Python Homework 2

Assigned: 11/29

Due: 12/07, 11:59pm

Possible Points: 100

Submit on Canvas a zip file containing your solution code for each problem (just the .py files, please do not submit the solution/project files), turn in at least one .py file for each problem. Each problem specifies what you should title the file containing your solution so that the autograder can find it. Incorrectly named programs will not be graded and thus will get 0 points for the problem.

Since the assignment will be graded using an automated grading system you should have your program's output match the sample output shown. If your program fails to run using Visual Studio 2017 it will receive zero points. If your program fails to run properly (runtime errors or wrong output) points will be deducted based on how correct is the program.

If you have any questions or concerns about the assignment do not hesitate to post on the discussion forum on canvas, send us message on canvas, or see us after class or during office hours.

Students are expected to write their source code from scratch, those who copy code from each other or web (including from Canvas examples) will be reported for cheating. We will also look at your code so do not just hard-code the answer and print it.

We strongly encourage to factor out the Sudoku reading, printing, and checking into a separate module.

Example Input File (same for all problems)

```
5 3 0 0 7 0 0 0 0
6 0 0 1 9 5 0 0 0
0 9 8 0 0 0 0 6 0
8 0 0 0 6 0 0 0 3
4 0 0 8 0 3 0 0 1
7 0 0 0 2 0 0 0 6
0 6 0 0 0 0 2 8 0
0 0 0 4 1 9 0 0 5
0 0 0 0 8 0 0 7 9
```

1. Reading Input: 20 points

File: `sudoku1.py`

A Sudoku game is played on a 9x9 grid and starts at some initial valid state. To play, you have to read the initial configuration of a grid first. Since the player may want to choose a different starting grid for each game, you need to allow the user to type in the name of the file that contains the initial grid.

At the start of the program, prompt user for the file path using the `input()` function. Open the specified file and read its content. The file will contain a 9x9 grid of numbers separated by spaces; 1-9 are valid start numbers and 0 denotes empty cell. To verify that the grid was read correctly, print it out. You can assume the input will always be a valid Sudoku grid.

Console Session

```
Enter input filepath: input.txt
5 3 0 0 7 0 0 0 0
6 0 0 1 9 5 0 0 0
0 9 8 0 0 0 0 6 0
8 0 0 0 6 0 0 0 3
4 0 0 8 0 3 0 0 1
7 0 0 0 2 0 0 0 6
0 6 0 0 0 0 2 8 0
0 0 0 4 1 9 0 0 5
0 0 0 0 8 0 0 7 9
```

2. Solving Sudoku: 20 points

File: `sudoku2.py`

Read a Sudoku grid as before from the user-specified file. To play the Sudoku game, a user must be able to enter numbers into the grid. After you printed the initial grid, prompt the user for a row, column, and number, set the content of the cell at position `(row, column)` to value `number` and print the updated grid. The input will be a valid grid, and the row and column will be the index of a cell that is empty (one containing a 0).

Note: you will have to convert the number of the input grid to integers to update the values in the grid. Each cell should have numbers in 0-9 range (with 0 indicating an empty cell).

Console Session

```
Enter input filepath: input.txt
5 3 0 0 7 0 0 0 0
6 0 0 1 9 5 0 0 0
0 9 8 0 0 0 0 6 0
8 0 0 0 6 0 0 0 3
4 0 0 8 0 3 0 0 1
7 0 0 0 2 0 0 0 6
0 6 0 0 0 0 2 8 0
0 0 0 4 1 9 0 0 5
0 0 0 0 8 0 0 7 9
Enter row: 1
Enter column: 2
Enter number: 1
5 3 0 0 7 0 0 0 0
6 0 1 1 9 5 0 0 0
0 9 8 0 0 0 0 6 0
8 0 0 0 6 0 0 0 3
4 0 0 8 0 3 0 0 1
7 0 0 0 2 0 0 0 6
0 6 0 0 0 0 2 8 0
0 0 0 4 1 9 0 0 5
0 0 0 0 8 0 0 7 9
```

3. Validating Steps: 20 points

File: `sudoku3.py`

You can assume that the input entered by the user is in a valid range: row and column are in the 0-8 range, and `number` is in the 0-9 range. In this exercise, you need to check if the input constitutes a valid step in solving the Sudoku based on the rules of the game.

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

Figure 1: Example of Sudoku grid which consists of nine 3x3 subgrids (source Wikipedia).

A valid configuration to the Sudoku grid must satisfy following rules:

- each cell has a number in the 0-9 range
- in each row, no number in the range 1-9 can be repeated
- in each column, no number in the range 1-9 can be repeated
- in each 3x3 subgrid, no number in the range 1-9 can be repeated

For example, entering number 3 at row 0 and column 2 will violate both the row rule and the subgrid rule.

In this exercise, read the starting Sudoku configuration by prompting a user for the file path as in part 1 or 2. When user enters a new number that

violates the Sudoku rules print a message containing the rule that was broken (`'Number X twice in row'`, `'Number X twice in column'`, or `'Number X twice in subgrid'`; **check and print them in this order**) and do not update the grid. If the number does not break any rule, update the grid and print it.

