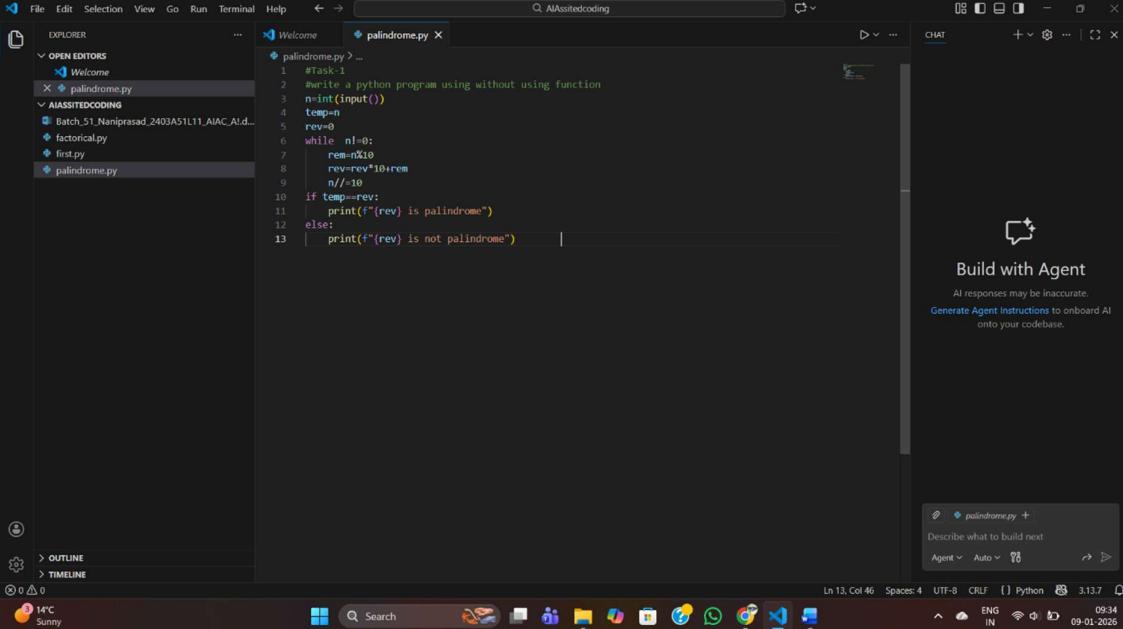


# 2403A51L35

## batch-52

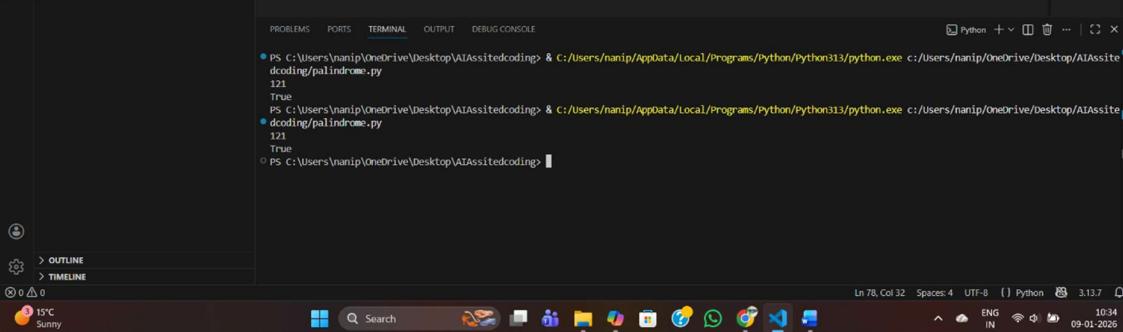
### #Task1

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\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:
  - o 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:
- $\text{rem} = \text{n} \% 10$
  - 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:
- $\text{n} //= 10$
  - 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))
```

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  
121  
True  
121  
True

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

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 the PyCharm IDE interface. The left sidebar displays project files: 'pycharm', 'EDITOR', 'ASSISTEDCODING', 'Batch 51 - Nanodegree\_24032011\_AIAAC', 'firuy', and 'palindrome.py'. The main editor window contains the following Python code:

```
1  #!/usr/bin/python
2  print("Is %s not a palindrome?" % input())
3
4  #check if number is a palindrome
5  def palindrome(num):
6      temp = num
7      rev = 0
8      while num > 0:
9          rem = num % 10
10         rev = (rev * 10) + rem
11         num = num // 10
12
13     if temp == rev:
14         return True
15     else:
16         return False
17
18 num = int(input())
19 print(palindrome(num))
```

The status bar at the bottom shows the path 'C:\Users\manip\PycharmProjects\Desktop\AssistedCoding' and the terminal output 'Is 123 not a palindrome?'. A floating window titled 'Build with Agent' is visible on the right.

## Output:

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 search bar for "AIAssistedcoding". The left sidebar has sections for Explorer, OPEN EDITORS, AIASSISTEDCODING, and OUTLINE/TIMELINE. The main area shows an "Welcome" file and an open editor for "palindrome.py". The code in the editor is as follows:

