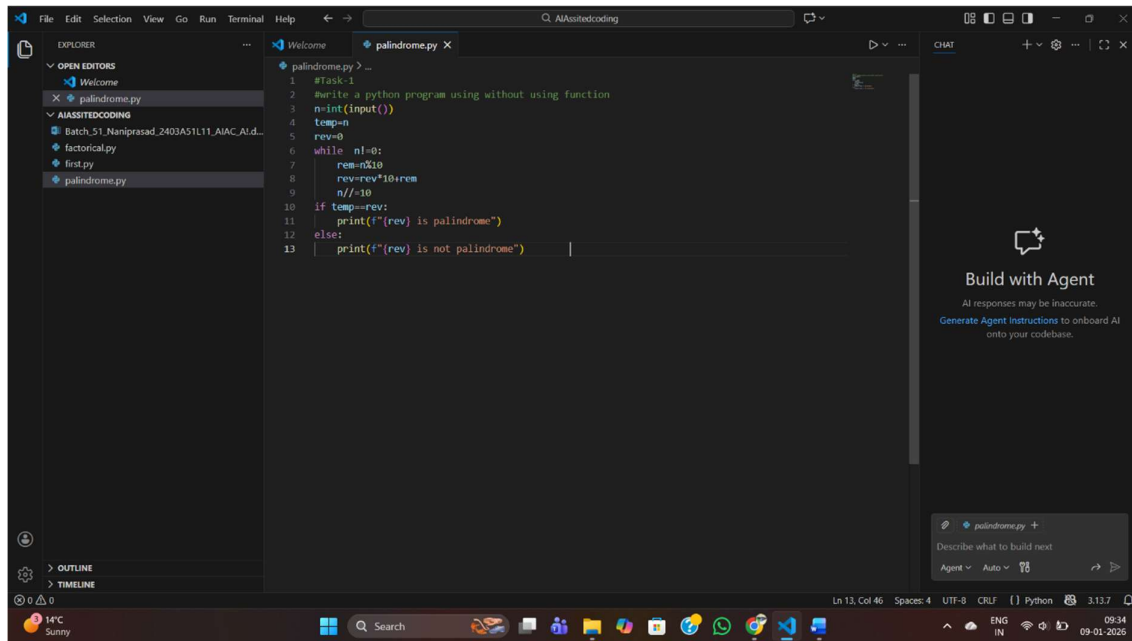


# 2403A51L35

## batch-52

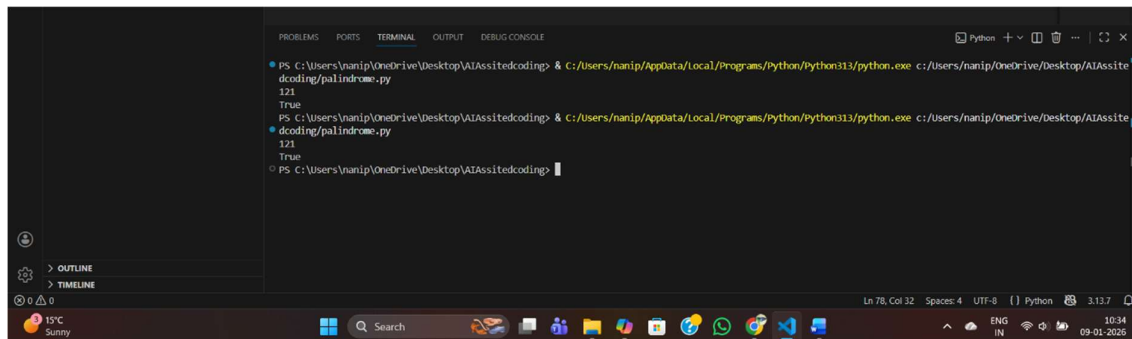
### #Task1

Write a python program for palindrome without using function



```
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")
```

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:
  - 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

```
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))
```

Terminal output:

```
PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding> & 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\AIAssistedcoding> & 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\AIAssistedcoding>
```

Output:

```
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))
```

Terminal output:

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

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

If not return False

### #Task 3

The screenshot displays a Windows desktop environment. On the left, a file explorer shows a project named 'palindrome.py' with a file named 'palindrome.py' selected. The main window is a code editor showing the following Python code:

```

1 # palindrome.py
2 def is_palindrome(s):
3     s = s.lower().replace(" ", "")
4     return s == s[::-1]
5
6 if __name__ == "__main__":
7     s = input("Enter a string: ")
8     if is_palindrome(s):
9         print("It is a palindrome")
10    else:
11        print("It is not a palindrome")

```

Below the code editor, a terminal window is open, showing the command prompt. The prompt is 'C:\Users\Amit>' and the command entered is 'python palindrome.py'. The output of the command is 'Enter a string: ' followed by a cursor. The terminal also shows the command 'python -m pip install --upgrade pip' and the output 'Requirement already satisfied: pip in c:\users\amit\appdata\local\programs\python\python311\python.exe (23.1.2)'. The terminal window is titled 'Python - C:\Users\Amit\appdata\local\programs\python\python311\python.exe'.

