

# Lab Assignment 3.5

(Thursday)

Name : Anoop Goud

Hallticket : 2303A51085

Batch : 02

## Lab Experiment: Prompt Engineering – Improving Prompts and Context Management

### Experiment – Prompt Engineering Techniques

#### Task Description

Design and refine prompts using different prompting strategies to generate Python programs for basic computational problems

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

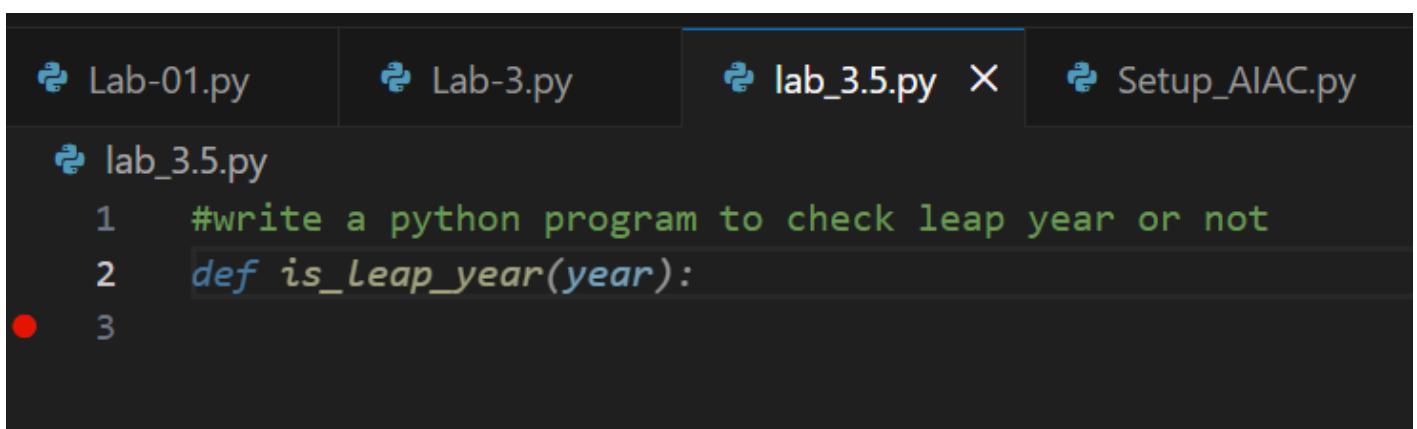
Write a zero-shot prompt to generate a Python function that checks whether a given year is a leap year.

#### Task:

- Record the AI-generated code.
- Test the code with multiple inputs.
- Identify any logical errors or missing edge-case handling.

#### Prompt:

#write a python program to check leap year or not



```
Lab-01.py Lab-3.py lab_3.5.py X Setup_AIAC.py
lab_3.5.py
1 #write a python program to check leap year or not
2 def is_leap_year(year):
3
```

#### Code:

The screenshot shows a code editor with multiple tabs at the top: Lab-01.py, Lab-3.py, lab\_3.5.py (which is the active tab), Setup\_AIAC.py, and fri.p. The code in the lab\_3.5.py tab is as follows:

```
1  #write a python program to check leap year or not and accept on
2  def is_leap_year(year):
3      if year < 0:
4          return False
5      if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
6          return True
7      return False
8  print(is_leap_year(2020))  # Example usage
9  print(is_leap_year(1900))  # Example usage
10 print(is_leap_year(2000)) # Example usage
```

Below the code editor, there are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is underlined), and PORTS. The TERMINAL tab shows the following output:

```
/AIAC/lab_3.5.py> OneDrive\Desktop\AIAC>
True
False
True
PS C:\Users\anoop\OneDrive\Desktop\AIAC>
```

