

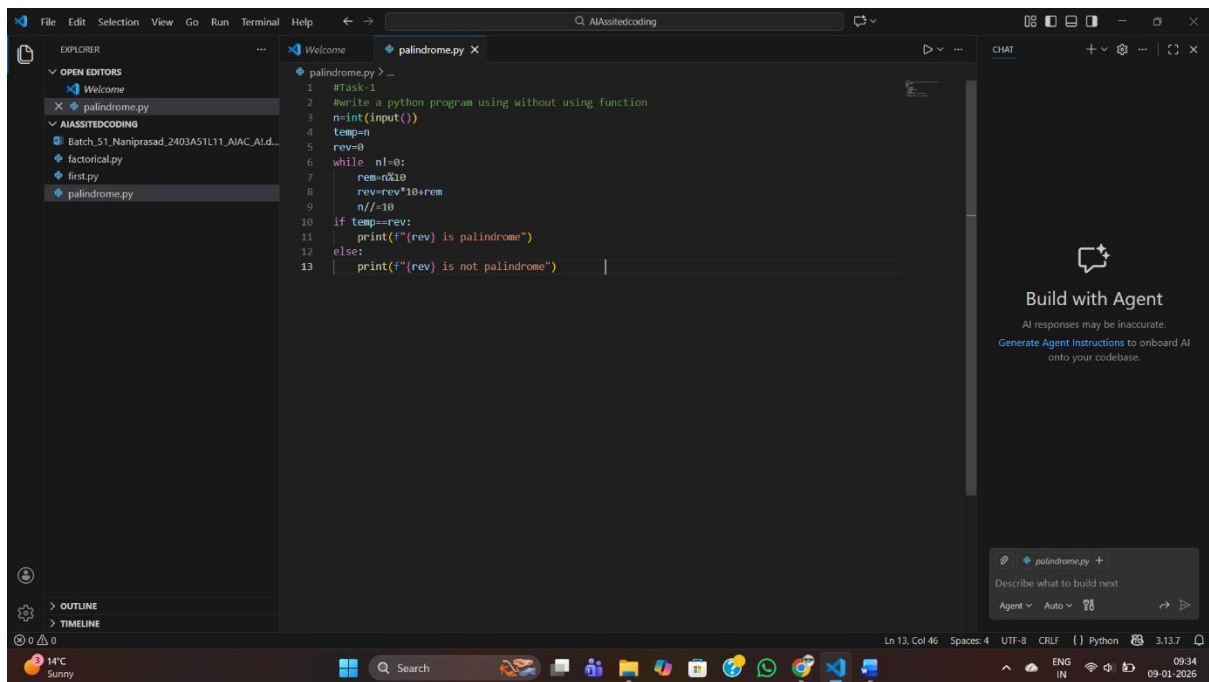
School of Computer Science and Artificial Intelligence

Lab Assignment # 4.2

Program : B. Tech (CSE)
Specialization :
Course Title : AI Assisted coding
Course Code :
Semester : II
Academic Session : 2025-2026
Name of Student : P Abhinav
Enrollment No. : 2403A51L38
Batch No. : 52
Date : 20-01-2026

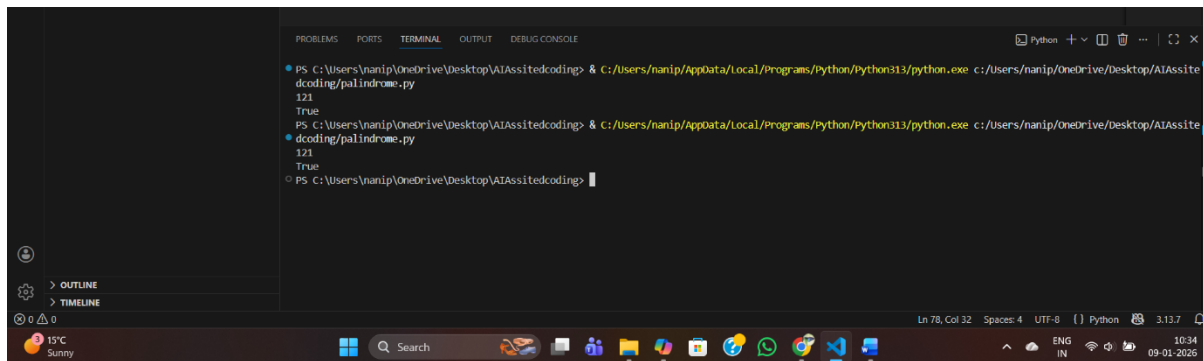
#Task1

Write a python program for palindrome without using function



```
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")
```

