ALGORITHMIQUE · FICHE PRATIQUE

Cheatsheet : Définir, appeler et comprendre une fonction

Paramètres, arguments, retour, portée et effets sur les données.

Par Ali ZAINOUL · Cours gratuit

Une fonction regroupe un traitement sous un nom. Sa définition décrit ses instructions ; son appel demande de les exécuter. Le résultat retourné est une valeur que l'appelant peut réutiliser : ce n'est pas la même chose qu'un affichage.

Cette fiche présente une fonction à plusieurs entrées, un appel complet puis un exemple d'effet sur des données. Les valeurs de chaque exemple sont fixées pour pouvoir vérifier le résultat à la main.

1. Paramètre, argument, corps et retour

Le paramètre est le nom écrit dans la définition. L'argument est la valeur transmise lors d'un appel. Avec compute_total(250, 3, 100), les trois valeurs deviennent respectivement price, quantity et delivery. L'ordre des arguments positionnels compte.

Le corps contient les instructions. L'appelant est la partie du programme qui demande l'exécution. Un retour termine l'appel et lui fournit le résultat. Une fonction qui affiche un nombre ne fournit pas automatiquement ce nombre comme résultat.

Relier les mots à l'exemple

Terme	Exemple	Rôle
Définition	compute_total(price, quantity, delivery)	Annonce la règle et les trois paramètres
Appel	compute_total(250, 3, 100)	Fournit les arguments dans cet ordre

Terme	Exemple	Rôle
Corps	price × quantity + delivery	Calcule un total en centimes
Retour	850	Valeur disponible pour l’appelant

2. Une fonction complète et son utilisation

Le contrat reçoit un prix unitaire en centimes, une quantité entière non négative et des frais en centimes. Les montants sont non négatifs. Le résultat est price × quantity + delivery. Cette règle ajoute les frais même lorsque la quantité vaut zéro : ne lui attribuez pas une politique commerciale implicite.

En Python, return fournit directement le résultat. En MATLAB, le nom de sortie est déclaré à gauche du signe égal dans la ligne function ; le corps affecte ce nom. Enregistrez la fonction MATLAB dans compute_total.m, puis le script d'appel dans use_compute_total.m, dans le même dossier.

De l’appel au résultat

Moment	Valeur connue
Entrée dans l’appel	price = 250 ; quantity = 3 ; delivery = 100
Produit	price × quantity = 750
Ajout des frais	750 + 100 = 850
Retour	total reçoit 850 chez l’appelant

Calculer une commande avec trois arguments

250 centimes × 3 articles + 100 centimes de frais doivent produire 850 centimes.

Pseudo-code

```
// example.pseudo
// compute_total
FUNCTION compute_total(price, quantity, delivery)
    RETURN price * quantity + delivery
END FUNCTION
total <- compute_total(250, 3, 100)
PRINT total
```

Python

```python
# compute_total.py
def compute_total(price, quantity, delivery):
    return price * quantity + delivery

total = compute_total(250, 3, 100)
print(total)
```

`def` définit la fonction. `return` transmet le nombre ; `print` l'affiche ensuite, à l'extérieur de la fonction.

MATLAB

```
% compute_total.m
function total = compute_total(price, quantity, delivery)
    total = price * quantity + delivery;
end
```

`function` déclare les entrées et le nom de sortie total. `end` ferme la définition.

MATLAB - appel

```
% use_compute_total.m
total = compute_total(250, 3, 100);
disp(total);
```

`disp` affiche le résultat reçu. Un appel sans affectation ne remplit pas automatiquement une variable total dans un autre script.

> **Résultat expliqué.** L'appelant reçoit 850. Pour compute_total(250, 0, 100), la même règle donne 100, et non zéro.

3. Portée locale, alias et modification

La portée d'un nom est la région où il peut être utilisé. Un nom local à une fonction appartient à son exécution. Mais un nom local peut désigner une donnée partagée : la localité du nom ne protège pas automatiquement le contenu.

Une liste Python est mutable : ses éléments peuvent être remplacés. Après alias = original, les deux noms désignent la même liste, pas deux copies. Dans l'exemple numérique MATLAB, l'appelant conserve la valeur de son vecteur ; le résultat modifié lui est retourné séparément.

La fonction suivante exige au moins un élément. Elle remplace le premier et retourne la suite obtenue. Le pseudo-code annonce la modification logique ; les versions précisent les effets observés chez l'appelant.

Remplacer le premier élément

L’entrée est [1, 2], la nouvelle première valeur est 9. L’effet sur les noms de l’appelant est différent selon le langage.

Pseudo-code

```
// example.pseudo
// replace_first
// Requires at least one element.
FUNCTION replace_first(values, replacement)
    values[0] <- replacement
    RETURN values
END FUNCTION
```

Python

```python
# replace_first.py
def replace_first(values, replacement):
    values[0] = replacement
    return values

original = [1, 2]
alias = original
updated = replace_first(original, 9)
print(original, alias, updated)
```

`return` transmet ici la liste déjà modifiée. Une copie distincte demanderait une opération supplémentaire.

MATLAB

```
% replace_first.m
function values = replace_first(values, replacement)
    values(1) = replacement;
end
```

MATLAB - appel

```
% use_replace_first.m
original = [1, 2];
alias = original;
updated = replace_first(original, 9);
disp(original);
disp(alias);
disp(updated);
```

`disp` montre [1 2], puis [1 2], puis [9 2]. Cette comparaison concerne des vecteurs numériques, pas tous les types d'objets MATLAB.

> **Résultat expliqué.** Python affiche trois listes [9, 2] car les noms désignent la même liste. MATLAB conserve [1 2] dans original et alias et donne [9 2] dans updated.

4. Choisir une interface compréhensible

Une interface de fonction précise les données reçues, le résultat et les effets autorisés. Distinguez modifier une donnée existante et réaffecter un nom à une nouvelle valeur. Une réaffectation locale ne remplace pas à elle seule le nom de l'appelant.

Un affichage aide à observer, mais mélange l'usage de l'écran avec le calcul. Pour réutiliser une règle dans un autre traitement, retournez d'abord la valeur puis affichez-la seulement là où c'est nécessaire.

> **Erreur à éviter.** En Python, une fonction sans return explicite renvoie None. En MATLAB, return quitte le script ou la fonction ; il ne s'écrit pas return suivi du résultat.

▼ Que vaut compute_total(100, 0, 50) ? Peut-on remplacer les arguments par (100, 50, 0) sans changer le problème ?

Le premier appel renvoie 50 centimes. Le second renvoie 5 000 centimes : il demande 50 articles sans frais. Les arguments sont associés aux paramètres par leur position.

Télécharger cette fiche en PDF · **Toutes les fiches par thème**

Version web : https://www.clearcode.fr/courses/programming/algorithmic/cheatsheet/functions