Assignment 4.3

Name: P. Umesh Reddy   Roll.no: 2403A510F9

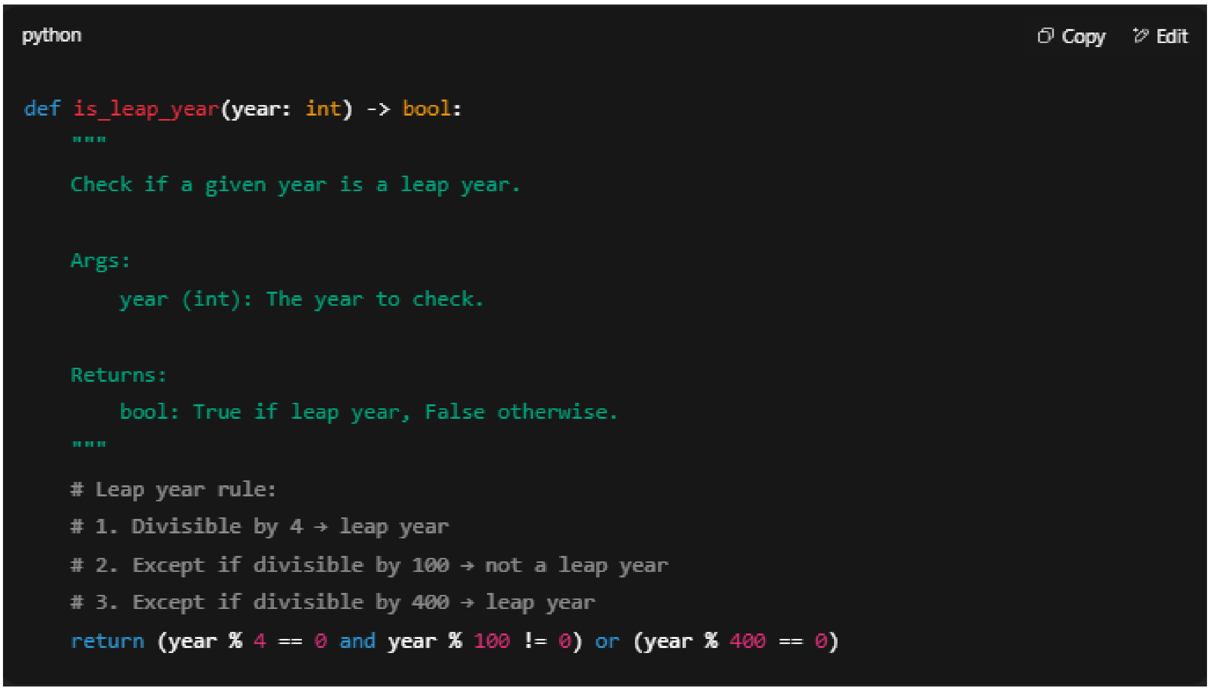
Batch.no: 06   course: AI Assisted coding

Task Description#1

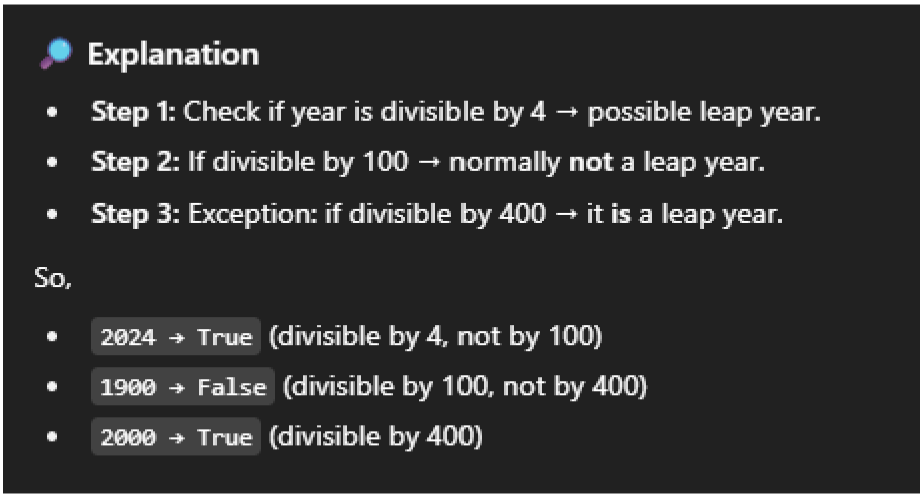
- Zero-shot: Prompt AI to write a function that checks whether a given year is a leap year.

Expected Output#1

- AI-generated function with no examples provided



```python
def is_leap_year(year: int) -> bool:
    """
    Check if a given year is a leap year.

    Args:
        year (int): The year to check.

    Returns:
        bool: True if leap year, False otherwise.
    """
    # Leap year rule:
    # 1. Divisible by 4 → leap year
    # 2. Except if divisible by 100 → not a leap year
    # 3. Except if divisible by 400 → leap year
    return (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0)
```



Explanation

- **Step 1:** Check if year is divisible by 4 → possible leap year.
- **Step 2:** If divisible by 100 → normally **not** a leap year.
- **Step 3:** Exception: if divisible by 400 → it **is** a leap year.

