Forum: Moteur de jeu GameBlender et alternatives

Topic: Bonne conception des classes (vertes, pour les enfants :-D)

Subject: Re: Python - l'instruction return

PostÃ© par: DaWaaaaghBabal

Contribution le : 22/3/2018 23:19:21

Citation :
Comment aurai-je pu savoir que le terme "objet" pour toi voulais dire "classe" ? Je ne suis pas télépathe
Hook a utilisé le mot "objet" à bon escient. C'est toi qui as un très gros problème de vocabulaire qu'il va falloir corriger très vite si tu veux comprendre quoi que ce soit à la POO. Je ne suis pas sûr, par exemple, que tu réalises ce que signifie "tout est objet".

Un *objet* est une brique de programme caractérisée par une identité (deux objets identiques restent séparés, de la même façon que deux jumeaux identiques restent deux personnes séparées), un état (des données) et un comportement (des fonctions, appelées méthodes).

Dans la majorité des langages orientés objet, l'état et le comportement d'un objet sont définis par une *classe*, dont il est une *instance*. Une classe déclare les champs et les méthodes possédés par les objets qui l'instancient.

La classe Utilisateur déclare qu'un utilisateur du forum a un nom, un avatar et un mot de passe, et qu'il peut poster_un_message(). DaWaaaaghBabal est une instance de la classe Utilisateur ; son nom est "DaWaaaaghBabal", son avatar est [cf à gauche], son mot de passe est [bien essayé], et il est en train de poster_un_message(ce_message).
DaWaaaaghBabal ***n'est pas*** une classe.

Fun fact : En Python, tout est objet. Donc une classe est un objet aussi. Donc une classe est décrite par une classe. Cette classe ? La classe Class. La classe Class déclare des champs comme "champs déclarés par la classe", "méthodes déclarées par la classe", et des méthodes comme "instancier".

Ici, il n'est absolument pas question de créer une classe dynamiquement. On peut le faire et c'est très rigolo, mais dans ce cas précis on fait quelque chose de beaucoup plus banal :

- On définit la classe Text. Elle déclare des états (en Python, pour déclarer un champ, on l'initialise dans le constructeur, d'où peut-être une partie de ta confusion) mais pas de comportement.
Une instance de la classe Text est donc juste un paquet de données qui ne sait rien faire, un texte à afficher et quelques métadonnées genre la couleur ou la police. Ça permet de bien associer un ensemble de paramètres dans une structure de données très simple.

- On définit la méthode add(...) qui crée un ***objet***, une instance de la classe Text, qui empaquète toutes les informations passées en paramètre. Si on ouvre la documentation, on constate que la méthode append() ne renvoie rien, donc même s'il y a un "return", cette méthode ne renvoie rien non plus. On aurait pu se passer du "return".

- On définit une méthode write(), qui initialise tout ce dont on a besoin pour jouer avec BGL, puis affiche tous les textes qu'on a ajoutés précédemment.

Tout ça est donc simplement une façon de stocker des textes au fur et à mesure (via la méthode add()) avant de tous les afficher d'un coup (via write()), comme le montrent les trois dernières lignes du code.