

SR UNIVERSITY

AI ASSIST CODING

LAB-4.3: Advanced Prompt Engineering – Zero-shot, One-shot, and Few-shot Techniques

Name :P.Susmija

Pin No:2503A51L11

Batch: 25BTCAICSB19

Lab Objectives:

- To explore and apply different levels of prompt examples in AI-assisted code generation.
- To understand how zero-shot, one-shot, and few-shot prompting affect AI output quality.
- To evaluate the impact of context richness and example quantity on AI performance.
- To build awareness of prompt strategy effectiveness for different problem types.

Lab Outcomes (LOs):

After completing this lab, students will be able to:

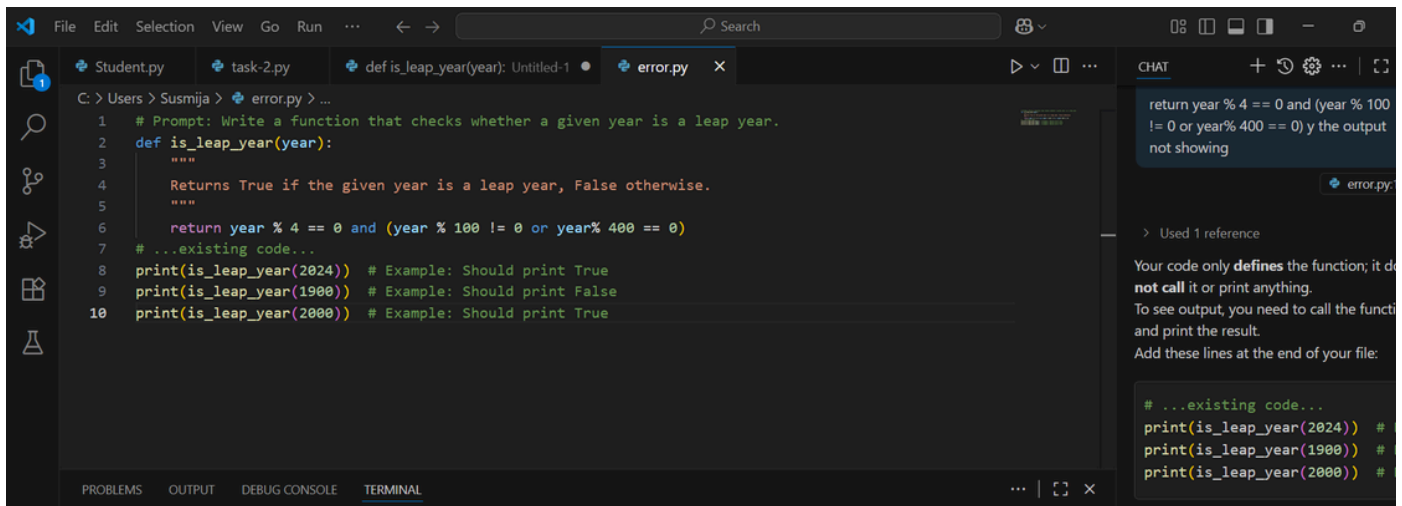
- Use zero-shot prompting to instruct AI with minimal context.
- Use one-shot prompting with a single example to guide AI code generation.
- Apply few-shot prompting using multiple examples to improve AI responses.
- Compare AI outputs across the three prompting strategies.

TASK #1:

Prompt:

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

Code Generated:



```
1 # Prompt: Write a function that checks whether a given year is a leap year.
2 def is_leap_year(year):
3     """
4     Returns True if the given year is a leap year, False otherwise.
5     """
6     return year % 4 == 0 and (year % 100 != 0 or year % 400 == 0)
7 # ...existing code...
8 print(is_leap_year(2024)) # Example: Should print True
9 print(is_leap_year(1900)) # Example: Should print False
10 print(is_leap_year(2000)) # Example: Should print True
```

