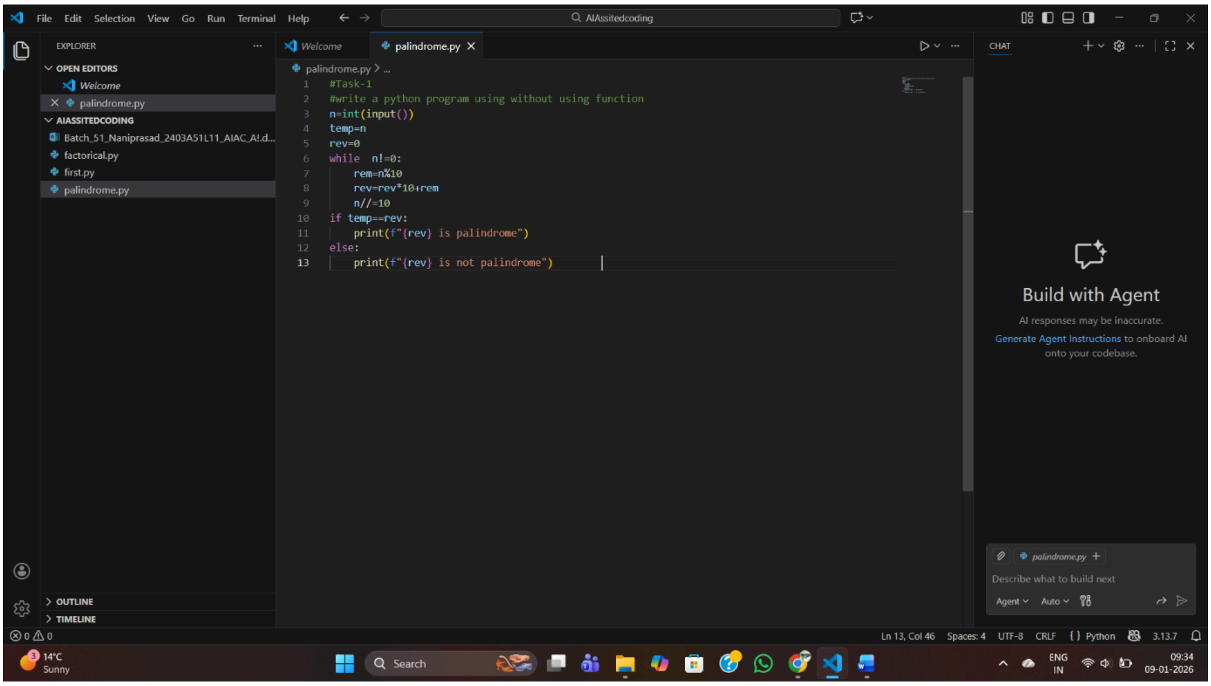
SR UNIVERSITY

2403A51L29

batch-51

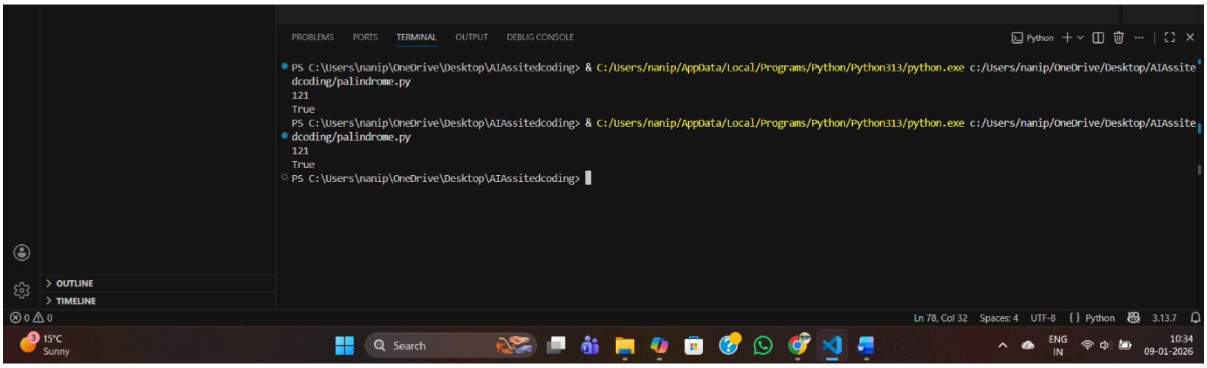
#Task1

Write a python program for palindrome without using function



```
#Task-1
#write a python program using without using function
n=int(input())
temp=n
rev=0
while  n!=0:
    rem=n%10
    rev=rev*10+rem
    n//=10
if temp==rev:
    print(f"{rev} is palindrome")
else:
    print(f"{rev} is not palindrome")
```

