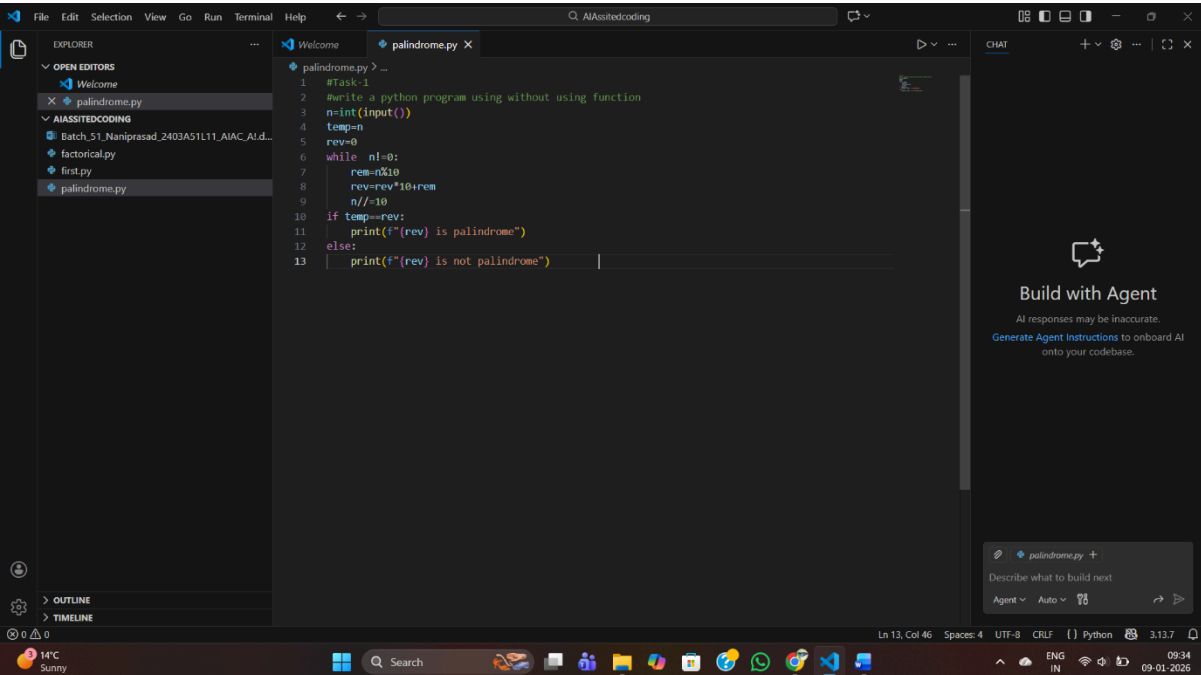


2403A51L29

batch-51

#Task1

Write a python program for palindrome without using function

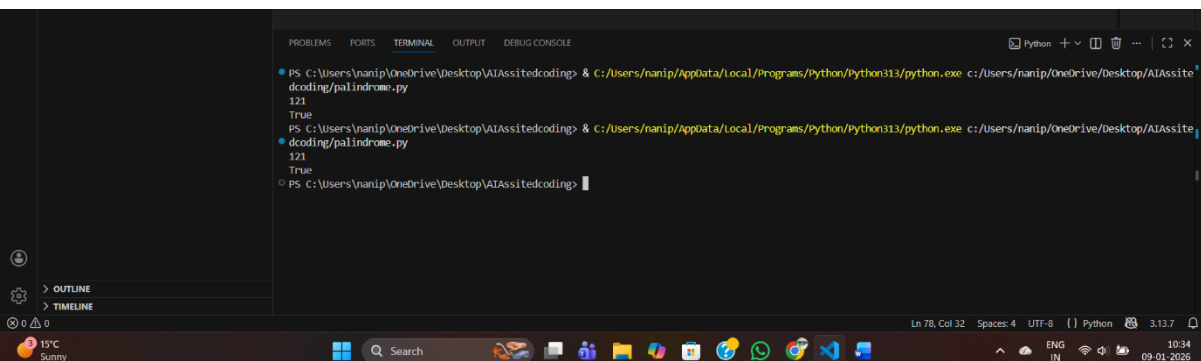


The screenshot shows a Visual Studio Code editor with a file named `palindrome.py` open. The code is as follows:

```
1 #Task-1
2 #write a python program using without using function
3 n=int(input())
4 temp=n
5 rev=0
6 while n!=0:
7     rem=n%10
8     rev=rev*10+rem
9     n//=10
10 if temp==rev:
11     print(f"{rev} is palindrome")
12 else:
13     print(f"{rev} is not palindrome")
```

The interface also shows an Explorer sidebar on the left with the file structure, a Chat sidebar on the right with a "Build with Agent" prompt, and a Windows taskbar at the bottom.

