

AI ASSISTED CODING

ASSIGNMENT – 3.5

M. Vennela

Roll No : (2303A51112)

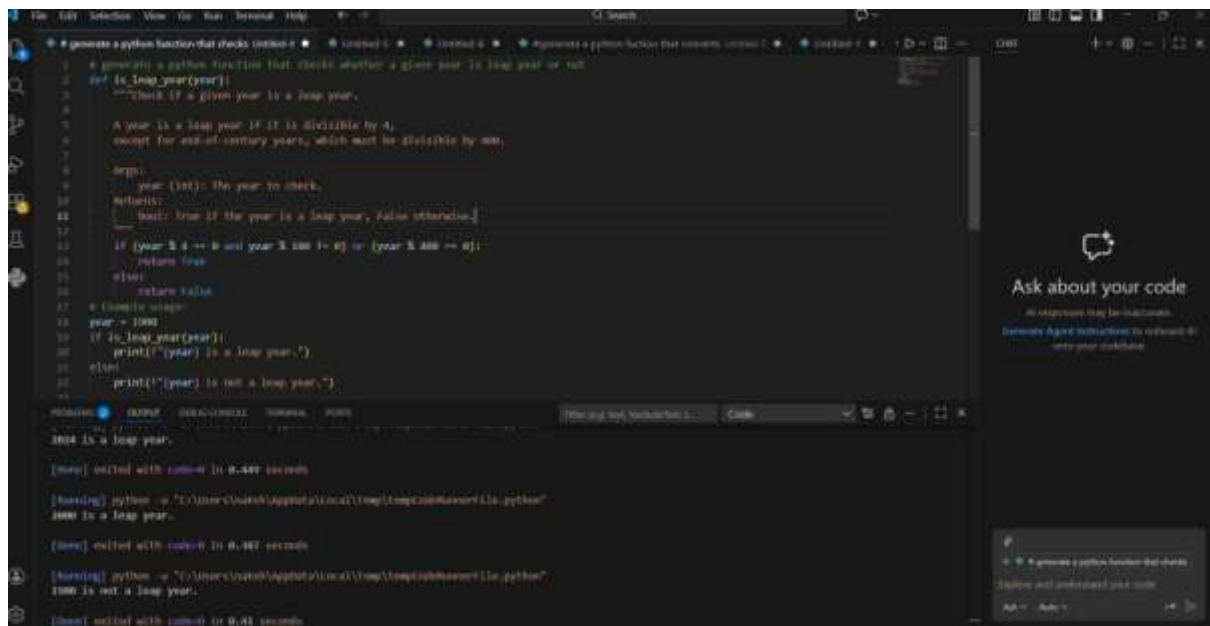
BATCH-03

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.



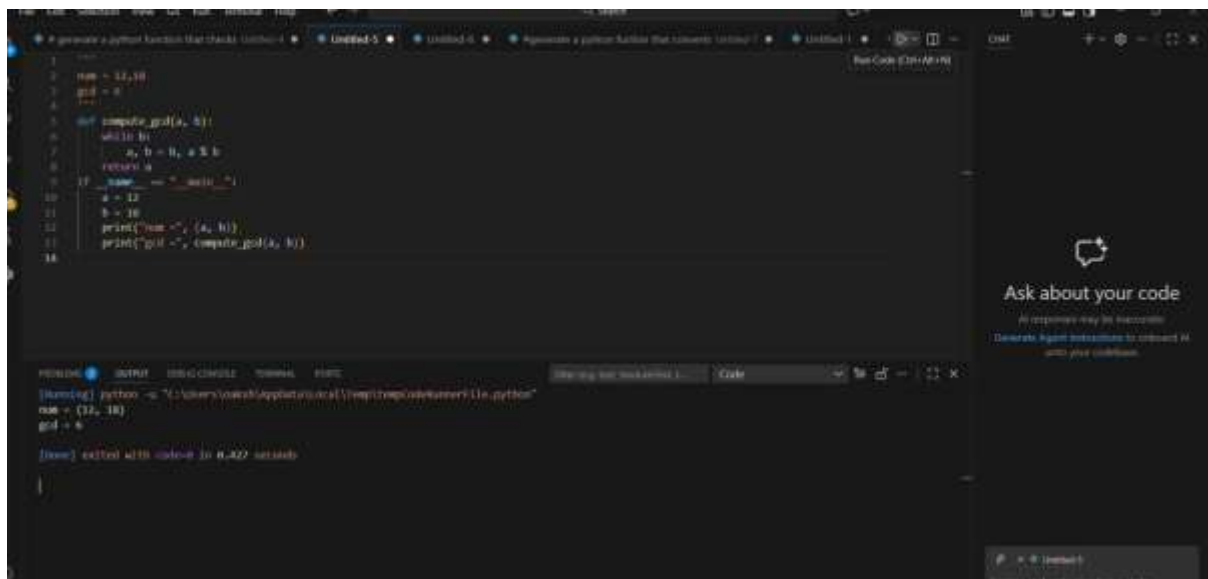
```
1 # generate a python function that checks whether a given year is leap year or not
2 def is_leap_year(year):
3     """Check if a given year is a leap year.
4
5     A year is a leap year if it is divisible by 4,
6     except for end-of-century years, which must be divisible by 400.
7
8     Args:
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 [main] exited with code 0 in 0.447 seconds
26
27 [testing] python -u "C:\Users\vennela\AppData\Local\Temp\TempCodeRunner\11a.py"
28 1900 is a leap year.
29
30 [main] exited with code 0 in 0.167 seconds
31
32 [testing] python -u "C:\Users\vennela\AppData\Local\Temp\TempCodeRunner\11a.py"
33 1900 is not a leap year.
34
35 [main] 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 takes two integers `a` and `b` as input and returns their GCD. Below the code, the output of the function is displayed in a terminal window, showing the GCD of 12 and 18 as 6.

