16.Write a program to print the first n perfect numbers. (Hint Perfect number means **a** positive integer that is equal to the sum of its proper divisors)

Sample Input:

N = 3

Sample Output:

First 3 perfect numbers are: 6 , 28 , 496

**Test Cases:**

1. N = 0
2. N = 5
3. N = -2
4. N = -5
5. N = 0.2

**PROGRAM:**

def sum\_of\_divisors(num):

divisors\_sum = 1

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

if num % i == 0:

divisors\_sum += i

if i != num // i:

divisors\_sum += num // i

return divisors\_sum

def perfect\_numbers(n):

perfect\_nums = []

num = 2

while len(perfect\_nums) < n:

if sum\_of\_divisors(num) == num:

perfect\_nums.append(num)

num += 1

return perfect\_nums

def main():

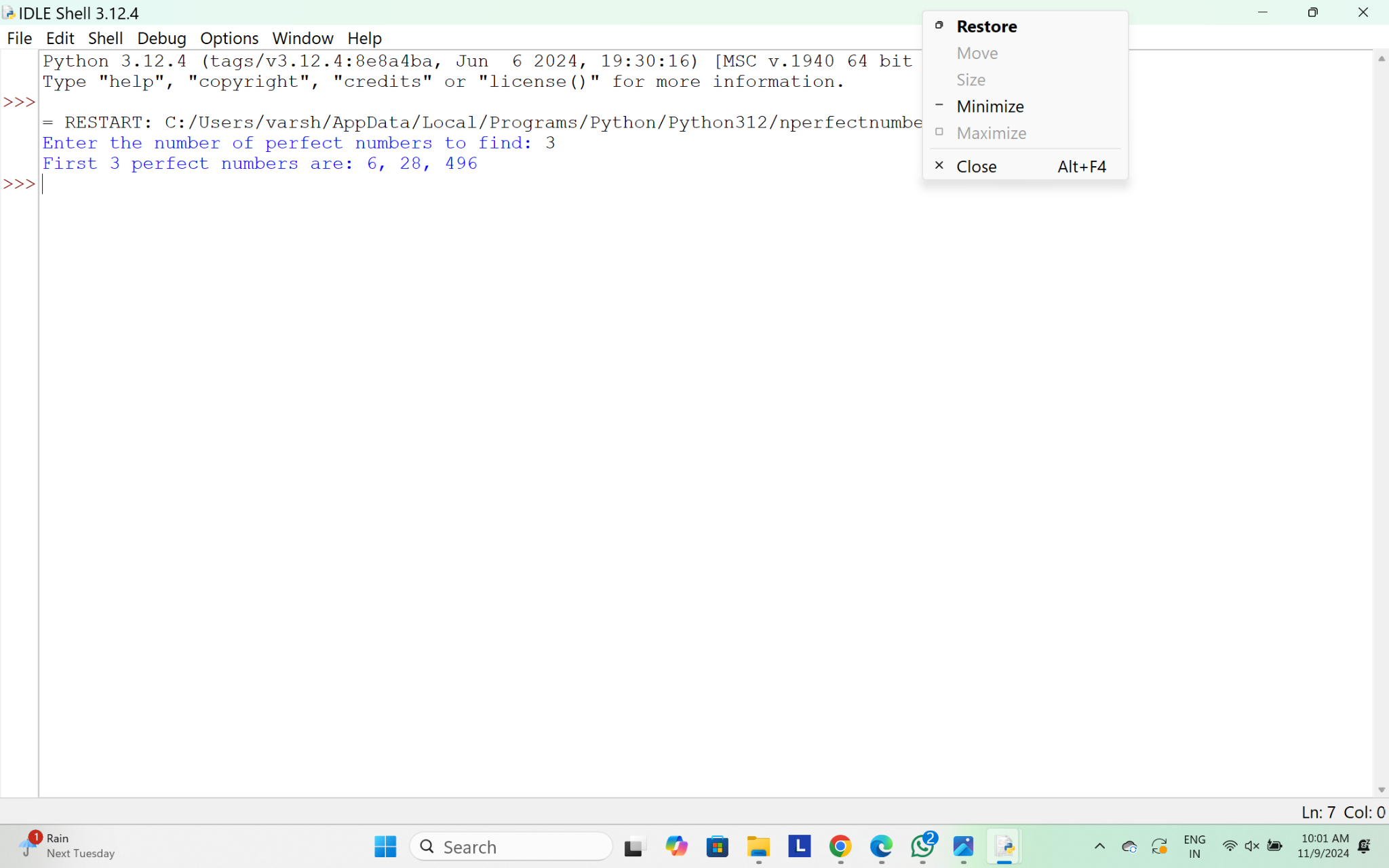
n = int(input("Enter the number of perfect numbers to find: ")

result = perfect\_numbers(n)

print(f"First {n} perfect numbers are:", ", ".join(map(str, result)))

main()

**OUTPUT:**

****

17.A Pythagorean triplet is a set of three integers a, b and c such that a2+ b2= c2. Given a limit, generate all Pythagorean Triples with values smaller than given limit?

