Python Project
Like a Tetris tune

Instructions & General Information:

⇒ This project is to be coded exclusively in Python language
⇒ Project attachements file :
  - File **diamond.txt**

⇒ Organization of teams:
  - This project is to be carried out in pairs (only one trinomial is allowed if an odd number of students)
  - The list of teams must be given to teachers no later than the end of the first project follow-up session

⇒ Important dates :
  - Publication date: **07/11/2022**
  - Date of submission of the project: Week **07/11/2022**
  - Follow-up date1 : Week **07/11/2022**
  - Follow-up date 2 : Week **05/12/2022**
  - Follow-up date 3 : Week **12/12/2022**
  - Submission date: **19/12/2022** à **23h59**
  - Defence date: Week **03/01/2023**

⇒ Final submission: an archive **.zip** contains
  - The code of the project contains all the files **.py** and **.txt**
  - The report in **.pdf**
  - A file **README.txt** listing the programs and how to use them in practice. This file must contain all the instructions necessary for execution; It is very important to explain to the user how to use the tool.

⇒ Submission of the project :
  - Will be comunicated later …

⇒ Evaluation
  - Indicative scale
  - Detailed scale: to be provided later
  - Final grade of the project = Code note + Report note + Defense note
  - Reminder: project note = 20% of the "Python Programming" course grade
  - Members of the same team may have different grades depending on the efforts made in carrying out this project.

⇒ Plagiarism
  - Any work with plagiarism will be severely sanctioned

Organisation of the code :

⇒ The notation of the code will mainly consider:
  - The implementation of the requested functionalities: progress at best but NOT off-topic
  - The quality of the code provided: organization in functions, comments, names of significant variables, respect for file names
  - Ease of use of the user interface

Pedagogical concepts covered:

⇒ The realization of this project will help you to apply the following pedagogical concepts
  - Basics, 1D lists, 2D lists, functions and files

⇒ To help you carry out this project, you can rely on:
  - Support of the course TI101 (I, B, R)
  - Books, various courses on the web. PLAGIAT ALERT !! It is not about copying entire programs.
  - Efrei teachers during the project follow-up sessions

Description

Tetris is a game that comes in the form of a matrix where blocks of different shapes must be laid so that the board is kept as long as possible not full. The idea is to place each block in the location that eliminates as many rows and/or columns as possible. These are automatically deleted when they are full. In this project, we want to take inspiration from Tetris and create a brand new version for it.

It is a question of starting from a 2-dimensional board of minimum size 21 x 21 boxes - whose lines are designated by capital letters ('A', 'B' ...) and the columns by lowercase letters ('a', 'b', ...) - on which will be delimited a valid playing surface. This valid playing surface can take three different shapes: circle, diamond or triangle (see three figures below).

