ALGORITHMIQUE · CHAPITRE 5 · FONDATIONS

Expressions, comparaisons et logique

Construisez des expressions et conditions : priorités, égalité, comparaisons et logique booléenne, avec exemples Python, MATLAB et tables de vérité.

Par Ali ZAINOUL · Cours gratuit

Ce que vous allez apprendre

- Évaluer une expression en respectant ses priorités.
- Distinguer affectation, égalité et comparaison.
- Combiner des propositions avec ET, OU et NON.

Prérequis. Savoir suivre quelques affectations de variables.

Que signifie "entre deux bornes" ?

Une température est considérée dans la plage attendue si elle est comprise entre 18 et 24 degrés, bornes incluses. À 18 exactement, la réponse doit donc être vraie. Le mot "entre" ne suffit pas à décider si les frontières sont incluses : il faut l'écrire.

Calculer et comparer

Une **expression** produit une valeur. `2 + 3 × 4` vaut 14 ; `(2 + 3) × 4` vaut 20. Une **comparaison** produit un booléen : `temperature ≥ 18` peut être vraie ou fausse. L'affectation enregistre ensuite ce résultat dans une variable. En Python et MATLAB, `==` compare des valeurs et `=` sert à affecter.

Lire les opérateurs avant de les combiner

Un **opérateur** décrit une opération sur des valeurs : `+` additionne, `*` multiplie et `/` divise dans les blocs de code. Un **opérande** est une valeur sur laquelle l'opérateur agit. Dans `2 + 3`, 2 et 3 sont les opérandes. Les parenthèses regroupent le calcul à effectuer en priorité.

Les comparaisons `<`, `>`, `<=` et `>=` signifient respectivement inférieur, supérieur, inférieur ou égal, supérieur ou égal. `==` teste l'égalité ; `!=` en Python et `~=` en MATLAB testent la différence. Notre pseudo-code écrit aussi `==` pour comparer et `<-` pour affecter. On lit donc `in_range <- temperature >= 18` en deux temps : comparer, puis conserver la réponse.

Scalaire signifie ici une seule valeur, par opposition à une suite de valeurs. Les conditions portent sur une seule température. En Python, `and`, `or` et `not` correspondent à ET, OU et NON. Pour ces booléens scalaires MATLAB, les écritures sont `&&`, `||` et `~`. Le pseudo-code utilise `AND`, `OR` et `NOT`. Ces correspondances ne signifient pas que les opérations sur des tableaux complets suivent toutes les mêmes règles.

Une borne constitue une frontière à vérifier, pas seulement un nombre présent dans le programme. Pour l'intervalle de 18 à 24 degrés Celsius, commencez par 17, 18, 24 et 25. Le test correct donne faux, vrai, vrai et faux. Écrire les résultats attendus avant l'exécution permet de repérer une confusion entre ET et OU.

Combiner deux propositions

Table de vérité de ET et OU

A	B	A ET B	A OU B
faux	faux	faux	faux
faux	vrai	faux	vrai
vrai	faux	faux	vrai
vrai	vrai	vrai	vrai

ET exige que les deux propositions soient vraies. **OU** accepte qu'au moins l'une soit vraie, y compris les deux. **NON** inverse vrai et faux. Pour notre intervalle, la température doit vérifier la borne basse *et* la borne haute.

Regrouper et évaluer une condition

En Python, le calcul arithmétique précède les comparaisons, puis viennent `not`, `and` et `or`. En MATLAB, `~` a une priorité supérieure aux comparaisons ; `&&` précède `||`. Notre pseudo-code regroupe explicitement les comparaisons, puis utilise NOT, AND, OR dans cet ordre de priorité. Pour nier un test dans les deux langages, écrivez donc `not (x == 0)` en Python et `~(x == 0)` en MATLAB.

Le **court-circuit** consiste à ne pas évaluer la seconde partie quand la première suffit à déterminer la réponse : ET s'arrête sur faux, OU sur vrai. Avec des nombres finis, `(denominator != 0) and (numerator / denominator > 2)` en Python protège ainsi la division par zéro. MATLAB écrit `(denominator ~= 0) && (numerator / denominator > 2)`. Placez la protection avant la division. Dans ce cours, AND et OR du pseudo-code suivent aussi cette convention.

Ces exemples emploient des booléens scalaires. En Python, `and` et `or` peuvent aussi renvoyer un de leurs opérandes lorsque ceux-ci ne sont pas des booléens. Ils ne doivent pas être confondus avec les opérateurs sur les bits. Pour plusieurs tests, `(x >= 18) and (x <= 24)` et sa traduction MATLAB restent plus explicites qu'une comparaison chaînée : `18 <= x <= 24` n'a pas le même sens dans les deux langages.

Traduire la même condition

Pseudo-code

```
// example.pseudo
// check_range
temperature <- 21
in_range <- (temperature >= 18) AND (temperature <= 24)
PRINT in_range
```

Python

```python
# check_range.py
temperature = 21
in_range = (temperature >= 18) and (temperature <= 24)
print(in_range)  # True
```

MATLAB

```matlab
% check_range.m
temperature = 21;
in_range = (temperature >= 18) && (temperature <= 24);
disp(in_range); % 1 : true
```

Les comparaisons portent ici sur une seule température. MATLAB utilise `&&` pour ce ET logique sur des conditions scalaires. Python utilise `and`. On écrira les deux comparaisons explicitement, plutôt que de traduire une comparaison chaînée Python directement en MATLAB.

Essayer les frontières

Vérifier une condition d'intervalle

Température	≥ 18	≤ 24	Dans la plage ?
17	faux	vrai	faux
18	vrai	vrai	vrai
24	vrai	vrai	vrai
25	vrai	faux	faux

Tester seulement 21 aurait masqué une erreur sur les bornes. Les valeurs juste avant, exactement sur et juste après une frontière révèlent souvent les ambiguïtés d'une condition.

Construire et nier une condition

▼ Pourquoi temperature ≥ 18 OU temperature ≤ 24 ne convient-il pas ?

Avec ces bornes, chaque température réelle vérifie au moins l'une des deux propositions : 10 vérifie la seconde et 30 la première. Cette condition n'impose donc pas de rester dans l'intervalle.

▼ Comment décrire une température hors de la plage ?

Elle est strictement inférieure à 18 OU strictement supérieure à 24. On peut aussi écrire NON(dans la plage). Les bornes restent exclues de cette condition, puisqu'elles appartiennent à la plage.

Du booléen à la décision

Une condition produit une réponse vrai/faux ; elle ne choisit pas encore une action à elle seule. Le chapitre sur les conditions utilisera ce résultat pour sélectionner un chemin d'exécution. Avant cela, nous étudions la représentation des nombres, les conversions et les collections, puis fixons une écriture de pseudo-code lisible.

À retenir - fiche mémo

Expression

Elle calcule une valeur ; une comparaison calcule une réponse vraie ou fausse.

AND / OR / NOT

ET exige les deux réponses vraies ; OU au moins une ; NON inverse la réponse.

Frontières

Testez juste avant, sur et juste après une borne.

Pseudo-code

```
// example.pseudo
// check_range
in_range <- (temperature >= 18) AND (temperature <= 24)
```

Point de vigilance. Affectation et comparaison sont différentes : <- affecte ; == compare dans notre pseudo-code.

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

Version web : https://www.clearcode.fr/courses/programming/algorithmic/foundations/expressions-and-logic