**PROGRAM:**

import math

def generate\_pythagorean\_triplets(limit):

triplets = []

for a in range(1, limit):

for b in range(a + 1, limit):

c = math.sqrt(a \* a + b \* b)

if c.is\_integer() and c < limit:

triplets.append((a, b, int(c)))

return triplets

def main():

limit = int(input("Enter the limit: "))

triplets = generate\_pythagorean\_triplets(limit)

if triplets:

print(f"Pythagorean Triplets with values smaller than {limit}:")

for triplet in triplets:

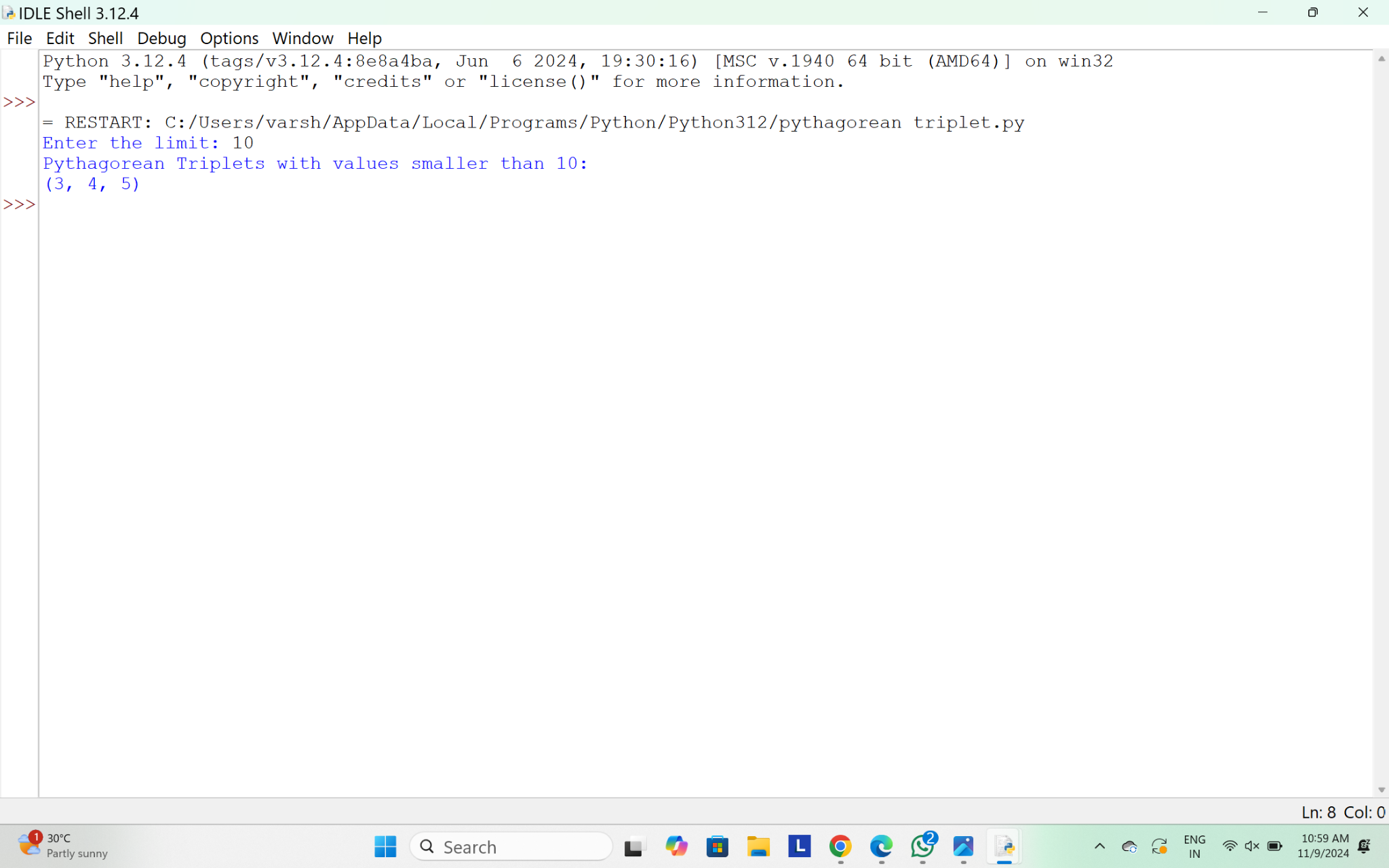
print(triplet)

else:

print(f"No Pythagorean Triplets found for values smaller than {limit}.")

main()

**OUTPUT:**

****

18.Write a program using function to calculate the simple interest. Suppose the customer is a senior citizen. He is being offered 12 percent rate of interest; for all other customers, the ROI is 10 percent.

Sample Input:

Enter the principal amount: 200000

Enter the no of years: 3

Is customer senior citizen (y/n): n

Sample Output:

Interest: 60000

**Test Cases:**

A.Principal: 2000 , Years: 0

B.Principal: 20000 , Years: -

C.Principal: -2000 , Years: 2

D.Principal:2,Years:2000

E.Principal:0,Years:5

**PROGRAM:**

def calculate\_simple\_interest(principal, years, is\_senior\_citizen):

if is\_senior\_citizen == 'y':

rate\_of\_interest = 12

else:

rate\_of\_interest = 10

interest = (principal \* rate\_of\_interest \* years) / 100

return interest

def main():

principal = float(input("Enter the principal amount: "))

years = int(input("Enter the number of years: "))

is\_senior\_citizen = input("Is customer senior citizen (y/n): ").lower()

