

AI-Assisted Coding

Assignment-3.5

Name: MD .Abdul Layeeq

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 `def is_leap_year(year: int) -> bool:`. The function logic checks for leap years based on the Gregorian calendar rules: divisible by 4, but not by 100 unless also divisible by 400. The code includes docstrings, type hints, and example usage. The terminal shows the execution of the code with inputs 1900, 2000, and 2024, and the output indicating whether each is a leap year. The chat panel on the right shows the 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." and the AI-generated code.

```
1 def is_leap_year(year: int) -> bool:
2     """
3     Args:
4         year (int): The year to check.
5
6     Returns:
7         bool: True if the year is a leap year, False otherwise.
8     """
9     if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
10        return True
11    else:
12        return False
13
14 # Example usage:
15 year = int(input("Enter a year: "))
16 if is_leap_year(year):
17    print(f"{year} is a leap year.")
18 else:
19    print(f"{year} is not a leap year.")
```

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.
```

Chat Panel:

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.

Used 1 reference

```
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, but 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
```

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

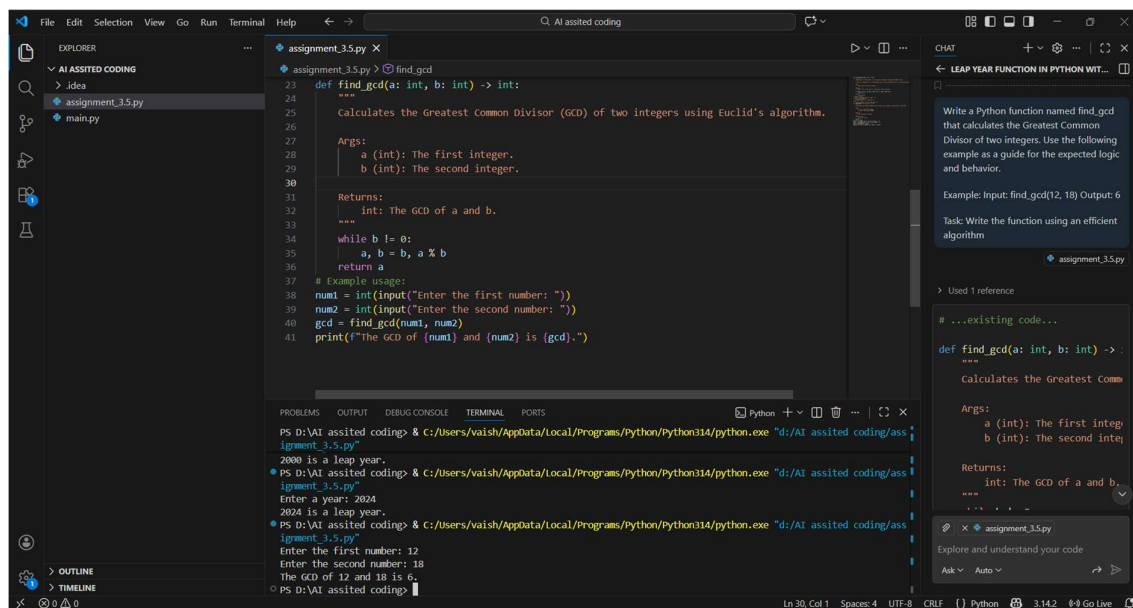
Write a one-shot prompt with one example to generate a Python function that finds the Greatest Common Divisor (GCD) of two numbers.

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 response shows the function definition and a comment: "Calculates the Greatest Common Divisor of two integers using Euclid's algorithm."

Question 3: Few-Shot Prompting (LCM Calculation)

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

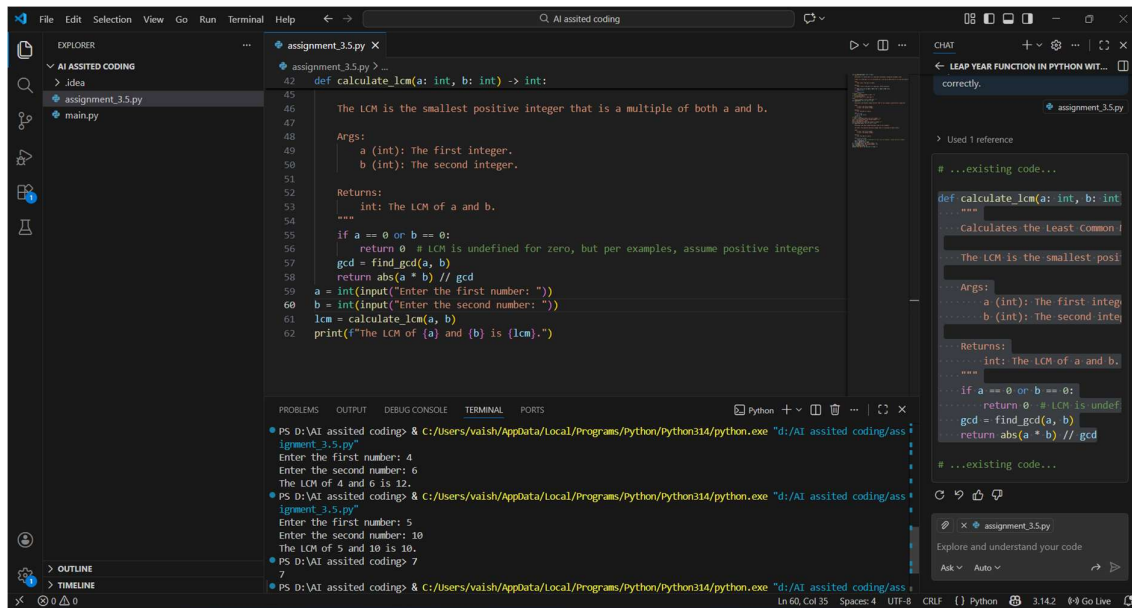
function that computes the Least Common Multiple (LCM).

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 and uses it to calculate the LCM for two input numbers. The terminal shows three test cases: (4, 6) resulting in 12, (5, 10) resulting in 10, and (7, 3) resulting in 21.

```
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}.")
```

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 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 environment 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 function code.

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 comments for arguments, returns, and example usage. The code is as follows:

```
def check_harshad_number(num: int) -> bool:
    """
    Args:
        num (int): The integer to check.

    Returns:
        bool: True if the number is a Harshad number, False otherwise.
    """
    if num <= 0:
        return False  # Harshad numbers are typically positive integers
    digit_sum = sum(int(digit) for digit in str(num))
    return num % digit_sum == 0

# Example usage:
num = int(input("Enter a number: "))
if check_harshad_number(num):
    print(f"{num} is a Harshad number.")
else:
    print(f"{num} is not a Harshad number.")
```
- Terminal:** Shows the execution of the script for three inputs:

```
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 number: 18
18 is a Harshad number.
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 number: 21
21 is a Harshad number.
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 number: 19
19 is not a Harshad number.
PS D:\AI assisted coding>
```
- Chat:** Displays a few-shot prompt for the function. The prompt includes the function signature, a description of Harshad numbers, and three examples with inputs and expected outputs:

```
<< LEAP YEAR FUNCTION IN PYTHON WIT...
> Used 1 reference

def(num: int) -> bool:
    """
    integer is a Harshad (Niven) number
    an integer that is divisible by
    integer to check.
    the number is a Harshad number, if
    Harshad numbers are typically p
    (digit) for digit in str(num))
    sum == 0

a number: ")
(num):
harshad number.")
t a Harshad number.")
```