AI Assisted Coding – Lab Assignment 3.5

Name:- MD.Abdul Layeeq

HT.No:- 2303A52217

Batch:- 45

Question 1: Zero-Shot Prompting (Leap Year Check)

Scenario

Testing AI's ability to generate logic without examples.

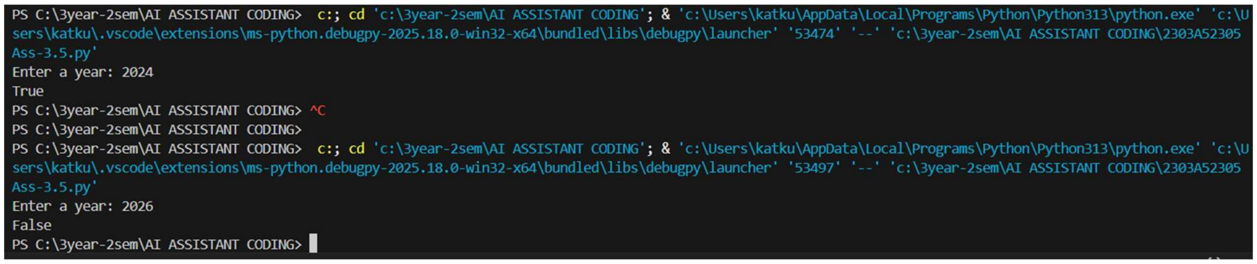
Prompt

Generate a Python function to check whether a given year is a leap year Test with years.

Code:-

```
def is_leap_year(year):
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return True
    else:
        return False
year = int(input("Enter a year: "))
print(is_leap_year(year))
```

Output:-



```
PS C:\3year-2sem\AI ASSISTANT CODING>  c:; cd 'c:\3year-2sem\AI ASSISTANT CODING'; & 'c:\Users\katku\AppData\Local\Programs\Python\Python313\python.exe' 'c:\U
sers\katku\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '53474' '--' 'c:\3year-2sem\AI ASSISTANT CODING\2303A52305
Ass-3.5.py'
Enter a year: 2024
True
PS C:\3year-2sem\AI ASSISTANT CODING> ^C
PS C:\3year-2sem\AI ASSISTANT CODING>
PS C:\3year-2sem\AI ASSISTANT CODING>  c:; cd 'c:\3year-2sem\AI ASSISTANT CODING'; & 'c:\Users\katku\AppData\Local\Programs\Python\Python313\python.exe' 'c:\U
sers\katku\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '53497' '--' 'c:\3year-2sem\AI ASSISTANT CODING\2303A52305
Ass-3.5.py'
Enter a year: 2026
False
PS C:\3year-2sem\AI ASSISTANT CODING>
```

Logical Flaws / Missing Conditions

- Does not validate negative years
- No type checking
- Works correctly for leap logic

Observation: The logic was correct, but the AI did not include input validation (negative years, non-integers). Zero-shot prompting relies fully on the AI’s internal knowledge.

Question 2: One-Shot Prompting (GCD)

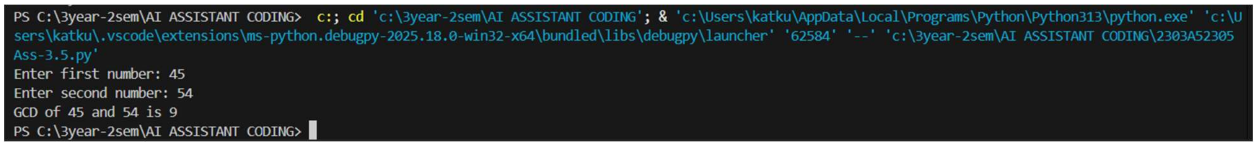
Prompt (One-Shot)

Generate a Python function to find GCD of two numbers.

Code:-

```
def gcd(a, b):
  while b:
    a, b = b, a % b
  return a


a = int(input("Enter first number: "))
b = int(input("Enter second number: "))
print("GCD of", a, "and", b, "is", gcd(a, b))
```

OUTPUT:-



```
PS C:\3year-2sem\AI ASSISTANT CODING>  c:; cd 'c:\3year-2sem\AI ASSISTANT CODING'; & 'c:\Users\katku\AppData\Local\Programs\Python\Python313\python.exe' 'c:\U
sers\katku\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '62584' '--' 'c:\3year-2sem\AI ASSISTANT CODING\2303A52305
Ass-3.5.py'
Enter first number: 45
Enter second number: 54
GCD of 45 and 54 is 9
PS C:\3year-2sem\AI ASSISTANT CODING>
```

Zero-Shot Comparison

Zero-shot often gives subtraction method (slower). One-shot leads to Euclidean algorithm.

