

Projet Python Comme un air de Tetris

Consignes & Informations générales :

- ⇒ Ce projet est à réaliser exclusivement en langage Python
- ⇒ Pièces jointes au sujet du projet :
 - Fichier **losange.txt**
- ⇒ Organisation des équipes :
 - Ce projet est à réaliser en binôme (**un seul trinôme est autorisé** si un **nombre impair** d'élèves)
 - La liste des équipes est à remettre aux enseignants **au plus tard** à la fin de la première séance de suivi de projet
- ⇒ Dates clé :
 - Date de publication : **07/11/2022**
 - Date de présentation du projet : Semaine du **07/11/2022**
 - Date de suivi 1 : Semaine du **07/11/2022**
 - Date de suivi 2 : Semaine du **05/12/2022**
 - Date de suivi 3 : Semaine du **12/12/2022**
 - Date de soumission : **19/12/2022 à 23h59**
 - Date de soutenance : Semaine du **03/01/2023**
- ⇒ Rendu final : Une archive **.zip** contenant
 - Le code du projet contenant les fichiers **.py** et **.txt**
 - Le rapport en **.pdf**
 - Un fichier **README.txt** donnant la liste des programmes et comment les utiliser en pratique. Ce fichier doit contenir toutes les instructions nécessaires à l'exécution ; il est très important pour expliquer à l'utilisateur comment se servir de l'outil.
- ⇒ Dépôt du projet :
 - Sera communiqué plus tard ...
- ⇒ Évaluation
 - Barème indicatif
 - Barème détaillé : sera fourni plus tard
 - Note finale du projet = Note code + Note rapport + Note soutenance
 - Rappel : note projet = 20% de la note du cours « Programmation Python »
 - Les membres d'une même équipe peuvent avoir des notes différentes en fonction des efforts fournis dans la réalisation de ce projet.
- ⇒ Plagiat
 - Tout travail présentant du plagiat sera sévèrement sanctionné

Organisation du code :

- ⇒ La notation du code tiendra compte principalement de :
 - L'implémentation des fonctionnalités demandées : avancer au mieux mais PAS en hors sujet
 - La qualité du code fourni : organisation en fonctions, **commentaires**, noms de variables significatifs, respect des noms de fichiers.
 - Facilité d'utilisation de l'interface utilisateur

Notions pédagogiques traitées :

- ⇒ La réalisation de ce projet vous aidera à appliquer les notions pédagogiques suivantes :
 - Notions des bases, listes 1D, listes 2D, les fonctions et les fichiers
- ⇒ Pour vous aider à la réalisation de ce projet, vous pouvez vous appuyer sur :
 - Les supports du cours TI101 (I, B, R)
 - Livres, cours divers sur le web. **ALERTE PLAGIAT !!** il ne s'agit pas de copier des programmes entiers.
 - Enseignants **Efrei** lors des séances de suivi du projet

Description

Le Tetris est un jeu qui se présente sous forme d'une matrice où des blocs de différentes formes doivent être posés de sorte que le plateau soit gardé le plus longtemps possible non plein. L'idée est de placer chaque bloc à l'emplacement qui permet d'éliminer un maximum de lignes et/ou de colonnes. Ces dernières sont supprimées automatiquement lorsqu'elles sont pleines. Dans ce projet, nous souhaitons s'inspirer du Tetris et lui créer une toute nouvelle version.

Il s'agit de partir d'un plateau à 2 dimensions de taille **minimum** 21 x 21 cases - dont les lignes sont désignées par des lettres majuscules ('A', 'B' ...) et les colonnes par des lettres minuscules ('a', 'b', ...) - sur lequel sera délimitée une surface de jeu valide. Cette surface de jeu valide peut prendre trois formes différentes : cercle, losange ou triangle (voir les trois figures ci-dessous).

