

AI-Assisted Coding

Assignment-3.5

Name: C .Vaishnav Reddy

Batch:45

Question 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.

Task:

- Record the AI-generated code.
- Test with years like 1900, 2000, 2024.
- Identify logical flaws or missing conditions.

The screenshot shows the Visual Studio Code interface with the AI-Assisted Coding extension. The main editor displays a Python file named `assignment_3.5.py` containing a function `is_leap_year` and its usage. The function checks if a year is a leap year based on the Gregorian calendar rules. The chat window on the right shows the prompt and the generated code. The terminal at the bottom shows the execution of the code with test cases.

```
def is_leap_year(year: int) -> bool:
    """
    Args:
        year (int): The year to check.

    Returns:
        bool: True if the year is a leap year, False otherwise.
    """
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return True
    else:
        return False

# Example usage:
year = int(input("Enter a year: "))
if is_leap_year(year):
    print(f"{year} is a leap year.")
else:
    print(f"{year} is not a leap year.")
```

Chat Prompt: Write a Python function named `is_leap_year` that accepts an integer representing a year. The function should return `True` if the year is a leap year according to the Gregorian calendar rules (divisible by 4, but not by 100 unless also divisible by 400), and `False` otherwise. Include type hinting and a docstring.

Chat Response: `def is_leap_year(year: int) -> bool:
 """
 Determines if a given year is a leap year.
 A year is a leap year if it is:
 - Divisible by 4 and not by 100, or
 - Divisible by 400.
 Args:
 year (int): The year to check.
 Returns:
 bool: True if the year is a leap year, False otherwise.
 """
 if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
 return True
 else:
 return False`

Terminal Output: `PS D:\AI assisted coding> & C:\Users\vaish\AppData\Local\Programs\Python\Python314\python.exe "d:/AI assisted coding/assignment_3.5.py"
Enter a year: 1900
1900 is not a leap year.
PS D:\AI assisted coding> & C:\Users\vaish\AppData\Local\Programs\Python\Python314\python.exe "d:/AI assisted coding/assignment_3.5.py"
Enter a year: 2000
2000 is a leap year.
PS D:\AI assisted coding> & C:\Users\vaish\AppData\Local\Programs\Python\Python314\python.exe "d:/AI assisted coding/assignment_3.5.py"
Enter a year: 2024
2024 is a leap year.`

Question 2: One-Shot Prompting (GCD of Two Numbers)

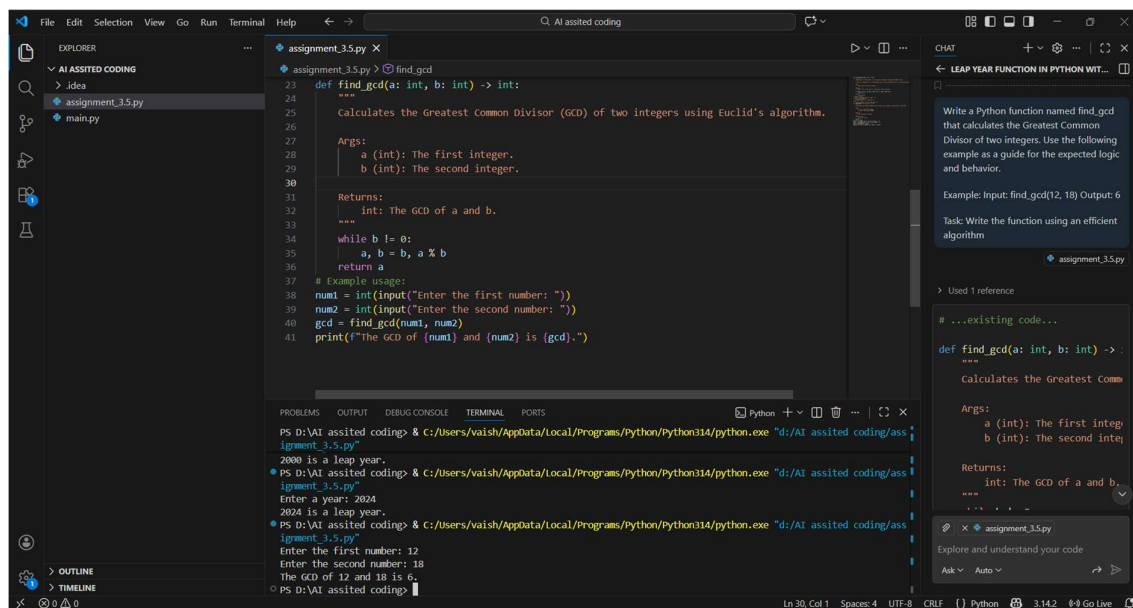
Write a one-shot prompt with one example to generate a Python function that finds the Greatest Common Divisor (GCD) of two numbers.

