01. **Mise en route**

Comprendre le mot “Algorithme”

Pourquoi ce mot paraît compliqué ?

Quand on entend le mot **algorithme**, on pense souvent à quelque chose de très complexe, réservé aux experts ou aux mathématiciens. Pourtant, la réalité est toute autre.

- Derrière ce mot savant se cache en fait **une idée simple** : une suite d'instructions précises pour atteindre un résultat.
- Chaque **technologie** que nous utilisons au quotidien – téléphone, GPS, réseaux sociaux – fonctionne grâce à des algorithmes.
- Et même sans le savoir, nous en utilisons tous les jours dans notre vie : suivre une recette de cuisine, trier des papiers, se brosser les dents, ce sont déjà des algorithmes.

👉 Un algorithme, ce n'est pas forcément lié à l'informatique : c'est avant tout une **façon d'organiser les étapes d'un problème**.

Définition simple .

*Un algorithme est une suite **finie** et **ordonnée** d'instructions **claires** et **précises** permettant de résoudre un problème ou d'atteindre un objectif.*

Exemple immédiat .

Une action aussi simple que se **préparer un thé** ☕ peut donner lieu à un algorithme :

1. Remplir la bouilloire avec un demi litre d'eau
2. Mettre en marche la bouilloire
3. Quand l'eau bout, verser dans la tasse
4. Infuser le sachet
5. Attendre 3 minutes
6. Retirer le sachet
7. Boire le thé

