LAB ASSIGNMENT – 4

HALL TICKET NO: 2303A51603

```
Lab 4: Advanced Prompt Engineering Zero-Shot, One-Shot, Few-Shot Techniques
Complete Running Code
```

Task 1: Zero-Shot Prompting – 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
- Checks whether the given year is a leap year
- Returns an appropriate result

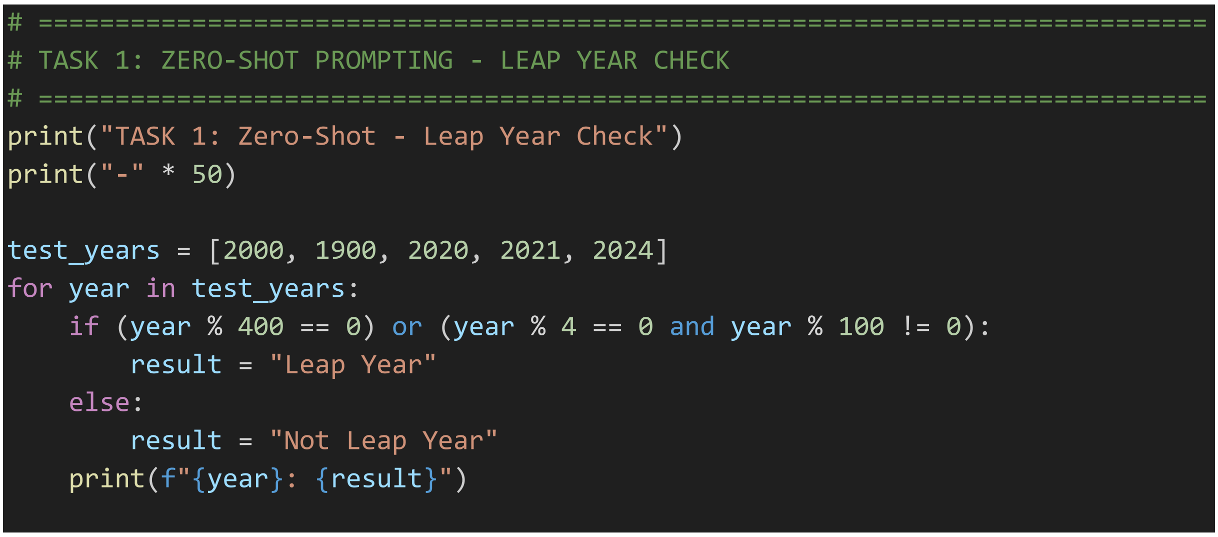
Note: No input-output examples should be provided in the prompt.

Expected Output

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

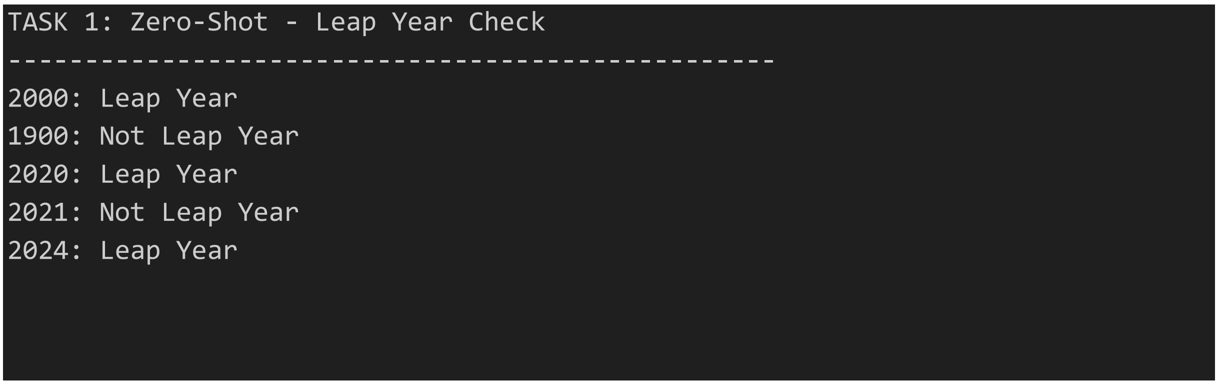
PROMPT:

Write a Python function that accepts a year as input, checks whether the given year is a leap year, and returns the result. Also show a sample input and output.



```python
# =============================================================================
# TASK 1: ZERO-SHOT PROMPTING - LEAP YEAR CHECK
# =============================================================================
print("TASK 1: Zero-Shot - Leap Year Check")
print("-" * 50)

test_years = [2000, 1900, 2020, 2021, 2024]
for year in test_years:
    if (year % 400 == 0) or (year % 4 == 0 and year % 100 != 0):
        result = "Leap Year"
    else:
        result = "Not Leap Year"
    print(f"{year}: {result}")
```

Output:



```
TASK 1: Zero-Shot - Leap Year Check
--------------------------------------------------
2000: Leap Year
1900: Not Leap Year
2020: Leap Year
2021: Not Leap Year
2024: Leap Year
```

# EXPLANATION:

# Zero-shot prompting requires AI to understand leap year logic from description alone

# A leap year is:

#   - Divisible by 400 (e.g., 2000), OR

#   - Divisible by 4 AND NOT divisible by 100 (e.g., 2020)

```
# CODE LOGIC:
# 1. Test multiple years against leap year rules
# 2. Use modulo operator (%) to check divisibility
# 3. Apply Boolean logic: (year % 400 == 0) or (year % 4 == 0 and year % 100 != 0)
```

Task 2: One-Shot Prompting – 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

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

PROMPT:

Write a Python function to convert centimeters to inches.

Example:

Input: 10 cm

Output: 3.94 inches

```
# ============================================================================
# TASK 2: ONE-SHOT PROMPTING - CM TO INCHES CONVERSION
# ============================================================================
print("\n\nTASK 2: One-Shot - CM to Inches Conversion")
print("-" * 50)

test_cm = [10, 25, 50, 100, 5]
for centimeters in test_cm:
    inches = centimeters * 0.3937
    inches = round(inches, 2)
    print(f"{centimeters} cm = {inches} inches")
```

Output:

```
TASK 2: One-Shot - CM to Inches Conversion
--------------------------------------------------
10 cm = 3.94 inches
25 cm = 9.84 inches
50 cm = 19.68 inches
100 cm = 39.37 inches
5 cm = 1.97 inches
```

# EXPLANATION:

# One-shot prompting provides a single input-output example

# Example from prompt: 10 cm = 3.94 inches

# This clarifies:

#   - The conversion formula to use (multiply by 0.3937)

#   - The precision needed (2 decimal places)

# CODE LOGIC:

# 1. For each test value in centimeters

# 2. Multiply by conversion factor 0.3937 (or divide by 2.54)

# 3. Round result to 2 decimal places

# 4. Display in format matching the example

Task 3: Few-Shot Prompting – 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

PROMPT:

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

Examples:

"John Smith" → "Smith, John"

"Anita Rao" → "Rao, Anita"

```
# ============================================================================
# TASK 3: FEW-SHOT PROMPTING - NAME FORMATTING
# ============================================================================
print("\n\nTASK 3: Few-Shot - Name Formatting")
print("-" * 50)

test_names = ["John Smith", "Anita Rao", "Mary Jane Watson", "Alice Johnson"]
for full_name in test_names:
    parts = full_name.strip().split()
    last_name = parts[-1]
    first_names = ' '.join(parts[:-1])
    formatted = f"{last_name}, {first_names}"
    print(f"'{full_name}' -> '{formatted}'")
```

Output:

```
TASK 3: Few-Shot - Name Formatting
--------------------------------------------------
'John Smith' -> 'Smith, John'
'Anita Rao' -> 'Rao, Anita'
'Mary Jane Watson' -> 'Watson, Mary Jane'
'Alice Johnson' -> 'Johnson, Alice'
```

