**Topic 6.4: Sudoku Puzzle Solver**

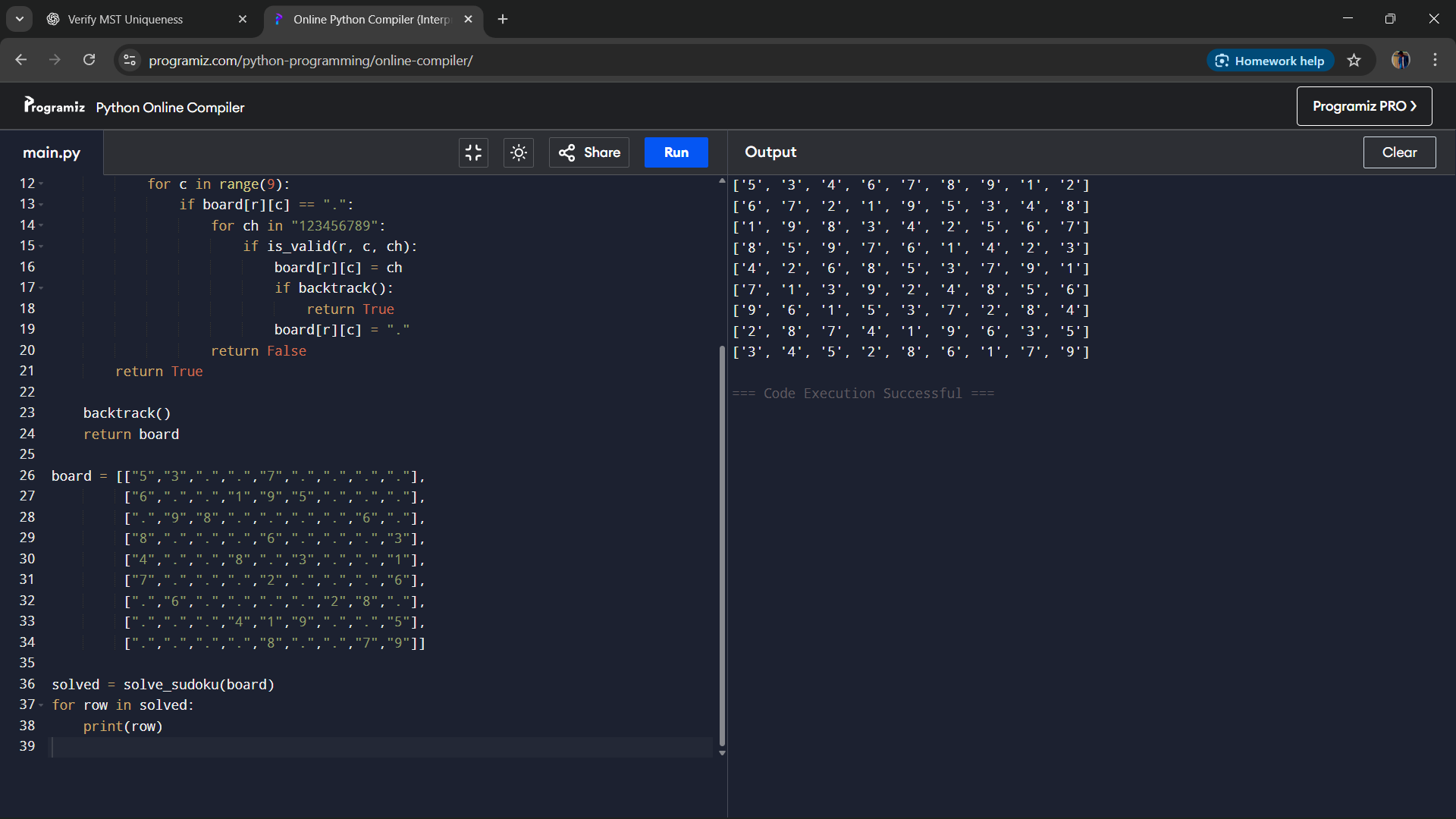
**Question**  
Write a program to solve a Sudoku puzzle by filling the empty cells.  
A Sudoku solution must satisfy all of the following rules:

* Each of the digits 1–9 must occur exactly once in each row.
* Each of the digits 1–9 must occur exactly once in each column.
* Each of the digits 1–9 must occur exactly once in each of the 9 sub-boxes of the grid.  
  The '.' character indicates empty cells.

**Aim**  
To implement a backtracking algorithm to solve a partially filled Sudoku board, ensuring all Sudoku rules are satisfied.

**Algorithm**

1. Start with the given Sudoku board.
2. Locate the first empty cell.
3. Try placing digits 1–9 in that cell.
4. For each placement, check validity:
   * Not already in the same row.
   * Not already in the same column.
   * Not already in the corresponding 3×3 sub-box.
5. If valid, place the number and continue recursively to solve the next empty cell.
6. If no number is valid, backtrack and try a new value in the previous cell.
7. Continue until all cells are filled, producing the solution.

**Output**

**Result**  
The Sudoku puzzle was solved successfully using backtracking and constraint checking.

**Performance Analysis**

* Time Complexity: O(9^n) in the worst case, where n is the number of empty cells.
* Space Complexity: O(1) for the fixed-size board and O(n) for the recursion stack.