Output:



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

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:

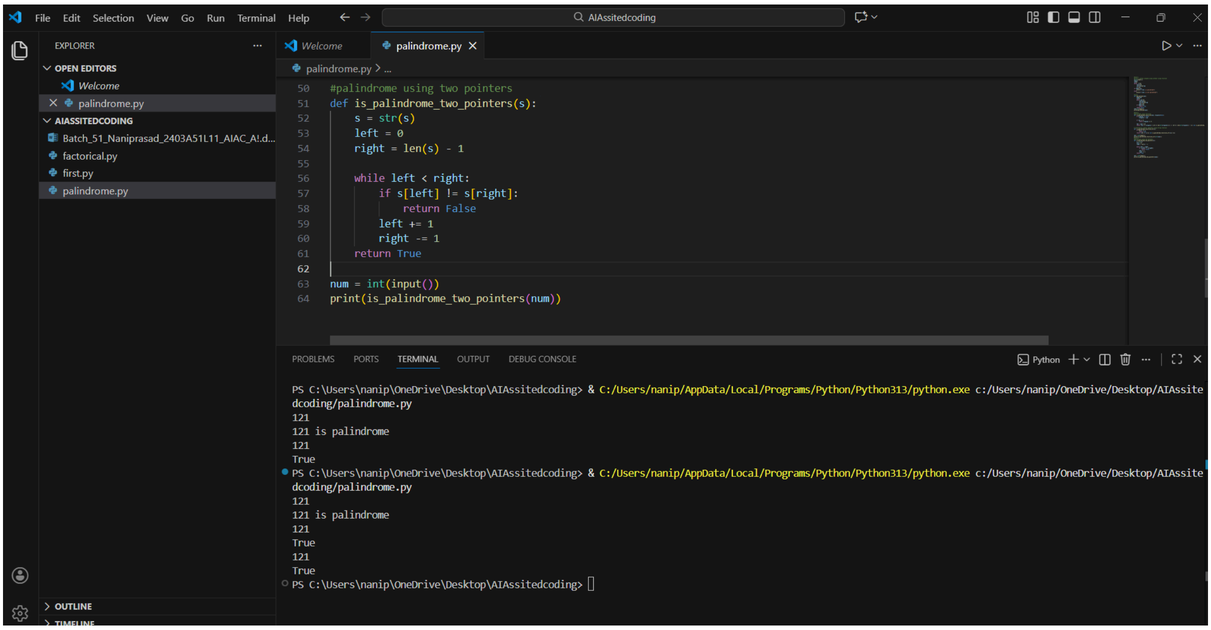
- If temp == rev, the original number reads the same backward → it's a palindrome.
- Otherwise, it's not a palindrome.

10.Output result:

- Print “rev is palindrome” if equal, else “rev is not palindrome”.

#Task2:

