IA BioScripting : Utilisation des IA génératives
comme appui à la programmation et au scripting pour la biologie

13 juin 2025 Paris (France)

généré avec Gemini

Accélérer sans déraper : maîtriser son code à l'ère de l'IA

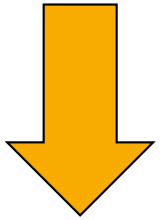
agAP INSTITUT
Amélioration génétique et adaptation des plantes méditerranéennes et tropicales

L'INSTITUT agro Montpellier

V. Ranwez

Changer de perspective par rapport à l'IA

codeur (débutant)



manager (débutant) de codeurs

Manager et déléguer efficacement (informatique)

Steps in effective delegation
Task Identification
Team Member Selection
Clear Communication
Support and Guidance
Performance Assessment

Effective delegation and leadership in software management
SD Mirkhan, et. al (2024)

délégation efficace
Fixez des objectifs
Évaluer les compétences
Fournir des ressources
Suivre les progrès
Offrir des commentaires
Encourager l'autonomie

https://www.linkedin.com/advice/0/heres-how-you-can-efficiently-delegate-tasks-professional-rbh0f?lang=fr&originalSubdomain=fr

Etape 1 : tester, tester, tester et encore tester

Des tests automatiques pour :

- clarifier les objectifs
- fournir des ressources/exemples
- valider les solutions
- refactoriser sereinement

Etape 1 : tester, tester, tester et encore tester

Des tests automatiques pour :

- clarifier les objectifs
- fournir des ressources/exemples
- valider les solutions
- refactoriser sereinement

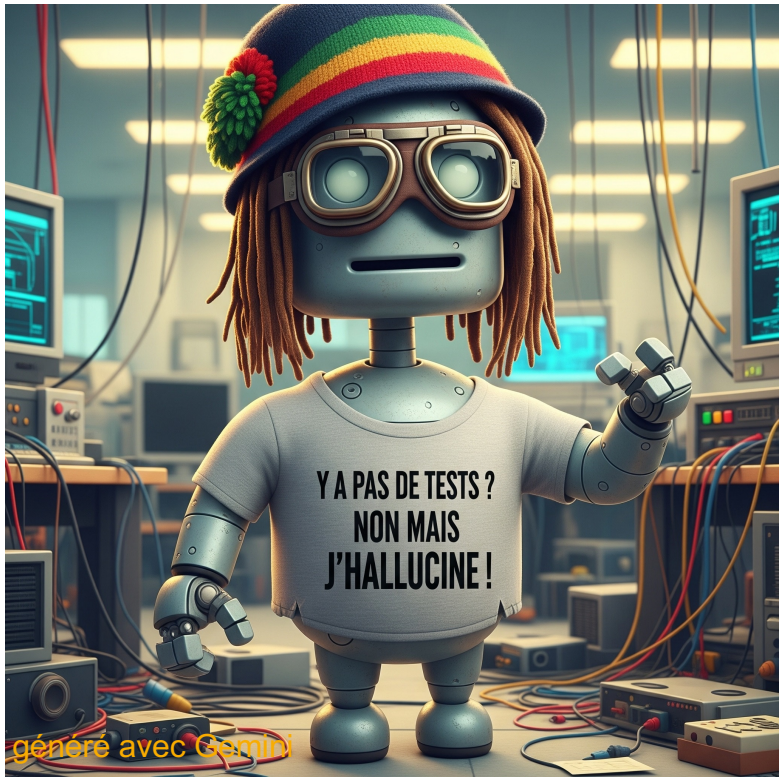
« Ouai les tests c'est cool mais moi j'peux pas j'fais des trucs compliqués ...» A. Nonyme

Etape 1 : tester les trucs compliqués

Etape 1 : tester les trucs compliqués

Tester une valeur exacte :

```
assert( pow(2,3) == 8)
```



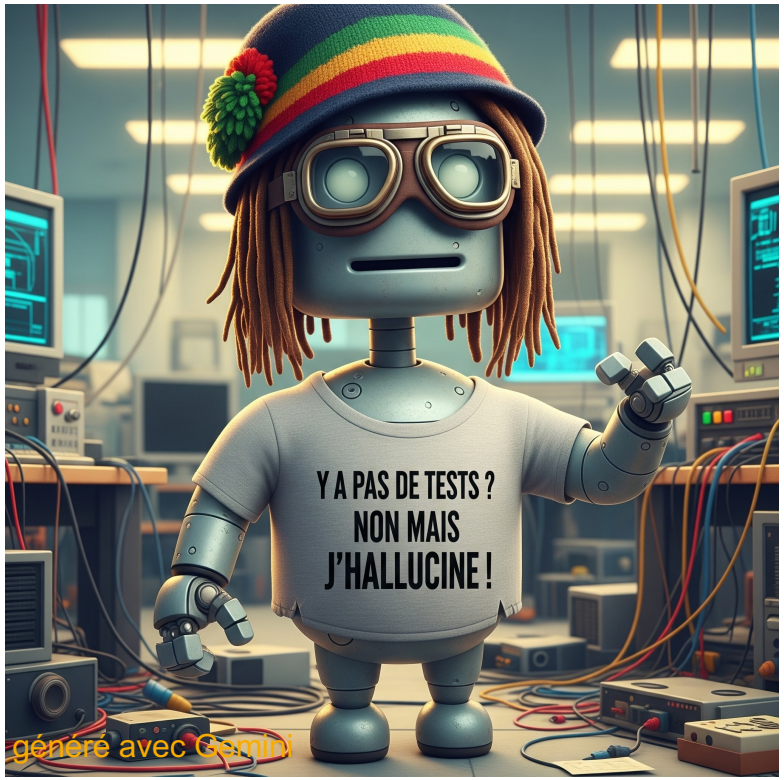
Etape 1 : tester les trucs compliqués

Tester une valeur exacte :

```
assert( pow(2,3) == 8)
```

Tester une tendance :

```
s=0.0;
for i in 1..1000
    s+= alea_entre(0,1)
mean=s/1000
assert ( abs(mean-0.5) < 0.01 )
```



