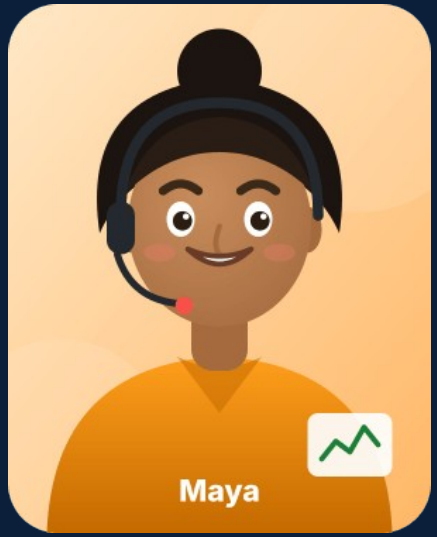
HEF-010 · MISSION 02 / 10

Variables, conditions et boucles

Trois briques suffisent pour écrire presque tous les algorithmes : mémoriser, décider, répéter.



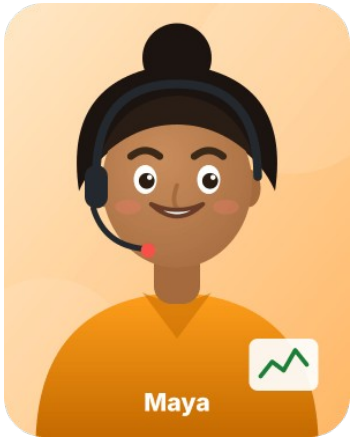


Badge : Chef d'orchestre

70 min · débutant

L'histoire

Kodi doit calculer la **moyenne des notes de quiz** d'un groupe de membres. Il écrit trois lignes… et le programme affiche n'importe quoi quand la liste est vide.



« Il manque une brique, dit Sam. Tu sais mémoriser et répéter. Mais as-tu pensé à décider ? »

Variables, conditions, boucles : les trois briques de base.

Ta mission

À la fin, je serai capable de…

1. utiliser une variable pour garder un état
2. écrire une condition avec si / sinon
3. choisir entre une boucle `for` et une boucle `while`
4. éviter une boucle infinie
5. suivre l'évolution des variables dans une trace

Défi d'échauffement

Défi d'échauffement

Tu as 10 pièces de 1 €, 5 de 2 € et 2 billets de 5 €. **Comment calculerais-tu le total sans le faire de tête ?** Écris les étapes. Quelle information gardes-tu en mémoire pendant que tu comptes ?

Le concept

La variable : une boîte étiquetée

Une **variable** est un nom qui désigne une valeur gardée en mémoire. Elle peut changer.

```
total = 0
total = total + 5   # total vaut maintenant 5
```

La ligne `total = total + 5` se lit : *calcule total + 5, puis mets le résultat dans total*. Ce n'est pas une égalité mathématique.

La condition : choisir

Le concept

La condition : choisir (suite)

```
note = 14
if note >= 10:
    print("validé")
else:
    print("à revoir")
```

On peut enchaîner avec `elif`. Le programme prend **un seul** chemin.

La boucle : répéter

- `for` : quand on sait quoi parcourir (une liste, une plage).
- `while` : quand on répète **tant qu'une condition est vraie**.

Le concept

La boucle : répéter (suite)

```
i = 0
while i < 3:
    print(i)
    i = i + 1   # sans cette ligne, la boucle ne s'arrête jamais !
```

⚠ ***Boucle infinie*** *: si la condition ne devient jamais fausse, le programme ne s'arrête pas. Vérifie toujours : qu'est-ce qui fait avancer la boucle vers sa fin ?*

L'état

L'**état** est l'ensemble des valeurs des variables à un instant donné. Suivre un algorithme, c'est suivre l'évolution de son état.

Exemple pas à pas

Problème : moyenne d'une liste de notes, sans planter si elle est vide.

```
def moyenne(notes):
    if len(notes) == 0:
        return None          # on décide : pas de moyenne
    total = 0
    for n in notes:
        total = total + n
    return total / len(notes)

print(moyenne([12, 8, 15]))
print(moyenne([]))
```

Trace pour `[12, 8, 15]` :

Tour	n	total
départ	:	0

Exemple pas à pas

Exemple pas à pas (suite)

Tour	n	total
1	12	12
2	8	20
3	15	35

Fin : `35 / 3 ≈ 11,67`.

