31- Les méthodes dans les objets (Programmation)

I – Rappel

II – Déclaration des méthodes

2.1 Principe de la déclaration

On déclare une méthode comme une fonction si ce n'est :

1. On la place **dans la Classe** (en décalant de 4 espaces du coup)
2. Il FAUT qu'elle possède **un premier paramètre qui pourra recevoir la référence-mémoire de l'objet.** Ce paramètre sera rempli automatiquement lors de l'appel de la méthode. On le nomme souvent **`self`.**

Voici un exemple d'appel de mystere (voir ci-contre) où **`heros`** est une instance de **`Personnage`** :

```
heros.mystere(5)
```

L'interpréteur va exécuter cet appel comme si vous aviez tapé ceci :

```
Personnage.mystere(self=heros, degats=5)
```

2.2 Appel depuis l'intérieur de la classe.

Comme self contient la référence de l'objet en cours d'appel, il suffit de taper ceci :

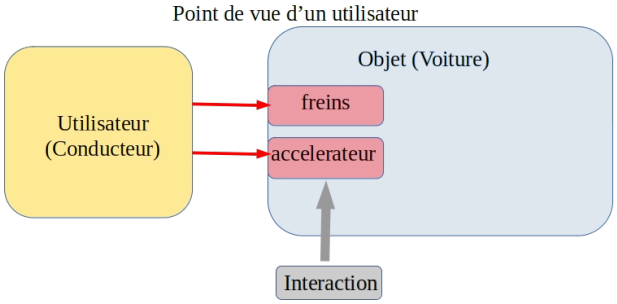
```
self.pas_negatif()
```

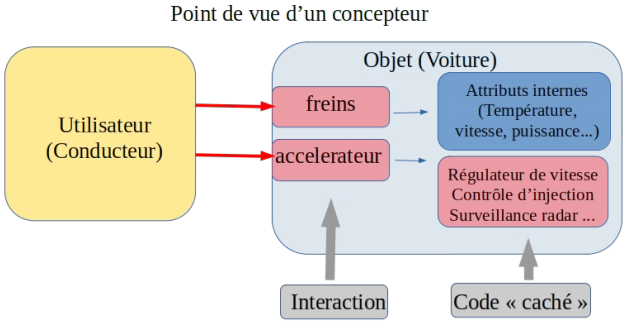
2.3 Principe d'encapsulation

L'utilisateur ne doit avoir accès qu'à un nombre restreint de **méthodes d'interface** lui permettant d'agir sur l'objet de façon contrôlée et sécurisée.
Les méthodes et les attributs qu'un utilisateur peut utiliser directement sont dits **PUBLICS**.

Le concepteur de la classe aura lui accès à d'autres méthodes pour réaliser le fonctionnement des méthodes publics notamment. On dit des méthodes et des attributs uniquement utilisables à l'intérieur de la Classe qu'ils sont **PRIVES.**

```
class Personnage:
    def __init__(self, niveau=0):
        self.niveau = niveau*5

    def mystere(self, degats):
        self.pv = self.pv - degats
        self.pas_negatif()

    def pas_negatif(self):
        if self.pv < 0:
            self.pv=0

heros = Personnage(niveau=10)

>>> heros.pv
50
>>> heros.mystere(5)
>>> heros.pv
45
```





III – Deux catégories de méthodes : les mutateurs et les accesseurs

Mutateur : méthode qui modifie l'état d'un objet (il modifie les attributs)

Accesseur : méthode qui renvoie un résultat correspondant à un attribut ou donnant une idée de la valeur de l'attribut.

IV – Tester le type d'un objet

```
>>> type(heros1) == Personnage
True
```

```
>>> isinstance(heros1, Personnage)
True
```