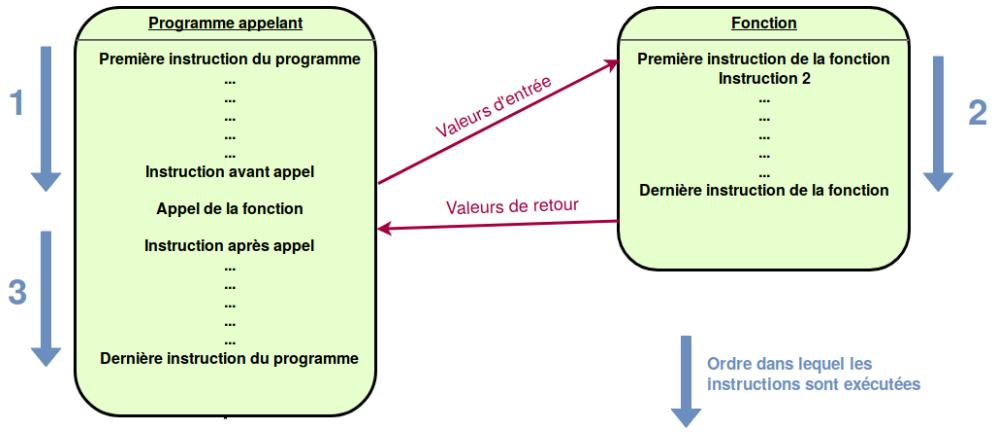
# Chapitre 4 : Fonctions

Une fonction est une séquence d'instructions effectuant une tâche précise (par exemple, un calcul ou un affichage) et est destinée à être utilisée par un autre algorithme.

Lors de l'appel d'une fonction :

- l'exécution du code ou *programme appelant* contenant l'appel de la fonction "se met en pause",
- l'exécution de la fonction commence :
  - les paramètres de la fonction sont initialisés avec les valeurs données par le programme appelant (on parle de *valeur d'entrée*),
  - la séquence d'instructions de la fonction est exécutée,
  - **Optionel :** la fonction retourne des valeurs (on parle de *valeurs de sortie*),
  - les variables locales de la fonction et les paramètres de la fonction sont détruits,
- l'exécution du code contenant l'appel de la fonction reprend, en utilisant si besoin les valeurs retournées par la fonction




### Définition d'une fonction sans paramètres et sans valeur de retour

```
def nom_fonction() :
    # Séquence d'instructions de la fonction
```

- `def` indique que l'on souhaite définir une fonction ;
- `nom_fonction` est le nom de la fonction, il suit les mêmes règles que les noms de variables ;
- les parenthèses ouvrantes et fermantes après le nom de la fonction sont **obligatoires** pour indiquer que l'on définit une fonction ;
- La séquence d'instructions est **indentée** par rapport au mot-clé `def`.

La première ligne est appelée en-tête ou signature de la fonction et la séquence d'instructions corps de la fonction.

```
In [ ]: # Définition de la fonction hello()

        def hello() :
            """Affiche un message de bienvenue"""
            print("Hello world!")


        hello() # Appel de la fonction => affiche Hello world!
```

### Définition d'une fonction avec paramètres et sans valeur de retour

On définit une fonction en précisant le nombre de paramètres, c'est-à-dire **le nombre de valeurs qui doivent être données lors de l'appel de la fonction**. Ces paramètres sont donnés entre les parenthèses dans l'en-tête de la fonction.

```
def nom_fonction(parametre1, parametre2, ..., parametrek) :
    # Séquence d'instructions de la fonction
    # Dans cette séquence, parametre1, parametre2, ..., parametrek
    # sont utilisés pour faire référence à la première, la deuxième, ..., et la kième valeurs
    # données lors de l'appel.
```

- `def` indique que l'on souhaite définir une fonction ;
- `nom_fonction` est le nom de la fonction, il suit les mêmes règles que les noms de variables ;
- `parametre1`, `parametre2`, ..., `parametrek` entre les parenthèses représentent les paramètres de la fonction. Leur nom suit les mêmes règles que les noms des variables.

**Exemple d'utilisation de la fonction** `affiche_age`

```
In [ ]: # Définition de la fonction affiche_perso()
        def affiche_perso(nom) :
            """ Affiche un message personnalisé suivant le nom personne. """
            print("Hello", nom, "!")

        affiche_perso("John") # Appel de la fonction => affiche Hello John!
```

**Mode de transmission des paramètres** Lorsque l'on appelle une fonction contenant des paramètres, on lui donne des **valeurs** d'entrée. Ces valeurs peuvent être des constantes ou des valeurs contenues dans des variables. Dans tous les cas, **les variables que l'on passe lors de l'appel d'une fonction ne sont pas modifiées**.

### Portées de variables

La portée (ou visibilité) d'une variable est l'endroit dans le code où cette variable est accessible. Une variable déclarée dans le corps d'une fonction (ou un paramètre de la fonction) n'est pas accessible en dehors du corps de cette fonction. On parle de portée locale ou variable locale.

**Vocabulaire** Fonction, paramètre, valeurs d'entrée, portée des variables, variables locales.