

# AI ASSISTED CODING ASSIGNMENT

## – 3.5

Nishth Afroz

Roll No : (2303A51212)

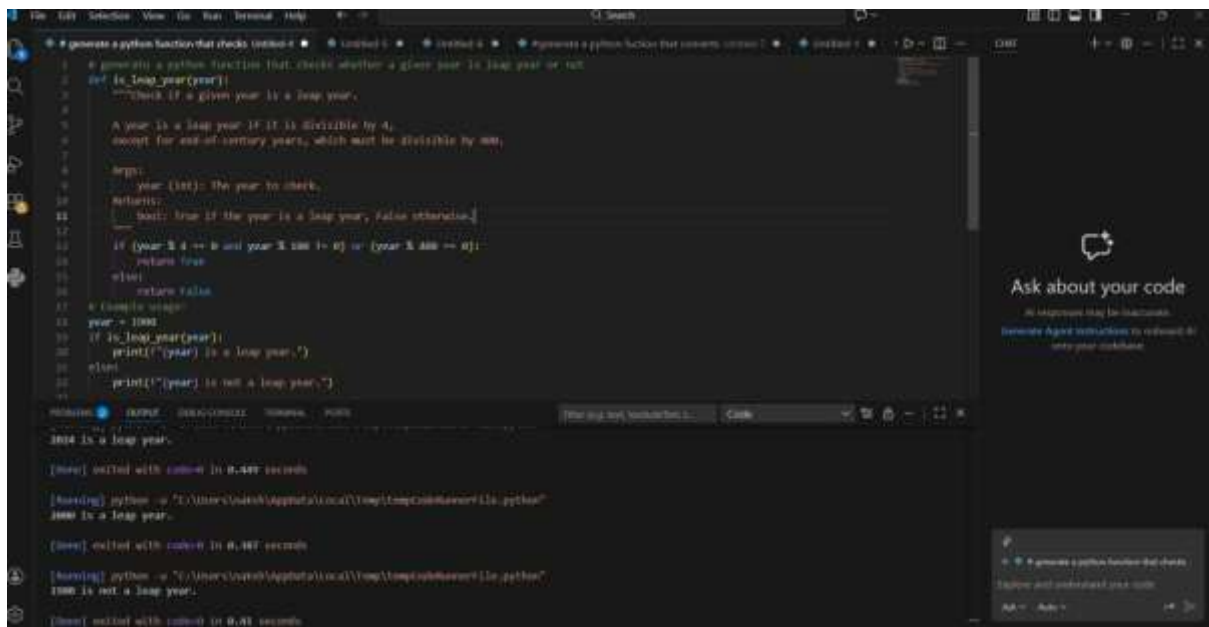
BATCH-04

### 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.



```
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 = 2024
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.")
```

2024 is a leap year.

[Note] exited with code=0 in 0.449 seconds

[Testing] python -u "C:\Users\Nishth\AppData\Local\Temp\TempCodeRunner\11a.py" 2024 is a leap year.

[Note] exited with code=0 in 0.167 seconds

[Testing] python -u "C:\Users\Nishth\AppData\Local\Temp\TempCodeRunner\11a.py" 1900 is not a leap year.

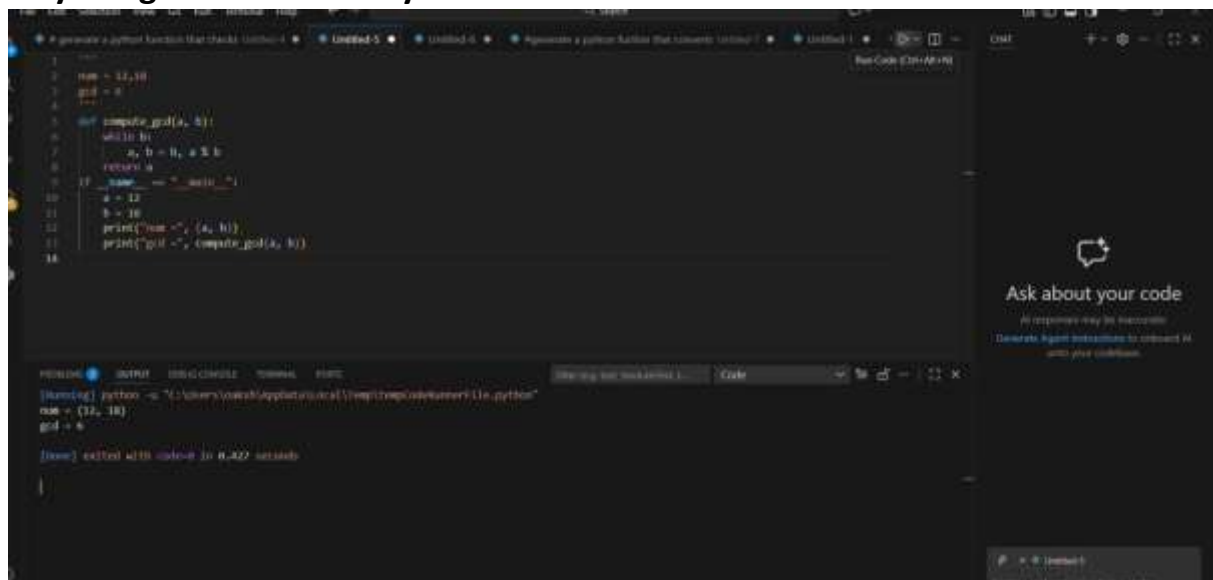
[Note] exited with code=0 in 0.161 seconds

