Sudoku Solver Assignment Documentation

Year 12 Session 5 2021 Networking and Cybersecurity Unit
Farid Fadil and Jerin George
Source Code: https://github.com/faridfadil/sudoku-game

Sudoku Solver Assignment Documentation

Year 12 Session 5 2021 Networking and Cybersecurity Unit
Farid Fadil and Jerin George
Source Code: https://github.com/faridfadil/sudoku-game

Project Schedule

Week	Rough Plan
13 (26-30 July)	• Know how to make a sudoku solver first, browse through some code online to ***get inspiration***. (Do not copy) • Make the UML diagram. • Algorithm Planning
14 (2-6 August)	• Begin Coding • Documentation • Test Logs
15(9-13 August) 4:00 PM 13 August Friday Due	• Presentation planning (PowerPoint with screenshots of product) • Finish presentation

Contribution Table

Both	Farid's Contribution	Jerin's Contribution
Test Logs	Research into existing algorithms for Sudoku Solvers. (Learned about Backtracking)	Test Plan And Logs
Research	Coded MainWindow class and SudokuSolver class and the functions	Research into brute force method and GUI interface
Planning for the overall product	Setup GitHub repository for contribution	Coded Difficulty Settings in Sudoku Solver
Contribution Table	Coded backtracking algorithm for sudoku solver	GUI, front-end aspect of the program (grid colours, top left favicon logo, buttons and labels)
Project Scheduling	Made code explanation video	Coded original grid layout
	Spruced up everything on the GitHub repository and created release v1.0.0 of the software	Made the flow chart for the sudoku solver program
	Project management	Documentation title page image

Sudoku Solver Assignment Documentation

Year 12 Session 5 2021 Networking and Cybersecurity Unit
Farid Fadil and Jerin George
Source Code: https://github.com/faridfadil/sudoku-game

Test Plan	

Objectives	• Generates random integers
	• Organises integers into sudoku form
	• Hides sudoku
	• Displays integers with specified difficulty
	• Solvable
	• Reliable code
	  - No errors
	• Efficient Code
	  - functions/modules
	  - classes with constructors
	• Graphical User Interface is functional and practical
	• Aesthetically pleasing

Deliverables	• Python Code
	  - GUI with Sudoku generator + solver
	• Relevant Images
	• Documentation
	  - Journal
	  - Test Logs
	  - Test Plan
	• Video Presentation

Resources Required → Testing	• Python IDLE - Code • Visual Studio Code - Collaboration • Github - Exchange of Code • Google Docs - Documentation • Discord - Communication
Test Strategy	1. Start with generating a sudoku board and matching it with GUI 2. Generate integers into Sudoku board 3. Randomise integers 4. Validate integers so that Sudoku is solvable 5. Hide all integers 6. Display integers with varying difficulties 7. Change the appearance of GUI 8. Edit and Submit

