

2403A51L18 batch-51

#Task1

Write a python program for palindrome without using function

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. In the top right corner, there is an AI assistance feature labeled "Build with Agent". Below it, a message says "AI responses may be inaccurate. Generate Agent Instructions to onboard AI onto your codebase." The main area displays a Python file named "palindrome.py" with the following code:

```
#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")
```

The status bar at the bottom indicates the code is in Python mode, has 3.13.7 syntax, and the current date is 09-01-2026.

Output:

The screenshot shows the terminal tab in VS Code. It displays the command 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 being run. The output shows two runs of the program, both returning "True" as the output. The status bar at the bottom indicates the code is in Python mode, has 3.13.7 syntax, and the current date is 09-01-2026.

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 $\text{rev} = 0$. 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:
- $\text{rem} = n \% 10$
 - This gives the rightmost digit of n .
6. Append digit to reversed number:
- $\text{rev} = \text{rev} * 10 + \text{rem}$
 - 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 $\text{temp} == \text{rev}$, 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))
```

PROBLEMS PORTS TERMINAL OUTPUT DEBUG CONSOLE

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

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))
```

PROBLEMS PORTS TERMINAL OUTPUT DEBUG CONSOLE

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> []

Explanation:

Create function

Pass the input with some value

In two pointer if last and first value are equal then

Last-=1

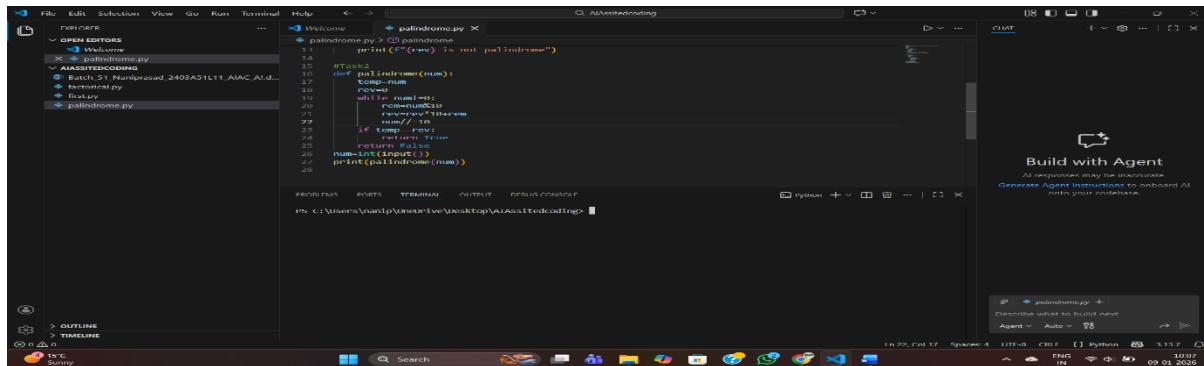
And first+=1

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

If not return False

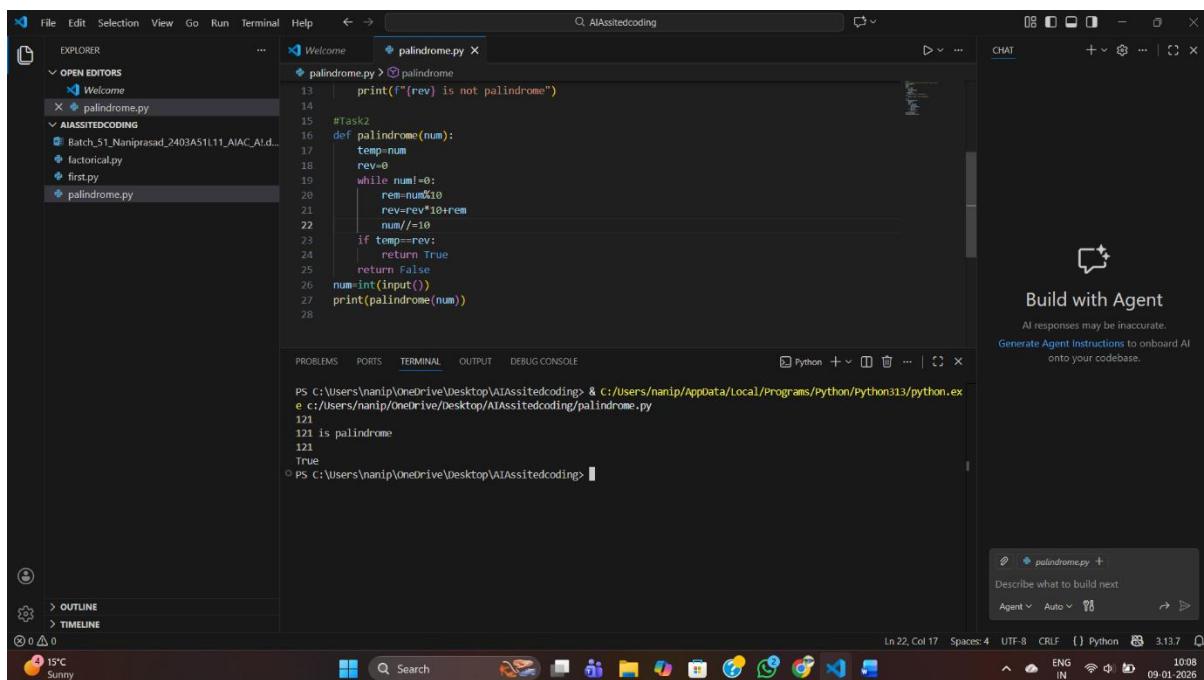
#Task 3

Write python program for palindrome using function



```
File Edit Selection View Go Run Terminal Help <- > C:\AIAssistedCoding
Welcome + palindrome.py x
palindrome.py > palindrome
print(f"{num} 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\AIAssistedCoding>
```

Output:



```
File Edit Selection View Go Run Terminal Help <- > C:\AIAssistedCoding
EXPLORER + palindrome.py x
Welcome + palindrome
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\AIAssistedCoding> & c:/Users/nanip/AppData/Local/Programs/Python/Python313/python.exe c:/Users/nanip/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

