Un piège courant

Complément - niveau basique

N'utilisez pas d'objet mutable pour les valeurs par défaut

En python il existe un piège dans lequel il est très facile de tomber. Aussi si vous voulez aller à l'essentiel: **n'utilisez pas d'objet mutable pour les valeurs par défaut**.

Si vous avez besoin d'écrire une fonction qui prend en argument par défaut une liste ou un dictionnaire vide, voici comment faire

```
# ne faites SURTOUT PAS ça
def ne_faites_pas_ca(options={}):
    "faire quelque chose"


# mais plutôt comme ceci
def mais_plutot_ceci(options=None):
    if options is None:
        options = {}
    "faire quelque chose"
```

Complément - niveau intermédiaire

Que se passe-t-il si on le fait ?

Pour exhiber l'exemple le plus simple possible, prenons le cas d'une fonction qui calcule une valeur – ici un entier aléatoire entre 0 et 10 –, et l'ajoute à une liste passée par l'appelant.

Et pour rendre la vie de l'appelant plus facile, on se dit qu'il peut être utile de faire en sorte que si l'appelant n'a pas de liste sous la main, on va créer pour lui une liste vide. Et pour ça on fait

```
import random
# l'intention ici est: si l'appelant ne fournit pas
# la liste en entrée on crée pour lui une liste vide
def ajouter_un_aleatoire(resultats=[]):
    resultats.append(random.randint(0,10))
    return resultats
```

Si on appelle cette fonction une première fois, tout semble bien aller

```
ajouter_un_aleatoire()
```

Sauf que, si on appelle la fonction une deuxième fois, on a une surprise

```
ajouter_un_aleatoire()
```

Pourquoi ?

Le problème ici est qu'une valeur par défaut - ici l'expression `[]` - est évaluée **une fois** au moment de la **définition** de la fonction.

Toutes les fois où la fonction est appelée avec cet argument manquant, on va utiliser le même objet, qui la première fois est bien une liste vide, mais qui se fait modifier par le premier appel.

Si bien que la deuxième fois on réutilise la même liste **qui n'est plus vide**. Pour aller plus loin, vous pouvez regarder la documentation python sur ce problème
(https://docs.python.org/2/faq/programming.html#why-are-default-values- shared-between-objects) .