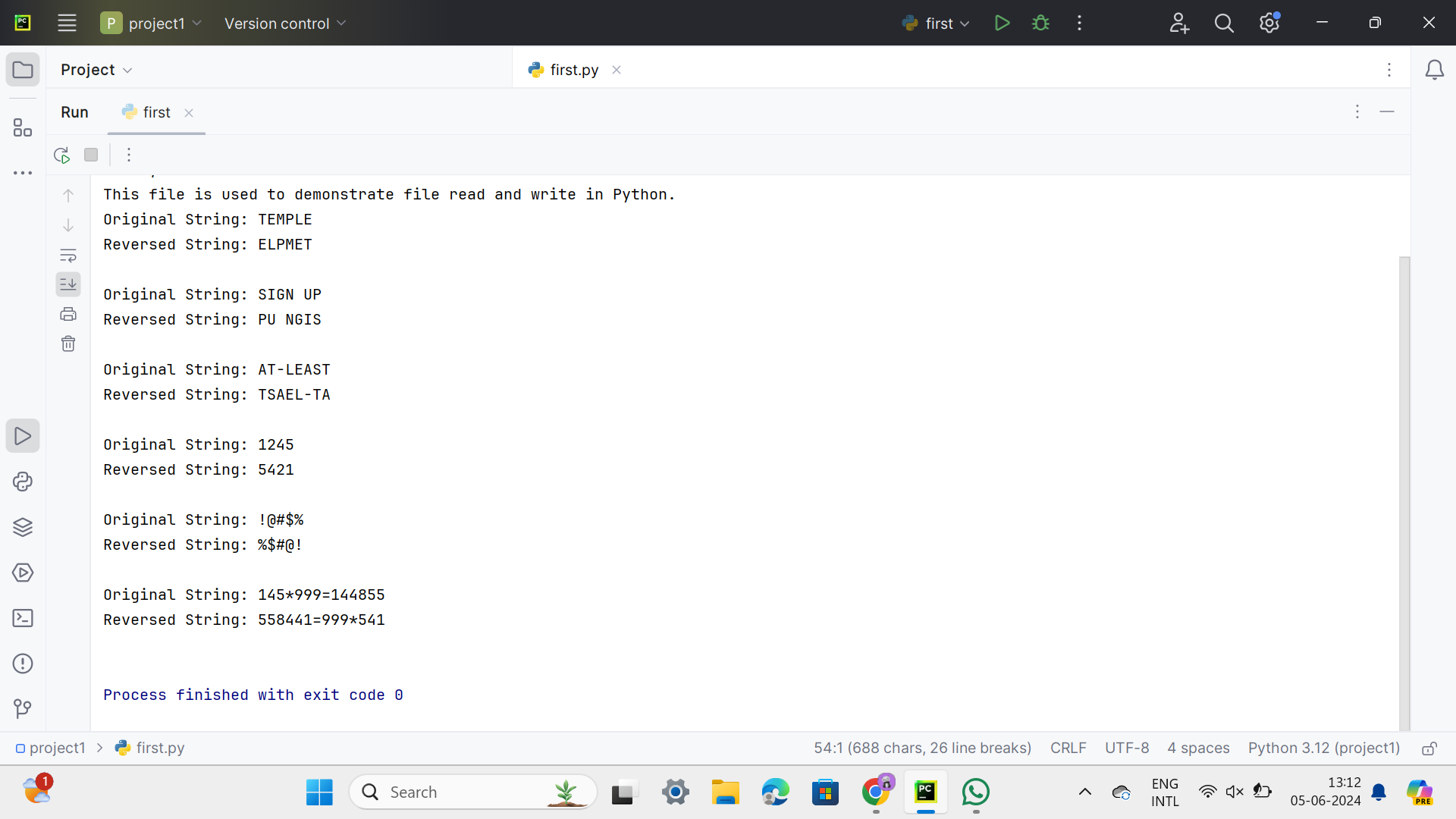
ASSIGNMENT-1

QUE-1 CODE

def reverse\_string(input\_string):  
 *# Initialize an empty string for the reversed string* reversed\_string = ''  
  
 *# Loop through the original string in reverse order* for i in range(len(input\_string) - 1, -1, -1):  
 reversed\_string += input\_string[i]  
  
 return reversed\_string  
  
  
*# Sample Input and Test Cases*test\_cases = [  
 "TEMPLE",  
 "SIGN UP",  
 "AT-LEAST",  
 "1245",  
 "!@#$%",  
 "145\*999=144855"  
]  
  
