MotionCut Python Programming Project 2

Design Choices:

1. Modularity:
    a. I opted for a modular design to enhance readability, maintainability, and reusability of the code. Each logical unit has its own function, making it easier to understand and modify specific parts of the code independently.

2. Interaction with Users / User Interface:
    a. The code provides a user-friendly interface with a welcome message and clear instructions for the user. This enhances the overall user experience and makes the program more accessible.
3. Input Validation:
    a. I included input validation to handle cases where the user might provide empty input or input with non-alphabetic characters. This ensures that the program works with valid input and provides informative error messages for the user.
4. Comments and Documentation:
    a. I added comments to explain the purpose of each function and important code blocks. This documentation enhances code readability and helps others understand the logic.

Additional Feature and Improvements:

1. Word Counting:
    a. The program accurately counts the number of words in the input text. It uses the **split ()** method to break the text into words based on spaces.
2. Letter Counting:
    a. The code counts the number of letters in the input string, considering only alphabetic characters. This is achieved using a generator expression and the **sum ()** function.
3. Input Validation:
    a. The code checks for empty input and non-alphabetic characters, providing appropriate error messages to guide the user.

Challenges Encountered:

1. Validating Inputs:
    a. It was important to give serious thought to how to handle different types of invalid input to ensure strong input validation. It was difficult to find a happy medium between software strength and user-friendliness.
2. Indexing of Correct Answers:
    a. One possible source of misunderstanding is the indexing for right responses, which starts from 1. It was critical to check that the user's choices correspond to the right responses in the code.

3. Clarity of the User Interface:
    a. Keeping the code short while maintaining a straightforward user interface with explicit instructions was no easy feat. Finding the sweet spot between thoroughness and ease of use was of the utmost importance.

4. Code Comments:
    a. It took careful consideration to make sure that comments added value without being repetitive, all while helping with comprehension.

By making these design choices, the code becomes more structured, user-friendly, and maintainable. The inclusion of input validation ensures that the program handles a variety of user inputs gracefully.