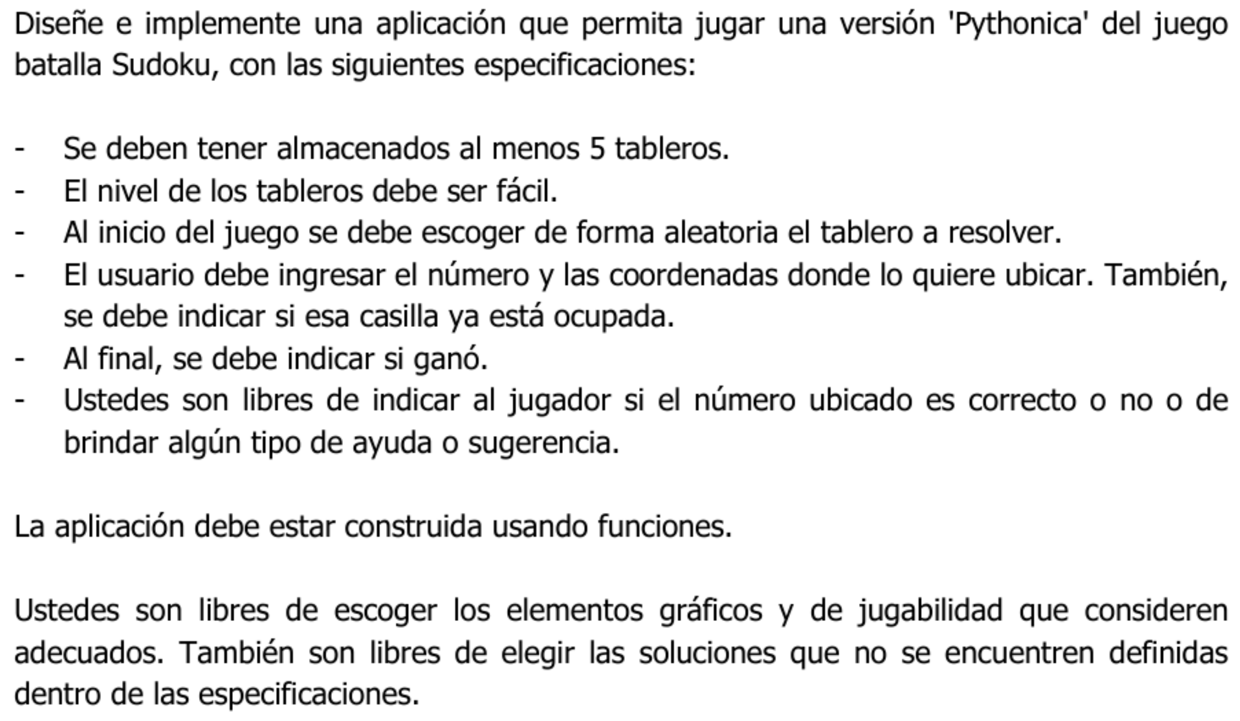
Group 5

Members

- Juan David Ocampo Gutierrez
- Nicolás Castro Pacheco
- Michell Valencia Berdugo
- Juan David Rivera Durán

Project #5: Sudoku



Diseñe e implemente una aplicación que permita jugar una versión 'Pythonica' del juego batalla Sudoku, con las siguientes especificaciones:

- Se deben tener almacenados al menos 5 tableros.
- El nivel de los tableros debe ser fácil.
- Al inicio del juego se debe escoger de forma aleatoria el tablero a resolver.
- El usuario debe ingresar el número y las coordenadas donde lo quiere ubicar. También, se debe indicar si esa casilla ya está ocupada.
- Al final, se debe indicar si ganó.
- Ustedes son libres de indicar al jugador si el número ubicado es correcto o no o de brindar algún tipo de ayuda o sugerencia.

La aplicación debe estar construida usando funciones.

Ustedes son libres de escoger los elementos gráficos y de jugabilidad que consideren adecuados. También son libres de elegir las soluciones que no se encuentren definidas dentro de las especificaciones.

