

AI Assisted Coding

Week2 – Wednesday

Name: Pruthagwin Challuri

Batch: 07

Roll No.: 2303A51951

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

Lab Objectives

- To explore and apply different levels of prompt examples in AI-assisted code generation
 - To understand how zero-shot, one-shot, and few-shot prompting affect AI output quality
 - To evaluate the impact of context richness and example quantity on AI performance
 - To build awareness of prompt strategy effectiveness for different problem types
- Week2 - Wednesday

Lab Outcomes (LOs)

After completing this lab, students will be able to:

- Use zero-shot prompting to instruct AI with minimal context
- Use one-shot prompting with a single example to guide AI code generation
- Apply few-shot prompting using multiple examples to improve AI responses
- Compare AI outputs across different prompting strategies

Task 1: Zero

Leap Year Check

Scenario

Zero-shot prompting involves giving instructions without providing examples. **Task Description**

Use zero-shot prompting to instruct an AI tool to generate a Python function that:

- Accepts a year as input

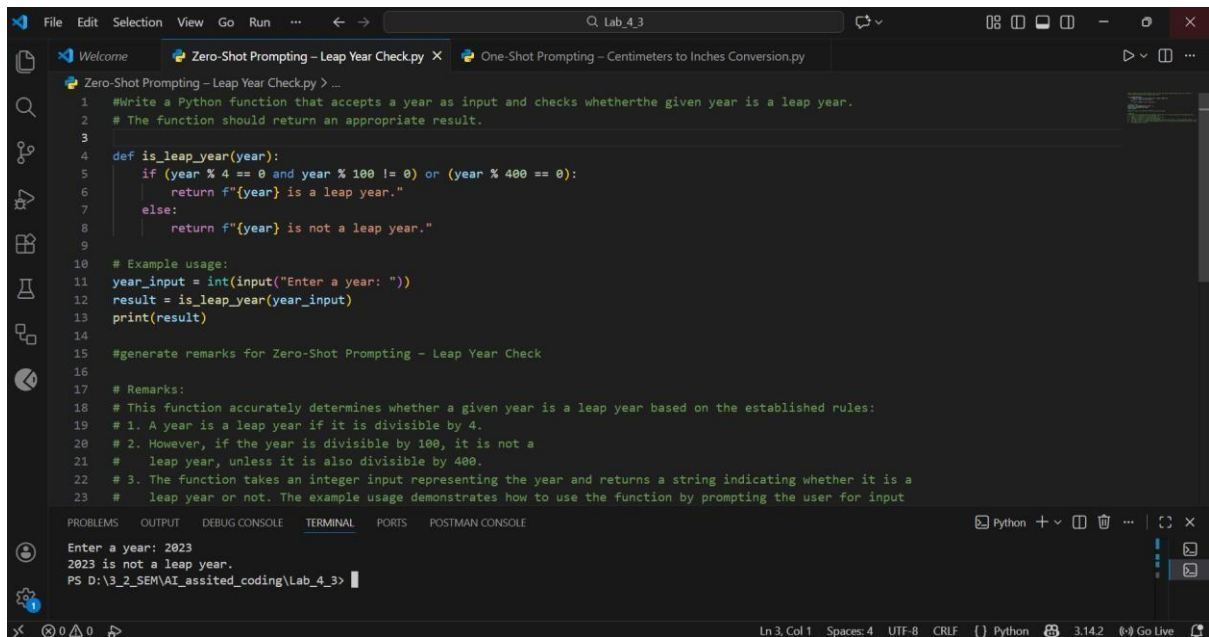
-Shot Prompting –

- Checks whether the given year is a leap year
- Returns an appropriate result

Note: No input-output examples should be provided in the prompt.

