AI ASSISTANCE CODING

ASSIGNMENT-3.5

NAME:RANGA SATYA NARAYANA

ROLL NO:2303A52501

BATCH-50

TASK-1:

Zero-Shot Prompting (Leap Year Check)

Write a zero-shot prompt to generate a Python function that checks

whether a given year is a leap year.

Week2 -

Task:

• Record the AI-generated code.

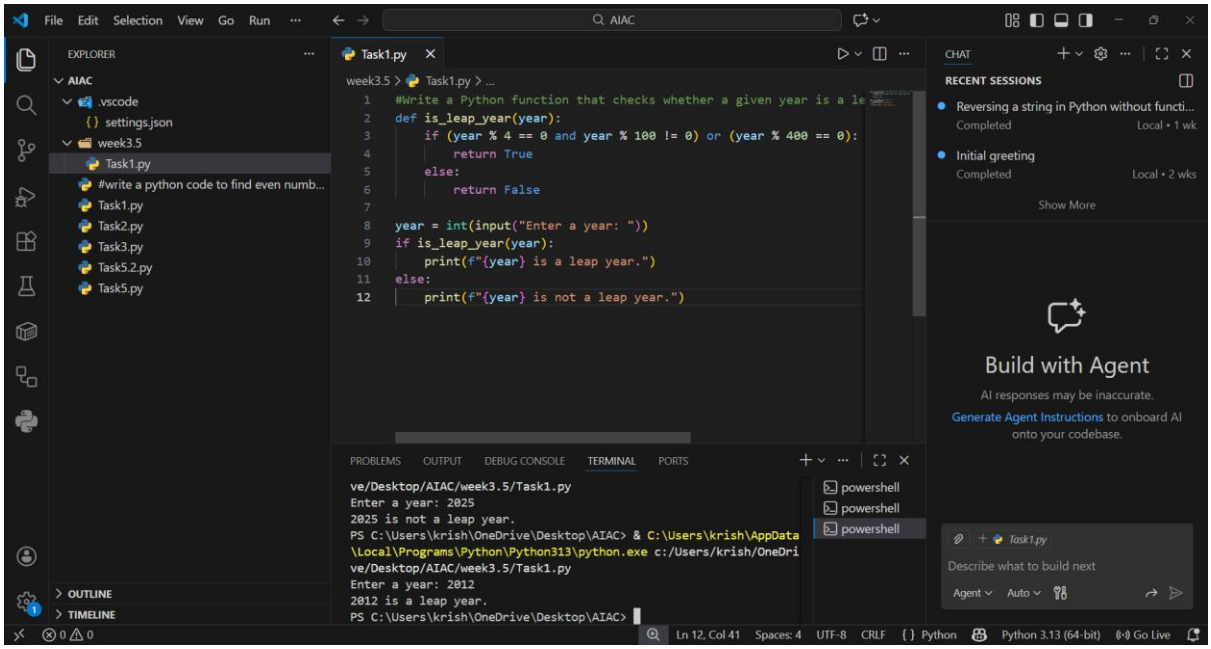
• Test with years like 1900, 2000, 2024.

• Identify logical flaws or missing conditions.

PROMPT:

“Write a Python function that checks whether a given year is a leap year or not by taking user input.”

OUTPUT:



EXPLANATION:

The initial zero-shot code only checked if a year was divisible by 4, which works for most cases but fails for century years like 1900. Leap years must follow the rule: divisible by 4, not divisible by 100 unless also divisible by 400. That's why 2000 is a leap year but 1900 is not. The corrected version adds these conditions to make the function logically complete.

Task-2:

One-Shot Prompting (GCD of Two Numbers)

Write a one-shot prompt with one example to generate a Python

function that finds the Greatest Common Divisor (GCD) of two numbers.

