

Partie 1 : Analyse Théorique

Thème : Créer des solutions logicielles intelligentes

Formation : Académie plp

Question 1 : Génération de code pilotée par l'IA

Comment les outils comme GitHub Copilot réduisent le temps de développement

Les outils de génération de code pilotés par l'intelligence artificielle, tels que GitHub Copilot, transforment la manière dont les développeurs abordent la programmation. En analysant le contexte du code en cours, ces outils suggèrent automatiquement des lignes ou des blocs de code pertinents, ce qui permet de :

- **Accélérer l'écriture du code** : les fonctions courantes, les boucles et les structures de données sont générées instantanément.
- **Réduire les erreurs syntaxiques** : les suggestions respectent les normes du langage utilisé.
- **Faciliter l'apprentissage** : les développeurs, notamment les débutants, peuvent observer les bonnes pratiques à travers les suggestions.
- **Automatiser les tâches répétitives** : ce qui libère du temps pour les aspects plus stratégiques du développement.

Limites et précautions

Malgré ses avantages, l'utilisation de ces outils comporte certaines limites :

- **Compréhension contextuelle limitée** : l'IA peut proposer du code qui ne correspond pas aux exigences métier ou à l'architecture du projet.
- **Dépendance excessive** : les développeurs risquent de ne pas comprendre pleinement le code généré.
- **Risques de sécurité** : le code proposé peut contenir des vulnérabilités ou ne pas respecter les normes de sécurité.
- **Qualité variable** : certaines suggestions peuvent être inefficaces ou non idiomatiques.

Conclusion

Ces outils sont des assistants puissants, mais leur utilisation doit rester encadrée par une expertise humaine pour garantir la qualité, la sécurité et la pertinence du code produit.

Question 2 : Apprentissage supervisé vs non supervisé dans la détection automatisée de bugs

Introduction

L'intelligence artificielle permet d'automatiser la détection de bugs dans les logiciels en analysant des données d'exécution, des journaux ou des comportements inattendus. Deux approches principales sont utilisées : l'apprentissage supervisé et l'apprentissage non supervisé.

Apprentissage supervisé

- **Définition** : Le modèle est entraîné sur des données étiquetées, où chaque exemple est classé comme "bug" ou "non-bug".
- **Avantages** :
 - Haute précision si les données sont bien annotées.
 - Permet de détecter des types de bugs connus avec fiabilité.
- **Limites** :
 - Nécessite un grand volume de données étiquetées.
 - Moins efficace pour découvrir des anomalies inédites.

Apprentissage non supervisé

- **Définition** : Le modèle explore les données sans étiquettes pour identifier des schémas inhabituels ou des anomalies.
- **Avantages** :
 - Utile pour détecter des bugs inconnus ou émergents.
 - Ne dépend pas d'un jeu de données annoté.
- **Limites** :
 - Moins précis, car il peut confondre des comportements inhabituels avec des bugs.
 - Interprétation des résultats plus complexe.

Conclusion

Dans le contexte de la détection automatisée de bugs, l'apprentissage supervisé est idéal pour les environnements bien documentés, tandis que l'apprentissage non supervisé est précieux pour explorer des systèmes complexes ou peu balisés. Une approche hybride peut souvent offrir les meilleurs résultats.

Question 3 : Atténuation des biais dans la personnalisation par l'IA

Pourquoi l'atténuation des biais est-elle essentielle lors de l'utilisation de l'IA pour la personnalisation de l'expérience utilisateur ?

L'intelligence artificielle permet de personnaliser l'expérience utilisateur en analysant des données comportementales, démographiques et contextuelles. Cette personnalisation peut

améliorer la pertinence des contenus, des recommandations et des interfaces. Cependant, si les données utilisées sont biaisées, l'IA risque de reproduire ou d'amplifier des discriminations existantes.

Risques liés aux biais

- **Discrimination involontaire** : certains groupes peuvent être défavorisés dans les recommandations ou les accès aux fonctionnalités.
- **Stéréotypes renforcés** : l'IA peut proposer des contenus qui enferment l'utilisateur dans des profils préconçus.
- **Perte de confiance** : les utilisateurs peuvent percevoir le système comme injuste ou opaque.

Importance de l'atténuation

Atténuer les biais permet de :

- **Garantir l'équité** : tous les utilisateurs bénéficient d'un traitement juste et équilibré.
- **Renforcer l'inclusivité** : les systèmes tiennent compte de la diversité des profils et des besoins.
- **Respecter les normes éthiques et légales** : notamment le RGPD et les principes de non-discrimination.

Conclusion

L'atténuation des biais est une condition essentielle pour que l'IA serve l'intérêt général, respecte les droits des utilisateurs et favorise une expérience personnalisée équitable et responsable.

