Fonctions en python

- Les fonctions en Python suivent le paradigme procédural, où les instructions sont exécutées séquentiellement.
- Un appel excessif de fonctions peut entraîner une "stack overflow", où la liste d'appels de fonctions devient trop grande et provoque un dépassement de mémoire.
- Chaque appel de fonction a un coût en termes de performances.
- La définition d'une fonction se fait avec le mot-clé def, suivi du nom de la fonction et de ses paramètres entre parenthèses.
- La documentation d'une fonction est placée entre triple guillemets ou triple apostrophes juste après la signature de la fonction. Cette documentation est disponible au moment de l'exécution (runtime).
- Une fonction Python retourne toujours quelque chose, soit explicitement avec un return, soit implicitement None si aucune valeur de retour n'est spécifiée.
- L'appel d'une fonction peut se faire avec des arguments positionnels, des arguments par mot-clé ou une combinaison des deux.
- Les valeurs par défaut peuvent être définies pour les paramètres d'une fonction.
- Pour spécifier différents types d'arguments, Python offre des mécanismes tels que les arguments obligatoires, les arguments par mot-clé, ainsi que des paramètres mixtes avec des contraintes positionnelles et par mot-clé.

Exemple de définition de fonction avec des paramètres mixtes :

```
def f(v1=0, v2=0, /, v3=0, *, v4=0, v5=0):
    # Oblige l'utilisation de valeurs positionnelles pour v1, v2 et v3
    # Oblige l'utilisation de valeurs par mot-clé pour v4 et v5
    pass
```

Dans cet exemple, les paramètres v1, v2, et v3 doivent être fournis en tant qu'arguments positionnels, tandis que v4 et v5 doivent être fournis en tant qu'arguments par mot-clé.

Les arguments avec des valeurs positionnelles et des valeurs par mot-clé sont deux méthodes différentes pour passer des arguments à une fonction en Python.

Arguments avec Valeur Positionnelle :

- Les arguments positionnels sont passés à une fonction dans l'ordre dans lequel ils sont définis dans la signature de la fonction.
- L'ordre et le nombre des arguments positionnels doivent correspondre à ceux définis dans la fonction.
- Les valeurs par défaut peuvent être définies pour les paramètres positionnels dans la signature de la fonction.

- L'utilisation d'arguments positionnels est généralement recommandée lorsque l'ordre des arguments est significatif et ne doit pas être changé.

```
def add(x, y):
    return x + y

result = add(3, 5)  # Arguments positionnels
```

Arguments avec Valeur par Mot-Clé :

- Les arguments par mot-clé sont passés à une fonction en spécifiant explicitement le nom du paramètre auquel ils sont destinés, suivis de la valeur à affecter.
- L'ordre des arguments par mot-clé n'est pas important.
- Les arguments par mot-clé sont utiles lorsqu'une fonction a de nombreux paramètres et que certains d'entre eux ont des valeurs par défaut.
- Ils rendent le code plus lisible et moins sujet à des erreurs d'ordre d'arguments.

```
def greet(name, message="Hello"):
    return f"{message}, {name}"

greet(message="Hi", name="John")  # Arguments par mot-clé
```

En résumé, les arguments positionnels et les arguments par mot-clé offrent une flexibilité dans la façon dont les arguments sont passés à une fonction, permettant aux développeurs de choisir la méthode qui convient le mieux à leur code en fonction de la clarté et de la flexibilité nécessaires.

Arguments valeurs mixtes :

Voici un exemple d'une fonction qui utilise à la fois des arguments positionnels et des arguments par mot-clé :

```
def describe_person(name, age, city="Unknown"):
    """
    Cette fonction décrit une personne en utilisant son nom, son âge et
éventuellement sa ville.
    :param name: Le nom de la personne.
    :param age: L'âge de la personne.
    :param city: La ville de résidence de la personne (par défaut "Unknown").
    :return: Une chaîne de caractères décrivant la personne.
    """
    description = f"{name} is {age} years old."
    if city != "Unknown":
        description += f" They live in {city}."
```

```
    return description

# Utilisation de la fonction avec des arguments positionnels et des arguments par
mot-clé
print(describe_person("Alice", 30))  # Utilisation d'arguments positionnels
print(describe_person("Bob", 25, city="New York"))  # Utilisation d'un argument
positionnel et d'un argument par mot-clé
```

Dans cet exemple :

- La fonction describe_person prend trois paramètres : name, age et city (avec une valeur par défaut "Unknown").
- L'appel de la fonction avec describe_person("Alice", 30) utilise des arguments positionnels pour fournir le nom et l'âge de la personne. La valeur par défaut de city est utilisée.
- L'appel de la fonction avec describe_person("Bob", 25, city="New York") utilise un argument positionnel pour le nom, un argument positionnel pour l'âge et un argument par mot-clé pour la ville. Cela permet de spécifier la ville sans se soucier de l'ordre des paramètres.

- `v1=0`, `v2=0`, `/`, `v3=0`: Ces paramètres sont des arguments positionnels. Ils doivent être passés dans cet ordre lors de l'appel de la fonction. Le slash (/) sépare les arguments positionnels des arguments par mot-clé. Ainsi, tous les paramètres avant le slash doivent être positionnels.
- `*`: Il s'agit de la marqueur de séparation entre les arguments positionnels et les arguments par mot-clé. Tout ce qui suit le * doit être spécifié en tant qu'arguments par mot-clé lors de l'appel de la fonction.
- `v4=0`,`v5=0`: Ce sont des arguments par mot-clé. Ils ont une valeur par défaut, ce qui signifie qu'ils peuvent être omis lors de l'appel de la fonction.

```
def f(v1=0, v2=0, /, v3=0, *, v4=0, v5=0):
    """
    Exemple de fonction avec différents types de paramètres.
    :param v1: Premier paramètre positionnel.
    :param v2: Deuxième paramètre positionnel.
    :param v3: Troisième paramètre positionnel.
    :param v4: Premier paramètre par mot-clé.
    :param v5: Deuxième paramètre par mot-clé.
    :return: Aucun
    """
    print(f"v1={v1}, v2={v2}, v3={v3}, v4={v4}, v5={v5}")

# Utilisation de la fonction avec différents types d'arguments
f(1, 2, 3, v4=4, v5=5)  # Arguments positionnels et arguments par mot-clé
```

Dans cet exemple :

- Les trois premiers paramètres (`v1`, `v2`, `v3`) sont des arguments positionnels.
- `v4` et `v5` sont des arguments par mot-clé, mais ils doivent être spécifiés en tant que tels lors de l'appel de la fonction.
- L'utilisation du slash (`/`) sépare les arguments positionnels des arguments par mot-clé.