AI ASSISTED CODING ASSIGNMENT – 3.5

Name :P.Pooja

Hallticket:2303A510F7

Batch:03

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

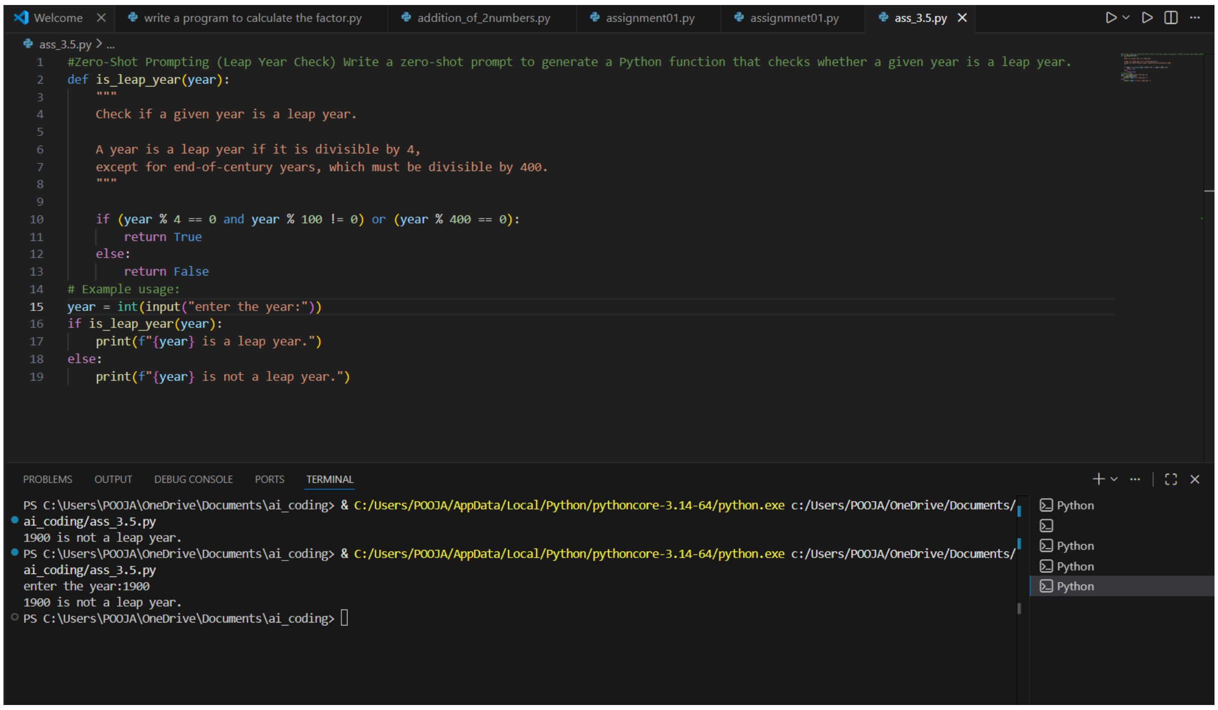
Write a zero-shot prompt to generate a Python function that checks whether a given year is a leap year.

Week2 - Task:

• Record the AI-generated code.

• Test with years like 1900, 2000, 2024.

• Identify logical flaws or missing conditions.



```python
#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.
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 = int(input("enter the year:"))
if is_leap_year(year):
    print(f"{year} is a leap year.")
else:
    print(f"{year} is not a leap year.")
```

```
PS C:\Users\POOJA\OneDrive\Documents\ai_coding> & C:/Users/POOJA/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/POOJA/OneDrive/Documents/
ai_coding/ass_3.5.py
1900 is not a leap year.
PS C:\Users\POOJA\OneDrive\Documents\ai_coding> & C:/Users/POOJA/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/POOJA/OneDrive/Documents/
ai_coding/ass_3.5.py
enter the year:1900
1900 is not a leap year.
PS C:\Users\POOJA\OneDrive\Documents\ai_coding>
```

