

AI ASSISTED CODING

ASSIGNMENT – 3.5

G.Sriram

Roll No : (2303A51191)

BATCH-03

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.

Week2 - Task:

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

The screenshot shows a code editor interface with several tabs open. The active tab contains Python code for checking if a year is a leap year. The code includes a docstring explaining the rules: a year is a leap year if it is divisible by 4, except for end-of-century years, which must be divisible by 400. It defines a function `is_leap_year` that takes an integer `year` and returns a boolean value. The code then tests this function with years 1900 and 2000, printing the results. The output panel shows the execution results for both years.

```
# generate a python function that checks whether a given year is leap year or not
def is_leap_year(year):
    """
    check if a given year is a leap year.

    A year is a leap year if it is divisible by 4,
    except for end-of-century years, which must be divisible by 400.

    Args:
        year (int): The year to check.
    Returns:
        bool: True if the year is a leap year, False otherwise.
    """
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return True
    else:
        return False
# Example usage:
year = 1900
if is_leap_year(year):
    print(f"{year} is a leap year.")
else:
    print(f"{year} is not a leap year.")

2024 is a leap year.

[Done] exited with code=0 in 0.449 seconds

[Running] python -u "C:\Users\naksh\AppData\Local\Temp\tempCodeRunnerFile.python"
2000 is a leap year.

[Done] exited with code=0 in 0.387 seconds

[Running] python -u "C:\Users\naksh\AppData\Local\Temp\tempCodeRunnerFile.python"
1900 is not a leap year.

[Done] exited with code=0 in 0.41 seconds
```

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.

The screenshot shows a code editor interface with a dark theme. In the top bar, there are tabs for 'File', 'Edit', 'Selection', 'View', 'Go', 'Run', 'Terminal', 'Help', and 'Search'. A 'CHAT' tab is also present. The main area contains the following Python code:

```
1  """
2  num = 12,18
3  gcd = 6
4  """
5  def compute_gcd(a, b):
6      while b:
7          a, b = b, a % b
8      return a
9  if __name__ == "__main__":
10     a = 12
11     b = 18
12     print("num =", (a, b))
13     print("gcd =", compute_gcd(a, b))
```

Below the code, the 'OUTPUT' tab is selected, showing the terminal output of the script running:

```
[Running] python -u "C:\Users\naksh\AppData\Local\Temp\tempCodeRunnerFile.py"
num = (12, 18)
gcd = 6

[Done] exited with code=0 in 0.427 seconds
```

On the right side of the interface, there is a sidebar with the heading 'Ask about your code' and a message: 'AI responses may be inaccurate. Generate Agent Instructions to onboard AI onto your codebase.'

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.

A screenshot of a code editor window. The menu bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and CHAT. The main area contains the following Python code:

```
1 ...
2 num = 4,6
3 lcm = 12
4 num = 5,6
5 lcm = 30
6 num = 7,3
7 lcm = 21
8 ...
9 def lcm (a, b):
10     if a > b:
11         greater = a
12     else:
13         greater = b
14
15     while True:
16         if greater % a == 0 and greater % b == 0:
17             lcm = greater
18             break
19         greater += 1
20
21     return lcm
```

The status bar at the bottom shows [Running] python -u "C:\Users\naksh\AppData\Local\Temp\tempCodeRunnerFile.py". The output pane displays "LCM of 4 and 6 is 12".

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.

A screenshot of a code editor window. The menu bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and CHAT. The main area contains the following Python code:

```
1 #generate a python function that converts a binary number to decimal
2 def binary_to_decimal(binary_str):
3     decimal_value = 0
4     binary_str = binary_str[::-1] # Reverse the string to process from least significant bit
5     for index, digit in enumerate(binary_str):
6         if digit == '1':
7             decimal_value += 2 ** index
8
9 # Example usage:
10 binary_number = "1101"
11 decimal_number = binary_to_decimal(binary_number)
12 print(f"The decimal value of binary {binary_number} is {decimal_number}")
```

The status bar at the bottom shows [Running] python -u "C:\Users\naksh\AppData\Local\Temp\tempCodeRunnerFile.py". The output pane displays "The decimal value of binary 1101 is 13".

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.

The screenshot shows a code editor interface with a dark theme. In the center, there is a code editor window containing the following Python code:

```
# generate a python function that converts a decimal number to binary
num = 10
binary_number = 1010
def decimal_to_binary(n):
    if n > 1:
        decimal_to_binary(n // 2)
        print(n % 2, end='')
num = 10
decimal_to_binary(num)
```

Below the code editor, there is a terminal window showing the output of running the script:

```
[Running] python -u "C:\Users\naksh\AppData\Local\Temp\tempCodeRunnerFile.py"
1010
[Done] exited with code=0 in 0.446 seconds
```

On the right side of the interface, there is a sidebar with a message bubble icon and the text "Ask about your code". Below it, smaller text reads "AI responses may be inaccurate." and "Generate Agent Instructions to onboard AI onto your codebase."

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**

The screenshot shows a code editor interface with a dark theme. In the top-left corner, there's a search bar with the placeholder "Search". The top menu bar includes "File", "Edit", "Selection", "View", "Go", "Run", "Terminal", and "Help". On the right side of the menu bar, there are several tabs labeled "Untitled-4", "Untitled-5", "Untitled-6", "#generate a python fuction that converts", "Untitled-7", "Untitled-1", "Untitled-2", and "CHAT". Below the menu bar, there's a toolbar with icons for file operations like "New", "Open", "Save", etc. A large text area contains the following Python code:

```
4 num = 21
5 print num is Harzard number
6 num = 19
7 print num is not Harzard number
8 """
9 def is_harshad_number(num):
10     digit_sum = sum(int(digit) for digit in str(num))
11     return num % digit_sum == 0
12 if __name__ == "__main__":
13     test_numbers = [18, 21, 19]
14     for num in test_numbers:
15         if is_harshad_number(num):
16             print(f"{num} is Harshad number")
17         else:
18             print(f"{num} is not Harshad number")
19 
```

Below the code editor, there's a "PROBLEMS" tab, an "OUTPUT" tab (which shows the execution log), a "DEBUG CONSOLE", "TERMINAL", and "PORTS" tab. The "OUTPUT" tab contains the following log:

```
[Running] python -u "C:\Users\naksh\AppData\Local\Temp\tempcodeRunnerFile.python"
18 is Harshad number
21 is Harshad number
19 is not Harshad number
```

The "OUTPUT" tab also indicates "[Done] exited with code=0 in 0.416 seconds". To the right of the code editor, there's a "CHAT" section with a "Ask about your code" button and a note that AI responses may be inaccurate. At the bottom right, there's a "Untitled-2" panel with the text "Explore and understand your code".