Example:

Input: 12, 18 → Output: 6

Task:

- Compare with a zero-shot solution.
- Analyze algorithm efficiency



The screenshot shows a VS Code editor with a Python file named `assignment_3.5.py`. The code defines a function `find_gcd` that calculates the Greatest Common Divisor (GCD) of two integers using Euclid's algorithm. The function takes two arguments, `a` and `b`, and returns the GCD. The code also includes a main function that prompts the user to enter two numbers and prints the GCD.

```
def find_gcd(a: int, b: int) -> int:
    """
    Calculates the Greatest Common Divisor (GCD) of two integers using Euclid's algorithm.

    Args:
        a (int): The first integer.
        b (int): The second integer.

    Returns:
        int: The GCD of a and b.
    """
    while b != 0:
        a, b = b, a % b
    return a

# Example usage:
num1 = int(input("Enter the first number: "))
num2 = int(input("Enter the second number: "))
gcd = find_gcd(num1, num2)
print(f"The GCD of {num1} and {num2} is {gcd}.")
```

The chat window on the right shows a prompt: "Write a Python function named `find_gcd` that calculates the Greatest Common Divisor of two integers. Use the following example as a guide for the expected logic and behavior. Example: Input: `find_gcd(12, 18)` Output: 6 Task: Write the function using an efficient algorithm". The chat also shows the function definition and the example usage.

Question 3: Few-Shot Prompting (LCM Calculation)

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

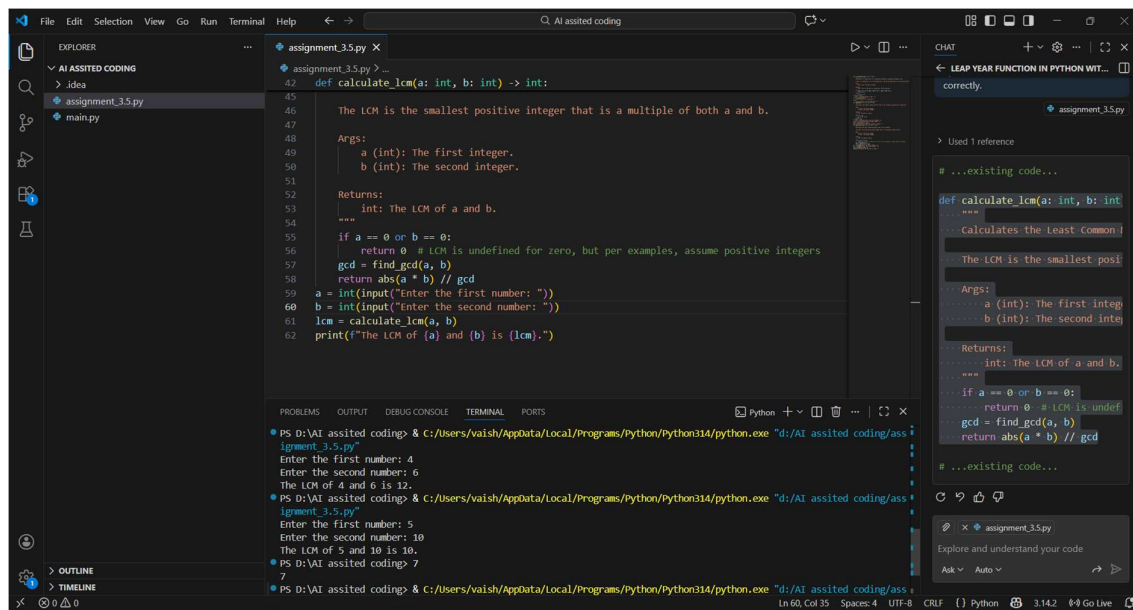
function that computes the Least Common Multiple (LCM).