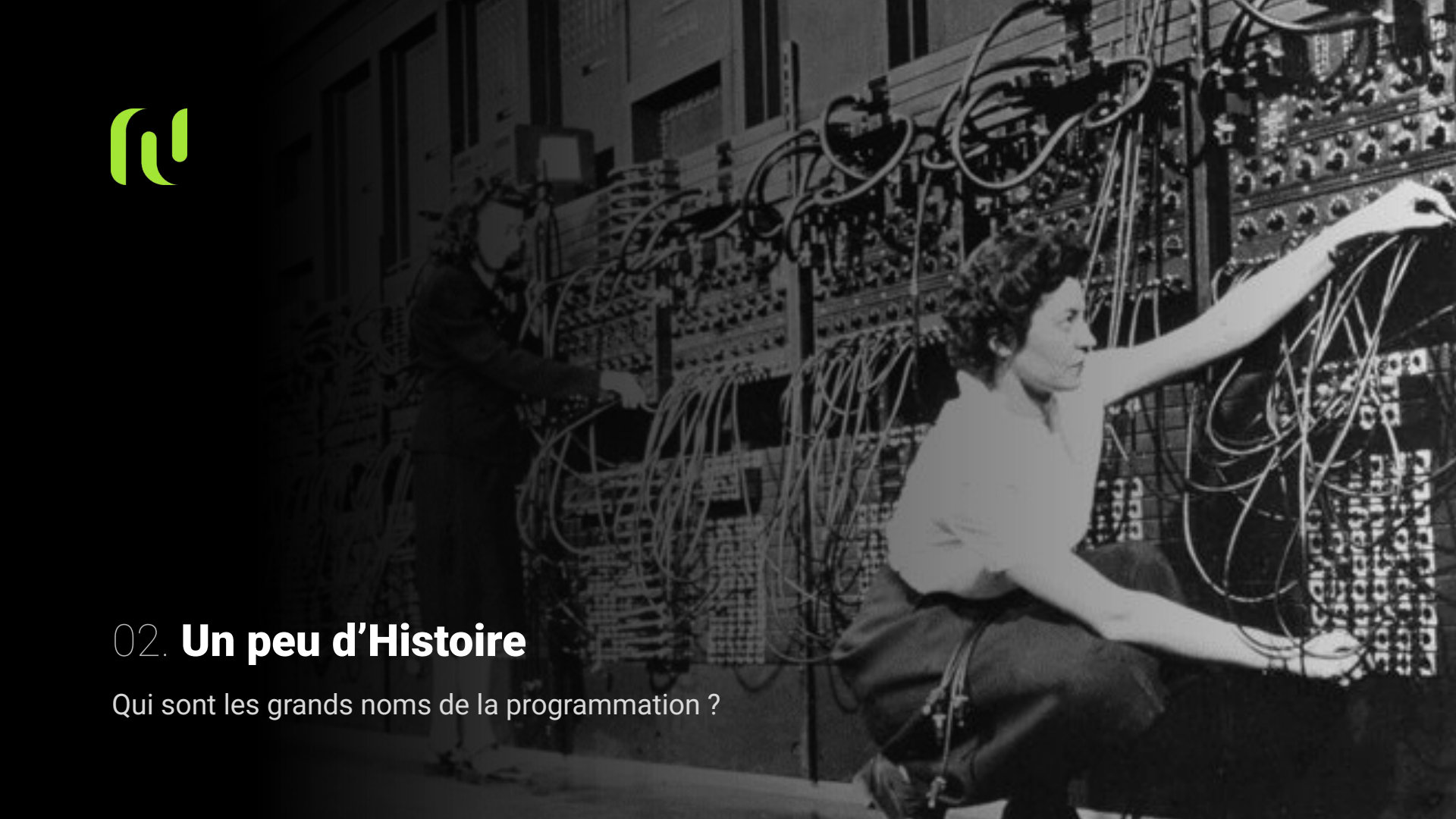
Différence algorithme/programme

Un **algorithme**, c'est avant tout une **idée** : une suite d'étapes logiques pour résoudre un problème. On peut l'écrire en français, en pseudo-code, ou même le représenter par un schéma.

Par exemple :

« Pour préparer un thé : faire bouillir de l'eau, la verser dans une tasse, ajouter un sachet de thé, attendre 3 minutes. »

Un **programme**, c'est cette même idée **traduite dans un langage compris par une machine**. C'est comme si on donnait la recette non plus en français, mais dans une langue que l'ordinateur comprend.



02. **Un peu d’Histoire**

Qui sont les grands noms de la programmation ?

Euclide : l’algorithme antique

Bien avant l’informatique, les mathématiciens de l’Antiquité utilisaient déjà des algorithmes. Euclide, **vers 300 av. J.-C.**, a décrit l’un des plus célèbres, encore enseigné aujourd’hui : le calcul du **Plus Grand Commun Diviseur (PGCD)**.

- Euclide est un mathématicien grec de l’Antiquité, auteur des **Éléments**.
- Son algorithme permet de trouver le **PGCD de deux nombres**.
- C’est l’un des **tout premiers algorithmes décrits formellement**.
- Principe : répéter des divisions successives jusqu’à obtenir un reste nul.

Origine des algorithmes

Al-Khwarizmi est un mathématicien perse du IXe siècle dont les travaux ont marqué l'histoire des mathématiques et donné naissance au mot "algorithme".

- **Né vers 780** à Khiva (Ouzbékistan actuel), savant de la Maison de la Sagesse à Bagdad.
- Auteur de l'ouvrage **"Al-jabr wa'l muqabala"** (origine du mot "algèbre").
- A introduit en Occident **le système de numération arabe** (chiffres 0 à 9).
- Son nom latinisé **"Algorithmi"** a donné le terme moderne **algorithme.**
- Ses méthodes de calculs ont révolutionné les mathématiques médiévales et posé les bases de l'informatique.

Blaise Pascal : la première machine à calculer

Au XVIIe siècle, un jeune génie français conçoit une machine capable d'automatiser les calculs arithmétiques. Blaise Pascal (1623–1662) devient ainsi un **précurseur de l'informatique mécanique**.

- Invente la **Pascaline** en 1642 pour aider son père, percepteur d'impôts.
- Fonction : réaliser automatiquement des additions et soustractions.
- **Première machine à calculer** mécanique de l'histoire.
- Importance : amorce l'idée que l'on peut **déléguer des calculs répétitifs** à une machine.
- Préfigure les ordinateurs en introduisant le concept d'automatisation du calcul.

