AI ASSISTED CODING

ASSIGNMENT-3.5

NAME: Hari Priya

H.T.No:2303A51104

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.

C: > Users > HARI PRIYA > Desktop > python > Untitled-1.py > ...

```python
#Generate a python function that reads year and checks if it is a leap year or not.
def is_leap_year(year):
    if year % 4 == 0:
        if year % 100 == 0:
            if year % 400 == 0:
                return True
            else:
                return False
        else:
            return True
    else:
        return False
try:
    year = int(input("Enter a year: "))
    if year < 0:
        print("Invalid year.")
    else:
        if is_leap_year(year):
            print(f"{year} is a leap year.")
        else:
            print(f"{year} is not a leap year.")
except ValueError:
    print("Invalid input")
```

```
PS C:\Users\HARI PRIYA> & "C:/Users/HARI PRIYA/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/HARI PRIYA/Desktop/python/Untitled-1.py"
python3.11.exe" "c:/Users/HARI PRIYA/Desktop/python/Untitled-1.py"
Enter a year: 2000
2000 is a leap year.
PS C:\Users\HARI PRIYA> & "C:/Users/HARI PRIYA/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/HARI Ppython3.11.exe" "c:/User
python3.11.exe" "c:/Users/HARI PRIYA/Desktop/python/Untitled-1.py"
Enter a year: 2024
2024 is a leap year.
```

1.The leap year logic is correct but unnecessarily complex due to deep nesting.

2.Checking for negative years is not meaningful for the standard calendar.

3.Year 0 is not handled even though it is not a valid year.

4.A simpler condition can replace multiple if–else statements.

5.Mixing input handling with logic reduces reusability of the function.

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.

One-shot

C: > Users > HARI PRIYA > Desktop > python > def armstrong_numbers_in_range(start, en.py > ...

```
'''
num=12,18
display 6
'''
def gcd(a,b):
    while b:
        a, b = b, a % b
    return a
try:
    num1 = int(input("Enter first number: "))
    num2 = int(input("Enter second number: "))
    if num1 <= 0 or num2 <= 0:
        print("Please enter positive integers only.")
    else:
        result = gcd(num1, num2)
        print(f"The GCD of {num1} and {num2} is: {result}")
except ValueError:
    print("Invalid input. Please enter valid integers.")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\HARI PRIYA> & "C:/Users/HARI PRIYA/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/HARI PRIYA/Desktop/python/def armstrong_numbers_in_range(start, en.py"
Enter first number: 12
Enter second number: 18
The GCD of 12 and 18 is: 6
PS C:\Users\HARI PRIYA> & "C:/Users/HARI PRIYA/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/HARI PRIYA/Desktop/python/def armstrong_numbers_in_range(start, en.py"
Enter first number: 10
Enter second number: 15
The GCD of 10 and 15 is: 5
```

Zero shot

C: > Users > HARI PRIYA > Desktop > python > #Generate a python code that reads two n.py > ...

```python
#Generate a python code that reads two numbers and displays GcD of the numbers
num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))
def gcd(a, b):
    while b:
        a, b = b, a % b
    return a
result = gcd(num1, num2)
print(f"The GCD of {num1} and {num2} is {result}")
```

```
PS C:\Users\HARI PRIYA> & "C:/Users/HARI PRIYA/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/HARI PRIYA/Desktop/python/#Gen
erate a python code that reads two n.py"
Enter first number: 2
Enter second number: 9
The GCD of 2 and 9 is 1
```

Efficiency comparison of the two GCD codes:

- Both programs use the **Euclidean Algorithm**, so their **time complexity is the same: O(log min(a, b))**.
- The **One shot code is slightly more efficient in structure** because the GCD logic is placed inside a function, making it reusable and cleaner.
- The **Zero shot code mixes input, logic, and output**, which reduces maintainability but does not change algorithmic efficiency.
- The try–except in the Zero shot code is unnecessary because int(input()) errors occur **before** the loop runs.
- Overall, **performance is equal**, but the **One shot code is better in design and clarity**, making it preferable in practice.

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.

Untitled-1.py

```python
'''
num=4 6
display 12
num=5 10
display 10
num=7 3
display 21
'''
def lcm(a, b):
    def gcd(x, y):
        while y:
            x, y = y, x % y
        return x
    return a * b // gcd(a, b)
try:
    num1=int(input("Enter first number: "))
    num2=int(input("Enter second number: "))
    print("The LCM of", num1, "and", num2, "is", lcm(num1, num2))
except ValueError:
    print("Invalid input. Please enter integer values.")
```

```
PS C:\Users\HARI PRIYA\Desktop\python> & "C:/Users/HARI PRIYA/AppData/Local/Microsoft/WindowsApps
/python3.11.exe" "c:/Users/HARI PRIYA/Desktop/python/Untitled-1.py"
Enter first number: 4
Enter second number: 6
The LCM of 4 and 6 is 12
PS C:\Users\HARI PRIYA\Desktop\python> & "C:/Users/HARI PRIYA/AppData/Local/Microsoft/WindowsApps
/python3.11.exe" "c:/Users/HARI PRIYA/Desktop/python/Untitled-1.py"
Enter first number: 5
Enter second number: 10
The LCM of 5 and 10 is 10
PS C:\Users\HARI PRIYA\Desktop\python> & "C:/Users/HARI PRIYA/AppData/Local/Microsoft/WindowsApps
/python3.11.exe" "c:/Users/HARI PRIYA/Desktop/python/Untitled-1.py"
Enter first number: 7
Enter second number: 3
The LCM of 7 and 3 is 21
PS C:\Users\HARI PRIYA\Desktop\python>
```

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.

