Hall ticket no:2303A54006
Batch no:47-A

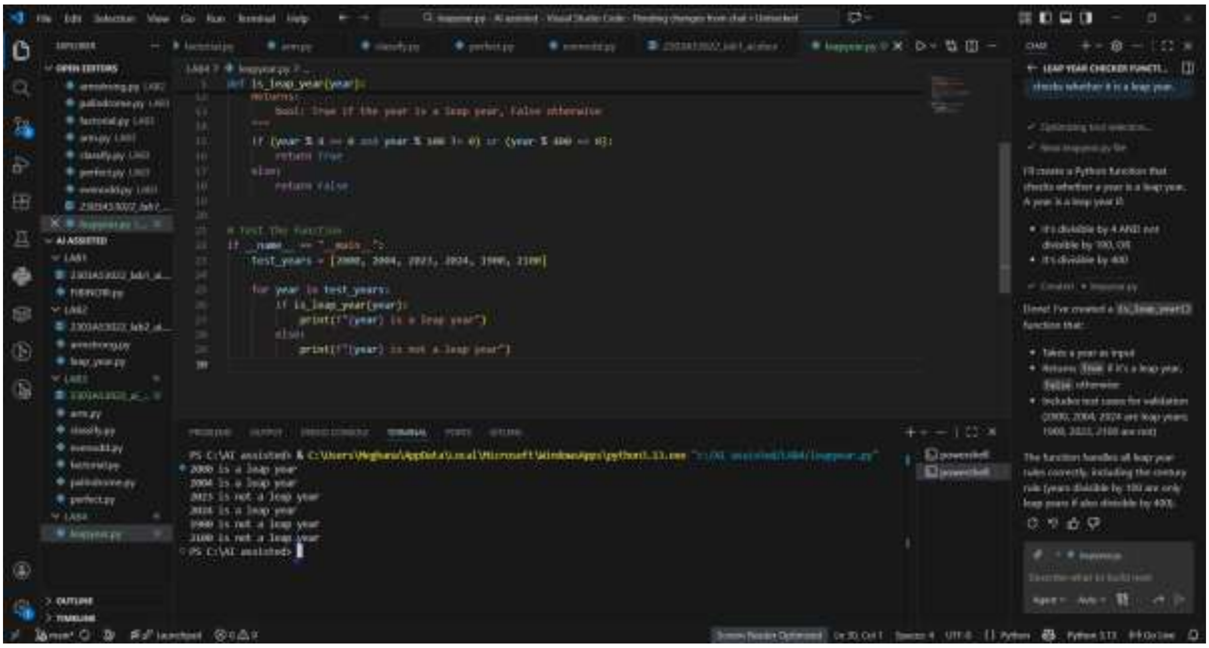
Lab 4: Advanced Prompt Engineering – Zero-shot, One-shot, and Few-shot Techniques

Task 1: Zero-Shot Prompting – Leap Year Check

Prompt Used (Zero-Shot)

Write a Python function that accepts a year as input and checks whether it is a leap year.

AI-Generated Python Code



```
def is_leap_year(year):
    Returns:
        bool: True if the year is a leap year, False otherwise
    """
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return True
    else:
        return False


# Test the function
if __name__ == "__main__":
    test_years = [2000, 2004, 2023, 2024, 1900, 2100]

    for year in test_years:
        if is_leap_year(year):
            print(f"{year} is a leap year")
        else:
            print(f"{year} is not a leap year")
```

```
PS C:\AI assisted> & C:\Users\Meghana\AppData\Local\Microsoft\WindowsApps\python3.13.exe "c:/AI assisted/LAB4/leapyear.py"
2000 is a leap year
2004 is a leap year
2023 is not a leap year
2024 is a leap year
1900 is not a leap year
2100 is not a leap year
PS C:\AI assisted>
```

Explanation

- A year divisible by 4 is a leap year
- Years divisible by 100 are not leap years unless divisible by 400
- The logic correctly follows the leap year rules