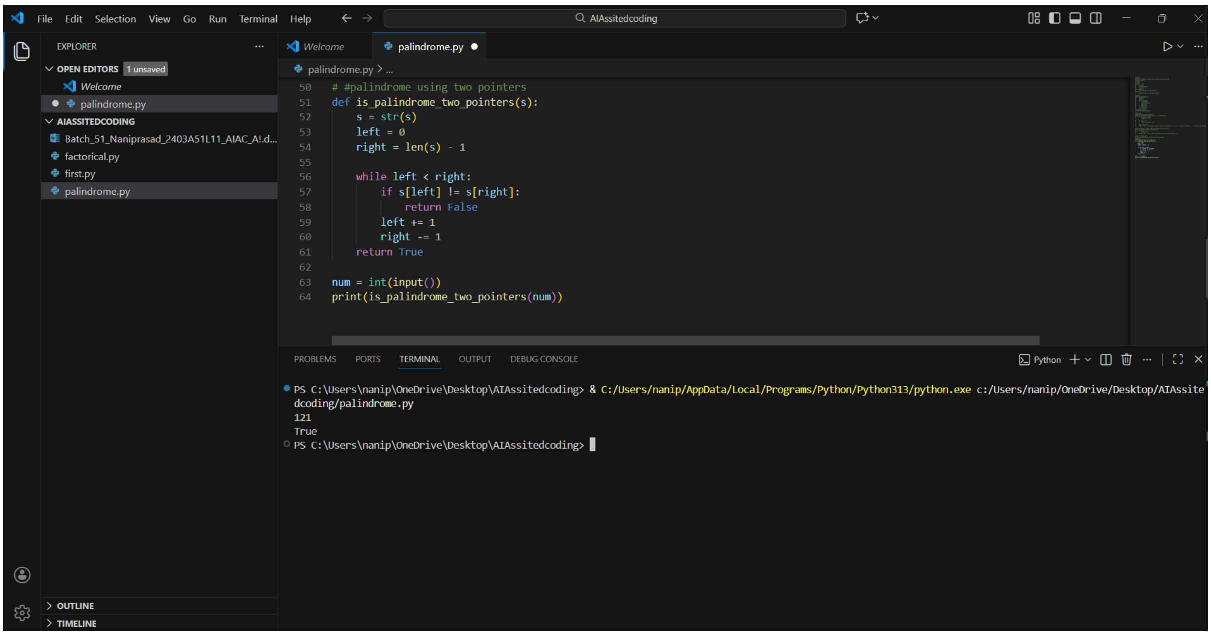
Write optimal solution for palindrome solution



```
#palindrome using two pointers
def is_palindrome_two_pointers(s):
    s = str(s)
    left = 0
    right = len(s) - 1

    while left < right:
        if s[left] != s[right]:
            return False
        left += 1
        right -= 1
    return True

num = int(input())
print(is_palindrome_two_pointers(num))
```

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

Output:



```
# #palindrome using two pointers
def is_palindrome_two_pointers(s):
    s = str(s)
    left = 0
    right = len(s) - 1

    while left < right:
        if s[left] != s[right]:
            return False
        left += 1
        right -= 1
    return True

num = int(input())
print(is_palindrome_two_pointers(num))
```

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

Explaination:

