Exercice 2 – Objets et classes en Python

Grand Tournois International de Jeux Vidéos

Corrigez l'atelier

Vérifiez que le fichier gtijv.py soit correct et complet par rapport à ce qui était demandé dans l'atelier 2.

Inspection de Tournament.py

Téléchargez le fichier tournament.py présent sur moodle et sauvez-le dans le même dossier que gtijv.py.

Faisons le tour que ce qui s'y trouve…

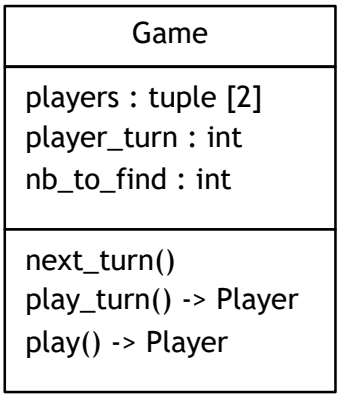
Quelques imports

```
import random
from gtijv import Player, Country
```

Nous avons importé les fichiers nécessaires. Notez que nous avons importé Player et Country du fichier gtijv. Pour que cela fonctionne, il doit se trouver dans le même dossier.

La classe Game

Cette classe représente une partie entre 2 joueurs. Une partie consiste à trouver un nombre entre 1 et 10. Le premier qui y arrive a gagné.



Remarquez la boucle `while not winner` dans la méthode `play()`.
A tout moment du jeu, winner contiendra None ou le joueur qui a gagné.

- None équivaut à False quand on le teste comme un booléen : il est **Falsy**
- Un objet équivaut à True quand on le teste comme un booléen : il est **Truthy**

Pour plus d'infos sur falsy et truthy : https://docs.python.org/2.4/lib/truth.html

La classe Tournament

Cette classe représente un tournoi pour lequel il y a une liste de joueurs participants. Les joueurs vont s'affronter de manière aléatoire à un jeu de type Game. Les perdants sont éliminés un à un. Le dernier en lice est le gagnant.

Remarquez la **précondition** posée avant la méthode `play_random_game()`. C'est un simple commentaire mais il indique que la méthode fonctionnera bien tant que cette condition est respectée. Si la précondition n'est pas respectée, le programmeur ne promet pas que sa méthode fonctionne.

Un autre programmeur qui voudrait utiliser cette fonction devra être prudent.

Le bloc principal

Pour finir, vous trouverez un bloc « main » à la fin du fichier. Celui-ci s'occupe de créer plein d'objet de type différents et de lancer un tournoi.

>>> Lancez le script et voyez ce qui se passe.

Lorsque vous lancez un script qui contient des imports, l'interpréteur va aussi lancer les scripts de ces imports.

Pourtant vous verrez que ce qui se trouvait dans votre bloc « main » du fichier gtijv.py ne s'est pas exécuté.

C'est normal. C'est parce que ce fichier n'était pas lancé comme fichier principal (« main ») donc son nom (`__name__`) ne valait pas « `__main__` ». La condition du if étant dès lors fausse, le contenu du bloc if ne s'est pas exécuté.

Complétez/corrigez tournament.py

1. Dans la méthode `play_random_game()` de la classe `Tournament`, deux joueurs sont choisis aléatoirement parmi la liste des joueurs. Cependant, il y a un risque que ce soit 2 fois le même joueur qui soit choisi.

