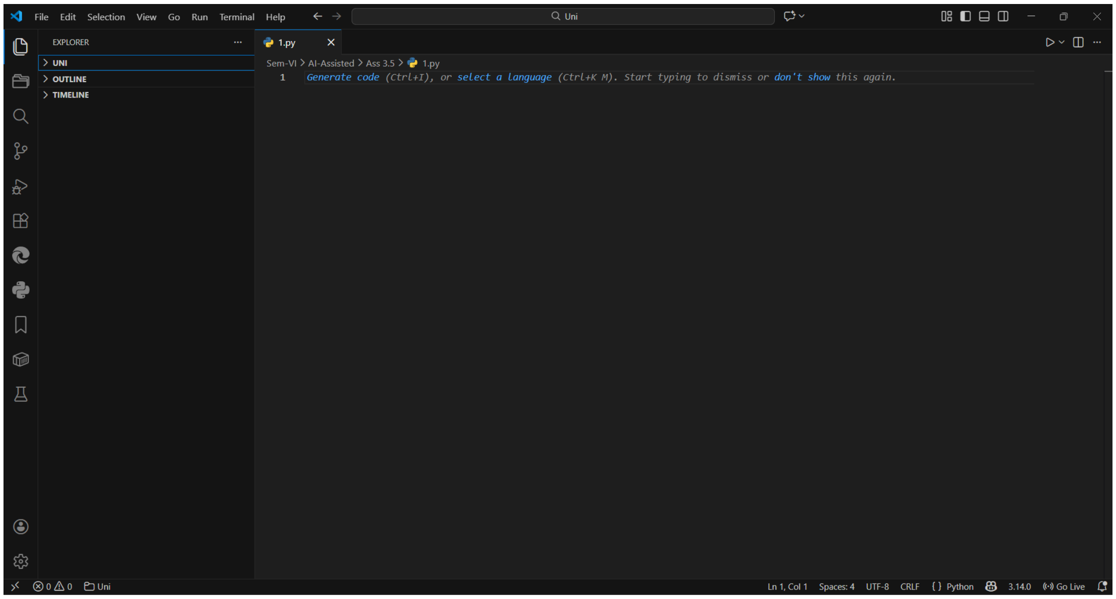
Lab Assignment 3.5

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:

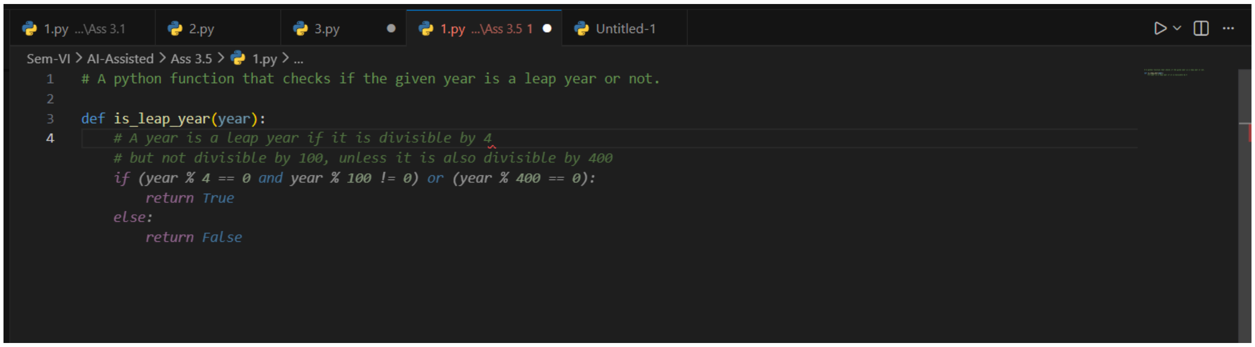
- Record the AI-generated code.
- Test with years like 1900, 2000, 2024.
- Identify logical flaws or missing conditions.



