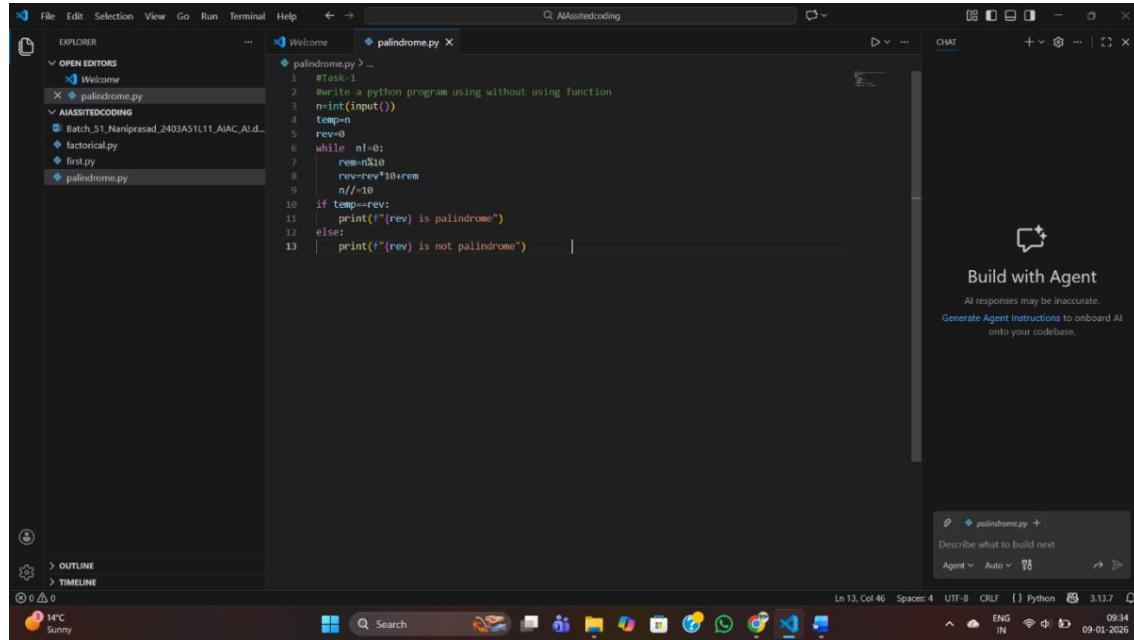


2403A51L24

batch-51

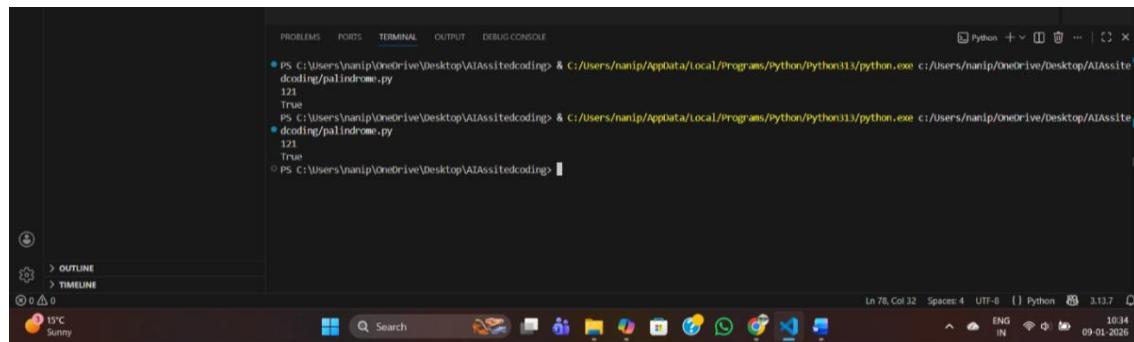
#Task1

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("{rev} is palindrome")
else:
    print("{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
o 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:

- 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

The screenshot shows a Visual Studio Code interface. The left sidebar has sections for EXPLORER, OPEN EDITORS, and AIASSISTEDCODING. The OPEN EDITORS section lists 'Welcome' and 'palindrome.py'. The AIASSISTEDCODING section lists 'Batch_51_Naniprasad_2403A51L11_AIAC_AId...' and 'palindrome.py'. The main editor area shows a Python script named 'palindrome.py' with the following code:

```
 50 #palindrome using two pointers
 51 def is_palindrome_two_pointers(s):
 52     i = 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))
```

The bottom terminal window 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> & 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
PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding>
```

Output:



The screenshot shows a Visual Studio Code (VS Code) interface. The left sidebar displays the 'OPEN EDITORS' section with 'palindrome.py' selected. The main editor area shows the following Python code:

```
palindrome.py > _  
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))
```

Below the editor, there are tabs for PROBLEMS, PORTS, TERMINAL, OUTPUT, and DEBUG CONSOLE. The TERMINAL tab is active, showing the 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
- 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

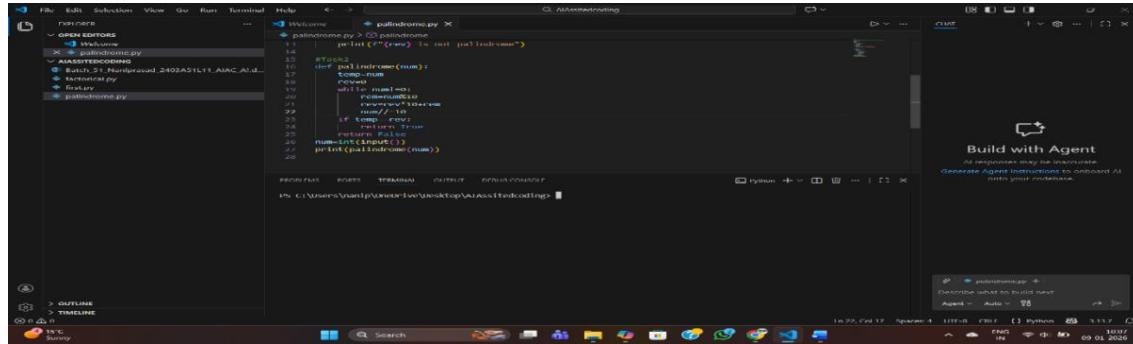
And first+=1

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

If not return False

#Task 3

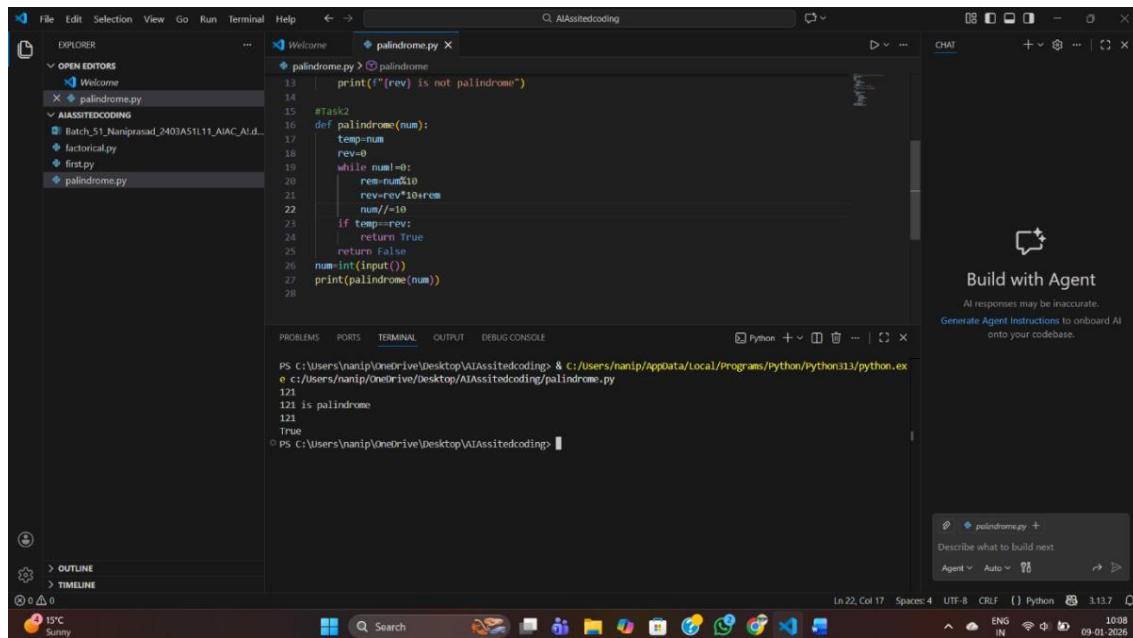
Write python program for palindrome using function



