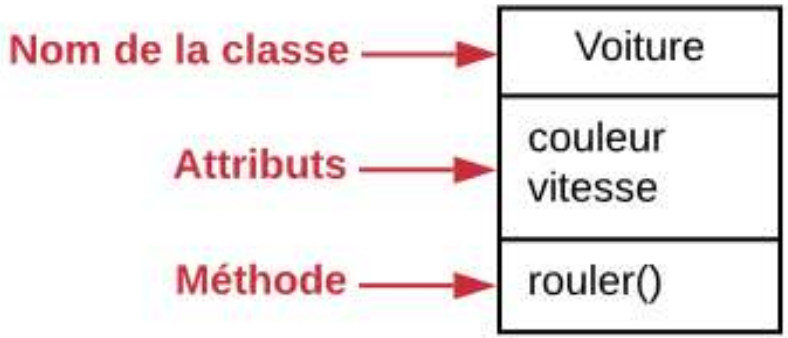
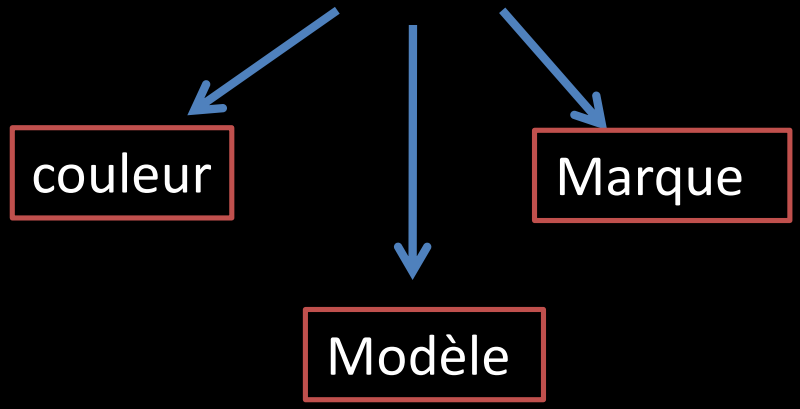
Brève introduction à la Programmation Orientée Objet

Une classe définit des attributs et des méthodes.
Imaginons une classe Voiture qui servira à créer des objets qui sont des voitures.
Cette classe va pouvoir définir un attribut couleur, un attribut vitesse, etc. Ces attributs correspondent à des propriétés qui peuvent exister pour une voiture. La classe Voiture pourra également définir une méthode rouler(). Une méthode correspond en quelque sorte à une action, ici l'action de rouler peut être réalisée pour une voiture.



un objet est une instance de classe

nous pourrons avoir plusieurs voitures qui seront chacune des instances (objets) bien distinctes, on va les distinguer par les valeurs des attributs ou propriétés relatives à la classe voiture.



Exemple

Définition de la Classe "point"

```
class Point:          #Définition d'un point géométrique
```

=> le nom identifiant une classe débute par une majuscule

Création d'un objet de type Point

```
Point()                    #création d'un objet de type Point
```

Pour créer une instance, on utilise le nom de la classe puis parenthèses

Affectation à une variable de la référence à un objet

```
p = Point()
```

```
>>> print(p)
<__main__.Point instance at 0x012CAF30>
```

Le message renvoyé par Python indique que p contient une référence à une instance de la classe Point, qui est définie elle-même au niveau principal du programme. Elle est située dans un emplacement bien déterminé de la mémoire vive, dont l'adresse apparaît ici en notation hexadécimale.


```
a = Point()
b = Point()
```

```
>>> print(a)
<__main__.Point instance at 0x012CADC8>
```

```
>>> print(b)
<__main__.Point instance at 0x012CAF08>
```

Nous avons ici 2 instances de la classe Point (2 objets) :

=> la première à laquelle on fait référence par la variable a,
=> la seconde à laquelle on fait référence par la variable b.