## 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:

Input: 12, 18 → Output: 6

```
Input: 12, 18 → Output: 6
def gcd(a, b):
    while b:
        a, b = b, a % b
    return a
```

```
12  ...
13  Input: 12, 18 → Output: 6
14  ...
15  def gcd(a, b):
16  |    while b:
17  |        a, b = b, a % b
18  |    return a
19  print(gcd(12, 18)) # 6
20  print(gcd(56, 98)) # 14
● 21  print(gcd(101, 10)) # 1
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

6

14

1

PS C:\Users\anoop\OneDrive\Desktop\AIAC>

### Comparison with Zero-Shot Solution

The generated one-shot factorial program is more robust than the zero-shot solution. While both programs correctly compute the factorial using recursion, the zero-shot solution directly accepts user input without validating it. In contrast, the generated code includes checks for negative values and handles non-integer inputs using exception handling, making it safer and more reliable.

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:

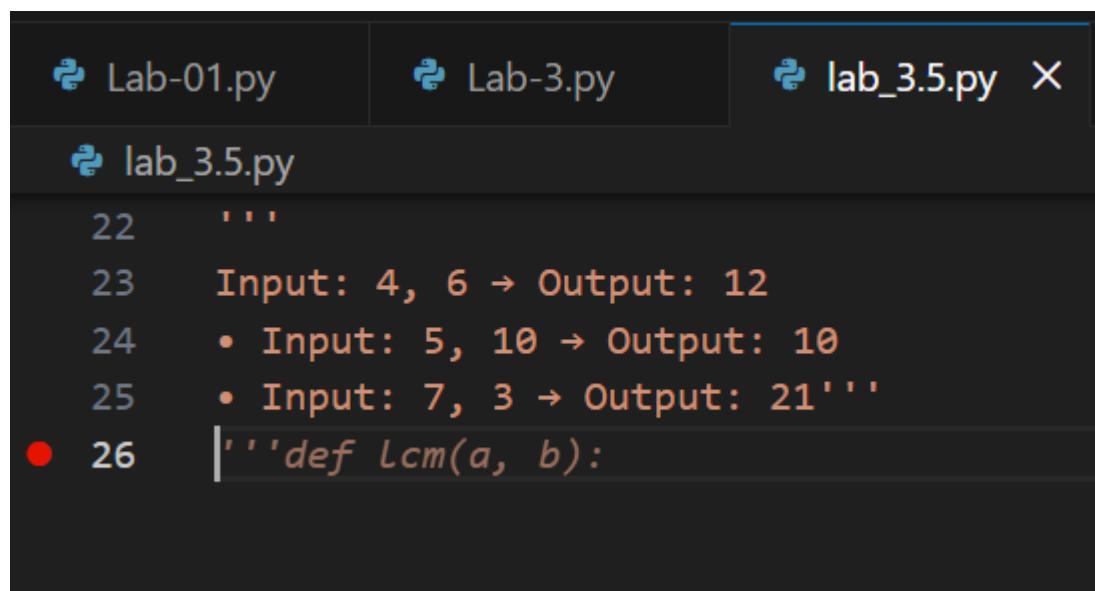
- Input: 4, 6 → Output: 12
- Input: 5, 10 → Output: 10
- Input: 7, 3 → Output: 21

Task:

- Examine how examples guide formula selection.

- **Test edge cases.**

Prompt:



The screenshot shows a code editor with three tabs at the top: "Lab-01.py", "Lab-3.py", and "lab\_3.5.py". The "lab\_3.5.py" tab is active. The code in the editor is as follows:

```
22     ...
23     Input: 4, 6 → Output: 12
24     • Input: 5, 10 → Output: 10
25     • Input: 7, 3 → Output: 21'''
● 26     '''def lcm(a, b):
```

A red dot is placed next to the line number 26, indicating it is the current line of interest or the next line to be edited.

Code:

```
Lab-01.py Lab-3.py lab_3.5.py X Setup_AIAC

lab_3.5.py > ...

22    ...
23    Input: 4, 6 → Output: 12
24    • Input: 5, 10 → Output: 10
25    • Input: 7, 3 → Output: 21
26    def lcm(a, b):
27        def gcd(x, y):
28            while y:
29                x, y = y, x % y
30            return x
31        return abs(a * b) // gcd(a, b)
32    print(lcm(4, 6)) # 12
33    print(lcm(5, 10)) # 10
● 34    print(lcm(7, 3)) # 21

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\anoop\OneDrive\Desktop\AIAC> & C:/Users/anoop/AIAC/lab_3.5.py
/AIAC/lab_3.5.py
12
10
21
PS C:\Users\anoop\OneDrive\Desktop\AIAC>
```

#### 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:

```
35
36     #write a python function that converts decimal to binary number
37     def decimal_to_binary(n):
        ...
```

**Code:**

The screenshot shows a code editor interface with a tab bar at the top containing files: -3.py, lab\_3.5.py (active), Lab\_4.1.py, Setup\_AIAC.py, fri.py, JAN-12.py, and print. Below the editor is a terminal window showing the execution of the lab\_3.5.py script.

```
-3.py          lab_3.5.py X  Lab_4.1.py  Setup_AIAC.py  fri.py  JAN-12.py  print

lab_3.5.py > ...
36     #write a python function that converts decimal to binary number
37     def decimal_to_binary(n):
38         if n == 0:
39             return "0"
40         binary = ""
41         while n > 0:
42             binary = str(n % 2) + binary
43             n = n // 2
44         return binary
45     try:
46         number = int(input("Enter a positive integer to convert to binary: "))
47         if number < 0:
48             print("Invalid input. Please enter a positive integer.")
49         else:
50             binary_representation = decimal_to_binary(number)
51             print(f"The binary representation of {number} is {binary_representation}.")
52     except ValueError:
53         print("Invalid input. Please enter a valid integer.")

