

**DEPARTEMENT OF COMPUTER SCIENCE**

**Course code: INT 404**

**Topic: Sudoku Problem Solver**

**SUBMITTED TO: MS. POOJA RANA**

**SUBMITTED BY:**

|  |  |  |
| --- | --- | --- |
| **NAME** | **REGISTRATION NUMBER** | **ROLL NUMBER** |
| **DIVYENDU JHA** | **11802112** | **64** |
| **TANISHA GUPTA** | **11807614** | **65** |
| **PRITHVI BUDHIRAJA** | **11805252** | **66** |

**CONTENT**

**1. ABSTRACT**

**2. INTRODUCTION**

**3. LITERATURE REVIEW**

**4. PROPOSED METHODOLOGY**

**5. CODE**

**6.RESULT AND DISCUSSION**

**7. CONCLUSION**

**8.BIBLIOGRAPHY**

**ABSTRACT**

The purpose of this project is to successfully demonstrate and create the production system Sudoku solver simulator. In order to fulfil, our strategies we implemented the concept of backtracking which comes under constraint satisfaction problem. Firstly, we require the production system to be maintained and control strategy to be fired. The Sudoku is solved using python language.

**INTRODUCTION**

Sudoku is the Japanese word for “single numbers”, and refers to numerical puzzle game that has become popular in newspapers and game publications all over the world. It comes under brain twisting exercises, and is a great game to solve in order to develop logical thinking skills.

Sudoku is a 9x9 matrix and each square in the grid consist of 3x3 matrices. The goal is to fill all the rows, columns and 3x3 matrices called mini-grid with numbers ranging from 1 to 9. The squares of the board already have some numbers filled and the rest have to be filled by the player (in this case, by the system).

A production system is a computer program typically used to provide some form of artificial intelligence, which consists primarily of a set of rules about behaviour and the mechanism necessary to follow those rules as the system responds to states of the world.

**LITERATURE REVIEW**

For solving the problem of sudoku we have certain rules to implement on it which are as follows:

1. Each of the digits from 1 to 9 must occur exactly once in each row.

2. Each of the digits from 1 to 9 must occur exactly once in each column.

3. Each of the digits from 1 to 9 must occur exactly once in each of the squares of the 3x3 matrix.

Sudoku Solver uses the basic concept of Backtracking to solve every piece of block in the puzzle. Backtracking is implemented using recursion. The best part of this recursion magic is that we will do the code for the first time and rest part is left to the recursive function.

The program places the first number in the first empty cell. If the choice is compatible with the existing clues, it continues to the second empty cell, where it places some other relevant number (in some other row, column, and mini-grid). When it encounters a conflict (which can happen very quickly), it erases the number just entered and inserts some other one or, if that is invalid, the next number is entered. After placing the first legal number possible, it moves to the next cell and starts again with a relevant digit (or a minimum possible acceptable value). If the number that has to be altered is a 9, which cannot be raised by one in a standard 9×9 Sudoku grid, the process backtracks and increases the number in the previous cell (or the next to the last number placed) by one. Then it moves forward until it hits a new conflict. In this way, the program may sometimes backtrack.

