

AI ASSISTED CODING ASSIGNMENT – 3.5

Name :P.Pooja

Hallticket:2303A510F7

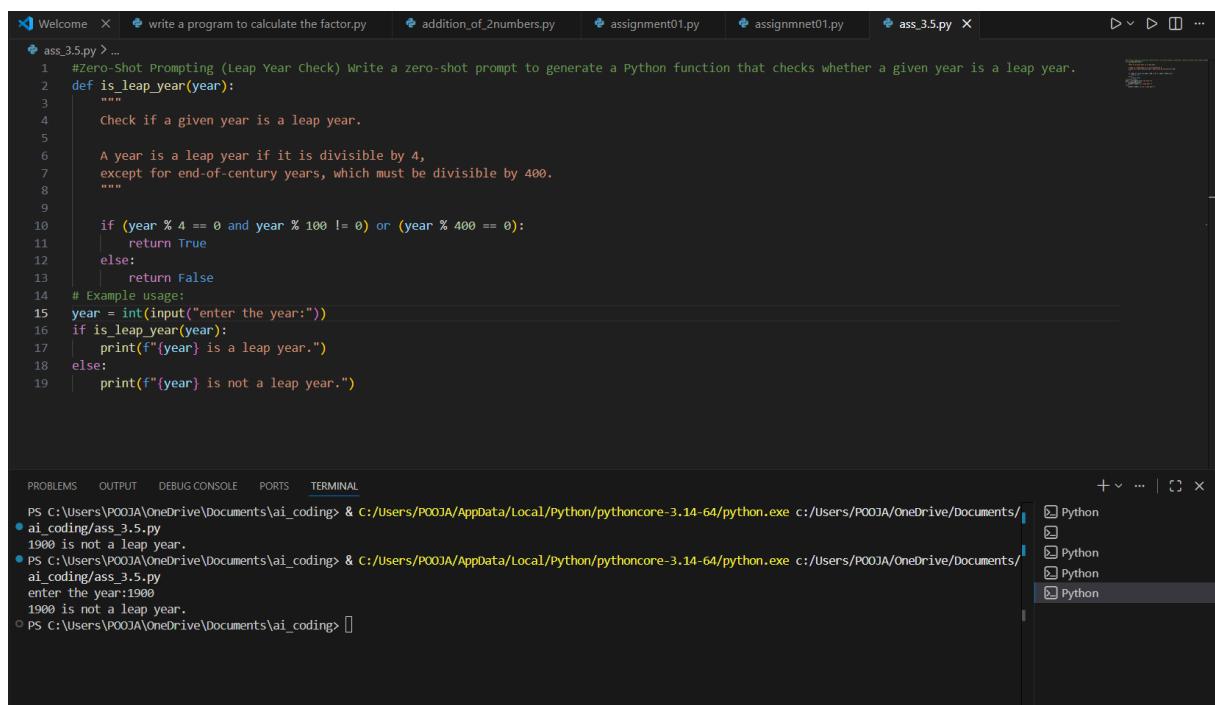
Batch:03

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.



The screenshot shows a code editor window with multiple tabs at the top: "Welcome", "write a program to calculate the factor.py", "addition_of_2numbers.py", "assignment01.py", "assgnmnet01.py", and "ass 3.5.py". The "ass 3.5.py" tab is active, displaying the following Python code:

```
#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.
def is_leap_year(year):
    """
    Check if a given year is a leap year.

    A year is a leap year if it is divisible by 4,
    except for end-of-century years, which must be divisible by 400.
    """

    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return True
    else:
        return False

# Example usage:
year = int(input("Enter the year:"))
if is_leap_year(year):
    print(f"{year} is a leap year.")
else:
    print(f"{year} is not a leap year.")
```

Below the code editor, there is a terminal window showing the execution of the script. The terminal output is:

```
PS C:\Users\POOJA\OneDrive\Documents\ai_coding> & C:/Users/POOJA/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/POOJA/OneDrive/Documents/ai_coding/ass_3.5.py
1900 is not a leap year.
PS C:\Users\POOJA\OneDrive\Documents\ai_coding> & C:/Users/POOJA/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/POOJA/OneDrive/Documents/ai_coding/ass_3.5.py
Enter the year:1900
1900 is not a leap year.
PS C:\Users\POOJA\OneDrive\Documents\ai_coding> []
```

