Objectifs de la feuille

- Analyse
- Formatage
- Nettoyage
- Refactoring
- Fusion

Présentation

Ce projet a pour but à la fois de pratiquer/utiliser Git et de coder proprement à l'aide d'un exemple d'application. Pour Git, il ne nous faut pas forcément de connaître toutes les commandes de Git (voir le document git-cmds-base.pdf sur CELENE). Il faut surtout connaître les concepts et savoir à quel moment il faut exécuter une action précise.
Pour ce qui est du code, il vous faut savoir comment coder proprement, savoir utiliser les outils pour formater automatiquement le code, pour chercher des problèmes de code et les fixer.

Création d'un compte GitHub

GitHub est le site principal fournissant des services d'hébergement de dépôt. GitHub est un service en ligne qui offre beaucoup de fonctionnalités :

- Stockage de dépôts en ligne
- Visualisation du dépôt
- Modification en direct

Vous devez créer un compte GitHub (si vous ne l'avez pas encore) et utiliser le vrai nom prénom pour ce compte car le TP est évalué directement dans les dépôts GitHub. Github ne permet pas d'utiliser un mot de passe pour travailler avec les dépôts, vous devrez créer des jetons d'accès personnels. Veuillez suivre les instructions du fichier personal_access_token.doc sur CELENE.

Contexte du projet

Nous allons vous fournir le code de base d'un jeu de Tetris. Ce code de base n'est pas complet et il y a de nombreux problèmes de la qualité. Vous devez compléter le code et améliorer la qualité du code dans les étapes suivantes.
Ce TP noté correspond à 30% de la note finale. La note est basé sur tous les commits que vous allez réaliser. Alors pour avoir une bonne note il faut bien attention sur les points suivants :

- Le message d'un commit doit être très clair, précis mais il ne faut pas trop long.
- Les commits pendant les séances de TP sont mieux notés. Les commits en dehors de ces heures ont un malus de 20%.
- Avoir les docstring dans tous les fichiers, les classes et les fonctions importantes.
- Avoir les commentaires au cas où vous ne pouvez pas expliquer le but par le code soi-même.
- Les outils Linter (pylint) et Formater (black) ne trouve pas de problèmes dans votre code.

Le projet est évalué directement sur le dépôt Git où vous ferez des commits, pendant les séances de TP. Nous vous offrons trois jours de retard gratuits : les commits passés ce délai de 72h après la dernière séance du projet ne sont plus pris en compte.

Vous pouvez travailler ensemble sur le débogage du programme, collaborer, relire le cours. Cependant les commits de code doivent être personnels. Le partage volontaire de votre travail constitue une violation du règlement et le risque est la note 0.

Récupérer le code du projet

Cliquer sur ce lien et suivre les instructions : https://classroom.github.com/a/P-T6spg9

Début du développement

Cloner le dépôt distant sur votre machine.
Il est recommandé d'utiliser Visual Studio Code pour éditer le code.
Vous aurez besoin d'installer pygame : `pip install pygame`
Créer le fichier readme.md pour ajouter vos informations, et commiter les modifications au dépôt distant.
Il faut utiliser le langage Markdown (https://daringfireball.net/projects/markdown/syntax)

A partir de maintenant, quand on dit commiter des modifications on comprend que ces commits sont aussi envoyés au dépôt distant.

Analyser le code

- Créer une branche « analyse » à partir de la branche main (master)
- Analyser le code pour comprendre le fonctionnement des fonctions, des blocs de code
- Ajouter des commentaires pour des lignes ou des blocs de code que vous pensez qu'ils sont nécessaires pour mieux comprendre le code. Ces commentaires sont temporels, une fois le code est amélioré, ils sont peut-être plus utiles.
- Commiter les modifications au dépôt distant sur la branche «analyse»

Formater le code source avec yapf

- Créer une branche « formatage » à partir de la branche main (master)
- Formater le code : les instructions pour installer et utiliser `yapf` se trouve sur CELENE dans le document `auto_verification_outlils.pdf (partie 2)`
- Utilisez `git status` pour vérifier le répertoire de travail
- Utilisez `git diff` pour regarder les modifs entre le répertoire de travail et ce qui est présent dans la dernière version du dépôt
- Ajouter des modif dans l'index avec git add -u
- Utilisez `git status` pour vérifier l'index
- Commiter les modifications
- Vérifier encore avec `git status`
- Envoyer le commit au dépôt distant sur la branche «formatage»
- Décrire dans le readme.md les modifications que `yapf` a fait

Nettoyer le code

- Créer une branche « nettoyage » à partir de la branche main (master)

  ATTENTION : dans cette étape, on reste toujours sur cette branche « nettoyage », il ne faut pas changer à main ou faire des merges
- Modulariser le code : créer un fichier pour les constantes constantes.py. Mettre toutes les constantes dans ce fichier. Importer le module constantes dans le fichier tetris.py. Commiter les modifications (un nouveau fichier et des modifications dans le fichier tetris.py)
- Noms significatifs : Mettre à jour les noms des fonctions, des variables au style PEP8 (snake_case) à la place du style camelCase. Vous pouvez regarder les slides du cours pour avoir des idées de modifications. Définir des constantes avec noms significatifs pour les nombres magiques. Commiter les modifications
- Annotators : Il faut ajouter des annotations python nécessaires pour améliorer la compréhension du code. Commiter les modifications
- Documentation : Il faut ajouter la documentation (docstring) pour classes et fonctions. Commiter les modifications

Re-factoring avec Pylint

- Créer une autre branche «refactoring» à partir de la branche main (master)

  ATTENTION : dans cette étape, on reste toujours sur cette branche «refactoring», il ne faut pas changer à main ou faire des merges

- Utiliser Pylint pour avoir des suggestions de pour améliorer la qualité du code. Les instructions pour installer et utiliser Pylint se trouve dans le document auto_verification_outlils.pdf (partie 1 et 3) sur CELENE.
- Fixer tous les avertissements du Pylint
- Ignorez les avertissements que vous ne pouvez pas fixer (voir partie 4 du même document) en utilisant soit des commentaires spécifiques dans le fichier de code ou modifier dans le fichier de configuration pylintrc.google fourni dans CELENE
- Commiter les modifications en indiquant votre score Pylint dans le readme

Fusionner vos branches

ATTENTION : Ne pas supprimer vos branches !!!

Chaque fusion ou merge doit passer par une Pull Request (PR). Démarche :

- choisir une branche (analyse ou formatage ou nettoyage ou refactoring) à l'aide de la commande switch
- faire le merge de la branche vers main
- en cas de conflits, résoudre les conflits et commitez les modifications
- créer une PR dans Github de la branche choisie vers main
- accepter la PR
- Récupérer la dernière version de la branche main depuis le dépôt distant
- Recommencez au début en changeant de branche parmi celles qui restent et ainsi de suite.

Vérifiez votre score Pylint sur la branche main. Ajustez au besoin et commitez les modifications.

Mettre à jour le readme.md

Depuis la branche main :

- Ajouter des informations du projet
- Ajouter des instructions pour lancer le jeu
- Ajouter des instructions pour jouer
- Commiter des modifications