Project Report: Sudoku Game in Python

Introduction

The Sudoku game project is a Python-based desktop application that provides users with a fully interactive Sudoku puzzle. The game features a graphical user interface (GUI) built using the Tkinter library, allowing players to generate new Sudoku puzzles, solve them, and receive instant feedback on their progress. The program also includes functionality to verify the correctness of user input and provide a solution to the puzzle.

Objective

The primary objective of this project was to create a user-friendly Sudoku game with the following features:
- Ability to generate new Sudoku puzzles.
- An interactive GUI for users to solve the puzzle.
- Validation of user input against Sudoku rules.
- A solver feature to automatically solve the puzzle.
- Feedback mechanism to inform users when they have successfully completed the puzzle.

Project Structure

The Sudoku game consists of several key functions and components, all integrated into a Tkinter-based GUI application.

Key Functions

1. get_contents():
   - Retrieves the current state of the Sudoku board, returning it as a list of lists representing rows and columns.

2. on_solve_game(e):
   - Solves the Sudoku puzzle and updates the GUI with the solution. Cells with provided values are locked to prevent further editing.

3. on_new_game(e):
   - Generates a new Sudoku puzzle, populates the board, and locks the cells with the generated numbers.

4. is_solved(output):
   - Checks if the puzzle is solved correctly by validating rows, columns, and 3x3 boxes.

5. on_focus_in(e) and on_focus_out(e):
   - Manage the highlighting of the selected cell and validate user input when focus shifts away from the cell.

6. on_key_press(e):
  - Handles user key presses to ensure only valid inputs are entered into the cells.

7. is_valid(r, c, num):
  - Validates whether placing a number in a specific cell is permissible according to Sudoku rules.

8. solve(input):
  - Implements a backtracking algorithm to solve the Sudoku puzzle.

9. generate_input():
  - Generates a new, solvable Sudoku board.

10. populate_entries():
  - Populates the Sudoku grid with Entry widgets for user interaction.

11. populate_actions_frame():
  - Populates the action buttons such as "Erase", "New Game", and "Solve the board".

12. add_message_frame():
  - Adds a message frame to the GUI to display feedback messages to the user.

Graphical User Interface (GUI)

The GUI is designed using the **Tkinter** library, which provides a simple and effective way to create desktop applications in Python.

Sudoku Grid: A 9x9 grid where each cell is represented by a Tkinter Entry widget. Users can enter numbers, and the grid responds to user interactions, such as cell selection and validation.

Action Buttons: Buttons are provided for user interaction:
Number Buttons (1-9): Allows users to input numbers into the selected cell.
Erase Button: Clears the content of the selected cell.
New Game Button: Generates a new Sudoku puzzle.
Solve the Board Button: Automatically solves the current puzzle.

Message Frame: Displays feedback, such as congratulating the user upon solving the puzzle.

Algorithm

The Sudoku solver uses a backtracking algorithm, a recursive method that attempts to fill each cell with a valid number and backtracks when an invalid state is encountered. The puzzle generation is done by first filling the middle 3x3 grid and then solving the rest, followed by removing numbers while ensuring the puzzle remains solvable.

Validation and Feedback

The program continuously validates the user's input to ensure compliance with Sudoku rules. If an invalid entry is made, the corresponding cell is highlighted in red. Upon completing the puzzle correctly, the user is notified with a congratulatory message.

Challenges and Solutions

Sudoku Puzzle Generation: Ensuring the puzzle is both challenging and solvable required careful balancing of number removal while maintaining a single solution. The implementation carefully removes numbers while verifying that the puzzle remains solvable.

Input Validation: Handling real-time input validation was challenging, especially ensuring that incorrect entries were immediately highlighted without interrupting the flow of the game.

Conclusion

The Python-based Sudoku game is a fully functional application that demonstrates the capabilities of Tkinter for building interactive GUI applications. The game is easy to use, visually appealing, and provides a robust platform for playing and solving Sudoku puzzles. Further enhancements could include adding different difficulty levels, implementing a timer, and integrating more advanced solving techniques.

Future Work

Difficulty Levels: Implementing varying levels of difficulty to cater to a wider range of players.
Timer: Adding a timer to track the time taken to solve the puzzle.
Hints Feature: Providing hints to players when they are stuck.
Mobile Version: Developing a version of the game for mobile platforms using a framework like Kivy.

References

- Tkinter Documentation: https://docs.python.org/3/library/tkinter.html
- Sudoku Solving Algorithms: https://en.wikipedia.org/wiki/Sudoku_solving_algorithms