Minesweeper

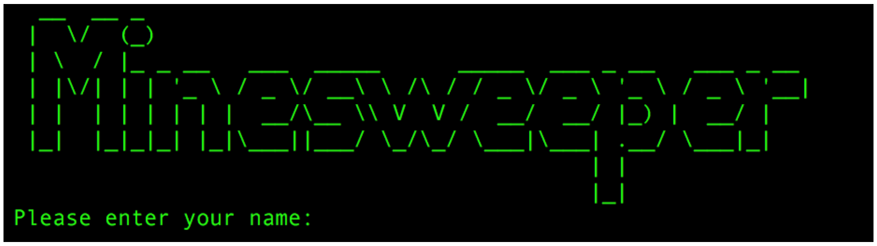
A Command Line based ASCII Game

I grew up playing the traditional Microsoft version of the Minesweeper game; alas, the MacBook does not come with this game pre-installed, so I decided to program my own. The gameplay is really simple — First, you select a difficulty level; then you select a cell by inputting its position and lastly you decide whether to put a flag at the cell in interest. This manual assumes that the reader is familiar with the minesweeper rules, if not click this.

In order to enjoy this game, direct to the folder with the `minesweeper.py` script using command line. And then you fire up the game by typing the following command —

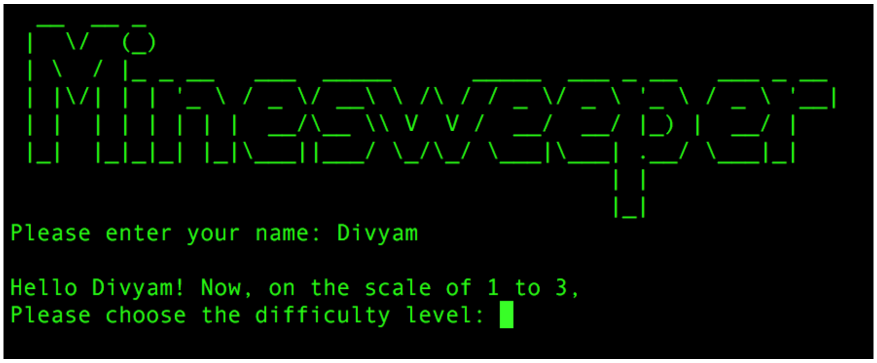
```
python minesweeper.py
```

This command should start the game and prompt user to insert his/her name like —