Sudoku Solver Assignment Documentation

Year 12 Session 5 2021 Networking and Cybersecurity Unit

Farid Fadil and Jerin George

Source Code: https://github.com/faridfadil/sudoku-game

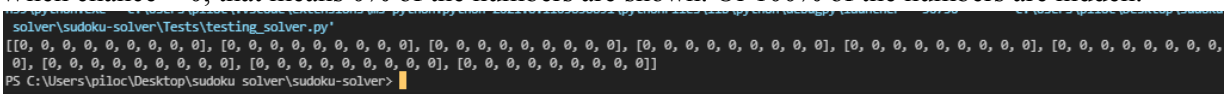
Test Logs

To begin with, the plan is to generate random numbers, solve the sudoku, hide the numbers and then display some so that it is solvable. Random number experimentation began and the code below works to display a number between 0 - 9 in different rows. The chance of a 0 occurring can either be increased or decreased and this is how the sudoku will change its difficulty.

```
#sample sudoku grid
sudoku_grid = [
        [0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0],
    ]

#chance to place a number, higher the value, the easier it is for the sudoku to be
solved
chance = 80
#choices of numbers
number_choices = [1, 2, 3, 4, 5, 6, 7, 8, 9]

#nested for loop for the 9x9 grid.
for column in range(9):
    for row in range(9):
        #random number generator from 1 to 100
        random_roll = random.randint(0, 100)

        #if number rolled is less than given chance
        if random_roll <= chance:
            # then place a number randomly chosen from the number_choices.
            sudoku_grid[row][column] = random.choice(number_choices)
        else:
            # otherwise, then put it in 0
            sudoku_grid[row][column] = 0

print(sudoku_grid)
```

Displaying the *main_sudoku_grid* to demonstrate the chance system:

When chance = 0, that means 0% of the numbers are shown. Or 100% of the numbers are hidden.



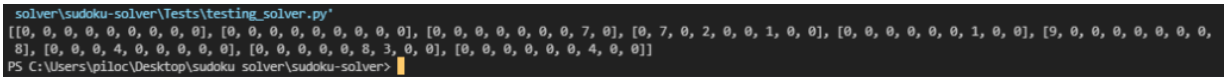
chance = 20 - 20% of numbers are shown

Sudoku Solver Assignment Documentation

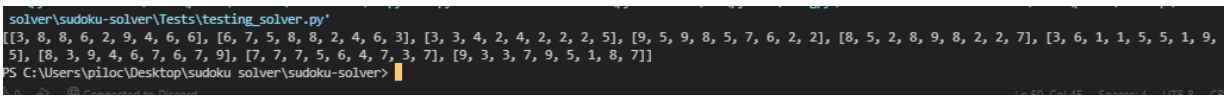
Year 12 Session 5 2021 Networking and Cybersecurity Unit

