Héritage entre class

Tout est objet

L’héritage

Nous avons vue dans les cours précédant comment créer de nouveaux objet en définissant des classes.

Toutefois ces objet ne dispose que des méthode qu’on leur à explicitement définit.

Ce qui serait intéressant se serait de créer un objet qui se comporte comme un autre objet déjà definit (une liste par exemple), avec les attributs associés, mais en lui ajoutant de nouvelles fonctionnalités.

L’héritage

Revenons à notre premier exemple, la class *superstring.*
Cette classe ne dispose que des méthode qu’on lui à explicitement définit.

Mais nous souhaitons lui associer toute les méthodes relative à une liste sans les redéfinir, la solution est d’utiliser **l’héritage**.

```
class superstring(list):
    def __init__(self, chaine):
        list.__init__(self, chaine)
    def upper(self):
        for (i, c) in enumerate(self):
            self[i] = c.upper()
    def __str__(self):
        return "".join(self)
```

```
s1 = superstring('Debout le dormant')
s1.append('s')
s1.insert(9, 's')
s1.upper()
print(s1)
```

```
DEBOUT LES DORMANTS
```

L’héritage

- Ligne 1 : On passe en argument dans la définition de la class le nom de celle dont elle hérite (ici la class *list*).
- La class *superstring* hérite donc des méthodes *append()* et *insert()* qui appartiennent nativement à la class *list*.
- Ligne 2 et 3 : On défini le constructeur en appelant le constructeur de *list.*
- Si on choisi quand même de définir ces méthodes dans la class fille (celle qui hérite) se seront celle-ci qui seront appliquées.
  C’est pour cela qu’il faut faire attention au nom qu’on choisie.
- Une class fille peut hériter de plusieurs class mère :
  class superstring(mère1, mère2):
  L’ordre des class mère à une importance.

Mise en pratique

1. Créer une class data qui hérite du formulaire et possède un attribut supplémentaire id.
   Une méthode doit permettre d'initialiser cette identifiant comme une combinaison de caractères pris dans chaque autre attribut.
   Exemple :

```
jd = data('Doe', 'Jhon', 1999)
ad = data('Doe', 'Alice', 1991)

jd.buildID()
ad.buildID()
print(jd.id)
print(ad.id)
```

```
DoJh21
DoAl29
```

Mise en pratique

2. *Recensement* est une classe qui prend pour entrée 3 listes de formulaire et qui à pour méthode *permanents()* qui retourne la liste des individus présent dans ces 3 listes.

```
class recensement:
    def __init__(self, l1, l2, l3):
        self.f2020 = l3
        self.f2019 = l2
        self.f2018 = l1

    def permanents(self):
        return [f for f in self.f2020 if
                f in self.f2019 and f in self.f2018]
```

Créer une class *listeelectorale* qui hérite de recensement et possède une méthode renvoyant tout les formulaires présent dans les trois listes et correspondants à des personnes majeurs.