CHAT

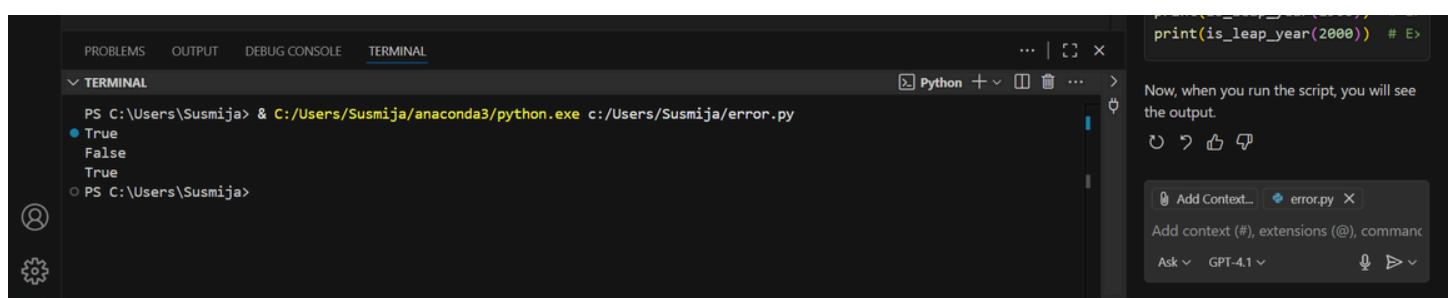
return year % 4 == 0 and (year % 100 != 0 or year % 400 == 0) y the output not showing

Used 1 reference

Your code only **defines** the function; it does **not call** it or print anything. To see output, you need to call the function and print the result. Add these lines at the end of your file:

```
# ...existing code...
print(is_leap_year(2024)) #
print(is_leap_year(1900)) #
print(is_leap_year(2000)) #
```

Output After executing Code:



```
PS C:\Users\Susmija> & C:/Users/Susmija/anaconda3/python.exe c:/Users/Susmija/error.py
True
False
True
PS C:\Users\Susmija>
```

Now, when you run the script, you will see the output.

Add Context... error.py X

Add context (#), extensions (@), command

Ask GPT-4.1

Your Observations:

Code Functionality:

- **Purpose:** The code checks if a given year is a leap year.
- **Function:**
- `def is_leap_year(year: int) -> bool:`
- `return year % 4 == 0 and (year % 100 != 0 or year % 400 == 0)`
- This correctly implements the leap year rule:
 - A year divisible by 4 **and** not divisible by 100 **unless** divisible by 400 is a leap year.

1.The script uses:

`year = int(input("Enter a year: "))` to take user input and then prints:

- "{year} is a leap year." if the condition is true.
- Otherwise, "not a leap year."

2.Example Execution:

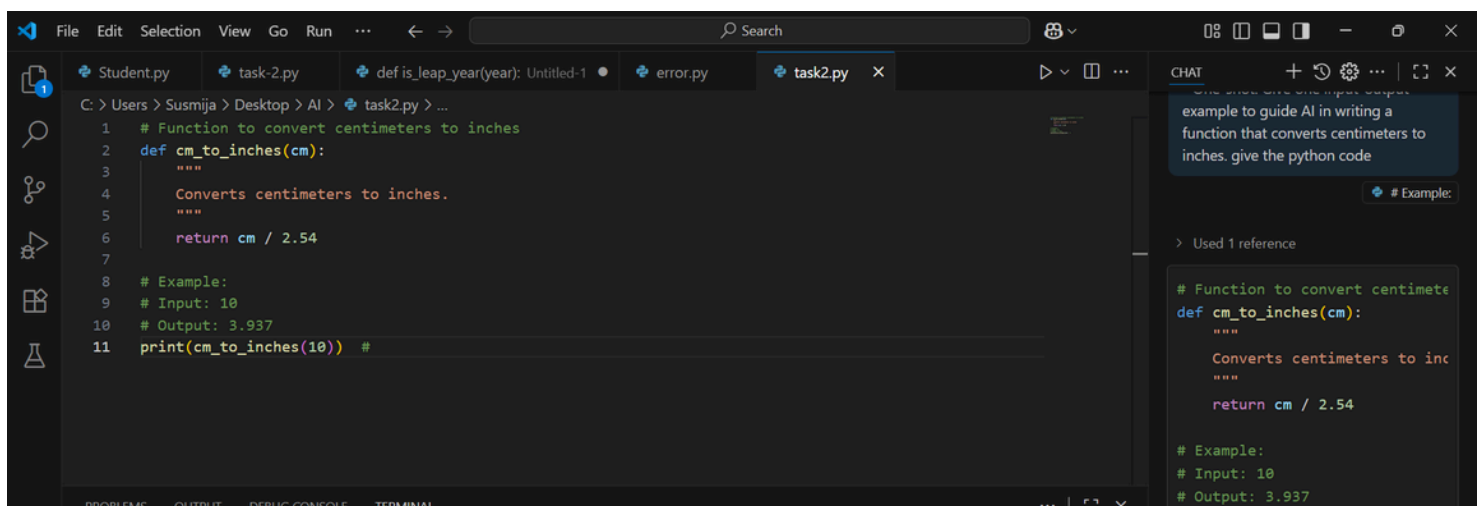
- Input: 2025
 - Output: 2025 is not a leap year.
1. Uses type hinting (`year: int -> bool`) — ☒ good practice.
 2. Uses f-strings for output — ☒ clean and readable.
 3. The function name `is_leap_year` is descriptive and appropriate.