*# Applying the reverse\_string function to each test case and printing the results*for case in test\_cases:  
 print(f"Original String: {case}")  
 print(f"Reversed String: {reverse\_string(case)}")  
 print() *# Print a newline for better readability*



QUE-2

CODE

def calculate\_pow(x, n):

result = 1

for \_ in range(n):

result \*= x

return result

def calculate\_add(x, n):

return x + n

def calculate\_sub(x, n):

return x - n

def calculate\_mul(x, n):

return x \* n

def calculate\_div(x, n):

if n == 0:

return "Division by zero is undefined"

return x / n

def main():

x = int(input("Enter the value for X: "))

n = int(input("Enter the value for N: "))

print("Choose the operation:")

print("1. Power (Pow(x, n))")

print("2. Addition (Add(x, n))")

print("3. Subtraction (Sub(x, n))")

print("4. Multiplication (Mul(x, n))")

print("5. Division (Div(x, n))")

choice = int(input("Enter your choice (1-5): "))

if choice == 1:

print(f"Pow({x}, {n}) = {calculate\_pow(x, n)}")

elif choice == 2:

print(f"Add({x}, {n}) = {calculate\_add(x, n)}")

elif choice == 3:

print(f"Sub({x}, {n}) = {calculate\_sub(x, n)}")

elif choice == 4:

print(f"Mul({x}, {n}) = {calculate\_mul(x, n)}")

elif choice == 5:

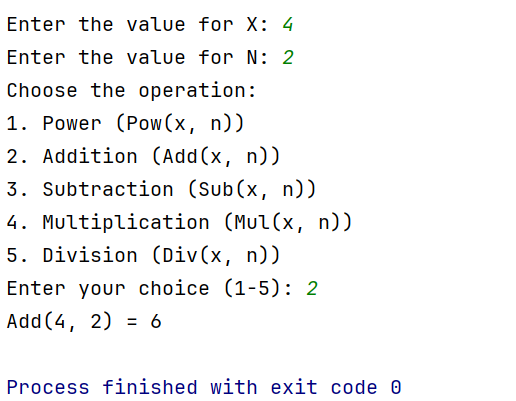
print(f"Div({x}, {n}) = {calculate\_div(x, n)}")

else:

print("Invalid choice!")

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

main()



QUE-3

CODE-

def is\_prime(num):

if num <= 1:

return False

if num == 2:

return True

if num % 2 == 0:

return False

for i in range(3, int(num\*\*0.5) + 1, 2):

if num % i == 0:

return False

return True

def count\_prime\_composite(numbers):

prime\_count = 0

composite\_count = 0

for num in numbers:

try:

num = int(num)

if num > 1:

if is\_prime(num):

prime\_count += 1

else:

composite\_count += 1

except ValueError:

# Skip non-integer inputs

continue

return prime\_count, composite\_count

def main():

# Input from user

user\_input = input("Enter the numbers (separated by spaces or commas): ")

# Splitting the input into a list of strings

numbers = user\_input.replace(',', ' ').split()

# Counting prime and composite numbers

prime\_count, composite\_count = count\_prime\_composite(numbers)

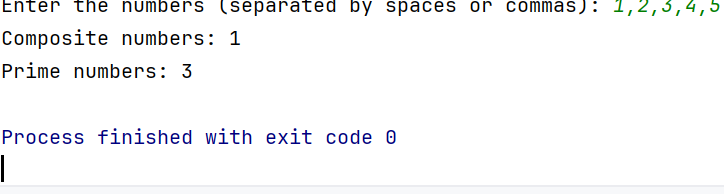
# Output the result

print(f"Composite numbers: {composite\_count}")

print(f"Prime numbers: {prime\_count}")

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

main()



QUE-4

CODE-

def is\_valid\_username(username):

# Check length of username

if len(username) < 3 or len(username) > 15:

return False

# Check if the username contains only alphanumeric characters

if not username.isalnum():

return False

# Check if the username does not start with a number

if username[0].isdigit():

return False

return True

def main():

# Get username input from the user

username = input("Enter the username: ")

# Check if the username is valid

if is\_valid\_username(username):

