Bonnes pratiques de programmation en Python

Le respect de certaines règles plus ou moins explicites permettent à tout programmeur de lire, comprendre et modifier le programme écrit par un autre. Nous allons voir ici quelques-unes de ces règles.

Syntaxe pythonesque !

1. Règles explicites du PEP8

Les règles de syntaxe de Python sont définies dans le document PEP8 (Python Enhancement Proposal) rédigé entre autres par Guido van Rossum lui-même!

A consulter sur https://www.python.org/dev/peps/pep-0008/

Ces règles définissent :

Conventions de nommage :

- Le nom des variables, fonctions et méthodes s'écrit toujours en snake_case (ex : vitesse_voiture)
- Le nom des classes (voir chapitre 3) s'écrit toujours en CamelCase (ex : VoitureSport)
- Le nom des constantes s'écrit toujours en lettres capitales (ex : VLUMIERE)

Gestion des espaces

- Les opérateurs sont censés être entourés d'un espace (sauf pour le passage de paramètres à une fonction ou pour des questions de lisibilité de grandes expressions mathématiques)
- Pas d'espace à l'intérieur de (), [] ou {}
- Pas d'espace avant : et , mais on en met **après**.

```
x = 2
a = x*2 - 1
b = x*x + y*y
c = (a+b) * (a-b)

def puissance(x, puis=3):
    if x == 5:
        return x**puis

print(puissance(5, 2))
```

Longueur des lignes

La longueur des lignes est limitée en principe à 79 caractères, on peut raccourcir une longue ligne grâce à :

- L'indentation
- Des parenthèses

```
dico {
    0: 'zero',
    1: 'un',
    2: 'deux'
}
```

Indentation = 4 espaces (ou une tabulation)

Saut de ligne

- 2 lignes entre les imports et les fonctions, puis entre chaque définition de fonction
- Mais une seule ligne entre chaque méthode d'une classe (voir Chapitre 3)

2. Règles implicites

- **Choisir** judicieusement le nom de ses variables :

```
for x in y:
ou
for lettre in chaine:
?
```

- **Documenter** chaque fonction avec un prototype (*docstring*). Ne pas hésiter à inclure un exemple simple et concret pour les fonctions compliquées

```
def calcul_moyenne(x, y):
 """
 Calcul de la moyenne de x et y

 param x: première valeur (int ou float)
 param y: seconde valeur (int ou float)
 return: moyenne de x et y

 Exemple:

 >>> calcul_moyenne(4, 7)
 5.5
 """
```

De plus, il faudra toujours :

- **Anticiper et gérer les bugs** en levant des exceptions et en incluant des tests les plus complets possibles de manière à avoir la meilleure couverture de code (pourcentage de code couvert par les tests) possible
- Chercher à optimiser le temps d'exécution du programme
- Assurer la mise à jour de son programme