So,

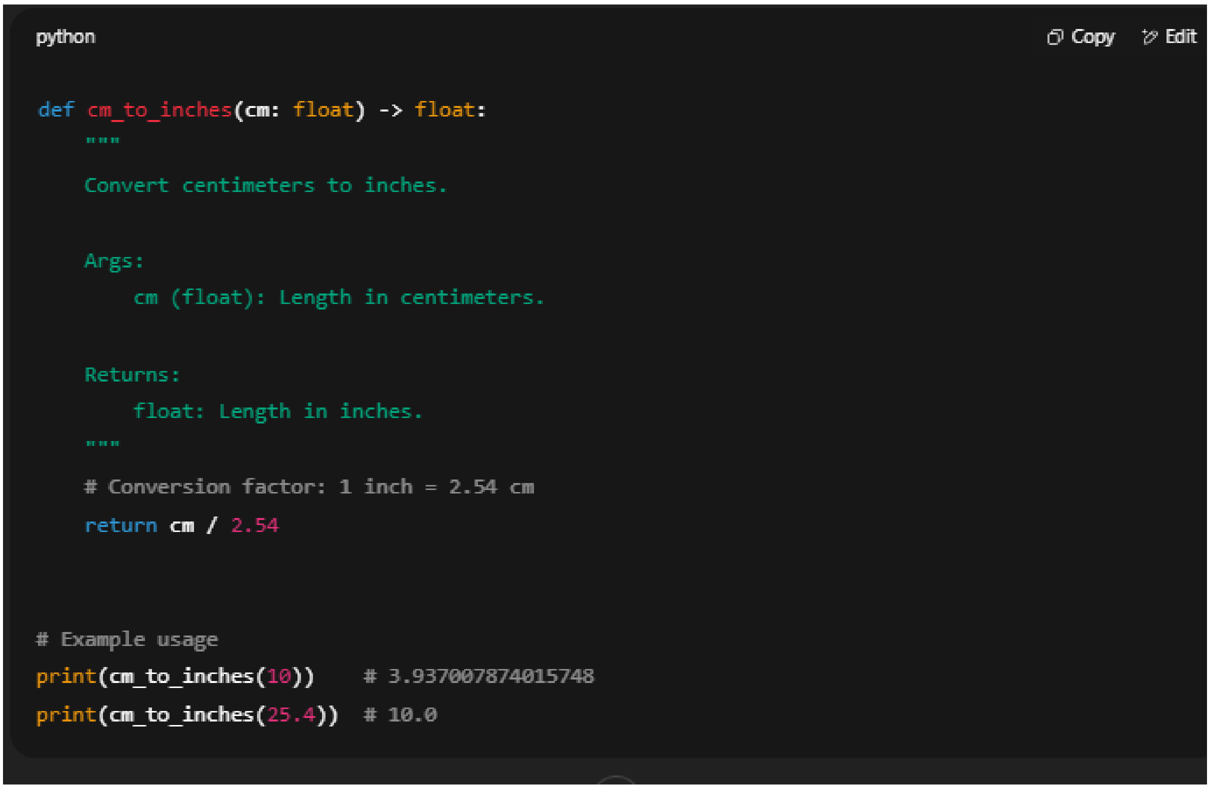
- `2024 → True` (divisible by 4, not by 100)
- `1900 → False` (divisible by 100, not by 400)
- `2000 → True` (divisible by 400)

Task Description#2

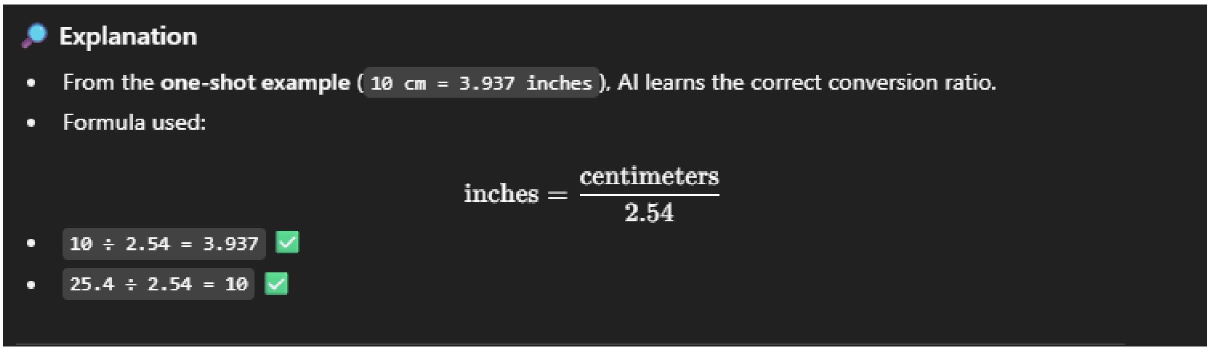
- One-shot: Give one input-output example to guide AI in writing a function that converts centimeters to inches.

