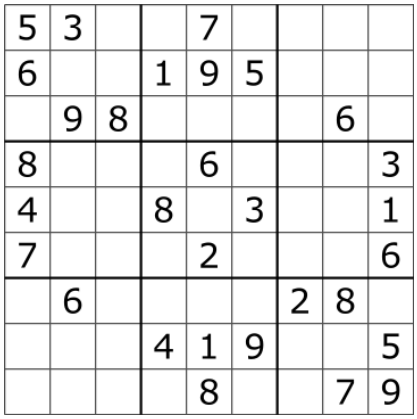
Mandatory assignment 1 – INF-1400 spring 2025

In this assignment, you are going to write a program that finds solutions to the popular puzzle known as Sudoku.



An introduction to sudoku

A sudoku board is a 9*9 board of integers. Some squares are filled in, and the goal is to fill in the rest. The rules are as follows:

1. A row, column and box (3*3 square as shown in the illustration) can only contain a single number once
2. All rows, columns and boxes must be filled with the numbers 1-9 for the game to be completed.

Precode

To help you start solving the assignment, you are given two Python files as well as some csv-files containing unsolved sudoku-boards for testing.

board.py

This file contains a class Board to represent a game that is played on a square board of an arbitrary size. The game could be chess, tick-tack-toe and a lot of other games, so you will make a subclass of Board to make it suitable for sudoku.

sudoku_reader.py

This file contains the class Sudoku_reader. The class will allow you to read sudoku boards from on the format given in the csv-files. The method next_board() returns a 9*9 2D list of integers.

Your assignment

You must implement the following classes:

SudokuBoard

This must be a subclass of the Board class from the precode, and objects of the class represent a single game of sudoku. The class should contain functionality allowing you to build a board out of squares and elements (see below), and a method that solves the board given the initial numbers.

Square

This class represents a single square on a sudoku board – thus, a SudokuBoard should contain exactly 81 (i.e., 9*9) squares. A square should be able to store its number, as well as references to the row, column and box it is part of. The Square class should contain a method to check whether a number is legal, checking whether it exists in its row, column or box. It should also contain a method that sets the number in the square if the value is legal.

Element

The Element class represents a row, column or box in your game, thus it should have references to 9 objects of Square, as all rows, columns and boxes keep track of 9 squares. It should have functionality to check whether any square it contains has a given value – you need to call this function in Square when checking if a value is legal.

In addition to the code, you must hand in a class diagram showing the relationships between the classes in your program. Remember to include inheritance arrows. We are not strict about diagrams conforming to the UML standard, but as a minimum your diagram should clearly show how the classes in your program are related to each other.

Sudoku algorithms

The simplest sudoku solving algorithm is the brute-force method. Solving a sudoku using brute force is done as follows:

1. Starting at the first square, find the lowest legal value between 1 and 9
2. Go to the next square and do the same
    a. If legal value is found, go to the next square and repeat step 2
    b. If NO values are legal, go back to the previous square and find the next legal value
3. The solution is found when all squares have gotten a number and no row, column or box contains the same value twice or more

Using this method, you can also find multiple solutions to a single sudoku board – simply keep going once you find a solution, roll back and try more values.

There are numerous other algorithms that you can experiment with, which will also be considerably quicker than the brute-force method. We do, however, recommend that you write a working brute-force solution first, just to make sure you have something you can hand in in case your more advanced algorithm doesn't work as intended.

Additional information

You are free to go beyond the assignment and write extensions such as a GUI to display sudoku boards, etc. We want you to have fun with programming and embrace creativity! You can of course also add additional functions, methods, classes and variables wherever you find it useful for your solution.

This is an individual assignment, and thus everything you hand in must be your own work unless you clearly specify that parts are quotes or references. Read more about the rules regarding mandatory assignments and exams on our Canvas page.

Report

In addition to the code, it is mandatory to hand in a report. The report should describe your implementation. Show that you understand how your implementation works. Remember that the code should be well enough commented and written so that it, in addition to the report, makes it easy for another person to understand how your program is put together.

In your report you must also try to answer the following questions; if you could choose the design of this program yourself, how would you do it? Would you use a different class structure? How is the cohesion and coupling in the solution outlined in this assignment?