

# Lab Assignment - 3.5

Name : P. Gouri Prasad Varma

Hall.t.No. : 2303A51035

Batch No. : 01

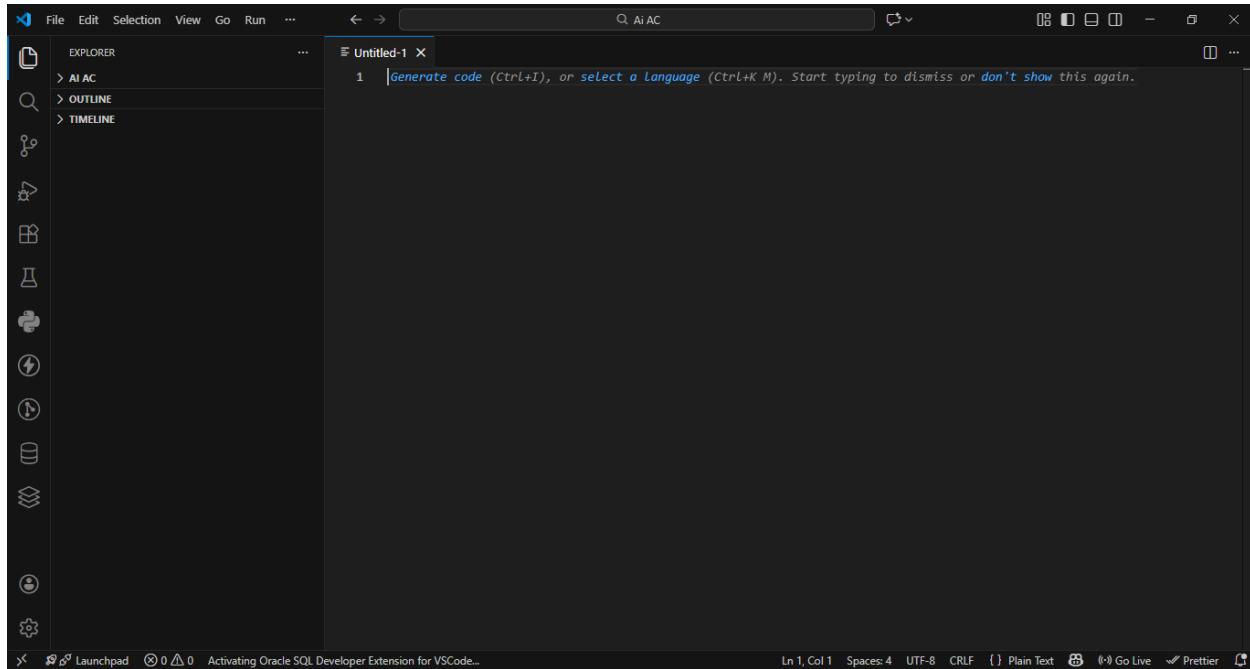
## Question 1: Zero-Shot Prompting (Leap Year Check)

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

### Task:

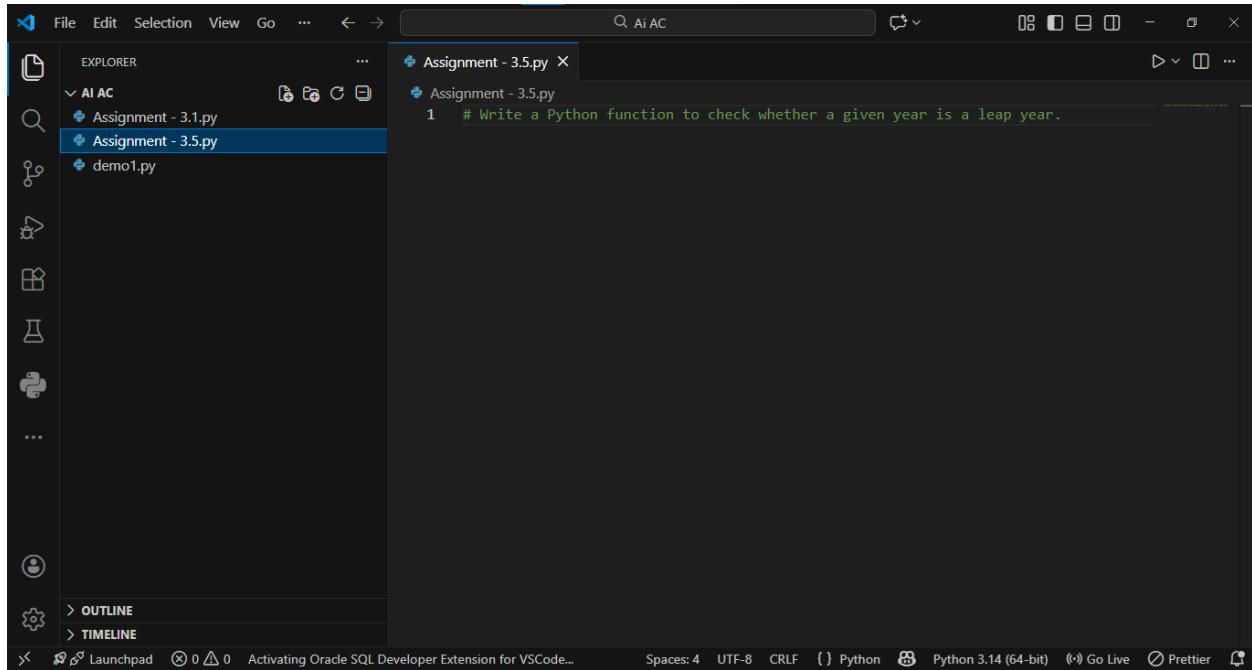
- Record the AI-generated code.
- Test with years like 1900, 2000, 2024.
- Identify logical flaws or missing conditions.