Expected Output#2

- Function with correct conversion logic



```python
def cm_to_inches(cm: float) -> float:
    """
    Convert centimeters to inches.

    Args:
        cm (float): Length in centimeters.

    Returns:
        float: Length in inches.
    """
    # Conversion factor: 1 inch = 2.54 cm
    return cm / 2.54


# Example usage
print(cm_to_inches(10))    # 3.937007874015748
print(cm_to_inches(25.4))  # 10.0
```



Explanation

- From the **one-shot example** (`10 cm = 3.937 inches`), AI learns the correct conversion ratio.
- Formula used:

$$\text{inches} = \frac{\text{centimeters}}{2.54}$$

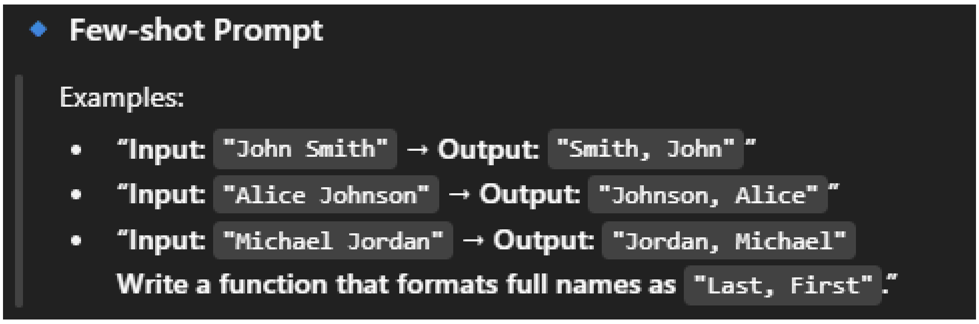
- `10 ÷ 2.54 = 3.937` ✅
- `25.4 ÷ 2.54 = 10` ✅

Task Description#3

- **Few-shot: Provide 2–3 examples to generate a function that formats full names as “Last, First”.**

