MineSweeper

MineSweeper was one of the first games available on Windows.

An ADT for a MineSweeper field:

- `Field(nrows, ncols, nbombs)` Create field and spread bombs.
- `display(show_bombs)` Display the field (with or without showing the bombs).
- `cell(row, col)` Get contents of cell.
- `uncover(row, col)` Open a cell. Returns `True` if there was a bomb.
- `mark(row, col)` Mark a cell.
- `all_visible()` Are all cells marked or visible?
- `num_marks()` Return number of marks on the field.

Two-dimensional Arrays

There are two strategies for implementing a higher-dimensional array.

- Make one array/List for the rows. Each slot is a reference to one array/List for the columns.
- Make a large array/List with `nrows * ncols` slots.

List of Lists

Creating the field:

```
self._field = [ None ] * nrows
for i in range(nrows):
  self._field[i] = [ '.' ] * ncols
```

Reading a cell:

```
return self._field[row][col]
```

Setting a cell:

```
self._field[row][col] = el
```


A single List

Creation:

```
self._field = [ '.' ] * (nrows * ncols)
```

Reading a cell:

```
return self._field[row * self._ncols + col]
```

Setting a cell:

```
self._field[row * self._ncols + col] = el
```

Comparison

For two-dimensional fields, the difference between the two techniques is not that large.
Lists of lists have nicer code for accessing elements, are slightly faster, but need a bit more memory.

For three- and more-dimensional fields, the "one long list" method is better, because it avoids wasting a lot of space.
One can speed up accessing elements by precomputing the factors for accessing the dimensions.

Minesweeper, Improved

When uncovering a cell with no neighboring bombs, we can immediately open all neighbors.

A good implementation does this automatically:

```
What cell do you want to check? F14
                      1 1 1 1 1 1 1 1 1 2
  1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
A . . . . . . . . . . . . . . . 1   1 . 1
B . . . . . . . . . . . . . . . 2   2 . 2
C . . . . . . . . . . . . . . . 2   1 . 1
D . . . . . 1 . . . . . 1 1 1 1   1 1 1
E . . . . . . . . . . . . 1
F . . . . . . . . . . 1 . 2 1         1 1 1
G . . . . . . . . . . . . 1 1 2 2 1 1 . .
H . . . . . . . . . . . . . . . . . . . .
```