### Copilot VSCode Workspace :



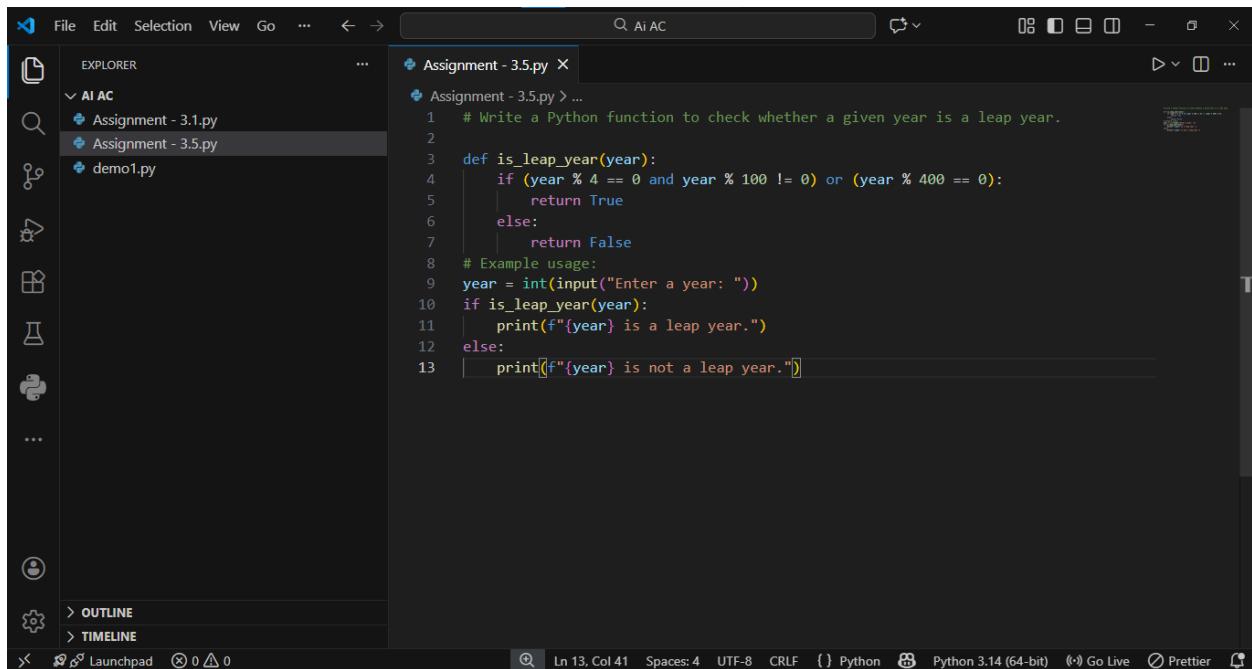
## Prompt Used (Zero-Shot):

Write a Python function to check whether a given year is a leap year.



A screenshot of the Visual Studio Code interface. The left sidebar shows the Explorer view with files: Assignment - 3.1.py, Assignment - 3.5.py (selected), and demo1.py. The center main area has a search bar at the top with the text "Ai AC". Below it is a code editor window titled "Assignment - 3.5.py" containing the prompt: "# Write a Python function to check whether a given year is a leap year.". The bottom status bar shows "Activating Oracle SQL Developer Extension for VSCode..." and other settings like "Spaces: 4", "UTF-8", "CRLF", "Python", "Python 3.14 (64-bit)", "Go Live", and "Prettier".

## Copilot Suggested Code :



A screenshot of the Visual Studio Code interface, identical to the previous one but with the code editor content filled by the AI copilot. The code now includes a function definition:

```
1 # Write a Python function to check whether a given year is a leap year.
2
3 def is_leap_year(year):
4     if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
5         return True
6     else:
7         return False
8 # Example usage:
9 year = int(input("Enter a year: "))
10 if is_leap_year(year):
11     print(f"{year} is a leap year.")
12 else:
13     print(f"{year} is not a leap year.")
```

The rest of the interface is the same, showing the Explorer view with the same files and the status bar indicating the extension is activating.