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

/AIAC/lab_3.5.py
Enter a positive integer to convert to binary: 10
The binary representation of 10 is 1010.
PS C:\Users\anoop\OneDrive\Desktop\AIAC> & C:/Users/anoop/AppData/Local/Programs/Python/Python314/p
/AIAC/lab_3.5.py
Enter a positive integer to convert to binary: -10
Invalid input. Please enter a positive integer.
PS C:\Users\anoop\OneDrive\Desktop\AIAC> []
```

### 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: write a python code to convert decimal number to binary number

input=10

output=1010

A screenshot of a code editor showing a partially completed Python script. The script starts with a multi-line comment and then defines a variable `input` with the value 10. The code is intended to print the binary representation of the input, which is shown as 1010. The code editor interface includes a status bar at the bottom with file navigation icons.

```
50  """
51  #write a python function that converts decimal to binary number
52  """
53
54  """
55
56  input=10
57  output=1010
```

Code:

The screenshot shows a code editor interface with multiple tabs at the top: -3.py, lab\_3.5.py (selected), Lab\_4.1.py, Setup\_AIAC.py, fri.py, and JAN-12.py. The main area displays the following Python code:

```
33     print("Please enter a valid integer: ")
54     ...
55
56     #write a python code to convert decimal number to binary number
57     def decimal_to_binary(n):
58         if n == 0:
59             return "0"
60         binary = ""
61         while n > 0:
62             binary = str(n % 2) + binary
63             n = n // 2
64         return binary
65     number = int(input("Enter a positive integer to convert to binary: "))
66     binary_representation = decimal_to_binary(number)
67     print(f"The binary representation of {number} is {binary_representation}.")
68
```

Below the code editor, there is a navigation bar with tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (underlined), and PORTS.

The terminal output window shows the execution of the script and its interaction with the user:

```
/AIAC/lab_3.5.py
Enter a positive integer to convert to binary: 10
The binary representation of 10 is 1010.
PS C:\Users\anoop\OneDrive\Desktop\AIAC> & C:/Users/anoop/AppData/Local/Programs/Python
/AIAC/lab_3.5.py
Enter a positive integer to convert to binary: -10
The binary representation of -10 is .
PS C:\Users\anoop\OneDrive\Desktop\AIAC>
```

Compared to a zero-shot output, this implementation is clearer and more readable because it explicitly shows each step of the decimal-to-binary conversion process. However, while the function correctly handles the case when the input is zero by returning "0", it does not handle negative numbers. Without validation, negative inputs would result in incorrect behavior, so additional checks are needed to restrict the input to non-negative integers.

#### 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:

input=18

output=harshad number

input=21

output=harshad number

input=19

output=not a harshad number

write a python code to check wheather a given number is harshad number or not

Prompt:

```
#write a python code to check wheather a given number is harshad number or not
def is_harshad_number(n):
    digit_sum = sum(int(digit) for digit in str(n))
    return n % digit_sum == 0
```

Code:

```
-3.py X lab_3.5.py X Lab_4.1.py Setup_AIAC.py fri.py JAN-12.py prime
lab_3.5.py > ...
69
70 #write a python code to check wheather a given number is harshad number or not
71 def is_harshad(num):
72     digit_sum = sum(int(digit) for digit in str(num))
73     return num % digit_sum == 0
74 try:
75     number = int(input("Enter a positive integer to check if it's a Harshad number: "))
76     if number < 0:
77         print("Invalid input. Please enter a positive integer.")
78     else:
79         if is_harshad(number):
80             print(f"{number} is a Harshad number.")
81         else:
82             print(f"{number} is not a Harshad number.")
83 except ValueError:
84     print("Invalid input. Please enter a valid integer.")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Enter a positive integer to check if it's a Harshad number: 18
18 is a Harshad number.
PS C:\Users\anoop\OneDrive\Desktop\AIAC> & C:/Users/anoop/AppData/Local/Programs/Python/Python314/pyt
/AIAC/lab_3.5.py
Enter a positive integer to check if it's a Harshad number: 21
21 is a Harshad number.
PS C:\Users\anoop\OneDrive\Desktop\AIAC> & C:/Users/anoop/AppData/Local/Programs/Python/Python314/pyt
/AIAC/lab_3.5.py
Enter a positive integer to check if it's a Harshad number: 19
19 is not a Harshad number.
PS C:\Users\anoop\OneDrive\Desktop\AIAC>
```

The code correctly identifies Harshad numbers for valid inputs such as 18 and 21, and correctly reports non-Harshad numbers like 19. It also handles invalid inputs such as negative numbers and non-integer values using input validation and exception handling. However, the logic does not explicitly handle the edge case where the input is 0, which would cause a division by zero error. Adding a specific check to reject 0 would complete the validation logic and make the program more robust.