Example:

Input: 12, 18 → Output: 6

Task:

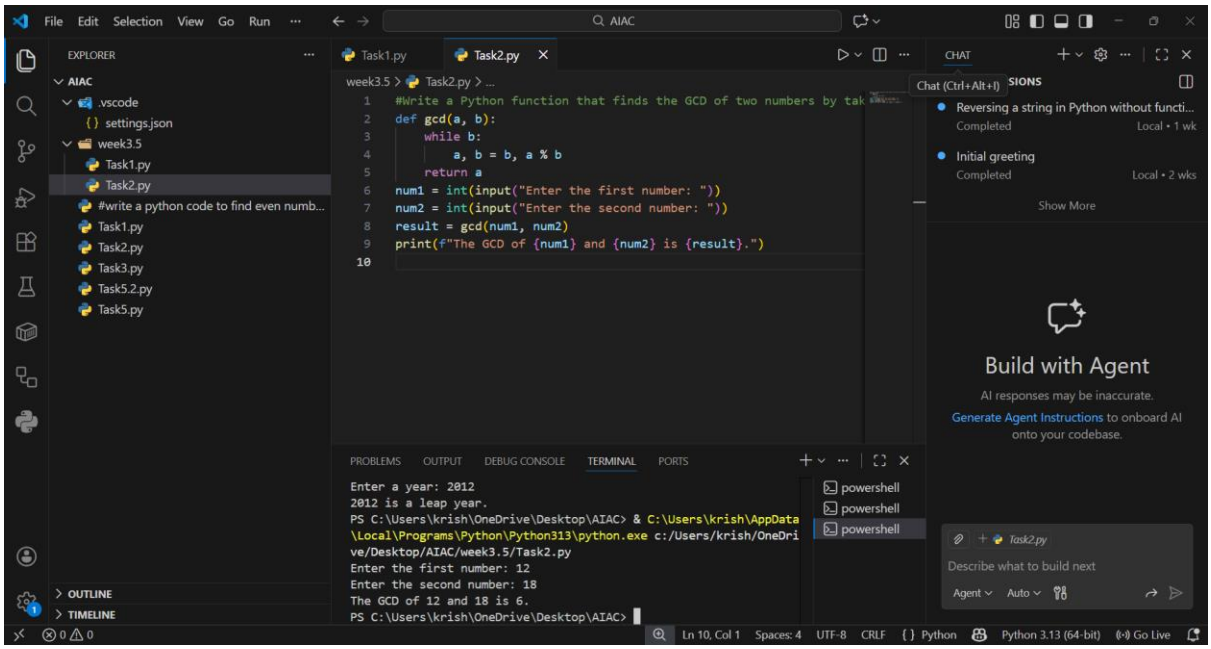
• Compare with a zero-shot solution.

• Analyze algorithm efficiency.

PROMPT:

“Write a Python function that finds the GCD of two numbers by taking user input.”

OUTPUT:



EXPLANATION:

The one-shot prompt guided the AI to use the Euclidean Algorithm, which repeatedly replaces numbers with their remainder until one becomes zero. This is much faster than checking all divisors manually. For example, with inputs 12 and 18, the algorithm quickly finds 6 as the GCD. The efficiency comes from reducing the problem size at each step.

Task-3:

Few-Shot Prompting (LCM Calculation)

Write a few-shot prompt with multiple examples to generate a Python

function that computes the Least Common Multiple (LCM).

Examples:

• Input: 4, 6 → Output: 12

• Input: 5, 10 → Output: 10

• Input: 7, 3 → Output: 21

Task:

