AI ASSISTED CODING

LAB 4: ADVANCED PROMPT TECHNIQUE

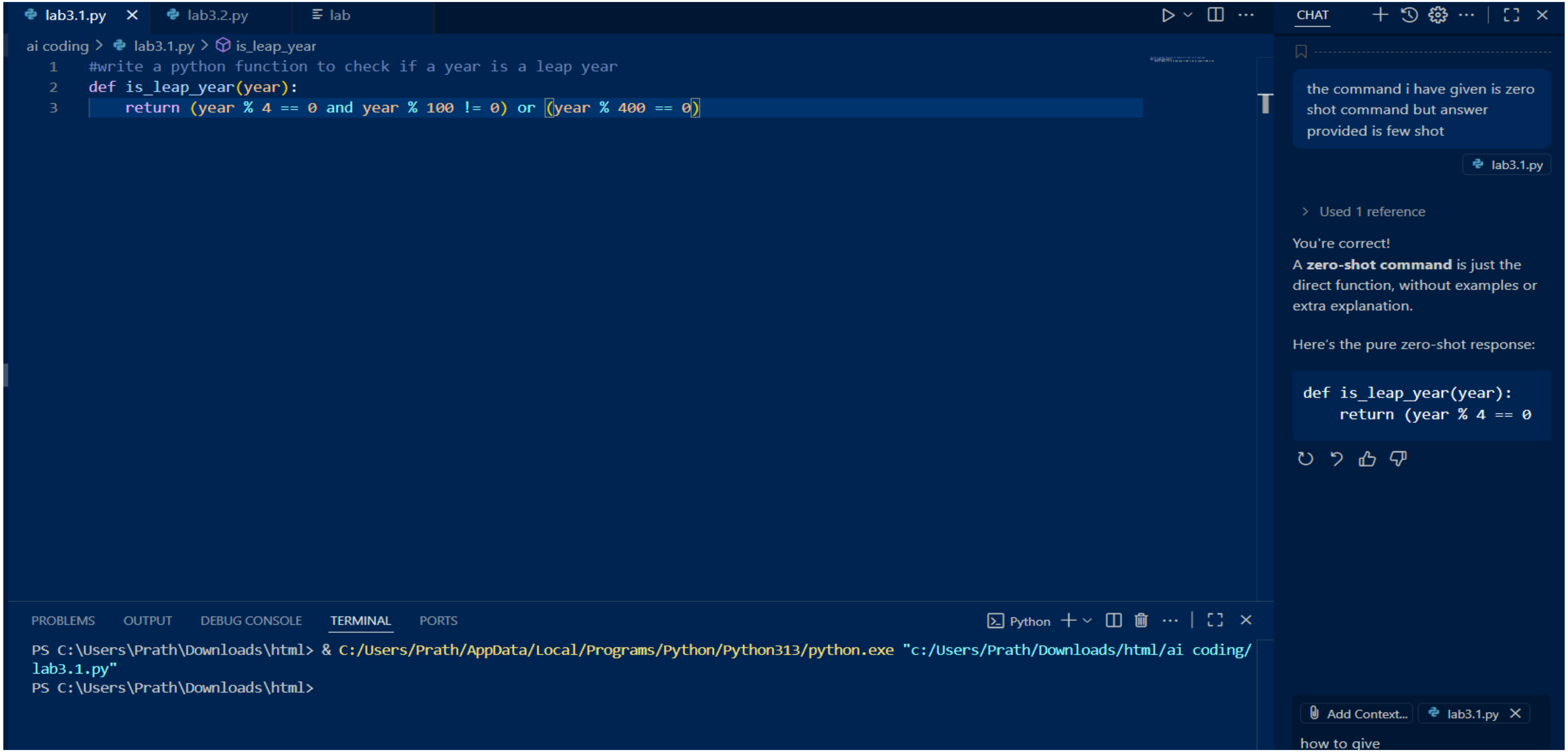
ZERO-SHOT

ONE-SHOT

FEW-SHOT

Task Description#1

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



```python
#write a python function to check if a year is a leap year
def is_leap_year(year):
    return (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0)
```

```
PS C:\Users\Prath\Downloads\html> & C:/Users/Prath/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/Prath/Downloads/html/ai coding/lab3.1.py"
PS C:\Users\Prath\Downloads\html>
```

the command i have given is zero shot command but answer provided is few shot

Used 1 reference

You're correct!
A **zero-shot command** is just the direct function, without examples or extra explanation.