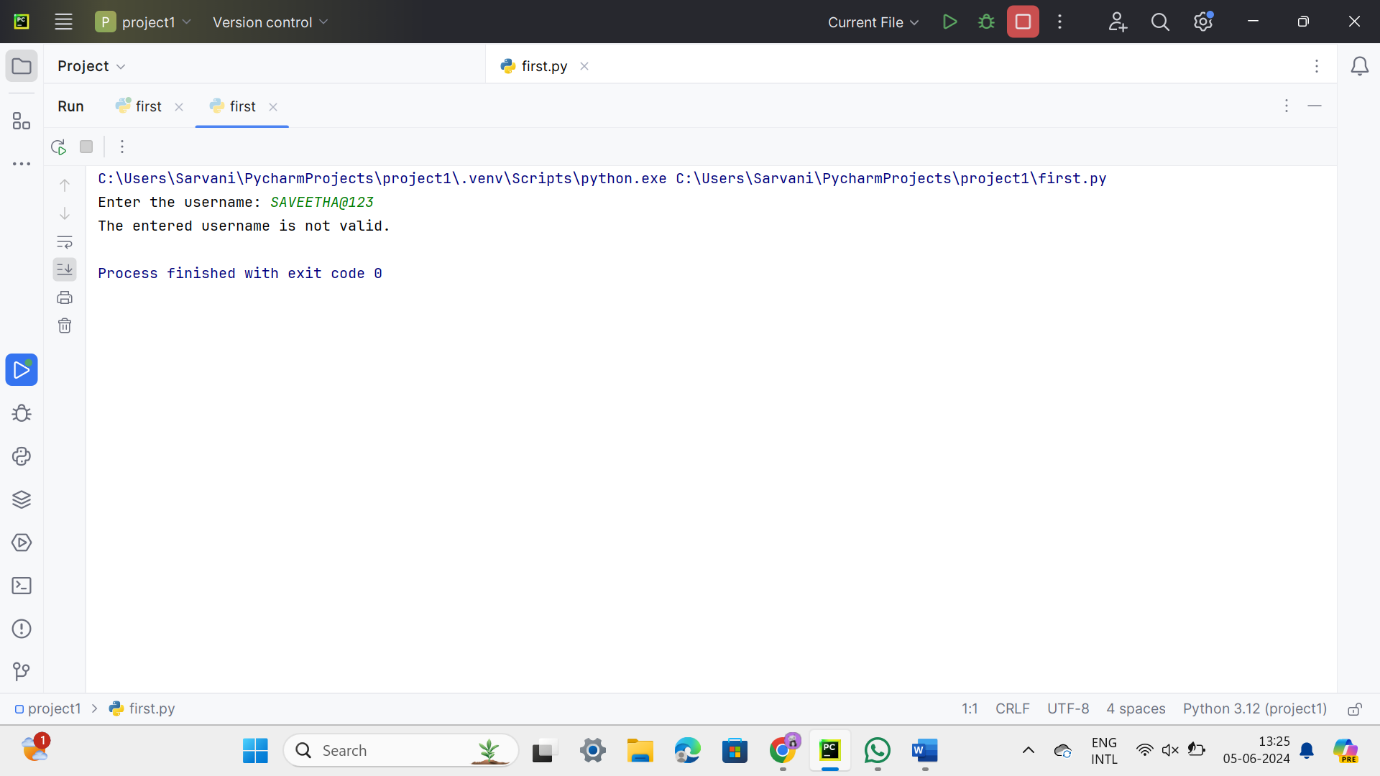
print("The entered username is valid.")

else:

print("The entered username is not valid.")

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

main()



QUE-5

CODE-

def find\_mth\_maximum(arr, M):

# Sort the array in descending order and return the Mth maximum element

sorted\_arr = sorted(arr, reverse=True)

if M <= 0 or M > len(sorted\_arr):

return None

return sorted\_arr[M-1]

def find\_nth\_minimum(arr, N):

# Sort the array in ascending order and return the Nth minimum element

sorted\_arr = sorted(arr)

if N <= 0 or N > len(sorted\_arr):

return None

return sorted\_arr[N-1]

def main():

# Sample input

arr = [14, 16, 87, 36, 25, 89, 34]

M = 1

N = 3

# Find Mth maximum and Nth minimum

mth\_max = find\_mth\_maximum(arr, M)

nth\_min = find\_nth\_minimum(arr, N)

if mth\_max is not None and nth\_min is not None:

# Calculate sum and difference

sum\_result = mth\_max + nth\_min

difference\_result = mth\_max - nth\_min

# Print the results

print(f"{M}th Maximum Number = {mth\_max}")

print(f"{N}th Minimum Number = {nth\_min}")

print(f"Sum = {sum\_result}")

