# Méthode de classe et méthode statiques

# Attribut de classe

- Un attribut de classe est un attribut qui caractérise la classe et non pas les instances
- Un attribut de classe est un attribut qui est **partagé** (**commun**) avec toutes les instances de cette classe.

```
class Voiture :

    fabriquant="Societe X"

        def methode1(...):
          .....

MaVoiture1=Voiture()
MaVoiture2=Voiture()
```

fabriquant est un attribut de classe

- Les attributs de classe sont définis **au niveau de la classe plutôt qu'à l'intérieur de la méthode __init__(self)**

# Attribut de classe

- En dehors de la classe, on **accède à** un attribut de classe **via le nom de la classe** ou même par le nom de l'instance.

```
print( Voiture.fabriquant )

Ou

print(MaVoiture1.fabriquant)
```

- **Remarque**: c'est conseillé d'utiliser le nom de la classe pour accéder à un attribut de classe

# Attribut de classe dans une méthode

- On accède à l'attribut de classe dans une méthode ou dans un constructeur par **nomclasse.nomattribut**, **soit pour récupérer sa valeur ou bien pour modifier sa valeur**:

```
class Voiture :
    fabriquant="societe X"

    def methode1(self,p1,p2,....):
        so=Voiture.fabriquant                    #récupérer la valeur
        .....
        Voiture.fabriquant="New Societe"         #modifier la valeur
```

- En python, un attribut de classe , on l'appelle aussi un **attribut statique**

# Types de méthodes

- Il existe trois types de méthodes:

1. **Méthodes d'instances** (ont obligatoirement un paramètre self, comme déjà vu)
2. **Méthodes de classe** , qui possède obligatoirement un paramètre cls
3. **Méthode statiques**, ne possède aucun paramètre obligatoire.

# Méthode de classe

- Une méthode de classe est une méthode qui définit un comportement de la classe et non pas d'un objet.  Ce type de méthode est **appelée via la classe et non pas via l'instance.**
- Une méthode de classe doit être précédée par **@classmethod**
- Une méthode de classe, possède un paramètre obligatoire **cls** qui représente le nom de la classe (au lieu de self pour les méthodes d'instances)
- Cette méthode **peut retourner ou pas une valeur et peut accepter ou non des paramètres** comme toutes méthode.

# Méthode de classe

- Exemple: cette méthode va nous permettre de modifier le nom du fabriquant.

```
@classmethod

    def modiFabriquant(cls , nvfab):
        cls.fabriquant=nvfab
```

Accessible via la classe notée **cls**

# Appel de la méthode de classe

- Pour utiliser une méthode de classe, on l'appelle par le nom de la classe comme l'exemple suivant:

```
Class voiture:
        ...
        @classmethod
         def modiFabriquant(cls , nvfab):
              cls.fabriquant=nvfab


V1=voiture("AB5478","ford",80000)
Voiture.modifFabriquant("Societe Y")
```

# Calculer le nombre d'instances crées

- Pour calculer le nombre d'instances crées, il faut:
  1. Créer un compteur initialisé à zéro (attribut de classe)
  2. Incrémenter ce compteur dans le constructeur
  3. Décrémenter ce compteur dans le destructeur
  4. Créer une méthode permettant de retourner la valeur de ce compteur

# Calculer le nombre d'instances crées

```
class voiture:
        c=0
       def __init__(self,matricule, marque,prix):
              voiture.c+=1
              self.matricule=matricule
              self.marque=marque
              self.prix=prix

       def __del__(self):
             voiture.c -=1

       @classmethod
        def  nombreInstances(cls):
                return cls.c

V1=voiture("AB5478","ford",950000)
V2=voiture("AB1254","ford",80000)
V3=voiture("AB9632","renault",140000)
print(voiture. nombreInstances())
```

# Méthode statique

- Une méthode statique est une méthode **utilitaire**( c'est à dire utile dans notre classe ),
- Une méthode statique, ne représente pas obligatoirement une action de la classe ou de l'instance, donc elle n'est pas appelé obligatoirement par un objet ou une classe, c'est-à-dire qu'elle ne possède pas le paramètre **self, ni cls**
- Une méthode statique est précédée par le mot **@staticmethod**

# Méthode statique

- Exemple:

```
class voiture :

@staticmethod
  def calcul(a , b):
      return a+(a*b)

  def prixVente(self , gain , prix):
       self.calcul(prix , gain)
```

- Ici on a définit une méthode statique qui retourne une valeur calculé selon les paramètres fournis(comme une simple fonction utilisée dans la programmation procédurale, sans self et sans cls en paramètre).
- On a utilisé cette méthode statique dans la méthode d'instance prixvente() et on a accédé via self. On peut accéder à une méthode statique aussi via le nom de la classe.