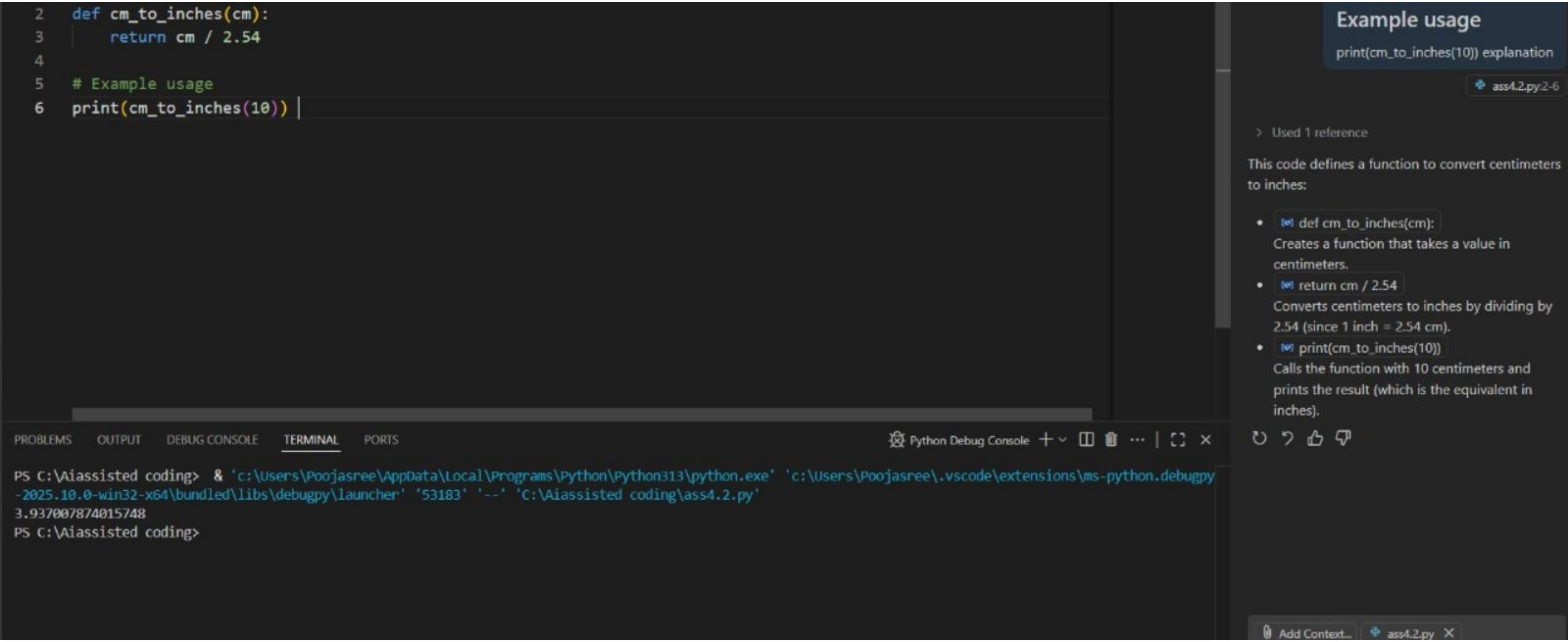
Here's the pure zero-shot response:

```python
def is_leap_year(year):
    return (year % 4 == 0
```

how to give

Task Description#2

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



```python
def cm_to_inches(cm):
    return cm / 2.54

# Example usage
print(cm_to_inches(10))
```

```
PS C:\Aiassisted coding>  & 'c:\Users\Poojasree\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\Poojasree\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '53183' '--' 'C:\Aiassisted coding\ass4.2.py'
3.937007874015748
PS C:\Aiassisted coding>
```