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.

```
ass_3_5.py > ...
20
21 def gcd_euclidean(a, b):
22     while b:
23         a, b = b, a % b
24     return a
25
26
27 def gcd_subtraction(a, b):
28     while a != b:
29         if a > b:
30             a = a - b
31         else:
32             b = b - a
33     return a
34
35
36 print("Example 1: GCD(12, 18)")
37 print(f"Euclidean: {gcd_euclidean(12, 18)}")
38 print(f"Subtraction: {gcd_subtraction(12, 18)}")
39
40 print("\nExample 2: GCD(48, 18)")
41 print(f"Euclidean: {gcd_euclidean(48, 18)}")
42 print(f"Subtraction: {gcd_subtraction(48, 18)}")
```

PROBLEMS OUTPUT DEBUG CONSOLE PORTS TERMINAL

```
1900 is not a leap year.
PS C:\Users\POOJA\OneDrive\Documents\ai_coding> & C:/Users/POOJA/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/POOJA/OneDrive/Documents/
ai_coding/ass_3_5.py
Example 1: GCD(12, 18)
Euclidean: 6
Subtraction: 6
Example 2: GCD(48, 18)
Euclidean: 6
Subtraction: 6
PS C:\Users\POOJA\OneDrive\Documents\ai_coding> []
```

+ v ... | ☰ x

Python
Python
Python
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.

```

44     def gcd(a, b):
45         while b:
46             a, b = b, a % b
47         return a
48     def lcm_formula(a, b):
49         return (a * b) // gcd(a, b)
50     def lcm_brute(a, b):
51         max_val = max(a, b)
52         multiple = max_val
53         while True:
54             if multiple % a == 0 and multiple % b == 0:
55                 return multiple
56             multiple += max_val
57     print("Example 1: LCM(4, 6) = 12")
58     print(f"Formula-based: {lcm_formula(4, 6)}")
59     print(f"Brute force: {lcm_brute(4, 6)}")
60     print("\nExample 2: LCM(5, 10) = 10")
61     print(f"Formula-based: {lcm_formula(5, 10)}")
62     print(f"Brute force: {lcm_brute(5, 10)}")
63     print("\nExample 3: LCM(7, 3) = 21")
64     print(f"Formula-based: {lcm_formula(7, 3)}")
65     print(f"Brute force: {lcm_brute(7, 3)}")
66     print("\nEdge Cases:")
67     print(f"LCM(1, 5) = {lcm_formula(1, 5)}")
68     print(f"LCM(10, 10) = {lcm_formula(10, 10)}")
69     print(f"LCM(100, 50) = {lcm_formula(100, 50)}")

```

PROBLEMS OUTPUT DEBUG CONSOLE PORTS TERMINAL

PS C:\Users\POOJA\OneDrive\Documents\ai_coding> & C:/Users/POOJA/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/POOJA/OneDrive/Documents/ai_coding/ass_3.5.py

Example 1: LCM(4, 6) = 12
Formula-based: 12
Brute force: 12

Example 2: LCM(5, 10) = 10
Formula-based: 10
Brute force: 10

Example 3: LCM(7, 3) = 21
Formula-based: 21
Brute force: 21

Edge Cases:
 $\text{LCM}(1, 5) = 5$
 $\text{LCM}(10, 10) = 10$
 $\text{LCM}(100, 50) = 100$

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.

The screenshot shows a code editor window with a dark theme. On the left is a code editor pane containing Python code for binary conversion and validation. On the right is a terminal pane showing the execution of the code and its output.

```
ass_3.5.py > ...
71 def binary_to_decimal(binary_string):
72     return int(binary_string, 2)
73
74
75 def binary_validated(binary_string):
76     if not all(c in '01' for c in binary_string):
77         return None
78     return int(binary_string, 2)
79
80
81 print("Valid Inputs:")
82 print(f"1010 = {binary_to_decimal('1010')}")
83 print(f"11111 = {binary_to_decimal('11111')}")
84
85 print("\nInvalid Inputs:")
86 print(f"1234 = {binary_validated('1234')}")
87 print(f"abc = {binary_validated('abc')}")
```

TERMINAL

```
PROBLEMS OUTPUT DEBUG CONSOLE PORTS TERMINAL + × ⋮ | ✎ ×
PS C:\Users\POOJA\OneDrive\Documents\ai_coding> & C:/Users/POOJA/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/POOJA/OneDrive/Documents/ai_coding/ass_3.5.py
LCM(1, 5) = 5
LCM(10, 10) = 10
LCM(100, 50) = 100
PS C:\Users\POOJA\OneDrive\Documents\ai_coding> & C:/Users/POOJA/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/POOJA/OneDrive/Documents/ai_coding/ass_3.5.py
Valid Inputs:
1010 = 10
11111 = 31
Invalid Inputs:
1234 = None
abc = None
PS C:\Users\POOJA\OneDrive\Documents\ai_coding>
```

