

# AI ASSISTANCE CODING

## ASSIGNMENT-3.5

**NAME: M HARIKRISHNA**

**ROLL NO:2203A52100**

**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 -

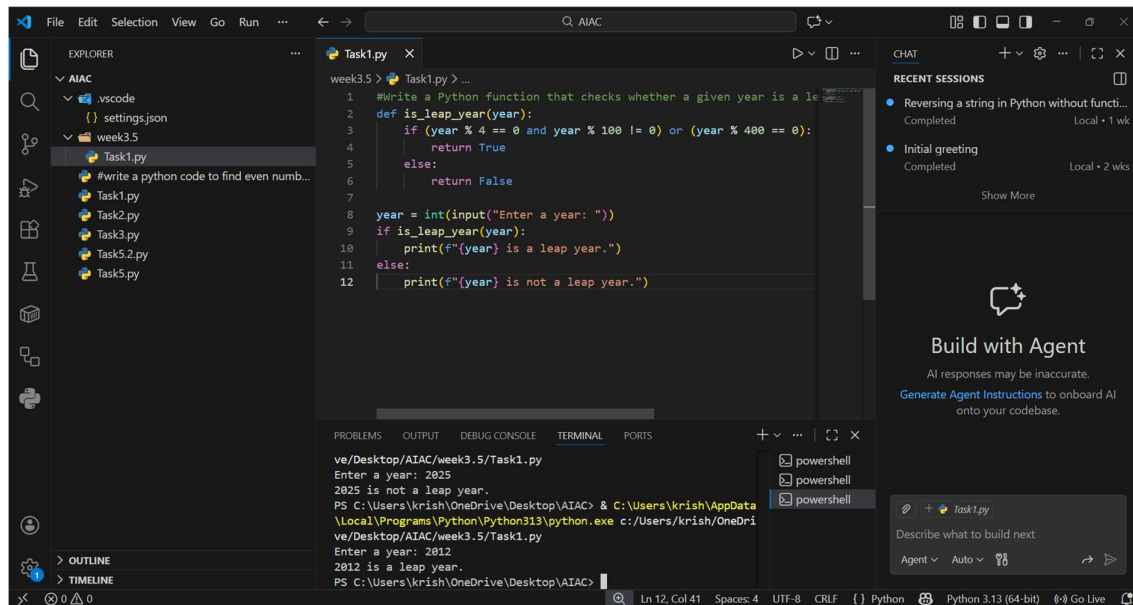
Task:

- 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:



The screenshot shows a VS Code editor with a file explorer on the left, a code editor in the center, and a terminal at the bottom. The code editor displays a Python script named `Task1.py` that defines a function `is_leap_year` and uses it to check if a user-entered year is a leap year. The terminal shows the execution of the script, with input and output for the years 2025 and 2012.

```
1 #Write a Python function that checks whether a given year is a leap year or not.
2 def is_leap_year(year):
3     if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
4         return True
5     else:
6         return False
7
8 year = int(input("Enter a year: "))
9 if is_leap_year(year):
10    print(f"{year} is a leap year.")
11 else:
12    print(f"{year} is not a leap year.")
```

Terminal Output:

```
ve/Desktop/AIAC/week3.5/Task1.py
Enter a year: 2025
2025 is not a leap year.
PS C:\Users\krish\OneDrive\Desktop\AIAC> & C:\Users\krish\AppData\Local\Programs\Python\Python313\python.exe c:/Users/krish/OneDrive/Desktop/AIAC/week3.5/Task1.py
Enter a year: 2012
2012 is a leap year.
PS C:\Users\krish\OneDrive\Desktop\AIAC>
```