Example usage

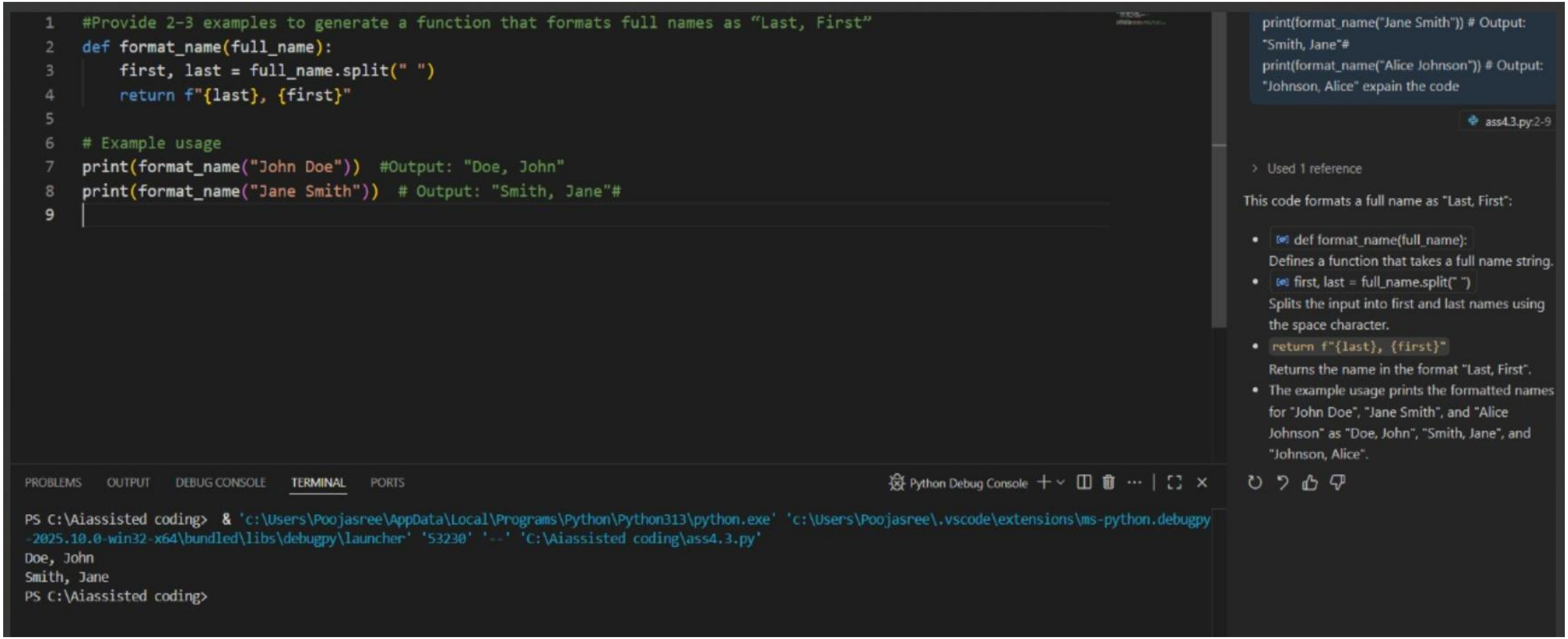
print(cm_to_inches(10)) explanation

This code defines a function to convert centimeters to inches:

- `def cm_to_inches(cm):` Creates a function that takes a value in centimeters.
- `return cm / 2.54` Converts centimeters to inches by dividing by 2.54 (since 1 inch = 2.54 cm).
- `print(cm_to_inches(10))` Calls the function with 10 centimeters and prints the result (which is the equivalent in inches).

Task Description#3

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



```python
#Provide 2-3 examples to generate a function that formats full names as “Last, First”
def format_name(full_name):
    first, last = full_name.split(" ")
    return f"{last}, {first}"

# Example usage
print(format_name("John Doe"))  #Output: "Doe, John"
print(format_name("Jane Smith"))  # Output: "Smith, Jane"#

```

print(format_name("Jane Smith")) # Output: "Smith, Jane"# print(format_name("Alice Johnson")) # Output: "Johnson, Alice" expain the code

