Informatique - lundi 15 juin 2026
Devoir nº1 Durée : 1 h

- **Aucun document autorisé. Calculatrice interdite.**
- **Ce sujet est constitué de 3 exercices indépendants.**

Exercice 1. On considère la suite (u_n) définie par : $u_0 = 1$ et $\forall n \in \mathbb{N},\ u_{n+1} = -u_n^2 + 3u_n + 5$.

1. Écrire une fonction `liste_u` prenant en argument $n \in \mathbb{N}$ et renvoyant `[`u_0`,`u_1`,...,`u_n`]`.
2. En utilisant la fonction `liste_u`, écrire un programme permettant de tracer u_k en fonction de k pour $k \in [\![0, 20]\!]$.
3. Écrire une fonction `nb_pair` prenant en argument une liste `L` de nombres entiers et renvoyant le nombre d'éléments pairs que contient `L`.
4. En utilisant les fonctions précédentes, écrire une fonction prenant en argument $n \in \mathbb{N}$ et renvoyant le nombre d'éléments pairs parmi les termes u_k pour $k \in [\![n, 2n]\!]$.

Exercice 2. On dit qu'une matrice M est *magique* lorsque la somme des éléments de chacune de ses lignes et de chacune de ses colonnes est la même. Par exemple, la matrice

$$M = \begin{pmatrix} 0 & 4 & 8 \\ 1 & 6 & 5 \\ 11 & 2 & -1 \end{pmatrix}$$

est magique, car la somme des éléments de chacune de ses lignes et de chacune de ses colonnes vaut 12. En revanche, la matrice

$$N = \begin{pmatrix} 6 & 7 & 8 \\ 8 & 7 & 6 \end{pmatrix}$$

n'est pas magique : la somme des éléments de sa première colonne vaut 14, tandis que celle des éléments de sa première ligne vaut 21.

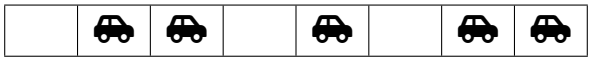
1. En prenant soin d'importer la bibliothèque appropriée, écrire une commande Python permettant de définir la matrice N proposée ci-dessus.
2. (a) Écrire une fonction `somme_ligne` prenant en arguments une matrice M et un indice de ligne i, et renvoyant la somme des éléments de la ligne i de M.
   (b) Écrire une fonction `somme_colonne` prenant en arguments une matrice M et un indice de colonne j, et renvoyant la somme des éléments de la colonne j de M.
3. En déduire une fonction `magique` prenant en argument une matrice et renvoyant `True` si la matrice est magique et `False` sinon.
4. Expliquez ce que renvoie la fonction `mystere` suivante :

```
def mystere(M):
    if magique(M):
        S = sum(sum(M))
        return S/np.size(M,0) == S/np.size(M,1)
```

   En déduire que si M est une matrice magique dont la somme sur chacune des lignes et des colonnes est non nulle, alors M est forcément carrée. Existe-t-il des matrices magiques non carrées ?

Exercice 3 (d'après Agro-Véto 2023, filière TB)**.** On modilise l'état du trafic routier dans un tunnel par une liste `T` de longueur n contenant uniquement des 0 et des 1. On assimile pour cela le tunnel à une suite de n cases représentant chacune un emplacement où il peut y avoir ou non un véhicule. On place alors un 1 dans la liste lorsqu'un emplacement est occupé par un véhicule et un 0 lorsqu'il est vide.

Par exemple `T = [0,1,1,0,1,0,1,1]` correspond au trafic suivant :



On suppose d'abord que le tunnel est à sens unique et n'a qu'une seule voie sur laquelle toutes les voitures ont le même comportement. On modélise ce comportement de manière discrète, en supposant qu'à chaque "tour" toutes les voitures suivent simultanément les règles d'évolution suivantes :

- une voiture avance d'une case vers la droite si la case à sa droite est libre,
- si la case à sa droite n'est pas libre alors la voiture reste immobile,
- si une voiture se trouve au bout du tunnel alors elle en sort obligatoirement.

Par exemple, si l'état du trafic à l'instant initial est représenté par `T = [0,1,1,0,1,0,1,1]` alors à l'instant suivant il sera représenté par `L = [0,1,0,1,0,1,1,0]`.

1. Donner, sans justifier, la liste correspondant à l'état du trafic à l'étape suivante celle représentée par `[0,1,0,1,0,1,1,0]`.
2. Écrire une fonction `compte` prenant en argument une liste `T` correspondant à un état du trafic et renvoyant le nombre de voitures dans le tunnel.
3. En utilisant les règles d'évolution du trafic, écrire une fonction `evol` prenant en argument une liste `T` correspondant à un état du trafic et renvoyant une autre liste `L` correspondant à l'état du trafic suivant `T`. Par exemple, `evol([0,1,1,0,1,0,1,1])` doit renvoyer `[0,1,0,1,0,1,1,0]`. *On pourra commencer par initialiser la liste `L` à la liste ne contenant que des* 0 *puis la modifier au fur et à mesure.*
4. En utilisant les fonctions précédentes, écrire une fonction `vider` prenant en argument une liste `T` correspondant à un état du trafic, et renvoyant le nombre d'étapes nécessaires avant que toutes les voitures aient quitté le tunnel.

On suppose maintenant que le tunnel a une deuxième voie où les véhicules circulent dans l'autre sens, et selon les mêmes règles d'évolution de la première voie. Le trafic est donc représenté par deux listes de 0 et de 1, notées `T` et `T_bis`.

5. Écrire une fonction `miroir` prenant en argument une liste `L` et renvoyant la liste dans l'autre sens. Par exemple `miroir([1,2,3])` doit renvoyer `[3,2,1]`.
6. En déduire une fonction `evol2` prenant en argument des listes `T` et `T_bis` correspondant à un état du trafic et renvoyant le couple de listes correpsondant à l'état du trafic suivant. Par exemple, si `T = [0,1,1,0,1,0,1,1]` et `T_bis = [1,0,0,1,1,0,1,0]` alors `evol2(T,T_bis)` doit renvoyer le couple `(L,L_bis)` où `L = [0,1,0,1,0,1,1,0]` et `L_bis = [0,0,1,0,1,1,0,0]`.
7. *Question bonus. Cette question sera corrigée uniquement si toutes les autres questions du sujet sont traitées proprement.* On suppose maintenant qu'il y a deux voies qui vont dans le même sens, et que les conducteurs de la voie la plus à droite peuvent doubler une voiture les précédant directement en passant par la voie de gauche. Énoncer des règles d'évolution du trafic correspondant à cette situation, et implémenter une fonction Python réalisant l'évolution du trafic.