

AI ASSISTED CODING ASSIGNMENT – 3.5

T.Sai Thrishool

Roll No : 2303A51127

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 a dark theme. In the center, there is a code editor window displaying the following Python code:

```
1 # generate a python function that checks whether a given year is leap year or not
2 def is_leap_year(year):
3     """Check if a given year is a leap year.
4
5     A year is a leap year if it is divisible by 4,
6     except for end-of-century years, which must be divisible by 400.
7
8     Args:
9         year (int): The year to check.
10    Returns:
11        bool: True if the year is a leap year, False otherwise.
12
13    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
14        return True
15    else:
16        return False
17 # Example usage:
18 year = 1900
19 if is_leap_year(year):
20     print(f"{year} is a leap year.")
21 else:
22     print(f"{year} is not a leap year.")
```

Below the code editor, the output window shows the results of running the code with different years:

```
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
```

On the right side of the interface, there is a sidebar titled "CHAT" with a message bubble icon and the text "Ask about your code". Below it, a note says "All responses may be inaccurate." and "Generate Agent Instructions to onboard AI onto your codebase." At the bottom right, there is a small window titled "# generate a python function that checks" with the text "Explore and understand 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:

```
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))
14 
```

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

```
[Running] python -u "c:\users\naksh\appdata\local\temp\tempCodeRunnerFile.python"
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 button labeled "Ask". Below it, there is a note: "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 the Visual Studio Code interface. The code editor shows a Python file named 'Untitled-6' with the following content:

```
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 terminal tab shows the output of running the script:

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

The status bar at the bottom indicates the code ran 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.

A screenshot of the Visual Studio Code interface. The code editor shows a Python file named 'Untitled-7' with the following content:

```
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     return decimal_value
9
10 # Example usage:
11 binary_number = "1101"
12 decimal_number = binary_to_decimal(binary_number)
13 print(f"the decimal value of binary {binary_number} is {decimal_number}")
```

The terminal tab shows the output of running the script:

```
[Running] python -u "C:/Users/naksh/AppData/Local/Temp/tempCodeRunnerFile.py"
The decimal value of binary 1101 is 13

[Done] exited with code=0 in 0.383 seconds
```

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

```
1  """
2  num = 10
3  binary_number = 1010
4  """
5  def decimal_to_binary(n):
6      if n > 1:
7          decimal_to_binary(n // 2)
8          print(n % 2, end=' ')
9  num = 10
10 decimal_to_binary(num)
11 """
```

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

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

The terminal window has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The OUTPUT tab is selected, displaying the output of the script.

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

A screenshot of the Visual Studio Code (VS Code) interface. The main editor window contains a Python script named `Untitled-2.py`. The code defines a function `is_harshad_number` that checks if a given number is a Harshad number (a number that is divisible by the sum of its digits). It includes a test block with three numbers: 18, 21, and 19.

```
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(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")
```

The output panel shows the execution results:

```
[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 status bar at the bottom indicates the script exited with `code=0` in 0.416 seconds.

In the top right corner, there is an AI-powered code review feature from ChatGPT. It includes a button to "Ask about your code" and a note that AI responses may be inaccurate. A tooltip suggests generating agent instructions to onboard AI into the codebase.