MotionCut Python Programming Project 1

Design Choices:

1. Structure of Functions:
    a. To make the code more modular and easier to read and understand, I divided it into three functions: main, ask_question, and welcome.
    b. Activities like showing a greeting, asking a question, and executing the main game loop are encapsulated in the functions.

2. Interaction with Users:
    a. A user-friendly welcome message and quiz-taking instructions are provided by the game.
    b. An easily understandable structure is used for multiple-choice questions, and the responses are input using letters (A, B, C, etc.).

3. Evaluation Method:
    a. A straightforward scoring system that gets the job done. With every right response, the user's score rises.
    b. The final score is displayed at the end of the quiz.

4. Validating Inputs:
    a. To guarantee that users provide legitimate answers, I instituted input validation. If the user inputs an incorrect value, the application will ask them to re-enter their decision.

5. 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. Customization:
    a. Users can easily edit the quiz to their liking by modifying the lists that hold the data (questions, options, and answers).

2. Dealing with Numeric Input:
    a. The input validation can also process numerical inputs, even though the quiz only accepts letters (A, B, C, etc.). The software will ask the user to input a letter if they type a number.

3. Scalability:
    a. The code is designed to handle additional questions simply by adding entries to the questions, options, and answers lists.

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.

Aiming on simplicity, modularity, and user-friendliness, the design prioritizes customization and makes sure both developers and users understand it well. Making sure the user experience was smooth and keeping the code clear were the key obstacles.