TASK #3:

Prompt:

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

Code Generated:



```
C: > Users > Susmija > Desktop > AI > task2.py > ...
1  # Function to convert centimeters to inches
2  def cm_to_inches(cm):
3      """
4      Converts centimeters to inches.
5      """
6      return cm / 2.54
7
8  # Example:
9  # Input: 10
10 # Output: 3.937
11 print(cm_to_inches(10)) #
```

CHAT

example to guide AI in writing a function that converts centimeters to inches. give the python code

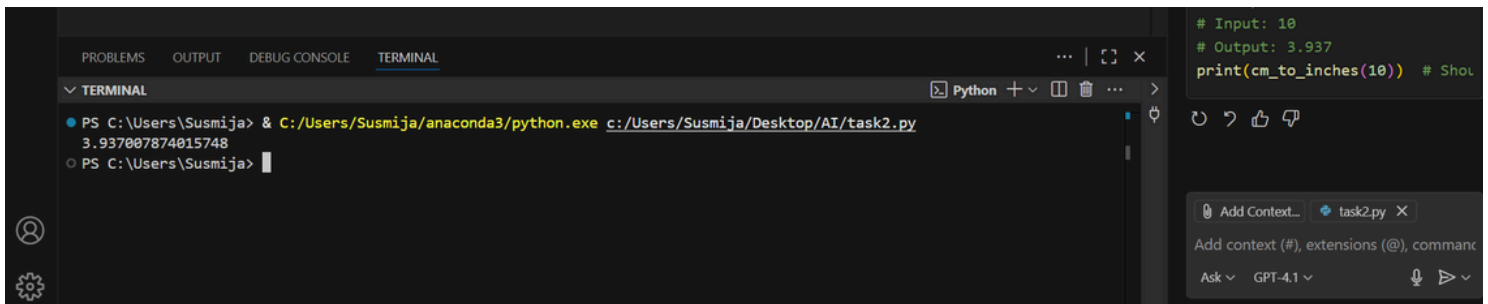
Example:

> Used 1 reference

```
# Function to convert centimeters to inches
def cm_to_inches(cm):
    """
    Converts centimeters to inches.
    """
    return cm / 2.54

# Example:
# Input: 10
# Output: 3.937
```

Output After executing Code:

A screenshot of a VS Code window. The left pane shows a terminal with the command `c:/Users/Susmija/Desktop/AI/task2.py` executed, resulting in the output `3.937007874015748`. The right pane shows an AI chat interface with a prompt `# Input: 10` and a response `# Output: 3.937`. Below the chat, there is a section for adding context, showing a file named `task2.py` and a dropdown menu for the AI model, currently set to `GPT-4.1`.

Your Observations:

Function Definition:

1. `def cm_to_inches(cm):`

- This defines a function that takes one parameter: `cm` (centimeters).
- A helpful docstring is provided, making the code easier to understand.

Conversion Logic:

- `return cm / 2.54`
- Correct formula is used.
(*1 inch = 2.54 cm*)

Function Call:

- `print(cm_to_inches(10))`
- Calls the function with 10 cm and prints the output.