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>
```

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:

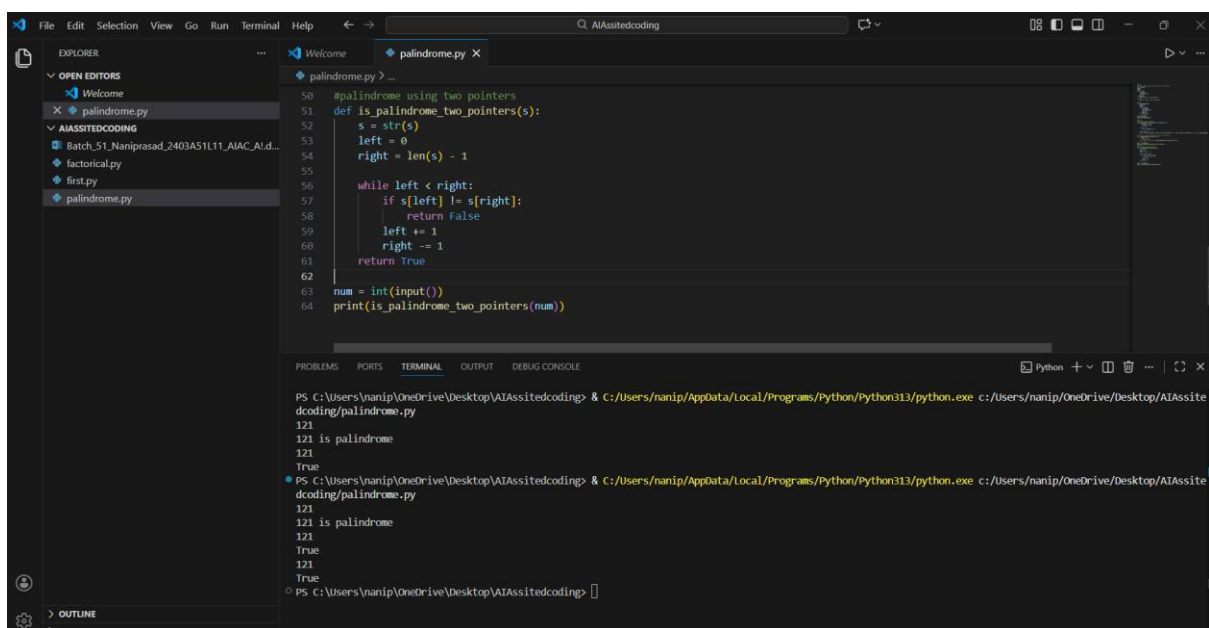
- If temp == rev, the original number reads the same backward → it's a palindrome.
- Otherwise, it's not a palindrome.

10. Output result:

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

#Task2:

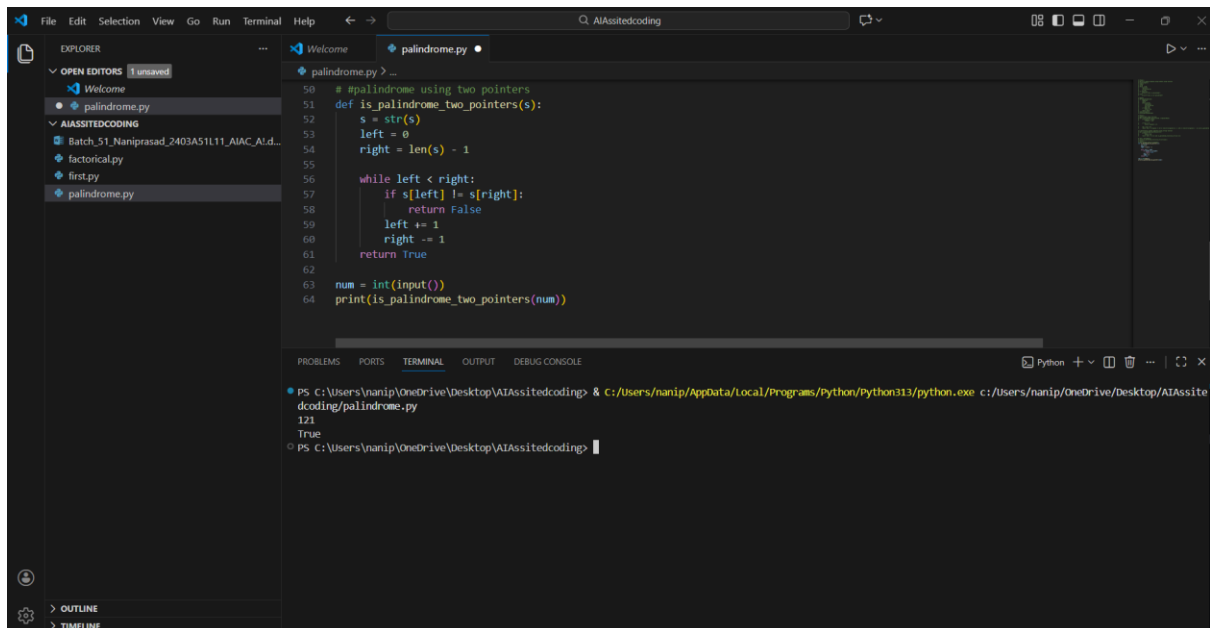
Write optimal solution for palindrome solution