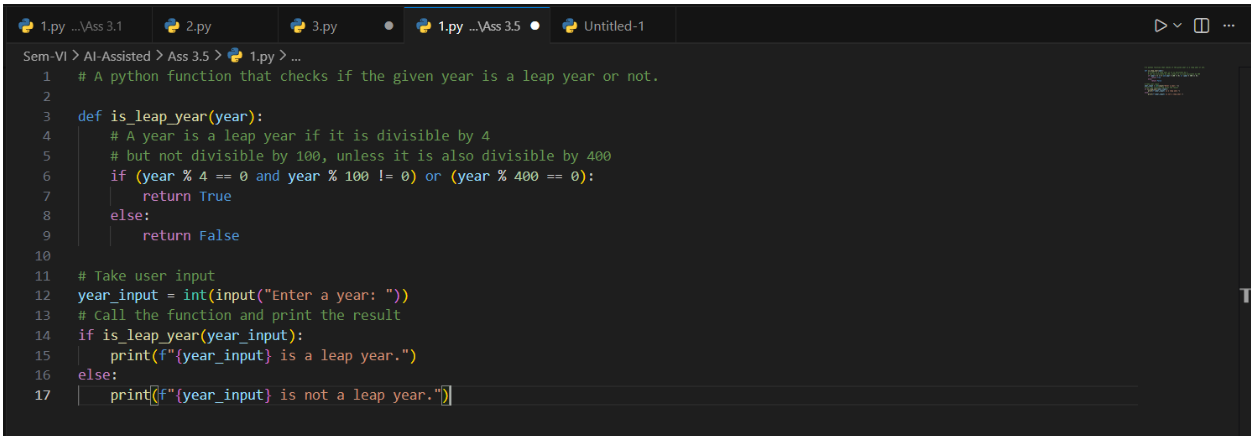
```
Generate code (Ctrl+I), or select a language (Ctrl+K M). Start typing to dismiss or don't show this again.
```

Prompt instructed:

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



```python
# 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
```

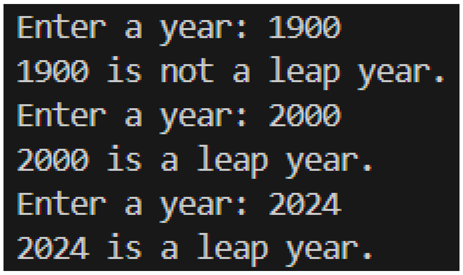


```python
# A python function that checks if the given year is a leap year or not.

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.")
```