Output:



The screenshot shows a terminal window with the following output:

```
PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding> & C:/Users/nanip/AppData/Local/Programs/Python/Python313/python.exe c:/Users/nanip/OneDrive/Desktop/AIAssistedcoding/palindrome.py
121
True
PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding> & C:/Users/nanip/AppData/Local/Programs/Python/Python313/python.exe c:/Users/nanip/OneDrive/Desktop/AIAssistedcoding/palindrome.py
121
True
PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding>
```

The terminal window is part of a VS Code interface, showing the command prompt and the execution of the `palindrome.py` file. The output shows the program running twice, both times taking the input '121' and outputting 'True'.

Palindrome check steps for the given code

1. Read input:

- Take an integer from the user and store it in `n`.

2. Store original number:

- Copy n into temp so you can compare later after reversing.

3. Initialize reverse:

- Set $rev = 0$. This will be built digit by digit into the reversed number.

4. Loop until n becomes 0:

- Keep extracting the last digit and removing it from n using integer division.

5. Extract last digit:

- $rem = n \% 10$
- This gives the rightmost digit of n .

6. Append digit to reversed number:

- $rev = rev * 10 + rem$
- Shifts existing digits in rev left and adds the new last digit.

7. Remove last digit from n :

- $n //= 10$
- Drops the rightmost digit from n to process the next one.

8. End of loop:

- When n becomes 0, rev now holds the full reversed number.

9. Compare original with reversed:

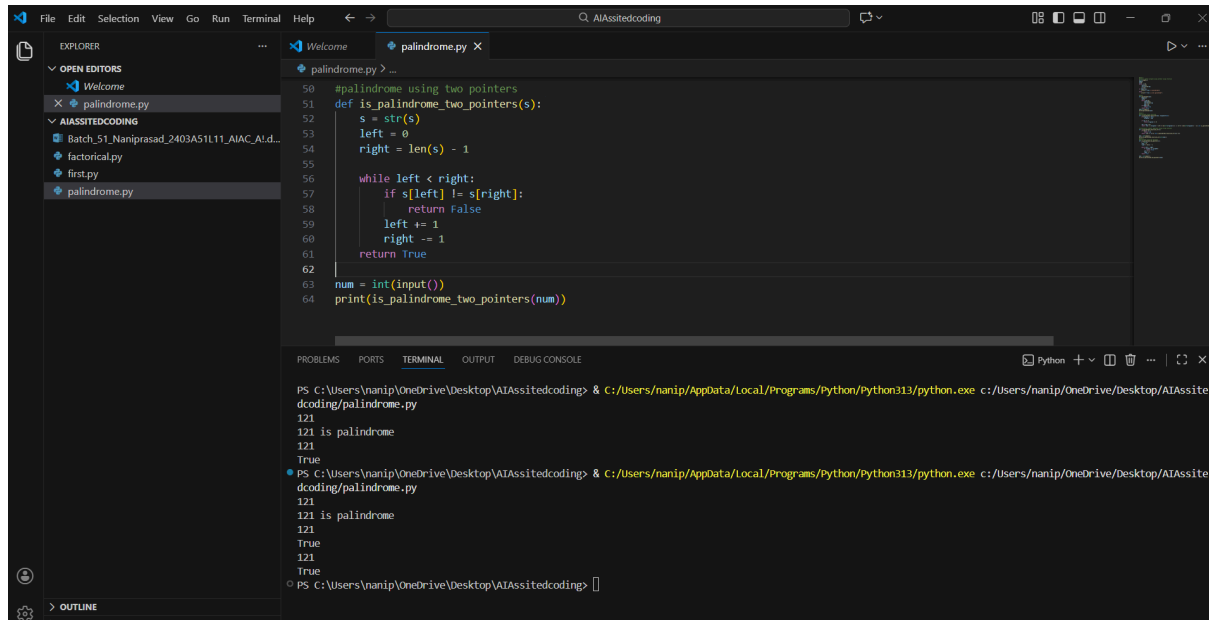
- If $temp == rev$, the original number reads the same backward → it's a palindrome.
- Otherwise, it's not a palindrome.