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



The screenshot displays an IDE with a Python script named `assignment_3.5.py` and its execution output in the terminal. The script defines a `calculate_lcm` function that takes two integers `a` and `b` as input and returns their Least Common Multiple (LCM). The function uses a helper function `find_gcd` to calculate the Greatest Common Divisor (GCD) and then computes the LCM as `abs(a * b) // gcd`. The script prompts the user to enter two numbers and prints the result.

```
def calculate_lcm(a: int, b: int) -> int:
    """
    The LCM is the smallest positive integer that is a multiple of both a and b.

    Args:
        a (int): The first integer.
        b (int): The second integer.

    Returns:
        int: The LCM of a and b.
    """
    if a == 0 or b == 0:
        return 0 # LCM is undefined for zero, but per examples, assume positive integers
    gcd = find_gcd(a, b)
    return abs(a * b) // gcd

a = int(input("Enter the first number: "))
b = int(input("Enter the second number: "))
lcm = calculate_lcm(a, b)
print(f"The LCM of {a} and {b} is {lcm}.")
```

The terminal output shows the script being executed with the following inputs and outputs:

```
PS D:\AI assisted coding> C:\Users\vaish\AppData\Local\Programs\Python\Python314\python.exe "D:\AI assisted coding\assignment_3.5.py"
Enter the first number: 4
Enter the second number: 6
The LCM of 4 and 6 is 12.
PS D:\AI assisted coding> C:\Users\vaish\AppData\Local\Programs\Python\Python314\python.exe "D:\AI assisted coding\assignment_3.5.py"
Enter the first number: 5
Enter the second number: 10
The LCM of 5 and 10 is 10.
PS D:\AI assisted coding> C:\Users\vaish\AppData\Local\Programs\Python\Python314\python.exe "D:\AI assisted coding\assignment_3.5.py"
Enter the first number: 7
Enter the second number: 3
The LCM of 7 and 3 is 21.
```

Question 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.

The screenshot displays a VS Code workspace with three main components:

- Editor (assignment_3.5.py):** Contains a Python function `decimal_to_binary` that takes an integer and returns its binary string representation. It includes a `while` loop for conversion and a docstring. Example usage is shown at the bottom.
- Terminal:** Shows the execution of the script. It prompts for a decimal number (10), calculates its binary equivalent (1010), and prints the result.
- Chat Window:** Contains a one-shot prompt: "Write a Python function named decimal_to_binary that accepts an integer and returns its binary representation as a string. Use the example below to understand the expected output format. Example: Input: decimal_to_binary(10) Output: '1010'. Task: Ensure the function handles the integer 0 correctly and returns a clean string without Python's default prefix." Below the prompt, it shows the generated code snippet for the function.

Question 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

The screenshot displays a VS Code environment with three main components:

- Editor:** Contains the Python function `check_harshad_number`. The function takes an integer `num` and returns a boolean. It includes docstrings for arguments and returns, and a comment about Harshad numbers being positive integers. The implementation calculates the sum of digits and checks if the number is divisible by that sum.
- Terminal:** Shows the execution of the script. It prompts the user to enter a number and displays the output for inputs 18, 21, and 19, confirming that 18 and 21 are Harshad numbers while 19 is not.
- Chat:** A chat window on the right shows a few-shot prompt for the function. The prompt includes the function signature, a description of Harshad numbers, and three examples with their corresponding inputs and outputs.