AI Generated Code:

```python
# 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:

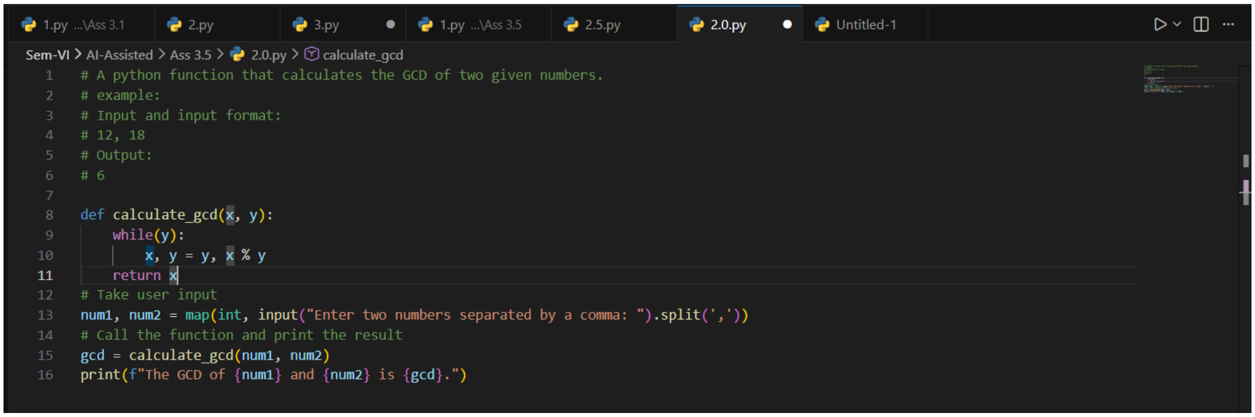
- Input: 12, 18 → Output: 6

Task:

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

One-Shot Prompt Instructed:

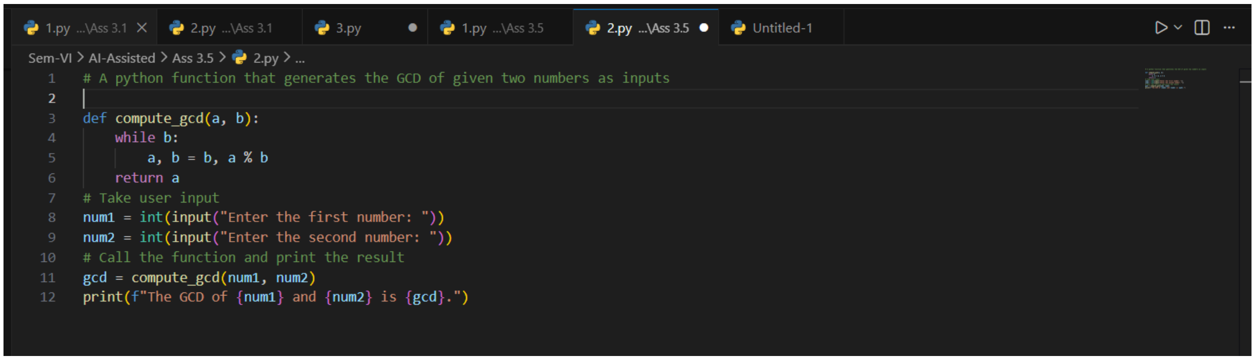
```
A python function that calculates the GCD of two given numbers.
example:
Input and input format:
12, 18
Output:
6
```



```python
# A python function that calculates the GCD of two given numbers.
# example:
# Input and input format:
# 12, 18
# Output:
# 6

