Comment coder proprement

Les conseils proposés dans ce guide sont tirés, en majeur partie, de l'ouvrage *Clean Code: A Handbook of Agile Software Craftsmanship* de Robert C. Martin. Il s'agit d'un ouvrage de référence sur la question des bonnes pratiques de programmation. Seul, le contenu des quatre premiers chapitres de l'ouvrage sont résumés dans ce guide. Nous vous invitons à consulter le livre si vous voulez aller plus loin.

1. Nommer les variables

Coder proprement nécessite d'abord et avant tout de bien nommer ses variables. Du code dont les variables sont mal nommées peut rapidement dewvenir incompréhensible, même pour son auteur.

Le grand principe à respecter lorsqu'on nomme ses variables : **révéler l'intention**. Pour le dire succintement il faut dire : >* **d'une variable -> ce que c'est.** >* **d'une fonction -> ce qu'elle fait.**

Quelques trucs à respecter pour bien nommer ses variables

1. **Éviter les noms de variables courts.**
2. **Éviter les abréviations.**
3. **Éviter des noms de variables qui varient peu entres eux.**
4. **Éviter d'utiliser deux mots différents pour dénoter la même chose/action** (un mot par concept).
5. **Utiliser un *verbe* qui décrit ce que fait une fonction lorsque vous la nommer.**

Conseil	Mauvais	Bon
1	`a = pd.read_csv("articles.csv")`	`articles = pd.read_csv("articles.csv")`
2	`tot_obs = 10`	`total_observations = 10`
3	`count_of_trees_in_the_city = 50` `count_of_plants_in_the_city = 40`	`trees_count = 50` `plants_count = 40`
4	`get_observations() fetch_annotations()`	`get_observations()` `get_annotations()`

Conseil	Mauvais	Bon
5	`def articles(file_path): [code]`	`def get_articles(file_path) [cod`

2. Les commentaires

Il y a une croyance assez répendue dans le monde de la programmation selon laquelle plus on commente notre code, mieux c'est. Selon l'auteur de *Clean Code*, cette croyance est fausse. Le grand danger réside dans le fait que l'on peut oublier de mettre à jour des commentaires. Lorsque cela se produit, les commentaires peuvent nous induire en erreur. Pour paraphraser Robert C. Martin : **«les commentaires peuvent mentir, mais le code dit la vérité»**. Sachant cela, il vaut mieux, la plupart du temps, utiliser son énergie pour essayer d'écrire du code propre au lieu d'écrire des commentaires. Voici donc quelques conseils en lien avec les commentaires Privilégier le code propre. >**Les choses à éviter** >1. Plutôt que d'écrire de longs commentaires afin d'expliquer comment fonctionne un bout de code malpropre, prenez ce temps pour essayer de **rendre votre code plus propre**. >2. **Éviter les commentaires qui expliquent ce que fait du code simple**. Vous pouvez tenir pour acquis que la personne qui lira votre code connaît les fonctionnalités de base du langage que vous utilisez. Il n'a pas besoin de plus d'explications. >3. **Ne pas laisser de code commenté**. On peut avoir tendance à commenter du code qui ne sert plus, mais dont on pense avoir besoin plus tard. Il vaut mieux l'effacer et se reposer sur notre outil de contrôle des versions (git).

Mauvais `# itération sur la liste d'articles pour aller chercher les titres # 'a' représente la liste des articles (qui est une liste de liste) list = [] for i in range(52): list.append(a[i][3])`
Bon `title_index = 3 number_of_articles = len(articles) article_titles = [] for article_index in range(number_of_articles): article_titles.append(articles[articl`

Mauvais `articles = get_articles(path) # articles = articles[:-1] # articles = do_something_funky(articles) save(articles)` **Bon** `articles = get_articles(path) save(articles)`

> **Les commentaires sont** utiles** pour** : 1. **Justifier un choix**, par exemple si vous faites un choix d'algorithme, un choix de libraire, ou un choix de conception qui mérite d'être justifié. 2. **Indiquer la durée que prend l'exécution d'un bout de code** (si c'est long). 3. **Documenter une fonction** (préciser le type des paramètres et les données retournées). Voir, par exemple, l'outil Docstrings en *python* 4. **Expliquer un bout de code complexe** (dont la logique serait difficile à comprendre même pour un programmeur ou une programmeuse d'expérience).