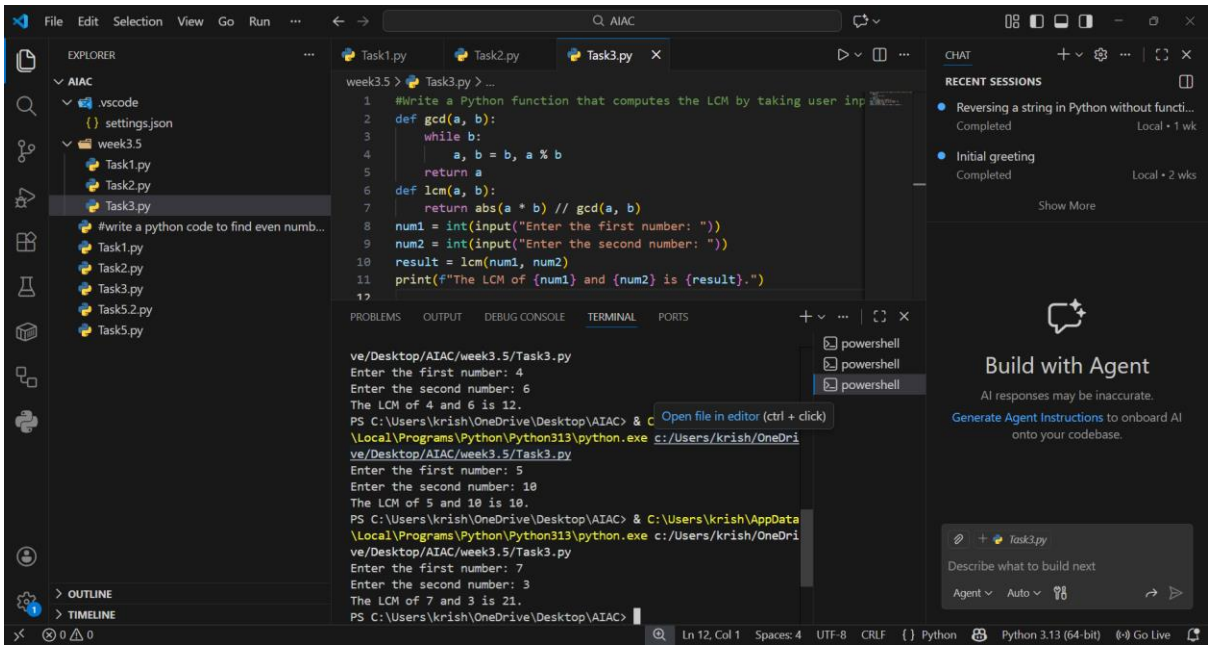
• Examine how examples guide formula selection.

• Test edge cases.

PROMPT:

“Write a Python function that computes the LCM by taking user input.Test edge cases.”

OUTPUT:



EXPLANATION:

With multiple examples, the AI recognized the formula for LCM: (a*b)//GCD(a,b). This avoids brute force searching for common multiples. For instance, LCM of 4 and 6 is 12, and of 7 and 3 is 21, which matches the examples. The use of examples ensures the AI applies the mathematical relationship rather than trial-and-error logic.

Task-4:

Zero-Shot Prompting (Binary to Decimal Conversion)

Write a zero-shot prompt to generate a Python function that converts a binary number to decimal.

Task:

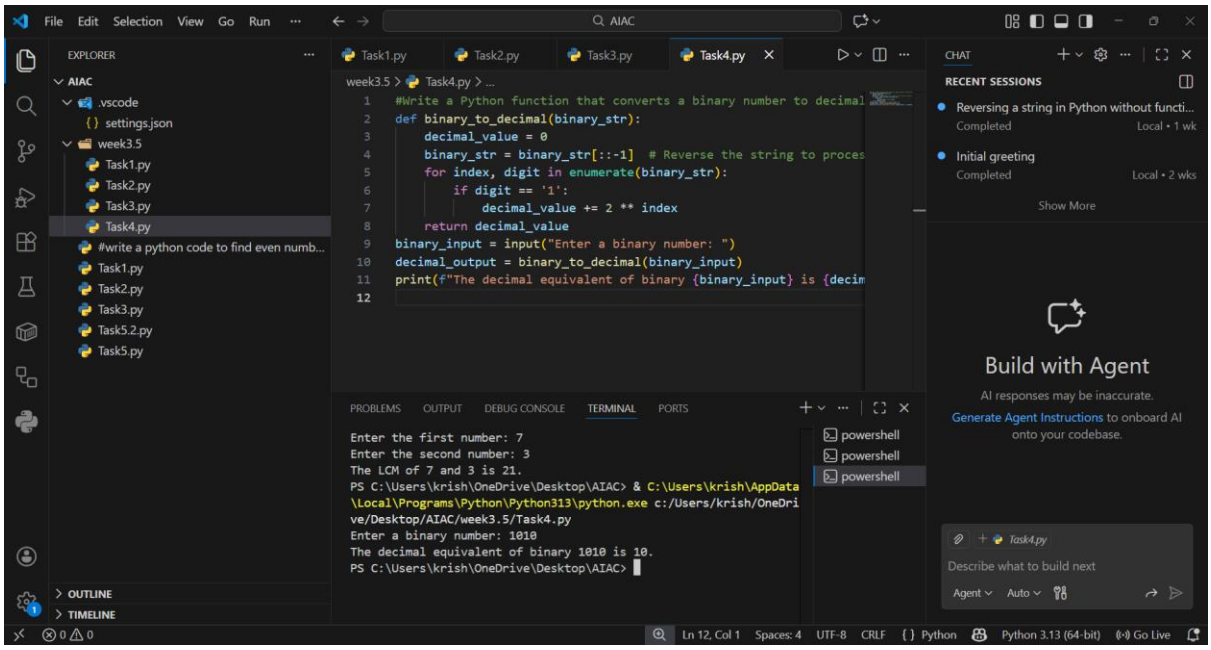
• Test with valid and invalid binary inputs.

