

ASSIGNMENT – 3.1

NAME : A.akshaya

HT.NO : 2303A52416

BATCH : 35

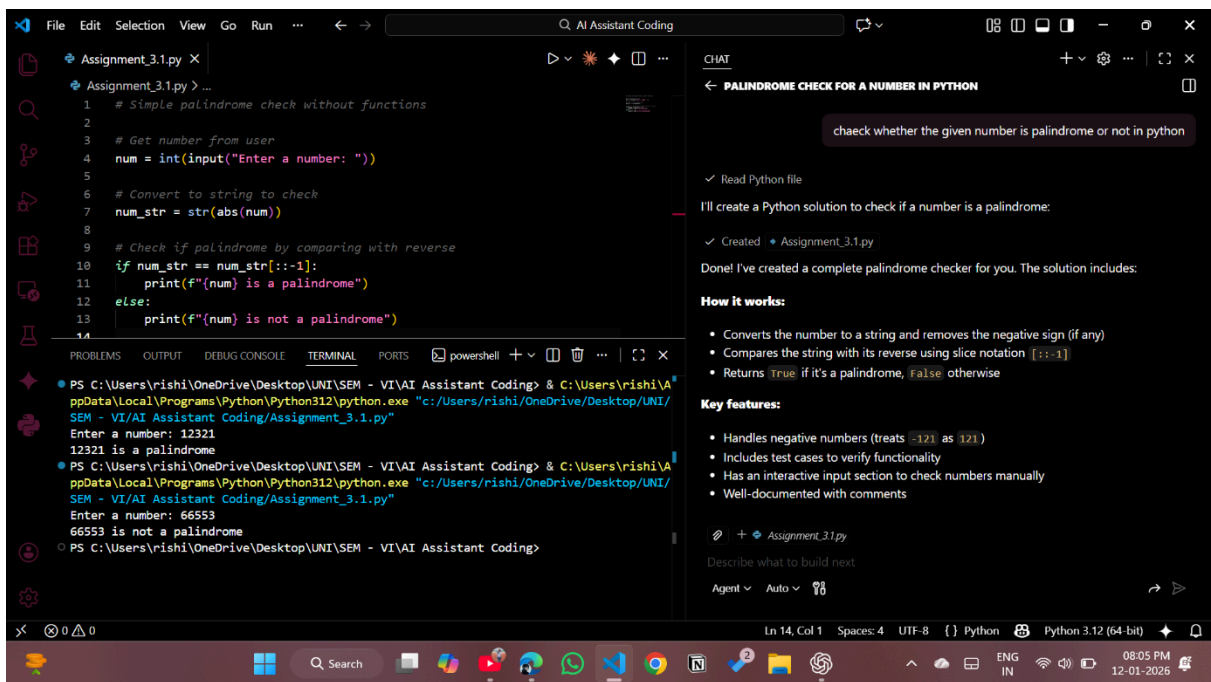
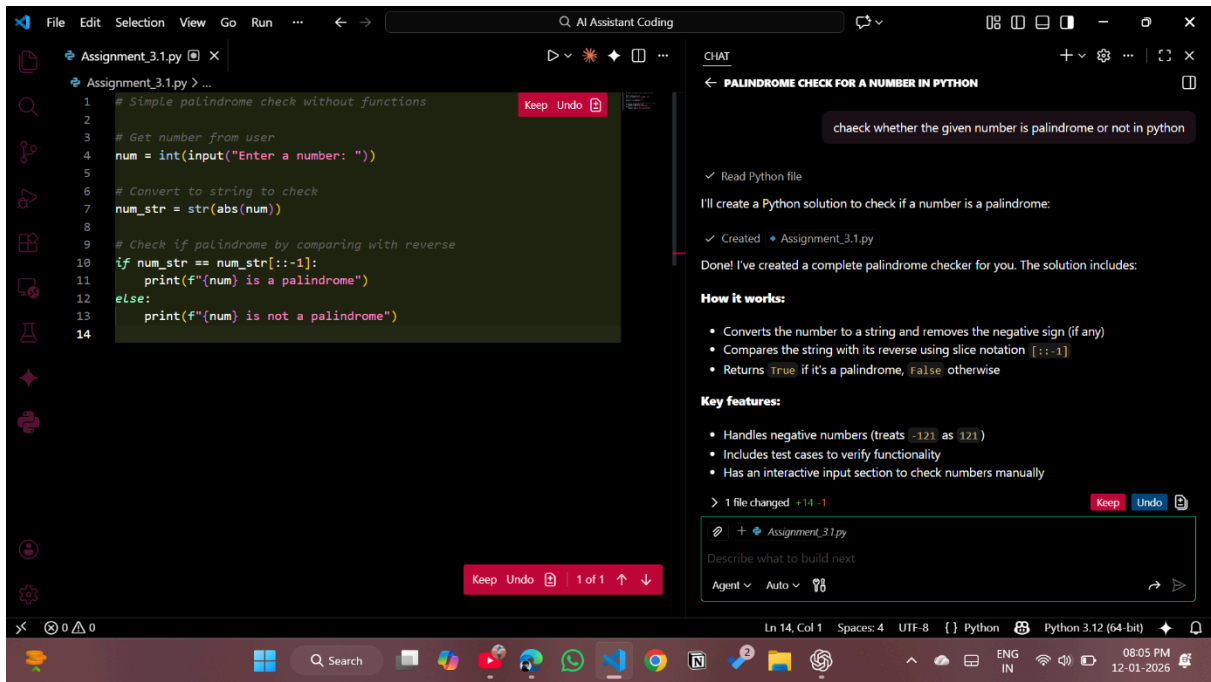
Question 1: Zero-Shot Prompting (Palindrome Number Program)

