Étapes clés : Développez un programme logiciel en Python

Dans ce document, vous trouverez un exemple des étapes clés à suivre pour mener à bien votre projet. Vous découvrirez :

- Des recommandations pour la réalisation de chaque étape ;
- Des erreurs à éviter ;
- Les ressources externes utiles pour chaque étape.

Ces étapes ne sont que des suggestions. Notez que :

- L'avancement du projet à chaque étape n'est qu'une ligne directrice et varie en fonction de votre rythme.
- Les livrables de l'étape font référence aux résultats attendus à chaque étape. Vous n'êtes pas obligé de réaliser tous ces livrables suggérés pour valider le projet, vous ne serez évalué que sur les livrables finaux.

Donnez-nous votre avis ! Ce document est une nouvelle ressource pour vous aider à faire votre projet. Remplissez ce formulaire de 3 minutes si vous souhaitez nous donner votre retour pour qu'on puisse l'améliorer.

Étape 1 : Se familiariser avec les classes et la programmation orientée objet

15 % de progression

🎯 **Cette étape n'est pas directement liée à un livrable du projet, mais elle vous sera très utile par la suite !** La meilleure façon de se familiariser avec la programmation orientée objet (POO), c'est de pratiquer. Dans cette étape, vous pouvez recommencer une tâche que vous avez déjà terminé sans la POO.

📌 **Recommandations :** Regardez à nouveau les consignes pour le deuxième projet du parcours. Recodez ce projet à partir de zéro, mais cette fois, réfléchissez en termes « d'objets ». Vous voudrez probablement penser à des entités telles qu'un `Livre`, une `Catégorie`, ou peut-être même une `RessourceEnLigne`.

C'est également le moment idéal pour suivre les cours recommandés pour ce projet : « Apprenez la programmation orientée objet avec Python » et « Écrivez du code Python maintenable ».

⚠️ **Les erreurs à éviter :**

- Vous serez peut-être tenté de réutiliser le code de votre projet précédent. Ne faites pas de copier-coller, mais réécrivez-le. Maintenant que vous êtes plus avancé en Python, il y a certainement des améliorations à apporter !
- Si vos livrables pour le deuxième projet sont sur GitHub, assurez-vous de créer une nouvelle branche, ou un nouveau repository pour cette tâche. Ne modifiez pas vos livrables originaux !

📃 **Ressources :**

- Documentation officielle de Python : classes et objets
- Les bases de la programmation orientée objet (RealPython) (en anglais)

Étape 2 : Définir (et coder) les modèles pour ce projet

35 % de progression

🎯 **Livrable de l'étape :** Code pour les modèles.

📌 **Recommandations :** Dans ce projet, vous devez utiliser le modèle « MVC » (Models, Views, Controllers). Le principal avantage de cette approche est qu'elle met en œuvre le principe de « séparation des responsabilités ». Chaque composant/classe est indépendant et responsable de sa propre tâche bien définie (contenir des données/attributs sur les entités, afficher quelque chose à l'écran, gérer la logique du programme...).

Cependant, mettre en œuvre ceci à partir de zéro peut être décourageant la première fois. Le meilleur moyen pour y parvenir est de commencer par les **modèles** : avec quelles entités votre programme va-t-il fonctionner ? Doivent-elles contenir des données (= attributs) ? Appliquent-elles des comportements spécifiques (= méthodes) ?

Pour commencer, votre programme s'appuiera (au minimum) sur des **tournois**, des **tours**, des **matchs** et des **joueurs**.

- Quels sont les attributs requis pour chacune de ces entités ?
- Quels sont les comportements de ces entités ? Par exemple, comment les joueurs sont-ils associés lors des tours ?

⚠️ **Les erreurs à éviter :**

- L'algorithme suisse permettant d'associer les joueurs peut être compliqué à mettre en œuvre correctement. Il est probable que vous vous retrouviez avec des correspondances non optimales. Ce n'est pas l'objectif principal du projet, faites au mieux !
- N'essayez pas encore de mettre en œuvre une conception MVC complète ! Commencez simplement avec vos modèles et assurez-vous qu'ils fonctionnent comme prévu. Vous pouvez utiliser des « données statiques » ou générer des valeurs de manière aléatoire pour une expérience plus authentique.
- Assurez-vous que vos modèles sont bien séparés. Vous pouvez écrire de petits « scripts » ou des programmes qui importent les modèles requis, créent quelques instances (voir ci-dessus), les font interagir et affichent les résultats. Exemple :
    - créer quelques joueurs ;
    - créer un tour, ajouter les joueurs ;
    - voir comment les joueurs sont associés (classements) ;
    - gagner/perdre des matchs de manière aléatoire ;
    - vérifier que les données des modèles sont correctement mises à jour.
- Ça pourrait être le bon moment pour vous familiariser avec les tests unitaires. Mais ce n'est pas requis dans les livrables du projet.

