TP 3 - TABLEAUX

Info1.Algo1 - 2022-2023 Semestre Impair

Objectif : Le but de cette séance est de manipuler les tableaux.

Tableaux

L'implémentation Python propose de manipuler les tableaux comme des listes dynamiques et hétérogènes, mais dans ce TP on se restreindra volontairement à manipuler des tableaux comme des vecteurs :

- composés d'un nombre **fixe** d'éléments, cette taille étant déterminée **dès l'initialisation**
- composés d'éléments **de même type**, chaque élément étant repéré par son indice.

indice	0	1	2	3	4	5	6	7
valeur	10	20	30	40	50	60	70	80

En particulier toutes les opérations de modification de la structure (`append`, `insert`, `del`, `remove`...) ainsi que les opérations enrichies (`clear`, `sorted`, `reverse`, ...) sont interdites sur ce TP.

Exemple :

```
# création d'un tableau de 20 entiers
tableau = [0]*20
# Ecriture dans le tableau à partir d'une saisie
tableau [0] = int(input())
# modification d'un élément du tableau
tableau[1] = 2 * tableau[0]

# parcourir de tous les éléments, indice allant de 0 à len(tableau)-1
for i in range(len(tableau)):
  tableau[i] = tableau[i] + 1
```

On notera que les **chaînes de caractères**, peuvent être manipulées comme des **tableaux** de caractères.

Applications directes

Exercice 1 - Initialisation, saisie, parcours ★

On souhaite pouvoir initialiser un tableau de notes (le nombre de notes étant connu à l'avance), saisir les différentes notes puis calculer la moyenne des celles-ci.

Compléter les quatre fonctions du fichier **ex01__initialisation__saisie__parcours.py** pour que les tests automatiques passent.

Exercice 2 - Recherche de l'indice d'un élément ★

Même si certaines structures de données sont plus adaptées à la recherche d'un élément dans une collection de valeurs, on a parfois besoin de déterminer si une valeur est présente parmi les valeurs contenues dans un tableau.

Compléter la fonction `recherche` du fichier **ex02__recherche__tableau.py** qui permet de rechercher une valeur dans un tableau.

La valeur peut être présente (éventuellement plusieurs fois) ou bien absente.

- Dans le cas où la valeur est présente, on doit retourner le premier indice du tableau qui la contient.
- Dans le cas où la valeur n'est pas présente, on retournera `-1`.

Votre programme doit fonctionner (sans modification) à la fois pour des tableaux contenant des nombres ou des tableaux contenant des chaines de caractères.

Exercice 3 - Calcul de l'étendue d'une série statistique donnée sous forme de tableau ★

Les tableaux de valeurs numériques modélisent souvent des données statistiques.

On est donc parfois amener à réaliser des calculs statistiques du des données présentées en tableau.

L'étendue d'une série statistique est la différence entre la valeur la plus grande et la valeur la plus petite de cette série.

Exemple :

L'étendue de la série statistique `[30,20,10,20,30,40]` est égale à `40-10` c'est-à-dire `30`.

Compléter la fonction `etendue` du fichier **ex03__etendue.py** qui permet de calculer l'étendue d'une série statistique qui lui est passé en paramètre.

Exercices Tableaux

Exercice 4 - Le chat, la souris et la chaîne de caractères ★

On vous donne en entrée une chaîne de caractère dans laquelle un chat est représenté par un `'C'` et une souris est représentée par un `'s'`.

Le reste de la chaîne de caractère est constituée du caractère `'.'` :

`...C.....s......`

Le chat essaie d'attraper la souris, il se prépare à sauter... Un chat (normalement constitué) est capable de sauter d'au plus **trois positions** (vers la droite ou vers la gauche).

Dans le fichier **ex04__matrice__chats__souris** compléter la fonction `chat_attrape_souris` qui prend en paramètre un matrice de caractères et un entier `saut` et retourne `True` si le chat peut attraper la souris en un saut, `False` sinon.

Exemple :

Le chat a la possibilité de sauter de 3 caractères, donc :

- `C.....s` : 5 caractères entre le chat et sa cible, la souris s'échappe, la fonction retourne `False`.
- `C...s` : 3 caractères entre les deux, le chat saute et attrape la souris, la fonction retourne `True`.

Exercice 5 - Moyenne glissante ★★

1) Dans le fichier **ex05__somme__glissante.py**, compléter le corps de la fonction `somme_deux_a_deux` qui prend en paramètre un **tableau** d'entiers contenant au moins deux éléments et retourne le **tableau** constitué des sommes deux à deux de ses éléments.

La fonction `somme_deux_a_deux` doit respecter l'abstraction tableau.

Exemple : si le tableau donné en paramètre est `[1,2,3,4]`, le tableau retourné par la fonction est `[3,5,7]`, c'est-à-dire `[1+2,2+3,3+4]`.

2) On souhaite généraliser la fonction `somme_deux_a_deux` au cas où l'on somme `n` éléments consécutifs du tableau donné en paramètre.