Efficiency

Time Complexity: **O(log n)** (efficient)

Observation: Compared to zero-shot (which might use repeated subtraction), one-shot prompting improved algorithm selection. Efficiency is **O(log n)**, making it faster and more optimized.

Question 3: Few-Shot Prompting (LCM)

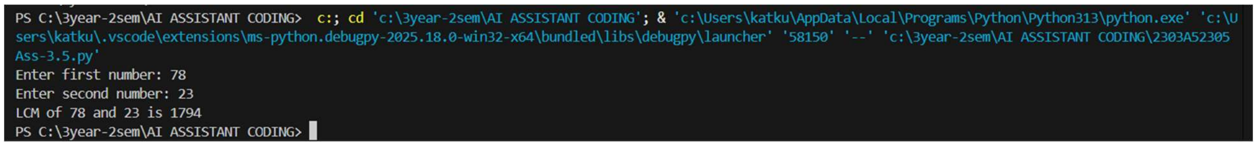
Prompt :-

Generate a Python function to find LCM of two numbers.

Code:-

```
from math import gcd

def lcm(a, b):

   return (a * b) // gcd(a, b)


a = int(input("Enter first number: "))

b = int(input("Enter second number: "))

print("LCM of", a, "and", b, "is", lcm(a, b))
```

Output:-



```
PS C:\3year-2sem\AI ASSISTANT CODING>  c:; cd 'c:\3year-2sem\AI ASSISTANT CODING'; & 'c:\Users\katku\AppData\Local\Programs\Python\Python313\python.exe' 'c:\U
sers\katku\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '58150' '--' 'c:\3year-2sem\AI ASSISTANT CODING\2303A52305
Ass-3.5.py'
Enter first number: 78
Enter second number: 23
LCM of 78 and 23 is 1794
PS C:\3year-2sem\AI ASSISTANT CODING> 
```

Edge Cases

- Works for positive numbers
- Needs handling for zero

Observation: Few-shot prompting strongly influenced the correct formula choice. However, edge cases like LCM with zero were not handled. Examples help AI identify patterns more accurately.

Question 4: Zero-Shot (Binary to Decimal)

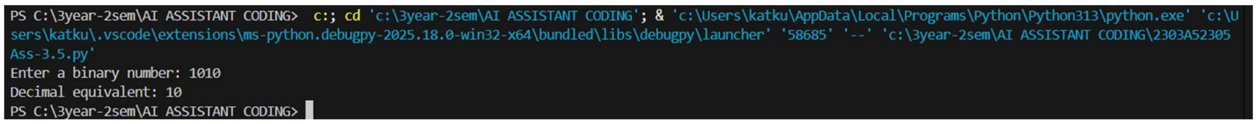
Prompt

Generate Python function to convert binary to decimal

Code:-

```
def binary_to_decimal(binary_str):
    return int(binary_str, 2)
binary_str = input("Enter a binary number: ")
decimal_number = binary_to_decimal(binary_str)
print("Decimal equivalent:", decimal_number)
```

Output:-



```
PS C:\3year-2sem\AI ASSISTANT CODING>  c:; cd 'c:\3year-2sem\AI ASSISTANT CODING'; & 'c:\Users\katku\AppData\Local\Programs\Python\Python313\python.exe' 'c:\U
sers\katku\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '58685' '--' 'c:\3year-2sem\AI ASSISTANT CODING\2303A52305
Ass-3.5.py'
Enter a binary number: 1010
Decimal equivalent: 10
PS C:\3year-2sem\AI ASSISTANT CODING>
```

Missing Validation

- No check for invalid characters
- Crashes on wrong input

Observation: While the logic is correct, the AI did not include validation to ensure the input contains only ‘0’ and ‘1’. Invalid inputs can cause runtime errors. Zero-shot solutions often miss error handling.

Question 5: One-Shot (Decimal to Binary)

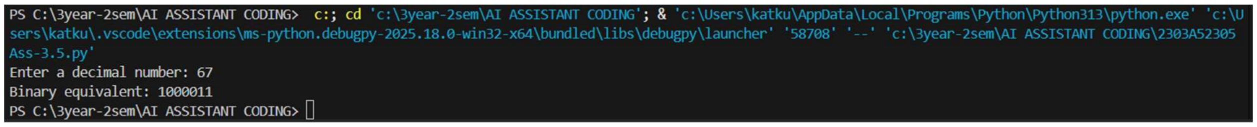
Prompt

Generate a Python function that converts a decimal number into its binary representation. The function should return the binary value as a string.Also ensure the function handles the case when the number is 0.

