**Exercise 2 Technical Design Document**

**Name:** Vanessa Belous

**Date Created:** September 13th, 2025

**Program Description:**

This program analyzes a user input email message to scan for potential spam keywords or phrases. Then, the program will provide the user with a “spam score” depending on how likely the email is to be spam.

**Functions used in the Program (listed in the order they’re called in):**

1. **Function Name:** main( )

**Description:** This function asks the user to input their email message, then calls the two practical functions. After that, this function prints out all of the details of the analysis, formatting them in the process.

**Parameters:** None

**Variables:**

1. global message = stores the email in a way that makes it accessible to the entire program file.

**Logical Steps:**

1. lists variables used
2. Asks user to input their email message. Converts every character in the email to lowercase so it can be compatible with the lowercase list of spam keywords/phrases.
3. prints the total spam score
4. prints the likelihood that the email is spam, as a percentage
5. Uses an if statement to check if there was any spam at all, to provide a customized message stating so. Otherwise, uses a for loop to print every word that was flagged.

**Returns:** None

1. **Function Name:** check\_for\_spam()

**Description:** Compares every word in the list of common spam keywords and phrases (created at the top of the file, outside of any functions) to see how much of it is likely spam.

**Parameters:** None

**Variables:**

1. global message = takes the email stored in main( ) so it can be compared to
2. global spam\_score = every time a spam word in the list is found in the email, this value accumulates
3. global spam\_in\_email\_list = will house every word that was flagged as spam in the email

**Logical Steps:**

1. list variables used in this function
2. A simple for loop. For every word in the list of common spam keywords and phrases, it will see if that word is in the email. If it is, it will append that word to spam\_in\_email\_list, and add one to spam\_score. It runs until every word in the list of common spam words is tested against the email

**Returns:** None

1. **Function Name:** likelihood\_spam( )

**Description:** Calculates how much of the inputted email is made up of common spam keywords or phrases.

**Parameters:** None

**Variables:**

1. global spam\_score = uses the value affected by the check\_for\_spam( ) function after it finishes running, counting how many words or phrases were flagged as spam in the email
2. global message = the email message taken as input in main( ), used to calculate likelihood
3. words\_in\_message = stores how many words are in the email message as an integer, to be used in an expression
4. likelihood = calculates the likelihood of spam by using the variables spam\_score and words\_in\_message. Rounds the result to two decimal places before returning the value to main( )

**Logical Steps:**

1. lists global variables used/referenced
2. calculates the length of the email using .split( ) nested within len( )
3. calculates the likelihood of the email being spam. spam\_score is divided by words\_in\_message, and that result is multiplied by 100 so it can be displayed as a percentage. All of that is within the round function, so the variable can store the rounded result

**Returns:** the calculated variable likelihood

**Program Logical Steps:**

1. Initializes any global variables, as well as the list of common spam keywords and phrases (spam\_words\_list)
2. main( ) is called
3. inside main( ):
   1. the user inputs their email message
   2. check\_for\_spam( ) is called, which checks if any of the words match what is stored in spam\_words\_list. If a common spam word/phrase is found, it will be appended to a list storing flagged words, and spam\_score goes up one
   3. likelihood\_spam( ) is called, which uses spam\_score and the value of the length of the email (that this function calculates) to calculate how much of the email is likely to be spam.
   4. finally, main( ) displays the findings of the program. The spam score is reported, and the likelihood is displayed as a percentage of the whole email. The words flagged as common spam words will be printed in a formatted way if there was spam found, otherwise will print a statement saying no spam words were found.

**Link to Repository:** <https://github.com/VBelous1/COP2373/tree/master/Week%205>

**Output Screenshot:**

**A screen shot of a computer

AI-generated content may be incorrect.**