## 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

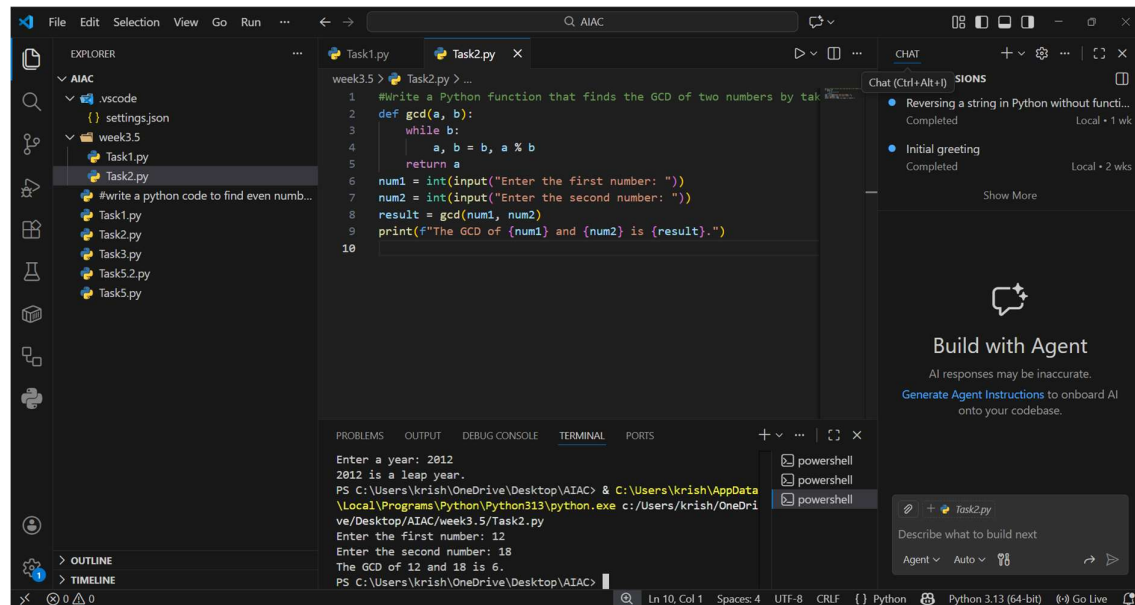
Task:

- 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:



The screenshot shows a Visual Studio Code editor with a file explorer on the left containing a project named 'AIAC' with files like 'Task1.py', 'Task2.py', 'Task3.py', 'Task4.py', 'Task5.py', and 'Task6.py'. The main editor window displays 'Task2.py' with the following Python code:

```
1 #Write a Python function that finds the GCD of two numbers by taking user input.
2 def gcd(a, b):
3     while b:
4         a, b = b, a % b
5     return a
6 num1 = int(input("Enter the first number: "))
7 num2 = int(input("Enter the second number: "))
8 result = gcd(num1, num2)
9 print(f"The GCD of {num1} and {num2} is {result}.")
10
```

Below the code editor is a terminal window showing the execution of the script:

```
PS C:\Users\krish\OneDrive\Desktop\AIAC> python Task2.py
Enter a year: 2012
2012 is a leap year.
PS C:\Users\krish\OneDrive\Desktop\AIAC> python Task2.py
Enter the first number: 12
Enter the second number: 18
The GCD of 12 and 18 is 6.
PS C:\Users\krish\OneDrive\Desktop\AIAC>
```

