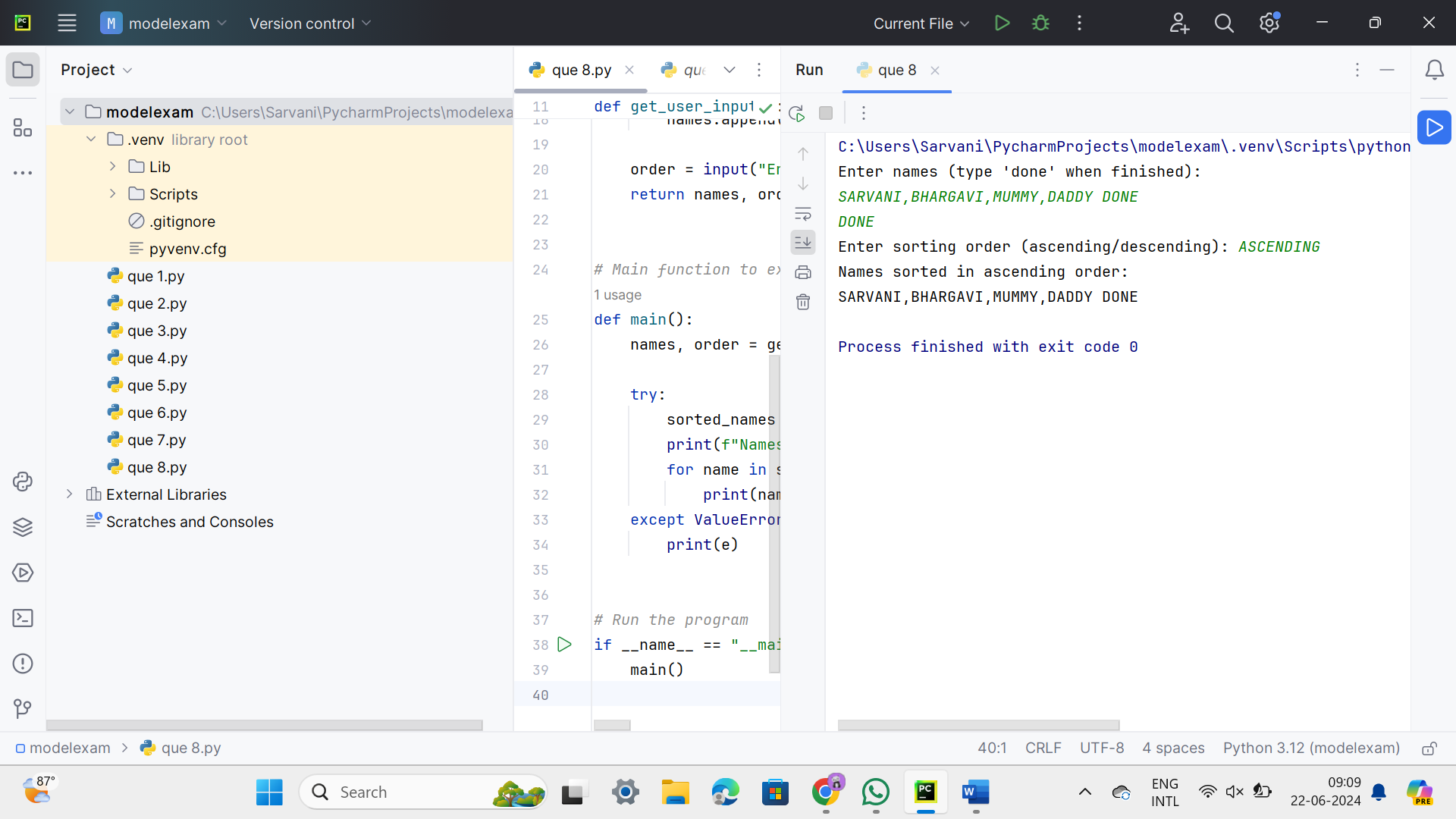
except ValueError as e:

print(e)

# Run the main function

if \_\_name\_\_ == "\_\_main\_\_":

main()



QUE-20

def multiplication\_table(m, n):

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

print(f"{m} × {i} = {m \* i}")

# Example usage:

m = 5

n = 10

print(f"Multiplication table of {m} up to {n}:")

multiplication\_table(m, n)