- o 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 \rightarrow rev = 121
 - Compare: temp == rev \rightarrow 121 == 121 \rightarrow 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 function and without using function

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and a search bar with the text "Q AIAssistedCoding". The left sidebar has sections for EXPLORER, OPEN EDITORS (with "Welcome" and "palindrome.py" listed), and AIASSISTEDCODING (with "Batch_51_Naniprasad_2403A51L11_AIAC_Ald..."). The main editor area displays a Python 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")
```

The status bar at the bottom shows "Ln 13, Col 46" and "Spaces: 4". The bottom right corner shows system icons for battery (14°C), network (Sunny), and date/time (09:34 09-01-2025). A floating "CHAT" window on the right side contains the text "Build with Agent" and "AI responses may be inaccurate. Generate Agent Instructions to onboard AI onto your codebase." A small icon for "palindrome.py" is also visible in the bottom right.

The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows several files: Welcome, palindrome.py, AIASSITEDCODING, Batch_51_Naniprasad_2403A51L11_AIAC_Ald..., factorial.py, first.py, and palindrome.py.
- Editor Area:** Displays the Python code for a palindrome checker using a stack. The code defines a function `is_palindrome_stack` that takes a string `s`, converts it to a list of characters, and then iterates through the characters, pushing them onto a stack. It then iterates through the characters again, popping from the stack each time. If at any point the character does not match the top of the stack, it returns `False`. If all characters match, it returns `True`. Finally, it reads an integer input from the user and prints the result of the function call.
- Terminal:** Shows the command line output of running the script. It shows the path `C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding`, the command `& C:/Users/nanip/AppData/Local/Programs/Python/Python313/python.exe c:/Users/nanip/OneDrive/Desktop/AIAssistedcoding/palindrome.py`, the execution time `121`, and the output `True`.

Output:

Step-by-Step

- Input:** User enters a number → stored in n.
- Save original:** temp = n keeps the original number safe.
- 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.
- Compare:** If temp == rev, the number is palindrome.
- Output:** Prints directly whether palindrome or not.

