

Assignment 3.1

AI Assisted Coding

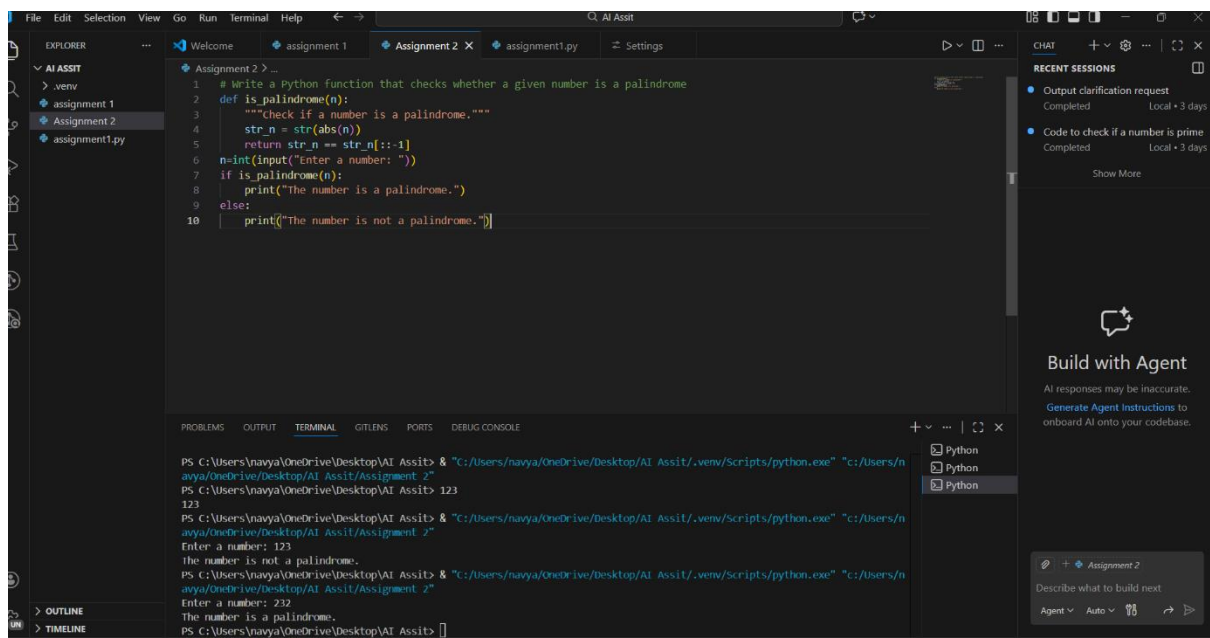
Name: Sunny Kolluri

HTNO: 2303A52444

Task 1:

Prompt:

Write a Python function that checks whether a given number is a palindrome



The screenshot shows a Visual Studio Code editor with a Python file named 'Assignment 2.py'. The code defines a function 'is_palindrome(n)' that checks if a number is a palindrome by converting it to a string, reversing it, and comparing the two. The function is then used in a main block to prompt the user for a number and print the result. The terminal at the bottom shows the execution of the script, with the user entering '123' and the output 'The number is not a palindrome.', followed by entering '232' and the output 'The number is a palindrome.'.

```
1 # Write a Python function that checks whether a given number is a palindrome
2 def is_palindrome(n):
3     """check if a number is a palindrome."""
4     str_n = str(abs(n))
5     return str_n == str_n[::-1]
6
7 n=int(input("Enter a number: "))
8 if is_palindrome(n):
9     print("The number is a palindrome.")
10 else:
11     print("The number is not a palindrome.")
```

```
PS C:\Users\navya\OneDrive\Desktop\AI Assit> "C:/Users/navya/OneDrive/Desktop/AI Assit/.venv/Scripts/python.exe" "C:/Users/navya/OneDrive/Desktop/AI Assit/Assignment 2"
123
The number is not a palindrome.
PS C:\Users\navya\OneDrive\Desktop\AI Assit> "C:/Users/navya/OneDrive/Desktop/AI Assit/.venv/Scripts/python.exe" "C:/Users/navya/OneDrive/Desktop/AI Assit/Assignment 2"
Enter a number: 232
The number is a palindrome.
PS C:\Users\navya\OneDrive\Desktop\AI Assit>
```

Observation:

- The program checks whether a given number is a palindrome by converting it into a string.
- The `abs(n)` function ensures that negative numbers are handled correctly.
- The string is compared with its reverse (`str_n[::-1]`).
- If both are the same, the number is identified as a palindrome; otherwise, it is not a palindrome.