10. Output result:

- Print "rev is palindrome" if equal, else "rev is not palindrome".

#Task2:

Write optimal solution for palindrome solution



```
50 #palindrome using two pointers
51 def is_palindrome_two_pointers(s):
52     s = str(s)
53     left = 0
54     right = len(s) - 1
55
56     while left < right:
57         if s[left] != s[right]:
58             return False
59         left += 1
60         right -= 1
61     return True
62
63 num = int(input())
64 print(is_palindrome_two_pointers(num))
```

PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding> & C:/Users/nanip/AppData/Local/Programs/Python/Python313/python.exe c:/Users/nanip/OneDrive/Desktop/AIAssistedcoding/palindrome.py

121

121 is palindrome

121

True

PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding> & C:/Users/nanip/AppData/Local/Programs/Python/Python313/python.exe c:/Users/nanip/OneDrive/Desktop/AIAssistedcoding/palindrome.py

121

121 is palindrome

121

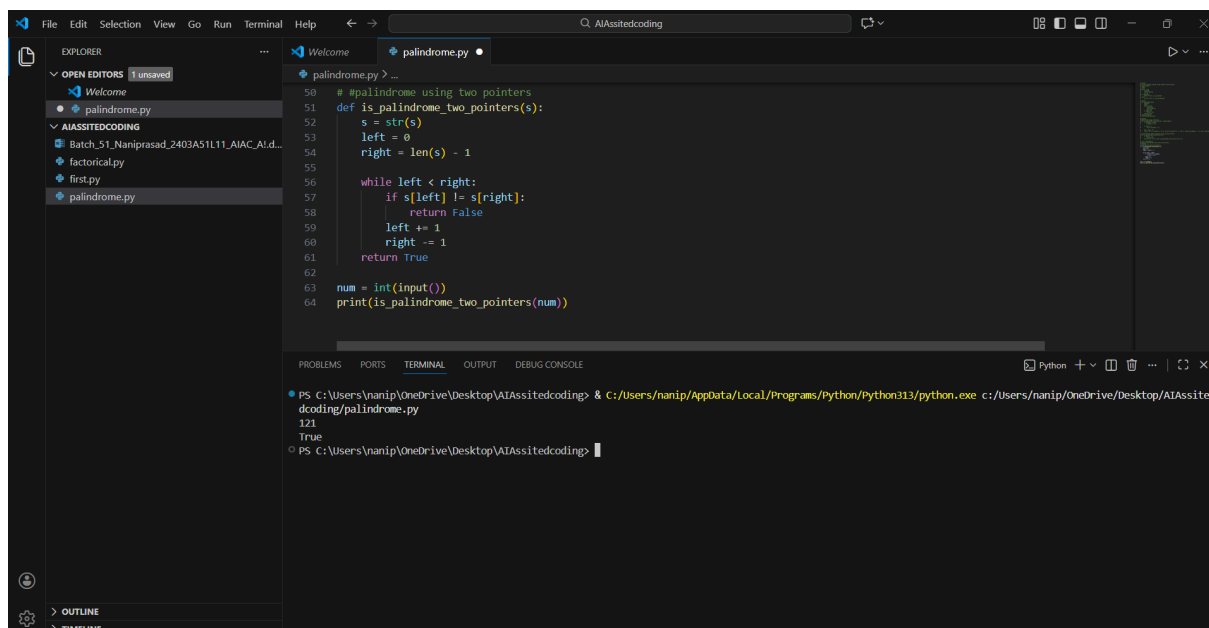
True

121

True

PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding>

Output:



```
50 #palindrome using two pointers
51 def is_palindrome_two_pointers(s):
52     s = str(s)
53     left = 0
54     right = len(s) - 1
55
56     while left < right:
57         if s[left] != s[right]:
58             return False
59         left += 1
60         right -= 1
61     return True
62
63 num = int(input())
64 print(is_palindrome_two_pointers(num))
```

PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding> & C:/Users/nanip/AppData/Local/Programs/Python/Python313/python.exe c:/Users/nanip/OneDrive/Desktop/AIAssistedcoding/palindrome.py

121

True

PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding>

Explanation:

Create function

Pass the input with some value

In two pointer if last and first value are equal then

Last-=1

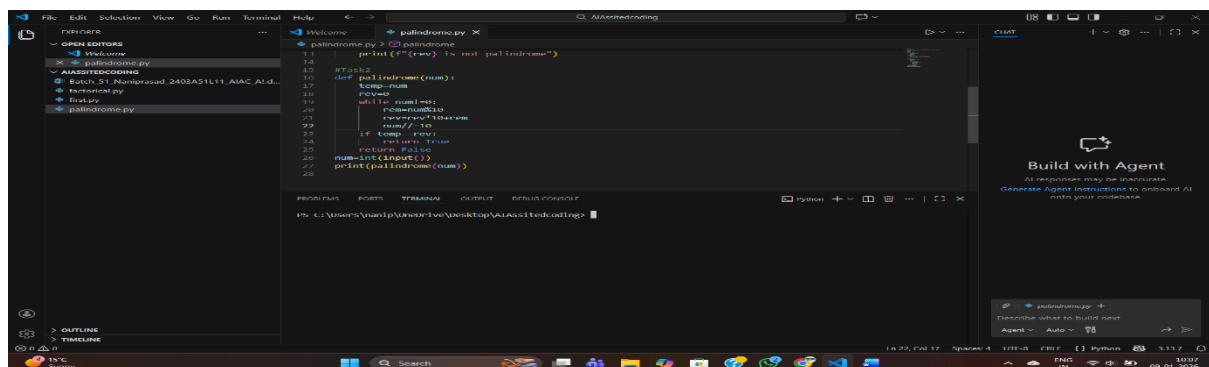
And first+=1

So if all index values are equal checking the last and first return True

If not return False

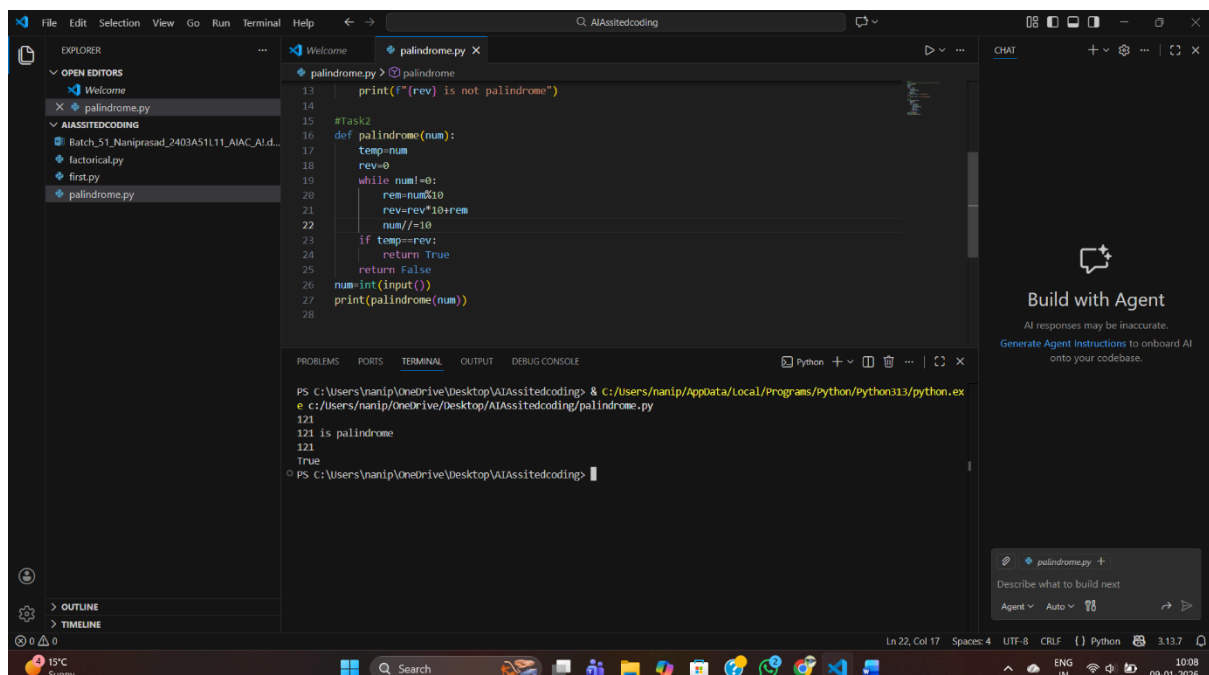
#Task 3

Write python program for palindrome using function



```
13 print(f"({rev}) is not palindrome")
14
15 #Task2
16 def palindrome(num):
17     temp=num
18     rev=0
19     while num!=0:
20         rem=num%10
21         rev=rev*10+rem
22         num//=10
23     if temp==rev:
24         return True
25     return False
26 num=int(input())
27 print(palindrome(num))
28
```

Output:



```
13 print(f"({rev}) is not palindrome")
14
15 #Task2
16 def palindrome(num):
17     temp=num
18     rev=0
19     while num!=0:
20         rem=num%10
21         rev=rev*10+rem
22         num//=10
23     if temp==rev:
24         return True
25     return False
26 num=int(input())
27 print(palindrome(num))
28
```

PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding> & C:/Users/nanip/Appdata/Local/Programs/python/python313/python.exe c:/Users/nanip/OneDrive/desktop/AIAssistedcoding/palindrome.py

121

121 is palindrome

121

True

PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding>

Explanation:

Step-by-Step Explanation

1. Function Definition

- `def palindrome(num):`
- A function named `palindrome` is created that takes one argument `num`.

2. Store Original Number

- `temp = num`
