TP2 : GESTION D'UN MAGASIN DE MUSIQUE

1. Avant de commencer

N'oubliez pas d'utiliser git ! Créez un nouveau projet pour l'occasion (petit tuto disponible sur ametice si vous avez oublié comment faire). Pensez également à tester chacune de vos fonctions !

2. La classe Instrument

(1) Vous allez définir une classe `Instrument` qui sera une classe générique représentant un instrument de musique. Son constructeur prendra en argument un prix, une marque, une année de création, et un volume.

(2) Surchargez la méthode `__str__` afin qu'elle retourne une chaîne de caractères qui vous semble bien représenter un instrument.

(3) Surchargez la méthode `__eq__` afin de tester si deux instruments sont égaux. Deux instruments sont égaux s'ils ont les mêmes valeurs d'attributs.

(4) Créer une méthode `is_new`, qui renverra vrai si l'instrument a été créé durant l'année courante. Attention, votre code doit toujours fonctionner dans 1, 5 ou même 10 ans. Pour ça, cherchez s'il existe un module standard qui vous permet de récupérer la date courante.

3. La classe Piano

(1) Définir une classe `Piano` qui hérite d'`Instrument`. Elle aura un attribut supplémentaire, `number_of_keys`, qui contiendra le nombre de touches d'un piano.

(2) Surchargez la méthode `__str__` de Piano afin qu'elle retourne une chaîne de caractères qui vous semble mieux prendre en compte les nouvelles informations que vous avez sur cet objet.

(3) Les pianos ont une pédale de sourdine qui permet de baisser leur volume. Vous ajouterez également une méthode `muted_volume` qui calcule et renvoie ce volume baissé. Le volume baissé correspond à la racine carré du volume. [1]

4. La classe Store

(1) Définir une classe `Store` qui est composée d'un attribut `turnover`, qui correspond à son chiffre d'affaire (initialisé à 0 au départ), et d'un attribut `list_pianos` représentant les pianos vendus dans le magasin. Vous vérifierez que chaque élément de cette liste est bien unique au moment de l'initialisation.

(2) Chaque élément de cette liste (chaque `Piano` du magasin donc) peut exister en plusieurs exemplaires. Vous aurez une deuxième liste, `stock_pianos`, de même taille que `list_pianos` qui contient le nombre de `Pianos` disponible actuellement dans le magasin. Vous initialiserez cette liste au moment de l'initialisation, en considérant que chaque `Piano` est disponible en un exemplaire dans le magasin. Voir la Table 1 pour un exemple.

(3) Définir une méthode `add(piano)` qui ajoute un `Piano` dans le magasin. Si ce `Piano` est déjà dans la `list_pianos`, alors on incrémente juste le nombre de `Piano` dans le `stock_pianos`. Sinon, on l'ajoute à la liste, ainsi que dans `stock_pianos` avec une valeur de 1.

(4) Définir une méthode `sell(piano)` qui vérifie si le `Piano piano` existe bien en stock. Si c'est le cas, alors on retire un piano du stock, et on ajoute à `turnover` le prix de vente du piano. Si non, on lève une erreur.

(5) Définir une méthode `inventory_value`, qui renvoie la somme totale que le magasin pourrait gagner si il vendait tout ses objets en stock.

1. Attention ! Cette fonction n'est absolument pas représentative de la réalité et est complètement inventée.

list_pianos	p1	p2	p3	...
stock_pianos	0	3	1	...

TABLE 1. Exemples de list_pianos et stock_pianos. Ici, il n'y a plus aucun piano de l'instance p1 disponible. Il reste 3 pianos p2, et un piano p3.