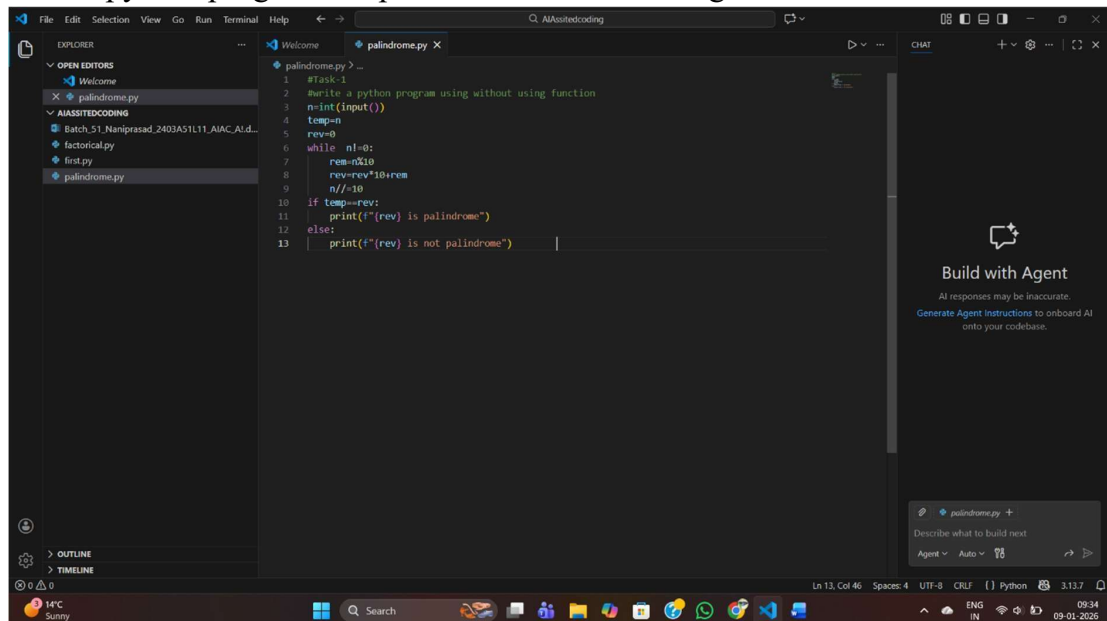


2203A51104

batch-52

#Task1

Write a python program for palindrome without using func on

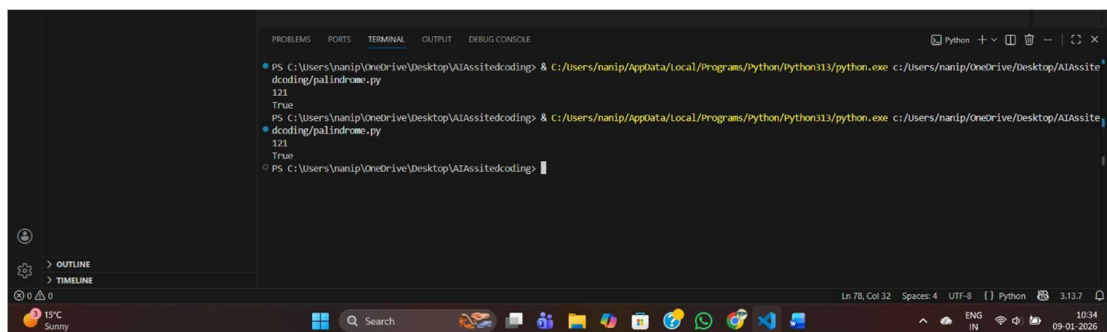


The screenshot shows a VS Code editor with a file named `palindrome.py` open. The code is as follows:

```
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 also shows a sidebar with the Explorer view, a Chat panel 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> .\palindrome.py
121
True
PS 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` script.

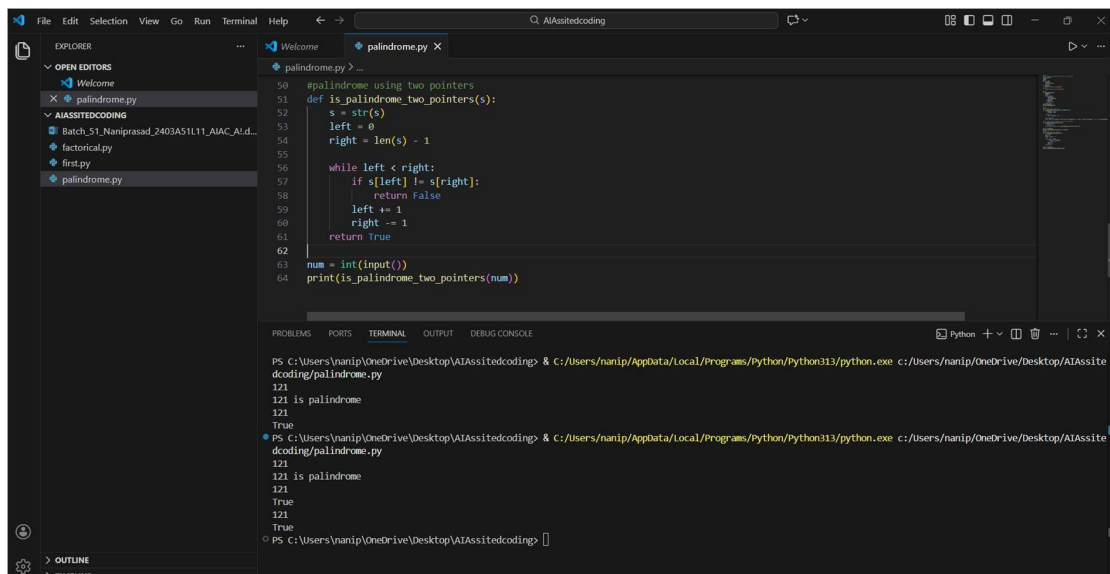
Palindrome check steps for the given code

