# SR UNIVERSITY

AI ASSIST CODING

# Lab Assignment 3 Prompt Engineering : Improving Prompts and Context Management

**Name**: K.SRIKAR

**Pin No**. : 2503A51L03

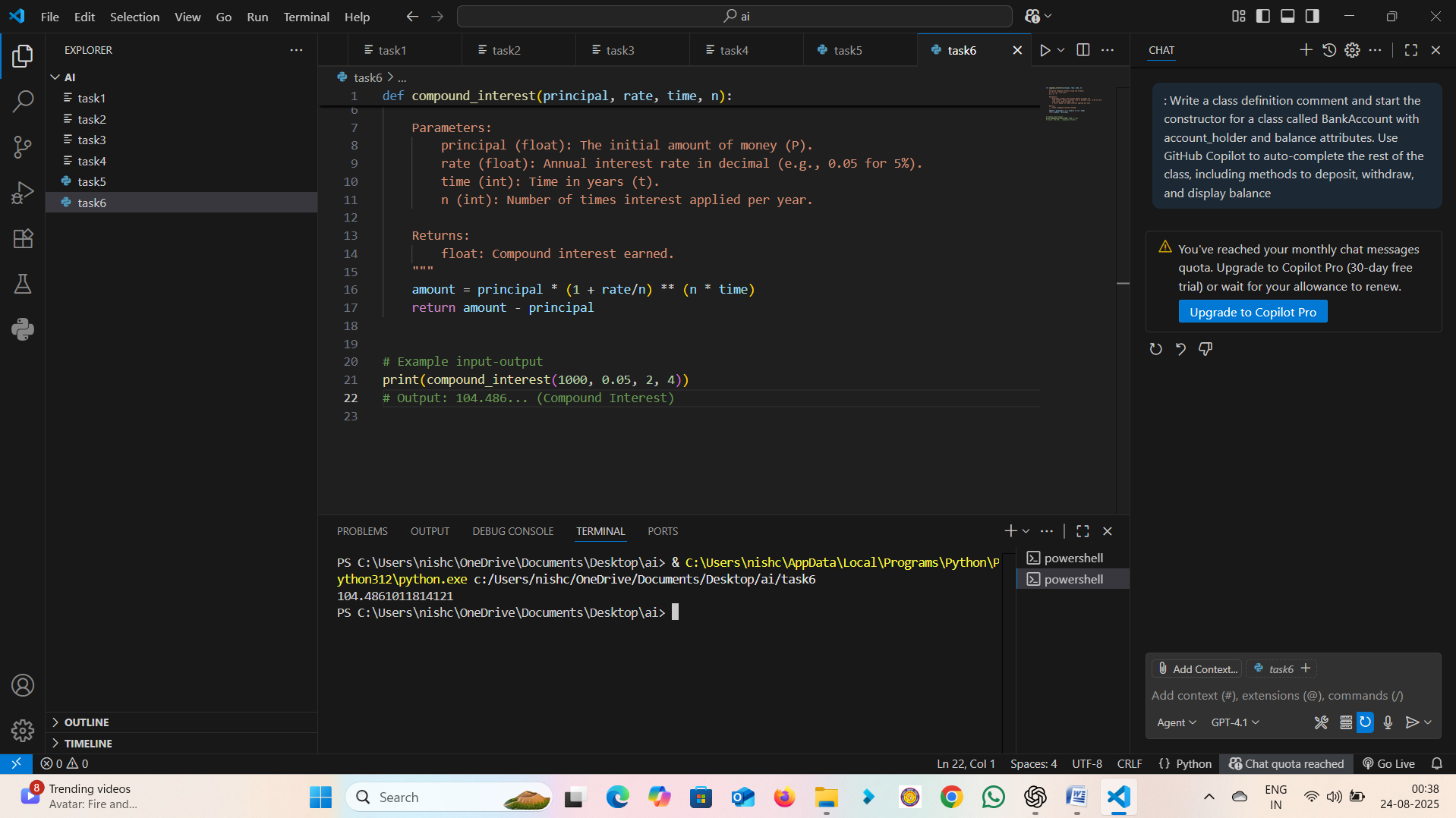
**Course Code**: 24CS002PC215.

