SUPINFO Academic Dept.

SUPINFO International University

INSTITUTE OF INFORMATION TECHNOLOGY

Algorithms and programming with Python

Second Mini - Project

Just Get 10

Version 1.0
Last update: 14/12/2016
Use: Students/Staff
Author: Laurent GODEFROY

SUMMARY

1 PREAMBLE 3

2 GENERAL INFORMATION ON THIS PROJECT 3

3 ALGORITHMS TO PLAY THE GAME 5

3.1 INITIALIZATION 5

3.2 POSSIBLE MOVE 6

3.3 FUSION OF CELLS 6

4 A GRAPHIC INTERFACE 8

4.1 DISPLAY OF THE BOARD 9

4.2 GAME PROCEDURE 10

4.3 MAIN PROCEDURE 11

5 BONUSES 11

5.1 SAVE A GAME 12

5.2 ANIMATION FOR A 10 12

5.3 DIFFERENT TYPES OF GAMING 12

6 INDICATIVE SCALE 12

© SUPINFO International University – http://www.supinfo.com

1 PREAMBLE

This exam must be made by a group of two students. In the single case where the number of students in the promotion is an odd number, one and only one group of three students is allowed.

Any form of plagiarism or using codes available online or any other type of support, even partial, is prohibited and will cause a 0, a cheater mention, and even a disciplinary board.

This mini-project will be presented by a viva. Your passing times will be announced by your campus.

The viva is also made up of groups of two. This will take **20 minutes** in which you will show to your examiner the proper functioning of your program with a demo. If you have not implemented the entire project, you will expose the functioning parts.

To support your presentation, you must prepare a PowerPoint file type, in which you will explain parts of the code that you judge important and significant. It is not necessary to send your file to your examiner. He will discover it day of your viva. A communication precising all these informations will be sent early January.

An **indicative** scale is given to you in the last part of the topic.

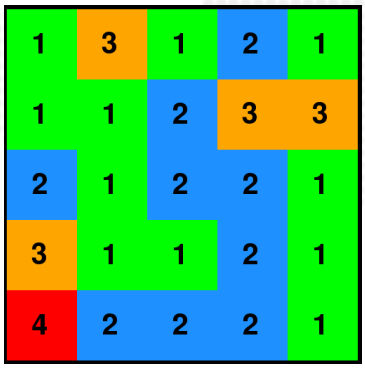
2 GENERAL INFORMATION ON THIS PROJECT

Important note: no code is requested in this sub part, where we will explain the rules.

The goal of this mini-project is to write in Python language a program to play the famous game of reflexion and logic « Just Get 10 ». This game has been created by the Veewo company and you will find the original version here.

Initially, numbers from 1 to 4 are randomly posed on a board of 5 rows and 5 columns. The number 1 has a bigger probability of appearing than the number 2. And the 2 has a bigger probability of appearing than the number 3, etc. Here is an example of an initial configuration:

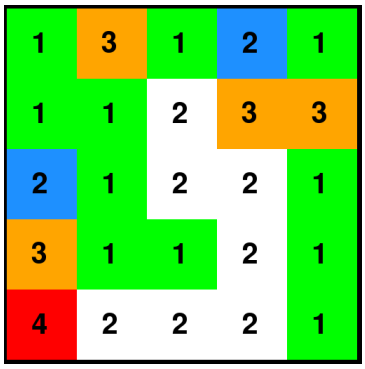
© SUPINFO International University – http://www.supinfo.com



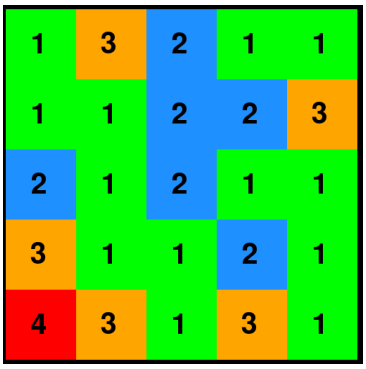
The goal is to get the value 10 by merging successively adjacent cells of same value. When a such set of cells merges, we get a cell whose value is incremented by 1. The others cells of the set disappear, cells which has not been impacted “fall down “by gravity, and columns which are not full, are filled randomly with the same probability when the board has been created.

Note that the term “adjacent “does not include cells touching by the apex in diagonal.