Write a zero-shot prompt (without providing any examples) to generate a Python function that checks whether a given number is a palindrome.

Task:

- Record the AI-generated code.
- Test the code with multiple inputs.
- Identify any logical errors or missing edge-case handling.

PROMPT : check whether the given number is palindrome or not in python.



Question 2: One-Shot Prompting (Factorial Calculation)

Write a one-shot prompt by providing one input-output example and ask the AI to generate a Python function to compute the factorial of a given number.

Example:

Input: 5 → Output: 120

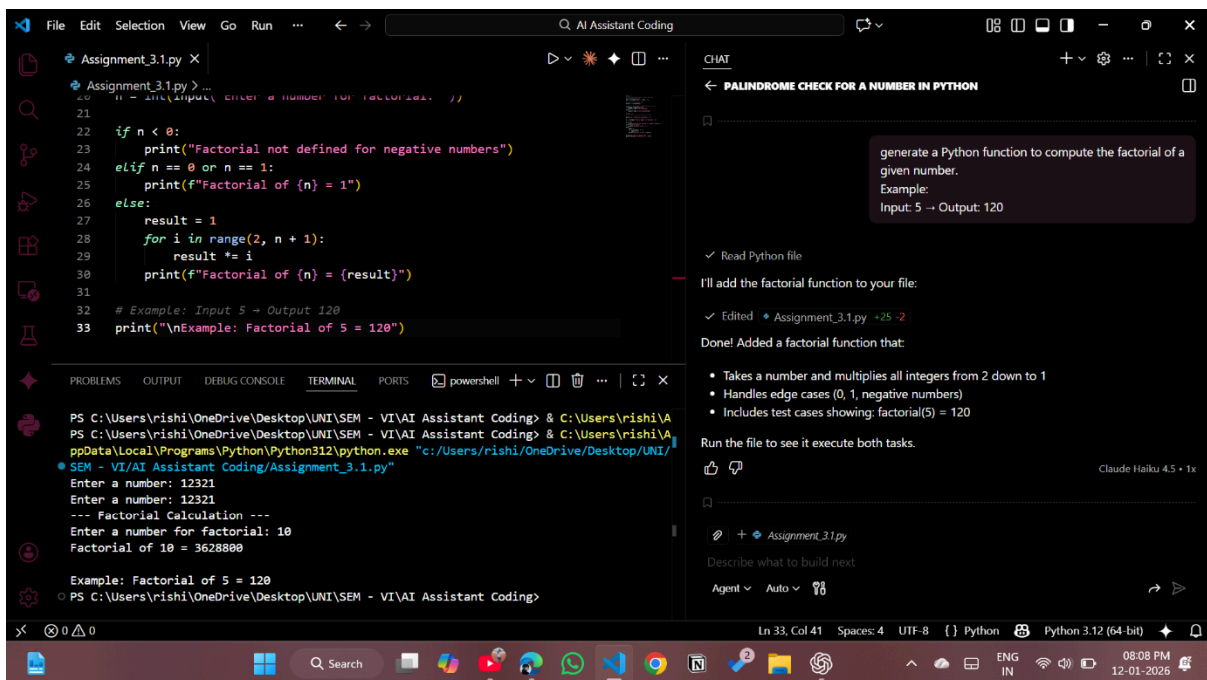
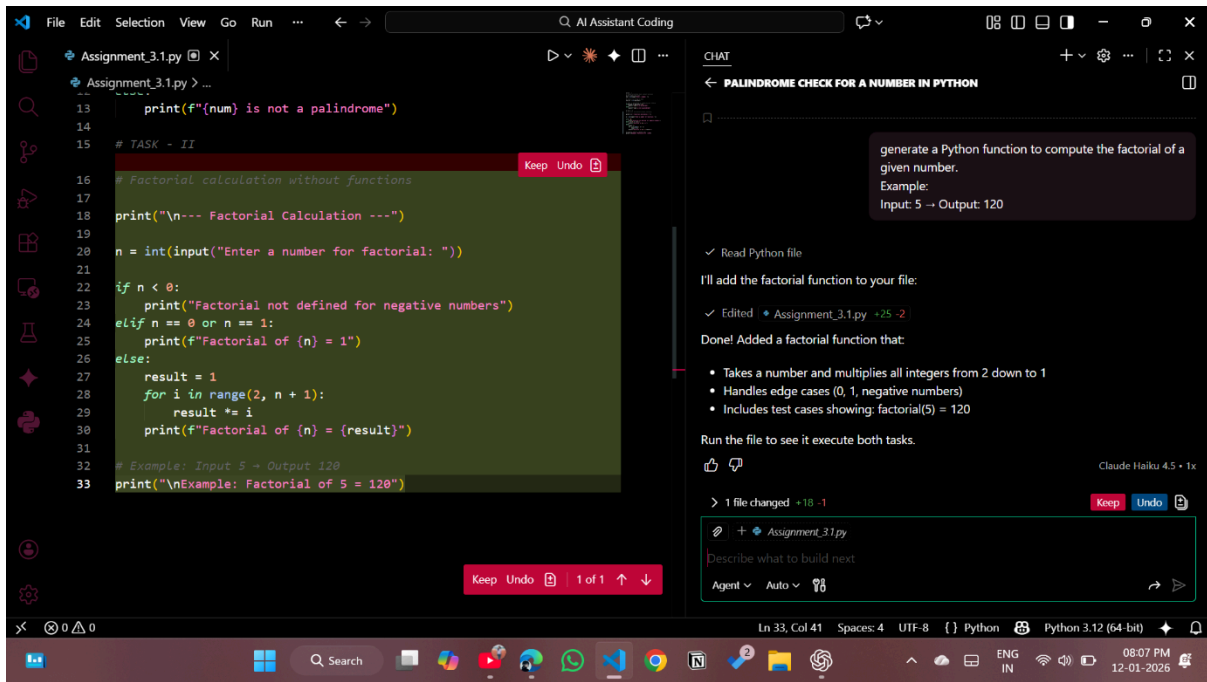
Task:

- Compare the generated code with a zero-shot solution.
- Examine improvements in clarity and correctness.

PROMPT : Generate a Python function to compute the factorial of a given number.