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 interface with several tabs at the top: 'Welcome', 'write a program to calculate the factor.py', 'addition_of_2numbers.py', 'assignment01.py', 'assignmnet01.py', 'ass_3.5.py' (which is the active tab), and '...'. The main area contains Python code for a function named `decimal_to_binary`. The code includes docstrings, type hints for arguments and return values, and example usage prints. Below the code editor is a terminal window showing the execution of the script `ass_3.5.py`. The terminal output shows the conversion of decimal numbers 10, 255, 0, and 7 to their binary equivalents. It also handles invalid input by raising a `ValueError`.

```

1 ass_3.5.py > print("abc = {binary_val}abc", abc)
2
3 #Write a one-shot prompt with an example to generate a Python function that converts a decimal number to binary.
4 def decimal_to_binary(n):
5     """
6         Convert a decimal number to its binary representation.
7
8     Args:
9         n (int): A decimal number.
10
11     Returns:
12         str: The binary representation of the decimal number.
13     """
14     if n < 0:
15         raise ValueError("Input must be a non-negative integer.")
16     binary_representation = bin(n).replace("0b", "")
17     return binary_representation
18
19 # Example usage:
20 print("Decimal: 10 -> Binary:", decimal_to_binary(10))
21 print("Decimal: 255 -> Binary:", decimal_to_binary(255))
22 print("Decimal: 0 -> Binary:", decimal_to_binary(0))
23 print("Decimal: 7 -> Binary:", decimal_to_binary(7))

PROBLEMS OUTPUT DEBUG CONSOLE PORTS TERMINAL
PS C:\Users\POOJA\OneDrive\Documents\ai_coding> & C:/Users/POOJA/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/POOJA/OneDrive/Documents/ai_coding/ass_3.5.py
1010 = 10
11111 = 31

Invalid Inputs:
1234 = None
abc = None
PS C:\Users\POOJA\OneDrive\Documents\ai_coding> & C:/Users/POOJA/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/POOJA/OneDrive/Documents/ai_coding/ass_3.5.py
Decimal: 10 -> Binary: 1010
Decimal: 255 -> Binary: 1111111
Decimal: 0 -> Binary: 0
Decimal: 7 -> Binary: 111
PS C:\Users\POOJA\OneDrive\Documents\ai_coding>

```

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

The screenshot shows a code editor interface with a dark theme. In the top tab bar, several files are listed: Welcome, write a program to calculate the factor.py, addition_of_2numbers.py, assignment01.py, assignmnet01.py, ass_3.5.py (which is the active file), and ass_3.5.py. The code in the editor is as follows:

```
ass_3.5.py > ...
  print("Decimal: / -> binary: , decimal_to_binary()")
109
110  # 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
111  def is_harshad_number(n):
112
113
114      # Calculate the sum of the digits
115      digit_sum = sum(int(digit) for digit in str(n))
116
117      # Check if n is divisible by the sum of its digits
118      return n % digit_sum == 0
119
120
121  # Example usage:
122  print("Is 18 a Harshad number", is_harshad_number(18))
```

Below the code editor is a terminal window showing the following session:

```
PROBLEMS OUTPUT DEBUG CONSOLE PORTS TERMINAL
PS C:\Users\POOJA\OneDrive\Documents\ai_coding> & C:/Users/POOJA/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/POOJA/OneDrive/Documents/ai_coding/ass_3.5.py
● PS C:\Users\POOJA\OneDrive\Documents\ai_coding> & C:/Users/POOJA/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/POOJA/OneDrive/Documents/ai_coding/ass_3.5.py
Decimal: 10 -> Binary: 1010
Decimal: 255 -> Binary: 11111111
Decimal: 0 -> Binary: 0
Decimal: 7 -> Binary: 111
● PS C:\Users\POOJA\OneDrive\Documents\ai_coding> & C:/Users/POOJA/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/POOJA/OneDrive/Documents/ai_coding/ass_3.5.py
Is 18 a Harshad number True
● PS C:\Users\POOJA\OneDrive\Documents\ai_coding> & C:/Users/POOJA/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/POOJA/OneDrive/Documents/ai_coding/ass_3.5.py
Is 18 a Harshad number False
○ PS C:\Users\POOJA\OneDrive\Documents\ai_coding>
```

On the right side of the interface, there is a sidebar with several Python-related items listed under "Python".