We will explain it with an example. We can select (they will appear in white) a set of adjacent cells of same value by clicking on any of these cells:



Then, we choose the cell to which they will merge by clicking on it, for example here, the cell on the fifth row and the second column:



The selected cells had a value of 2, therefore a 3 appeared on the fifth row and the second

© SUPINFO International University – http://www.supinfo.com

column. The others twos of the selected set therefore disappeared. The remaining cells of columns 3 and 4 then fell because of gravity, and these uncompleted columns were randomly filled.

The goal is to reach the number 10. If one gets stuck before reaching this number, the game is over.

You might need to read several times the instructions before getting a good overall view of the subject. Take the time you need in order to fully understand the instructions before starting to code in the following parts.

In part 3, we will implement the algorithms needed for the progression of the game.

In part 4, we will add a graphic interface.

Finally, in part 5, we will offer some bonuses and possible extensions.

3 ALGORITHMS TO PLAY THE GAME

It is strongly recommended to read all the content of this part before coding.
The requested work is noticeable with its blue colour.

Important note: In this part we will only work in console mode. The board will obviously be a two-dimensional list of integers.

3.1 INITIALIZATION

In a file that we will name « bases.py », implement the following subroutines:

- A function with a t-uple of three real numbers (x_1,x_2,x_3) in parameters checking $0<x_1<x_2<x_3<1$. A number between 0 and 1 will at first be taken randomly thanks to the random function. If this number is smaller than x_1 the function returns 4, if it is between x_1 and x_2, the function will return 3, if this number is between x_2 and x_3, the function will return 2. Else it will return 1.
- A function with an integers **n** and a t-uple of three real numbers (x_1,x_2,x_3) in parameters where $0<x_1<x_2<x_3<1$. It will return a two-dimensional list of **n** rows and **n** columns where every value is an integer between 1 and 4 which has been obtained thanks to the previous function.

© SUPINFO International University – http://www.supinfo.com

- A procedure enabling to display of the values of a two-dimensional list put in parameters. This procedure will only be use to check the good functioning of the subroutines of this part.

Example of functioning : the following code

```
n = 5
proba=(0.05,0.30,0.6)
gameBoard = newBoard(n,proba)
display(gameBoard,n)
```

must achieve to a similar result of this:

```
1 1 1 3 1
2 2 4 1 2
2 2 1 1 1
1 2 3 2 1
1 1 3 2 1
```

3.2 POSSIBLE MOVE

In a file that we will name « possibles.py », implements the following subroutines:

- A function taking in parameters an integer **n**, a two-dimensional list of **n** rows and **n** columns, and two integers **i** and **j** where **0≤i<n** and **0≤j<n**. It will return **True** if the cell with the coordinate **i**, **j** owns an adjacent case with the same value. Else, it will return **False**.
- A function taking in parameters an integer **n** and a two-dimensional list of **n** rows and **n** columns. It will return **True** if a move is still possible on the board. In others words, if at least one cell has an adjacent case with the same value. Otherwise, its will return **False**.
- A function taking in parameters an integer **n** and a two-dimensional list of **n** rows and **n** columns. It will return the maximum value of the list.

3.3 FUSION OF CELLS

In a file that we will name « merge.py », implementer the following subroutines:

- A procedure taking in parameters an integer n, a two-dimensional list of n rows and n

© SUPINFO International University – http://www.supinfo.com

columns, a t-uple of two integers **(i,j)** where **0≤i<n** and **0≤j<n**, and a list of t-uple of two integers which verify the previous inequality. We will suppose that when a call of this procedure is done by another subroutine, the list put in parameter initially contain the t-uple which has been put in parameter too. At the end of the execution of the procedure, this list will contain the coordinates (as t-uples) of the set of adjacent cells with the same value of the cell which coordinates are those of the t-uple.

- A procedure taking in parameters an integer **n**, a two-dimensional list of **n** rows and **n** columns, and a list of t-uple of two integers **(i,j)** where **0≤i<n** and **0≤j<n**. It will increment by 1 the value of the cell which coordinates are the first element of the list, and it will set to 0 the values of the cells which the coordinates are the others elements of the list.
- A procedure taking in parameters an integer **n**, a two-dimensional list of **n** rows and **n** columns, and a t-uple of three real numbers **(x_1,x_2,x_3)** where **$0<x_1<x_2<x_3<1$**. By gravity i twill make it fall cells located on the top of cells which the value is equal to 0. Finally, it will fill the uncompleted columns thanks to the first function of the sub part 3.1.

