
SOMME DES $n + 1$ PREMIERS ENTIERS

Problématique Pour définir la somme des $n + 1$ premiers entiers, on a l'habitude d'écrire la formule suivante :

$$0 + 1 + 2 + \dots + n$$

Bien que cette formule nous paraisse simple et intuitive, il n'est pas si évident de l'utiliser pour écrire en Python une fonction `somme(n)` qui renvoie la somme des $n + 1$ premiers entiers. L'une des difficultés est de trouver un moyen de programmer la répétition des calculs qui est représentée par la notation $+\dots+$.

Tout d'abord, nous chercherons un moyen d'implémenter de cette fonction à l'aide d'une méthode itérative. L'introduction d'une définition mathématique de cette somme, nous permettra ensuite d'aborder ce problème par une nouvelle méthode de programmation ; *la récursivité*.

Méthode itérative Utilise une boucle `for` et une variable locale pour accumuler la somme des entiers de 0 à n pour implémenter une fonction `somme(n)` renvoyant la somme des $n+1$ premiers entiers.

Méthode récursive Une autre manière d'aborder ce problème est possible si l'on considère la réécriture de cette somme à l'aide de la fonction mathématique $somme(n)$ qui, pour tout entier naturel n , donne la somme des $n + 1$ premiers entiers de la manière suivante :

$$\text{pour tout } n \in \mathbb{N}, \text{ } somme(n) = \begin{cases} 0 & \text{si } n = 0, \\ n + somme(n - 1) & \text{si } n > 0. \end{cases}$$

1. Recopie sur ton carnet de bord la définition mathématique de la fonction $somme$ donnée ci-dessus. Que remarques-tu de particulier dans cette définition ?
2. À partir de cette définition, écris, en développant leur expression, $somme(0)$, $somme(1)$, $somme(2)$ et $somme(3)$.
3. À partir de la définition mathématiques, implémente une nouvelle version de la fonction Python `somme(n)`, cette fois-ci sans utiliser de boucle, ni d'instruction affectation.
Indication : Pour cela il faut distinguer les deux cas de la définition, en se posant pour chacun la question « Que doit retourner la fonction ? »
4. Ajoute un point d'arrêt à la dernière ligne exécutée dans la fonction `somme`, puis exécute ton code en mode *débogage* en appelant `somme(3)`.
5. Observe la pile d'appel du programme¹ (ou *call stack* en anglais) en notant l'ordre des appels successifs de `somme` et la valeur de la variable locale `n`.
6. Sur ton carnet de bord, essaie de représenter cette succession d'appels, de manière similaire à la question 2., en développant chaque appel de fonction par la valeur qu'elle renvoie.
7. Par inadvertance, Gaston a mal recopié la définition mathématique de la fonction $somme(n)$:

$$\text{pour tout } n \in \mathbb{N}, \text{ } somme(n) = \begin{cases} 0 & \text{si } n = 0, \\ n - 1 + somme(n) & \text{si } n > 0. \end{cases}$$

Que se passera-t-il si on implémente en Python la fonction de Gaston ?
Teste-le dans un éditeur Python et observe le résultat.

1. il s'agit de la liste de toutes les fonctions qui sont encore en cours d'appel.