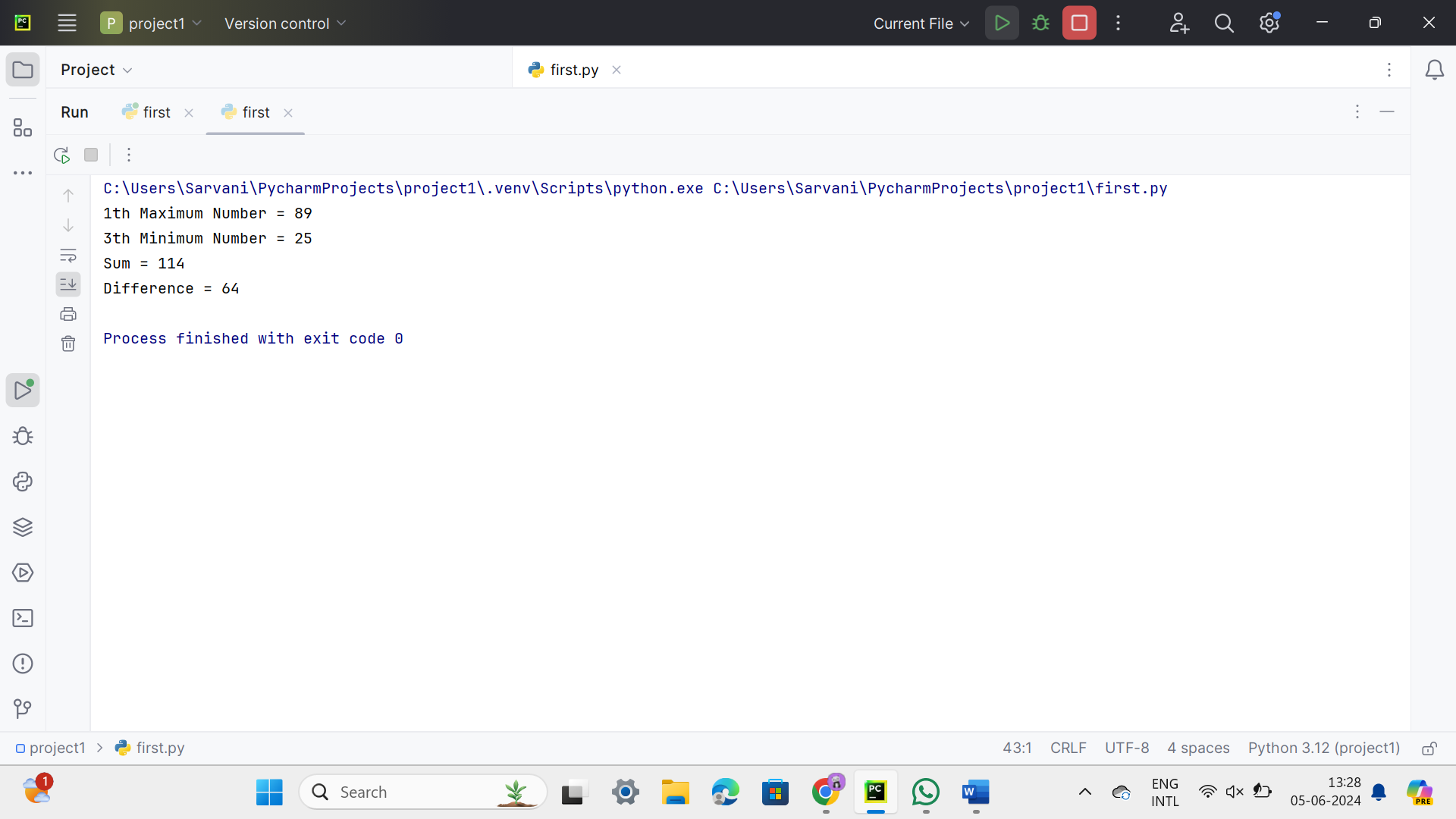
print(f"Difference = {difference\_result}")

else:

print("Invalid M or N value for the given array length.")

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

main()



QUE-6

CODE-

def reverse\_number(number):

# Check if the number is negative

is\_negative = number < 0

if is\_negative:

number = -number # Make the number positive for reversal

reversed\_number = 0

while number > 0:

digit = number % 10

reversed\_number = reversed\_number \* 10 + digit

number = number // 10

if is\_negative:

reversed\_number = -reversed\_number # Restore the negative sign

return reversed\_number

def main():

user\_input = input("Enter the number: ")

try:

number = int(user\_input)

reversed\_number = reverse\_number(number)