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

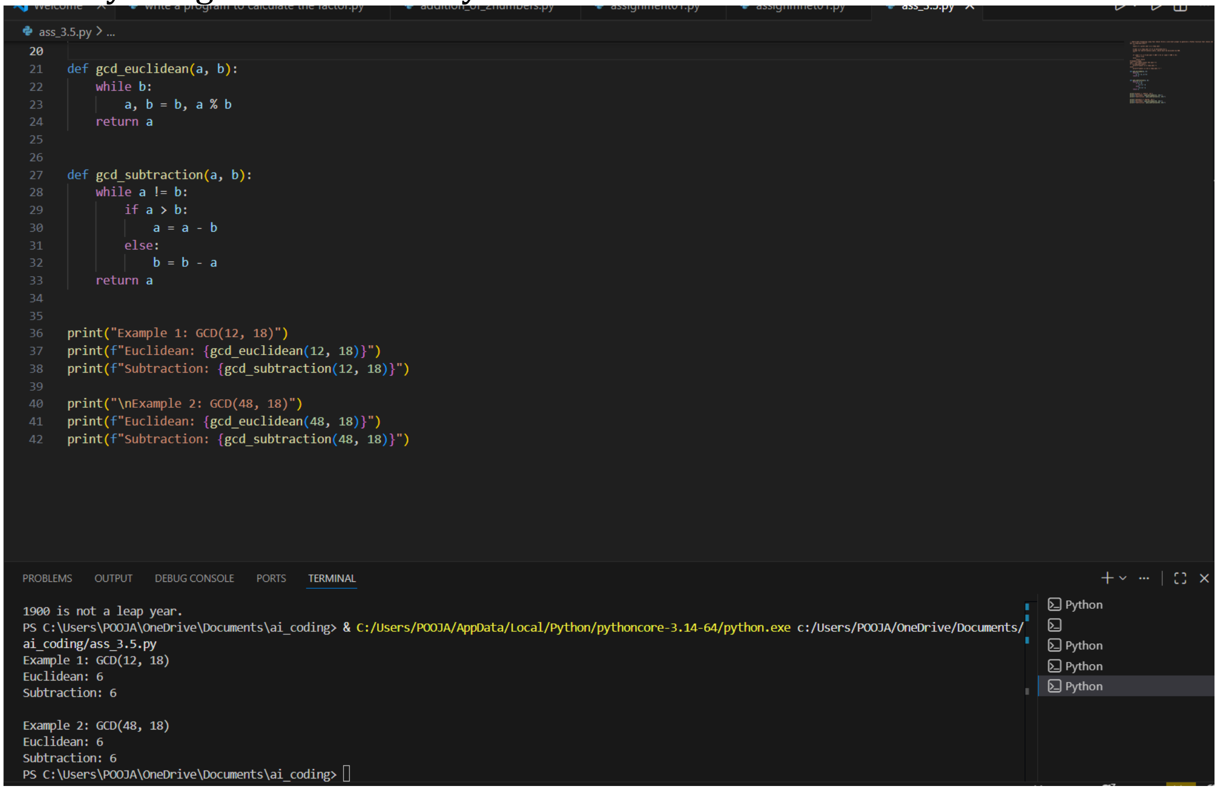
Write a one-shot prompt with one example to generate a Python function that finds the Greatest Common Divisor (GCD) of two numbers.

Example:

Input: 12, 18 → Output: 6

Task:

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



```python
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)}")
```

```
1900 is not a leap year.
PS C:\Users\POOJA\OneDrive\Documents\ai_coding> & C:/Users/POOJA/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/POOJA/OneDrive/Documents/
ai_coding/ass_3.5.py
Example 1: GCD(12, 18)
Euclidean: 6
Subtraction: 6

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

Question 3: Few-Shot Prompting (LCM Calculation)

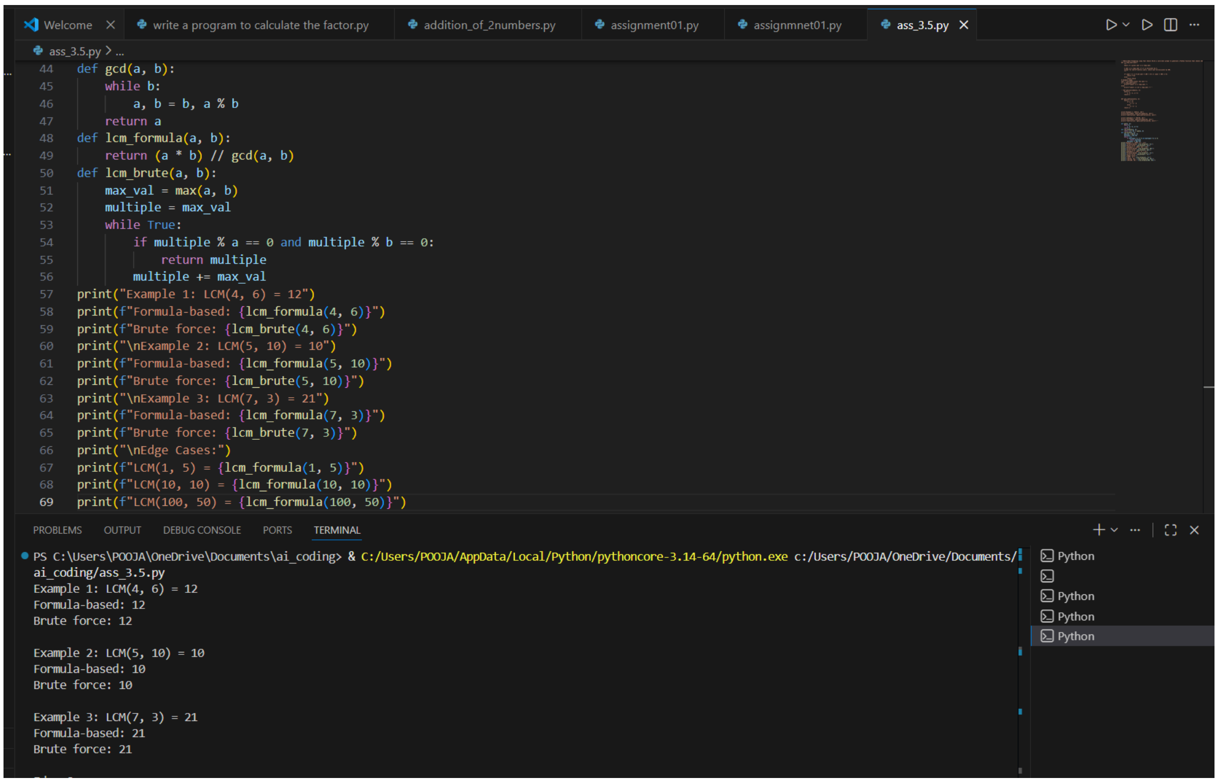
Write a few-shot prompt with multiple examples to generate a Python function that computes the Least Common Multiple (LCM).

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.

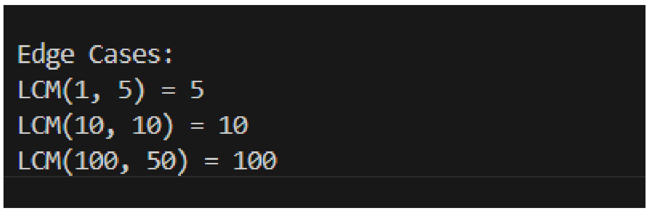


```python
def gcd(a, b):
    while b:
        a, b = b, a % b
    return a
def lcm_formula(a, b):
    return (a * b) // gcd(a, b)
def lcm_brute(a, b):
    max_val = max(a, b)
    multiple = max_val
    while True:
        if multiple % a == 0 and multiple % b == 0:
            return multiple
        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)}")
```

