Les fonctions en Python

Introduction

Les **fonctions** sont une des notions les plus importantes en programmation. Elles permettent de structurer votre code, de le rendre plus **réutilisable**, plus **lisible** et plus **organisé**. Une fonction est un bloc de code qui effectue une tâche spécifique et qui peut être appelée autant de fois que nécessaire. Python dispose de nombreuses fonctions intégrées comme print(), input(), ou encore len(), mais vous pouvez également **créer vos propres fonctions**.

Les fonctions sont essentielles pour :

1. **Réduire la duplication de code :** Si vous avez besoin d'exécuter plusieurs fois une même tâche dans votre programme, vous pouvez écrire cette tâche une seule fois dans une fonction, puis l'appeler autant de fois que nécessaire.
2. **Rendre le code plus lisible :** Les fonctions permettent de regrouper un ensemble d'instructions sous un nom clair et descriptif, ce qui améliore la compréhension globale du programme.
3. **Faciliter la maintenance :** Si un problème ou une modification est nécessaire, vous n'avez qu'à modifier la fonction une seule fois, au lieu de chercher toutes les occurrences dans le programme.
4. **Encourager la modularité :** Les fonctions permettent de diviser votre programme en petits blocs autonomes et logiques.

Partie 1. Qu'est-ce qu'une fonction en Python ?

Une **fonction** est un **bloc de code** avec un nom, qui peut être exécuté lorsqu'il est appelé. Une fonction peut :

- Accepter des **données en entrée** (appelées **paramètres**).
- Effectuer un ensemble d'**instructions**.
- **Retourner un résultat** (optionnel).

En Python, une fonction est définie avec le mot-clé def, suivi du nom de la fonction et d'un ensemble d'instructions.

Exemple simple :

```
def saluer():
    print("Bonjour, tout le monde !")

# Appel de la fonction
saluer()
```

Sortie :

```
Bonjour, tout le monde !
```

Partie 2. Structure d’une fonction

Voici la syntaxe générale pour définir une fonction en Python :

```
def nom_de_la_fonction(param1, param2, ...):

    # Instructions de la fonction
    return valeur  # (Optionnel)
```

- **def :** Le mot-clé utilisé pour définir une fonction.
- **nom_de_la_fonction :** Le nom que vous choisissez pour votre fonction. Ce nom doit être descriptif et respecter les règles de nommage en Python (pas d’espaces, pas de caractères spéciaux, etc.).
- **Paramètres (optionnels) :** Les données que vous pouvez transmettre à la fonction.
- **Instructions :** Les opérations que la fonction effectue.
- **return (optionnel) :** La valeur que la fonction renvoie au programme principal.

Partie 3. Créer et utiliser une fonction

3.1 Définir une fonction simple

Une fonction simple est une fonction qui n'accepte **aucun paramètre** et **ne retourne aucune valeur**.

Exemple :

```
def afficher_bienvenue():
    print("Bienvenue dans le programme !")

# Appel de la fonction
afficher_bienvenue()
```

Sortie :

```
Bienvenue dans le programme !
```

3.2 Passer des paramètres à une fonction

Les **paramètres** permettent de transmettre des données à une fonction pour personnaliser son comportement. Vous devez les définir entre parenthèses après le nom de la fonction.

Exemple :

```
def saluer_utilisateur(nom):
    print(f"Bonjour, {nom} !")

# Appel de la fonction avec un paramètre
saluer_utilisateur("Alice")
saluer_utilisateur("Bob")
```

Sortie :

```
Bonjour, Alice !
Bonjour, Bob !
```

3.3 Retourner une valeur avec return

Une fonction **peut retourner une valeur** avec l'instruction return. Cela permet de récupérer un résultat et de l'utiliser dans le programme principal.

