Programmation Orientée Objet en Python

#2 Encapsulation

par David ALBERT

2023

Table des matières

01

Objets et classes

Objets et classes

Définitions

Un **objet** est une variable. Il faut notamment qu'il soit déclaré avec son type. Le type d'un objet est un type composite (constitué de types primitifs, de fonctions et d'autres objets) => appelé **classe**.

Un **objet** n'est autre qu'une **instance de classe**.

Un **classe** est un **type de données** (au même titre que les entiers, les chaînes de caractères, etc).

Une **classe** regroupe un ensemble de choses:

- des données que l'on appelle **attributs** (variables primitives ou objets)
- de **méthodes** de traitement de ces données et/ou de données extérieures à la classe

Objets et classes

Différence classe et objet

Il n'existe qu'une classe CitroenC3 mais il peut exister de nombreuses instances de cette classe.

```
voitureDidier : CitroenC3 = CitroenC3()
#------------   ---------   -----------
#   objet        classe     constructeur
```

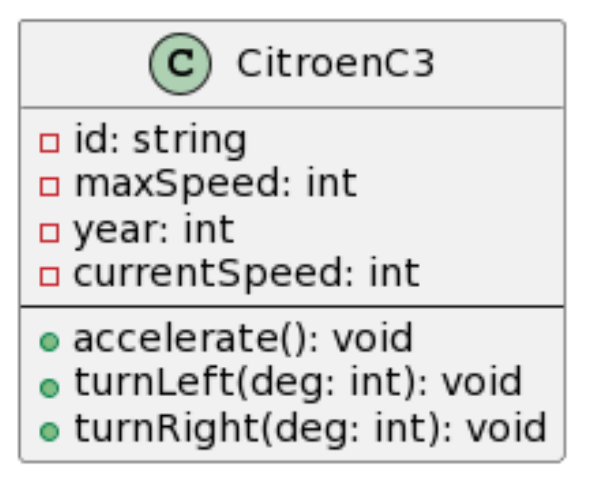
💡 **Pour mieux comprendre**

Classe = Le moule pour fabriquer des objets = type de données contenant des données (attributes) et des fonctions (méthodes)

Objet = une instance de classe (l'objet une fois créé) = une donnée spécifique

Définir une classe

Syntaxe UML



Syntaxe python

```python
class CitroenC3:

    # Constructeur
    def __init__(self):
        self._id = 0
        self._currentSpeed = 0
        self._maxSpeed = 210
        self._year = 2010

    def accelerate(self):
        pass

    def turnLeft(self, deg):
        pass

    def turnRight(self, deg):
        pass
```

Instancier un objet

Syntaxe UML

Syntaxe python

Pour instancier un objet en python on fait appel au constructeur.

```
maVoiture = CitroenC3()
```

