Guide de style pour Python

1. Introduction

Un programme sera relu et modifié par un humain : son auteur ou un autre développeur. Adopter un style d'écriture standard facilite cette relecture.
Par conséquent, il est recommandé de respecter des éléments de style et de suivre les conventions d'écritures de la **PEP 8** et de la **PEP 257**.

- La **PEP 8** (https://www.python.org/dev/peps/pep-0008/) a pour objectif de définir des règles communes entre développeurs.
- La **PEP 257** (https://www.python.org/dev/peps/pep-0257/), quant à elle définit les conventions sur les docstrings.

2. Le nommage

Syntaxiquement, les noms de variables, de fonctions, de classes, de méthodes, d'attributs peuvent comporter des lettres, des chiffres, des caractères « _ » (underscore) et ne doivent pas commencer par un chiffre. On choisira un **nom évocateur**.

- **Modules** : noms courts, lettres minuscules + underscores.

```
crypto
```

- **Variables, fonctions et méthodes** : lettres minuscules + underscores.

```
une_variable = 10

def une_fonction():
    return var
```

- **Classe** : écriture en CamelCase (minuscules, sans underscore, avec une majuscule au début de chaque mot.

```
class CeciEstUneClasse:
    def methode(self):
        pass
```

- Constante : majuscules + underscores

```
TOTAL_MAX = 10
```

3. Indentation

Les blocs Python sont délimités par l'indentation.
Une indentation correspond à **4 espaces** (pas de tab). La plupart des IDE Python utilisent ce réglage.

4. Lignes

Une ligne ne doit pas exécuter 79 caractères. Lorsqu'une instruction court sur plusieurs lignes, on facilite la lecture en indentant.

```
# Alignement avec la parenthèse ouvrante
foo = nom_de_fonction_long(var_one, var_two,
                           var_three, var_four)
```

On laisse deux lignes vides entre les différentes fonctions ou classes à l'intérieur d'un module.
Au sein d'une classe, les méthodes sont séparées par une seule ligne.

5. Espaces

- Les opérateurs doivent être entourés d'espaces.

```
# Correct
var = 2
x == y
1 + 2
```

```
# Wrong
var=2
x==y
1+2
```

  - Exception pour grouper les opérateurs mathématiques ayant la priorité la plus haute pour distinguer les groupes.

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

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

  - Exception pour le signe = dans la déclaration d'arguments et le passage de paramètres.

```
def fonction(arg='valeur '):
    pass

resultat = fonction(arg='valeur')
```

- Pas d'espaces après « `([{` », ni avant « `)]}` »
- Pas d'espaces avant les deux points « `:` » et les virgules « `,` » mais après oui.
  - Exception pour les « `:` » dans les slices (pas d'espace avant et après).

```
# Correct
def fonction(x, y):
    pass
dico = {'a': 1, 'b': 2, 'c': 3}
lst = [1, 1, 2, 3, 5, 8, 9]
print(lst[2:6])
```

```
# Wrong
def fonction(x , y) :
    pass
dico = {'a' : 1, 'b' : 2, 'c' : 3}
lst = [1,1,2,3,5,8,9]
print( lst[2 : 6] )
```

6. Docstrings (PEP 257)

On utilise toujours des triples quotes :

```
def fonction_avec_docstring_courte():
    """Résumé en une ligne """
    pass
```

Si la docstring est longue :

```
def fonction():
    """Résumé en une seule ligne suivi d'une ligne vide

    Description longue qui se termine par des
    triple quotes sur une ligne
    """
```

Source : https://www.python.org/dev/peps/pep-0008/
https://www.python.org/dev/peps/pep-0257/