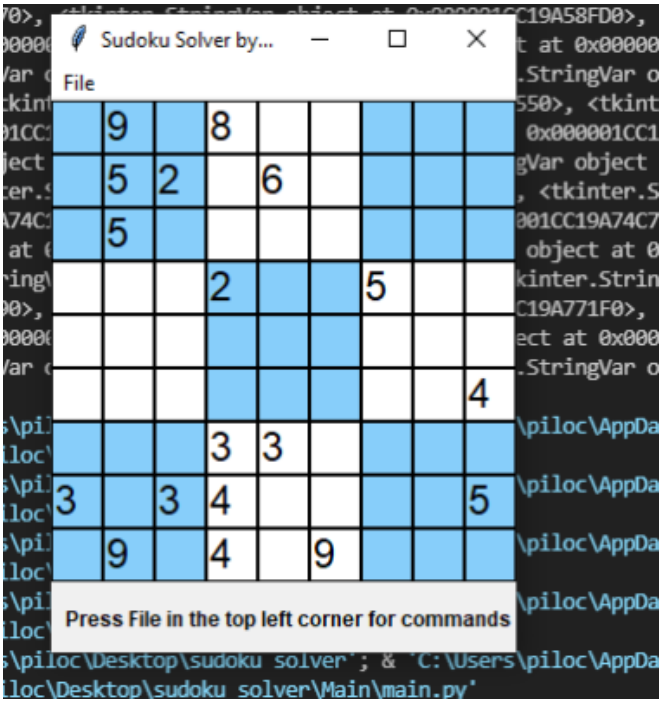
Farid Fadil and Jerin George

Source Code: https://github.com/faridfadil/sudoku-game

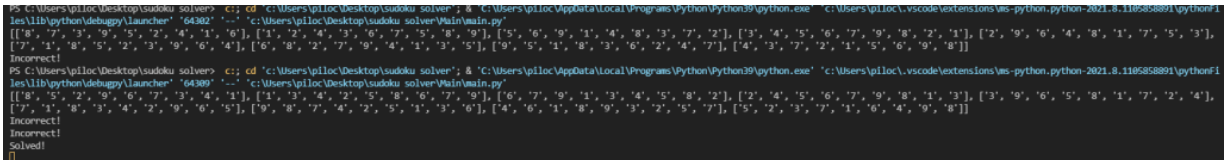


chance = 100 - 100% of numbers are shown

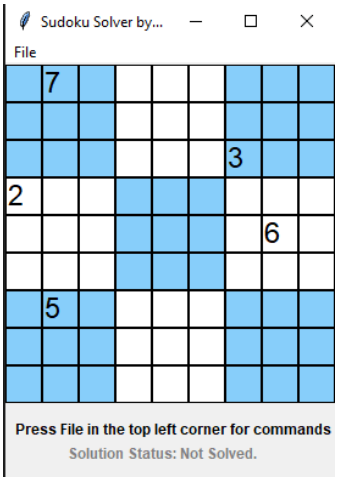


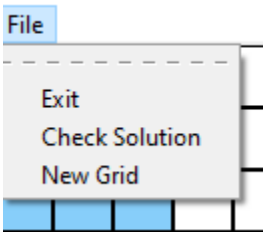


Although the random number generation works, it does not print them into the sudoku boxes so that they are valid. As seen above there are duplicate inputs in either the rows, column or square. Might have to just solve the whole grid first then hide the numbers.



The code has been proving to be successful in printing the sudokus, now the check button is being tested as seen above. If the sudoku is solved and matches the numbers generated before being hidden, it will display ‘correct!’. If it does not match, it displays ‘incorrect!’.






```
50          # menu dropdown for player controls in the top left corner.
51          menu = Menu(gui)
52          gui.config(menu = menu)
53
54          file = Menu(menu)
55          menu.add_cascade(label = 'File', menu = file)
56          file.add_command(label = 'Exit', command = gui.quit)
57          file.add_command(label = 'Solve', command = self.solve_grid)
58          file.add_command(label = 'Clear', command = self.clear_grid)
59
60          #NOTE ammend this function so that VALID numbers are placed on the grid.
61          self.fill_grid(chance_to_place = 20)
62
```

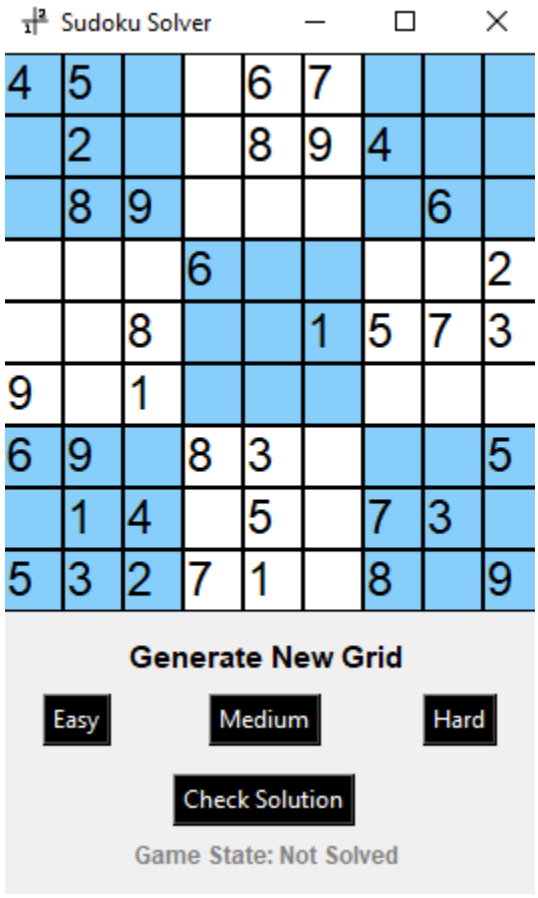
The file command is unnecessary in being used as the buttons for the sudoku grid. It will be easier and more beneficial if the file menu is replaced with buttons attached to the GUI. This can be done using the same method of command = ‘’.

```
# replace slots that are chosen at random with empty cells.
def hide_solution(self):
    CHANCE_TO_HIDE = 90
    for column in range(GRID_SIZE):
        for row in range(GRID_SIZE):
            #random number generator from 1 to 100
            random_roll = random.randint(0, 100)
            #if number rolled is less than given chance
            if random_roll < CHANCE_TO_HIDE:
                # then set the current cell to be none
                main_sudoku_grid[row][column].set('')

# save the grid so that the check_solution() function later on can check again
```

