

AI Assisted Coding

Assignment – 4.3

Name: A.Rithika

Roll No: 2303A54048

Batch: 48

Lab 4: Advanced Prompt Engineering – Zero-shot, One-shot, and Few-shot Techniques

Task 1: Zero-Shot Prompting – Leap Year Check

Scenario:

Zero-shot prompting involves giving instructions without providing examples.

Prompt used : #Write a Python function that takes a year as input and checks whether it is a leap year.

#The function should return an appropriate message indicating whether the year is a leap year or not.

#Do not include any input-output examples.

- **Generated code :**

```
def is_leap_year(year):  
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):  
        return True  
    else:  
        return False  
  
year = int(input("Enter year: "))  
print(is_leap_year(year))
```

- **Sample Input :**

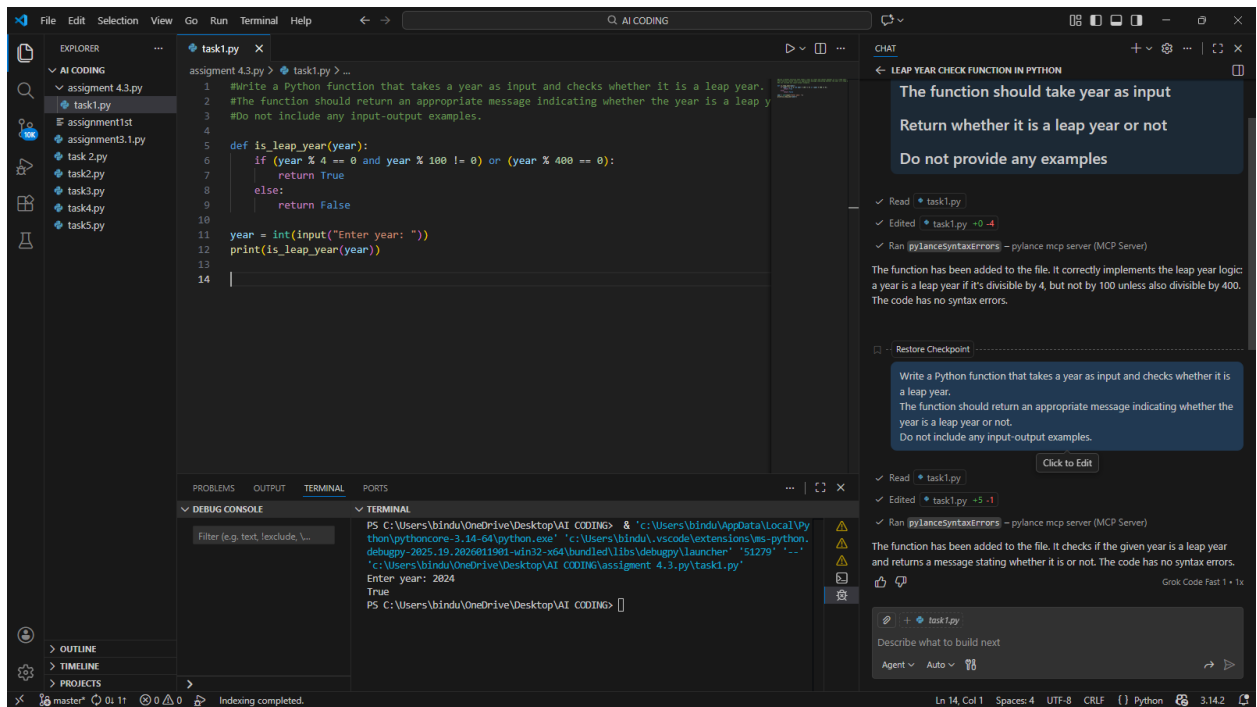
Enter year:2024

- **Sample Output :**

True

- **Short Explanation of Logic :**

the program checks the leap year conditions using logical operators .a year is a leap year if the it is divisible by 400 or divisibe by 4 but not divisible by 100 . the logic is implemented directly without providing any prior .



Task 2: One-Shot Prompting – Centimeters to Inches Conversion

❖ Scenario : One-shot prompting guides AI using a single example.