```
1.3 |     print(f"(rev) is not palindrome")
1.4 |
1.5 | #task2
1.6 def palindrome(num):
1.7     temp_num
1.8     rev=0
1.9     while num!=0:
2.0         rem=num%10
2.1         rev=rev*10+rem
2.2         num/=10
2.3     if temp==rev:
2.4         return True
2.5     return False
2.6 num=int(input())
2.7 print(palindrome(num))
```

Below the editor are tabs for PROBLEMS, PORTS, TERMINAL, OUTPUT, and DEBUG CONSOLE. The TERMINAL tab shows command-line 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
121 is palindrome
121
121
True
○ PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding
```

A floating window titled "Build with Agent" is visible on the right, with the message "AI responses may be inaccurate. Generate Agent Instructions to onboard AI onto your codebase." At the bottom, there's a status bar with icons for battery, temperature (15°C), weather (Sunny), and system information (Windows 10, ENG IN, 10:08, 09-01-2026). The bottom right corner shows the Python extension version 3.13.7.

## 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:
  - 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

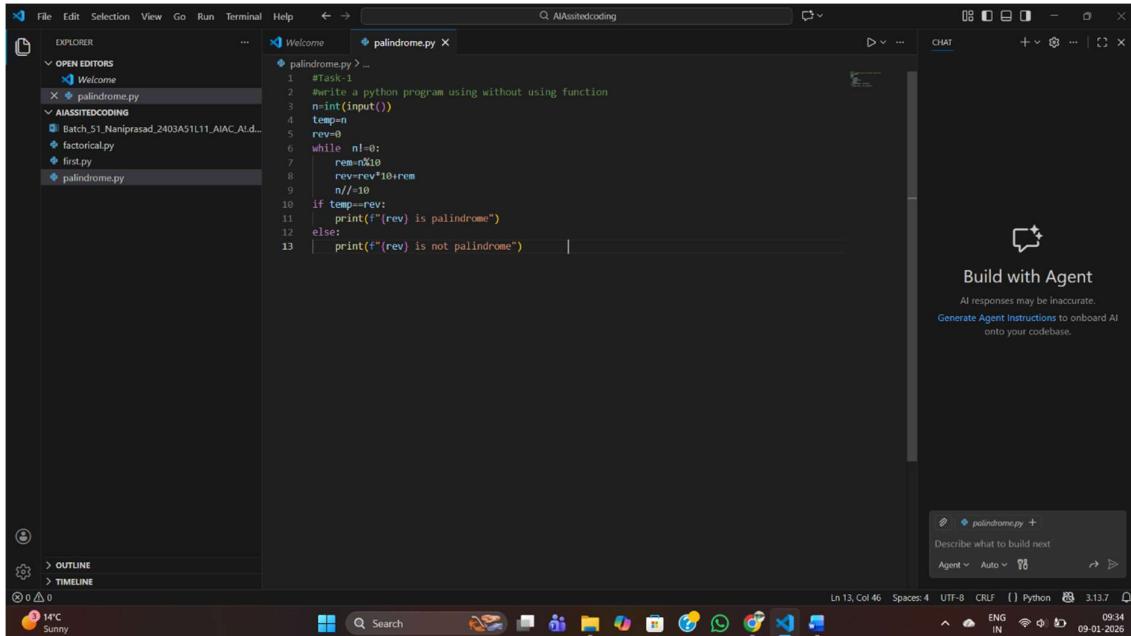
- Loop:
  - Iteration 1: rem = 1, rev = 1, num = 12
  - Iteration 2: rem = 2, rev = 12, num = 1
  - Iteration 3: rem = 1, rev = 121, num = 0
- Loop ends → rev = 121
- Compare: temp == rev → 121 == 121 → True
- Output: True

If input is 123:

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

#Task4:

Write Python program with using function and without using function



```

File Edit Selection View Go Run Terminal Help <- > Q AIAssistedCoding
EXPLORER OPEN EDITORS ...
Welcome palindrome.py ...
AIASSISTEDCODING
Batch_51_Haniprasad_2403A51L11_AiAC_A.d...
factorial.py
first.py
palindrome.py
Welcome palindrome.py ...
#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")
Ln 13, Col 46 Spaces: 4 UTF-8 CRLF () Python 3.13.7
14°C Sunny
Search