To change the difficulty of the sudoku the above integer can be changed. The chance to hide is the percentage in which the sudoku is displayed with 0's. The 0 is replaced with '' and so the number remains hidden. To change the difficulty 3 buttons can be added and the code duplicated so that it displays a different percentage each time.

```
# replace slots that are chosen at random with empty cells.
def hide_solutione(self):
    CHANCE_TO_HIDE = 63 # how frequent should the cells be made empty per loop iteration?
    for column in range(GRID_SIZE):
        for row in range(GRID_SIZE):
            #random number generator from 1 to 100
            random_roll = random.randint(0, 100)
            #if number rolled is less than given chance
            if random_roll < CHANCE_TO_HIDE:
                # then set the current cell to be none
                main_sudoku_grid[row][column].set('')
def hide_solutionm(self):
    CHANCE_TO_HIDE = 77 # how frequent should the cells be made empty per loop iteration?
    for column in range(GRID_SIZE):
        for row in range(GRID_SIZE):
            #random number generator from 1 to 100
            random_roll = random.randint(0, 100)
            #if number rolled is less than given chance
            if random_roll < CHANCE_TO_HIDE:
                # then set the current cell to be none
                main_sudoku_grid[row][column].set('')
def hide_solutionh(self):
    CHANCE_TO_HIDE = 87 # how frequent should the cells be made empty per loop iteration?
    for column in range(GRID_SIZE):
        for row in range(GRID_SIZE):
            #random number generator from 1 to 100
            random_roll = random.randint(0, 100)
            #if number rolled is less than given chance
            if random_roll < CHANCE_TO_HIDE:
                # then set the current cell to be none
                main_sudoku_grid[row][column].set('')
```

The final GUI has been changed and the difficulty updated. A logo also appears in the corner rather than the Tkinter logo.



