

Lab Assignment # 1

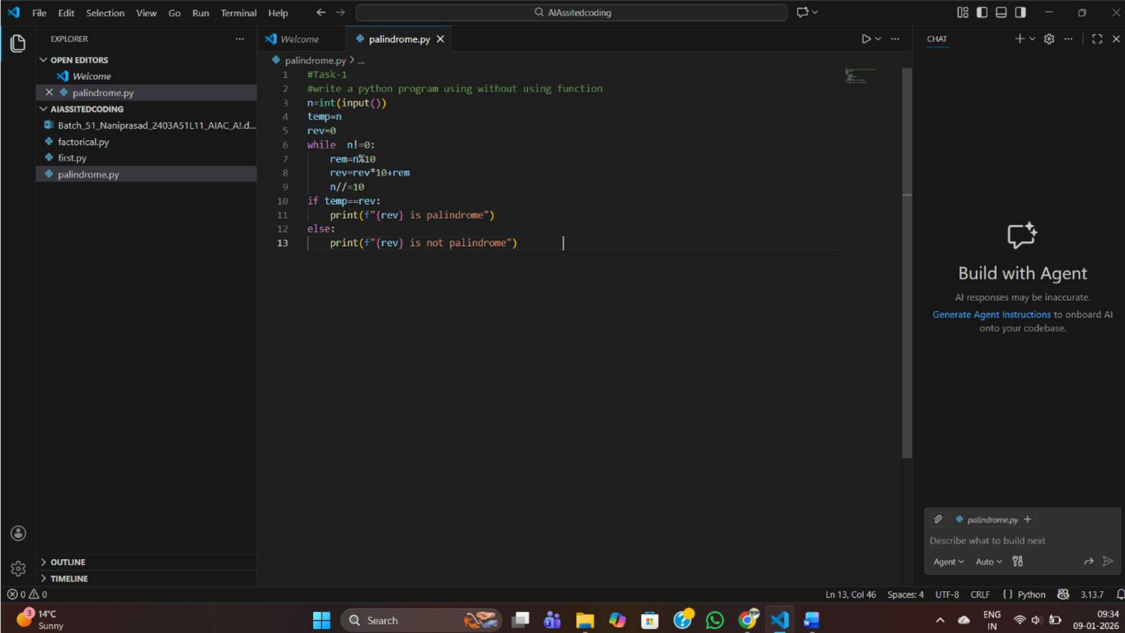
Program : B. Tech (CSE)
Specialization :
Course Title : AI Assisted coding
Course Code :
Semester : II
Academic Session : 2025-2026
Name of Student : S. Bharath
Enrollment No. : 2403A51L17
Batch No. : 51
Date :06-01-2026

Submission Starts here

OUTPUT :
SCREENSHOTS:

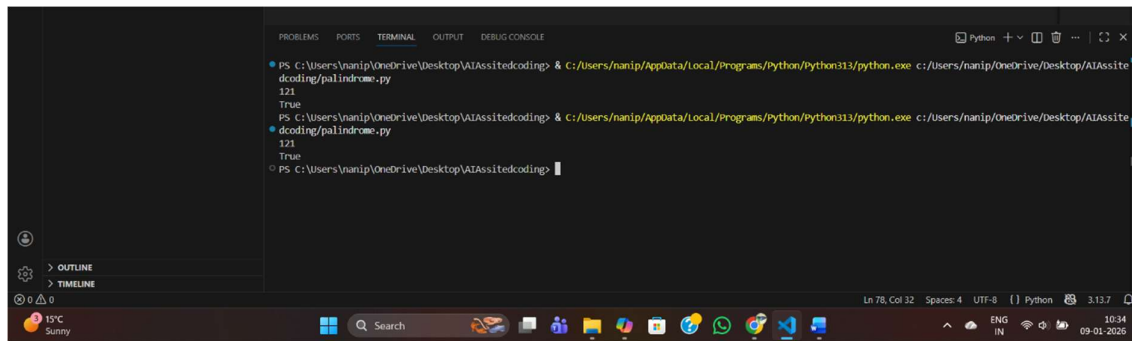
#Task1

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:

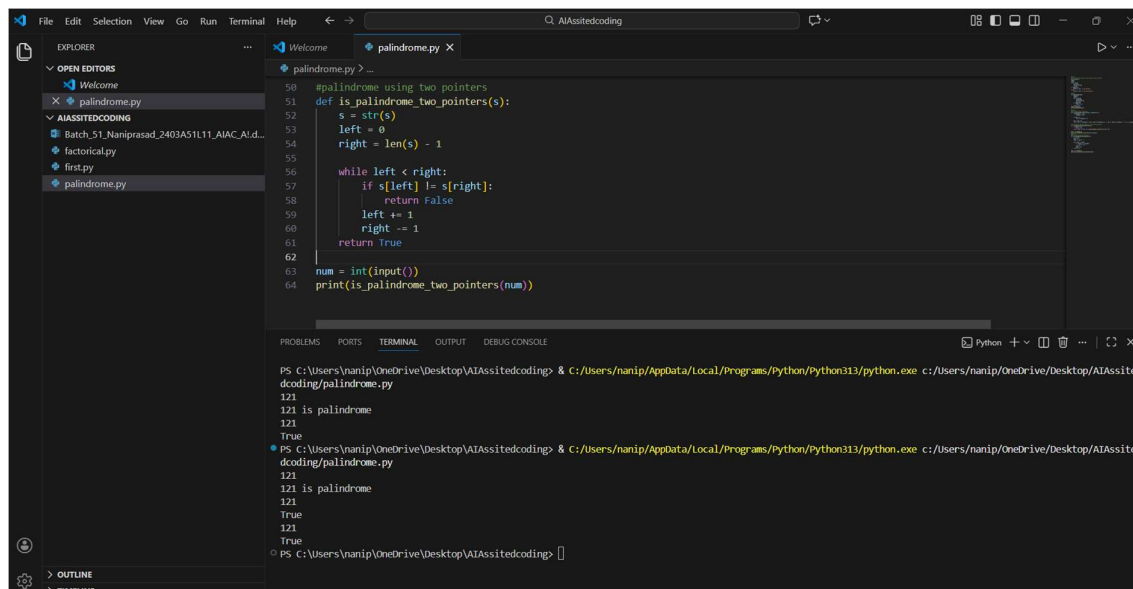
- If temp == rev, the original number reads the same backward → it's a palindrome.
- Otherwise, it's not a palindrome.

10. Output result:

- Print “rev is palindrome” if equal, else “rev is not palindrome”.

#Task2:

Write optimal solution for palindrome solution



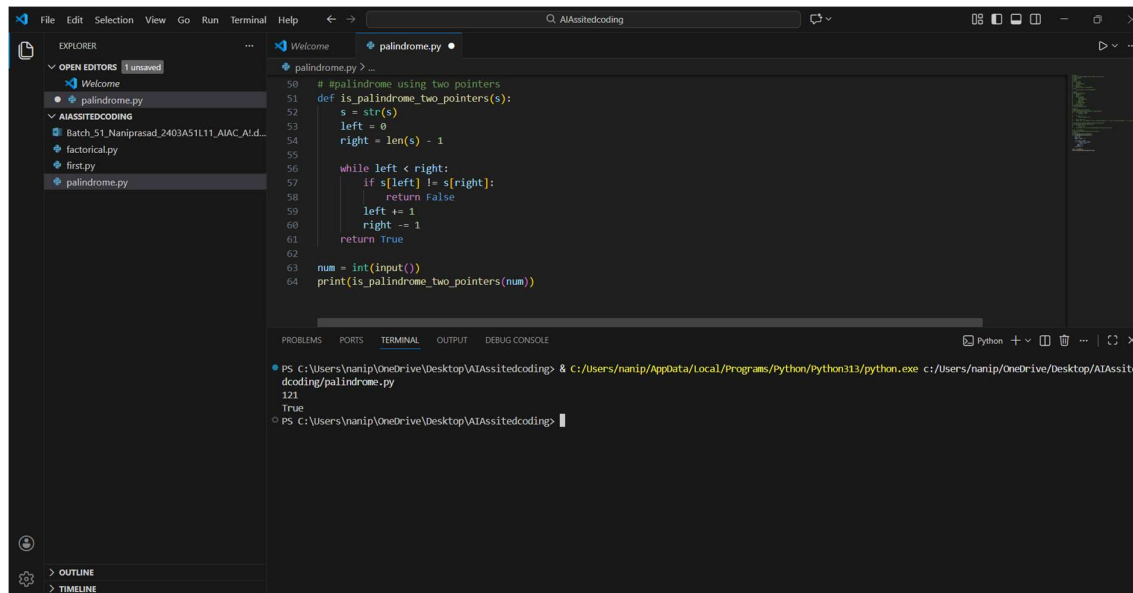
The screenshot shows a Python IDE with a file explorer on the left, a code editor in the center, and a terminal at the bottom. The code editor contains a Python script named `palindrome.py` that implements a two-pointer approach to check if a string is a palindrome. The script defines a function `is_palindrome_two_pointers(s)` which compares characters from both ends of the string towards the center. It then takes user input, converts it to a string, and prints the result of the function.

```
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))
```