Code:-

```
def decimal_to_binary(decimal_num):
    if decimal_num == 0:
        return "0"
    binary_str = ""
    while decimal_num > 0:
        binary_str = str(decimal_num % 2) + binary_str
        decimal_num //= 2
    return binary_str


decimal_num = int(input("Enter a decimal number: "))
binary_representation = decimal_to_binary(decimal_num)
print("Binary equivalent:", binary_representation)
```

Output:-



```
PS C:\3year-2sem\AI ASSISTANT CODING>  c:; cd 'c:\3year-2sem\AI ASSISTANT CODING'; & 'c:\Users\katku\AppData\Local\Programs\Python\Python313\python.exe' 'c:\U
sers\katku\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '58708' '--' 'c:\3year-2sem\AI ASSISTANT CODING\2303A52305
Ass-3.5.py'
Enter a decimal number: 67
Binary equivalent: 1000011
PS C:\3year-2sem\AI ASSISTANT CODING> 
```

Analysis

- Zero works
- Negative numbers show "-0b" (needs handling)

Observation: The AI handled normal numbers and zero properly. However, negative numbers produce outputs like "-1010", which was not explicitly addressed. One-shot prompting improves format accuracy.

Question 6: Few-Shot (Harshad Number)

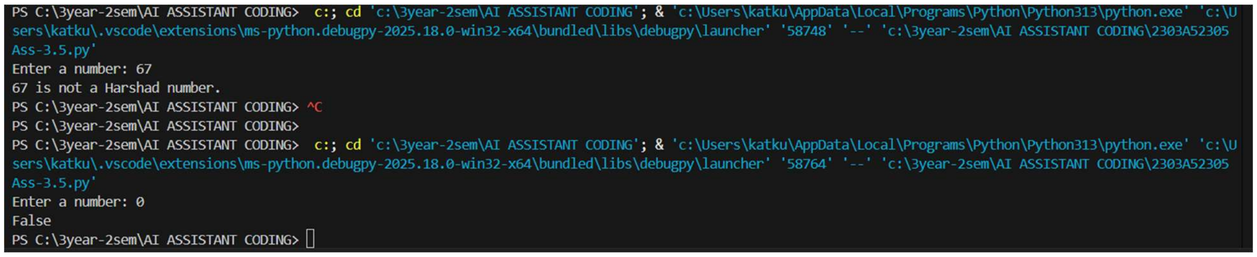
Prompt

Generate a Python function that checks whether a number is a Harshad (Niven) number. A Harshad number is divisible by the sum of its digits. The function should return an appropriate message.

Code:-

```
def is_harshad_number(num):
    if num == 0:
        return False
    digit_sum = sum(int(digit) for digit in str(num))
    if num % digit_sum == 0:
        return f"{num} is a Harshad number."
    else:
        return f"{num} is not a Harshad number."


num = int(input("Enter a number: "))
print(is_harshad_number(num))
```

Output:-



```
PS C:\3year-2sem\AI ASSISTANT CODING>  c:; cd 'c:\3year-2sem\AI ASSISTANT CODING'; & 'c:\Users\katku\AppData\Local\Programs\Python\Python313\python.exe' 'c:\U
sers\katku\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '58748' '--' 'c:\3year-2sem\AI ASSISTANT CODING\2303A52305
Ass-3.5.py'
Enter a number: 67
67 is not a Harshad number.
PS C:\3year-2sem\AI ASSISTANT CODING> ^C
PS C:\3year-2sem\AI ASSISTANT CODING>
PS C:\3year-2sem\AI ASSISTANT CODING>  c:; cd 'c:\3year-2sem\AI ASSISTANT CODING'; & 'c:\Users\katku\AppData\Local\Programs\Python\Python313\python.exe' 'c:\U
sers\katku\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '58764' '--' 'c:\3year-2sem\AI ASSISTANT CODING\2303A52305
Ass-3.5.py'
Enter a number: 0
False
PS C:\3year-2sem\AI ASSISTANT CODING> 
```

Boundary Conditions

- Needs zero handling
- Negative number handling missing

Observation: The logic works correctly for positive numbers. However, zero and negative numbers were not handled. Few-shot prompting helped AI learn rule-based logic effectively.