Create function

Pass the input with some value

In two pointer if last and first value are equal then

Last-=1

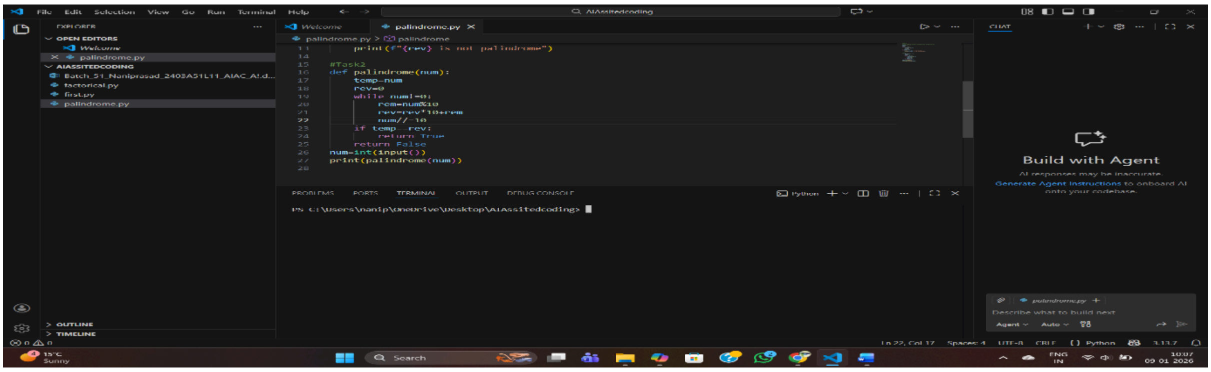
And first+=1

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

If not return False

#Task 3

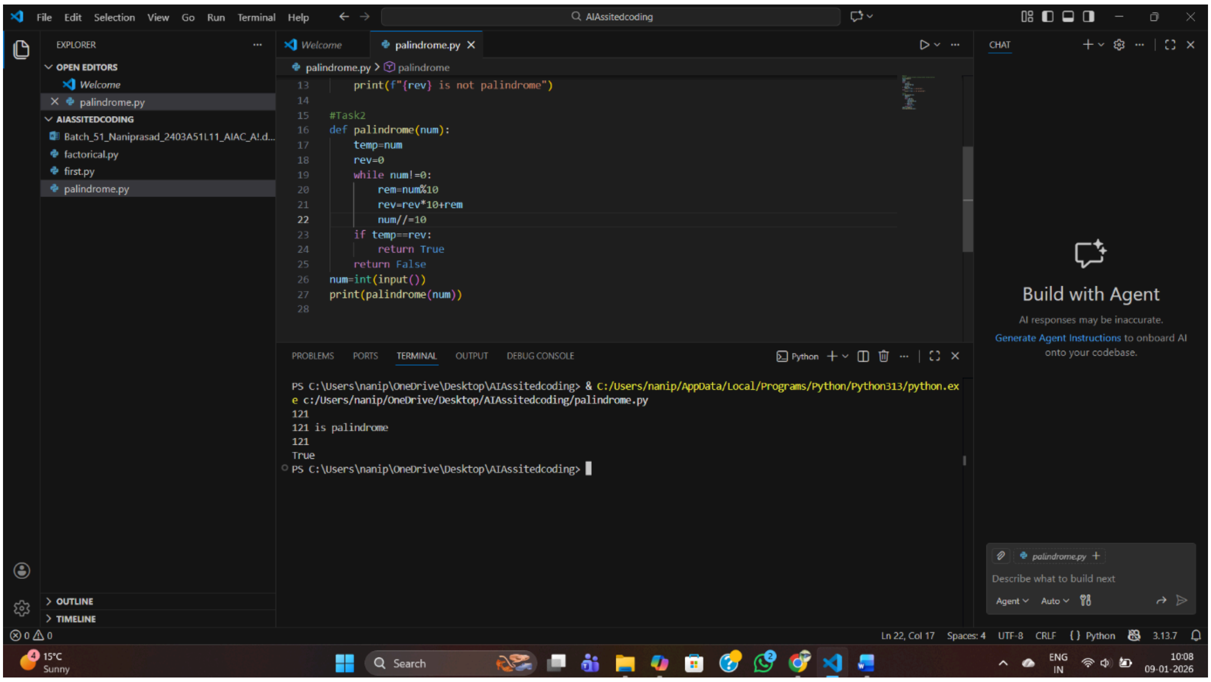
Write python program for palindrome using function



```
    print(f"{rev} is not palindrome")

#Task2
def palindrome(num):
    temp=num
    rev=0
    while num!=0:
        rem=num%10
        rev=rev*10+rem
        num//=10
    if temp==rev:
        return True
    return False
num=int(input())
print(palindrome(num))
```

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

Output:



```
    print(f"{rev} is not palindrome")

#Task2
def palindrome(num):
    temp=num
    rev=0
    while num!=0:
        rem=num%10
        rev=rev*10+rem
        num//=10
    if temp==rev:
        return True
    return False
num=int(input())
print(palindrome(num))
```

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

Explaination:

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:

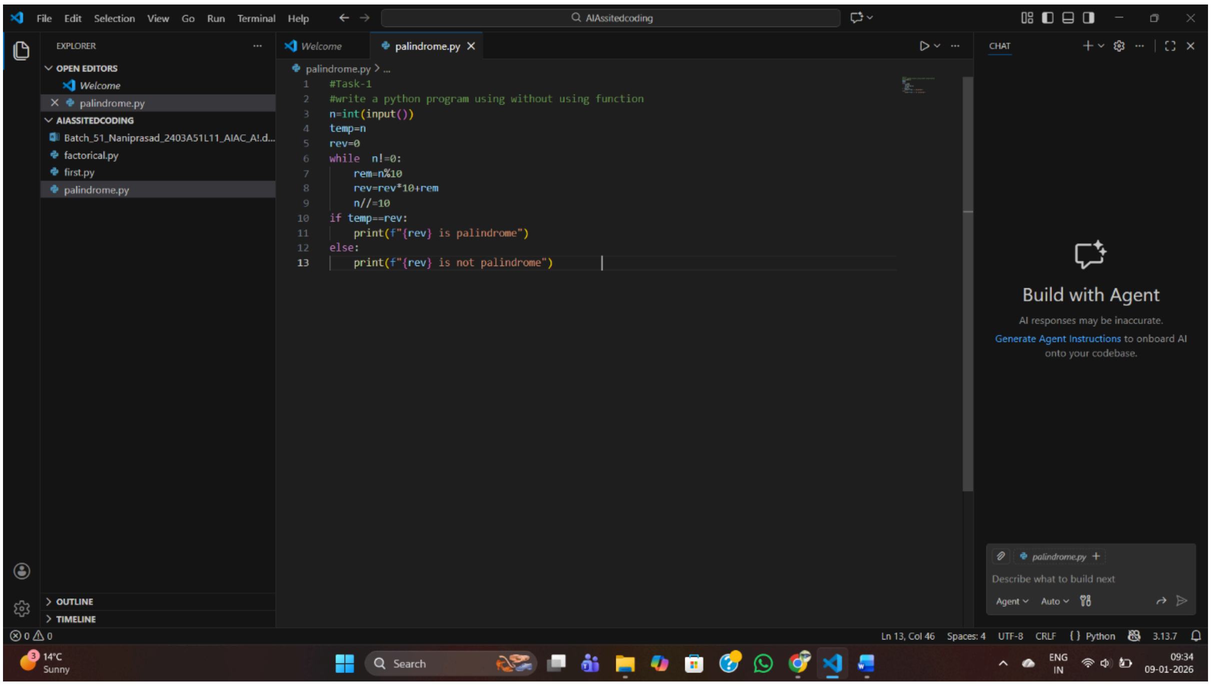
- temp = 121, rev = 0
- Loop:
  - Iteration 1: rem = 1, rev = 1, num = 12
  - Iteration 2: rem = 2, rev = 12, num = 1
  - Iteration 3: rem = 1, rev = 121, num = 0
- Loop ends → rev = 121
- Compare: temp == rev → 121 == 121 → True
- Output: True

If input is 123:

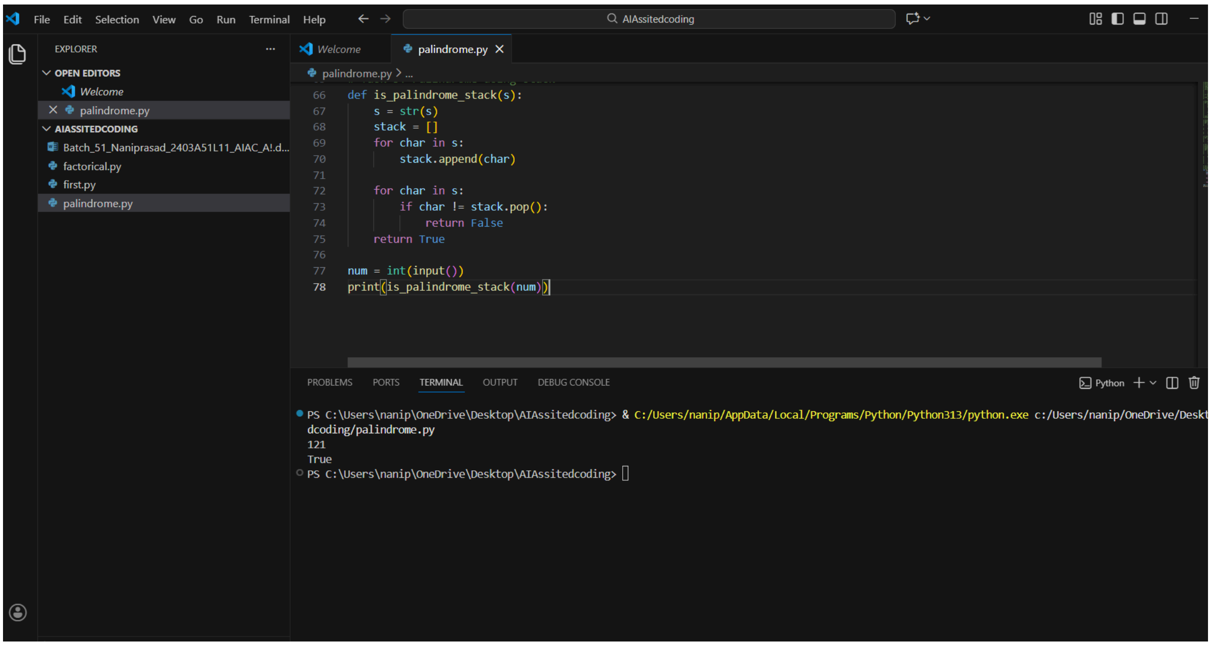
- Reverse becomes 321
- Compare: 123 != 321 → False
- Output: False