## **Input / Output test cases screenshot :**

```
● PS C:\Btech\3 - 2\Ai AC> & C:\Python314\python.exe "c:/Btech/3 - 2/Ai AC/Assignment - 3.5.py"
 1900 is not a leap year.
 PS C:\Btech\3 - 2\Ai AC> & C:\Python314\python.exe "c:/Btech/3 - 2/Ai AC/Assignment - 3.5.py"
● 2000 is a leap year.
 PS C:\Btech\3 - 2\Ai AC> & C:\Python314\python.exe "c:/Btech/3 - 2/Ai AC/Assignment - 3.5.py"
 Enter a year: 2024
● 2024 is a leap year.
 PS C:\Btech\3 - 2\Ai AC> & C:\Python314\python.exe "c:/Btech/3 - 2/Ai AC/Assignment - 3.5.py"
 Enter a year: 2023
● 2023 is not a leap year.
 PS C:\Btech\3 - 2\Ai AC> & C:\Python314\python.exe "c:/Btech/3 - 2/Ai AC/Assignment - 3.5.py"
● Enter a year: 2100
 2100 is not a leap year.
```

## **Observation & Logical Flaws :**

- Fails for **century years**
- Missing condition:

Years divisible by **100** are **not leap years** unless divisible by **400**.

## **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.

### **Prompt Used (One-Shot):**

Write a Python function to find GCD of two numbers.

**Example:** Input: 12, 18 → Output: 6

A screenshot of a dark-themed code editor window titled "Assignment - 3.5.py". The code in the editor is:

```
14
15 # "Write a Python function to find GCD of two numbers.
16 # Example: Input: 12, 18 → Output: 6"
```

The status bar at the bottom shows: Ln 16, Col 38, Spaces:4, UTF-8, CRLF, Python, Python 3.14 (64-bit), Go Live, Prettier.

## Copilot Suggested Code :

A screenshot of a dark-themed code editor window titled "Assignment - 3.5.py > ...". The code in the editor is:

```
14
15 # "Write a Python function to find GCD of two numbers.
16 # Example: Input: 12, 18 → Output: 6"
17
18 v def gcd(a, b):
19   v   while b:
20     |     a, b = b, a % b
21   return a
22 # Example usage:
23 num1 = int(input("Enter first number: "))
24 num2 = int(input("Enter second number: "))
25 print(f"The GCD of {num1} and {num2} is {gcd(num1, num2)}")
```

The status bar at the bottom shows: Ln 25, Col 60, Spaces:4, UTF-8, CRLF, Python, Python 3.14 (64-bit), Go Live, Prettier.

## **Input / Output test cases screenshot :**

```
● PS C:\Btech\3 - 2\Ai AC> & C:\Python314\python.exe "c:/Btech/3 - 2/Ai AC/Assignment - 3.5.py"
  Enter first number: 12
  Enter second number: 18
  The GCD of 12 and 18 is 6
○ PS C:\Btech\3 - 2\Ai AC>
```

## **Comparison with Zero-Shot :**

<b>Aspect</b>	<b>Zero-Shot</b>	<b>One-Shot</b>
Clarity	Medium	High
Algorithm	Trial-based	Euclidean
Efficiency	Low	High

## **Efficiency Analysis :**

- Time Complexity: **O(log min(a, b))**
- Very efficient for large numbers

## **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 Used (Few-Shot Prompt):

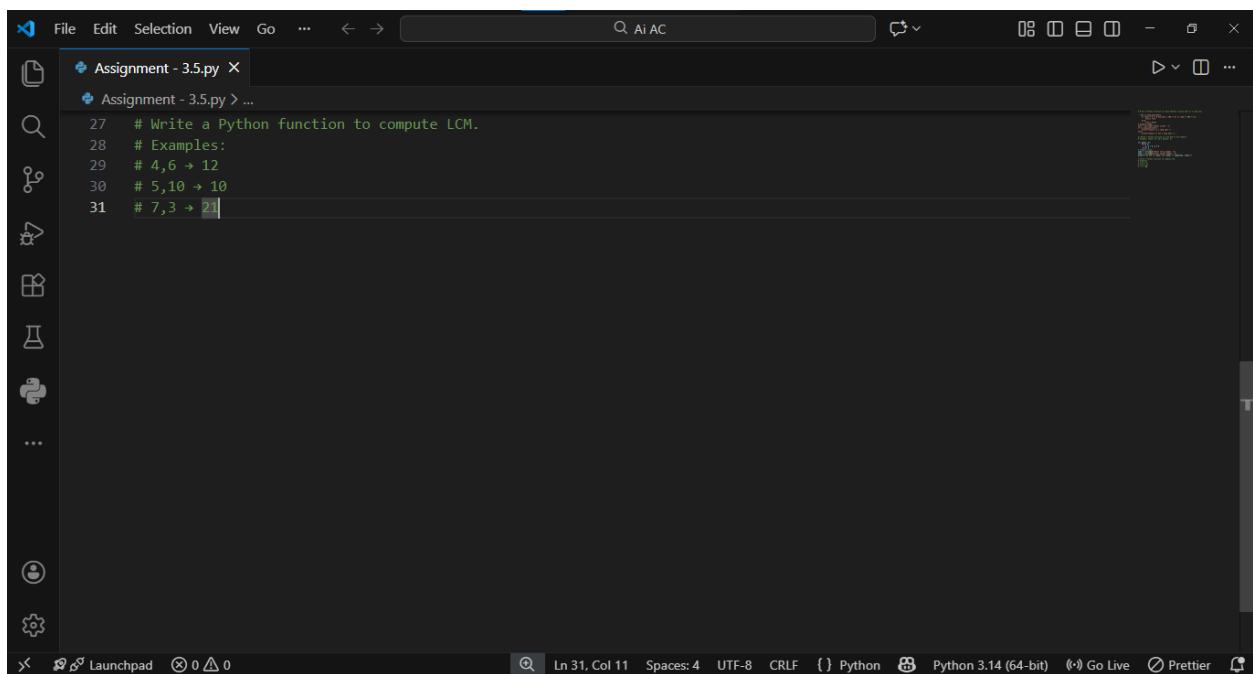
Write a Python function to compute LCM.