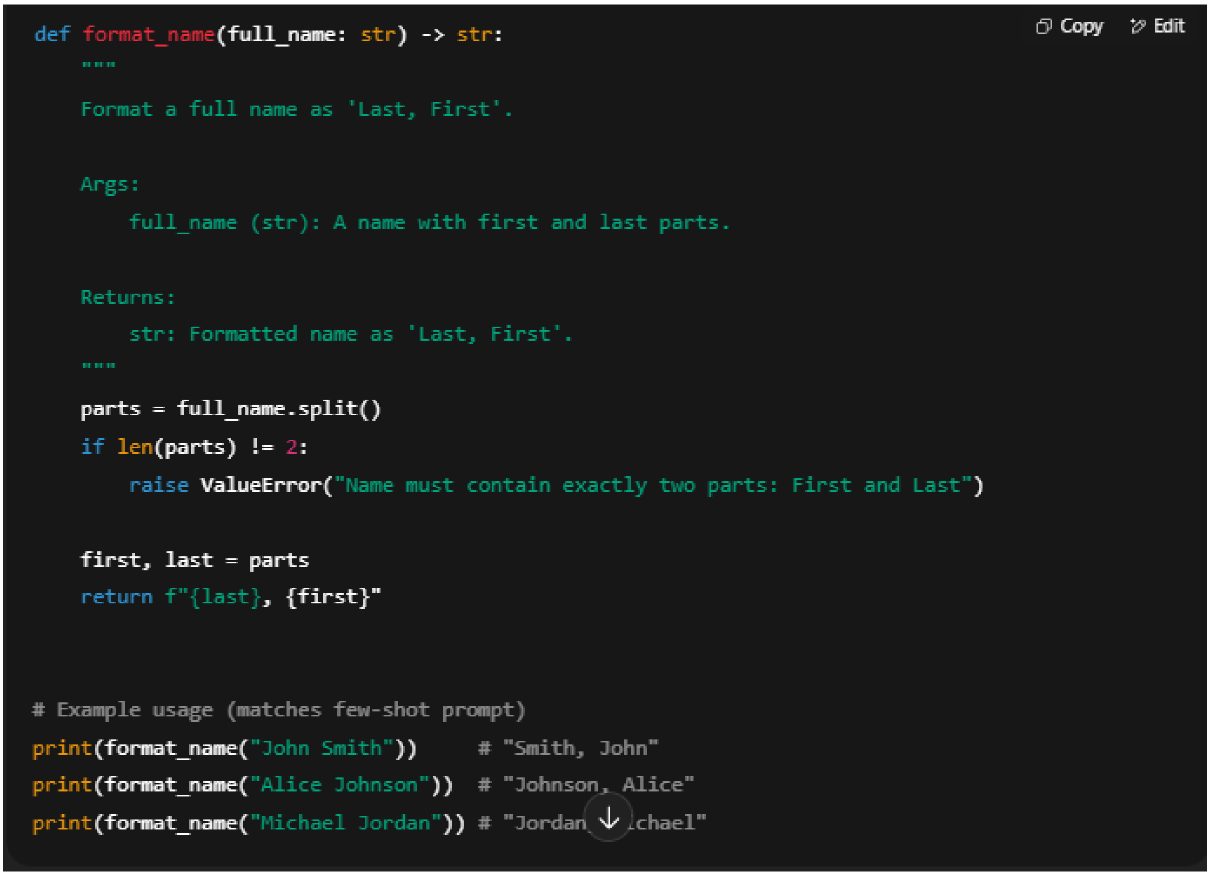
Expected Output#3

- **Well-structured function respecting the examples**

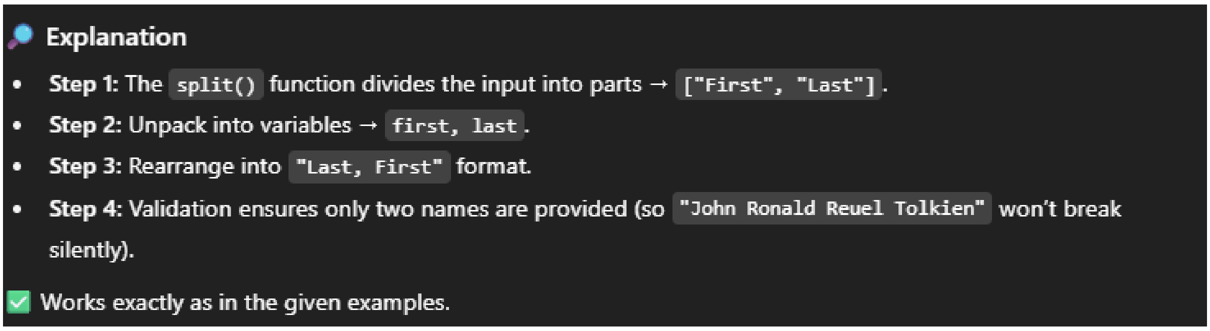


Few-shot Prompt

> Examples:
>
> - “Input: `"John Smith"` → Output: `"Smith, John"` ”
> - “Input: `"Alice Johnson"` → Output: `"Johnson, Alice"` ”
> - “Input: `"Michael Jordan"` → Output: `"Jordan, Michael"`
>
> Write a function that formats full names as `"Last, First"`.”



```python
def format_name(full_name: str) -> str:
    """
    Format a full name as 'Last, First'.

    Args:
        full_name (str): A name with first and last parts.

    Returns:
        str: Formatted name as 'Last, First'.
    """
    parts = full_name.split()
    if len(parts) != 2:
        raise ValueError("Name must contain exactly two parts: First and Last")

    first, last = parts
    return f"{last}, {first}"


# Example usage (matches few-shot prompt)
print(format_name("John Smith"))     # "Smith, John"
print(format_name("Alice Johnson"))  # "Johnson, Alice"
print(format_name("Michael Jordan")) # "Jordan, ↓chael"
```



Explanation

- **Step 1:** The `split()` function divides the input into parts → `["First", "Last"]`.
- **Step 2:** Unpack into variables → `first, last`.
- **Step 3:** Rearrange into `"Last, First"` format.
- **Step 4:** Validation ensures only two names are provided (so `"John Ronald Reuel Tolkien"` won't break silently).

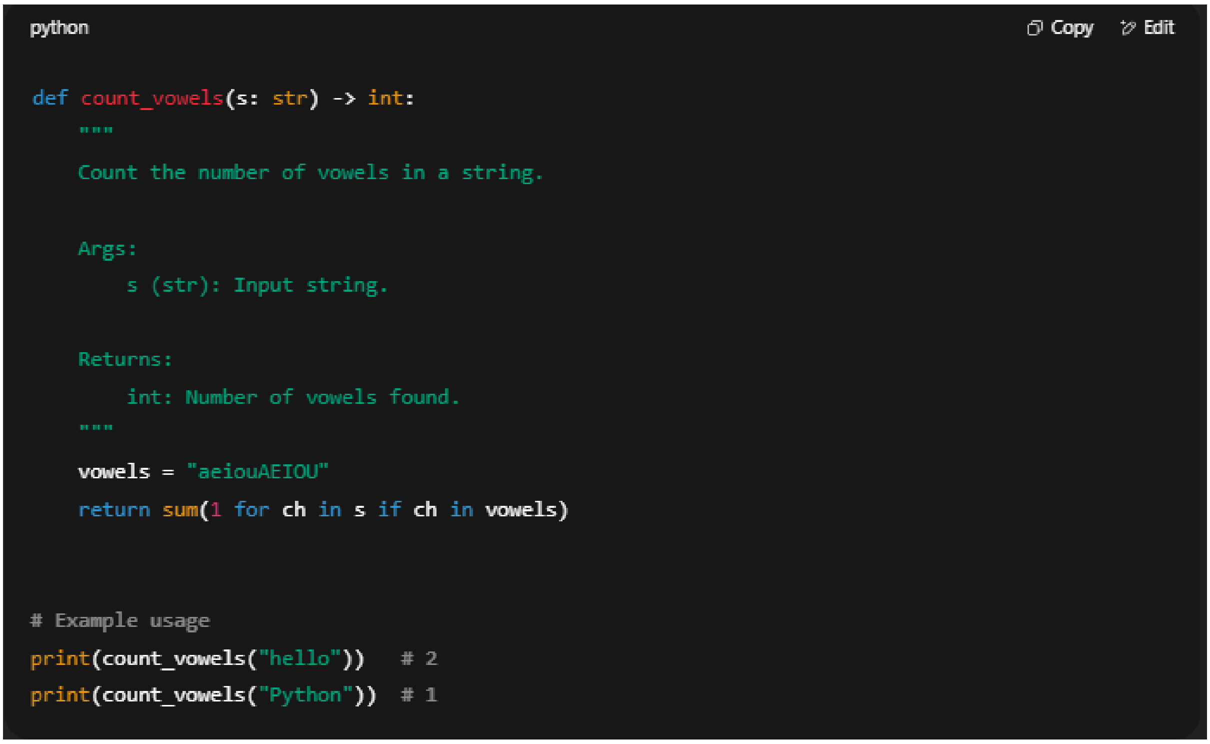
✅ Works exactly as in the given examples.

