

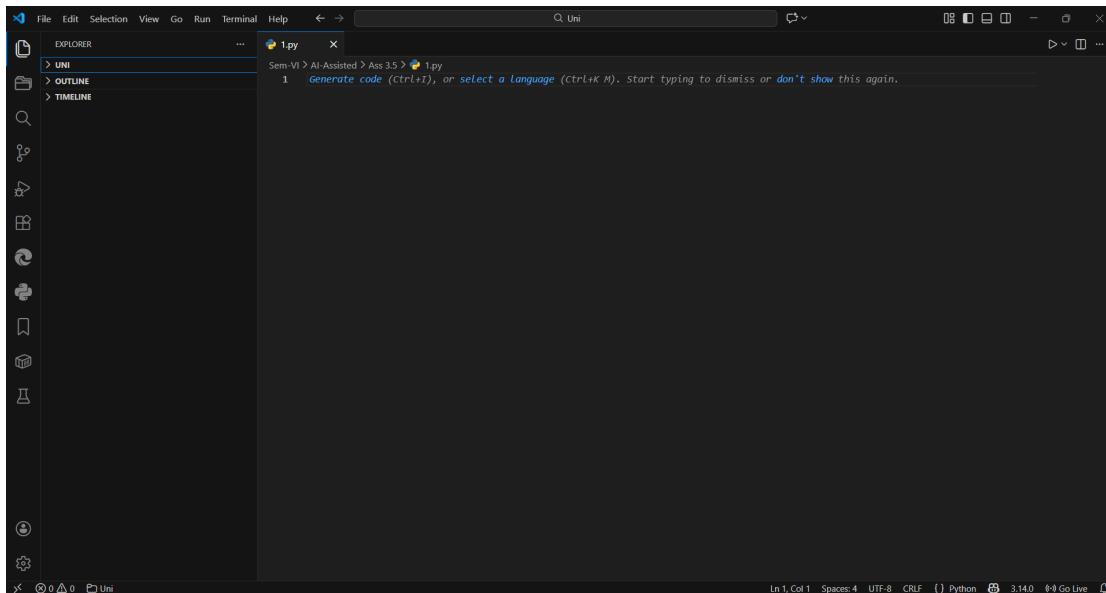
# Lab Assignment 3.5

## 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.



Prompt instructed:

A python function that checks if the given year is a leap year or not.

```
# A python function that checks if the given year is a leap year or not.
def is_leap_year(year):
    # A year is a leap year if it is divisible by 4
    # but not divisible by 100, unless it is also divisible by 400
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return True
    else:
        return False
```

The screenshot shows a code editor window with multiple tabs. The active tab contains Python code for determining if a year is a leap year. The code includes a function definition, user input handling, and print statements. The code is color-coded for syntax.

```
1 # A python function that checks if the given year is a leap year or not.
2
3 def is_leap_year(year):
4     # A year is a leap year if it is divisible by 4
5     # but not divisible by 100, unless it is also divisible by 400
6     if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
7         return True
8     else:
9         return False
10
11 # Take user input
12 year_input = int(input("Enter a year: "))
13 # Call the function and print the result
14 if is_leap_year(year_input):
15     print(f"{year_input} is a leap year.")
16 else:
17     print(f"{year_input} is not a leap year.")
```

AI Generated Code:

```
# A python function that checks if the given year is a leap year or not.

def is_leap_year(year):
    # A year is a leap year if it is divisible by 4
    # but not divisible by 100, unless it is also divisible by 400
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return True
    else:
        return False

# Take user input
year_input = int(input("Enter a year: "))
# Call the function and print the result
if is_leap_year(year_input):
    print(f"{year_input} is a leap year.")
else:
    print(f"{year_input} is not a leap year.)
```

Test with years like 1900, 2000, 2024:

```
Enter a year: 1900
1900 is not a leap year.
Enter a year: 2000
2000 is a leap year.
Enter a year: 2024
2024 is a leap year.
```

Identified Logical Flaws or Missing Conditions:

1. No Input Validation

- The function assumes the input year is always a valid integer.
- It does not handle:
- Non-integer inputs (strings, floats)

- Empty input
  - Invalid values such as 0 or negative years
2. No Error Handling
- If an invalid type (e.g., string or float) is passed, the program will raise a runtime error.
  - The program lacks exception handling (try-except) to prevent crashes.

