LE JEU DE LA VIE

TP

Imaginé en 1970 par le mathématicien britannique John H. Conway. Ce dernier est décédé le 11 avril 2020, du CoViD-19. Il avait 82 ans.

Partie I — Dessins animés

Exercice 1 — Télécharger le fichier-modèle et le tester. On le trouvera au bout du lien ci-dessous.

`http://lanuitcestfaitpourdormir.org/Documents/jeu_de_la_vie.py`

Exercice 2 — Le programme `EntAléa(`a`, `b`)` renvoie un entier choisi au hasard, avec probabilité uniforme, dans l'intervalle $\{a\,;\,a+1\,;\,\ldots\,;\,b\}$.

```
from random import randint as EntAléa
```

1) Écrire un programme `Choisir(L)` qui étant donnée une liste (non vide) renvoie l'un de ses éléments au hasard.

2) Écrire un programme `Animation_2(M, N, L)` qui colorie chaque case d'une grille M × N avec une couleur choisie aléatoirement dans une liste `L` prescrite de couleurs, en recommençant à chaque instant.

Exercice 3 — En utilisant

```
from time import time as Maintenant
```

modifier les deux programmes précédents pour qu'ils affichent, en sortie, le nombre moyen d'images par secondes (« F.P.S. », *frames per second*) affichées.

Exercice 4

1) Créer deux constantes `NOIR = (0.0, 0.0, 0.0)` et `BLANC = (1.0, 1.0, 1.0)`.

2) Écrire un programme `GrilleBlanche(M, N)` qui crée une grille (représentée par une liste de listes, comme dans le programme fourni en exemple) de taille M × N, dont toutes les cases sont blanches.

3) Écrire un programme `PetitCarréFrénétique(M, N)` qui crée une grille blanche, y place aléatoirement un petit carré noir, et change celui-ci de place à chaque instant.

Exercice 5

1) Écrire un programme `Voisines(M, N, i, j)` qui construit la liste des (coordonnées des) voisines de la case (i, j) dans une grille M × N. Il y en a au plus huit ; moins si l'on est au bord ou dans un coin.

2) En déduire le programme `PetitCarréAgité(M, N)` qui déplace le petit carré noir à chaque instant vers l'une des cases adjacentes à celle où il se trouve.

Exercice 6

1) Écrire un programme `Fondre(L)` qui étant donnée une liste de couleurs renvoie la couleur obtenue en moyennant chaque composante.

2) On suppose disposer d'une liste `C` dont les éléments sont de la forme `(i, j, c)` avec (i, j) des coordonnées dans une grille et c une couleur. Écrire un programme `PlacerLesPetitsCarrés(M, N, C)` qui construit une grille blanche de taille $M \times N$ et y colorie les cases correspondant à la liste `C`. Si plusieurs clients occupent la même position, on fera la moyenne de leurs couleurs.

3) Écrire le programme `DéplacerLesPetitsCarrés(M, N, C)` qui modifie la liste `C` de sorte à ce que chaque petit carré occupe une case voisine de celle où il était.

4) En déduire le programme `PetitsCarrésAgités(M, N, k)`, qui crée une grille $M \times N$ blanche, avec k petits carrés de couleurs aléatoires, et déplace à chaque instant ce petit monde.

Partie II — Le jeu de la vie

Une grille contient des cases blanches et noires. À chaque instant, on la transforme en une nouvelle grille, selon les règles suivantes :

- *si la case (i, j) est blanche, et si elle possède trois voisines noires, elle devient noire, sinon elle reste blanche,*
- *si la case (i, j) est noire, et si elle possède deux ou trois voisines noires, alors elle reste noire, sinon elle devient blanche.*

Exercice 7

1) Écrire un programme `RègleJdlV(c, n)` qui étant donnée une couleur `c` (correspondant à une case dans une grille) valant obligatoirement `NOIR` ou `BLANC` et un entier n, donne la couleur de cette case dans la grille transformée, si on suppose qu'elle possède n voisines noires.