```
File Edit Selection View Go Run Terminal Help
OPEN EDITORS
  Welcome
  palindrome.py
AI ASSISTED CODING
  Batch_51_Naniprasad_2403A51L11_AIAC_A...
  factorial.py
  first.py
  palindrome.py
Help < AIAssistedCoding
  Welcome > palindrome.py
  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
    else:
      num=int(input())
    print(palindrome(num))
  121
  121 is palindrome
  True
  PS C:\Users\nanip\OneDrive\Desktop\AIAssistedCoding>
```

The screenshot shows the VS Code interface with the Python file 'palindrome.py' open. The code defines a function 'palindrome' that takes a number as input, reverses it, and checks if the reversed number is equal to the original. If they are equal, it prints 'True'; otherwise, it prints 'False'. The terminal window shows the execution of the script and the output for the number 121.

Output:



```
File Edit Selection View Go Run Terminal Help
OPEN EDITORS
  Welcome
  palindrome.py
AI ASSISTED CODING
  Batch_51_Naniprasad_2403A51L11_AIAC_A...
  factorial.py
  first.py
  palindrome.py
Help < AIAssistedCoding
  Welcome > palindrome.py
  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
    else:
      num=int(input())
    print(palindrome(num))
  121
  121 is palindrome
  True
  PS C:\Users\nanip\OneDrive\Desktop\AIAssistedCoding>
```

This screenshot shows a second version of the VS Code interface with the same Python code for a palindrome checker. The output in the terminal window is identical to the first one, showing the script running and printing 'True' for the input 121.

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

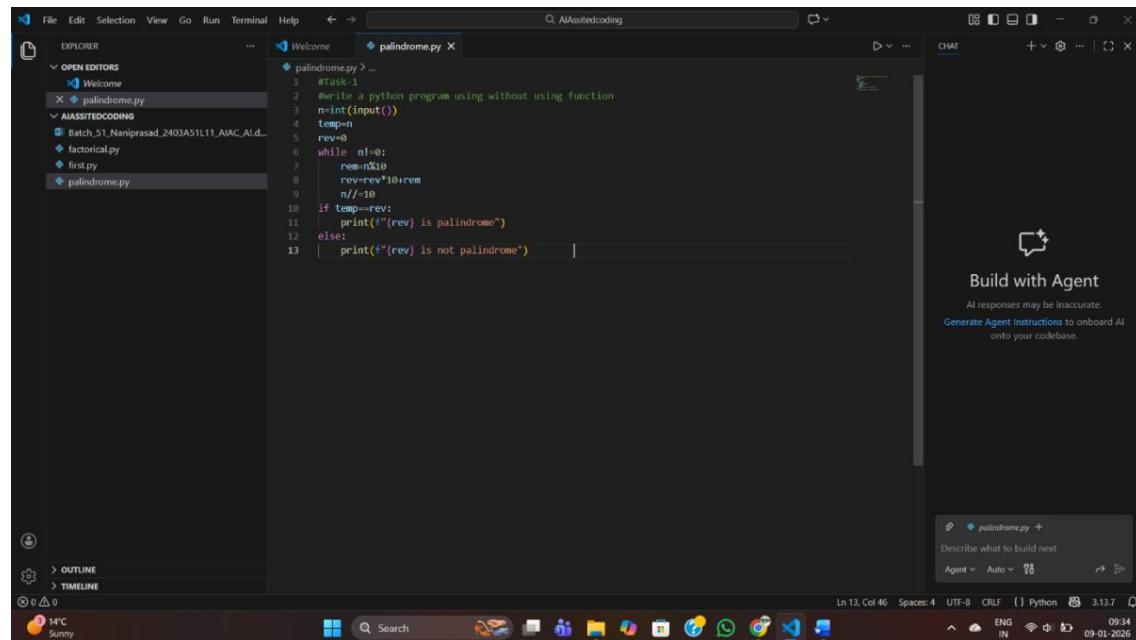
- Loop:
 - Iteration 1: rem = 1, rev = 1, num = 12
 - Iteration 2: rem = 2, rev = 12, num = 1
 - Iteration 3: rem = 1, rev = 121, num = 0
- Loop ends → rev = 121
- Compare: temp == rev → 121 == 121 → True
- Output: True

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 the Visual Studio Code interface with a dark theme. The Explorer sidebar on the left lists files: 'Welcome', 'palindrome.py' (the active file), 'Batch_51_Naniprasad_2403A51L11_AIAC_Aid...', 'factorial.py', 'first.py', and 'palindrome.py'. The 'palindrome.py' file contains 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("{rev} is palindrome")
12 else:
13     print("{rev} is not palindrome")

