

# Chapitre 7 : Classe

Yves Guidet pour Édugroupe

V1.4.8 April 28, 2017

Python est un langage objet, comme l'«ancêtre» Smalltalk, Ada95, Java ou C++ ; ces langages existent depuis les années 80, mais là encore, ce qui est plus rare, c'est qu'il s'agit d'un langage de script.

- ▶ la double origine de l'Objet
  - ▶ encapsulation
  - ▶ héritage
- ▶ objet = "anything you can do things to"
- ▶ classe = type
- ▶ objet = instance d'une classe
  - ▶ état, comportement et identité

- ▶ un paragraphe d'un document
- ▶ une fenêtre d'une station de travail
- ▶ une politique d'ordonnancement

# une classe Python

```
>>> class Verbe:
...     "un verbe à conjuguer"
...
>>> dir(Verbe)
['__doc__', '__module__']
>>> v = Verbe()
>>> dir(v)
['__doc__', '__module__']
>>> print v.__doc__
un verbe à conjuguer
>>>
>>> v
<__main__.Verbe instance at 0xb7ebfe2c>
```

Noter la majuscule. Elle est conseillée pour les classes, déconseillée ailleurs (voir le bouquin de Swinnen).

- ▶ plus simplement dit : *attributs*

```
>>> v.infinifif = "programmer"
```

- ▶ première apparition de la *notation pointée*

- ▶ fonctions « encapsulées »

```
>>> class Verbe:  
...     "comme d'hab"  
...     def conjuguerPresent (self):  
...     # à suivre
```

- ▶ noter l'autoréférence (ne pas oublier le *self*)
- ▶ la notation pointée pour les méthodes :

```
v.conjuguerPresent();
```

- ▶ *main* par défaut
- ▶ chaque classe possède le sien
- ▶ chaque objet possède le sien

```
>>> v=Verbe()  
>>> dir(v)  
['__doc__', '__module__', 'conj']  
>>> v.conj  
<bound method Verbe.conj of <__main__.Verbe instance at 0xb...
```



- ▶ spécialisation
- ▶ plus de propriétés (attributs, méthodes)

```
class ObjetVolant:  
...  
class Avion (ObjetVolant):  
...  
class Oiseau (ObjetVolant):  
...
```

Pour la suite, voir les bouquins gratuits, et le tuto Django :  
<http://www.formation-django.fr/python/programmation-objet.html>

## une autre classe : Point

Rien de très original, on définit des *Points* avec :

- ▶ données membre : *x*, *y* flottants
- ▶ méthodes : *printit* (*print* est un mot-clé en Python2) et *translate()* qui reçoit un *delta\_x* et un *delta\_y*.

Un joli (?) diagramme :

### Point

-x: float

-y: float

+print()

+translate(in dx:float,in dy:float)

# classe Point (suite)

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

class Point :
    "un point du plan"
    def __init__(self, x=0, y=0) :
        self.x = x
        self.y = y

    def printit (self) :    # print motclé
        print 'x = ', self.x, 'y = ', self.y

    def translate (self, dx, dy) :
        self.x += dx
        self.y += dy
```

On notera, comme pour les fonctions, la *doc*.

Répétons que le premier argument des méthodes (y compris le *constructeur* `__init__`) est toujours l'objet « courant » *self*.

Ces quelques lignes à la fin de *Point.py* nous permettent de voir que tout va bien.

```
p = Point(1, -1)
p.printit()
p.translate(1, 1)
p.printit()
```

Peut-on importer Point ? Bien sûr, comme on a déjà vu pour les *modules*.

Concevoir et implémenter une classe Cercle.  
Les plus courageux pourront compléter la classe Verbe.