```
PS C:\Users\POOJA\OneDrive\Documents\ai_coding> & C:/Users/POOJA/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/POOJA/OneDrive/Documents/
ai_coding/ass_3.5.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
```



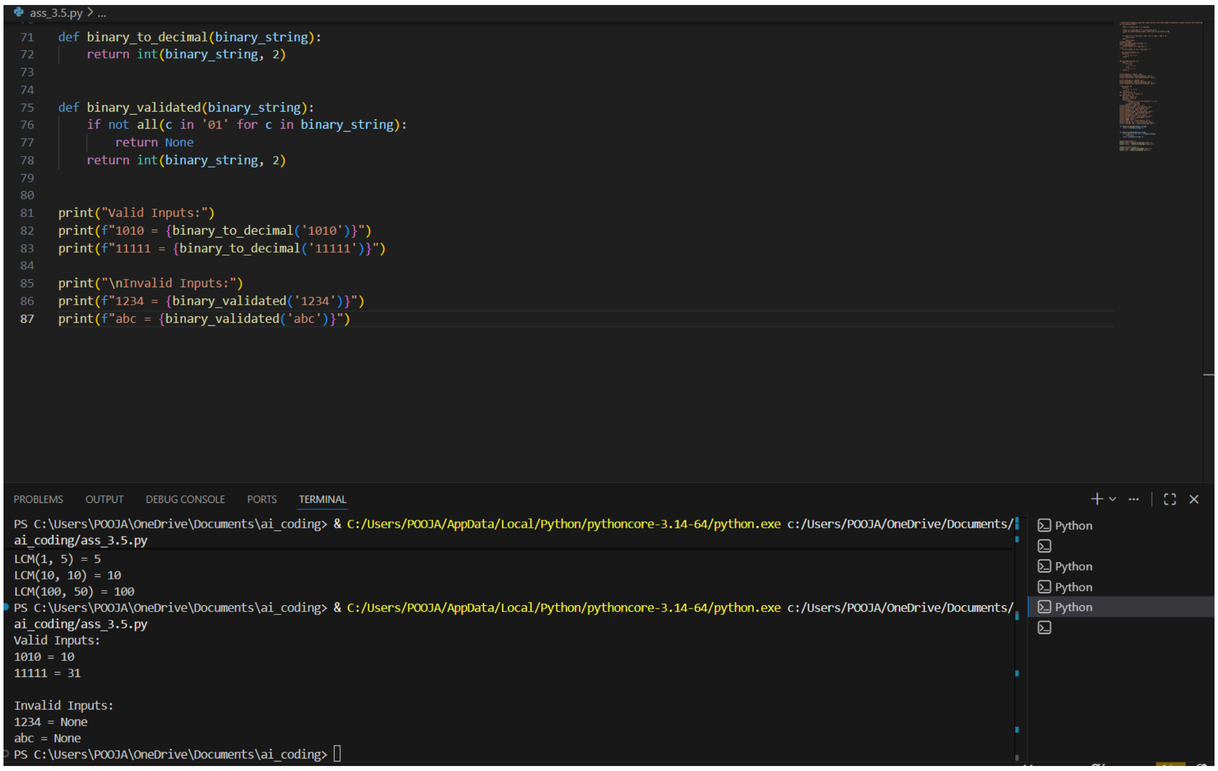
```
Edge Cases:
LCM(1, 5) = 5
LCM(10, 10) = 10
LCM(100, 50) = 100
```

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.



```python
def binary_to_decimal(binary_string):
    return int(binary_string, 2)


def binary_validated(binary_string):
    if not all(c in '01' for c in binary_string):
        return None
    return int(binary_string, 2)


print("Valid Inputs:")
print(f"1010 = {binary_to_decimal('1010')}")
print(f"11111 = {binary_to_decimal('11111')}")

print("\nInvalid Inputs:")
print(f"1234 = {binary_validated('1234')}")
print(f"abc = {binary_validated('abc')}")
```

```
PS C:\Users\POOJA\OneDrive\Documents\ai_coding> & C:/Users/POOJA/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/POOJA/OneDrive/Documents/ai_coding/ass_3.5.py
LCM(1, 5) = 5
LCM(10, 10) = 10
LCM(100, 50) = 100
PS C:\Users\POOJA\OneDrive\Documents\ai_coding> & C:/Users/POOJA/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/POOJA/OneDrive/Documents/ai_coding/ass_3.5.py
Valid Inputs:
1010 = 10
11111 = 31