**PROPOSED METHODOLOGY**

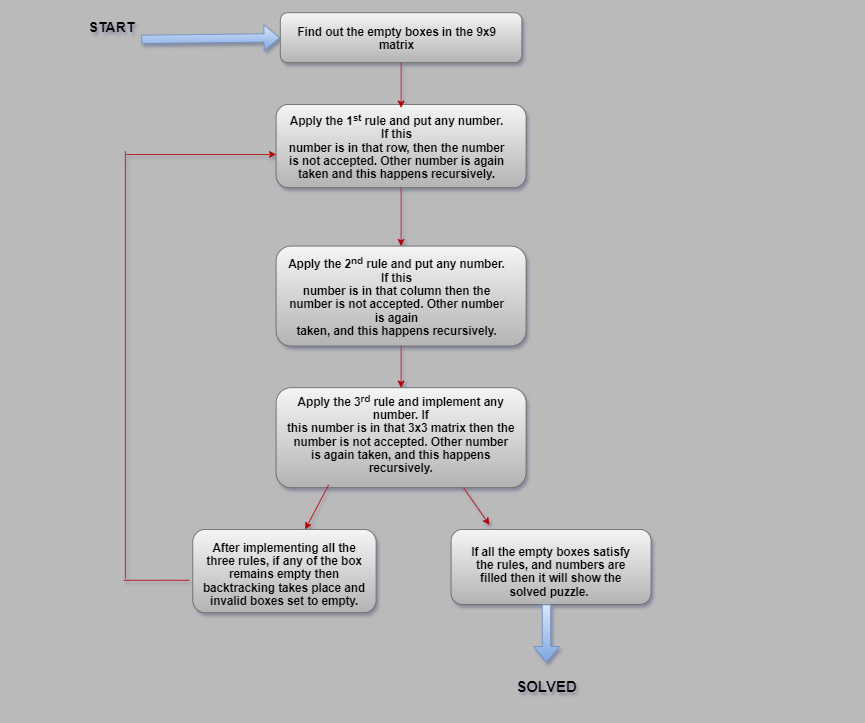
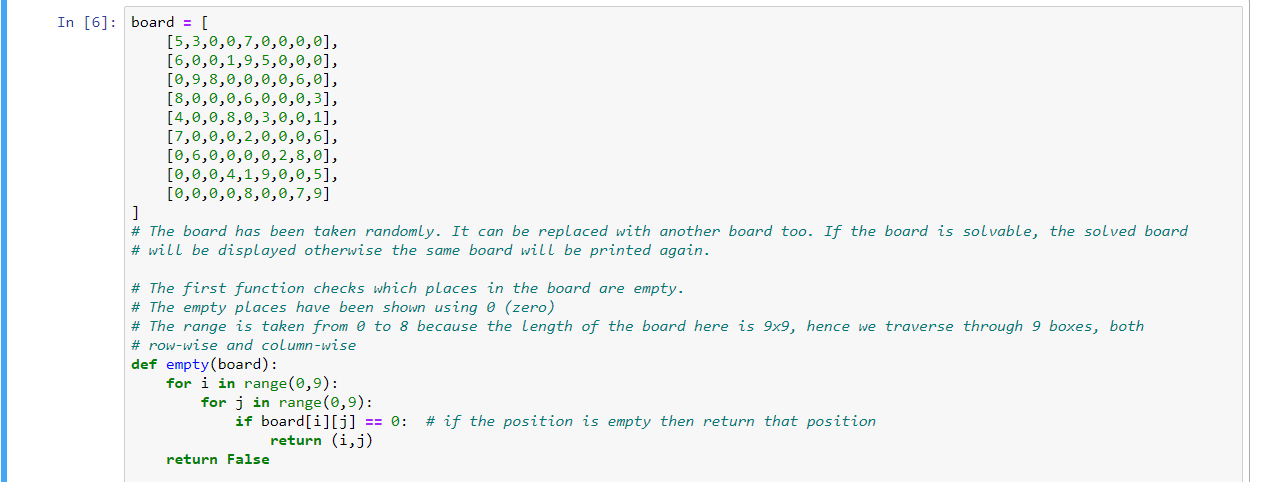
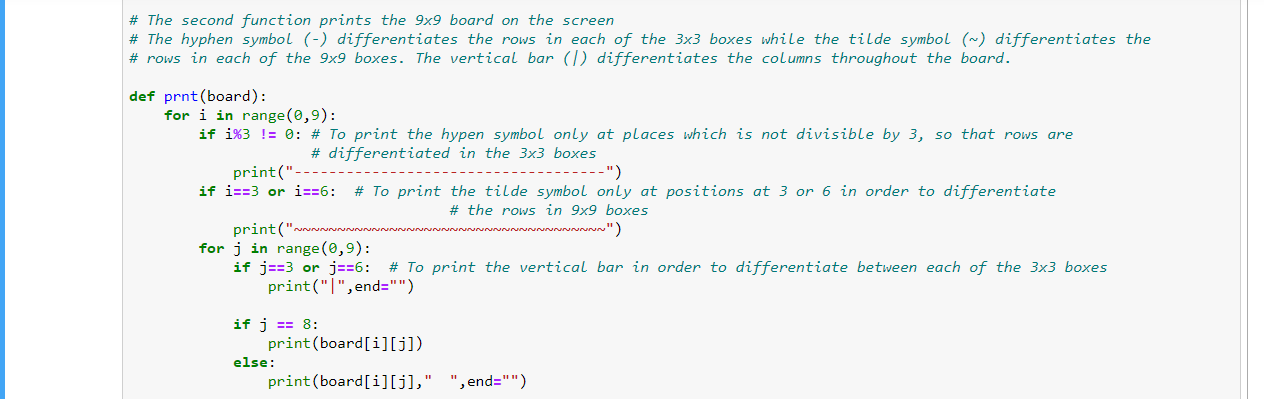
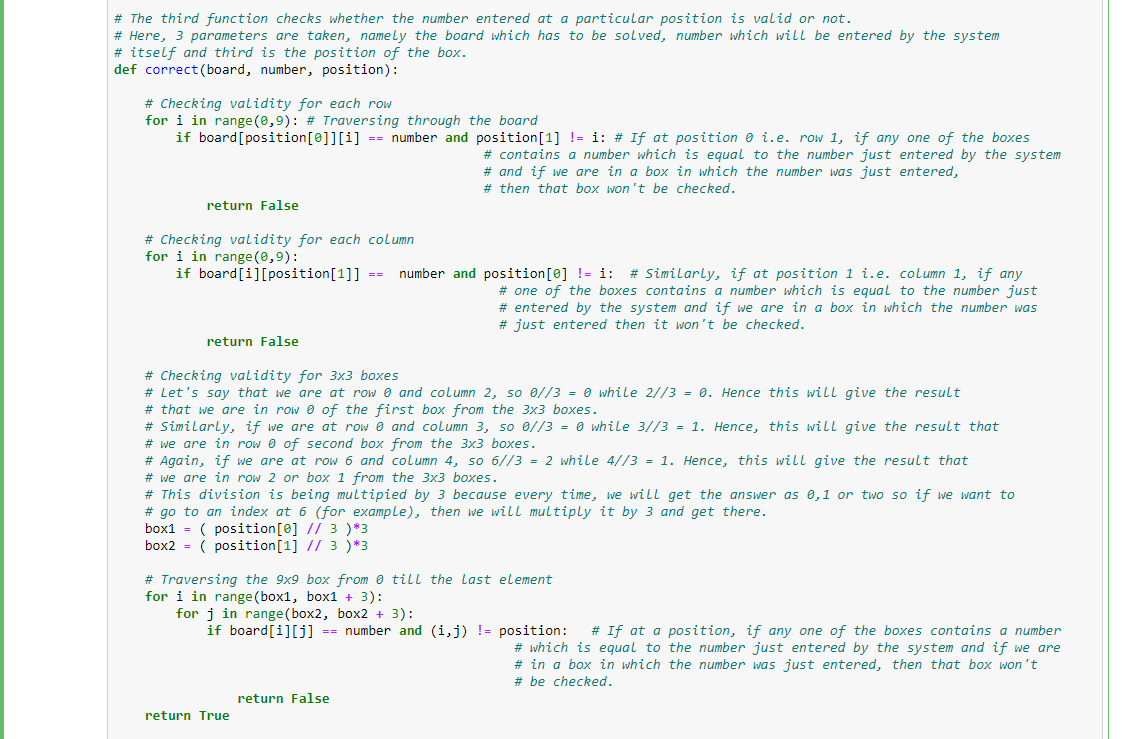


