new idea

Conseils

Maîtrisez d'abord l'algorithme

Prenez le temps de bien comprendre chaque étape avant d'écrire une seule ligne de code. Si un point reste flou, revenez-y, et dessinez un schéma.
Un algorithme clair dans votre tête se traduit par un code clair à l'écran.

Divisez pour mieux régner

Décomposez le problème en sous-tâches simples et gérables.
Cela rend le travail moins intimidant et facilite le débogage.

Validez étape par étape

Ne passez pas à la tâche $t + 1$ avant d'avoir vérifié que la tâche t fonctionne correctement. Testez avec des exemples simples, puis des cas limites. Cette habitude vous évitera des erreurs en cascade.

Un code qui évolue pas à pas, c'est comme un puzzle qu'on assemble pièce par pièce. Chaque étape validée est une petite victoire qui vous rapproche de la solution finale.

Fonctions

entrée → **fonction** → sortie

Pour chaque fonction, soyez très clair sur son *entrée* et sa *sortie*.

Une fois qu'une fonction est écrite, on doit pouvoir l'utiliser simplement en connaissant son **nom**, ses **paramètres** et sa **valeur de retour**.

Dessins

Dessiner des tableaux et des scénarios : une étape cruciale

Pour suivre l'évolution des indices et des références

Pour éviter les erreurs logiques (décalage, modification inattendue)

Un simple dessin rend la récursion **concrète**

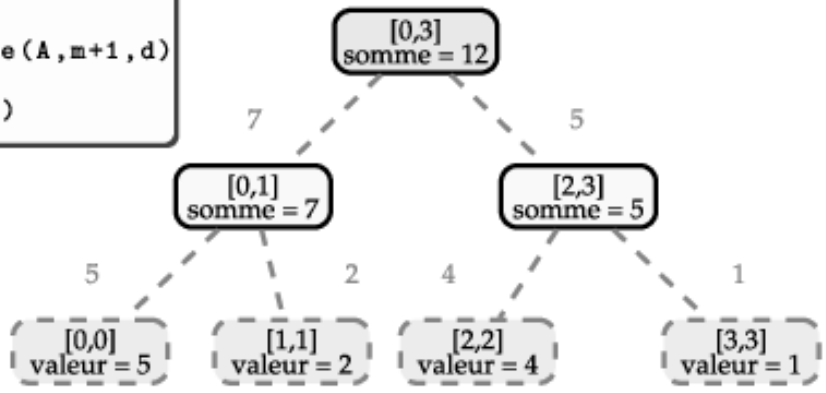
visualiser la décomposition d'un problème

```
def somme(A, g, d):
    if g == d:
        return A[g]

    m = (g+d)//2

    return somme(A,g,m) + somme(A,m+1,d)

print( somme([5,2,4,1], 0, 3) )
```



CONSEILS

Partie PYTHON est relativement facile,
partie ALGORITHMIQUE est **difficile**.

Ordre suggéré pour la révision du matériel pour évaluations :

urgence (sur 10)	tâche
10	**lisez les diapositives des CMs**
10	**refaire soigneusement les questions pour TOUS les TD**
6	**lecture et exercices du notebook Jupyter**
4	**lisez le chapitre du livre pour chaque CM**
4	**assurez-vous de bien comprendre les TPs**