## Examples:

4,6 → 12

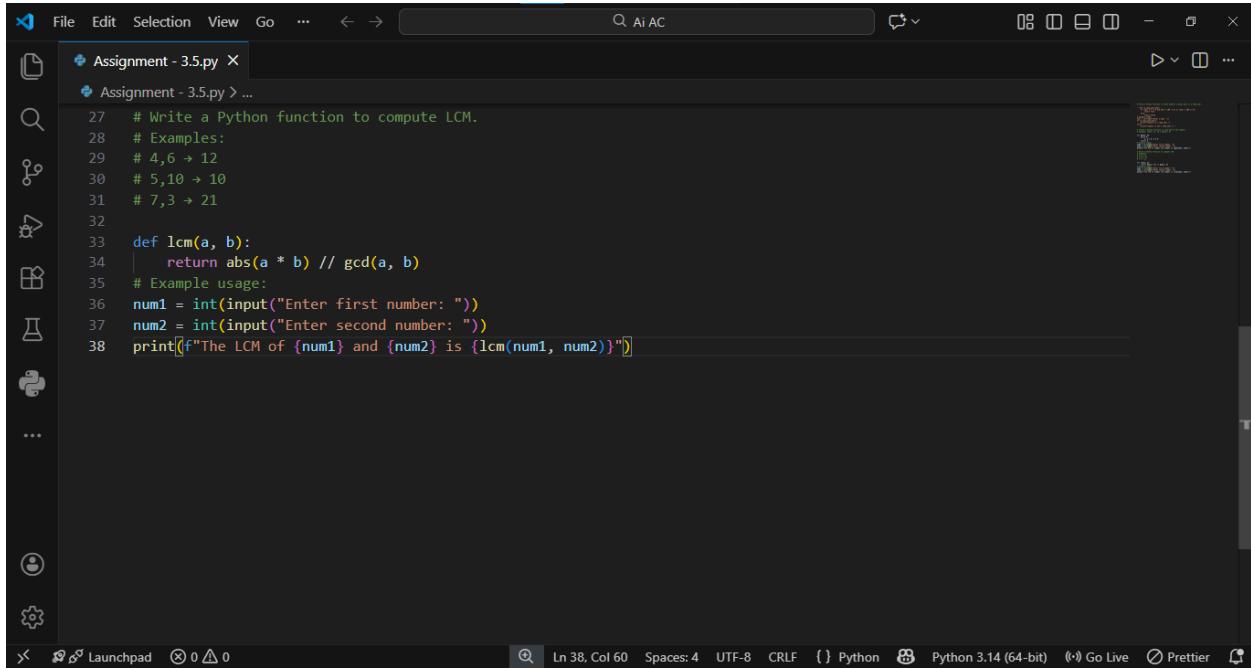
5,10 → 10

7,3 → 21



```
# Write a Python function to compute LCM.
# Examples:
# 4,6 → 12
# 5,10 → 10
# 7,3 → 21
```

## Copilot Suggested Code :

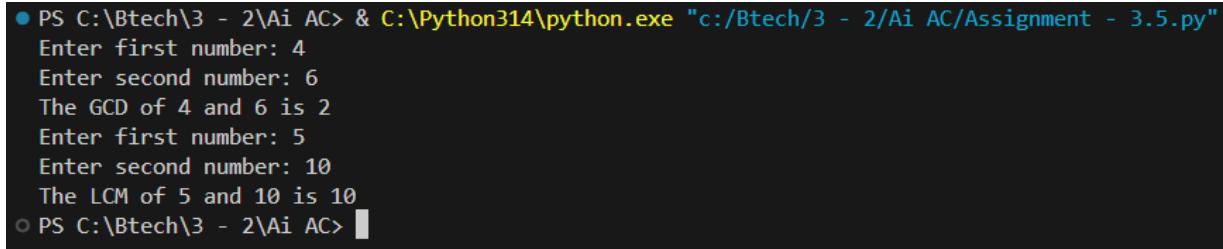


A screenshot of a code editor window titled "Assignment - 3.5.py". The code is as follows:

```
27 # Write a Python function to compute LCM.
28 # Examples:
29 # 4,6 → 12
30 # 5,10 → 10
31 # 7,3 → 21
32
33 def lcm(a, b):
34     return abs(a * b) // gcd(a, b)
35 # Example usage:
36 num1 = int(input("Enter first number: "))
37 num2 = int(input("Enter second number: "))
38 print(f"The LCM of {num1} and {num2} is {lcm(num1, num2)}")
```

The editor interface includes a sidebar with icons for file operations, a search bar, and a status bar at the bottom showing file path, line/col, encoding, and Python version.