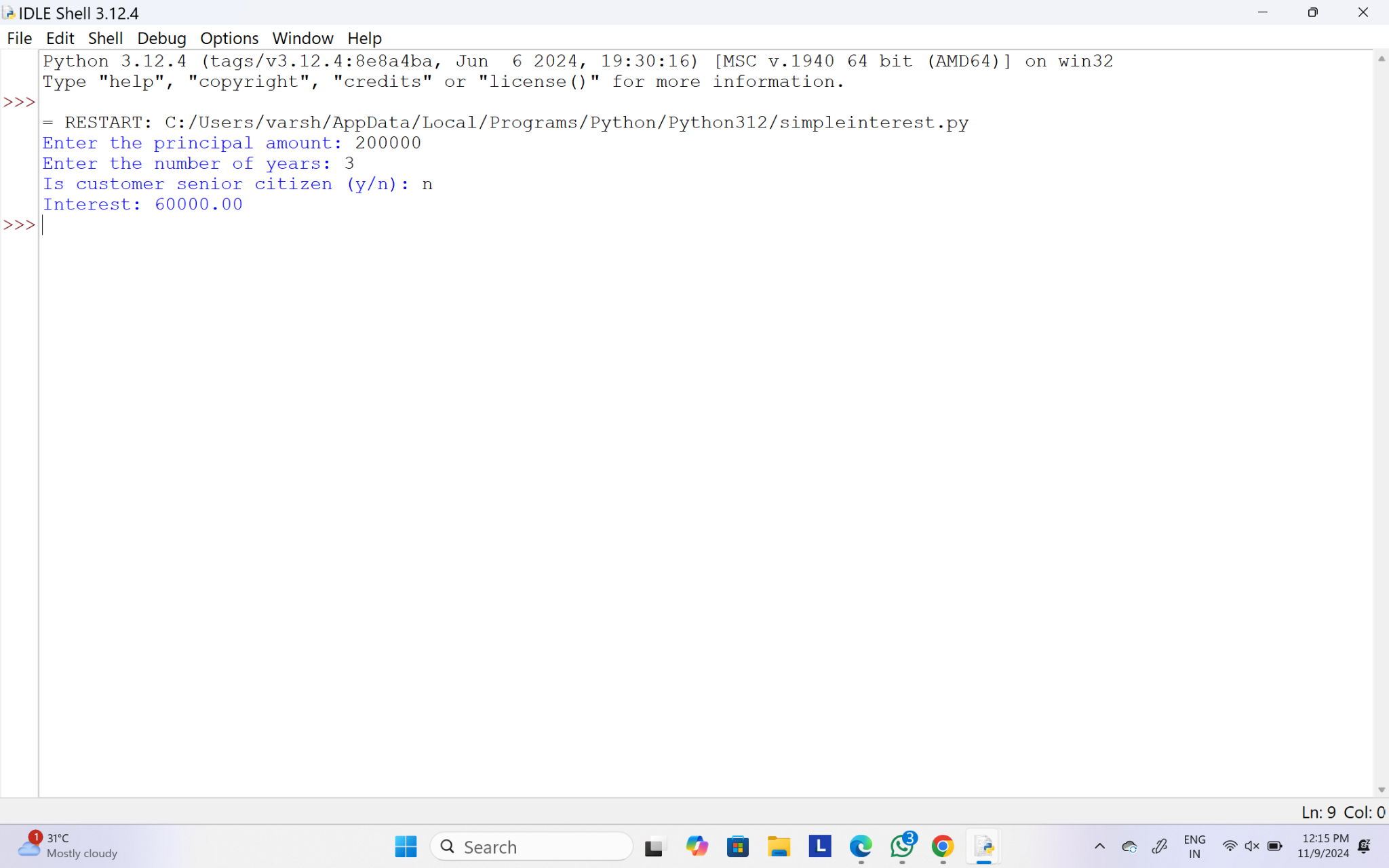
interest = calculate\_simple\_interest(principal, years, is\_senior\_citizen)