Invalid Inputs:
1234 = None
abc = None
PS C:\Users\POOJA\OneDrive\Documents\ai_coding> 
```

Question 5: One-Shot Prompting (Decimal to Binary Conversion)

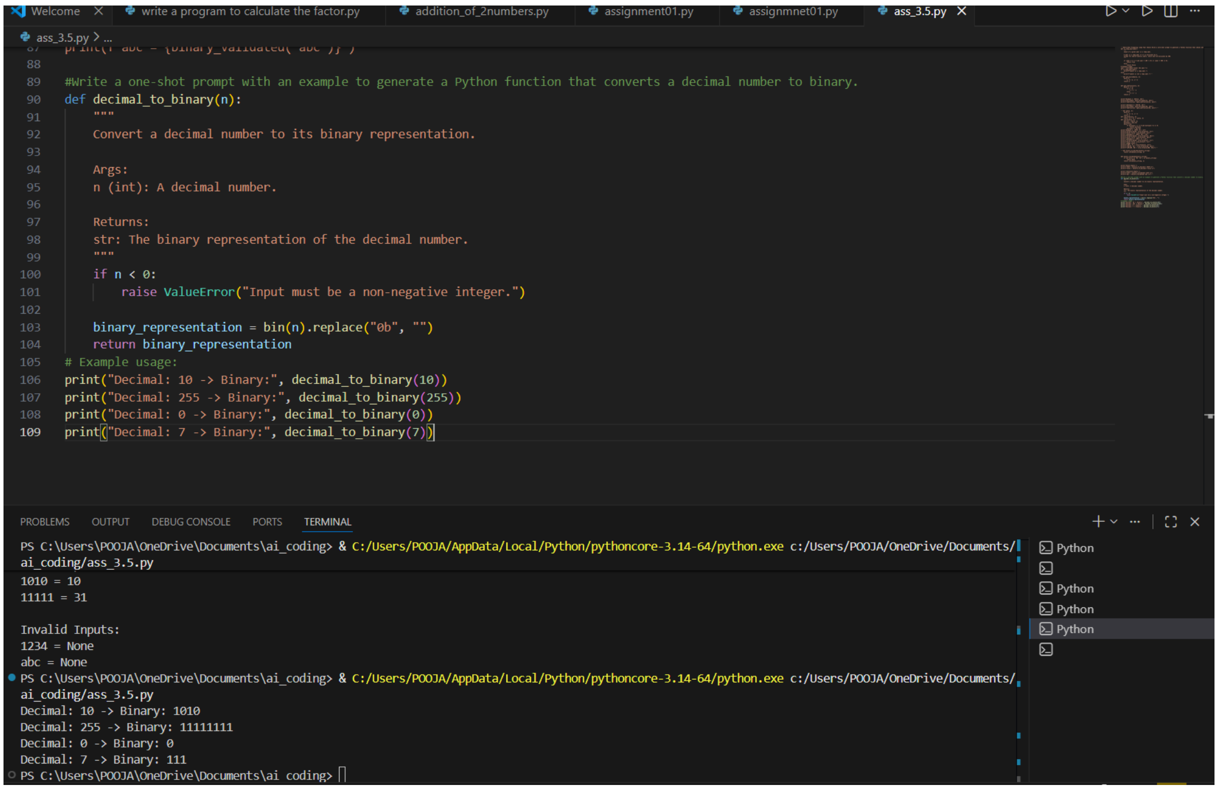
Write a one-shot prompt with an example to generate a Python function that converts a decimal number to binary.

Example:

Input: 10 → Output: 1010

Task:

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



```python
#Write a one-shot prompt with an example to generate a Python function that converts a decimal number to binary.
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))
```

```
PS C:\Users\POOJA\OneDrive\Documents\ai_coding> & C:/Users/POOJA/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/POOJA/OneDrive/Documents/
ai_coding/ass_3.5.py
1010 = 10
11111 = 31