## Input / Output test cases screenshot :



```
● PS C:\Btech\3 - 2\Ai AC> & C:\Python314\python.exe "c:/Btech/3 - 2/Ai AC/Assignment - 3.5.py"
Enter first number: 4
Enter second number: 6
The GCD of 4 and 6 is 2
Enter first number: 5
Enter second number: 10
The LCM of 5 and 10 is 10
○ PS C:\Btech\3 - 2\Ai AC> █
```

## How Examples Guided the Model :

- Correctly inferred  $\text{LCM} = (a \times b) / \text{GCD}$
- Used efficient GCD internally

## Edge Case Testing :

```
print(lcm(0, 5))    # Output: 0
print(lcm(1, 1))    # Output: 1
```

## Question 4: Zero-Shot Prompting (Binary to Decimal Conversion)

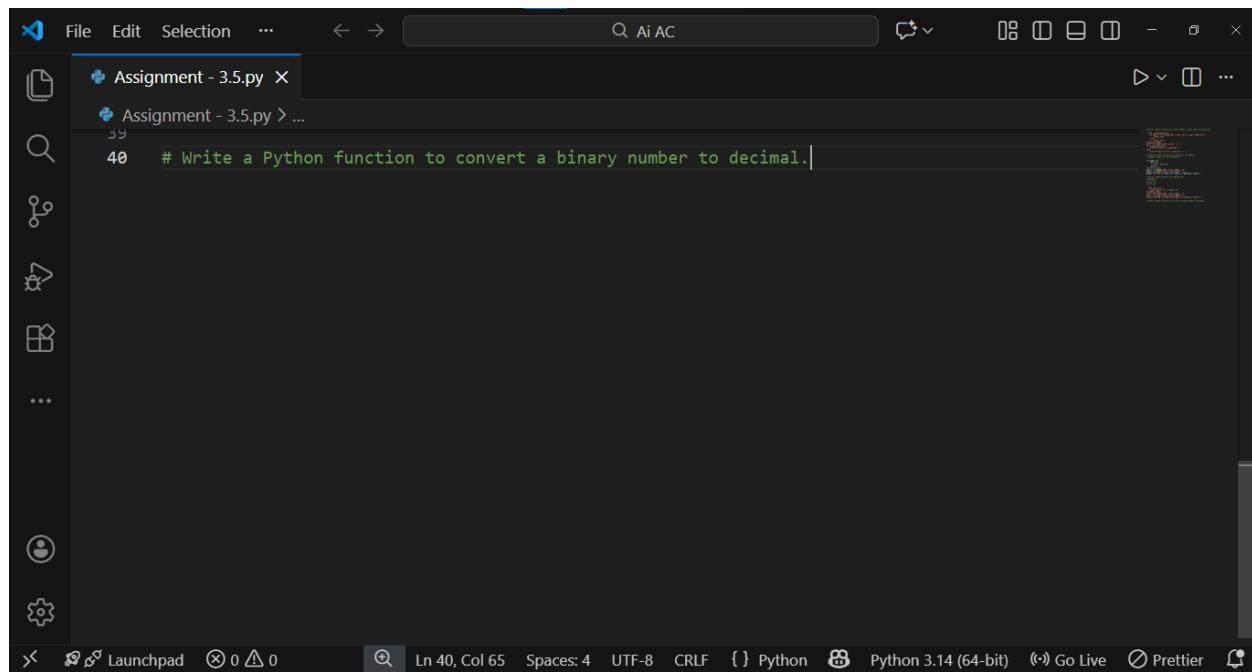
Write a zero-shot prompt to generate a Python function that converts a binary number to decimal.