Partie 2 : Mise en œuvre pratique

Tâche 1 : Complétion de code alimentée par l'IA

Outil utilisé : GitHub Copilot

Langage : Python

Objectif : Écrire une fonction pour trier une liste de dictionnaires par une clé spécifique, puis comparer la version générée par l'IA avec une version manuelle.

Code généré par GitHub Copilot

```
# Tri d'une liste de dictionnaires par une clé donnée

def trier_par_cle(liste, cle):

    return sorted(liste, key=lambda x: x[cle])
```

Implémentation manuelle

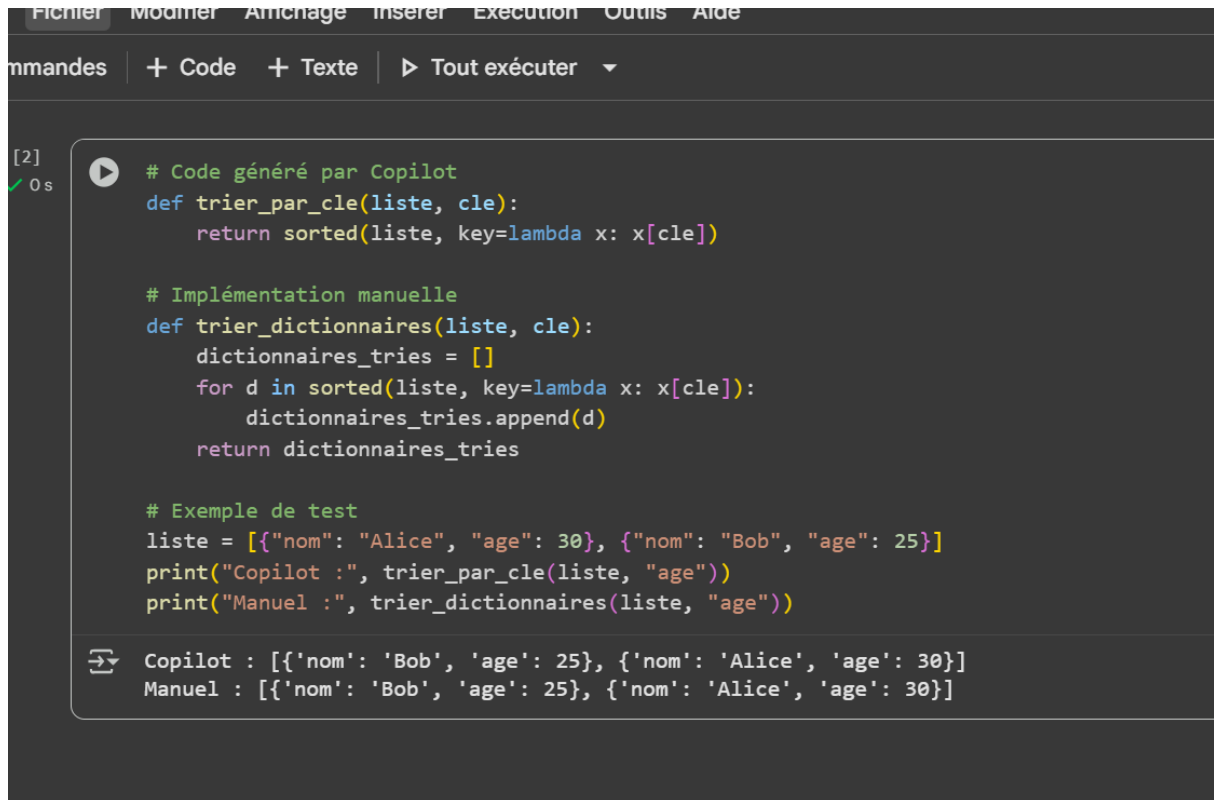
```
def trier_dictionnaires(liste, cle):
```

```
dictionnaires_tries = []

for d in sorted(liste, key=lambda x: x[cle]):

    dictionnaires_tries.append(d)

return dictionnaires_tries
```



The screenshot shows a code editor with a dark theme. At the top, there's a menu bar with 'Fichier', 'Modifier', 'Anchorage', 'Insérer', 'Execution', 'Outils', and 'Aide'. Below it is a toolbar with 'Commandes', '+ Code', '+ Texte', and '▶ Tout exécuter'. The main area displays Python code. On the left, a green checkmark and '[2] 0 s' indicate successful execution. The code is divided into three sections: a Copilot-generated function, a manual implementation, and a test example. The Copilot version is concise, using a lambda function in the sorted() call. The manual version is more verbose, using a loop to build the sorted list. The test example shows a list of dictionaries and prints the results of both functions, which are identical.

```
[2] 0 s # Code généré par Copilot
def trier_par_cle(liste, cle):
    return sorted(liste, key=lambda x: x[cle])

# Implémentation manuelle
def trier_dictionnaires(liste, cle):
    dictionnaires_tries = []
    for d in sorted(liste, key=lambda x: x[cle]):
        dictionnaires_tries.append(d)
    return dictionnaires_tries

# Exemple de test
liste = [{"nom": "Alice", "age": 30}, {"nom": "Bob", "age": 25}]
print("Copilot :", trier_par_cle(liste, "age"))
print("Manuel :", trier_dictionnaires(liste, "age"))

Copilot : [{'nom': 'Bob', 'age': 25}, {'nom': 'Alice', 'age': 30}]
Manuel : [{'nom': 'Bob', 'age': 25}, {'nom': 'Alice', 'age': 30}]
```