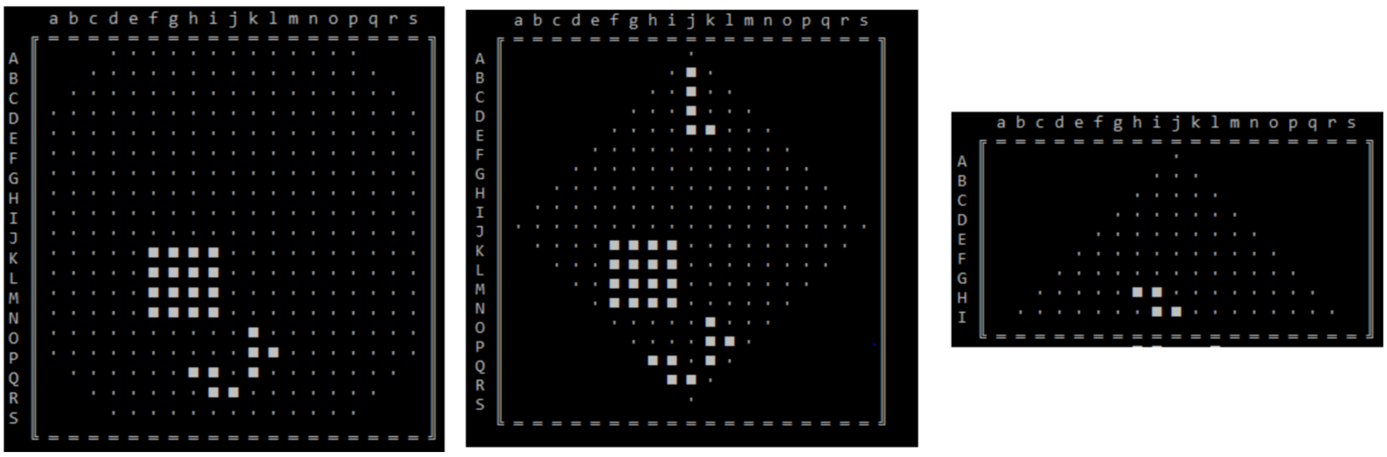

a b c d e f g h i j k l m n o p q r s
A B C D E F G H I J K L M N O P Q R S
a b c d e f g h i j k l m n o p q r s
A B C D E F G H I J K L M N O P Q R S
a b c d e f g h i j k l m n o p q r s
A B C D E F G H I

Figure 1

In this game, the user has a set of blocks that he will have to place in turn on the valid surface of the board by entering the coordinates of the place where he wants to insert them. Some blocks to be laid are common to all three shapes, but to each of them other more suitable forms can be added (see last section).

Application to be carried out

When the game launches, a home screen should appear with two options:

- Start playing
- Display the rules of the game.

The content of this home screen is left open and everyone can propose the interface they want.

When the game starts, the home screen should disappear and the user is prompted to set up their game:

1. He must first choose the size of his game board and its shape from:

    -Circle

    -Dimond

    -Triangle

2. Choose from two suggestion policies:

- Display at each turn of the game all the available blocks and the user selects one.
- Display only 3 randomly selected blocks.

Note: the settings made at this stage remain unchanged throughout the game period.

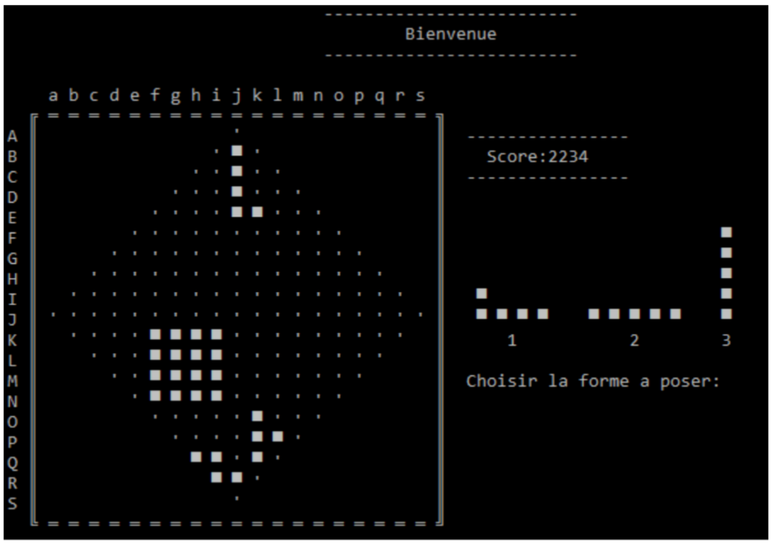
During the game, the application must be able to check if the contact information entered by the user is valid:

-They are valid if the squares of the chosen block can all be placed on empty squares on the valid surface of the game.

- They are not valid if some - or all - squares of the chosen block are outside the valid surface of the game or if the selected location does not have enough squares to accommodate the chosen block.

Each time a block is inserted, a test is performed to check if there are solid rows and/or columns. In this case, these must be canceled (reset).

In the case of cancelling the line, the solid squares above must go down to pile up at the bottom of the playing surface. On the other hand, when the column is cancelled, no offset is to be expected. In addition, canceling rows/columns must result in the calculation of a score that is displayed near the game board. A simple function for this one would be to count the number of canceled boxes. However, the proposal of any other function is accepted.



```
                         -------------------------
                                 Bienvenue
                         -------------------------

  a b c d e f g h i j k l m n o p q r s
A                                          ---------------
B                                            Score:2234
C                                          ---------------
D
E
F
G
H
I
J
K                                            1          2       3
L
M                                          Choisir la forme a poser:
N
O
P
Q
R
S
```

Figure 2

Stop Condition

1. To insert a block on the tray, the user has three attempts to enter valid coordinates. If after 3 successive attempts, the chosen positions are invalid each time, then the game stops. At the end of the game, a message should be displayed on the screen reminding you of the score obtained.
2. The game may also be interrupted at the request of the user ("Stop playing")

Technical aspects

The realization of this project requires careful thought about the storage structure that will represent all your elements, as well as a division of your code into functions.

