**WEEK 1 PYTHON INTERSHIP QUIZ GAME REPORT**

1. Introduction and Setup:
   * The quiz() function is defined to encapsulate the quiz game logic.
   * A list called questions is created, containing dictionaries for each question. Each dictionary includes the question, correct answer, and multiple choice options.
2. Quiz Game Logic:
   * The quiz() function prints a welcome message and instructions.
   * The list of questions is shuffled randomly using random.shuffle(questions) to ensure a different order in each game.
   * The function then iterates through each question in the shuffled list.
     + For each question, the options are randomly shuffled.
     + The user is prompted to select the correct option by entering a number.
     + The user's input is compared with the correct answer, and the user is informed if their response is correct or incorrect.
     + The score (correct and incorrect answers) is updated accordingly.
3. Scoring:
   * The correct and incorrect answers are tracked using variables (correct\_answers and incorrect\_answers).
   * After all questions are answered, the final score is printed, including the number of correct and incorrect answers and the percentage of correct answers.
4. Play Again Loop:
   * The main part of the code is within the \_\_main\_\_ block.
   * The quiz is played in a loop, and after each round, the user is prompted if they want to play again (play\_again variable).
   * If the user enters "yes," the loop continues, and a new quiz is generated. If the user enters anything other than "yes," the loop breaks, and the program exits.
5. Input Validation:
   * The code assumes that the user enters a valid numerical option during the quiz. No input validation is performed in case the user enters a non-numeric or out-of-range option.
6. Randomization:
   * The use of random.shuffle() ensures that the order of questions and options is different in each quiz session.
7. Percentage Calculation:
   * The percentage of correct answers is calculated and displayed in the final score.

Overall, this code provides a basic structure for a quiz game, allowing users to answer multiple-choice questions with randomized options. It's a simple and interactive program suitable for learning purposes.

**The possible challenges faces when designing this quiz**

Designing a quiz application like the one provided in the code can involve various challenges. Here are some potential challenges and considerations:

1. **User Input Validation:**
   * The code assumes that the user will always enter a valid numerical option during the quiz. Adding input validation to handle non-numeric or out-of-range inputs would make the application more robust.
2. **Scalability:**
   * The code is designed for a fixed set of questions. If you want to scale the quiz with a dynamic set of questions, you need a mechanism to load questions from an external source, such as a file or a database.
3. **Question and Option Formatting:**
   * The code assumes simple text-based questions and options. If your quiz involves more complex content, such as images, multimedia, or formatted text, you would need a way to handle and display these elements.
4. **User Feedback and Interaction:**
   * The code provides basic feedback on whether an answer is correct or incorrect. Enhancing the user experience could involve providing more detailed feedback, explanations, or even hints based on the user's responses.