School of Computer Science and Artificial Intelligence

Lab Assignment-03

Course Title : AI Assistant Coding
Name of Student : Shaik Naved Ahmed
Enrollment No. : 2303A54053
Batch No. : 47-B

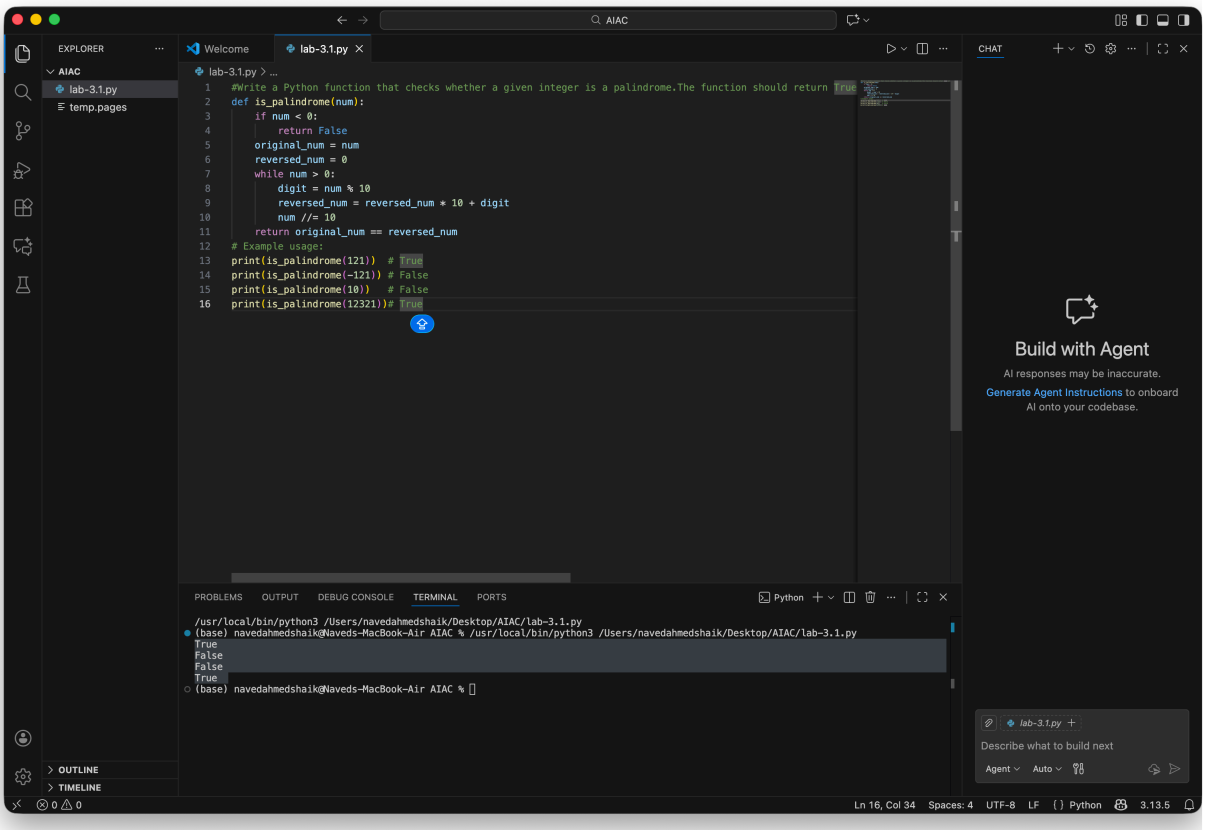
QUESTION 1: Zero-Shot Prompting (Palindrome Number Program)

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

PROMPT:

Write a Python function that checks whether a given integer is a palindrome. The function should return True if the number is a palindrome and False otherwise. Do not convert the number to a string.