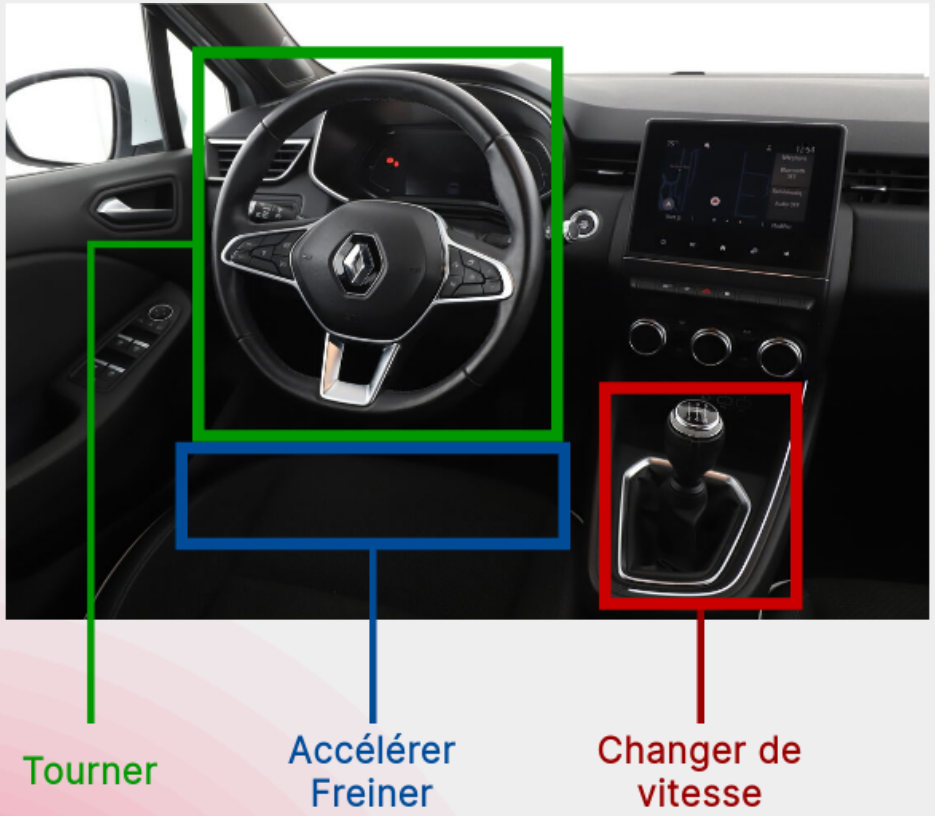
02

Encapsulation

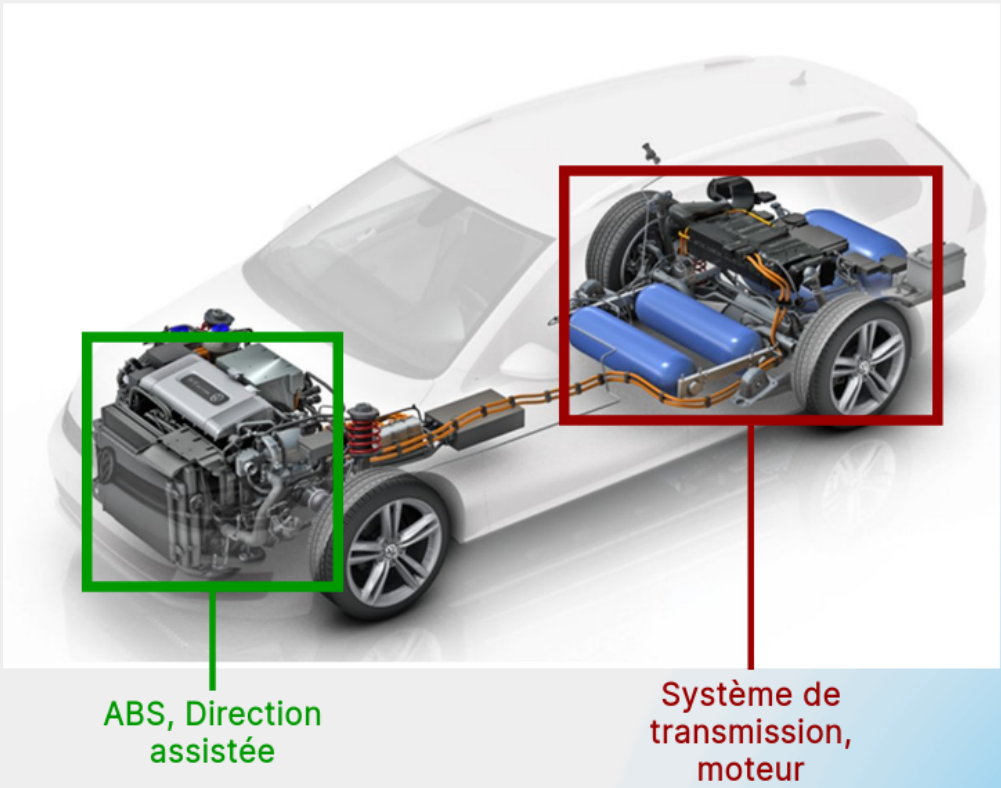
Encapsulation

Princpe

Usage simple et visible



Fonctionnement complexe et caché



Encapsulation

Visibilité des données

Lors de la conception d'un programme orienté-objet, le programmeur doit:

- identifier les objets et les données appartenant à chaque objet
- les droits d'accès qu'ont les autres objets sur ces données

L'encapsulation de données dans un objet permet de cacher ou non leur existence aux autres objets du programme. Une donnée peut être déclarée en accès :

- **public** : les autres objets peuvent accéder à la valeur de cette donnée ainsi que la modifier
- **privé** : les autres objets n'ont pas le droit d'accéder directement à la valeur de cette donnée (ni de la modifier). En revanche, ils peuvent le faire indirectement par des méthodes publiques de l'objet concerné

Encapsulation

Bonnes pratiques

- ne rendre publique que le stricte minimum
  - les fonctions nécessaires à l'usage (`accelerate`, `turnLeft`, `turnRight`)
  - et pas plus (`increaseSpeed`, `turnLeftWheel`)
- suivre le principe de **responsabilité unique** - **S** de **S**OLID
  - exemple Animal / AnimalDB
- n'utiliser que des **attributs privés** donc, si c'est raisonnable de penser qu'on a besoin:
  - de lire leur valeur depuis l'extérieur, on utilise un **getter**
  - de modifier leur valeur depuis l'extérieur, on utilise un **setter**

Encapsulation

Syntaxe python

03

Constructeur

04

Le mot clé : self