```
1 def compute_gcd(a, b):
2     while b:
3         a, b = b, a % b
4     return a
5
6 if __name__ == "__main__":
7     a = 12
8     b = 18
9     print("GCD of {a} and {b} is: {gcd}")
10    print("GCD of 12 and 18 is: {gcd}")
```

Output:

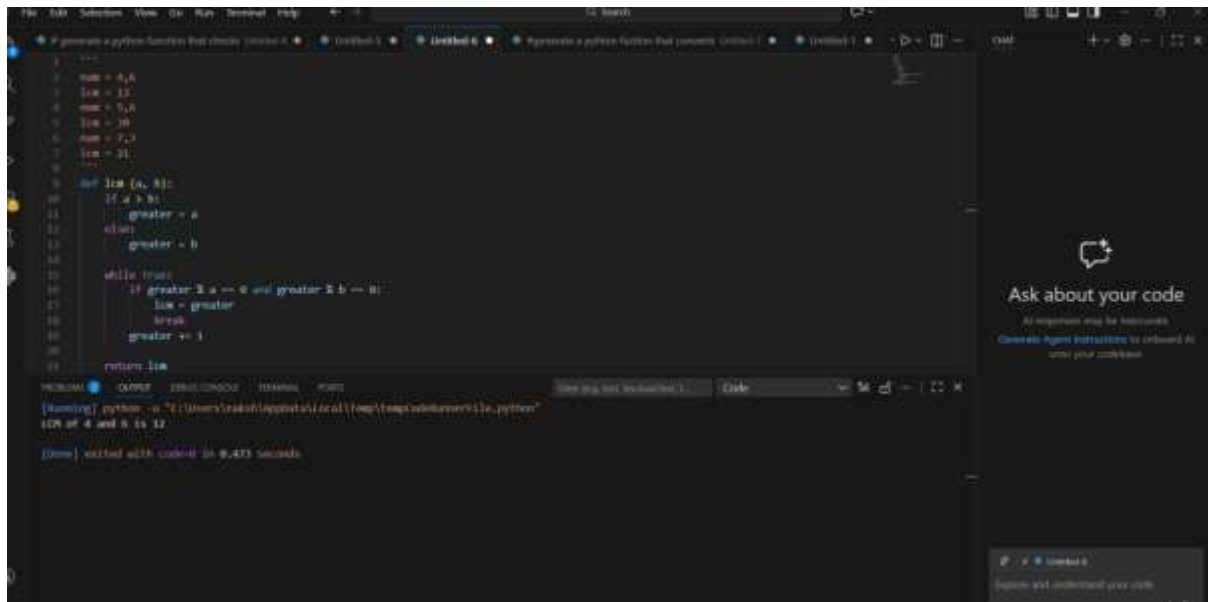
```
[Running] python -u "C:\Users\user\AppData\Local\Temp\codeeditor\1.py"
GCD of 12 and 18 is: 6
GCD of 12 and 18 is: 6
[Done] exited with code 0 in 0.422 seconds
```

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.



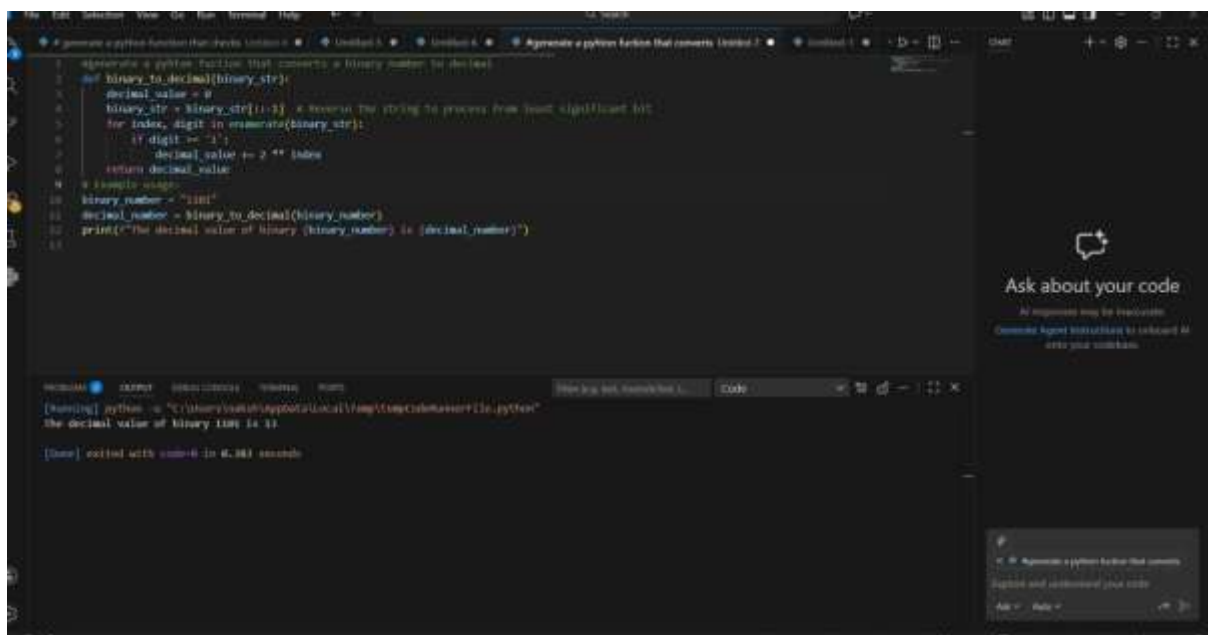
```
1 # generate a python function that checks whether a > b
2
3 a = 4
4 b = 12
5
6 max = 0
7
8 lim = 10
9
10 max = 7
11
12 lim = 21
13
14
15 def find(a, b):
16     if a > b:
17         greater = a
18     else:
19         greater = b
20
21     while True:
22         if greater > a == 0 and greater > b == 0:
23             lim = greater
24             break
25         greater += 1
26
27     return lim
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

```
[Running] python -u "C:\Users\ashish\AppData\Local\Temp\tempcodeeditorfile.py"
IDN of a and b 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
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\ashish\AppData\Local\Temp\tempcodeeditorfile.py"
The decimal value of binary 1101 is 13
[Done] exited with code=0 in 0.383 seconds
```

