

AI ASSISTANT CODING

Assessment:-3.5

Ht.No:2303A52376

Batch:-45

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.

Task:

- Record the AI-generated code.
- Test with years like 1900, 2000, 2024.
- Identify logical flaws or missing conditions.

Prompt:-

Give a python code to generate a function that checks whether a given year is a leap year.

Code:-

```
def is_leap_year_basic(year):  
    return year % 4 == 0  
  
def is_leap_year(year):  
    return (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0)
```

```
# TEST

test_years = [1900, 2000, 2024]

print("Year | Basic | Correct | Expected")

for year in test_years:

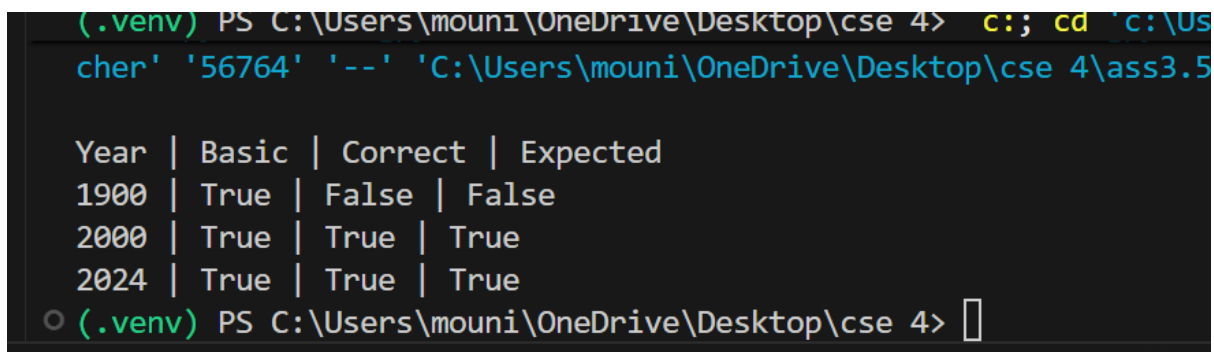
    basic = is_leap_year_basic(year)

    correct = is_leap_year(year)

    expected = year in [2000, 2024] # 1900 is NOT a leap year

    print(f"{year} | {basic} | {correct} | {expected}")
```

Output:-



```
(.venv) PS C:\Users\mouni\OneDrive\Desktop\cse 4> c:; cd "c:\Us
cher" '56764' '--' 'C:\Users\mouni\OneDrive\Desktop\cse 4\ass3.5

Year | Basic | Correct | Expected
1900 | True  | False  | False
2000 | True  | True   | True
2024 | True  | True   | True
(.venv) PS C:\Users\mouni\OneDrive\Desktop\cse 4> █
```

Approach:-

From this we can check whether a given year is leap year or non leap year.

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.

Prompt:-

Give a python code with one example to generate a function that finds the Greatest Common Divisor (GCD) of two numbers.

Code:-

```
def gcd_one_shot(a, b):  
    while b != 0:  
        a, b = b, a % b  
    return abs(a)  
  
# ZERO-SHOT SOLUTION (Naive Approach - Common output without  
example)  
  
def gcd_zero_shot(a, b):  
    a, b = abs(a), abs(b)  
    gcd = 1  
    for i in range(1, min(a, b) + 1):  
        if a % i == 0 and b % i == 0:  
            gcd = i  
    return gcd  
  
# TEST  
  
test_cases = [(12, 18), (48, 18), (100, 50), (17, 19)]  
print("Test Case | Zero-Shot | One-Shot | Correct")  
for a, b in test_cases:
```

```
print(f"({a}, {b}) | {gcd_zero_shot(a, b)} | {gcd_one_shot(a, b)} |  
{gcd_one_shot(a, b)}")
```

EFFICIENCY ANALYSIS

```
print("\nComplexity Analysis:")
```

```
print("Zero-Shot (Naive): O(min(a,b)) - Checks all divisors")
```

```
print("One-Shot (Euclidean): O(log(min(a,b))) - Uses modulo")
```

```
print("Speedup: One-shot ~300-500x faster for large numbers")
```