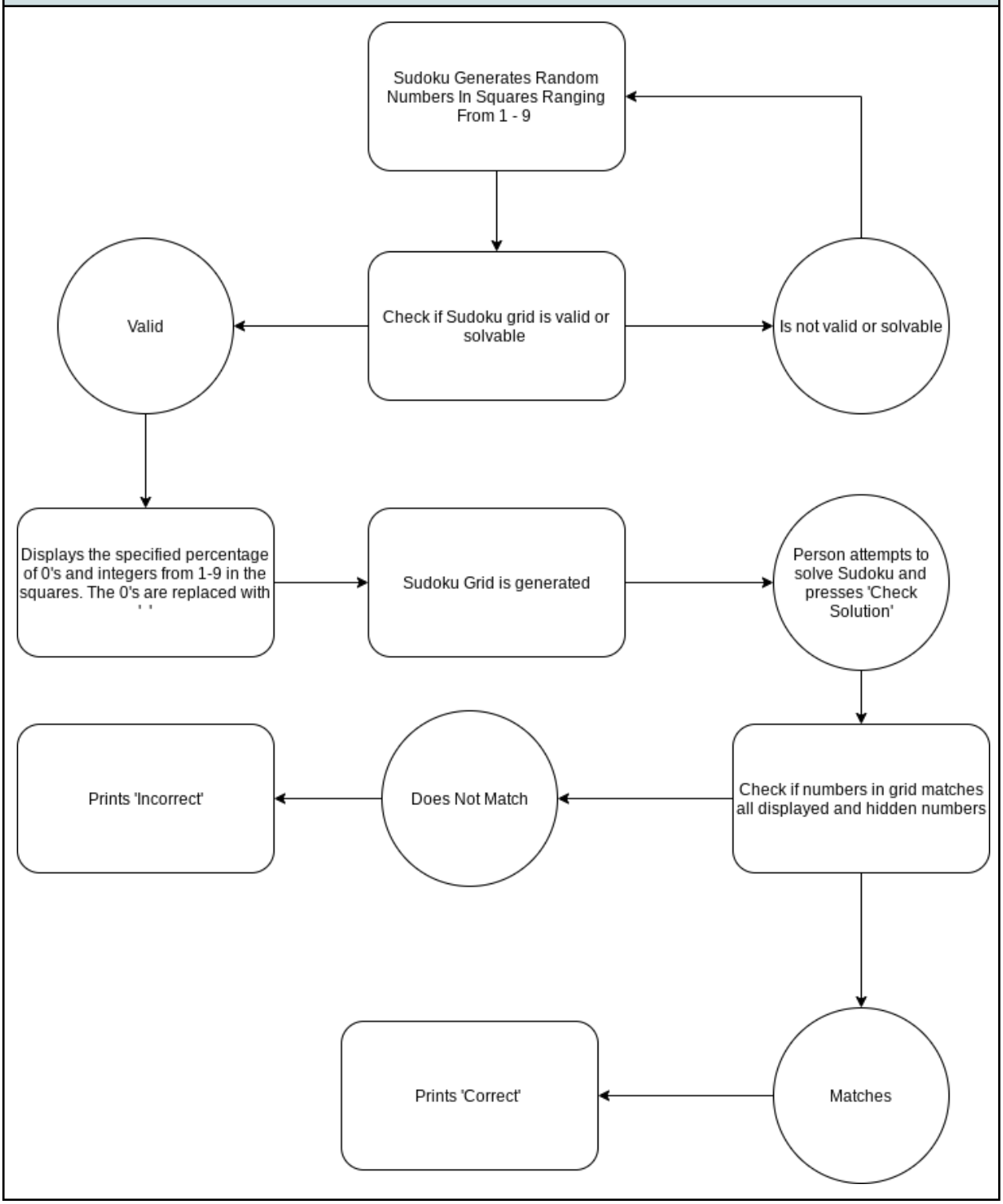
Sudoku Solver Assignment Documentation

Year 12 Session 5 2021 Networking and Cybersecurity Unit

Farid Fadil and Jerin George

Source Code: https://github.com/faridfadil/sudoku-game

Flowchart Diagram of Sudoku Solver



The Logical Approach to Generating the Sudoku Grid

To generate the Sudoku game, we first randomize the arrangement of the numbers in the top row of the Sudoku grid. Then, by using a backtracking algorithm that linearly steps through each cell and places a number in an empty cell based on the rules of Sudoku (only 1 unique number in the row and column and 3x3 grid), that fully solves the grid (we call the solved grid *main_sudoku_grid* in code). Then, the solved grid is saved in memory by looping through all the cells of *main_sudoku_grid* and assigning each value in the latter to the corresponding spots in a 2-dimensional list called *correct_solution_grid*. Then, a function loops through each grid cell of *main_sudoku_grid* with a chance to replace the currently displayed number with an empty cell (an empty string). The chance determines how hard the sudoku is to solve with a greater chance to hide making the puzzle harder as fewer numbers will be visible and a lower chance for an easier puzzle as more numbers will be visible.

Module Dependencies

Tkinter was used to build the GUI for the sudoku solver program as both contributors had prior experience with the library and because it is built into Python, anyone intending to test/run our code can easily do so without the overhead of installing additional libraries (like Pygame for instance). We also used the random module which helped in generating the randomized sudoku grid and other back-end logic.