## 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 it with a zero-shot solution.
- Analyze algorithm efficiency.

### One-Shot Prompt Instructed:

A python function that calculates the GCD of two given numbers.

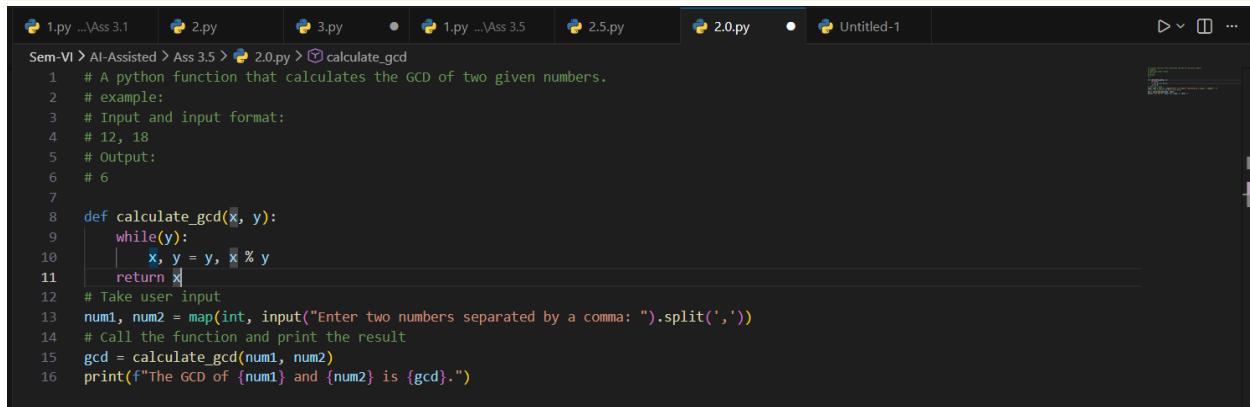
example:

Input and input format:

12, 18

Output:

6



```

Sem-VI > AI-Assisted > Ass 3.5 > calculate_gcd.py
1 # A python function that calculates the GCD of two given numbers.
2 # example:
3 # Input and input format:
4 # 12, 18
5 # Output:
6 # 6
7
8 def calculate_gcd(x, y):
9     while(y):
10         x, y = y, x % y
11     return x
12
13 # Take user input
14 num1, num2 = map(int, input("Enter two numbers separated by a comma: ").split(','))
15 # Call the function and print the result
16 gcd = calculate_gcd(num1, num2)
17 print(f"The GCD of {num1} and {num2} is {gcd}.")

```

### Zero-Shot Prompt Instructed:

A python function that generates the GCD of given two numbers as inputs



Sem-VI > AI-Assisted > Ass 3.5 > 2.py > ...

```
1 # A python function that generates the GCD of given two numbers as inputs
2
3 def compute_gcd(a, b):
4     while b:
5         a, b = b, a % b
6     return a
7
8 # Take user input
9 num1 = int(input("Enter the first number: "))
10 num2 = int(input("Enter the second number: "))
11 # call the function and print the result
12 gcd = compute_gcd(num1, num2)
13 print(f"The GCD of {num1} and {num2} is {gcd}.")
```

## Comparing and analyzing the algorithm efficiency:

- Both have the same time complexity of O(n)
- The difference between them is the way of giving inputs, as the examples influenced and guided the way we give inputs.