SCREENSHOT OF GENERATED CODE:



INPUT & OUTPUT:

Input	Output	Reason
121	TRUE	Reversed number is 121, which is equal to the original number.
-121	FALSE	Negative numbers are not considered palindromes.
10	FALSE	Reversed number is 01 (1), which is not equal to 10.
12321	TRUE	Reversed number is 12321, which matches the original number.

EXPLANATION:

The function first checks if the number is negative and immediately returns False because negative numbers cannot be palindromes. It then stores the original number in a separate variable for comparison later and initializes a variable to build the reversed number. Using a while loop, it repeatedly takes the last digit of the number with the modulo operator, appends this digit to the reversed number by multiplying the current reversed value by 10 and adding the digit, and then removes the last digit from the original number using integer division. This process continues until all digits are processed. Finally, the function compares the reversed number with the original number and returns True if they are the same, indicating a palindrome, or False otherwise.

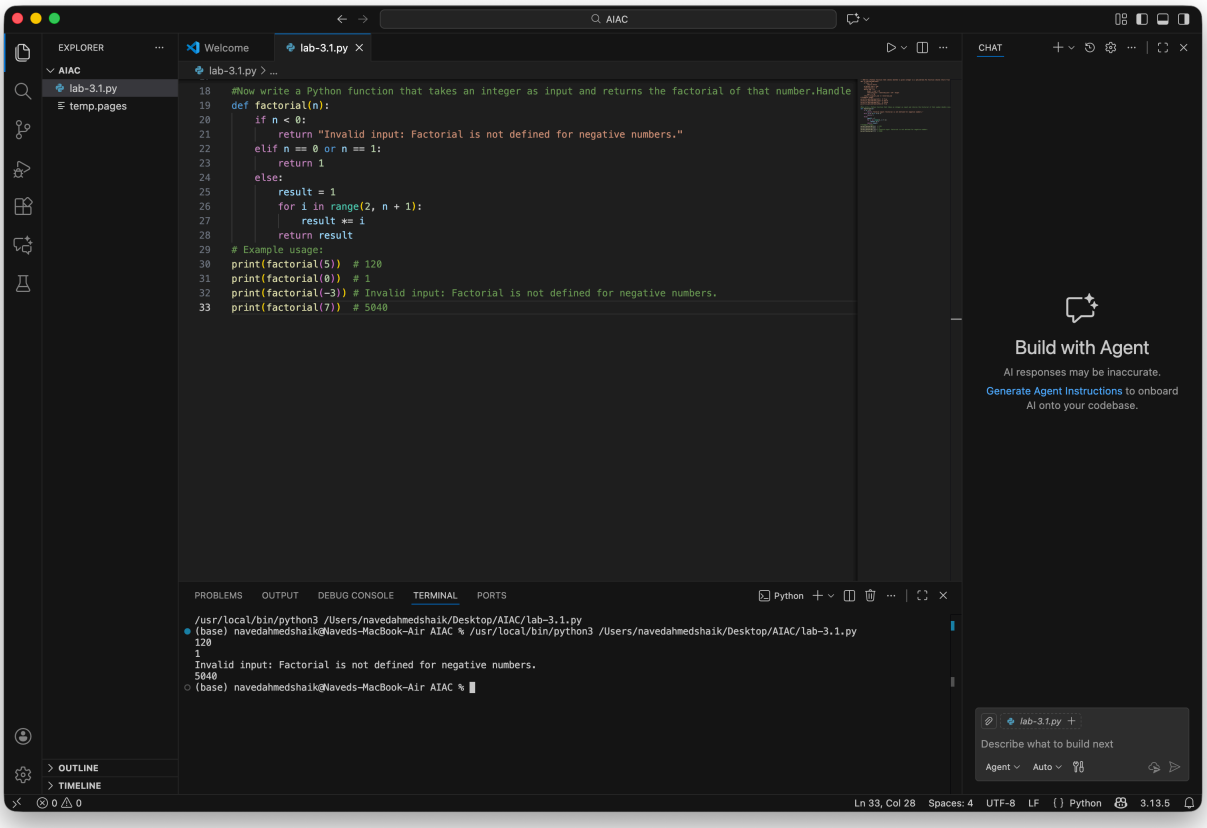
Question 2: One-Shot Prompting (Factorial Calculation)

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.

