INTRODUCTION TO COMPUTER SCIENCE I
PROJECT 2A
Sudoku: The Beginning

The first in a two part project in which we will write a program that solves Sudoku puzzles. In the first part, we will lay the foundation, writing functions that read in a puzzle from a file, print that puzzle, and test the puzzle to be sure that it does not violate the Sudoku rules.

1 Overview

Start a new module, `sudoku.py`, that contains the following functions (which may, in turn, call on other functions that you write):

1. `readGrid(filename)`: Open the `filename` given as a parameter and read an unsolved Sudoku puzzle from a file into a 2-dimensional grid of values (that is, a list of lists of integers). See Section 2.1 for details.

2. `printGrid(grid)`: Print the 2D grid of values in a format that matches the format of the input file. See Section 2.2 for details.

3. `checkGrid(grid)`: Determine whether the grid is a valid Sudoku board, with no duplicate (non-zero) digits in any row, column, or subgrid. See Section 2.3 for details.

A complete solution for this project should contain a `main()` function that uses the functions above in the following form:

```
def main ():
    filename = input('Provide the name of a file
                     that contains a sudoku puzzle: ')
    grid = readGrid(filename)
    printGrid(grid)
    valid = checkGrid(grid)
    if valid:
        print('Grid OK')
    else:
        print('Not a valid grid!')
```

2 Some details

Below are somewhat more details descriptions of the work that each function should do, and how it might go about it.

2.1 Reading a puzzle file

There are some essential functions that you will need to use in order to open and read a Sudoku puzzle. But first, to consider the task of reading such a file, we must contemplate what a Sudoku puzzle file looks like. Specifically, it is a **text file** that contains something like the following:

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

This 9-by-9 grid of digits represents a complete Sudoku puzzle by placing the digit 0 wherever a blank entry in the puzzle should appear, and then placing all non-zero digits in their proper places. The goal of the `readGrid()` function is to open a file like this one, and then read its digits (including the blank-place-holding zeros) into a list-of-lists-of-ints that represent the puzzle.

The functions that should help with this task are:

- `open(filename)`: This function *opens* the named file (that is, `filename` points to a string that contains the name of a file), preparing it for reading. It returns a special thing known as a *file object*. We don't need to know much about these objects other than their role in allowing us to now read from that file. Thus, this function should be called in something like the following form:

  ```
  f = open(filename)
  ```

- `f.readline()`: This dotted method can be called on a file object (such as `f`, above). It reads the **next line of text in the given file**, returning it as a string. So, it would be called like so:

  ```
  s = f.readline()
  ```

- `s.split()`: Another dotted method, but this one is called on a string (e.g., `s`, above). It assumes that a space (or any whitespace character) is a *delimiter*, and then creates a list of strings, where each entry is a string that occurred between delimiters.

  Here, assuming a string that is a sequence of text digits separated by spaces, the `split()` method will create a list of strings, where each string is a single text digit. By using this method, the digits of a line of a Sudoku puzzle file can be separated into a list, with each digit (represented as a string) occupying one entry in the list.

The list produced by using the string's `split()` method should then be converted into a list of `int` values. Thus, each row from the file can be converted into a list (of length 9) of digits, each representing a value in one row of the puzzle. Each such row can then be added to another list, which itself points to each row that is read and converted.

2.2 Printing the grid

Once the grid is constructed, it will be quite helpful to be able to print it in a manner that is easy to consume visually. Specifically, the `printGrid(grid)` function should reproduce the format of the text-based puzzle file, described above in Section 2.1, as a printed output.

2.3 Validating the grid

For any given grid, we want to know whether the placement of values (or, in our case, non-zero digits) follows the rules of the game. For that to be the case, there cannot be duplicate (non-zero) digits in any *row, column, or subgrid*.

Therefore, the `checkGrid(grid)` function should test each row, column, and subgrid for such duplicate digits. It must return `False` if any such duplicates are found, and `True` otherwise.

It is likely that this function should call on other functions of your creation. For example, it may be valuable to create a function such as `checkRow(grid, rowNum)` that checks row number `rowNum` from `grid` for duplicates. Similar `checkCol` and `checkSubgrid` functions would similarly be helpful to the `checkGrid` function. Moreover, these all are likely to be helpful in Project 2b, when we attempt to create a Sudoku-solving function.

3 Your assignment

Write the functions described above such that the `main()` function in Section 1 should work as advertised.

4 Submitting your work

On the CS submission site, upload your module, which I expect will be named `sudoku.py`.

This assignment is due by Monday, Nov-10, at 11:59 pm.