

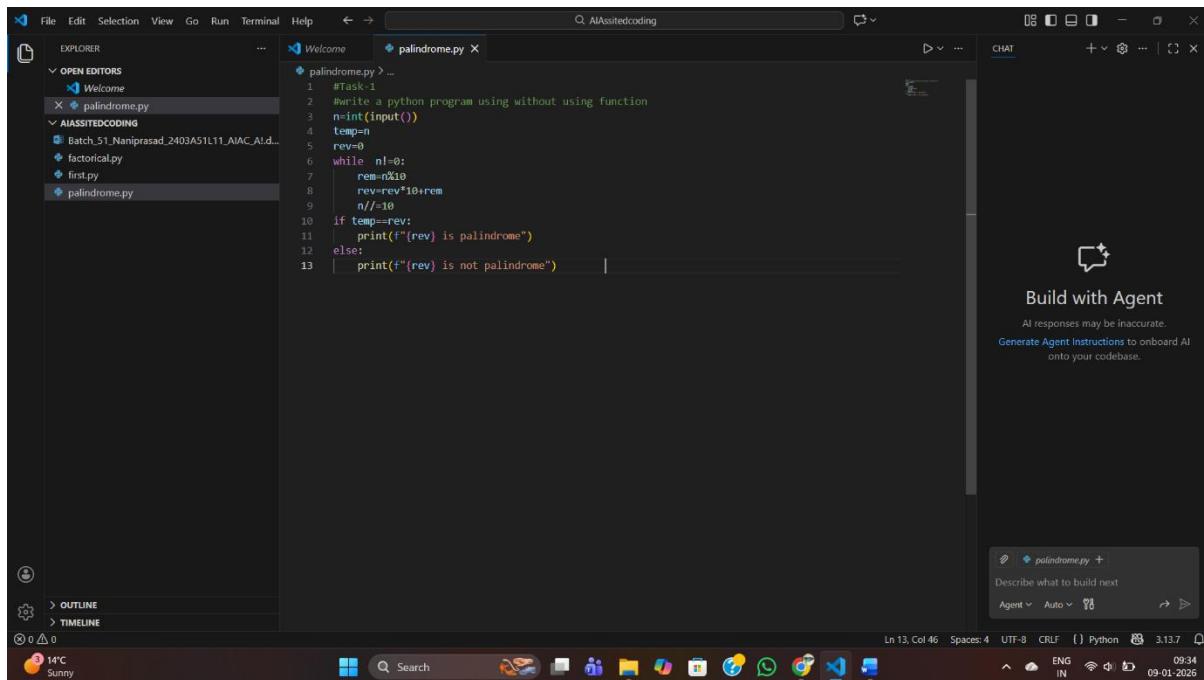
School of Computer Science and Artificial Intelligence

Lab Assignment # 4.2

Program : B. Tech (CSE)
Specialization :
Course Title : AI Assisted coding
Course Code :
Semester : II
Academic Session : 2025-2026
Name of Student : P Abhinav
Enrollment No. : 2403A51L38
Batch No. : 52
Date : 20-01-2026

#Task1

Write a python program for palindrome without using function



The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The left sidebar displays a file tree with several Python files: 'Welcome.py', 'palindrome.py' (which is currently open), 'factorial.py', and 'first.py'. In the center, the code editor contains the following Python script:

```
#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")
```

The right side of the screen features the 'CHAT' panel, which includes a 'Build with Agent' section. The status bar at the bottom shows the file path 'C:\Users\Abhinav\OneDrive\Desktop\AIAssistedcoding\palindrome.py', line count 'Ln 13, Col 46', spaces used 'Spaces: 4', encoding 'UTF-8', and file type 'Python'. It also displays system information like 'CRLF', 'Auto', and the date '09-01-2026'.

Output:

```
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 10:34 ENG IN 09-01-2026

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:

- 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 the Visual Studio Code interface with the following details:

- File Explorer:** Shows open files including "Welcome", "palindrome.py", "Batch_51_Naniprasad_2403A51L11_AIAC_Ad..", "factorial.py", "first.py", and "palindrome.py".
- Editor:** The "palindrome.py" file is open, displaying 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
62         return True
63
64 num = int(input())
65 print(is_palindrome_two_pointers(num))
```
- Terminal:** The terminal shows the execution of the script and its 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
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>
```
- Sidebar:** Includes sections for "OUTLINE" and "TIMELINE".

Output:

The screenshot shows a dark-themed instance of Visual Studio Code. In the center, there is a code editor window titled "palindrome.py". The code implements a function `is_palindrome_two_pointers` that checks if a string is a palindrome using two pointers. It also includes a main block that reads a number from input and prints the result of the function. Below the code editor, the terminal window shows the output of running the script, which correctly identifies the number 121 as a palindrome.