## TASK 1: Compound Interest Function (Step by Step)

Prompt Used: write a function to calculate compound interest, starting with only the function name. Then add a docstring, then input-output example

**AI Generated Code**:

# Step 1  
def compound\_interest():  
 pass  
  
# Step 2 - with docstring  
def compound\_interest(principal, rate, time, n):  
 """  
 Calculate compound interest using formula:  
 A = P \* (1 + r/n)^(n\*t)  
 CI = A - P  
 """  
  
# Step 3 - with full logic and example  
def compound\_interest(principal, rate, time, n):  
 amount = principal \* (1 + rate/n)\*\*(n\*time)  
 return amount - principal  
  
print(compound\_interest(1000, 0.05, 2, 4)) # 104.486...

**Explanation:** The function is refined step by step: first just the name, then a docstring, and finally the full implementation. Formula ensures compound interest is computed correctly. 

**Output: Input**: P=1000, r=5%, t=2 years, n=4 → **Output**: 104.486 (compound interest)

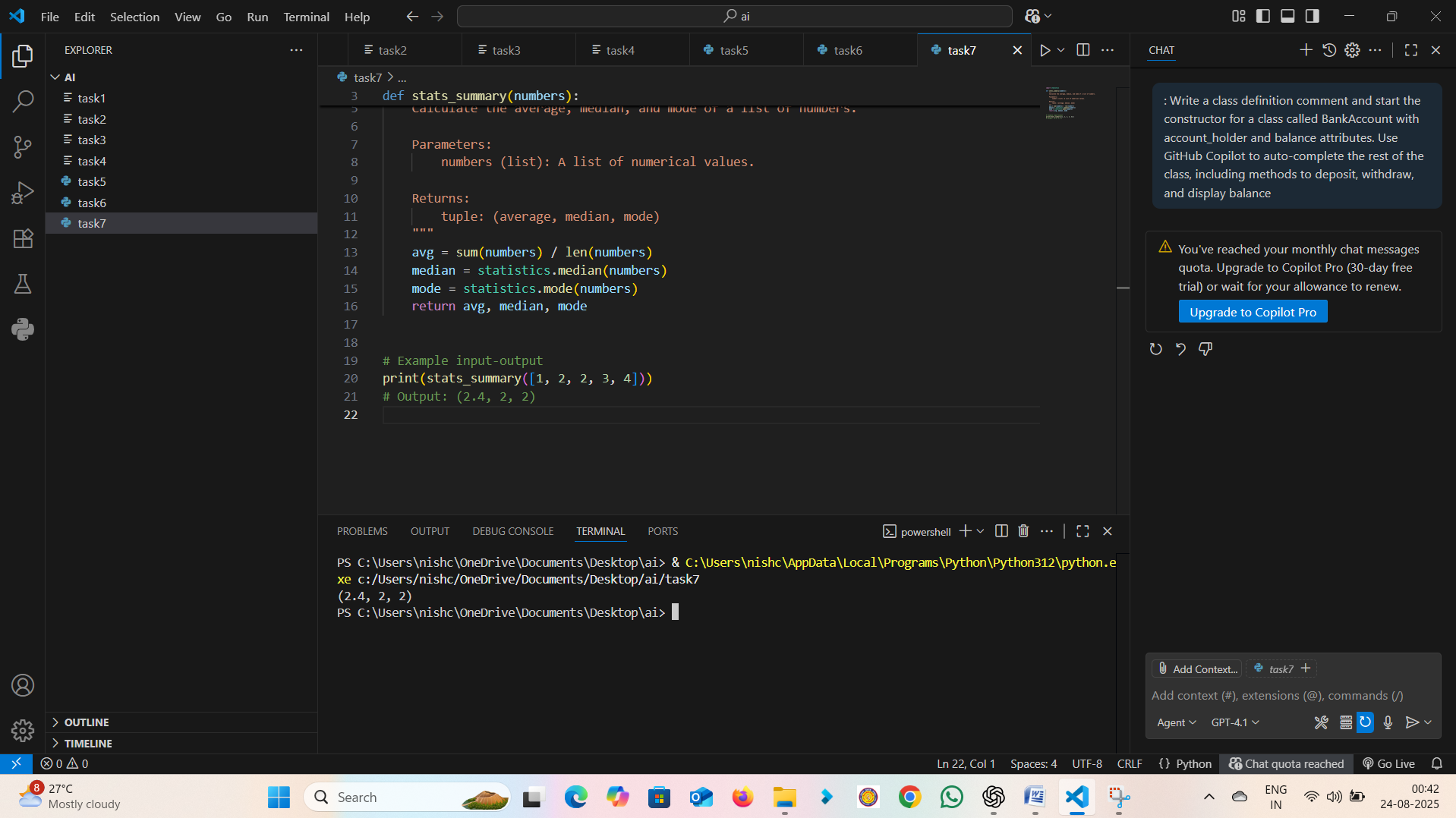
**Observation**: The task showed how step-by-step prompting helps in building a complete function. Starting with only the function name, then adding a docstring, and finally implementing the logic, made the AI produce accurate and clear code for calculating compound interest.