Invalid Inputs:
1234 = None
abc = None
PS C:\Users\POOJA\OneDrive\Documents\ai_coding> & C:/Users/POOJA/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/POOJA/OneDrive/Documents/
ai_coding/ass_3.5.py
Decimal: 10 -> Binary: 1010
Decimal: 255 -> Binary: 11111111
Decimal: 0 -> Binary: 0
Decimal: 7 -> Binary: 111
PS C:\Users\POOJA\OneDrive\Documents\ai_coding>
```

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

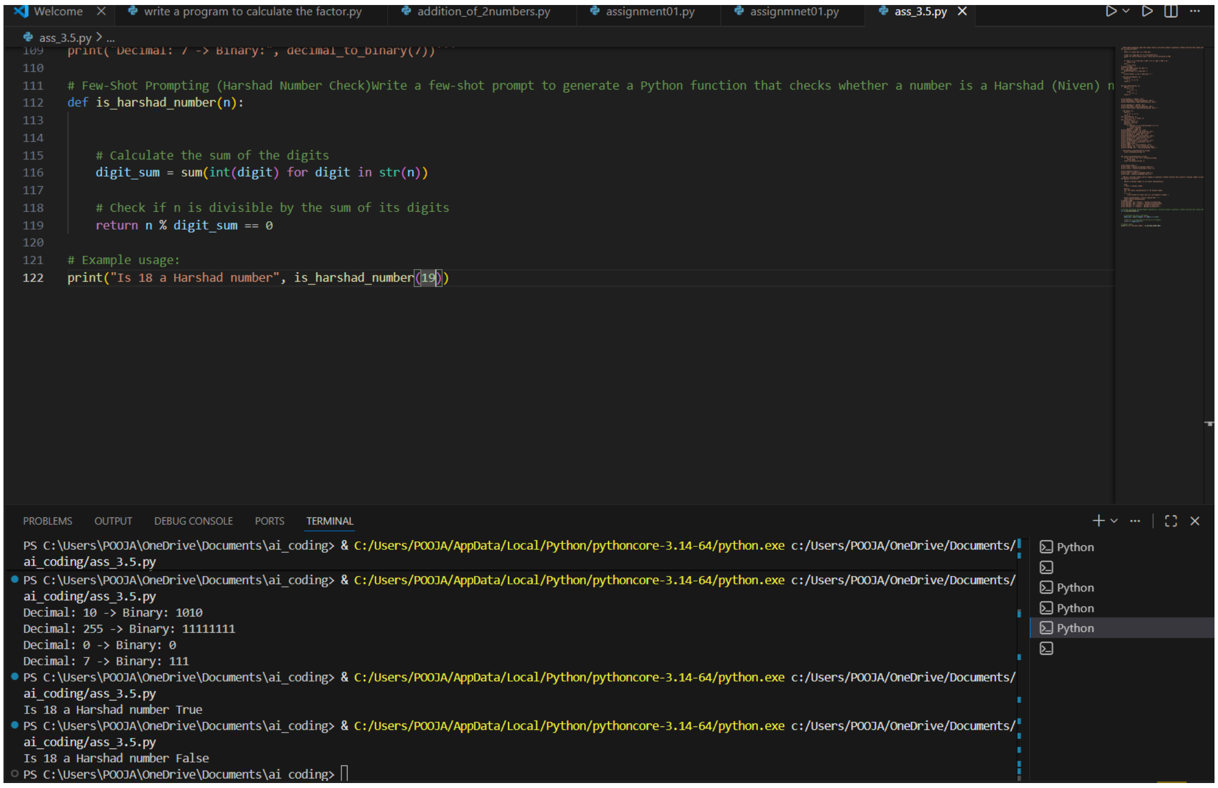
Write a few-shot prompt to generate a Python function that checks whether a number is a Harshad (Niven) number.

Examples:

- Input: 18 → Output: Harshad Number
- Input: 21 → Output: Harshad Number
- Input: 19 → Output: Not a Harshad

Number Task:

- Test boundary conditions.
- Evaluate robustness



```python
print("Decimal: 7 -> Binary:", decimal_to_binary(7))

# 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) n
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(19))
```

```
PS C:\Users\POOJA\OneDrive\Documents\ai_coding> & C:/Users/POOJA/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/POOJA/OneDrive/Documents/
ai_coding/ass_3.5.py
PS C:\Users\POOJA\OneDrive\Documents\ai_coding> & C:/Users/POOJA/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/POOJA/OneDrive/Documents/
ai_coding/ass_3.5.py
Decimal: 10 -> Binary: 1010
Decimal: 255 -> Binary: 11111111
Decimal: 0 -> Binary: 0
Decimal: 7 -> Binary: 111
PS C:\Users\POOJA\OneDrive\Documents\ai_coding> & C:/Users/POOJA/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/POOJA/OneDrive/Documents/
ai_coding/ass_3.5.py
Is 18 a Harshad number True
PS C:\Users\POOJA\OneDrive\Documents\ai_coding> & C:/Users/POOJA/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/POOJA/OneDrive/Documents/
ai_coding/ass_3.5.py
Is 18 a Harshad number False
PS C:\Users\POOJA\OneDrive\Documents\ai_coding>
```