• Identify missing validation logic.

PROMPT:

“Write a Python function that converts a binary number to decimal by taking user input.”

OUTPUT:



EXPLANATION:

The zero-shot code used Python’s built-in int(binary_str, 2) to convert binary strings to decimal. This works perfectly for valid inputs like "1010" → 10. However, it fails with invalid inputs like "1021" because no validation is included. Adding a check for only 0 and 1 characters makes the function more robust.

Task-5:

One-Shot Prompting (Decimal to Binary Conversion)

Write a one-shot prompt with an example to generate a Python function that converts a decimal number to binary.

Example:

Input: 10 → Output: 1010

Task:

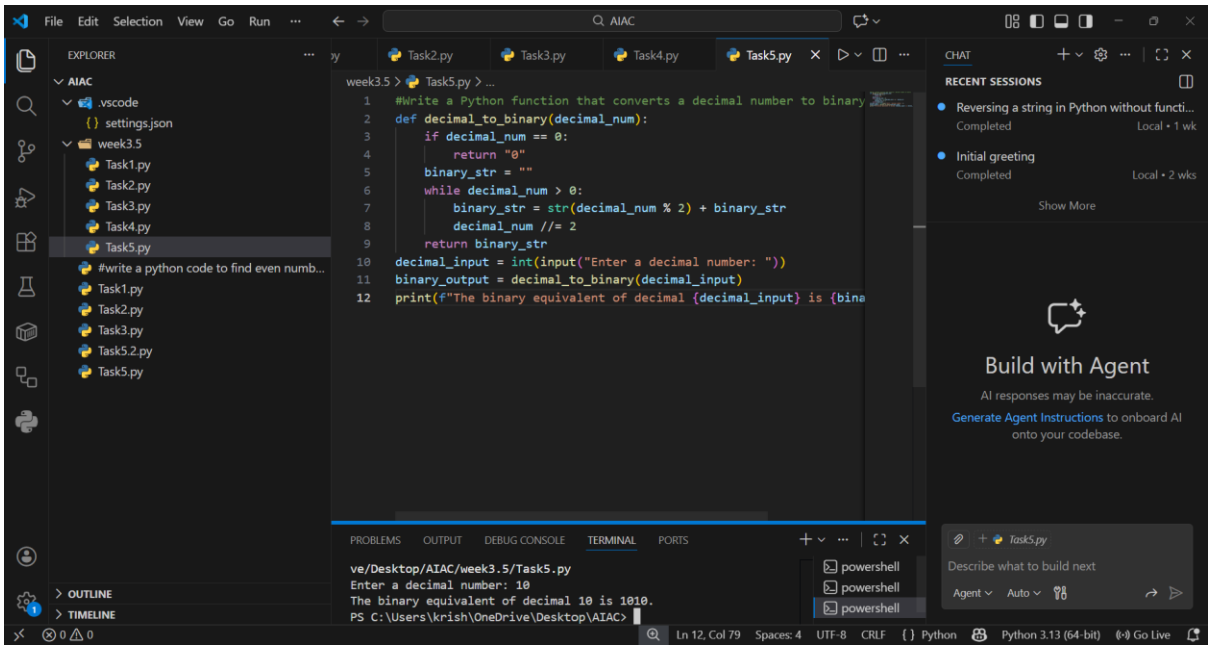
• Compare clarity with zero-shot output.