2) D'une manière générale, écrire un programme `ConstruireRègle(P, Q)` qui construit et renvoie un programme `Règle(c, n)` qui donne la couleur, dans la grille transformée, d'une case de couleur c possédant n voisines noires, avec `P` (resp. `Q`) la liste des valeurs de n qui donnent une case noire dans la grille transformée si la case est initialement blanche (resp. noire).

Exercice 8 — Écrire un programme `CompterVoisinesNoires(M, N, G, i, j)` qui calcule le nombre de voisines noires de la case (i, j) dans la grille `G`, supposée de taille $M \times N$.

Exercice 9

1) Écrire un programme `Bernoulli(p)` qui renvoie `True` avec la probabilité p, et `False` sinon. On remarquera qu'il y a dans le fichier-modèle une fonction pré-importée `Aléa`, qui renvoie des nombres choisis dans $[0\,;\,1[$ avec une probabilité uniforme.

2) En déduire un programme `DamierAléa(M, N, p)` qui construit une nouvelle grille de taille $M \times N$ dont les cases sont aléatoirement noires et blanches, avec les probabilités p et $1 - p$ respectivement.

Exercice 10

1) Écrire un programme `Transformation(M, N, G, Règle)` qui construit une nouvelle grille, obtenue en appliquant les règles de transformation à la grille `G`. L'entrée `Règle` est un programme du genre de ceux de **Ex. 7**.

2) En déduire le programme `JeuDeLaVie(M, N, Règle, p = 0.5)` qui part d'une grille de taille $M \times N$ dont les cases sont aléatoirement noires ou blanches (avec les probabilités p et $1 - p$ respectivement), et transforme celle-ci à chaque instant avec la `Règle`.

Exercice 11

1) Deux entiers M et N étant donnés, proposer une bijection φ de $[\![0\,;\,M-1]\!] \times [\![0\,;\,N-1]\!]$ dans $[\![0\,;\,MN-1]\!]$.

2) En déduire un programme `Signature(M, N, G)` qui à toute grille (noire et blanche) associe, de manière bijective, un entier de $[\![0\,;\,\mathrm{MN}-1]\!]$.

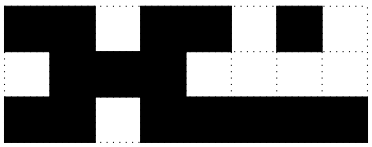
3) Utilisons un peu les `sets` de Pyton. Que font les instructions `E = set()`, `E.add(`x`)` et x `in E` ?

4) Modifier le programme `JeuDeLaVie` pour qu'il s'arrête dès qu'il obtient une image déjà rencontrée. Il renverra le nombre d'images différentes engendrées depuis la configuration de départ.

Partie III — Zoologie

Exercice 12 — Une *structure* est la donnée d'un petit rectangle $m \times n$ de cases noires et blanches. On souhaite pouvoir les construire facilement, et voir comment elles évoluent.

Chacune des m lignes de la structure sera représentée par une liste `[`x_0`, `x_1`, `x_2`, `x_3`]` signifiant que la ligne commence par x_0 cases noires, suivies de x_1 cases blanches, puis à nouveau x_2 cases noires, etc. Ainsi la structure



peut être représentée de manière compacte par la liste de listes

```
[[2, 1, 2, 1, 1],
 [0, 1, 3],
 [2, 1, 5]]
```

1) Écrire un programme `PlacerStructure(G, `i`, `j`, S)` qui peint en noir et en blanc les cases de `G`, pour y faire apparaître la structure `S` (donnée dans sa représentation compacte), de sorte à ce que le coin supérieur gauche de celle-ci soit la case (i, j).

2) En déduire un programme `ÉtudierStructure(M, N, `i`, `j`, S)` qui crée une grille de taille M × N, y place la structure `S` en position (i, j), et détermine le *temps de vie* de la structure (le nombre d'images différentes qu'elle engendre). Cette quantité peut dépendre de la position initiale. Le programme renverra un couple `(`n`, `b`)` où n est le temps de vie, et b est un booléen indiquant si la dernière image est blanche ou non (autrement dit si la structure finit par disparaître ou si elle est persistante).

Exercice 13