# Norme de programmation

1. Écrire les noms de variables, fonctions et classe en anglais. Ainsi que les commentaires et la documentation
2. Utilisez des noms de variables significatives et décrivez clairement ce qu'ils représentent. Les noms doivent être en minuscule séparer les mots avec un underscore. Exemple : “liste\_orginin”, “first\_name”, “last\_name”.
3. Utilisez des verbes pour les noms de fonctions et commencez chaque mot avec une majuscule sauf le premier mot. Exemple : "calculerTaxes", "enregistrerUtilisateur", "afficherMessageErreur".
4. Utilisez des noms de classes significatifs et décrivez clairement ce qu'ils représentent, en commençant chaque mot avec une majuscule. Exemple : "Utilisateur", "Commande", "Produit". Éviter les verbes dans les noms de classe.
5. Utilisez des commentaires pour décrire le code lorsque nécessaire et documenter les fonctions et les classes. Éviter les commentaires redondants ou pouvant porter à confusion.
6. Évitez les fonctions trop longues, en les limitant à une seule tâche.
7. Évitez d'utiliser des variables globales, car elles peuvent causer des effets secondaires imprévus.
8. Évitez d'utiliser des constantes magiques (nombres ou chaînes de caractères utilisés dans le code sans explication), en les remplaçant par des variables avec des noms significatifs.
9. Évitez les redondances dans le code.
10. Évitez les fonctions qui retournent plusieurs valeurs. Privilégier plutôt une structure claire comme une classe, une liste ou un tuple.
11. Évitez les [effets de bord](https://fr.wikipedia.org/wiki/Effet_de_bord_(informatique)#:~:text=En%20informatique%2C%20une%20fonction%20est,autre%20que%20retourner%20une%20valeur.) dans les fonctions (c'est-à-dire les effets sur les

variables en dehors de la fonction).

1. Évitez d'utiliser des goto, car ils peuvent rendre le code difficile à suivre.
2. Utilisez des assertions pour vérifier que les préconditions et les postconditions sont respectées. Les préconditions devraient se trouver avant l’exécution du code principal.
3. Évitez les exceptions silencieuses, car elles peuvent rendre le débogage difficile. (c’est-à-dire les arrêts de programmes involontaire qui ne sont pas fournis de code d’erreur ou encore les try-catch avec aucune gestion de l’erreur)
4. Utiliser un logger afin de pouvoir mieux comprendre le comportement des bugs survenu.
5. Utilisez des exceptions personnalisées (dérivée de la classe exception) au lieu de code d’erreur sous forme d’enum.
6. Évitez les fonctions qui ont un grand nombre d'arguments.
7. Utilisez des tests unitaires pour vérifier que les fonctions fonctionnent correctement.
8. Utilisez des outils de gestion de versions tels que Git pour suivre les modifications apportées au code.
9. Éviter l’indentation qui dépasse deux niveaux de profondeur dans une fonction. Si celle-ci dépasse de deux niveaux, il est probable que le code pourrait être extrait dans d’autres fonctions.
10. Évitez d'utiliser des fonctions obsolètes ou dépréciées.

Liste conçue à l’aide de ChatGPT et de la documentation Coder proprement fournis en classe.