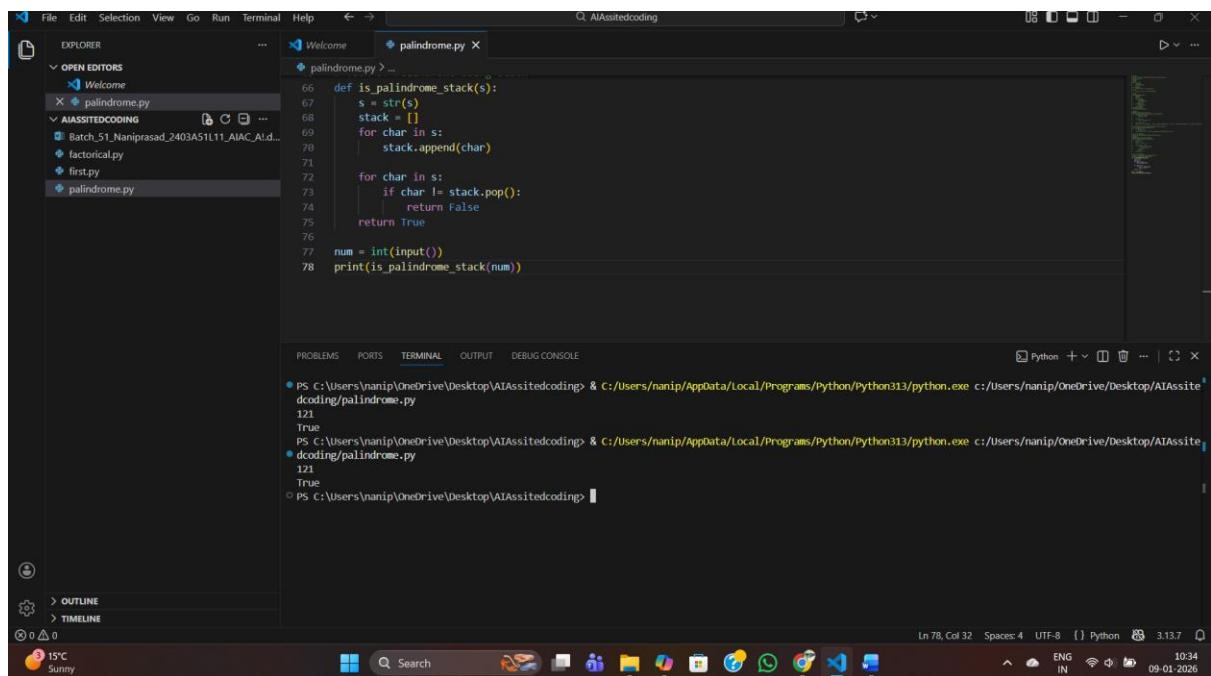
Step-by-Step

- Function defined:** `palindrome(num)` encapsulates the logic.
- Inside function:**
 - Store original number in temp.
 - Reverse the number using same loop logic.

- o Compare temp with rev.
- o 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).



The screenshot shows a Visual Studio Code (VS Code) interface. The left sidebar displays a file tree with several Python files: Welcome, palindrome.py, AIAssistedCoding, Batch_51_Naniprasad_2403A51L11_AIAC_A1d..., factorial.py, first.py, and palindrome.py. The main editor area contains 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 terminal at the bottom shows the execution of the script:

```

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 status bar at the bottom right indicates the file is saved (S), the current line is 78, column 32, and the file size is 3.137 KB. It also shows the date and time as 09-01-2026.

#Task5:

Write python program for palindrome using recursion

The screenshot shows a Visual Studio Code (VS Code) interface. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and a back/forward navigation icon. The left sidebar displays the Explorer, showing open editors: Welcome (1 unsaved), palindrome.py, and AIAssistedCoding (Batch_51_Naniprasad_2403A51L11_AIAC_A1d.., factorial.py, first.py, and palindrome.py). The main editor area contains the following Python code:

```
File Edit Selection View Go Run Terminal Help < > Q AIAssistedCoding Explorer OPEN EDITORS 1 unsaved Welcome palindrome.py AIASSISTEDCODING Batch_51_Naniprasad_2403A51L11_AIAC_A1d.. factorial.py first.py palindrome.py

palindrome.py >...
1 def palindrome(num):
2     if num < 0:
3         return False
4     num=int(input())
5     print(palindrome(num))
6
7 #Task-3
8 #Palindrome using recursion
9 def is_palindrome_recursive(num, original=None):
10     if original is None:
11         original = num
12
13     if num == 0:
14         return original == 0
15
16     rem = num % 10
17     return rem == (original % (10 ** len(str(original)))) // (10 ** (len(str(original)) - 1)) and is_palindrome_recursive(num // 10)
18
19 # Alternative simpler approach using string reversal
20 def is_palindrome_recursive_str(s):
21     if len(s) <= 1:
22         return True
23     return s[0] == s[-1] and is_palindrome_recursive_str(s[1:-1])
24
25 num = int(input())
26 print(is_palindrome_recursive(str(num)))
```

Output:

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The title bar reads "AIAssitedcoding". The left sidebar has sections for "EXPLORER", "OPEN EDITORS", and "AIIASSTEDCODING". The "OPEN EDITORS" section lists "Welcome", "palindrome.py", and "first.py". The "AIIASSTEDCODING" section lists "Batch 51 Naniprasad 240A51111 AIAC AI...". The main editor area displays a Python script named "palindrome.py". The code defines a function `isPalindrome` that checks if a number is a palindrome using recursion. It includes a docstring and several test cases at the bottom. Below the editor are tabs for "PROBLEMS", "PORTS", "TERMINAL", "OUTPUT", and "DEBUG CONSOLE". The terminal tab shows command-line interactions with Python scripts. The status bar at the bottom indicates "In 30, Col 28" and "Spacebar 4" for keyboard input, along with icons for file operations, search, and system status.

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) \rightarrow equal \rightarrow recurse on "2".
- Step 2: "2" has length 1 \rightarrow base case \rightarrow return True.
- Final result: True.