


# SUDOKU GENERATOR

## Networking and Security

Phebe Le  
Planning Journal

Session 1.5: 6/5/2023-23/5/23

 Sudoku Board - Easy

**Solve by filling in the numbers**

9	5	7	6	1	3	2	8	4
4	8	3	2	5	7	1	9	6
6	1	2	8	4	9	5	3	7
1	7	8	6	3	4	9	5	2
5	2	4	9	7	1	3	6	8
3	6	9	5	2	9	7	4	1
8	4	5	7	9	2	6	1	3
2	9	1	4	3	6	8	7	5
7	3	6	1	8	5	4	2	9

The puzzle has been solved

## 1. Introduction and project overview

This task is to solve a CSP (Constraint Satisfaction problem). This being a sudoku board. The logic of a game is to try and substitute numbers from 1-9 into a pre-formatted board with numbers. The substituted number must not have another identical number within the same row, column and square. These sections are called houses.

## 2. Project objective

The objective of this project is to a functioning Sudoku game within Python. The design of the game will be segregated into three modes: easy, medium and hard. Each mode will be based in the same board size but have a different number and location of hints. The number and location will depend on the chosen mode. Alongside this, there will be two different ways for the answer to be found: though the user and through the computer-generated answer. The user may iteratively add numbers and check their answer to eventually reach the answer or the user may click a button that displays the correct answer.

The sudoku must be able to perform the following functions

- Allow the user to choose which difficulty mode to select
- Create a functioning sudoku board
- Allow the user to add digits to 'solve' the puzzle
- Produce a correct answer
- Store and cross-check user's solution with the correct answer
- Randomly generate different game templates

## 3. Project scope

There are numerous possible methods for creating and solving a sudoku algorithm. Yet the large list has been narrowed down to three.

1. Recursion, a backtracking technique. This technique is not 'intelligent' due to its tedious method of testing every digit until there are no more digits. This method could be optimised significantly. The computing and memory size of the program may also be significantly longer due to its algorithm.
2. Bit mask. This method is a slight optimisation of Recursion as it uses a bitmask/bitwise operation to determine a bitmask and a value element
3. Cross hatching with backtracking. This method a significant optimisation of both algorithms due to its 'pruning' technique. As a result, the size of the computing and memory size will be significantly smaller.

The final choice of the methods will be cross hatching with backtracking. This method is ideal as it is an optimised option within the backtracking algorithm. Whilst this is optimal, due to time constraints backtracking may have to be used.

The steps to create sudoku which uses a cross-hatching technique consists of several steps.

1. Initialise the board
2. Function to make sure the numbers 1-9 are not repeated in each house
3. Function to solve the board

The logic of this program will consist of several components.

- Search and sort techniques: Insertion sort for the checking validity of number in function
- Tuples, dictionaries and lists: To create a sudoku board
- Classes: To create window and button
- Functions: To create main sections of the code
- Modules: \*or tk

## 4. Assumption

There are several assumptions for this task.

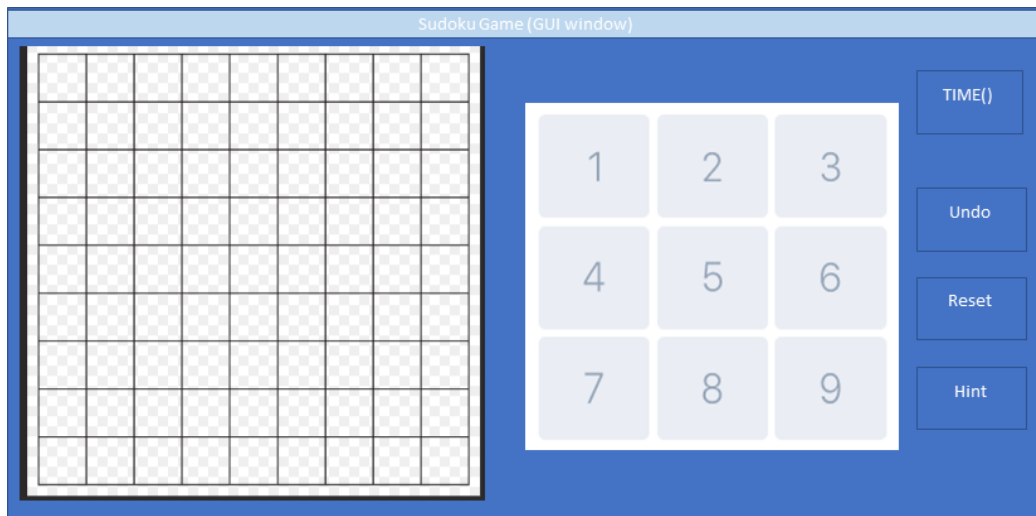
- The test environment will be initialised through Tkinter in Python Idle. This will allow the design to include custom Tkinter widgets and improve the UI of the game. This will mean the game must be interactive within the Tkinter GUI window instead of an external site or program.
- The game will be unique due to the cycle of the different templates for different modes.
- The task will be completed within three weeks. The week will be segmented into different stages. The creation of a board and GUI design, solving function, validity check function, validity of next check function.

