Sudoku Generator and Solver

~by Rohit Karnawat

Problem Statement:

Solving Sudoku has been a difficult task. The goal has been to apply a backtracking algorithm to minimize computing times. There are several different Sudoku variations available right now, including 4X4 grids, 9X9 grids, and 16X16 grids. The 9X9 board's traditional and classic Sudoku are the main subject of this project. The objective is to fill in the vacant cells in a partially filled 9x9 2D array with digits (from 1 to 9), so that every row, column, and submatrix of size 3x3 includes exactly one occurrence of the digits from 1 to 9.

Algorithm description:

For generating a valid Sudoku,

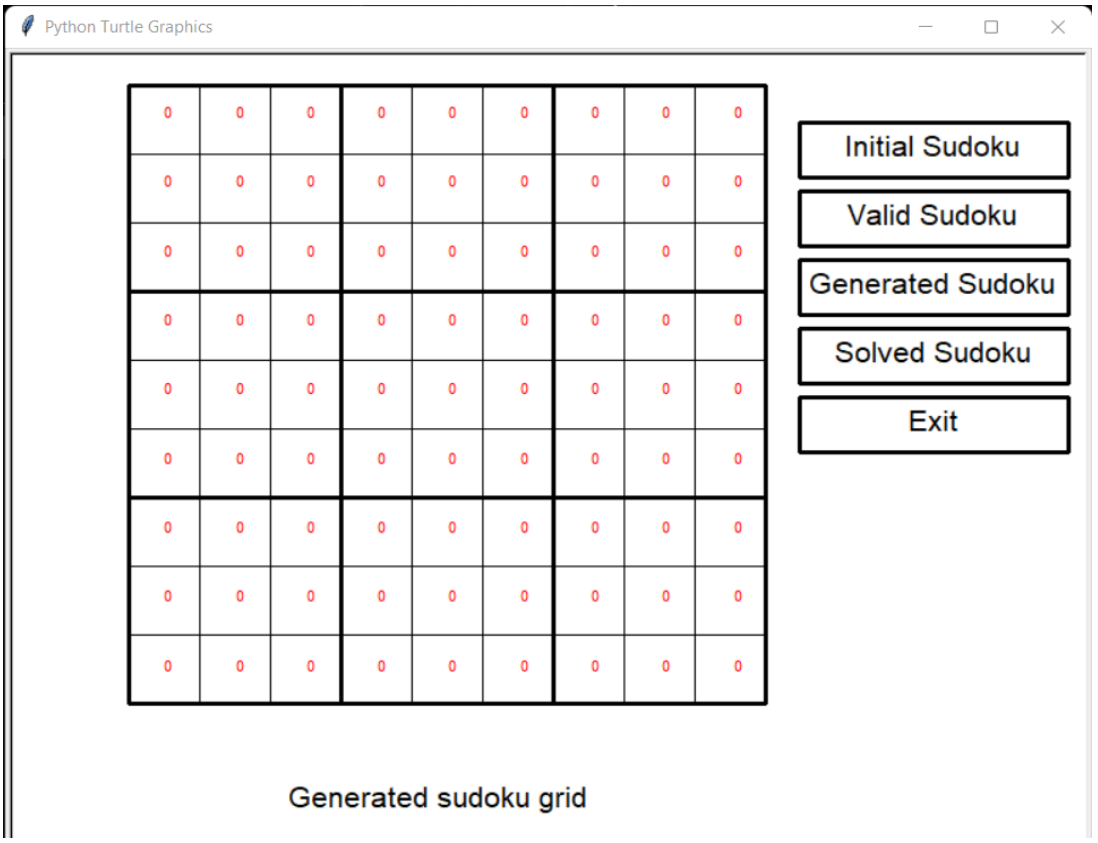
1. Initialize the 2d array of 9x9 elements with 0.
2. Start filling the diagonal element first as they are independent of other elements and will have to do 3x3 matrix checks only.
3. Now fill the non-diagonal elements by checking repetition in row, column and 3x3 matrix.
4. If it fails then backtrack i.e. undo the changes and try again by recursively calling the function again.
5. Once a valid sudoku is generated randomly select elements and make it 0. Thus a valid sudoku is generated.

