Pense bête 6

Les Classes – Héritage

L'héritage est un mécanisme très puissant qui permet à une classe d'être définie comme fille d'une classe mère.

Dans ce cas, la classe fille hérite de tous les atributs et méthodes de sa classe mère.

Exemple :

```
class Bateau():
    def __init__(self):
        self.tonnage = 0
        self.nom = "monBateau"
        self.a_quai = True
    def accoster(self):
        if self.a_quai:
            print("Le bateau est déjà à quai")
        else:
            self.a_quai = True
            print("Le bateau accoste")
    def quitter_quai(self):
        if not self.a_quai:
            print("Le bateau n'est pas à quai.")
        else:
            self.a_quai= False
#----------------------------------------
b=Bateau()
b.accoster()
b.quitter_quai()
b.quitter_quai()
b.accoster()
```

```
b.accoster()
```

Résultat :

Le bateau est déjà à quai

Le bateau n'est pas à quai.

Le bateau accoste

Le bateau est déjà à quai

Nous créons une classe sous-marin, qui doit pouvoir faire tout ce que fait un bateau, mais qui doit aussi disposer des méthodes : plonger et remonter.

Nous définissons la classe SousMarin comme étant fille de la classe Bateau et nous lui ajoutons les deux méthodes qui lui sont propres.

```
class SousMarin(Bateau):
    def plonger(self):
        print("Je plonge")
    def remonter(self):
        print("je remonte")
```

Utilisation :

```
s=SousMarin()
s.accoster() #appel de la méthode héritée de bateau
s.plonger()
s.remonter()
print(s.nom)
```

Résultats :

Le bateau est déjà à quai

Je plonge

je remonte

monBateau

Nous voyons que le sous-marin **s** a hérité de tous les attributs du bateau et que lors de sa construction, comme nous n'avons pas défini de constructeur propre au sous-marin (def __init__(self) :) c'est le constructeur de la classe Bateau qui a été appelé : le nom du sous-marin est celui donné par le constructeur de la classe Bateau.

Si maintenant nous voulons ajouter des attributs au sous-marin et les initialiser dans un constructeur, pour bénéficier des initialisations de la classe mère, nous devrons appeler explicitement le constructeur de cette dernière.

```
class SousMarin(Bateau):
    def __init__(self):
        #appel du constructeur de la classe mère
        super().__init__()
        self.base=""
        self.en_plonger = False


    def plonger(self):
        if not self.en_plonger :
            self.en_plonger = True
            print("Je plonge")
        else:
            print("Je suis déjà en plongée")


    def remonter(self):
        if self.en_plonger :
            self.en_plonger = False
            print("Je remonte")
        else:
            print("Je suis déjà en surface")
#--------------------------
s=SousMarin()
#appel de la méthode héritée
s.accoster()
```

```
s.plonger()
s.plonger()
s.remonter()
#afficher le nom hérité
print(s.nom)
s.base="Lorient"
print(s.base)
```

Résultat :

```
Le bateau est déjà à quai
Je plonge
Je suis déjà en plongée
Je remonte
monBateau
Lorient
```

Il est possible de redéfinir une méthode de la classe mère. Il suffit dans la classe fille de donner le même nom à la méthode qu'elle redéfinit.