- This approach is easy to implement, efficient, and avoids complex mathematical operations.

Task 2:

Prompt: #Input: 5 → Output: 120

#Write a Python function to calculate factorial of a number.

The screenshot shows a VS Code editor with a Python file containing a recursive factorial function. The code is as follows:

```

10 | print("The number is not a palindrome.")
11 | #Input: 5 → Output: 120
12 | #Write a Python function to calculate factorial of a number.
13 | def factorial(n):
14 |     if n == 0:
15 |         return 1
16 |     else:
17 |         return n * factorial(n-1)
18 | n=int(input("Enter a number: "))
19 | print(factorial(n))

```

The terminal output shows the execution of the program. It prompts the user to enter a number. The first input is 123, which results in the message "The number is not a palindrome." The second input is 232, which also results in the same message. The third input is 5, which results in the output 120.

```

PS C:\Users\navya\OneDrive\Desktop\AI Assit> "C:\Users\navya\OneDrive\Desktop\AI Assit\.venv\Scripts\python.exe" "C:\Users\navya\OneDrive\Desktop\AI Assit\Assignment 2"
Enter a number: 123
The number is not a palindrome.
PS C:\Users\navya\OneDrive\Desktop\AI Assit> "C:\Users\navya\OneDrive\Desktop\AI Assit\.venv\Scripts\python.exe" "C:\Users\navya\OneDrive\Desktop\AI Assit\Assignment 2"
Enter a number: 232
The number is not a palindrome.
Enter a number: 5
120
PS C:\Users\navya\OneDrive\Desktop\AI Assit>

```

Observation:

The program calculates the **factorial of a number** using **recursion**.

The base case $n == 0$ returns 1, which stops the recursive calls. For values greater than 0, the function calls itself with $n-1$.

The result is obtained by multiplying all numbers from n down to 1.

This method clearly demonstrates the **concept of recursion** in Python.

Task3:

Prompt: #Input: 153 → Output: Armstrong Number

#Input: 370 → Output: Armstrong Number

#Input: 123 → Output: Not an Armstrong Number

#Write a Python function to check whether a number is an Armstrong number.

```
18 n=int(input("Enter a number: "))
19 print(factorial(n))
20 #Input: 151 -> Output: Armstrong Number
21 #Input: 370 -> Output: Armstrong Number
22 #Input: 123 -> Output: Not an Armstrong Number
23 #Write a Python function to check whether a number is an Armstrong number.
24 def is_armstrong(n):
25     str_n = str(n)
26     num_digits = len(str_n)
27     sum_of_powers = sum(int(digit) ** num_digits for digit in str_n)
28     return sum_of_powers == n
29 n=int(input("Enter a number: "))
30 if is_armstrong(n):
31     print("The number is an Armstrong number.")
32 else:
33     print("The number is not an Armstrong number.")
```

Enter a number: 232
The number is a palindrome.
Enter a number: 12
479001600
PS C:\Users\navya\OneDrive\Desktop\AI_Assit> & "C:/Users/navya/OneDrive/Desktop/AI_Assit/.venv/scripts/python.exe" "C:/Users/navya/OneDrive/Desktop/AI_Assit/Assignment_2"
Enter a number: 232
The number is a palindrome.
Enter a number: 5
120
Enter a number: 153
The number is an Armstrong number.
PS C:\Users\navya\OneDrive\Desktop\AI_Assit>

Code to check if a number is prime
Completed Local • 4 days
Show More

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

Assignment 2
Describe what to build next
Agent Auto

Observation:

- The program checks whether a given number is an Armstrong number.
- The number is first converted into a string to count the total number of digits.
- Each digit is raised to the power of the total number of digits and added together.
- If the sum of these powers is equal to the original number, it is identified as an Armstrong number.
- This approach is straightforward and avoids complex calculations.