## 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:

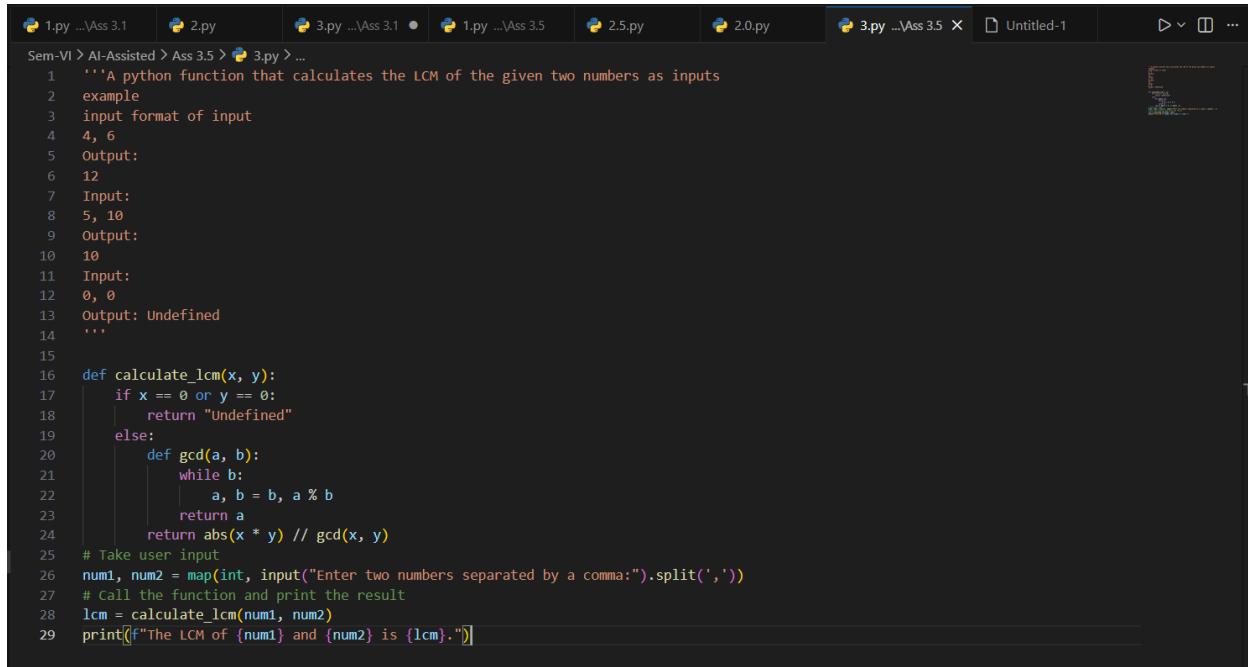
- Input: 4, 6 → Output: 12
- Input: 5, 10 → Output: 10
- Input: 7, 3 → Output: 21

Task:

- Examine how examples guide formula selection.
- Test edge cases.

## Prompt Instructed:

```
A python function that calculates the LCM of the given two numbers as
inputs
example
input format of input
4, 6
Output:
12
Input:
5, 10
Output:
10
Input:
0, 0
Output: Undefined
```



```
1  """A python function that calculates the LCM of the given two numbers as inputs
2  example
3  input format of input
4  4, 6
5  Output:
6  12
7  Input:
8  5, 10
9  Output:
10 10
11 Input:
12 0, 0
13 Output: Undefined
14 """
15
16 def calculate_lcm(x, y):
17     if x == 0 or y == 0:
18         return "Undefined"
19     else:
20         def gcd(a, b):
21             while b:
22                 a, b = b, a % b
23             return a
24         return abs(x * y) // gcd(x, y)
25
26 # Take user input
27 num1, num2 = map(int, input("Enter two numbers separated by a comma:").split(','))
28 # Call the function and print the result
29 lcm = calculate_lcm(num1, num2)
30 print(f"The LCM of {num1} and {num2} is {lcm}.")
```

Testing Edge cases:

```
Enter two numbers separated by a comma: 1, 1
The LCM of 1 and 1 is 1.
```

```
Enter two numbers separated by a comma: 1, 100
The LCM of 1 and 100 is 100.
```

```
Enter two numbers separated by a comma:0, 0
The LCM of 0 and 0 is Undefined.
```

#### 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.

Prompt Instructed:

```
Python Program to convert Binary number to Decimal number
```

