

School of Computer Science and Artificial Intelligence**Lab Assignment # 1**

Program : B. Tech (CSE)
Specialization :
Course Title : AI Assisted coding
Course Code :
Semester : II
Academic Session : 2025-2026
Name of Student : Bandi Vivek
Enrollment No. : 2403A51L45
Batch No. : 52

Submission Starts here :**Task 1****Write a python program for palindrome without using function**

The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar showing a project structure with files like 'palindrome.py', 'first.py', and 'factorial.py'. The main editor window displays the following Python code:

```
#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")
```

The status bar at the bottom indicates the code has 13 lines, 46 columns, and is saved in Python (Python 3.13.7). It also shows system information like battery level (14%), weather (Sunny), and date/time (09-01-2026).

Output:

The screenshot shows a terminal window titled "Python" with the following command-line interaction:

```
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>
```

The terminal is part of a larger interface with various icons and status bars at the bottom.

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 $\text{rem} = \text{n} \% 10$
 - o This gives the rightmost digit of n.
6. Append digit to reversed number:
 - o $\text{rev} = \text{rev} * 10 + \text{rem}$
 - o Shifts existing digits in rev left and adds the new last digit.
7. Remove last digit from n:
 - o $\text{n} // 10$
 - o 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:

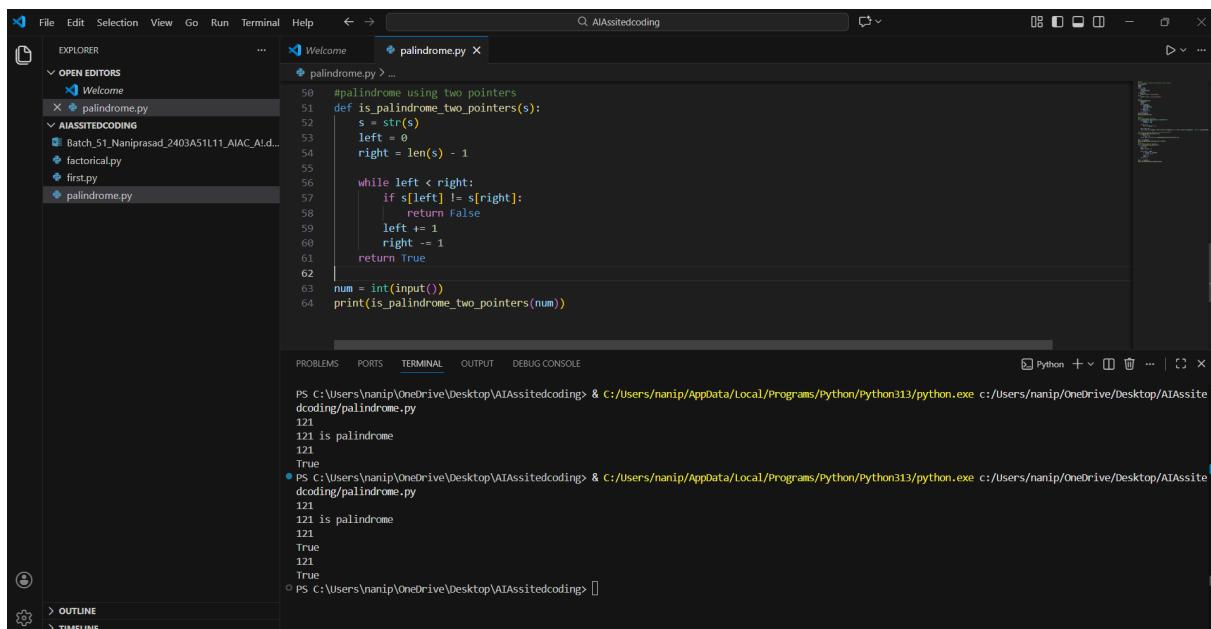
- If temp == rev, the original number reads the same backward → it's a palindrome.
- Otherwise, it's not a palindrome.

10. Output result:

- Print "rev is palindrome" if equal, else "rev is not palindrome".

