

ASSIGNMENT - 3.5

Name: A.Sai Varshith

Roll Number : 2303A51831

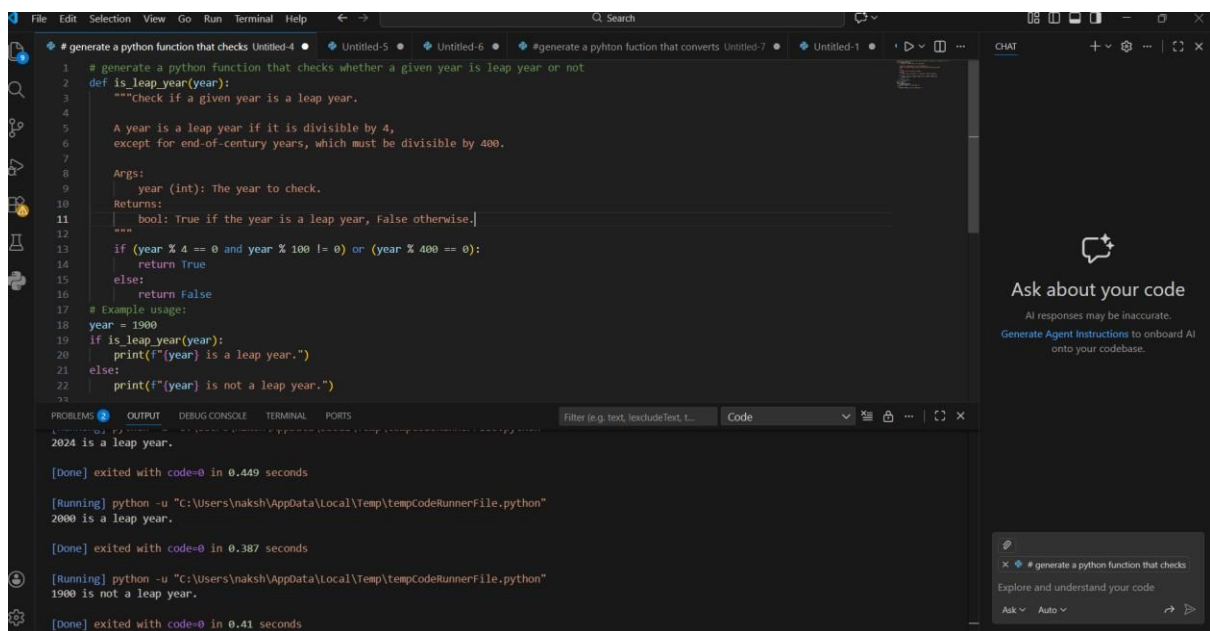
Batch - 04

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 with a Python function `is_leap_year` and its execution results in the terminal. The function checks if a year is a leap year based on the following logic: a year is a leap year if it is divisible by 4, except for end-of-century years, which must be divisible by 400. The function returns `True` if the year is a leap year, and `False` otherwise. The terminal output shows the function being tested with years 1900, 2000, and 2024. The results are: 1900 is not a leap year, 2000 is a leap year, and 2024 is a leap year.

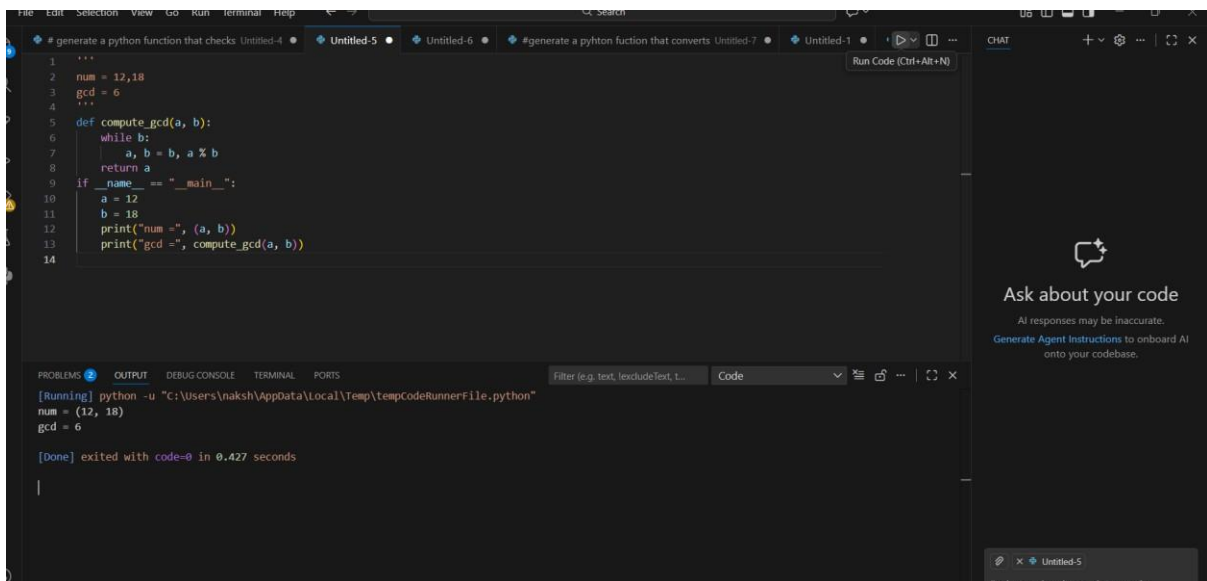
```
1 # generate a python function that checks whether a given year is leap year or not
2 def is_leap_year(year):
3     """Check if a given year is a leap year.
4
5     A year is a leap year if it is divisible by 4,
6     except for end-of-century years, which must be divisible by 400.
7
8     Args:
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
18 # Example usage:
19 year = 1900
20 if is_leap_year(year):
21     print(f"{year} is a leap year.")
22 else:
23     print(f"{year} is not a leap year.")
24
25
26 [Done] exited with code=0 in 0.449 seconds
27
28 [Running] python -u "C:\Users\naksh\AppData\Local\Temp\tempCodeRunnerFile.python"
29 2000 is a leap year.
30
31 [Done] exited with code=0 in 0.387 seconds
32
33 [Running] python -u "C:\Users\naksh\AppData\Local\Temp\tempCodeRunnerFile.python"
34 1900 is not a leap year.
35
36 [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 with a Python script and its execution output. The script defines a function `compute_gcd(a, b)` using a while loop to find the GCD of two numbers. It then calls this function with `a=12` and `b=18` and prints the result. The terminal output shows the execution of the script, confirming that the GCD of 12 and 18 is 6.