Exemple :

```
def ajouter(a, b):
    resultat = a + b
    return resultat

# Appel de la fonction et utilisation du résultat
somme = ajouter(3, 5)
print(f"La somme est : {somme}")
```

Sortie :

```
La somme est : 8
```

Détail :

- **ajouter**(3, 5) exécute la fonction avec a = 3 et b = 5.
- La fonction retourne la valeur 8 grâce à **return**.
- Cette valeur est **stockée** dans la variable somme.

3.4 Paramètres par défaut

Vous pouvez définir des **valeurs par défaut** pour les paramètres. Cela signifie que si vous n'entrez pas de valeur lors de l'appel de la fonction, le paramètre prendra automatiquement sa valeur par défaut.

Exemple :

```
def saluer(nom="inconnu"):
    print(f"Bonjour, {nom} !")

# Appel sans paramètre (valeur par défaut utilisée)
saluer()

# Appel avec un paramètre (la valeur par défaut est écrasée)
saluer("Alice")
```

Sortie :

```
Bonjour, inconnu !
Bonjour, Alice !
```

3.5 Les fonctions avec plusieurs paramètres

Une fonction peut accepter **plusieurs paramètres** pour effectuer des opérations plus complexes.

Exemple :

```
def calculer_volume(longueur, largeur, hauteur):
    return longueur * largeur * hauteur

# Appel de la fonction
volume = calculer_volume(2, 3, 4)
print(f"Le volume est : {volume}")
```

Sortie :

```
Le volume est: 24
```

3.6 Fonction sans return

Si une fonction ne contient **pas l'instruction return**, elle ne renvoie aucune valeur (elle retourne **par défaut None**).

Exemple :

```
def afficher_message():
    print("Ceci est un message.")

resultat = afficher_message()
print(resultat)  # Affiche : None
```

Partie 4. Bonnes pratiques avec les fonctions

1. **Donnez un nom clair à vos fonctions :**
   - Le nom doit indiquer clairement ce que fait la fonction. Par exemple, préférez calculer_volume() à cv().

2. **Évitez les fonctions trop longues :**
   - Une fonction doit idéalement se concentrer sur une seule tâche. Si une fonction est trop longue, divisez-la en plusieurs fonctions plus petites.

3. **Documentez vos fonctions :**
   - Utilisez des commentaires ou des chaînes de documentation (docstrings) pour expliquer ce que fait la fonction.

Exemple :

```
def ajouter(a, b):
    """
    Cette fonction ajoute deux nombres.
    Paramètres :
    - a : premier nombre
    - b : deuxième nombre
    Retourne :
    - La somme de a et b
    """
    return a + b
```

EXERCICES

Exercice 1 : Fonction simple

Créez une fonction appelée **afficher_bienvenue** qui affiche :

```
Bienvenue dans le programme !
```

Appelez cette fonction **trois** fois.

Exercice 2 : Fonction avec un paramètre

Écrivez une fonction appelée **saluer_utilisateur** qui prend un nom en **paramètre** et affiche :

```
Bonjour, [nom] !
```

Appelez cette fonction avec **trois** noms différents.

Exercice 3 : Addition simple

Créez une fonction appelée addition qui :
- Prend **deux** nombres en paramètres.
- **Retourne** la somme de ces deux nombres.

Appelez la fonction avec les nombres 7 et 5, et affichez le résultat.

Exercice 4 : Calcul d'aire

Écrivez une fonction calculer_aire qui :
- Prend deux paramètres : la **longueur** et la **largeur** d'un rectangle.
- **Retourne** l'aire du rectangle.

Appelez la fonction avec les dimensions 5 et 3, et affichez l'aire.

Exercice 5 : Paramètres par défaut

Créez une fonction **presentation** qui :
- Prend deux **paramètres** : un nom et un âge.
- Le paramètre âge doit avoir une valeur **par défaut** de 18.
- Affiche une phrase comme :

```
Bonjour, je m'appelle [nom] et j'ai [âge] ans.
```

Appelez la fonction **avec** et **sans** spécifier l'âge.