```
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
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))
```

Terminal 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>
```

Explanation:

Create function

Pass the input with some value

In two pointer if last and first value are equal then

Last-=1

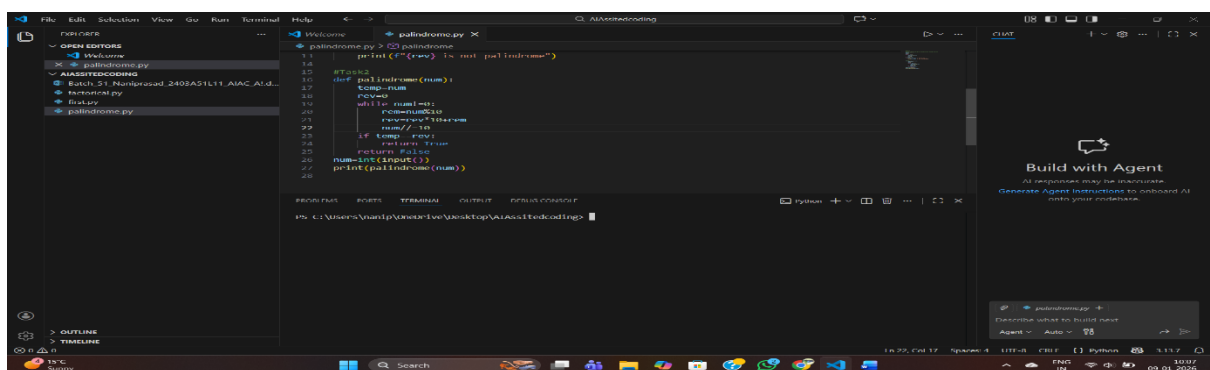
And first+=1

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

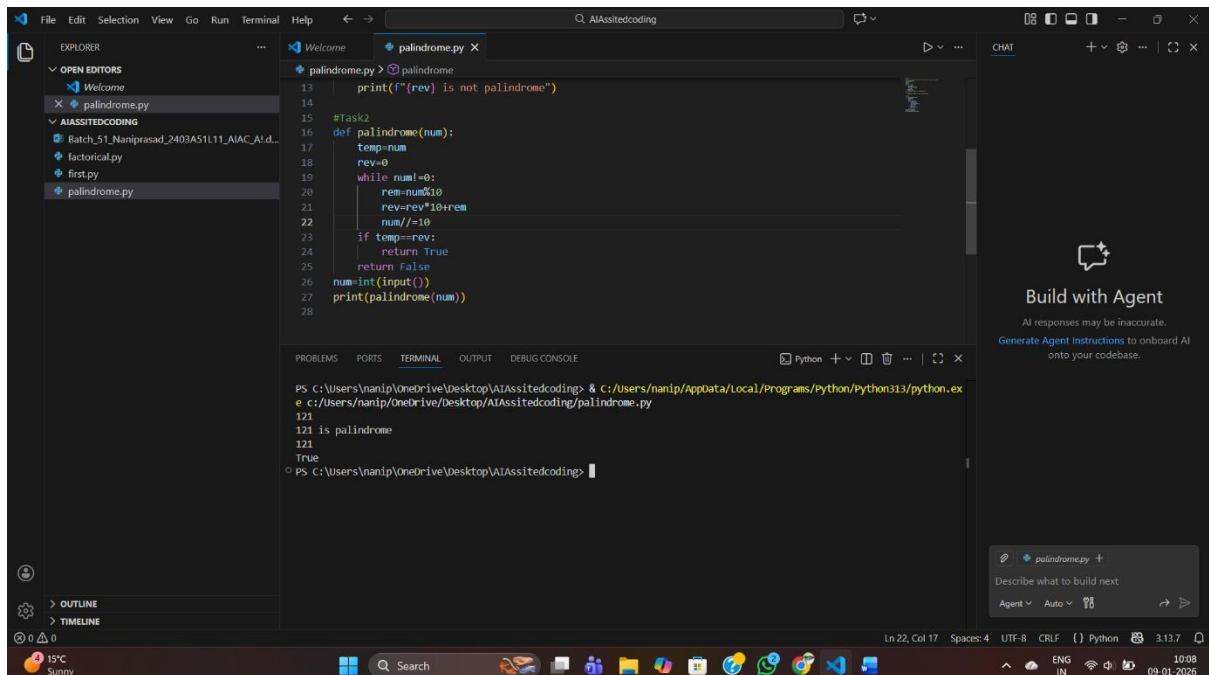


```
1 # palindrome using function
2 def is_palindrome(num):
3     temp = num
4     rev = 0
5     while num > 0:
6         dig = num % 10
7         rev = rev * 10 + dig
8         num = num // 10
9     if temp == rev:
10         return True
11     else:
12         return False
13
14 num = int(input())
15 print(is_palindrome(num))
```

Terminal output:

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

Output:



The screenshot shows a Visual Studio Code editor with a file named `palindrome.py` open. The code defines a function `palindrome(num)` that checks if a number is a palindrome. It uses a while loop to reverse the number and compares it with the original. The terminal shows the command `python c:/Users/nanip/AppData/Local/Programs/Python/Python313/python.exe c:/Users/nanip/OneDrive/Desktop/AIAssistedcoding/palindrome.py` being executed, and the output is `121 is palindrome` and `True`.

```
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         reverse=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
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:

- $\text{rem} = \text{num} \% 10 \rightarrow$ extract the last digit.
- $\text{rev} = \text{rev} * 10 + \text{rem} \rightarrow$ build the reversed number digit by digit.
- $\text{num} //= 10 \rightarrow$ 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 \rightarrow return True.
- Otherwise \rightarrow return False.

🔍 Main Program

- $\text{num} = \text{int}(\text{input}()) \rightarrow$ take user input.
- $\text{print}(\text{palindrome}(\text{num})) \rightarrow$ call the function and print the result (True or False).

Example Walkthrough

Suppose input is 121:

- $\text{temp} = 121, \text{rev} = 0$
- Loop:
 - Iteration 1: $\text{rem} = 1, \text{rev} = 1, \text{num} = 12$
 - Iteration 2: $\text{rem} = 2, \text{rev} = 12, \text{num} = 1$
 - Iteration 3: $\text{rem} = 1, \text{rev} = 121, \text{num} = 0$
- Loop ends $\rightarrow \text{rev} = 121$
- Compare: $\text{temp} == \text{rev} \rightarrow 121 == 121 \rightarrow \text{True}$
- Output: True

If input is 123:

- Reverse becomes 321
- Compare: $123 != 321 \rightarrow \text{False}$

- Output: False

#Task4:

Write Python program with using function and without using function

The screenshot shows a VS Code editor with a file named `palindrome.py`. The code implements a loop-based algorithm to check if a number is a palindrome. It takes an input `n`, stores it in `temp`, and then reverses the digits of `n` by repeatedly taking the last digit (`n%10`) and building a new number `rev`. Finally, it compares `temp` and `rev` to determine if the number is a palindrome.