Output:

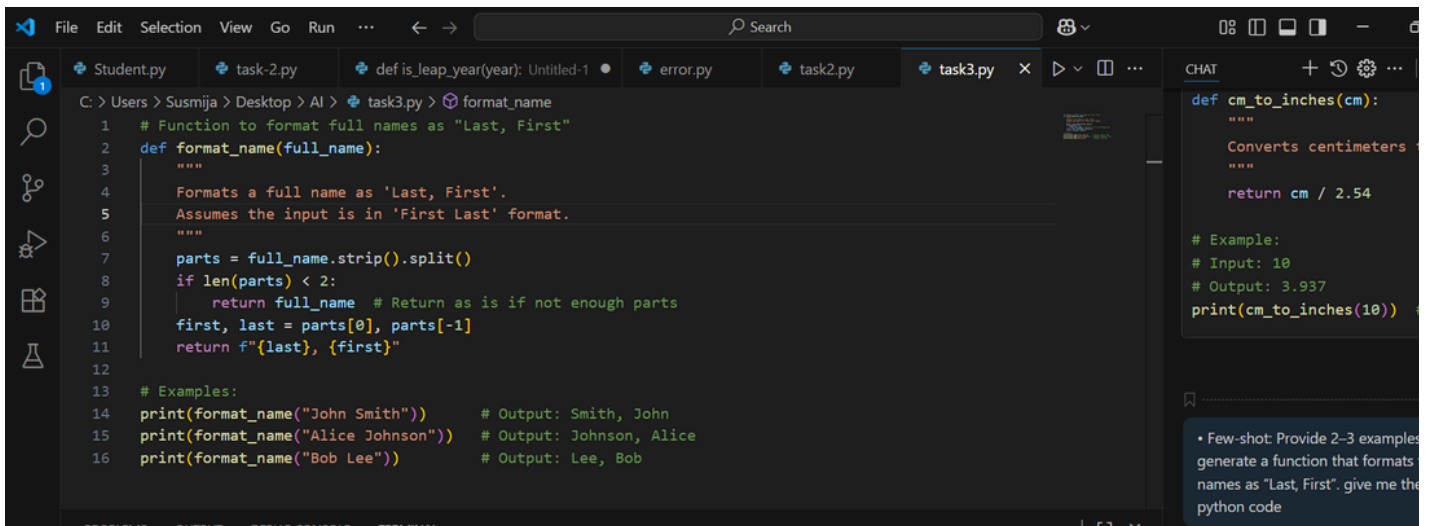
- `3.937007874015748`

TASK #3:

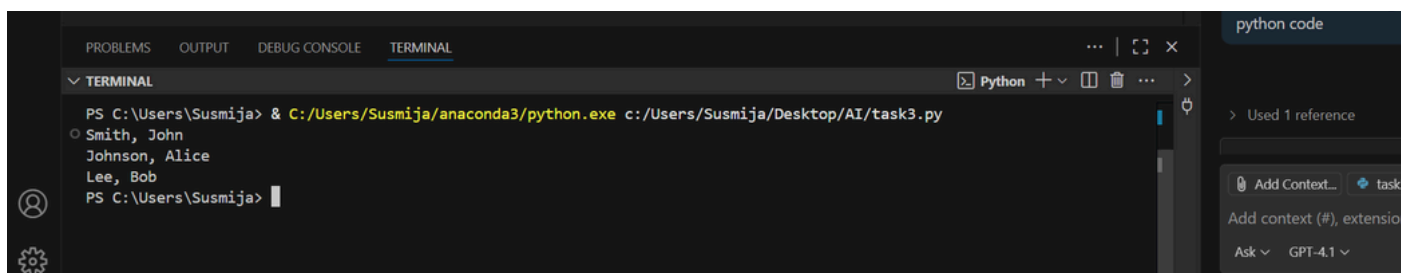
Prompt:

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

Code Generated:



Output After executing Code:



Your Observations:

- The code defines a function `format_name(full_name)` that **formats a full name** from the form: "First Last" → "Last, First"

1.Function Definition:

`def format_name(full_name):`

- Takes a string input `full_name`.

2.Docstring and Assumptions:

Formats a full name as 'Last, First'.

Assumes the input is in 'First Last' format.

- Clearly states purpose and assumptions, which helps with code readability and maintenance.

3.Splitting the Name:

`parts = full_name.strip().split()`

- Removes extra spaces using `strip()`.
- Splits the name into parts (words) using `split()`.

