ALGORITHMIQUE · CHAPITRE 2 · FONDATIONS

Valeurs, variables et affectation

Distinguez valeurs, types, variables et constantes. Suivez une affectation et un échange de valeurs avec des traces en pseudo-code, Python et MATLAB.

Par Ali ZAINOUL · Cours gratuit

Ce que vous allez apprendre

- Distinguer valeur, type, variable et constante nommée.
- Lire une affectation sans la confondre avec une égalité.
- Échanger deux valeurs et expliquer chaque état intermédiaire.

Prérequis. Avoir identifié les entrées et sorties d'un petit calcul.

Pourquoi cet échange échoue-t-il ?

Vous voulez échanger les valeurs de deux variables. Si `left` vaut 4 et `right` vaut 9, écrire d'abord "left reçoit right", puis "right reçoit left" ne suffit pas. Après la première instruction, la valeur 4 n'est plus accessible par `left`. La seconde relit donc 9.

Valeur, type et état

Une **valeur** peut être un nombre, un texte ou un booléen vrai/faux. Une **variable** permet de désigner une valeur par un nom. L'**état** du calcul est l'ensemble des valeurs utiles à un instant donné. L'affectation `left ← right` signifie : évaluer ce qui est à droite, puis associer le résultat au nom de gauche. Ce n'est pas une égalité mathématique à résoudre.

Observer le remplacement

Trace d'un échange incorrect

Instruction exécutée	left	right
État initial	4	9
left ← right	9	9
right ← left	9	9

Le tableau décrit l'état **après** chaque instruction. Cette convention de lecture sera conservée dans le cours. Nous n'avons pas échangé les valeurs : nous avons copié la même valeur dans les deux variables.

Distinguer une variable, un type et une constante

Un **type** indique la nature d'une valeur et les opérations qui lui conviennent. Le nombre entier `4`, le texte `"4"` et le booléen `True` ne représentent pas la même chose. Un **booléen** ne possède que deux valeurs logiques : vrai ou faux. Nous n'en faisons pas encore une décision ; le chapitre sur la logique les combinera dans des conditions.

Une **constante** est une valeur que l'on décide de ne pas modifier pendant le traitement. La conversion d'une durée utilise toujours 60 secondes par minute. Nommer cette règle `SECONDS_PER_MINUTE` explique son rôle, tandis que `total_seconds` peut changer d'un essai à l'autre. Écrire un nom en majuscules est ici une **convention de nommage**, c'est-à-dire une règle de lisibilité partagée, pas une protection automatique du langage.

Python ne rend pas un nom impossible à réaffecter parce qu'il est écrit en majuscules. Dans les scripts MATLAB montrés ici, une affectation ordinaire reste également modifiable. Le pseudo-code utilise `CONSTANT` pour annoncer notre intention ; Python et MATLAB reposent ici sur la discipline du programmeur. Cette précision évite de confondre une règle du problème avec une garantie technique inexistante.

Pseudo-code

```
// example.pseudo
// convert_minutes
CONSTANT SECONDS_PER_MINUTE <- 60
total_seconds <- 125
minutes <- floor_divide(total_seconds, SECONDS_PER_MINUTE)
PRINT minutes
```

Python

```python
# convert_minutes.py
SECONDS_PER_MINUTE = 60
total_seconds = 125
minutes = total_seconds // SECONDS_PER_MINUTE
print(minutes)  # 2
```

MATLAB

```
% convert_minutes.m
SECONDS_PER_MINUTE = 60;
total_seconds = 125;
minutes = floor(total_seconds / SECONDS_PER_MINUTE);
disp(minutes); % 2
```

Avec 125, le résultat vaut 2 minutes ; avec 180, il vaut 3 minutes. Dans les deux essais, `SECONDS_PER_MINUTE` reste égal à 60. Une constante nommée réduit le risque de modifier une occurrence de 60 en oubliant une autre. Pour les détails de la convention Python, voir les **règles de nommage des constantes de la PEP 8**.

Conserver une valeur temporaire

Pour déplacer deux objets, on peut poser provisoirement le premier dans un emplacement libre. La variable `temporary` joue ce rôle : elle conserve 4 avant son remplacement. Ici, nous choisissons des nombres simples ; les références vers des objets partagés seront abordées

avec la portée et l'état.

Pseudo-code

```
// example.pseudo
// swap_values
left <- 4
right <- 9
temporary <- left
left <- right
right <- temporary
PRINT left, right
```

Python

```python
# swap_values.py
left = 4
right = 9
temporary = left
left = right
right = temporary
print(left, right)  # 9 4
```

MATLAB

```
% swap_values.m
left = 4;
right = 9;
temporary = left;
left = right;
right = temporary;
disp([left, right]); % [9 4]
```

Construire la bonne trace

Trace de l'échange avec une variable temporaire

Instruction	temporary	left	right
État initial	Non définie	4	9
temporary ← left	4	4	9
left ← right	4	9	9
right ← temporary	4	9	4

Refaites la trace avec 4 et 4 : les deux valeurs restent égales. La méthode n'a pas besoin de savoir si elles sont différentes. Puis remplacez les nombres par deux textes et expliquez pourquoi les étapes restent compréhensibles.

Lire avant d’exécuter

▼ Après x ← 2 puis x ← x + 3, que vaut x ?

La seconde instruction lit d'abord l'ancienne valeur 2, calcule 5, puis remplace la valeur associée à x. L'état final est x = 5. Il n'y a aucune contradiction : le symbole d'affectation décrit un changement dans le temps.

▼ Peut-on supprimer temporary après left ← right ?

Pas avant d'avoir réutilisé sa valeur pour affecter right. Elle est précisément le seul nom qui conserve encore l'ancienne valeur de left dans cet exemple.

Garder une convention de noms

Les identifiants `left`, `right` et `temporary` ont le même rôle dans les trois écritures. Le pseudo-code utilise une flèche pour séparer visuellement affectation et comparaison. Dans les langages, ce sont les instructions et le contexte qui donnent leur sens aux symboles.

À retenir - fiche mémo

Variable

Un nom associé à une valeur ; une affectation peut changer cette association.

Constante

Une valeur que le programme choisit de ne pas modifier, comme 60 secondes par minute.

Échange

Une variable temporaire conserve la première valeur avant son remplacement.

Pseudo-code

```
// example.pseudo
// swap_values
temporary <- left
left <- right
right <- temporary
```

Point de vigilance. Les majuscules ne rendent pas un nom automatiquement immuable en Python ou dans nos scripts MATLAB.

Consulter la cheatsheet : Fondations · **Télécharger la fiche du thème en PDF**

Version web : https://www.clearcode.fr/courses/programming/algorithmic/foundations/values-and-variables