**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 function `compute_gcd(a, b)` implemented using the Euclidean algorithm. The function uses a `while` loop to repeatedly replace `a` with `b` and `b` with `a % b` until `b` becomes 0. The final value of `a` is the GCD. The code is tested with `a = 12` and `b = 18`, resulting in a GCD of 6. The output console shows the execution of the code and the resulting GCD value.

```
1 a = generate a python function that checks whether a
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
10 if __name__ == "__main__":
11     a = 12
12     b = 18
13     print("num = ", (a, b))
14     print("gcd = ", compute_gcd(a, b))
15
```

Running: python -u "C:\Users\vaish\AppData\Local\Temp\codeinterpreter\11e.py" num = (12, 18) gcd = 6

[name] exited with output 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.

```
1 """
2
3 sum = 4,6
4 lim = 12
5 sum = 5,6
6 lim = 20
7 sum = 7,2
8 lim = 24
9
10 def lim(a, b):
11     if a > b:
12         greater = a
13     else:
14         greater = b
15
16     while True:
17         if greater < a == 0 and greater < b == 0:
18             lim = greater
19             break
20         greater += 1
21
22 return lim
```

[Running] python -u "C:\Users\user\AppData\Local\Temp\tempcodeeditor\lim.py" 409 of 4 and 5 is 12

[Done] exited with code=0 in 0.433 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
3 def binary_to_decimal(binary_str):
4     decimal_value = 0
5     binary_str = binary_str[::-1] # Reverse the string to process from least significant bit
6     for index, digit in enumerate(binary_str):
7         if digit == '1':
8             decimal_value += 2 ** index
9     return decimal_value
10
11 # Example usage:
12 binary_number = "1101"
13 decimal_number = binary_to_decimal(binary_number)
14 print(f"The decimal value of binary {binary_number} is {decimal_number}")
15
```

[Running] python -u "C:\Users\user\AppData\Local\Temp\tempcodeeditor\lim.py" 409 of 4 and 5 is 12

[Done] exited with code=0 in 0.383 seconds

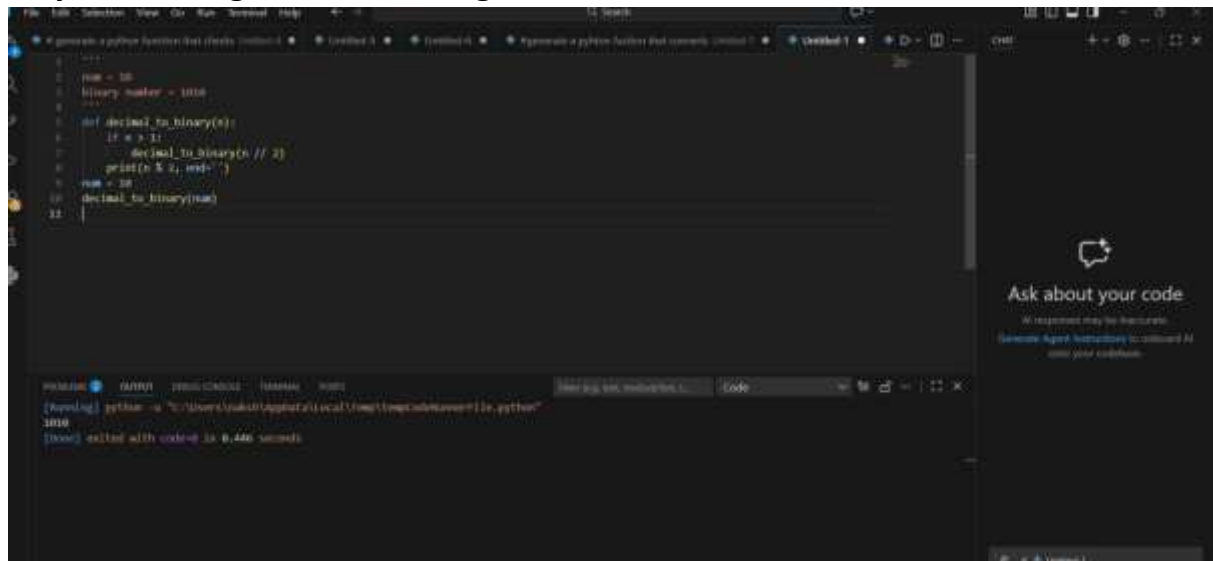
**Question 5: One-Shot Prompting (Decimal to Binary Conversion)**

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. The script defines a function `decimal_to_binary(n)` that uses a recursive approach to convert a decimal number to binary. The function calls itself with `n // 2` and prints the remainder `n % 2`. The script then calls the function with `10`. The output in the terminal shows the binary representation `1010`.

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

Output: 1010

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