#Generate a python function that reads a.py > ...

```python
#Generate a python function that reads a binary number and converts it to decimal number.
def binary_to_decimal(binary_str):
    decimal = 0
    power = 0
    for digit in reversed(binary_str):
        if digit == '1':
            decimal += 2 ** power
        power += 1
    return decimal
try:
    binary_input = input("Enter a binary number: ")
    decimal_output = binary_to_decimal(binary_input)
    print(f"The decimal equivalent of binary {binary_input} is {decimal_output}.")
except ValueError:
    print("Invalid input. Please enter a valid binary number.")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python

```
.exe" "c:/Users/HARI PRIYA/Desktop/python/#Generate a python function that reads a.py"

Enter a binary number: 0101
The decimal equivalent of binary 0101 is 5.
PS C:\Users\HARI PRIYA\Desktop\python> & "C:/Users/HARI PRIYA/AppData/Local/Microsoft/WindowsApps/python3.11
.exe" "c:/Users/HARI PRIYA/Desktop/python/#Generate a python function that reads a.py"

Enter a binary number: 0001011
a python function that reads a.py"

Enter a binary number: -2
The decimal equivalent of binary -2 is 0.
```

1. It does **not check for empty input** (an empty string passes all() and is treated as valid).
2. It does **not handle leading/trailing spaces** (e.g., " 101 " should be stripped).
3. The try–except ValueError is **unnecessary**, because no ValueError can occur in this code.
4. It does not give feedback if the input is **too large**, which could cause performance issues.
5. The validation only checks characters, not **user intent or format** (like negative sign or prefixes such as 0b).

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.

#Generate a python function that reads a.py | Untitled-1.py

C: > Users > HARI PRIYA > Desktop > New folder > Untitled-1.py > ...

```
'''
n=10
display 1010
'''
def decimal_to_binary(n):
    if n == 0:
        return "0"
    binary = ""
    while n > 0:
        binary = str(n % 2) + binary
        n = n // 2
    return binary
try:
    n = int(input("Enter a decimal number: "))
    if n < 0:
        print("Invalid input")
    else:
        binary_result = decimal_to_binary(n)
        print(binary_result)
except ValueError:
    print("Invalid input")
```

```
Enter a decimal number: 8
1000
PS C:\Users\HARI PRIYA> & "C:/Users/HARI PRIYA/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/HARI PRIYA/Desktop/New
 folder/Untitled-1.py"
Enter a decimal number: 0
0
PS C:\Users\HARI PRIYA> & "C:/Users/HARI PRIYA/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/HARI PRIYA/Desktop/New
 folder/Untitled-1.py"
Enter a decimal number: -2
Invalid input
```

C: > Users > HARI PRIYA > Desktop > New folder > Untitled-1.py > ...

```python
#Generate a python funtion that reads a number and coverts it to binary number
def convert_to_binary(num):
    if num == 0:
        return "0"
    binary = ""
    while num > 0:
        binary = str(num % 2) + binary
        num = num // 2
    return binary
try:
    n = int(input("Enter a number to convert to binary: "))
    if n < 0:
        print("Invalid input")
    else:
        binary_number = convert_to_binary(n)
        print(binary_number)
except ValueError:
    print("Invalid input")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python

```
PS C:\Users\HARI PRIYA> & "C:/Users/HARI PRIYA/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/HARI PRIYA/Desktop
 folder/Untitled-1.py"
Enter a number to convert to binary: 10
1010
PS C:\Users\HARI PRIYA> & "C:/Users/HARI PRIYA/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/HARI PRIYA/Desktop
 folder/Untitled-1.py"
Enter a number to convert to binary: 7
111
```

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

#Generate a python function that reads a.py | Untitled-1.py | ass.py

C: > Users > HARI PRIYA > Desktop > New folder > ass.py > ...

```
'''
n=18
display Harshad numbers
n=19
display Not a Harshad number
n=5
display Invalid input
n=abc
display Invalid input
'''
def is_harshad_number(num):
    if num <= 0:
        return "Invalid input"
    digit_sum = sum(int(digit) for digit in str(num))
    if num % digit_sum == 0:
        return "Harshad number"
    else:
        return "Not a Harshad number"
try:
    n = int(input("Enter a positive integer: "))
    result = is_harshad_number(n)
    print(result)
except ValueError:
    print("Invalid input")
```

```
Enter a positive integer: 19
Not a Harshad number
PS C:\Users\HARI PRIYA> & "C:/Users/HARI PRIYA/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/HARI PRIYA/Desktop/New
 folder/ass.py"
Enter a positive integer: 21
Harshad number
PS C:\Users\HARI PRIYA> & "C:/Users/HARI PRIYA/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/HARI PRIYA/Desktop/New
 folder/ass.py"
Enter a positive integer: a
Invalid input
```

Robustness of the code:

- The code safely handles **non-numeric input** using try–except, preventing crashes.
- It checks for **invalid values (zero or negative numbers)** before processing.
- The digit sum is calculated reliably by converting the number to a string.
- Returning clear messages like "Harshad Number" and "Invalid Input" improves user feedback.
- Overall, the program is **robust for normal user input**, though returning booleans instead of strings would improve reusability.