La première programmeuse de l'Histoire .

Aux origines de la programmation, une visionnaire imagine déjà le potentiel des machines :

- **Ada Lovelace**, mathématicienne britannique, souvent considérée comme **la première programmeuse de l'histoire**.
- Elle a rédigé **le premier algorithme** destiné à être exécuté par une machine.
- Elle pressent déjà, à l'époque, que les machines pourraient **manipuler autre chose que des chiffres** (sons, symboles…)
- Son travail pose les **bases de la programmation moderne**, bien avant l'invention des ordinateurs.

Alan Turing : l’homme qui fit penser les machines .

En pleine Seconde Guerre mondiale, un génie pose les fondations de l’informatique moderne.

- **Alan Turing,** mathématicien britannique, considéré comme l’un des **pères de l’informatique.**
- En 1936, il formalise la notion de "**machine de Turing**", modèle abstrait de calcul encore utilisé aujourd’hui.
- Pendant la guerre, il joue un rôle clé dans le **déchiffrage du code Enigma** utilisé par les nazis.
- Son travail démontre qu’une machine peut **manipuler des symboles et simuler un raisonnement humain.**
- Il ouvre la voie à l’idée **d’intelligence artificielle** avec sa célèbre question : "*Les machines peuvent-elles penser ?*"

John von Neumann : les machines peuvent-elles penser ?

John von Neumann a posé en 1945 les bases de l'architecture des ordinateurs modernes, encore utilisée aujourd'hui.

- Mathématicien et physicien hongrois-américain.
- Formalise l'**architecture de von Neumann**.
- Une même **mémoire** contient les données et les instructions.
- Un **processeur** exécute les instructions **séquentiellement**.
- Modèle toujours présent dans PC, smartphones, consoles.

Donald Knuth : *The Art of Computer Programming*

Donald Knuth est un informaticien américain, considéré comme l'un des **pères de l'informatique moderne**. Son travail a profondément marqué l'étude et la pratique des algorithmes.

- Né en 1938, professeur à Stanford.
- Auteur de la série de livres *The Art of Computer Programming*.
- A popularisé les notations O(n), O(log n), O(n^2) pour comparer les **performances des algorithmes**.
- Inventeur du langage **TeX** (mise en page scientifique).

Points importants :

Ses travaux permettent de dire si un algorithme est rapide ou lent, surtout quand les données deviennent énormes.

Référence encore incontournable aujourd'hui pour les chercheurs et développeurs.

Focus : *Turing & Enigma*

La machine **Enigma**, utilisée par les nazis, chiffrait les messages militaires grâce à un système complexe de substitutions. Son décryptage a montré la puissance des algorithmes appliqués à un problème concret.

- Enigma repose sur un **algorithme de chiffrement** : une suite d'étapes systématiques pour transformer un texte clair en texte codé.
- Chaque jour, la configuration changeait → rendant l'algorithme encore plus difficile à casser.
- Turing et son équipe ont conçu **des algorithmes de décryptage**, puis des machines pour les exécuter rapidement.
- C'est l'une des premières fois que l'on a dû créer des machines informatiques pour exécuter un algorithme trop complexe pour les humains.

03. **Les algorithmes du quotidien**

Vous les utilisez tous les jours… 😁

Deux type d’algorithmes

Nous pouvons distinguer deux types d’algorithmes :

Les algorithmes **visibles**

Les algorithmes **invisibles**

Les algorithmes du quotidien

La tarte aux poires

Ingrédients :

- 1 pâte sablée
- 4 poires bien mûres
- 80 g de poudre d'amandes
- 60 g de sucre
- 50 g de beurre mou
- 1 œuf
- 1 sachet de sucre vanillé

