

# AI Assisted Coding

## Assignment 3.5

Name: Dinesh

Hallticket:2303A52329

### Task-01

Prompt: py function to check whether a given year is leap year or not

```
def is_leap_year(year):
    """
    Check whether a given year is a leap year or not.

    A year is a leap year if:
    - It is divisible by 4 AND not divisible by 100, OR
    - It is divisible by 400
    """
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return True
    return False

# Example usage
if __name__ == "__main__":
    test_years = [2000, 2004, 2100, 2024, 1900, 2023]
    for year in test_years:
        print(f"{year}: {is_leap_year(year)}")
```

Output:

```
C:\AI-Assisted-Coding\Lab_3_5.py
2000: True
2024: True
1900: False
PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding>
```

## Task-02

**Prompt:** Write a Python function to find the Greatest Common Divisor (GCD) of two numbers.

Example:

Input: 12, 18

Output: 6

The function should take two integers as input and return their GCD.

```
def gcd(a, b):
    """
    Find the Greatest Common Divisor (GCD) of two numbers.

    Uses the Euclidean algorithm.
    """
    while b:
        a, b = b, a % b
    return abs(a)

# Example usage
if __name__ == "__main__":
    test_pairs = [(12, 18), (48, 18), (100, 50)]
    for num1, num2 in test_pairs:
        print(f"GCD of {num1} and {num2}: {gcd(num1, num2)}")
```

**Output:**

```
PS C:\Users\Vivek> python -u "c:\Users\Vivek\OneDrive\Desktop\Python\Task-02.py"
GCD of 12 and 18: 6
GCD of 48 and 18: 6
GCD of 100 and 50: 50
```

## Task 03

**prompt:**

**Write a Python function to compute the Least Common Multiple (LCM) of two numbers.**

**Examples:**

**Input: 4, 6 Output: 12**

**Input: 5, 10 Output: 10**

**Input: 7, 3 Output: 21**

**The function should take two integers as input and return their LCM.**

**CODE:**

```
def lcm(a, b):
    """
    Find the Least Common Multiple (LCM) of two numbers.

    The LCM is the smallest positive integer that is divisible by both numbers.
    Uses the relationship: LCM(a, b) = (a * b) / GCD(a, b)
    """

    def gcd(x, y):
        while y:
            x, y = y, x % y
        return abs(x)

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

# Example usage
if __name__ == "__main__":
    test_pairs = [(4, 6), (5, 10), (7, 3)]
    for num1, num2 in test_pairs:
        print(f"LCM of {num1} and {num2}: {lcm(num1, num2)}")
```

**Output:**

```
PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding> python
LCM of 4 and 6: 12
LCM of 5 and 10: 10
LCM of 7 and 3: 21
LCM of 10 and 6: 30
LCM of 63 and 25: 1575
PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding>
*! ⌂ ⊗ 4 ▲ 4
```

## Task 4 :

**Write a Python program to convert a binary string to its decimal value. Also give explanation whether the input is valid or invalid**  
**Test the program using multiple valid and invalid binary examples.**

```
def binary_to_decimal(binary_str):
    """
    Convert a binary number (string) to its decimal equivalent.

    Args:
        binary_str: A string representing a binary number (e.g., "1010")

    Returns:
        The decimal equivalent of the binary number
    """
    return int(binary_str, 2)

# Example usage
if __name__ == "__main__":
    test_binaries = ["1010", "11111", "10000", "1"]
    for binary in test_binaries:
        print(f"Binary {binary}: {binary_to_decimal(binary)}")

def binary_to_decimal_validated(binary_str):
    """
    Convert a binary number (string) to its decimal equivalent with validation.

    Args:
        binary_str: A string representing a binary number

    Returns:
        A tuple of (decimal_value, is_valid, explanation)
    """
    # Check if input is a string
    if not isinstance(binary_str, str):
        return None, False, "Invalid: Input must be a string"

    # Check if string is empty
    if not binary_str:
        return None, False, "Invalid: Input cannot be empty"

    # Check if all characters are 0 or 1
    if not all(c in '01' for c in binary_str):
        return None, False, f"Invalid: '{binary_str}' contains non-binary characters"

    # Convert to decimal
    decimal_value = int(binary_str, 2)
    return decimal_value, True, f"Valid: Binary '{binary_str}' = Decimal {decimal_value}"

# Test with validation
print("\nBinary to Decimal with Validation:")
test_cases = ["1010", "11111", "10000", "1", "1201", "", "abc"]
for binary in test_cases:
    decimal, is_valid, explanation = binary_to_decimal_validated(binary)
```

