

EXERCICE 1: Une grille de Takuzu, ou Binaïro, est une grille carrée à n lignes et autant de colonnes avec n pair. Au départ, chaque case peut contenir 0, 1 ou être vide. En Python, une grille est codée par une liste de listes, les cases vides étant représentées par None. Par exemple, la grille

0	1		0
1		0	1
0	0	1	1
1	1	0	

est codée par la liste `[[0,1, None, 0 ],[1, None, 0, 1] , [0, 0, 1, 1] , [1, 1, 0, None ]]`. Dans la suite, on identifie la grille et la liste qui la représente.

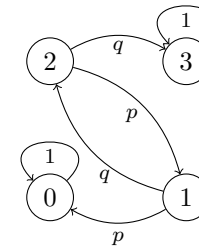
1. Écrire une fonction prenant en argument une grille de taille non précisée et renvoyant le nombre de cases non remplies.
2. Écrire une fonction prenant en argument une grille de taille non précisée et renvoyant un booléen testant s'il existe deux lignes identiques.

EXERCICE 2: Soit $p \in]0, 1[$.

Une puce peut occuper quatre positions numérotées de 0 à 3. À chaque instant $t = n$, où $n \in \mathbb{N}$, elle se déplace (ou pas). On suppose que :

- 0 et 3 sont des puits : si la puce s'y trouve, elle y reste définitivement ;
- si la puce est en 1, alors elle saute à la position 0 à l'instant suivant avec la probabilité p ou à la position 2 avec la probabilité $q = 1 - p$;
- si la puce est en 2, alors elle saute en 1 avec la probabilité p ou en 3 avec la probabilité q .

À $t = 0$, la puce se trouve en 1. Les conditions de déplacement peuvent être représentées schématiquement comme suit :



On suppose que la bibliothèque `random` a été importée avec l'instruction `import random`. On rappelle que pour a un nombre entre 0 et 1, l'expression `random.random() < a` vaut `True` avec une probabilité a .

1. Écrire un programme `saut` prenant en argument le paramètre p et la position de la puce à l'instant n et renvoyant la position de la puce à l'instant suivant $n + 1$.
2. Écrire un programme `position` prenant en argument le paramètre p et un entier naturel n non nul et renvoyant la liste des n premières positions de la puce.

EXERCICE 3:

1. Écrire une fonction python, d'arguments d'entrée i et N , qui simule une marche aléatoire sur $\mathbb{Z}$: le marcheur démarre sur l'entier i et à chaque pas, il avance de 1 ou recule de 1 avec probabilité $1/2$. La marche s'arrête après le N -ième pas et la fonction renvoie l'entier sur lequel le marcheur s'est arrêté.

Un plateau de jeu est constitué de $n + 1$ cases numérotées de 0 à n avec $n > 1$, les cases 0 et n contenant des valeurs a et b , comme ci-dessous où $n = 10$, $a = 1$ et $b = 3$:

1										3
0	1	2	3	4	5	6	7	8	9	10

Pour $i \in \{0, 1, \dots, n\}$, on considère le jeu J_i suivant : on place un pion dans la case de numéro i et tant que l'on n'est pas dans la case 0 ou dans la case n , on lance une pièce équilibrée : si elle tombe sur face, on se déplace vers la gauche ; sinon, on se déplace vers la droite. On admet que la partie se termine presque sûrement : le gain du joueur est la valeur (a ou b) contenue dans la case finale.

2. Écrire une fonction qui prend en paramètres les valeurs n, a, b et i , qui simule ce jeu et qui renvoie le gain.

EXERCICE 4:

1. Écrire une fonction Python qui simule une série de N lancers d'une pièce équilibrée, et qui renvoie la liste des résultats de ces lancers ("Pile" est codé par 1, et "Face" par 0.)
2. Écrire une fonction Python qui simule une série de lancers d'une pièce équilibrée jusqu'à obtention de la configuration "Pile, Pile, Face", et qui renvoie le nombre de lancers nécessaires à l'apparition de cette configuration. À l'aide de cette fonction, évaluer le temps moyen d'attente de cette configuration.

EXERCICE 5:

1. Écrire une fonction `somme(f,a,b,N)` qui retourne $\sum_{k=0}^{N-1} f\left(a + k\frac{b-a}{N}\right)$ où f est une fonction, a et b deux réels ($a < b$) et N un entier supérieur à 1.
2. Écrire une fonction prenant en entrée une fonction `f`, deux réels `a`, `b` qui retourne une valeur approchée de $\int_a^b f(t)dt$.

Utiliser cette fonction pour donner une valeur approchée de $\int_0^{+\infty} e^{-t^2} dt$ à l'aide de l'égalité :

$$\int_0^{+\infty} f(t)dt = \int_0^1 \frac{1}{(1-t)^2} f\left(\frac{t}{1-t}\right) dt$$