

Projet Python : Création d'un BOT Discord favorisant l'apprentissage de Python

Auteurs : FEHRI MEHDI, ZELLER EMILE, SCHNEIDER HUGO







Introduction

- Objectif : Créer un outil interactif et intelligent pour l'apprentissage de Python.
- Approche : Mélanger éducation et gamification pour stimuler la motivation.
- Principe : Apprendre en relevant des défis, accumulant de l'XP, progressant en niveaux et bénéficiant d'une expérience personnalisée.

? Problématique

- Comment rendre l'apprentissage du Python plus motivant et progressif ?
- Comment suivre, enregistrer et encourager la progression individuelle des utilisateurs ?

Solution proposée

- Un Bot éducatif interactif qui propose :
 - Défis automatiques  (adaptés au niveau de l'utilisateur)
 - Suivi personnalisé d'XP et montée en niveau 
 - Mémoire individuelle de la progression 
 - Citations motivantes pour garder la forme 
- Toutes les données sont gérées et stockées de manière flexible via des fichiers JSON.

Architecture générale

Composant	Rôle principal	Détails clés
bot.py	Interface Discord, commandes slash, tâches périodiques	Async/await complet, XP & niveaux, leader-board, relances d'inactivité
ai.py	Génération IA : réponses, cours, QCM	Prompt-engineering, validation JSON, adaptations au profil
utils.py	Persistance JSON & logique métier	XP /niveaux, sélection de défis, gestion historique
Fichiers .json	Mini-base de données	users , levels , cours , citations , logs ...

➔ Cette architecture sépare clairement l'I/O Discord, l'intelligence GPT et la persistance de données, ce qui facilite la maintenance et l'évolution du projet.

Bot.py - Rôle et flux principal

- **Pont Discord ↔ OpenAI** : reçoit les slash-commands, envoie les prompts à `ai.py`, répond en embed.
- **Événements clés** : `on_member_join` (création de profil) & `on_ready` (sync commandes).
- **Commandes cœur** : `/prof`, `/stats`, `/classement`, `/citation`, `/reset`.
- **100 % asynchrone** → aucune commande ne bloque le bot.

Bot.py — Gamification & background tasks

- **XP + niveaux** : `progress_bar()` + `xp_to_next()` ; cache anti-spam `_levelup_cache` .
- **Badges auto** : `give_badge()` attribue des rôles Discord (100 / 500 / 1000 XP).
- **Tâches périodiques** :
 - `leaderboard_task` (24 h) → classement hebdo.
 - `inactivity_ping` (12 h) → DM après 72 h d'absence.
- **Robustesse** : try/except sur DM, timeout court sur appels IA.

Utils.py — Persistence & profils

- **Mini-DB JSON** : `load_json` / `save_json` (écriture atomique).
- **Gestion utilisateurs** : `create_user` , `get_user` , `update_user_xp` .
- **Historique** : `append_exercise` , `append_course` , `append_qcm` (titre + timestamp).
- **Portabilité** : basculer vers SQLite = changer uniquement ces fonctions.

Utils.py — XP, niveaux & adaptation

- **Level-up** : `check_level_up()` + `get_level_up_message()` (embed motivant).
- **Difficulté dynamique** : `get_appropriate_challenge_level()` analyse les 5 derniers défis.
- **Sélection contenu** : `get_all_cours`, `choisir_qcm`, `format_cours` → prêt à afficher.
- **Complexité** : niveaux **O(L)** ($L \approx 100$) ; historique 7 jours **O(H)** (H faible).

Ai.py - Que fait `ai.py` ?

1. Configuration & client

Charge `config.json` pour récupérer la clé et le modèle GPT, puis instancie `AsyncOpenAI` en mode asynchrone non-bloquant.

2. Définition du personnage

Un bloc **PERSONALITY** décrit Professeur Pipithon et impose un format JSON strict à chaque réponse.

3. Génération de contenu pédagogique

- `generer_cours_complet()` : crée un prompt (chapitre, niveau) → renvoie cours + défi + QCM en < 45 s.
- `creer_cours()` : même logique pour un mini-cours express.

4. Réponse principale – `professeur_repond()`

- Récupère le profil (XP, niveau, historique).

Ai.py - Comment avons-nous construit son intelligence ?