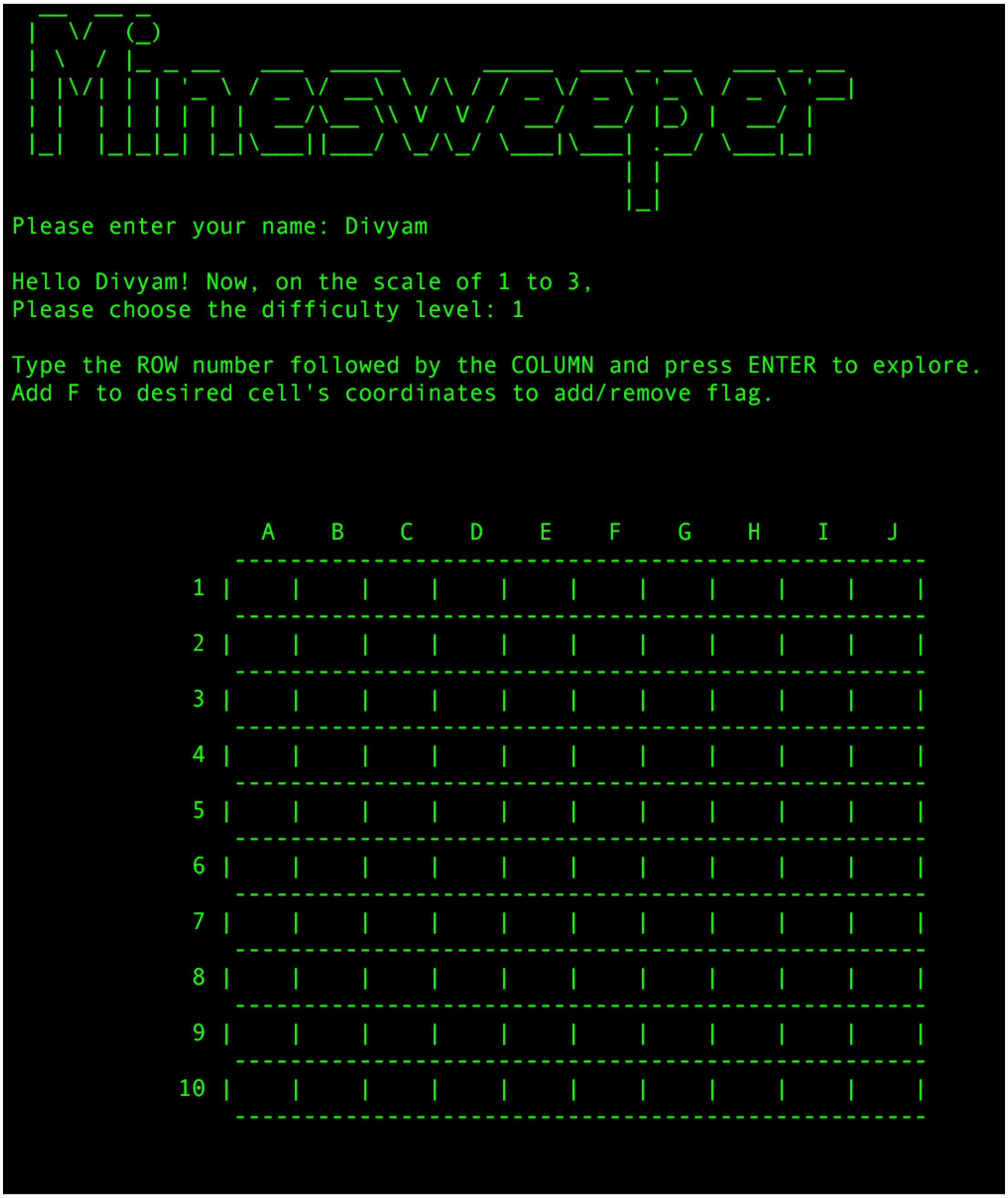
```
 __  __ _
|  \/  (_)_ __   ___  _____      _____  ___ _ __   ___ _ __
| |\/| | | '_ \ / _ \/ __\ \ /\ / / _ \/ _ \ '_ \ / _ \ '__|
| |  | | | | | |  __/\__ \\ V  V /  __/  __/ |_) |  __/ |
|_|  |_|_|_| |_|\___||___/ \_/\_/ \___|\___| .__/ \___|_|
                                           |_|
Please enter your name:
```

After typing your name, press enter —



```
 __  __ _
|  \/  (_)_ __   ___  _____      _____  ___ _ __   ___ _ __
| |\/| | | '_ \ / _ \/ __\ \ /\ / / _ \/ _ \ '_ \ / _ \ '__|
| |  | | | | | |  __/\__ \\ V  V /  __/  __/ |_) |  __/ |
|_|  |_|_|_| |_|\___||___/ \_/\_/ \___|\___| .__/ \___|_|
                                           |_|
Please enter your name: Divyam

Hello Divyam! Now, on the scale of 1 to 3,
Please choose the difficulty level: █
```

