Les fonctions en Python

Résumé

En cuisine, une recette permet de préparer le même plat chaque fois qu'on en a besoin, sans tout réinventer. En programmation, c'est pareil : une **fonction** est une sorte de recette qui prend des ingrédients (les **paramètres**), effectue une suite d'opérations et produit un résultat. Au lieu de recopier les mêmes lignes de code à chaque fois, on les met dans une fonction qu'on appelle par son nom. C'est la clé pour écrire des programmes clairs, courts et faciles à modifier.

Prérequis : Les types de base.

① Mémo

Définir une fonction

```
def nom_de_la_fonction(parametre1, parametre2):
    instructions
    return resultat
```

- Le mot-clé `def` introduit la définition.
- Les **paramètres** sont les entrées de la fonction.
- Le mot-clé `return` renvoie le résultat.
- Le corps de la fonction est indenté (quatre espaces).

Appeler une fonction

```
resultat = nom_de_la_fonction(valeur1, valeur2)
```

- On passe des valeurs entre parenthèses lors de l'appel.
- La valeur renvoyée par `return` peut être stockée dans une variable.
- Si la fonction ne contient pas de `return`, elle renvoie `None`.

Définitions 1 (Vocabulaire : paramètre formel et paramètre effectif)

Dans `def double(n):`, le `n` entre parenthèses est un **paramètre formel** : c'est le nom symbolique que la fonction utilise pour désigner la valeur qu'elle recevra.

Quand on appelle la fonction avec `double(5)`, la valeur `5` est un **paramètre effectif** : c'est la valeur concrète passée à la fonction au moment de l'appel.

- Les paramètres formels apparaissent dans la **définition**.
- Les paramètres effectifs apparaissent dans l'**appel**.
- On dit simplement « paramètre » quand le contexte ne prête pas à confusion.

Définition 1 (Différence entre `return` et `print`)

- `return` renvoie une valeur : on peut la stocker et la réutiliser.
- `print` affiche à l'écran mais ne renvoie rien (`None`).

Exemple. `print(abs(-5))` fonctionne car `abs` *renvoie* 5. Si `abs` ne faisait qu'afficher, on ne pourrait pas utiliser le résultat.

⚠ Pièges fréquents

- Oublier `return` : la fonction renvoie alors `None`.
- Confondre `return` et `print`.
- Oublier les parenthèses à l'appel : `ma_fonction` désigne la fonction, `ma_fonction()` l'exécute.
- Placer du code après `return` : il ne sera jamais exécuté.

② Exemples

Programme 1 · Aire d'un disque

```
import math

def aire_disque(rayon):
    return math.pi * rayon ** 2

print(aire_disque(5))     # 78.539...
```

Programme 2 · Conversion Celsius-Fahrenheit

```
def celsius_vers_fahrenheit(c):
    return c * 9 / 5 + 32

print(celsius_vers_fahrenheit(100))   # 212.0
print(celsius_vers_fahrenheit(0))     # 32.0
```

Programme 3 · Différence return vs print

```
def double_return(n):
    return n * 2

def double_print(n):
    print(n * 2)

a = double_return(5)   # a vaut 10
b = double_print(5)    # affiche 10, mais b vaut None

print(a + 1)            # 11 (on peut calculer avec a)
# print(b + 1)          # ERREUR : None + 1 impossible
```

③ Exercices

Exercice 1 *Premières fonctions* Écrire les fonctions suivantes :

1. `cube(n)` qui renvoie n^3.
2. `perimetre_rectangle(longueur, largeur)` qui renvoie le périmètre d'un rectangle.

Exercice 2 *Return ou print* Sans exécuter le code, prédire ce qui est affiché par chaque programme :

```
# Programme A
def mystere(x):
    x * 2 + 1

print(mystere(5))

# Programme B
def calcul(a, b):
    return a + b
    print("Terminé")

resultat = calcul(3, 4)
print(resultat)

# Programme C
def afficher_double(n):
    print(n * 2)

x = afficher_double(6)
print(x)
```

Exercice 3 *Conversion de durée*

1. Écrire une fonction `convertir_duree(secondes)` qui renvoie un tuple `(heures, minutes, secondes)` (un *tuple* est un groupe de valeurs entre parenthèses, séparées par des virgules).
2. Tester : vérifier que `convertir_duree(3725)` renvoie `(1, 2, 5)`.

Solutions des exercices

Corrigé de l'exercice 1.

```
def cube(n):
    return n ** 3

def perimetre_rectangle(longueur, largeur):
    return 2 * (longueur + largeur)
```

Vérification :

- `cube(3)` renvoie `27` car $3^3 = 27$;
- `perimetre_rectangle(5, 3)` renvoie `16` car $2 \times (5 + 3) = 16$.

Corrigé de l'exercice 2.

- **Programme A :** affiche `None`. La ligne `x * 2 + 1` calcule mais ne renvoie rien (`return` oublié).
- **Programme B :** affiche `7`. Le `return` interrompt la fonction : `print("Terminé")` n'est jamais exécuté.
- **Programme C :** affiche d'abord `12` (le `print` dans la fonction), puis `None` (car pas de `return`).

Corrigé de l'exercice 3.

```
def convertir_duree(secondes):
    h = secondes // 3600
    m = (secondes % 3600) // 60
    s = secondes % 60
    return (h, m, s)
```

Vérification : `convertir_duree(3725)` donne 3725//3600 = 1, reste 125, 125//60 = 2, reste 5, soit `(1, 2, 5)`.

On vérifie : $1 \times 3600 + 2 \times 60 + 5 = 3725$. ✓