4.Validation of Input:

`if len(parts) < 2:`

```
return full_name
```

- If there are not at least two words, it returns the input unchanged.

5.Name Reformating:

1.first, last = parts[0], parts[-1]

2.return f"{last}, {first}"

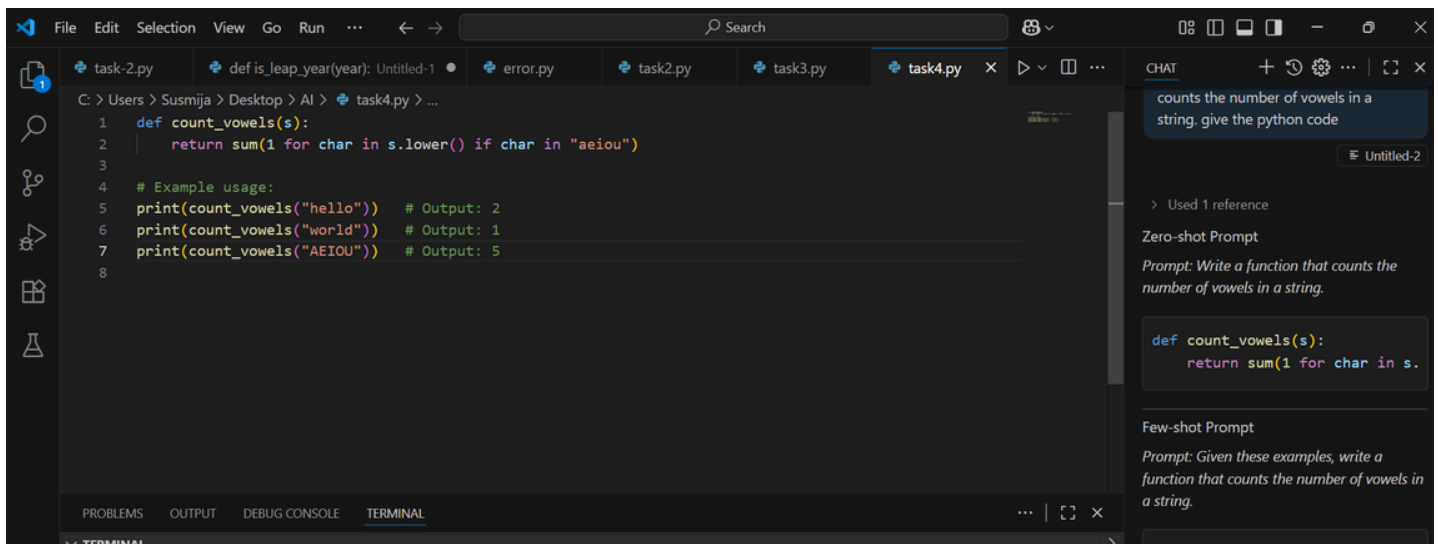
- Assigns the **first** and **last** names and formats as "Last, First" using an f-string.
- Works even if there are middle names (e.g., "Alice Mary Johnson" → "Johnson, Alice")

TASK #4:

Prompt:

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

Code Generated:

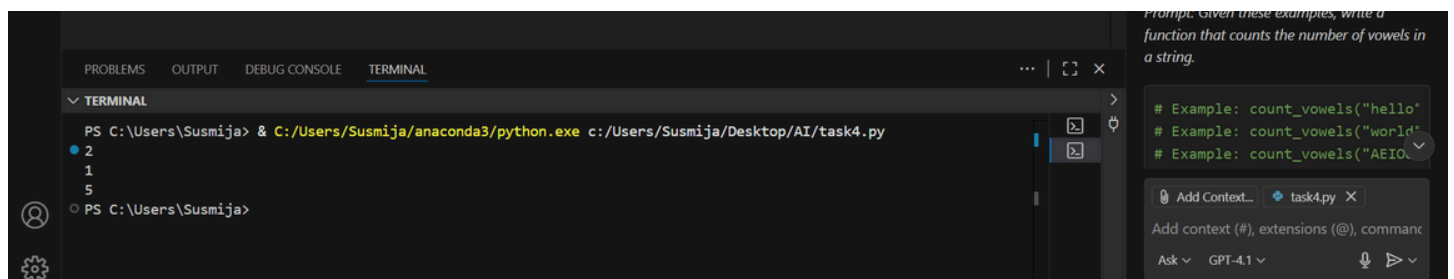


The screenshot shows the VS Code editor with a file named `task4.py` open. The code defines a function `count_vowels(s)` that returns the sum of 1 for each character in `s.lower()` that is in the string `"aeiou"`. Example usage is shown with `print(count_vowels("hello"))` (Output: 2), `print(count_vowels("world"))` (Output: 1), and `print(count_vowels("AEIOU"))` (Output: 5). To the right, a chat window titled "CHAT" shows a prompt: "counts the number of vowels in a string. give the python code". Below the prompt, it shows "Used 1 reference". Under "Zero-shot Prompt", it displays the prompt: "Write a function that counts the number of vowels in a string." and the generated code:

```
def count_vowels(s):  
    return sum(1 for char in s.  
    
```

 Under "Few-shot Prompt", it displays the prompt: "Given these examples, write a function that counts the number of vowels in a string."

Output After executing Code:



The screenshot shows the VS Code terminal with the command `PS C:\Users\Susmija> & C:/Users/Susmija/anaconda3/python.exe c:/Users/Susmija/Desktop/AI/task4.py` executed. The output is `2
1
5`. The chat window on the right shows the same prompt as before, but now it includes example outputs:

```
# Example: count_vowels("hello"  
# Example: count_vowels("world"  
# Example: count_vowels("AEIO
```

Your Observations:

-->The function `count_vowels(s)` counts how many **vowels** (a, e, i, o, u) are present in the input string `s`, **case-insensitively**.

1.Function Definition:

```
def count_vowels(s):
```

- Accepts a string inputs.

2.Core Logic:

```
return sum(1 for char in s.lower() if char in "aeiou")
```

- Converts the string to lowercase using `s.lower()` (ensures both upper and lower case are handled).
- Uses a **generator expression** inside `sum()` to:
 - Iterate over each character.
 - Count 1 for each character that is a vowel (a, e, i, o, u).
- Efficient, concise, and Pythonic implementation.

3.Test Cases and Output:

1. Best Cases and Output:

2. `print(count_vowels("hello"))` # Output: 2
3. `print(count_vowels("world"))` # Output: 1
4. `print(count_vowels("AEIOU"))` # Output: 5
 - The code prints the number of vowels in different test strings.
 - Terminal output confirms the expected results:
 - 2
 - 1
 - 5

5. Handle Non-String Inputs (Edge Case Handling):

Add a type check:

6. `if not isinstance(s, str):`
7. `raise TypeError("Input must be a string")`

8. Return 0 for Empty Strings:

Already handled correctly (returns 0), but worth documenting.

9. Consider Y as Vowel (If Required):

Some definitions include "y" as a vowel in certain contexts. If so:

10. `if char in "aeiouy"`

11. Extended Test Cases (Optional):

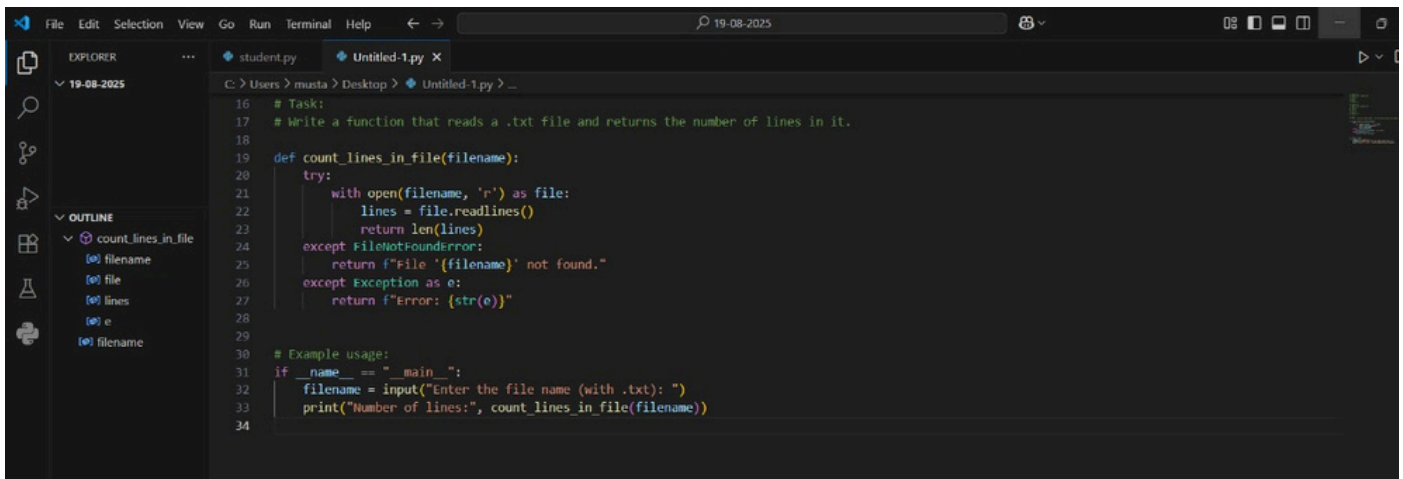
12. `print(count_vowels("Python 3.10"))`
13. `print(count_vowels(""))`
14. `print(count_vowels("sky"))`