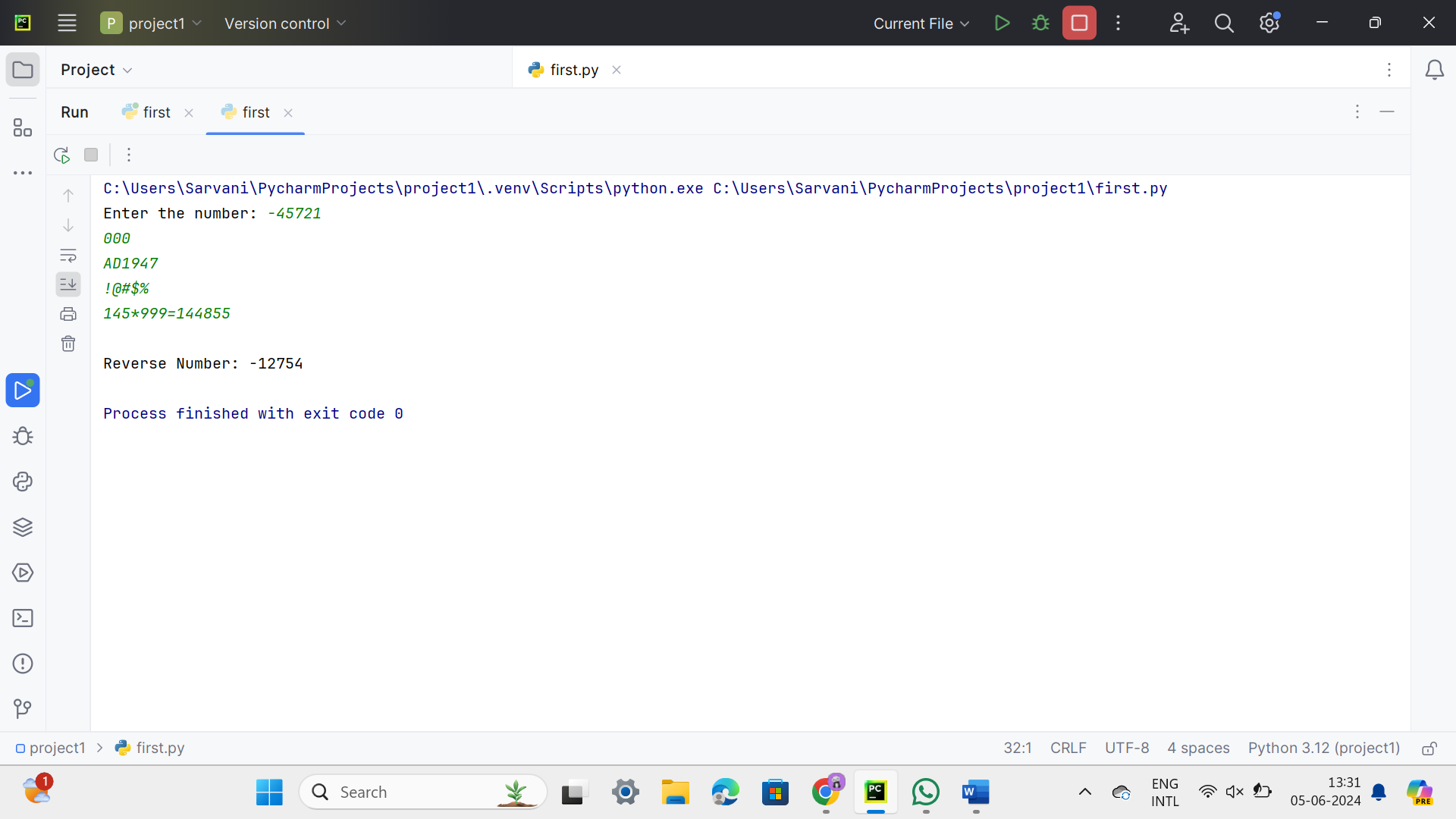
print(f"Reverse Number: {reversed\_number}")

except ValueError:

print("Invalid input. Please enter a valid integer.")

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

main()



QUE-7

CODE-

def check\_voting\_eligibility(age):

try:

age = float(age)

if age < 0:

return "Invalid input. Age cannot be negative."

if age % 1 != 0:

return "Invalid input. Age should be a whole number."

age = int(age)

if age >= 18:

return "You are eligible to vote."

else:

years\_left = 18 - age

return f"You are allowed to vote after {years\_left} years."

except ValueError:

return "Invalid input. Please enter a valid age."

def main():

