Minesweeper Lab

Overview

Minesweeper is a classic logic puzzle originally played on personal computers. The game features a grid of hidden squares, with “mines” scattered throughout. Your goal is to clear all safe squares without detonating a mine. Clues are given by numbers showing how many mines are adjacent to a square.

You and your team will design and implement Minesweeper in Python. Along the way, you will practice:
- 3D lists
- Input validation
- Iteration
- Modular programming (multiple .py files)
- Testing and debugging
- Computational thinking: decomposition, flowcharts, and algorithm design

Step 1: Understanding the Game

Steps of Play:
1. Declare a board size (rows × columns). (ask for input)
2. Place a set number of mines randomly on the board. (ask for input)
3. For each non-mine square, count how many mines touch it (8-neighbors: NE, N, NW, W,
   SW, S, SE, E).
4. Store that number in the cell. If 0 → leave blank.
5. Keep this as your base board (hidden from the player).
6. Create a display board for the player filled with ♦.
7. On each move, the player chooses a cell:
   - If it’s a mine → Game Over.
   - If it’s a number → reveal just that cell.
   - If it’s blank → reveal it and flood-fill all connected blanks until bordered by numbers.
   - Repeat until all safe cells are revealed (→ You Win!).

Step 2: Example Base Board

Suppose we have a 5×5 board with 2 mines:

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

Step 3: Player’s Display Board

At the start, show a board of hidden squares:

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

Step 4: Sample Playthrough

Player chooses (1,1)

	0	1	2	3	4
0	♦	♦	♦	♦	♦
1	♦		♦	♦	♦
2	♦	♦	♦	♦	💣
3	♦	♦	♦	♦	♦
4	♦	♦	💣	♦	♦

Since (1,1) is blank , all adjacent blanks and numbers are revealed:

We place on the stack all the blanks uncovered

	0	1	2	3	4
0				♦	♦
1				♦	♦
2				♦	💣
3	♦	♦	♦	♦	♦
4	♦	♦	💣	♦	♦

stack: 0:0, 0:1, 0:2,
1:2, 2:2, 2:1,
2:0, 1:0

0:0, 0:1 -> nothing to uncover

0:2 -> uncover adjacent cells

	0	1	2	3	4
0					♦
1				1	♦
2				♦	💣
3	♦	♦	♦	♦	♦
4	♦	♦	💣	♦	♦

stack: 1:2, 2:2, 2:1, 2:0, 1:0,
0:3

1:2 -> uncover adjacent cells

	0	1	2	3	4
0					♦
1				1	♦
2				1	💣
3	♦	♦	♦	♦	♦
4	♦	♦	💣	♦	♦

stack: 2:2, 2:1, 2:0, 1:0, 0:3

2:2-> uncover adjacent cells

	0	1	2	3	4
0					♦
1				1	♦
2				1	💣
3	♦	1	1	2	♦
4	♦	♦	💣	♦	♦

stack: 2:1, 2:0, 1:0, 0:3

2:1-> uncover adjacent cells

	0	1	2	3	4
0					♦
1				1	♦
2				1	💣
3		1	1	2	♦
4	♦	♦	💣	♦	♦

stack: 2:0, 1:0, 0:3, 3:0

2:0, 1:0 -> nothing to uncover
0:3-> uncover adjacent cells

	0	1	2	3	4
0					
1				1	1
2				1	💣
3		1	1	2	♦
4	♦	♦	💣	♦	♦

stack: 3:0, 0:4

3:0-> uncover adjacent cells

	0	1	2	3	4
0					
1				1	1
2				1	💣
3		1	1	2	♦
4		1	💣	♦	♦

stack: 0:4, 4:0

0:4, 4:0 -> nothing to uncover, stack is empty. Next move

Step 5: Lab Assignment

Requirements:

1) You may work in teams of 3
2) Each member must be responsible for at least 2 functions
3) Each function must be in a separate file
4) Suggested function/file list:
   count_adjacent_mines.py
   game_won.py
   get_adjacent_cells.py
   get_validated_input.py
   globals.py (given)
   initialize_board.py
   is_mine_at.py
   place_random_mines.py
   play_minesweeper.py
   print_board.py (given)
   update_board.py
   utils.py (given)

- You must use the files that are given in red
- Based on your design, you may change, add or delete suggested functions

5) Include a flow chart of the game
6) You win the game by uncovering all the ♦ that are not covering 💣
7) Each team may do 2 webex video conferences with me provided all team members are present before Thursday, October 23rd.

Rubric (100 points)

- Flowchart + design docs (10)
- Board initialization & mine placement (20)
- Reveal algorithm correctness (25)
- Input handling & UX (10)
- Win/lose logic (10)
- Code quality (docstrings, naming, modularity) (10)
- Team collaboration (commit history + file ownership) (15)