# EXPLANATION:

# Few-shot prompting provides 2-3 examples to clarify exact format requirement

# Examples from prompt:

#   - 'John Smith' -> 'Smith, John'

#   - 'Anita Rao' -> 'Rao, Anita'

#   - 'Mary Jane Watson' -> 'Watson, Mary Jane'

#

# From these examples, the pattern is:

#   - Last word = surname

#   - All other words = first/middle names

#   - Format: "LastName, FirstNames"

```
# CODE LOGIC:

# 1. Split full name into word list

# 2. Extract last element as surname

# 3. Join all other elements as first/middle names

# 4. Return in "Last, First" format
```

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

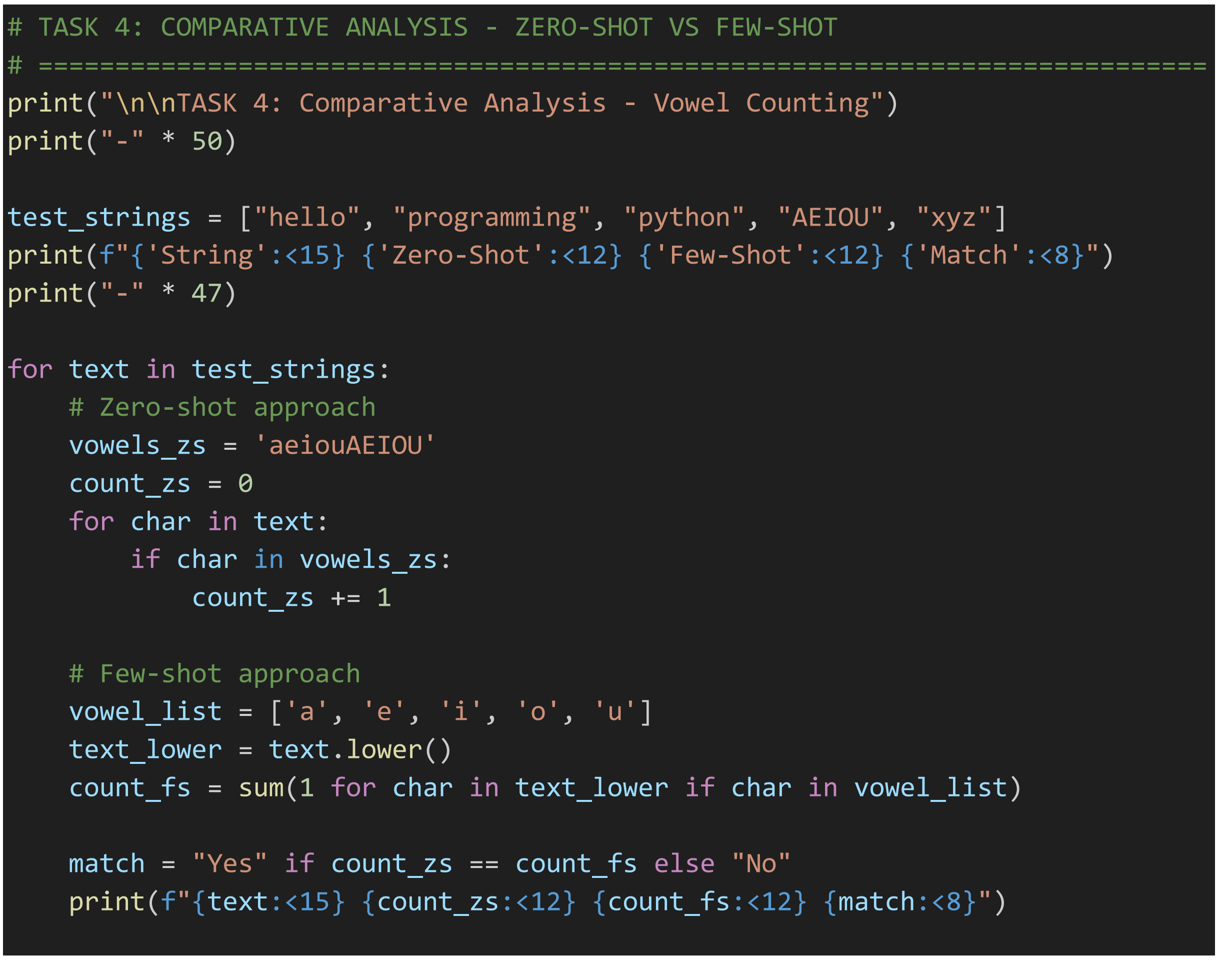
• Comparison table or short reflection paragraph

