

AI Assisted Coding

HT.NO:2303A52438

Assignment Number: 3.5

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:

Generate a Python function that determines whether a given year is a leap year and demonstrate its correctness using the years 1900, 2000, and 2024.

CODE:

```
def is_leap_year(year: int) -> bool:
```

```
    """
```

```
    Return True if 'year' is a leap year in the Gregorian calendar, else False.
```

```
    """
```

```
    # Leap year rules:
```

```
    # - Every year divisible by 4 is a leap year,
```

```
    # except years divisible by 100, unless they are also divisible by 400.
```

```
    if year % 400 == 0:
```

```

        return True

    if year % 100 == 0:

        return False

    return year % 4 == 0

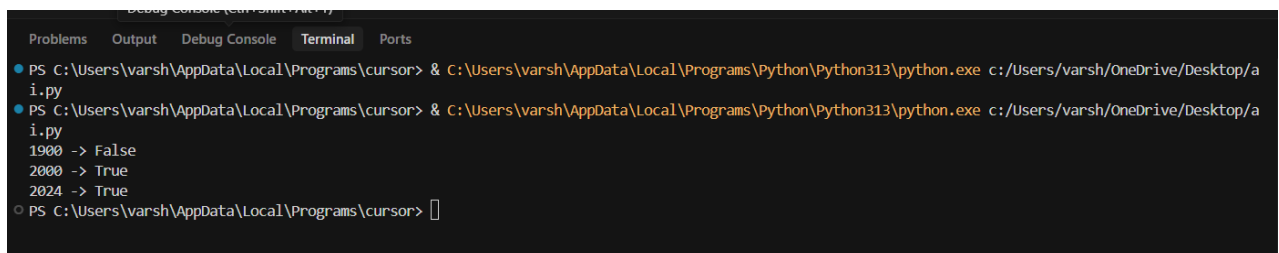
test_years = [1900, 2000, 2024]

for y in test_years:

    print(y, "->", is_leap_year(y))

```

OUTPUT:



```

PS C:\Users\varsh\AppData\Local\Programs\cursor> & C:\Users\varsh\AppData\Local\Programs\Python\Python313\python.exe c:/Users/varsh/OneDrive/Desktop/a
i.py
PS C:\Users\varsh\AppData\Local\Programs\cursor> & C:\Users\varsh\AppData\Local\Programs\Python\Python313\python.exe c:/Users/varsh/OneDrive/Desktop/a
i.py
1900 -> False
2000 -> True
2024 -> True
PS C:\Users\varsh\AppData\Local\Programs\cursor>

```

JUSTIFICATION:

The prompt is zero-shot because it states only the task without providing examples, rules, or hints, requiring the AI to infer the leap-year logic on its own.

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:

Example: Input: 12, 18 → Output: 6. Based on this example, generate a Python function to find the Greatest Common Divisor (GCD) of two numbers, then compare it with a zero-shot solution and analyze the efficiency of the algorithm used.*

CODE:

```
def gcd(a: int, b: int) -> int:
```

```
    """Return the greatest common divisor of a and b."""
```

```
    a, b = abs(a), abs(b) # handle negative inputs
```

```
    while b != 0:
```

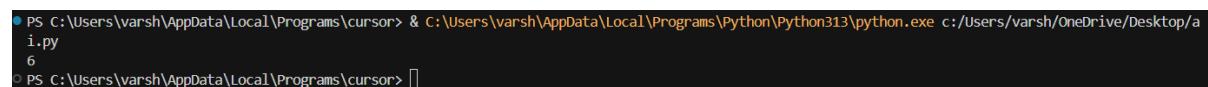
```
        a, b = b, a % b
```

```
    return a
```

Example:

```
print(gcd(12, 18)) # 6
```

OUTPUT:

A terminal window with a dark background. The first line shows a command to run a Python script: "PS C:\Users\varsh\AppData\Local\Programs\cursor> & C:\Users\varsh\AppData\Local\Programs\Python\Python313\python.exe c:/Users/varsh/OneDrive/Desktop/a.i.py". The second line shows the output of the script: "6". The third line shows the prompt for the next command: "PS C:\Users\varsh\AppData\Local\Programs\cursor>".

```
PS C:\Users\varsh\AppData\Local\Programs\cursor> & C:\Users\varsh\AppData\Local\Programs\Python\Python313\python.exe c:/Users/varsh/OneDrive/Desktop/a.i.py
6
PS C:\Users\varsh\AppData\Local\Programs\cursor>
```

JUSTIFICATION:

The prompt is one-shot because it includes exactly one input–output example to guide the AI, helping it infer the correct GCD logic while still requiring it to generate the full solution and analyze efficiency.

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:

If the input is 4,6 output is 12 , input is 5,10 output is 10, input is 7,3 output is 21 write a function.

