Charte de codage :

I. Normes de codage générales

- Encodez-en **UTF-8,** les programmes
- Il est essentiel de choisir un nom de fichier court et significatif.
- Le nom des fichiers doit être en anglais, en minuscule et les mots doivent être séparés par des underscores « _ »
- Recommandation Github : le nom du fichier doit contenir le numéro de version
- Écrivez en début de fichier en **commentaire le numéro de la version avec les changements apportés**.
- Nommez vos versions de la façon suivante **vX.Y.Z** où X ϵ {1,..,9} et Y,Z ϵ IN

  X : changement **majeur** de version

  → ajout, suppression d'une fonctionnalité ou restructuration du programme

  Y : changement **mineur** de version

  → ajout, suppression d'une fonction

  Z : **Révision** de la version précédente

  → correction de bug, optimisation d'une fonction
- **Écrivez les noms des variables en minuscule.**
- Écrivez le nom des constantes en **MAJUSCULE**.
- Utilisez «_» pour séparer les mots dans le nom des variables et des constantes.
- Nommez vos variables avec des noms **cohérents**, **compréhensibles** et en **anglais.**

  Exemple: On ne peut pas appeler une variable «a», ce nom n'est pas compréhensible. «mean» est un nom de variable trop vague, on ne sait pas de quelle moyenne il s'agit. Le nom de variable idéal est par exemple mean_height.
- **Vérifiez que les variables soient toujours utilisées.**
- **Écrivez le nom des fonctions en minuscule et séparez les différents mots à l'aide d'une majuscule.**
- Les **noms des fonctions** doivent être en **anglais.**
- Utilisez une majuscule pour commencer le nom d'une classe.
- N'utilisez pas les lettres accentuées et les "kanas" (ç, œ …).
- **Espacez** bien votre code pour le rendre lisible.
- Ajoutez des espaces à côté de vos opérateurs.

  Exemple : Ne faites pas a=1 mais plutôt a = 1.
- Limitez-vous à une instruction par ligne.

- Découpez les lignes après les opérateurs, dans le cas où elles sont trop longues.
- Commentez vos codes en français, au-dessus de la fonction ou de la ligne concernée. Soyez clair et concis. Écrivez des phrases courtes mais complètes.
- Commentez avant chaque fonction de la manière suivante :

  Entrée :

  Sortie :

  Objectif de la fonction :

- Les commentaires ne doivent pas dépasser 30% de la longueur de code

II. SQL

Attention ce langage n'est pas sensible à la casse, c'est-à-dire qu'il ne distingue pas les majuscules des minuscules

- Veillez à ce que le nom des tables soit
  - Toujours en minuscule ;
  - Toujours au singulier ;
  - Sans accents ;
  - Sans abréviations ;
  - Si c'est une table de jointure, l'écrire dans l'ordre alphabétique
- Veillez à ce que le nom des tables ne soit pas un mot réservé de SQL
- Veillez à ce que le nom des colonnes soit pour :
  - Une clef primaire : id_+ nom de la table ;
  - Un libellé : lib_+ nom de la table ;
- Nommez les contraintes des clés étrangères en commençant par « fk_» suivi du nom de la tables. Pour les autres contraintes, commencez par «ck_ »
- Définissez les contraintes au niveau de la table et non des colonnes.
- Veillez à ce que tous les mots clé de SQL soit en majuscule

III. Python

- Sautez une ligne après chaque méthode.
- Sautez deux lignes après chaque classe.
- Préférez l'utilisation de la programmation fonctionnelle dans vos scripts. Il s'agit d'utiliser le plus possible des fonctions pour les actions qui sont répétées plusieurs fois.
- Assurez- vous qu'une fonction retourne quelques choses dans tous les cas.

  Attention que le seul return ne se situe pas dans un if
- Veillez ç ce que les imports de plusieurs modules soient sur plusieurs ligne.
- Pour importer vos modules, préférez des chemins relatifs
- Veillez à ne pas mettre d'espace avant les deux points et les virgules mais après.
- Evitez d comparer une variable booléenne avec True/False. Utilisez directement la variable.
- Respectez les directives de la PEP8

Avec Spyder vous pouvez afficher des alertes lorsque votre code viole une des directive PEP8. Pour les actives, allez dans outils-> préférences-> éditeur-> Introspection et analyse de code et cochez la case à côté de Real-time code style analysis (PEP8)