**Name:** Jaeger Liebster

**Date Created:** September 21, 2025

**Program Description:**

The purpose of this program is to check a message for common spam keywords and assign it a probability of being spam based on how many keywords it detects.

**Functions used in the Program**

1. Function Name:

scan\_message

* **Description:** This function checks messages for spam by looking for specific words, counting how often they appear to calculate a score, and then showing which words were found.
* **Parameters:**
  1. message (string): The user-provided email message to be analyzed.
  2. spam\_keywords (list): A list of spam words and phrases to search for within the message.
* **Variables:**
  1. score (integer): A counter that holds the total spam score.
  2. found\_keywords (set): A set used to store the unique keywords found in the message to avoid duplicates in the final report.
  3. normalized\_message (string): A lowercase version of the input message to ensure case-insensitive matching.
  4. occurrences (integer): The number of times a specific keyword appears in the message.
* **Logical Steps:**
  1. Initialize the  
      score to 0 and found\_keywords to an empty set.
  2. Convert the input  
      message to lowercase and store it in normalized\_message.
  3. Loop through each keyword in the  
      spam\_keywords list.
  4. For each keyword, count its total  
      occurrences within normalized\_message.
  5. If a keyword occurs one or more times, add the number of  
      occurrences to the score and add the keyword to the found\_keywords set.
* **Returns:** A tuple containing the final integer score and a sorted list of the unique found\_keywords.

2. Function Name:

rate\_likelihood

* **Description:** This function takes a numerical score and returns a human-readable string that rates the likelihood of the message being spam.
* **Parameters:** score (integer): The spam score calculated by the scan\_message function.
* **Variables:** This function uses no internal variables.
* **Logical Steps:**
  1. Use an if/elif/else structure to evaluate the  
      score.
  2. Return a specific rating string based on which range the  
      score falls into (e.g., 0 for "Very low likelihood", 1-2 for "Low likelihood", etc.).
* **Returns:** A string describing the likelihood of the message being spam (e.g., "High likelihood").

3. Function Name:

main

* **Description:** This is the primary function that runs the application. It handles user I/O and calls the other functions in the correct order to produce the final analysis.
* **Parameters:** None.
* **Variables:**
  1. lines (list): A list to hold each line of the user's multi-line input.
  2. user\_message (string): The complete message after joining all input lines.
  3. spam\_score (integer): The score returned from the scan\_message function.
  4. detected\_words (list): The list of trigger words returned from the scan\_message function.
  5. likelihood (string): The rating string returned from the rate\_likelihood function.
* **Logical Steps:**
  1. Print a welcome message and instructions for the user.
  2. Create a loop to capture multi-line input from the user, storing each line in the lines list. The loop breaks when the user submits an empty line.
  3. Join the  
      lines into the single user\_message string.
  4. Call  
      scan\_message with the user\_message and SPAM\_KEYWORDS list to get the spam\_score and detected\_words.
  5. Call rate\_likelihood with the spam\_score to get the likelihood.
  6. Print the final analysis report, including the score, likelihood, and trigger words.
* **Returns:** None.

**Logical Steps**

1. The  
    main function is called when the script is executed.
2. Within  
    main, the scan\_message function is called to analyze the user's input.
3. Next, the rate\_likelihood function is called to convert the score from scan\_message into a rating.
4. Finally, main prints the results from the previous two function calls.

**Link to your repository:**https://github.com/RoarinThundah/COP2373\_Jaeger

**Output Screenshot:**