Group 5

Step 1: Analysis

1	Identify possible inputs	● Number between 1 and 9 ● Coordinates (row and column, indices 0–8) ● Final answer (yes/no) to decide whether to play another round
2	Identify outputs	● Initial board randomly selected ● Board updated after each move ● Validation messages: ○ Invalid input (number out of range, coordinates out of range) ○ Cell already occupied ○ Invalid move according to Sudoku rules ● Victory message when board is completed ● Farewell message if player chooses not to play again
3	Decomposition	● get_board(boards): ○ Input: List of at least 5 predefined boards ○ Process: ■ Randomly select one board using a pseudo-random generator ■ Make a copy of the selected board to avoid modifying the original ○ Output: Initial board (9x9 matrix) ● display_board(board): ○ Input: 9x9 matrix (with numbers and zeros) ○ Process: ■ Iterate through rows and columns of the board ■ Print each number, showing a “.” instead of zeros for readability ■ Insert separators every 3 rows and columns to visualize Sudoku blocks ○ Output: Visual print of the board in console

		● get_move(board): ○ Input: Current board ○ Process: ■ Ask the user for a number between 1 and 9 ■ Ask for row (0–8) and column (0–8) ■ Validate that inputs are correct numbers (handle errors if letters or out-of-range values are entered) ■ Return row, column, and number ○ Output: Tuple (row, column, number) or error message if input is invalid ● validate_move(board, row, col, number): ○ Input: Current board, row, column, number ○ Process: ■ Check if the cell is already occupied (either initially or by a previous move) ■ Ensure the number is not repeated in the selected row ■ Ensure the number is not repeated in the selected column ■ Ensure the number is not repeated in the corresponding 3x3 subgrid ○ Output: True if valid, False otherwise ● update_board(board, row, col, number): ○ Input: Current board, row, column, valid number ○ Process: ■ Insert the number into the indicated cell ○ Output: Modified board

- board_completed(board):
    - Input: Current board
    - Process:
        - Iterate through all cells
        - Check if any cell contains zero (empty)
    - Output: True if no zeros remain, False if there are still empty spaces
- play_sudoku():
    - Input: None
    - Process:
        - Show welcome message
        - Select a random board using get_board
        - While the board is not complete:
            - Display the board
            - Ask the user for a move (get_move)
            - Validate the move (validate_move)
            - If valid → update the board (update_board)
            - If invalid → show error message and ask for another move
        - When the board is complete → show victory message
    - Output: On-screen interaction
- main():
    - Input: None
    - Process:
        - Run play_sudoku()
        - Ask the user if they want to play another round
        - If they say “yes”, repeat the process
        - If they say “no”, show farewell message
    - Output: On-screen interaction

4	Restrictions	● Numbers must be between 1 and 9 ● Coordinates must be within 0–8 ● Do not overwrite cells already filled in the initial board ● Follow Sudoku rules (no duplicates in row, column, or subgrid)

Step 2: Design

a. Pseudocode

```
START
   SHOW "==========================================="
   SHOW "        Welcome to Pythonic Sudoku        "
   SHOW "==========================================="
   REPEAT
      PLAY_SUDOKU()
      SHOW "Do you want to play again? (Y/N): "
      READ answer
   UNTIL answer != "Y"
   SHOW "Thanks for playing. See you next time!"
END

PROCEDURE GET_BOARD(list_of_boards)
   board = random choice from list_of_boards
   RETURN board
END PROCEDURE

PROCEDURE DISPLAY_BOARD(board)
   SHOW "Current Sudoku Board:"
   FOR each row in board DO
      FORMAT row with spaces and 3x3 separators
      SHOW row
   END FOR
END PROCEDURE

PROCEDURE ASK_MOVE()
   SHOW "Enter a number (1–9): "
   READ number
   SHOW "Enter row (0–8): "
   READ row
   SHOW "Enter column (0–8): "
   READ col
   RETURN row, col, number
END PROCEDURE

PROCEDURE VALIDATE_MOVE(board, row, col, number)
   IF board[row][col] != 0 THEN
      SHOW "This cell is already occupied."
      RETURN FALSE
   END IF

   IF number exists in row THEN
      SHOW "Number already in this row."
      RETURN FALSE
   END IF
```

```
    IF number exists in column THEN
        SHOW "Number already in this column."
        RETURN FALSE
    END IF

    IF number exists in 3x3 subgrid THEN
        SHOW "Number already in this 3x3 box."
        RETURN FALSE
    END IF

    RETURN TRUE
END PROCEDURE

PROCEDURE UPDATE_BOARD(board, row, col, number)
    board[row][col] = number
END PROCEDURE

PROCEDURE IS_BOARD_FULL(board)
    FOR each row in board DO
        IF row contains 0 THEN
            RETURN FALSE
        END IF
    END FOR
    RETURN TRUE
END PROCEDURE

PROCEDURE PLAY_SUDOKU()
    board = GET_BOARD(predefined_boards)
    WHILE NOT IS_BOARD_FULL(board) DO
        DISPLAY_BOARD(board)
        row, col, number = ASK_MOVE()
        valid = VALIDATE_MOVE(board, row, col, number)
        IF valid == TRUE THEN
            UPDATE_BOARD(board, row, col, number)
        ELSE
            SHOW "Invalid move. Try again."
        END IF
    END WHILE
    SHOW "Congratulations! You completed the Sudoku board!"
END PROCEDURE
```

