AI ASSISTANT CODING

ASSIGNMENT-3.5

Ht.no: 2303A510F7

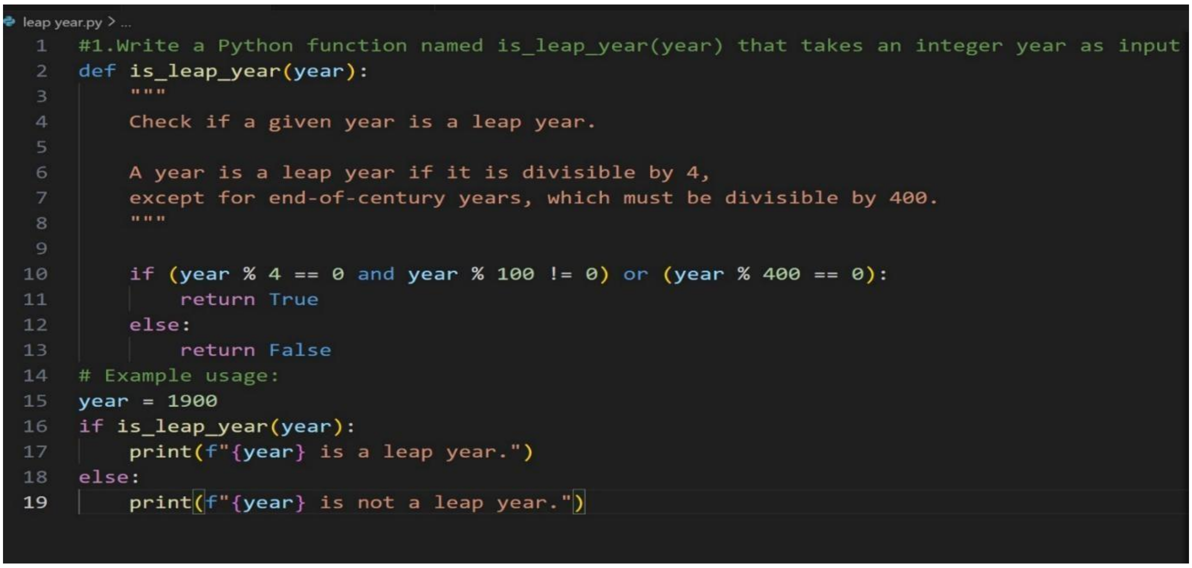
Batch:03

1. **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.
Week2 -
Task:
- Record the AI-generated code.
- Test with years like 1900, 2000, 2024.
- Identify logical flaws or missing conditions.

Prompt:

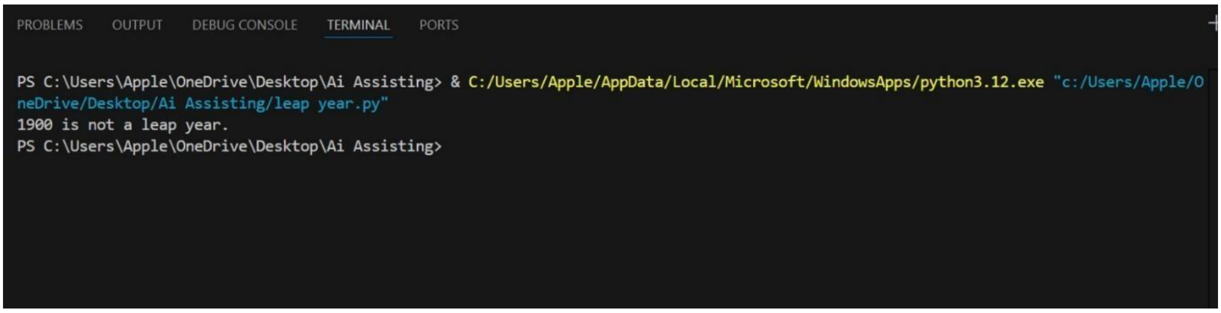
Write a Python function named is_leap_year(year) that takes an integer year as input and returns True if the year is a leap year and False otherwise.

