

# AI ASSISTED CODING ASSIGNMENT – 3.5

N.Hasini

Roll No : 2303A510G3

BATCH-03

## 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.

```
lab1.py AIAC3_2.py AIAC3_5.py X AIAC2_5.py
AIAC3_5.py > ...
1 #Zero-Shot Prompting (Leap Year Check) Write a zero-shot prompt to generate a Python function that checks whether a
2 def is_leap_year(year):
3     """
4     Check if a given year is a leap year.
5
6     A year is a leap year if it is divisible by 4,
7     except for end-of-century years, which must be divisible by 400.
8     """
9
10    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
11        return True
12    else:
13        return False
14
15    # Example usage:
16    year = 1900
17    if is_leap_year(year):
18        print(f"{year} is a leap year.")
19    else:
20        print(f"{year} is not a leap year.")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS Python Debug Console

```
PS C:\Users\91938\OneDrive\Documents\Desktop\AI> & 'c:\Users\91938\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\91938\
vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundle\libs\debugpy\launcher' '54253' '--' 'c:\Users\91938\OneDrive\Documents\De
sktop\AI\AIAC3_5.py'
1900 is not a leap year.
PS C:\Users\91938\OneDrive\Documents\Desktop\AI>
```

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

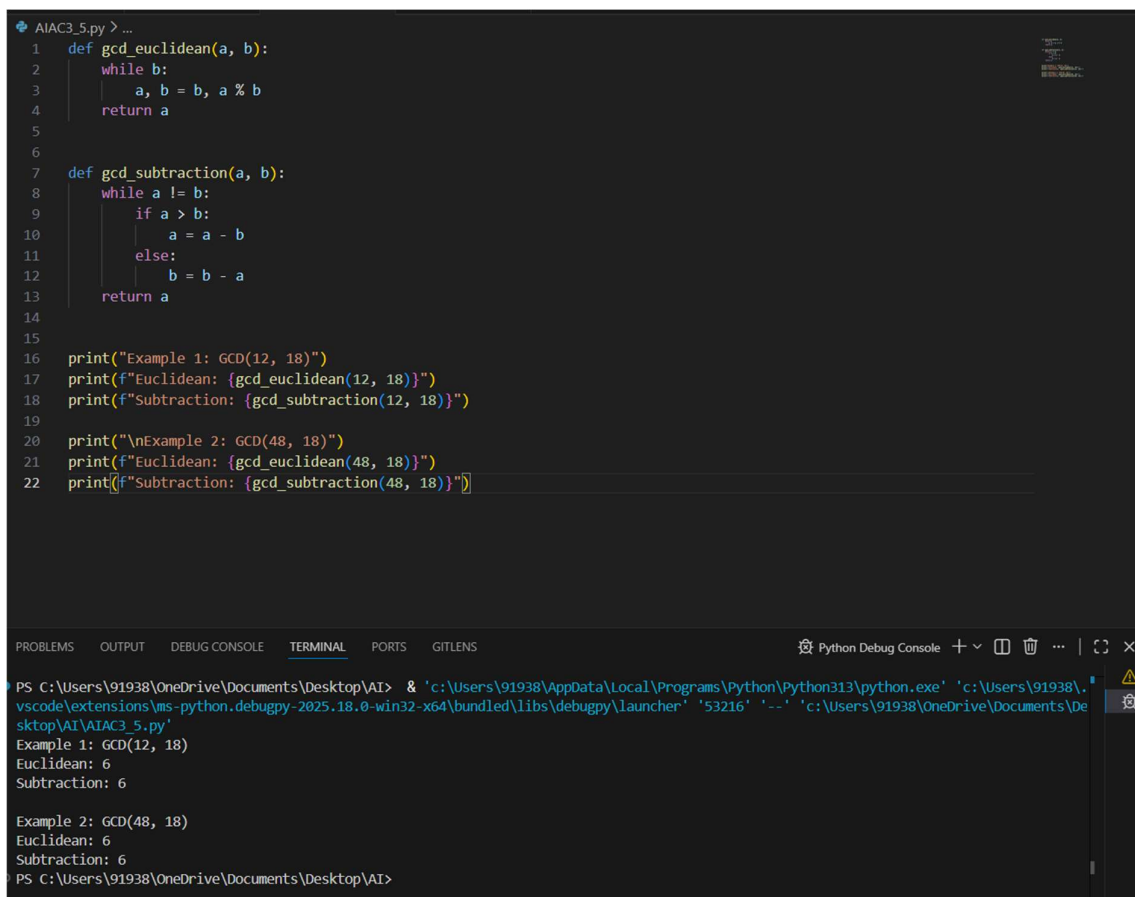
Write a one-shot prompt with one example to generate a Python function that finds the Greatest Common Divisor (GCD) of two numbers.

Example:

Input: 12, 18 → Output: 6

Task:

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