Exemple : Lorsque le tableau donné en paramètre est `[1,2,3,4]` :

- si `n` vaut `3`, le tableau retourné par la fonction est `[6,9]`, c'est-à-dire `[1+2+3,2+3+4]`.
- si `n` vaut `4`, le tableau retourné par la fonction est `[10]`, c'est-à-dire `[1+2+3+4]`.

Compléter le corps de la fonction `somme_glissante` qui accepte les paramètres suivants :

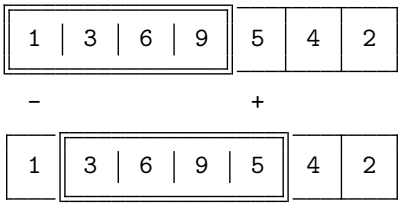
- `tableau` : un tableau d'entiers.
- `n` : un entier.

La fonction retourne alors le tableau constitué des sommes de `n` éléments consécutifs de `tableau`.

3) Selon comment la question **2** a été résolue, la fonction peut bloquer *(sans faire d'erreur)* lors du test de la fonction `somme_deux_a_deux` optimisée. Si c'est le cas, l'objectif de cette question **3** est donc de passer ce dernier test.

Une manière d'optimiser le code de la fonction `somme_deux_a_deux` est de ne pas recalculer séparément chacune des sommes du tableau retourné.

Exemple :

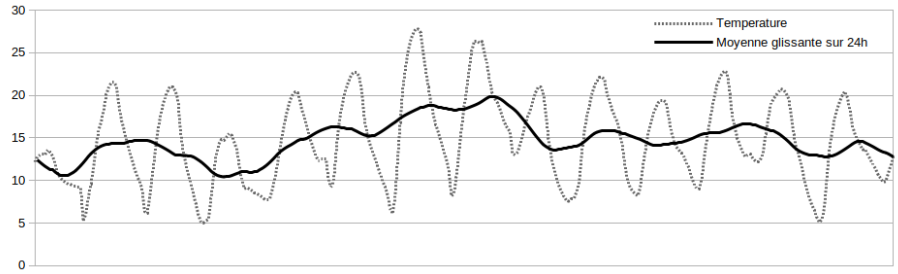


Dans l'exemple ci-dessus, plutôt que de calculer séparément les sommes `1+3+6+9` et `3+6+9+5`, on peut retrancher le nombre `1` puis ajouter le nombre `5`.

Optimiser la fonction `somme_deux_a_deux` à l'aide de la méthode suivante :

- Calculer la première somme.
- Chaque nouvelle somme est calculée à partir de la précédente en retranchant / ajoutant les 2 nombres appropriés.

  Application : Si l'on divise chaque somme par **n**, on obtient une moyenne glissante, qui peut être utilisée pour lisser des données présentant une forte variation locale. Sur le graphique ci-dessous, on a mesuré la température heure par heure. On observe l'alternance jour/nuit. La moyenne glissante sur 24h permet de mieux apprécier les variations d'un jour à l'autre.



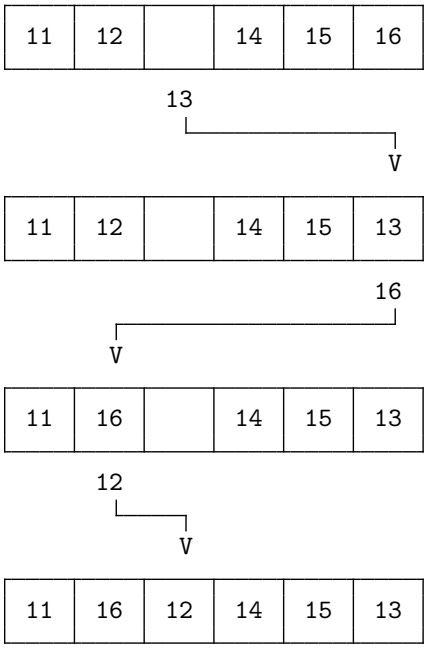
Exercice 6 - Permutation circulaire ★★

Une permutation circulaire ou **cycle** est un type particulier de permutation des éléments d'un tableau, et fonctionne de la façon suivante :

Si le cycle `[2,5,1]` est appliqué au tableau `[11,12,13,14,15,16]`

Alors :

- L'élément d'indice `2` (c'est-à-dire `13`) est envoyé à l'indice `5`
- L'élément d'indice `5` (c'est-à-dire `16`) est envoyé à l'indice `1`
- L'élément d'indice `1` (c'est-à-dire `12`) est envoyé à l'indice `2`



Ainsi, aucun des éléments du tableau n'est perdu *(les indices de cycles doivent donc tous être différents)* et le tableau devient `[11,16,12,14,15,13]`

Remarque : Un cycle vide, ou de taille `1`, ne modifie pas le tableau.

Dans le fichier **ex06_appliquer_cycle.py**, compléter le code de la fonction `appliquer_cycle` qui accepte en paramètres le tableau à modifier ainsi que le cycle à appliquer, et renvoie le tableau modifié. La fonction `appliquer_cycle` doit respecter l'abstraction tableau.