```
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
```

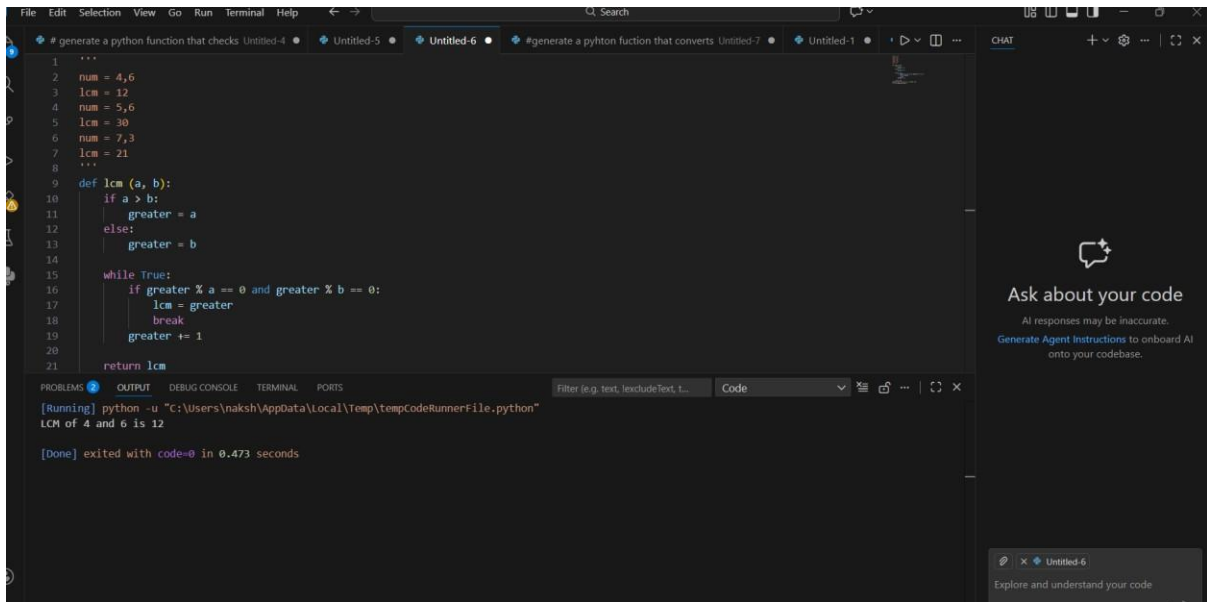
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
[Running] python -u "C:\Users\naksh\AppData\Local\Temp\tempCodeRunnerFile.py"
num = (12, 18)
gcd = 6
[Done] exited with code=0 in 0.427 seconds

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.