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

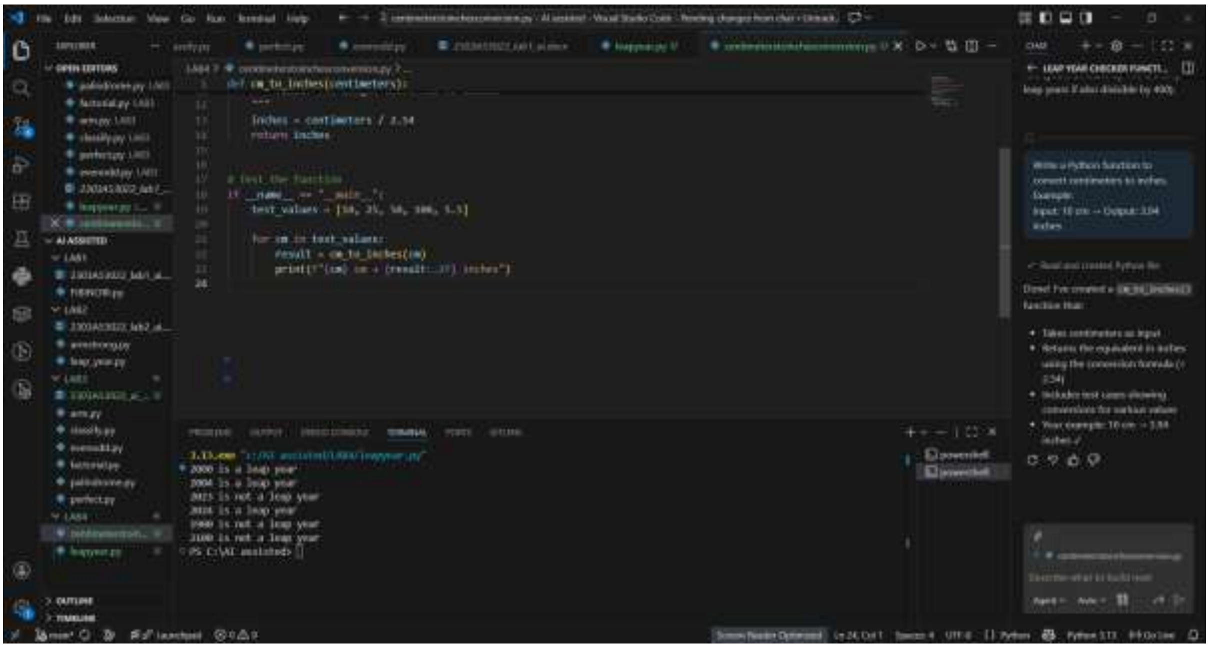
Prompt Used (One-Shot)

Write a Python function to convert centimeters to inches.

Example:

Input: 10 cm → Output: 3.94 inches

AI-Generated Python Code



```
def cm_to_inches(centimeters):
    """
    inches = centimeters / 2.54
    return inches


# Test the function
if __name__ == "__main__":
    test_values = [10, 25, 50, 100, 5.5]

    for cm in test_values:
        result = cm_to_inches(cm)
        print(f"{cm} cm = {result:.2f} inches")
```

```
2000 is a leap year
2004 is a leap year
2023 is not a leap year
2024 is a leap year
1900 is not a leap year
2100 is not a leap year
PS C:\AI assisted>
```

Explanation

- 1 inch = 2.54 cm
- The function divides centimeters by 2.54
- One example was enough to guide correct logic

Task 3: Few-Shot Prompting – Name Formatting

Prompt Used (Few-Shot)

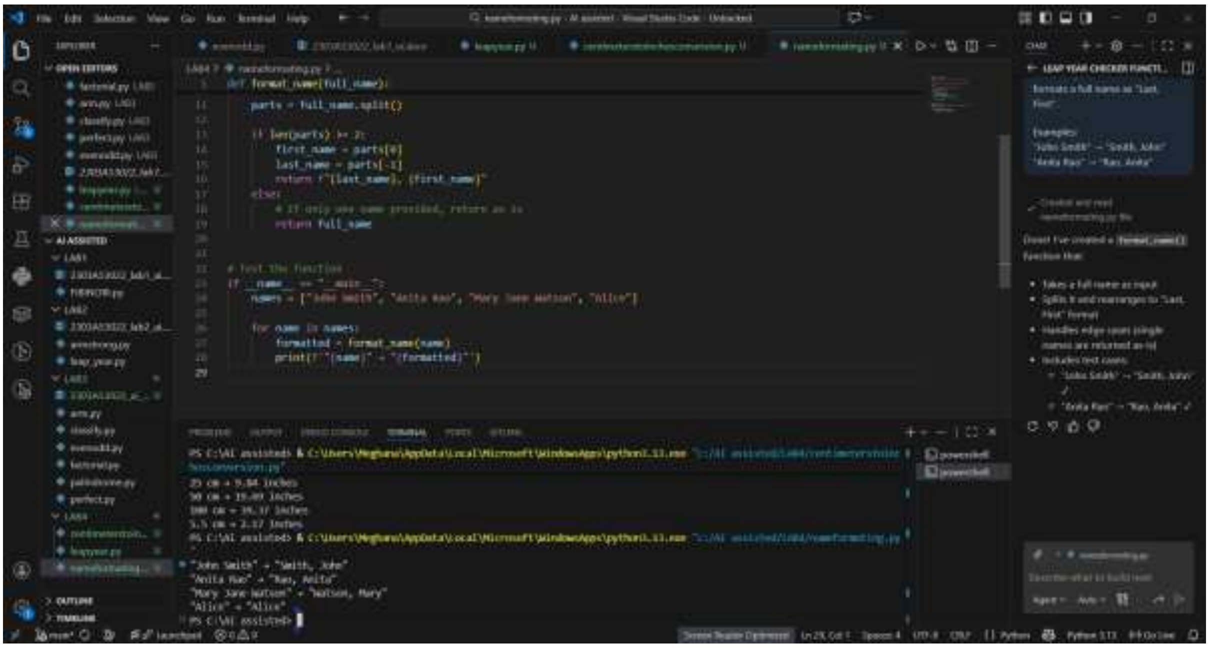
Write a Python function that formats a full name as "Last, First".