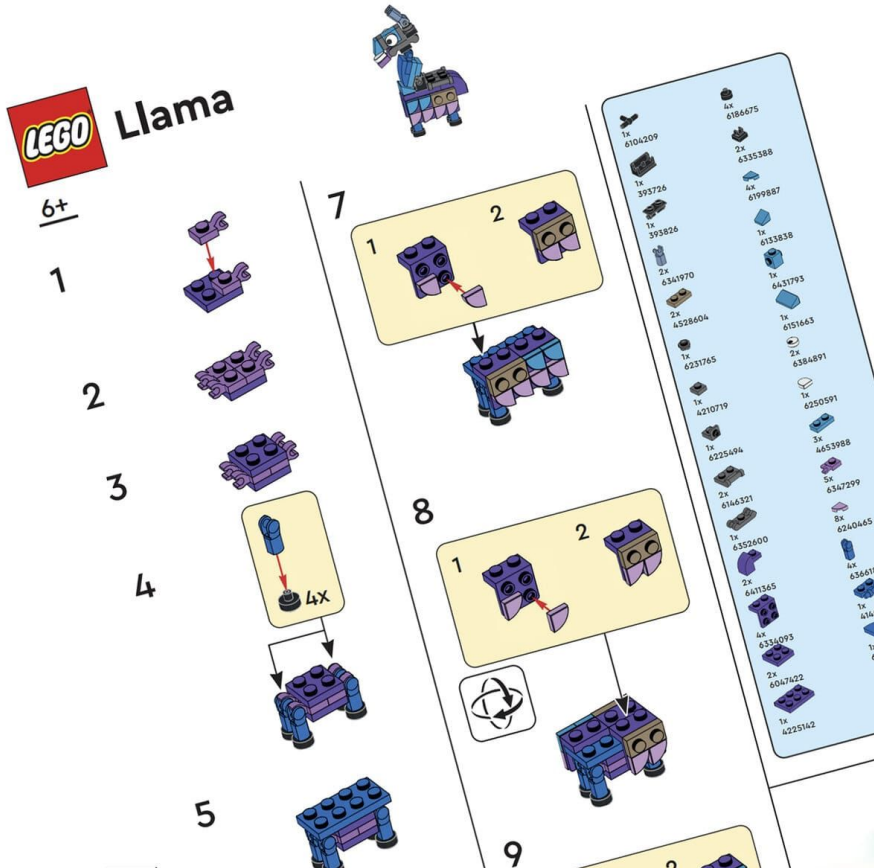
Préparation :

1. Éplucher les poires, les couper en deux et retirer le cœur.
2. Préparer la crème d'amandes : mélanger le beurre mou, le sucre, le sucre vanillé, l'œuf et la poudre d'amandes.
3. Étaler la pâte sablée dans un moule à tarte.
4. Si le bord de la pâte est trop large, le découper.
5. Répartir la crème d'amandes sur la pâte.
6. Disposer les demi-poires sur la crème, face bombée vers le haut.
7. Préchauffer le four à 180 °C.
8. Enfourner la tarte pendant environ 30 minutes, jusqu'à ce que la crème d'amandes soit dorée.
9. Laisser tiédir avant de servir, éventuellement saupoudrer de sucre glace.

Les algorithmes du quotidien

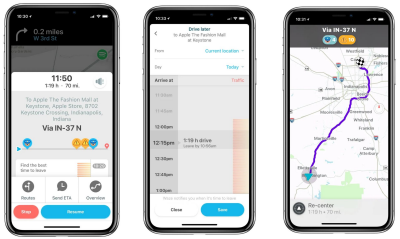
D’autres exemples d’algorithmes visibles…

- Les recettes de cuisine
- Les notices de montage de meubles
- Les notices LEGO



Les algorithmes du quotidien

D’autres exemples d’algorithmes invisibles…



GPS

Calcul d’itinéraire le plus court

NETFLIX

Netflix / Spotify

Recommandation de contenus

Banque

Calcul des intérêts de vos livrets d’épargne

🎯 Exercice - les algorithme du quotidien

Réfléchissez à des situations de la vie courante qui ressemblent à des algorithmes.

