New Guidelines for SudokuPy3
November 2020

1. Main Files

The repository *SudokuPy3* is a sudoku solving program which originates from an intuitive idea that a sudoku puzzle could be much simplified when blanks of two feasible numbers (dual blanks) are filled accordingly. Then, different sudoku-solving techniques could be applied to complete the sudoku puzzle. All programs are written in Python 3. Main files include:

File Name	**Description**
SudokuPy3.py	Main program
SudokuPy3Fun.py	Contains essential algorithmic functions
SudokuPy3Bank.py	Create the data structure of sudoku puzzles
SudokuPy3PulpSolver.py	Solve sudoku puzzle using the Python module PuLP
mySudokuBank.txt	Examples of sudoku puzzles

Required modules: NumPy, Matplotlib, SymPy

2. Demonstration

Every sudoku puzzle is stored in an 9×9 ndarray of integral entries. As an example, a sudoku puzzle (PuzzleNo1008) is given by

$$\left[\begin{array}{ccc|ccc|ccc} 8 & 7 & 1 & 3 & 4 & 5 & 0 & 0 & 9 \\ 0 & 4 & 0 & 0 & 0 & 0 & 1 & 0 & 8 \\ 0 & 0 & 0 & 0 & 9 & 0 & 3 & 0 & 4 \\ \hline 0 & 3 & 0 & 4 & 7 & 9 & 5 & 8 & 0 \\ 0 & 9 & 8 & 0 & 0 & 2 & 4 & 0 & 7 \\ 7 & 0 & 4 & 0 & 0 & 0 & 9 & 0 & 0 \\ \hline 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 6 & 8 & 0 & 0 & 4 & 3 \\ 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right].$$

While some entries are filled by 1/2/3/4/5/6/7/8/9 according to their prescribed values in a sudoku puzzle, other unfilled entries are temporarily indicated by the value 0. To solve the above puzzle (No1008), readers could run SudokuPy3.py directly. The first line of output requests readers to type the puzzle number.

Type the puzzle number: 1008

A list of information about No1008 is shown, followed by different options that readers could choose to solve No1008.

```
Option 1:    myBootstrap
Option 2:    myExhaustFill
Option 3:    mySolveEngine
Option 4:    NDBsolveEngine
Option 5:    PulpSolver
Option 0:    QUIT
```

The main solving machine developed in this project is called by Option 3.

```
Choose Option: 3
```

An autorun will then be carried out. In this process, output lines are printed to display the progress of the running program. At the end, readers could access the solution to No1008. The solution is first displayed as a graph in the new window. Then, back to the console, the solution is printed as an ndarray.

```
The solution is
[[ 8 7 1 3 4 5 6 2 9 ]
 [ 3 4 9 7 2 6 1 5 8 ]
 [ 2 5 6 8 9 1 3 7 4 ]
 [ 1 3 2 4 7 9 5 8 6 ]
 [ 5 9 8 1 6 2 4 3 7 ]
 [ 7 6 4 5 3 8 9 1 2 ]
 [ 4 2 7 9 1 3 8 6 5 ]
 [ 9 1 5 6 8 7 2 4 3 ]
 [ 6 8 3 2 5 4 7 9 1 ]]
```

Readers would choose to save or not to save the above solution by typing Y or N respectively.

```
Save Answer(Y/N)? N
```

When the table of options show up again, readers may input 0 to exit the program.

```
Choose Option: 0
```

3.The Main Solver

The main solver of this sudoku solving program, is the function *mySolveEngine* found in the file SudokuPy3Fun.py. The solving process is divided into three steps: (1) fill by exhaustion,
(2) fill by choices of dual blanks and (3) fill by bootstrapping. These phrases are not formal names in sudoku, so we would discuss them one by one.

3.1 Fill by Exhaustion

Filling blanks by exhaustion is a direct method to solve a sudoku puzzle, which founds on the concept that in every row, column, or block, each number from 1 to 9 must occur once. Take the puzzle No1008 as an example.

$$\left[\begin{array}{ccc|ccc|ccc} 8 & 7 & 1 & 3 & 4 & 5 & 0 & 0 & 9 \\ 0 & 4 & 0 & 0 & 0 & 0 & 1 & 0 & 8 \\ 0 & 0 & 0 & 0 & 9 & 0 & 3 & 0 & 4 \\ \hline 0 & 3 & 0 & 4 & 7 & 9 & 5 & 8 & 0 \\ 0 & 9 & 8 & 0 & 0 & 2 & 4 & 0 & 7 \\ 7 & 0 & 4 & 0 & 0 & 0 & 9 & 0 & 0 \\ \hline 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 6 & 8 & 0 & 0 & 4 & 3 \\ 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right].$$

The last row reads $[6, 0, 0, 0, 0, 0, 0, 0, 0]$. For every entry on the last row, we could list out all possible numbers to be filled.