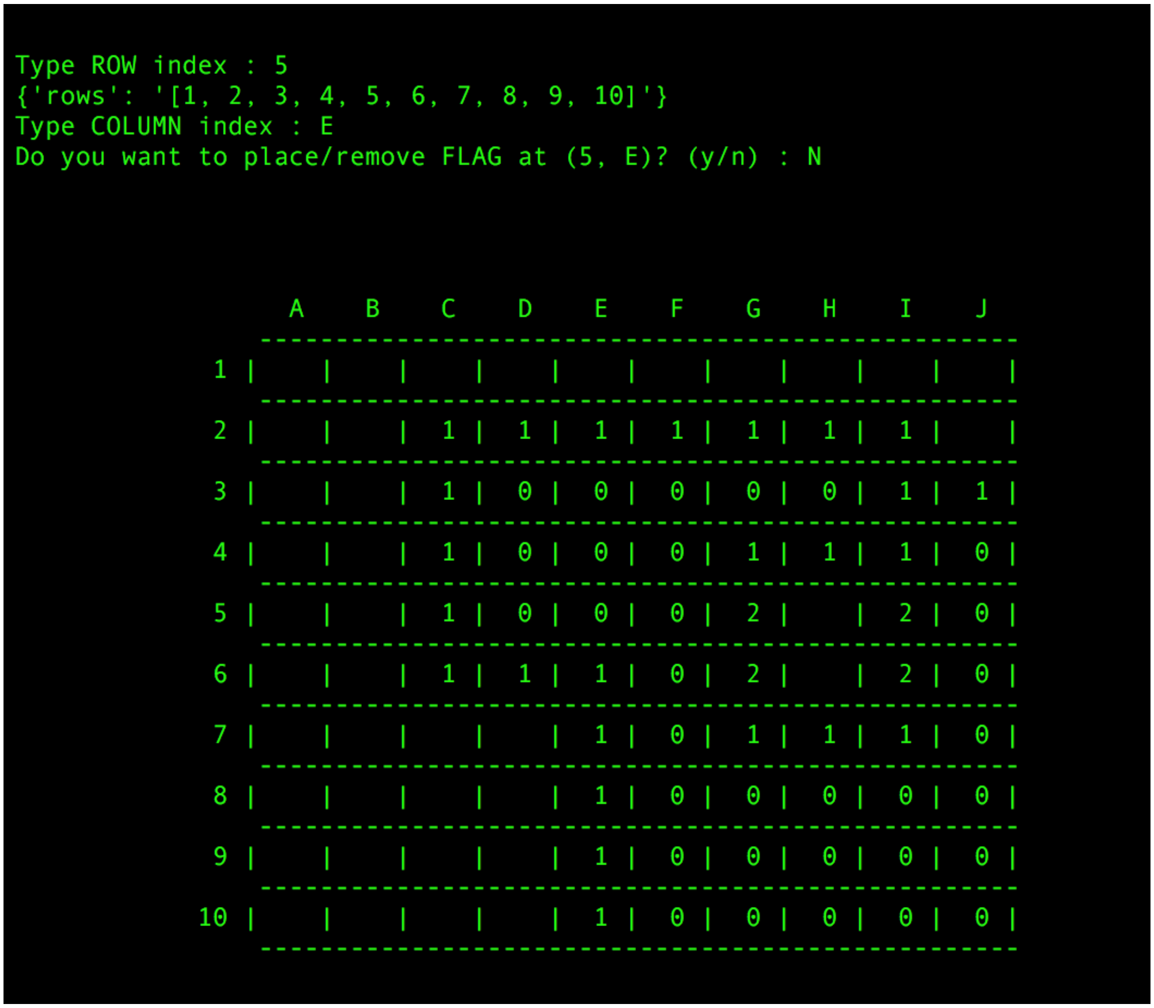
Now, select the difficulty level with 1 being easiest and 3 being hardest —



```
 __  __ _
|  \/  (_)_ __   ___  _____      _____  ___ _ __   ___ _ __
| |\/| | | '_ \ / _ \/ __\ \ /\ / / _ \/ _ \ '_ \ / _ \ '__|
| |  | | | | | |  __/\__ \\ V  V /  __/  __/ |_) |  __/ |
|_|  |_|_|_| |_|\___||___/ \_/\_/ \___|\___| .__/ \___|_|
                                           |_|
Please enter your name: Divyam

Hello Divyam! Now, on the scale of 1 to 3,
Please choose the difficulty level: 1

Type the ROW number followed by the COLUMN and press ENTER to explore.
Add F to desired cell's coordinates to add/remove flag.



             A    B    C    D    E    F    G    H    I    J
           ---------------------------------------------------
         1 |    |    |    |    |    |    |    |    |    |    |
           ---------------------------------------------------
         2 |    |    |    |    |    |    |    |    |    |    |
           ---------------------------------------------------
         3 |    |    |    |    |    |    |    |    |    |    |
           ---------------------------------------------------
         4 |    |    |    |    |    |    |    |    |    |    |
           ---------------------------------------------------
         5 |    |    |    |    |    |    |    |    |    |    |
           ---------------------------------------------------
         6 |    |    |    |    |    |    |    |    |    |    |
           ---------------------------------------------------
         7 |    |    |    |    |    |    |    |    |    |    |
           ---------------------------------------------------
         8 |    |    |    |    |    |    |    |    |    |    |
           ---------------------------------------------------
         9 |    |    |    |    |    |    |    |    |    |    |
           ---------------------------------------------------
        10 |    |    |    |    |    |    |    |    |    |    |
           ---------------------------------------------------
```