Un algorithme, c’est simplement une suite d’étapes précises qu’on suit pour arriver à un résultat.

👉 **Votre tâche** :

Citez quelques exemples d’algorithmes que vous utilisez dans votre journée (à la maison, dans la rue, au travail, etc.).

Expliquez **pourquoi** vous pensez que c’est un algorithme.

🎯 Réflexion .

Dans les algorithmes cités, que se passe-t-il s’il manque une étape ?

04. **Les structures fondamentales**

Comment est composé un bon algo ?

Les 3 structures fondamentales

Séquences

Conditions

Répétitions

Les séquences .

Une séquence, c'est une suite d'actions qui s'exécutent **dans l'ordre**, l'une après l'autre.

- Chaque étape **dépend** de l'étape précédente.
- Si on saute une étape, le résultat peut **être faux**.
- C'est la structure la plus simple des algorithmes.

👉 **Exemple de la vie courante :** faire un sandwich.

1. Prendre deux tranches de pain.
2. Étaler du beurre.
3. Ajouter une tranche de jambon.
4. Refermer avec la deuxième tranche.

Les conditions .

Une condition permet de **prendre une décision** : l'algorithme choisit entre plusieurs chemins possibles.

- Si … alors …
- On teste une situation, et en fonction de la réponse (oui / non, vrai / faux), on choisit quoi faire.

👉 **Exemple de la vie courante :** le feu tricolore 🚦

- **Si le feu est vert** → j'avance
- **Si le feu est orange ou rouge** → je m'arrête

Les répétitions .

Une répétition, c'est quand on demande à l'algorithme de **répéter une même action plusieurs fois**.

- "Tant que ..." (condition de fin)
- "Pour chaque ..." (boucle sur une liste)