### Task:

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

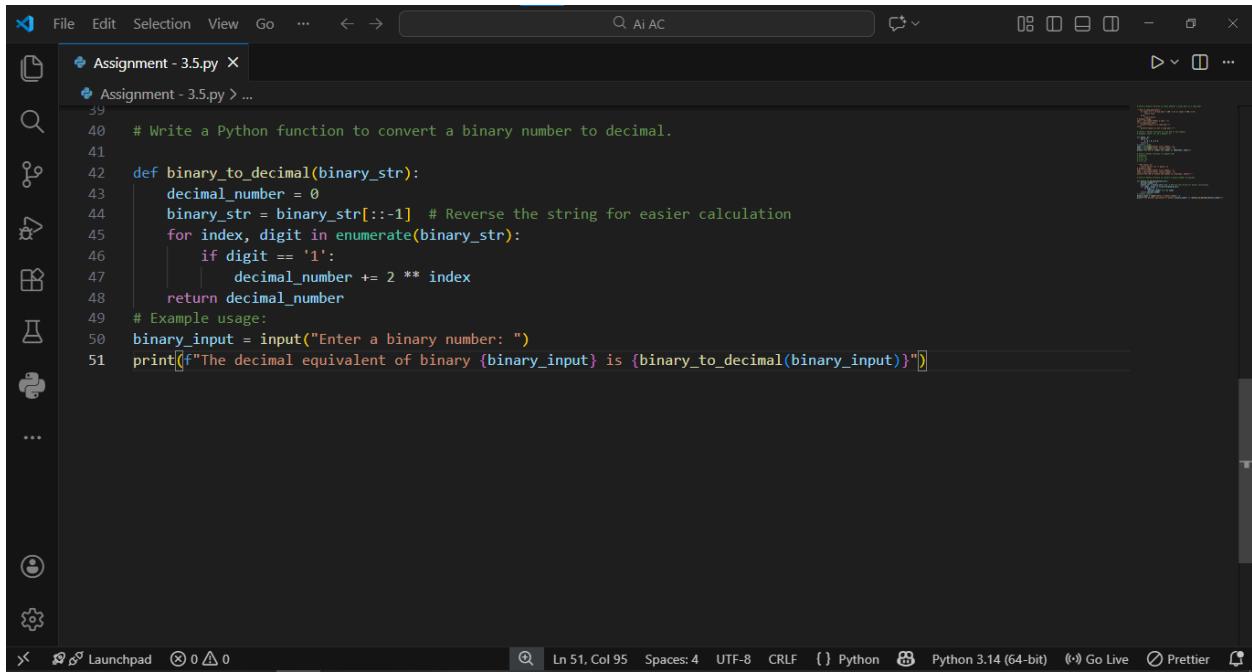
### Prompt Used (Zero-Shot):

Write a Python function to convert a binary number to decimal.



A screenshot of a code editor window titled "Assignment - 3.5.py". The code editor shows a single line of Python code: "40 # Write a Python function to convert a binary number to decimal.". The interface includes a toolbar with icons for file operations, a search bar, and various status indicators at the bottom.

## Copilot Suggested Code :

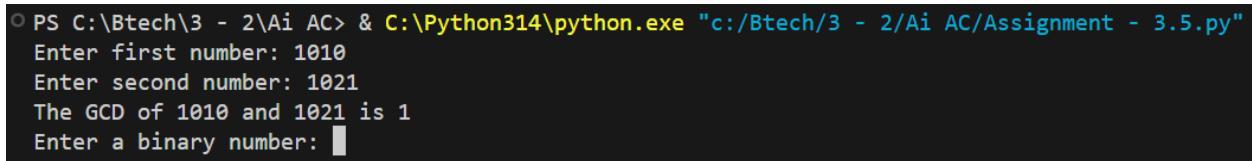


A screenshot of a code editor window titled "Assignment - 3.5.py". The code is as follows:

```
39 # Write a Python function to convert a binary number to decimal.
40
41 def binary_to_decimal(binary_str):
42     decimal_number = 0
43     binary_str = binary_str[::-1] # Reverse the string for easier calculation
44     for index, digit in enumerate(binary_str):
45         if digit == '1':
46             decimal_number += 2 ** index
47     return decimal_number
48
49 # Example usage:
50 binary_input = input("Enter a binary number: ")
51 print(f"The decimal equivalent of binary {binary_input} is {binary_to_decimal(binary_input)}")
```

The editor has a dark theme with light-colored syntax highlighting. The status bar at the bottom shows "Ln 51, Col 95" and "Spaces: 4".

## Input / Output test cases screenshot :



A screenshot of a terminal window showing the following interaction:

```
PS C:\Btech\3 - 2\Ai AC> & C:\Python314\python.exe "c:/Btech/3 - 2/Ai AC/Assignment - 3.5.py"
Enter first number: 1010
Enter second number: 1021
The GCD of 1010 and 1021 is 1
Enter a binary number: 
```