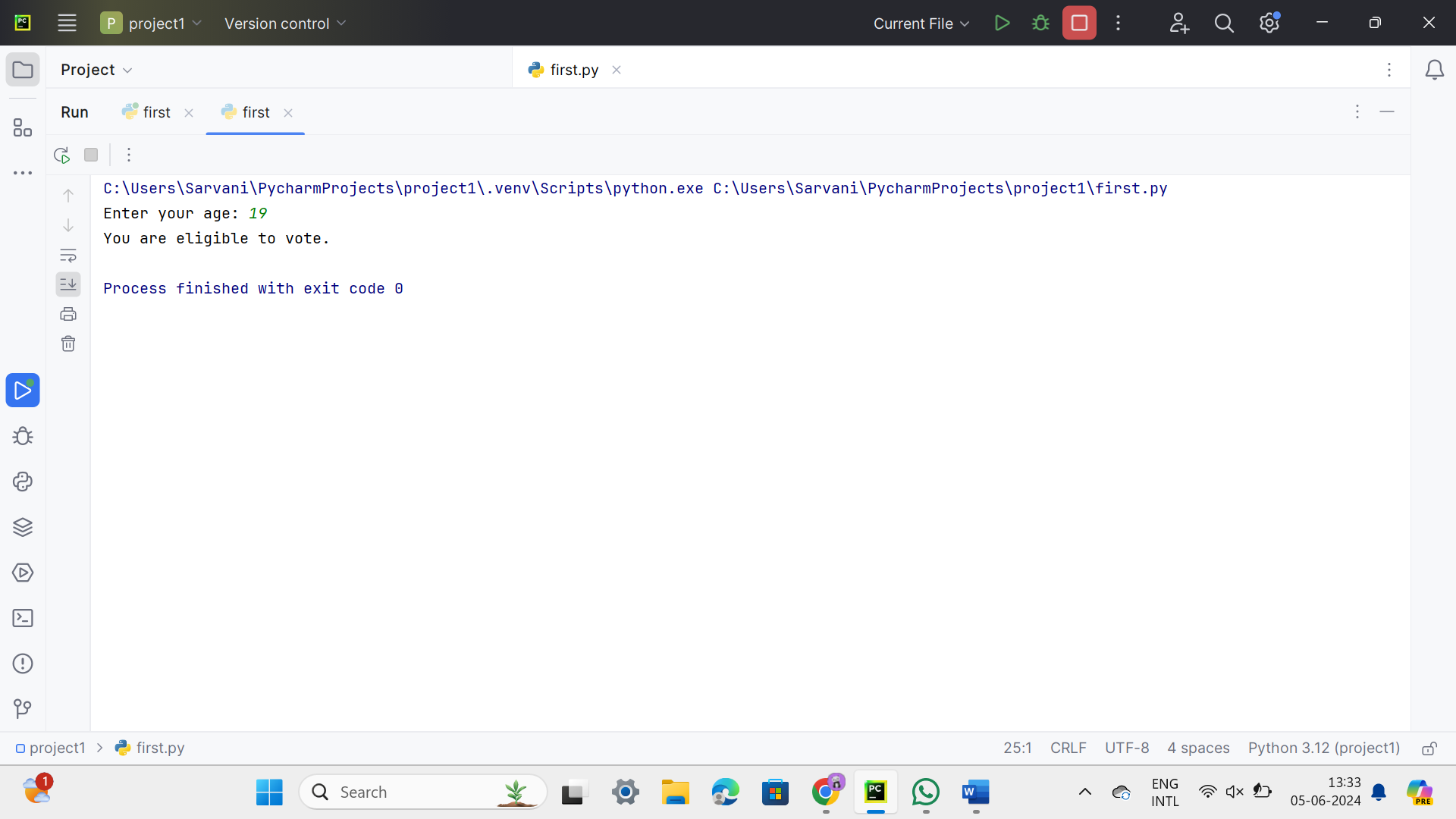
user\_input = input("Enter your age: ")

result = check\_voting\_eligibility(user\_input)

print(result)

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

main()



QUE-8

CODE-

def calculate\_atm\_balance(denominations, notes):

total\_balance = 0

for denomination, count in zip(denominations, notes):

total\_balance += denomination \* count

return total\_balance

def main():

denominations = []

notes = []

for i in range(1, 5):

try:

denomination = int(input(f"Enter the {i}st Denomination: "))

if denomination not in [2000, 500, 200, 100]:

print("Invalid denomination entered. Please enter one of the following: 2000, 500, 200, 100.")

return

num\_notes = int(input(f"Enter the {i}st Denomination number of notes: "))

if num\_notes < 0:

print("The number of notes cannot be negative.")

return

denominations.append(denomination)

notes.append(num\_notes)

except ValueError:

print("Invalid input. Please enter valid numbers.")