Task Description#4

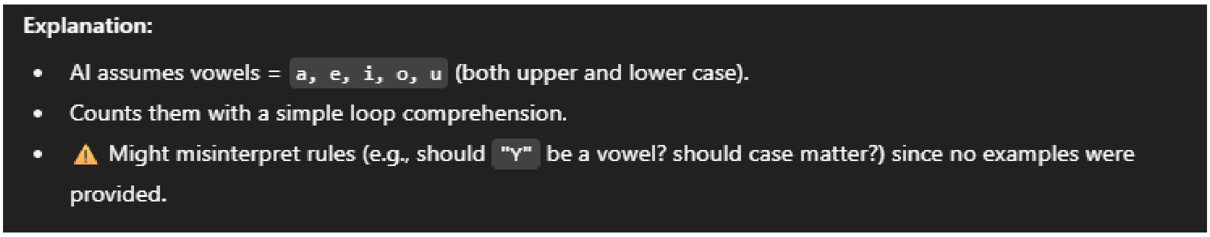
- Compare zero-shot and few-shot prompts for writing a function that counts the number of vowels in a string.

Expected Output#4

- Functional output and comparative reflection

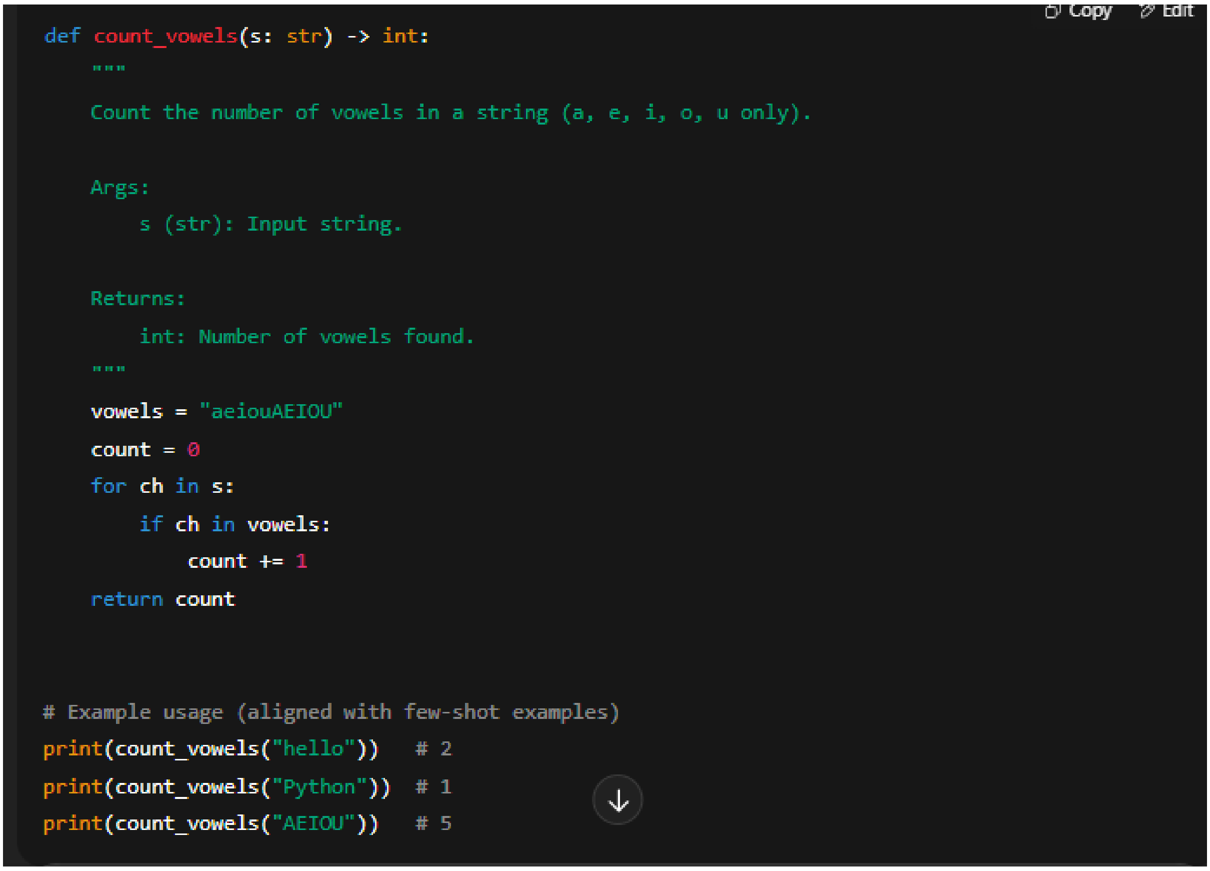


```python
def count_vowels(s: str) -> int:
    """
    Count the number of vowels in a string.

    Args:
        s (str): Input string.

    Returns:
        int: Number of vowels found.
    """
    vowels = "aeiouAEIOU"
    return sum(1 for ch in s if ch in vowels)


# Example usage
print(count_vowels("hello"))   # 2
print(count_vowels("Python"))  # 1
```

