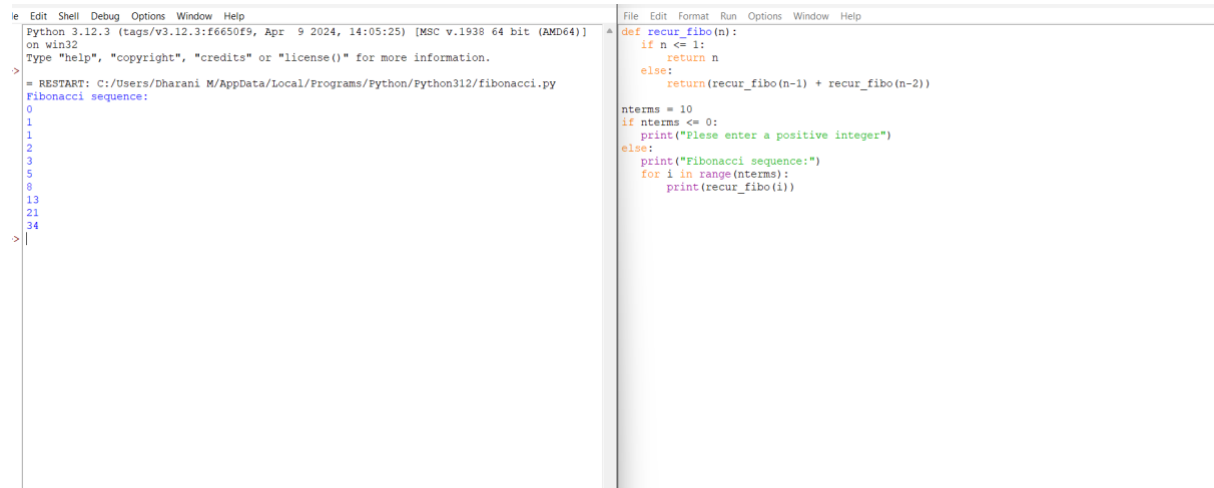


1. Write a program to Print Fibonacci Series using recursion.

$$F(n) = F(n-1) + F(n-2)$$

TIME COMPLEXITY:  $O(2n)$



The screenshot shows a Python IDE with two windows. The left window displays the output of a Fibonacci sequence program, showing the sequence from 0 to 34. The right window shows the source code of the program, which uses a recursive function to calculate the Fibonacci sequence.

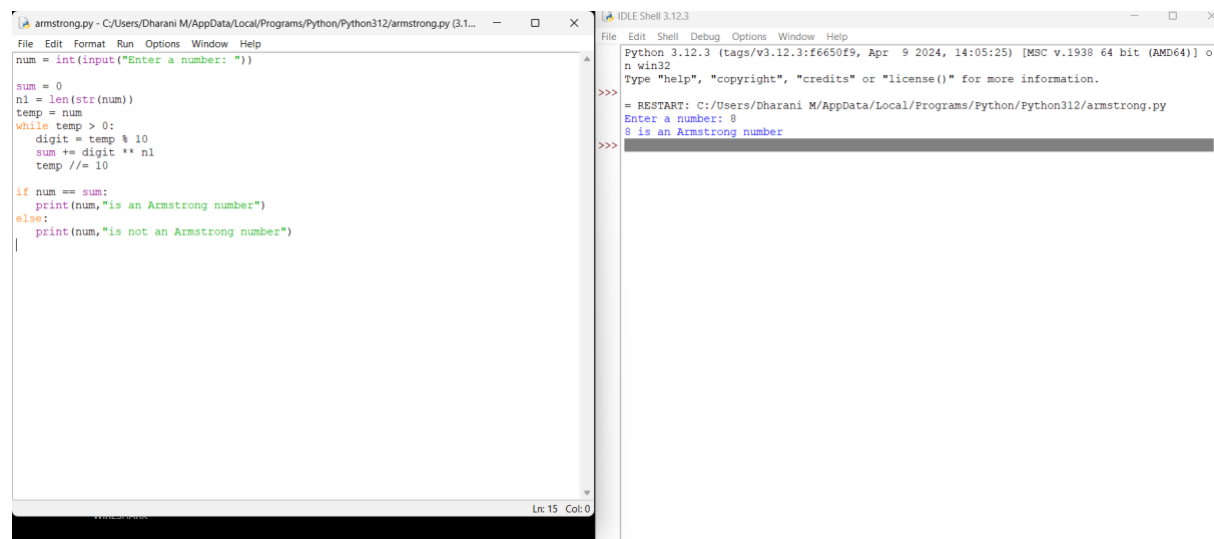
```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/fibonacci.py
Fibonacci sequence:
0
1
1
2
3
5
8
13
21
34
>

def recur_fibo(n):
    if n <= 1:
        return n
    else:
        return(recur_fibo(n-1) + recur_fibo(n-2))

nterms = 10
if nterms <= 0:
    print("Please enter a positive integer")
else:
    print("Fibonacci sequence:")
    for i in range(nterms):
        print(recur_fibo(i))
```