## Task 2: Refined Prompt for Average, Median, and Mode

Prompt Used: Write a function to calculate average, median, and mode of a list of numbers.

**AI Generated Code**:

import statistics  
  
def stats\_summary(numbers):  
 avg = sum(numbers)/len(numbers)  
 median = statistics.median(numbers)  
 mode = statistics.mode(numbers)  
 return avg, median, mode  
  
print(stats\_summary([1,2,2,3,4])) # (2.4, 2, 2)

**Explanation:** By refining the vague prompt, AI generates a function that calculates average, median, and mode. It uses Python's statistics library for median and mode, while average is manually computed. 

**Output: Input**: [1,2,2,3,4] → **Output**: Average=2.4, Median=2, Mode=2

**Observation:** A vague prompt produced poor results, but refining it led to a precise function that correctly calculates average, median, and mode. This highlights the importance of giving clear, specific prompts to obtain reliable code.

## Task 3: Few-shot Prompting for Convert to Binary

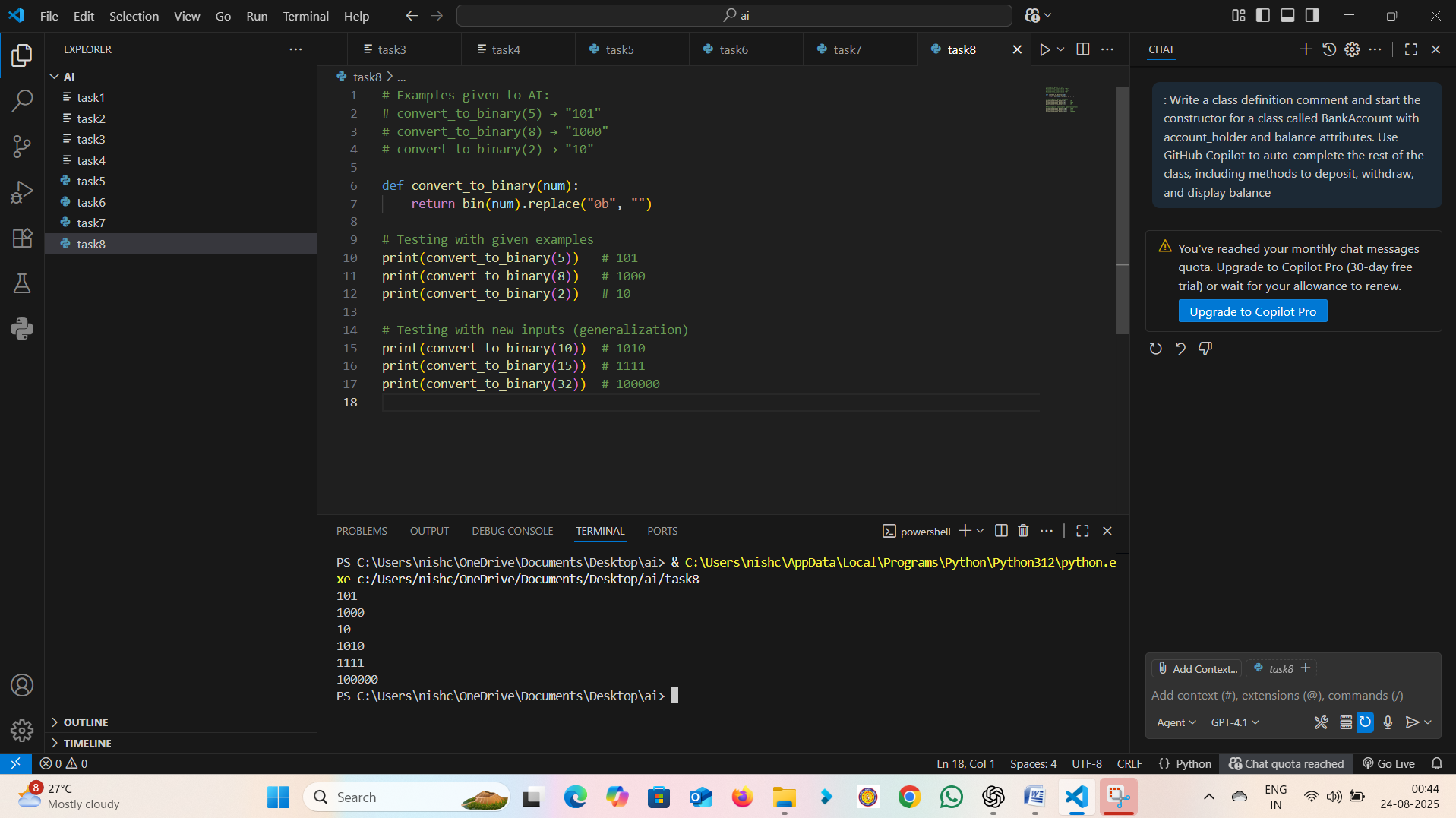
Prompt Used: Provide multiple examples of input-output to the AI for convert\_to\_binary(num) function. Observe how AI uses few-shot prompting to generalize.