```
AIAC3_5.py > ...
1 def gcd_euclidean(a, b):
2     while b:
3         a, b = b, a % b
4     return a
5
6
7 def gcd_subtraction(a, b):
8     while a != b:
9         if a > b:
10            a = a - b
11        else:
12            b = b - a
13    return a
14
15
16 print("Example 1: GCD(12, 18)")
17 print(f"Euclidean: {gcd_euclidean(12, 18)}")
18 print(f"Subtraction: {gcd_subtraction(12, 18)}")
19
20 print("\nExample 2: GCD(48, 18)")
21 print(f"Euclidean: {gcd_euclidean(48, 18)}")
22 print(f"Subtraction: {gcd_subtraction(48, 18)}")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS Python Debug Console

```
PS C:\Users\91938\OneDrive\Documents\Desktop\AI> & 'c:\Users\91938\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\91938\OneDrive\Documents\Desktop\AI\AIAC3_5.py'
Example 1: GCD(12, 18)
Euclidean: 6
Subtraction: 6

Example 2: GCD(48, 18)
Euclidean: 6
Subtraction: 6
PS C:\Users\91938\OneDrive\Documents\Desktop\AI>
```

### Question 3: Few-Shot Prompting (LCM Calculation)

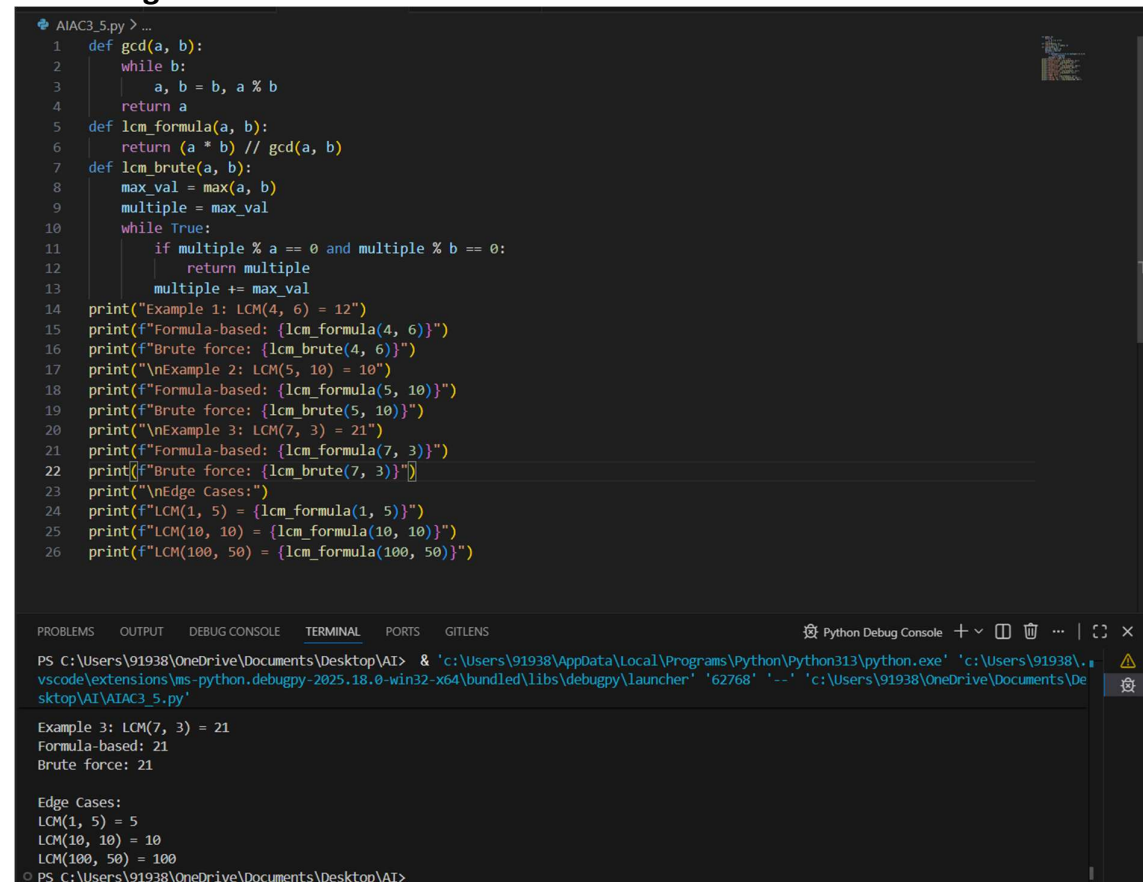
Write a few-shot prompt with multiple examples to generate a Python function that computes the Least Common Multiple (LCM).