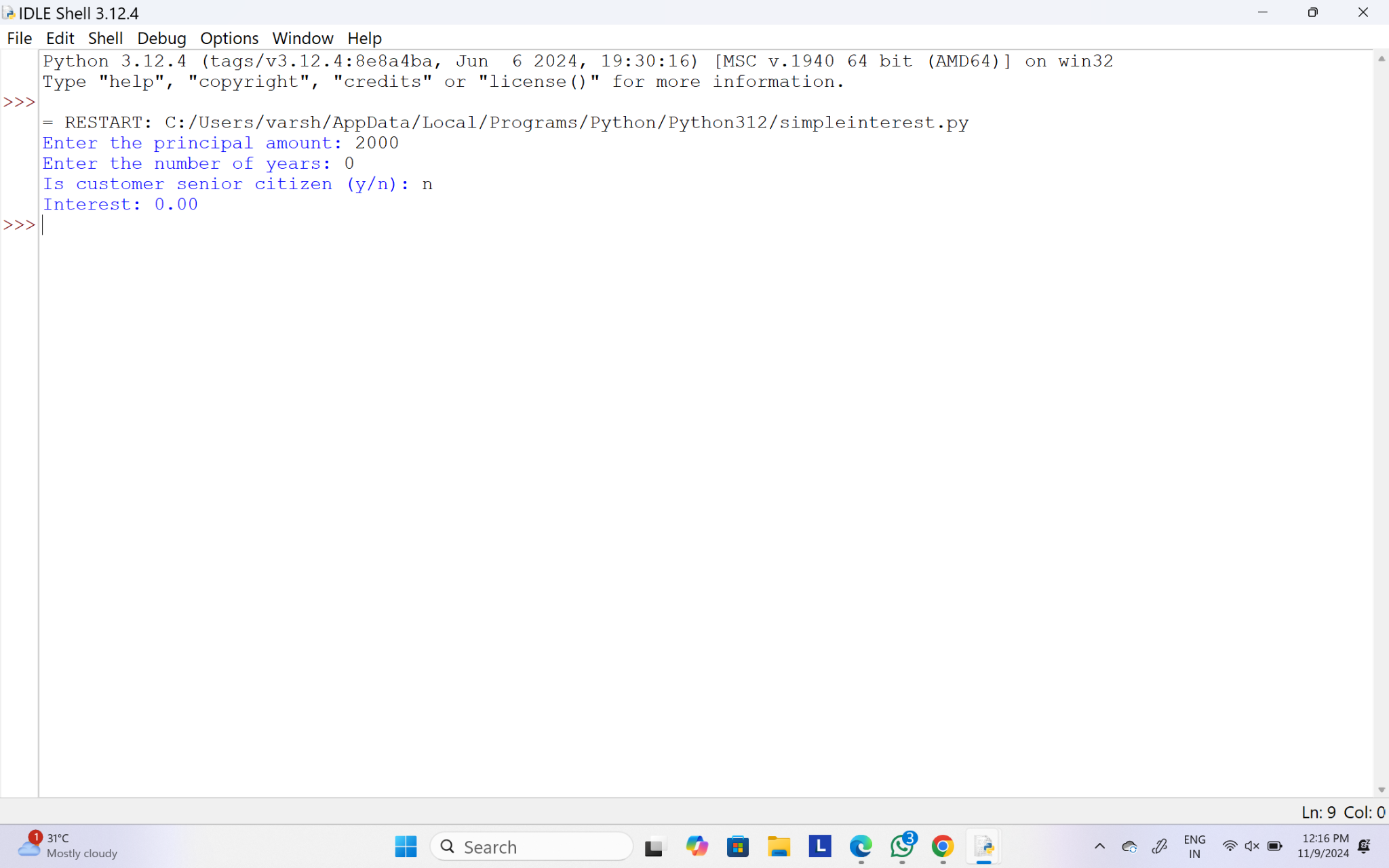
print(f"Interest: {interest:.2f}")

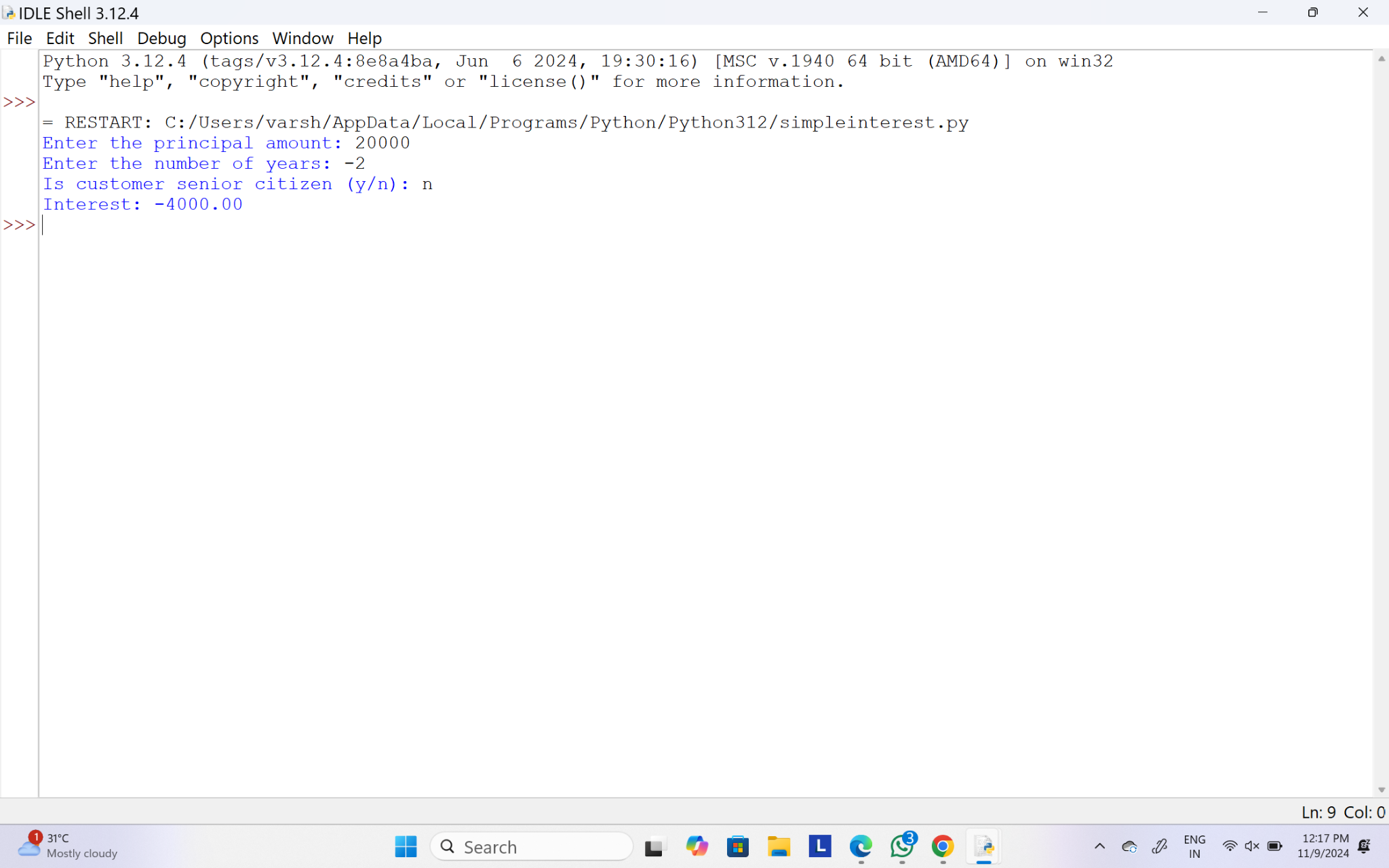
if \_\_name\_\_ == "\_\_main\_\_":