👉 **Exemple de la vie courante :** le feu tricolore 🚦

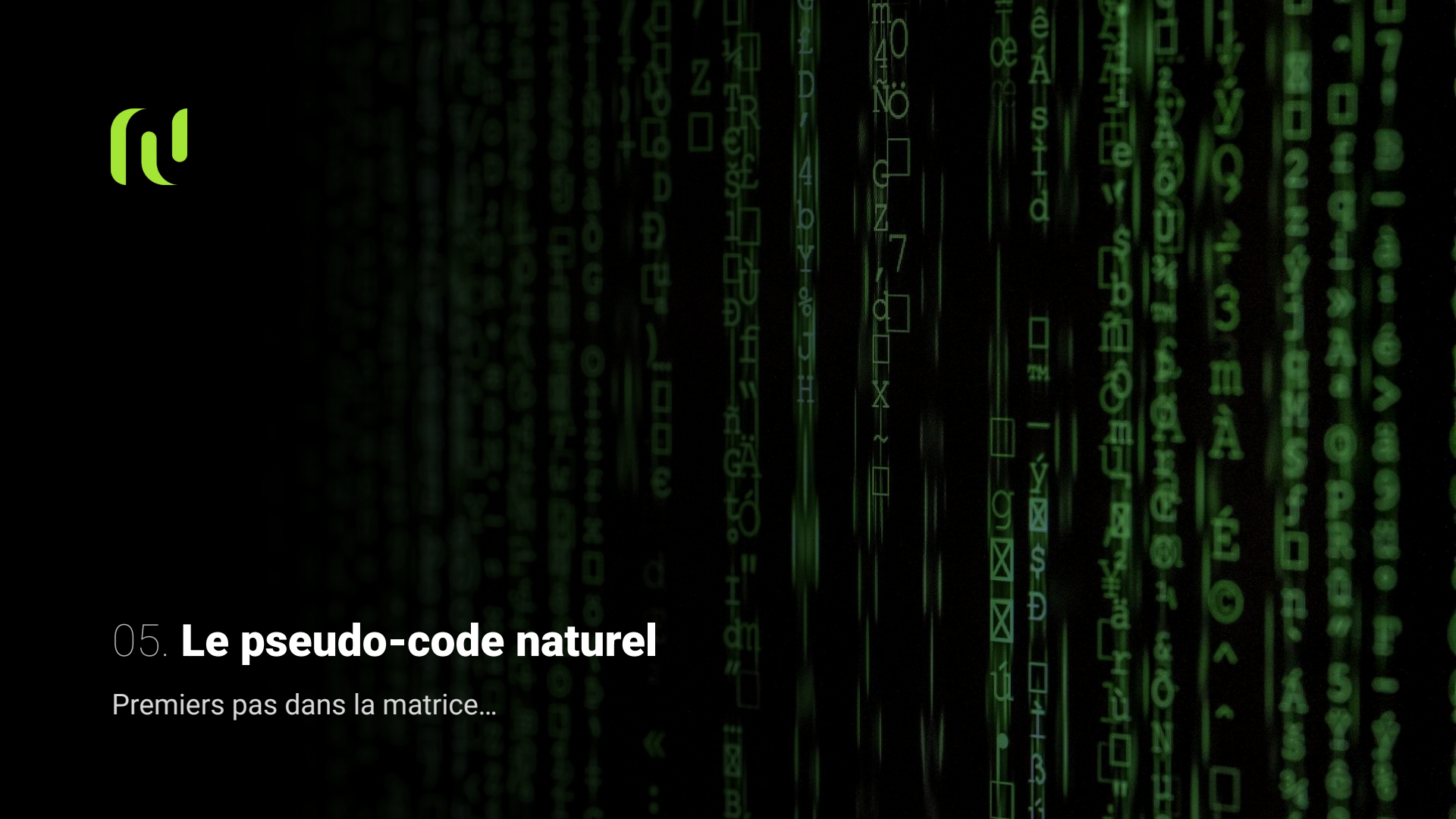
- **Faire 10 pompes :** on répète la même action jusqu'à atteindre le nombre demandé.
- **Jouer aux cartes :** distribuer une carte à chaque joueur jusqu'à ce que tout le monde en ait.

Identifier les structures

La tarte aux poires

Préparation :

1. Éplucher les poires, les couper en deux et retirer le cœur.
2. Préparer la crème d'amandes : mélanger le beurre mou, le sucre, le sucre vanillé, l'œuf et la poudre d'amandes.
3. Étaler la pâte sablée dans un moule à tarte.
4. Si le bord de la pâte est trop large, le découper.
5. Répartir la crème d'amandes sur la pâte.
6. Disposer les demi-poires sur la crème, face bombée vers le haut.
7. Préchauffer le four à 180 °C.
8. Enfourner la tarte pendant environ 30 minutes, jusqu'à ce que la crème d'amandes soit dorée.
9. Laisser tiédir avant de servir, éventuellement saupoudrer de sucre glace.



05. **Le pseudo-code naturel**

Premiers pas dans la matrice…

Pourquoi du pseudo-code ?

Le pseudo-code est une manière d'écrire un algorithme sans utiliser un vrai langage informatique. C'est un outil de transition entre le langage naturel (comme le français) et le langage machine (comme Python).

- **Lisible par tous** : pas besoin de connaître la programmation.
- **Précis** : plus structuré qu'une phrase en français libre.
- **Universel** : tout le monde peut comprendre le raisonnement, même sans être informaticien.
- **Pratique** : c'est un brouillon avant de passer au code.

🍳 Analogie cuisine

Langage naturel → quelqu'un vous dit "prépare-moi un café".

Pseudo-code → une liste d'étapes précises, mais écrites simplement.

Programme → la même liste, mais traduite dans la langue d'un robot ou d'un ordinateur.