Code:



```python
#1.Write a Python function named is_leap_year(year) that takes an integer year as input
def is_leap_year(year):
    """
    Check if a given year is a leap year.

    A year is a leap year if it is divisible by 4,
    except for end-of-century years, which must be divisible by 400.
    """

    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return True
    else:
        return False
# Example usage:
year = 1900
if is_leap_year(year):
    print(f"{year} is a leap year.")
else:
    print(f"{year} is not a leap year.")
```

Output:



```
PS C:\Users\Apple\OneDrive\Desktop\Ai Assisting> & C:/Users/Apple/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/Apple/O
neDrive/Desktop/Ai Assisting/leap year.py"
1900 is not a leap year.
PS C:\Users\Apple\OneDrive\Desktop\Ai Assisting>
```

Code Explanation:

The function is_leap_year(year) is used to determine whether a given year is a leap year based on the standard Gregorian calendar rules. It first checks if the year is divisible by 4 but not divisible by 100, which covers most common leap years. It also checks whether the year is divisible by 400, which is necessary for century years to be considered leap years. If either of these conditions is satisfied, the function returns True, indicating that the year is a leap year; otherwise, it returns False. In the example usage, the year 1900 is tested. Although 1900 is divisible by 4 and 100, it is not divisible by 400, so the function correctly identifies it as not a leap year and prints the corresponding message.

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

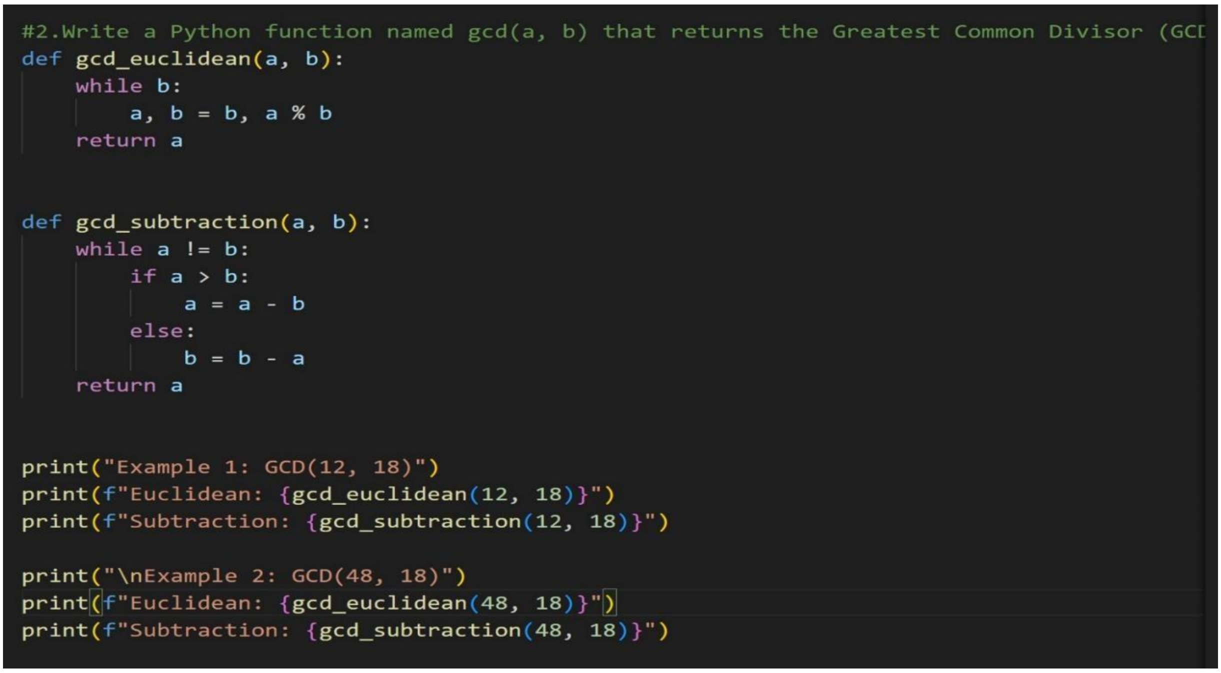
Task:

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

Prompt:

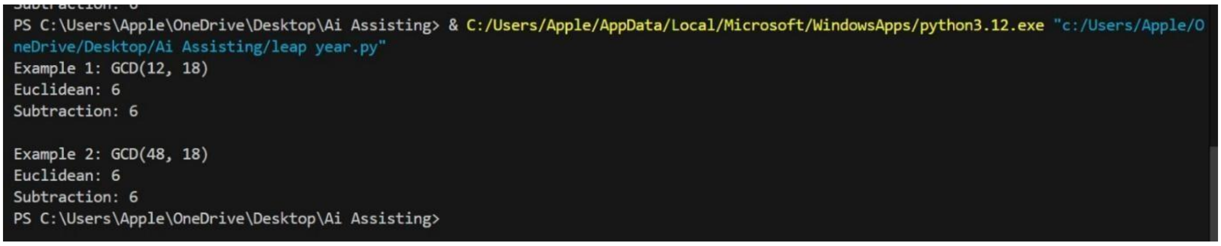
Write a Python function named gcd(a, b) that returns the Greatest Common Divisor (GCD) of two positive integers.

Code:



```python
#2.Write a Python function named gcd(a, b) that returns the Greatest Common Divisor (GC
def gcd_euclidean(a, b):
    while b:
        a, b = b, a % b
    return a


def gcd_subtraction(a, b):
    while a != b:
        if a > b:
            a = a - b
        else:
            b = b - a
    return a


print("Example 1: GCD(12, 18)")
print(f"Euclidean: {gcd_euclidean(12, 18)}")
print(f"Subtraction: {gcd_subtraction(12, 18)}")

print("\nExample 2: GCD(48, 18)")
print(f"Euclidean: {gcd_euclidean(48, 18)}")
print(f"Subtraction: {gcd_subtraction(48, 18)}")
```

Output:



```
PS C:\Users\Apple\OneDrive\Desktop\Ai Assisting> & C:/Users/Apple/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/Apple/O
neDrive/Desktop/Ai Assisting/leap year.py"
Example 1: GCD(12, 18)
Euclidean: 6
Subtraction: 6

Example 2: GCD(48, 18)
Euclidean: 6
Subtraction: 6
PS C:\Users\Apple\OneDrive\Desktop\Ai Assisting>
```

Code Explanation:

The program computes the Greatest Common Divisor (GCD) of two numbers using two approaches: the Euclidean method and the subtraction method. The Euclidean algorithm repeatedly replaces the numbers using the remainder operation until the remainder becomes zero, yielding the GCD efficiently. The subtraction method repeatedly subtracts the smaller number from the larger one until both values become equal. Both methods correctly produce the same GCD for the given examples. However, the Euclidean algorithm is significantly faster and more efficient than the subtraction approach, especially for large inputs.

