

# AI ASSISTED CODING ASSIGNMENT

## – 3.5

Nishth Afroz

Roll No : (2303A51212)

BATCH-04

### 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 window with the following Python code:

```
# generate a python function that checks if a given year is a 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 = 2000
if is_leap_year(year):
    print(f"{year} is a leap year.")
else:
    print(f"{year} is not a leap year.")

[User] exited with status 0 in 0.449 seconds

[Running] python -u "C:\Users\Nishth\appdata\local\temp\Temp\LeapYearCheck11a.py"
2000 is a leap year.

[User] exited with status 0 in 0.447 seconds

[Running] python -u "C:\Users\Nishth\appdata\local\temp\Temp\LeapYearCheck11a.py"
2000 is not a leap year.

[User] exited with status 0 in 0.441 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 with a Python script named `gcd.py`. The code implements the Euclidean algorithm to find the GCD of two numbers, `a` and `b`. It includes a docstring and a check function. The code is run in a terminal window, showing the output of running the script with inputs 12 and 18, resulting in a GCD of 6. A tooltip "Ask about your code" is visible on the right.

```
1 #!/usr/bin/env python3
2
3 def compute_gcd(a, b):
4     while b != 0:
5         a, b = b, a % b
6     return a
7
8 if __name__ == "__main__":
9     a = 12
10    b = 18
11    print("GCD of", a, "and", b, "is", compute_gcd(a, b))
```

```
[Running] gcd.py --> "C:\Users\Yash\Appliances\Udemy\Code\Python"
code = (12, 18)
gcd = 6

[Done] exited with status 0 in 0.422 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.

A screenshot of a code editor window titled "generate a python function that finds lcm". The code is as follows:

```
num = 45
lcm = 12
lcm += 6
lcm += 18
lcm += 36
lcm += 72
lcm += 144

def lcm(a, b):
    if a > b:
        greater = a
    else:
        greater = b

    while True:
        if greater % a == 0 and greater % b == 0:
            lcm = greater
            break
        greater += 1

    return lcm

print(lcm(4, 6))

```

The output window shows the command being run and the result:

```
[running] python -m "E:\Users\rajan\Downloads\Local\Temp\TemporaryFolder\file1.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.

A screenshot of a code editor window titled "generate a python function that converts binary number to decimal". The code is as follows:

```
# 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

# simple usage
binary_number = "1010"
decimal_number = binary_to_decimal(binary_number)
print(f"The decimal value of binary {binary_number} is {decimal_number}")


```

The output window shows the command being run and the result:

```
[running] python -m "E:\Users\rajan\Downloads\Local\Temp\TemporaryFolder\file1.py"
The decimal value of binary 1010 is 10

[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.

A screenshot of a Jupyter Notebook interface. The code cell contains the following Python code:num = 10
binary\_number = bin(num)

def decimal\_to\_binary(n):
 if n < 0:
 decimal\_to\_binary(-n)
 print('S', n, end='')
 else:
 num = n
 decimal\_to\_binary(n // 2)
 print(n % 2, end='')

decimal\_to\_binary(num)The output cell shows the result of running the code: "1010". Below the output, it says "Done! edited with codecell in 0.446 seconds".

**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. On the left is the code editor pane containing Python code. On the right is the terminal pane showing the execution and output of the code.

```
1 # Function that checks if number is Harshad number
2
3 def is_harshad_number(num):
4     sum = 0
5     if num < 10:
6         print(num, "is not Harshad number")
7         return False
8     else:
9         for digit in str(num):
10             digit_as_int = int(digit)
11             sum += digit_as_int
12
13     if sum == num:
14         print(str(num), "is Harshad number")
15     else:
16         print(str(num), "is not Harshad number")
17
18
19
20 test_numbers = [19, 21, 48]
21 for num in test_numbers:
22     if is_harshad_number(num):
23         print(str(num), "is Harshad number")
24     else:
25         print(str(num), "is not Harshad number")
```

TERMINAL

```
[running] python -m pygments.console TERMINAL -P0001
[running] python -m pygments.console
19 is Harshad number
21 is Harshad number
48 is not Harshad number
(Done) exited with status 0 in 0.000 seconds
```

Ask about your code

All responses may be inaccurate.

Generate Agent Instructions to continue AI refine your codebase.

# 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

Explore and understand your code