

Devoir surveillé n°1 - Test de rentrée - Jeudi 4 septembre 2025

Version n°1

Rédiger sur une copie double. La calculatrice n'est pas autorisée.

Consignes de présentation :

- Nom Prénom lisibles en haut à gauche, classe en haut à droite, DS n°1 au milieu et encadré à la règle.
- Vous laissez ensuite 5 ou 6 centimètres pour mes éventuels commentaires, délimités par deux traits tracés à la règle.
- S'il n'y a pas de marge déjà tracée sur votre feuille, vous en tracez une à 4 cm du bord de la feuille.
- Les codes Python demandés doivent être écrit très lisiblement, avec une indentation très claire.

-
- Exercice 1 :** 1. Ecrire une fonction `cherche(L,elt)` qui renvoie **un booléen** indiquant si l'élément `elt` est présent ou non dans `L`, sans utiliser le test `in`.
2. Donner, en justifiant, la complexité en temps dans le pire des cas, en fonction de la taille n de la liste `L`.

Exercice 2 : Ecrire une fonction `maximum(L)` qui renvoie **un couple** constitué de la valeur de l'élément maximum de la liste `L`, et à la position de ce maximum dans la liste (si ce maximum est atteint plusieurs fois, l'indice renvoyé sera le plus petit).

- Exercice 3 :** 1. Recopier et compléter le script suivant (lignes 6, 9, 11, pour que la fonction trie la liste `L` selon les valeurs croissantes, en suivant le principe du **tri par insertion**.

Remarque : cette fonction ne retourne rien, quand on l'exécute elle modifie la liste passée en paramètre.

```
1 def tri_insertion (L) :  
2     for k in range(1, len(L)) :  
3         x = L[k] # élément à placer  
4         j = k  
5         # on cherche la place de x parmi le début de la liste , triée.  
6         while ..... : # tant que pas au début de la liste , et  
7             # que la future place de x n'est pas trouvée  
8             L[j] = L[j-1] # on décale le terme d'avant  
9             j = ..... # on décale vers la gauche le compteur qui cherche la future place de x  
10            ..... # on place x dans la position libérée
```

2. Donner, en justifiant, la complexité en temps dans le pire des cas, en fonction de la taille n de la liste `L`.

Devoir surveillé n°1 - Test de rentrée - Jeudi 4 septembre 2025

Version n°2

Rédiger sur une copie double. La calculatrice n'est pas autorisée.

Consignes de présentation :

- Nom Prénom lisibles en haut à gauche, classe en haut à droite, DS n°1 au milieu et encadré à la règle.
- Vous laissez ensuite 5 ou 6 centimètres pour mes éventuels commentaires, délimités par deux traits tracés à la règle.
- S'il n'y a pas de marge déjà tracée sur votre feuille, vous en tracez une à 4 cm du bord de la feuille.
- Les codes Python demandés doivent être écrit très lisiblement, avec une indentation très claire.

-
- Exercice 1 :** 1. Ecrire une fonction `somme_positifs(S)` qui renvoie la somme des éléments **positifs** de `L`, sans utiliser la fonction `sum`.
2. Donner, en justifiant, la complexité en temps dans le pire des cas, en fonction de la taille n de la liste `L`.

Exercice 2 : Ecrire une fonction `minimum(L)` qui renvoie **un couple** constitué de la valeur de l'élément minimum de la liste `L`, et à la position de ce minimum dans la liste (si ce minimum est atteint plusieurs fois, l'indice renvoyé sera le plus petit).

- Exercice 3 :** 1. Recopier et compléter le script suivant (lignes 4, 5, 13, 14), pour que la fonction retourne un booléen indiquant si `elt` est présent ou non dans la liste `L`, en suivant le principe du **la recherche par dichotomie**. On suppose que la liste `L` est **triée selon les valeurs croissantes**.

```
1 def cherche_dicho(L, elt) :  
2  
3     ind_deb = .....  
4     ind_fin = .....  
5     while ind_deb <= ind_fin: # tant que les compteurs ne se sont pas rejoints  
6         m = ..... # indice médian  
7         if L[m] == elt : # elt trouvé à la position m  
8             return True  
9         elif L[m] < elt : # on garde la partie droite  
10            ind_deb = m+1  
11        else: # on garde la partie gauche  
12            ind_fin = .....  
13    return .....
```

2. Donner, en justifiant brièvement, la complexité en temps dans le pire des cas, en fonction de la taille n de la liste `L`.