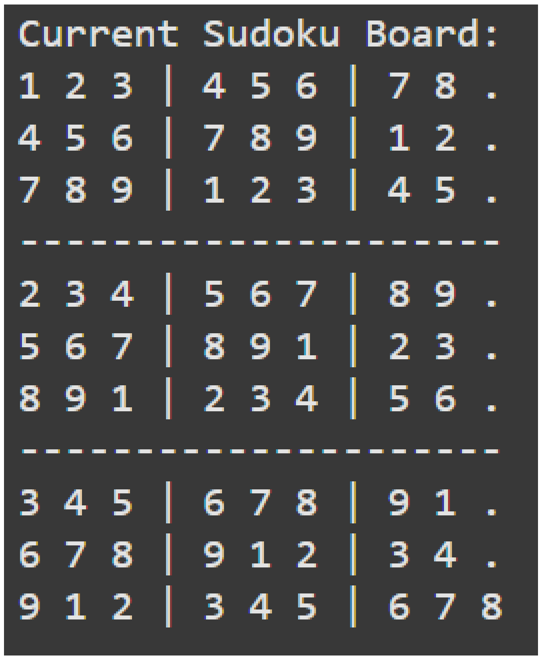
Group 5

b. Flowchart

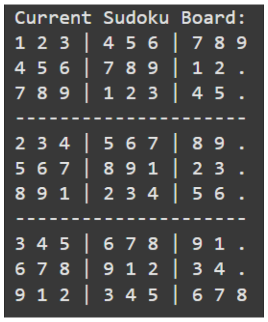
sudoku.drawio.png

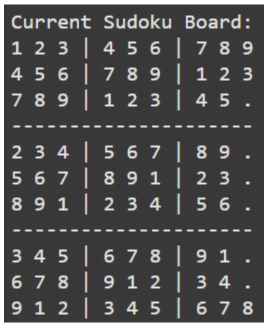
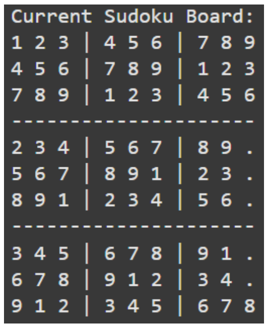
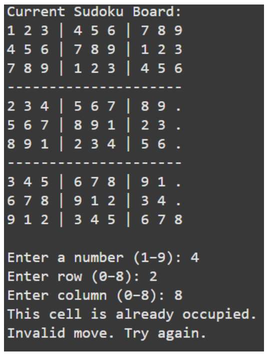
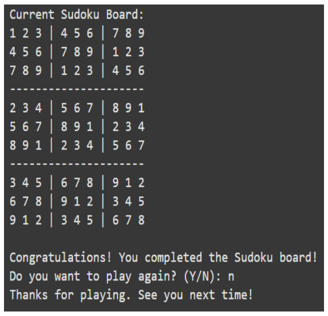
c. Desktop Test