#Task2:

Write optimal solution for palindrome solution



```
# 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\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 dark-themed instance of Visual Studio Code. In the Explorer sidebar, there are several files: 'Welcome', 'palindrome.py' (which is the active editor), 'Batch_51_Naniprasad_2403A51L11_AIAC_A1.d...', 'factirical.py', 'first.py', and 'palindrome.py'. The 'palindrome.py' file contains the following code:

```
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))
```

In the Terminal tab, the output of running the script is shown:

```
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

The screenshot shows a dark-themed instance of Visual Studio Code. In the Explorer sidebar, there are several files: 'Welcome', 'palindrome.py' (which is the active editor), 'Batch_51_Naniprasad_2403A51L11_AIAC_A1.d...', 'factirical.py', 'first.py', and 'palindrome.py'. The 'palindrome.py' file contains the following code:

```
1  print("Is %s is not palindrome")
2
3  #Task 2 palindrome(num)
4  def palindrome(num):
5      temp_num = num
6      reverse_num = 0
7      while num != 0:
8          remainder = num % 10
9          reverse_num = (reverse_num * 10) + remainder
10         num = num // 10
11     if temp_num == reverse_num:
12         return True
13     else:
14         return False
15
16 print(palindrome(121))
```

In the Terminal tab, the output of running the script is shown:

```
PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding>
Build with Agent
All responses may be incomplete.
Generate Agent Instructions to onboard AI
Describe what to build next
Agent > Auto > > >
```

Output:

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

- File Explorer:** Shows files like 'Welcome', 'palindrome.py', 'factorial.py', and 'first.py'.
- Editor:** Displays the code for 'palindrome.py'. The code defines a function 'palindrome' that takes a number 'num' and reverses it digit by digit to check if it's a palindrome.
- Terminal:** Shows the command-line output of running the script. It prints '121 is palindrome' and returns 'True'.
- Bottom Status Bar:** Shows the current file is 'palindrome.py', the line and column are 'Ln 22, Col 17', the encoding is 'UTF-8', and the date is '09-01-2026'.

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:

- $\text{rem} = \text{num} \% 10 \rightarrow$ extract the last digit.
- $\text{rev} = \text{rev} * 10 + \text{rem} \rightarrow$ build the reversed number digit by digit.
- $\text{num} // 10 \rightarrow$ 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 \rightarrow return True.
- Otherwise \rightarrow return False.

Main Program

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

If input is 123:

- Reverse becomes 321
- Compare: $123 != 321 \rightarrow \text{False}$

- 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_Naniprasad_2403A51L11_AIAC_AId...
factorial.py first.py palindrome.py
palindrome.py > ...
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")

```

Build with Agent

AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.

Describe what to build next
Agent Auto

Ln 13, Col 46 Spaces: 4 UTF-8 CRLF Python 3.13.7 09:34 ENG IN 09-01-2026

```

File Edit Selection View Go Run Terminal Help ← →
Q AIAssistedCoding
EXPLORER OPEN EDITORS Welcome palindrome.py ...
AIASSISTEDCODING
Batch_51_Naniprasad_2403A51L11_AIAC_AId...
factorial.py first.py palindrome.py
palindrome.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\AIAssistedCoding> & C:/Users/nanip/AppData/Local/Programs/Python/Python313/python.exe c:/Users/nanip/OneDrive/Desktop\AIAssistedCoding\palindrome.py
True

PS C:\Users\nanip\OneDrive\Desktop\AIAssistedCoding>

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", and "Batch_51_Naniprasad_2403A51L11_AIAC_Ald..".
- Editor:** The "palindrome.py" file is open, containing the following code:

```
66 def isPalindrome_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(isPalindrome_stack(num))
```
- Terminal:** The terminal shows the output of 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 temperature (15°C), weather (Sunny), and system information (Ln 78, Col 32, Spaces: 4, UTF-8, ENG IN, 10:34, 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", and "Batch_51_Naniprasad_2403A51L11_AIAC_Ald..".
- Editor:** The "palindrome.py" file is open, containing the following code:

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

Output:

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

- File Explorer:** Shows a tree view with "OPEN EDITORS" expanded, displaying "palindrome.py".
- Code Editor:** The main pane contains the following Python code for checking if a number is a palindrome using recursion:

```
palindrome.py >
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
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
```

- Terminal:** The terminal shows command-line interactions with the script:

```
PS C:\Users\manip\Desktop\AIAssitedcoding> & c:/users/manip/appdata/local/programs/python/python311/python.exe c:/users/manip/Desktop/AIAssitedcoding/palindrome.py
121
121 is palindrome
121
True
PS C:\Users\manip\Desktop\AIAssitedcoding> & c:/users/manip/appdata/local/programs/python/python311/python.exe c:/users/manip/Desktop/AIAssitedcoding/palindrome.py
121
121 is palindrome
121
True
121
True
PS C:\Users\manip\Desktop\AIAssitedcoding>
```

- Bottom Status Bar:** Shows "In 30, Col 28" and "Python" along with system icons.

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.