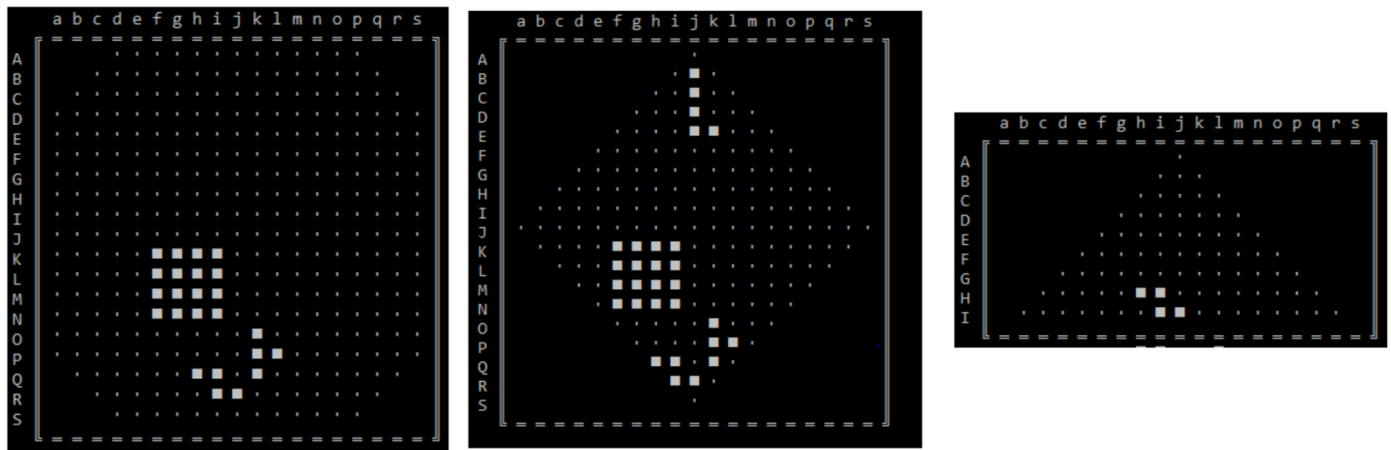


Figure 1

Dans ce jeu, l'utilisateur dispose d'un ensemble de blocs qu'il devra placer tour à tour sur la surface valide du plateau en saisissant les coordonnées de l'endroit où il veut les insérer. Certains blocs à poser sont communs aux trois formes, mais à chacune d'entre elles d'autres formes plus adaptées peuvent s'ajouter (Cf. dernière section).

Application à réaliser

Au lancement du jeu, un écran d'accueil doit s'afficher proposant deux options :

- Commencer à jouer
- Afficher les règles du jeu.

Le contenu de cet écran d'accueil est laissé ouvert et chacun peut proposer l'interface qu'il souhaite.

Lorsque le jeu commence, l'écran d'accueil doit disparaître et l'utilisateur est invité à paramétrer son jeu :

1. Il doit d'abord choisir la dimension de son plateau de jeu et sa forme parmi :

- Cercle

- Losange
- Triangle

2. Choisir parmi deux politiques de suggestion des blocs :

- Afficher à chaque tour de jeu l'ensemble des blocs disponibles et l'utilisateur en sélectionne un.
- Afficher uniquement 3 blocs sélectionnés aléatoirement.

Note : le paramétrage réalisé à cette étape reste inchangé durant toute la période de jeu.

Pendant la partie, l'application doit être capable de vérifier si les coordonnées saisies par l'utilisateur sont valides :

- Elles sont valides si les cases du bloc choisi peuvent être toutes placées sur des cases vides se trouvant sur la surface valide du jeu.
- Elles ne sont pas valides si certaines - ou toutes les - cases du bloc choisi se situent en dehors de la surface valide du jeu ou que l'emplacement sélectionné n'a pas suffisamment de cases pour accueillir le bloc choisi.

À chaque insertion d'un bloc, un test est effectué pour vérifier s'il y a des lignes et/ ou des colonnes pleines.

Dans ce cas, celles-ci doivent être annulées (réinitialisées).

Dans le cas de l'annulation de la ligne, les cases pleines se retrouvant au-dessus doivent descendre pour s'empiler au fond de la surface de jeu. En revanche, à l'annulation de la colonne, aucun décalage n'est à prévoir.