Entries	Current Values	Possible Numbers
(9, 1)	6	6
(9, 2)	0	1, 2, 5, 8
(9, 3)	0	2, 3, 5, 7, 9
(9, 4)	0	1, 2, 5, 7, 9
(9, 5)	0	1, 2, 3, 5
(9, 6)	0	1, 3, 4, 7
(9, 7)	0	2, 7, 8
(9, 8)	0	1, 2, 7, 9
(9, 9)	0	1, 2

The last row must contain a copy of 4, but the number 4 is possible only at the entry (9, 6). Therefore, the entry (9, 6) must be filled by 4.

$$\left[\begin{array}{ccc|ccc|ccc} 8 & 7 & 1 & 3 & 4 & 5 & 0 & 0 & 9 \\ 0 & 4 & 0 & 0 & 0 & 0 & 1 & 0 & 8 \\ 0 & 0 & 0 & 0 & 9 & 0 & 3 & 0 & 4 \\ \hline 0 & 3 & 0 & 4 & 7 & 9 & 5 & 8 & 0 \\ 0 & 9 & 8 & 0 & 0 & 2 & 4 & 0 & 7 \\ 7 & 0 & 4 & 0 & 0 & 0 & 9 & 0 & 0 \\ \hline 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 6 & 8 & 0 & 0 & 4 & 3 \\ 6 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 \end{array}\right].$$

This method to simplify a sudoku puzzle, is called *filling by exhaustion* in our context. It is carried out by the function *myExhaustFill* in SudokuPy3Fun.py.

3.2 Fill by the choices of dual blanks

In general, there may be blanks on the puzzle which admits only two possible numbers. We would call them *dual blanks*. Once they are filled by possible numbers, the puzzle is much simplified. To illustrate this concept further, we turn back to the puzzle No1008.

$$\left[\begin{array}{ccc|ccc|ccc} 8&7&1&3&4&5&0&0&9\\ 0&4&0&0&0&0&1&0&8\\ 0&0&0&0&9&0&3&0&4\\ \hline 0&3&0&4&7&9&5&8&0\\ 0&9&8&0&0&2&4&0&7\\ 7&0&4&0&0&0&9&0&0\\ \hline 4&0&0&0&0&0&0&0&5\\ 0&0&0&6&8&0&0&4&3\\ 6&0&0&0&0&0&0&0&0 \end{array}\right]$$

We list out all the dual blanks of No1008. In practice, we are choosing at most 15 of them.

Entries	Possible Numbers
(1, 7)	2, 6
(1, 8)	2, 6
(2, 4)	2, 7
(2, 5)	2, 6
(2, 6)	6, 7
(3, 1)	2, 5
(4, 1)	1, 2
(4, 3)	2, 6
(5, 1)	1, 5
(5, 4)	1, 5
(8, 6)	1, 7
(8, 7)	2, 7
(9, 9)	1, 2

There 13 dual blanks on No1008, so there is a total of $2^{13} = 8192$ ways of filling dual blanks. For instance, one way of filling is

$$\left[\begin{array}{ccc|ccc|ccc} 8&7&1&3&4&5&2&6&9\\ 0&4&0&2&6&7&1&0&8\\ 5&0&0&0&9&0&3&0&4\\ \hline 1&3&6&4&7&9&5&8&0\\ 5&9&8&1&0&2&4&0&7\\ 7&0&4&0&0&0&9&0&0\\ \hline 4&0&0&0&0&0&0&0&5\\ 0&0&0&6&8&1&7&4&3\\ 6&0&0&0&0&0&0&0&2 \end{array}\right].$$

Once the sudoku puzzle remains consistent after filling dual blanks with numbers, we proceed to either filling by exhaustion, filling by bootstrapping or another round of dual blank filling.

3.3 Fill by Bootstrapping

After several round of dual blank filling, if the total number of combinations of the sudoku puzzle is less than 10^6, then we procced with boostrapping in order to find a solution to the sudoku puzzle. In the sudoku solving program, bootstrapping is carried out by the function *myBootstrap* in SudokuPy3Fun.py.

The overall rundown of the main solver *mySolveEngine*, is described in the following flow chart.