Critères de notation

Quatre critères de notation :

Pour l'évaluation de vos programmes, nous nous nous baserons sur les critères suivants

- 0. Date de rendu
- 1. Fonctionnalités implémentées
- 2. Style et structure du code
- 3. Décomposition fonctionnelle
- 4. IHM et ergonomie

0- Date de rendu

- C'est sûrement le critère **le plus important**. Vous devez rendre dans vos programmes avant la date limite. Vous êtes aussi évalués sur la capacité à respecter ce critère. cela implique de votre part de planifier et d'organiser votre travail.

1. Fonctionnalités implémentées

Selon les projets, il vous est demandé d'implémenter des fonctionnalités .

On peut distinguer distinguer :

- **Les fonctionnalités de bases :**
  - le programme ne remplirait absolument pas le cahier des charges sans leur présence.
- **Les fonctionnalités recommandées :**
  - pour que le programme correspondent à ce qui est attendu
- **Les fonctionnalités "bonus"**
  - non demandées mais qui enrichissent l'utilisation de votre programme

2. Style et structure du code

La raison pour laquelle vous vous concentrez sur le style de programmation est que vos programmes doivent être lisibles et maintenables par vous-même et par les autres.
Le style du code que vous soumettez dans vos projets sera évalué par vos professeurs.
Voici quelques éléments à prendre en compte :

Choix approprié des noms de fonctions?

- Les fonctions sont-elles nommées de façon sensée pour indiquer ce qu'elles font?
- Le préfixe `est` (ou `a`) été utilisé pour les fonctions qui retournent `True` / `False`?
- Les procédures (c'est-à-dire, les fonctions qui ne renvoient pas de valeur) sont-elles nommées d'après ce qu'elles font?

Des docstrings de fonctions claires?

- Les doctrines des fonctions indiquent-elles clairement ce que chaque fonction fait?
- Chaque docstring explique-t-elle clairement les paramètres de la fonction et ce qui sera renvoyé par la fonction (le cas échéant).

Une docstring du programme claire?

- Est-ce que votre docstring de programme indique que vous êtes l'auteur du programme et la date à laquelle vous avez écrit le programme?

Est-ce que votre code est trop complexe?

- Vous ne devez pas utiliser plus de trois niveaux d'indentation (après le premier niveau) dans une fonction ou une méthode.

Nom des constantes

- Avez-vous utilisé des constantes bien nommées pour vos valeurs numériques?

Est-ce que votre code est assez espacé?

- Avez-vous mis au moins deux lignes vides entre les définitions de fonctions?
- Avez-vous utilisé des lignes vides pour diviser des blocs de code dans les fonctions?

3. Décomposition fonctionnelle

Vos fonctions ne font qu'une tâche

- Les bons programmes consistent en un ensemble de fonctions chacune accomplissant une tâche bien définie.

Décomposition hiérarchique

- Afin d'accomplir leurs tâches en quelques lignes (au plus 40), les fonctions de niveau supérieur appellent des fonctions de niveau inférieur et ainsi de suite.

Votre code ne contient pas de répétition

- Vous avez utilisé suffisamment le processus d'extraction de fonction et de refactoring

Votre code est bien organiser

- Vous avez traduit le 'programme logique' en code :
  - Les détails en instruction
  - Les instructions liées en fonctions
  - Les fonctions liées en modules

4. IHM et ergonomie

Le nom du programme est bien choisi

- il indique ce que fait le programme

Le nom des dossiers sont bien choisis

- ils indiquent leur contenu
  - images
  - son...
  -

Un Readme est présent dans le dossier

- il contient les informations de base sur le programme
- il est au format `.txt` ou `.md`

le programme débute avec un message de présentation ou un menu

- Pour donner les premières informations
- pour pouvoir faire des choix d'utilisation

Votre interface utilisateur a une bonne organisation visuelle :

- la quantité d'informations à analyser est limitée
- les messages de la console sont bien organisés

L'information est facilement disponible

- le programme fournit suffisamment d'informations
- des feedbacks sont prévus en cas d'interaction avec l'utilisateur

Votre programme s'adapte à l'utilisateur

- en fonction de ses préférence
- en fonctions de ses réponses précédentes

Le programme gère les erreurs des utilisateurs

- il empêcher si possible les erreurs
- il permet de repérer et de comprendre les erreurs