NORMES DE CODAGE

1. Normes de codage générales

- Encodez en **UTF-8,** les programmes
- Mettez en haut en commentaire du fichier : **Groupe**[Numéro_de_groupe] .
- 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 ∈ {1,..,9} et Y,Z ∈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 ...).
- **Indentez** vos structures de contrôles (if, for, while ...) avec des **tabulations** pour mieux s'y retrouver. N'utilisez pas des espaces pour l'indentation.
- Les instructions des structures de contrôles sont attendues entre accolades.
- **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.
- **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:

  Objectif de la fonction:

  Sortie:
- Tous ce qui peut être **visible sur le site/application** doit être en **français**

2. Normes de codage spécifiques à certain langage

2.1. SQL

- Privilégiez les 3ème forme normale.
- Nommez les attributs de telle sorte que le nom soit **unique** à la base de données.

  Exemple : Toutes les tables ne peuvent pas avoir un attribut «id». S'il s'agit de la table User, on choisira «id_user».
- Définissez les contraintes au niveau de la table et non des colonnes.
- Nommez les contraintes des clés primaires en commençant par «pk_» suivi du nom de la table. On aura donc pk_NomTable.
- Nommez les contraintes des clés étrangères en commençant par «fk_» suivi du nom de la table fille puis de la table mère. On aura donc fk_TableFille_TableMère.
- Nommez les autres contraintes en commençant par «ck_».
- Écrivez le nom des paramètres, variables, tables, vues, curseur, trigger … en minuscules.
- **Écrivez les *keywords* en MAJUSCULES** (SELECT, FROM, WHERE).
- Présentez les requêtes de la forme suivante :

  SELECT ….

  FROM ….

  WHERE ….

  GROUP BY ….

  HAVING ….

  ORDER BY ….
- Utilisez des alias pour les tables et les vues dans vos requêtes.
- Privilégiez t1.attribut = t2.attribut à t1.attribut IN (SELECT attribut FROM t2) pour les jointures.

2.2. R

- Tous les fichiers doivent être indépendants.
- Mettez **rm(list = ls())** en début de fichier.
- Importez les librairies en début de fichier
- **Utilisez** dans tous les cas **les accolades**, même si elles sont facultatives. Chaque accolade fermante doit être verticalement alignée à l'instruction correspondante à l'accolade ouvrante.
- Séparez les blocs d'instructions ou les fonctions par une ligne.
- Utilisez **<-** pour les affectations plutôt que =.
- N'utilisez pas d'abréviations pour FALSE et TRUE.
- **Evitez** les boucle **for**

2.3. Python

- Référez-vousà https://www.python.org/dev/peps/pep-0008/ et https://gist.github.com/sloria/7001839. Le premier lien correspond aux directives dites *PEP8* qui sont utilisées dans l'industrie. 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 sytle analysis (PEP8)**.
- Préférez l'utilisation de la programmation fonctionnelle dans vos scripts. Il s'agit utiliser le plus possible des fonctions pour les actions qui sont répétées plusieurs fois.
- Découpez les lignes après les opérateurs, dans le cas où elles sont trop longues.
- Placez les directives d'importations en tête du fichier avec un saut de ligne entre chaque. Les directives d'importation doivent être divisées en trois groupes, dans l'ordre, celles faisant référence:
    - à la bibliothèque standard
    - à des bibliothèques tierces
    - à des modules de votre projet.
- Utilisez des chemins absolus et non des relatifs pour les directives d'importation. Ajoutez un saut de ligne entre chaque groupe de directives d'importation.
- N'utilisez pas d'espace:
    - après les accolades, crochets et des parenthèses ouvrantes
    - avant les accolades, crochets et des parenthèses fermantes
    - avant les virgules, les points virgules et les deux points
    - avant la parenthèse ouvrante qui introduit la liste des paramètres d'une fonction
    - avant le crochet ouvrant indiquant une indexation ou sélection
- Séparez par deux sauts de ligne la définition d'une fonction et la définition d'une classe.
- 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.
- Précédez les noms des méthodes/fonctions protected par un «_».
- Précédez les noms des méthodes/fonctions privées par deux «_», c'est-à-dire __[nomDeLaFonction].

2.4. HTML/CSS

- Indentez les balises, ouvrez-les et fermez-les au même niveau.
- Commentez votre code en utilisant la syntaxe suivante : <!-- Ceci est un commentaire -->
- Utilisez les URL canoniques.
- Faites la mise en page dans une feuille de style (CSS)

2.5. JavaScript

- Commencez le nom des fonctions constructeurs par une majuscule. Cela ne change rien, n'importe qu'elle fonction peut devenir un constructeur si elle est instanciée avec le mot clé new, mais cela facilite la lecture du code.
- Utilisez des guillemets simples pour les strings.
- N'utilisez pas de point virgule.
- Mettez un espace après les mots clés et après le nom des fonctions
  exemple: if (condition) {...}
  function name (arg) {…}

2. 6. PHP

- Commencez le nom des interfaces par un I (i majuscule) :

  exemple : INomInterface
- Utilisez des guillemets simples pour les chaînes de caractères
- Revenez obligatoirement à la ligne pour mettre les instructions après un if.
- N'utilisez pas d'espace :
  - après les accolades, crochets et des parenthèses ouvrantes
  - avant les accolades, crochets et des parenthèses fermantes
  - avant les virgules, les points virgules et les deux points
  - avant la parenthèse ouvrante qui introduit la liste des paramètres d'une fonction
  - avant le crochet ouvrant indiquant une indexation ou sélection
- Utilisez un espace:
  - Après un « if », « elseif », « else », « while », « for », « foreach »
  - Après chaque « , ».
- N'utilisez pas de « switch/case »