Chapitre 3 : Programmation — Exercices

Exercice 1 ★ :

a) Écrire une fonction **`somme(tab)`** qui calcule et renvoie la somme de tous les éléments d'un tableau d'entiers.
b) En utilisant la fonction **`somme`**, écrire une fonction **`moyenne(tab)`** qui calcule et renvoie la moyenne des éléments du tableau tab, que l'on supposera non vide.
c) À l'aide de votre fonction, calculez la moyenne du tableau T = [1, 4, 7, 6] et affichez le résultat à l'aide de l'instruction print.

Exercice 2 ★ :

Écrire une fonction `echange(tab, i, j)` qui échange les éléments i et j du tableau tab.

Exercice 3 ★★ :

Écrire une fonction `inverse(tab)` qui prend en paramètre un tableau tab et renvoie un tableau inversé. Ex : `inverse([2, 5, 1, 12])` renvoie `[12, 1, 5, 2]` .

Exercice 4 ★★ :

a) Écrire une fonction `verifieTaille(tab1, tab2)` qui renvoie True si les tableaux d'entiers tab1 et tab2 sont de même longueur et False sinon.
b) Écrire une fonction `hamming(tab1, tab2)` qui prend deux tableaux en paramètres et qui renvoie le nombre de valeurs auxquels les deux tableaux diffèrent. Si les deux tableaux ne font pas la même taille, on renverra le nombre -1.

Exercice 5 ★★★ : fusion de tableaux

En informatique, on souhaite souvent trier dans l'ordre des données sous forme de tableaux. Certains méthodes (tri-fusion par exemple) nécessitent de fusionner des tableaux en respectant l'ordre croissant.

1) Considérons les deux tableaux d'entiers : tab1 = [4, 6, 8] et tab2 = [1, 2, 5, 12] . En utilisant la logique ci-dessous, sur une feuille, fusionnez ces deux tableaux dans un nouveau tableau tab.

   Dans tab1, on regarde l'élément 0 : c'est 4 .
   Dans tab2, on regarde l'élément 0 : c'est 1 .
   1 < 4, donc on rajoute 1 à tab et on va regarder l'élément 1 de tab2 : tab = [1]
   Dans tab1, on regarde l'élément 0 : c'est 4 .
   Dans tab2, on regarde l'élément 1 : c'est 6 .
   ……

2) En vous aidant de la question 1, écrivez une fonction fusion(tab1, tab2) qui fusionne deux tableaux triés dans l'ordre croissant et renvoie un nouveau tableau trié dans l'ordre croissant.

Problème ★★★ : système de Lotka-Volterra

a. Dans une réserve naturelle, on considère qu'il n'y a que deux espèces (bonjour la biodiversité…) : des lapins et des lynx. On suppose qu'initialement il y a 53000 lapins et 9000 lynx et on se demande comment vont évoluer les populations de ces deux espèces. Pour simplifier la situation, on suppose qu'il n'y a aucune intervention extérieure.

Si on note u_n le nombre de lapins l'année numéro n et v_n le nombre de lynx à l'année numéro n, alors on peut modéliser leurs évolutions par :

$$\begin{cases} u_{n+1} = (1 + a - bv_n) \times u_n \\ v_{n+1} = (1 - c + du_n) \times v_n \end{cases}$$

où : a (resp. c) représente le taux d'évolution des lapins (reps des lynx) et b et d représentent les taux d'évolution liées aux lynx qui mangent les lapins.

Nous allons prendre pour valeur a=0.09, b=0.00001, c=0.25 et d=0.000005.

1) Que représente u_{n+1} et v_{n+1} ?

2) Nous allons stocker l'ensemble de l'histoire des populations dans deux tableaux appelés **lapins** et **lynx**. Imaginons que les populations aient eu cette évolution jusqu'à l'année 4 :

lapins = [53000, 51000, 42000, 47000, 37000]

lynx = [9000, 9500, 12000, 8000, 9000]

Vous souhaitez calculer le nombre de lapins et de lynx une année plus tard. Quel élément de vos tableaux allez-vous utiliser ? Comment accédez-vous à ces éléments en langage Python ?

3) Créez une fonction nombreProies(proie, predateur) et une fonction nombrePreds(proie, predateur) prenant en paramètre un tableau proie et un tableau predateur et renvoyant le nombre de proies (resp. de prédateurs) à l'année suivante. On utilisera le résultat de la question 2 pour répondre à cette question.

4) Écrire une fonction genererPopulation(nMax) prenant en paramètre nMax, le nombre d'années de simulation en renvoyant deux tableaux contenant l'historique des populations de lapins et de lynx pour les nMax années à venir.

