Soit `L` une liste. Une **tranche** de `L` est une sous-liste de `L` composée d'éléments consécutifs.
Par exemple, si `L = [1, -3, 4, -2, 1, -5, 7]` alors `[1, -3, 4]` et `[-2, 1]` sont des tranches de `L`.
Étant donnée une liste `L`, on souhaite calculer la somme minimum d'une tranche de `L`.
Par exemple, si `L = [1, -3, 4, -2, 1, -5, 7]`, alors la tranche de somme minimum est `[-2, 1, -5]`, de somme -6 ($= -2 + 1 - 5$). En effet, toutes les autres tranches ont une somme supérieure.

I Première méthode : force brute

1. Écrire une fonction `somme` telle que `somme(L, i, j)` renvoie la somme des éléments de `L` de l'indice `i` à `j - 1` (c'est-à-dire $\sum_{k=i}^{j-1}$ `L[k]`).
   Par exemple, `somme([1, -3, 4, -2, 1, -5, 7])` doit renvoyer 3 ($= 4 - 2 + 1$).
2. En déduire une fonction `tranche_min1` renvoyant la tranche minimum d'une liste, en appelant `somme(L, i, j)` pour tout indice `i`, `j` valides et en conservant le minimum.
   Vérifier que `tranche_min1([1, -3, 4, -2, 1, -5, 7])` renvoie -6.
3. Quelle est la complexité de la fonction précédente, en fonction de la taille n de la liste ?

Dans la suite, on essaie de résoudre le même problème mais avec une meilleure complexité.

II Deuxième méthode : en stockant les sommes partielles

1. Écrire une fonction `sommes_partielles` telle que, si `L` `sommes_partielles(L)` renvoie une liste `S` telle que `S[i]` contienne la somme des i premiers éléments de `L` (c'est-à-dire `S[i]` $= \sum_{k=0}^{i-1}$ `L[k]`).
   Par exemple, `sommes_partielles([1, -3, 4, -2, 1, -5, 7])` doit renvoyer `[0, 1, -2, 2, 0, 1, -4, 3]`.
2. Comment exprimer $\sum_{k=i}^{j}$ `L[k]` en fonction de `S[j + 1]` et `S[i]` ?
3. En déduire une fonction `tranche_min2` renvoyant la tranche minimum d'une liste en utilisant `sommes_partielles` au lieu de `somme`.
4. Quelle est la complexité de la fonction précédente, en fonction de la taille n de la liste ?

III Troisième méthode : algorithme de Kadane

On admet que l'algorithme suivant permet de calculer, dans `s`, la tranche minimum de `L` :

```
s = 0
s_cur = 0
Pour chaque element e de L:
    s_cur = s_cur + e
    Si e < s_cur:
        s_cur = e
    Si s_cur < s:
        s = s_cur
```

1. Traduire ce pseudo-code en une fonction `tranche_min3` et vérifier qu'elle fonctionne.
2. Quelle est la complexité de cette méthode ? Comparer avec les 2 autres méthodes.