- The original number is stored in `temp` so we can compare later.

3. Initialize Reverse

- `rev = 0`
- This variable will hold the reversed number.

4. Loop to Reverse Number

- `while num != 0:` → keep looping until `num` becomes 0.
- Inside the loop:
- `rem = num % 10` → extract the last digit.
- `rev = rev * 10 + rem` → build the reversed number digit by digit.
- `num //= 10` → remove the last digit from `num`.

5. Check Palindrome

- After the loop ends, `rev` contains the reversed number.
- Compare `temp` (original number) with `rev`.
- If they are equal → return `True`.
- Otherwise → return `False`.



Main Program

- `num = int(input())` → take user input.

- `print(palindrome(num))` → call the function and print the result (True or False).

Example Walkthrough

Suppose input is 121:

- `temp = 121, rev = 0`
- Loop:
 - Iteration 1: `rem = 1, rev = 1, num = 12`
 - Iteration 2: `rem = 2, rev = 12, num = 1`
 - Iteration 3: `rem = 1, rev = 121, num = 0`
- Loop ends → `rev = 121`
- Compare: `temp == rev` → `121 == 121` → True
- Output: True

If input is 123:

- Reverse becomes 321
- Compare: `123 != 321` → False
- Output: False

#Task4:

Write Python program with using function and without using function

The screenshot shows the VS Code editor with a file named `palindrome.py` open. The code is as follows:

```
1 #Task-1
2 #write a python program using without using function
3 n=int(input())
4 temp=n
5 rev=0
6 while n!=0:
7     rem=n%10
8     rev=rev*10+rem
9     n//=10
10 if temp==rev:
11     print(f"{rev} is palindrome")
12 else:
13     print(f"{rev} is not palindrome")
```

The interface includes an Explorer sidebar on the left, a main editor window, and a Chat sidebar on the right with a "Build with Agent" prompt. The status bar at the bottom shows "Ln 13, Col 46" and "Python 3.12.7".