Task 4:

Prompt:

#Write an optimized Python program to classify a number as Prime, Composite, or Neither.

#Constraints: Validate input Handle numbers less than or equal to 1 Use efficient logic

```
34 #write an optimized Python program to classify a number as Prime, composite, or Neither.
35 #constraints: Validate input Handle numbers less than or equal to 1 use efficient logic
36 def is_prime(n):
37     if n <= 1:
38         return False
39     if n <= 3:
40         return True
41     if n % 2 == 0 or n % 3 == 0:
42         return False
43     i = 5
44     while i * i <= n:
45         if n % i == 0 or n % (i + 2) == 0:
46             return False
47         i += 6
48     return True
49
50 n = int(input("Enter a number: "))
51 if is_prime(n):
52     print("The number is a Prime number.")
53 elif n > 1:
54     print("The number is a composite number.")
55 else:
56     print("The number is Neither Prime nor composite.")
```

Enter a number: 56
The number is a composite number.
PS C:\Users\navya\OneDrive\Desktop\AI Assit>

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

Observation:

The program classifies a given number as Prime, Composite, or Neither. Numbers less than or equal to 1 are correctly identified as Neither prime nor composite.

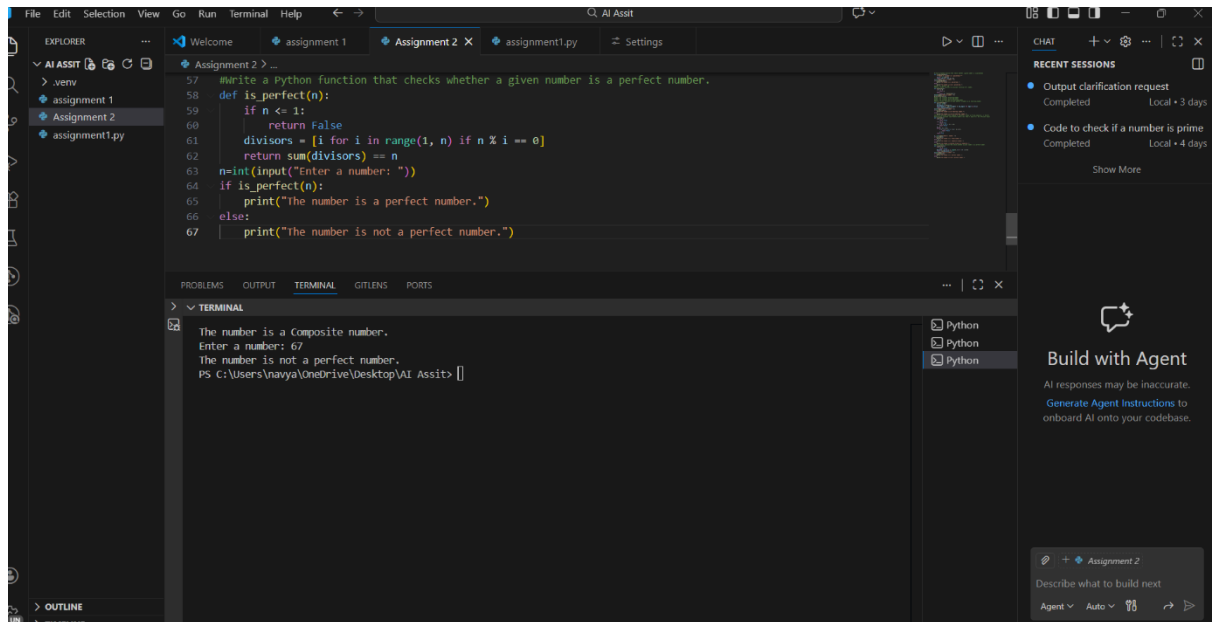
The function uses an optimized prime-checking logic by testing divisibility only up to \sqrt{n} . It skips unnecessary checks by eliminating multiples of 2 and 3 and then checking numbers of the form $6k \pm 1$.

