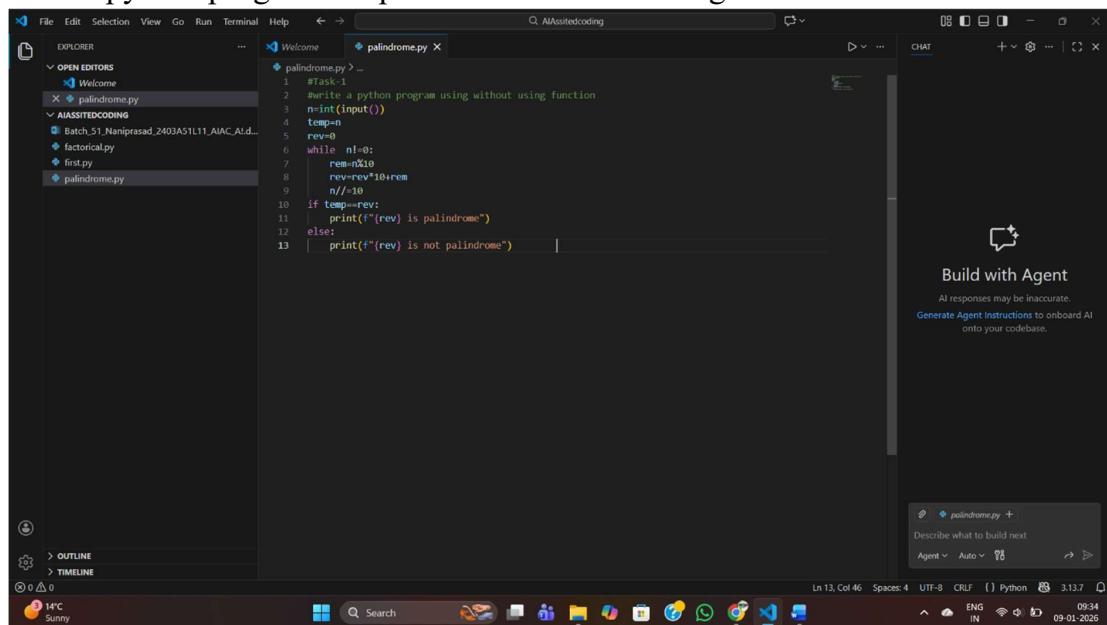


2203A51104

batch-52

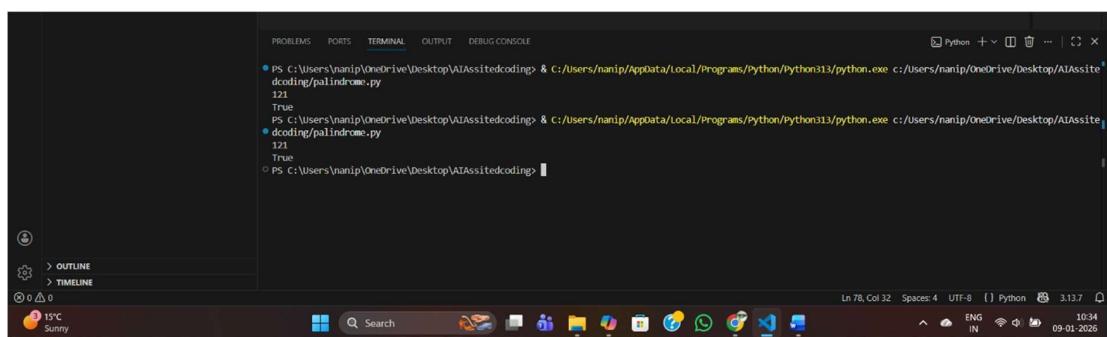
#Task1

Write a python program for palindrome without using func on



```
#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:
 - o Copy n into temp so you can compare later after reversing.

3. Initialize reverse:

- o Set rev = 0. This will be built digit by digit into the reversed number.

4. Loop until n becomes 0:

- o Keep extracting the last digit and removing it from n using integer division.

5. Extract last digit:

- o rem = n % 10 o This gives the rightmost digit of n.

6. Append digit to reversed number:

- o rev = rev * 10 + rem
- o Shifts existing digits in rev left and adds the new last digit.

7. Remove last digit from n:

- o n //= 10 o Drops the rightmost digit from n to process the next one.

8. End of loop:

- o When n becomes 0, rev now holds the full reversed number.

9. Compare original with reversed:

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

10. Output result:

- o 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
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
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))
```

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>

Explaina on:

Create func on

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 top navigation bar includes File, Edit, Subversion, View, Go, Run, Terminal, Help, and AIAssistedCoding. On the left, the Project tool window displays a file tree with 'Welcome' as the root, containing 'palindrome.py'. The code editor shows the following Python script:

```
1. #!/usr/bin/python
2. # This program checks if a number is a palindrome
3. # Author: [REDACTED]
4. # Date: [REDACTED]
5.
6. def palindrome(num):
7.     temp = num
8.     rev = 0
9.     while num > 0:
10.         rem = num % 10
11.         rev = (rev * 10) + rem
12.         num = num // 10
13.     if temp == rev:
14.         return True
15.     else:
16.         return False
17.
18. print(palindrome(12321))
```

The bottom status bar shows the path 'C:\Users\manipal\source\repos\AIAssistedCoding>', the current file 'palindrome.py', and the message '1 file changed, 1 insertion(+), 0 deletions(-)'. The bottom right corner shows the date '09/01/2025'.

