CIT 590 Assignment 3: Three Musketeers

Fall 2016, David Matuszek

Purposes of this assignment

- To give you experience with using lists and tuples
- To get you started writing unit tests

General idea of the assignment

Write a program to play the game of Three Musketeers--human against computer.

You have probably never heard of this game. That's okay, the rules are simple--that's why I chose it. The link above is to a Wikipedia article that explains the game, and I can answer questions if necessary. It also happens to be a pretty good game.

Details

Get starter files

Copy the files three_musketeers.py and three_musketeers_test.py to your computer.

Writing tests first

It is better to write the tests before writing the code to be tested. Since in this assignment I'm providing stubs for both methods and their tests, this is an easy time to try doing things the recommended way.

I have an entire writeup on Test-Driven Design (TDD), but here's the capsule version:

```
pick a method that doesn't depend on other, untested methods
while the method isn't complete:
    write a test for the desired feature
    run all tests and make sure the new one fails
    while any test fails:
        add/fix just enough code to try to pass the tests
    refactor the code to make it cleaner
```

About the provided code

See the Wikipedia article for the rules of the game.

The "board" is represented as a list of five lists; each of these lists represents a row, and each list contains five elements. The first list represents the first row; the first element in each list represents the first column. The values in the list must each be one of three things: an `'M'`, representing a Musketeer, an `'R'`, representing one of Cardinal Richelieu's men; and a `'-'`, representing an empty space. `board` is the one and only global variable in this program.

Directions are given as one of the four strings `'left'`, `'right'`, `'up'`, and `'down'`.

Your job is to complete the program, both `three_musketeers.py` and `three_musketeers_test.py`.

- In `three_musketeers_test.py`,
  - The `setUp(self)` function is called before each and every test, and gives you a fresh new `board` (in `three_musketeers.py`, not here in the test file). This is done so that the result of one test does not depend on what other tests may have done. (You can use a different initial test board if you like.)
  - The `set_board` method (in `three_musketeers.py`) can be called when you want to perform tests on some other board that the one in `setUp`.
  - The `self.fail()` command causes the test to fail. You should replace each such command with real tests.
- In `three_musketeers.py`,
  - Where I have written `assert condition` at the beginning of a function, that means you can assume the condition holds as you write the function. It also means you should only call the function with parameters that satisfy the condition.
  - The `pass` statement does nothing. You should replace every `pass` statement with actual working code.

Some brief Python reminders

Here are some Python things to remember:

- Methods can call other methods. Sometimes this means that a fairly complex job can be accomplished in just a line or two.
- It's easy to get things in and out of tuples. If `location` is a two-tuple, all of the following work:
  - `(row, column) = location`
  - `location = (row, column)`
  - `column = location[1]`
- Although you can sometimes leave the parentheses off a tuple, you must have them when calling a function: do_something((3, 4))

- Sequences of the same type can be "added." For example, `[1, 2] + [3, 4]` gives `[1, 2, 3, 4]`.
- `append` adds an element to a sequence. For example, if `a = [1, 2]`, `a.append(3)` changes `a` to `[1, 2, 3]`.
- If `b` is a list of lists, `b[0]` is the first list, and `b[0][0]` is the first element in the first list. For example, if `c` is `[[1, 2], [3, 4, 5]]`, then `c[1][2]` is `5`.

The computer does **not** have to play a great game--it does have to play legally. However, if the computer plays stupidly, it won't be much fun. If you would like to implement a simple strategy, here's a start:

- Cardinal Richelieu's men should try to all move in the same direction.
- The three Musketeers should try to stay as far away from one another as possible.

Don't spend any more time than necessary on strategy until **after** the program is working!

Due date

Zip your two files and turn them in to Canvas before **11:59pm Tuesday, September 20**.