1.py ...Ass 3.1 2.py 3.py ...Ass 3.1 1.py ...Ass 3.5 2.5.py 2.0.py 3.py ...Ass 3.5 4.py 1 Sem-VI > AI-Assisted > Ass 3.5 > 4.py > binary\_to\_decimal

```
1 """
2 Python Program to convert Binary number to Decimal number
3 """
4 ✘
5 def binary_to_decimal():
6     # Take user input
7     binary_num = input("Enter a binary number: ")
8
9     # Check if the input is a valid binary number
10    if not all(bit in '01' for bit in binary_num):
11        return "Invalid Binary Number"
12
13    # Convert binary to decimal
14    decimal_num = 0
15    binary_num_reversed = binary_num[::-1]
16
17    for index, digit in enumerate(binary_num_reversed):
18        decimal_num += int(digit) * (2 ** index)
19
20    return f"Decimal Number: {decimal_num}"
```

```
1.py ...Ass 3.1 2.py 3.py ...Ass 3.1 ● 1.py ...Ass 3.5 2.5.py 2.0.py 3.py ...Ass 3.5 4.py ● Un D v ...  
Sem-VI > AI-Assisted > Ass 3.5 > 4.py > ...  
1 1  
2 Python Program to convert Binary number to Decimal number  
3 """  
4  
5 def binary_to_decimal():  
6     # Take user input  
7     binary_num = input("Enter a binary number: ")  
8  
9     # Check if the input is a valid binary number  
10    if not all(bit in '01' for bit in binary_num):  
11        return "Invalid binary number"  
12  
13    # Convert binary to decimal  
14    decimal_num = 0  
15    binary_num_reversed = binary_num[::-1]  
16  
17    for index, digit in enumerate(binary_num_reversed):  
18        decimal_num += int(digit) * (2 ** index)  
19  
20    return f"Decimal Number: {decimal_num}"  
21  
22 # Call the function and print the result  
23 result = binary_to_decimal()  
24 print(result)
```

## AI Generated Code:

```
def binary_to_decimal():
    # Take user input
    binary_num = input("Enter a binary number: ")

    # Check if the input is a valid binary number
    if not all(bit in '01' for bit in binary_num):
        return "Invalid binary number"

    # Convert binary to decimal
    decimal_num = 0
    binary_num_reversed = binary_num[::-1]

    for index, digit in enumerate(binary_num_reversed):
```

```

    decimal_num += int(digit) * (2 ** index)

    return f"Decimal Number: {decimal_num}"

# Call the function and print the result
result = binary_to_decimal()
print(result)

```

### Identified Missing Validation Logic & Other Issues:

1. No Check for Empty Input
  - If the user presses Enter without typing anything, the program:
  - Passes the all() check
  - Incorrectly treats it as a valid binary number
  - This results in a misleading output (Decimal Number: 0).
2. No Handling of Leading or Trailing Spaces
  - Input like " 1010 " fails validation even though it is logically valid.
  - The program does not strip whitespace before validation.
3. No Support for Signed Binary Numbers
  - Binary values with signs (-1010) are not handled.
  - The program assumes only unsigned binary numbers.

### 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.

### One shot prompt instruction:

Python program that converts a Decimal number to Binary number

Example:

Input:

10

Output:

1010

```
1.py ...\\Ass 3.1 2.py 3.py ...\\Ass 3.1 4.py 5.py 6.py 7.py 8.py 9.py 10.py 11.py 12.py 13.py 14.py 15.py 16.py 17.py 18.py 19.py 20.py 21.py 22.py 23.py 24.py 25.py 26.py 27.py 28.py 29.py 30.py 31.py 32.py 33.py 34.py 35.py 36.py 37.py 38.py 39.py
```

Sem-VI > AI-Assisted > Ass 3.5 > 5.0.py > ...

```
1 """
2 Python program that converts a Decimal number to Binary number with zero and negative error handling.
3 Example:
4 Input:
5 10
6 Output:
7 1010
8 """
9
10 def decimal_to_binary():
11     # Take user input
12     decimal_num = input("Enter a decimal number: ")
13
14     # Check if the input is a valid integer
15     if not decimal_num.lstrip("-").isdigit():
16         return "Invalid decimal number"
17
18     # Convert input to integer
19     number = int(decimal_num)
20
21     # Handle the case for 0 explicitly
22     if number == 0:
23         return "Binary Number: 0"
24
25     # Handle negative numbers
26     if number < 0:
27         return "Negative numbers cannot be converted to binary"
28
29     # Convert decimal to binary
30     binary_num = ""
31     while number > 0:
32         binary_num = str(number % 2) + binary_num
33         number //= 2
34
35     return f"Binary Number: {binary_num}"
36
37 # Call the function and print the result
38 result = decimal_to_binary()
39 print(result)
```

