Odd Semester (2024)



**BINUS UNIVERSITY**



**BINUS INTERNATIONAL**



**Assignment Cover Letter**

**(Individual Work** **)**

**Student Information**: **Surname**  **Given Names Student ID Number**

1. Alvin Nathaniel 2440042430

**Course Code**  **:** COMP6056   **Course Name**  **:** Program Design Methods

**Class**  **:** L1AC   **Name of Lecturer(s)**  **:** Jude Joseph Lamug Martinez

**Major**  **:** Computer Science

**Title of Assignment** : Sudoku Solver

**Type of Assignment**  **:** Final Project

**Submission Pattern**

**Due Date**  **:** 13-01-2021   **Submission Date**  **:**

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer’s instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

**Plagiarism/Cheating**

BiNus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

**Declaration of Originality**

By signing this assignment, I understand, accept, and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student: (Name of Student)

Nathaniel Alvin

**“Sudoku Solver”**

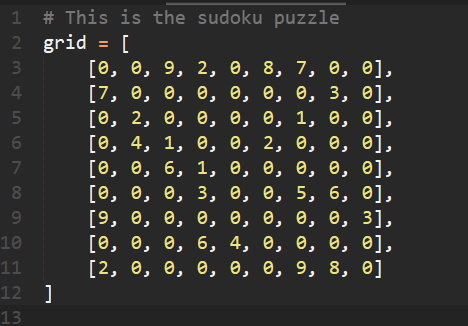
**Name : Nathaniel Alvin**

**ID : 2440042430**

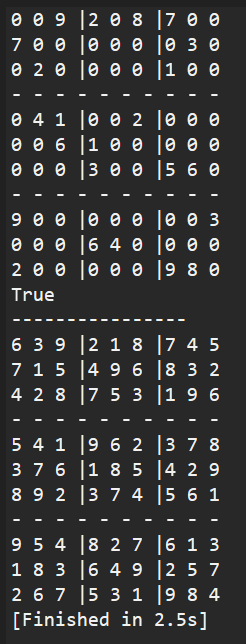
1. **Program Description**

This application allows people to solve sudoku puzzles within seconds. It uses backtracking algorithm to deal with the problem. For now, we have to manually input the puzzle to the code.

1. **How the program works**



As said before, the puzzle needs to be manually inputted to the code. 0 means blank space.

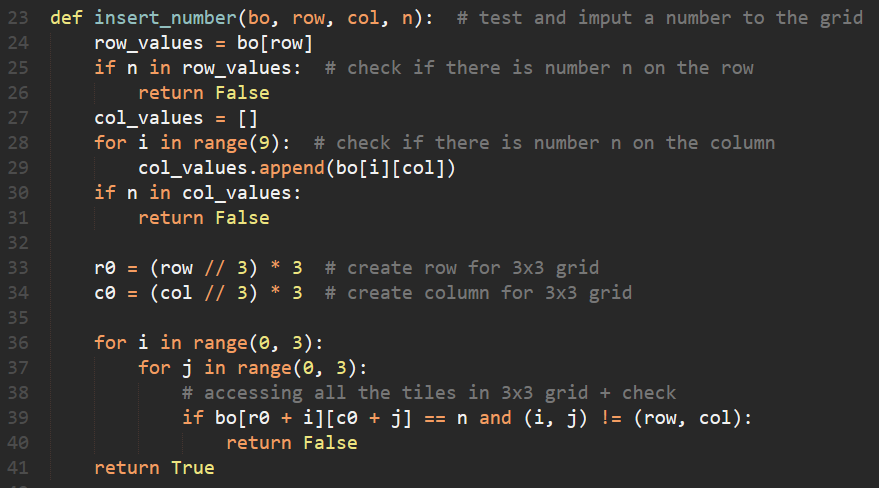


Then, run the program. The unsolved puzzle will be printed on top and the solution under it. True means the puzzle is solvable, and False means it is not.

