Ces conventions visent essentiellement à améliorer la lisibilité du code et des dossiers : elles doivent permettre d'identifier du premier coup d'œil un maximum de choses dans le code, de se repérer facilement et de savoir où trouver les choses. Vous trouverez donc ici un ensemble de règles et de conseils à adopter durant le projet.

1) Convention logiciel

Pour ce projet nous allons **tous** travailler sur la **même version de python** afin d'éviter toute erreur lorsque l'on fusionne les codes.
Pour ceci nous choisirons d'utiliser **python 3.6** car cette version est stable.
Il est possible de vous créer un environnement spécial pour le projet à l'aide d'anaconda.
Lien d'explication : https://conda.io/docs/user-guide/tasks/manage-python.html

2) Convention de nommage des fichiers

- Pour la version finale du fichier sur le serveur, chaque nom de fichier doit commencer par l'identifiant du groupe, suivi d'un underscore et du nom de votre fichier.
  Exemple : g8_synonym.py
- Il est essentiel de choisir un nom de **fichier court et significatif**.
- Le nom du fichier doit être en **anglais** et écrit en **lettres minuscule**.
- Si le nom de votre fichier est composé, utiliser des underscores « _ » pour séparer les éléments.
- Recommandation sur Github :
  - Nommez tous vos fichiers comme dans l'exemple : g3_predict_vX.Y.py
  - Nommez vos versions de la façon suivante vX.Y. où X,Y∈ {1,..,9} :
  X : changement majeur de version
  Ajout, suppression d'une fonctionnalité ou restructuration du programme
  Y : changement mineur de version
  Ajout, suppression d'une fonction

Attention, sur le serveur, il ne faudra mettre que la version finale.

3) Convention de nommage des variables, fonctions et classes

- Veillez à ne coder qu'en **une seule langue : l'anglais**. Veillez donc à ne pas utiliser de lettres accentuées.
- Veillez à écrire vos **variables en minuscule**. Si une variable qui contient plusieurs mots, séparez-les avec des underscores « _ ».
- Evitez les variables trop longues, peu compréhensibles et quasi-semblables.
  Exemple à ne pas reproduire : this_is_a_variable et this_is_a_variable2.
- Veillez à écrire vos **constantes en majuscule**. Si une constante contient plusieurs mots, séparez-les avec des underscores « _ ».
- Veillez à écrire le nom de vos **fonctions en minuscule** et séparez-les avec des underscores « _ » si elles contiennent plusieurs mots.
- Veillez à **commencer** le nom de vos **classes en majuscule**. Si une classe qui contient plusieurs mots, séparez-les en commençant chaque mot par une majuscule.

4) Commentaires du code et des fonctions

- Chaque code doit commencer par un commentaire indiquant votre groupe, ainsi que les initiales du membre du groupe qui a réalisé ce code.
  Exemple :
  """
  Created on Wed Jan 10 10:01:47 2018
  Group 10
  @author: A.B.
  """
- Evitez les commentaires inutiles.
- Toutes vos fonctions doivent être commentées, de façon à indiquer ce que prend la fonction en entrée et ce qu'elle retourne, suivie d'un petit résumé de ce qu'elle effectue.
  Exemple : def function f(x):
  """
  input : explain what the parameters are.
  ex.: x: json data.
  ouput : explain what the function returns.
  This is what the function is doing.
  """
- Les commentaires ne doivent pas dépasser 30% de la longueur du code.
- Indiquez dans les commentaires ce que fait le code, pas comment il le fait (le code doit être suffisamment clair pour que le « comment » se lis tout seul).

5) En ce qui concerne le code :

- Chaque code doit être encodé en utf-8.
- Veillez à bien indenter votre code ! Une bonne indentation du code permet de mieux s'y retrouver et d'éviter souvent de nombreuses erreurs.
- Aérez votre code avec des retours à la ligne. Cela améliore sa structure.
- Veillez à toujours bien espacer vos variables de vos opérateurs.
  Exemple : var = var1 + var2
- Veillez à ne pas écrire des lignes de code trop longues. Sur une ligne, 80 caractères semblent être la limite actuelle pour ne pas utiliser la barre de défilement afin de voir la fin du code.
- Dans le cas d'une ligne qui dépasse 80 caractères, découpez-la après les opérateurs.
- Préférez écrire du code simple qui est facile à comprendre. Le code le plus simple n'est pas nécessairement le plus petit, soyez prêt à faire des compromis de taille en faveur de la simplicité.
- Préférez les méthodes simples et les fonctions qui prennent un petit nombre de paramètres. Évitez les méthodes et les fonctions qui sont longues et complexes.
- Évitez de mettre beaucoup d'idées sur une seule ligne de code.

- Veillez à ce que toutes les variables que vous avez définies ainsi que les packages que vous avez importés soient utilisés.
- Ne déclarez pas une variable qui n'est utilisée qu'une seule fois.

6) 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 quelque chose dans tous les cas.
- Attention que le seul « return » ne se situe pas dans un « if » dont la condition n'est pas toujours vérifiée.
- Veillez à ce que les imports de plusieurs modules soient sur plusieurs ligne. Bien sûr, cela n'est pas valable pour « from x import y,z ».
- Pour importer vos modules, Préférez des chemins relatifs. Exemple : « import sys » et non « from sys import * ».
- Veillez à ne pas mettre d'espace avant les deux points et les virgules, mais après.
- Veillez à ne pas mettre d'espace à l'intérieur des parenthèses, crochets ou accolades.

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

7) 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 :
  - o Toujours en minuscule ;
  - o Toujours au singulier ;
  - o Sans accents ;
  - o Sans abréviations ;
  - o 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 :
  - o Une clef primaire : id_ + nom de la table ;
  - o Un libellé : lib_ + nom de la table.
- Nommez les contraintes des clés étrangères en commençant par «fk_» suivi du nom de la table fille puis de la table mère.
- Nommez les contraintes des clés primaires en commençant par «pk_» suivi du nom de la table. 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.