Software Workshop I
Assignment 2
Marks available: 40
Date of assessment:
06/12/2023, Wednesday at 19h00
Set by: Abdul Wahid

Background:

This assignment is based on 3x3 Sudoku board game. However, for this assignment there are different rules required to play the game. These will be discussed below.

Instructions:

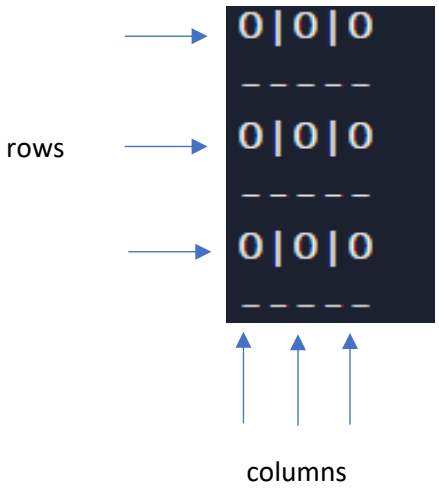
1. Create a new Python project called your SurnameStudentNumber, for example, Smith123456
2. Create a file called board-game.py
3. Copy the contents of the template given to you into this file and make use of the instructions below to complete the code for the game.
4. Do not put any code that gets user input.
5. You may add more functions, but you must not remove existing ones.
6. Please familiarise yourself with the template before starting.
7. There is no need to develop by GUI.

Submission instructions

Submit your work by zipping the project (for example, Smith123456 – see step 1 above) which includes your boardgame.py file and upload the .zip file to Canvas.

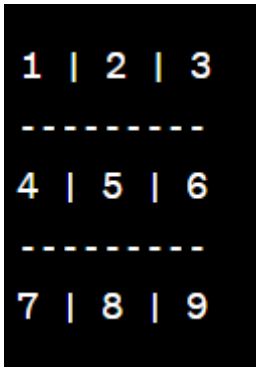
Now you are ready to start your assignment solution:

1. generate_board() **Function [5 marks]**
    a) The function generate_board() should create a **3x3** Sudoku board. **[3]**
    b) The default values of rows = 3 and columns = 3 have been provided which means that your Sudoku game is fixed to 3x3 only.
    c) The function should return a 2D list representing the Sudoku board. Each element initially holds the value 0, indicating an empty cell. (for a sample the screenshot is as follows) **[2]**



2. print_board(board) **Function [5 marks]**

The function print_board(board) should print the Sudoku board in the following format: The board is passed in as a parameter to the print_board(board) function. The number 1,2,3 …9 represent the already placed numbers. If this print_board(board) function is called in the start all these numbers will be zeros as mentioned in point 1(above).



3. place_number(board, number, row, col) **Function [10 marks]**

a) The function place_number(board, number, row, col) should allow players to place a number on the board. **[5]**
b) The board is passed in as a parameter.
c) The number parameter holds the value of a number between 1 and 9.
d) The row and col parameters represent the position where a number is to be placed on the board.
e) Additionally, a number cannot be placed in a position if a number already exists. Print a message to provide an error message. **[3]**
f) Return **True** if you have successfully placed the number else return **False** **[2]**

4. check_win(board, current_player) **Function [10 marks]**

a) The check_win(board, current_player) function receives board and current_player as its arguments. The current_player is used for determining the current player of the game.

b) The function check_win(board, current_player) should determine a winner based on the following rules:

- Rows win: If there is a row where the sum of numbers equals **15**, this is a win. **[3]**
- Columns win: If there is a column where the sum of numbers equals **15**, this is a win. **[3]**
- Diagonal win: If there is either a diagonal or a reversed diagonal where the sum of numbers equals **15**, this is a win. **[3]**

c) This function must return a string containing an appropriate message e.g. "player 1 or 2 wins by summing to **15** horizontally", "player 1 or 2 wins by summing to **15** vertically", "player 1 or 2 wins by summing to **15** diagonally, or "No win". **[1]**

5. def play_sudoku() **Function [10 marks]**:

a) The game must start with player 1. (already done in Skelton code)

b) the game should continue unless a player wins or all 9 slots are filled and no one have won the game **[1]**

c) the validation checks should be in this function

- the row and column number should be between 0, 1, or 2 (already done)

d) Call the place_number (board, number, row, col) function to place the number on the board **[3]**

e) Check for a winner using the check_win (board, current_player) function **[3]**

f) Update the current player for the next turn **[3]**

The following is a sample of the output when the input "`moves = [(5,0,0),(5,0,1),(5,1,1),(5,2,2)]`" is provided. These inputs are already given in the Skelton code (`#Automated moves`). The first digit in (5,0,0) represent the number to be placed, the next two digits represent row and column numbers. Similar automated moves are already provided in the Skelton code. You can uncomment those and check the output.

```
0|0|0
-----
0|0|0
-----
0|0|0
-----
It's Player 1's turn.
5|0|0
-----
0|0|0
-----
0|0|0
-----
It's Player 2's turn.
5|5|0
-----
0|0|0
-----
0|0|0
-----
It's Player 1's turn.
5|5|0
-----
0|5|0
-----
0|0|0
-----
```

```
It's Player 2's turn.
5|5|0
-----
0|5|0
-----
0|0|5
-----
2 wins by summing to 15 diagonally

Process finished with exit code 0
```