Comparaison entre le code généré par Copilot et l'implémentation manuelle pour trier une liste de dictionnaires.

Analyse comparative

La version générée par GitHub Copilot est concise et efficace. Elle utilise directement la fonction `sorted()` avec une expression `lambda`, ce qui est une pratique courante et idiomatique en Python. Elle respecte les standards du langage et est facilement lisible.

La version manuelle, bien que fonctionnelle, est plus verbeuse. Elle ajoute une étape intermédiaire avec une boucle `for` pour construire la liste triée, ce qui n'est pas nécessaire ici. Cela peut être utile dans des cas plus complexes, mais pour une tâche simple comme celle-ci, la version Copilot est plus élégante.

En termes de performance, les deux versions sont équivalentes, mais la version IA est plus directe et montre une bonne compréhension des conventions Python. Cela illustre bien comment l'IA peut accélérer le développement tout en respectant les bonnes pratiques.

Tâche 2 : Tests automatisés avec l'IA

Outil utilisé : Selenium IDE

Objectif : Automatiser un test de connexion avec identifiants valides et invalides

Script de test (extrait simplifié)

```
// Connexion valide
```

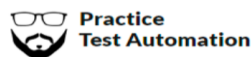
```
open | https://practicetestautomation.com/practice-test-login/
```

```
type | id=username | student
```

```
type | id=password | Password123
```

```
click | id=submit
```

```
assertText | css=.post-title | Logged In Successfully
```



DOMICILE PRATIQUE COURS BLOG CONTACT

Connexion réussie

Félicitations étudiant. Vous vous êtes connecté avec succès !

Se déconnecter

Tester la connexion

Il s'agit d'une simple page de connexion. Les étudiants peuvent utiliser cette page pour s'entraîner à écrire des tests de connexion positifs et négatifs simples. La fonctionnalité de connexion est quelque chose que la plupart des ingénieurs en automatisation des tests doivent automatiser.

Utilisez les informations d'identification suivantes pour exécuter la connexion :

Nom d'utilisateur : **étudiant**

Mot de passe : **Password123**

Nom d'utilisateur

student

Mot de passe

.....

Envoyer



Selenium IDE is
recording...



Selenium IDE is recording...

Résultat du test automatisé avec Selenium IDE – Connexion réussie avec identifiants valides.

```
// Connexion invalide

open | https://practicetestautomation.com/practice-test-login/

type | id=username | fakeuser

type | id=password | wrongpass

click | id=submit

assertText | css=.error | Your username is invalid!
```

Tester la connexion

Il s'agit d'une simple page de connexion. Les étudiants peuvent utiliser cette page pour s'entraîner à écrire des tests de connexion positifs et négatifs simples. La fonctionnalité de connexion est quelque chose que la plupart des ingénieurs en automatisation des tests doivent automatiser.

Utilisez les informations d'identification suivantes pour exécuter la connexion :

Nom d'utilisateur : **étudiant**
Mot de passe : **Password123**

Nom d'utilisateur

fakeuser

Mot de passe

.....

Envoyer



• Selenium IDE is recording..

Utilisez les informations d'identification suivantes pour exécuter la connexion :

Nom d'utilisateur : **étudiant**
Mot de passe : **Password123**

Nom d'utilisateur

Mot de passe

Envoyer

Votre nom d'utilisateur n'est pas valide !



• Selenium IDE is recording...

Résultat du test automatisé avec Selenium IDE – Connexion échouée avec identifiants incorrects.

Résumé

L'automatisation des tests avec Selenium IDE permet de simuler des scénarios utilisateur sans écrire de code complexe. Dans ce test, j'ai vérifié deux cas : une connexion réussie avec des identifiants valides, et une connexion échouée avec des identifiants incorrects. L'outil a enregistré chaque étape et les a rejouées automatiquement, ce qui garantit la reproductibilité du test.

Cette méthode est plus rapide et plus fiable que les tests manuels. Elle permet de détecter les erreurs d'interface ou de logique avant la mise en production. De plus, Selenium IDE offre une interface intuitive qui facilite la création de scénarios même pour les débutants. Ce type de test est essentiel dans les environnements DevOps où les mises à jour sont fréquentes.