The screenshot shows the VS Code editor with the `palindrome.py` file updated. The code is as follows:

```
66 def is_palindrome_stack(s):
67     s = str(s)
68     stack = []
69     for char in s:
70         stack.append(char)
71
72     for char in s:
73         if char != stack.pop():
74             return False
75     return True
76
77 num = int(input())
78 print(is_palindrome_stack(num))
```

The bottom of the image shows the terminal output:

```
PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding> c:/Users/nanip/AppData/Local/Programs/Python/python313/python.exe c:/Users/nanip/OneDrive/Desktop/coding/palindrome.py
121
True
PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding>
```

Output:

Step-by-Step

1. **Input:** User enters a number → stored in `n`.
2. **Save original:** `temp = n` keeps the original number safe.
3. **Reverse logic:**
 - Extract last digit using `rem = n % 10`.

- Build reversed number: $rev = rev * 10 + rem$.
 - Remove last digit: $n //= 10$.
 - Repeat until n becomes 0.
4. **Compare:** If $temp == rev$, the number is palindrome.
5. **Output:** Prints directly whether palindrome or not.

Step-by-Step

1. **Function defined:** `palindrome(num)` encapsulates the logic.
2. **Inside function:**
 - Store original number in `temp`.
 - Reverse the number using same loop logic.
 - Compare `temp` with `rev`.
 - Return `True` if palindrome, else `False`.
3. **Main program:**
 - Take input from user.
 - Call the function: `palindrome(num)`.
 - Print the returned result (`True` or `False`).


```

66 def is_palindrome_stack(s):
67     s = str(s)
68     stack = []
69     for char in s:
70         stack.append(char)
71
72     for char in s:
73         if char != stack.pop():
74             return False
75     return True
76
77 num = int(input())
78 print(is_palindrome_stack(num))

```

Terminal output:

```

PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding> & C:/Users/nanip/AppData/Local/Programs/Python/Python313/python.exe c:/Users/nanip/OneDrive/Desktop/AIAssite
dcoding/palindrome.py
121
True
PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding> & C:/Users/nanip/AppData/Local/Programs/Python/Python313/python.exe c:/Users/nanip/OneDrive/Desktop/AIAssite
dcoding/palindrome.py
121
True
PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding>

```

#Task5:

Write python program for palindrome using recursion

```

16 def palindrome(num):
24     return True
25     return False
26 num=int(input())
27 print(palindrome(num))
28
29 #Task-3
30 #palindrome using recursion
31 def is_palindrome_recursive(num, original=None):
32     if original is None:
33         original = num
34
35     if num == 0:
36         return original == 0
37
38     rem = num % 10
39     return rem == (original % (10 ** len(str(original)))) // (10 ** (len(str(original)) - 1)) and is_palindrome_recursive(num // 10)
40
41 # Alternative simpler approach using string reversal
42 def is_palindrome_recursive_str(s):
43     if len(s) <= 1:
44         return True
45     return s[0] == s[-1] and is_palindrome_recursive_str(s[1:-1])
46
47 num = int(input())
48 print(is_palindrome_recursive_str(str(num)))

```

Output:

```
16 def palindrome(num):
17     return True
18     return False
19 num=int(input())
20 print(palindrome(num))
21
22 #Task 3
23 #palindrome using recursion
24 def is_palindrome_recursive(num, original=None):
25     if original is None:
26         original = num
27
28     if num == 0:
29         return original == 0
30
31     rem = num % 10
32     return rem == (original % (10 ** len(str(original)))) // (10 ** (len(str(original)) - 1)) and is_palindrome_recursive(num // 10)
33
34
35
36
37
38
39
40
```

```
PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding> & c:/Users/nanip/Appdata/Local/Programs/python/python313/python.exe c:/Users/nanip/OneDrive/Desktop/AIAssite
ddcoding/palindrome.py
121
121 is palindrome
True
PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding> & c:/Users/nanip/Appdata/Local/Programs/python/python313/python.exe c:/Users/nanip/OneDrive/Desktop/AIAssite
ddcoding/palindrome.py
121
121 is palindrome
True
PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding>
```

Step-by-Step Explanation

1. Convert number to string

- `str(num)` turns the input number into a string.
- Example: if user enters 121, then `s = "121"`.

2. Recursive function logic

- `is_palindrome_recursive_str(s)` checks if the string `s` is a palindrome.

3 Execution Example: Input = 121

- `s = "121"`
- Step 1: Compare "1" (first) and "1" (last) → equal → recurse on "2".
- Step 2: "2" has length 1 → base case → return True.
- Final result: True.