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

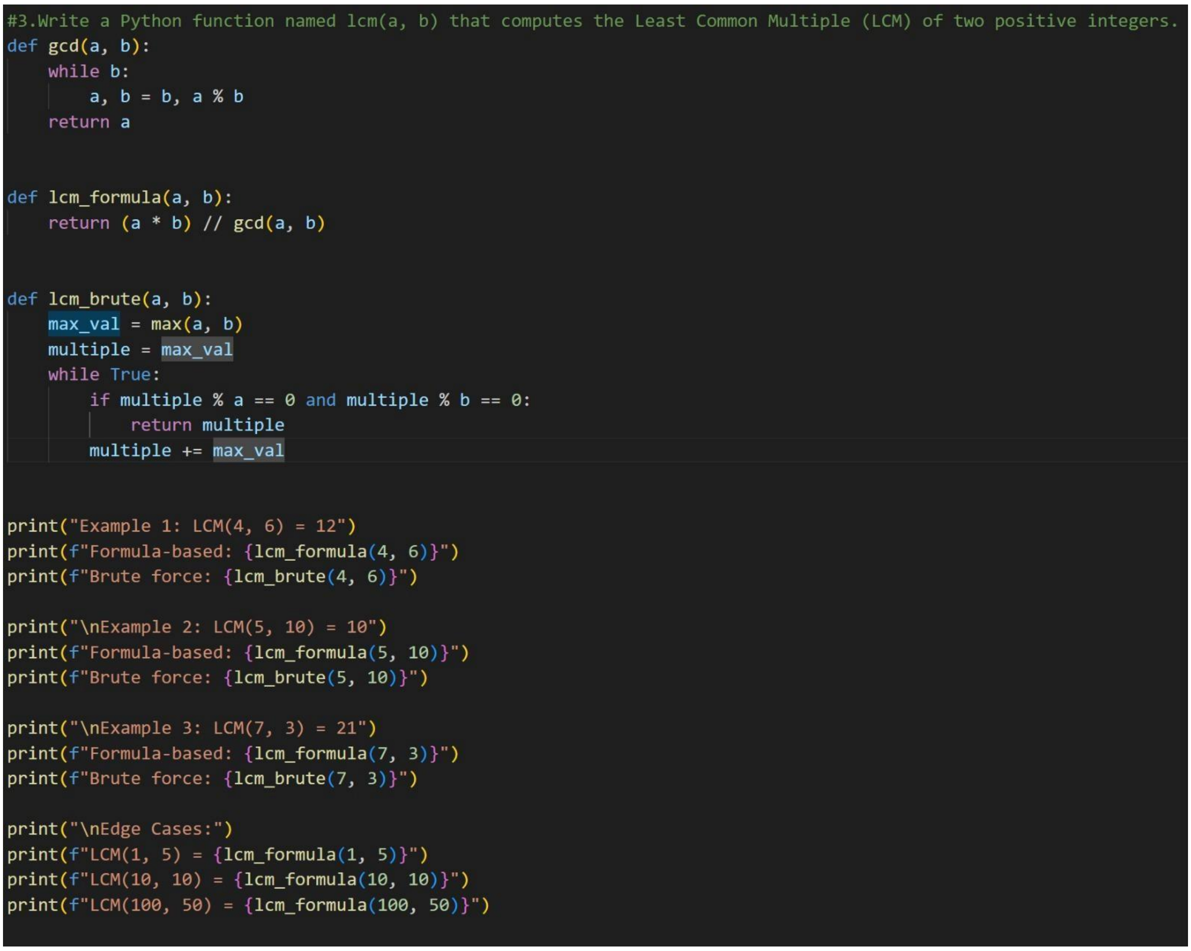
Task:

- Examine how examples guide formula selection.
- Test edge cases.

Prompt:

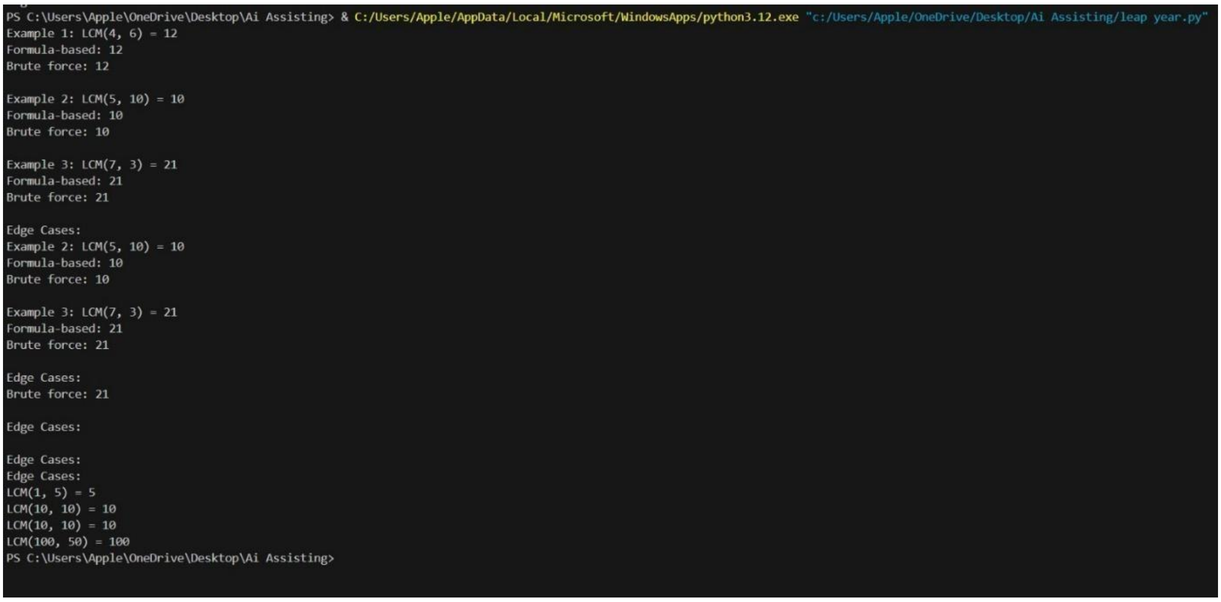
Write a Python function named lcm(a, b) that computes the Least Common Multiple (LCM) of two positive integers.