1. Storage of trays

Each of the game boards is stored in a text file as a matrix where a non-game board square is designated by a 0, a square that is part of the valid board area is designated by a 1, and a box on the valid board surface and has been occupied by a block is designated by a 2.

In this project, the diamond.txt file is provided as an example of an empty tray in the form of a diamond. It is requested to generate the other two files corresponding to the circle and triangle trays

2. Storage of the blocks :

- All blocks to be used in the game are stored in a single 1D list
- Each box in the 1D list contains a block represented as a matrix

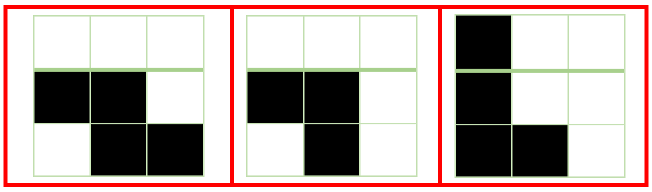


Figure 3

- The matrix size of each block is N x M where N is the greatest height among all the blocks proposed in the series, and M is the largest width among all the blocks proposed in the series.

Example: If we have the blocks:

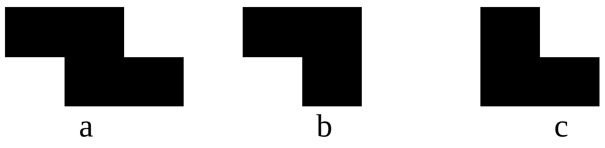
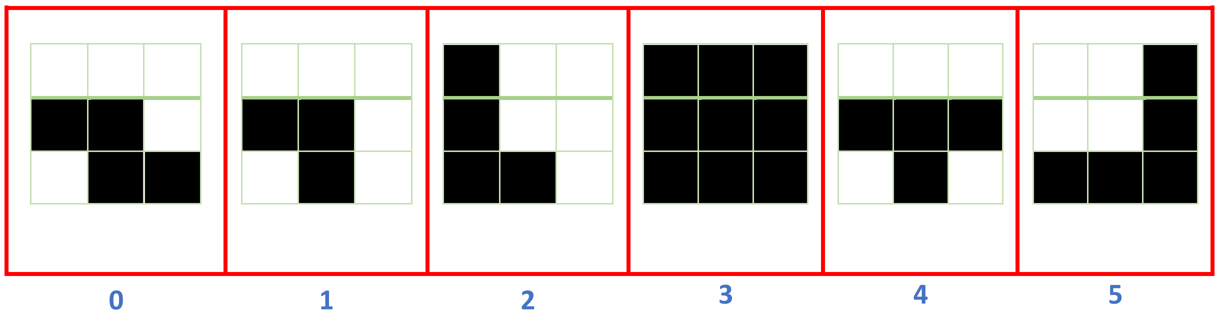


Figure 4

The size of the matrices where they will be stored will be 3x3 because the block in Figure 4.c is the largest in height (=3) and the block in Figure 4.a is the widest (width = 3).

- For each form of board, a 1D list is associated whose contents of the boxes represent the indices where the blocks that are authorized to it are stored (see last section). In this game, there will be three lists of different sizes: diamond_list, circle_list and triangle_list.

Example: Let be the list of the following blocks called blocs_list



The three lists associated with the different game boards will be in this example of the following form :

Diamond_list	**0**	**1**	**2**	**4**
circle_list	**0**	**3**	**2**	**5**
triangle_list	**0**	**1**	**3**	

Lists in different sizes with manual filling.

Thus, access to the block of index 5 for the tray of form Circle for example, will be done by: blocs_list[circle_list[3]] .

Functions to implement

To perform this project, you must write at least the following functions:

Game boards

A grid `grid` is a square matrix of integer such that:

- `grid[i][j] = 0` If the box is not part of the game board (diamond, circle or triangle)
- `grid[i][j] = 1` If the box is an empty box on the game board
- `grid[i][j] = 2` if the square is a square occupied by a piece of one of the Tetris blocks

a. Write a function `read_grid(path)` that returns a valid grid read from the contents of the file specified by `path`.

b. Write a function `save_grid(path, grid)` that save a grid `grid` in a file specified by `path`.

c. Write a function `print_grid(grid)` that displays the status of the grid `grid`. The display will be done using the ASCII codes of the different symbols and characters. The choice of its latest is free, the only condition is that the display is clear on the screen.

2. Blocks

a. Write a function `print_blocs(grid)` which takes as parameters the shape of the chosen tray, and which displays the list of all the blocks associated with it.