def calculate_gcd(x, y):
    while(y):
        x, y = y, x % y
    return x
# Take user input
num1, num2 = map(int, input("Enter two numbers separated by a comma: ").split(','))
# Call the function and print the result
gcd = calculate_gcd(num1, num2)
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
```



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

def compute_gcd(a, b):
    while b:
        a, b = b, a % b
    return a
# Take user input
num1 = int(input("Enter the first number: "))
num2 = int(input("Enter the second number: "))
# Call the function and print the result
gcd = compute_gcd(num1, num2)
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:

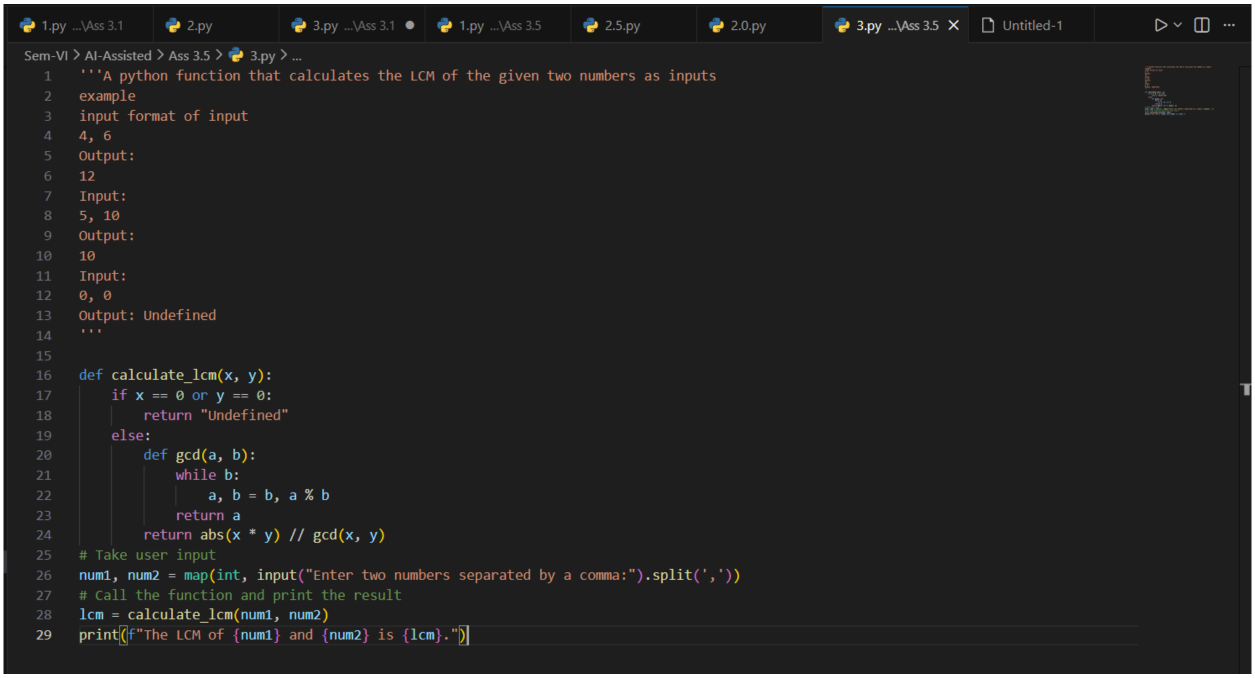
- Input: 4, 6 → Output: 12
- Input: 5, 10 → Output: 10
- Input: 7, 3 → Output: 21

Task:

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

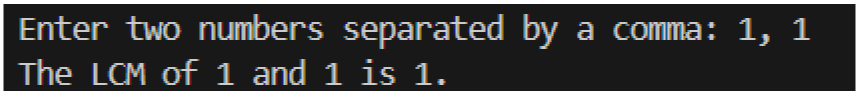
Prompt Instructed:

```
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
```

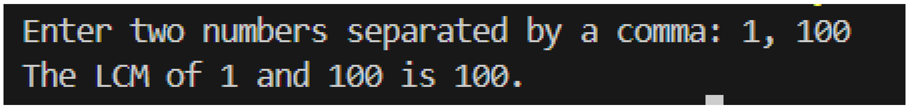


```python
'''A python function that calculates the LCM of the given two numbers as inputs
example
input format of input
4, 6
Output:
12
Input:
5, 10
Output:
10
Input:
0, 0
Output: Undefined
'''

def calculate_lcm(x, y):
    if x == 0 or y == 0:
        return "Undefined"
    else:
        def gcd(a, b):
            while b:
                a, b = b, a % b
            return a
        return abs(x * y) // gcd(x, y)
# Take user input
num1, num2 = map(int, input("Enter two numbers separated by a comma:").split(','))
# Call the function and print the result
lcm = calculate_lcm(num1, num2)
print(f"The LCM of {num1} and {num2} is {lcm}.")
```

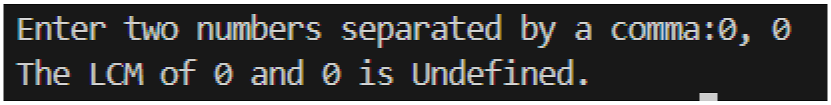
Testing Edge cases:



```
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)