```
# 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\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>
```

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

This screenshot shows a similar setup to the previous one, but it includes a "Build with Agent" feature on the right side of the interface. The code in the editor is identical to the previous example, but it now includes a function `is_palindrome` that takes a number as input and returns True if it is a palindrome. The terminal output shows the script being run and correctly identifying the number 121 as a palindrome.

```
def is_palindrome(num):
    num = str(num)
    rev_num = num[::-1]
    if num == rev_num:
        return True
    else:
        return False

num = int(input())
print(is_palindrome(num))
```

```
PS C:\Users\nanip\OneDrive\Desktop\AIAssitedcoding>
Build with Agent
All responses may be incomplete.
Generate Agent Instructions to onboard AI into your pipeline.

PS C:\Users\nanip\OneDrive\Desktop\AIAssitedcoding>
```

Output:

The screenshot shows a code editor interface with a dark theme. On the left, the Explorer sidebar lists several files under 'OPEN EDITORS' and 'AIASSISTEDCODING'. The 'palindrome.py' file is open in the center editor area. The code defines a function 'palindrome' that takes a number 'num' as input. It initializes 'temp_num' to 'num', sets 'rev=0', and then enters a loop where it repeatedly divides 'num' by 10 and adds the remainder to 'rev' multiplied by 10. The loop continues until 'num' becomes 0. Finally, it compares 'temp' with 'rev' and prints the result. The terminal below shows the execution of the script and its output: '121 is palindrome'.

```
File Edit Selection View Go Run Terminal Help ← → AIAssistedcoding
EXPLORER OPEN EDITORS Welcome palindrome.py
AIASSISTEDCODING Batch_51_Naniprasad_2403A51L1J_AIAC_A1d.py factorial.py first.py palindrome.py
Welcome palindrome.py
13 |     print(f"{rev} is not palindrome")
14 |
15 #task2
16 def palindrome(num):
17     temp_num = 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))
28

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 is palindrome
121
True
PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding>

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

palindrome.py +
Describe what to build next
Agent Auto 88 1008 10:08 09-01-2026
```

Explanation:

Step-by-Step Explanation

1. Function Definition

- o `def palindrome(num):`
- o A function 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:

- $\text{rem} = \text{num} \% 10 \rightarrow$ 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 AIASSITEDCODING 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")
Ln 13, Col 46 Spaces: 4 UTT-8 CRLF () Python 3.13.7 09:34
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
Agent Auto Python

```

```

File Edit Selection View Go Run Terminal Help ← → Q AIAssistedcoding
EXPLORER OPEN EDITORS Welcome AIASSITEDCODING 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))
Ln 78, Col 46 Spaces: 4 UTT-8 CRLF () Python 3.13.7 09:34
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\coding/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 $\text{rem} = n \% 10$.
- Build reversed number: $\text{rev} = \text{rev} * 10 + \text{rem}$.
- Remove last digit: $n // 10$.
- 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 Windows desktop environment with the Visual Studio Code (VS Code) application open. The code editor displays a Python file named `palindrome.py` containing the following 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))
```

The terminal tab at the bottom 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>
```

The status bar at the bottom right indicates the current date and time as 09-01-2026.

#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` containing the following code:

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

#task-3
#palindrome using recursive
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 palindrome.py
AIASSISTEDCODING
  Batch 51 Naniprasad.2403A51L11.AIAC.A.I.D...
    palindrome.py
    factorial.py
    first.py
    palindrome.py

Welcome palindrome.py
1 def palindrome(num):
2     if num == 0:
3         return True
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     num = num // 10
18     if original % (10 ** len(str(original))) == (10 ** (len(str(original)) - 1)) * is_palindrome_recursive(num // 10)
19         return is_palindrome_recursive(num // 10, original)
20     else:
21         return False
22
23 is_palindrome_recursive(121)
24
25 121 is palindrome
26
27 True
28
29 True
30
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
121 is palindrome
121
True
121
True
31
32
33
34
35
36
37
38
39
40
PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding>

```

In 30, Col 28 Spaces: 4 UTF-8 CRLF Python ENG IN 10:17 09-01-2026

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(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.