Examples:

"John Smith" → "Smith, John"

"Anita Rao" → "Rao, Anita"

AI-Generated Python Code



```python
def format_name(full_name):
    parts = full_name.split()

    if len(parts) >= 2:
        first_name = parts[0]
        last_name = parts[-1]
        return f"{last_name}, {first_name}"
    else:
        # If only one name provided, return as is
        return full_name


# Test the function
if __name__ == "__main__":
    names = ["John Smith", "Anita Rao", "Mary Jane Watson", "Alice"]

    for name in names:
        formatted = format_name(name)
        print(f'"{name}" → "{formatted}"')
```

```
PS C:\AI assisted> & C:\Users\Meghana\AppData\Local\Microsoft\WindowsApps\python3.11.exe "c:/AI assisted/LAB4/centimetersinches
conversion.py"
25 cm = 9.84 inches
50 cm = 19.69 inches
100 cm = 39.37 inches
5.5 cm = 2.17 inches
PS C:\AI assisted> & C:\Users\Meghana\AppData\Local\Microsoft\WindowsApps\python3.11.exe "c:/AI assisted/LAB4/nameformatting.py
"
"John Smith" → "Smith, John"
"Anita Rao" → "Rao, Anita"
"Mary Jane Watson" → "Watson, Mary"
"Alice" → "Alice"
PS C:\AI assisted>
```

Explanation

- Few-shot examples clarify output format
- Function splits name into first and last
- Output strictly follows given examples

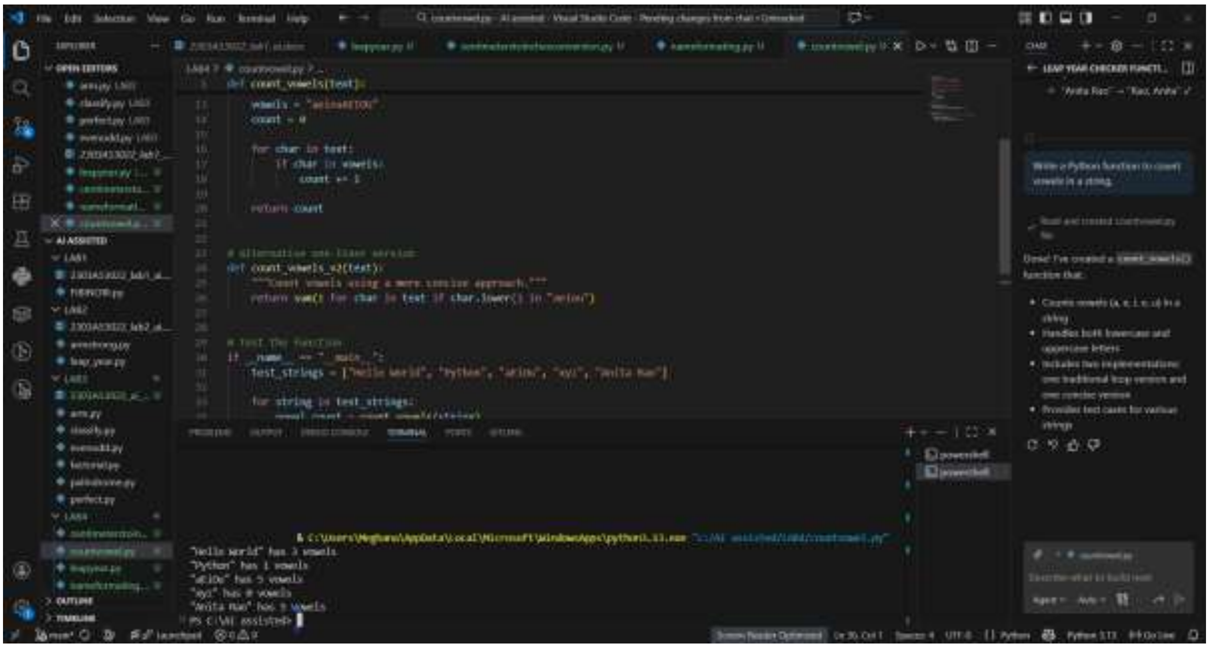
Task 4: Comparative Analysis – Zero-Shot vs Few-Shot