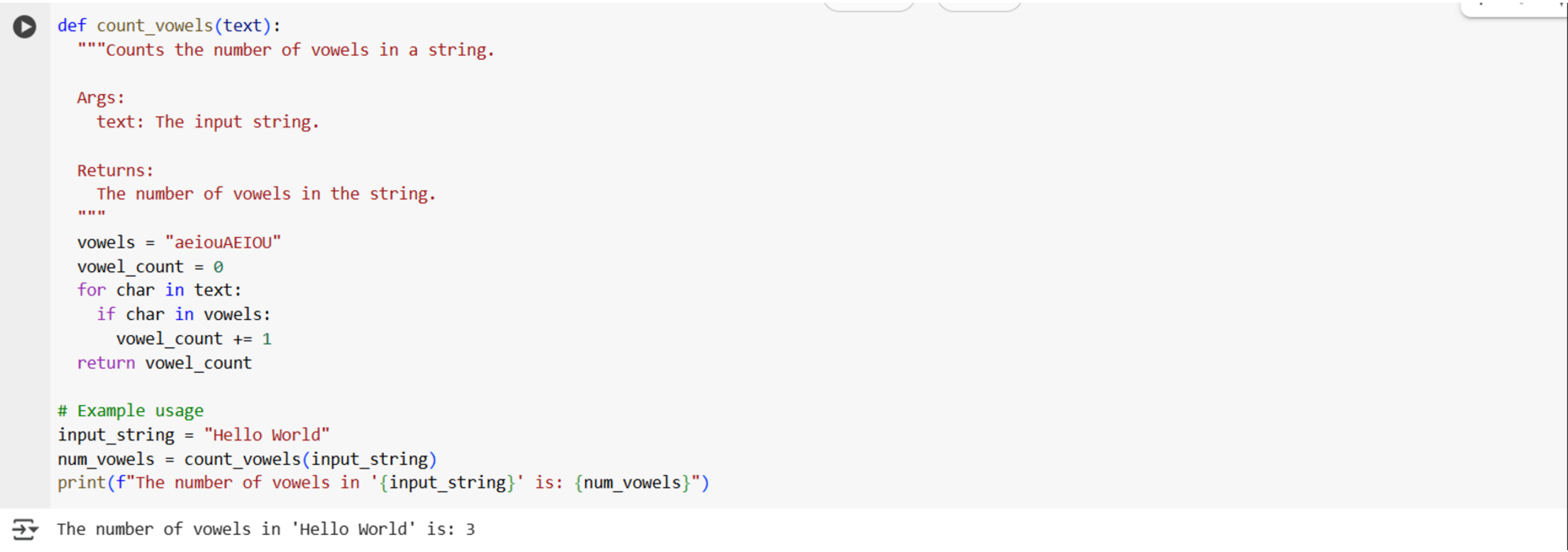
This code formats a full name as "Last, First":

- `def format_name(full_name):` Defines a function that takes a full name string.
- `first, last = full_name.split(" ")` Splits the input into first and last names using the space character.
- `return f"{last}, {first}"` Returns the name in the format "Last, First".
- The example usage prints the formatted names for "John Doe", "Jane Smith", and "Alice Johnson" as "Doe, John", "Smith, Jane", and "Johnson, Alice".

```
PS C:\Aiassisted coding>  & 'c:\Users\Poojasree\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\Poojasree\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '53230' '--' 'C:\Aiassisted coding\ass4.3.py'
Doe, John
Smith, Jane
PS C:\Aiassisted coding>
```