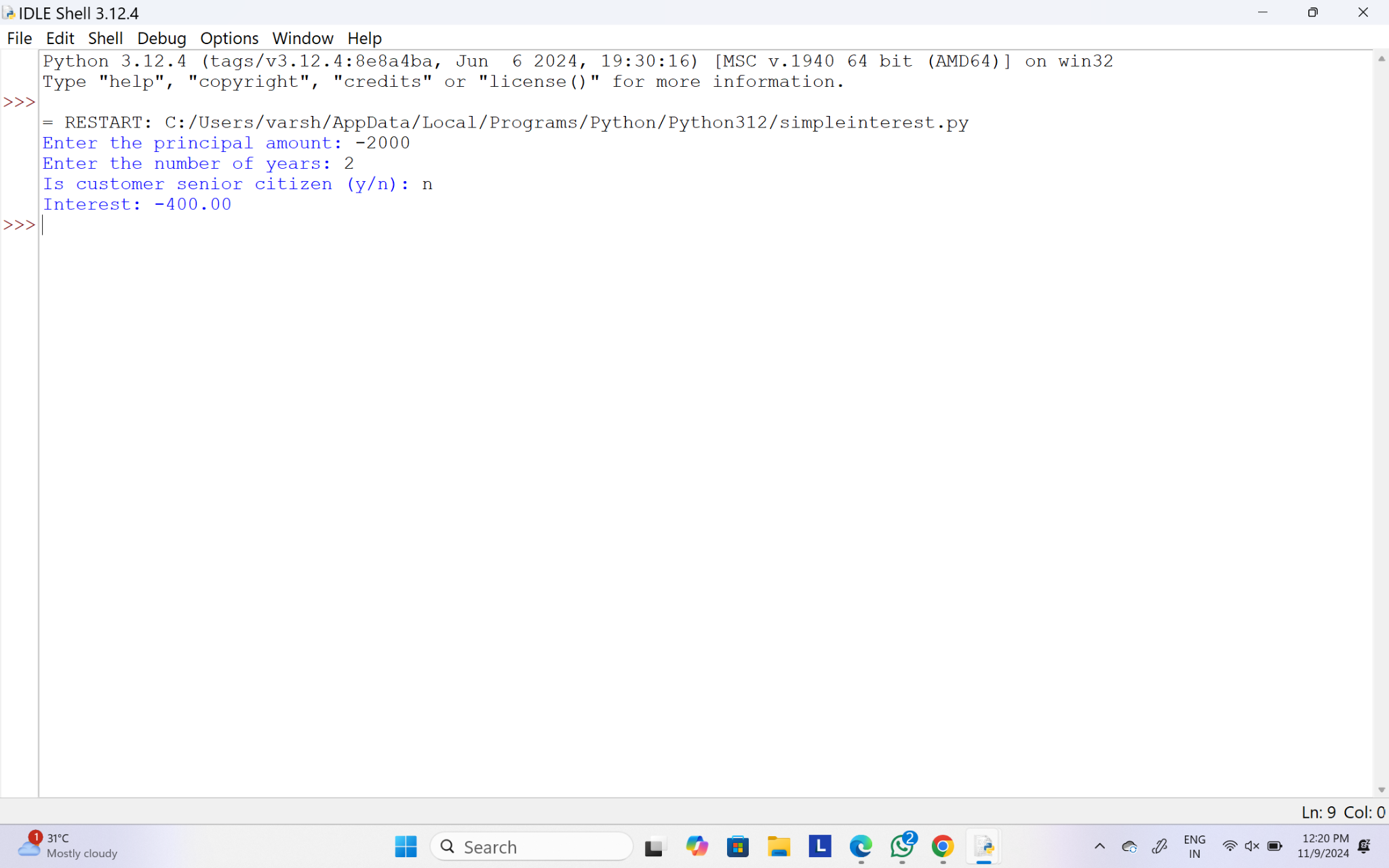
main()