Examples given: convert\_to\_binary(5) → '101', convert\_to\_binary(8) → '1000', convert\_to\_binary(2) → '10'

**AI Generated Code**:

def convert\_to\_binary(num):  
 return bin(num).replace("0b", "")  
  
print(convert\_to\_binary(5)) # 101  
print(convert\_to\_binary(8)) # 1000  
print(convert\_to\_binary(2)) # 10

**Explanation:** With multiple examples, the AI learns to generalize and uses Python's built-in bin() function for binary conversion, removing the '0b' prefix.



Output: Input: 5 → Output: 101  
Input: 8 → Output: 1000  
Input: 2 → Output: 10

Input:10->output:1010

Input:15->output:1111

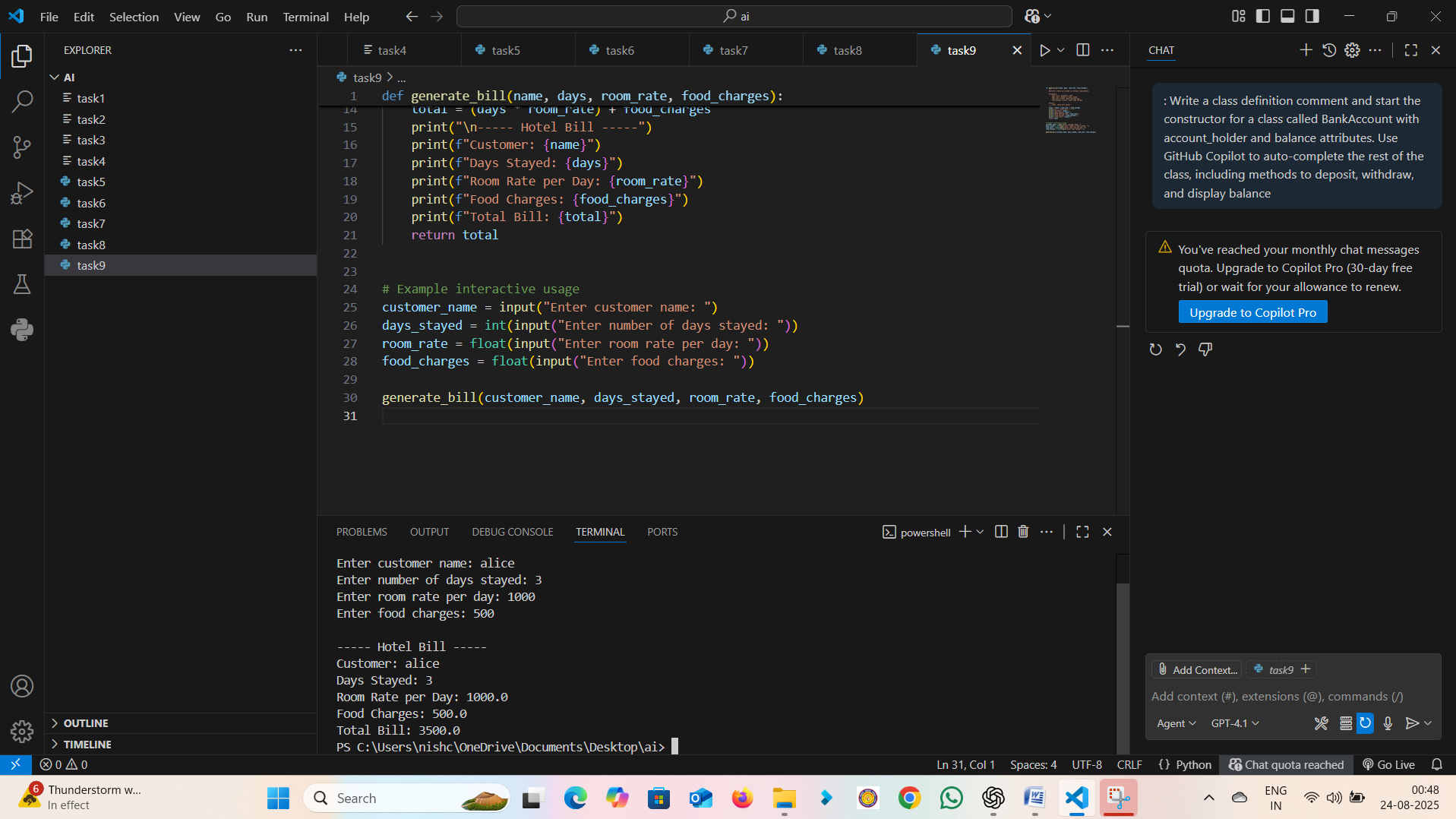
Input:32->output:100000

**Observation**: By providing examples, the AI learned the expected output format and generated correct code using Python’s built-in bin() function. Few-shot prompting improved accuracy and reduced ambiguity.

## Task 4: Hotel Billing User Interface Prompt: Create a user interface for a hotel to generate bill based on customer requirements.

**AI Generated Code:**

def generate\_bill(name, days, room\_rate, food\_charges):  
 total = (days \* room\_rate) + food\_charges  
 print("----- Hotel Bill -----")  
 print(f"Customer: {name}")  
 print(f"Days Stayed: {days}")  
 print(f"Room Rate: {room\_rate}")  
 print(f"Food Charges: {food\_charges}")  
 print(f"Total Bill: {total}")  
 return total  
  
generate\_bill("Alice", 3, 1000, 500)

**Explanation:** This function takes customer details and calculates total bill as (days \* room\_rate + food\_charges). It prints a formatted bill as a simple text-based UI. 