Console Session

```
Enter input filepath: input.txt
5 3 0 0 7 0 0 0 0
6 0 0 1 9 5 0 0 0
0 9 8 0 0 0 0 6 0
8 0 0 0 6 0 0 0 3
4 0 0 8 0 3 0 0 1
7 0 0 0 2 0 0 0 6
0 6 0 0 0 0 2 8 0
0 0 0 4 1 9 0 0 5
0 0 0 0 8 0 0 7 9
Enter row: 1
Enter column: 2
Enter number: 1
Number 1 twice in row
5 3 0 0 7 0 0 0 0
6 0 0 1 9 5 0 0 0
0 9 8 0 0 0 0 6 0
8 0 0 0 6 0 0 0 3
4 0 0 8 0 3 0 0 1
7 0 0 0 2 0 0 0 6
0 6 0 0 0 0 2 8 0
0 0 0 4 1 9 0 0 5
0 0 0 0 8 0 0 7 9
```

4. Grid GUI: 20 points

File: `sudoku4.py`

In the quest for building a user-friendly Sudoku application, we have to create GUI as most users dislikes console. In this part, you need to write Qt program that constructs a 9x9 grid consisting of nine 3x3 subgrids. Cells in the subgrids should be combo boxes that allow the user to select a number (see code presented in class for example). Read the input file and fill the GUI with the initial grid. The user can perform moves using the combo boxes in the GUI but you are not required to check for the validity of the moves based on Sudoku rules.

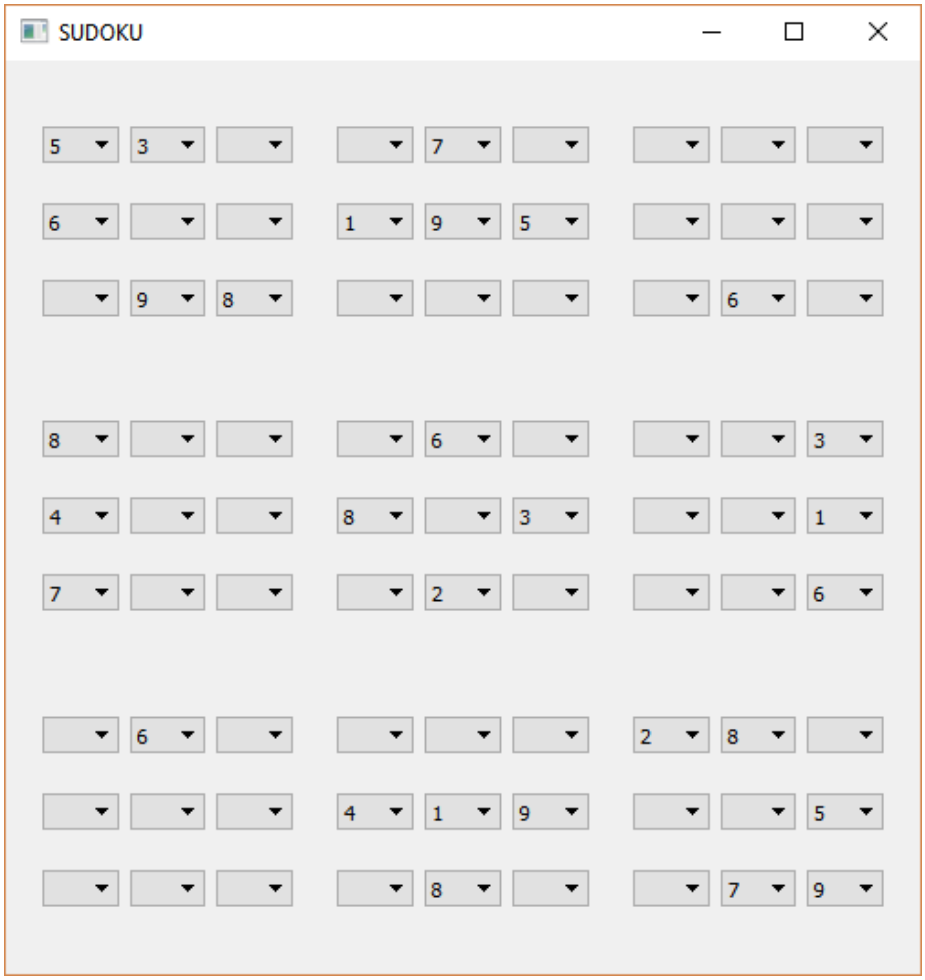

Figure 2: Example Sudoku GUI

5. Validation and Hints in GUI: 20 points

File: `sudoku5.py`

Anybody doing Sudoku on paper knows that it is quite suboptimal experience, but we use computers and sky is the limit. Enhance the Sudoku GUI and check for the three rules. If the user makes a mistake, show a message with the specifics of the mistake and revert back to the previous value in the grid cell.

You need to check these rules:

- in each row, no number in the range 1-9 can be repeated
- in each column, no number in the range 1-9 can be repeated
- in each 3x3 subgrid, no number in the range 1-9 can be repeated

Moreover, it is common to write down the possible numbers for each cell when solving Sudoku on paper. Modify your code so that each empty combo box shows only the list of valid numbers from the range 1-9. For each non-empty combo box show only the option to maintain the current value or clear the cell. For example, the possible choices that do not violate any of the rules in row 0 and column 2 are 1, 2, and 4. Similarly, for row 3 and column 3 the user should only have the two options of clearing the cell or maintaining the value 8.