Quiz Game Design and Implementation Document

Design Choices:

1. Object Oriented Design:

* I chose to encapsulate the quiz functionality within a class (`Quiz Game`) to promote modularity and maintainability.

* The class includes methods for initializing the quiz, running the quiz, and displaying the final score.

2. Customization:

* I implemented customization by creating a list (`quiz_data`) that holds the questions, options, and correct answers.

* This design choice allows users to easily modify the quiz by updating the `quiz_data` list.

3. User Interaction:

* User interaction is facilitated through the `input()` function, allowing users to input their answers.

* Input is converted to uppercase to ensure case*insensitive comparison for correct answers.

4. Scoring System:

* The user's score is tracked using the `user_score` attribute within the `QuizGame` class.

* For each correct answer, the score is incremented, providing feedback to the user on their performance.

5. Code Organization:

* The code is organized into functions and methods, promoting readability and ease of maintenance.

* The main functionality is encapsulated within the `QuizGame` class, with a separate `main()` function for execution.

Features:

1. Customization:

* Users can easily customize the quiz by modifying the `quiz_data` list with new questions, options, and correct answers.

* This allows for flexibility in adapting the quiz to different topics or preferences.

2. User Feedback:

* Users receive immediate feedback on each question, whether their answer is correct or incorrect.

* For incorrect answers, the correct answer is displayed, providing an educational aspect to the quiz.

3. Final Score Display:

* At the end of the quiz, users are presented with their final score out of the total number of questions.

4. Object*Oriented Structure:

* The use of a class (`QuizGame`) promotes code organization and encapsulation.

* Methods within the class handle specific aspects of the quiz, making the code more modular and easier to understand.

Challenges:

1. Validation:

* Basic input validation is lacking in the current implementation. Enhancing this aspect to handle unexpected inputs or ensuring valid answers could be a future improvement.

2. Additional Features:

* While the current code fulfills the basic project objectives, additional features such as a timer, category selection, or a more sophisticated scoring system could be explored for a more comprehensive quiz experience.

Conclusion:

The design and implementation of the quiz game prioritize simplicity, customization, and user feedback. The code structure allows for easy modification of quiz content and sets the foundation for future enhancements. Balancing simplicity with the potential for expansion ensures that the code remains accessible to users with varying levels of Python experience.