PROMPT:

Now write a Python function that takes an integer as input and returns the factorial of that number.Handle invalid inputs such as negative numbers appropriately.

SCREENSHOT OF GENERATED CODE:



INPUT & OUTPUT:

Input	Output	Reason
5	120	5! = 5 × 4 × 3 × 2 × 1 = 120.
0	1	By definition, 0! is equal to 1.
-3	Invalid input: Factorial is not defined for negative numbers.	Factorial is not defined for negative numbers.
7	5040	7! = 7 × 6 × 5 × 4 × 3 × 2 × 1 = 5040.

EXPLANATION:

The function first checks if the input number is negative and returns an error message because factorial is not defined for negative numbers. If the number is 0 or 1, it directly returns 1 since the factorial of both is 1. For any positive number greater than 1, it initializes a result variable to 1 and multiplies it by each integer from 2 up to the given number using a loop. After completing the loop, it returns the final computed factorial value.

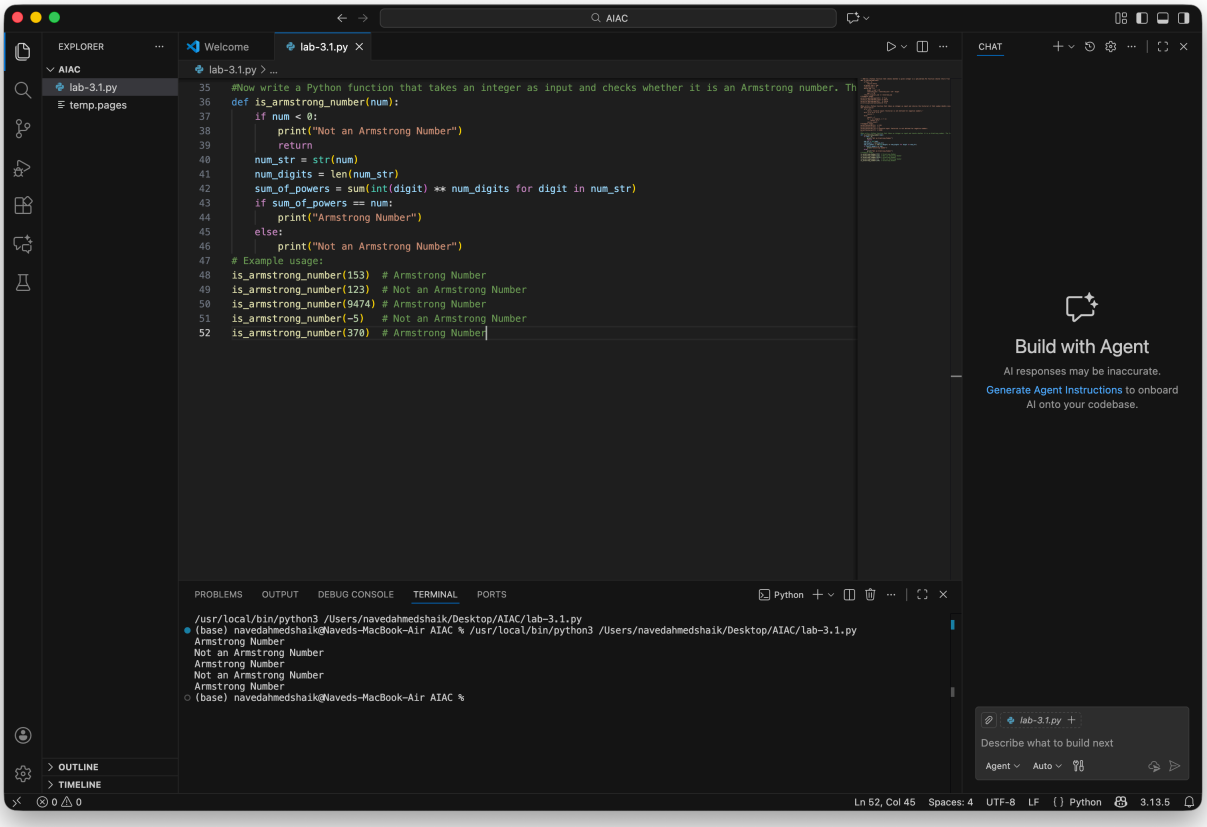
Question 3: Few-Shot Prompting (Armstrong Number Check)

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.