```

The status bar at the bottom shows: Line 13, Col 46, Spaces: 4, UTF-8, CRLF, Python 3.13.7, ENG IN, 09:34, 09-01-2026.

The screenshot shows a dark-themed instance of Visual Studio Code. In the Explorer sidebar, there are several open files: 'Welcome', 'palindrome.py', 'Batch_S1_Naniprasad_2403A51L11_AIAC_Atd...', 'factorial.py', 'first.py', and 'palindrome.py' again. The main editor area contains the following Python code:

```
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))
```

Below the editor, the terminal window shows the output of running the script:

```
PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding> & t:/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:**
 - 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.

- 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).

```

File Edit Selection View Go Run Terminal Help ← → C:\AIAssitedcoding
EXPLORER OPEN EDITORS palindrom.py ...
OPEN EDITORS Welcome
X palindrome.py
AIASSITEDCODING Batch_51_Naniprasad_2403A51L11_AIAC_AI.d...
factorial.py first.py palindrome.py
palindrom.py > ...
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\AIAssitedcoding> & C:/Users/nanip/AppData/Local/Programs/Python/Python313/python.exe c:/users/nanip/OneDrive/Desktop/AIAssitedcoding/palindrome.py
121
True
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>

```

OUTLINE TIMELINE 0 15°C Sunny

Search

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

10:34 ENG IN 09-01-2026

#Task5:

Write python program for palindrome using recursion

The screenshot shows a Visual Studio Code (VS Code) interface. The title bar includes the standard file menu (File, Edit, Selection, View, Go, Run, Terminal, Help) and a search bar for 'AIAssistedCoding'. The left sidebar has sections for 'EXPLORER', 'OPEN EDITORS' (with 1 unsaved file), 'AIASSISTEDCODING' (Batch_51_Naniprasad_2403A51L11_AIAC_Ald...), 'factorial.py', 'first.py', and 'palindrome.py'. The main editor area displays Python code for palindrome detection. The code uses two methods: a recursive approach and a string reversal approach. The recursive approach is annotated with '#Task-3'. The string reversal approach is annotated with '# Alternative simpler approach using string reversal'. The code is as follows:

```
def palindrome(num):
    if num < 0:
        return False
    num=int(input())
    print(palindrome(num))

#Task-3
def isPalindrome_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 isPalindrome_recursive(num // 10, original)

# Alternative simpler approach using string reversal
def isPalindrome_recursive_str(s):
    if len(s) <= 1:
        return True
    return s[0] == s[-1] and isPalindrome_recursive_str(s[1:-1])

num = int(input())
print(isPalindrome_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

- 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.