Exemple de pseudo-code simple

Préparer un thé 🫖 :

```
Algorithme PréparerThé
Entrée : eau, thé
Sortie : thé prêt
Début
  Faire bouillir l'eau
  Verser l'eau dans la tasse
  Mettre le sachet de thé
  Attendre 3 minutes
Fin
```

🎯 Activité pratique

Réécrire la recette de la tarte aux poires en pseudo code.

Ingrédients :

- 1 pâte sablée
- 4 poires bien mûres
- 80 g de poudre d'amandes
- 60 g de sucre
- 50 g de beurre mou
- 1 œuf
- 1 sachet de sucre vanillé

Préparation :

1. Éplucher 4 poires, les couper en deux et retirer le cœur.
2. Préparer la crème d'amandes : mélanger le beurre mou, le sucre, le sucre vanillé, l'œuf et la poudre d'amandes.
3. Étaler la pâte sablée dans un moule à tarte.
4. Si le bord de la pâte est trop large, le découper.
5. Répartir la crème d'amandes sur la pâte.
6. Disposer les demi-poires sur la crème, face bombée vers le haut.
7. Préchauffer le four à 180 °C.
8. Enfourner la tarte pendant environ 30 minutes, jusqu'à ce que la crème d'amandes soit dorée.
9. Laisser tiédir avant de servir, éventuellement saupoudrer de sucre glace.

🎯 Activité pratique - Identifiez les erreurs

```
Algorithme PréparerTarteAuxPoires
Entrée : pâte sablée, poires, beurre, sucre, amandes, œuf, sucre vanillé
Sortie : tarte cuite
Étapes :
  Éplucher les poires
  Couper les poires en 2
  Retirer le coeur des poires
  Mélanger beurre, sucre, œuf, amandes, sucre vanillé
  Étaler la pâte dans un moule
  Si la pâte dépasse → découper
  Répartir crème + poires
  Préchauffer four à 180°C
  Si le four est chaud → Cuire 30 min
```

06. **Notion de performances**

La performance au coeur des problématiques des devs

Qu’est ce qu’un bon algorithme ?

Un algorithme n’est pas seulement une suite d’étapes, il doit avoir certaines qualités.

- **Finitude** : il doit toujours s’arrêter (sinon → boucle infinie).
- **Précision** : chaque étape doit être claire et sans ambiguïté.
- **Efficacité** : il doit atteindre son but, avec le moins d’efforts possibles.

Exemple de finitude

Que se passe-t-il si un algorithme ne s'arrête pas ?

- **Recette de cuisine** → si on dit « mélanger jusqu'à ce que ce soit parfait » → jamais fini.
- **Brossage de dents** → « brosser jusqu'à ce que ce soit bien » → trop vague.
- 👉 Toujours prévoir une condition de fin **claire**.

Exemple de précision

Pourquoi la précision est essentielle :

- **« Mettre du dentifrice »** → Combien ? Où ?
- **« Faire cuire »** → Combien de minutes ? Quelle température ?
- 👉 Sans précision, la machine (ou un autre humain) risque de mal exécuter.

Exemple de deux méthodes différentes

Deux algorithmes peuvent résoudre le même problème, mais pas avec la même efficacité.

Par exemple, chercher un mot dans un dictionnaire papier.

- **Méthode 1** → feuilleter page par page (long).
- **Méthode 2** → couper le dictionnaire au milieu (rapide).

➡️ **Quelle méthode choisiriez-vous pour un dictionnaire de 1000 pages ? Pourquoi ?**

Exemple de deux méthodes différentes

Deux algorithmes peuvent résoudre le même problème, mais pas avec la même efficacité.

Par exemple, chercher un mot dans un dictionnaire papier.

- **Méthode 1** → feuilleter page par page (long).
- **Méthode 2** → couper le dictionnaire au milieu (rapide).

➡️ **Quelle méthode choisiriez-vous pour un dictionnaire de 1000 pages ? Pourquoi ?**

Feuilleter page par page → jusqu'à 1000 étapes.