```

The screenshot shows a dark-themed instance of Visual Studio Code. In the Explorer sidebar, there are several projects and files listed under 'OPEN EDITORS' and 'AIASSITEDCODING'. The main editor area contains a Python script 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))
```

Below the editor, the terminal tab is active, showing the command-line output of running the script:

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

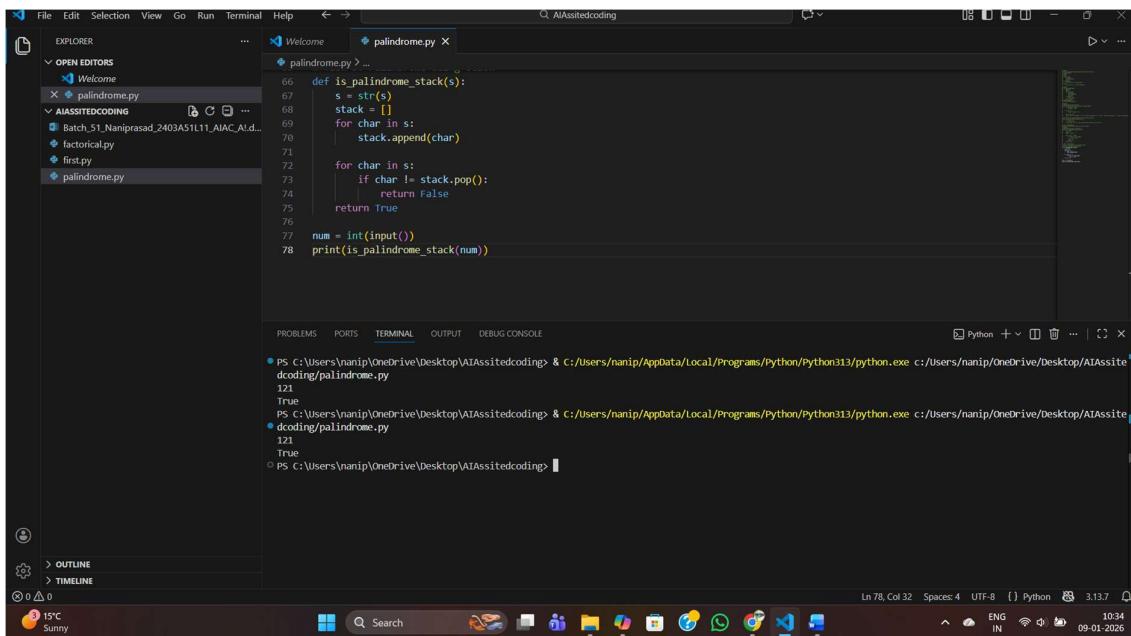
### Step-by-Step

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).



The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows multiple files including "palindrome.py", "Welcome", "factorial.py", and "first.py".
- Editor:** Displays the code for "palindrome.py" which defines a function `is\_palindrome\_stack` that checks if a string is a palindrome using a stack.
- Terminal:** Shows the command-line output of running the script, indicating it correctly identifies "121" as a palindrome.
- Bottom Status Bar:** Provides information like file path (C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding\palindrome.py), line/col (Ln 78, Col 32), and date/time (09-01-2026).

```

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))

```

## #Task5:

Write python program for palindrome using recursion



The screenshot shows a Visual Studio Code (VS Code) interface with the following details:

- File Explorer:** Shows multiple open files: "Welcome", "palindrome.py", "Batch\_51\_Naniprasad\_2403A51L11\_AIAC\_Ald...", "factorial.py", "first.py", and "palindrome.py".
- Editor:** The main editor window displays Python code for palindrome detection. The code uses two methods: a recursive approach and a string reversal approach.
- Search Bar:** At the top center, it says "AAAssistedcoding".
- Bottom Status Bar:** Shows "OUTLINE" and "PALETTE".

```
16 def palindrome(num):
17     if num < 0:
18         return False
19     num=int(input())
20     print(palindrome(num))
21
22
23 #Task -3
24 #palindrome using recursion
25 def is_palindrome_recursive(num, original=None):
26     if original is None:
27         original = num
28
29     if num == 0:
30         return original == 0
31
32     rem = num % 10
33     return rem == (original % (10 ** len(str(original)))) // (10 ** (len(str(original)) - 1)) and is_palindrome_recursive(num // 10, original)
34
35 # Alternative simpler approach using string reversal
36 def is_palindrome_recursive_str(s):
37     if len(s) <= 1:
38         return True
39     return s[0] == s[-1] and is_palindrome_recursive_str(s[1:-1])
40
41 num = int(input())
42 print(is_palindrome_recursive(str(num)))
```

## Output:

## 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.