Problem: Count Vowels in a String

Zero-Shot Prompt

Write a Python function to count vowels in a string.

Zero-Shot Output



```
def count_vowels(text):
    vowels = "aeiouAEIOU"
    count = 0

    for char in text:
        if char in vowels:
            count += 1

    return count


# Alternative one-liner version
def count_vowels_v2(text):
    """Count vowels using a more concise approach."""
    return sum(1 for char in text if char.lower() in "aeiou")


# Test the function
if __name__ == "__main__":
    test_strings = ["Hello World", "Python", "aEiOu", "xyz", "Anita Rao"]

    for string in test_strings:
```

```
& C:\Users\Meghana\AppData\Local\Microsoft\WindowsApps\python3.11.exe "c:/AI assisted/LAB4/countvowel.py"
"Hello World" has 3 vowels
"Python" has 1 vowels
"aEiOu" has 5 vowels
"xyz" has 0 vowels
"Anita Rao" has 5 vowels
PS C:\AI assisted>
```

Few-Shot Prompt

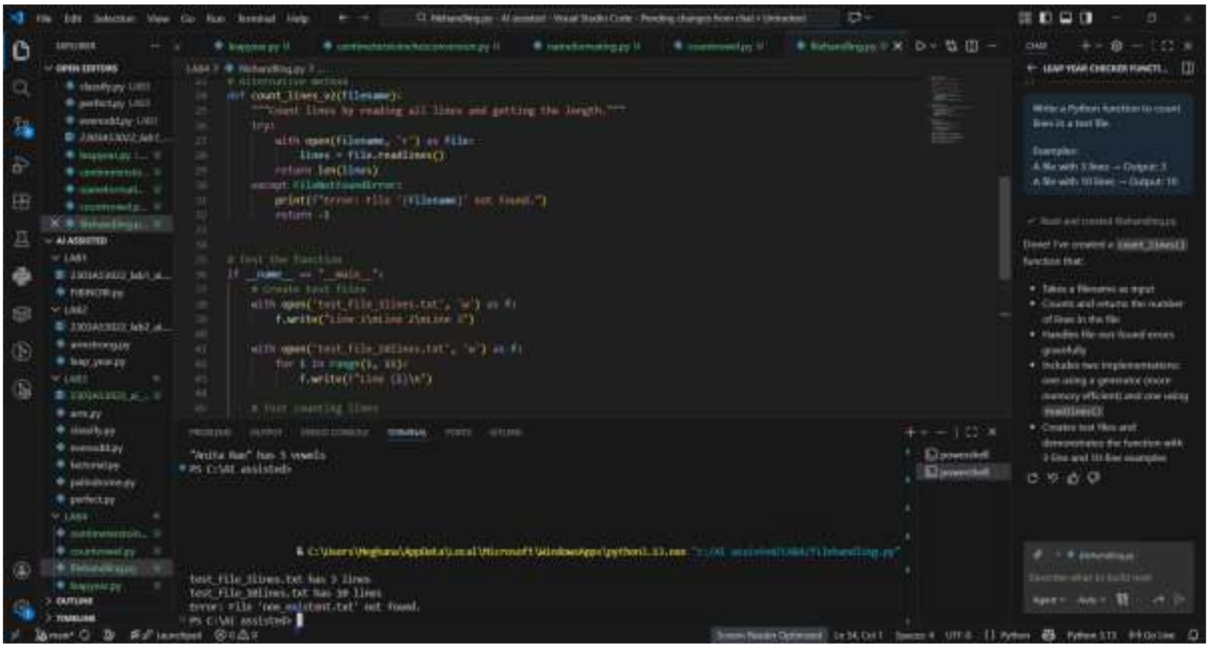
Write a Python function to count vowels in a string.