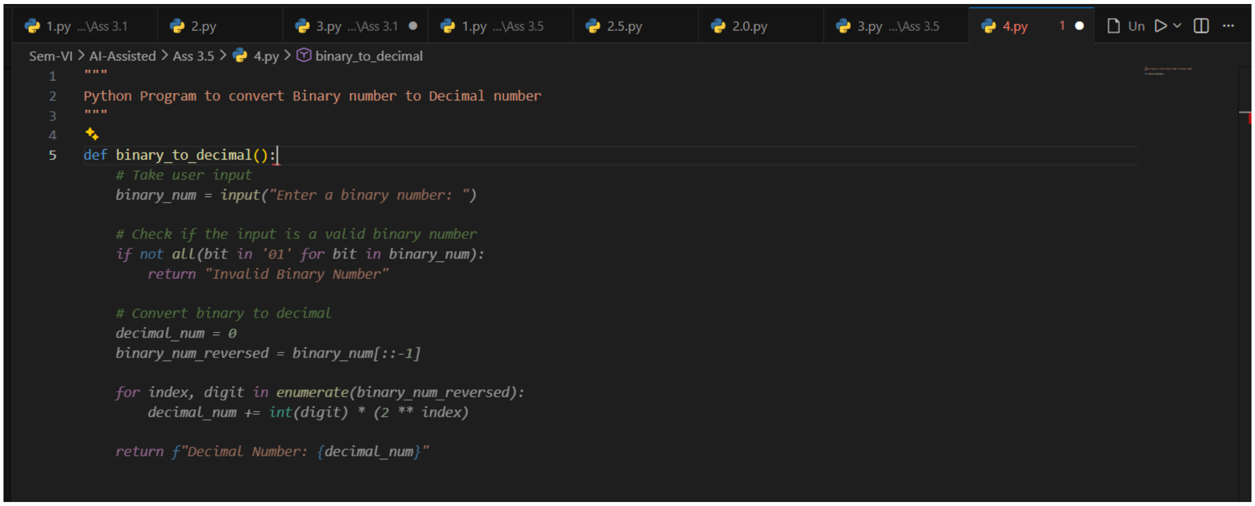
Write a zero-shot prompt to generate a Python function that converts a binary number to decimal.

Task:

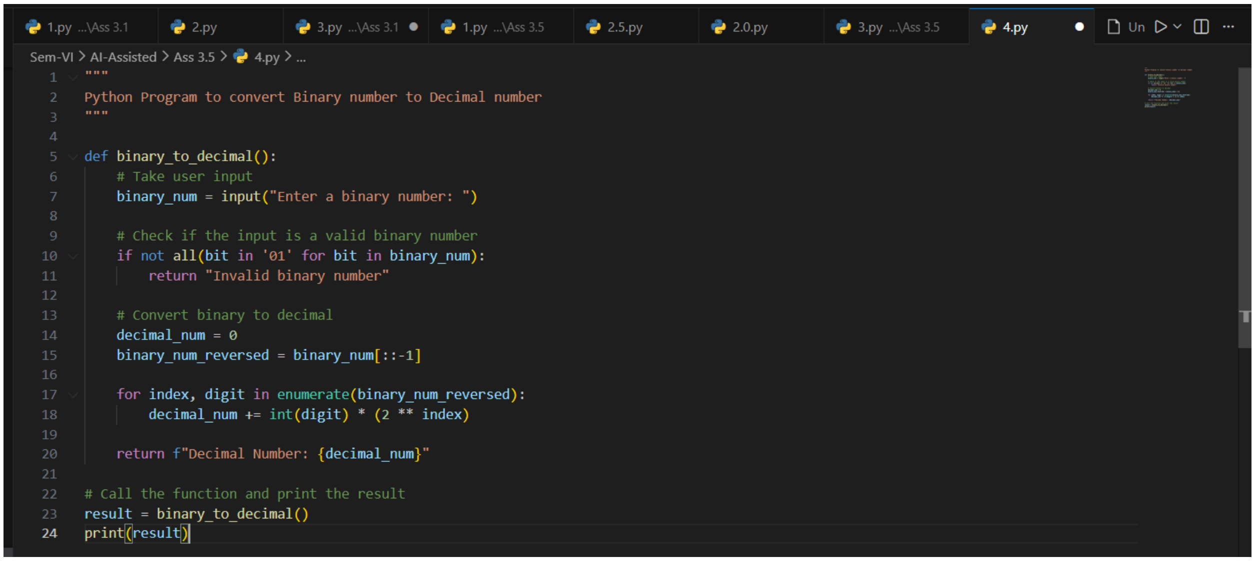
- Test with valid and invalid binary inputs.
- Identify missing validation logic.