• Conclusion on prompt effectiveness

PROMPT:

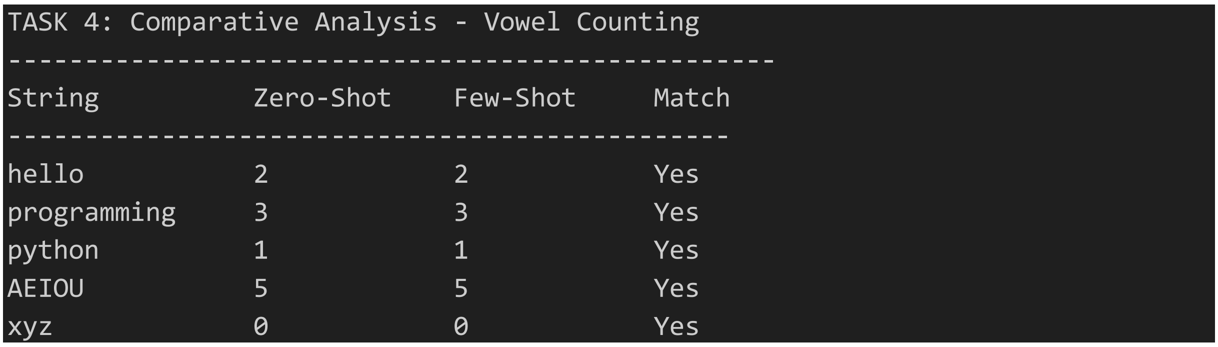
Write a Python function that counts the number of vowels in a given string.

```
# ============================================================================
```



```python
# TASK 4: COMPARATIVE ANALYSIS - ZERO-SHOT VS FEW-SHOT
# ============================================================================
print("\n\nTASK 4: Comparative Analysis - Vowel Counting")
print("-" * 50)

test_strings = ["hello", "programming", "python", "AEIOU", "xyz"]
print(f"{'String':<15} {'Zero-Shot':<12} {'Few-Shot':<12} {'Match':<8}")
print("-" * 47)

for text in test_strings:
    # Zero-shot approach
    vowels_zs = 'aeiouAEIOU'
    count_zs = 0
    for char in text:
        if char in vowels_zs:
            count_zs += 1

    # Few-shot approach
    vowel_list = ['a', 'e', 'i', 'o', 'u']
    text_lower = text.lower()
    count_fs = sum(1 for char in text_lower if char in vowel_list)

    match = "Yes" if count_zs == count_fs else "No"
    print(f"{text:<15} {count_zs:<12} {count_fs:<12} {match:<8}")
```

Output:



```
TASK 4: Comparative Analysis - Vowel Counting
--------------------------------------------------
String          Zero-Shot    Few-Shot     Match
-----------------------------------------------
hello           2            2            Yes
programming     3            3            Yes
python          1            1            Yes
AEIOU           5            5            Yes
xyz             0            0            Yes
```

# EXPLANATION:

# Same problem (count vowels), two different prompting strategies

#

# ZERO-SHOT: "Generate function to count vowels in a string"

#   - AI infers what vowels are (a, e, i, o, u)

#   - Uses explicit loop approach

```
#
# FEW-SHOT: "Count vowels. Example: 'hello' = 2 (e, o)"
#   - Examples clarify what to count
#   - Uses more Pythonic comprehension approach

# CODE LOGIC:
# 1. For each test string, apply both approaches
# 2. Zero-shot: explicit loop through characters
# 3. Few-shot: Pythonic generator expression with sum()
# 4. Compare results to verify correctness
```