## Zero-Shot Prompt Instruction:

```
1.py ...\\Ass 3.1 2.py 3.py ...\\Ass 3.5 4.py 5.py 6.py 7.py 8.py 9.py 10.py 11.py 12.py 13.py 14.py 15.py 16.py 17.py 18.py 19.py 20.py 21.py 22.py 23.py 24.py 25.py 26.py 27.py 28.py 29.py 30.py
```

Sem-VI > AI-Assisted > Ass 3.5 > 5.5.py > ...

```
1 """
2 Python program that converts a Decimal number to Binary number
3 """
4
5 def decimal_to_binary():
6     # Take user input
7     decimal_num = input("Enter a decimal number: ")
8
9     # Check if the input is a valid integer
10    if not decimal_num.isdigit():
11        return "Invalid decimal number"
12
13    # Convert input to integer
14    number = int(decimal_num)
15
16    # Handle the case for 0 explicitly
17    if number == 0:
18        return "Binary Number: 0"
19
20    # Convert decimal to binary
21    binary_num = ""
22    while number > 0:
23        binary_num = str(number % 2) + binary_num
24        number //= 2
25
26    return f"Binary Number: {binary_num}"
27
28 # Call the function and print the result
29 result = decimal_to_binary()
30 print(result)
```

Analyzing handling of zero and negative numbers in one shot prompt:

Enter a decimal number: 0

Binary Number: 0

Enter a decimal number: -1

Negative numbers cannot be converted to binary

## 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

Prompt Instructed:

```
Python program to check whether a number is a Harshad number or not
```

Examples:

Input:

**18**

Output:

Harshad Number

Input:

**21**

Output:

Not a Harshad Number

Input:

**-19**

Output:

Negative numbers cannot be Harshad Numbers

Input:

**I9**

Output:

Invalid input

```

1  """
2  Python program to check whether a number is a Harshad number or not
3  # Parameters
4  # Input:
5  #   19
6  # Output:
7  #   Harshad Number
8  # Input:
9  #   19
10 # Output:
11 #   Not a Harshad Number.
12 # Input:
13 #   19
14 # Output:
15 #   Negative numbers cannot be Harshad Numbers
16 # Input:
17 #   19
18 # Output:
19 #   Invalid input
20 # Input:
21
22 def is_harshad_number():
23     # Take user input
24     num = input("Enter an integer: ")
25
26     # Check if the input is a valid integer
27     if not num.isdigit() or num == "0":
28         return "Invalid input"
29
30     # Convert input to integer
31     number = int(num)
32
33     # Handle negative numbers
34     if number < 0:
35         return "Negative numbers cannot be Harshad Numbers"
36
37     # Calculate the sum of the digits
38     sum_of_digits = sum(int(digit) for digit in num)
39
40     # Check if the number is divisible by the sum of its digits
41     if number % sum_of_digits == 0:
42         return "Harshad Number"
43     else:
44         return "Not a Harshad Number"
45
46 # Call the function and print the result
47 result = is_harshad_number()
48 print(result)

```

Testing Boundary Conditions:

Enter an integer: 19

Invalid input

Enter an integer: 100

Harshad Number

Enter an integer: 1

Harshad Number

Evaluating Robustness of the code:

#### 1. Handling of Valid Inputs

- Correctly identifies Harshad and non-Harshad numbers.
- Works for small, large, and multi-digit integers.

#### 2. Handling of Invalid Inputs

- Rejects non-numeric inputs (19, abc).
- Rejects floating-point values (12.5).
- Handles empty input safely.

#### 3. Handling of Negative Numbers

- Explicitly checks for negative values.
- Provides a meaningful error message instead of incorrect computation.

#### 4. Division Safety

- Prevents division by zero when digit sum is zero (e.g., input 0).
- Ensures the program does not crash due to runtime errors.

#### 5. Boundary Coverage

Covers:

- Lower boundary (1)
- Upper boundary (large numbers)
- Invalid boundary (negative, non-numeric)