2. Write a program to check the given no is Armstrong or not using recursive function.

TIME COMPLEXITY:  $O(1)$ .



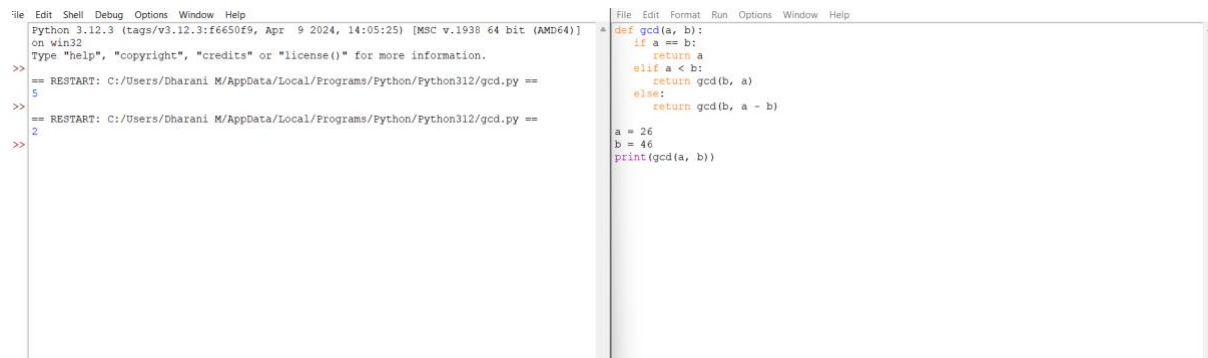
The screenshot shows a Python IDE with two windows. The left window displays the source code of a program that checks if a number is an Armstrong number. The right window shows the output of the program, where the user enters the number 8 and the program outputs "8 is an Armstrong number".

```
armstrong.py - C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/armstrong.py (3.1...
File Edit Format Run Options Window Help
num = int(input("Enter a number: "))
sum = 0
n1 = len(str(num))
temp = num
while temp > 0:
    digit = temp % 10
    sum += digit ** n1
    temp //= 10
if num == sum:
    print(num,"is an Armstrong number")
else:
    print(num,"is not an Armstrong number")
|

Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] o
n win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/armstrong.py
Enter a number: 8
8 is an Armstrong number
>>>
```

### 3. Write a program to find the GCD of two numbers using recursive factorization

TIME COMPLEXITY:  $O(\log(\min(a,b)))$ ,



The screenshot shows a Python IDE with two windows. The left window displays the execution of a script named 'gcd.py'. The output shows the program restarting twice, with the first run returning 5 and the second run returning 2. The right window shows the source code of the 'gcd.py' file, which defines a recursive function 'gcd(a, b)' and calls it with 'a = 26' and 'b = 46'.