Pilier	Mise en œuvre	Bénéfice
Prompt-engineering strict	Persona + exemples + schéma JSON imposé ; validation par <code>deep_merge()</code>	Réponses homogènes et parseables
Contexte adaptatif	Injection dynamique du niveau, XP, historique + taux de réussite	Personnalisation fine, motivation constante
Réglages de créativité	<code>temperature</code> 0.5 (cours) / 0.6 (réponses), <code>timeout</code> court	Style cohérent, latence maîtrisée
Asynchronicité	<code>AsyncOpenAI</code> + <code>await</code> partout	Concurrence élevée sans blocage
Résilience & logs	<code>try/except</code> , traces, test <code>_test()</code>	Débogage rapide, fiabilité

Extrait Code – Appel OpenAI asynchrone (ai.py)

```
async def generer_cours_complet(chapitre: str, level: int) -> dict | None:
    prompt = COURSE_FULL_PERSONALITY.format(level=level) + f"\nChapitre : {chapitre}"
    rsp = await client.chat.completions.create(    # ← async / await
        model=GPT_MODEL,
        messages=[{"role": "user", "content": prompt}],
        temperature=0.5,
        timeout=45,
    )
    return json.loads(rsp.choices[0].message.content.strip())
```

- Utilisation de AsyncOpenAI pour ne pas bloquer le bot.
- Prompt = template dynamique (chapitre + niveau) → contenu ciblé.
- Parse direct du JSON renvoyé : pas de string-post-processing.

Extrait Code – Difficulté adaptative (utils.py)

```
def get_appropriate_challenge_level(user_id):
    base = get_user(str(user_id))["level"]
    recent = get_user(str(user_id)).get("completed_challenges", [])[-5:]
    success = sum(c["success"] for c in recent) / max(1, len(recent))
    if success > 0.8:
        return base + 1          # on hausse le niveau
    if success < 0.3 and base > 1:
        return base - 1          # on baisse pour ne pas décourager
    return base
```

- apport pédagogique : montre un algorithme simple $O(1)$ qui adapte la difficulté à la performance récente et maintient l'élève dans sa zone d'apprentissage.



Utilisation des fichiers JSON

- `users.json` : sauvegarde permanente de l'expérience, du niveau et de l'historique utilisateur.
- `levels.json` : structure des niveaux et paliers d'expérience.
- `logs.json` : enregistrement complet de toutes les actions effectuées par les utilisateurs.
- `messages.json` : messages standards (réponses aux QCM, cours, défis).
- `xp_rules.json` : règles précises de gain et perte d'XP.
- `citations.json` : citations motivantes et humoristiques envoyées de façon dynamique.





Système XP et niveaux

- Chaque action (défi, QCM, cours) rapporte de l'XP calculé selon les règles dynamiques.
- Le franchissement d'un palier d'XP attribue automatiquement un nouveau niveau et un titre motivant.

```
{ "niveau": 10, "xp": 250, "titre": "🔥 Initié des Boucles" }
```

- Objectif : maintenir l'engagement par une progression visible et gratifiante.

Points forts du projet

-  Mémoire individuelle complète : XP, niveaux, historique sauvegardés à chaque interaction.
-  Adaptation automatique des défis au niveau de l'utilisateur grâce à l'IA.
-  Progression ludique et gratifiante avec un système complet de niveaux et titres personnalisés.
-  Utilisation efficace de fichiers JSON pour séparer données et logique de code.

Une petite démo !

Limites identifiées

- **Dépendance à l'API OpenAI** : seuils de quota, coûts variables et nécessité d'une connexion internet stable.
- **JSON comme stockage** : pratique pour un POC, mais pas adapté à >10 000 utilisateurs ni aux accès concurrents.
- **Sécurité minimale** : pas de chiffrement des données ni de vérification anti-spam sur les inputs utilisateurs.
- **Gestion d'erreurs encore perfectible** : certains `try/except` génériques qui masquent la vraie cause des bugs.
- **Pas d'internationalisation** : le bot ne parle que français pour l'instant.
- **Interface Discord seulement** : aucune API publique ni appli mobile / web autonome.



Perspectives d'amélioration

- Implémentation complète de cours détaillés et interactifs.
- Amélioration de l'IA pour générer des défis plus complets.
- Système de badges, trophées et récompenses spéciales.
- Création de mini-jeux éducatifs complémentaires.
- Ajout de fonctionnalités supplémentaires (reset profile etc...)



Conclusion

- Notre projet propose une app d'apprentissage motivante, progressive et mémorielle.
- Il combine un suivi personnalisé, des défis adaptatifs, une progression gamifiée et une architecture souple.
- Il ouvre la voie à de nombreux développements futurs pour enrichir encore l'expérience utilisateur.

? Questions

Merci pour votre attention ! 

N'hésitez pas si vous voulez tester notre plateforme et relever un premier défi !