Projets en Programmation Python

Consignes générales 2023-2024

1 Présentation

Ce projet permet à l'étudiant de réaliser un programme Python en passant par toutes les étapes de son cycle de vie : spécifications, analyse et conception, codage, vérification et maintenance. Il doit montrer que celui-ci est capable d'entrer dans la réalisation d'une application ambitieuse, bien structurée et maîtrisée qui pourra ensuite être utilisée voire modifiée par d'autres.

Le sujet du projet est constitué des TD 3 à 10, à l'exception peut-être du TD 8 car il n'est pas obligatoire de rendre une interface en Dash. Le rendu est demandé en 3 versions qui correspondent, chacune, à une évolution du projet (voir ci-dessous).

2 Date limite et rendu

Le rapport et le programme Python doivent être rendus le **10 janvier 2024**. Le détail du rapport est précisé ci-dessous ; il doit impérativement comporter l'adresse du dépôt d'où peut être téléchargé le code de l'application. Le rapport doit être déposé sur la page moodle de l'enseignement par chaque étudiant du binôme.

3 Eléments à fournir :

1. Un **rapport** (format .pdf uniquement, d'une taille inférieure à 5Mo). Celui-ci rappelle la problématique du projet et en donne les spécifications. Il détaille ensuite l'analyse effectuée par les étudiants et comment celle-ci est implémentée (conception). Un soin particulier doit être pris pour expliquer comment les diverses tâches ont été réparties entre les différents étudiants travaillant sur le projet. En effet, au moment de l'évaluation, l'examinateur se réserve le droit d'interroger chaque étudiant individuellement sur le projet et de prendre en compte les réponses pour attribuer une note individuelle. Il est très important également de rappeler que ce rapport **ne doit pas** comporter les lignes de codes du logiciel. Enfin, le rapport doit absolument présenter les **tests** effectués, une conclusion et des perspectives. Les différents éléments du rapport sont détaillés dans la section 4.
2. Le lien du **dépôt Github** contenant le code source doit être clairement mentionné dans le rapport. Il comprendra l'ensemble des fichiers nécessaires à l'exécution du programme. Le code doit être correctement présenté (indentation), clair (utilisation de noms lisibles) et commenté (intelligemment). Les librairies éventuelles doivent également être précisées afin de pouvoir facilement exécuter le programme. Le bon fonctionnement du programme sera jugé sur une version 3.10 du langage Python. Trois versions du code doivent être fournies, si possible via le système de *tags* de github :
   — **v**1 : version qui correspond aux TDs 3 à 5 (socle de base de l'application)
   — **v**2 : version qui correspond aux TDs 3 à 7 (avec le moteur de recherche)
   — **v**3 : version qui correspond aux TDs 3 à 7 + TD 9-10 (avec l'interface)

4 Le rapport :

Le rapport doit comporter :

1. Les spécifications : partie normalement rédigée conjointement avec le client qui comprend une description en langage naturel de ce que doit faire le programme. Dans notre cas, les spécifications seront rédigées à partir des énoncés et des questions posées à l'enseignant.
2. L'analyse : cette partie explique comment les spécifications vont être réalisées. Elle comporte :

- quel sera l'environnement de travail (vous pouvez justifier le choix de votre environnement) ?
- quelles sont les données identifiées dans les spécifications ?
- un diagramme des classes (simplifié bien sûr), c'est-à-dire :
  — préciser les classes utilisées (nom et méthodes principales),
  — préciser les relations d'héritage entre classes,
  — préciser les relations d'utilisation entre les classes,

3. La conception : cette partie explique comment l'analyse est implémentée, c'est-à-dire :
   - préciser clairement la manière dont vous les tâches ont été partagées entre les différents étudiants travaillant sur le même projet,
   - détailler les algorithmes spécifiques à l'application et d'une certaine complexité (si nécessaire),
   - détailler les problèmes rencontrés, comment vous les avez résolus, contournés ou non.
   - donnez un exemple commenté d'utilisation du programme du début à la fin.
4. La validation : spécifier les procédures de test de votre logiciel, c'est-à-dire :
   - tests unitaires : comment les méthodes ont été testées individuellement ? Donnez quelques résultats ;
   - test globaux : en utilisant votre interface, quels cas particuliers avez-vous testés ?
5. La maintenance : donnez quelques évolutions possibles du logiciel. Quelles sont les fonctionnalités que l'on peut rajouter et comment ? Pourquoi cela sera-t-il facile ou difficile ?

5 Notation du projet :

La notation du projet tiendra compte des éléments suivant :

1. Fonctionnement général du programme (6 pts) :
   - état d'avancement du projet : jusqu'où ont été réalisées les spécifications sans bugs (étant entendu par bugs tout plantage inopiné du programme).
   - amélioration du projet : quels sont les "plus" apportés aux spécifications initiales (pris en compte seulement si les spécifications ont d'abord été réalisées).
   - les codes non valides (dans le cas où le programme ne marche pas ou mal), s'ils sont identifiés et commentés comme tels.
2. Code (8 pts) :
   - présentation générale du code (ex. indentation, séparation des différentes classes),
   - commentaire : en lisant les commentaires, on doit pouvoir comprendre le programme.
   - qualité : qualité/efficacité du code écrit vs. tâche réalisée (importance des tests).
3. Dossier (6 pts) :
   - analyse : qualité du choix des classes, prise en compte des mécanismes objets, qualité des algorithmes du programme (mis en avant ici),
   - conception : correspondance avec la réalité, qualité de l'interface,
   - présence de chaque partie avec la mise en avant des informations pertinentes,
   - parties supplémentaires (tests et maintenance).

6 Remarques complémentaires :

- Il faut respecter le caractère multi-plateforme de Python. Aussi, veillez à ne pas utiliser d'éléments spécifiques à un certain OS (ex. : ne pas utiliser d'adresse absolue comme "C :\...\...").
- Pensez à fournir le nom des librairies nécessaires à l'exécution de votre programme (vous pouvez générer un fichier de "requirements" : `https://pip.pypa.io/en/latest/user_guide/#requirements-files`).
- Il est *nettement* préférable que les données (en volume raisonnable) se trouvent également sur le répertoire (export de la table des données) afin de ne pas avoir à re-télécharger les données avec les API.
- Il faut à minima proposer une petite interface textuelle pour interagir avec la base d'articles, ce qui peut être fait avec un carnet de note (*notebook*) ou avec une librairie plus ambitieuse (comme `dash`, mais il y en a d'autres). **Attention :** il est interdit de rendre *tout* le code sous forme de notebook ! Cela doit être réservé à l'interface.

- Un programme qui fonctionne mais qui fait peu de choses sera mieux noté qu'un programme qui ne fonctionne pas du tout (par définition qui ne fait rien). Autrement dit : l'étendue des fonctionnalités réalisables par l'interface (et qui fonctionne) est un critère important.
- L'utilisation du code d'autres auteurs (et libre de droit) est autorisée dans la mesure où : a) vous signalez très clairement la source de ce code (sous peine d'être accusé de plagiat, ce qui amène automatiquement à une note de 0/20, à minima), b) cela ne représente qu'une partie du code original que vous produisez pour réaliser ce projet.
  Cette remarque vaut également pour l'utilisation de ChatGPT, et autres solutions de ce type comme Github Copilot. Ces solutions sont autorisées du moment que vous signalez comment et où vous les avez utilisées.

Bonne chance !