Full Code Explanation

Full code explanation video: Python Sudoku Solver Code Explanation

NOTE There have been a few minor changes to the code since the video was made and below gives a run-down of the changes made and why we made them.

Sudoku Solver Assignment Documentation

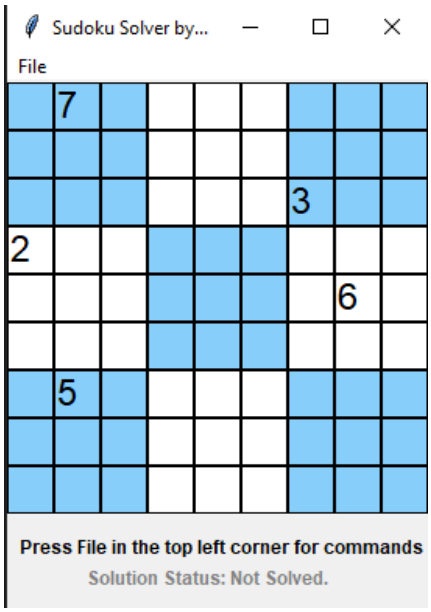
Year 12 Session 5 2021 Networking and Cybersecurity Unit

Farid Fadil and Jerin George

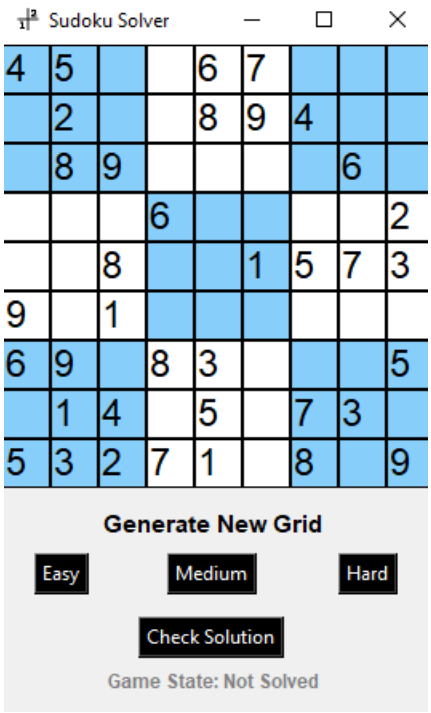
Source Code: https://github.com/faridfadil/sudoku-game

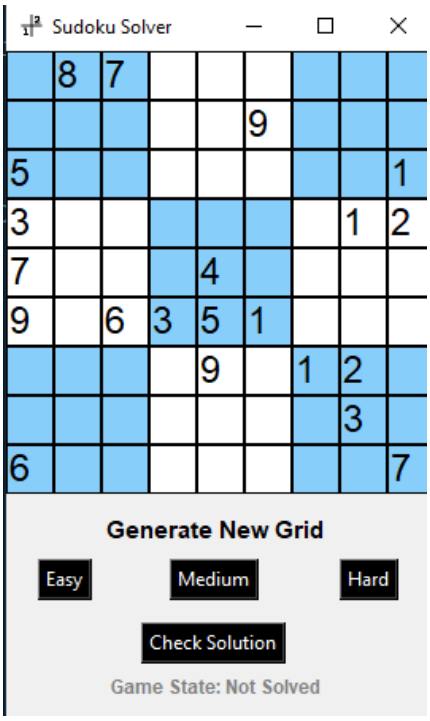
Adding Difficulties

In accordance with the assessment rubric, we had to include difficulty levels in the sudoku grid. This was done by creating separate functions for generating the grid based on an alteration of the *CHANCE_TO_HIDE* constant variable in the *hide_solution* function. Easy meant chance to hide was smaller and Hard meant a chance to hide was higher. As a result of adding the difficulty options, the previous GUI's file options menu was removed from the final GUI, shown on the right, to improve the user experience and convenience in the navigation of the program.



Final Product





Easy Grid (left), Hard Grid (right). Note the difference in numbers shown between the two difficulties