Wow! Look you have a nice grid now! From here, you can start playing the game by selecting a cell from the command like by inserting the row and the column. Also you have to decide whether to put a flag at this cell or not.

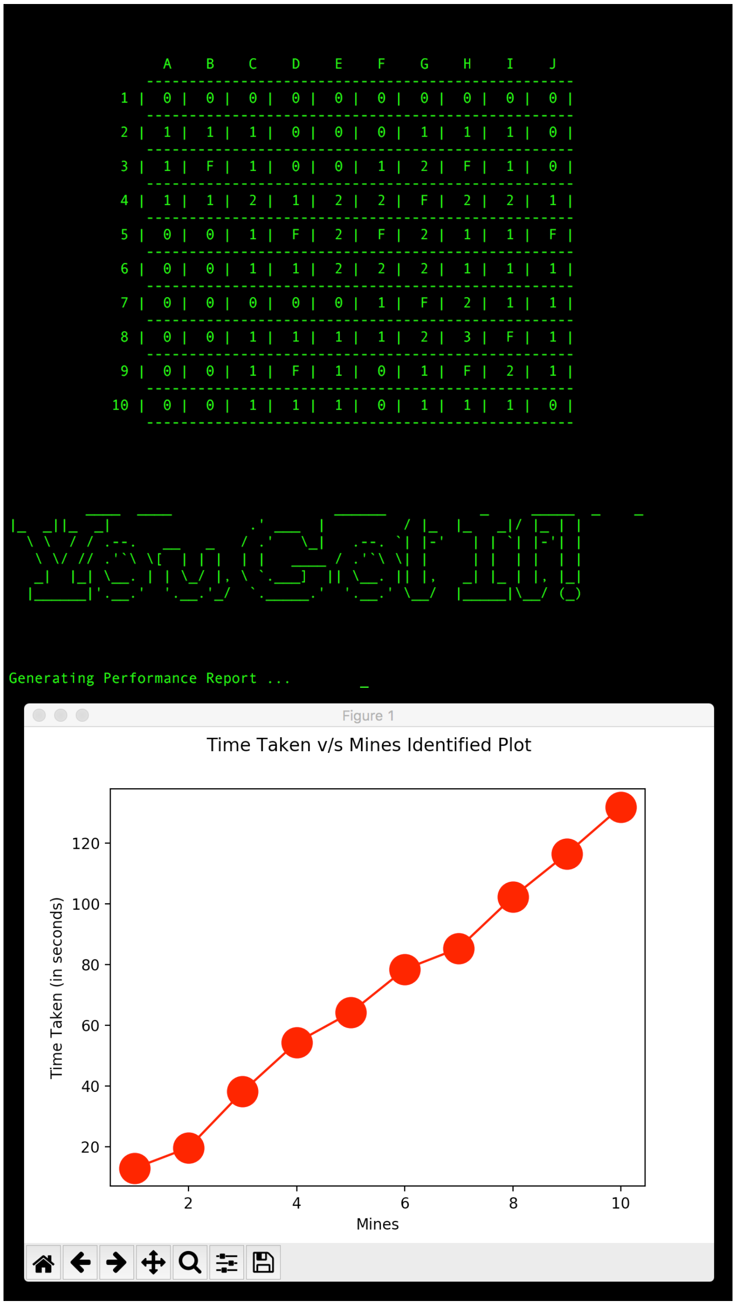
Let's go for row 5 and column E in our case —



```
Type ROW index : 5
{'rows': '[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]'}
Type COLUMN index : E
Do you want to place/remove FLAG at (5, E)? (y/n) : N




                  A    B    C    D    E    F    G    H    I    J
                --------------------------------------------------
             1 |    |    |    |    |    |    |    |    |    |    |
                --------------------------------------------------
             2 |    |    |  1 |  1 |  1 |  1 |  1 |  1 |  1 |    |
                --------------------------------------------------
             3 |    |    |  1 |  0 |  0 |  0 |  0 |  0 |  1 |  1 |
                --------------------------------------------------
             4 |    |    |  1 |  0 |  0 |  0 |  1 |  1 |  1 |  0 |
                --------------------------------------------------
             5 |    |    |  1 |  0 |  0 |  0 |  2 |    |  2 |  0 |
                --------------------------------------------------
             6 |    |    |  1 |  1 |  1 |  0 |  2 |    |  2 |  0 |
                --------------------------------------------------
             7 |    |    |    |    |  1 |  0 |  1 |  1 |  1 |  0 |
                --------------------------------------------------
             8 |    |    |    |    |  1 |  0 |  0 |  0 |  0 |  0 |
                --------------------------------------------------
             9 |    |    |    |    |  1 |  0 |  0 |  0 |  0 |  0 |
                --------------------------------------------------
            10 |    |    |    |    |  1 |  0 |  0 |  0 |  0 |  0 |
                --------------------------------------------------
```