Code:



```python
#3.Write a Python function named lcm(a, b) that computes the Least Common Multiple (LCM) of two positive integers.
def gcd(a, b):
    while b:
        a, b = b, a % b
    return a


def lcm_formula(a, b):
    return (a * b) // gcd(a, b)


def lcm_brute(a, b):
    max_val = max(a, b)
    multiple = max_val
    while True:
        if multiple % a == 0 and multiple % b == 0:
            return multiple
        multiple += max_val


print("Example 1: LCM(4, 6) = 12")
print(f"Formula-based: {lcm_formula(4, 6)}")
print(f"Brute force: {lcm_brute(4, 6)}")

print("\nExample 2: LCM(5, 10) = 10")
print(f"Formula-based: {lcm_formula(5, 10)}")
print(f"Brute force: {lcm_brute(5, 10)}")

print("\nExample 3: LCM(7, 3) = 21")
print(f"Formula-based: {lcm_formula(7, 3)}")
print(f"Brute force: {lcm_brute(7, 3)}")

print("\nEdge Cases:")
print(f"LCM(1, 5) = {lcm_formula(1, 5)}")
print(f"LCM(10, 10) = {lcm_formula(10, 10)}")
print(f"LCM(100, 50) = {lcm_formula(100, 50)}")
```

Output:



```
PS C:\Users\Apple\OneDrive\Desktop\Ai Assisting> & C:/Users/Apple/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/Apple/OneDrive/Desktop/Ai Assisting/leap year.py"
Example 1: LCM(4, 6) = 12
Formula-based: 12
Brute force: 12

Example 2: LCM(5, 10) = 10
Formula-based: 10
Brute force: 10

Example 3: LCM(7, 3) = 21
Formula-based: 21
Brute force: 21

Edge Cases:
Example 2: LCM(5, 10) = 10
Formula-based: 10
Brute force: 10

Example 3: LCM(7, 3) = 21
Formula-based: 21
Brute force: 21

Edge Cases:
Brute force: 21

Edge Cases:

Edge Cases:
Edge Cases:
LCM(1, 5) = 5
LCM(10, 10) = 10
LCM(10, 10) = 10
LCM(100, 50) = 100
PS C:\Users\Apple\OneDrive\Desktop\Ai Assisting>
```

Code Explanation:

This code demonstrates how to find the Least Common Multiple (LCM) of two numbers using both an optimized and a basic approach. First, the gcd function calculates the Greatest Common Divisor using the Euclidean algorithm. The lcm_formula function then uses this GCD value to compute the LCM efficiently with a direct mathematical formula. In contrast, the lcm_brute function repeatedly checks multiples of the larger number until it finds one divisible by both numbers. The examples and edge cases confirm that both methods produce correct results. However, the formula-based method is faster and more suitable for larger numbers.

Question 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.