Task Description#4

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

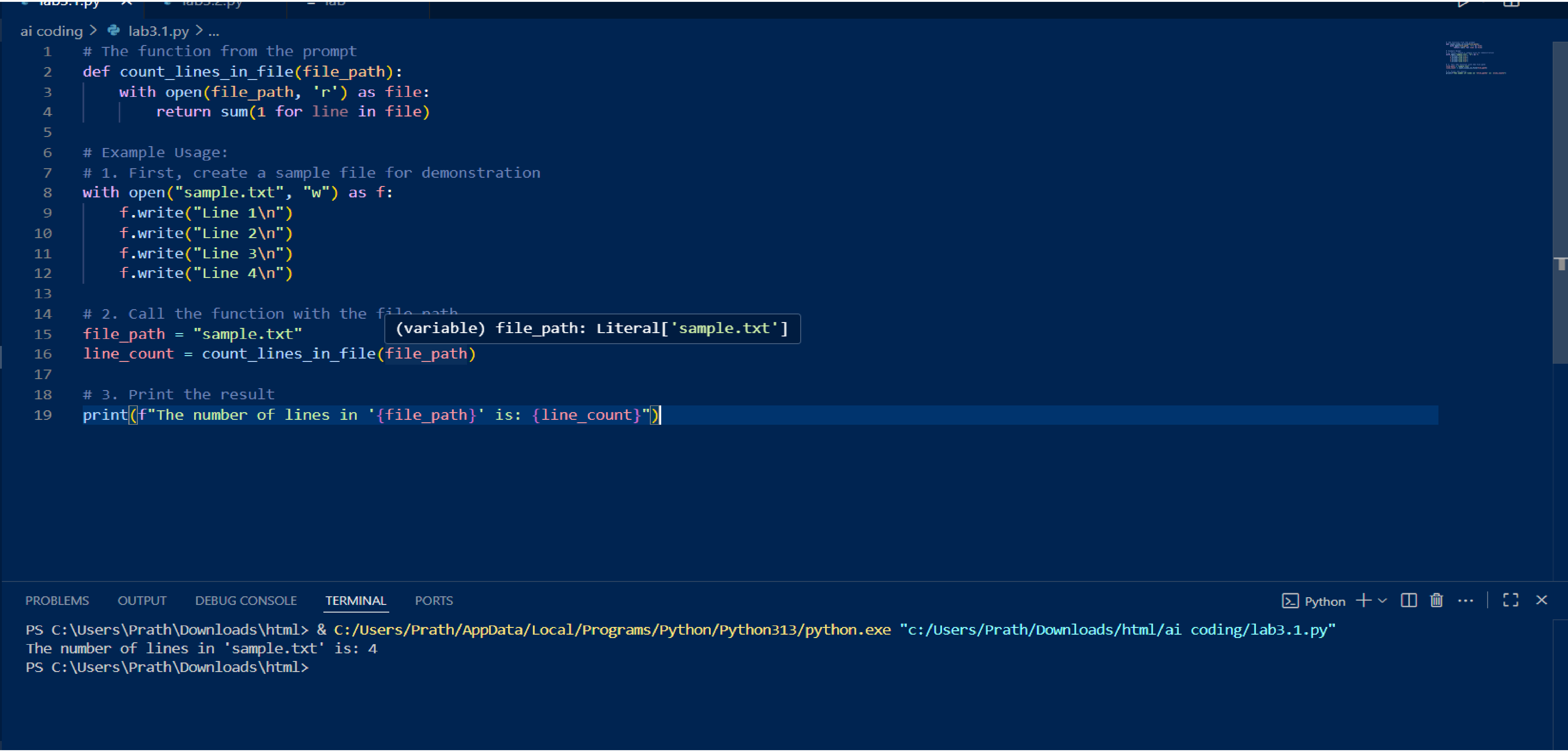


```python
def count_vowels(text):
  """Counts the number of vowels in a string.

  Args:
    text: The input string.

  Returns:
    The number of vowels in the string.
  """
  vowels = "aeiouAEIOU"
  vowel_count = 0
  for char in text:
    if char in vowels:
      vowel_count += 1
  return vowel_count

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

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

Task Description#5

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

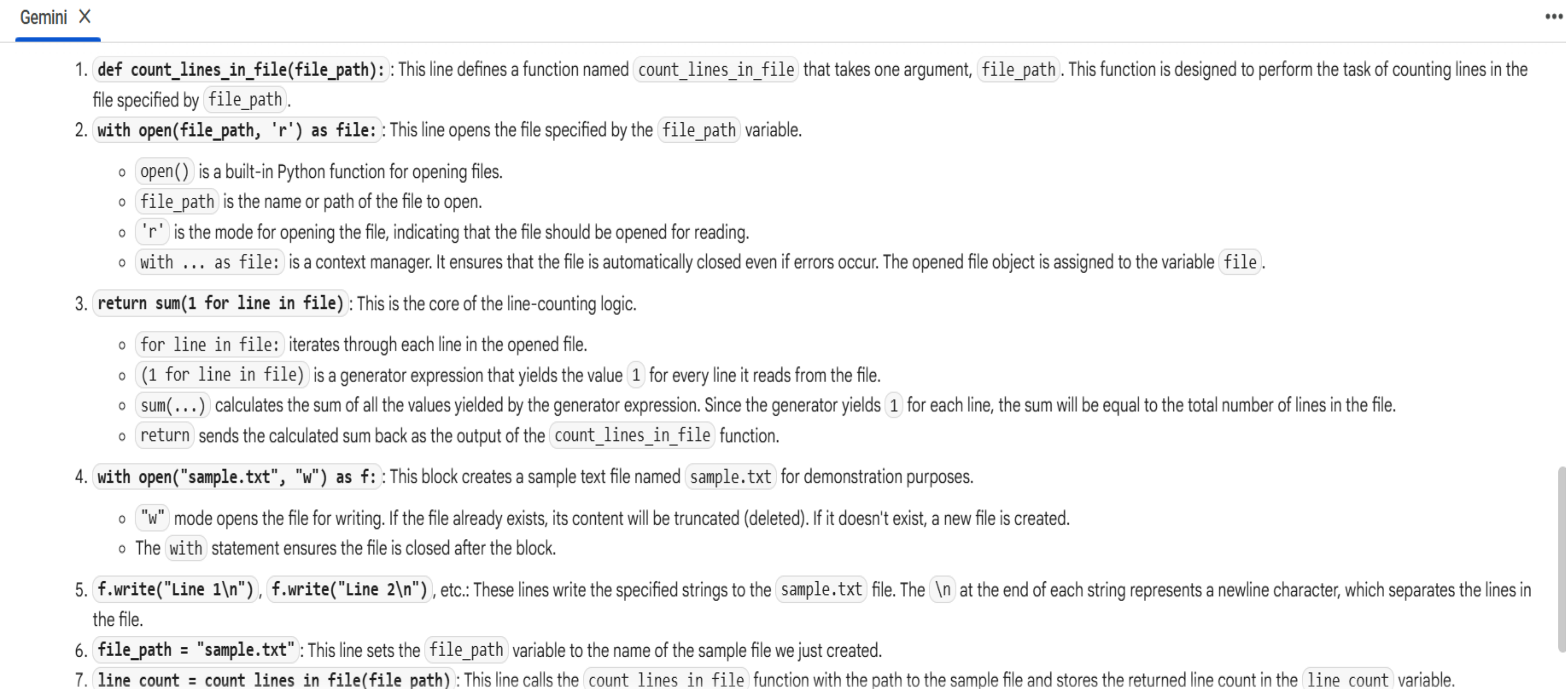