Looks like 5E was really a good choice! Now by following the traditional minesweeper rules, we can go ahead and finish this game. Be right back!

Phew! That took a while. Finally, we have —



```
      A    B    C    D    E    F    G    H    I    J
    --------------------------------------------------
 1 |  0 |  0 |  0 |  0 |  0 |  0 |  0 |  0 |  0 |  0 |
    --------------------------------------------------
 2 |  1 |  1 |  1 |  0 |  0 |  0 |  1 |  1 |  1 |  0 |
    --------------------------------------------------
 3 |  1 |  F |  1 |  0 |  0 |  1 |  2 |  F |  1 |  0 |
    --------------------------------------------------
 4 |  1 |  1 |  2 |  1 |  2 |  2 |  F |  2 |  2 |  1 |
    --------------------------------------------------
 5 |  0 |  0 |  1 |  F |  2 |  F |  2 |  1 |  1 |  F |
    --------------------------------------------------
 6 |  0 |  0 |  1 |  1 |  2 |  2 |  2 |  1 |  1 |  1 |
    --------------------------------------------------
 7 |  0 |  0 |  0 |  0 |  0 |  1 |  F |  2 |  1 |  1 |
    --------------------------------------------------
 8 |  0 |  0 |  1 |  1 |  1 |  1 |  2 |  3 |  F |  1 |
    --------------------------------------------------
 9 |  0 |  0 |  1 |  F |  1 |  0 |  1 |  F |  2 |  1 |
    --------------------------------------------------
10 |  0 |  0 |  1 |  1 |  1 |  0 |  1 |  1 |  1 |  0 |
    --------------------------------------------------
```

```
Generating Performance Report ...
```

Wow! What's that graph? That's something new! That's right! For my advanced feature, I added a performance report to summarize your moves. How cool is that?!

FAQs

1) How did you implement this game?

Well, on the abstract level I created two 2-dimensional arrays to fulfill the purpose of having a grid/board. One of them is kept hidden from the user as it contains the solution to a given gameplay and the another one is used for display purposes, the one user sees.

2) Which version of Python you wrote this for?

Good question! Python 2.7.13 can support this game.

3) Any Python packages you have used in your project.

Yes, 3 of them. All 3 packages are very common and can be installed by simply doing pip install.

- matplotlib
- random
- time

4) Have you tested this game?

Of course! I made a manual test plan to extensively test this game.

5) What did you use for code versioning?

GitHub

Github is my best friend! My repo is on — https://github.com/divyamguptaedu