#Task4:

Write Python program with using function and without using function



```
#Task-1
#write a python program using without using function
n=int(input())
temp=n
rev=0
while  n!=0:
    rem=n%10
    rev=rev*10+rem
    n//=10
if temp==rev:
    print(f"{rev} is palindrome")
else:
    print(f"{rev} is not palindrome")
```



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

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.

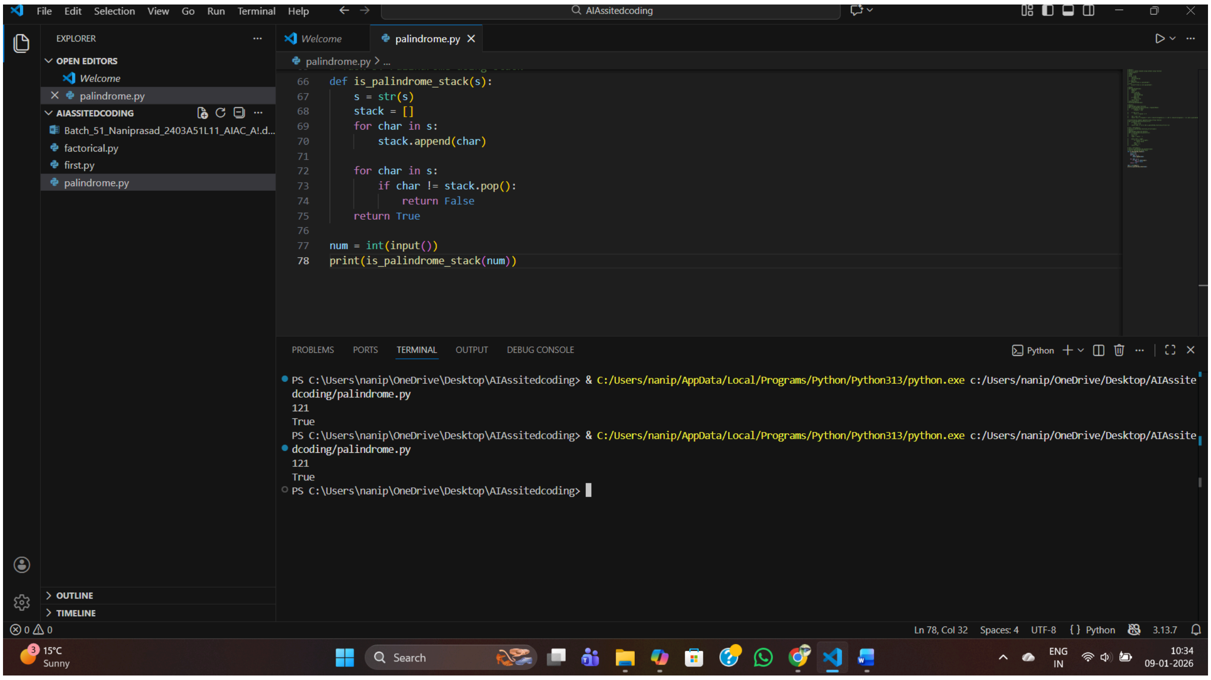
- Build reversed number: rev = rev * 10 + rem.
- Remove last digit: n //= 10.
- Repeat until n becomes 0.

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).