Ce qu'il faut remarquer : la condition protège la division par zéro. Sans elle, `moyenne([])` planterait.

À toi, avec aide

Écris une fonction `compter_pairs(liste)` qui renvoie combien de nombres pairs contient la liste. Teste avec `[1, 2, 3, 4, 6]` : le résultat doit être 3.

Un nombre est pair si `n % 2 == 0` (le reste de la division par 2 est 0).

À toi, avec aide · réponse

Indices progressifs

Indice 1

Il te faut un compteur qui démarre à 0.

Indice 2

Dans une boucle `for`, teste la condition `n % 2 == 0`.

Indice 3

Quand la condition est vraie, ajoute 1 au compteur. Renvoie-le après la boucle.

À toi, avec aide · réponse

Réponse

```
def compter_pairs(liste):
    compteur = 0
    for n in liste:
        if n % 2 == 0:
            compteur = compteur + 1
    return compteur

print(compter_pairs([1, 2, 3, 4, 6]))  # 3
```

À toi, en solo

Écris `premier_negatif(liste)` qui renvoie la **position** (index) du premier nombre négatif, ou `-1` s'il n'y en a pas. Utilise une boucle `while`.
Teste : `[4, 7, -2, 5, -9]` → 2 ; `[1, 2]` → -1.

À toi, en solo · réponse

Pistes de correction

```
def premier_negatif(liste):
    i = 0
    while i < len(liste):
        if liste[i] < 0:
            return i
        i = i + 1
    return -1

print(premier_negatif([4, 7, -2, 5, -9]))  # 2
print(premier_negatif([1, 2]))             # -1
```

Vérifie que `i` augmente à chaque tour : sinon, boucle infinie.

Pièges classiques

Piège	Pourquoi c'est un problème	Comment l'éviter
Oublier d'avancer dans un `while`	Boucle infinie	Chercher la ligne qui rapproche de la fin
Confondre `=` et `==`	= donne une valeur, == compare	Lire = comme « devient »
Initialiser au mauvais endroit	Le compteur repart de 0 à chaque tour	Initialiser avant la boucle
Erreur de un (off-by-one)	On saute le dernier élément	Tester sur 0, 1 et 2 éléments

Mini-boss

Mini-boss

Écris `statistiques(notes)` qui renvoie un dictionnaire avec : `minimum`, `maximum`, `moyenne`, et `reussites` (nombre de notes >= 10).
Pour une liste vide, renvoie `None`. N'utilise pas `min`, `max` ni `sum` : seulement des boucles et des conditions.

Teste avec `[12, 8, 15, 9, 10]` : minimum 8, maximum 15, moyenne 10.8, réussites 3.

Mini-boss · réponse

Pistes de correction

```
def statistiques(notes):
    if len(notes) == 0:
        return None
    mini = notes[0]
    maxi = notes[0]
    total = 0
    reussites = 0
    for n in notes:
        if n < mini:
            mini = n
        if n > maxi:
            maxi = n
        total = total + n
        if n >= 10:
            reussites = reussites + 1
    return {"minimum": mini, "maximum": maxi,
            "moyenne": total / len(notes), "reussites": reussites}

print(statistiques([12, 8, 15, 9, 10]))
```

Mini-boss · réponse

Pistes de correction (suite)

```
print(statistiques([]))
```

Quiz éclair

Quiz éclair

1. `total = total + 5` signifie :

A. total est égal à total + 5 (équation)
B. total vaut toujours 5
C. calcule total + 5, puis range le résultat dans total

2. Un `while` s'arrête quand :

A. la condition devient fausse
B. le programme est long
C. on a appelé print

3. Une boucle infinie vient souvent :

A. d'un nom trop long

Quiz éclair

Quiz éclair (suite)

B. d'une variable de boucle jamais modifiée
C. d'une liste trop courte

4. 🔁 *Révision mission 01* : Un algorithme est :

A. un langage
B. un fichier
C. une suite d'étapes précises qui se termine

5. 🔁 *Révision mission 01* : La trace sert à :

A. vérifier un algorithme à la main
B. compiler le code
C. effacer les erreurs

Quiz éclair · réponse

Réponses

1. **C**.
2. **A**.
3. **B**.
4. **C**.
5. **A**.