1. Read input:
 - o Take an integer from the user and store it in n.
2. Store original number:
 - o 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:
 - $rev = rev * 10 + rem$
 - Shift 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:

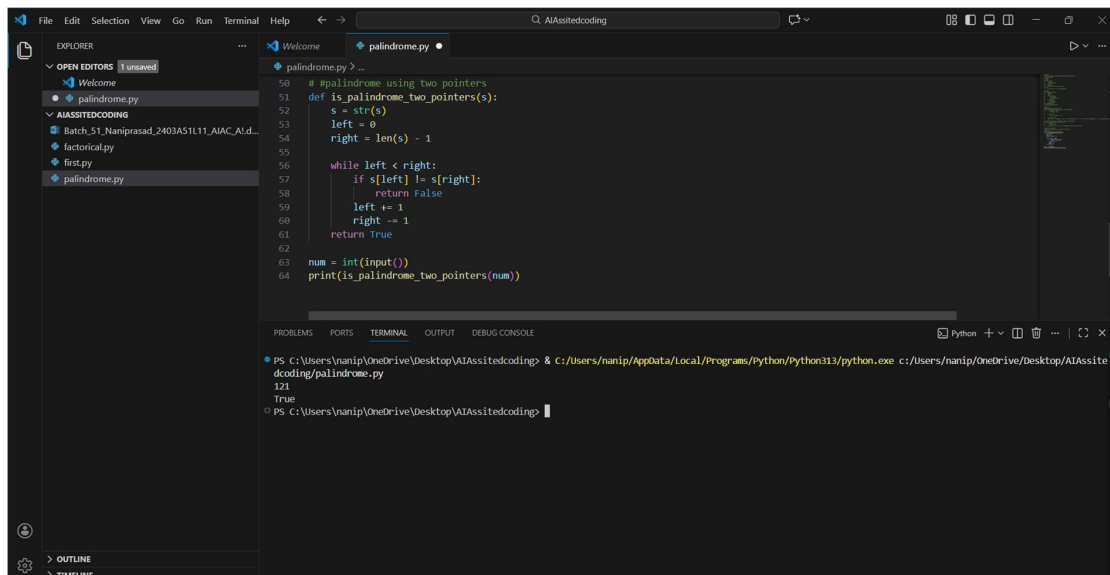
Write optimal solution for palindrome solution



```
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/AIAssite
doding/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/AIAssite
doding/palindrome.py
121
121 is palindrome
121
True
121
True
PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding>