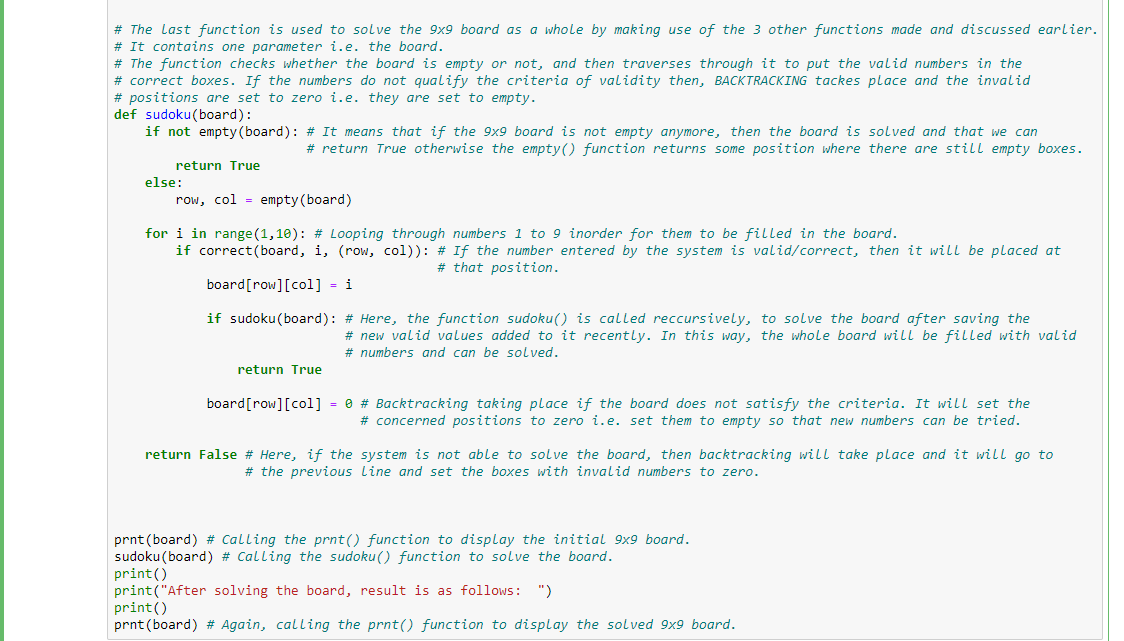
Fig. 1 Production system and control strategy