Je réfléchis

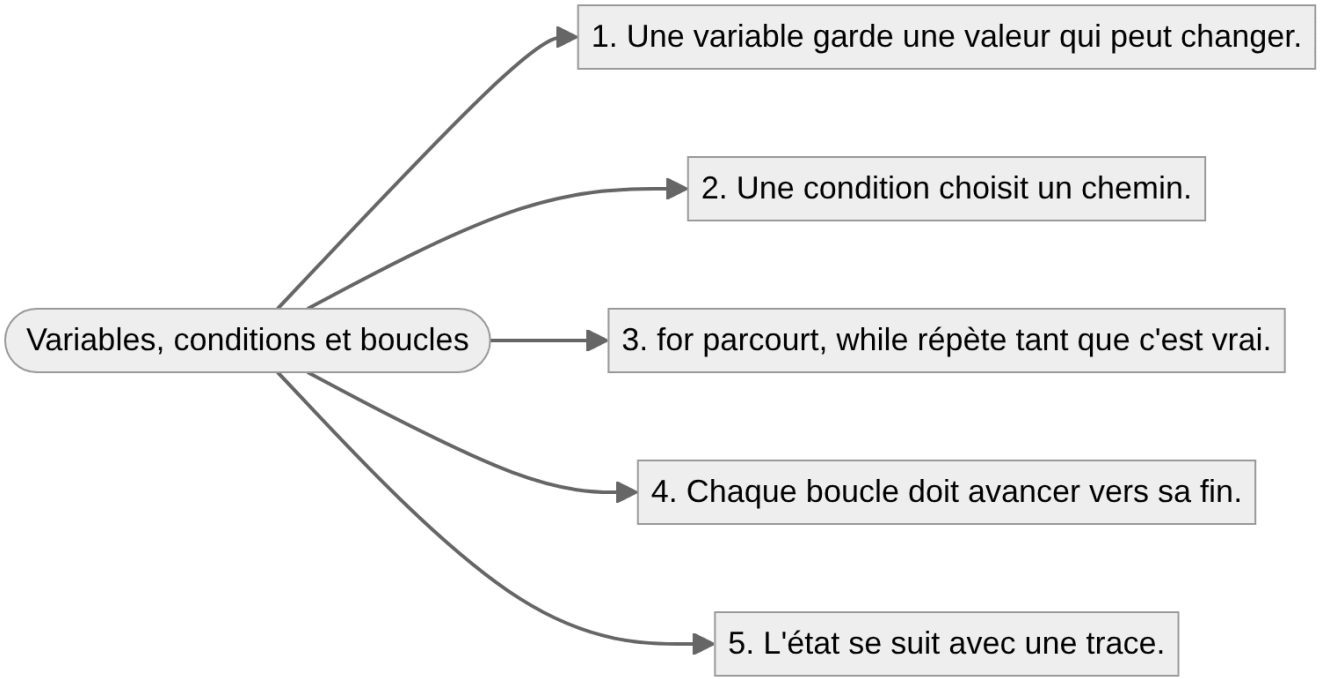
1. Quelle brique (variable, condition, boucle) t'a le plus posé de difficultés ? Pourquoi ?
2. Où, dans ta vie, répètes-tu une action « tant que » quelque chose n'est pas atteint ?

À retenir

1. Une variable garde une valeur qui peut changer.
2. Une condition choisit un chemin.
3. `for` parcourt, `while` répète tant que c'est vrai.
4. Chaque boucle doit avancer vers sa fin.
5. L'état se suit avec une trace.

À retenir

Le schéma de la mission



MISSION ACCOMPLIE

+100 XP

Badge débloqué : Chef d'orchestre

Prochaine mission : Fonctions, invariants et contrats