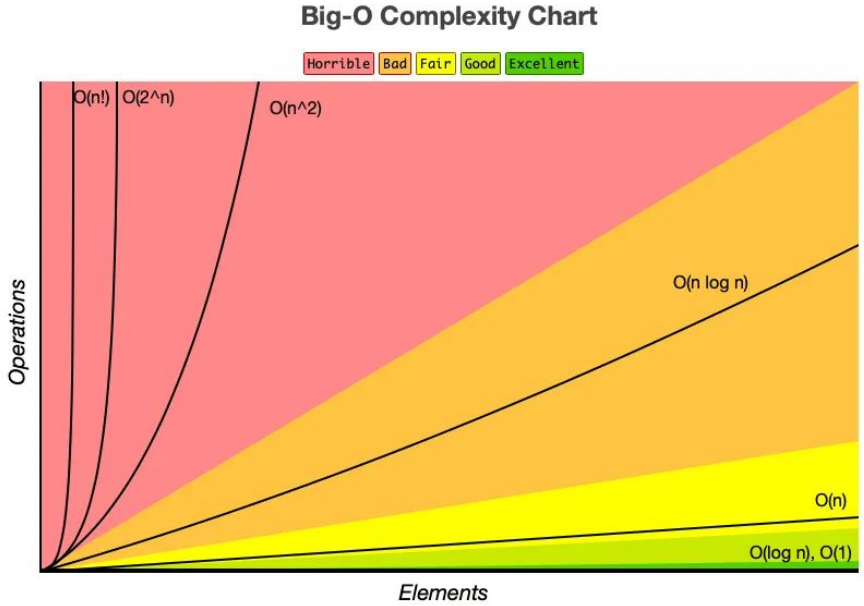
Couper en deux → environ 10 étapes.

👉 Les bons algorithmes changent tout quand la quantité de données augmente.

Mesurer la performance : notation Big O

La notation **Big O** est une manière de mesurer l'efficacité d'un algorithme. Elle décrit combien d'étapes un algorithme a besoin en fonction de la taille du problème (nombre d'éléments à traiter).

- Sert à comparer des algorithmes entre eux.
- Ne donne pas le temps exact en secondes, mais une tendance
- S'écrit avec la lettre O suivie d'une expression : O(1), O(n), O(log n)…
- Plus la complexité est basse, plus l'algorithme est efficace quand les données grandissent.



Bonus : prévoir l'imprévu

Un bon algorithme ne traite pas uniquement le *Happy Path*.

Il doit tenir compte des situations exceptionnelles qu'il pourra rencontrer lors de son exécution.

Par exemple, cela implique :

- **Une bonne gestion des erreurs** → Que se passe-t-il si une des entrées est incorrecte ? Si l'une des dépendances (serveurs, API…) n'est pas disponible ?
- **Une sortie *graceful* en cas d'échec** → L'algorithme -et plus tard le programme- devront gérer de la façon la plus transparente possible les échecs pour que l'utilisateur conserve une expérience optimale.

🎯 Activité #1

Décrire un algorithme pour les situations suivantes :

```
Algorithme PréparerThé
Entrée : eau, thé
Sortie : thé prêt
Début
  Faire bouillir l'eau
  Verser l'eau dans la tasse
  Mettre le sachet de thé
  Attendre 3 minutes
Fin
```

- **Aller de votre domicile aux locaux de la Metz Numeric School**
  Quelles sont les différentes étapes suivies pour vous rendre à l'école aujourd'hui ? (imaginez un trajet fictif si besoin)
- **Prendre l'ascenseur**
  D'un point de vue usager, quelles sont les étapes pour prendre un ascenseur ?
- **Jeu "Devine le nombre"**
  Le joueur doit deviner un nombre entre 1 et 1000 choisi par l'ordinateur.
- **Classement d'objets**
  Comment classer 3 objets (A, B, C) du plus petit au plus grand ?
- **Gestion d'une file d'attente (ex. médecin)**
  Imaginez l'algorithme que représente la gestion d'une file d'attente.