Guide d’étapes clés : Développer un programme logiciel en utilisant Python

Comment utiliser ce document ?

Ce guide vous propose un découpage du projet en étapes. Vous pouvez suivre ces étapes selon vos besoins. Dans chacune, vous trouverez :

- des recommandations pour compléter la mission ;
- les points de vigilance à garder en tête ;
- une estimation de votre avancement sur l’ensemble du projet (attention, celui-ci peut varier d’un apprenant à l’autre).

Suivre ce guide vous permettra ainsi :

- d’organiser votre temps ;
- de gagner en autonomie ;
- d’utiliser les cours et ressources de façon efficace ;
- de mobiliser une méthodologie professionnelle que vous pourrez réutiliser.

Gardez en tête que votre progression sur les étapes n’est qu’une estimation, et sera différente selon votre vitesse de progression.

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

15% de progression

Avant de démarrer cette étape, je dois avoir :

- lu tout l’énoncé du projet et le document des spécifications techniques ;
- regardé les cours recommandés du projet, notamment :
    - Apprenez la programmation orientée objet avec Python et Écrivez du code Python maintenable.

Une fois cette étape terminée, je devrais avoir :

- une meilleure compréhension de l'utilisation de la programmation orientée objet (POO) dans une application Python.
- codé une version en POO d'un programme déjà codé sans POO.

Recommandations :

- La meilleure façon de se familiariser avec la programmation orientée objet (POO), c'est de pratiquer. Dans cette étape, nous vous recommandons fortement de recommencer un projet que vous avez déjà terminé sans la POO.
- Regardez à nouveau les consignes pour le projet Utilisez les bases de Python pour l'analyse de marché . 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 ».
    - Il n'est pas nécessaire de compléter tous les cours avant de démarrer le projet ! Vous pouvez consulter les chapitres pertinents des cours au fur et à mesure.

Points de vigilance :

- 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 projet Utilisez les bases de Python pour l'analyse du marché 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

40% de progression

Avant de démarrer cette étape, je dois avoir :

- revu les spécifications techniques ;
- compris la conception MVC, notamment le rôle des modèles.

Une fois cette étape terminée, je devrais avoir :

- défini les modèles requis pour le projet ;
- écrit le code pour les modèles.

Recommandations :

- Dans ce projet, vous devez utiliser le patron« 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) ?
    - Possèdent-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 ?
- L'application chargera les données du tournoi à partir d'un fichier JSON.
- Les données des joueurs seront disponibles dans le fichier JSON du tournoi, mais vous devrez les associer avec les autres données disponibles dans les fichiers JSON des clubs.
    - Gardez en tête l'identifiant national d'échecs (dans les spécifications techniques).

Points de vigilance :

- En associant les joueurs par votre algorithme, 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 tout de suite 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 (points) ;
    - 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 :

- La POO en Python (rédigé en anglais)

Étape 3 : Mettre en œuvre la conception MVC et itérer

80% de progression

Avant de démarrer cette étape, je dois avoir :

- codé tous les modèles ;
- compris les rôles des vues et des contrôleurs dans la conception MVC.

Une fois cette étape terminée, je devrais avoir :

- codé une application (contenant toutes les **vues** et tous les **contrôleurs**) qui fonctionne bien :
    - en gérant les données ;
    - en suivant les tournois ;
    - en générant les rapports à voir.

Recommandations :

- Maintenant que vos modèles sont fonctionnels, complets et testés, vous pouvez commencer à travailler sur les autres composants : les **vues** et les **contrôleurs**.
    - Les vues affichent des données à l'utilisateur.
    - Les contrôleurs gèrent la logique du programme : ils créent et modifient les données.
- Dans ce programme, les vues s'afficheront sur la console. Même si elles utilisent principalement des `print` (et peut-être des `input`) pour traiter les données, il est important de les garder séparées des modèles et contrôleurs pour respecter la séparation 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.

        - En plus d'afficher des données, les vues peuvent également en capturer en utilisant `input` pour demander des informations à l'utilisateur.
- 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).
- Assurez-vous que l'application sauvegarde les données dans des fichiers JSON dès qu'une information est saisie ou est modifiée. Vous aurez besoin des méthodes de sérialisation sur les instances (par exemple, pour les dates ou les structures de données complexes).
    - 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.

Points de vigilance :

- Essayez de « faire simple ».
- N'essayez pas d'implémenter toutes les fonctionnalités en une seule fois. **Itérez** plusieurs cycles de développement. D'habitude, les fonctionnalités ont deux composants associés : la vue et le contrôleur. Par exemple :
    - *rechercher tous les tournois des fichiers* vs. *afficher une liste de tous les tournois disponibles dans le terminal et permettre à l'utilisateur d'en sélectionner un ;*
    - *charger un tournoi d'un fichier* vs. *afficher des informations sur un tournoi dans le terminal*.
- Vous aurez peut-être des questions pour 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 (rédigé en anglais)

Étape 4 : Faire une pause et nettoyer tout !

100% de progression

Avant de démarrer cette étape, je dois avoir :
- L'application fonctionnelle qui assure la persistance de données.

Une fois cette étape terminée, je devrais avoir :
- Le repository final et nettoyé qui inclut un fichier README.md.

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 nettoyer 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.

Points de vigilance :
- Ne laissez pas de fichiers inutiles ou de gros morceaux de code commentés ! Prenez le temps de nettoyer votre projet et de lui donner un aspect professionnel.
- N'oubliez pas le fichier __init__.py dans les packages !
- 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 (toutes rédigées en anglais) :
- Python PEP-8
- Documentation de Flake 8
- Documentation de Black
- Formatage et nettoyage dans Visual Studio Code

Projet terminé !