This approach improves efficiency and reduces execution time for large numbers.

Task 5:

Prompt:

#Write a Python function that checks whether a given number is a perfect number.



Observation:

- The program checks whether a given number is a **perfect number**.
- Numbers less than or equal to **1** are immediately excluded.
- All **proper divisors** of the number (excluding the number itself) are collected.
- The sum of these divisors is compared with the original number.
- If both are equal, the number is identified as a **perfect number**.

Task 6:

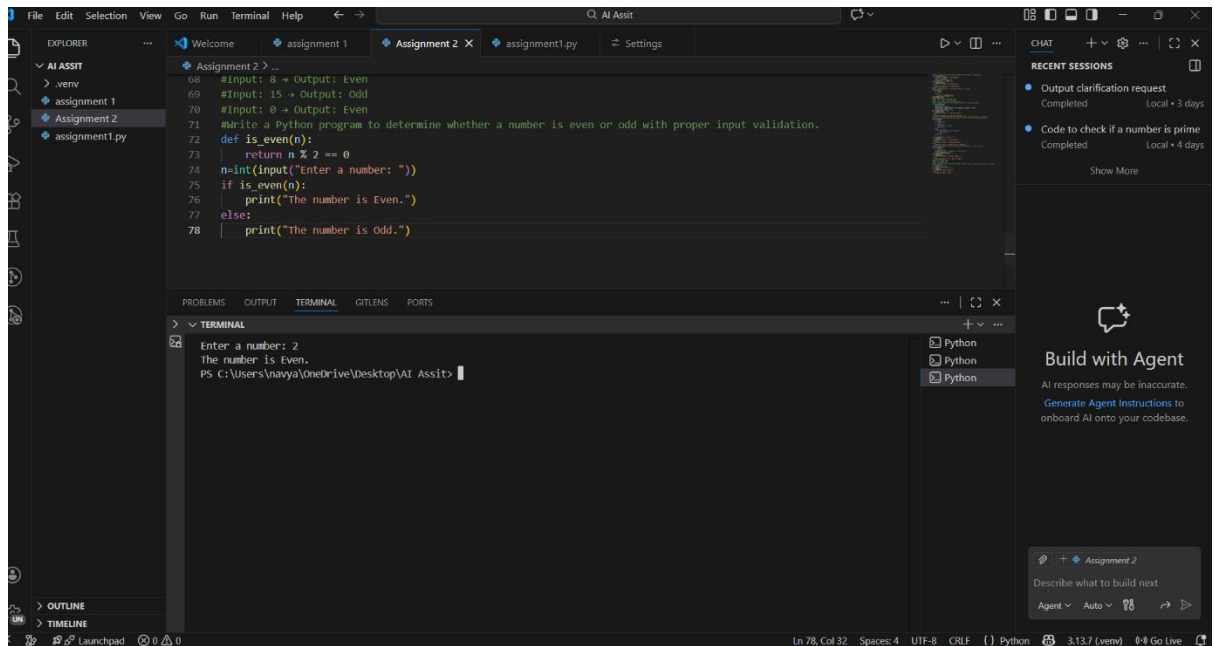
Prompt:

#Input: 8 → Output: Even

#Input: 15 → Output: Odd

#Input: 0 → Output: Even

#Write a Python program to determine whether a number is even or odd with proper input validation.



Observation:

- The program determines whether a given number is even or odd using the modulo (%) operator.
- If a number gives a remainder of 0 when divided by 2, it is classified as Even.
- Otherwise, the number is classified as Odd.
- The program correctly identifies 0 as an even number.
- This method is efficient and works for all integer inputs.