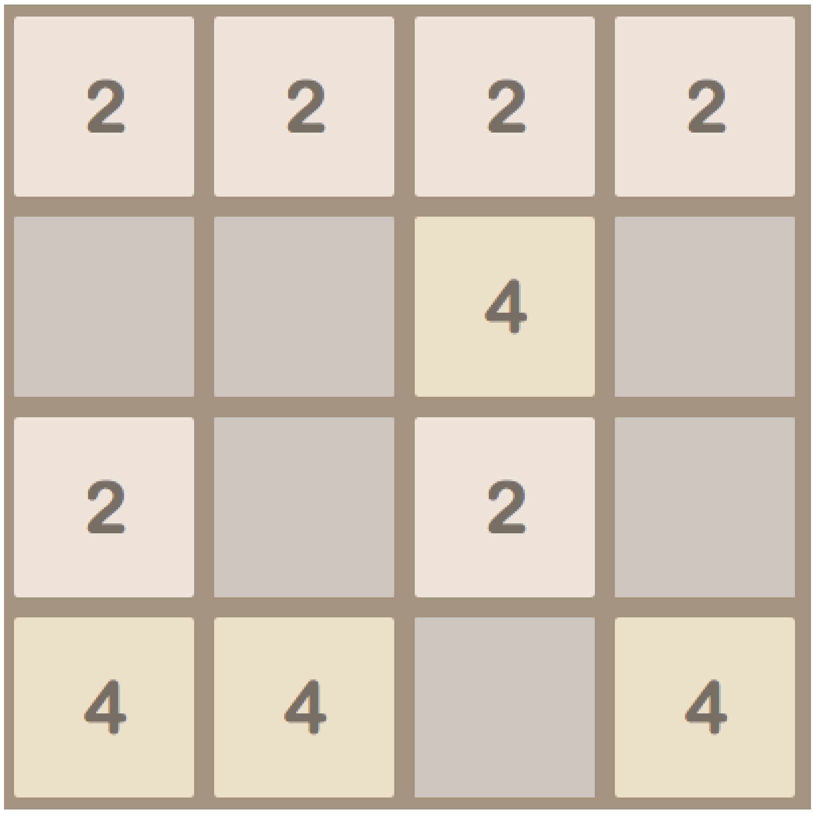
Assignment 02: 2048

Overview

2048 is a simple grid-based numbers game. The object of the game is to combine tiles with the same number to make larger numbered tiles. You "win" when you create a 2048 tile. If you haven't played it already, go on and play it on your pc/phone.

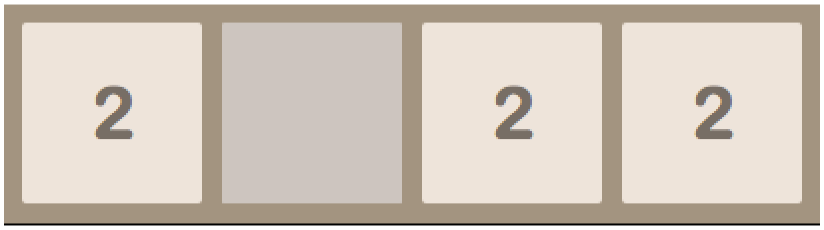


Merge

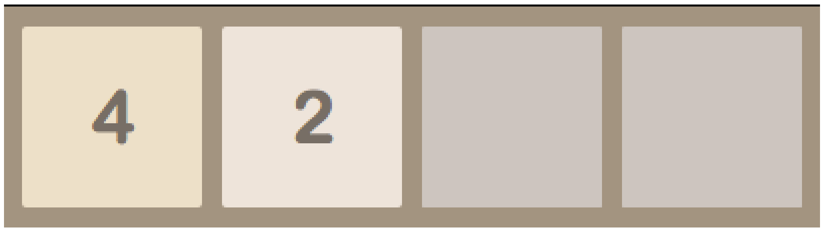
For this assignment, you will implement a function merge(line) that models the process of merging all of the tile values in a single row or column. This function takes the list line as a parameter and returns a new list with the tile values from line slid towards the front of the list and merged. Note that you should return a new list and you should not modify the input list. This is one of the more challenging parts of implementing the game.

In this function, you are always sliding the values in the list towards the front of the list (the list position with index 0). You will make sure you order the entries in line appropriately when you call this function later in the next assignment. Empty grid squares with no displayed value will be assigned a value of zero in our representation.

For example, if a row of the board started as follows:



And you slide the tiles left, the row would become:



Note that the two leftmost tiles merged to become a 4 and the third 2 just slides over next to the 4. A given tile can only merge once on any given turn, although many pairs of tiles could merge on a single turn.