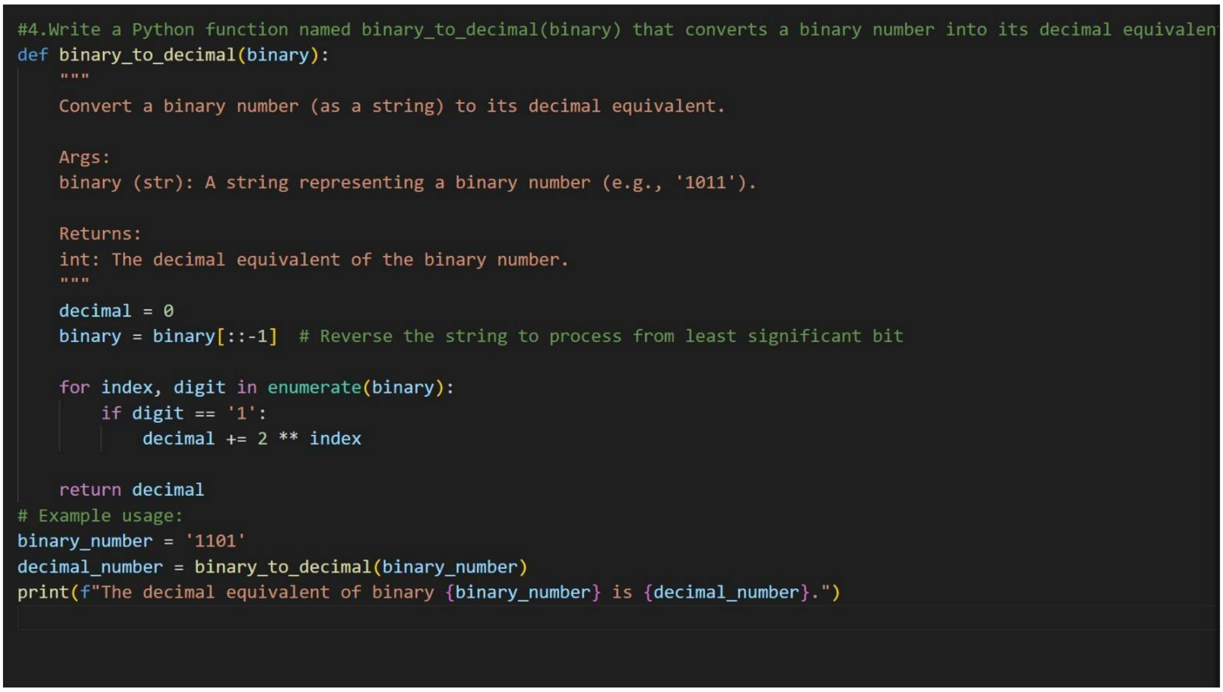
Task:

- Test with valid and invalid binary inputs.
- Identify missing validation logic.

Prompt:

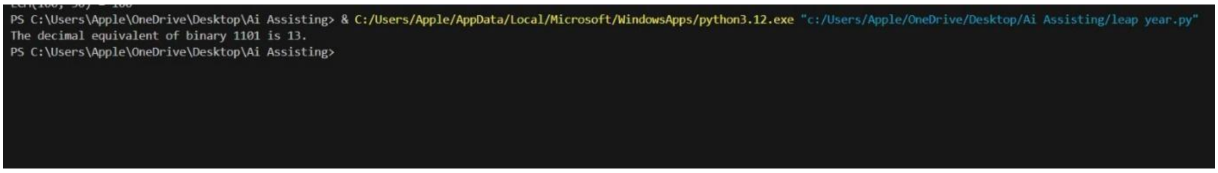
Write a Python function named binary_to_decimal(binary) that converts a binary number into its decimal equivalent.

Code:



```python
#4.Write a Python function named binary_to_decimal(binary) that converts a binary number into its decimal equivalen
def binary_to_decimal(binary):
    """
    Convert a binary number (as a string) to its decimal equivalent.

    Args:
    binary (str): A string representing a binary number (e.g., '1011').

    Returns:
    int: The decimal equivalent of the binary number.
    """
    decimal = 0
    binary = binary[::-1]  # Reverse the string to process from least significant bit

    for index, digit in enumerate(binary):
        if digit == '1':
            decimal += 2 ** index

    return decimal
# Example usage:
binary_number = '1101'
decimal_number = binary_to_decimal(binary_number)
print(f"The decimal equivalent of binary {binary_number} is {decimal_number}.")
```

