Newton Coding Style

I. Introduction

1. Organisation

Le Newton Coding Style est un jeu de règles que vous devez suivre pour que votre code soit lisible aisément et compréhensible par tout le monde. (Tout non respect de ses règles entrainera des pertes de points voir la non lecture de vos exercices).
Il couvre :

- L'organisation de votre dossier projet
- Le nom de vos variables
- L'organisation de vos fichiers codes (les paragraphes)
- L'écriture de vos lignes de codes

2. Pourquoi ?

Ecrire un code avec le NCS rend la lecture du code simple pour les autres. Cela facilite donc le travail en groupe, l'entraide et la correction de vos codes. Cela vous permet en plus de structurer et de rendre votre code plus clair. Mais surtout ça vous facilite la vie dans :

- La lecture et relecture
- Le debuggage (plus gros travail de l'informaticien)
- La maintenance
- La conception logique de votre code
- La réutilisabilité de votre code (que vous ne fassiez pas 2 fois la même chose)

3. Niveaux de règles

Les règles sont répartis en 3 catégories de sévérités :

- Les **Règles Majeures** : Elles concernent la structure du code et sont extrememement dommageable à l'écriture de votre code. Transgressée une règle majeure est un problème majeur (même qu'une fois) et doit être réglé le plus rapidement possible. Les règles majeures seront écrites en Rouge
- Les *Règles Mineures* : Elles concernent la représentation visuelle du code. Les suivre permet de rendre votre code plus lisible. Transgresser de façon répétée des erreurs mineures doit être évitée car cela rend le code difforme et peu lisible. Les règles mineures seront écrites en Vert
- Les Règles Informatives : Elles concernent les points spécifiques triviaux qui ne sont pas aussi importants que les règles mineures et majeures. Chacune de ces

règles représente de bonnes pratiques et doit être suivies pour assurer un code de la meilleure qualité possible. Les règles informatives seront écrites en Gris

Théorème 0.1.
Le Coding Style est une convention de syntaxe seulement et ne peut nullement être utilisée comme une excuse si votre code ne fonctionne pas!

R Le document est inspiré du Coding Style d'Epitech qui est lui inspiré du Linux Kernel Coding Style.

4. Tarifs

- La note maximale à un code qui ne fonctionne pas est de 5/20.
- Une infraction à une règle majeure enlève 3 points.
- Une infraction à une règle mineure enlève 1 point.
- Une infraction à une règle informative enlève 1/2 point.

II. Organisation de votre fichier source

- **O-1** : PEP8 : Le nommage des variables, nom de fonctions, nom de fichier doit suivre le guide de style python PEP8. Les points importants sont :
  - Les noms de variables et de fonctions sont en minuscules
  - Les espaces sont remplacés par des underscores _.
- **O-2** : Nommage des fichiers, dossiers et fonctions : Le nom des fichiers doit définir de façon claire, précise, explicite, non ambiguë ce que le fichier fait : ~~exercice 1.py~~, résolution_exercice_1.py, correction_exercice_1.py
- **O-3** : Abréviations Les abréviations, les noms de variables de 1 caractère sont interdits.

III. Echelle globale du programme

- **G-1** : Séparation des fonctions Les fonctions doivent être séparées par une et une seule ligne vide.
- **G-2** : Variables globales : Les variables globales dans les fonctions doivent être éviter. Si cela ne peut pas être éviter, il faut les déclarer au début des fonctions par un : *global exemple_de_variable_globale*. Les seules variables globales doivent être les constantes. Ces dernières sont écrites toute en CAPITALE.
- **G-3** : Espace de fin de ligne Il ne doit pas y avoir d'espace ou de tabulation à la fin des lignes
- **G-4** : Indentation initiale : Il ne doit pas y avoir d'espace ou de tabulation en début de fichier. La première ligne de code ne peut jamais contenir en premier caractère un espace.

IV. Les Fonctions

- **F-1** : Cohérence des fonction Une fonction ne doit faire qu'une chose, elle ne peut pas mélanger plusieurs degrés d'abstraction. Elle doit aussi respecter le

principe de responsabilité unique et n'est responsable que d'une tache.
- **F-2** : Nommage des fonctions Le nom d'une fonction doit définir la tache qu'elle execute et doit contenir un verbe. Dans la mesure du possible les fonctions doivent être écrites en anglais (cela sera bonifier).
- **F-3** : Largeur du code La longueur d'une ligne ne peut pas dépasser une longueur de 80 caractères.
- **F-4** : Longueur de fonctions et de scripts Le corps d'un script ou d'une fonction doit être le plus court possible et ne doit pas dépasser plus de 20 lignes.
- **F-5** : Nombre de paramètres Une fonction ne peut pas avoir plus de 4 arguments
- **F-6** : Docstring Une fonction doit avoir un petit commentaire décrivant l'usage et le type des arguments ainsi qu'une ligne de texte expliquant ce que fait la fonction.
- **F-7** : Fonctions emboitées Les fonctions à l'interieur de fonction sont interdites car elles ralentissent énormément le code.

V. A l'interieur d'une fonction

- **I-1** : Contenu d'une ligne Une ligne ne doit contenir qu'une seule action/déclaration
- **I-2** : Espaces Quand vous utilisez des espaces, mettez en juste un seul (pas de double ou de triples espaces)

VI. Structure de Contrôle

- **C-1** : Branchement successifs Les blocs while, for, if/else ne peuvent pas avoir une profondeur de 3 ou plus. Pas de for dans un autre for qui lui même est dans un auter for.