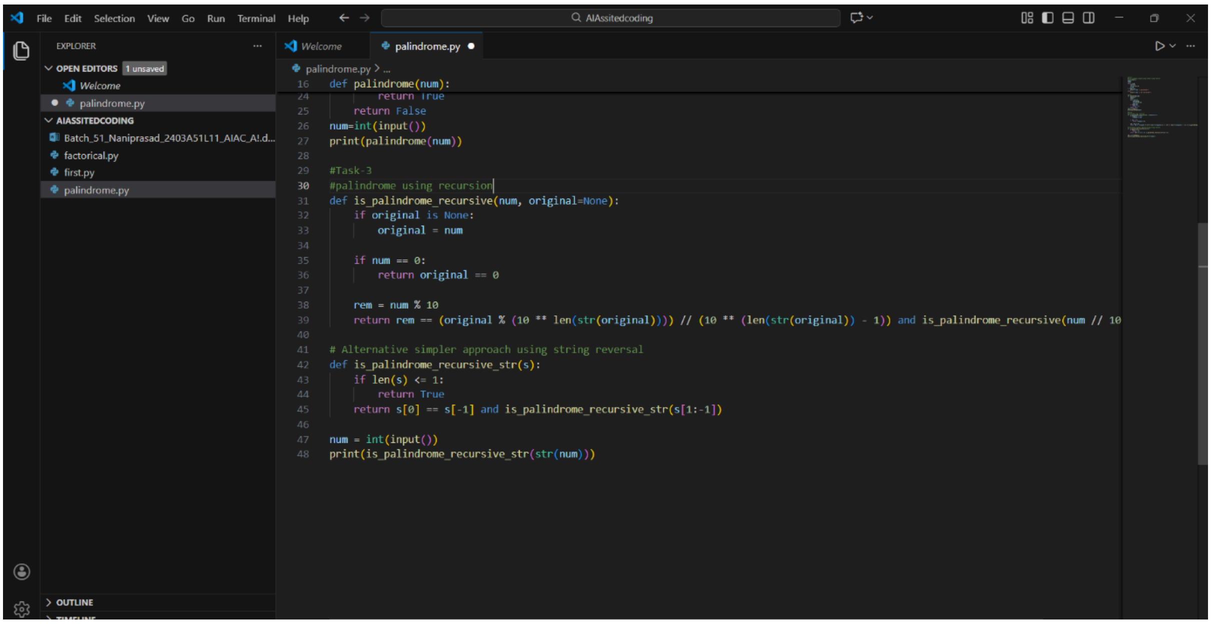


```
def is_palindrome_stack(s):
    s = str(s)
    stack = []
    for char in s:
        stack.append(char)

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

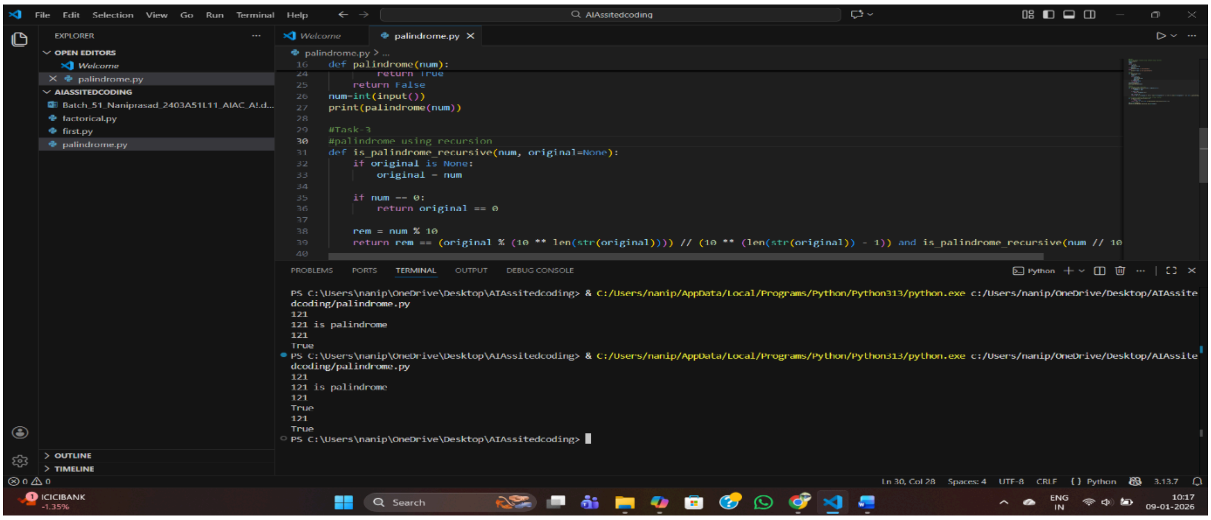
#Task5:

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=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
```

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

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.