- **Prompt used :**
- # Write a Python function to convert centimeters to inches
- # Use the formula inches = centimeters / 2.54
- # Example:
- # Input: 10 cm
- # Output: 3.94 inches
- **Generated code :**

```
def cm_to_inches(centimeters):
    """
    Convert centimeters to inches.

    Args:
        centimeters: A number representing length in centimeters

    Returns:
        A float representing the length in inches
    """
    inches = centimeters / 2.54
    return round(inches, 2)

# Example usage
if __name__ == "__main__":
    cm_value = 10
    result = cm_to_inches(cm_value)
    print(f"Input: {cm_value} cm")
    print(f"Output: {result} inches")
```

- **Sample Input :**
- Enter value in cm: 25.4
- **Sample Output :**
- 10.0 inches

Short Explanation of Logic :

The function takes centimeters as input and converts it into inches using the standard formula $\text{Cm} \times 2.54$ one example in the prompt helps the ai to generate the logic accurately in one short

```

1  def cm_to_inches(centimeters):
2      """
3      Convert centimeters to inches.
4
5      Args:
6          centimeters: A number representing length in centimeters
7
8      Returns:
9          A float representing the length in inches
10     """
11     inches = centimeters / 2.54
12     return round(inches, 2)
13
14
15 # Example usage
16 if __name__ == "__main__":
17     cm_value = 10
18     result = cm_to_inches(cm_value)
19     print(f"Input: {cm_value} cm")
20     print(f"Output: {result} inches")
  
```

TERMINAL

```

PS C:\Users\bindu\OneDrive\Desktop\AI CODING> & C:\Users\bindu\AppData\Local\Python\pythoncore-3.14-64\python.exe "c:\Users\bindu\OneDrive\Desktop\AI CODING\assignment 4.3.py\task1.py"
PS C:\Users\bindu\OneDrive\Desktop\AI CODING> ^C
PS C:\Users\bindu\OneDrive\Desktop\AI CODING> & C:\Users\bindu\AppData\Local\Python\pythoncore-3.14-64\python.exe "c:\Users\bindu\OneDrive\Desktop\AI CODING\assignment 4.3.py\task2.py"
Input: 10 cm
Output: 3.94 inches
PS C:\Users\bindu\OneDrive\Desktop\AI CODING>
  
```

Task3:Few-shot prompting-Name Formatting

Scenario:

Few-shot prompting improves accuracy by providing multiple examples.



- **Prompt used :**
- # Write a Python function to format names as "Last, First"
- # Examples:
- # "John Smith" -> "Smith, John"
- # "Anita Rao" -> "Rao, Anita"

- **Generated code :**

```
def format_name(full_name):  
    """  
    Format a full name as "Last, First"  
  
    Args:  
        full_name (str): A full name in "First Last" format  
  
    Returns:  
        str: The name formatted as "Last, First"  
    """  
    parts = full_name.split()  
    if len(parts) >= 2:  
        first_name = parts[0]  
        last_name = parts[-1]  
        return f"{last_name}, {first_name}"  
    return full_name  
  
# Test examples  
print(format_name("John Smith"))      # Smith, John  
print(format_name("Anita Rao"))       # Rao, Anita
```

- **Sample Input :**

Enter full name: John Smith

- **Sample Output : Smith, John**

Short Explanation of Logic : The function splits the full name into first and last names and rearranges them in the required format it providing multiple examples improves few short

```

1 def format_name(full_name):
2     """
3     Format a full name as "Last, First"
4
5     Args:
6         full_name (str): A full name in "First Last" format
7
8     Returns:
9         str: The name formatted as "Last, First"
10    """
11    parts = full_name.split()
12    if len(parts) >= 2:
13        first_name = parts[0]
14        last_name = parts[-1]
15        return f"{last_name}, {first_name}"
16    return full_name
17
18 # Test examples
19 print(format_name("John Smith")) # Smith, John
20 print(format_name("Anita Rao")) # Rao, Anita

```

Task 4: Comparative Analysis – Zero-Shot vs Few-Shot

- **Scenario:**
Different prompt strategies may produce different code quality.
- **Prompt 1: Zero-Shot Prompting**

Write a Python function that counts the number of vowels in a given string.

The function should return the total count.

Do not provide any examples.

Generated code:

```

def count_vowels(string):
    vowels = "aeiouAEIOU"
    count = 0
    for char in string:
        if char in vowels:

```

- **Sample Input :**

Enter string: Hello World

- **Sample Output :**

Number of vowels: 3

- **Prompt 2: Few-Shot Prompting**

- Write a Python function to count vowels in a string.
- Examples:
- Input: "hello" → Output: 2
- Input: "AI Assisted Coding" → Output: 7

