Minesweeper, Part 3: A Text-Based Minesweeper Game

SHAPE Summer 19, Introductory Computer Science

July 5, 2019

Update the minesweeper file you worked on yesterday.

In today's sheet, we will create a simple text-based version of Minesweeper. We will add the User Inteface next week.

1 Changing the Display Once More

On the last sheet, you wrote a function `user_view(gameboard)` that displays the gameboard as seen by the user. In our Minesweeper game, the user will type in x,y coordinates to uncover bombs. To make that easier, print the row and column numbers next to the user view of the map. Modify the `user_view` function so that it displays the user map as in the following example

```
   0 1 2 3 4 5 6 7
   ---------------
0| ? ? ? ? 1 0 0 0
1| ? ? 1 1 1 0 1 1
2| ? ? 1 0 0 0 1 ?
3| ? ? 2 2 2 2 2 ?
4| ? ? ? ? ? ? ? ?
5| ? ? ? ? ? ? ? ?
6| ? ? ? ? ? ? ? ?
7| ? ? ? ? ? ? ? ?
```

2 Creating the Game

Write a function `game(height, width, n)` that plays the Minesweeper game. Call `game` at the end of your program at the top level (i.e. no indentation). You can call the game with a pre-selected width and height. Within the `game()` function, first create a new gameboard and then bury n mines. Then enter into the main game loop as described below.

2.1 Checking if the Game is Won

The game is won if the player has uncovered all cells in which no bomb is buried. Write a function `check_won(gameboard)` that returns `True` if the game is won, and `False` otherwise.

2.2 Game Loop

The main game will run in a loop. In each iteration, the game first displays the current user view of the gameboard. It then asks the user to enter an x,y coordinate pair. It updates the board based on the user input. If the game is not won or lost, it should run through the loop again.

- If the coordininates entered have a bomb in them, end the game immediately. In this case, print out the actual board using `print_mines` to show the player where the bombs were buried.
- If the coordinates entered do not have a bomb in them, run the `uncover_board` function to uncover an area on the board.
- Check if the user has won the game by calling the `check_won(gameboard)` function. In this case, end the game and print a message.

The user input should be one line of two comma-separated integer values. So the user would type "1,2" instead of two separate lines.

3 Measuring Time (optional)

Measure the time it takes from the start of the game to the end of the game. You can measure time as shown in the following example:

```
import time

start_time = time.time()

# your code is in between these lines

elapsed_time = time.time() - start_time
```

Display the current time before each turn of the game together with the game board. Also display the time when the user has won or lost.

4 Marking Bombs (optional)

Allow the user to mark certain cells as bombs (red flags). The user should enter "m4,2" instead of just the coordinates to mark cell 4,2. Display these marked cells in the user view. How would you represent marked cells on the board? If you can't represent them on the board, can you come up with another representation?