**Output:** Customer: Alice, Days=3, Room Rate=1000/day, Food=500 → Total Bill = 3500

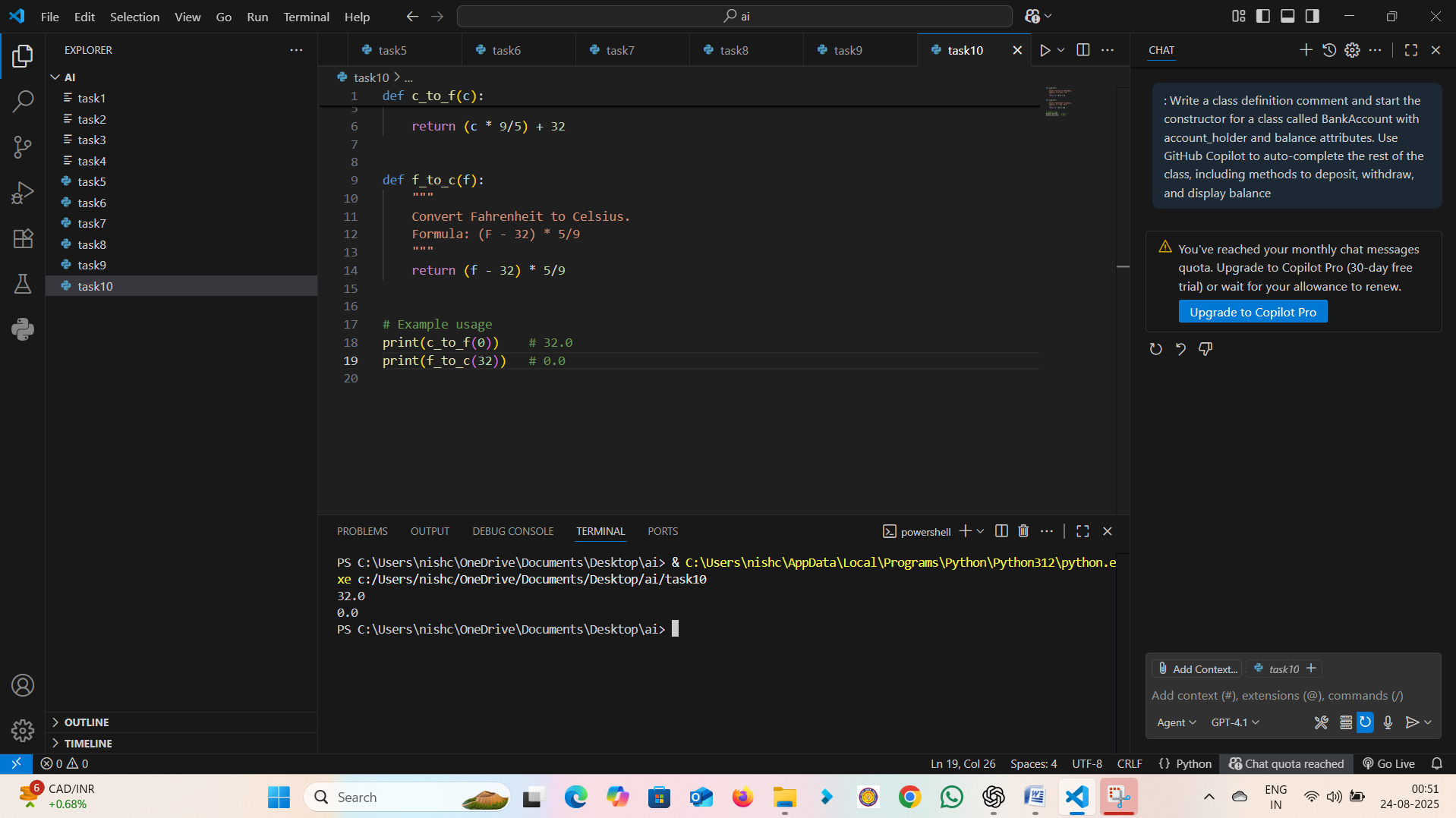
**Observation:** The AI generated a simple, text-based billing system that calculates total charges and prints a formatted bill. This shows how descriptive prompts can produce practical, user-friendly solutions tailored to real-world applications.

## Task 5: Prompt Specificity: Temperature Conversion

**Prompt:** 'Write a function to convert Celsius to Fahrenheit and Fahrenheit to Celsius with examples.'

**AI Generated Code:**

def c\_to\_f(c):  
 return (c \* 9/5) + 32  
  
def f\_to\_c(f):  
 return (f - 32) \* 5/9  
  
print(c\_to\_f(0)) # 32.0  
print(f\_to\_c(32)) # 0.0

**Explanation:** The vague prompt gave unclear results. By making the prompt specific, the AI generated accurate conversion functions for both Celsius→Fahrenheit and Fahrenheit→Celsius. 

Output: Input: 0°C → Output: 32°F  
Input: 32°F → Output: 0°C

**Observation:** The vague prompt gave incomplete or unclear results, while the specific prompt resulted in accurate conversion functions for Celsius and Fahrenheit. This demonstrates that prompt specificity directly affects the quality of AI-generated output.