PROMPT:

Now write a Python function that takes an integer as input and checks whether it is an Armstrong number. The function should print "Armstrong Number" if it is, otherwise "Not an Armstrong Number".

SCREENSHOT OF GENERATED CODE:



INPUT & OUTPUT:

Input	Output	Reason
153	Armstrong Number	(1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153), which equals the original number.
123	Not an Armstrong Number	(1^3 + 2^3 + 3^3 = 1 + 8 + 27 = 36), not equal to 123.
9474	Armstrong Number	(9^4 + 4^4 + 7^4 + 4^4 = 6561 + 256 + 2401 + 256 = 9474).
-5	Not an Armstrong Number	Negative numbers are not considered Armstrong numbers.
370	Armstrong Number	(3^3 + 7^3 + 0^3 = 27 + 343 + 0 = 370), which equals the original number.

EXPLANATION:

The function first checks if the number is negative and immediately prints "Not an Armstrong Number" because Armstrong numbers cannot be negative. It then converts the number into a string to count how many digits it has. For each digit, it raises the digit to the power of the total number of digits and adds these values together. If the resulting sum is equal to the original number, the function prints "Armstrong Number"; otherwise, it prints "Not an Armstrong Number".

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.

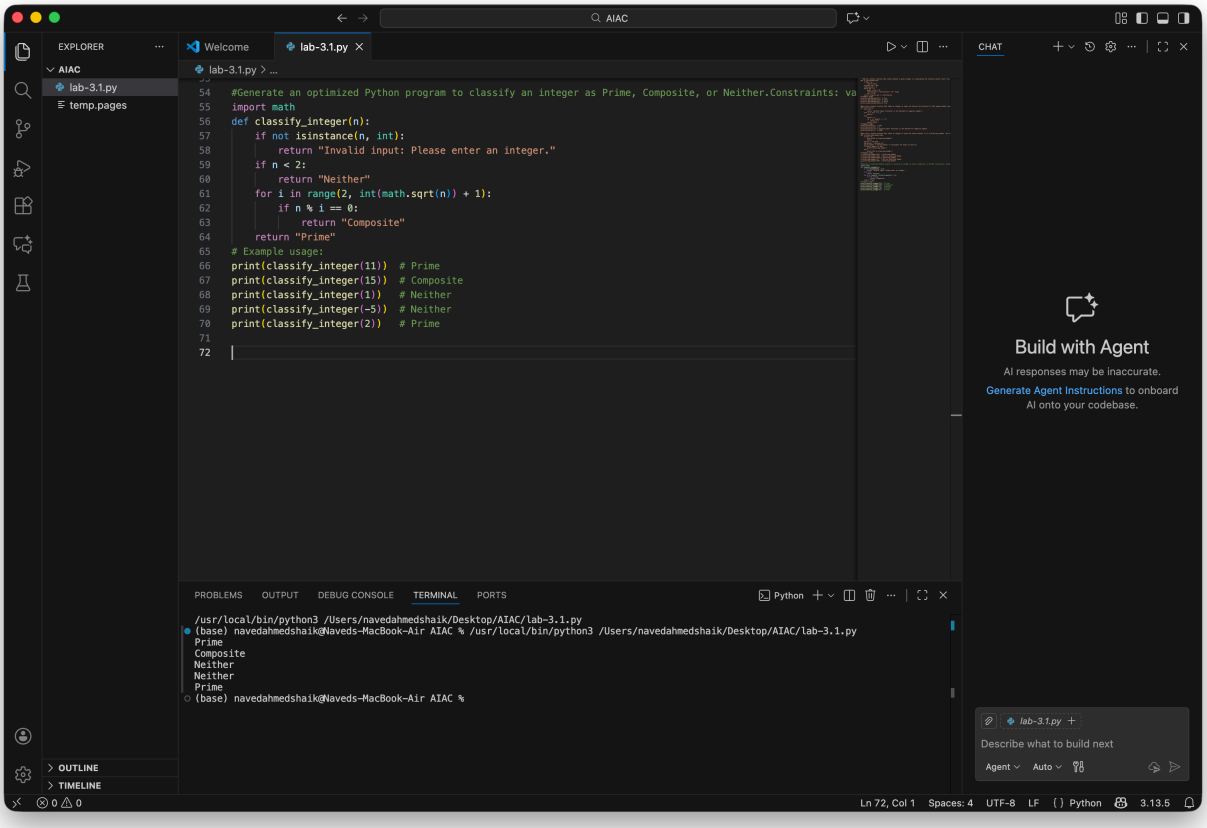
• Optimize the logic for efficiency.