Prompt Instructed:

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



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

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}"
```



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

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)
```

AI Generated Code:

```python
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:

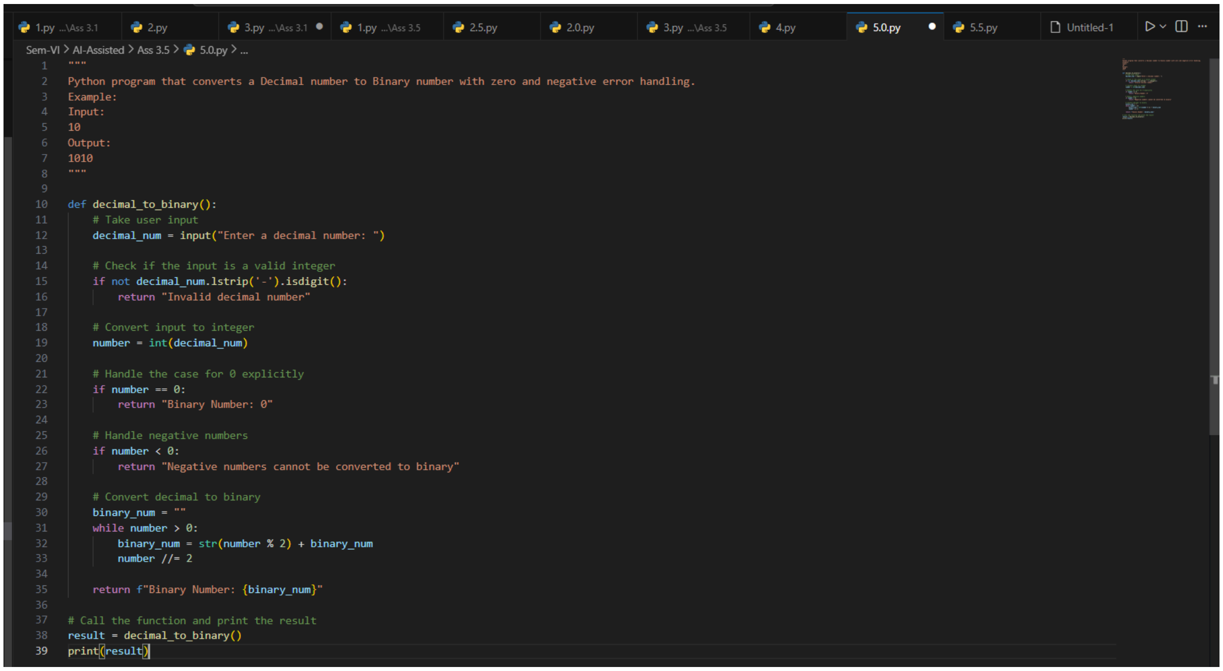
- Input: 10 → Output: 1010

Task:

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

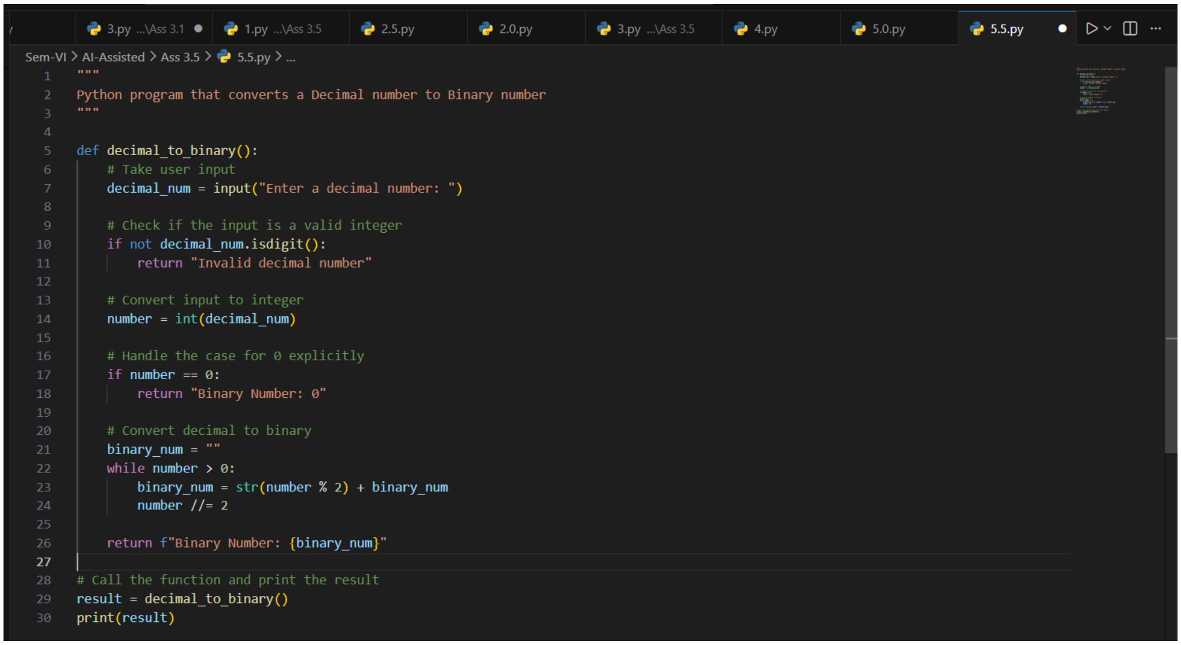
One shot prompt instruction:

```
Python program that converts a Decimal number to Binary number
Example:
Input:
10
Output:
1010
```



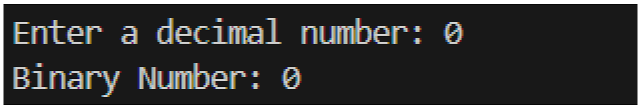
```python
"""
Python program that converts a Decimal number to Binary number with zero and negative error handling.
Example:
Input:
10
Output:
1010
"""

def decimal_to_binary():
    # Take user input
    decimal_num = input("Enter a decimal number: ")

    # Check if the input is a valid integer
    if not decimal_num.lstrip('-').isdigit():
        return "Invalid decimal number"

    # Convert input to integer
    number = int(decimal_num)

    # Handle the case for 0 explicitly
    if number == 0:
        return "Binary Number: 0"

    # Handle negative numbers
    if number < 0:
        return "Negative numbers cannot be converted to binary"

    # Convert decimal to binary
    binary_num = ""
    while number > 0:
        binary_num = str(number % 2) + binary_num
        number //= 2

    return f"Binary Number: {binary_num}"

# Call the function and print the result
result = decimal_to_binary()
print(result)
```

Zero-Shot Prompt Instruction:

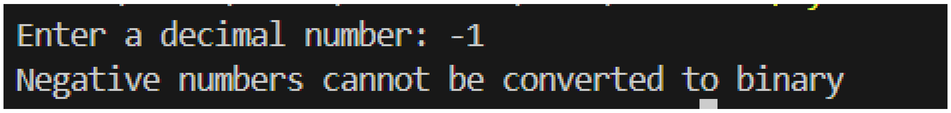


```python
"""
Python program that converts a Decimal number to Binary number
"""

def decimal_to_binary():
    # Take user input
    decimal_num = input("Enter a decimal number: ")