## Testing :

```
print(binary_to_decimal("1010"))    # Output: 10
print(binary_to_decimal("1021"))    # Error
```

## Missing Validation :

- No check for invalid binary digits

## 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

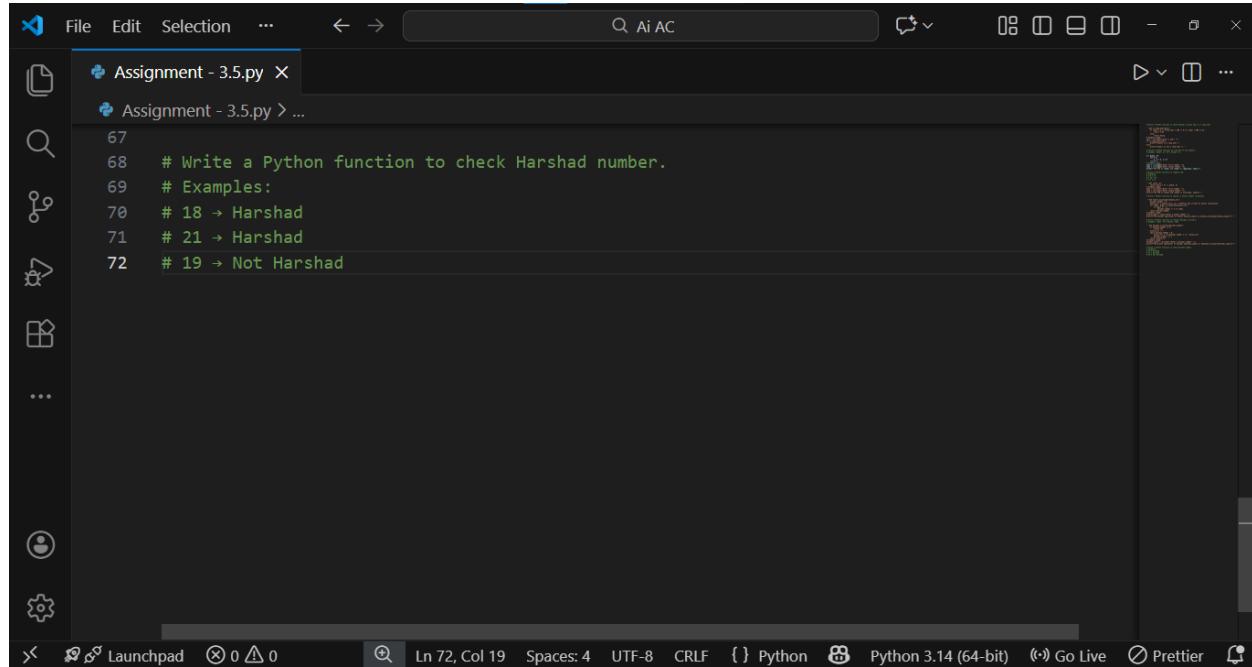
### Task:

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

### Prompt Used (One-Shot):

Write a Python function to convert decimal to binary.

**Example:** Input: 10 → Output: 1010

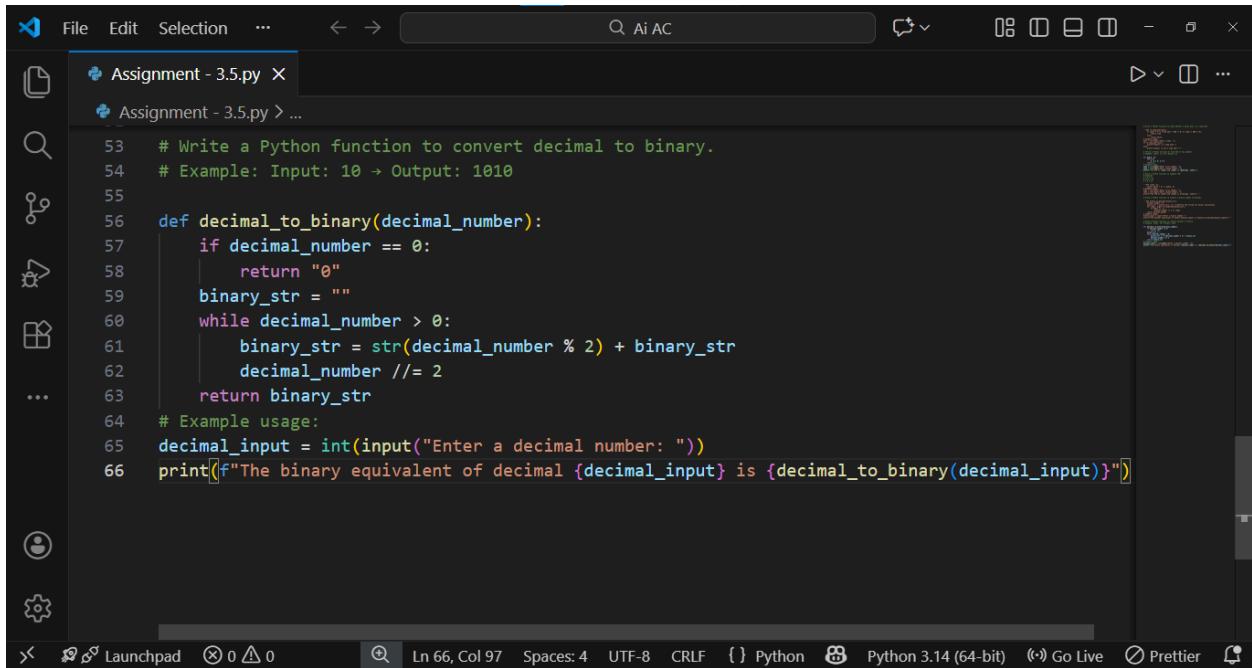


The screenshot shows a code editor interface with a dark theme. On the left is a sidebar with various icons: a file folder, a magnifying glass, a circular arrow, a star, a document, and three dots. The main area displays a Python script named 'Assignment - 3.5.py'. The code is as follows:

```
67
68     # Write a Python function to check Harshad number.
69     # Examples:
70     # 18 → Harshad
71     # 21 → Harshad
72     # 19 → Not Harshad
```

The status bar at the bottom shows the following information: 'Launchpad' icon, '(x) 0 ▲ 0' icon, a search icon, 'Ln 72, Col 19', 'Spaces: 4', 'UTF-8', 'CRLF', a Python icon, 'Python 3.14 (64-bit)', '(o) Go Live' icon, a Prettier icon, and a GitHub icon.