## 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:

The screenshot shows a VS Code editor with a file explorer on the left, a code editor in the center, and a terminal at the bottom. The file explorer shows a project named 'AIAC' with a subdirectory 'week3.5' containing files 'Task1.py', 'Task2.py', and 'Task3.py'. The code editor displays the content of 'Task3.py', which is a Python script for calculating the LCM of two numbers. The script defines a 'gcd' function using a while loop and a 'lcm' function using the formula  $\text{lcm}(a, b) = \frac{a * b}{\text{gcd}(a, b)}$ . It then takes user input for two numbers and prints the LCM. The terminal shows the execution of the script, with prompts for the first and second numbers and the resulting LCM values for the test cases (4, 6) and (5, 10).

```

1 #Write a Python function that computes the LCM by taking user input
2 def gcd(a, b):
3     while b:
4         a, b = b, a % b
5     return a
6 def lcm(a, b):
7     return abs(a * b) // gcd(a, b)
8 num1 = int(input("Enter the first number: "))
9 num2 = int(input("Enter the second number: "))
10 result = lcm(num1, num2)
11 print(f"The LCM of {num1} and {num2} is {result}.")
12
ve/Desktop/AIAC/week3.5/Task3.py
Enter the first number: 4
Enter the second number: 6
The LCM of 4 and 6 is 12.
PS C:\Users\krish\OneDrive\Desktop\AIAC> & C:\Users\krish\AppData\Local\Programs\Python\Python313\python.exe c:/Users/krish/OneDrive/Desktop/AIAC/week3.5/Task3.py
Enter the first number: 5
Enter the second number: 10
The LCM of 5 and 10 is 10.
PS C:\Users\krish\OneDrive\Desktop\AIAC> & C:\Users\krish\AppData\Local\Programs\Python\Python313\python.exe c:/Users/krish/OneDrive/Desktop/AIAC/week3.5/Task3.py
Enter the first number: 7
Enter the second number: 3
The LCM of 7 and 3 is 21.
PS C:\Users\krish\OneDrive\Desktop\AIAC>

```

## EXPLANATION:

With multiple examples, the AI recognized the formula for LCM:  $(a*b)//\text{GCD}(a,b)$ . 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.

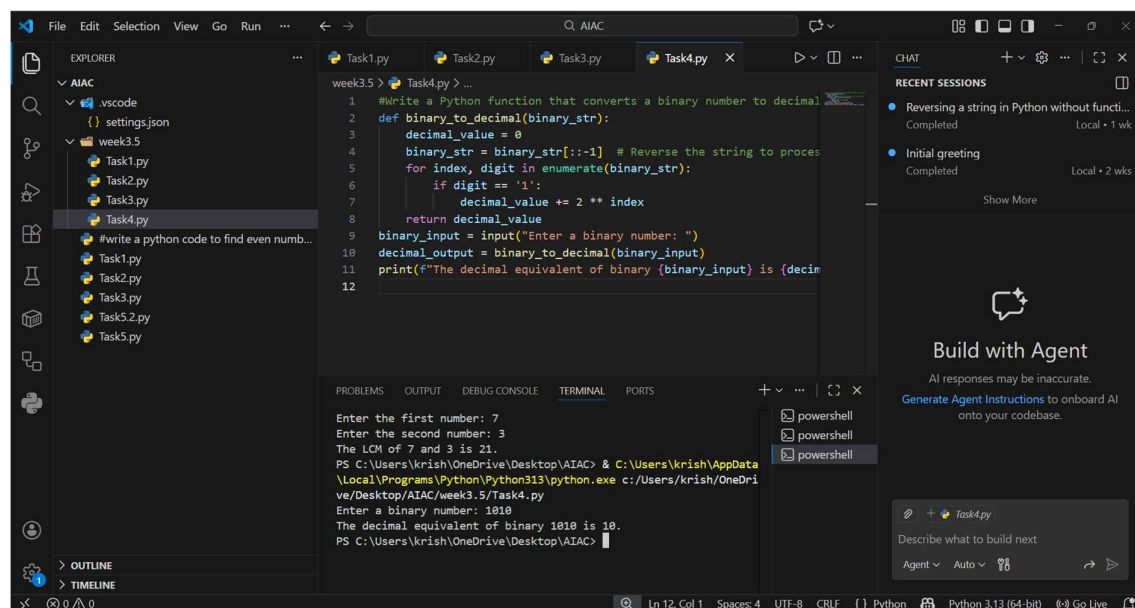
Task:

- 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:



The screenshot shows a Visual Studio Code editor with a file explorer on the left, a code editor in the center, and a terminal at the bottom. The code editor displays a Python script named `Task4.py` that defines a function `binary_to_decimal` to convert a binary string to a decimal value. The script uses `int(binary_str, 2)` for conversion. The terminal shows the execution of the script, where the user enters a binary number '1010' and the program outputs 'The decimal equivalent of binary 1010 is 10.'.

```
1 #Write a Python function that converts a binary number to decimal
2 def binary_to_decimal(binary_str):
3     decimal_value = 0
4     binary_str = binary_str[::-1] # Reverse the string to process
5     for index, digit in enumerate(binary_str):
6         if digit == '1':
7             decimal_value += 2 ** index
8     return decimal_value
9
10 binary_input = input("Enter a binary number: ")
11 decimal_output = binary_to_decimal(binary_input)
12 print(f"The decimal equivalent of binary {binary_input} is {decimal_output}")
```

Enter the first number: 7  
Enter the second number: 3  
The LCM of 7 and 3 is 21.  
PS C:\Users\krish\OneDrive\Desktop\AIAC> & C:\Users\krish\AppData\Local\Programs\Python\Python313\python.exe c:\Users\krish\OneDrive\Desktop\AIAC\week3.5\Task4.py  
Enter a binary number: 1010  
The decimal equivalent of binary 1010 is 10.  
PS C:\Users\krish\OneDrive\Desktop\AIAC>

### 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

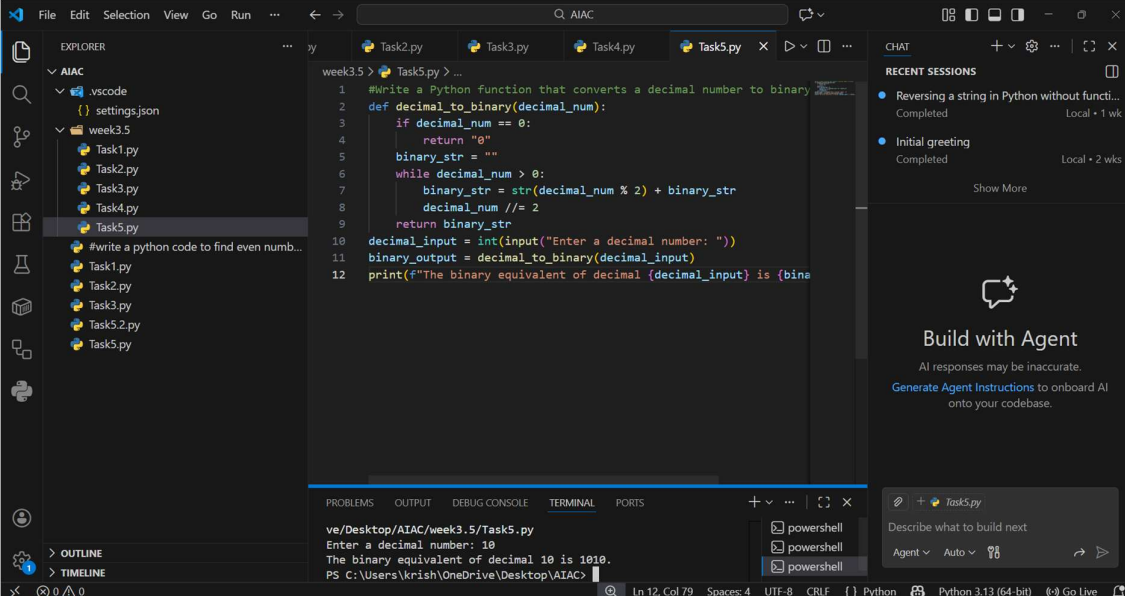
Task:

- Compare clarity with zero-shot output.
- Analyze handling of zero and negative numbers.