De plus, l'annulation des lignes/colonnes doit entraîner le calcul d'un score qui s'affiche près du plateau de jeu. Une fonction simple pour celui-ci serait de compter le nombre de cases annulées. Cependant, la proposition de toute autre fonction est acceptée.



Figure 2

Condition d'arrêt

1. Pour insérer un bloc sur le plateau, l'utilisateur dispose de trois tentatives pour saisir des coordonnées valides. Si à l'issue de 3 tentatives successives, les positions choisies sont à chaque fois invalides, alors le jeu s'arrête. A la fin de la partie, un message doit s'afficher à l'écran rappelant le score obtenu.
2. Le jeu peut également être interrompu à la demande de l'utilisateur (« Arrêter de jouer »)

Aspects techniques

La réalisation de ce projet nécessite de bien réfléchir à la structure de stockage qui représentera tous vos éléments, ainsi qu'un découpage de votre code en fonctions.

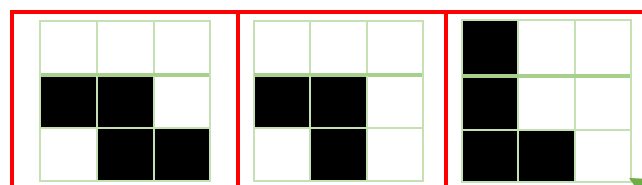
1. Stockage des plateaux

Chacun des plateaux de jeu est stocké dans un fichier texte sous la forme d'une matrice où une case n'appartenant pas au plateau de jeu est désignée par un 0, une case faisant partie de la surface valide du plateau est désignée par un 1, et une case se trouvant sur la surface valide du plateau et ayant été occupée par un bloc est désignée par un 2.

Dans ce projet, le fichier **losange.txt** est fourni comme exemple de plateau vide sous forme d'un losange. Il est demandé de générer les deux autres fichiers correspondants aux plateaux cercle et triangle.

2. Stockage des blocs :

- L'ensemble des blocs à utiliser dans le jeu est stocké dans une seule liste 1D
- Chaque case de la liste 1D contient un bloc représenté comme étant une matrice



Blocs_list : Liste qui permet de stocker tous les blocs du jeu, tous plateaux confondus. Ainsi chaque case ici contient une liste 2D → L'ensemble forme une liste 3D.

Bloc : Liste 2D de taille NxM.
Astuce : utiliser 0 pour représenter les cases vides, 1 pour les autres cases.

Figure 3

- La taille de la matrice de chaque bloc est de N x M où N est la plus grande hauteur parmi tous les blocs proposés dans la série, et M est la plus grande largeur parmi tous les blocs proposés dans la série.

Exemple : Si l'on dispose des blocs :

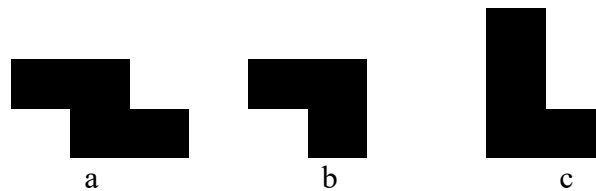
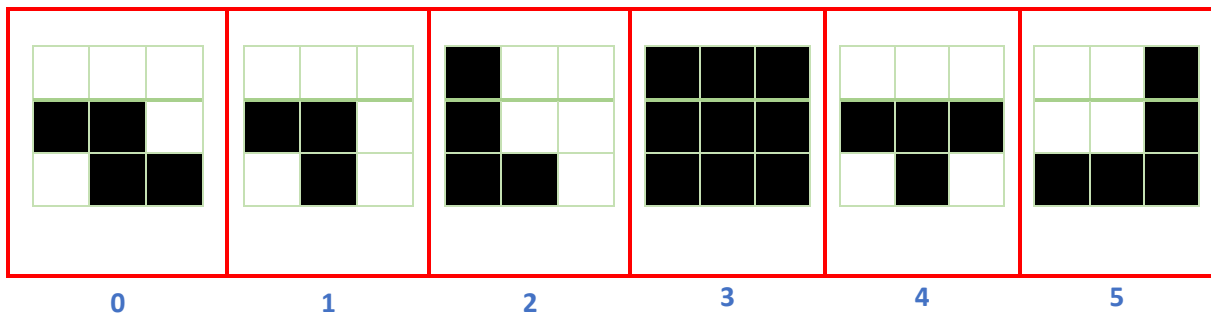