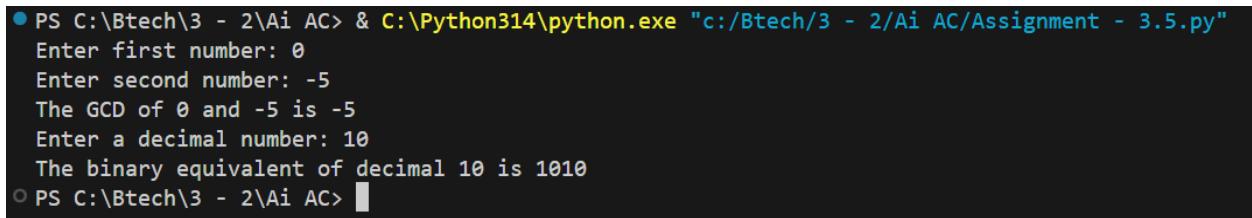
## Copilot Suggested Code :



A screenshot of a code editor window titled "Assignment - 3.5.py". The code is a Python function named "decimal\_to\_binary" that converts a decimal number to binary. It includes a docstring, a check for zero, a loop to build the binary string from remainders, and an example usage with user input and output.

```
53 # Write a Python function to convert decimal to binary.
54 # Example: Input: 10 → Output: 1010
55
56 def decimal_to_binary(decimal_number):
57     if decimal_number == 0:
58         return "0"
59     binary_str = ""
60     while decimal_number > 0:
61         binary_str = str(decimal_number % 2) + binary_str
62         decimal_number //= 2
63     return binary_str
64 # Example usage:
65 decimal_input = int(input("Enter a decimal number: "))
66 print(f"The binary equivalent of decimal {decimal_input} is {decimal_to_binary(decimal_input)}")
```

## Input / Output test cases screenshot :



A screenshot of a terminal window showing the execution of the "decimal\_to\_binary" function. The user enters several decimal numbers, and the program prints their binary equivalents. The terminal also shows the command used to run the script.

```
● PS C:\Btech\3 - 2\Ai AC> & C:\Python314\python.exe "c:/Btech/3 - 2/Ai AC/Assignment - 3.5.py"
Enter first number: 0
Enter second number: -5
The GCD of 0 and -5 is -5
Enter a decimal number: 10
The binary equivalent of decimal 10 is 1010
○ PS C:\Btech\3 - 2\Ai AC>
```

## Edge Case Analysis :

```
print(decimal_to_binary(0))      # Output: 0

print(decimal_to_binary(-5))     # Output: b101
(incorrect)
```

## **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 Used (Few-Shot Prompt):**

Write a Python function to check Harshad number.

### **Examples:**

18 → Harshad

21 → Harshad

19 → Not Harshad

A screenshot of a code editor window titled "Assignment - 3.5.py". The code is as follows:

```
67
68     # Write a Python function to check Harshad number.
69     # Examples:
70     # 18 → Harshad
71     # 21 → Harshad
72     # 19 → Not Harshad
```

## Copilot Suggested Code :

A screenshot of a code editor window titled "Assignment - 3.5.py". The code has been extended with Copilot's suggestions:

```
67
68     # Write a Python function to check Harshad number.
69     # Examples:
70     # 18 → Harshad
71     # 21 → Harshad
72     # 19 → Not Harshad
73
74     def is_harshad_number(number):
75         digit_sum = sum(int(digit) for digit in str(number))
76         return number % digit_sum == 0
77     # Example usage:
78     num = int(input("Enter a number: "))
79     if is_harshad_number(num):
80         print(f"{num} is a Harshad number.")
81     else:
82         print(f"{num} is not a Harshad number.")
```

## **Input / Output test cases screenshot :**

```
● PS C:\Btech\3 - 2\Ai AC> & C:\Python314\python.exe "c:/Btech/3 - 2/Ai AC/Assignment - 3.5.py"
  Enter first number: 18
  Enter second number: 19
  The GCD of 18 and 19 is 1
  Enter a number: 21
  21 is a Harshad number.

○ PS C:\Btech\3 - 2\Ai AC> █
```

## **Testing :**

```
print(is_harshad(18))  # True
print(is_harshad(21))  # True
print(is_harshad(19))  # False
```

## **Boundary Conditions :**

- Fails for  $n = 0$  (division by zero)

## **Robust Version :**

```
def is_harshad(n):
    if n <= 0:
        return False

    digit_sum = sum(int(d) for d in str(n))

    return n % digit_sum == 0
```