Output:



```
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>
```

Explaina on:

Create func on

Pass the input with some value

In two pointer if last and first value are equal then

Last-=1

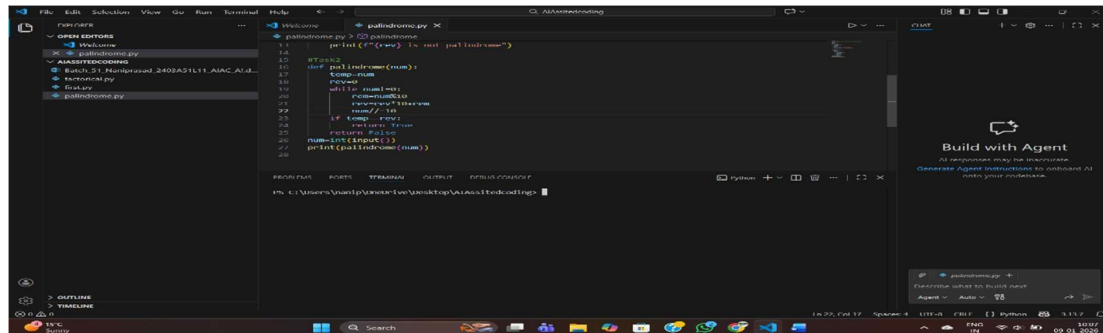
And first+=1

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

If not return False

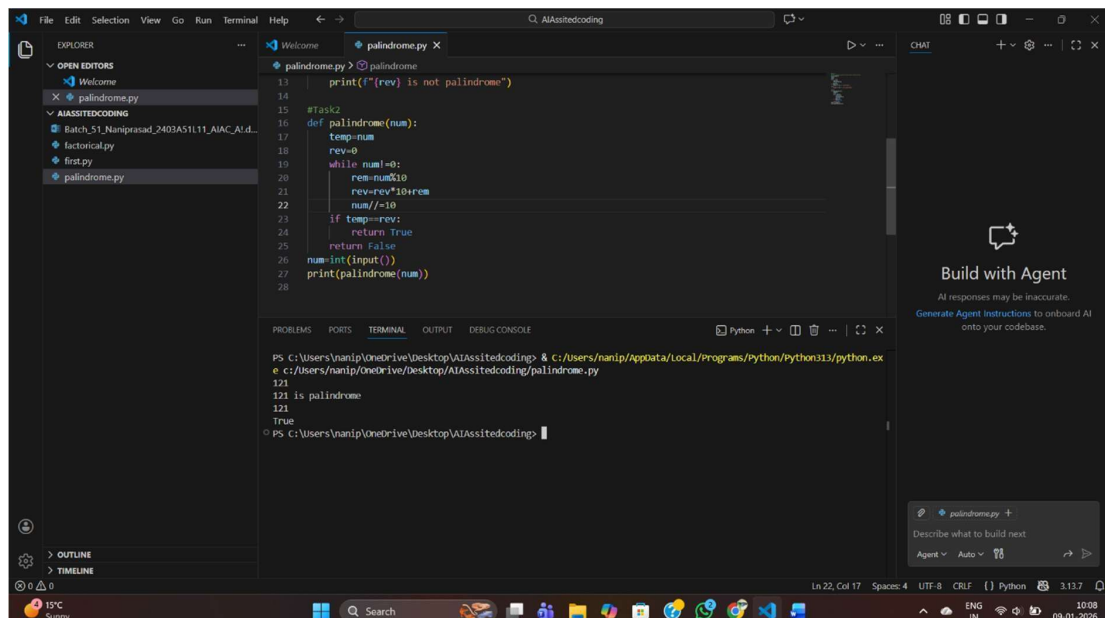
#Task 3

Write python program for palindrome using func on



```
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         res=num%10
21         rev=rev*10+res
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         res=num%10
21         rev=rev*10+res
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 is palindrome

True

Explaina on:

Step-by-Step Explana on

1. Func on Defini on o def palindrome(num):
 - o A func on 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 func on 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: $123 \neq 321 \rightarrow \text{False}$
- Output: False #Task4:

Write Python program with using func on and without using func on

The image displays two screenshots of a Visual Studio Code editor interface, showing Python programs for checking palindromes.

Top Screenshot: The editor shows a file named `palindrome.py` with the following code:

```
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")
```

Bottom Screenshot: The editor shows a file named `palindrome.py` with the following code:

```
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 screenshot also shows the terminal output:

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

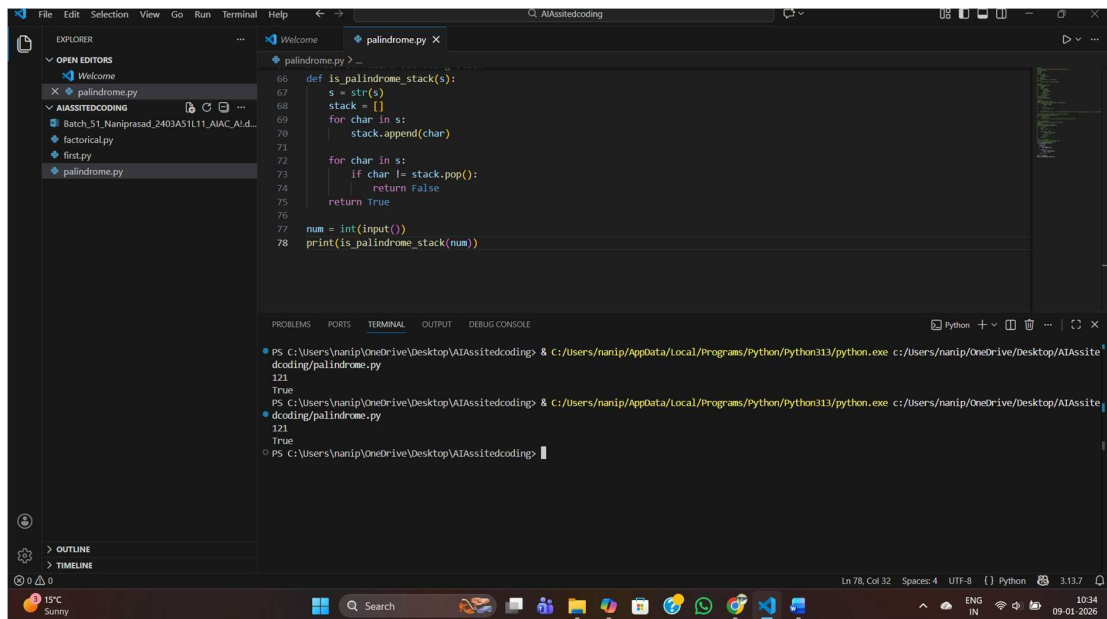
Output:

Step-by-Step

1. Input: User enters a number \rightarrow 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. Func on defined: `palindrome(num)` encapsulates the logic.
2. Inside func on:
 - 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 func on: `palindrome(num)`.
 - Print the returned result (`True` or `False`).



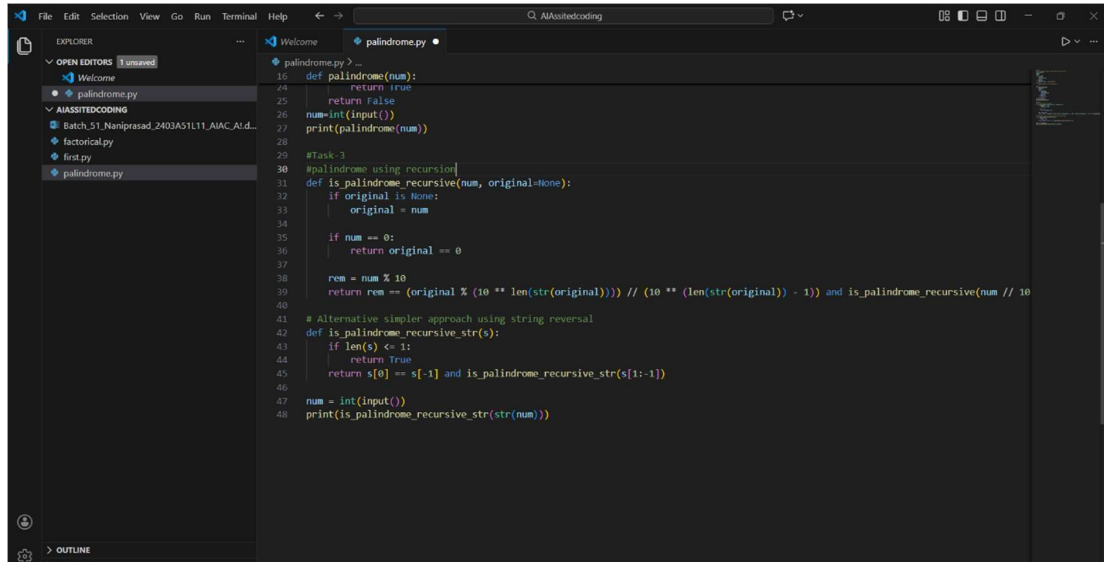
```
def is_palindrome_stack(s):
    s = str(s)
    stack = []
    for char in s:
        stack.append(char)
    for char in s:
        if char != stack.pop():
            return False
    return True

num = int(input())
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\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>
```