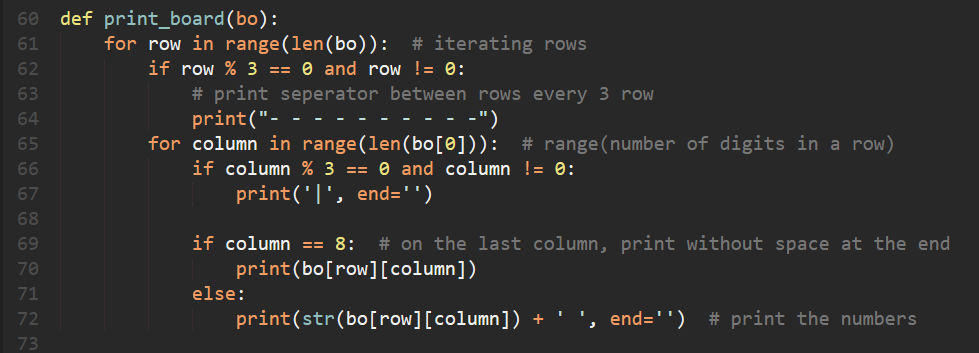
1. **Lessons that have been learned**

Python is a new language for me therefore there is much room for improvement. This project teaches me a new algorithm called backtracking and how to implement it using python. It has been a fun project as every trial and error teaches me new things. Not only the algorithm, but the function to print the puzzle is also a bit tricky for me and it definitely improves my problem-solving skill.

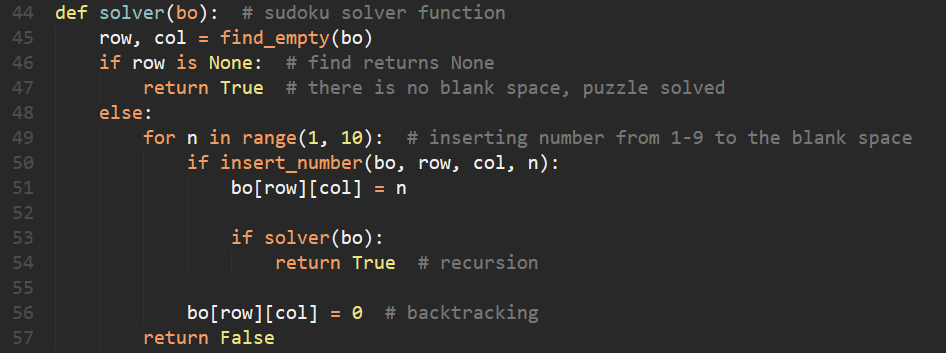
1. **Code explanation**

****

This insert\_number function is to check whether it is possible to input a number into the blank space. It will return True if the number is possible and False if not possible. This function takes 4 arguments. Bo which is short for board, row and col or the position of the blank space and n which is the number we want to input. First, it will check if the number inputted is within the row and column. By using the in statement, this is very possible. For the column however, I need to append the values in the row into a list to make it easier. In a sudoku puzzle, there is a 3x3 grid or so-called sub-grid. Besides the row and column, each number can only appear once inside this sub-grid. To make this sub-grid, I need to make r0 and c0 to make an imaginary box.



This is the function that prints the puzzle. Line 62 prints a separator every 3 rows. On line 65, it will print a bar every 3 column. This represents the sub-grid so that it is easier to see. And row 69-72 is just printing the number with space at the end except on column 8 or the last column.



This is the main function for this project. As the name implies, it solves the puzzle. Line 46 checks if there is still any blank space. If there is none, the puzzle is finished. Else, we will try to insert a number from 1-9 in an empty space. This number cannot break the board meaning it has to pass the insert\_number function that has been explained before. If this number is valid, it will be inputted to the space. This will go through all the blank spaces by doing recursion. This means it will call the function again and find a new blank space for it to try a number. However, if no solution is possible, it will backtrack and go back to the previous blank space and set it to 0 as seen in line 56. It will use the for loop in line 49 again and try a new number that might fit the space. This is done until the board is filled.

**IX. Project Link**

<https://github.com/1miaocat/Sudoku-solver>

# X. References

* <http://www.counton.org/sudoku/rules-of-sudoku.php#:~:text=The%20classic%20Sudoku%20game%20involves,a%20row%2C%20column%20or%20box>.