Example:

Input: 5 → Output: 120



Question 3: Few-Shot Prompting (Armstrong Number Check)

Write a few-shot prompt by providing multiple input-output examples to guide the AI in generating a Python function to check whether a given number is an Armstrong number.

Examples:

- Input: 153 → Output: Armstrong Number
- Input: 370 → Output: Armstrong Number
- Input: 123 → Output: Not an Armstrong Number

Task:

- Analyze how multiple examples influence code structure and accuracy.
- Test the function with boundary values and invalid inputs.

PROMPT : Python function to check whether a given number is an Armstrong number.

Examples:

- Input: 153 → Output: Armstrong Number
- Input: 370 → Output: Armstrong Number
- Input: 123 → Output: Not an Armstrong Number

```
35 # TASK - III
36 # Armstrong Number check without functions
37 # Armstrong number: sum of digits raised to power of number of digits
38
39 print("\n--- Armstrong Number Check ---")
40
41 num = int(input("Enter a number: "))
42
43 # Convert to string to get digits
44 num_str = str(num)
45 num_digits = len(num_str)
46
47 # Calculate sum of each digit raised to power of total digits
48 sum_of_powers = 0
49 for digit in num_str:
50     sum_of_powers += int(digit) ** num_digits
51
52 # Check if Armstrong number
53 if sum_of_powers == num:
54     print(f"{num} is an Armstrong Number")
55 else:
56     print(f"{num} is Not an Armstrong Number")
57
58 # Examples
59 print("\nExamples:")
60 print("153 is an Armstrong Number (1^3 + 5^3 + 3^3 = 153)")
61 print("370 is an Armstrong Number (3^3 + 7^3 + 0^3 = 370)")
62 print("123 is Not an Armstrong Number (1^3 + 2^3 + 3^3 = 36)")
63 print("9474 is an Armstrong Number (9^4 + 4^4 + 7^4 + 4^4 = 9474)")
```

PALINDROME CHECK FOR A NUMBER IN PYTHON

Examples:

- Input: 153 → Output: Armstrong Number
- Input: 370 → Output: Armstrong Number
- Input: 123 → Output: Not an Armstrong Number

How it works:

- Gets the number of digits
- Raises each digit to that power and sums them
- If the sum equals the original number, it's an Armstrong number
- Examples: $153 = 1^3 + 5^3 + 3^3$, $370 = 3^3 + 7^3 + 0^3$

```
31
32 # Example: Input 5 → Output 120
33 print("\nExample: Factorial of 5 = 120")'''
34
35 # TASK - III
36 # Armstrong Number check without functions
37 # Armstrong number: sum of digits raised to power of number of digits
38
39 print("\n--- Armstrong Number Check ---")
40
41 num = int(input("Enter a number: "))
42
43 # Convert to string to get digits
44 num_str = str(num)
45 num_digits = len(num_str)
```

PS C:\Users\rishi\OneDrive\Desktop\UNI\SEM - VI\AI Assistant Coding> C:\Users\rishi\AppData\Local\Programs\Python\Python312\python.exe "c:/Users/rishi/OneDrive/Desktop/UNI/SEM - VI/ AI Assistant Coding/Assignment_3.1.py"

--- Armstrong Number Check ---
Enter a number: 234
234 is Not an Armstrong Number

Examples:
153 is an Armstrong Number (1^3 + 5^3 + 3^3 = 153)
370 is an Armstrong Number (3^3 + 7^3 + 0^3 = 370)
123 is Not an Armstrong Number (1^3 + 2^3 + 3^3 = 36)

PS C:\Users\rishi\OneDrive\Desktop\UNI\SEM - VI\AI Assistant Coding>

PALINDROME CHECK FOR A NUMBER IN PYTHON

Examples:

- Input: 153 → Output: Armstrong Number
- Input: 370 → Output: Armstrong Number
- Input: 123 → Output: Not an Armstrong Number

How it works:

- Gets the number of digits
- Raises each digit to that power and sums them
- If the sum equals the original number, it's an Armstrong number
- Examples: $153 = 1^3 + 5^3 + 3^3$, $370 = 3^3 + 7^3 + 0^3$

(Optional Extension)

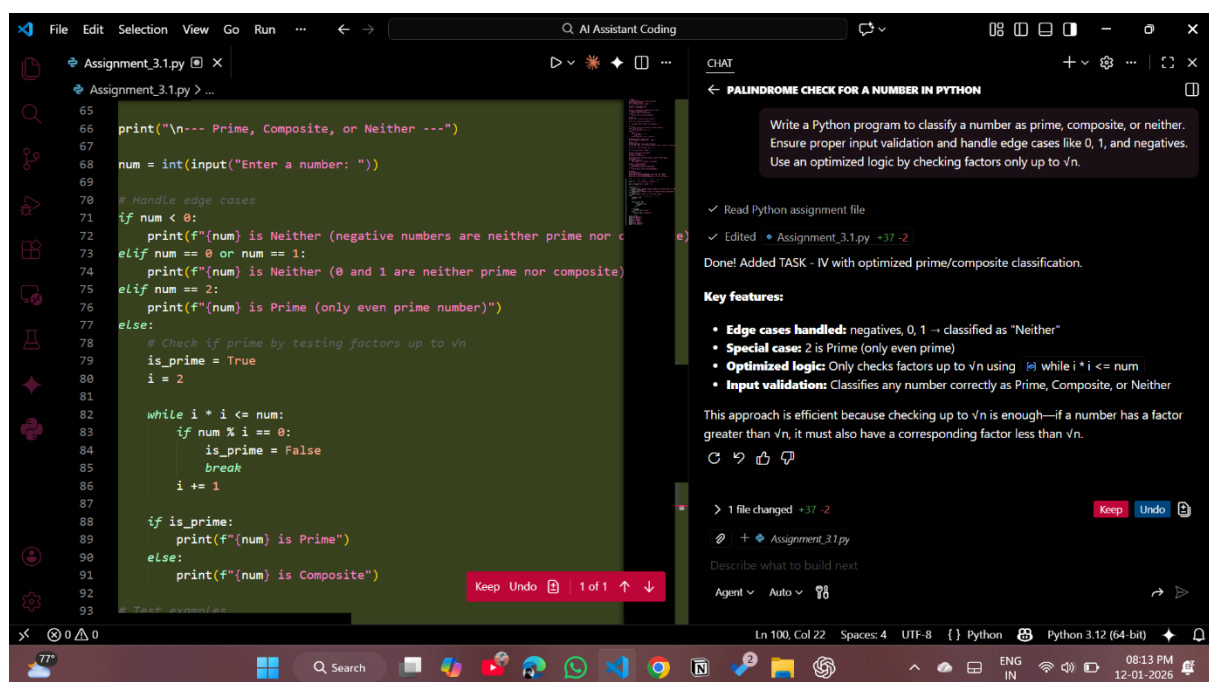
Question 4: Context-Managed Prompting (Optimized Number Classification)

Design a context-managed prompt with clear instructions and constraints to generate an optimized Python program that classifies a number as prime, composite, or neither.

Task:

- Ensure proper input validation.
- Optimize the logic for efficiency.
- Compare the output with earlier prompting strategies.

PROMPT : Write a Python program to classify a number as prime, composite, or neither. Ensure proper input validation and handle edge cases like 0, 1, and negatives. Use an optimized logic by checking factors only up to \sqrt{n} .