Figure 4

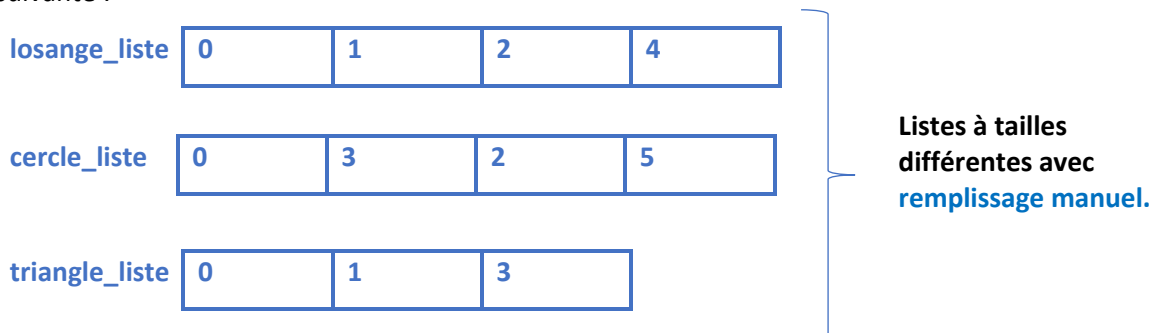
La taille des matrices où ils seront stockés sera de 3x3 car le bloc de la Figure 4.c est le plus grand en hauteur (=3) et le bloc de la Figure 4.a est le plus large (largeur = 3).

- Pour chaque forme de plateau, une liste 1D est associée dont le contenu des cases représente les indices où sont stockés les blocs qui lui sont autorisés (Cf. dernière section). Dans ce jeu, il y aura donc trois listes de tailles différentes : **losange_liste**, **cercle_liste** et **triangle_liste**.

Exemple : Soit la liste des blocs suivantes appelée **blocs_liste**



Les trois listes associées aux différents plateaux de jeu seront dans cet exemple de la forme suivante :



Ainsi, l'accès au bloc d'indice 5 pour le plateau de forme Cercle par exemple, se fera par : **blocs_liste[cercle_liste[3]]** .

Fonctions à implémenter

Pour réaliser ce projet, vous devez écrire au minimum les fonctions suivantes :

1. Plateaux de jeu

Une grille `grid` est une matrice carrée d'entier telle que :

- `grid[i][j] = 0` si la case ne fait pas partie du plateau de jeu (losange, cercle ou triangle)
 - `grid[i][j] = 1` si la case est une case vide du plateau de jeu
 - `grid[i][j] = 2` si la case est une case occupé par une pièce d'un des blocs de Tetris
- a. Écrire une fonction `read_grid(path)` qui retourne une grille valide lue à partir du contenu du fichier spécifié par `path`.
 - b. Écrire une fonction `save_grid(path, grid)` qui sauvegarde la grille `grid` dans un fichier spécifié par `path`.
 - c. Écrire une fonction `print_grid(grid)` qui affiche l'état de la grille `grid`. L'affichage se fera en utilisant les codes ASCII des différents symboles et caractères. Le choix de ses derniers est libre, la seule condition est que l'affichage soit clair à l'écran.