    # Check if the input is a valid integer
    if not decimal_num.isdigit():
        return "Invalid decimal number"

    # Convert input to integer
    number = int(decimal_num)

    # Handle the case for 0 explicitly
    if number == 0:
        return "Binary Number: 0"

    # Convert decimal to binary
    binary_num = ""
    while number > 0:
        binary_num = str(number % 2) + binary_num
        number //= 2

    return f"Binary Number: {binary_num}"

# Call the function and print the result
result = decimal_to_binary()
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:

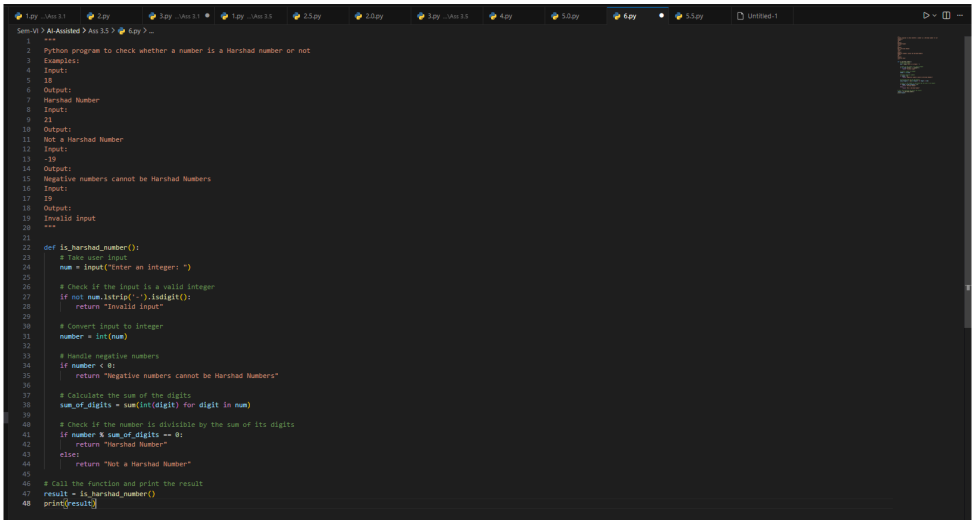
- Input: 18 → Output: Harshad Number
- Input: 21 → Output: Harshad Number
- Input: 19 → Output: Not a Harshad Number

Task:

- Test boundary conditions.
- Evaluate robustness

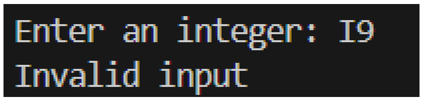
Prompt Instructed:

```
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
```

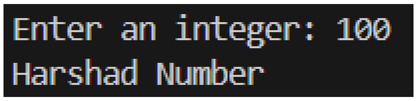


```python
"""
Python program to check whether a number is a Harshad number or not
Examples:
Input:
18
Output:
Harshad Number
Input:
21
Output:
Not a Harshad Number
Input:
-19
Output:
Negative numbers cannot be Harshad Numbers
Input:
I9
Output:
Invalid input
"""

def is_harshad_number():
    # Take user input
    num = input("Enter an integer: ")

    # Check if the input is a valid integer
    if not num.lstrip('-').isdigit():
        return "Invalid input"

    # Convert input to integer
    number = int(num)

    # Handle negative numbers
    if number < 0:
        return "Negative numbers cannot be Harshad Numbers"

    # Calculate the sum of the digits
    sum_of_digits = sum(int(digit) for digit in num)

    # Check if the number is divisible by the sum of its digits
    if number % sum_of_digits == 0:
        return "Harshad Number"
    else:
        return "Not a Harshad Number"

# Call the function and print the result
result = is_harshad_number()
print(result)
```

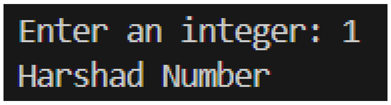
Testing Boundary Conditions:



```
Enter an integer: I9
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 (I9, 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)