Output:



```
PS C:\Users\Apple\OneDrive\Desktop\Ai Assisting> & C:/Users/Apple/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/Apple/OneDrive/Desktop/Ai Assisting/leap year.py"
The decimal equivalent of binary 1101 is 13.
PS C:\Users\Apple\OneDrive\Desktop\Ai Assisting>
```

Code Explanation:

The function binary_to_decimal converts a binary number given as a string into its decimal equivalent. It first reverses the binary string so that each digit can be processed from the least significant bit. Using a loop with enumerate, it checks each digit and adds the corresponding power of 2 when the digit is 1. The variable decimal accumulates the total decimal value during this process. Finally, the function returns the computed decimal number, which is printed for the example input.

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

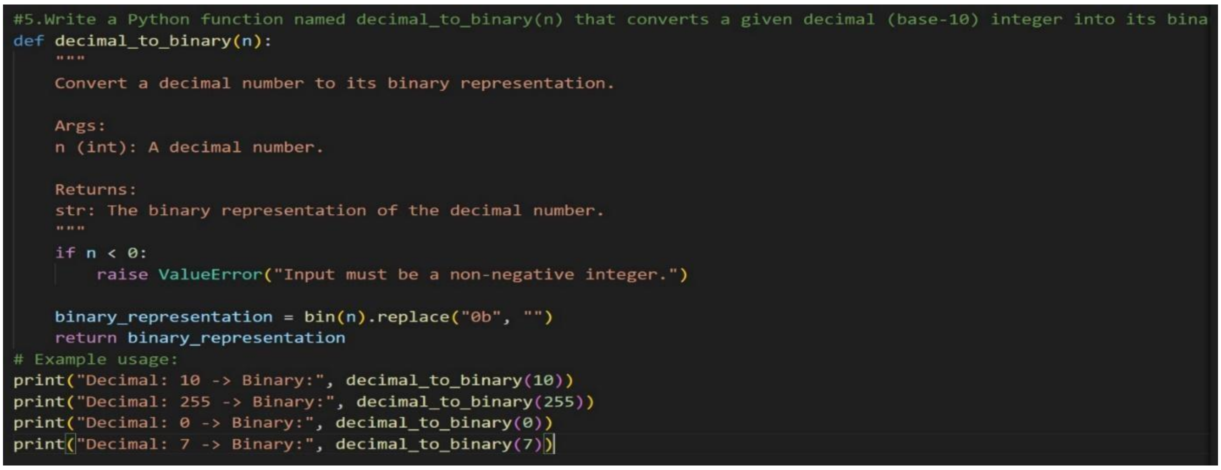
Task:

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

Prompt:

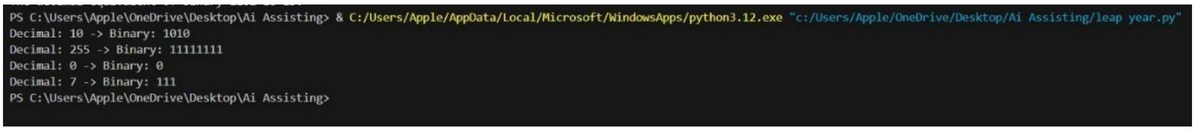
Write a Python function named decimal_to_binary(n) that converts a given decimal (base-10) integer into its binary (base-2) representation and returns it as a string.

Code:



```python
#5.Write a Python function named decimal_to_binary(n) that converts a given decimal (base-10) integer into its bina
def decimal_to_binary(n):
    """
    Convert a decimal number to its binary representation.

    Args:
    n (int): A decimal number.

    Returns:
    str: The binary representation of the decimal number.
    """
    if n < 0:
        raise ValueError("Input must be a non-negative integer.")

    binary_representation = bin(n).replace("0b", "")
    return binary_representation
# Example usage:
print("Decimal: 10 -> Binary:", decimal_to_binary(10))
print("Decimal: 255 -> Binary:", decimal_to_binary(255))
print("Decimal: 0 -> Binary:", decimal_to_binary(0))
print("Decimal: 7 -> Binary:", decimal_to_binary(7))
```