For solving Sudoku,

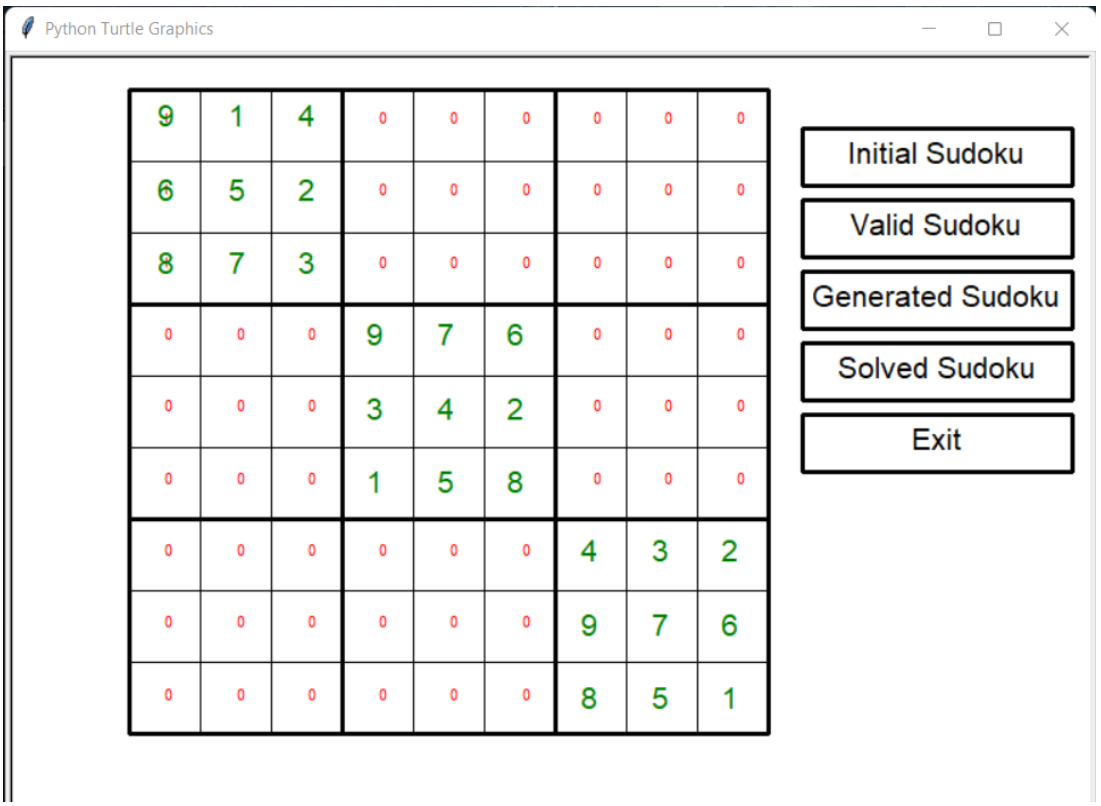
1. Find the empty element i.e element with 0 in 2d array.
2. If no empty element is found then stop i.e sudoku is solved.
3. Else sequentially fill the elements from 1 to 9 by checking the row, column and 3x3 matrix condition.
4. If it fails then backtrack i.e. undo the changes and try again by recursively calling the function again.

Thus a valid sudoku is generated and solved using backtracking algorithm

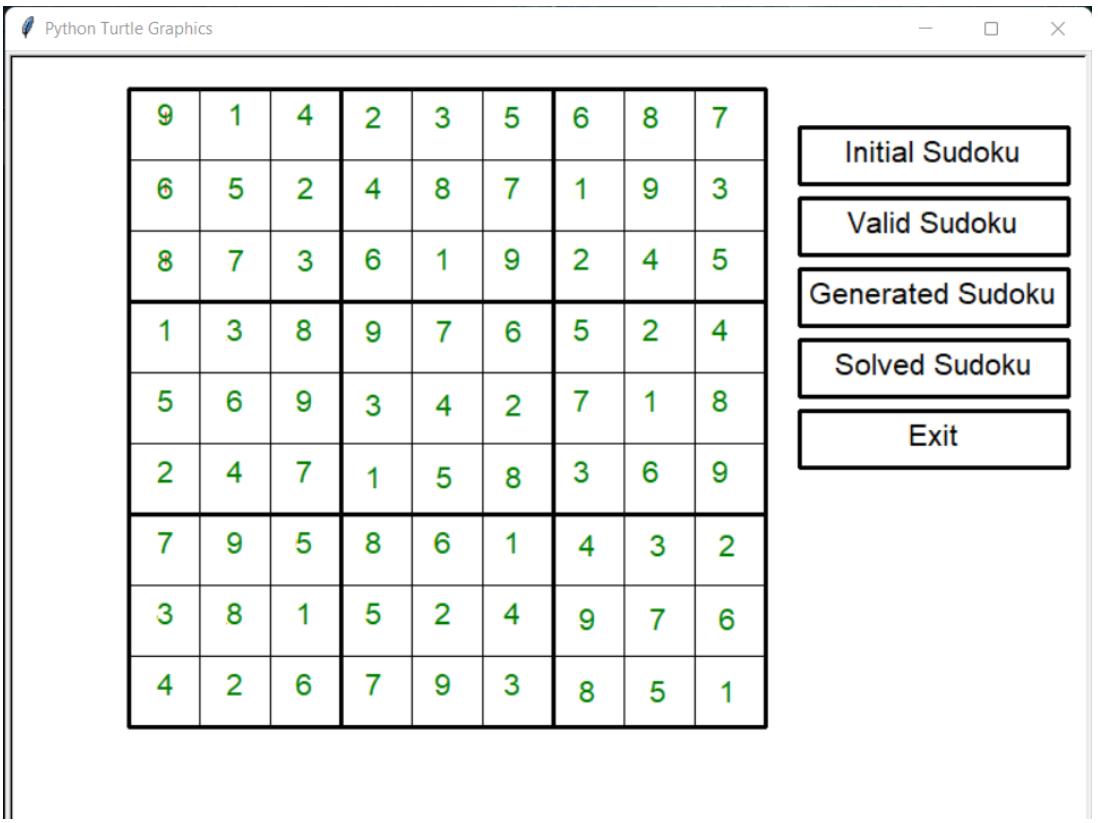
Results and discussion:



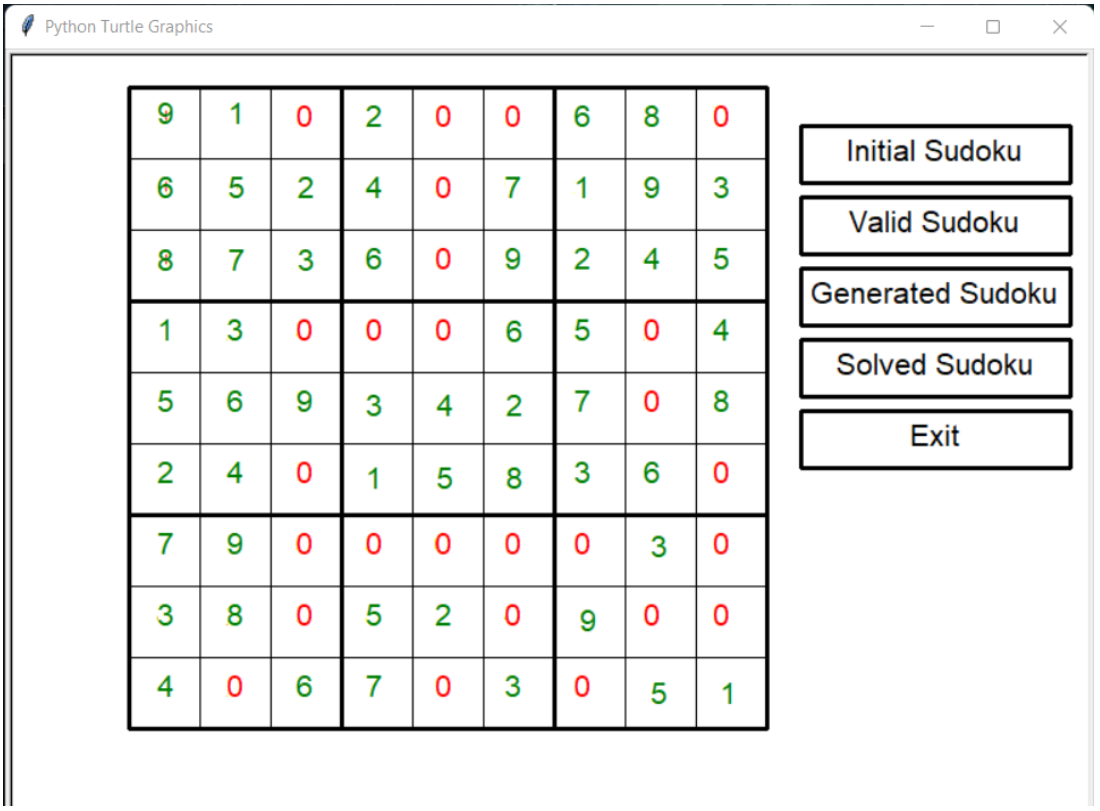
a.Generate sudoku grid



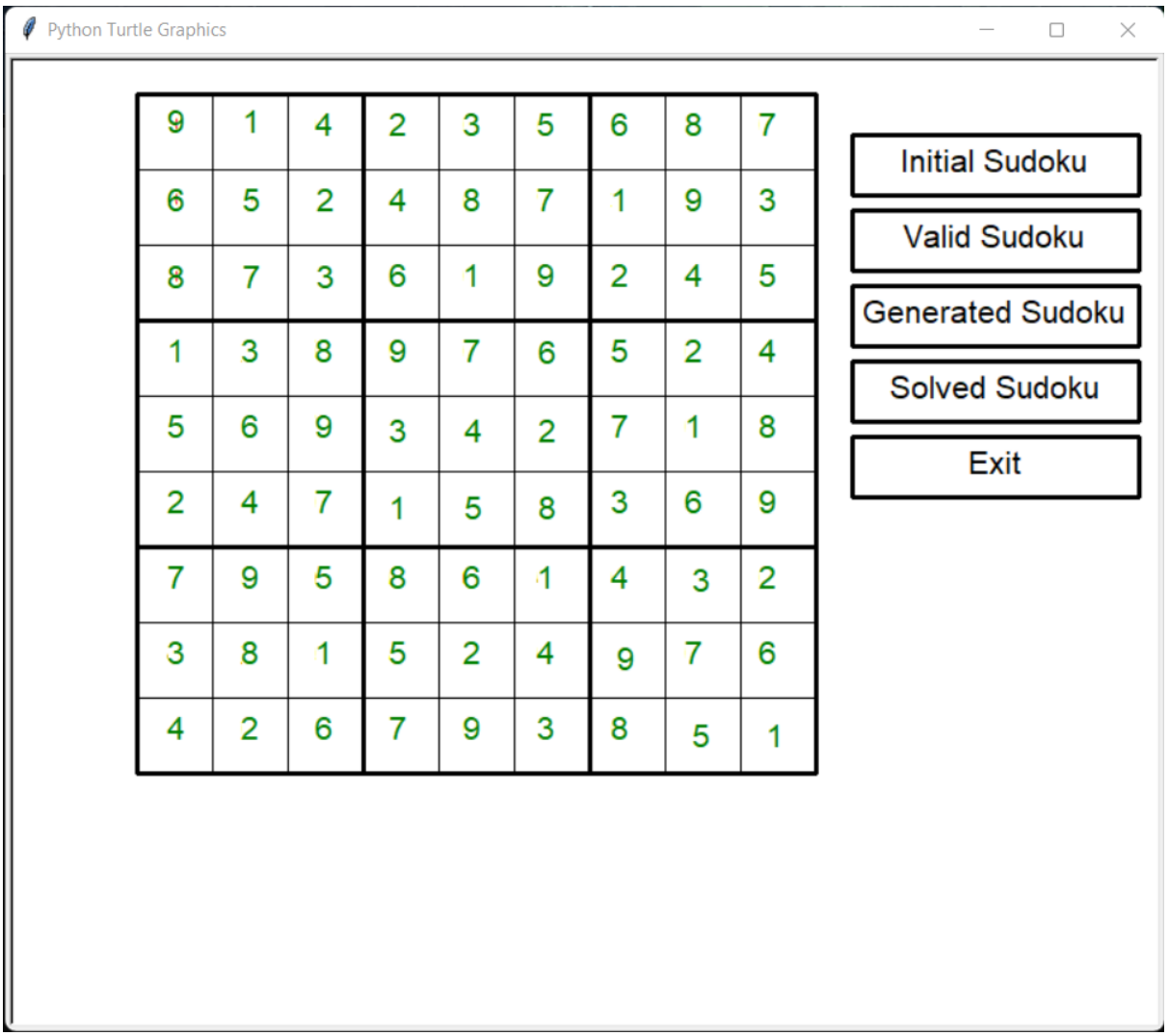
b. Fill diagonal elements



c. Valid Sudoku



d. Generate Sudoku



e. Solved Sudoku

After each step the program will print the output in the console; Also after solving the sudoku one can regenerate the sudoku puzzle by clicking on the generated sudoku. There are few UI related bugs in the program as it does not actually remove elements since it will remove all the elements from the screen thus I just placed a small white box above the previous elements, thus sometimes this method leaves small parts of previous number visible for e.g. first three elements in the first column.

Time complexity: The backtracking algorithm requires a time complexity of $O(9^m)$ in the worst case, where m is the number of unfilled cells in the Sudoku. This is because there are 9 ways to explore each unfilled cell.

References:

- https://en.wikipedia.org/wiki/Sudoku_solving_algorithms
- https://www.geeksforgeeks.org/backtracking-introduction/?ref=lbp
- https://www.simplilearn.com/tutorials/data-structure-tutorial/backtracking-algorithm
- https://docs.python.org/3/library/turtle.html
- https://www.youtube.com/watch?v=JzONv5kaPJM