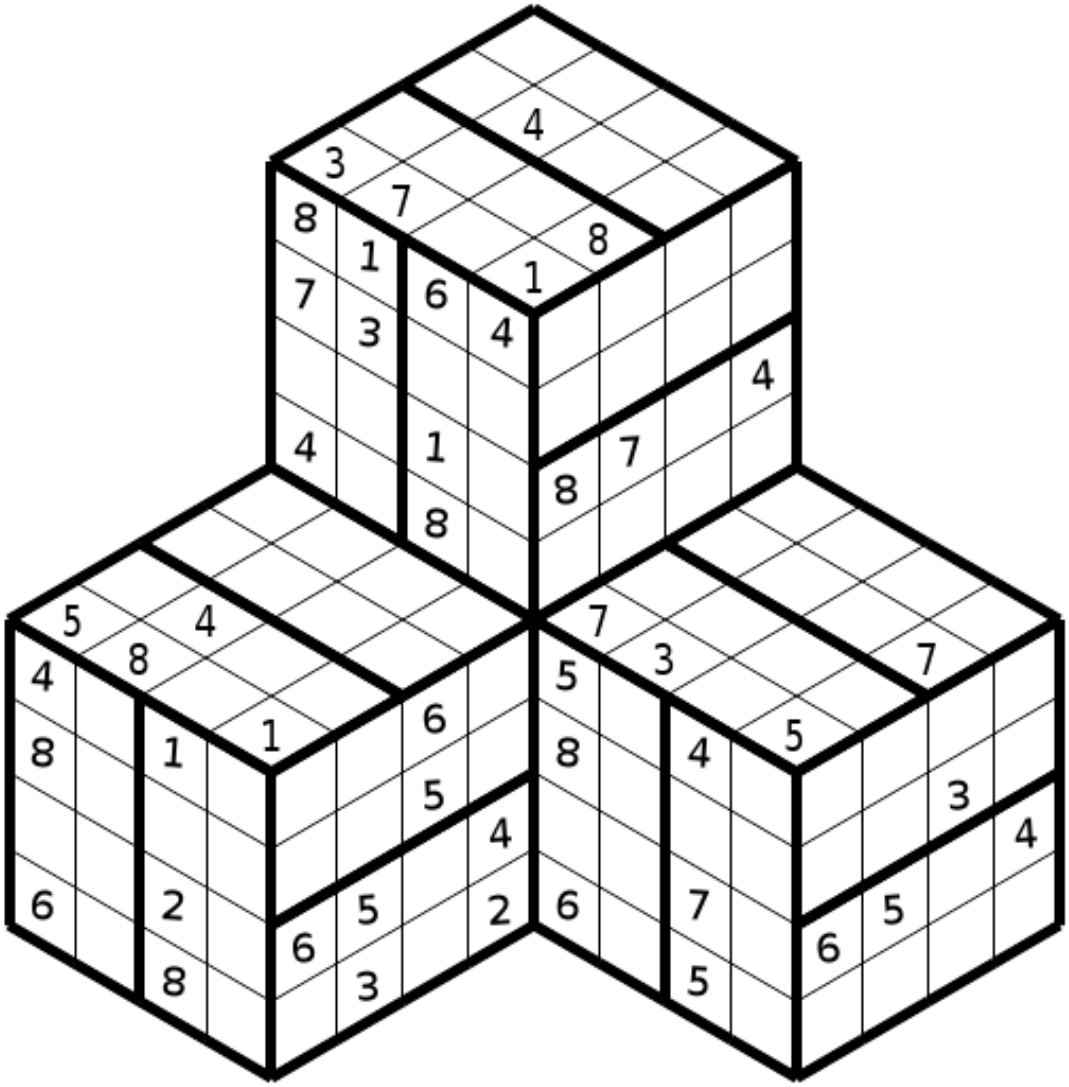
Universiteit van Amsterdam

Programmeertalen

Week 4 - Python

Jouke Witteveen

Sudoku



Introduction

Try to find pythonic code for the following exercises. Put those functions into `exercises.py`.

Exercise 1

Create a script that reads a list of lists of *integers*, one list per line, from a file. Use the following construct:

Python snippet

```
with open( file ) as f:
    # 'f' is now an iterator that iterates over the lines of 'file'
```

After reading from the file, the script should output the numbers, one line per list. The numbers should be separated by a single space and there should be no characters other than whitespace and digits. The input file follows the same layout, so that the output of the script is identical to the contents of the input file.

Exercise 2

Write a function that tests whether a sequence of `n` integers contains all the integers from 1 to `n`, inclusive.

Exercise 3

Write a generator that takes a sequence of `n` integers and produces all integers from 1 to `n` that are not part of the sequence.

Exercise 4a

Let a Sudoku grid be represented as a list of `n` lists of `n` elements, each of the `n` lists representing a row in the grid. Assuming `n` is a square, use nested list comprehension to build a list of lists of block members according to the Sudoku rules. That is, the members of any list in the resulting list of lists formed a block in the original Sudoku grid.

Exercise 4b

Given a row and column index in the original grid, come up with a means to calculate the row index of the corresponding block in the list of lists of block members.

> For the assignment you may, but need not, use any of the functions you have just written. The assignment leaves room for very clever techniques. If you choose to use some, be careful that you test early and thoroughly.

Sudoku

Description

Sudoku is a well-known logic puzzle. Background information and the rules of Sudoku are readily available on the internet. You will implement a general Sudoku solver, `sudoku.py`. This program should take as its sole argument the filename of a file containing a Sudoku puzzle of an arbitrary size `n` that is a square. If the Sudoku puzzle can be solved, the program outputs a solution. Call this program `sudoku.py`.

Input/Output Format

A Sudoku puzzle of size `n` is represented by `n` lines of `n` whitespace separated numbers, each line representing a row in the Sudoku grid in the obvious way. In the input, the number `0` represents an open cell. Hence, it is the task of the program to replace all `0`s with numbers so that the resulting grid is a valid solution to the Sudoku puzzle.

The Algorithm

The algorithm implemented in your program should perform an exhaustive search through potential solutions. As soon as a solution is found, the search is terminated and the solution output. However, there are special requirements.

Your program should not consider just any possible value for the open cells. For any open cell, it should only consider the values not present in either the row, the column, or the block to which the open cell belongs. Thus, for the Sudoku puzzle

1	2	3	4
0	0	2	1
2	1	4	3
0	4	1	2

a potential search tree would be

```
.
|-- (0,1) = 3
|
`-- (0,1) = 4
    |
    `-- (1,1) = 3
        |
        `-- (0,3) = 3
```

but depending on the order in which the open cells are processed, different search trees are possible.

> Additionally, your program must not depend on the call stack in its traversal of the search tree. This means that any form of recursion is disallowed and you should use your own variables to guide the search.

Extra Features

In order to obtain the maximal score, your program should terminate with a helpful message in case its input is incorrectly formatted or presented. Additionally, you should write a short (maximum 3 pages) report analysing the effect of not considering all possible numbers for each open cell. Your findings should be backed by experiments.

The extra features will only benefit your score if your program works according to the requirements. For example, failing gracefully does no good to your score if your solver is implemented recursively.

sectionSubmission The assignment has to be packed before submission, using the packing script provided on Blackboard. Files that are not packed using this script will not be corrected!