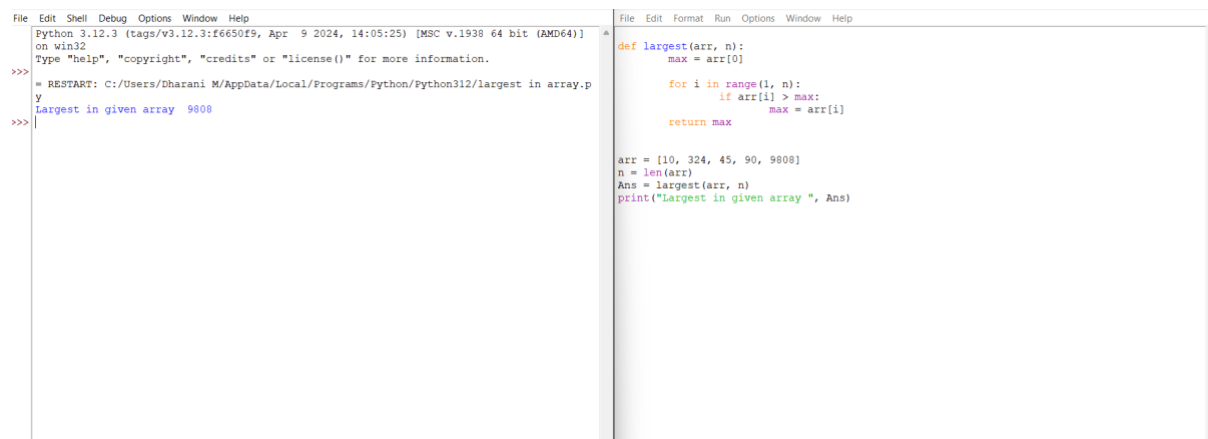
```
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> == RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/gcd.py ==
5
>>> == RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/gcd.py ==
2
>>>
```

```
File Edit Format Run Options Window Help
def gcd(a, b):
    if a == b:
        return a
    elif a < b:
        return gcd(b, a)
    else:
        return gcd(b, a - b)

a = 26
b = 46
print(gcd(a, b))
```

### 4. Write a program to get the largest element of an array.

TIME COMPLEXITY:  $O(N)$ ,



The screenshot shows a Python IDE with two windows. The left window displays the execution of a script named 'largest in array.py'. The output shows the program restarting once, with the largest element in the array being 9808. The right window shows the source code of the 'largest in array.py' file, which defines a function 'largest(arr, n)' and calls it with 'arr = [10, 324, 45, 90, 9808]' and 'n = len(arr)'.

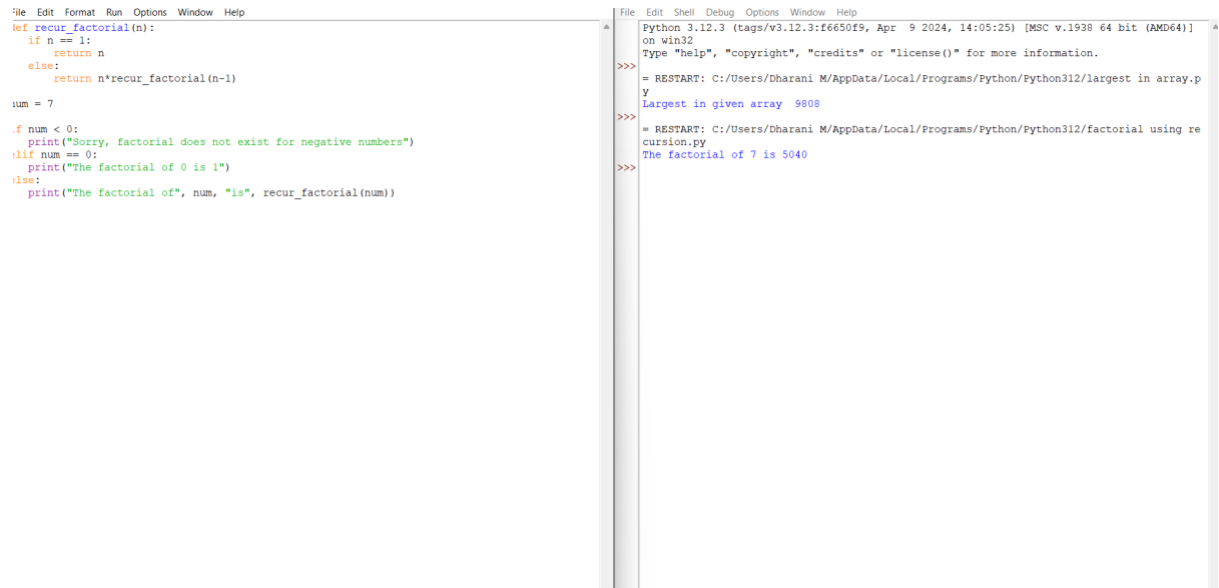
```
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> == RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/largest in array.py
Largest in given array 9808
>>>
```

```
File Edit Format Run Options Window Help
def largest(arr, n):
    max = arr[0]
    for i in range(1, n):
        if arr[i] > max:
            max = arr[i]
    return max

arr = [10, 324, 45, 90, 9808]
n = len(arr)
Ans = largest(arr, n)
print("Largest in given array ", Ans)
```