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

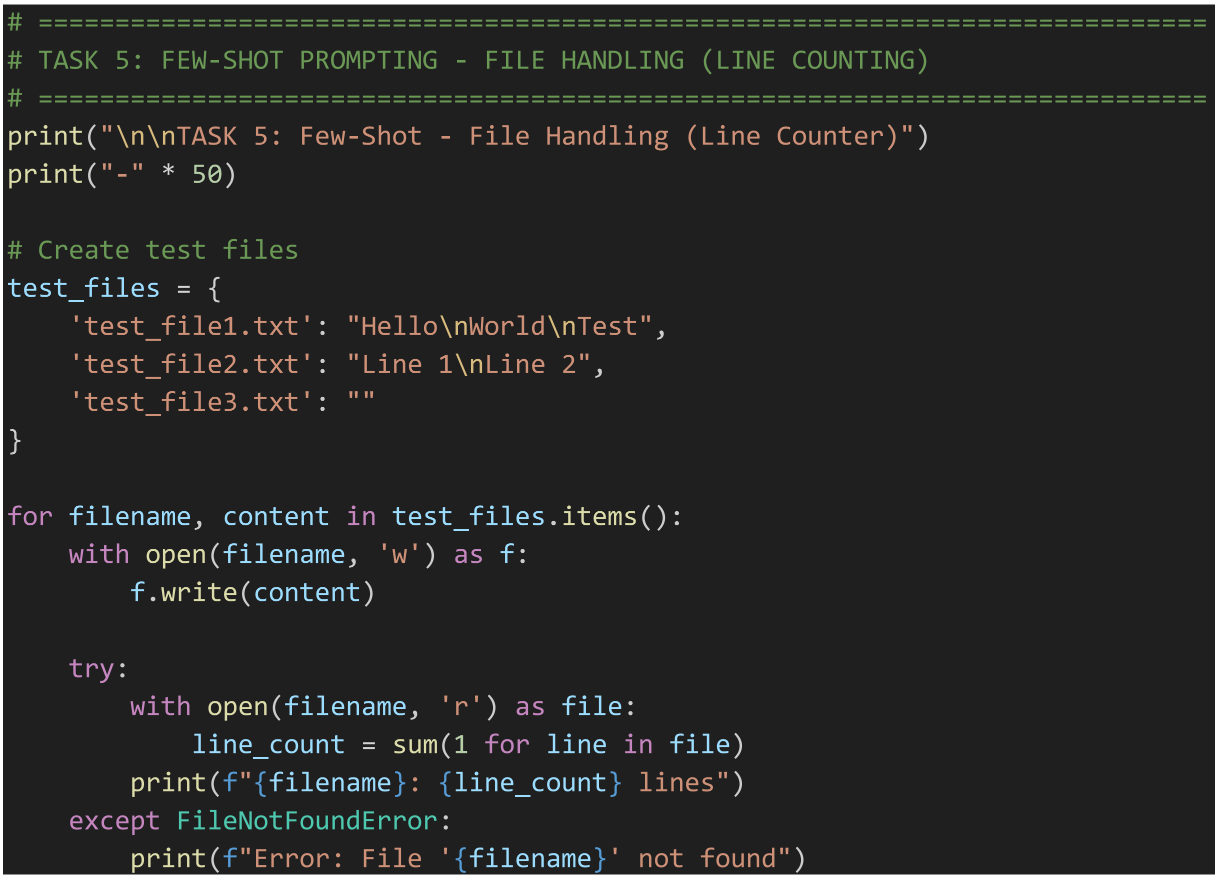
PROMPT:

Write a Python function that reads a .txt file and counts the number of lines.

Examples:

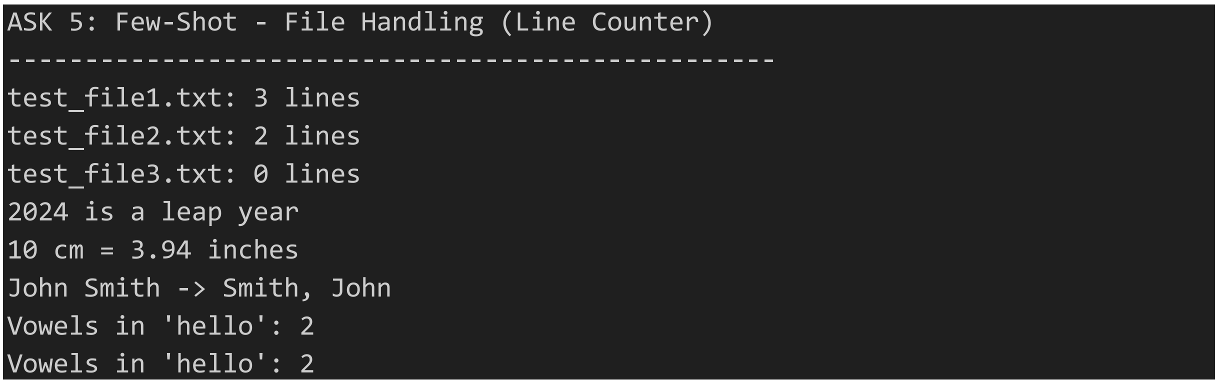
File with 3 lines → Output: 3

File with 10 lines → Output: 10

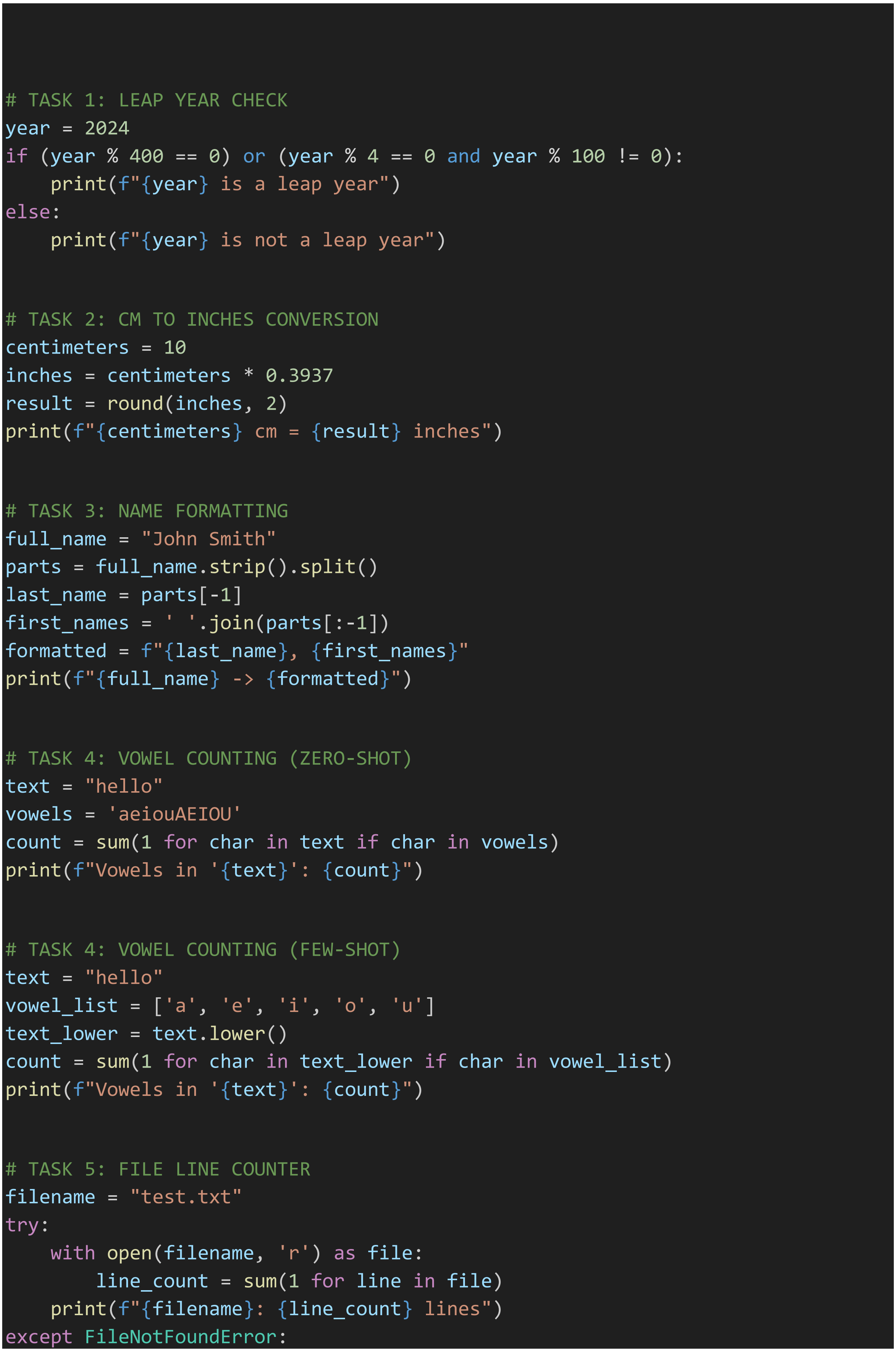


```
# ============================================================================
# TASK 5: FEW-SHOT PROMPTING - FILE HANDLING (LINE COUNTING)
# ============================================================================
print("\n\nTASK 5: Few-Shot - File Handling (Line Counter)")
print("-" * 50)

# Create test files
test_files = {
    'test_file1.txt': "Hello\nWorld\nTest",
    'test_file2.txt': "Line 1\nLine 2",
    'test_file3.txt': ""
}

for filename, content in test_files.items():
    with open(filename, 'w') as f:
        f.write(content)

    try:
        with open(filename, 'r') as file:
            line_count = sum(1 for line in file)
        print(f"{filename}: {line_count} lines")
    except FileNotFoundError:
        print(f"Error: File '{filename}' not found")
```