Example of functioning: the following code (by calling respectively the three previous procedures “propagation“, “modification“ and “gravity“)

© SUPINFO International University – http://www.supinfo.com

```
n = 5
proba=(0.05,0.30,0.6)
gameBoard = [[1,1,1,1,3],[1,2,4,1,1],[2,2,4,1,1],[4,2,2,3,3],[3,3,2,3,3]]
display(gameBoard,n)
current = (3,2)
L = [current]
propagation(gameBoard,L,current,n)
print(L)
print()
modification(gameBoard,L,n)
display(gameBoard,n)
gravity(gameBoard,n,proba)
display(gameBoard,n)
```

Must achieve to this result:

```
1 1 1 1 3
1 2 4 1 1
2 2 4 1 1
4 2 2 3 3
3 3 2 3 3

[(3, 2), (4, 2), (3, 1), (2, 1), (1, 1), (2, 0)]

1 1 1 1 3
1 0 4 1 1
0 0 4 1 1
4 0 3 3 3
3 3 0 3 3

3 3 2 1 3
1 2 1 1 1
1 3 4 1 1
4 1 4 3 3
3 3 3 3 3
```

4 A GRAPHIC INTERFACE

I It is strongly recommended to read all the content of this part before coding. The requested work is noticeable with its blue colour.
We must use the graphic library Pygame. Using another library will not be taken into account.

© SUPINFO International University – http://www.supinfo.com

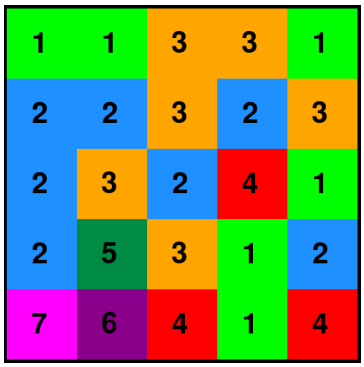
Note 1: in this part we will continue to model the gaming board by a two-dimensional list of integers.

Note 2: in this part we will restrict ourselves with boards of 5 rows and 5 columns. In other words, we will respect the original rule of the game.

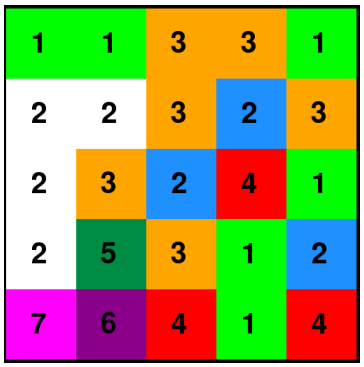
Note 3: you are free to design your game as you want as long as the features are available. The screenshots of this part are only here to give you an idea.

4.1 DISPLAY OF THE BOARD

The goal of this part is to implement procedure able to display the board. Each values will have his own colour, so after many turns, you should see a display looking like this:



Once selected, a set of adjacent cells with the same value will have to be highlight, with white for example:



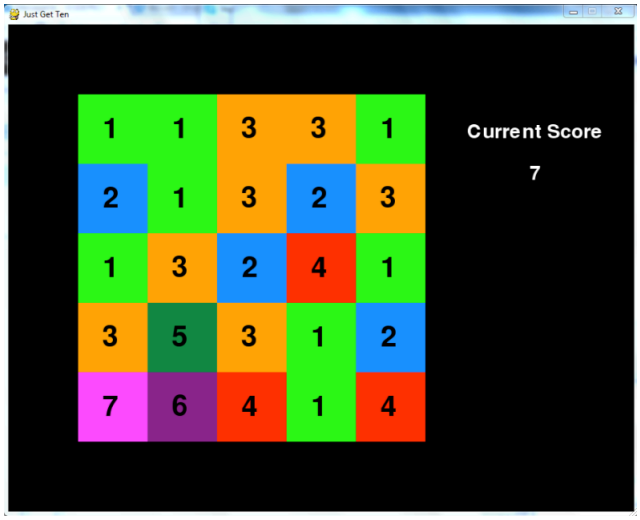
Notice that the selection of a set of cells will not be achieved by the procedures of this sub-part but by those of the following part.