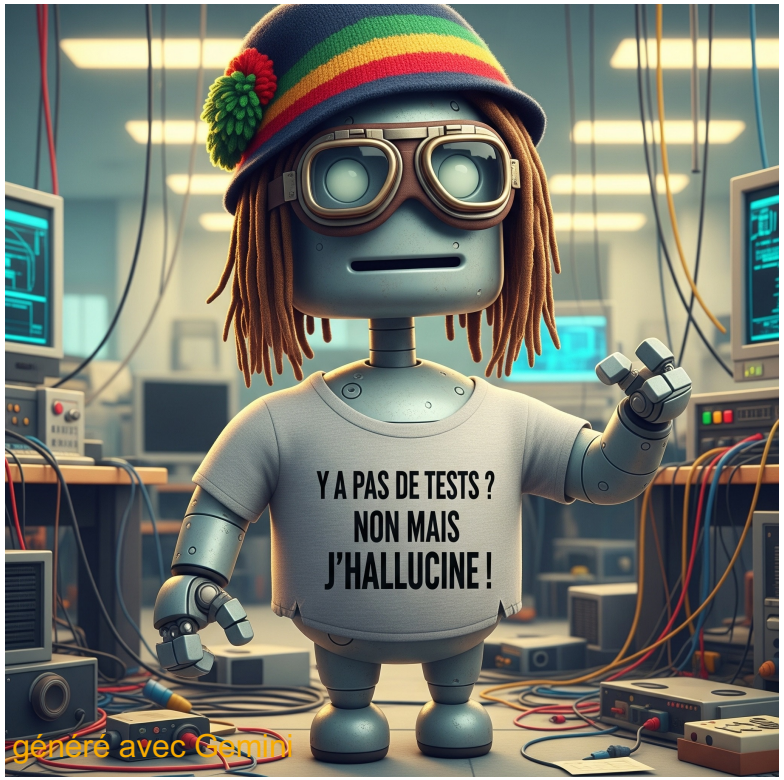
Etape 1 : tester les trucs compliqués

Tester une valeur exacte :

```
assert( pow(2,3) == 8)
```

Tester une tendance :

```
s=0.0;
for i in 1..1000
    s+= alea_entre(0,1)
mean=s/1000
assert ( abs(mean-0.5) < 0.01 )
```



Tester une propriété :

```
pred1=reads2geno(a=2; c=1; g=10; t=5)
pred2=reads2geno(a=2; c=1; g=11; t=5)
assert (pred1.GG <= pred2.GG)
```

Etape 2 : générer un code qui marche

Exprimer ses besoins :

- les tests aident
- des bases d'algorithmique sont un plus

Valider les réponses de l'IA :

- des bases du langage utilisé sont un plus
- des tests sont absolument obligatoires

« la confiance n'exclut pas le contrôle » Vladimir Ilitch Lénine

« Tout code non testé est buggé» Alex Périence

Etape 3 : améliorer le code sereinement

Ne pas s'arrêter quand ça marche, refactoriser pour :

- optimiser la lisibilité et la maintenabilité du code
- réorganiser si besoin (fonction, classes, modules, etc.)
- optimiser les performances (**si nécessaire**)
- rendre le code plus homogène avec le reste du projet

Etape 3 : améliorer le code sereinement

Ne pas s'arrêter quand ça marche, refactoriser pour :

- optimiser la lisibilité et la maintenabilité du code
- réorganiser si besoin (fonction, classes, modules, etc.)
- optimiser les performances (**si nécessaire**)
- rendre le code plus homogène avec le reste du projet

Et tout ça sans stress grâce :

- aux tests (on s'assure de ne rien casser)
- au versioning (on s'assure de pouvoir revenir en arrière si on casse)

Utiliser l'IA à chaque étape : il suffit de demander…

l'IA peut générer des tests unitaires
=> vous n'avez qu'à les valider/corriger

l'IA peut générer du code (et l'expliquer)
=> vous n'avez qu'à le valider/corriger

l'IA peut générer la doc du code
=> vous n'avez qu'à la valider/corriger

l'IA peut proposer d'améliorer du code (refactoring, debugging, code conventions)
=> vous n'avez qu'à les valider/corriger

l'IA peut proposer des optimisations algorithmiques
=> vous n'avez qu'à les valider/corriger

Utiliser l'IA à chaque étape : il suffit de demander…

l'IA peut générer des tests unitaires
=> vous n'avez qu'à les valider/corriger

l'IA peut générer du code (et l'expliquer)
=> vous n'avez qu'à le valider/corriger

l'IA peut générer la doc du code
=> vous n'avez qu'à la valider/corriger

l'IA peut proposer d'améliorer du code (refactoring, debugging, code conventions)
=> vous n'avez qu'à les valider/corriger

l'IA peut proposer des optimisations algorithmiques
=> vous n'avez qu'à les valider/corriger

=> Et oui c'est pas toujours rigolo la vie de chef …