## Output:

```
Binary to Decimal with Validation:  
Valid: Binary '1010' = Decimal 10  
Valid: Binary '11111' = Decimal 31  
Valid: Binary '10000' = Decimal 16  
Valid: Binary '1' = Decimal 1  
Invalid: '1201' contains non-binary characters  
Invalid: Input cannot be empty  
Invalid: 'abc' contains non-binary characters  
PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding>  
n*! ⌂ ④ ⊗ 4 ▲ 4
```

## Task-5

**prompt :**

**generate a Python function that converts a decimal number to binary.**

**Example:**

**Input: 10 → Output: 1010**

**also give a one word explanation for outputs**

**Code:**

```
def decimal_to_binary(n):  
    """  
        Convert a decimal number to its binary representation.  
  
        Args:  
            n: A non-negative integer in decimal form  
  
        Returns:  
            Binary representation as a string  
    """  
    if n == 0:  
        return "0"  
  
    binary = ""  
    while n > 0:  
        binary = str(n % 2) + binary  
        n //= 2  
    return binary  
  
  
# Example usage  
if __name__ == "__main__":  
    test_numbers = [10, 5, 15, 32, 100]  
    for num in test_numbers:  
        result = decimal_to_binary(num)  
        print(f"{num} → {result} (Conversion)")
```

**Output:**

```
Binary 1: 1

Binary to Decimal with Validation:
Valid: Binary '1010' = Decimal 10
Valid: Binary '11111' = Decimal 31
Valid: Binary '10000' = Decimal 16
Valid: Binary '1' = Decimal 1
Invalid: '1201' contains non-binary characters
Binary to Decimal with Validation:
Valid: Binary '1010' = Decimal 10
Valid: Binary '11111' = Decimal 31
Valid: Binary '10000' = Decimal 16
Valid: Binary '1' = Decimal 1
Invalid: '1201' contains non-binary characters
Valid: Binary '1010' = Decimal 10
Valid: Binary '11111' = Decimal 31
Valid: Binary '10000' = Decimal 16
Valid: Binary '1' = Decimal 1
Invalid: '1201' contains non-binary characters
Valid: Binary '11111' = Decimal 31
Valid: Binary '10000' = Decimal 16
Valid: Binary '1' = Decimal 1
Invalid: '1201' contains non-binary characters
Valid: Binary '10000' = Decimal 16
Valid: Binary '1' = Decimal 1
Invalid: '1201' contains non-binary characters
Invalid: Input cannot be empty
Invalid: '1201' contains non-binary characters
Invalid: Input cannot be empty
Invalid: Input cannot be empty
Invalid: 'abc' contains non-binary characters
PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Co
```

**Task-06:****Prompt:**

Write a Python function to check whether a given number is a Harshad (Niven) number.

Examples:

Input: 18 Output: Harshad Number

Input: 21 Output: Harshad Number

Input: 19 Output: Not a Harshad Number

The function should calculate the sum of digits and check divisibility.

Also test boundary conditions such as single-digit numbers, zero, and negative values to ensure robustness.

## Code

```
def is_harshad_number(n):
    """
    Check whether a given number is a Harshad (Niven) number.

    A Harshad number is an integer that is divisible by the sum of its digits.

    Args:
        n: An integer (can be negative or zero)

    Returns:
        A string indicating whether the number is a Harshad number or not.
    """
    if n < 0:
        return "Not a Harshad Number"

    digit_sum = sum(int(digit) for digit in str(abs(n)))

    if digit_sum == 0:
        return "Not a Harshad Number"

    if n % digit_sum == 0:
        return "Harshad Number"
    else:
        return "Not a Harshad Number"

# Example usage
if __name__ == "__main__":
    test_numbers = [18, 21, 19, 0, -18, 9]
    for num in test_numbers:
        result = is_harshad_number(num)
        print(f"{num}: {result}")
```

## Output:

```
18: Harshad Number
21: Harshad Number
19: Not a Harshad Number
0: Not a Harshad Number
-18: Not a Harshad Number
9: Harshad Number
PS C:\Users\Vivek\OneDrive\Desktop\3:
```