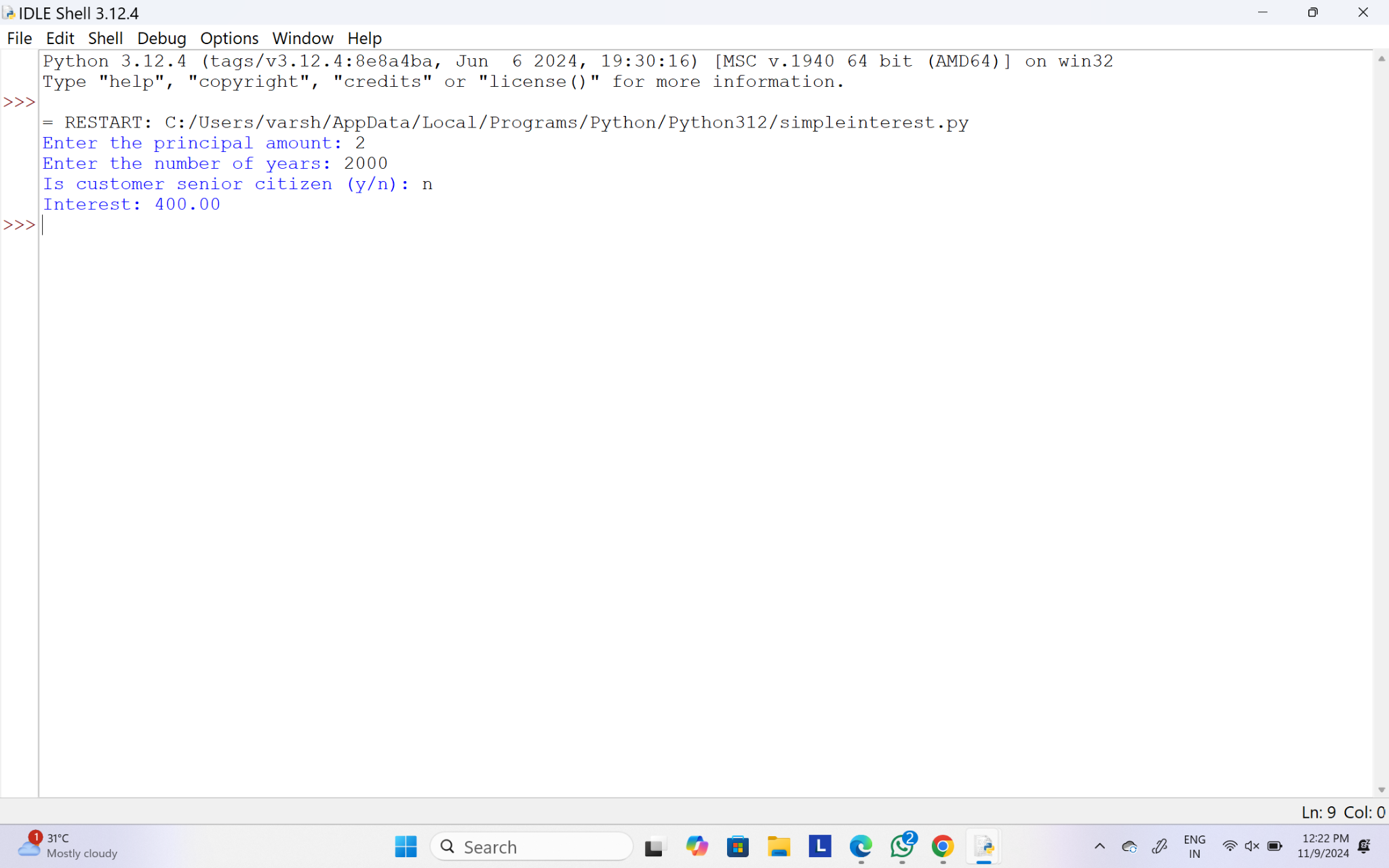
**OUTPUT:**

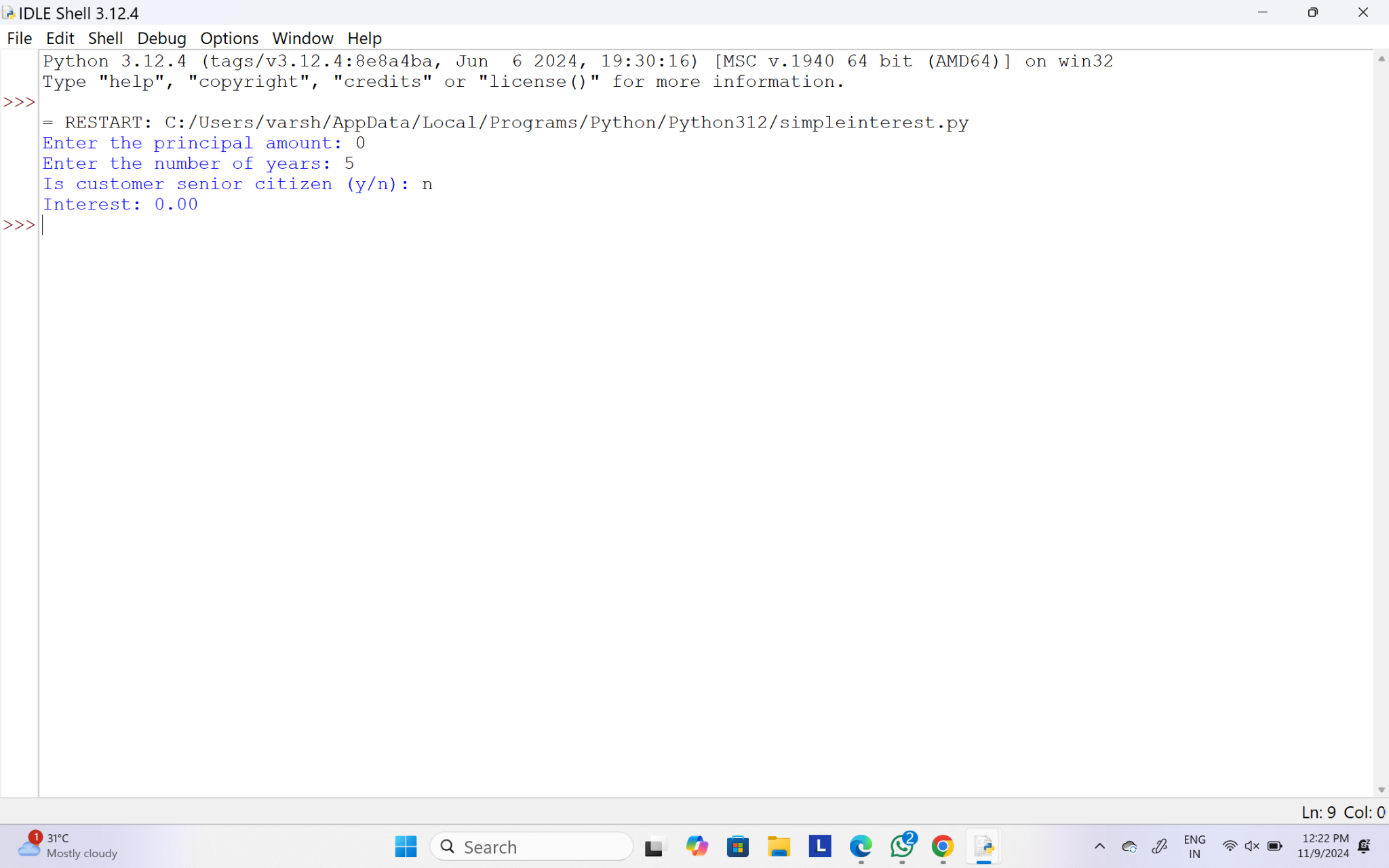
****

****

****

****

****

****

19.Write a program to print number of factors and to print nth factor of the given number.

Sample Input:

Given Number: 100

N = 4

Sample Output:

Number of factors = 9

4th factor of 100 = 5

**Test Cases:**

1. Given Number = 512 , N = 6
2. Given Number = 343 , N = 7
3. Given Number = 1024 , N = 0
4. Given Number = -6561 , N = 3
5. Given Number = 0 , N = 2

**PROGRAM:**

def get\_factors(number):

"""Returns a list of factors of the given number."""

factors = []

if number <= 0:

return factors # No factors for 0 or negative numbers

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

if number % i == 0:

factors.append(i)