• Analyze handling of zero and negative numbers.

PROPMT:

“Write a Python function that converts a decimal number to binary.”

OUTPUT:



EXPLANATION:

The one-shot prompt led to using Python’s bin() function, which directly converts decimals to binary strings. For example, 10 becomes "1010". It handles zero correctly but produces odd results for negatives without extra

logic. By adding a condition for negative numbers, the function can return a proper binary string with a minus sign.

Task-6:

Few-Shot Prompting (Harshad Number Check)

Write a few-shot prompt to generate a Python function that checks

whether a number is a Harshad (Niven) number.

Examples:

• Input: 18 → Output: Harshad Number

• Input: 21 → Output: Harshad Number

• Input: 19 → Output: Not a Harshad Number

Task:

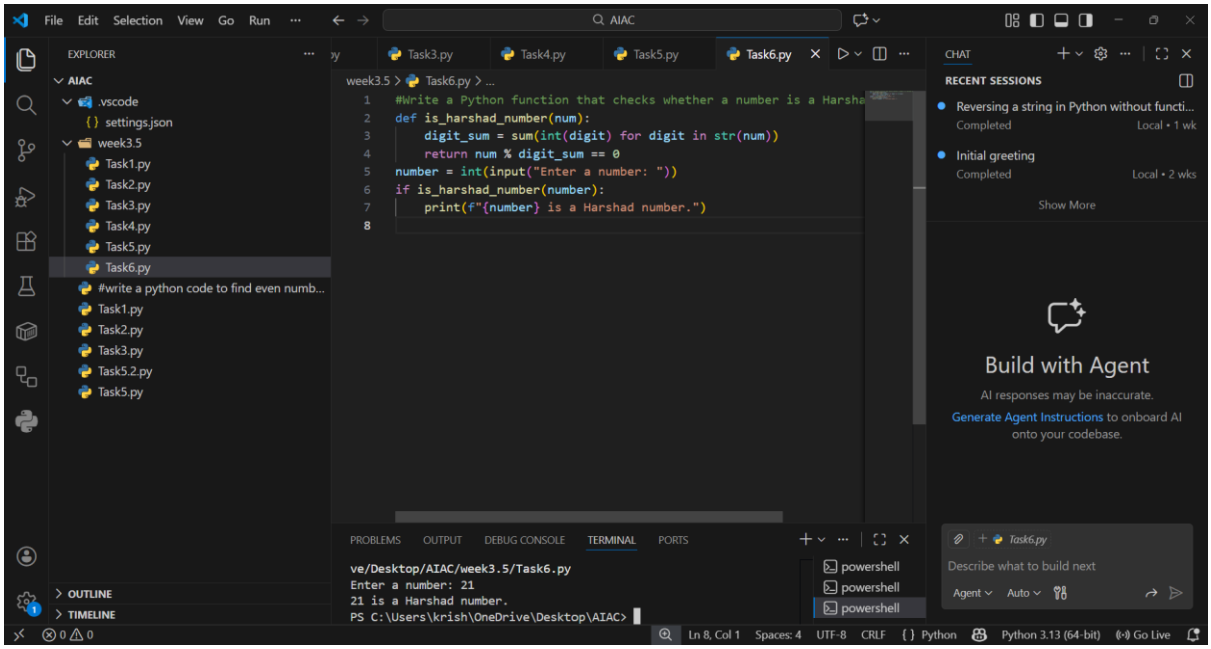
• Test boundary conditions.

• Evaluate robustness

PROMPT:

“Write a Python function that checks whether a number is a Harshad number. “

OUTPUT:



EXPLANATION:

The few-shot prompt taught the AI that a Harshad number is divisible by the sum of its digits. The code calculates the digit sum and checks divisibility, correctly identifying numbers like 18 and 21 as Harshad. However, input 0 causes a division by zero error, so adding a special case for zero improves robustness. This shows how examples guide the AI toward the right definition.