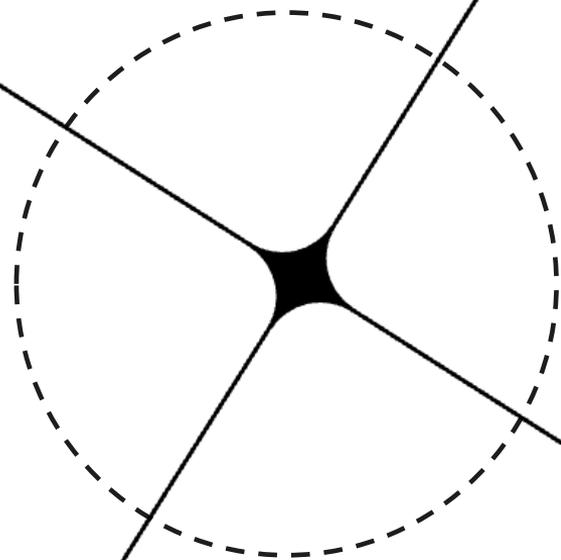


# *La programmation à l'heure de l'IA générationnelle*

Julien Romero



# La programmation à l'heure de l'IA générative

- Des outils comme ChatGPT ou GitHub Copilot peuvent maintenant écrire du code
- Il est maintenant possible de construire des applications sans taper une seule ligne de code
- Alors... pourquoi devons-nous apprendre Java ?

Apprendre Java (et la programmation), ce n'est pas juste apprendre une syntaxe.

Le but, c'est d'apprendre à penser comme un ingénieur.

**Pourquoi** on construit un produit, et pas seulement comment.

**Conception** de systèmes, faire des compromis, gérer la complexité.

**Communiquer** avec des utilisateurs, équipes, et son soi du futur.

Résoudre de **nouveaux problèmes**, pas seulement suivre un guide.

# *Oui, une IA peut coder*

- ChatGPT peut créer des programmes complets à partir d'un simple prompt
  - Prompt = instruction
- GitHub Copilot complète automatiquement les fonctions que vous tapez
- Les délais de commercialisation diminuent, même pour des applications complexes

# Oui, une IA peut coder

Écris un tic tac toe en Python.

Voici un exemple simple d'un jeu de **Tic Tac Toe** (morpion) en Python, jouable à deux joueurs dans le terminal :

```
python Copier Modifier  
  
def afficher_grille(grille):  
    print("\n")  
    for ligne in grille:  
        print(" | ".join(ligne))  
        print("-" * 9)  
    print("\n")  
  
def verifier_victoire(grille, joueur):  
    # Vérifie lignes, colonnes et diagonales  
    for i in range(3):  
        if all(grille[i][j] == joueur for j in range(3)): # ligne  
            return True  
        if all(grille[j][i] == joueur for j in range(3)): # colonne  
            return True  
    if all(grille[i][i] == joueur for i in range(3)): # diagonale principale  
        return True  
    if all(grille[i][2 - i] == joueur for i in range(3)): # diagonale secondaire  
        return True  
    return False  
  
def case_valide(grille, x, y):  
    # Vérifie si la case est libre
```

Poser une question

+ Outils 🗣️ ⬆️

```
X | X | O  
-----  
  | O |  
-----  
  |  |  
-----
```

Joueur X, entrez la ligne (0-2) : 1  
Joueur X, entrez la colonne (0-2) : 0

```
X | X | O  
-----  
X | O |  
-----  
  |  |  
-----
```

Joueur O, entrez la ligne (0-2) : 2  
Joueur O, entrez la colonne (0-2) : 0

```
X | X | O  
-----  
X | O |  
-----  
O |  |  
-----
```

🎉 Joueur O a gagné !

## *Oui, une IA peut coder*

- ChatGPT peut créer des programmes complets à partir d'un simple prompt
  - Prompt = instruction
- GitHub copilot complète automatiquement les fonctions que vous tapez
- Les délais de commercialisation diminuent, même pour des applications complexes

Que pouvez-vous apporter quand une IA peut déjà coder ?

## *Mais l'IA a des limites*

- L'IA fait des erreurs, avec beaucoup de confiance
- L'IA a une vue très limitée sur le contexte dans la réalité
- L'IA peut introduire des bugs, des biais, ou du code non sécurisé
  - Qui est responsable ?

Vous êtes toujours utile pour guider et corriger la machine.  
Au final, vous serez responsable s'il y a un problème.

# Mais l'IA a des limites

Can you update the fields to zero-out?

Modification de Feature Classifier Baseline

Updated `load_candidates`: the "zero-out" section now clears **all** LLM-only fields you introduced (`top_university`, `multi_degree`, `volunteer_experience`, `programming_languages`, etc.). Nothing else changed.

Give it a try—let me know if any field slipped through!



You forgot "job\_category", "expertise\_area".

Added `job_cat` and `expertise` to the zero-out lists for both `load_candidates` and `load_jobs`, plus the remaining llm-specific list columns (`hard_skills`, `soft_skills`, `programming_languages`, etc.) on the jobs side.

That ensures the ablation flag `--use_llm_features` truly blanks every LLM-only field. Let me know if any other keys need clearing!



wrong, the good answer was: [  
"industry", "values", "langs\_req", "senior\_req", "diploma\_req",  
"remote",

# ***Pourquoi quand même apprendre la programmation ?***

- Le langage n'est qu'un moyen d'expression
  - La programmation vous apprend la **logique et la résolution de problèmes**
- Coder n'est pas seulement de l'écriture
  - Il s'agit de **comprendre des systèmes**
- Vous ne pouvez pas déboguer un programme si vous ne le comprenez pas

N'utiliser que l'IA vous empêchera de développer les modèles mentaux plus complexes.