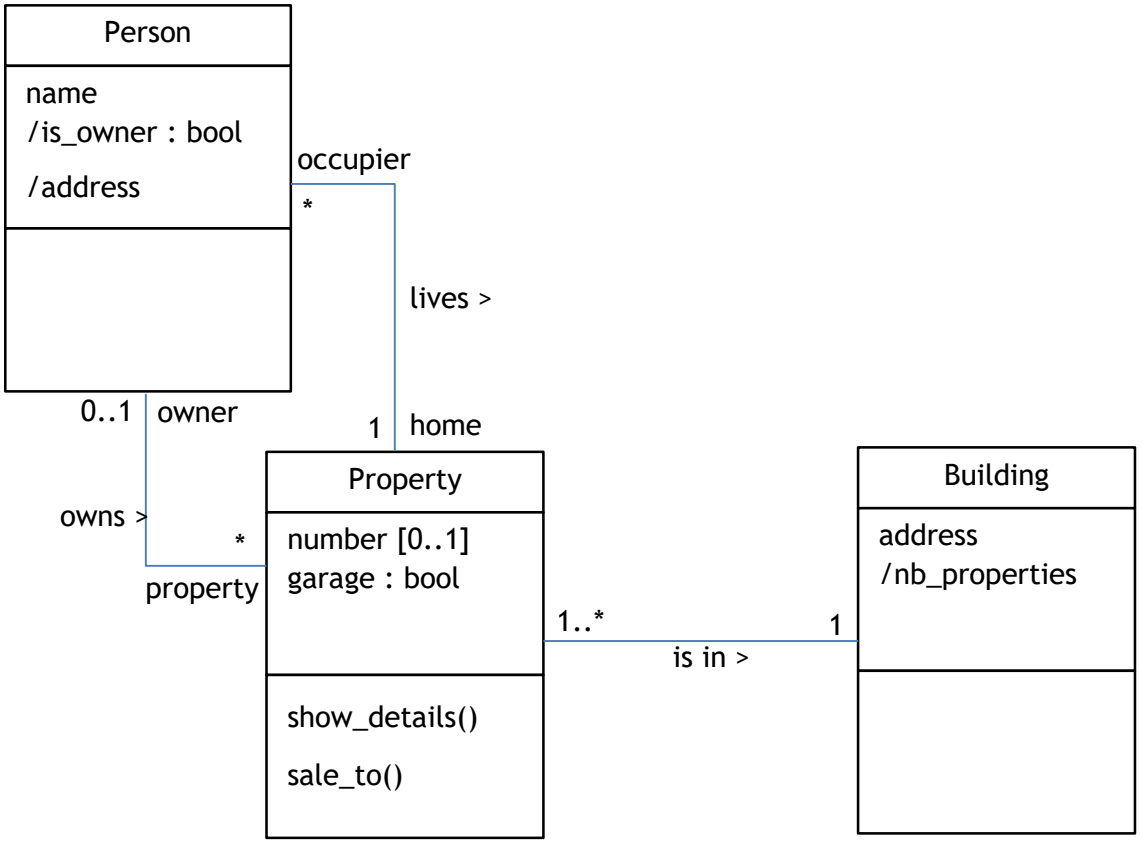
>>> Faites en sorte que ce soit 2 joueurs différents avant de lancer la partie.

2. Dans la méthode `play()` de la classe `Tournament`, le tournois commence et tant qu'il y a au moins 2 joueurs présents dans la liste, une partie est lancée via `play_random_game()`. Pour finir, l'unique joueur restant dans la liste est affiché comme gagnant.

>>> Que se passera-t-il si la méthode `play()` est appelée alors qu'il n'y a aucun participant ?

>>> Affichez « Aucuns inscrits, tournois annulé » s'il n'y a pas au moins 2 participants lorsque `play()` est appelé.

Système de gestion de location de propriétés



Voici le diagramme UML correspondant à l'énoncé suivant :

Une personne a un nom, est propriétaire ou non et a une adresse (celle de l'endroit où il habite). Un immeuble a une adresse et contient un certain nombre de propriétés (au moins 1). Une propriété se trouve dans un immeuble et a éventuellement un numéro (s'il s'agit d'un appartement). On retient également si une propriété possède un garage ou non.

Une personne peut posséder plusieurs propriétés mais il arrive que certaines propriétés n'appartiennent pas à une personne (mais plutôt à une organisation ou à la commune). Chaque personne habite dans une propriété et une propriété peut abriter plusieurs occupants.

On veut également pouvoir afficher les détails d'une propriété et la vendre.

Transformation du schéma

1. Transformez le diagramme UML correspondant à l'énoncé pour retirer les relations et pouvoir le traduire en classes python.

2. Qu'allez-vous faire des attributs dérivables ?

3. Vous noterez que le lien entre Building et Property et de type min1-min1. Quel problème va en découlez ?

Pour régler ce problème, changez la multiplicité du côté de Property en mettant un 0..*.

Traduire en classes Python

Traduisez le schéma intermédiaire en classes python en suivant ces étapes. Pensez à tester régulièrement votre code.

1. Créez 3 classes (`class … :`) et mettez-y le constructeur correspondant.
   Pour `Property`, si la présence su garage n'est pas indiquée, on suppose qu'il n'y en a pas.