return factors

def main():

number = int(input("Given Number: "))

n = int(input("N = "))

if number <= 0

print("Number should be positive to have factors.")

return

factors = get\_factors(number)

print(f"Number of factors = {len(factors)}")

if n > 0 and n <= len(factors):

print(f"{n}th factor of {number} = {factors[n - 1]}")

else:

print(f"{n}th factor of {number} is invalid.")

def test\_cases():

test\_data = [

(100, 4),

(512, 6),

(343, 7),

(1024, 0),

(-6561, 3),

(0, 2)

]

for number, n in test\_data:

print(f"Given Number = {number}, N = {n}")

print("Output:")

global input

input = lambda prompt: str([number, n][test\_data.index((number, n)) == 0]) if prompt.startswith('Given') else str(n)

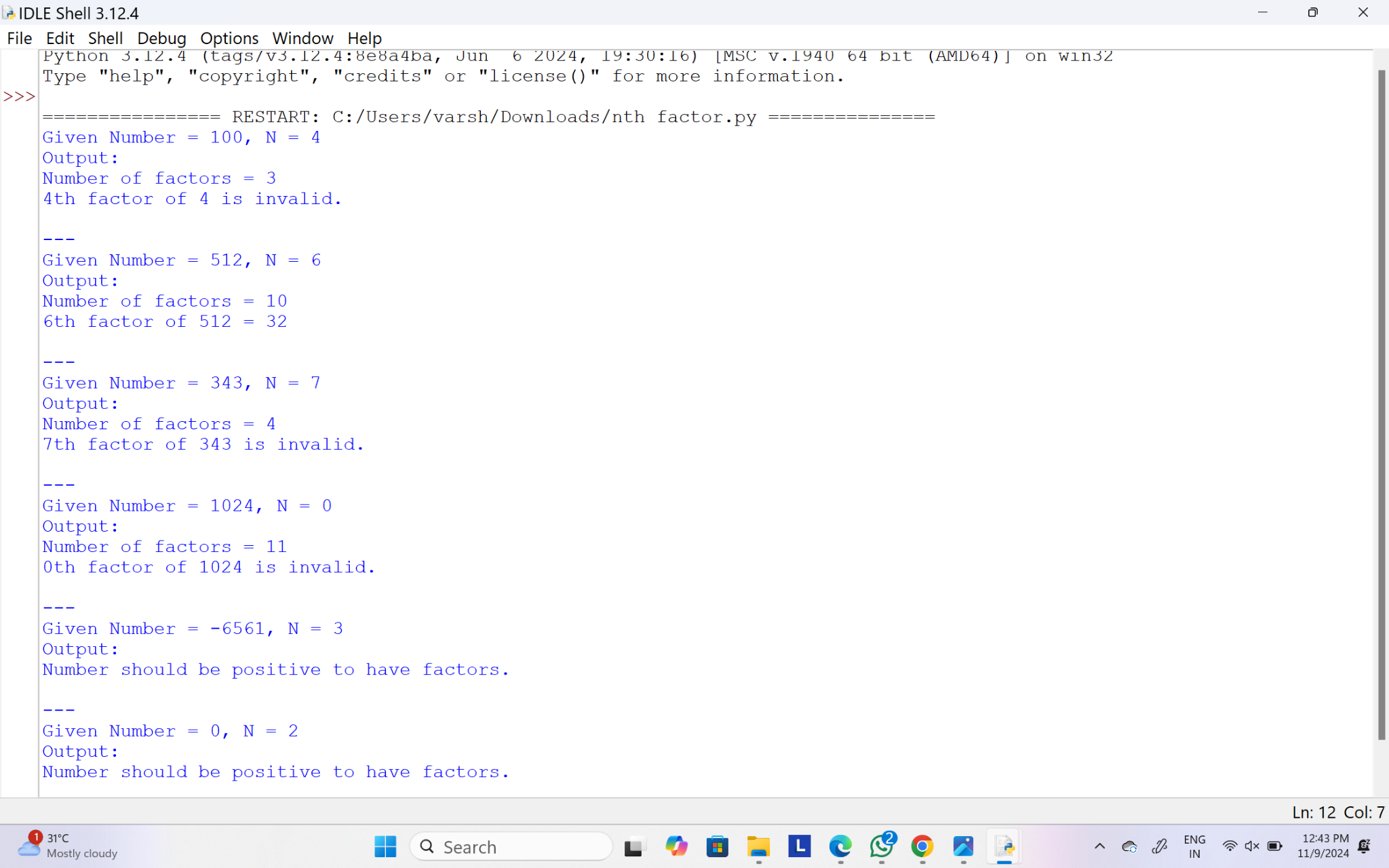
main()