Board:



```
Current Sudoku Board:
1 2 3 | 4 5 6 | 7 8 .
4 5 6 | 7 8 9 | 1 2 .
7 8 9 | 1 2 3 | 4 5 .
---------------------
2 3 4 | 5 6 7 | 8 9 .
5 6 7 | 8 9 1 | 2 3 .
8 9 1 | 2 3 4 | 5 6 .
---------------------
3 4 5 | 6 7 8 | 9 1 .
6 7 8 | 9 1 2 | 3 4 .
9 1 2 | 3 4 5 | 6 7 8
```

Iteration	User input	Process (functions and validations)	Output (board/message)
1	Num 9, row 0, col 8	validate_move: Correct update_board: Correct	 Current Sudoku Board: 1 2 3 | 4 5 6 | 7 8 9 4 5 6 | 7 8 9 | 1 2 . 7 8 9 | 1 2 3 | 4 5 . --------------------- 2 3 4 | 5 6 7 | 8 9 . 5 6 7 | 8 9 1 | 2 3 . 8 9 1 | 2 3 4 | 5 6 . --------------------- 3 4 5 | 6 7 8 | 9 1 . 6 7 8 | 9 1 2 | 3 4 . 9 1 2 | 3 4 5 | 6 7 8

2	Num 3, row 1, col 8	validate_move: Correct update_board: Correct	
3	Num 6, row 2, col 8	validate_move: Correct update_board: Correct	
4	Num 4, row 2, col 8	validate_move: Incorrect update_board: Incorrect	
…	…	…	…
Final	-	Is_board_full == True	

```
Current Sudoku Board:
1 2 3 | 4 5 6 | 7 8 9
4 5 6 | 7 8 9 | 1 2 3
7 8 9 | 1 2 3 | 4 5 .
---------------------
2 3 4 | 5 6 7 | 8 9 .
5 6 7 | 8 9 1 | 2 3 .
8 9 1 | 2 3 4 | 5 6 .
---------------------
3 4 5 | 6 7 8 | 9 1 .
6 7 8 | 9 1 2 | 3 4 .
9 1 2 | 3 4 5 | 6 7 8
```

```
Current Sudoku Board:
1 2 3 | 4 5 6 | 7 8 9
4 5 6 | 7 8 9 | 1 2 3
7 8 9 | 1 2 3 | 4 5 6
---------------------
2 3 4 | 5 6 7 | 8 9 .
5 6 7 | 8 9 1 | 2 3 .
8 9 1 | 2 3 4 | 5 6 .
---------------------
3 4 5 | 6 7 8 | 9 1 .
6 7 8 | 9 1 2 | 3 4 .
9 1 2 | 3 4 5 | 6 7 8
```

```
Current Sudoku Board:
1 2 3 | 4 5 6 | 7 8 9
4 5 6 | 7 8 9 | 1 2 3
7 8 9 | 1 2 3 | 4 5 6
---------------------
2 3 4 | 5 6 7 | 8 9 .
5 6 7 | 8 9 1 | 2 3 .
8 9 1 | 2 3 4 | 5 6 .
---------------------
3 4 5 | 6 7 8 | 9 1 .
6 7 8 | 9 1 2 | 3 4 .
9 1 2 | 3 4 5 | 6 7 8

Enter a number (1-9): 4
Enter row (0-8): 2
Enter column (0-8): 8
This cell is already occupied.
Invalid move. Try again.
```

```
Current Sudoku Board:
1 2 3 | 4 5 6 | 7 8 9
4 5 6 | 7 8 9 | 1 2 3
7 8 9 | 1 2 3 | 4 5 6
---------------------
2 3 4 | 5 6 7 | 8 9 1
5 6 7 | 8 9 1 | 2 3 4
8 9 1 | 2 3 4 | 5 6 7
---------------------
3 4 5 | 6 7 8 | 9 1 2
6 7 8 | 9 1 2 | 3 4 5
9 1 2 | 3 4 5 | 6 7 8

Congratulations! You completed the Sudoku board!
Do you want to play again? (Y/N): n
Thanks for playing. See you next time!
```