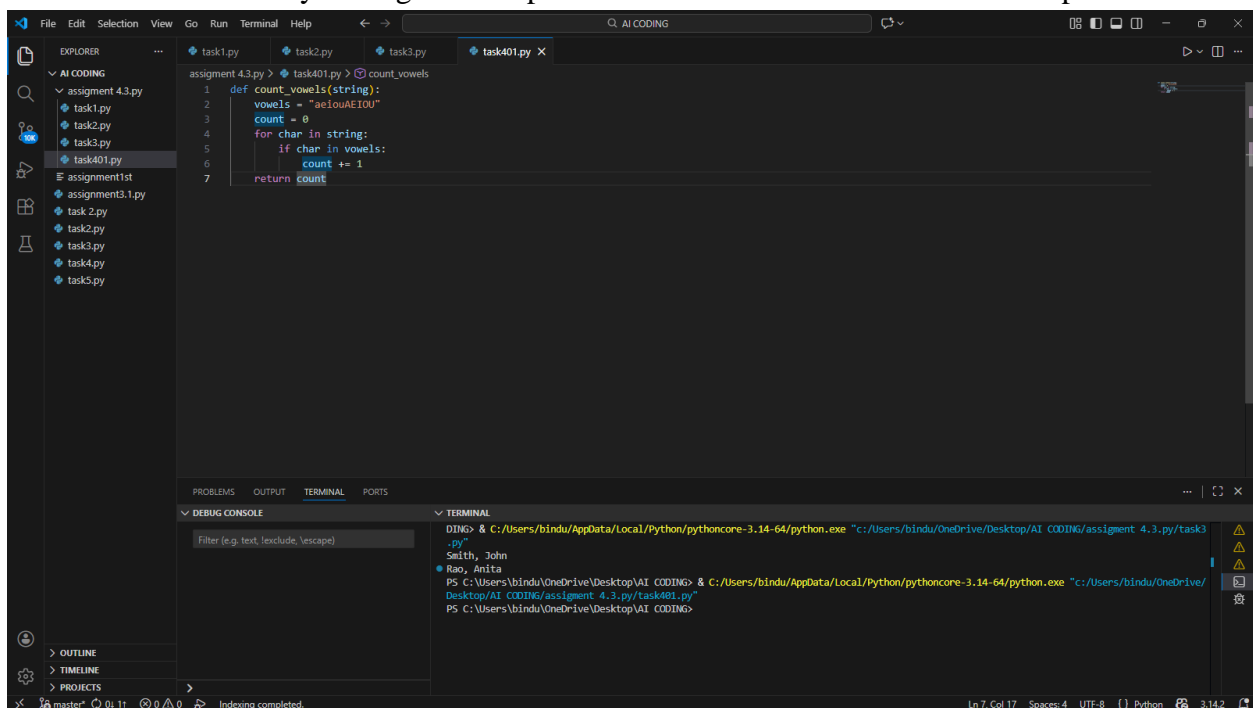
❖ **Generated code:**

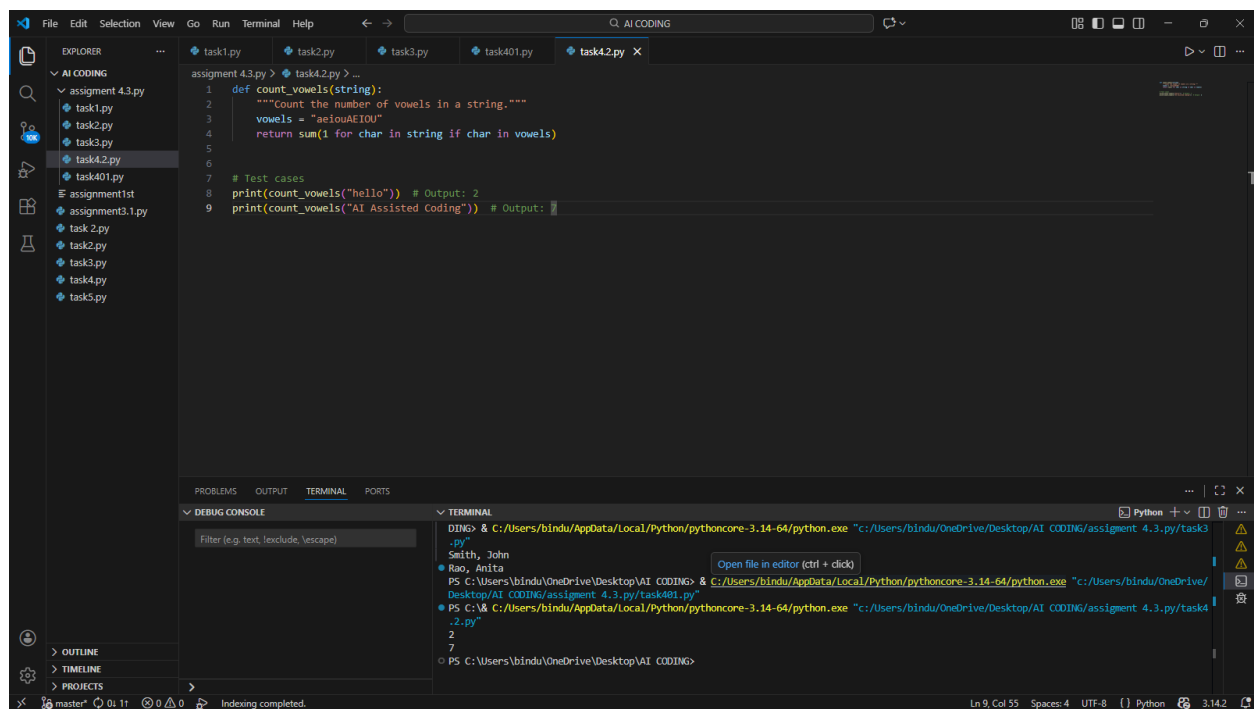
```
❖ def count_vowels(string):  
❖     """Count the number of vowels in a string."""  
❖     vowels = "aeiouAEIOU"  
❖     return sum(1 for char in string if char in vowels)  
❖  
❖ # Test cases  
❖ print(count_vowels("hello")) # Output: 2  
❖ print(count_vowels("AI Assisted Coding")) # Output: 7
```