## 5. Project schedule

4	5	6	7	8	9	10
	Planning and developing concept	Begin initial development of the code				Flowchart+GUI
	START					
	Western Austr					
11	12	13	14	15	16	17
Flowchart+GUI design						Develop integr
	NULL: From Exam week					Uer code
	King's Birthday					
18	19	20	21	22	23	24
Develop integral logic and coding			Finish developing code		FINISH	
Uer code		Recursion/other method code			Finish docume	

## 6.1. Phase 1: research, planning, early code development

### GUI Window



Aim: For the GUI window, I intend to segregate the section into three: the sudoku board, the number pad and the 4 button/widgets. The number pad will input a number into the given square the mouse has clicked. The time widget will begin once a user begins inputting numbers, the undo button will undo the previous action, the reset will reset to the beginning puzzle the user begun with. The hint button will input one number into the sudoku puzzle.

### Menu

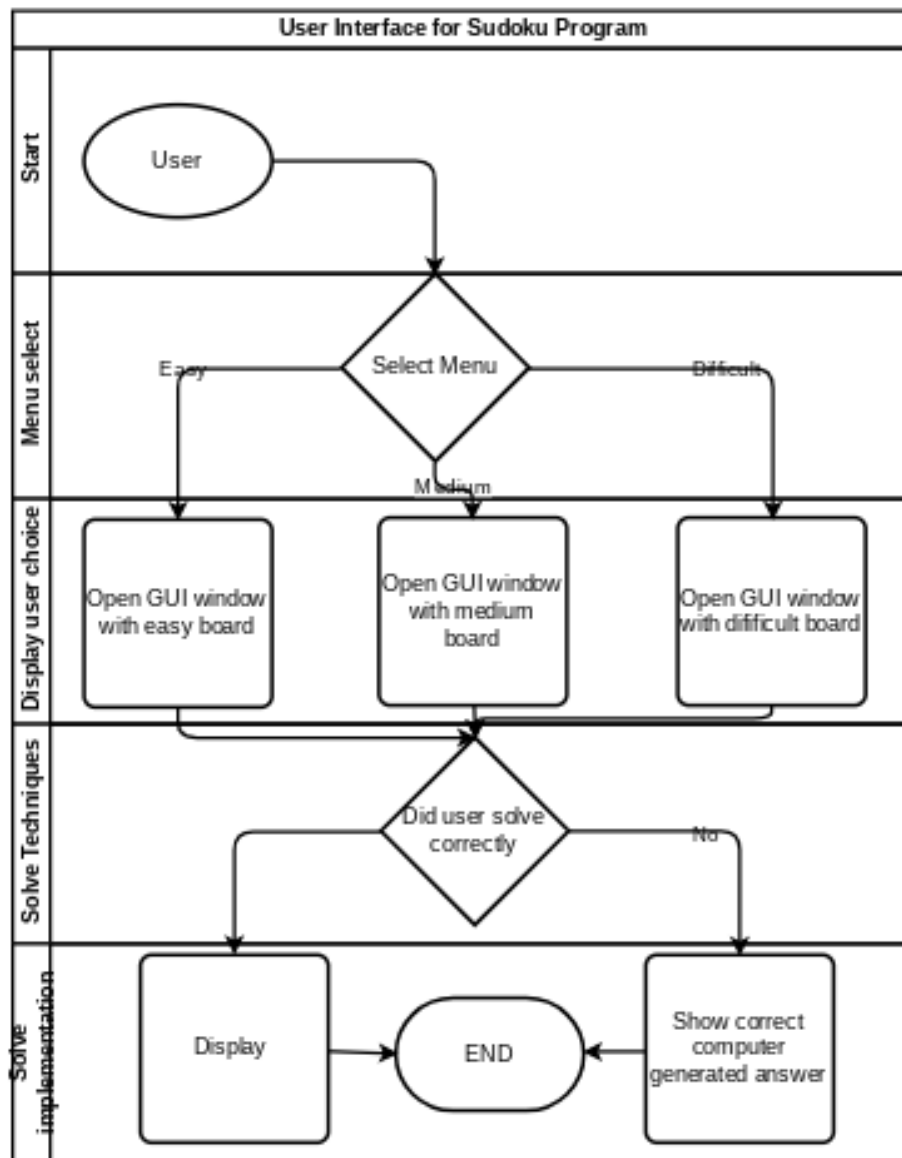


Aim: The menu will be created on the 'main screen' of the GUI window. This will then lead into the different sudoku modes and randomised designs.

### Sudoku square

To create the sudoku code, the use of a list will be used as a prototype. This will initially be a list with 81 numbers to replicate a 9x9 square.

## 6.2. Flowchart



## 6.3. Deliverables

The expected delivered ideas will be a complete sudoku board that has random generation, an algorithm and a menu. Components such as different grid sizes or other external components will be attempted yet due to the current frame of time considering exam week and other external aspects, I will be unlikely that this assignment will be finished to the degree which is stated on the assignments sheet.

## **6.4. Estimation of different aspects of the assignment**

Planning: Due to my enjoyment of planning, this section will likely be enjoyable and interesting to me.

Development: The development will be difficult due to time constraints and due to my lack of knowledge on properly being able to implement such aspects. I will also spend too much time on meaningless features

Documentation: The documentation section may also be fun to do, yet it will be quite time consuming.

## **6.5. Resources**

Components such as a working computer and proper save features will be required. The internet will also be required due to the lack of knowledge.

## **6.6. Reflection**

In this assignment there was a lot which could have been improved, created, finished or fixed. I think that my process within these last couple days has been quite strong due to my limited amount of time. I think that watching python tutorials has helped me understand and learn how different concepts are beneficial to the development of my code. If I were to redo aspects of the code, I would focus on getting the random generation to work in the way it should. I would also try to pass the parameter of the for loop to the other for loop to display on the main board (def Solved)