• Compare the output with earlier prompting strategies.

PROMPT:

Generate an optimized Python program to classify an integer as Prime, Composite, or Neither. Constraints: validate input, check divisibility up to √n, print only one result (Prime, Composite, or Neither), and briefly compare with a basic approach.

SCREENSHOT OF GENERATED CODE:



INPUT & OUTPUT:

Input	Output	Reason
11	Prime	No divisors found from 2 to √11, so it is a prime number.
15	Composite	Divisible by 3 (15 % 3 = 0), so it is a composite number.
1	Neither	Numbers less than 2 are neither prime nor composite.
-5	Neither	Negative numbers are neither prime nor composite.
2	Prime	Only divisible by 1 and itself; loop doesn't find any other divisors.

EXPLANATION:
The function first validates whether the input is an integer; if not, it returns an invalid input message. For integers less than 2, it returns "Neither" because numbers like 0, 1, and negatives are neither prime nor composite. For valid integers ≥ 2, it checks divisibility only up to the square root of the number, which is more efficient than checking all numbers up to n. If any divisor is found, it returns "Composite"; otherwise, it returns "Prime".

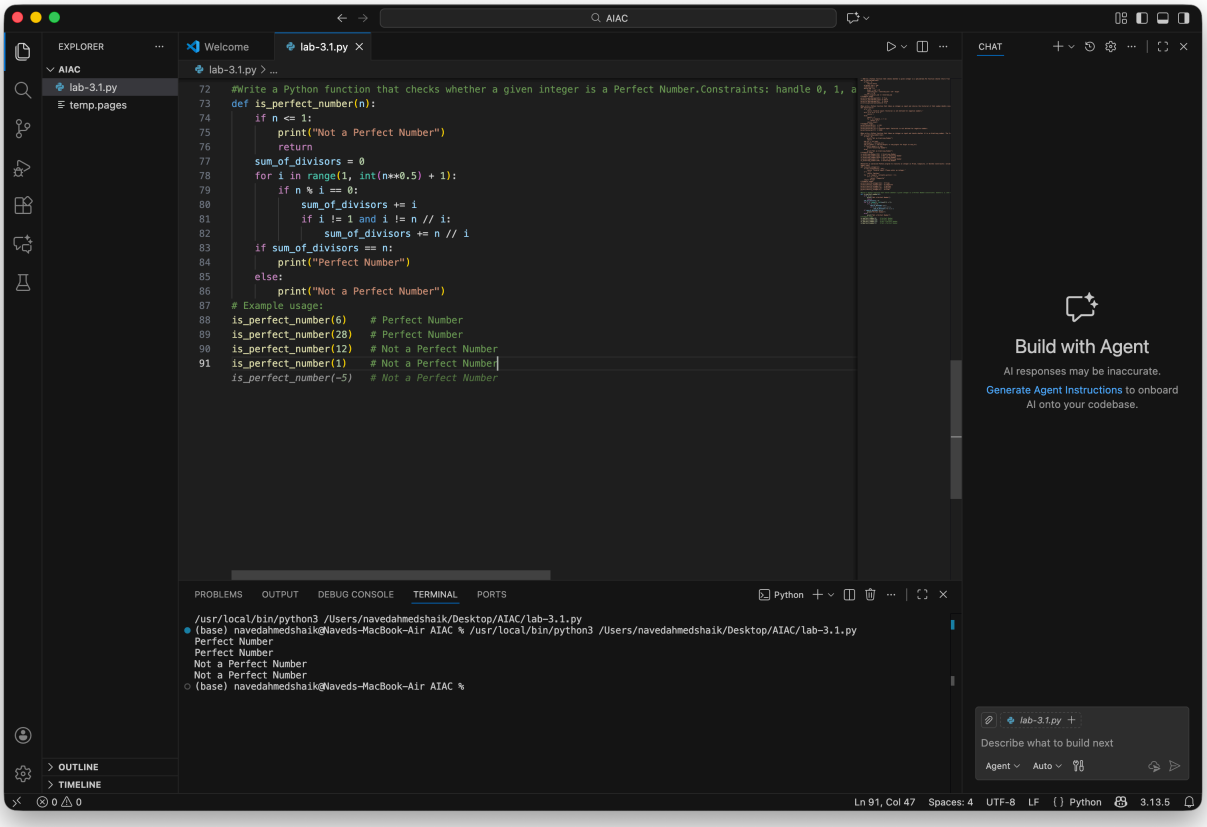
Question 5: Zero-Shot Prompting (Perfect Number Check)

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:

Write a Python function that checks whether a given integer is a Perfect Number. Constraints: handle 0, 1, and negative numbers, use an efficient divisor check, and print Perfect Number or Not a Perfect Number.

SCREENSHOT OF GENERATED CODE:
INPUT & OUTPUT:



Input	Output	Reason
6	Perfect Number	Divisors: 1, 2, 3 → Sum = 6, which equals the number.
28	Perfect Number	Divisors: 1, 2, 4, 7, 14 → Sum = 28, which equals the number.
12	Not a Perfect Number	Divisors: 1, 2, 3, 4, 6 → Sum = 16, not equal to 12.
1	Not a Perfect Number	1 has no proper divisors that sum to itself.
0	Not a Perfect Number	0 is not considered a perfect number.
-5	Not a Perfect Number	Negative numbers are not considered perfect numbers.

EXPLANATION:

The function first checks if the number is less than or equal to 1 and prints "Not a Perfect Number" because perfect numbers are positive integers greater than 1. It then initializes a variable to store the sum of proper divisors. Using an efficient loop that runs only up to the square root of the number, it finds divisors in pairs. For each divisor found, it adds both the divisor and its pair (if they are different and not the number itself) to the sum. Finally, it compares the sum of divisors with the original number and prints "Perfect Number" if they are equal; otherwise, it prints "Not a Perfect Number".