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.



```
AIAC3_5.py > ...
1 def gcd(a, b):
2     while b:
3         a, b = b, a % b
4     return a
5 def lcm_formula(a, b):
6     return (a * b) // gcd(a, b)
7 def lcm_brute(a, b):
8     max_val = max(a, b)
9     multiple = max_val
10    while True:
11        if multiple % a == 0 and multiple % b == 0:
12            return multiple
13        multiple += max_val
14 print("Example 1: LCM(4, 6) = 12")
15 print(f"Formula-based: {lcm_formula(4, 6)}")
16 print(f"Brute force: {lcm_brute(4, 6)}")
17 print("\nExample 2: LCM(5, 10) = 10")
18 print(f"Formula-based: {lcm_formula(5, 10)}")
19 print(f"Brute force: {lcm_brute(5, 10)}")
20 print("\nExample 3: LCM(7, 3) = 21")
21 print(f"Formula-based: {lcm_formula(7, 3)}")
22 print(f"Brute force: {lcm_brute(7, 3)}")
23 print("\nEdge Cases:")
24 print(f"LCM(1, 5) = {lcm_formula(1, 5)}")
25 print(f"LCM(10, 10) = {lcm_formula(10, 10)}")
26 print(f"LCM(100, 50) = {lcm_formula(100, 50)}")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS Python Debug Console

```
PS C:\Users\91938\OneDrive\Documents\Desktop\AI> & 'c:\Users\91938\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\91938\OneDrive\Documents\Desktop\AI\AIAC3_5.py'
Example 3: LCM(7, 3) = 21
Formula-based: 21
Brute force: 21

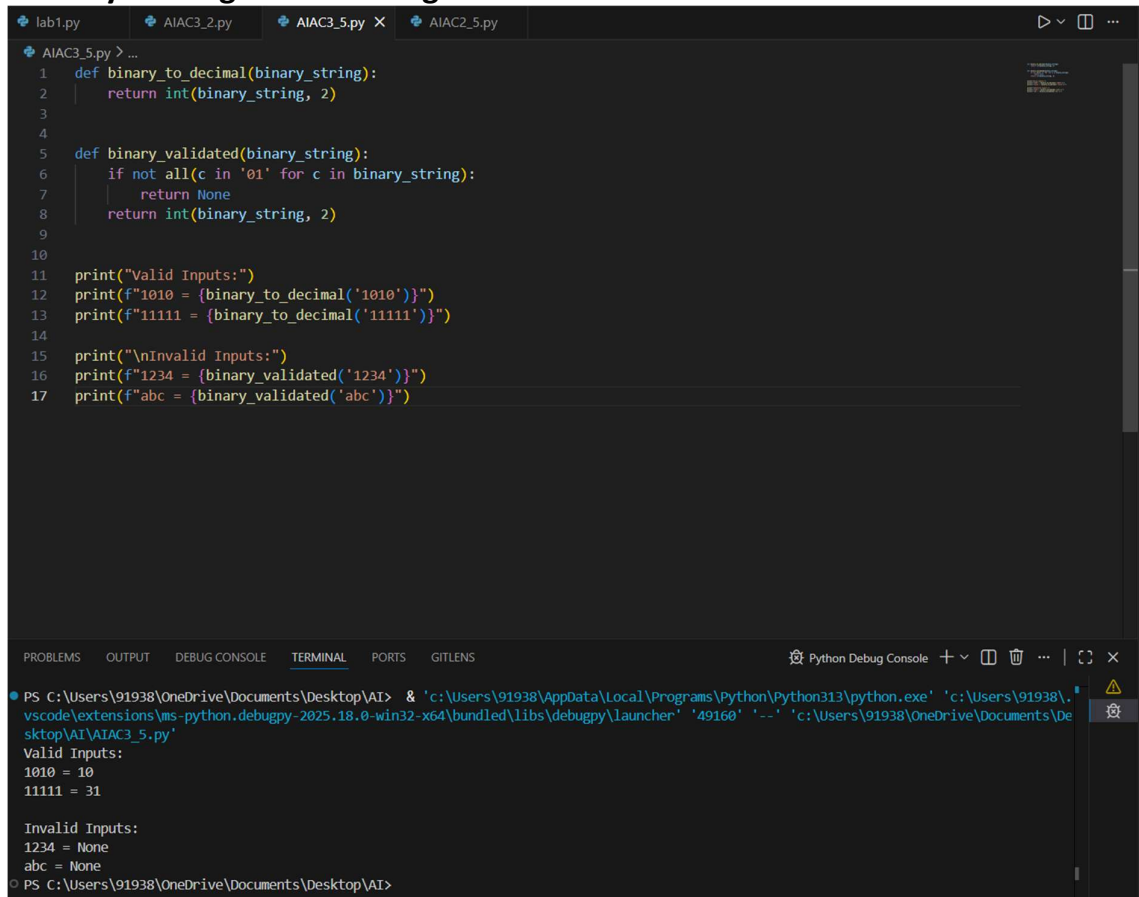
Edge Cases:
LCM(1, 5) = 5
LCM(10, 10) = 10
LCM(100, 50) = 100
PS C:\Users\91938\OneDrive\Documents\Desktop\AI>
```

#### 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.



The screenshot shows a VS Code editor with a Python file named `AIAC3_5.py`. The code defines two functions: `binary_to_decimal` and `binary_validated`. The `binary_to_decimal` function uses `int(binary_string, 2)` for conversion. The `binary_validated` function checks if the input string contains only '0' and '1' characters; if not, it returns `None`, otherwise it calls `binary_to_decimal`. The script includes test cases for both valid and invalid inputs.