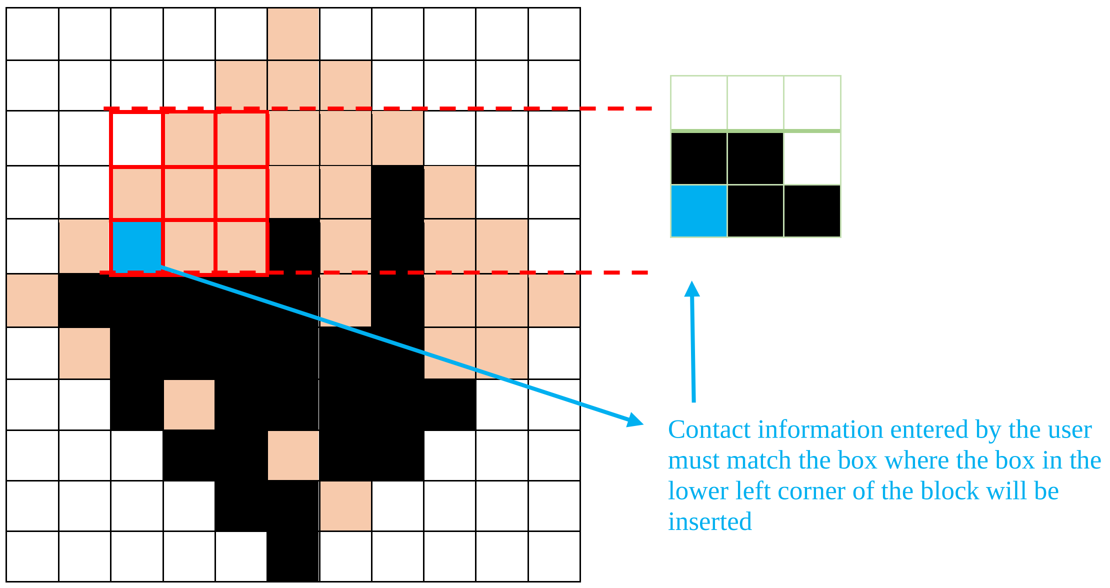
b. Write a function `select_bloc()` which allows you to select the blocks to be proposed to the user according to one of the 2 policies explained above and the type of tray chosen.

3. Positioning

a. Write a function `valid_position(grid, bloc, i, j)` which checks whether the block block can be placed on the slot `grid[i][j]` in such a way that the lower left box of the block `bloc` is positioned on `grid[i][j]`.

   It is assumed that the insertion of a block on the tray will be done by superimposing the matrix that contains it on the place where it is to be inserted. To do this, it will be necessary to match the cell of the lower left corner of the block with the coordinates entered by the user.

Example :


Contact information entered by the user must match the box where the box in the lower left corner of the block will be inserted

b. Write a function `emplace_bloc(grid, bloc, i, j)` that positions the block `bloc` on position `(i, j)` of the grid `grid` if and only this is a valid position, and thus modifies the grid.

4. Cancelling rows/columns and calculating the score

a. Write a function `row_state(grid, i)` that verify if all the lines `i` in a grid `grid` is full.

b. Write a function `col_state(grid, j)` that verify if all the columns `j` in a grid `grid` is full.

c. Write a function `row_clear(grid, i)` that cancels the row `i` in a grid `grid` by shifting all lines from the top of a unit down. Be careful because depending on the shape of the board, some boxes from the previous line may no longer be present in the board.

d. Write a function `col_clear(grid, j)` that cancels the column `j` in a grid `grid`

e. Write a function `update_score()` which updates the score each time a row is cancelled (by as many points as the number of cancelled squares in the row).

5. Go further…

a. Offer the user the option to rotate a block and write the function `rotate_bloc(bloc, dir)` that make the rotation of a block `bloc` a quarter turn clockwise if `dir == True`, and in the opposite direction otherwise.

b. Provide a backup feature that will save the state of the grid.

c. Offer a loading feature that will allow you to resume a saved game.

