



## Les notions abordées

Au cours de ce TP vous allez aborder les notions de base de la POO comme :

- Le codage de classes et de son constructeur.
- La création d'instances.
- L'héritages de classes.
- La redéfinition (overriding in english) de méthodes.

## Principe de l'héritage : rappels...

Le principe de l'héritage et de la redéfinition est illustré par l'exemple suivant :

```
class Parent:
    def afficher(self):
        print("Méthode de la classe Parent")

class Enfant(Parent):
    def afficher(self): # Redéfinition (overriding)
        print("Méthode de la classe Enfant")

# Test
obj = Enfant()
obj.afficher()
```

On place entre (...) le nom de la classe dont on hérite les éléments...

Il y a redéfinition de la méthode de la classe PARENT lorsque la méthode de la classe ENFANT porte le même nom...

Méthode de la classe Enfant

Il est toutefois possible d'appeler la méthode « *cachée* » par la redéfinition en utilisant la fonction « `super()` ». L'exemple suivant illustre cet aspect :

```
class Parent:
    def afficher(self):
        print("Méthode de la classe Parent")

class Enfant(Parent):
    def afficher(self): # Redéfinition (overriding)
        print("Méthode de la classe Enfant")
        super().afficher()

# Test
obj = Enfant()
obj.afficher()
```

On fait appel à la méthode « `afficher()` » de la classe PARENT...

Méthode de la classe Enfant  
Méthode de la classe Parent



*Remarque : La mise en oeuvre de la fonction « `super()` » est possible pour toutes les méthodes et en particulier pour le constructeur : « `__init__()` »...*



## Partie 1 : la classe PARENT...

Vous allez coder une classe « **Animal** » au sein d'un script « `animal.py` ». Cette classe dispose de DEUX attributs d'instances :

- « **nom** » : Surnom de l'animal.
- « **age** » : Estimation de son âge en années...

Codez le script « `animal.py` » en suivant les étapes suivantes :

1. Créez la classe « **Animal** ».
2. Créez son constructeur qui dispose de deux arguments pour définir les deux attributs de la classe.
3. Créez une méthode « **se\_présenter(self)** » qui affiche un texte ressemblant à celui-là :

**Je me nomme *nom*, j'ai *age* ans...**

4. Testez votre code en y ajoutant en dehors de la classe, le code suivant :

```
animaux = [Animal("Simba", 5), Animal("Beethoven", 3), Animal("César", 26), Animal("Dumbo", 1)]

for animal in animaux:
    animal.se_présenter()
```

## Partie 2 : la classe ENFANT...

Vous allez coder une seconde classe nommée « **Mammifere** » toujours dans le script « `animal.py` ». Cette classe dispose de CINQ attributs d'instances :

- « **nom** » : Surnom de l'animal.
- « **age** » : Estimation de son âge en années...
- « **race** » : Sa race (chien, chat, baleine, etc...).
- « **type\_pelage** » : (poils courts, poils longs, peau nue, etc...).
- « **couleur** » : Celle du pelage/peau en question....

Complétez le script « `animal.py` » en suivant les étapes suivantes :

5. Créez la classe « **Mammifere** » comme ENFANT de la classe « **Animal** » qui prend le rôle de classe PARENT.
6. Créez son constructeur qui fait appel à la fonction « **super().\_\_init\_\_(...)** » pour construire les attributs « **nom** » et « **age** » avant de définir classiquement les attributs propres à cette classe ENFANT : « **race** », « **type\_pelage** » et « **couleur** »...

**Indication :** Les arguments de « **super().\_\_init\_\_(...)** » doivent être cohérents avec le constructeur de la classe PARENT...

7. Testez votre code en apportant la modification suivante à la liste « **animaux** » :

```
animaux = [Mammifere("Simba", 5, "lion", "poils courts", "fauve clair"),
            Mammifere("Beethoven", 3, "chien", "poils longs", "blanche & fauve"),
            Mammifere("César", 26, "singe", "poils courts", "marron"),
            Mammifere("Dumbo", 1, "éléphant", "peau nue", "grise")]
```



*Remarque : Quand bien même les attributs « **nom** » et « **age** » ont été construit par la classe PARENT, ils sont accessible (car hérités) au niveau des instances de la classe ENFANT...*



8. Créez (dans la classe ENFANT) une méthode « **se\_présenter(self)** » qui affiche un texte ressemblant à celui-là :

**Je suis un(e) *race* revêtu de *type\_pelage* de couleur : *couleur*.**

**Résultat :** L’affichage du texte produit par la méthode « **se\_présenter(...)** » de la classe ENFANT se substitue à celui de la classe PARENT. La méthode « **se\_présenter(...)** » de la classe ENFANT a opéré une redéfinition de sa « jumelle » de la classe PARENT...

9. Complétez le code de la méthode « **se\_présenter(...)** » de la classe ENFANT pour avoir un appel de la méthode « **jumelle** » de la classe PARENT...

**Indication :** Utilisez la fonction « **super()** »...

10. Testez le résultat...  
11. Permutez les deux lignes de la méthode « **se\_présenter(...)** » de la classe ENFANT.  
12. Testez le résultat...

## Partie 3 : une seconde classe ENFANT...

Vous allez coder une troisième classe nommée « **Oiseau** » toujours dans le script « **animal.py** ». Cette classe dispose également de QUATRE attributs d’instances :

- « **nom** » : Surnom de l’animal.
- « **age** » : Estimation de son âge en années...
- « **ordre** » : C’est le type d’oiseaux (rapace, perroquet, passereau, etc...)
- « **envergure** » : Valeur en cm.

On aurait aimé mettre « **type** » comme attribut...  
Mais c’est impossible car « **type** » est un mot clé de Python !

Complétez le script « **animal.py** » en suivant les étapes suivantes :

13. Créez la classe « **Oiseau** » comme ENFANT de la classe « **Animal** ».  
14. Créez son constructeur en faisant appel au constructeur de la classe PARENT...  
15. Créez (dans la classe ENFANT) une méthode « **se\_présenter(self)** » qui affiche un texte ressemblant à celui-là :

**Je suis un oiseau de type *ordre* et mon envergure est de *envergure* cm.**

16. Testez votre code en apportant la modification suivante à la liste « **animaux** » :

```
animaux = [ Mammifere("Simba", 5, "lion", "poils courts", "fauve clair"),
            Mammifere("Beethoven", 3, "chien", "poils longs", "blanche & fauve"),
            Mammifere("César", 26, "singe", "poils courts", "marron"),
            Mammifere("Dumbo", 1, "éléphant", "peau nue", "grise"),
            Oiseau("Hedwige", 7, "rapace", 90),
            Oiseau("Blu", 5, "perroquet", 100),
            Oiseau("Lago", 12, "perroquet", 60),
            Oiseau("Zazu", 40, "passereau", 40)]
```

17. Complétez le code de la méthode « **se\_présenter(...)** » de la classe ENFANT pour avoir un appel de la méthode « **jumelle** » de la classe PARENT...  
18. Testez le résultat...