## PROMPT:

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

## OUTPUT:



```
1 #Write a Python function that converts a decimal number to binary
2 def decimal_to_binary(decimal_num):
3     if decimal_num == 0:
4         return "0"
5     binary_str = ""
6     while decimal_num > 0:
7         binary_str = str(decimal_num % 2) + binary_str
8         decimal_num //= 2
9     return binary_str
10 decimal_input = int(input("Enter a decimal number: "))
11 binary_output = decimal_to_binary(decimal_input)
12 print(f"The binary equivalent of decimal {decimal_input} is {binary_output}")
```

ve/Desktop/AIAC/week3.5/Task5.py  
Enter a decimal number: 10  
The binary equivalent of decimal 10 is 1010.  
PS C:\Users\krish\OneDrive\Desktop\AIAC>

## 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

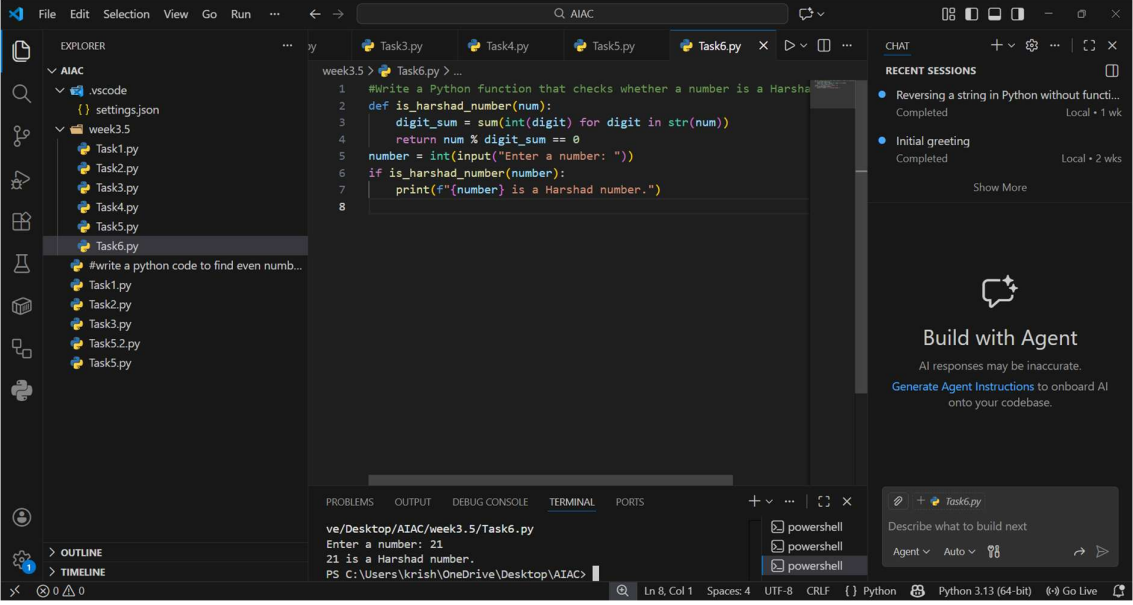
Task:

- Test boundary conditions.
- Evaluate robustness

#### **PROMPT:**

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

## OUTPUT:



The screenshot shows the Visual Studio Code interface. The Explorer panel on the left displays a file tree with a folder named 'AIAC' containing several Python files (Task1.py through Task6.py). The main editor window is open to 'Task6.py', which contains the following Python code:

```
1 #Write a Python function that checks whether a number is a Harshad number or not
2 def is_harshad_number(num):
3     digit_sum = sum(int(digit) for digit in str(num))
4     return num % digit_sum == 0
5
6 number = int(input("Enter a number: "))
7 if is_harshad_number(number):
8     print(f"{number} is a Harshad number.")
```

The bottom panel shows the 'TERMINAL' output, indicating the script is running in a PowerShell window at the path 've/Desktop/AIAC/week3.5/Task6.py'. The execution shows the prompt 'Enter a number: 21' followed by the output '21 is a Harshad number.'.

## 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.