Word Puzzles Game in Python

Project Overview

The Word Puzzles Game is a console-based game where the player is presented with a jumbled version of a word and must guess the correct word. The game includes multiple levels with increasing difficulty, a scoring system, and the ability to track player progress.

Step 1: Setup the Game

1. **Initialize Word List:**
    - Create a list of words to be used in the game.
    - Include words of varying difficulty levels.
2. **Function: `initialize_words` Purpose:**
    - To initialize the list of words for the game.
3. **Description:**
    - Initializes a list of words that the game will use.
    - Categorizes words into different difficulty levels if desired.

Step 2: Jumble the Word

3. **Function: `jumble_word` Purpose:**
    - To create a jumbled version of a given word.
4. **Description:**
    - Takes a word as input and returns a new string with the characters of the word shuffled.

Step 3: Get Player's Guess

4. **Function: `get_player_guess` Purpose:**
    - To prompt the player to guess the correct word.
5. **Description:**
    - Prompts the player to input their guess for the jumbled word.
    - Validates the input to ensure it is a non-empty string.

Step 4: Provide Feedback

5. **Function: `provide_feedback` Purpose:**
    - To provide feedback on the player's guess.
6. **Description:**
    - Compares the player's guess to the correct word.
    - Informs the player if the guess is correct or incorrect.

Step 5: Update Score

6. **Function: `update_score` Purpose:**
    - To update the player's score based on their guess.
7. **Description:**
    - Increments the player's score for a correct guess.
    - May include a scoring system that varies points based on difficulty level or speed of guessing.

Step 6: Check for End of Game

7. **Function: `check_end_of_game` Purpose:**
    - To determine if the game should end.
8. **Description:**
    - Checks if there are no more words left to guess or if the player has met a specific condition to end the game.

Step 7: Display Score and Progress

8. **Function: `display_score` Purpose:**
    - To display the current score and progress of the player.
9. **Description:**
    - Prints the player's current score and any other relevant statistics.

Step 8: Main Game Loop

9. **Function: `play_game` Purpose:**
    - To manage the overall game flow and player interactions.
10. **Description:**
    - Initializes the game and player score.
    - Continuously jumbles words, prompts for guesses, provides feedback, and updates the score.
    - Ends the game when all words are guessed or another end condition is met.

Full Function Descriptions

Function: `initialize_words`

- Initializes a list of words that the game will use.
- Categorizes words into different difficulty levels if desired.

Function: `jumble_word`

- Takes a word as input and returns a new string with the characters of the word shuffled.

Function: `get_player_guess`

- Prompts the player to input their guess for the jumbled word.
- Validates the input to ensure it is a non-empty string.

Function: `provide_feedback`

- Compares the player's guess to the correct word.
- Informs the player if the guess is correct or incorrect.

Function: `update_score`

- Increments the player's score for a correct guess.
- May include a scoring system that varies points based on difficulty level or speed of guessing.

Function: `check_end_of_game`

- Checks if there are no more words left to guess or if the player has met a specific condition to end the game.
- Returns `True` if the game should end, otherwise `False`.

Function: `display_score`

- Displays the player's current score and any other relevant statistics.

Function: `play_game`

- Manages the overall game flow and player interactions.
- Initializes the game and player score.
- Continuously jumbles words, prompts for guesses, provides feedback, and updates the score.
- Ends the game when all words are guessed or another end condition is met.

Implementation Tips

1. **Word List:** Use a list to store words and optionally categorize them by difficulty.
2. **Randomization:** Use Python's `random` module to shuffle the characters in the word.
3. **Input Validation:** Ensure that the player's input is valid and handle invalid input gracefully.
4. **Scoring System:** Implement a scoring system that rewards correct guesses and optionally includes bonuses for difficulty or speed.
5. **Game Loop:** Use a loop to manage the game flow, ensuring the player is continuously prompted until the game ends.
6. **User Feedback:** Provide clear and immediate feedback to the player on each guess.