```
1 def binary_to_decimal(binary_string):
2     return int(binary_string, 2)
3
4
5 def binary_validated(binary_string):
6     if not all(c in '01' for c in binary_string):
7         return None
8     return int(binary_string, 2)
9
10
11 print("Valid Inputs:")
12 print(f"1010 = {binary_to_decimal('1010')}")
13 print(f"11111 = {binary_to_decimal('11111')}")
14
15 print("\nInvalid Inputs:")
16 print(f"1234 = {binary_validated('1234')}")
17 print(f"abc = {binary_validated('abc')}")
```

The terminal output shows the execution results:

```
Valid Inputs:
1010 = 10
11111 = 31

Invalid Inputs:
1234 = None
abc = None
```

#### 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.

```
lab1.py AIAC3_2.py AIAC3_5.py X AIAC2_5.py
AIAC3_5.py > ...
1 #Write a one-shot prompt with an example to generate a Python function that converts a decimal number to binary.
2 def decimal_to_binary(n):
3     """
4     Convert a decimal number to its binary representation.
5
6     Args:
7     n (int): A decimal number.
8
9     Returns:
10    str: The binary representation of the decimal number.
11    """
12    if n < 0:
13        raise ValueError("Input must be a non-negative integer.")
14
15    binary_representation = bin(n).replace("0b", "")
16    return binary_representation
17
18 # Example usage:
19 print("Decimal: 10 -> Binary:", decimal_to_binary(10))
20 print("Decimal: 255 -> Binary:", decimal_to_binary(255))
21 print("Decimal: 0 -> Binary:", decimal_to_binary(0))
22 print("Decimal: 7 -> Binary:", decimal_to_binary(7))