```python
# The function from the prompt
def count_lines_in_file(file_path):
    with open(file_path, 'r') as file:
        return sum(1 for line in file)

# Example Usage:
# 1. First, create a sample file for demonstration
with open("sample.txt", "w") as f:
    f.write("Line 1\n")
    f.write("Line 2\n")
    f.write("Line 3\n")
    f.write("Line 4\n")

# 2. Call the function with the file path
file_path = "sample.txt"
line_count = count_lines_in_file(file_path)

# 3. Print the result
print(f"The number of lines in '{file_path}' is: {line_count}")
```

(variable) file_path: Literal['sample.txt']

```
PS C:\Users\Prath\Downloads\html> & C:/Users/Prath/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/Prath/Downloads/html/ai coding/lab3.1.py"
The number of lines in 'sample.txt' is: 4
PS C:\Users\Prath\Downloads\html>
```

Explaination



1. `def count_lines_in_file(file_path):`: This line defines a function named `count_lines_in_file` that takes one argument, `file_path`. This function is designed to perform the task of counting lines in the file specified by `file_path`.
2. `with open(file_path, 'r') as file:`: This line opens the file specified by the `file_path` variable.
    - `open()` is a built-in Python function for opening files.
    - `file_path` is the name or path of the file to open.
    - `'r'` is the mode for opening the file, indicating that the file should be opened for reading.
    - `with ... as file:` is a context manager. It ensures that the file is automatically closed even if errors occur. The opened file object is assigned to the variable `file`.
3. `return sum(1 for line in file)`: This is the core of the line-counting logic.
    - `for line in file:` iterates through each line in the opened file.
    - `(1 for line in file)` is a generator expression that yields the value `1` for every line it reads from the file.
    - `sum(...)` calculates the sum of all the values yielded by the generator expression. Since the generator yields `1` for each line, the sum will be equal to the total number of lines in the file.
    - `return` sends the calculated sum back as the output of the `count_lines_in_file` function.
4. `with open("sample.txt", "w") as f:`: This block creates a sample text file named `sample.txt` for demonstration purposes.
    - `"w"` mode opens the file for writing. If the file already exists, its content will be truncated (deleted). If it doesn't exist, a new file is created.
    - The `with` statement ensures the file is closed after the block.
5. `f.write("Line 1\n")`, `f.write("Line 2\n")`, etc.: These lines write the specified strings to the `sample.txt` file. The `\n` at the end of each string represents a newline character, which separates the lines in the file.
6. `file_path = "sample.txt"`: This line sets the `file_path` variable to the name of the sample file we just created.
7. `line_count = count_lines_in_file(file_path)`: This line calls the `count_lines_in_file` function with the path to the sample file and stores the returned line count in the `line_count` variable.