Output:



```
ASK 5: Few-Shot - File Handling (Line Counter)
--------------------------------------------------
test_file1.txt: 3 lines
test_file2.txt: 2 lines
test_file3.txt: 0 lines
2024 is a leap year
10 cm = 3.94 inches
John Smith -> Smith, John
Vowels in 'hello': 2
Vowels in 'hello': 2
```



```python
# TASK 1: LEAP YEAR CHECK
year = 2024
if (year % 400 == 0) or (year % 4 == 0 and year % 100 != 0):
    print(f"{year} is a leap year")
else:
    print(f"{year} is not a leap year")


# TASK 2: CM TO INCHES CONVERSION
centimeters = 10
inches = centimeters * 0.3937
result = round(inches, 2)
print(f"{centimeters} cm = {result} inches")


# TASK 3: NAME FORMATTING
full_name = "John Smith"
parts = full_name.strip().split()
last_name = parts[-1]
first_names = ' '.join(parts[:-1])
formatted = f"{last_name}, {first_names}"
print(f"{full_name} -> {formatted}")


# TASK 4: VOWEL COUNTING (ZERO-SHOT)
text = "hello"
vowels = 'aeiouAEIOU'
count = sum(1 for char in text if char in vowels)
print(f"Vowels in '{text}': {count}")


# TASK 4: VOWEL COUNTING (FEW-SHOT)
text = "hello"
vowel_list = ['a', 'e', 'i', 'o', 'u']
text_lower = text.lower()
count = sum(1 for char in text_lower if char in vowel_list)
print(f"Vowels in '{text}': {count}")


# TASK 5: FILE LINE COUNTER
filename = "test.txt"
try:
    with open(filename, 'r') as file:
        line_count = sum(1 for line in file)
    print(f"{filename}: {line_count} lines")
except FileNotFoundError:
    print(f"File not found: {filename}")
```

```
# EXPLANATION:
# Few-shot prompting with file operations
# Examples show:
#   - File with content 'Hello\nWorld\nTest' = 3 lines
#   - File with content 'Line 1\nLine 2' = 2 lines
#   - Empty file '' = 0 lines
#
# This clarifies edge cases and expected behavior

# CODE LOGIC:
# 1. Create test files with different line counts
# 2. Open each file and count lines
# 3. Use generator expression for efficiency: sum(1 for line in file)
# 4. Handle FileNotFoundError exception
# 5. Return line count (or -1 for error)
```