## *Du code aux solutions*

- Le but n'est pas d'écrire du code
  - Nous devons **résoudre des problèmes réels**, qui répondent à des besoins
- L'IA vous aidera à développer plus vite, mais seulement si la solution est bien conçue
- La vitesse sans la compréhension mène à du travail supplémentaire et à des échecs
  - Une application fonctionnelle qui ne répond pas aux besoins est un échec

L'IA augmente votre productivité, mais ne remplace pas une bonne conception, de l'empathie, ou une réflexion stratégique.

# Croissance cognitive ou dépendance ?

- L'apprentissage doit être **actif**
  - Il n'y a pas de progrès sans difficulté
- L'IA vous soulage trop tôt et trop vite
  - le piège : L'IA vous donne l'illusion de la compétence
- D'abord réfléchissez, ensuite demandez à l'IA

# L'IA et l'intégrité académique

- Copiez du code généré par une IA sans le mentionner sera considéré comme de la triche
  - L'IA l'a peut-être copié sur un autre code trouvé en ligne (plagiat)
- Si vous ne pouvez pas expliquer un code, vous ne le comprenez pas
  - C'est votre compréhension que nous voulons évaluer
  - Souvent évident pour votre enseignant quand il vous évalue si vous comprenez ou non
- L'examen final sera sur papier (sans ChatGPT)

# Votre valeur en tant qu'ingénieur

- Le prompting n'est pas une compétence utile sur le long-terme
  - Devient de plus en plus simple
  - Tout le monde y arrive
  - Une IA peut elle-même écrire des prompts (<https://www.promptcowboy.ai/>)
- Votre vraie valeur : esprit critique, conception de systèmes complexes, compréhension de problèmes mal définis, communication
- L'IA ne vous remplacera pas...

# *Votre valeur en tant qu'ingénieur*

- Le prompting n'est pas une compétence utile sur le long-terme
  - Devient de plus en plus simple
  - Tout le monde y arrive
- Votre vraie valeur : esprit critique, conception de systèmes complexes, compréhension de problèmes mal définis, communication
- L'IA ne vous remplacera pas...
  - ... sauf si vous n'apportez rien de plus

# Comment étudier intelligemment avec l'IA

- D'abord, résoudre le problème vous-même
  - Trouver la solution n'est pas important (des centaines d'étudiants l'ont trouvé avant vous)
  - L'important, c'est de pratiquer et de se frotter à la difficulté
- Ensuite seulement, demander à l'IA s'il y a des erreurs ou des points d'amélioration
  - Pourquoi est-ce que cette version est meilleure ?
  - Est-elle plus lisible ?
  - Plus efficace ?

# Comment étudier intelligemment avec l'IA

Peux-tu m'aider à améliorer la solution à ce problème :

Écrire un algorithme qui dit si un chiffre donné en entrée est dans une liste triée.

Ma solution:

```
python
def find(x, l):
    for y in l:
        if y == x:
            return True
        else:
            pass
    return False
```

Donne moi tous les points d'amélioration.

Bien sûr, voici tous les points d'amélioration que je peux te proposer pour ta solution à ce problème. On va couvrir **la logique**, **les performances**, **le style**, et **la lisibilité**.

## 🔍 1. Optimisation : profiter du fait que la liste est triée

**Problème** : Ta fonction utilise une **recherche linéaire** ( $O(n)$ ), ce qui est inefficace pour une **liste triée**.

**Amélioration** : Utiliser une **recherche dichotomique** (↓ `binary search`), ce qui donne une complexité en  $O(\log n)$ .

**Correction** :

```
python Copier Modifier

def find(x, l):
    left, right = 0, len(l) - 1
    while left <= right:
        mid = (left + right) // 2
        if l[mid] == x:
            return True
        elif l[mid] < x:
            left = mid + 1
        else:
            right = mid - 1
    return False
```

## 🔧 2. Code inutile : `else: pass`

**Problème** : La clause `else: pass` est redondante et alourdit le code.

**Correction** :

```
python Copier Modifier

for y in l:
    if y == x:
        return True
# Pas besoin de else: pass
return False
```



# Comment étudier intelligemment avec l'IA

## 3. Nom des variables

**Problème :** `x` et `l` sont peu explicites.

**Amélioration :** Choisir des noms plus clairs comme `target` et `sorted_list`.

**Exemple :**

```
python Copier Modifier  
  
def find(target, sorted_list):  
    for value in sorted_list:  
        if value == target:  
            return True  
    return False
```

## 4. Utiliser les outils Python standards (optionnel)

Si tu n'as pas besoin d'implémenter toi-même la recherche binaire (par exemple pour un exercice), tu peux utiliser le module standard `bisect`.

```
python Copier Modifier  
  
import bisect  
  
def find(target, sorted_list):  
    index = bisect.bisect_left(sorted_list, target)  
    return index < len(sorted_list) and sorted_list[index] == target
```



# *Le futur : Collaboration Humain-IA*

- Les architectes utilisent déjà des outils de DAO (Design Assistée par Ordinateur) comme AutoCAD
  - Nous aurons l'IA
- L'IA fera partie de notre travail, mais ne nous remplacera pas
  - Elle sera votre partenaire
  - Vous devez être créatif sur les solutions apportées
  - La responsabilité finale vous reviendra
- La réflexion et le jugement resteront faits par des humains

## *En résumé : pensez comme des ingénieurs*

- Les outils changent, la réflexion reste
  - Java n'est qu'un outil
- Nous n'avons jamais été là uniquement pour écrire du code
  - Nous résolvons des problèmes
- Votre cerveau reste votre IDE principal