Keep in mind, however, that any tile should only be merged once and that these merges should happen in order from lowest index to highest index. For instance, on the input [2, 0, 2, 4], the result should be [4, 4, 0, 0] and **not** [8, 0, 0, 0]

As you work on your merge function, here are some simple tests you should try:

• [2, 0, 2, 4]	should return [4, 4, 0, 0]
• [0, 0, 2, 2]	should return [4, 0, 0, 0]

- [2, 2, 0, 0]   should return [4, 0, 0, 0]
- [2, 2, 2, 2]   should return |[4, 4, 0, 0]
- [8, 16, 16, 8]   should return [8, 32, 8, 0]

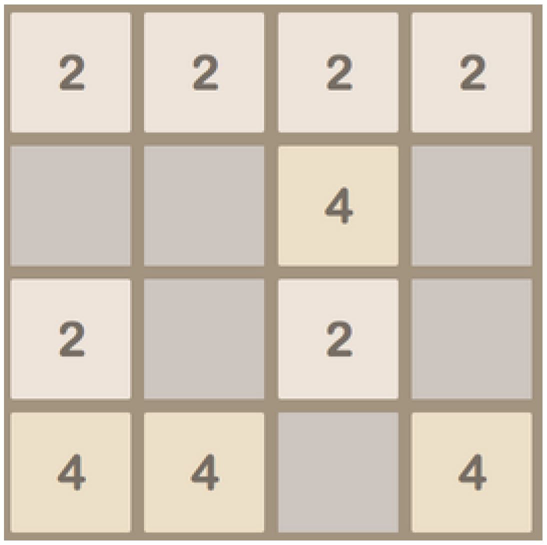
These tests are by no means exhaustive and are just meant to get you started.

After building the single merge design the game with 4x4 grid which can be abstracted to list of four lists. If you can think of a better (more efficient) approach you are welcome to use your own structure.

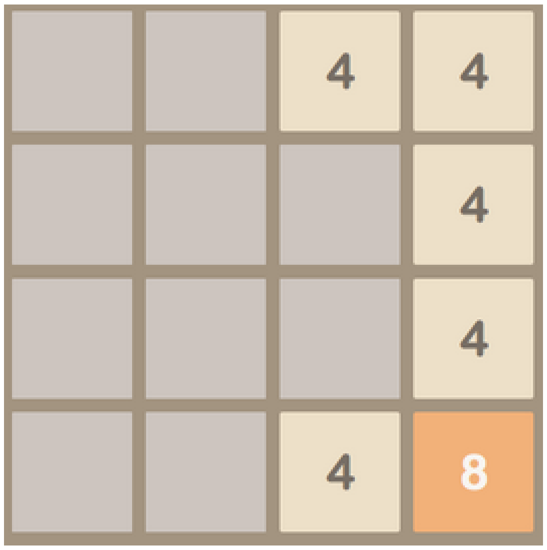
Moving: Sliding and Merging

On each turn, you may slide all of the tiles on the board in one direction (left, right, up, or down). When you do so, all of the tiles on the board slide as far as they can go in the given direction leaving no empty squares between the tiles. Further, if two tiles of the same number end up next to each other, they merge to form a new tile with twice the value. If no tiles would slide or combine in a given direction, then that is not a legal move, and you cannot make that move on the current turn.

To understand how the tiles slide and merge, consider the following game board:



If you move right, then the tiles would slide and merge to yield the following final position:



Note the following about how the tiles slide and merge:

- In the top row, the first two "2" tiles merged to become a "4" tile. The second two "2" files also merged to become another "4" tile. However, the two resulting "4" tiles did not merge. A tile can only merge *once* on any given move.
- In the second row, the single "4" tile just slides all the way to the right.
- In the bottom two rows, a pair of matching tiles merge. Note that it does not matter if there were empty squares between the tiles before they slid. The merge happens to adjacent tiles after they have slid over as far as possible.
- In the bottom row, only one pair of "4" tiles merge. In particular, the two tiles furthest to the *right* are the two that merged. This is because you moved right. Had you moved left, for instance, the two tiles furthest to the *left* would have merged instead.

Coding Guidelines

In this class, you will be asked to strictly follow a set of coding style guidelines. Good programmers not only get their code to work, but they also write it in a way that enables others to easily read and understand their code. Please read the style guidelines carefully and get into the habit of following them right from the start.

Documentation strings ("docstrings") are an integral part of the Python language. They need to be in the following places:

- At the top of each file describing the purpose of the module.
- Below each class definition describing the purpose of the class.
- Below each function definition describing the purpose of the function.

Docstrings describe what is being done in a module, class, method, or function, not how it is being done. Except in rare cases where the how is part of the contract

Comments

Comments should describe how a section of code is accomplishing something. You should not comment obvious code. Instead, you should document complex code and/or design decisions. Comments and docstrings are not interchangeable. Comments start with the "#" character. While you will see some Python programmers do this, you should not comment your code by putting a multi-line string in the middle of your program. That is not actually a comment, rather it is just a string in the middle of your program!

Global variables

Global variables should be avoided in this class as much as possible. Avoiding their use is good programming practice in any language. There is one exception to this rule: you may have global constants. Because the Python language does not actually support constants, by convention, Python programmers use global variables with names that are in all capital letters for constants.

Names

All variable, function, class, and method names **must** be at least 3 characters. The first character of a name should follow these conventions:

- Variable names should be meaningful avoid variable names like a, i, ab … and use descriptive variable names
- Variable names should always start with a lower-case letter. (Except for variables that represent constants, which should use all upper-case letters.)
- Function names should always start with a lower-case letter.