2. Blocs

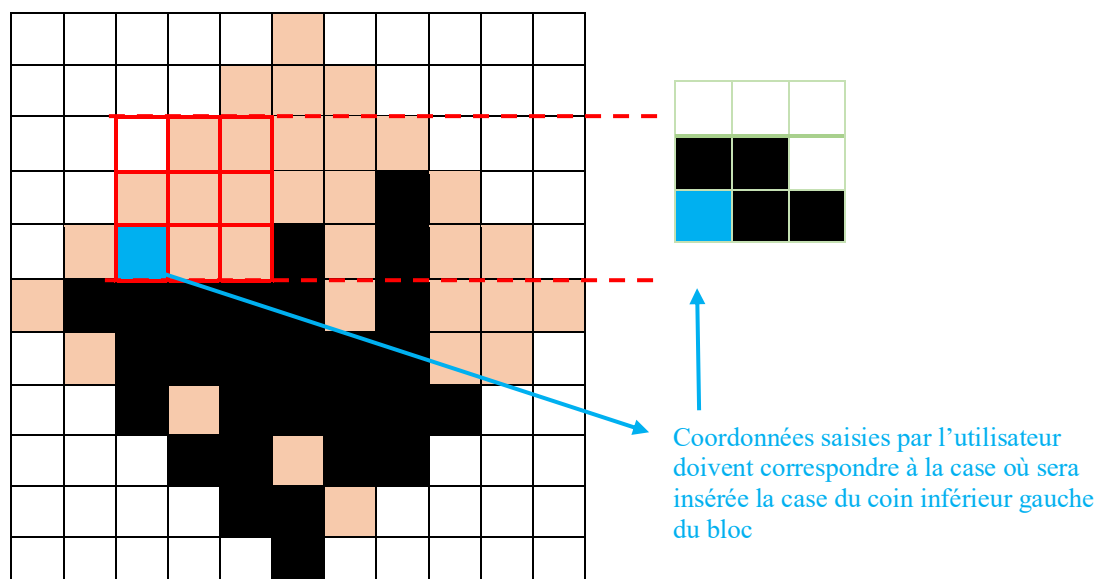
- a. Écrire une fonction `print_blocs(grid)` qui prend en paramètres la forme du plateau choisi, et qui affiche la liste de tous les blocs qui lui sont associés.
- b. Écrire une fonction `select_bloc()` qui permet de sélectionner les blocs à proposer à l'utilisateur selon l'une des 2 politiques expliquées ci-dessus et le type de plateau choisi.

3. Positionnement

- a. Écrire une fonction `valid_position(grid, bloc, i, j)` qui vérifie si le bloc `bloc` peut être placé sur l'emplacement `grid[i][j]` de telle façon que la case inférieure gauche du bloc `bloc` soit positionnée sur `grid[i][j]`.

L'on suppose que l'insertion d'un bloc sur le plateau se fera par la superposition de la matrice qui le contient sur l'endroit où l'on veut l'insérer. Pour cela, il faudra faire coïncider la cellule du coin inférieur gauche du bloc avec les coordonnées saisies par l'utilisateur.

Exemple :



- b. Écrire une fonction `emplace_bloc(grid, bloc, i, j)` qui positionne le bloc `bloc` à la position `(i, j)` de la grille `grid` si et seulement celle-ci est une position valide, et modifie ainsi la grille.

4. Annulation de lignes/colonnes et calcul du score

- a. Écrire une fonction `row_state(grid, i)` qui vérifie si toute la ligne `i` d'une grille `grid` est pleine.
- b. Écrire une fonction `col_state(grid, j)` qui vérifie si toute la colonne `j` d'une grille `grid` est pleine.
- c. Écrire une fonction `row_clear(grid, i)` qui annule la ligne `i` de la grille `grid` en décalant toutes les lignes du haut d'une unité vers le bas. Faire attention car selon la forme du plateau, il se peut que certaines cases de la ligne précédente ne soient plus présentes dans le plateau.
- d. Écrire une fonction `col_clear(grid, j)` qui annule la colonne `j` de la grille `grid`.
- e. Écrire une fonction `update_score()` qui met à jour le score à chaque fois qu'une ligne est annulée (d'autant de points que le nombre de cases annulées dans la ligne).