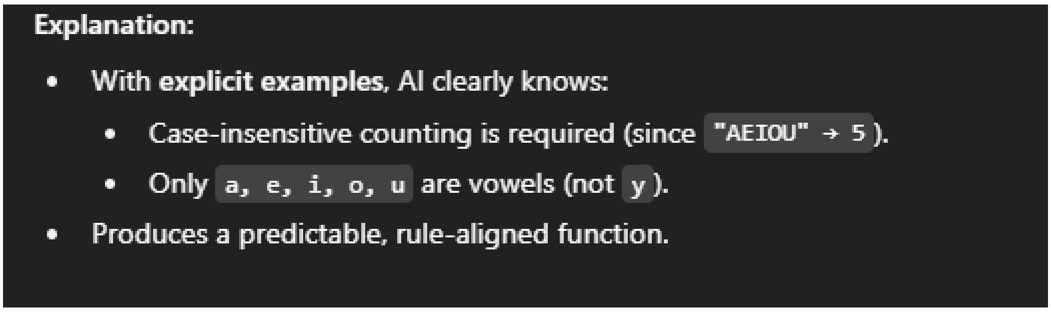


Explanation:

- AI assumes vowels = `a, e, i, o, u` (both upper and lower case).
- Counts them with a simple loop comprehension.
- ⚠ Might misinterpret rules (e.g., should `"Y"` be a vowel? should case matter?) since no examples were provided.

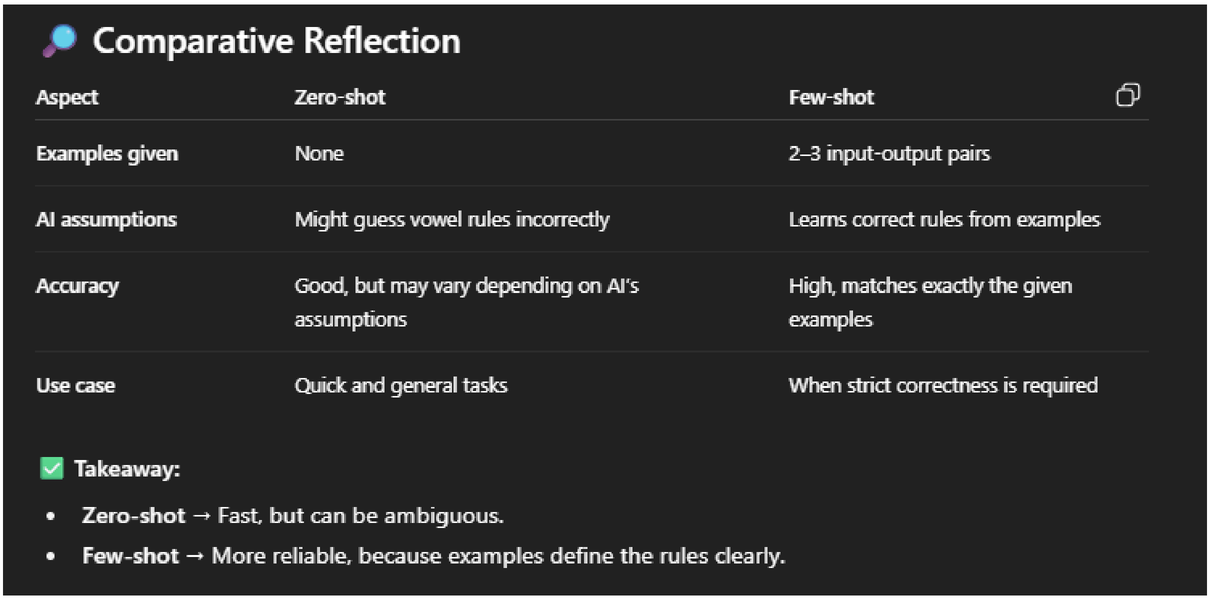


```python
def count_vowels(s: str) -> int:
    """
    Count the number of vowels in a string (a, e, i, o, u only).

    Args:
        s (str): Input string.

    Returns:
        int: Number of vowels found.
    """
    vowels = "aeiouAEIOU"
    count = 0
    for ch in s:
        if ch in vowels:
            count += 1
    return count


# Example usage (aligned with few-shot examples)
print(count_vowels("hello"))   # 2
print(count_vowels("Python"))  # 1
print(count_vowels("AEIOU"))   # 5
```



Explanation:

- With **explicit examples**, AI clearly knows:
    - Case-insensitive counting is required (since `"AEIOU" → 5`).
    - Only `a, e, i, o, u` are vowels (not `y`).
- Produces a predictable, rule-aligned function.