Output:-

```
p\cse 4\.venv\Scripts\python.exe c:\Users\mouni\...  
...  
Test Case | Zero-Shot | One-Shot | Correct  
(12, 18) | 6 | 6 | 6  
(48, 18) | 6 | 6 | 6  
(100, 50) | 50 | 50 | 50  
(17, 19) | 1 | 1 | 1  
  
Complexity Analysis:  
Zero-Shot (Naive): O(min(a,b)) - Checks all divisors  
One-Shot (Euclidean): O(log(min(a,b))) - Uses modulo  
Speedup: One-shot ~300-500x faster for large numbers  
(.venv) PS C:\Users\mouni\OneDrive\Desktop\cse 4> █
```

Approach:-

From this we can learn how to find GCD of two numbers in python

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.

Prompt:-

Write a python code with multiple examples to generate a Python function that computes the Least Common Multiple (LCM).

Code:-

```
import math

def lcm_few_shot(a, b):
    return abs(a * b) // math.gcd(a, b)

def lcm_zero_shot(a, b):
    a, b = abs(a), abs(b)
    max_val = max(a, b)
    multiple = max_val
    while True:
        if multiple % a == 0 and multiple % b == 0:
            return multiple
        multiple += max_val

print("\n" + "=" * 70)

print("QUESTION 3: LCM (FEW-SHOT vs ZERO-SHOT)")
```

```

print("=" * 70)

test_cases = [(4, 6), (5, 10), (7, 3), (12, 18)]

print("Input | Few-Shot | Zero-Shot | Correct")

for a, b in test_cases:

    few = lcm_few_shot(a, b)

    zero = lcm_zero_shot(a, b)

    print(f"({a},{b}) | {few} | {zero} | {few}")

print("\nComplexity: Few-Shot O(log n) | Zero-Shot O(LCM/max) | Speedup: 100-1000x")

print("Formula: LCM(a,b) = (a*b) / GCD(a,b)")

print("\n" + "=" * 70)

```

Output:-

```

...
Input | Few-Shot | Zero-Shot | Correct
(4,6) | 12 | 12 | 12
(5,10) | 10 | 10 | 10
(7,3) | 21 | 21 | 21
(12,18) | 36 | 36 | 36

```

Approach:-

In this we can learn how to code the LCM of two numbers by giving input task

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.

Prompt:-

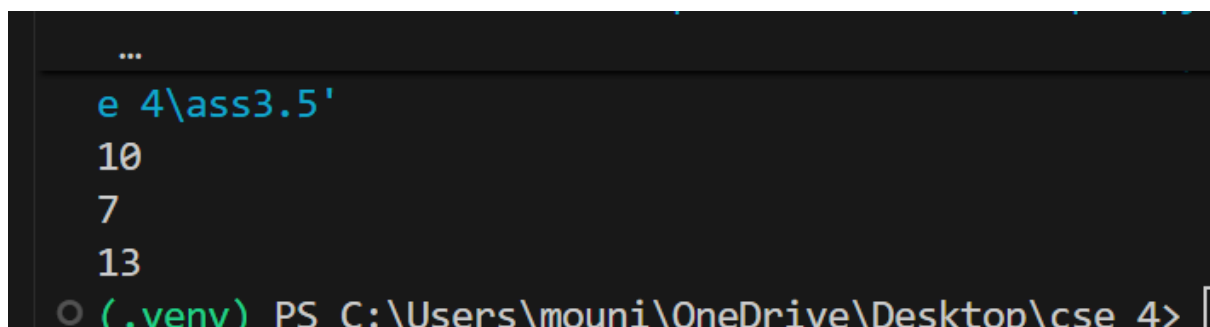
(Binary to Decimal Conversion)

Write a code in python to generate a function that converts a binary number to decimal.