Output:



```
PS C:\Users\Apple\OneDrive\Desktop\Ai Assisting> & C:/Users/Apple/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/Apple/OneDrive/Desktop/Ai Assisting/leap year.py"
Decimal: 10 -> Binary: 1010
Decimal: 255 -> Binary: 11111111
Decimal: 0 -> Binary: 0
Decimal: 7 -> Binary: 111
PS C:\Users\Apple\OneDrive\Desktop\Ai Assisting>
```

Code Explanation:

The function decimal_to_binary converts a given decimal integer into its binary representation. It first checks whether the input number is non-negative and raises an error if a negative value is provided. The built-in bin() function is then used to convert the decimal number into binary format. The prefix 0b added by bin() is removed to get a clean binary string. Finally, the function returns the binary value and prints the result for different example inputs.

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

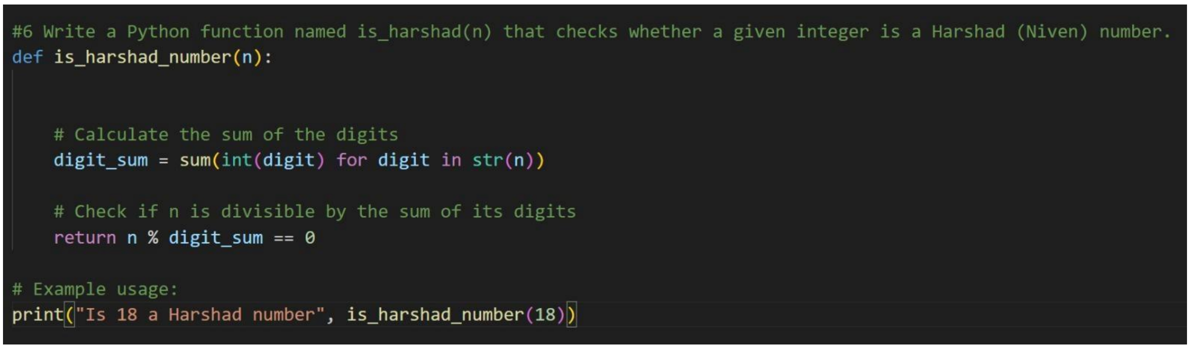
Task:

- Test boundary conditions.
- Evaluate robustness

Prompt:

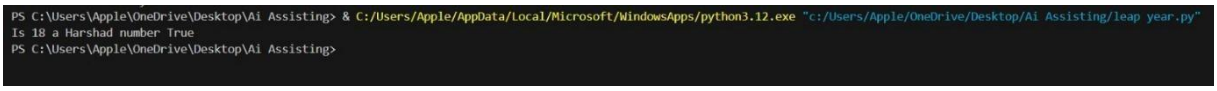
Write a Python function named is_harshad(n) that checks whether a given integer is a Harshad (Niven) number.

Code:



```python
#6 Write a Python function named is_harshad(n) that checks whether a given integer is a Harshad (Niven) number.
def is_harshad_number(n):


    # Calculate the sum of the digits
    digit_sum = sum(int(digit) for digit in str(n))

    # Check if n is divisible by the sum of its digits
    return n % digit_sum == 0

# Example usage:
print("Is 18 a Harshad number", is_harshad_number(18))
```

Output:



```
PS C:\Users\Apple\OneDrive\Desktop\Ai Assisting> & C:/Users/Apple/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/Apple/OneDrive/Desktop/Ai Assisting/leap year.py"
Is 18 a Harshad number True
PS C:\Users\Apple\OneDrive\Desktop\Ai Assisting>
```

Code Explanation:

The function is_harshad_number checks whether a given number is a Harshad (Niven) number. It first converts the number into a string and calculates the sum of its digits using a generator expression. Then, it checks whether the original number is divisible by this digit sum. If the number is divisible without any remainder, the function returns True; otherwise, it returns False. In the example provided, the number 18 is correctly identified as a Harshad number because it is divisible by the sum of its digits (1 + 8 = 9).