RELEVANT BACKGROUND INFORMATION AND PRE-REQUISITS

What is SOS game

SOS game is a two-player game where you have a 4 x 4 or 5 x 5 grid, each player can either put and S or O in a cell of his choice, we will refer to it as a "move". After a player makes a move, it becomes the other's turn to make a move.

The goal of the game is to make the most SOS sequences of characters. After each move, you count the number of sequences created by that move and add to the current player's score and the current player continues to make moves. The player with the most score wins.

How does the game work

Game starts with printing an empty grid. Then the first player is asked to enter the coordinates of the cell he wants to make his move in, he is asked to enter values in form "x y" where x represents the column and y is the row. After the player is asked to enter the character he want to play, either "s", "o", "S", "O". Then the game print the grid and the score of both players, then the game switches players. The process is repeated until the grid is full.

Once the grid is full, the game determines who is the winner or whether it was a tie or not.

Understanding the code.

First, define the global variable that are used across the script.

- board: is the 5 x 5 grid that is used to store the moves of each player, each cell has either 0 -> empty, 1 -> ‘O’ or 2-> ‘S’

- player_turn: is an integer representing the current player, it has value 0 for first player, 1 for the second player.

  you can easily toggle it from 0 to 1 and vice versa by using the ‘not’ operator.

- score0, score1: are the current score of each player

Second, defining the used functions:

- print_score(): print the score of each player

- print_board(): print the board variable ie the grid based on the value of each cell, 0 -> ‘_’ , 1 -> ‘O’, 2 ->’S’

  each two adjacent cell is separated by ‘|’ for appeal

- values_in(): it tries to take integer input from user, if the user fail the input a valid integer, it repeats the process recursively. It ensures the input is integer. it returns a tuple of the two integers inputted.

- get_play():it uses the values returned from values_in and checks if the values are a valid play; and gets the character played and applies it the grid

  A play is valid when:
  - board[x][y] is empty
  - x is in range (1, 5)

    - y is in range (1, 5)
- calculate_score(x,y): takes the coordinate of a cell, and calculates the number of sequences completed by the cell, then add the number to the score of the current player.

- gameover(): prints the winner who has the higher score, a tie in case the scores are equal.

- main(): is where the game loop is. first inform the players whose turn it is. Then, get play from the player, calculate the score caused by such play and add the score to the current player. After that, change the current player, repeat until the grid is full.