PEP

Python Enhancement Proposal (PEP) is a document that proposes a new feature, improvement, or change to the Python programming language. Ces règles ne sont pas obligatoires, mais elles sont fortement conseillées pour écrire du code Python de qualité. Elles facilitent la compréhension, la maintenance et la collaboration sur le code.

PEP257 for doctrings

La PEP 257 recommande d'écrire des docstrings avec des triples doubles guillemets, c'est-à-dire
"""blabla""" mais pas '''blabla'''
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.
"""
__authors__ = ("A", "B")
__version__= "1.2.3"
import module ...
classes & methods etc

pycodestyle & pydocstyle & pylint

pycodestyle: vérifier la conformité avec PEP 8
$ pycodestyle script.py
pydocstyle: vérifier la conformité avec la PEP 257
$ pydocstyle script.py
L'outil pylint va lui aussi vérifier une partie de ces règles mais il va également essayer de comprendre le contexte du code et proposer des éléments d'amélioration.

PEP8

indentation = 4 espaces
Import module
le chargement d'un module se fait avec l'instruction import module plutôt qu'avec from module import *

PEP8 (cont)

Dans un script Python, on importe en général un module par ligne.
D'abord les modules internes (classés par ordre alphabétique), c'est-à-dire les modules de base de Python, puis les modules externes (ceux que vous avez installés en plus).
Si le nom du module est trop long, on peut utiliser un alias. (as)
Règles de nommage
Les modules (fichiers Python) doivent avoir des noms courts en minuscules et peuvent contenir des traits de soulignement (_).
Les paquets (répertoires contenant des modules) doivent avoir des noms courts en minuscules, de préférence sans trait de soulignement.
Les classes doivent utiliser la convention CapWords, c'est-à-dire commencer chaque mot par une majuscule et ne pas séparer les mots par des traits de soulignement.
Les fonctions et les variables doivent utiliser la convention snake_case, c'est-à-dire écrire les mots en minuscules et les séparer par des traits de soulignement. Par exemple, print, sum, my_function, my_variable.
Les constantes (variables dont la valeur ne change pas) doivent être écrites en majuscules et séparées par des traits de soulignement. Par exemple, PI, GRAVITY, MY_CONSTANT.
Les noms des arguments des fonctions doivent être clairs et informatifs, et éviter les abréviations ou les noms génériques comme x, y, z. Par exemple, def add(a, b): est moins clair que def add(number1, number2):.
Les noms des attributs et des méthodes d'une classe doivent suivre les mêmes conventions que les fonctions et les variables. Par exemple, class Person: name = "Alice" def greet(self): print(f"Hello, I am {self.name}").
Espace
La PEP 8 recommande d'entourer les opérateurs (+, -, /, *, ==, !=, >=, not, in, and, or...) d'un espace avant et d'un espace après.
Il n'y a, par contre, pas d'espace à l'intérieur de crochets, d'accolades et de parenthèses
Ni juste avant la parenthèse ouvrante d'une fonction ou le crochet ouvrant d'une liste ou d'un dictionnaire
On met un espace après les caractères : et , (mais pas avant)
Par contre, pour les tranches de listes, on ne met pas d'espace autour du :
on n'ajoute pas plusieurs espaces autour du = ou des autres opérateurs pour faire joli
le caractère \ permet de couper des lignes trop longues.
On peut aussi utiliser les parenthèses pour évaluer un expression trop longue, par exampl, if (condition1 and condition 2):

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com