15.2 Les *docstrings* et la PEP 257

Les *docstrings*, que l'on pourrait traduire par « chaînes de documentation » en français, sont un élément essentiel de nos programmes Python comme on l'a vu au chapitre 14 *Création de modules*. À nouveau, les développeurs de Python ont émis des recommandations dans la PEP 8 et plus exhaustivement dans la PEP 257[4] sur la manière de rédiger correctement les *docstrings*. En voici un résumé succinct.

De manière générale, écrivez des *docstrings* pour les modules, les fonctions, les classes et les méthodes. Lorsque l'explication est courte et compacte comme dans certaines fonctions ou méthodes simples, utilisez des *docstrings* d'une ligne :

```
"""Docstring simple d'une ligne se finissant par un point."""
```

Lorsque vous avez besoin de décrire plus en détail un module, une fonction, une classe ou une méthode, utilisez une *docstring* sur plusieurs lignes.

```
"""Docstring de plusieurs lignes, la première ligne est un résumé.

Après avoir sauté une ligne, on décrit les détails de cette docstring.
blablabla
blablabla
blublublu
bliblibli
On termine la docstring avec les triples guillemets sur la ligne suivante.
"""
```

Remarque

La PEP 257 recommande d'écrire des *docstrings* avec des triples doubles guillemets, c'est-à-dire

`"""Ceci est une docstring recommandée."""`

mais pas

`'''Ceci n'est pas une docstring recommandée.'''`.

Comme indiqué dans le chapitre 14 *Création de modules*, n'oubliez pas que les *docstrings* sont destinées aux utilisateurs des modules, fonctions, méthodes et classes que vous avez développés. Les éléments essentiels pour les fonctions et les méthodes sont :

1. ce que fait la fonction ou la méthode,
2. ce qu'elle prend en argument,
3. ce qu'elle renvoie.

Pour les modules et les classes, on ajoute également des informations générales sur leur fonctionnement.

Pour autant, la PEP 257 ne dit pas explicitement comment organiser les *docstrings* pour les fonctions et les méthodes. Pour répondre à ce besoin, deux solutions ont émergées :

— La solution Google avec le *Google Style Python Docstrings*[5].

— La solution *NumPy* avec le *NumPy Style Python Docstrings*[6]. *NumPy* qui est un module complémentaire à Python, très utilisé en analyse de données et dont on parlera dans le chapitre 17 *Quelques modules d'intérêt en bioinformatique*.

On illustre ici la solution *NumPy* pour des raisons de goût personnel. Sentez-vous libre d'aller explorer la proposition de Google. Voici un exemple très simple :

```
def multiplie_nombres(nombre1, nombre2):
    """Multiplication de deux nombres entiers.

    Cette fonction ne sert pas à grand chose.

    Parameters
    ----------
    nombre1 : int
        Le premier nombre entier.
    nombre2 : int
        Le second nombre entier.

        Avec une description plus longue.
        Sur plusieurs lignes.

    Returns
    -------
    int
        Le produit des deux nombres.
    """
    return nombre1 * nombre2
```

Lignes 6 et 7. La section `Parameters` précise les paramètres de la fonction. Les tirets sur la ligne 7 permettent de souligner le nom de la section et donc de la rendre visible.

Lignes 8 et 9. On indique le nom et le type du paramètre séparés par le caractère deux-points. Le type n'est pas obligatoire. En dessous, on indique une description du paramètre en question. La description est indentée.

Lignes 10 à 14. Même chose pour le second paramètre. La description du paramètre peut s'étaler sur plusieurs lignes.

Lignes 16 et 17. La section `Returns` indique ce qui est renvoyé par la fonction (le cas échéant).

Lignes 18 et 19. La mention du type renvoyé est obligatoire. En dessous, on indique une description de ce qui est renvoyé par la fonction. Cette description est aussi indentée.

Attention

L'être humain a une fâcheuse tendance à la procrastination (le fameux « Bah je le ferai demain… ») et écrire de la documentation peut être un sérieux motif de procrastination. Soyez vigilant sur ce point, et rédigez vos *docstrings* au moment où vous écrivez vos modules, fonctions, classes ou méthodes. Passer une journée (voire plusieurs) à écrire les *docstrings* d'un gros projet est particulièrement pénible. Croyez-nous !