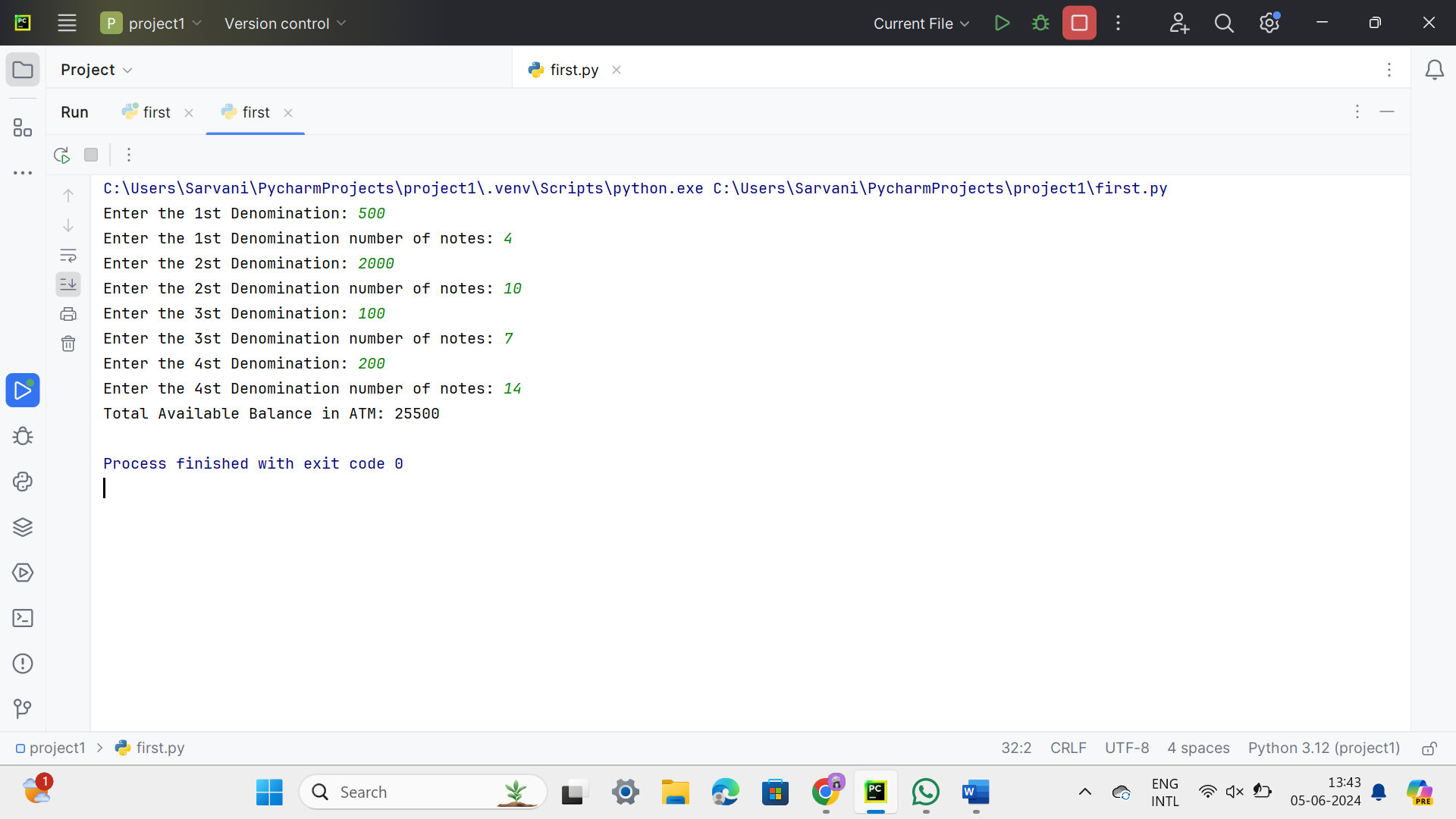
return

total\_balance = calculate\_atm\_balance(denominations, notes)

print(f"Total Available Balance in ATM: {total\_balance}")