The screenshot displays a code editor with a Python file named 'Assignment_3.1.py'. The code implements a prime classification algorithm with input validation and optimized logic. The chat window on the right shows a prompt for writing a Python program to classify a number as prime, composite, or neither, with instructions for input validation and optimized logic.

```
65
66 print("\n--- Prime, Composite, or Neither ---")
67
68 num = int(input("Enter a number: "))
69
70 # Handle edge cases
71 if num < 0:
72     print(f"{num} is Neither (negative numbers are neither prime nor c
73 elif num == 0 or num == 1:
74     print(f"{num} is Neither (0 and 1 are neither prime nor composite)
75 elif num == 2:
76     print(f"{num} is Prime (only even prime number)")
77 else:
78     # Check if prime by testing factors up to sqrt(n)
79     is_prime = True
80     i = 2
81
82     while i * i <= num:
83         if num % i == 0:
84             is_prime = False
85             break
86         i += 1
87
88     if is_prime:
89         print(f"{num} is Prime")
90     else:
91         print(f"{num} is Composite")
92
93 # Test examples
```

PALINDROME CHECK FOR A NUMBER IN PYTHON

Write a Python program to classify a number as prime, composite, or neither. Ensure proper input validation and handle edge cases like 0, 1, and negatives. Use an optimized logic by checking factors only up to \sqrt{n} .

✓ Read Python assignment file
✓ Edited • Assignment_3.1.py +37 -2

Done! Added TASK - IV with optimized prime/composite classification.

Key features:

- **Edge cases handled:** negatives, 0, 1 → classified as "Neither"
- **Special case:** 2 is Prime (only even prime)
- **Optimized logic:** Only checks factors up to \sqrt{n} using `while i * i <= num`
- **Input validation:** Classifies any number correctly as Prime, Composite, or Neither

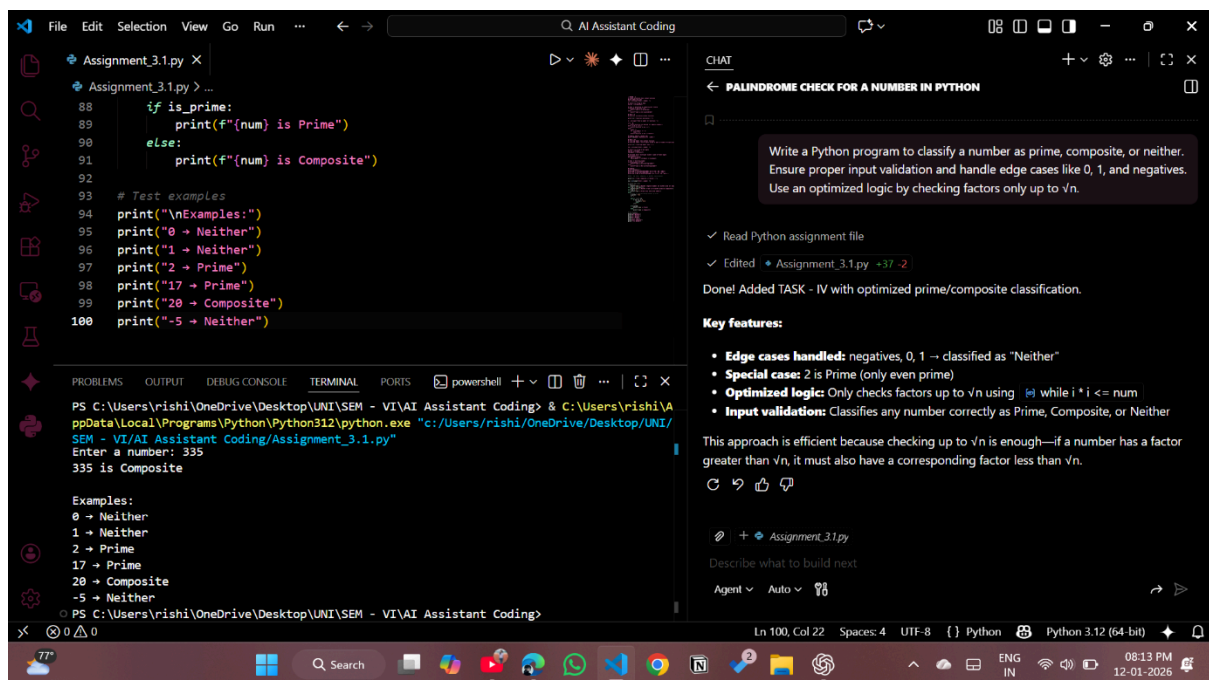
This approach is efficient because checking up to \sqrt{n} is enough—if a number has a factor greater than \sqrt{n} , it must also have a corresponding factor less than \sqrt{n} .