5. Pour aller plus loin

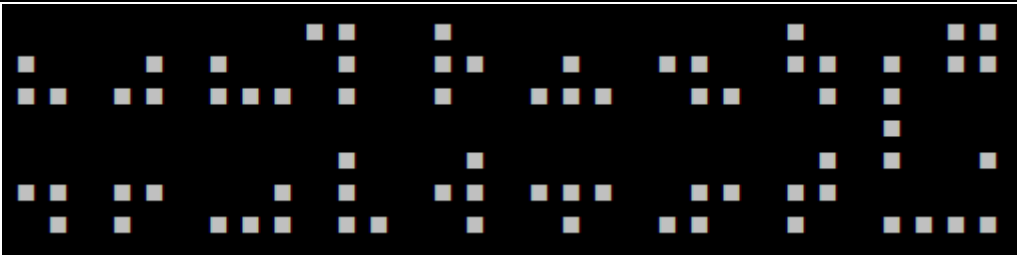
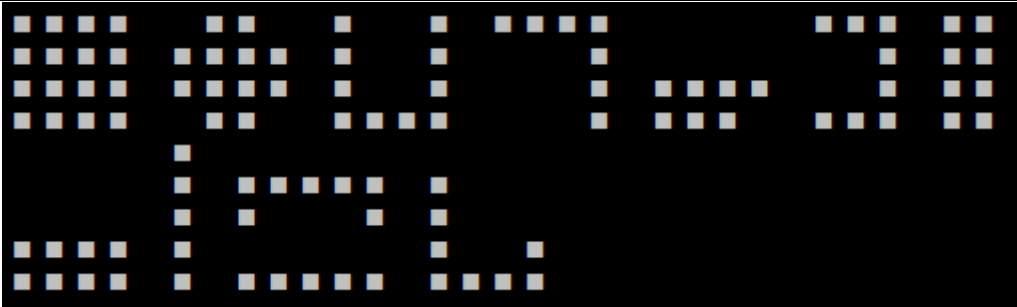
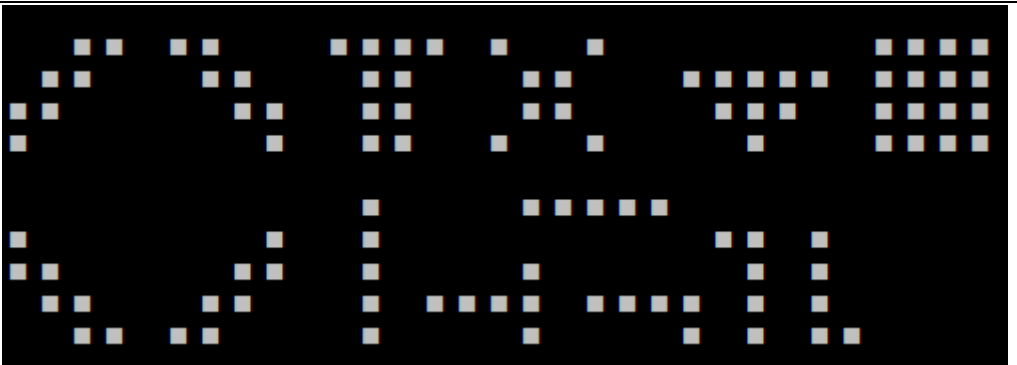

- a. Proposer à l'utilisateur l'option de rotation d'un bloc et écrire la fonction `rotate_bloc(bloc, dir)` qui effectue la rotation du bloc `bloc` d'un quart de tour dans le sens des aiguilles d'une montre si `dir == True`, et dans le sens contraire sinon.
- b. Proposer une fonctionnalité de sauvegarde qui permettra de sauvegarder l'état de la grille.
- c. Proposer une fonctionnalité de chargement qui permettra de reprendre une partie sauvegardée.
- d. En fonction de la valeur du score, ajouter de la difficulté au jeu en suggérant des blocs pas toujours faciles à insérer.

Programme principal

Écrire le programme principal en utilisant les fonctions développées précédemment, et d'autres si nécessaire, qui permet le lancement du jeu.

Le code doit être organisé en modules et en fonctions.

Blocs associés à chaque plateau de jeu

Plateau	Liste des blocs
Tous (Blocs communs à tous les plateaux)	
Cercle	
Losange	
Triangle	

Remarques générales :

- ⇒ Les saisies sécurisées sont à effectuer systématiquement même si elles ne sont pas toujours demandées explicitement
- ⇒ Ne pas hésiter à afficher des messages aux utilisateurs lorsqu'une action n'est pas possible.
- ⇒ Afficher le menu à la fin de chaque action pour pouvoir basculer vers une autre action.