5. Write a program to find the Factorial of a number using recursion.

TIME COMPLEXITY:  $O(N)$



```
File Edit Format Run Options Window Help
def recur_factorial(n):
    if n == 1:
        return n
    else:
        return n*recur_factorial(n-1)

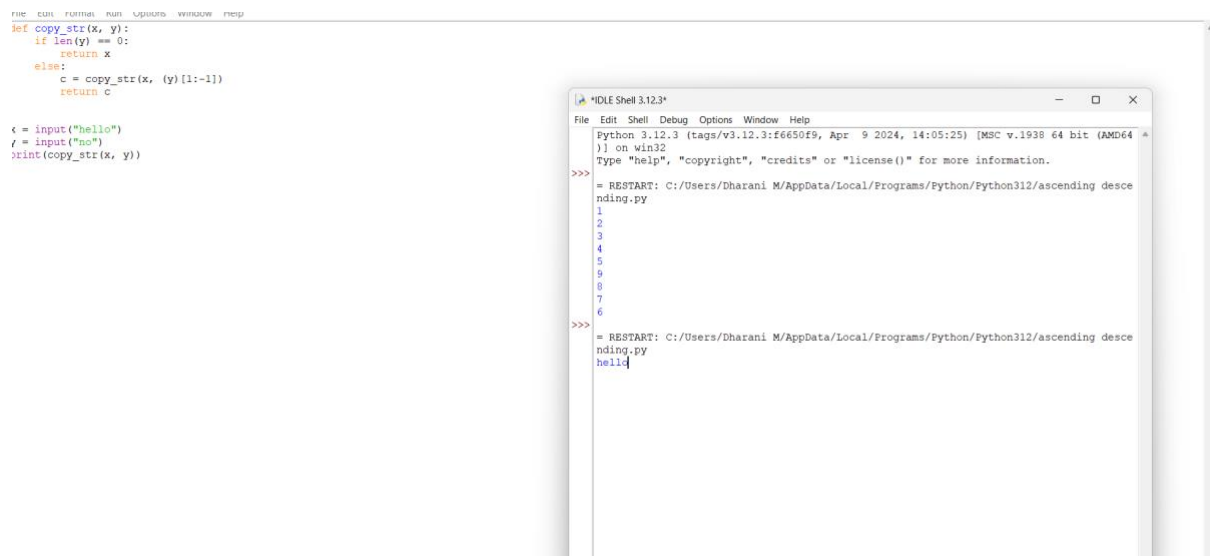
num = 7

if num < 0:
    print("Sorry, factorial does not exist for negative numbers")
elif num == 0:
    print("The factorial of 0 is 1")
else:
    print("The factorial of", num, "is", recur_factorial(num))
```

```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/largest in array.p
y
Largest in given array 9808
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/factorial using re
cursion.py
The factorial of 7 is 5040
>>>
```

6. Write a program for to copy one string to another using recursion

TIME COMPLEXITY:  $O(m)$



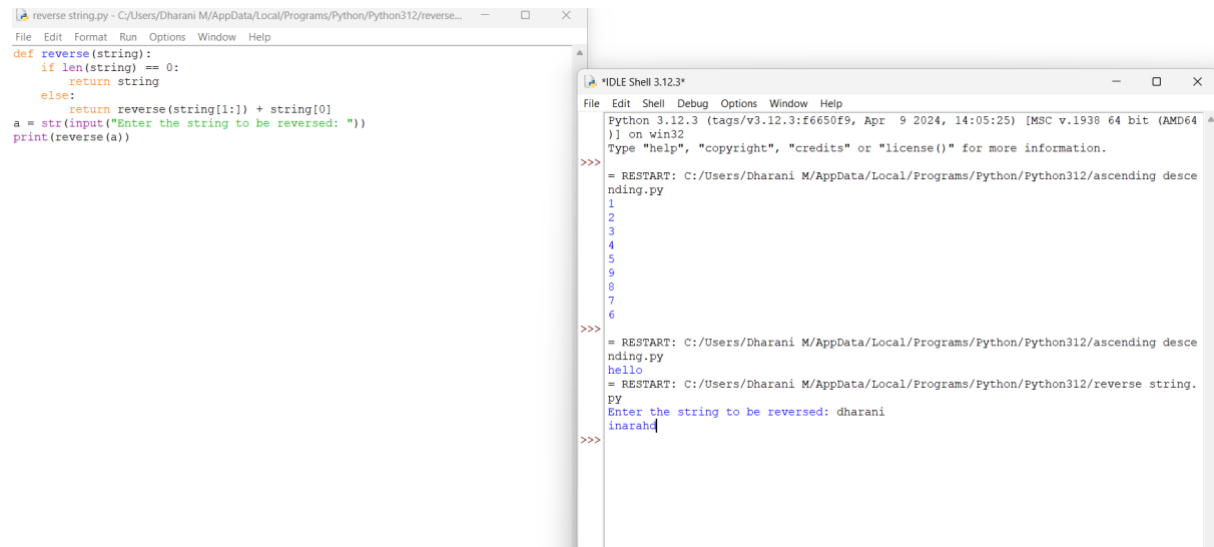
```
File Edit Format Run Options Window Help
def copy_str(x, y):
    if len(y) == 0:
        return x
    else:
        c = copy_str(x, (y)[1:-1])
        return c

x = input("hello")
y = input("no")
print(copy_str(x, y))
```

```
IDLE Shell 3.12.3
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/ascending desce
nding.py
1
2
3
4
5
6
7
8
9
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/ascending desce
nding.py
hello
```