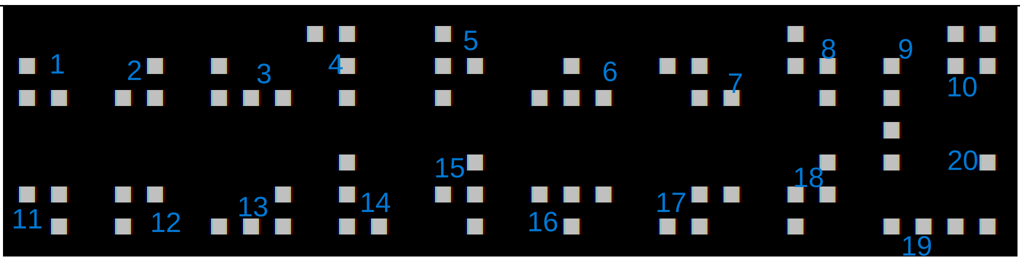
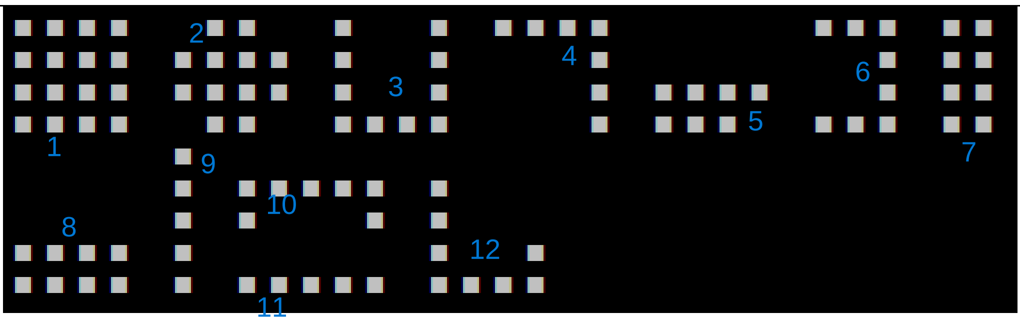
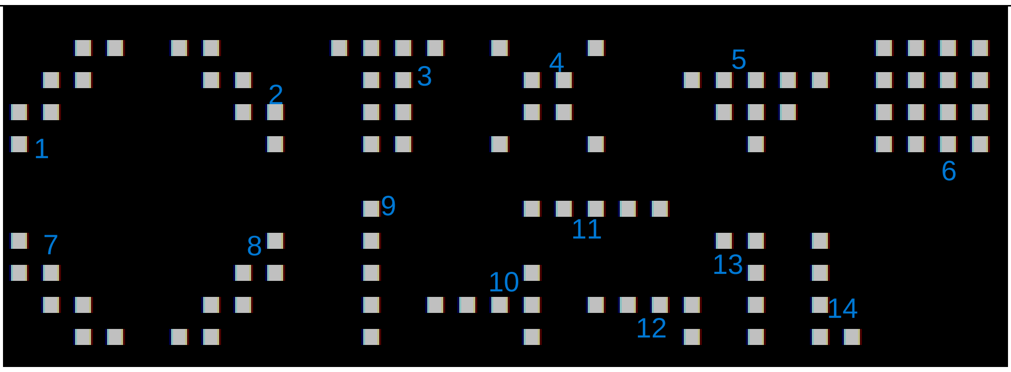
d. Depending on the score value, add difficulty to the game by suggesting blocks that are not always easy to insert.

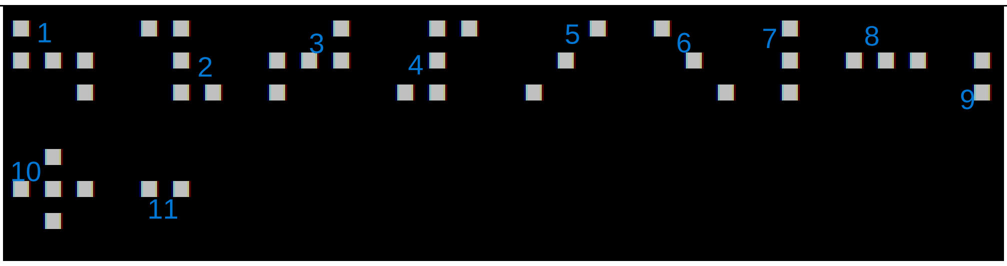
Main Program

Write the main program using the functions developed previously, and others if necessary, that allows the launch of the game.

Code must be organized into modules and functions.

Blocks associated with each game board

Board	List of blocks
All (Blocks common to all trays)	
Circle	
Diamond	

Triangle	

General remarks:

⇒ Secure entries are to be carried out systematically even if they are not always explicitly requested
⇒ Do not hesitate to display messages to users when an action is not possible.
⇒ Display the menu at the end of each action to be able to switch to another action.