The image shows a Windows desktop with a dark-themed Visual Studio Code (VS Code) editor. The editor has three main panes. The left pane is the Explorer, showing a file tree with 'palindrome.py' selected. The middle pane is the Editor, displaying the code for 'palindrome.py'. The code defines a function 'palindrome(num)' that checks if a number is a palindrome by reversing its digits and comparing it to the original. The right pane is the Chat interface, titled 'CHAT', with a large 'Build with Agent' button and a text input field. Below the chat pane is a terminal window showing the execution of the script. The terminal output indicates that the number 121 is a palindrome. The Windows taskbar at the bottom shows the system clock as 10:08 on 09-01-2024, and various application icons are visible.

**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

The screenshot shows a Visual Studio Code editor window with a file named 'palindrome.py'. The code implements a loop to reverse a number and then compares it to the original number. The interface includes a sidebar with file explorer, a main editor area, and a chat panel on the right.

```

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 chat panel on the right displays a 'Build with Agent' message, indicating that AI-generated code is being used. The status bar at the bottom shows the current line and column (Ln 13, Col 46) and the Python version (3.11.7).

```
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 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>
```

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.

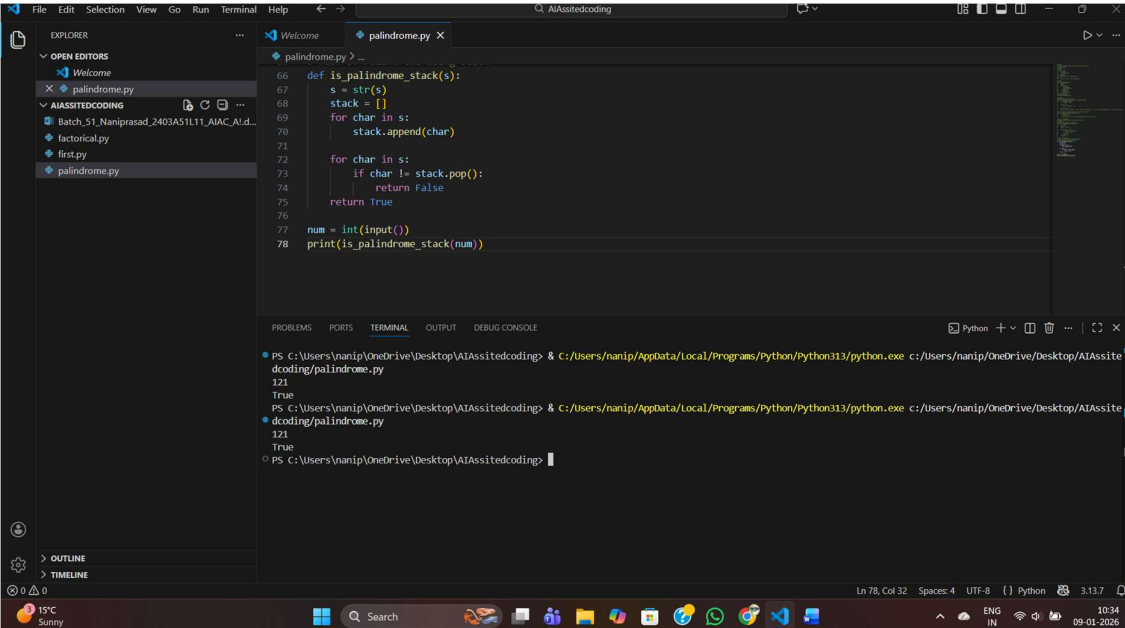
## Step-by-Step

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).



```
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))
```

PROBLEMS PORTS TERMINAL OUTPUT DEBUG CONSOLE

```
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>
```

Ln 78, Col 32 Spaces: 4 UTF-8 Python 3.13.7

## #Task5:

Write python program for palindrome using recursion



```

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

```

## Output:

```

PS C:\Users\nanip\OneDrive\Desktop\VAAssitedcoding> .\palindrome.py
121
121 is palindrome
121
True
PS C:\Users\nanip\OneDrive\Desktop\VAAssitedcoding>

```

## 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.