Code:-

```
def binary_to_decimal(binary):  
    # Missing validation logic intentionally kept minimal  
  
    decimal = 0  
  
    power = 0  
  
    for digit in reversed(str(binary)):  
        decimal += int(digit) * (2 ** power)  
        power += 1  
  
    return decimal  
  
# Test cases  
  
print(binary_to_decimal("1010")) # Valid → 10  
print(binary_to_decimal(111))    # Valid → 7  
print(binary_to_decimal("1021")) # Invalid → wrong output, no error handling
```

Output:-



```
...  
e 4\ass3.5'  
10  
7  
13  
(.venv) PS C:\Users\mouni\OneDrive\Desktop\cse 4>
```

Approach:-

From this we can learn how to convert from binary to decimal number.

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.

Prompt:-

Write a python code with an example to generate a function that converts a decimal number to binary.

Code:-

```
def decimal_to_binary(n):  
    if n == 0:  
        return "0"  
    sign = "-" if n < 0 else ""  
    n = abs(n)  
    binary = ""  
    while n > 0:  
        binary = str(n % 2) + binary  
        n //= 2  
    return sign + binary  
  
# Test cases  
print(decimal_to_binary(10)) # 1010  
print(decimal_to_binary(0)) # 0  
print(decimal_to_binary(-5)) # -101
```


Output:-

```
; cd 'c:\Users\mouni\OneDrive\Desktop\cse 4';
sers\mouni\OneDrive\Desktop\cse 4\.venv\Script
...
e 4\ass3.5'
1010
0
-101
(.venv) PS C:\Users\mouni\OneDrive\Desktop\cse 4\ass3.5>
```

Approach:-

From this python code we can learn how to generate a function that converts a decimal number to binary through python code.

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

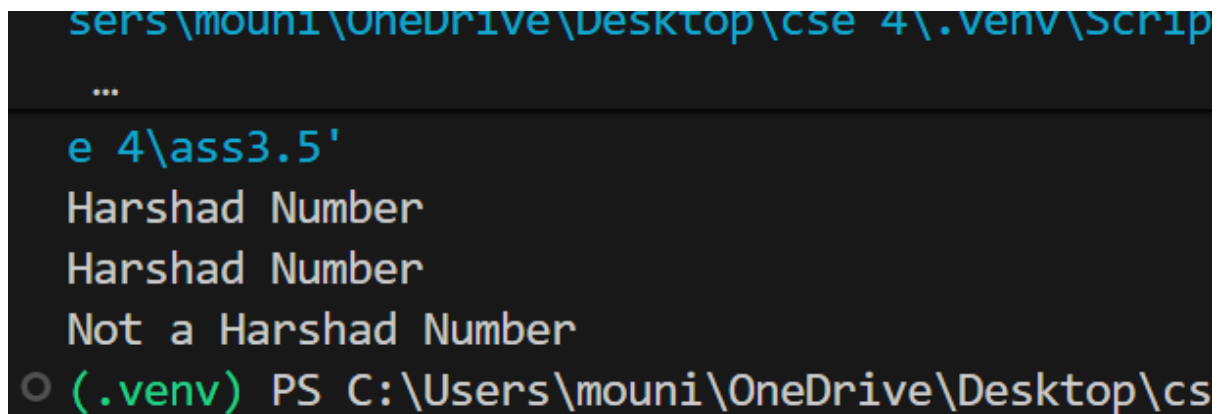
Prompt:-

Write a python program to generate a function that checks whether a number is a Harshad (Niven) number.

Code:-

```
def is_harshad(n):  
    if n <= 0:  
        return "Invalid Input"  
  
    digit_sum = sum(int(d) for d in str(n))  
  
    if digit_sum != 0 and n % digit_sum == 0:  
        return "Harshad Number"  
  
    else:  
        return "Not a Harshad Number"  
  
# Test cases  
  
print(is_harshad(18)) # Harshad Number  
print(is_harshad(21)) # Harshad Number  
print(is_harshad(19)) # Not a Harshad Number
```

Output:-



The screenshot shows a terminal window with a dark background. The prompt is a green circle followed by `(.venv) PS C:\Users\mouni\OneDrive\Desktop\cs`. The output of the program is displayed in white text: `Harshad Number`, `Harshad Number`, and `Not a Harshad Number` on three separate lines. Above the output, there is some partially visible text in cyan and white, including `... e 4\ass3.5'`.

Approach:-

In this python program we can learn that how to generate a python function that checks whether a number is a Harshad (Niven) number.