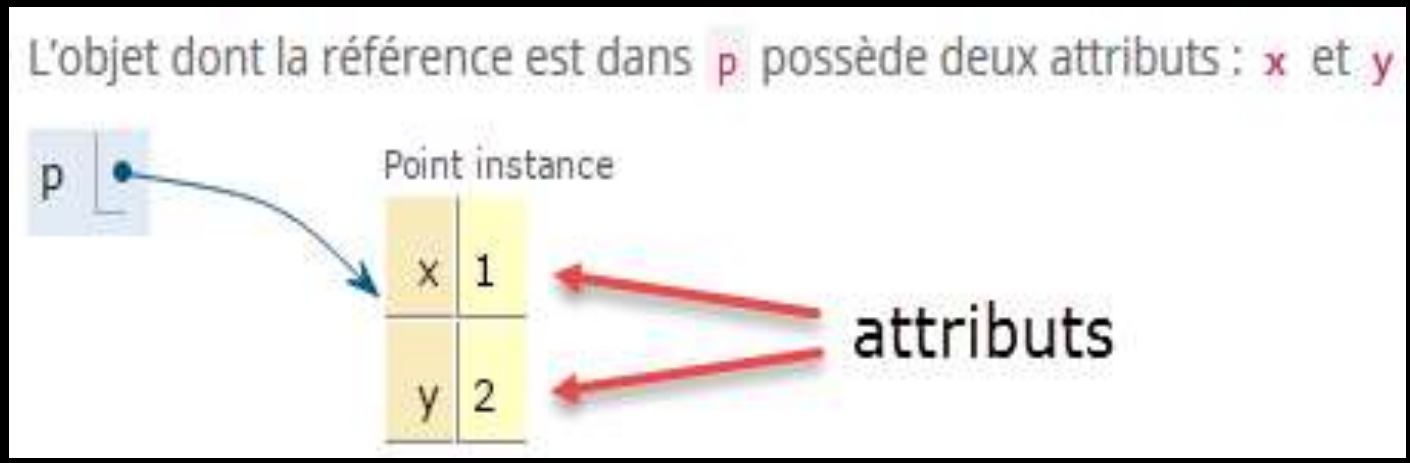
Définition des attributs

```
class Point:
    "Definition d'un point geometrique"

p = Point()
p.x = 1
p.y = 2
print("p : x =", p.x, "y =", p.y)
```

p : x = 1 y = 2

La syntaxe pour accéder à un attribut est la suivante :
on va utiliser la variable qui contient la référence à l'objet et on va mettre un point . puis le nom de l'attribut.


L'objet dont la référence est dans p possède deux attributs : x et y
p
Point instance
x
1
y
2
attributs


```
class Point:
    "Definition d'un point geometrique"

a = Point()
a.x = 1
a.y = 2
b = Point()
b.x = 3
b.y = 4
print("a : x =", a.x, "y =", a.y)
print("b : x =", b.x, "y =", b.y)
```

Par abus de langage on parlera parfois de l'objet a alors qu'il s'agira en fait de *l'objet auquel a fait référence*.

Définition des méthodes

```
class Point:
    def deplace(self, dx, dy):
        self.x = self.x + dx
        self.y = self.y + dy
```

On vient de définir la méthode "deplace"

Pour définir une méthode:
1- indiquer son nom (ici deplace())
2- indiquer les arguments entre des parenthèses. Le premier argument d'une méthode doit être ***self***

Pour accéder à une méthode:

1- le nom de la variable qui fait référence à cet objet

2- un point

3- le nom de la méthode

```
a.deplace(3, 5)
```

La notion de constructeur

Si lors de la création d'un objet nous voulons qu'un certain nombre d'actions soit réalisées (par exemple une initialisation), nous pouvons utiliser un **constructeur**. C'est une méthode sans valeur de retour qu'on appelle _init_(). Elle est appelée lors de la création de l'objet

```
class Point:
    def __init__(self, abs, ord):
        self.x = abs
        self.y = ord

a = Point(1, 2)
print("a : x =", a.x, "y =", a.y)
```