The terminal shows the execution of the script. It prompts for input, and the user enters '121'. The output is '121 is palindrome'. The prompt is then repeated, and the user enters '122'. The output is '122 is not palindrome'.

```
PS C:\Users\nanip\OneDrive\Desktop\AIAssitedcoding> .\palindrome.py
121
121 is palindrome
121
PS C:\Users\nanip\OneDrive\Desktop\AIAssitedcoding> .\palindrome.py
122
122 is not palindrome
122
PS C:\Users\nanip\OneDrive\Desktop\AIAssitedcoding>
```

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\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

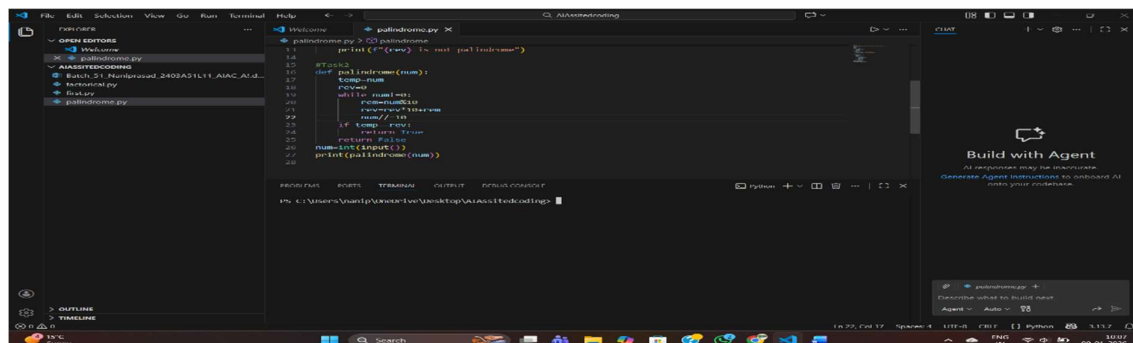
And first+=1

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

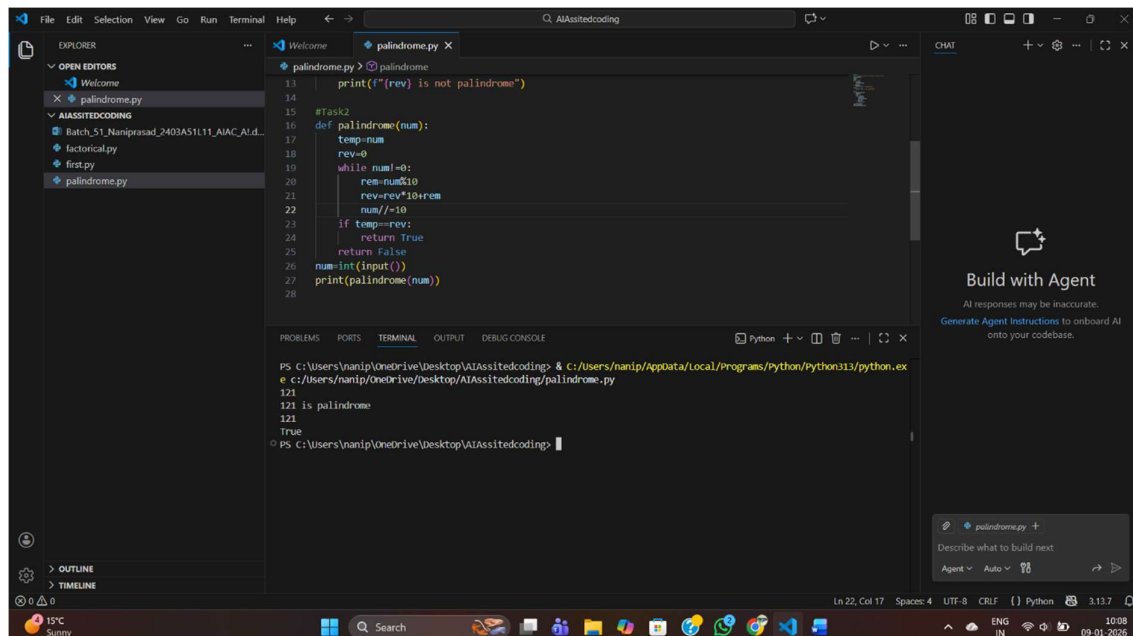


```
11 print ("Enter a number")
12 num = int(input())
13
14 def is_palindrome(num):
15     if num < 10:
16         return True
17     else:
18         num = num // 10
19         return is_palindrome(num)
20
21 print(is_palindrome(num))
```

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:



```
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))
28
```

PROBLEMS PORTS TERMINAL OUTPUT DEBUG CONSOLE

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

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:

- $rem = num \% 10 \rightarrow$ extract the last digit.
- $rev = rev * 10 + rem \rightarrow$ build the reversed number digit by digit.
- $num //= 10 \rightarrow$ 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

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

If input is 123:

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

- Output: False

#Task4:

Write Python program with using function and without using function

The screenshot shows a VS Code editor window with a file named `palindrome.py`. The code is as follows:

```
1 #Task-4