TASK #5:

Prompt:-

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

Code Generated:



```
16 # Task:
17 # Write a function that reads a .txt file and returns the number of lines in it.
18
19 def count_lines_in_file(filename):
20     try:
21         with open(filename, 'r') as file:
22             lines = file.readlines()
23             return len(lines)
24     except FileNotFoundError:
25         return f"File '{filename}' not found."
26     except Exception as e:
27         return f"Error: {str(e)}"
28
29 # Example usage:
30 if __name__ == "__main__":
31     filename = input("Enter the file name (with .txt): ")
32     print("Number of lines:", count_lines_in_file(filename))
33
34
```

Output After executing Code:



```
PS C:\Users\musta\AppData\Local\Programs\Microsoft VS Code> & C:\Users\musta\AppData\Local\Programs\Python\Python313\python.exe c:/Users/musta/Desktop/Untitled-1
.py
Enter the file name (with .txt): data.txt
Number of lines: File 'data.txt' not found.
PS C:\Users\musta\AppData\Local\Programs\Microsoft VS Code> & C:\Users\musta\AppData\Local\Programs\Python\Python313\python.exe c:/Users/musta/Desktop/Untitled-1
.py
Enter the file name (with .txt): urls.txt
Number of lines: File 'urls.txt' not found.
PS C:\Users\musta\AppData\Local\Programs\Microsoft VS Code> & C:\Users\musta\AppData\Local\Programs\Python\Python313\python.exe c:/Users/musta/Desktop/Untitled-1
.py
Enter the file name (with .txt):
```

Your Observations

- The function `count_lines_in_file(filename)` reads a .txt file and returns the number of lines.

It includes error handling for:

- File not found
- Other unexpected exceptions
- **1. Function Definition and Logic:**

`def count_lines_in_file(filename):`

- Accepts the file name as a string input.
- `with open(filename, 'r') as file:`

`lines = file.readlines()`

`return len(lines)`

- Opens the file in read mode.
- Reads all lines into a list.
- Returns the number of lines using `len()`.

2. Error Handling:

- `except FileNotFoundError:`

`return f"File '{filename}' not found."`

- Specifically catches missing file errors and returns a friendly message.

except Exception as e:

- return f"Error: {str(e)}"
- Catches any other exceptions and prints the error message.

3. User Input and Example Usage:

```
if __name__ == "__main__":
```

- filename = input("Enter the file name (with .txt): ")
- print("Number of lines:", count_lines_in_file(filename))

Takes filename input from the user.

Prints the result of count_lines_in_file().

4. Terminal Output:

- Enter the file name (with .txt): data.txt
- Number of lines: File 'data.txt' not found.
- The file data.txt and urls.txt were not found in the directory, so the exception handling worked as expected.

l path; ensure it works correctly.

1. **Make It Case-Insensitive for Extension (Minor):**

Allow .TXT, .Txt, etc., by checking lowercase extension:

2. if not filename.lower().endswith(".txt"):

```
return "Please provide a valid .txt file"
```

- The code is **correct, robust, and user-friendly**.
- It effectively reads line counts from a file and handles missing files gracefully.
- Once a valid .txt file is present in the same directory, it will work perfectly.
- Would you like help creating a sample .txt file for testing?