📃 **Ressources (en anglais) :**

- La POO en Python

Étape 3 : Mettre en œuvre la conception MVC

70 % de progression

🎯 **Livrable de l'étape :** Vues et contrôleurs pour le code.

📌 **Recommandations :** Maintenant que vos modèles sont fonctionnels, complets et testés, vous pouvez commencer à travailler sur les autres composants : les contrôleurs et les vues. Les contrôleurs sont responsables de la logique du programme et les vues sont responsables de l'affichage (et en partie, de la réception) des données à destination (ou en provenance) de l'utilisateur.

- Dans ce programme, les vues s'afficheront sur la console. Même si elles utiliseront principalement des `print` (et peut-être des `input`) pour traiter les données, il est important de les garder séparées pour respecter la distinction des responsabilités dans la conception MVC. C'est le principe de la « responsabilité unique » : chaque composant ne doit avoir qu'un seul rôle à jouer dans le code.
- Les contrôleurs peuvent être imbriqués.
    - Vous pouvez avoir un contrôleur « Application », qui instanciera :
        - un contrôleur `MenuManager`;
        - un contrôleur `TournamentManager`;
        - un contrôleur `UserManager`.
    - Ce ne sont là que des exemples ! C'est votre projet et vous pouvez choisir les contrôleurs à utiliser.
- En plus de la conception MVC, vous pouvez vous familiariser (et utiliser) d'autres modèles de conception. Pour les interfaces utilisateur, il est courant d'utiliser le modèle `Command` et parfois le modèle `State` (pour les menus).

⚠️ **Les erreurs à éviter :**

- Essayez de « faire simple ».
- Vous verrez peut-être des débats sur la question de savoir si une vue doit ou non être capable de « recevoir des données de l'utilisateur » ou seulement « d'afficher des données à l'utilisateur ». Chacun son point de vue ! Les modèles de conception sont des directives générales et chaque équipe ou projet peut avoir sa propre interprétation/mise en œuvre.
- Le plus important dans ce projet est de comprendre le principe de la « responsabilité unique ». Votre conception MVC n'est peut-être pas parfaite, mais elle reste acceptable. Vérifiez auprès de votre mentor !

📃 **Ressources :**

- Structurer votre code Python
- Mise en page des applications en Python (en anglais)

Étape 4 : Implémenter la sérialisation

85 % de progression

🎯 **Livrable de l'étape :** Framework TinyDB pour le code

📌 **Recommandations :** Votre projet devrait maintenant être en grande partie fonctionnel. Ce qui lui manque, c'est la possibilité de lire et d'enregistrer des données de manière continue. Vous utiliserez le framework TinyDB, qui est une base de données persistante de type JSON. N'hésitez pas à en apprendre davantage sur JSON et la persistance des données, sur lesquels TinyDB s'appuie et que vous continuerez à utiliser.

- Si vos modèles n'en disposent pas encore, mettez en place un moyen de créer des instances à partir d'un « dictionnaire » ou de données orientées texte.
- Implémentez une méthode `save` sur vos modèles, qui sérialise tous les attributs de vos entités.
- Connectez ces deux méthodes à la base de données.
- Essayez de conserver une approche orientée objet lorsque vous traitez avec la base de données !

⚠️ **Les erreurs à éviter :**

- Essayez de vous assurer que vous ne lisez ou n'écrivez pas trop depuis/vers la base de données. N'apportez des modifications que lorsque cela est nécessaire !

📃 **Ressources (en anglais) :**

- Documentation officielle de TinyDB
- Repository GitHub pour TinyDB

Etape 5 : Faites une pause et nettoyez tout !

100 % de progression

🎯 **Livrable de l'étape :** Le code final.

📌 **Recommandations :** Votre projet devrait maintenant être entièrement conforme au cahier des charges. Relisez votre code, prenez le temps de vous assurer qu'il répond aux meilleures pratiques (docstrings, PEP8, etc.).

- Assurez-vous que votre code est bien organisé. Il semble logique d'utiliser des packages tels que `views`, `controllers` et `models`. Ceux-ci devraient à leur tour contenir des modules Python, généralement un pour chacune de vos classes.
- Vous aurez besoin de mettre en place des outils pour dégrossir et embellir votre code : `Flake8`, `Black` ou `Isort`.
- Vous voudrez peut-être publier votre code sur GitHub, assurez-vous qu'il comporte un fichier README contenant les informations explicatives sur votre programme.

⚠️ **Les erreurs à éviter :**

- Ne laissez pas de fichier dépassé ou de gros morceaux de code commentés ! Prenez le temps de nettoyer votre projet et de lui donner un aspect professionnel.
- Le test complet de votre programme peut prendre du temps. Il est important de vous assurer qu'il fonctionne à 100 % comme prévu. Vous pouvez profiter de la construction `if __name__ == "__main__"` pour exécuter indépendamment le code de vos modules.

📃 **Ressources (en anglais) :**

- Python PEP-8
- Documentation de Flake 8
- Documentation de Black
- Formatage et nettoyage dans Visual Studio Code

🎉 Projet terminé !