In a file that we will call « justGetTenGUI.py », implement the following subroutines :

© SUPINFO International University – http://www.supinfo.com

- A procedure taking in parameter a two-dimensional list representing the board, the surface on which it will be drawn, the coordinates of a cell and a Boolean. According to the value of this answer, the procedure will display the cell in question (in the right place of course), either with the associated colour in the background, or with a white background.
- A procedure taking in parameter a two-dimensional list representing the board, the surface on which it will be drawn, a list of coordinates of cells and a Boolean. According to the previous principle, this procedure will display the value of the cells with the coordinates figuring in the list put into parameters.
- A procedure taking into parameter a two-dimensional list representing the board, its number **n** of rows and columns, and the surface on which it will be drawn. This procedure will display the value of each cell of the board with the associated colour in the background.
- A score procedure displaying the maximum value of the board.

- Your game window will look like the following one:



4.2 GAME PROCEDURE

Implement in the previous file a procedure managing the succession of the player's moves:

- This procedure takes into parameter a two-dimensional list representing the board, its number **n** of rows and columns, the surface onto which it will be drawn, and a t-uple of three real numbers (x_1, x_2, x_3), verifying $0<x_1<x_2<x_3<1$.
- As long as a move is still possible, the player is invited to play:
  - One click on an ''isolated'' cell does not have any effect.
  - One click on any random cell from a set of adjacent cells of the same colour colours this set in white.
  - One click on one of the cells of a selected set generates the merging towards this cell. The board is also modified according to the rules of the game. If a set is

© SUPINFO International University – http://www.supinfo.com

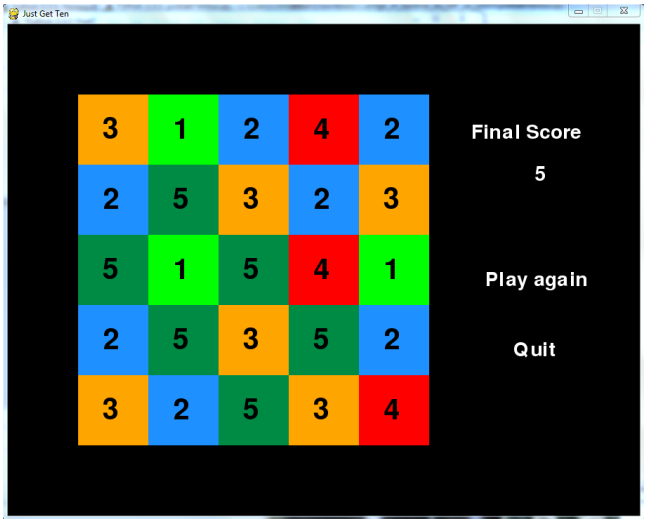
selected, a click on a cell that doesn't belong in the set re-colorates it, and can de-colorate another set.
    - At any moment the player can quit the game by clicking on the cross or by pressing on the escape button.
- After every move the score is put up to date.

4.3 MAIN PROCEDURE

Implement in the previous file a procedure allowing to play Just Get 10:

- This procedure will create a graphic surface, initializing a board according to a t-uple of probability and making the player play.
- At the end of the game we will give the opportunity to the player to either replay or quit the game.

Here is an example of the end of a game:



5 BONUSES

You or free to implement either zero, one or several of the following bonuses.

© SUPINFO International University – http://www.supinfo.com

5.1 SAVE A GAME

Adding an option allows one to save the state of a game in a text file.
Also implement the possibility of taking back a saved game in a text file.

5.2 ANIMATION FOR A 10

Adding a visual and sound animation (document yourself on this point) when the player reaches 10. Make sure that the game can resume.

5.3 DIFFERENT TYPES OF GAMING

Offer a choice of different types of gaming. Here are some examples :

- Smaller board (4 rows and 4 columns)
- Bigger board (6 rows and 6 columns)
- Limited time for every move
- Limited time to reach 10

…

6 INDICATIVE SCALE

This scale can be set to change; it is therefore only indicative.

- Part 3 : 20 points
- Part 4 : 20 points
- Bonus : 10 points

This adds up to a total of 40 points, your viva being evaluated on 20 points. The total grade will then be brought proportionally to a grade based on 20 points.

© SUPINFO International University – http://www.supinfo.com