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 interface includes a sidebar with Explorer, Search, and Run and Debug views. The bottom status bar shows the file is at line 13, column 46, with 4 spaces, UTF-8 encoding, CRLF line endings, and Python 3.13.7.

The screenshot shows the same VS Code editor window with the `palindrome.py` file. The code is as follows:

```
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))
```

Below the code editor, the TERMINAL view shows the execution output:

```
PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding> & c:\Users\nanip\AppData\Local\Programs\Python\Python313\python.exe c:/Users/nanip/OneDrive/Desktop/
doding/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`).


```
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

```
def palindrome(num):
    return True
    return False
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)

# 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(str(num)))
```

Output:

```
palindrome.py
16 def palindrome(num):
17     return True
24 return False
25 num=int(input())
27 print(palindrome(num))
28
29
30 #Task 3
31 #palindrome using recursion
32 def is_palindrome_recursive(num, original_str):
33     if original is None:
34         original = num
35
36     if num == 0:
37         return original == 0
38
39     rem = num % 10
40     return rem == (original // (10 ** len(str(original)))) // (10 ** (len(str(original)) - 1)) and is_palindrome_recursive(num // 10)

PROBLEMS PORTS TERMINAL OUTPUT DEBUG CONSOLE
Python + + + + +
PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding> & C:\Users\nanip\AppData\Local\Programs\Python\Python113\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\Python113\python.exe c:\Users\nanip\OneDrive\Desktop\AIAssistedcoding\palindrome.py
121
121 is palindrome
121
True
PS C:\Users\nanip\OneDrive\Desktop\AIAssistedcoding>
```

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.