CHAPITRE 2 — Récursivité

« **Auteur inconnu**

Vous ne pouvez pas comprendre la récursivité sans avoir d'abord compris la récursivité. »

1 Récursivité : notion et exemples.

1.1 Quelques exemples

Toutes ces images ont un point commun : ....
....
....
....
....

Définition 1. Un procédé (au sens large) est dit récursif lorsqu'il possède la propriété de faire référence à lui-même à un moment donné lors de la description de ce procédé.

Exemple 1. Certains objets de la vie courante peuvent être définis de manière récursive :

- ....
....
- ....
....

1.2 Application aux algorithmes

Théorème 1. *Toute fonction calculable (au sens intuitif) est récursive (au sens large).*

Exemple 2.
Énoncé du problème. Pour tout entier $n \in \mathbb{N}^*$, on définit le problème $\mathcal{P}(n)$ suivant :

« Calculer la somme des n premiers entiers : $S = 1 + 2 + 3 + \ldots + n$ »

- **Méthode 1.** Il est possible d'écrire une fonction python itérative (avec une boucle), qui prenne en entrée un entier `n` et qui renvoie l'entier `S` associé :

Code python

```python
def somme(n):
    """ int -> int"""
    ................................
    ................................
    ................................
    ................................
```

• **Méthode 2.** Il est possible d'écrire une fonction python *récursive* (faisant référence à elle-même), qui prenne en entrée un entier `n` et qui renvoie l'entier `S` associé :

1. On sait résoudre le problème dans le cas . . . .

. . . .

2. Dans le cas général, on veut résoudre le problème de difficulté n, c'est à dire $\mathcal{P}(n)$: « Calculer la somme des n premiers entiers : $S = 1 + 2 + 3 + \ldots + n$ ».

a. On suppose que l'on connait la réponse S' au problème de difficulté $n - 1$. . . . .

. . . .

b. À l'aide de S', on construit S la solution au problème $\mathcal{P}(n)$. . . . .

. . . .

3. On aboutit donc au code python ci-dessous :

Code python

```python
def somme(n):
    """ int -> int """
    # Cas de base:
    if ............................:
        return .....................
    # Cas général
    else:
        ............................ # appel dit récursif
        return .....................
```

Remarque. On peut faire le lien avec la définition par récurrence de la suite $u_n = \text{somme}(n)$, définie pour tout $n \geq 1$ par :

Méthode : Résolution récursive d'un problème

On considère un problème à résoudre, qui dépend d'un certain paramètre $n \in \mathbb{N}$, que l'on appelle le **rang** (la "difficulté") du problème.
Pour écrire un algorithme de manière récursive, la méthode générale est la suivante :

- **Identifier le cas de base.** Il s'agit de trouver dans quel cas la résolution du problème est "simple" (souvent $n = 0$ ou 1)
- **Résoudre le problème de manière récursive.**
  - *On suppose connue* la solution des problèmes de rang *strictement inférieur* à n.
  - On construit la solution du problème de rang n **à l'aide** de la solution d'un problème de rang inférieur.

Exercice 1. **1.** (u_n) est une suite arithmétique de raison 5 et de premier terme 2. Écrire une fonction python récursive `u` qui étant donné un entier n calcule u_n.

2. (v_n) est une suite arithmétique de raison -2 et de premier terme 8. Écrire une fonction python récursive `v` qui étant donné un entier n calcule v_n.