Description:

2048 is a single-player sliding block puzzle game. The objective is to slide numbered tiles on a grid to combine them and create a tile with the number 2048.

Features to Implement:

- Display the 4x4 grid with numbered tiles.
- Allow the player to slide tiles in four directions (up, down, left, right).
- Combine tiles with the same number when they collide.
- Handle game-over conditions when no more moves are possible.

Step by Step Guide

1. `initialize_grid`

Purpose:

- To create and set up the initial 4x4 game grid for the 2048 game.

Description:

- Initializes a 4x4 grid filled with zeros.
- Adds two new tiles, either 2 or 4, at random empty positions in the grid.

2. `add_new_tile`

Purpose:

- To add a new tile (either 2 or 4) at a random empty position on the grid.

Description:

- Identifies all empty positions (cells with a value of 0) in the grid.
- Selects a random position from these empty positions.
- Places a new tile with a value of either 2 or 4 in the selected position.

3. `print_grid`

Purpose:

- To display the current state of the game grid to the console.

Description:

- Iterates through each row of the grid.
- Prints each row of the grid in a formatted manner to ensure readability.

4. `get_user_input`

Purpose:

- To obtain the next move from the player.

Description:

- Prompts the player to enter a move (W for up, A for left, S for down, D for right).
- Validates the input to ensure it is one of the accepted commands.
- Returns the validated move input.

5. `slide_and_merge_row_left`

Purpose:

- To slide and merge a single row of tiles to the left.

Description:

- Removes all zeros from the row to simulate the sliding of tiles.
- Merges adjacent tiles with the same value by summing them and placing a zero in the merged tile's position.
- Repeats the process to ensure all possible merges are completed.
- Returns the updated row, with zeros added to the end to maintain the row length.

6. `move_left`

Purpose:

- To move and merge all tiles in the grid to the left.

Description:

- Applies the `slide_and_merge_row_left` function to each row of the grid.

7. `move_right`

Purpose:

- To move and merge all tiles in the grid to the right.

Description:

- Reverses each row of the grid.
- Applies the `slide_and_merge_row_left` function to each reversed row.
- Reverses each row back to its original order after merging.

8. `move_up`

Purpose:

- To move and merge all tiles in the grid upwards.

Description:

- Treats each column of the grid as a row.
- Applies the `slide_and_merge_row_left` function to each column.
- Updates the grid with the merged columns.

9. `move_down`

Purpose:

- To move and merge all tiles in the grid downwards.

Description:

- Treats each column of the grid as a row.
- Reverses each column.
- Applies the `slide_and_merge_row_left` function to each reversed column.
- Reverses each column back to its original order after merging.
- Updates the grid with the merged columns.

10. `check_win`

Purpose:

- To check if the player has won the game.

Description:

- Iterates through each row of the grid.
- Checks if any tile has reached the value 2048.
- Returns `True` if a 2048 tile is found, indicating a win.

11. `check_game_over`

Purpose:

- To check if the game is over (i.e., no more valid moves are possible).

Description:

- Iterates through each cell in the grid.
- Checks if there are any empty cells (value of 0).
- Checks if adjacent tiles can be merged (i.e., have the same value).
- Returns `True` if no valid moves are possible, indicating game over.

12. `play_game`

Purpose:

- To manage the overall game loop, handling the game flow and user interactions.

Description:

- Initializes the game grid.
- Continuously displays the grid, handles user input, and updates the grid based on moves.
- Checks for win or game over conditions after each move.
- Ends the game loop if the player wins or the game is over.