🔎 Comparative Reflection

Aspect	Zero-shot	Few-shot
Examples given	None	2–3 input-output pairs
AI assumptions	Might guess vowel rules incorrectly	Learns correct rules from examples
Accuracy	Good, but may vary depending on AI's assumptions	High, matches exactly the given examples
Use case	Quick and general tasks	When strict correctness is required

✅ **Takeaway:**

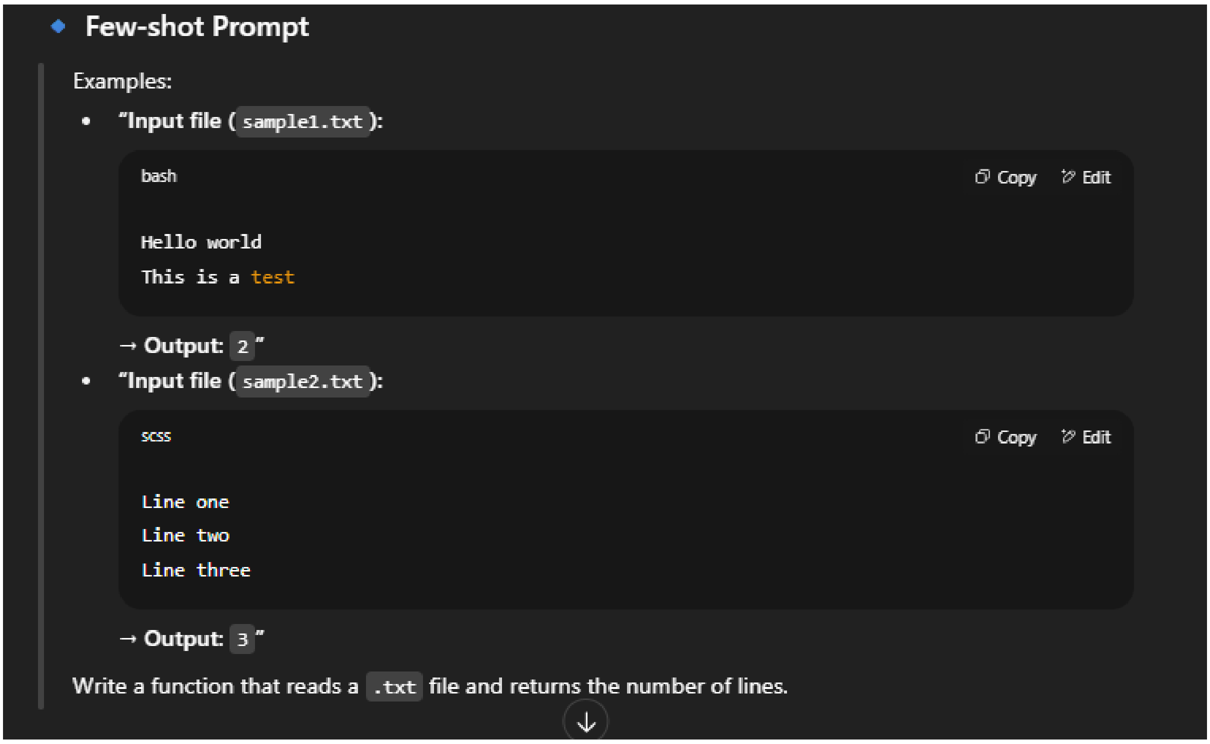
- **Zero-shot** → Fast, but can be ambiguous.
- **Few-shot** → More reliable, because examples define the rules clearly.

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 login



◆ **Few-shot Prompt**

Examples:

- **"Input file (`sample1.txt`):**

```bash
Hello world
This is a test
```

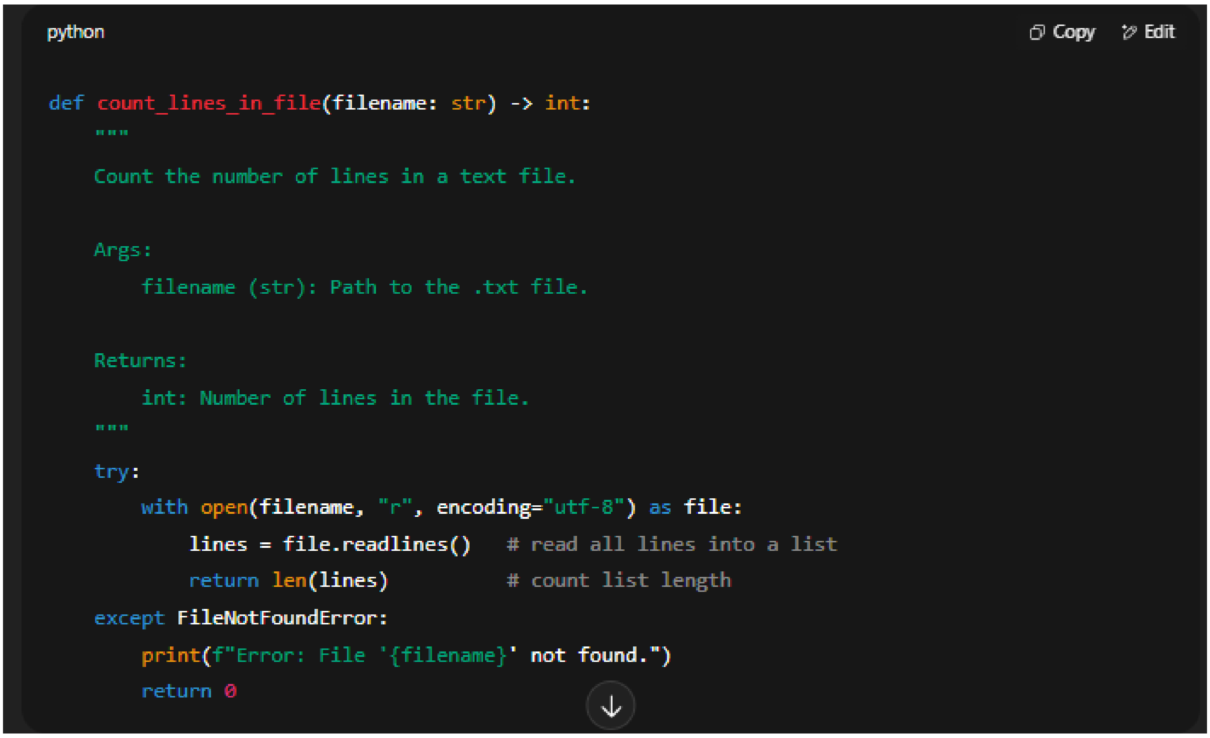
→ **Output:** `2`"