#Task5:

Write python program for palindrome using recursion



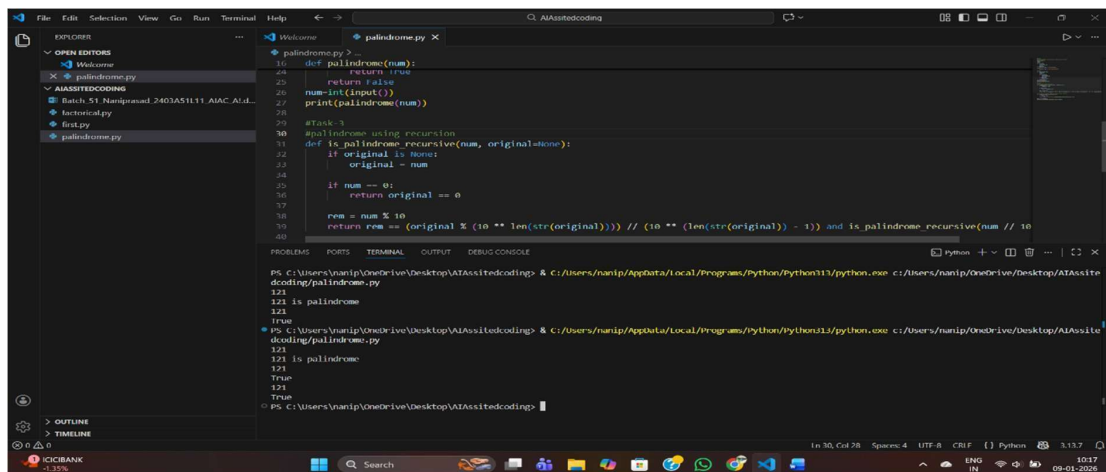
```
def palindrome(num):
    return True
    return False
num = int(input())
print(palindrome(num))

#Task-3
#palindrome using recursion
def is_palindrome_recursive(num, original=None):
    if original is None:
        original = num
    if num == 0:
        return original == 0
    rem = num % 10
    return rem == (original % (10 ** len(str(original)))) // (10 ** (len(str(original)) - 1)) and is_palindrome_recursive(num // 10)

# Alternative simpler approach using string reversal
def is_palindrome_recursive_str(s):
    if len(s) <= 1:
        return True
    return s[0] == s[-1] and is_palindrome_recursive_str(s[1:-1])

num = int(input())
print(is_palindrome_recursive_str(str(num)))
```

Output:



```
def palindrome(num):  
    return True  
    return False  
num = int(input())  
print(palindrome(num))  
  
#Task 3  
#palindrome using recursion  
def is_palindrome_recursive(num, original_str):  
    if original == num:  
        original = num  
    if num == 0:  
        return original == 0  
    rem = num % 10  
    return rem == (original % (10 ** len(str(original)))) and is_palindrome_recursive(num // 10, original_str[:-1])
```

```
PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding> python .\palindrome.py  
121  
121 is palindrome  
True  
PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding>
```

Step-by-Step Explana on

1. Convert number to string ◦ `str(num)` turns the input number into a string. ◦ Example: if user enters 121, then `s = "121"`.
2. Recursive func on logic
◦ `is_palindrome_recursive_str(s)` checks if the string `s` is a palindrome.
- 3 Execu on 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.