Atelier 2 – Objets et classes en Python

Cet atelier est prévu pour 2h.

Bonnes pratiques

Pour des projets conséquents, une bonne pratique est de créer un dossier qui contiendra toutes les classes du projet et d'y mettre un fichier par classe. Chaque fichier contiendra alors la définition d'une (et une seule) classe.

De plus, la plupart des projets contiennent un fichier « principal » (« main » en anglais) qui contient le code principal du projet. C'est le code contenu dans ce fichier qui s'occupe de créer différents objets et de les faire interagir.

Il arrive aussi que, pour des projets plus petits, le code principal soit mis à la suite d'une définition de classe. Dans ce cas, il est d'usage de le mettre dans le bloc d'instruction suivant[1] :

```
if __name__ == "__main__" :
    # mon code principal …
```

Pour la suite des exercices, pour des raisons de **lisibilité** et étant donné qu'il ne s'agit pas d'un projet conséquent, nous vous demandons de faire un fichier par exercice et de découper votre code comme suit :

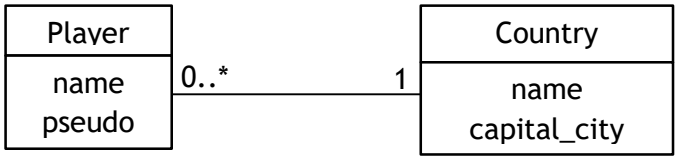
```
class My_first_class :
    # définition de la classe

class My_second_class :
    # définition de la classe

...

if __name__ == "__main__" :
    # code principal
```

Notez que la **définition d'une classe** ne fait que **définir** quelque chose. Si vous ne créez pas d'objet instance de cette classe, il ne se passera rien. La création d'objet se fait en dehors de la classe (dans le code principal).

[1] https://docs.python.org/fr/3/library/__main__.html

Grand Tournois International de Jeux Vidéos

Voici le schéma correspondant au GTIJV



Écrire les classes

Dans un fichier de script, écrivez les classes correspondant à ce schéma en suivant les étapes :

1. Retirer l'association
2. Traduire en Python

Sauvez ce fichier sous le nom gtijv.py.

> Faites bien **attention** à l'attribut qui va devoir stocker la **liste des joueurs** dans un pays. N'hésitez pas à aller revoir dans le cours comment traduire ça en Python.

Tester les classes

Dans le bloc `if __name__ == "__main__" :`

- créez 2 objets de type `Country`, `usa` et `bel`. Pour le moment, ils n'ont pas de joueurs.
- Exécutez votre script et, en console, grâce à `__dict__`, affichez les informations des 2 pays.
- Toujours en console, changez la capitale de `usa` par New York et vérifiez si le changement est fait.

Revenez au script et, à la suite de ce qui est déjà fait, créez 2 joueurs belges et 2 joueurs américains. Ajoutez ensuite les joueurs dans la liste de joueurs du pays correspondant.

Testez votre script et regarder un peu se qui se trouve dans chaque attribut.

Ajouter des méthodes

Ajoutez maintenant une méthode `show_details` dans Pays permettant d'afficher en détail les infos d'un pays suivit de la **liste** des pseudos des joueurs du pays. Par exemple :

Belgique (capitale : Bruxelles) :

- SuperMario

- Jojo007

- ALB3RT

A la suite du bloc « main », affichez les détails de chaque pays en utilisant la méthode que vous venez de creer. Relancez votre script pour vérifier.

En console, testez la commande `print()` et lui donnant un de vos joueurs en paramètre.

- Que signifie ce qui s'affiche ?
- Ce n'est pas très lisible n'est-ce pas ? Comment pouriez vous faire pour que quand vous faites un print avec un joueur ca donne quelque chose de mieux ? Faites-le.

Dans Pays, ajoutez une méthode `add_player` qui prend un joueur en paramètre et l'ajoute à la liste.

Améliorez `add_player` pour que la méthode change automatiquement le pays du joueur en question

En console, affichez un joueur américain. Ensuite, utilisez `add_player` pour l'ajouter à la Belgique. Pour finir, réaffichez le joueur et affichez les détails de la Belgique.

Affichez maintenant les détails des usa… Votre joueur y figure toujours. Enlevez-le.

Pour aller plus loin

Attribut de classe

Dans la classe Player,

- Ajoutez un attribut de classe `nb_players` qui est de 0 au départ.
- Faites en sorte que quand on crée (avec le constructeur) un nouvel objet Player, `nb_players` soit incrémenté de 1.
- Relancez le script et vérifiez ce que vous avez fait en affichant la valeur de l'attribut de classe `nb_player`. Correspond-elle à la réalité ?

Méthode statique

Dans la classe Player, ajouter une méthode statique `play()` qui demande à l'utilisateur d'entrer un nombre entre 1 et 10 et renvoie ce nombre.

- Dans un premier temps, supposez que l'utilisateur rentre bien un nombre entre 1 et 10
- Dans un second temps, tant que le nombre rentré n'est pas entre 1 et 10, redemandez un nombre.