```
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
```

[Running] python -u "C:\Users\naksh\AppData\Local\Temp\tempCodeRunnerFile.python"

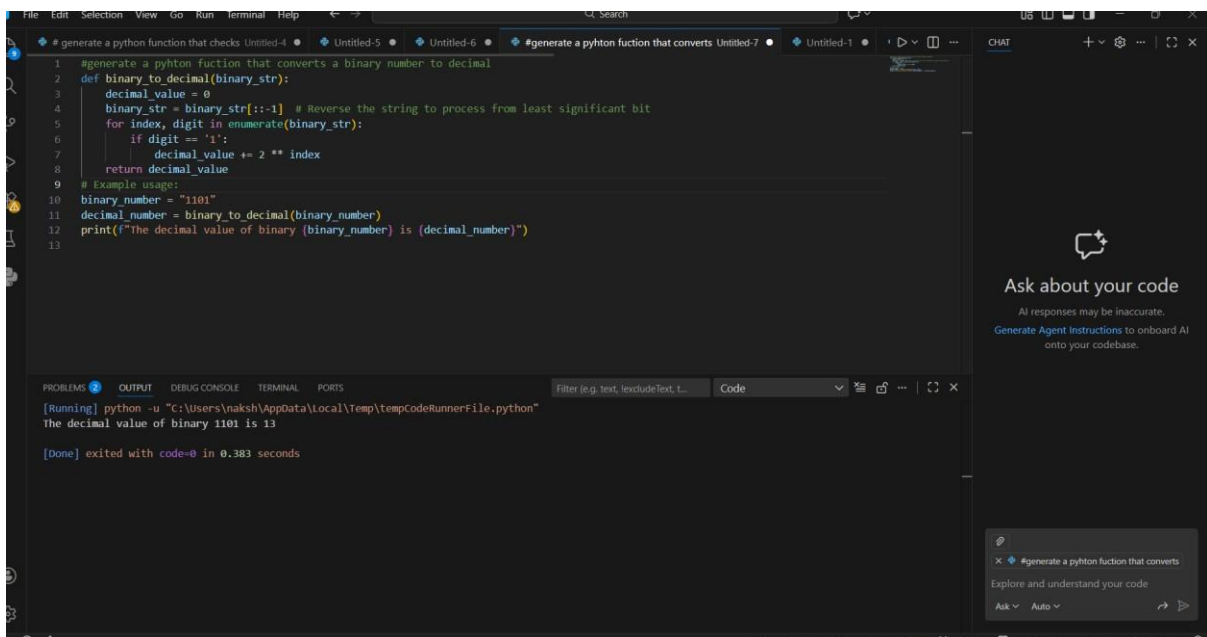
LCM of 4 and 6 is 12

[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.



```
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}")
```

[Running] python -u "C:\Users\naksh\AppData\Local\Temp\tempCodeRunnerFile.python"

The decimal value of binary 1101 is 13

[Done] exited with code=0 in 0.383 seconds

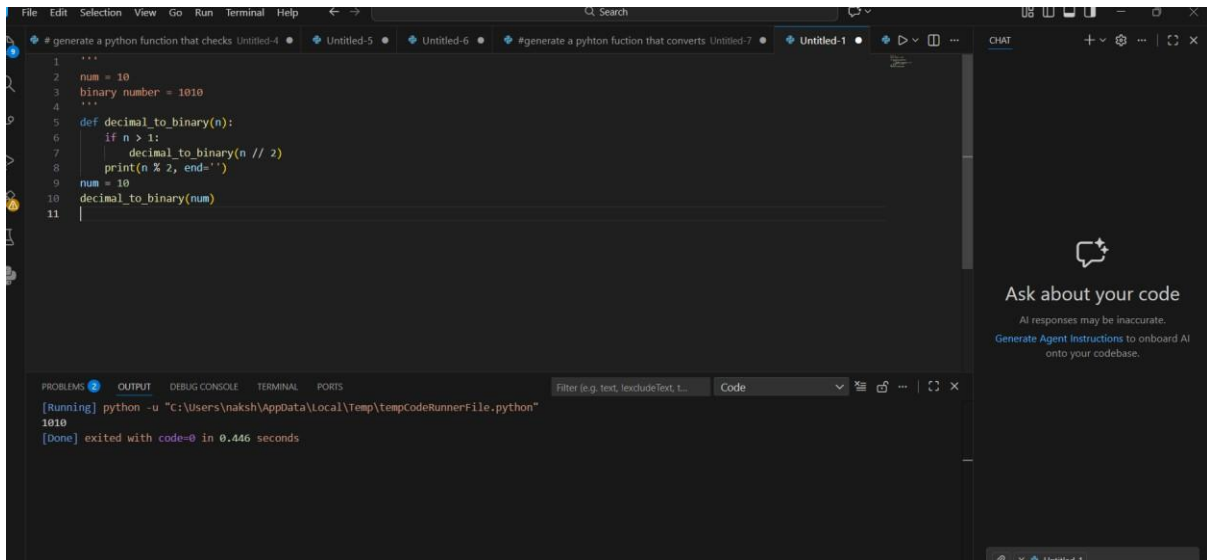
Question 5: One-Shot Prompting (Decimal to Binary Conversion)

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 with a Python script and its execution output. The script defines a recursive function `decimal_to_binary(n)` that prints the binary representation of a decimal number `n`. The function is called with `num = 10`. The output shows the binary representation `1010`.

```
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 |
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

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

