Bonnes pratiques en Python

Cette page est inspirée sur ce document.

De même que R, Python est "juste" un langage de programmation, il est donc possible d'écrire du code Python dans un éditeur de texte et de lancer ce code via un terminal. Bien que ce soit faisable, il est plus simple d'utilser un éditeur de code. Bien qu'il soit techniquement possible d'utiliser RStudio comme éditeur pour Python, je vous recommanderais dans utiliser spécialiser comme *PyCharm*. Le format *notebook* est aussi très populaire en Python. Dans ce cas, je vous conseillerais la librarie `marimo` qui permet de transformer ses fichiers `.py` en notebook.

De plus, il vaut mieux s'assurer de travailler avec la dernière version stable de Python comptatible avec les packages les utilisés (e.g. `numpy`, `matplotlib`, `sklearn`). En effet, très souvent, il y a un délai pour que les dernières versions des packages soient comptatibles avec la dernière version de Python. Il est donc important de partager la version des packages utilisés avec l'analyse.

Concernant la documentation, le standard en Python est de suivre les recommendations du PEP 257. La libraries `Sphinx` permet de générer une documentation à partir des *docstring*.

Une syntaxe uniforme rend un code beaucoup plus facile à lire et à comprendre. Un guide de style énonce des normes pour avoir une syntaxe uniforme. Le guide de style PEP 8 est recommandé en Python.

Retour à la ligne et indentation

L'indentation fait partie de Python, i.e. les blocks `if...else` et `for` sont définis grâce aux indentations. Il est possible d'utiliser un *tab* ou des *espaces*. Le choix vous appartient mais il est important d'être consistant dans votre code.

Convention de noms

L'utilisation des noms de variables avec un seul caractère est à éviter. Sinon, les variables, fonctions, méthodes, packages et modules sont nommés avec des lettres minuscules, des nombres et l'underscore `_`. Les classes et exceptions doivent être nommés avec des majuscules pour séparer les mots (`UneClasse`). Les constants sont en majuscules.

Organisation du code

De même que pour R, lorsque le code commence à devenir long, il devient avantageux de le séparer en plusieurs fichiers. Par exemple, on peut avoir un fichier par partie de l'analyse (un pour le nettoyage des données, un pour l'analyse, un pour la visualisation, …). On peut aussi créer plusieurs sous-dosssiers.

Quelques trucs à faire

1. Ne pas faire de comparaison à `True`, `False` or `None`.

```
if attr:
    print("True!")

if not attr:
    print("False!")

if attr is None:
    print("None!")
```

2. Utiliser la compréhension de liste lorsque cela est possible.

```
a = [3, 4, 5]
b = [i for i in a if i > 4]
```

3. Charger un fichier avec `with`. Cela permet d'être sur que la connexion avec le fichier est fermé une fois qu'il a été lu.

```
with open("file.txt") as f:
    read(f)
```

4. Utiliser un maximum de 80 caractères par ligne.
5. Utiliser des parenthèses pour aller à la ligne dans les longues chaînes de caractères.