Task Description#5
- Use few-shot prompting to generate a function that reads a .txt file and returns the number of lines.

Expected Output#5
- Working file-processing function with AI-guided logic

```python
def count_lines_in_file(filename):
    """Reads a text file and returns the number of lines.

    Args:
        filename: The path to the text file.

    Returns:
        The number of lines in the file.
    """
    with open(filename, 'r') as f:
        return len(f.readlines())

# Few-shot example
# file: /content/sample.txt
# output: 10
```

Task Description#4
- Compare zero-shot and few-shot prompts for writing a function that counts the number of vowels in a string.

Expected Output#4
- Functional output and comparative reflection

```python
def count_vowels(text):
    """Counts the number of vowels in a string."""
    vowels = "aeiouAEIOU"
    vowel_count = 0
    for char in text:
        if char in vowels:
            vowel_count += 1
    return vowel_count

# Example usage
text = "Hello World"
print(f"The number of vowels in '{text}' is: {count_vowels(text)}")

text = "Artificial Intelligence"
print(f"The number of vowels in '{text}' is: {count_vowels(text)}")
```

```
The number of vowels in 'Hello World' is: 3
The number of vowels in 'Artificial Intelligence' is: 10
```

Task Description#3
- Few-shot: Provide 2–3 examples to generate a function that formats full names as "Last, First".

Expected Output#3
- Well-structured function respecting the examples

```python
def format_name(full_name):
    """Formats a full name as 'Last, First'.

    Args:
        full_name: A string containing the full name (e.g., "John Doe").

    Returns:
        A string with the name formatted as 'Last, First' (e.g., "Doe, John").
    """
    parts = full_name.split()
    if len(parts) > 1:
        return f"{parts[-1]}, {' '.join(parts[:-1])}"
    else:
        return full_name  # Handle cases with single names or empty strings

# Examples
print(format_name("John Doe"))
print(format_name("Jane Smith"))
print(format_name("Peter Jones"))
```

```
Doe, John
Smith, Jane
```

Task Description#2
- One-shot: Give one input-output example to guide AI in writing a function that converts centimeters to inches.

Expected Output#2
- Function with correct conversion logic

```python
def cm_to_inches(cm):
    """Converts centimeters to inches."""
    # 1 inch = 2.54 centimeters
    inches = cm / 2.54
    return inches

# One-shot example
print(cm_to_inches(10) == 3.937007874015748)
```

```
True
```

Task Description#1
- Zero-shot: Prompt AI to write a function that checks whether a given year is a leap year.

Expected Output#1
- AI-generated function with no examples provided

```python
year_to_check = 2024
if is_leap(year_to_check):
    print(f"{year_to_check} is a leap year.")
else:
    print(f"{year_to_check} is not a leap year.")

year_to_check = 1900
if is_leap(year_to_check):
    print(f"{year_to_check} is a leap year.")
else:
    print(f"{year_to_check} is not a leap year.")

year_to_check = 2000
if is_leap(year_to_check):
    print(f"{year_to_check} is a leap year.")
else:
    print(f"{year_to_check} is not a leap year.")
```

```
2024 is a leap year.
1900 is not a leap year.
2000 is a leap year.
```

Task Description#1
- Zero-shot: Prompt AI to write a function that checks whether a given year is a leap year.

Expected Output#1
- AI-generated function with no examples provided

```python
year_to_check = 2024
if is_leap(year_to_check):
    print(f"{year_to_check} is a leap year.")
else:
    print(f"{year_to_check} is not a leap year.")

year_to_check = 1900
if is_leap(year_to_check):
    print(f"{year_to_check} is a leap year.")
else:
    print(f"{year_to_check} is not a leap year.")

year_to_check = 2000
if is_leap(year_to_check):
    print(f"{year_to_check} is a leap year.")
else:
    print(f"{year_to_check} is not a leap year.")
```

```
2024 is a leap year.
1900 is not a leap year.
2000 is a leap year.
```