- **"Input file (`sample2.txt`):**

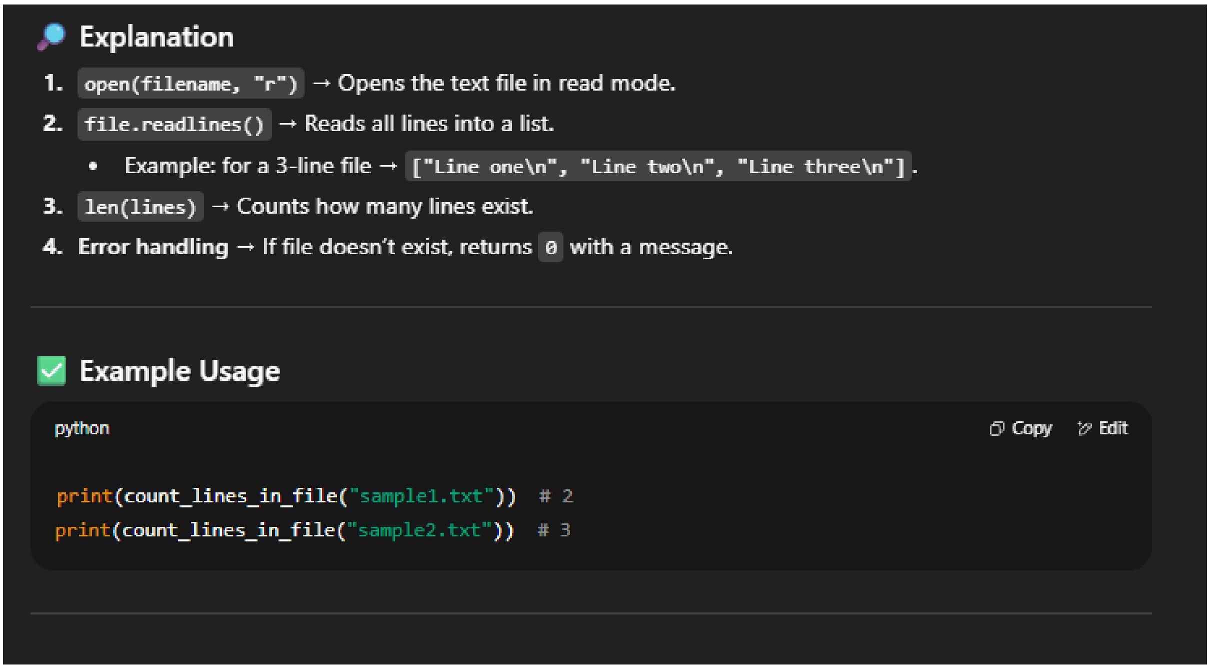
```scss
Line one
Line two
Line three
```

→ **Output:** `3`"

Write a function that reads a `.txt` file and returns the number of lines.



```python
def count_lines_in_file(filename: str) -> int:
    """
    Count the number of lines in a text file.

    Args:
        filename (str): Path to the .txt file.

    Returns:
        int: Number of lines in the file.
    """
    try:
        with open(filename, "r", encoding="utf-8") as file:
            lines = file.readlines()   # read all lines into a list
            return len(lines)          # count list length
    except FileNotFoundError:
        print(f"Error: File '{filename}' not found.")
        return 0
```



🔎 Explanation

1. `open(filename, "r")` → Opens the text file in read mode.
2. `file.readlines()` → Reads all lines into a list.
   - Example: for a 3-line file → `["Line one\n", "Line two\n", "Line three\n"]`.
3. `len(lines)` → Counts how many lines exist.
4. **Error handling** → If file doesn't exist, returns `0` with a message.

✅ Example Usage

python

```python
print(count_lines_in_file("sample1.txt"))  # 2
print(count_lines_in_file("sample2.txt"))  # 3
```