



## Présentation ADRAR Pôle Numérique

Suivez-nous... LinkedIn  et Twitter  @Adrar\_Numerique



# ADRAR\o/ PÔLE NUMERIQUE

PÔLE NUMERIQUE DU CENTRE DE FORMATION ADRAR

- > SUPPORT, ADMINISTRATION SYSTEMES & RESEAUX
- > DEVELOPPEMENT D'APPLICATIONS WEB & MOBILES
- > WEBDESIGN & DEVELOPPEMENT INTERFACES
- > TRANSFORMATION NUMERIQUE DES ENTREPRISES

<http://www.adrar-numerique.com>

Suivez-nous sur LinkedIn  Twitter  @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)



Suivez-nous sur LinkedIn  Twitter  @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

# LA PROGRAMMATION ORIENTEE

## OBJET : POO

Suivez-nous sur LinkedIn  Twitter  @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

La programmation orientée objet (POO) est une méthode de structuration de code permettant de rendre votre programme mieux organisé et plus lisible. Comme vous le savez en python TOUT est objet, c'est donc un langage propice à l'apprentissage de la POO. Quelques définitions :

- Classe : Ensemble de code regroupant des variables et des méthodes permettant de créer des objets.
- Objet : Instance d'une classe disposant d'attributs et de méthodes.
- Constructeur : Méthode qui va être utilisée pour instancier un objet.
- Assesseur (getter) : Méthode qui va être utilisée pour afficher un attribut.
- Mutateur (setter) : Méthode qui va être utilisée pour modifier un attribut.
- Héritage : Principe voulant qu'une classe hérite des propriétés de son parent. De base toutes les classes héritent de la classe Object

Suivez-nous sur LinkedIn 

Twitter  @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

# CLASSES ET OBJETS

Suivez-nous sur LinkedIn  Twitter  @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

Nous pouvons visualiser une classe comme étant une famille d'éléments avec chaque élément représenté par un objet. Chaque instance de la classe aura

Prenons par exemple une classe Voiture.

Classe : Voiture
Attributs :
Couleur : str
Prix : str
nbRoues : int
Méthodes:
Démarrer()
Accélérer(vitesse)

Objet : MaVoiture
Attributs :
Noir
12 000
4
Méthodes:
Démarrer()
Accélérer(vitesse)

Suivez-nous sur LinkedIn  Twitter  @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

Pour pouvoir déclarer notre classe voiture nous avons besoin du mot clé « class ». Si notre classe hérite d'une autre classe il faudra le préciser entre parenthèses lors de la définition de la classe. Par défaut toutes les classes héritent de la class « object », nous pouvons donc omettre sa déclaration.

Ces deux déclarations sont donc identique :

```
class Voiture:  
    #Corp de la classe  
  
class Voiture(object):  
    #Corp de la classe
```



Suivez-nous sur LinkedIn  Twitter  @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

Pour pouvoir créer un objet notre classe a besoin d'une méthode particulière appelé constructeur. Dans le constructeur on va déclarer tous les attributs obligatoires de notre objet. Ce constructeur est la méthode `__init__`. Voici comment il se déclare :

```
class Voiture:
    nbRoues = 4 #Variable de classe

    def __init__(self, couleur, prix):
        self.couleur = couleur #Variables d'instance
        self.prix = prix

MaVoiture = Voiture(couleur="noir", prix="12000")
```

Pour déclarer un objet il suffit d'appeler la classe de l'objet en déclarant ses attributs



Suivez-nous sur LinkedIn Twitter @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

```
class Voiture:
    nbRoues = 4 #Variable de classe

    def __init__(self, couleur, prix):
        self.couleur = couleur #Variables d'instance
        self.prix = prix

MaVoiture = Voiture(couleur="noir", prix="12000")
```

Plusieurs éléments sont à remarquer dans cet exemple:

- Class
- `__init__`
- Self
- Variable de classe
- Variable d'instance

*Class* est le mot clé permettant de créer notre classe.

`__init__` est le nom de notre méthode constructeur.

*Self* fait référence à l'instance de notre classe, l'objet.

Une variable de classe va être commune à toutes les instances de notre classe.

Une variable d'instance va être potentiellement différente à chaque nouvelle instance

Nous commençons notre classe par dire « toutes les instances de ma classe auront toujours 4 roues ».

Dans ce constructeur on retrouve deux lignes : `self.couleur = couleur` et `self.prix = prix`. Ces lignes signifient :

« lorsque je déclare un nouvel objet de ma classe, les paramètres *couleur* et *prix* vont devenir les attributs de mon objet (`self.couleur` et `self.prix`) »

Suivez-nous sur LinkedIn Twitter @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

Pour définir une méthode de classe il suffit d'écrire une déclaration de fonction au sein de la classe (délimité par l'indentation, comme toujours en python).

```
class Voiture:
    nbRoues = 4 #Variable de classe

    def Demarrer(self):
        print("Démarrage de la voiture")
        self.roule = True

    def Accelerer(self, vitesse):
        if not self.roule:
            self.Demarrer()
        self.vitesse = vitesse
        print("Accélération effectuée, nous roulons à un vitesse de {}km/h".format(self.vitesse))
```

Omission du constructeur pour plus de lisibilité

Suivez-nous sur LinkedIn  Twitter  @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

Nous pouvons ensuite créer notre objet Voiture et utiliser ses méthodes :

```
MaVoiture = Voiture(couleur="noir", prix="12000")
MaVoiture.Accelerer(100)
Voiture.Accelerer(MaVoiture, 100)
```

Les deux syntaxes (*MaVoiture.Accelerer(100)* et *Voiture.Accelerer(MaVoiture, 100)* ) sont équivalente. La deuxième syntaxe vous permet de comprendre l'utilisation du self.

La méthode *Accelerer* prend deux paramètres : self et vitesse. C'est une méthode de la classe voiture. On appelle donc la méthode *Accelerer* de la classe Voiture en lui indiquant l'objet Voiture à accélérer et la vitesse.

Sachez cependant que la première syntaxe est la plus utilisée. On commence par indiquer l'objet Voiture concerné, puis la méthode à utiliser avec ses paramètres (sans le self qui a été indiqué en début de ligne)

Suivez-nous sur LinkedIn  Twitter  @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

La déclaration précédente va donner ce résultat :

```
Démarrage de la voiture  
Accélération effectuée, nous roulons à un vitesse de 100km/h  
Accélération effectuée, nous roulons à un vitesse de 100km/h
```

Pour rendre la méthode Demarrer() correcte j'ai du modifié mon constructeur :

```
def __init__(self, couleur, prix):  
    self.couleur = couleur #Variables d'instance  
    self.prix = prix  
    self.roule = False
```

# EXERCICE

Classe : Compte

Attributs :

Date création : int

Solde : int

Propriétaire: str

Banque: str

Méthodes:

Créditer(montant)

Débiter(montant)

Afficher(solde)

Créez la classe Compte dans un fichier réalisez un script dans un autre fichier qui demande le nom de l'utilisateur pour lui créer son compte et qui lui propose par la suite de créditer son compte, de le débiter ou d'afficher son solde.

Après une action de l'utilisateur le script lui demande si il a terminé. Le script ne se termine que si l'utilisateur répond oui.  
Le compte appartiendra à la banque portant votre nom.