```

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 screenshot shows a VS Code editor with a file named `palindrome.py`. The code defines a function `is_palindrome_stack(s)` that uses a stack to check if a string `s` is a palindrome. It pushes each character onto the stack and then pops them to compare with the original string. The main part of the code takes an input `num`, converts it to a string, and calls the `is_palindrome_stack` function.

```

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 terminal output shows the execution of the program:

```

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>

```

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 $\text{rem} = n \% 10$.
- Build reversed number: $\text{rev} = \text{rev} * 10 + \text{rem}$.
- Remove last digit: $n //= 10$.
- Repeat until n becomes 0.

4. **Compare:** If $\text{temp} == \text{rev}$, 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))
```

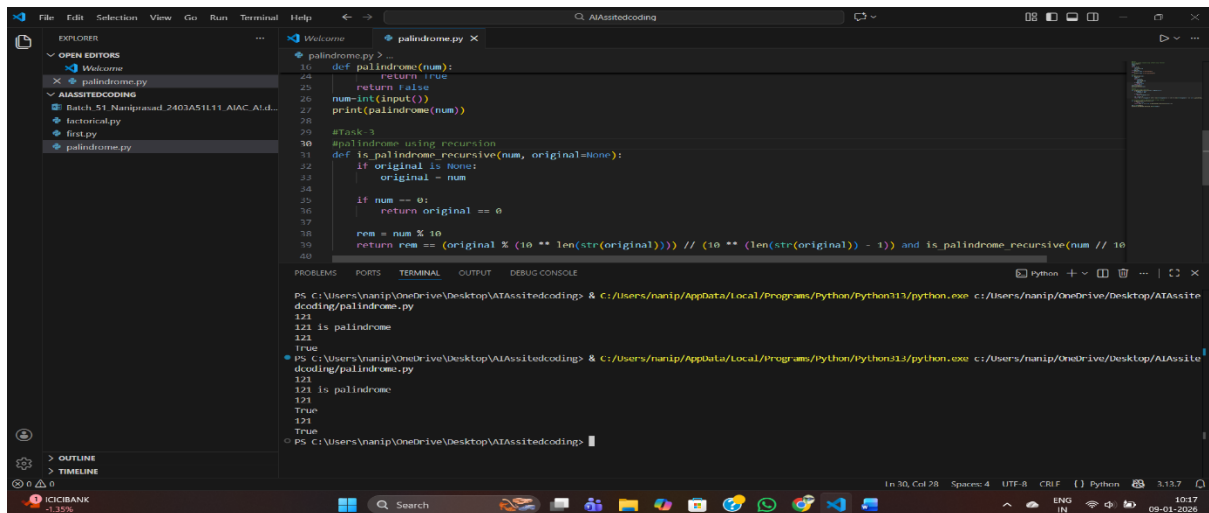
```
PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding> & C:/Users/nanip/AppData/Local/Programs/Python/Python313/python.exe c:/Users/nanip/OneDrive/Desktop/AIAssite
doding/palindrome.py
121
True
PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding> & C:/Users/nanip/AppData/Local/Programs/Python/Python313/python.exe c:/Users/nanip/OneDrive/Desktop/AIAssite
doding/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     num = num // 10
32     return num == (original % (10 ** len(str(original)))) // (10 ** (len(str(original)) - 1)) and is_palindrome_recursive(num // 10, original % 10)
33
34
35
36
37
38
39
40
```

```
PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding> & C:\Users\nanip\AppData\Local\Programs\Python\Python113\python.exe c:/Users/nanip/OneDrive/Desktop/AIAssite
doding/palindrome.py
121
121 is palindrome
121
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.