print("\n---")

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

test\_cases()

**OUTPUT:**



20.Write a program to print hollow square symbol pattern? Get the symbol input from the user

**PROGRAM:**

def print\_hollow\_square(size, symbol):

for i in range(size):

for j in range(size):

if i == 0 or i == size - 1 or j == 0 or j == size - 1:

print(symbol, end=' ') # Print symbol at borders

else:

print(' ', end=' ')

print()

def main():

size = int(input("Enter the size of the square: "))

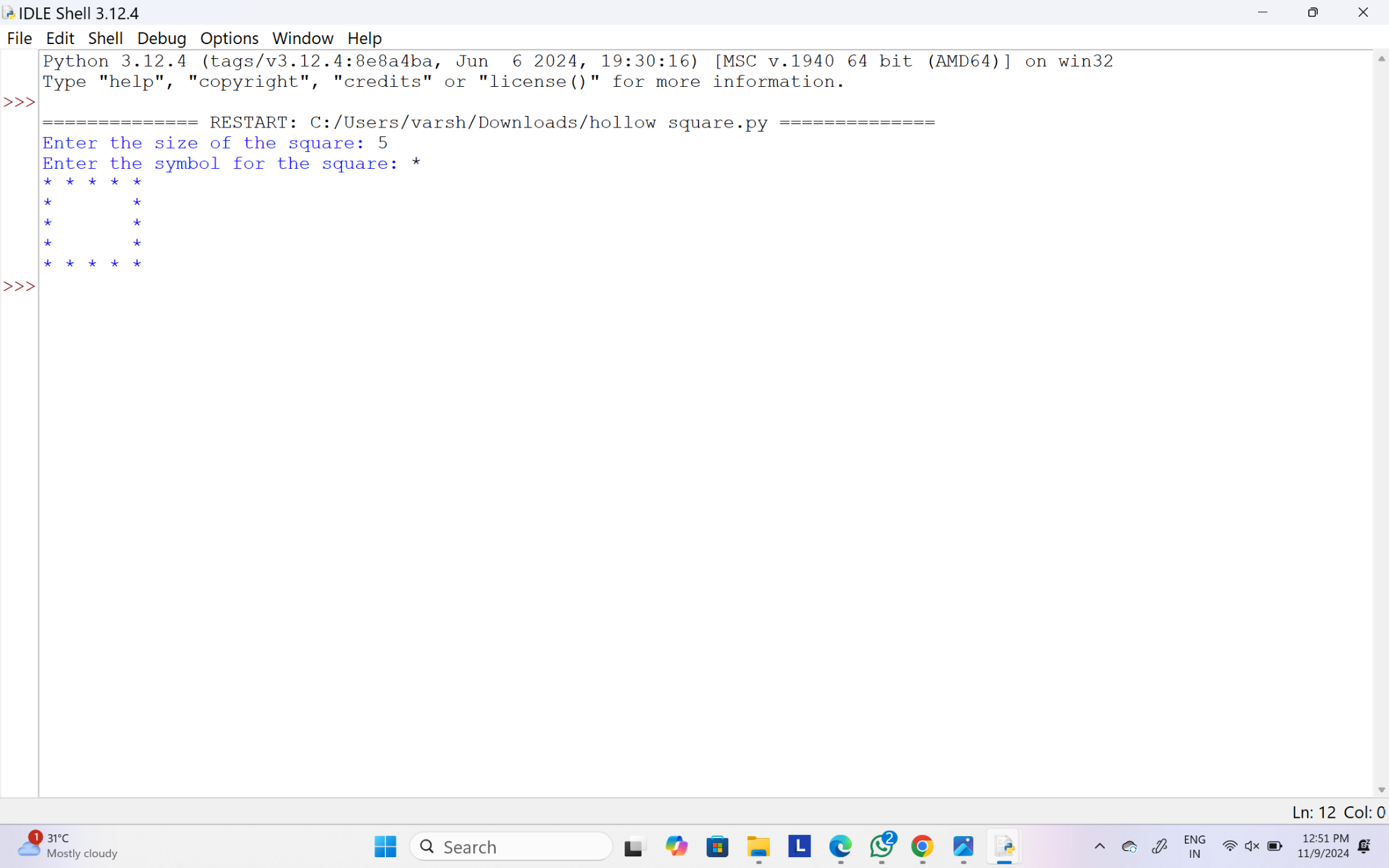
symbol = input("Enter the symbol for the square: ")

print\_hollow\_square(size, symbol)

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

main()

**OUTPUT:**

****