A Quiz Game (Scramble Strike)

Final project for completing *Basic Programming with Python* in EDGE program at Department of Information and Communication Technology, Mawlana Bhashani Science and Technology University.

<u>Submitted by:</u>
Md. Masud Nabi
Batch BPP-22
Basic Programming with Python in EDGE program at Department of Information and Communication Technology, Mawlana Bhashani Science and Technology University.

Overview of the Project

This is a text-based, two-player competitive puzzle game written in Python. The core objective is to unscramble English words correctly to "conquer" cells in a 3x3 grid. The game combines vocabulary skills with reaction speed, using a precision timing system to penalize slow answers. The game continues until the grid is full or a strict turn limit is reached.

Handbook

In **Scramble Strike** game, two players compete to dominate a 3x3 grid by correctly unscrambling vocabulary words under strict time pressure. Players take alternating turns where they are presented with a scrambled word ranging from 4 to 7 characters, with the difficulty automatically increasing as more of the matrix is conquered. The defining rule is the "Instant Strike" timer: the clock begins the exact moment the word appears, with no separate start signal, meaning players must read, process, and type their answer immediately to minimize their time log. A correct answer claims the next available cell in the grid and awards 10 points, while an incorrect answer forfeits the turn. The game concludes when all 9 cells are filled or the 27-turn limit is reached, and the winner is determined by a unique efficiency score that subtracts a time-based penalty from the total points earned, ensuring that the fastest and most accurate player prevails.

Initialization and Data Structures

The Matrix: The game board is represented as a 3x3 list of lists. Initially, all cells are filled with placeholders (`_`). As players win turns, these placeholders are replaced by the player's marker (`P1` or `P2`).

Player Database: A dictionary stores the state for Player 1 and Player 2. It tracks four specific attributes for each player (Name: Display name; Mark: The symbol used in the matrix; Cells: The count of successful unscrambles; Time: The cumulative total of seconds spent thinking and typing across all turns.)

Vocabulary Library: The program processes a hardcoded block of raw text containing common English words. It cleans this text (stripping whitespace, converting to lowercase) and filters it to create a list of unique words that are strictly between 4 and 7 characters in length.

Adaptive Difficulty System: The game features a dynamic difficulty curve that scales based on the progress of the match. The program checks how many cells in the matrix are currently filled to determine the length of the next word. Early games (0–2 cells filled) of the program selects words with 4 or 5 characters. Mid games (3–5 cells filled) contain words with 5 or 6 characters. Late game (6–8 cells filled) selects words with 6 or 7 characters.

The Turn Logic: This is the central interaction loop, designed to measure reaction time accurately.

Word Preparation: A target word is selected based on the current difficulty. The characters are shuffled randomly. A check ensures the "scrambled" version is not identical to the original word.

The Ready Phase: The program displays the player's name and issues a warning. It then forces a 1.5-second pause (`time.sleep`) to allow the player to mentally prepare.

Automatic Timer: The unique feature of this program is the timer placement. The system captures the `start_time` immediately after capturing the time, the `input()` prompt appears with the scrambled word. The user types their answer and hits Enter. The system captures the `end_time` immediately after the Enter key is pressed. The duration (`end_time - start_time`) is calculated and added to the player's cumulative time log.

Matrix Progression: If a player inputs the correct word, they are awarded a "cell." The matrix fills sequentially rather than by player choice. The logic calculates the position using integer division and modulus on the `filled_cells` counter, where row: calculated as `count // 3` and column: calculated as `count % 3`. This ensures the grid fills from top-left to bottom-right (Row 1, then Row 2, then Row 3).

Winning and Scoring Algorithm: When the game ends, the winner is not decided solely by who answered the most questions, but by an efficiency formula. Every cell won is worth 10 points (Matrix Points). The total time (in seconds) the player used throughout the entire game is divided by 50 (Time Penalty). Final Score = (Cells Won × 10) - (Total Time / 50) 9 (Final Equation). This creates a scenario where a player with fewer cells could theoretically beat a player with more cells if the second player was significantly slower in typing their answers.

Termination Conditions: The main game loop runs continuously until one of two conditions is met. The `filled_cells` counter reaches 9, meaning the board is full (Matrix Saturation). The `total_turns_played` counter reaches 27. This acts as a "fail-safe" to prevent an infinite loop if players consistently fail to unscramble words (Turn Exhaustion).