GENERAL KNOWLEDGE QUIZ GAME

PROJECT OVERVIEW

Project Title

General Knowledge Quiz Game

Project Description

A Python-based command-line quiz application that tests users on general knowledge across various categories including science, geography, history, and pop culture. The game features a robust question management system, real-time scoring, and an engaging user interface with immediate feedback.

GitHub Repository

GitHub Repo Link

TECHNICAL SPECIFICATIONS

Technologies Used

- **Programming Language:** Python 3.x
- **Libraries:**
  - `csv` - For reading question data
  - `random` - For question randomization
- **Data Format:** CSV (Comma-Separated Values)
- **Architecture:** Object-Oriented Programming (OOP)

Key Features

- **Randomized Question Order:** Questions shuffle each session for varied experience
- **Input Validation:** Only accepts valid responses (A/B/C/D)
- **Accurate Scoring:** Tracks only legitimate attempts
- **Early Exit Option:** Users can quit anytime by typing 'quit'
- **Case-Insensitive Matching:** Handles various input formats

- **Real-time Feedback:** Immediate correct/incorrect indicators
- **Comprehensive Question Bank:** 100 diverse general knowledge questions

System Requirements

- Python 3.6 or higher
- Any operating system (Windows, macOS, Linux)

IMPLEMENTATION DETAILS

Class Structure

```
QuizGame
├── __init__(filename)   # Initializes game with questions
├── load_questions()      # Static method to read CSV data
└── play()                # Main game loop and logic
```

Data Management

- **Source:** `general_knowledge_quiz.csv`
- **Format:** CSV with columns: question, option_a, option_b, option_c, option_d, answer
- **Size:** 100 questions across multiple categories
- **Validation:** Automatic column verification on load

Error Handling

The application includes comprehensive error handling:

- **File Not Found:** Graceful handling of missing CSV files
- **CSV Corruption:** Validation for malformed or incomplete data
- **Invalid Input:** User-friendly messages for incorrect inputs
- **Missing Columns:** Verification of required data structure

```
# Example of robust error handling
try:
    with open(filename, 'r', newline='', encoding='utf-8') as csvfile:
        # File operations
except FileNotFoundError:
    print(f"Error: Quiz file '{filename}' not found.")
    raise
```

CHALLENGES & SOLUTIONS

Challenge 1: Question Randomization

Problem: Ensure each game session presents questions in a different order to maintain engagement and prevent memorization patterns.

Solution:

Shuffle questions so they appear in random order each time

random.shuffle(self.questions)

Impact: Creates unique gameplay experience for each session, enhancing replay value.

Challenge 2: Accurate Score Tracking

Problem: Prevent invalid inputs from affecting the score calculation while maintaining user-friendly interaction.

Solution:

```
# Count only valid A/B/C/D inputs as attempts
if answer in option_map:
    self.attempts += 1
    # Process answer...
else:
    print("Invalid input. Please choose A, B, C, or D.\n")
    # Do not increment attempts for invalid input
```

Impact: Ensures scoring integrity while providing clear user guidance.

Challenge 3: Case-Insensitive Answer Matching

Problem: Users might input answers in various cases (upper, lower, mixed).

Solution:

```
# Compare selected option text with correct answer (case-insensitive)
if option_map[user_input].lower() == question['answer'].lower():
```

Impact: Improved user experience with flexible input acceptance.

Challenge 4: CSV Data Integrity

Problem: Ensure the question file contains all required columns and valid data.

Solution:

```
# Validate that required columns exist in CSV
required_columns = {'question', 'option_a', 'option_b', 'option_c', 'option_d', 'answer'}
if not required_columns.issubset(reader.fieldnames):
    missing = required_columns - set(reader.fieldnames)
    raise ValueError(f"CSV missing required columns: {missing}")
```

Impact: Prevents runtime errors due to malformed data files.

FUTURE ENHANCEMENT PLANS

Short-term Improvements

- Add difficulty levels (Easy, Medium, Hard)
- Implement category-based question selection
- Create a timer for each question

Medium-term Enhancements

- Develop a web interface using Flask/Django
- Add user authentication and score history
- Create an admin panel for question management
- Implement multiplayer functionality

Long-term Vision

- Mobile app development (React Native/Flutter)
- Integration with external question APIs
- AI-powered adaptive difficulty
- Global leaderboard system

INSTALLATION & USAGE

Setup Instructions

1. Ensure Python 3.6+ is installed
2. Save `quiz_game.py` and `general_knowledge_quiz.csv` in the same directory
3. Run: `python quiz_game.py`

Game Commands

- `A/B/C/D` - Select answer choice
- `quit` - Exit the game early
- Any other input - Prompts for valid input

LEARNING OUTCOMES

Technical Skills Demonstrated

- Object-Oriented Programming principles
- File I/O operations with CSV
- Data validation and error handling
- User input processing and sanitization
- Documentation and code organization

CONCLUSION

The General Knowledge Quiz Game represents a well-architected Python application that successfully addresses core programming challenges while delivering an engaging user experience. The project demonstrates strong software engineering principles and provides a solid foundation for future enhancements.

Document Version: 1.0

Last Updated: 27/11/2025

Author: Seyi Ajanaku