> 1 file changed +37 -2 Keep Undo

Assignment_3.1.py

Describe what to build next

Agent Auto 🔍



Question 5: Zero-Shot Prompting (Perfect Number Check)

Write a zero-shot prompt (without providing any examples) to generate a Python function that checks whether a given number is a perfect number.

Task:

- Record the AI-generated code.
- Test the program with multiple inputs.
- Identify any missing conditions or inefficiencies in the logic.

PROMPT : check whether the number is perfect number or not in python.

The screenshot shows the Visual Studio Code editor with a file named `Assignment_3.1.py`. The code is a Python script to check if a number is a perfect number. It includes comments, a function to find divisors, and a main loop that takes user input and prints the result. The chat window on the right shows a prompt: "check whether the number is perfect number or not in python". The chat response includes a confirmation that the task was added and a list of examples: 6 is Perfect (1 + 2 + 3 = 6), 28 is Perfect (1 + 2 + 4 + 7 + 14 = 28), and 10 is Not Perfect (1 + 2 + 5 = 8).

```
184 # Perfect number: equals the sum of its proper divisors (divisors excl
185
186 print("\n--- Perfect Number Check ---")
187
188 num = int(input("Enter a number: "))
189
190 if num <= 0:
191     print(f"{num} is Not a Perfect Number (must be positive)")
192 else:
193     # Find sum of all divisors except the number itself
194     sum_of_divisors = 0
195
196     for i in range(1, num):
197         if num % i == 0:
198             sum_of_divisors += i
199
200     # Check if perfect number
201     if sum_of_divisors == num:
202         print(f"{num} is a Perfect Number")
203         print(f"Divisors of {num}: 1 to {num-1}, Sum = {sum_of_divisor
204     else:
205         print(f"{num} is Not a Perfect Number")
206         print(f"Sum of divisors: {sum_of_divisors}")
207
208 # Examples
209 print("\nExamples:")
210 print("6 is Perfect (1 + 2 + 3 = 6)")
211 print("28 is Perfect (1 + 2 + 4 + 7 + 14 = 28)")
212 print("10 is Not Perfect (1 + 2 + 5 = 8)")
```

The screenshot shows the same Visual Studio Code editor with the `Assignment_3.1.py` file. The terminal window at the bottom is open, showing the output of the script. It displays the prompt "Enter a number: 87" and the response "87 is Not a Perfect Number". The chat window on the right is the same as in the first screenshot.

```
117 if num % i == 0:
118     sum_of_divisors += i
119
120 # Check if perfect number
121 if sum_of_divisors == num:
122     print(f"{num} is a Perfect Number")
123     print(f"Divisors of {num}: 1 to {num-1}, Sum = {sum_of_divisor
124 else:
125     print(f"{num} is Not a Perfect Number")
126     print(f"Sum of divisors: {sum_of_divisors}")
127
128 # Examples
129 print("\nExamples:")
130 print("6 is Perfect (1 + 2 + 3 = 6)")
131 print("28 is Perfect (1 + 2 + 4 + 7 + 14 = 28)")
132 print("10 is Not Perfect (1 + 2 + 5 = 8)")
```

Question 6: Few-Shot Prompting (Even or Odd Classification with Validation)

Write a few-shot prompt by providing multiple input-output examples to guide the AI in generating a

Python program that determines whether a given number is even or odd, including proper input validation.

Examples:

- Input: 8 → Output: Even
- Input: 15 → Output: Odd
- Input: 0 → Output: Even

Task:

- Analyze how examples improve input handling and output clarity.
- Test the program with negative numbers and non-integer inputs.

PROMPT : Python program that determines whether a given number is even or odd, including proper input validation.

Examples:

- Input: 8 → Output: Even
- Input: 15 → Output: Odd
- Input: 0 → Output: Even