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS Python Debug Console
PS C:\Users\91938\OneDrive\Documents\Desktop\AI> & 'c:\Users\91938\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\91938\
vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '62824' '--' 'c:\Users\91938\OneDrive\Documents\De
sktop\AI\AIAC3_5.py'
PS C:\Users\91938\OneDrive\Documents\Desktop\AI> c;; cd 'c:\Users\91938\OneDrive\Documents\Desktop\AI'; & 'c:\Users\91938\AppData\Local\
Programs\Python\Python313\python.exe' 'c:\Users\91938\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launc
her' '62845' '--' 'c:\Users\91938\OneDrive\Documents\Desktop\AI\AIAC3_5.py'
Decimal: 10 -> Binary: 1010
Decimal: 255 -> Binary: 11111111
Decimal: 0 -> Binary: 0
Decimal: 7 -> Binary: 111
PS C:\Users\91938\OneDrive\Documents\Desktop\AI>
```

## Question 6: Few-Shot Prompting (Harshad Number Check)

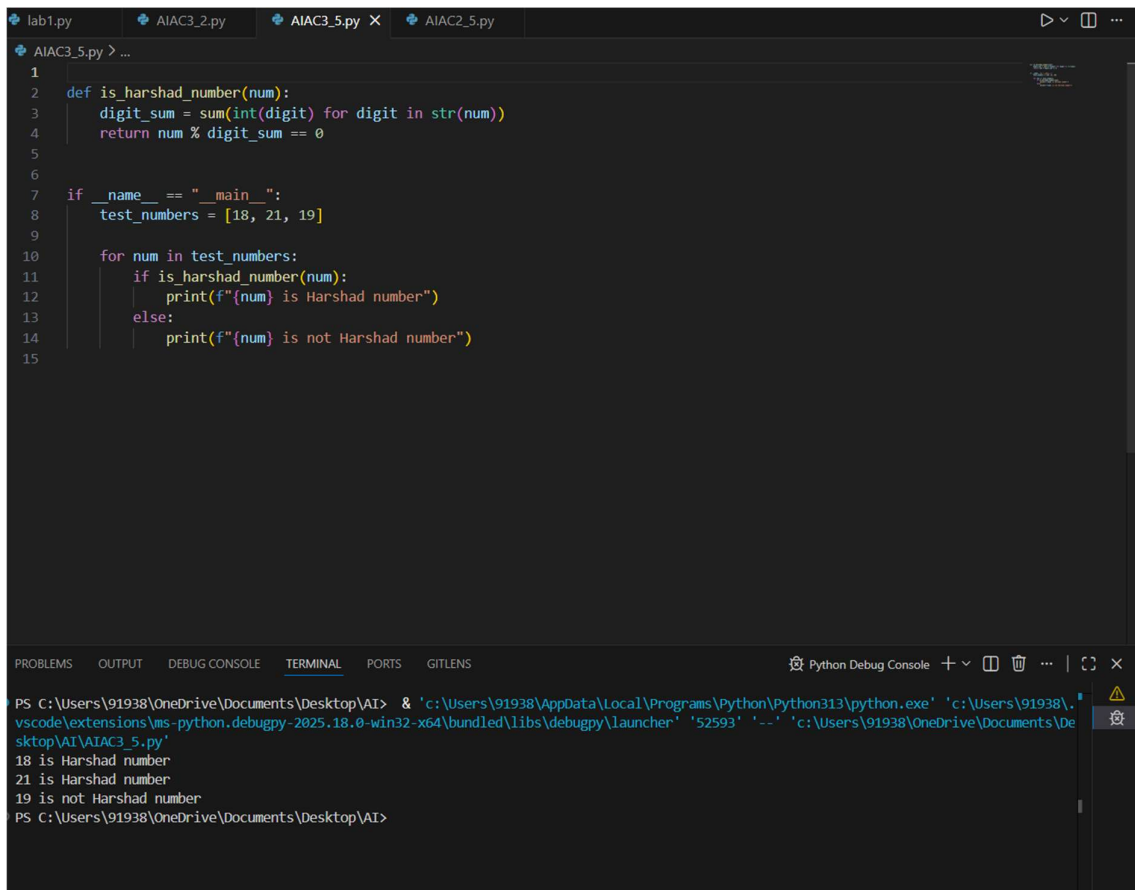
Write a few-shot prompt to generate a Python function that checks whether a number is a Harshad (Niven) number.

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 shows a VS Code editor with a Python file named `AIAC3_5.py`. The code defines a function `is_harshad_number` and tests it with the numbers 18, 21, and 19. The terminal output shows the results of the execution.

```
1
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
7 if __name__ == "__main__":
8     test_numbers = [18, 21, 19]
9
10    for num in test_numbers:
11        if is_harshad_number(num):
12            print(f"{num} is Harshad number")
13        else:
14            print(f"{num} is not Harshad number")
15
```

Terminal Output:

```
PS C:\Users\91938\OneDrive\Documents\Desktop\AI> & 'c:\Users\91938\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\91938\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundle\libs\debugpy\launcher' '52593' '--' 'c:\Users\91938\OneDrive\Documents\Desktop\AI\AIAC3_5.py'
18 is Harshad number
21 is Harshad number
19 is not Harshad number
PS C:\Users\91938\OneDrive\Documents\Desktop\AI>
```