Pour renvoyer deux tableaux, on utilise l'instruction :

```
return tableau1, tableau2
```

Pour obtenir ces deux tableaux, on utilisera l'instruction :

```
lapins, lynx = genererPopulation(nMax)
```

5) Recopier les instructions pour voir graphiquement les évolutions des populations. L'intervention de l'homme est-elle utile pour conserver les deux espèces ?

```
from matplotlib.pyplot import *
plot(list(range(nMax)), lapins, 'r')
plot(list(range(nMax)), lynx, 'b')
show()
```

Chapitre 3 : Programmation — Exercices

Exercice 1 ★ :

a) Écrire une fonction **`somme(tab)`** qui calcule et renvoie la somme de tous les éléments d'un tableau d'entiers.
b) En utilisant la fonction **`somme`**, écrire une fonction **`moyenne(tab)`** qui calcule et renvoie la moyenne des éléments du tableau tab, que l'on supposera non vide.
c) À l'aide de votre fonction, calculez la moyenne du tableau T = [1, 4, 7, 6] et affichez le résultat à l'aide de l'instruction print.

Exercice 2 ★ :

Écrire une fonction `echange(tab, i, j)` qui échange les éléments i et j du tableau tab.

Exercice 3 ★★ :

Écrire une fonction `inverse(tab)` qui prend en paramètre un tableau tab et renvoie un tableau inversé. Ex : `inverse([2, 5, 1, 12])` renvoie `[12, 1, 5, 2]` .

Exercice 4 ★★ :

a) Écrire une fonction `verifieTaille(tab1, tab2)` qui renvoie True si les tableaux d'entiers tab1 et tab2 sont de même longueur et False sinon.
b) Écrire une fonction `hamming(tab1, tab2)` qui prend deux tableaux en paramètres et qui renvoie le nombre de valeurs auxquels les deux tableaux diffèrent. Si les deux tableaux ne font pas la même taille, on renverra le nombre -1.

Exercice 5 ★★★ : fusion de tableaux

En informatique, on souhaite souvent trier dans l'ordre des données sous forme de tableaux. Certains méthodes (tri-fusion par exemple) nécessitent de fusionner des tableaux en respectant l'ordre croissant.

3) Considérons les deux tableaux d'entiers : tab1 = [4, 6, 8] et tab2 = [1, 2, 5, 12] . En utilisant la logique ci-dessous, sur une feuille, fusionnez ces deux tableaux dans un nouveau tableau tab.

   Dans tab1, on regarde l'élément 0 : c'est 4 .
   Dans tab2, on regarde l'élément 0 : c'est 1 .
   1 < 4, donc on rajoute 1 à tab et on va regarder l'élément 1 de tab2 : tab = [1]
   Dans tab1, on regarde l'élément 0 : c'est 4 .
   Dans tab2, on regarde l'élément 1 : c'est 6 .
   ……

4) En vous aidant de la question 1, écrivez une fonction fusion(tab1, tab2) qui fusionne deux tableaux triés dans l'ordre croissant et renvoie un nouveau tableau trié dans l'ordre croissant.

Problème ★★★ : système de Lotka-Volterra

a. Dans une réserve naturelle, on considère qu'il n'y a que deux espèces (bonjour la biodiversité…) : des lapins et des lynx. On suppose qu'initialement il y a 53000 lapins et 9000 lynx et on se demande comment vont évoluer les populations de ces deux espèces. Pour simplifier la situation, on suppose qu'il n'y a aucune intervention extérieure.

Si on note u_n le nombre de lapins l'année numéro n et v_n le nombre de lynx à l'année numéro n, alors on peut modéliser leurs évolutions par :

$$\begin{cases} u_{n+1} = (1 + a - bv_n) \times u_n \\ v_{n+1} = (1 - c + du_n) \times v_n \end{cases}$$

où : a (resp. c) représente le taux d'évolution des lapins (reps des lynx) et b et d représentent les taux d'évolution liées aux lynx qui mangent les lapins.

Nous allons prendre pour valeur a=0.09, b=0.00001, c=0.25 et d=0.000005.

1) Que représente u_{n+1} et v_{n+1} ?

2) Nous allons stocker l'ensemble de l'histoire des populations dans deux tableaux appelés **lapins** et **lynx**. Imaginons que les populations aient eu cette évolution jusqu'à l'année 4 :

lapins = [53000, 51000, 42000, 47000, 37000]

lynx = [9000, 9500, 12000, 8000, 9000]

Vous souhaitez calculer le nombre de lapins et de lynx une année plus tard. Quel élément de vos tableaux allez-vous utiliser ? Comment accédez-vous à ces éléments en langage Python ?

3) Créez une fonction nombreProies(proie, predateur) et une fonction nombrePreds(proie, predateur) prenant en paramètre un tableau proie et un tableau predateur et renvoyant le nombre de proies (resp. de prédateurs) à l'année suivante. On utilisera le résultat de la question 2 pour répondre à cette question.

4) Écrire une fonction genererPopulation(nMax) prenant en paramètre nMax, le nombre d'années de simulation en renvoyant deux tableaux contenant l'historique des populations de lapins et de lynx pour les nMax années à venir.

Pour renvoyer deux tableaux, on utilise l'instruction :

```
return tableau1, tableau2
```

Pour obtenir ces deux tableaux, on utilisera l'instruction :

```
lapins, lynx = genererPopulation(nMax)
```

5) Recopier les instructions pour voir graphiquement les évolutions des populations. L'intervention de l'homme est-elle utile pour conserver les deux espèces ?

```
from matplotlib.pyplot import *
plot(list(range(nMax)), lapins, 'r')
plot(list(range(nMax)), lynx, 'b')
show()
```