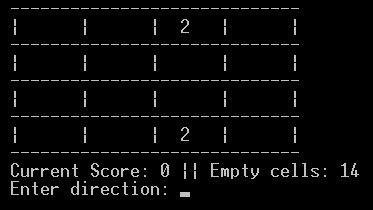
Problem : The 2048 Game

2048 is a single-player sliding block puzzle game. The game's objective is to slide numbered tiles typically on a 4x4 grid to combine them to create a tile with the number 2048. The tiles are slid all together either upwards, downwards, left or right and go all together to one side of the grid, stopped only by either the border of the grid or a neighbouring filled tile. Two adjacent tiles in the same direction of the slide with the same values are summed in one tile. A tile participates in a merge only once in each round. At each round a new empty tile is filled with a 2 or 4 and the player slides the tiles of the board in one direction (up, down, left or right) again. The player wins if the value 2048 is reached, and loses if all the tiles are filled and no sliding is possible.

Here is a typical snapshot of a game of 2048 in progress played on a smartphone, tablet or browser.  


You are asked to write a python program to allow a user to play the game on the computer. The game consists of displaying the grid with its tiles, let the user interact by choosing a direction to slide, and consequently updating the grid and maintaining the score.

The output should not be graphic with colours as above but using simple ASCII characters as below:



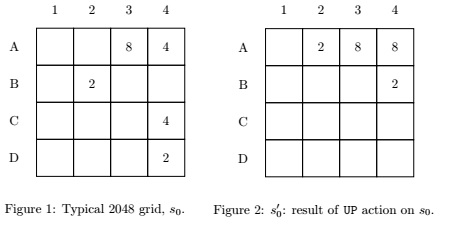
Notice that in addition to the score, we would like to display the number of empty tiles.

The game is an iteration that continues until the user wins by forming a tile with the value 2048 or the grid is full and the tiles cannot be collapsed.

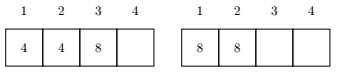
**How are tiles created and how do we collapse the tiles?**

At each round, a randomly selected empty tile is filled with the value 2 or value 4, with 2 having a 90% chance to be picked over the 4. When the game starts however, two tiles randomly selected and filled with the value 2.

During each move, the player can take one of four possible actions: {LEFT, RIGHT, UP, DOWN} which collapses the grid left, right, up or down respectively. When the grid is collapsed in any direction, all the numbers shift to blank tiles in that direction till they reach the end of the grid or reach a tile with a number in it already. During the shift, identical numbers are added together and merged in the first tile with that number in the direction of the slide, while different numbers stack next to each other. For example, for state s0, if the player chooses the UP action, the table will be collapsed upward resulting in the new state, s'0 (see Figure 1 and 2).



In this state s'0 ( Figure 2), we can see that the **2** in B2 moved up to A2 and the **8** in A3 remained where it was as it was at the topmost end of the column initially. The **4** in A4 also did not move but when the **4** in C4 moved up, it was added to the first one since identical numbers are merged (summed up). The **2** in D4 moved up to B4. However, if a tile with a number was already merged with an identical number during one round, it will not be merged again with another identical tile during the same move.



In the example above, when sliding to the left the row on the left, it results in the row on the right. The two **4**s are merged to give an 8, but now the two resulting **8**s are not merged in this current move.

NB: When the user attempts a direction in which the grid is NOT collapsible, the round is cancelled, i.e. no random **2** or **4** is added in an empty tile.

**How do we score?**

Points are scored for any tile merge that occurs during a move with the value being equal to the sum of the value of the two merged tiles- i.e., the action to go from s0 to s'0 in Figures 1 and 2 results in 8 points.

Note that if tiles continuously merge, their value would increase by power of 2: 2-4-8-16-32-64-128-256-512-1024-2048. The goal is to reach the value **2048**.

Since after each move, a **2** or a **4** is placed on a random empty tile, over time the number of blank tiles decreases. The game stops when the grid is no longer collapsible in any of the four directions (when there are no legal actions available.) To get used to the rules of the game you can play the game at <https://gabrielecirulli.github.io/2048/>. Be careful, it is very addictive.

**Tasks to do**

Your task is to implement the game in python. To do so, you need to design a data structure to hold the necessary information and implement an interface to access and manipulate this data structure with the game program. The following is the algorithm that the game program should follow:

Initialize Game

end\_of\_Game <- false

while (not end\_of\_Game) repeat

display grid

if game\_is\_won then

display congratulatory message

else if no free tiles left and not collapsable then

display Sorry you lose

end\_of\_Game <- true

else

get selected direction among up, down, left, right or exit

if exit then

end\_of\_Game <- true

else case direction is

up then slide\_UP

down then slideDOWN

left then slideLEFT

right then slideRIGHT

end\_case

Add 2 or 4 in random empty tile

end\_if

end\_if

end\_while

The direction  up, down, left, and right, as well as the exit, can be represented by character input like "P" for exit, "D" for right, "A" for left, "S" for down and "W" for up.

This algorithm considers the existence of a data structure that knows how to display itself, stores the state of the game grid and the score and has a protocol with a set of getters and setters.

Build a class to represent a 2048 game instance which stores the n x n grid as a matrix using a list of lists. The grid should be initialized with zeros to represent empty tiles. In addition to the grid property, the instance should also store the score as well as the number of rows and columns if these are not fixed to 4 x 4. The constructor of the instance should also initialize two random tiles with the value **2**.

You are provided with a skeleton code in the file game.py (see below). This file contains a class: Game2048. You should work on completing this class and the main program. The class contains properties to store the state of the grid and a set of methods to perform the needed actions involving the grid of the game. Some of these methods are already completed, others are enumerated but not completed. They have the instruction "**pass**" or comments that start with "# #####". Complete these methods. When doing so, you may need other methods that are not enumerated in this skeleton program. Write those methods too.

**How to proceed?**

Proceed step by step and test your program as you go along. For instance, first complete the constructor \_\_init\_\_() then finish the print() method and test whether your grid is displayed as required. Yous should get an output as follow:

-----------------------------

| | | 2| |

-----------------------------

| | | | 2|

-----------------------------

| | | | |

-----------------------------

| | | | |

-----------------------------

Current Score: 0 || Empty cells: 14

---------------------------------------

Then write the methods getNbEmptyTiles() and getListEmptyTiles() and test them in isolation. Then continue with the methods win()and collapsible() and do a unit test. Afterwards write the main program even before completing the methods for sliding the tiles and test the input validation for the directions. The main program should take care of the addition of the random tiles which will let you test the endgame when the grid is full. Finally focus on the 4 methods for sliding the tiles. Note what is consistently similar and avoid repeating it by creating helper functions. Enjoy!

Here is an example of a situation where the user loses. The grid is full and there is no possible collapse up, down, left or right.

-----------------------------

| 2| 16| 2| 4|

-----------------------------

| 4| 32| 64| 8|

-----------------------------

| 64| 256| 1024| 16|

-----------------------------

| 2| 4| 2| 4|

-----------------------------

Current Score: 11620 || Empty cells: 0

---------------------------------------

Sorry! Game Over. You lose.

Here is an example of a winning case. The value 2048 was reached.

-----------------------------

| | | | |

-----------------------------

| | | 4| 64|

-----------------------------

| | 64| 128| 16|

-----------------------------

| | 256| 512| 2048|

-----------------------------

Current Score: 27828 || Empty cells: 8

---------------------------------------

Congratulations! You win.

Assignment Deliverables

* You are going to submit one Python program that conforms to the specifications above and allows a user to play the game of 2048.