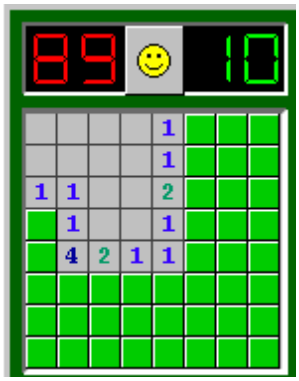
CS 101
Fall 2013
Program 7
This rather large program is an implementation of Minesweeper. You will be given the outline of a Minesweeper class, which you will need to complete and use in your actual program.

Overview
Minesweeper is a popular game that has been a standard part of Windows for years. (Start->Games-> Minesweeper or Start->Accessories->Games->Minesweeper.) It is also available in several forms online (one is at http://www.mine-sweeper.com/).



How to play:
The player is given a grid of size N x N hidden cells. Any hidden cell contains one of the following:

a) A Mine
b) Blank Cell
c) A number (call it X) indicating that the cell has X mines in its neighborhood. Note that a blank cell could be represented by the number 0.

The mines are randomly placed on the grid.
The rules of the game:

1) If the player chooses a cell that has a hidden mine, the game is over. BOOM!
2) If the player chooses a cell that has a hidden (nonzero) number, then the cell is uncovered and the number is now visible.
3) If the player chooses a cell that doesn't have a mine or number, the ripple effect takes place (see "Ripple effect," below) uncovering all the blanks in the neighborhood until either the grid boundary is reached or a number is reached along its path.

The objective of the game is to identify, logically, all the mines on the grid.

Task
Your task is to implement the game in Python *using classes.* Before implementing the game, please play the game at least 5 times. It will help you understand the project.

Program Specification:

1) To reduce the complexity of the program, we'll fix the size of the grid to 5x5, with 3 mines on the grid.
2) On the grid, blank cells can be represented with a number 0, indicating that the cell has 0 mines in its neighborhood. ***The neighborhood of a cell consists of all the cells that are immediately adjacent to the given cell horizontally, vertically, or diagonally.***
3) The program should have the following error checks:
    a. If a user enters non-integer input for the row or column of the cell, notify the user that the input is incorrect and prompt again.
    b. If the user chooses a cell that's not on the grid, report an error and ask the user for a valid cell.
    c. Any others you can think of…
4) If the player gives the position of a grid cell that's already open (uncovered), do nothing.
5) Part of the grade will be the appropriateness of your class, methods, and any functions you use. The quality of the code will matter as well as the performance. ***Your program must be object-oriented to receive credit.***

High level algorithm:

1) Complete the class for the Minesweeper game. A few basic data structures and methods have been provided for you. Your implementation must meet the specifications for each method. This includes error handling. You may need some additional methods as well. If so, provide the same specifications (parameters, return, user interaction, modification to data structures, errors handled) as the examples. If a method states that it assumes parameters have been checked, then it should not check those parameters; the code that calls that method should check them before the call.
2) Randomly place the mines on the grid. Complete the placeMines() method for doing this.
3) After that, get the counts for each non-mine cell on the grid. Complete the getCounts() method to do this.
4) At this stage, the grid is ready with the mines and numbers, and the player could start playing.

Actual grid

```
1 1 1 1 1
1 M 1 2 M
3 3 2 2 M
M M 1 1 1
2 2 1 0 0
```

Game in progress (player view):

```
  1 2 3 4 5
1 H H 1 1 1
2 1 H 1 2 H
3 H H 2 2 H
4 H H 1 1 1
5 H H 1 0 0
```

5) Display the grid in the form of the "Player view." In the beginning it will be all H's, and as the player uncovers the view you will update the view to reflect that.
6) The display of the player view should include row and column numbers to make it easier to play. Rows are 1-5 from top to bottom, columns are 1-5 from left to right, as shown.
7) Prompt the player for the position (row, column) on the grid that he/she wants to explore.
8) If the player hits a mine, give a message that a mine was hit and the game is over. Display the actual state of the grid (with everything uncovered, the 'actual view' above).
9) If the player hits a number (nonzero), just show the number in the updated player view.
10) If the player hits a blank, then show the ripple effect (see "Ripple effect," below).
11) If the player has uncovered all non-mine cells, the player has won. Print a message congratulating the player and offering to play again.
12) Repeat steps 5-11 until the game is over (the player hits a mine, or all cells without mines are uncovered).
13) When the game is over (won or lost), ask the user if they want to play again; if so, continue; otherwise exit.
14) Most of the above could be programmed in the main program, calling methods of the Minesweeper class to carry out such tasks as uncovering a cell, displaying the board, etc.

Ripple effect:
Case 1: Requirement. This is expected in your program:
If a player chooses a blank cell on the grid, consider only the row and column to which the cell belongs. Starting from the cell, move horizontally (both left and right) and vertically (both up and down) until you encounter a nonzero number or the edge of the grid, whichever comes first. If a number is encountered, it should be uncovered and then the ripple effect should stop.

Case 2: Not expected. 10 POINTS EXTRA CREDIT IF IMPLEMENTED CORRECTLY.
In the ideal case, all the neighbors of the cell should be considered. Then, for each of the neighbors, all of *those* cells' neighbors should be considered, and so on. Additionally, the boundary conditions have to be dealt with, i.e. when we reach an edge or corner of the grid during the ripple effect. (Hint: Think recursively. Or, use a queue and breadth-first search.)

Deliverables:
A completed Minesweeper.py file. Do not change the name of this file.
A completed program file that imports Minesweeper to use a Minesweeper object to play the game.

For each class method that you add, and each function in your main program, you must specify:

- What that method expects as input via parameters (if any, or a statement that no input is expected);
- What is returned (if anything, or a statement that nothing is returned);
- What user interaction there is (input or output, including to any files), or, again, a statement that there is none;
- How any data structures of the object are modified by the function (or, surprise, a statement that there are no modifications to the object's data); and
- What errors (if any) that method handles.

This documentation should include any other features of the method that are relevant.

Tips & Hints:

1) As a starting point, identify the variables and methods (member functions) that your Minesweeper class will need. Complete the stub methods provided for you. Test each separately before going on to the next.
2) Start with a basic game; for example, you may want to start with a version that places mines and then asks the user for input, and just prints out a brief message whether the row and column specified were within range. Then add code handling each case (blank cell, mined cell, cell with a mine in the neighborhood) separately.
3) Test early, test often.
4) Most of the development work is going to go into the Minesweeper class. Your main program will probably be surprisingly short.