7. Write a program to print the reverse of a string using recursion

TIME COMPLEXITY:  $O(n)$



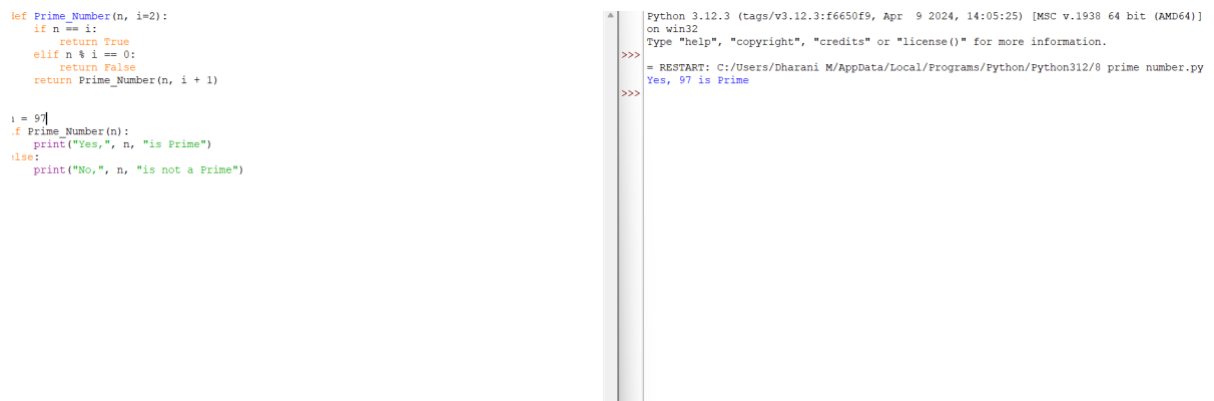
The image shows a Python IDE with two windows. The left window displays the source code for a recursive string reversal program. The right window shows the execution output.

```
reverse string.py - C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/reverse...
File Edit Format Run Options Window Help
def reverse(string):
    if len(string) == 0:
        return string
    else:
        return reverse(string[1:]) + string[0]
a = str(input("Enter the string to be reversed: "))
print(reverse(a))
```

```
*IDLE Shell 3.12.3*
File Edit Shell Debug Options Window Help
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/ascending descending.py
1
2
3
4
5
9
8
7
6
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/ascending descending.py
hello
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/reverse string.py
Enter the string to be reversed: dharani
inarahd
>>>
```

8. Write a program to generate all the prime numbers using recursion

TIME COMPLEXITY:  $O(\sqrt{N})$ .



The image shows a Python IDE with two windows. The left window displays the source code for a recursive prime number generation program. The right window shows the execution output.

```
def Prime_Number(n, i=2):
    if n == i:
        return True
    elif n % i == 0:
        return False
    return Prime_Number(n, i + 1)

i = 97
if Prime_Number(n):
    print("Yes,", n, "is Prime")
else:
    print("No,", n, "is not a Prime")
```

```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/8 prime number.py
Yes, 97 is Prime
>>>
```

9. Write a program to check a number is a prime number or not using recursion.

TIME COMPLEXITY:  $O(n)$

```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/prime number witho
ut recur.py
Not Prime
>>>
```

10. Write a program for to check whether a given String is Palindrome or not using recursion

TIME COMPLEXITY:  $O(n)$

```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/pallindrome.py
Yes
>>>
```

```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/pallindrome.py
Yes
>>>
```