- **Sample Input :**
Enter string: Hello World
- **Sample Output :3**

Explanation (Few-Shot);

The function uses a predefined vowel set and python's sum with the generator expression to count vowels efficiently the logic is compact and easier to understand due to example .





Comparison Table

Criteria	Zero-Shot	Few-Shot
Accuracy	Correct	Correct
Readability	Medium	High
Logical Clarity	Simple	Very clear
Code Length	Longer	Shorter

Task 5: Few-Shot Prompting – File Handling (Line Count)

Scenario : You are building a text file analyzer that counts the number of lines in a file. Clear examples help ensure correct file handling.

Prompt used :

Write a Python function to read a text file

The function should take the file name as input

Open the file in read mode# Count the total number of lines in the file

#Return the line count as an integer

Example:

If "file.txt" contains 3 lines, the output should be 3

Generated code :

```
def count_lines(filename):
    """
    Read a text file and count the total number of lines.

    Args:
        filename (str): The name of the file to read

    Returns:
        int: The total number of lines in the file
    """
    try:
        with open(filename, 'r') as file:
            line_count = sum(1 for line in file)
            return line_count
    except FileNotFoundError:
        print(f"Error: File '{filename}' not found.")
        return 0

# Example usage
if __name__ == "__main__":
    result = count_lines("file.txt")
    print(f"Total lines: {result}")
```

- **Sample Input:**
 - `print(count_lines("file.txt"))`
 - **Sample Output:**
- 3

Short Explanation of Logic:

The program reads a text file in read mode and counts the number of lines present in it a counter variable is used to keep track of the number of lines. Each time a line is read from the file the counter increases by one. After reading the entire file, the function returns the total number of lines. This method ensures accurate line counting and is easy to understand

File Edit Selection View Go Run Terminal Help

task1.py task2.py task3.py task401.py task42.py task5.py X

assignment 4.3.py > task5.py > ...

```
1 def count_lines(filename):
2     """
3     Read a text file and count the total number of lines.
4
5     Args:
6         filename (str): The name of the file to read
7
8     Returns:
9         int: The total number of lines in the file
10    """
11    try:
12        with open(filename, 'r') as file:
13            line_count = sum(1 for line in file)
14            return line_count
15    except FileNotFoundError:
16        print(f"Error: File '{filename}' not found.")
17        return 0
18
19
20 # Example usage
21 if __name__ == "__main__":
22     result = count_lines("file.txt")
23     print(f"Total lines: {result}")
```

Run and Debug

To customize Run and Debug create a launch.json file.

Debug using a terminal command or in an interactive chat.

Show automatic Python configurations

PROBLEMS OUTPUT TERMINAL PORTS

DEBUG CONSOLE

Filter (e.g. text, exclude, \escape)

TERMINAL

```
PS C:\Users\bindu\OneDrive\Desktop\AI CODING> & C:\Users\bindu\AppData\Local\Python\pythoncore-3.14-64\python.exe "c:/Users/bindu/OneDrive/Desktop/AI CODING/assignment 4.3.py/task5.py"
Error: File 'file.txt' not found.
Total lines: 0
PS C:\Users\bindu\OneDrive\Desktop\AI CODING>
```

BREAKPOINTS

- ☐ Raised Excepti...
- ☒ Uncaught Exce...
- ☐ User Uncaught...

master* 0:11 0 0 0 Indexing completed.

Ln 23, Col 36 Spaces: 4 UTF-8 Python 3.142