Output:

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The top navigation 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, and AIAssistedCoding, with "palindrome.py" selected. The main editor area displays the following Python 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/Python313/python.exe c:/Users/nanip/OneDrive/Desktop/AIAssistedCoding/palindrome.py
121
121 is palindrome
121
True
PS C:\Users\nanip\OneDrive\Desktop\AIAssistedCoding>
```

A floating "Build with Agent" card is visible on the right, with the message "Build with Agent" and "All responses may be inaccurate. Generate Agent Instructions to onboard AI onto your codebase." A status bar at the bottom shows "Ln 22, Col 17 Spaces: 4 UTF-8 CRLF () Python" and system icons.

Explanation:

Step-by-Step Explanation

1. Func on Defini on o def palindrome(num):
 - o A func on named palindrome is created that takes one argument num.
 2. Store Original Number

- o temp = num o The original number is stored in temp so we can compare later.
- 3. Initialize Reverse
 - o rev = 0 o This variable will hold the reversed number.
- 4. Loop to Reverse Number o while num != 0: → keep looping until num becomes 0.
 - o Inside the loop: o rem = num % 10 → extract the last digit.
 - o rev = rev * 10 + rem → build the reversed number digit by digit.
 - o num //= 10 → remove the last digit from num.
- 5. Check Palindrome o After the loop ends, rev contains the reversed number. o Compare temp (original number) with rev.
 - o If they are equal → return True.
 - o 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:
 - o Iteration 1: rem = 1, rev = 1, num = 12 o Iteration 2: rem = 2, rev = 12, num = 1
 - o 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: $123 \neq 321 \rightarrow \text{False}$
- Output: False #Task4:

Write Python program with using func on and without using func on

The screenshot displays two instances of the Visual Studio Code (VS Code) interface, each showing a different implementation of a palindrome checker.

Top Editor (Function-Based Approach):

```
1 task_1
2     #write a python program using without using function
3     num=int(input())
4     temp=0
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")
```

Bottom Editor (Stack-Based Approach):

```
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))
```

Both editors show the same output in the terminal:

```
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
```

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:
 - o Extract last digit using rem = n % 10.
 - o Build reversed number: rev = rev * 10 + rem.
 - o Remove last digit: n // 10.
 - o 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:
 - o Store original number in temp.
 - o 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 Windows desktop environment with the Visual Studio Code (VS Code) application open. The code editor displays a Python file named `palindrome.py`. The code uses a stack-based approach to reverse a string and compare it with the original. The terminal tab shows the execution of the script, which correctly identifies the input number as a 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\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>
```

#Task5:

Write python program for palindrome using recursion

The screenshot shows a Windows desktop environment with the Visual Studio Code (VS Code) application open. The code editor displays a Python file named `palindrome.py`. The code implements a recursive function to check if a number is a palindrome. It handles both integer and string inputs and includes a check for alternative approaches using string reversal.

```
def palindrome(num):
    if num == 0:
        return True
    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, original)

# 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(num)))
```

Output:

```

File Edit Selection View Go Run Terminal Help ← → 🔍 AIAssistedCoding
OPEN EDITORS Welcome
AIASSISTEDCODING
Batch 51 Naniprasad_2403A51L11_AIAc_AJL
factorial.py
first.py
palindrome.py
palindrome.py
Welcome
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)) - 1)) and is_palindrome_recursive(num // 10, original)
18
19
20 #Palindrome using recursion
21 def is_palindrome():
22     num = int(input("Enter a number: "))
23
24     if num < 0:
25         print("Number must be positive")
26     else:
27         if is_palindrome_recursive(num):
28             print("The number is a palindrome")
29         else:
30             print("The number is not a palindrome")
31
32
33
34
35
36
37
38
39
40

```

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 is palindrome

True

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 is palindrome

121 True

121 True

PS C:\Users\Nanip\OneDrive\Desktop\AIAssistedcoding>

ICIBANK 0:23%

Search

PROBLEMS PORTS TERMINAL OUTPUT DEBUG CONSOLE

PS C:\Users\Nanip\OneDrive\Desktop\AIAssistedcoding>

130 Col 28 Spaces: 4 UTF-8 CRLF ENG IN 10:17 09-01-2025

Step-by-Step Explanation

- Convert number to string o str(num) turns the input number into a string. o Example: if user enters 121, then s = "121".
- Recursive function logic
 - o is_palindrome_recursive(s) checks if the string s is a palindrome.
- Execution Example: Input = 121
 - o s = "121" o Step 1: Compare "1" (first) and "1" (last) → equal → recurse on "2".
 - o Step 2: "2" has length 1 → base case → return True.
 - o Final result: True.