Capstone Project 1

Interactive Quiz Game in Python

Project Overview:

In this project, you will create an interactive quiz game using Python in groups. The game will present a series of questions to the player, track their score, and provide feedback on their performance. The player will be able to select their answers and receive immediate results. This project is a great way to practice Python programming concepts and engage users in an educational and entertaining game experience.

Objective:

The objective of the Interactive Quiz Game project is to create an engaging and educational quiz game using Python. The project aims to provide an interactive learning experience for players while practicing Python programming concepts.

Learning Outcomes:

By completing this project, learners will:

1. Gain hands-on experience in Python programming.
2. Learn to design and implement a quiz game with interactive features.
3. Understand how to handle user input and validate answers.
4. Practice working with data structures, such as lists or dictionaries, to store and manage quiz questions.
5. Develop skills in implementing loops, conditionals, and score tracking mechanisms.
6. Enhance their ability to provide feedback and display results based on user interaction.
7. Gain familiarity with optional enhancements such as timers, difficulty levels, and high scores.
8. Strengthen problem-solving and logical thinking skills through designing and implementing a functional game.

Key Features needs to implement:

1. Questions and Answers: Prepare a collection of questions and their corresponding answers. Each question should have multiple choices, with only one correct answer.
2. User Interaction: Allow the user to select their answer choice for each question. Provide clear instructions on how to input their answer (e.g., by typing the corresponding letter or number).
3. Scoring and Feedback: Keep track of the player's score throughout the quiz. Provide immediate feedback after each question, indicating whether their answer was correct or incorrect. Display the final score at the end of the quiz.
4. Timer (Optional): For an additional challenge, you can incorporate a timer for each question. Allow the player a limited amount of time to answer each question and display how much time they have remaining. (High level Challenge)
5. Difficulty Levels (Optional): Implement different difficulty levels for the questions. Assign varying point values to each question based on the difficulty level, providing a more dynamic scoring system.
6. High Scores (Optional): Store and display high scores to create a competitive element in the game. Allow players to compare their scores with previous players or themselves.

Implementation Help/Steps:

Step 1: Set up the project

Create a new Python file called "quiz_game.py" and open it in your preferred code editor.

Step 2: Prepare the quiz questions

Define a list of questions, each containing a question statement and multiple choices for answers.

Each question should have a correct answer associated with it.

Step 3: Display the instructions

Print a welcome message and instructions for the quiz game.

Step 4: Iterate through the questions

Loop through each question from the list prepared in Step 2.

Display the question statement and multiple choices to the user.
Prompt the user to select their answer.

Step 5: Check the answer.

Compare the user's answer with the correct answer for the current question.
Keep track of the user's score by awarding a point for each correct answer.

Step 6: Display the result.

Print the user's score at the end of the quiz.

Step 7: Challenge: Add a timer.

For an additional challenge, students can implement a timer for each question, allowing users a limited amount of time to answer.

Marking Scheme (out of 50):

1. Setting up the project: 10 marks
2. Displaying instructions and welcoming message: 10 marks
3. Looping through questions and obtaining user's answers: 10 marks
4. Checking the answers and calculating the score: 10 marks
5. Displaying the final score: 10 marks
6. Bonus (implementing timer)

Outline the steps/plan for your project:

Students should provide a clear explanation for each step of their code, describing the purpose and functionality of the code segment. They can add comments or write explanations as separate comments within the code. The explanation should demonstrate their understanding of the concepts used in the project.

Add the link to your project here:

Link to access project:

Deadline: Until Release of Project 2(One Week)