Expected Output

- AI-generated leap year checking function
- Correct logical conditions
- Sample input and output
- Screenshot of AI-generated response (if required)



```
1 #Write a Python function that accepts a year as input and checks whether the given year is a leap year.
2 # The function should return an appropriate result.
3
4 def is_leap_year(year):
5     if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
6         return f"{year} is a leap year."
7     else:
8         return f"{year} is not a leap year."
9
10 # Example usage:
11 year_input = int(input("Enter a year: "))
12 result = is_leap_year(year_input)
13 print(result)
14
15 #generate remarks for Zero-Shot Prompting - Leap Year Check
16
17 # Remarks:
18 # This function accurately determines whether a given year is a leap year based on the established rules:
19 # 1. A year is a leap year if it is divisible by 4.
20 # 2. However, if the year is divisible by 100, it is not a
21 #    leap year, unless it is also divisible by 400.
22 # 3. The function takes an integer input representing the year and returns a string indicating whether it is a
23 #    leap year or not. The example usage demonstrates how to use the function by prompting the user for input
```

Enter a year: 2023
2023 is not a leap year.
PS D:\3_2_SEM\AI_assisted_coding\Lab_4_3>

Task 2: One

Centimeters to Inches Conversion

Scenario

One-shot prompting guides AI using a single example.

Task Description

Use one-shot prompting by providing one input-output example to generate a Python function that:

- Converts centimeters to inches
- Uses the correct mathematical formula

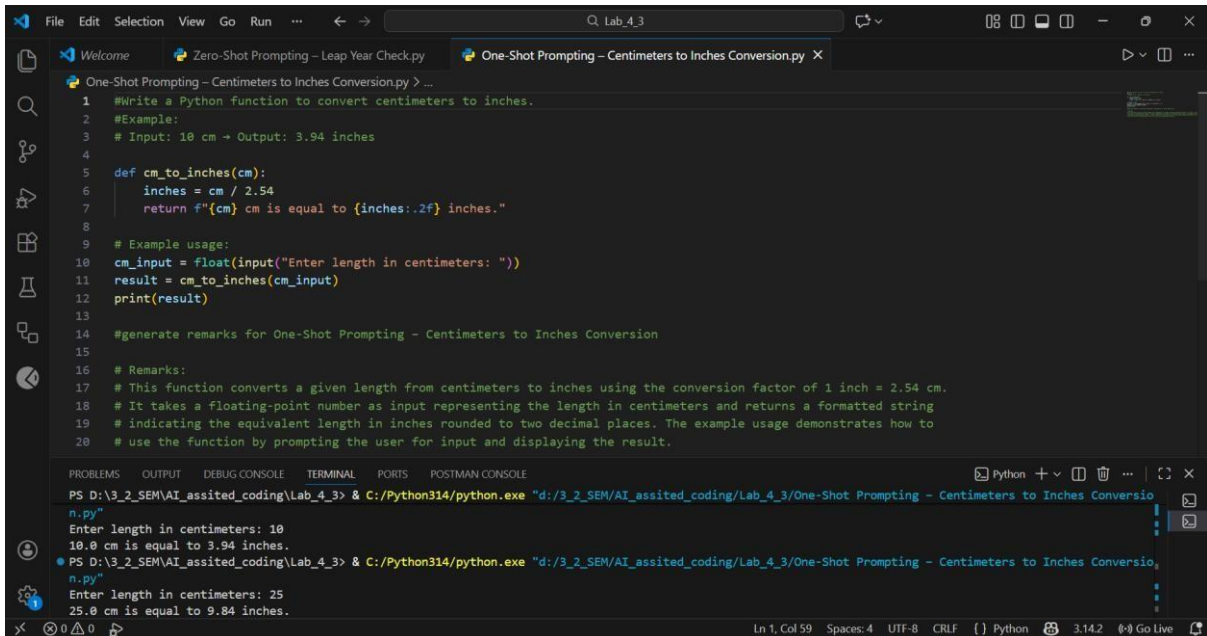
Example provided in prompt:

Input: 10 cm → Output: 3.94 inches

Expected Output

-Shot Prompting –

- Python function with correct conversion logic
- Accurate calculation
- Sample test cases and outputs



```
1 #Write a Python function to convert centimeters to inches.
2 #Example:
3 # Input: 10 cm → Output: 3.94 inches
4
5 def cm_to_inches(cm):
6     inches = cm / 2.54
7     return f"{cm} cm is equal to {inches:.2f} inches."
8
9 # Example usage:
10 cm_input = float(input("Enter length in centimeters: "))
11 result = cm_to_inches(cm_input)
12 print(result)
13
14 #generate remarks for One-Shot Prompting - Centimeters to Inches Conversion
15
16 # Remarks:
17 # This function converts a given length from centimeters to inches using the conversion factor of 1 inch = 2.54 cm.
18 # It takes a floating-point number as input representing the length in centimeters and returns a formatted string
19 # indicating the equivalent length in inches rounded to two decimal places. The example usage demonstrates how to
20 # use the function by prompting the user for input and displaying the result.
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE

```
PS D:\3_2_SEM\AI_assited_coding\Lab_4_3> & C:/Python314/python.exe "d:/3_2_SEM/AI_assited_coding/Lab_4_3/One-Shot Prompting - Centimeters to Inches Conversion.py"
Enter length in centimeters: 10
10.0 cm is equal to 3.94 inches.
PS D:\3_2_SEM\AI_assited_coding\Lab_4_3> & C:/Python314/python.exe "d:/3_2_SEM/AI_assited_coding/Lab_4_3/One-Shot Prompting - Centimeters to Inches Conversion.py"
Enter length in centimeters: 25
25.0 cm is equal to 9.84 inches.
```

Task 3: Few

Name Formatting

Scenario

Few-shot prompting improves accuracy by providing multiple examples.

Task Description

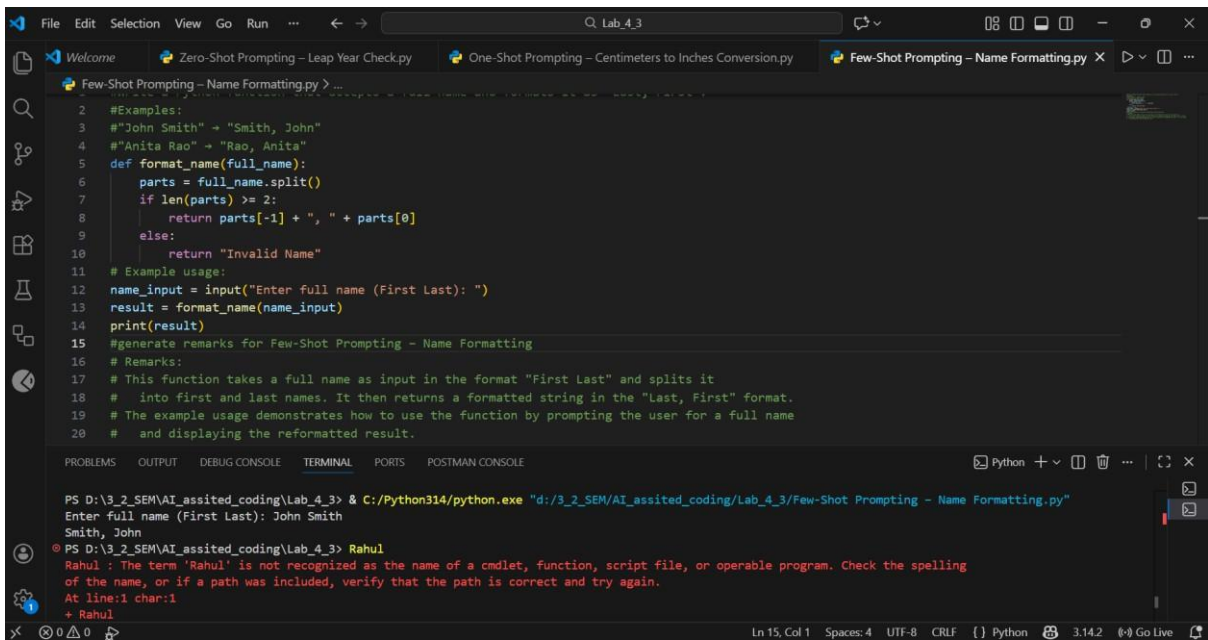
Use few-shot prompting with 2–3 examples to generate a Python function that:

- Accepts a full name as input
- Formats it as “Last, First” Example formats:
- "John Smith" → "Smith, John"
- "Anita Rao" → "Rao, Anita"

Expected Output

- Well-structured Python function
- Output strictly following example patterns
- Correct handling of names
- Sample inputs and outputs

-Shot Prompting –



The image shows a Visual Studio Code editor window with three tabs: 'Welcome', 'Zero-Shot Prompting – Leap Year Check.py', and 'One-Shot Prompting – Centimeters to Inches Conversion.py'. A fourth tab, 'Few-Shot Prompting – Name Formatting.py', is active and shows a Python script. The script defines a function `format_name` that takes a full name and returns it in 'Last, First' format. It includes example usage and remarks. The terminal at the bottom shows the script being executed, with input 'John Smith' and output 'Smith, John'. A second command 'Rahul' is entered, resulting in an error message: 'The term 'Rahul' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spelling of the name, or if a path was included, verify that the path is correct and try again.'