**CODE:**

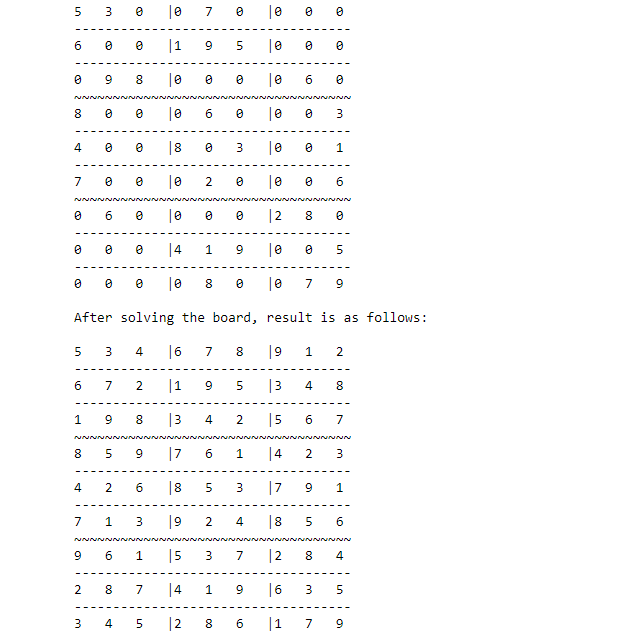








**RESULT AND DISCUSSION**



Here, we have taken a board in which some of the numbers were already filled.

By following the rules of the sudoku implemented in the code, the output came in which we can see the numbers are from 1 to 9 in the 3x3 matrix and every column and rows have 1 to 9 without any repetition.

**CONCLUSION:**

We finally reach to a conclusion that this method is actually efficient when it comes to solving Sudoku puzzles. The best part of this code is the number of lines needed to implement the logic which is very feasible. The implementation could likely be improved to make it work faster. Python programming is feasible to get the solution of Sudoku.

**REFERENCES:**

1. <https://en.wikipedia.org/wiki/Sudoku>
2. Research paper on Sudoku: <http://www.ijsrp.org/research-paper-0513/ijsrp-p1735.pdf>
3. Backtracking <https://en.wikipedia.org/wiki/Backtracking>
4. Constraint satisfaction problem: <https://medium.com/my-udacity-ai-nanodegree-notes/solving-sudoku-think-constraint-satisfaction-problem-75763f0742c9>