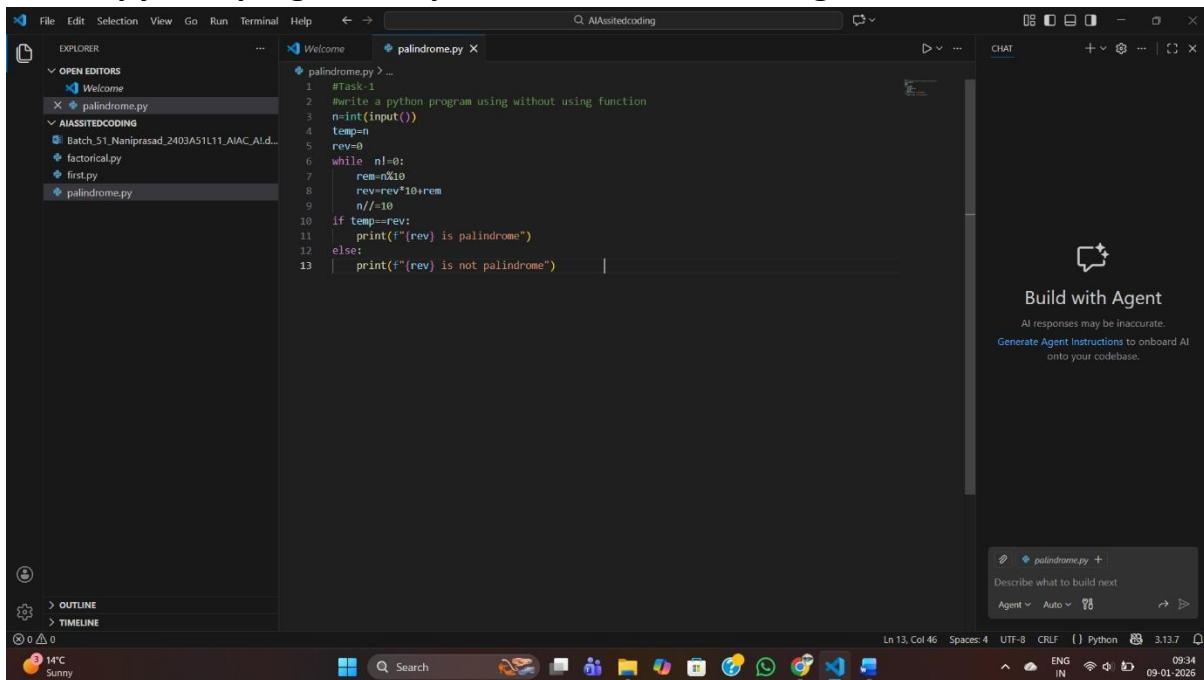


2403A51L37 batch-52

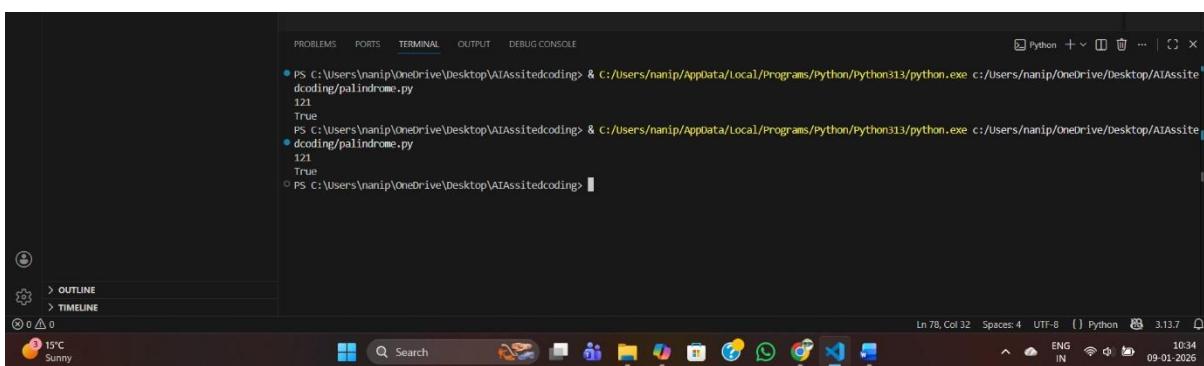
#Task1

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:
 - 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:
- $\text{rem} = \text{n} \% 10$ ○ 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:
- $\text{n} //\text{= } 10$ ○ 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

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows a folder structure under "OPEN EDITORS" containing "Welcome", "palindrome.py", and "AIASSITEDCODING" which includes "Batch_51_Naniprasad_2403A51L11_AIAC_A1d..." and other files.
- Editor:** Displays the code for "palindrome.py".

```
# 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))
```
- Terminal:** Shows the output of running the script.

```
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:

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows a folder structure under "OPEN EDITORS" containing "Welcome", "palindrome.py" (marked as unsaved), and "AIASSITEDCODING" which includes "Batch_51_Naniprasad_2403A51L11_AIAC_A1d..." and other files.
- Editor:** Displays the code for "palindrome.py".
- Terminal:** Shows the output of running the script.

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

The screenshot shows a dark-themed instance of Visual Studio Code. The left sidebar displays a file tree with several open files under 'OPEN EDITORS'. One file, 'palindrome.py', is currently active and shows the following code:

```
1  # Welcome to palindrome.py
2
3  print("rev is not palindrome")
4
5  # Task
6  def isPalindrome(num):
7      temp_num = num
8      rev = 0
9      while num > 0:
10          rem = num % 10
11          num = num // 10
12          rev = rev * 10 + rem
13
14      if temp_num == rev:
15          return True
16      else:
17          return False
18
19  number = 12321
20
21  print(isPalindrome(number))
```

The bottom status bar indicates the file path as 'C:\Users\yashp\OneDrive\Desktop\VAI\AssistedCoding>' and the current line as '16, 22, Col 17'. A floating 'CHAT' window on the right side contains the message 'Build with Agent'.

Output:

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 "AIAssistedcoding". The left sidebar has sections for EXPLORER, OPEN EDITORS, and AIASSISTEDCODING, with files like "palindrome.py" and "factorial.py" listed. The main editor area displays a Python script named "palindrome.py" with the following code:

```
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))
```

The terminal below shows the execution of the script:

```
PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding> & c:/users/nanip/appdata/local/programs/python/python333/python.exe c:/users/nanip/OneDrive/Desktop/AIAssistedcoding/palindrome.py
121
121 is palindrome
121
True
○ PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding>
```

A right-hand sidebar titled "CHAT" contains a message from "Build with Agent": "Build with Agent" and "All responses may be inaccurate. Generate Agent Instructions to onboard AI onto your codebase." A small icon for the AI agent is also present.

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 ○ $\text{rev} = 0$ ○ This variable will hold the reversed number.
4. Loop to Reverse Number ○ while $\text{num} \neq 0$: → keep looping until num becomes 0. ○ Inside the loop: ○ $\text{rem} = \text{num} \% 10$ → extract the last digit. ○ $\text{rev} = \text{rev} * 10 + \text{rem}$ → build the reversed number digit by digit.
 - $\text{num} // 10$ → 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.

2 Main Program

- $\text{num} = \text{int}(\text{input}())$ → take user input.
- $\text{print}(\text{palindrome}(\text{num}))$ → 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 → $\text{rev} = 121$
 - Compare: $\text{temp} == \text{rev} \rightarrow 121 == 121 \rightarrow \text{True}$
 - Output: True

If input is 123:

- Reverse becomes 321
- Compare: 123 != 321 → False
- Output: False #Task4:

Write Python program with using function and without using function

The screenshot shows two instances of the Visual Studio Code (VS Code) interface. Both instances have the title bar "File Edit Selection View Go Run Terminal Help" and the status bar at the bottom showing "AIAssistedCoding".

Top Instance (Non-Functional Approach):

- Explorer:** Shows files like "Welcome", "palindrome.py", "Batch_51_Naniprasad_2403A51L11_AIAC_AI.d...", "factorial.py", "first.py", and "palindrome.py".
- Editor:** Displays the following Python 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")

```
- Terminal:** Shows the command "python palindrome.py" being run.

Bottom Instance (Functional Approach):

- Explorer:** Shows the same files as the top instance.
- Editor:** Displays the following Python 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))

```
- Terminal:** Shows the command "python palindrome.py" being run.

Output:

Step-by-Step

1. **Input:** User enters a number → stored in n.

2. **Save original:** $\text{temp} = \text{n}$ keeps the original number safe.

3. **Reverse logic:**

- Extract last digit using $\text{rem} = \text{n} \% 10$.
- Build reversed number: $\text{rev} = \text{rev} * 10 + \text{rem}$. ○ Remove last digit: $\text{n} // 10$. ○ 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).

The screenshot shows a VS Code interface with the following details:

- File Explorer:** Shows files in the "OPEN EDITORS" section: "Welcome", "palindrome.py", "Batch_S1_Naniprasad_2403A51L11.AIAC.At.d", "factorial.py", "first.py", and "palindrome.py".
- Editor:** Displays the code for "palindrome.py".

```
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))
```
- Terminal:** Shows command-line output for running 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>
```
- Bottom Status Bar:** Shows the current file path as "C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding", line 78, column 32, spaces: 4, encoding: UTF-8, Python 3.13.7, and the date/time as 09-01-2026.

#Task5:

Write python program for palindrome using recursion

The screenshot shows a VS Code interface with the following details:

- File Explorer:** Shows files in the "OPEN EDITORS" section: "Welcome", "palindrome.py", "Batch_S1_Naniprasad_2403A51L11.AIAC.At.d", "factorial.py", "first.py", and "palindrome.py".
- Editor:** Displays the code for "palindrome.py".

```
def palindrome(num):  
    if num == 0:  
        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_reversive(s):  
    if len(s) <= 1:  
        return True  
    return s[0] == s[-1] and is_palindrome_reversive(s[1:-1])  
  
num = int(input())  
print(is_palindrome_reversive(str(num)))
```
- Bottom Status Bar:** Shows the current file path as "C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding", line 48, column 1, spaces: 4, encoding: UTF-8, Python 3.13.7, and the date/time as 09-01-2026.

Output:

```

File Edit Selection View Go Run Terminal Help <- > 🔍 AIAssistedCoding
OPEN EDITORS ... Welcome palindrome.py ✘
OPEN EDITORS ... Welcome palindrome.py ✘
AIASSISTEDCODING palindrome.py
Batch 51 Naniprasad_2403A51L11_AIAC_Ald...
factorial.py
first.py
palindrome.py
PROBLEMS PORTS TERMINAL OUTPUT DEBUG CONSOLE
PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding> & C:/Users/nanip/AppData/Local/Programs/Python/Python311/python.exe c:/Users/nanip/OneDrive/Desktop/AIAssistedcoding/palindrome.py
121
121 is palindrome
121
True
True
True
True
PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding>

```

In 30, Col 28 Spaces: 4 UTF-8 CRLF () Python 3.13.7 ENG IN 10:17 09-01-2026

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(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.