Outils de programmation et gestion de versions
Examen - Projet Python avec Git et Documentation

Kévin Michoud
kmichoud@unistra.fr

Date limite : Vendredi 10 janvier 2026

Projet individuel ou en binôme
Tous documents autorisés, usage de l'IA autorisé

Présentation générale

L'objectif de cet examen est de démontrer votre maîtrise des outils et bonnes pratiques vus en cours :

- Utilisation de **Git** avec des commits logiques et organisés
- Écriture de **tests unitaires** avec pytest
- **Documentation** du code avec docstrings et génération automatique via pdoc
- **Type hints** pour clarifier les signatures de fonctions
- Choix d'une **licence** appropriée

Le projet peut être réalisé **individuellement ou en binôme**.

Choix du projet

Vous êtes libres de choisir votre sujet de projet. Voici quelques suggestions :

- Une petite application web avec **Streamlit**
- Un outil en ligne de commande (CLI) avec **argparse**
- Un petit projet de **machine learning** (analyse de données, prédiction simple, etc.)
- Tout autre projet qui vous intéresse

Important :

- Le code en lui-même n'a pas besoin d'être massif. La qualité et l'organisation priment sur la quantité.
- Il est recommandé (mais pas obligatoire) de me consulter une fois que vous avez choisi votre sujet.
- Le code doit être écrit en **anglais** (noms de variables, fonctions, classes, etc.)
- Les commentaires, documentation et explications peuvent être en **français ou en anglais**

Inspiration : Vous pouvez vous inspirer de ce dépôt pour voir un exemple de structure :
`https://github.com/KnuxV/strong_password_generator`
C'est suffisant niveau quantité de code mais bien-sûr vous pouvez en faire plus.

Critères d'évaluation

1. Utilisation de Git (4 points)

Ce que j'attends :

— Des commits **logiques et thématiques** avec des messages clairs

× Mauvais exemple : `git add . && git commit -m "up" && git push`

Exigences minimales :

— Au moins **une branche secondaire** avec un **merge** dans la branche principale

— Si vous travaillez à **deux**, au moins une **pull request** doit être créée et fusionnée

Partage du projet :

— Ajoutez la deuxième personne à votre dépôt en tant que collaboratrice

— Sur **GitHub** : Settings → Collaborators → Ajouter

— Sur **GitLab UniStra** : Settings → Members → Ajouter mon compte

— Ajoutez-moi également, ou partagez-moi le projet par URL via email

2. Documentation et Type Hints (4 points)

Docstrings :

— Tous les **scripts, fonctions, classes et méthodes** doivent avoir une docstring (en français ou anglais)

— La docstring doit expliquer ce que fait le code

— **Astuce** : Vous pouvez utiliser l'IA (ChatGPT, Claude, Copilot, etc.) pour générer les docstrings et gagner du temps. N'oubliez pas de les relire et de les adapter si nécessaire.

Type hints :

— Obligatoire pour toutes les **fonctions et méthodes**

— Indiquer clairement les types en entrée et en sortie

— Optionnel pour les variables (si vous pensez que c'est pertinent)

Génération de documentation avec pdoc :
Vous avez deux options :
Option 1 - Documentation locale :

```
pdoc --html --output-dir docs/ votre_module/
```

Option 2 - Documentation en ligne (recommandé - bonus) :

— Utilisez **GitHub Actions** pour déployer automatiquement votre documentation

— Suivez le guide officiel : `https://pdoc.dev/docs/pdoc.html#deploying-to-github-pages`

3. Tests avec pytest (4 points)

(pas besoin de les mettre dans la documentation)

Organisation des tests (suggestion) :

— Créez un dossier `tests/` à la racine de votre projet

— Ajoutez un fichier `__init__.py` dans ce dossier (pour en faire un package Python)

— Créez des scripts de test thématiques (par exemple : `test_validation.py`, `test_utils.py`, etc.)

Nombre de tests :

— Le nombre exact est libre, mais l'objectif est de montrer que vous comprenez le concept

— Testez au moins quelques cas standards et quelques cas limites (edge cases)

Exécution des tests :

```
pytest
```

Bonus :

— Automatisez l'exécution des tests avec **GitHub Actions**

— Utilisez cette action : `https://github.com/marketplace/actions/run-pytest`

4. Licence (1 point)

Choisissez une licence pour votre projet :

— Consultez `https://choosealicense.com/` pour vous aider à choisir

— Ajoutez un fichier `LICENSE.txt` ou `LICENSE` à la racine de votre dépôt

— Justifiez brièvement votre choix dans le document de justifications (voir ci-dessous)

5. README (1 point)

Créez un fichier **README.md** (en français ou anglais) qui contient :

— Une **description** de votre projet : qu'est-ce qu'il fait ?

— Les **instructions d'installation** (dépendances, packages à installer)

— Les **instructions d'utilisation** (comment lancer le programme)

— Toute autre information pertinente pour comprendre et utiliser votre projet

6. Justifications (3 points)

Créez un document appelé `justifications.md` (ou `.pdf` si vous préférez) de **1 à 2 pages** (plus si nécessaire).

Ce document doit contenir :

- **Git** : Comment avez-vous organisé vos commits ? Avez-vous rencontré des difficultés avec les branches/merges ?
- **Tests** : Quels cas de test avez-vous choisis ? Avez-vous identifié des edge cases particuliers ?
- **Documentation et type hints** : Y a-t-il des choix particuliers que vous souhaitez justifier ?
- **Licence** : Pourquoi avez-vous choisi cette licence pour votre projet ?
- **Utilisation de l'IA** : Comment avez-vous utilisé l'IA pour ce projet ?
  - Quels outils avez-vous utilisés ? (ChatGPT, Claude, Copilot, etc.)
  - Pour quelles parties ? (génération de code, docstrings, tests, débogage, justifications, etc.)
  - Donnez un exemple concret où l'IA vous a aidé
  - Avez-vous dû corriger ou adapter ce que l'IA a généré ? Si oui, donnez un exemple

Important :

- Vous pouvez utiliser l'IA pour vous aider sur toutes les parties du projet, mais vous devez **comprendre** ce que vous écrivez et en être **responsable**
- Si il est clair que vous avez utilisé l'IA sans comprendre, la moindre erreur sera comptabilisée
- Soyez honnête sur votre utilisation de l'IA dans les justifications
- Soyez concis et pertinent. Je ne veux pas de kilomètres de texte, juste des explications claires.

Barème récapitulatif

Critère	Points
Code	3
Git (commits, branches, merge/PR)	4
Documentation (docstrings, type hints, pdoc)	4
Tests (pytest, organisation, pertinence)	4
Licence	1
README	1
Justifications (dont utilisation IA)	3
Total	**20**

Bonus :

— Déploiement de la documentation via GitHub Actions

— Automatisation des tests via GitHub Actions

— Qualité et originalité du projet

Conseils

— **Commitez régulièrement** : ne faites pas tout d'un coup à la fin

— **Testez votre code** : écrivez les tests au fur et à mesure

— **Soyez honnête** sur votre utilisation de l'IA dans les justifications

— **N'hésitez pas à me poser des questions** si vous êtes bloqué.e, je répondrais avec plaisir