Gestion de la mémoire

Complément - niveau basique

L'objet de ce complément est de vous montrer qu'avec python, vous n'avez pas à vous préoccuper de la mémoire. Pour cela toutefois il nous faut donner un certain nombre de détails sur d'autres langages comme C et C++. Si vous souhaitez suivre ce cours à un niveau basique vous pouvez ignorer ce complément, et seulement retenir que python se charge de tout pour vous.

Complément - niveau intermédiaire

Langages de bas niveau

Dans un langage traditionnel de bas niveau comme C ou C++, le programmeur est en charge de l'allocation - et donc de la libération - de la mémoire.

Ce qui signifie que, sauf pour les valeurs stockées dans la pile, le programmeur est amené

- à réclamer de la mémoire à l'OS en appelant explicitement `malloc` (C) ou `new` (C++)
- et réciproquement à rendre cette mémoire à l'OS lorsqu'elle n'est plus utilisée, en appelant `free` (C) ou `delete` (C++)

Avec ce genre de langage, la gestion de la mémoire est un aspect important de la programmation, qui là encore offre une grande flexibilité mais au prix d'un coût élevé en termes de vitesse de développement.

Il est en effet assez facile d'oublier de libérer la mémoire après usage, ce qui peut conduire à épuiser les ressources disponibles. À l'inverse utiliser une zone mémoire non allouée peut conduire à des bugs très difficiles à localiser.

Langages de haut niveau

Pour toutes ces raisons, avec un langage de plus haut niveau comme python, le programmeur est libéré de cet aspect de la programmation.

Pour anticiper un peu sur le cours des semaines suivantes, voici ce qu'il peut être utile de garder en tête s'agissant de la gestion mémoire en python:

- vous créez vos objets au fur et à mesure de vos besoins
- vous n'avez pas besoin de les libérer explicitement, le "*Garbage Collector*" de python va s'en charger pour, comme le nom le suggère, recycler la mémoire lorsque c'est possible

- python a tendance à être assez gourmand en mémoire, comparé à un langage de bas niveau, car tout est objet; et chaque objet est assorti de ***méta- informations*** qui occupent une place non négligeable. Par exemple, chaque objet possède notamment
    - une référence vers son type - c'est le prix du typage dynamique;
    - un compteur de références - le nombre d'autres valeurs (variables ou objets) qui pointent vers l'objet. Cette information est notamment utilisée, précisément, par le ***Garbage Collector*** pour déterminer si la place utilisée par un objet peut être libérée ou non;
    - cette liste est non exhaustive...
- un certain nombre de types prédéfinis et non mutables sont implémentés en python comme des ***singletons***, c'est-à-dire qu'un seul objet est créé et partagé; c'est le cas par exemple pour les petits entiers, et les chaînes de caractères;
- lorsqu'on implémente une classe il est possible de lui conférer cette caractéristique de singleton, de manière à optimiser la mémoire nécessaire pour exécuter un programme.