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

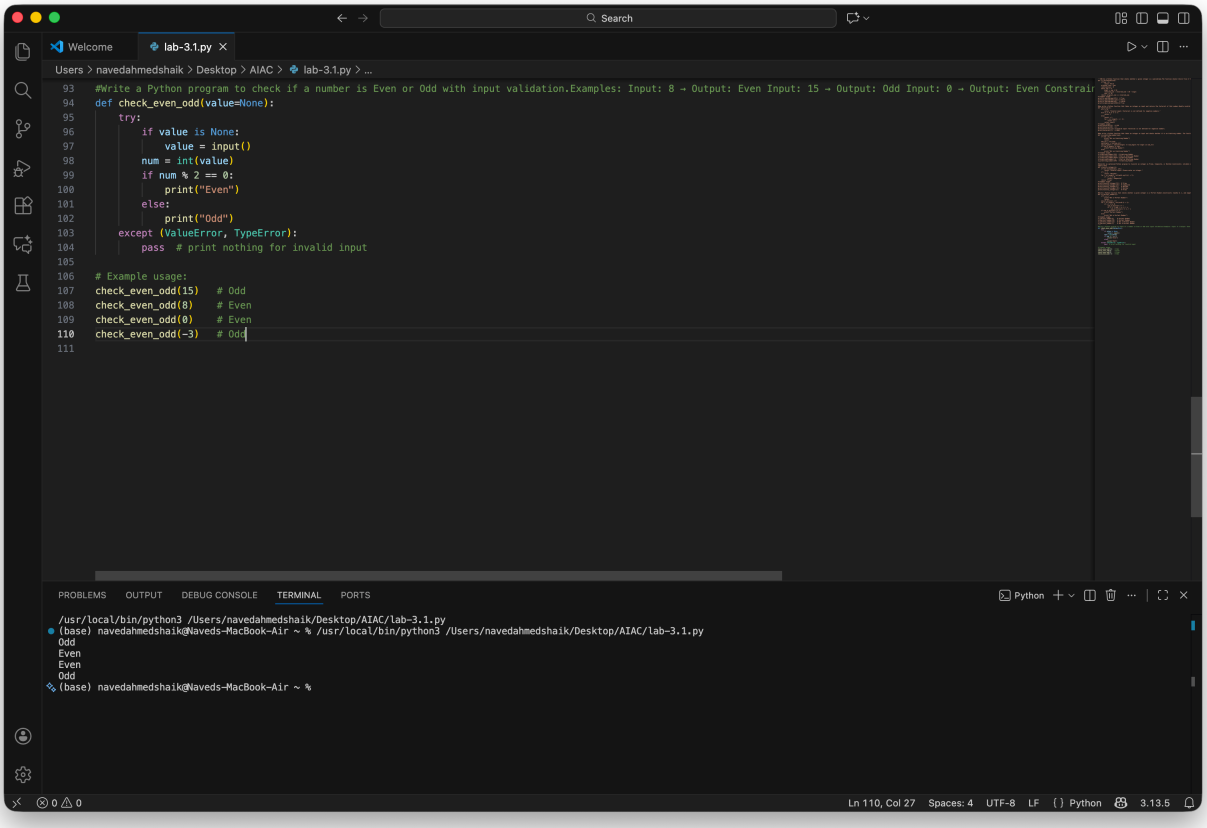
Write a few-shot prompt by providing multiple input-output examples to guide the AI in generating a Python program that determines whether a given number is even or odd, including proper input validation.
Examples:
• Input: 8 → Output: Even
• Input: 15 → Output: Odd
• Input: 0 → Output: Even
Task:
• Analyze how examples improve input handling and output clarity.
• Test the program with negative numbers and non-integer inputs.

PROMPT:

Write a Python program to check if a number is Even or Odd with input validation.
Examples:
Input: 8 → Output: Even
Input: 15 → Output: Odd
Input: 0 → Output: Even
Constraints:

- **Handle non-integer inputs and negative numbers.**
- **Print only Even or Odd.**

SCREENSHOT OF GENERATED CODE:



INPUT & OUTPUT:

Input	Output	Reason
15	Odd	15 is not divisible by 2 (15 % 2 = 1).
8	Even	8 is divisible by 2 (8 % 2 = 0).
0	Even	0 is divisible by 2 (0 % 2 = 0).
-3	Odd	-3 is not divisible by 2 (-3 % 2 = 1).
"abc"	—	Invalid input; cannot be converted to an integer.

EXPLANATION:

The function checks whether a given input is even or odd. If no value is passed, it takes input from the user. It then tries to convert the input into an integer. If the conversion is successful, it checks divisibility by 2: if the number is divisible by 2, it prints "Even", otherwise it prints "Odd". If the input is invalid (not a number or wrong type), the function catches the error and prints nothing.