15-112, Spring 2015

Assignment 6

Submission

You must submit one ZIP file on Autolab. This ZIP file must be named *userid*-hw6.zip and must include all of the Python files that you have written for this assignment. Please write your name and userid as a comment at the beginning of each Python file.

Recommendations

- **Make sure to test your program**.
- Make sure that your program is executable. If you are unable to complete portions of the assignment, comment out the part of the code that does not work properly, and explain what you did, what worked, and what did not. It is your responsibility to explain as carefully as you can why you think you were unable to get the code working, what you think is wrong, and how you might go about fixing it. The quality of such an explanation will be important to us in deciding whether to give you partial credit.

1 Sudoku [100 points]

In this exercise, we want to write a program to play Sudoku. If you are not familiar with the Sudoku game, I recommend you to read the corresponding Wikipedia article and play few games before starting this assignment. You will find many programs to play Sudoku on the Web.

`http://en.wikipedia.org/wiki/Sudoku`

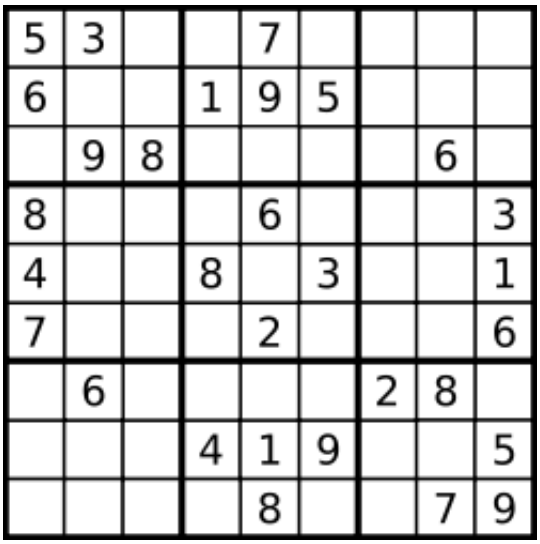

Figure 1: An example of an incomplete Sudoku grid (source *Wikipedia*)

We model a Sudoku grid in Python as a `list of list of numbers`. A Sudoku grid is a list of rows and rows are list of numbers. When a cell of the Sudoku grid is empty, we use the number the number `0` (which is not a valid Sudoku number). For instance, the Sudoku grid given in figure 1 is represented as follows:

```
myGrid = [ [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] ]
```


1.1 Checking an arbitrary list of numbers [15 points]

Write a function `checkList(l)` that takes `l` (as `list of numbers`) and returns `True` if the same number does not appear more than once in the list `l` (`0` that is not taken into account) . It returns `False` otherwise.

- `l` is the list of numbers.

1.2 Checking a row [15 points]

Write a function `checkRow(grid, row)` that takes a Sudoku `grid` (as `list of list of numbers`), a `row` index (as `number`) and returns `True` if the `row` index of `grid` is valid according to the rules of Sudoku. It returns `False` otherwise.

- `grid` is the Sudoku grid.
- `row` is the row index to check. We assume that this index is between 0 and 8.

1.3 Checking a column [15 points]

Write a function `checkColumn(grid, column)` that takes a Sudoku `grid` (as `list of list of numbers`), a `column` index (as `number`) and returns `True` if the `column` index of `grid` is valid according to the rules of Sudoku. It returns `False` otherwise.

- `grid` is the Sudoku grid.
- `column` is the column index to check. We assume that this index is between 0 and 8.

1.4 Checking a square [15 points]

Write a function `checkSquare(grid, square)` that takes a Sudoku `grid` (as `list of list of numbers`), a `square` index (as `number`) and returns `True` if the `square` index of `grid` is valid according to the rules of Sudoku. It returns `False` otherwise.

- `grid` is the Sudoku grid.
- `square` is the square index to check. We assume that this index is between 0 and 8.

The square indexes are mapped as shown in figure 2

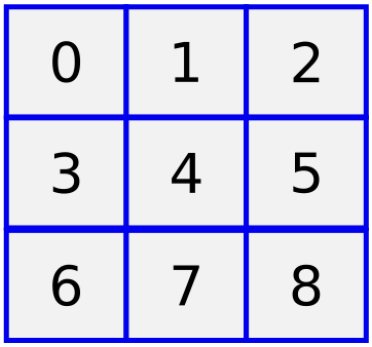

Figure 2: Square indexes

1.5 Adding a number in a Sudoku Grid [20 points]

Write a function `addNumber(grid, row, column, n)` that takes a Sudoku `grid` (as `list of list of numbers`), a `row` index, a `column` index , a number `n` (all as `number`) and returns `True` if the number fits in the position (`row`, `column`) of `grid` according to the rules of Sudoku. It returns `False` otherwise.

- `grid` is the Sudoku grid.
- `row` is the row index.
- `column` is the column index.
- `n` is the number to insert.

1.6 Verifying a solution [20 points]

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

Figure 3: Solution to the grid given in figure 1 (source *Wikipedia*)

We model a solution to a Sudoku grid as a `list of list of nu2mbers`. For instance, the solution shown in figure 3 is encoded as follows:

```
mySolution = [ [4,6,8,9,1,2],
               [7,2,3,4,8],
               [1,3,4,2,5,7],
               [5,9,7,1,4,2],
               [2,6,5,7,9],
               [1,3,9,4,8,5],
               [9,1,5,3,7,4],
               [2,8,7,6,3],
               [3,4,5,2,6,1] ]
```

Write a function `checkSolution(grid, solution)` that takes a Sudoku `grid` (as `list of list of numbers`), a `solution` (as `list of list of numbers`) and returns `True` if `solution` is valid solution for `grid` according to the rules of Sudoku. It returns `False` otherwise.

- `grid` is the Sudoku grid.
- `solution` is the collection of numbers to verify.