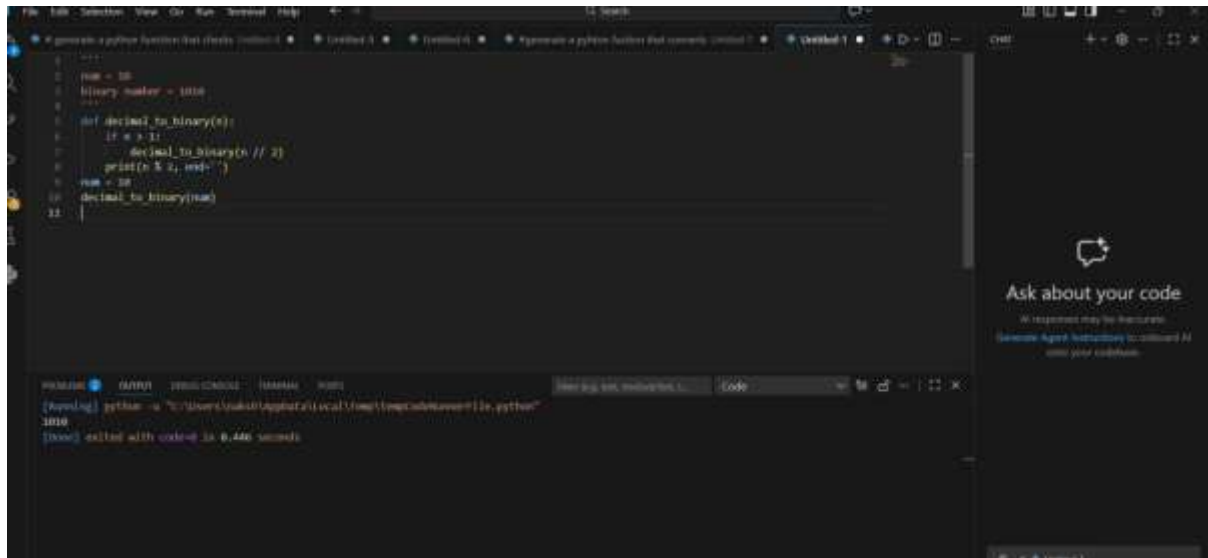
Question 5: One-Shot Prompting (Decimal to Binary Conversion)

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 with a Python function `decimal_to_binary(n)` that recursively converts a decimal number to binary. The function uses a base case of `n < 2` and a recursive call `decimal_to_binary(n // 2)`. The code is executed in a terminal, showing the output `1010` for the input `10`.

```
def decimal_to_binary(n):  
    if n < 2:  
        return str(n)  
    return decimal_to_binary(n // 2) + str(n % 2)  
  
n = 10  
print(decimal_to_binary(n))
```

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