```
2 #Examples:
3 #John Smith → "Smith, John"
4 #Anita Rao → "Rao, Anita"
5 def format_name(full_name):
6     parts = full_name.split()
7     if len(parts) >= 2:
8         return parts[-1] + ", " + parts[0]
9     else:
10        return "Invalid Name"
11 # Example usage:
12 name_input = input("Enter full name (First Last): ")
13 result = format_name(name_input)
14 print(result)
15 #generate remarks for Few-Shot Prompting - Name Formatting
16 # Remarks:
17 # This function takes a full name as input in the format "First Last" and splits it
18 # into first and last names. It then returns a formatted string in the "Last, First" format.
19 # The example usage demonstrates how to use the function by prompting the user for a full name
20 # and displaying the reformatted result.
```

PS D:\3_2_SEM\AI_assited_coding\Lab_4_3> & C:/Python314/python.exe "d:/3_2_SEM/AI_assited_coding/Lab_4_3/Few-Shot Prompting - Name Formatting.py"

Enter full name (First Last): John Smith

Smith, John

PS D:\3_2_SEM\AI_assited_coding\Lab_4_3> Rahul

Rahul : The term 'Rahul' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spelling of the name, or if a path was included, verify that the path is correct and try again.

At line:1 char:1

+ Rahul

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

Scenario

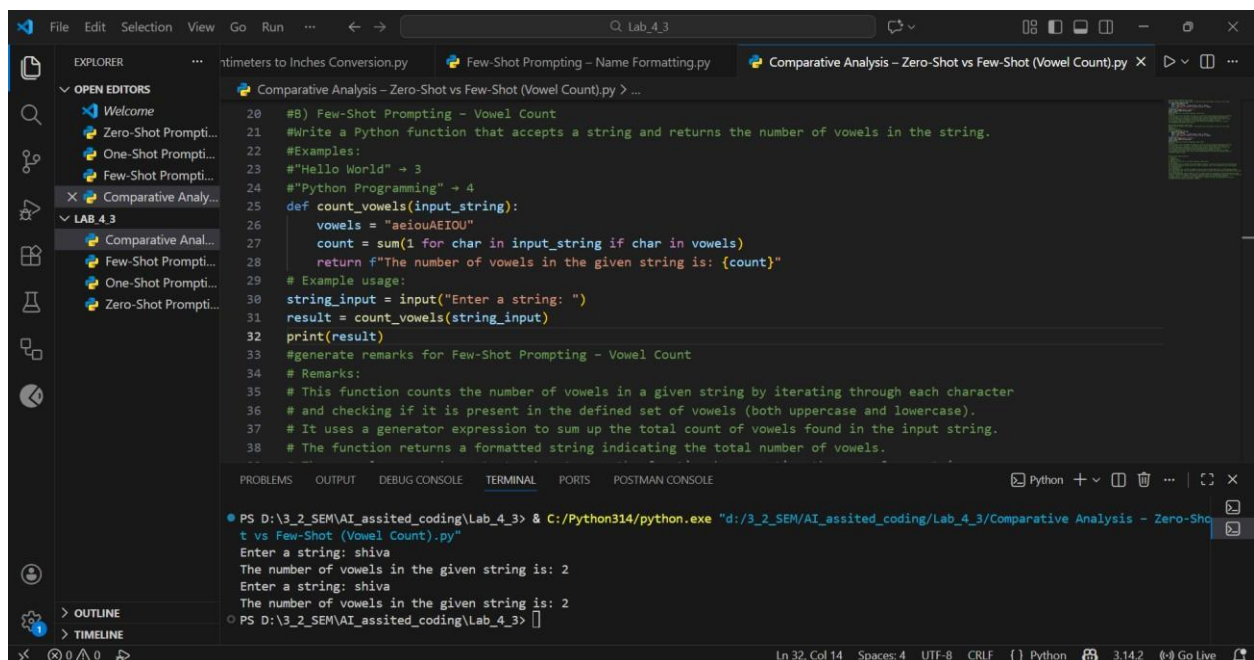
Different prompt strategies may produce different code quality.

Task Description

- Use zero-shot prompting to generate a function that counts vowels in a string
- Use few-shot prompting for the same problem
- Compare both outputs based on:
 - o Accuracy
 - o Readability
 - o Logical clarity

Expected Output

- Two vowel-counting functions
- Comparison table or short reflection paragraph
- Conclusion on prompt effectiveness

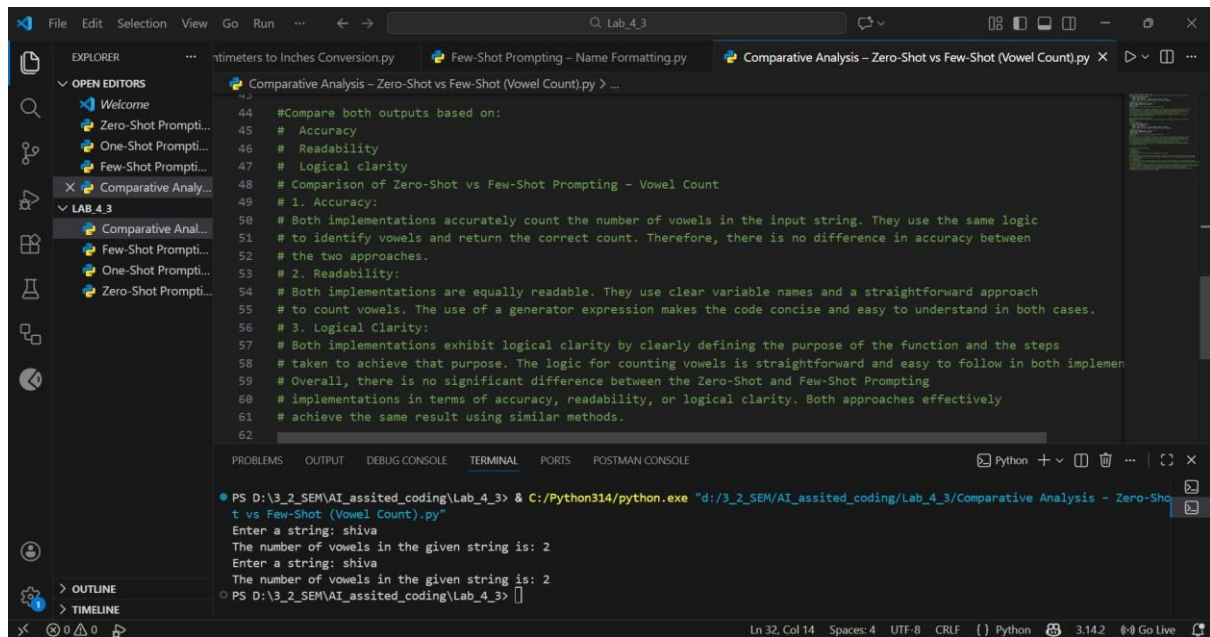


The screenshot shows a Visual Studio Code editor with a file named 'Comparative Analysis - Zero-Shot vs Few-Shot (Vowel Count).py'. The code defines a function 'count_vowels' that takes an input string and returns the number of vowels. It includes comments and an example usage. The terminal output shows the script being executed, and the user entering 'shiva' twice, resulting in the output 'The number of vowels in the given string is: 2'.