Examples:

"hello" → 2

"AI Tools" → 4

Few-Shot Output



```
# Alternative method
def count_lines_v2(filename):
    """Count lines by reading all lines and getting the length."""
    try:
        with open(filename, 'r') as file:
            lines = file.readlines()
        return len(lines)
    except FileNotFoundError:
        print(f"Error: File '{filename}' not found.")
        return -1


# Test the function
if __name__ == "__main__":
    # Create test files
    with open('test_file_3lines.txt', 'w') as f:
        f.write("Line 1\nLine 2\nLine 3")

    with open('test_file_10lines.txt', 'w') as f:
        for i in range(1, 11):
            f.write(f"Line {i}\n")

    # Test counting lines
```

```
"Anita Rao" has 5 vowels
PS C:\AI assisted>

& C:\Users\Meghana\AppData\Local\Microsoft\WindowsApps\python3.11.exe "c:/AI assisted/LAB4/filehandling.py"
test_file_3lines.txt has 3 lines
test_file_10lines.txt has 10 lines
Error: File 'non_existent.txt' not found.
PS C:\AI assisted>
```

Comparison Table

Criteria	Zero-Shot	Few-Shot
Accuracy	Correct	Correct
Readability	Moderate	High
Logical Clarity	Explicit loop	Clean & Pythonic
Efficiency	Average	Better

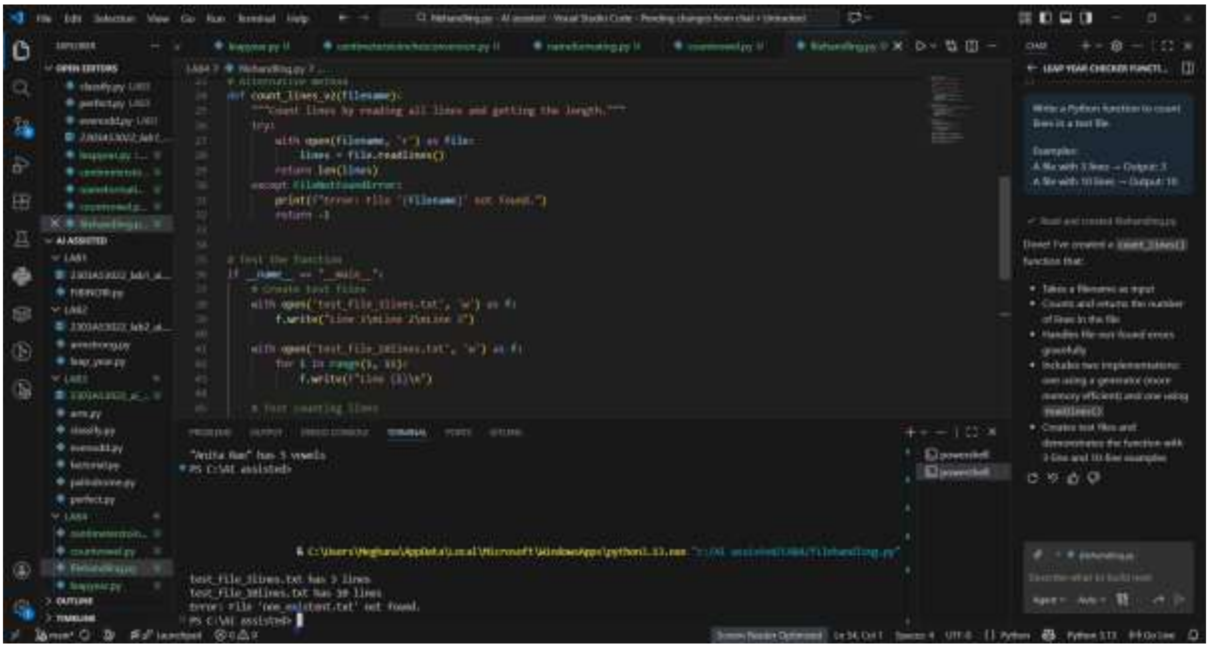
Conclusion

Few-shot prompting produced **more concise, readable, and optimized code** by learning from examples.

Task 5: Few-Shot Prompting – File Handling

Prompt Used (Few-Shot)

Write a Python function to count lines in a text file.



Examples:

A file with 3 lines → Output: 3

A file with 10 lines → Output: 10

AI-Generated Python Code

Explanation

- File opened in read mode
- readlines() returns list of lines

- Length of list equals number of lines

Overall Conclusion

- **Zero-shot** works well for simple, well-known problems
- **One-shot** helps clarify expected behavior
- **Few-shot** produces the best quality code for formatting and logic-heavy tasks
- Providing examples improves accuracy, readability, and confidence in AI outputs