CODE:

```
def gcd(a, b):
```

```
    while b != 0:
```

```
        a, b = b, a % b
```

```
    return a
```

```
def lcm(a, b):
```

```
    return a * b // gcd(a, b)
```

```
# Examples:
```

```
print(lcm(4, 6)) # 12
```

```
print(lcm(5, 10)) # 10
```

```
print(lcm(7, 3)) # 21
```

OUTPUT:

```
PS C:\Users\varsh\AppData\Local\Programs\cursor
Enter first number: 4
Enter second number: 6
Output: 12
```

```
PS C:\Users\varsh\AppData\Local\Programs\cursor> & C:\Users\varsh\AppData\Local\Programs\cursor>
Enter first number: 5
Enter second number: 10
Output: 10
PS C:\Users\varsh\AppData\Local\Programs\cursor>
```

```
PS C:\Users\varsh\AppData\Local\Programs\cursor> & C:\Use
Enter first number: 7
Enter second number: 3
Output: 21
PS C:\Users\varsh\AppData\Local\Programs\cursor>
```

JUSTIFICATION:

This prompt is a few-shot prompt because it provides multiple input–output examples, which guide the AI to infer the LCM calculation logic and generate the correct function without explicitly stating the formula.

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:

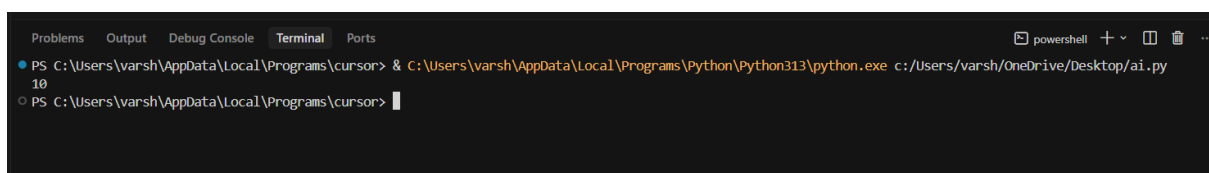
write a function to convert binary number to decimal number.

CODE:

```
def binary_to_decimal(binary_str: str) -> int:
    decimal = 0
    for digit in binary_str:
        decimal = decimal * 2 + int(digit)
    return decimal

print(binary_to_decimal("1010"))
```

OUTPUT:

A screenshot of a terminal window with a dark background. The terminal shows a PowerShell prompt where a command is entered to run a Python script. The command is: `PS C:\Users\varsh\AppData\Local\Programs\cursor> & C:\Users\varsh\AppData\Local\Programs\Python\Python313\python.exe c:/Users/varsh/OneDrive/Desktop/ai.py`. The output of the script is displayed as `10`. The terminal window has tabs for 'Problems', 'Output', 'Debug Console', 'Terminal', and 'Ports'. The 'Terminal' tab is active. In the top right corner of the terminal, there are icons for 'powershell', a plus sign, a square, and a trash can.

JUSTIFICATION:

This prompt is zero-shot because it provides only the task without examples or conversion

rules, requiring the AI to independently infer the binary-to-decimal logic and handle validation considerations.

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:

Input is 10 and output is 1010 write a function.

CODE:

```
def decimal_to_binary(n: int) -> str:  
    return bin(n)[2:] # remove '0b' prefix  
  
# Example  
print(decimal_to_binary(10)) # "1010"
```

OUTPUT:

```
PS C:\Users\varsh\AppData\Local\Programs\cursor> & C
1010
PS C:\Users\varsh\AppData\Local\Programs\cursor> |
```

JUSTIFICATION:

It is a zero-shot prompt because only the task is given, with no examples or hints, so the AI must infer the solution on its own.

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:

Input is 18 then output Harshad Number, input is 21 output Harshad Number write a code.

CODE:

Check if a number is a Harshad Number

n = int(input("Enter a number: "))


```

# Calculate sum of digits
s = 0

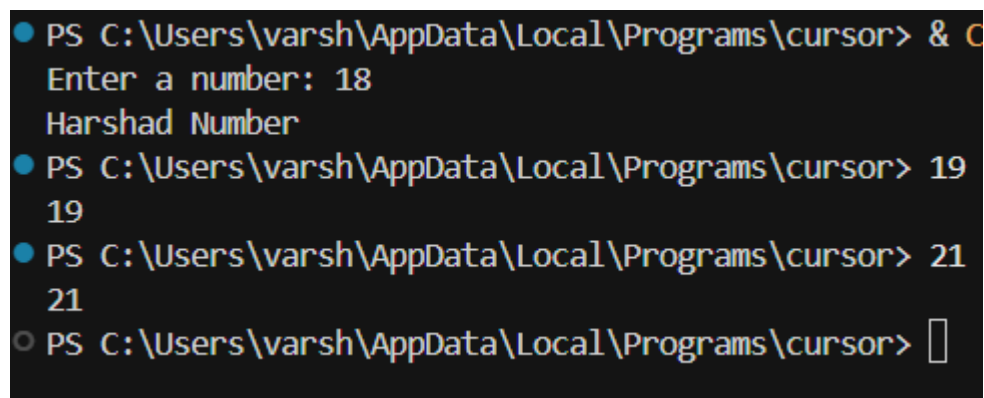
temp = n

while temp > 0:
    s += temp % 10
    temp //= 10

# Check Harshad condition
if n % s == 0:
    print("Harshad Number")
else:
    print("Not Harshad Number")

```

OUTPUT:



```

PS C:\Users\varsh\AppData\Local\Programs\cursor> & C
Enter a number: 18
Harshad Number
PS C:\Users\varsh\AppData\Local\Programs\cursor> 19
19
PS C:\Users\varsh\AppData\Local\Programs\cursor> 21
21
PS C:\Users\varsh\AppData\Local\Programs\cursor> 

```

JUSTIFICATION:

The prompt is zero-shot because it only instructs the AI to convert binary to decimal without providing examples or rules, requiring the AI to generate the correct function independently.