2. Implémentez les méthodes correspondant aux attributs dérivables.

3. Implémentez une méthode `add_property` dans Building qui prendra en paramètre une propriété et l'ajoutera à la liste des propriétés de l'immeuble.

4. Réécrivez la méthode `__str__` de `Building` pour que quand on souhaite afficher un immeuble, il s'affiche sous la forme : **Building** *address* **(***nb properties***)**. Pensez à réutiliser la méthode que vous avez déjà écrite.

5. Réécrivez la méthode `__str__` de `Person` pour que quand on souhaite afficher une personne, elle s'affiche sous la forme : *name* **(***home address***)**.

6. Réécrivez la méthode `__str__` de `Property`. S'il s'agit d'un appartement (possède un numéro), il faudra qu'il s'affiche : **Apartment** *number*, *building address*. S'il s'agit d'une maison (ne possède pas de numéro), il faudra qu'il s'affiche : **House** *building address*.

7. Testez ce que vous avez fait. Pour cela, dans le bloc main :
   a. Créez un building `b1` se trouvant à l'adresse « Commonstreet nb 5 » et un building `b2` se trouvant à l'adresse « Magicstreet nb 18 ».
   b. Créez un appartement `ap1` qui se trouve dans l'immeuble `b1` au numéro 1 et qui dispose d'un garage. Ajoutez cet appartement à `b1`.
   c. Créez 3 autres appartements `ap2`, `ap3` et `ap4` se trouvant également dans l'immeuble `b1`, respectivement aux numéros 2, 3 et 4. N'oubliez pas de les rajouter dans `b1`.
   d. Créez une maison `h` qui se trouve dans `b2` et dispose d'un garage. Pensez à ajouter la maison ainsi créée dans `b2`.
   e. Créez 3 personnes : Bob, Alice et Bobby Jr qui vivent ensemble dans la maison sur Magicstreet. Bob est propriétaire de la maison et des appartements `ap1` et `ap2`.
   f. Créez Chris qui habite dans l'appartement numéro 1.
   g. Créez David qui habite et possède l'appartement numéro 3.
   h. Créez Elen qui habite et possède l'appartement numéro 4.
   i. Faites les changements nécessaires dans les propriétés.
   j. Grâce au `print()`, affichez les infos des 2 immeubles, de la maison et de l'appartement numéro 2. Affichez également Bob, Alice et Elen.

Si vous avez correctement programmé jusqu'ici, voici ce qui devrait s'afficher :

```
Building Commonstreet nb 5 (4)
Building Magicstreet nb 18 (1)

House Magicstreet nb 18
Apartment 2, Commonstreet nb 5

Bob (Magicstreet nb 18)
Alice (Magicstreet nb 18)
Elen (Commonstreet nb 5/4)
```

8. Implémentez une méthode `show_details` pour une propriété qui affiche tous les détails de cette dernière comme ceci :

```
>>> h.show_details()
--- House Magicstreet nb 18 ---
Owner : Bob (Magicstreet nb 18)
Occupiers :
- Bob
- Alice
- Bobby Jr
Garage : YES

>>> ap2.show_details()
--- Apartment 2, Commonstreet nb 5 ---
Owner : Bob (Magicstreet nb 18)
Occupiers :
Garage : NO
Nb Apart in Building : 4

>>> ap3.show_details()
--- Apartment 3, Commonstreet nb 5 ---
Owner : David (Commonstreet nb 5/3)
Occupiers :
- David
Garage : NO
Nb Apart in Building : 4
```

9. Dans `Person`, définissez une méthode qui affiche le détail de toutes les propriétés que la personne possède.

10. A la fin du bloc main, appelez la méthode que vous venez de définir pour voir le détail des propriétés de Bob.

11. Écrivez la méthode `sale_to` dans la classe `Property`. Elle prend en argument la personne à qui la propriété est vendue et s'occupe de gérer la vente. C'est-à-dire changer le propriétaire de la propriété, enlever la propriété de celles de l'ancien propriétaire et l'ajouter à celles du nouveau propriétaire.

12. À la fin du bloc `main`, vendez l'appartement 2 à Elen. Affichez ensuite le détail des propriétés de Bob et d'Elen pour vérifier que la vente s'est bien passée.