ALGORITHMIQUE · FICHE PRATIQUE

Cheatsheet : Choisir et répéter des instructions

Conditions, seuils, boucles, accumulateurs, indices et parcours imbriqués.

Par Ali ZAINOUL · Cours gratuit

Une condition choisit un chemin ; une boucle répète un groupe d'instructions. Cette fiche explique les mots-clés et montre les valeurs effectivement parcourues, pas seulement des formes à recopier.

Un bloc est un groupe d'instructions. Python le délimite par l'indentation, les espaces au début des lignes. MATLAB utilise les mots-clés de contrôle et end. Dans le pseudo-code de cette fiche, chaque bloc est fermé explicitement.

1. Un choix exclusif avec plusieurs seuils

IF teste une condition ; THEN introduit la branche correspondante. ELSE IF ajoute un test lorsque les précédents sont faux. ELSE représente le cas restant. Dans une chaîne, seule la première branche vraie est retenue ; deux if indépendants ne possèdent pas cette exclusivité.

Le score est un entier de 0 à 20. La règle est : 16 à 20 → Excellent ; 10 à 15 → Passed ; 0 à 9 → Try again. Les données hors de cet intervalle ne sont pas validées par ce petit script.

Frontières du classement

Score	Premier test	Second test	Message
9	Faux	Faux	Try again
10	Faux	Vrai	Passed
15	Faux	Vrai	Passed
16	Vrai	Non évalué	Excellent

Classer un score de 16

Les seuils sont testés du plus exigeant au moins exigeant.

Pseudo-code

```
// example.pseudo
// classify_score
score <- 16
IF score >= 16 THEN
    message <- "Excellent"
ELSE IF score >= 10 THEN
    message <- "Passed"
ELSE
    message <- "Try again"
END IF
PRINT message
```

Python

```python
# classify_score.py
score = 16
if score >= 16:
    message = "Excellent"
elif score >= 10:
    message = "Passed"
else:
    message = "Try again"
print(message)
```

`elif` n'est évalué que si le `if` précédent n'a pas retenu de branche. `else` ne possède pas son propre test.

MATLAB

```
% classify_score.m
score = 16;
if score >= 16
    message = 'Excellent';
elseif score >= 10
    message = 'Passed';
else
    message = 'Try again';
end
disp(message);
```

`elseif` ajoute le second seuil et `end` ferme la chaîne. `disp` affiche le message retenu.

> **Résultat expliqué.** Le premier test est vrai : Excellent est choisi sans évaluer le seuil de 10.

2. WHILE : initialiser, tester, exécuter, mettre à jour

Une itération est un passage dans le corps de boucle. WHILE teste avant chaque itération. Si la condition est fausse dès le départ, le corps n'est jamais exécuté. L'état doit progresser vers la sortie : oublier la mise à jour peut produire une boucle infinie.

Dans le décompte, remaining est un entier non négatif. Il est affiché puis diminué de 1. La suite des tests reçoit 3, 2, 1 puis 0 : trois tours, mais quatre tests.

Les quatre évaluations du test

Valeur avant test	remaining > 0	Action
3	Vrai	Afficher 3 ; passer à 2
2	Vrai	Afficher 2 ; passer à 1
1	Vrai	Afficher 1 ; passer à 0

Valeur avant test	remaining > 0	Action
0	Faux	Quitter le corps

Un décompte qui atteint zéro

La sortie normale est 3, 2, 1 puis Done. Une entrée de 0 n’affiche que Done.

Pseudo-code

```
// example.pseudo
// countdown
remaining <- 3
WHILE remaining > 0 DO
    PRINT remaining
    remaining <- remaining - 1
END WHILE
PRINT "Done"
```

Python

```python
# countdown.py
remaining = 3
while remaining > 0:
    print(remaining)
    remaining = remaining - 1
print("Done")
```

`while` refait le test après la soustraction. Le dernier `print` se situe après la boucle.

MATLAB

```
% countdown.m
remaining = 3;
while remaining > 0
    disp(remaining);
    remaining = remaining - 1;
end
disp('Done');
```

`while` reprend le test avec la nouvelle valeur ; `end` n’initialise pas remaining de nouveau.

> **Résultat expliqué.** Les valeurs affichées sont 3, 2 et 1 ; l'état final est remaining = 0, puis Done est affiché.

3. FOR et accumulateur : additionner un parcours

Un accumulateur garde un résultat partiel. Pour une somme, total commence à zéro avant la boucle. Le pseudo-code parcourt de 1 à n, bornes incluses. Python utilise range(1, n + 1), car sa borne finale est exclue ; MATLAB utilise ici 1:n avec un pas de 1.

Pour n = 4, les valeurs parcourues sont 1, 2, 3 et 4. Les totaux successifs sont 1, 3, 6 et 10. Pour n = 0, le parcours est vide et le total reste zéro.

Additionner les entiers de 1 à 4

Le contrat porte sur un entier n non négatif.

Pseudo-code

```
// example.pseudo
// sum_integers
n <- 4
total <- 0
FOR value FROM 1 TO n INCLUSIVE DO
    total <- total + value
END FOR
PRINT total
```

Python

```python
# sum_integers.py
n = 4
total = 0
for value in range(1, n + 1):
    total = total + value
print(total)
```

`for` parcourt les éléments fournis par `range`. Le nom value reçoit successivement 1, 2, 3 puis 4.

MATLAB

```
% sum_integers.m
n = 4;
total = 0;
for value = 1:n
    total = total + value;
end
disp(total);
```

`for` parcourt ici un vecteur ligne ; chaque tour fournit une valeur. `end` ferme le corps répété.

> **Résultat expliqué.** Le total final vaut 10. Réinitialiser total dans la boucle aurait détruit l’accumulation.

4. Indices, imbrication et arrêts anticipés

Un indice est un numéro de position. Une liste Python de trois éléments utilise les indices 0, 1 et 2 ; un vecteur MATLAB de trois éléments utilise 1, 2 et 3. Une longueur est un nombre d'éléments, pas nécessairement le dernier indice.

Une boucle imbriquée se trouve dans le corps d'une autre. Pour chaque ligne d'une grille, la boucle intérieure reprend toutes les colonnes. Deux lignes de trois colonnes donnent les couples (1,1), (1,2), (1,3), (2,1), (2,2), (2,3) : six visites.

BREAK quitte la boucle courante ; CONTINUE passe au prochain tour sans exécuter le reste du tour courant. Ces instructions ne remplacent pas une règle d'arrêt correctement annoncée.

> **Erreur à éviter.** Une suite vide n'a pas de premier élément. Traitez ce cas avant de lire values[0] en Python ou values(1) en MATLAB.

▼ Pourquoi commencer le classement par score >= 10 ne permet-il plus d’atteindre Excellent pour 18 ?

18 satisfait déjà le premier seuil. La chaîne choisit cette première branche et ignore les suivantes. Testez 16 avant 10, ou annoncez des conditions d’intervalles explicites.

Télécharger cette fiche en PDF · Toutes les fiches par thème

Version web : https://www.clearcode.fr/courses/programming/algorithmic/cheatsheet/control-flow