```
20 #8) Few-Shot Prompting - Vowel Count
21 #Write a Python function that accepts a string and returns the number of vowels in the string.
22 #Examples:
23 # "Hello World" -> 3
24 # "Python Programming" -> 4
25 def count_vowels(input_string):
26     vowels = "aeiouAEIOU"
27     count = sum(1 for char in input_string if char in vowels)
28     return f"The number of vowels in the given string is: {count}"
29 # Example usage:
30 string_input = input("Enter a string: ")
31 result = count_vowels(string_input)
32 print(result)
33 #generate remarks for Few-Shot Prompting - Vowel Count
34 # Remarks:
35 # This function counts the number of vowels in a given string by iterating through each character
36 # and checking if it is present in the defined set of vowels (both uppercase and lowercase).
37 # It uses a generator expression to sum up the total count of vowels found in the input string.
38 # The function returns a formatted string indicating the total number of vowels.
```

Terminal Output:

```
PS D:\3_2_SEM\AI_assited_coding\Lab_4_3> & C:/Python314/python.exe "d:/3_2_SEM/AI_assited_coding/Lab_4_3/Comparative Analysis - Zero-Shot vs Few-Shot (Vowel Count).py"
Enter a string: shiva
The number of vowels in the given string is: 2
Enter a string: shiva
The number of vowels in the given string is: 2
PS D:\3_2_SEM\AI_assited_coding\Lab_4_3>
```



Task 5: Few-Shot Prompting – File Handling

Scenario

File processing requires clear logical understanding.

Task Description

Use few-shot prompting to generate a Python function that:

- Reads a .txt file
- Counts the number of lines in the file
- Returns the line count

Expected Output

- Working Python file-processing function
- Correct line count
- Sample .txt input and output
- AI-assisted logic explanation

The screenshot shows the Visual Studio Code editor with the file `Few-Shot Prompting - File Handling (Line Count).py` open. The Explorer sidebar on the left shows the project structure with folders `LAB_4_3` and `demo.txt`. The main editor area contains the following Python code:

```
1 #Write a Python function that reads a .txt file and counts the number of lines.
2 # Examples:
3 # File content:
4 # Hello
5 # World
6 # + Output: 2 lines
7 # File content:
8 # Python
9 # AI
10 # Lab
11 # + Output: 3 lines
12 def count_lines_in_file(file_path):
13     try:
14         with open(file_path, 'r') as file:
15             lines = file.readlines()
16             return f"The number of lines in the file is: {len(lines)}"
17     except FileNotFoundError:
18         return "File not found. Please check the file path."
19 # Example usage:
20 file_path_input = input("Enter the path to the .txt file: ")
21 result = count_lines_in_file(file_path_input)
22 print(result)
```

The bottom panel shows the TERMINAL with the following output:

```
PS D:\3_2_SEM\AI_assited_coding\Lab_4_3> C:\Python314\python.exe "d:/3_2_SEM/AI_assited_coding/Lab_4_3/Few-Shot Prompting - File Handling (Line Count).py"
Enter the path to the .txt file: D:\3_2_SEM\AI_assited_coding\Lab_4_3\demo.txt
The number of lines in the file is: 3
PS D:\3_2_SEM\AI_assited_coding\Lab_4_3>
```

The screenshot shows the same Visual Studio Code editor with the file `Few-Shot Prompting - File Handling (Line Count).py` open. The code is now complete with comments and a main execution block. The main editor area contains the following Python code:

```
12 def count_lines_in_file(file_path):
13     lines = file.readlines()
14     return f"The number of lines in the file is: {len(lines)}"
15 except FileNotFoundError:
16     return "File not found. Please check the file path."
17 # Example usage:
18 file_path_input = input("Enter the path to the .txt file: ")
19 result = count_lines_in_file(file_path_input)
20 print(result)
21 #generate remarks for Few-Shot Prompting - File Handling (Line Count)
22 # Remarks:
23 # This function reads a specified .txt file and counts the number of lines it contains.
24 # It uses a try-except block to handle potential file not found errors gracefully.
25 # The function opens the file in read mode, reads all lines into a list, and returns the count of lines.
26 # The example usage demonstrates how to use the function by prompting the user for the file path
27 # and displaying the line count result. The provided examples illustrate the expected output for
28 # different file contents.
```

The bottom panel shows the TERMINAL with the following output:

```
PS D:\3_2_SEM\AI_assited_coding\Lab_4_3> C:\Python314\python.exe "d:/3_2_SEM/AI_assited_coding/Lab_4_3/Few-Shot Prompting - File Handling (Line Count).py"
Enter the path to the .txt file: D:\3_2_SEM\AI_assited_coding\Lab_4_3\demo.txt
The number of lines in the file is: 3
PS D:\3_2_SEM\AI_assited_coding\Lab_4_3>
```