

AI ASSISTED CODING ASSIGNMENT – 3.5

P.Aneesh Reddy

Roll No : (2303A51120)

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 multiple tabs. The active tab contains the following Python code:

```
# 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.
```

Below the code editor, the terminal window shows the output of running the script with year 2024. The AI sidebar on the right is visible with the message "Ask about your code".

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 multiple tabs. The active tab contains the following Python code:

```
# generate a python function that checks if two numbers are coprime
def compute_gcd(a, b):
    while b:
        a, b = b, a % b
    return a
if __name__ == "__main__":
    a = 12
    b = 18
    print("num =", (a, b))
    print("gcd =", compute_gcd(a, b))
```

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

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

The terminal also indicates the process exited with code=0 in 0.427 seconds.

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.

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and a search bar. There are seven tabs open: Untitled-4, Untitled-5, Untitled-6 (the active tab), Untitled-7, Untitled-1, Untitled-2, and Untitled-3. A ChatGPT integration window titled "Ask about your code" is visible on the right, featuring a message input field and a "Send" button. Below the tabs, the code editor displays a Python script for calculating the Least Common Multiple (LCM) of two numbers. The script uses a while loop with the Euclidean algorithm. The bottom status bar shows tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS, along with a code search bar and a "Code" dropdown. The terminal pane shows the command "python -u "C:/Users/naksh/AppData/Local/Temp/tempCodeRunnerfile.py"" was run, resulting in "LCM of 4 and 6 is 12". The status bar also indicates the code exited with a status of 0 in 0.473 seconds.

```
# generate a python function that checks Untitled-4 • Untitled-5 • Untitled-6 • #generate a python fuction that converts Untitled-7 • Untitled-1 • Untitled-2 • Untitled-3 •
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
```

PROBLEMS ② OUTPUT DEBUG CONSOLE TERMINAL PORTS

[Running] python -u "C:/Users/naksh/AppData/Local/Temp/tempCodeRunnerfile.py"
LCM of 4 and 6 is 12

[Done] exited with code=0 in 0.473 seconds

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.

The screenshot shows a code editor with several tabs open. The active tab contains Python code to convert binary strings to decimal values. The code uses a for loop with enumerate to iterate through the reversed binary string, calculating the decimal value by summing $2^{\text{index}} \times \text{digit}$ for each bit. It includes a usage example for the number "1101".

```
# generate a python function that converts a binary number to decimal
def binary_to_decimal(binary_str):
    decimal_value = 0
    binary_str = binary_str[::-1] # Reverse the string to process from least significant bit
    for index, digit in enumerate(binary_str):
        if digit == '1':
            decimal_value += 2 ** index
    return decimal_value

# Example usage:
binary_number = "1101"
decimal_number = binary_to_decimal(binary_number)
print(f'the decimal value of binary {binary_number} is {decimal_number}')

```

Below the code editor is a terminal window showing the execution of the script. The command `python -u "C:/Users/naksh/AppData/Local/Temp/tempCodeRunnerFile.pytton"` is run, followed by the output: "The decimal value of binary 1101 is 13".

At the bottom right, there is an AI interface with a message bubble icon and the text "Ask about your code".

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 top-left, there's a navigation bar with 'File', 'Edit', 'Selection', 'View', 'Go', 'Run', 'Terminal', 'Help'. Below it is a search bar. On the right side, there's a 'CHAT' section with a message icon and a placeholder 'Ask about your code'. A note below says 'AI responses may be inaccurate.' and 'Generate Agent Instructions to onboard AI onto your codebase.' At the bottom, there are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', and 'PORTS'. The 'OUTPUT' tab is active, showing the command 'python -u "c:\users\naksh\AppData\Local\Temp\tempCodeRunnerFile.py"' followed by the output '1010' and '[Done] exited with code=0 in 0.446 seconds'. The main code area contains the following Python code:

```
# generate a python function that converts a decimal number to binary.
...
1 num = 10
2 binary_number = 1010
3
4 def decimal_to_binary(n):
5     if n > 1:
6         decimal_to_binary(n // 2)
7         print(n % 2, end='')
8     num = 10
9 decimal_to_binary(num)
10
11
```

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. At the top, there's a menu bar with File, Edit, Selection, View, Go, Run, Terminal, Help, and a search bar. Below the menu is a tab bar with several tabs: Untitled-4, Untitled-5, Untitled-6, Untitled-7, Untitled-1, Untitled-2, and CHAT. The Untitled-2 tab is currently active.

The main area contains the following Python code:

```
4 num = 21
5 print(num is Harshad number)
6 num = 19
7 print(num is not Harshad number)
8 ...
9 def is_harshad(number):
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):
16             print(f"{num} is Harshad number")
17         else:
18             print(f"{num} is not Harshad number")
19 
```

Below the code editor is a "PROBLEMS" tab, an "OUTPUT" tab showing the execution results, a "DEBUG CONSOLE" tab, a "TERMINAL" tab, and a "PORTS" tab. The "OUTPUT" tab displays:

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

[Done] exited with code=0 in 0.416 seconds
```

To the right of the code editor, there's a "CHAT" section with a message bubble icon and the text "Ask about your code". Below it, smaller text says "AI responses may be inaccurate." and "Generate Agent Instructions to onboard AI onto your codebase." A small "Untitled-2" tab is also visible in the bottom right corner.