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

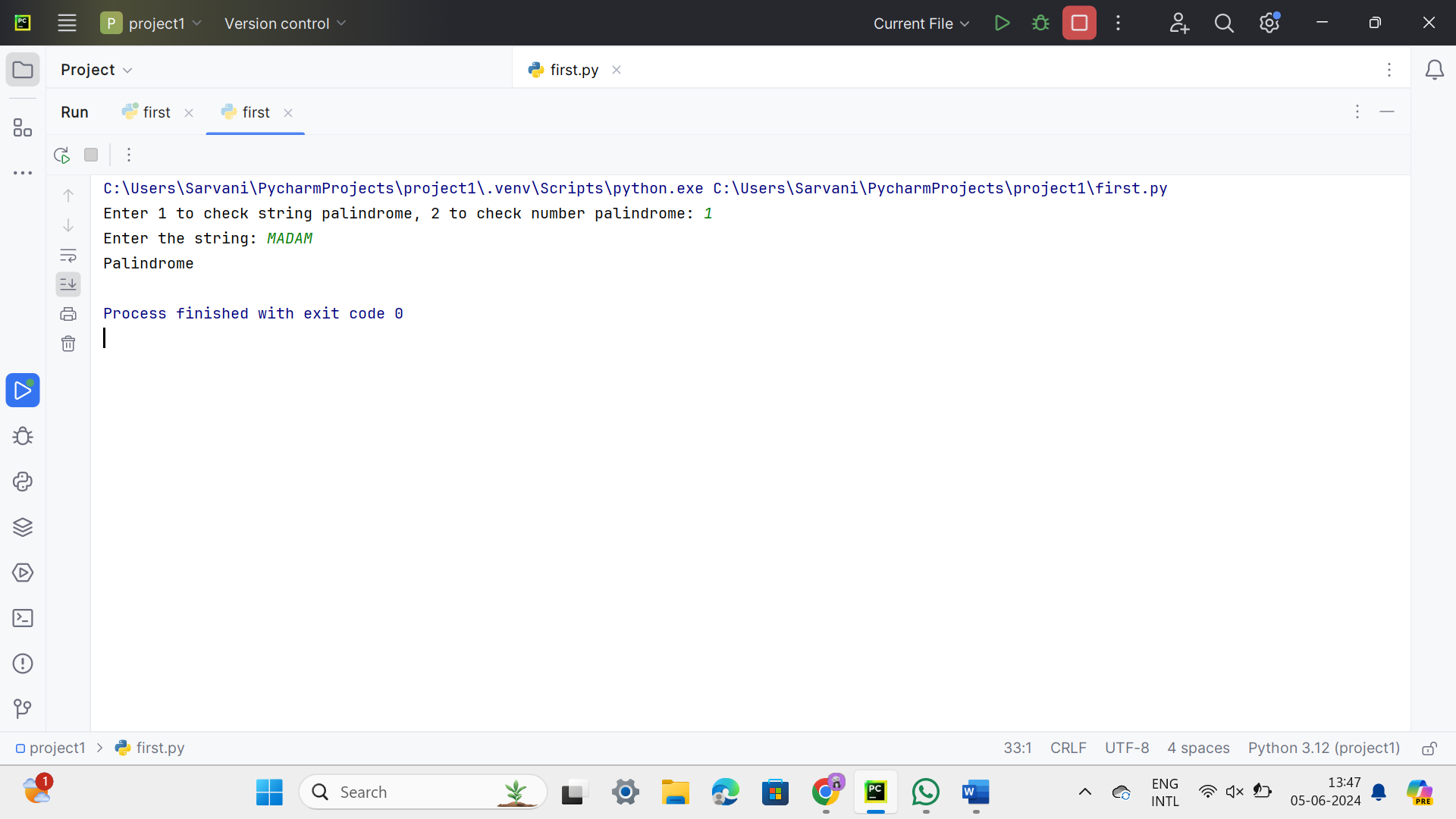
main()



QUE-9

CODE-

def is\_palindrome\_string(s):  
 s = s.lower().replace(" ", "")  
 return s == s[::-1]  
  
def is\_palindrome\_number(n):  
 n = str(n)  
 return n == n[::-1]  
  
def main():  
 choice = input("Enter 1 to check string palindrome, 2 to check number palindrome: ")  
  
 if choice == '1':  
 user\_input = input("Enter the string: ")  
 if is\_palindrome\_string(user\_input):  
 print("Palindrome")  
 else:  
 print("Not a palindrome")  
 elif choice == '2':  
 user\_input = input("Enter the number: ")  
 try:  
 float(user\_input) *# Validate the input is a number* if is\_palindrome\_number(user\_input):  
 print("Palindrome")  
 else:  
 print("Not a palindrome")  
 except ValueError:  
 print("Invalid input. Please enter a valid number.")  
 else:  
 print("Invalid choice. Please enter 1 or 2.")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()



QUE-10

CODE-

import math

from functools import reduce

def gcd\_multiple(numbers):

return reduce(math.gcd, numbers)

def lcm(a, b):

return abs(a \* b) // math.gcd(a, b)

def lcm\_multiple(numbers):

return reduce(lcm, numbers)

def main():

try:

n = int(input("Enter the N value: "))

if n <= 0:

print("N should be a positive integer.")

return

numbers = []

for i in range(n):

num = int(input(f"Enter number {i + 1}: "))

numbers.append(num)

gcd\_result = gcd\_multiple(numbers)

lcm\_result = lcm\_multiple(numbers)

print(f"LCM = {lcm\_result}")

print(f"GCD = {gcd\_result}")

except ValueError:

print("Invalid input. Please enter valid integers.")

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

main()

