

Thibault
HILAIRE

thibault.hilaire@lip6.fr

Fadwa Rekik

rekikfadwa@gmail.com

MAIN - Python

CM4

MAIN — 2016-2017



A decorative border on the left side of the page, featuring a dark blue background with overlapping, semi-transparent geometric shapes in various shades of blue and purple. The shapes include squares, rectangles, and triangles, creating a complex, layered effect. The right edge of this border is jagged and irregular, resembling torn paper or a rough cut.

Les classes

On a vu ensemble des variables de type *simples* :

- entiers, réels
- booléens

Des types plus compliqués mais contenant qu'un type de données :

- les chaînes de caractères
→ on a de nombreuses **méthodes** s'appliquant sur ces **objets**

Et des types plus complexes, agrégeant de manière ordonnée ou pas, différentes valeurs

- listes, tuples
- dictionnaires

Motivations

Exemple : on cherche à stocker des informations correspondant à des personnes

- nom,
- prénom,
- date de naissance,
- adresse

On peut utiliser une liste

```
toi = [ "Dupont", "André", (22,11,1987), "rue_de_la_  
pompe, 75016_Paris"]  
moi = [ "Hilaire", "Thibault", (27,10,1977), "rue_de_  
la_soif, 35000_Rennes"]
```

On accèderait au nom par `toi[0]`, `moi[0]`, au prénom par `toi[1]`, `moi[1]`, etc.

C'est une approche intéressante, mais un peu limitée (surtout si le nombre de champs devient grand).

On peut aussi éventuellement utiliser un dictionnaire

```
toi = { "nom": "Dupont", "prénom": "andré", "naissance": (22,11,1987), "adresse": "rue_de_la_pompe, 75016_Paris" }
moi = { "nom": "Hilaire", "prénom": "Thibault", "naissance": (27,10,1977), "adresse": "rue_de_la_soif, 35000_Rennes" }
```

C'est un peu mieux, plus long, mais pas encore idéal.

L'idéal serait d'avoir un *objet* "Personne" que l'on pourrait manipuler, avec des *méthodes* qui s'appliquent dessus, pour l'afficher, le transformer, etc. Exactement comme on fait pour des listes ou chaînes de caractères.

Classe

Une **classe** est un type de données qui contient des variables (attributs, propriétés) et des méthodes

Objet

Un **objet** (ou une **instance** de classe) est **un** représentant particulier (avec des attributs donnés) d'une classe

`"totolito"` est un *objet* de la classe des chaînes de caractères.
`[12,3,"huhu"]` est un *objet* de la la classe des listes.

Des méthodes, communes à tous les objets de la classe des chaînes de caractères, peuvent s'appliquer sur l'objet `"totolito"`.

Des méthodes, communes à tous les objets de la casse des listes, peuvent s'appliquer sur l'objet `[12,3,"huhu"]`

Ex : On peut imaginer une classe *Personne* qui contiendrait les attributs :

- nom, prénom
- date de naissance
- adresse

Et avec (par exemple) des méthodes

- `calculeAge()` qui calcule l'âge d'une personne
- `affichePersonne()` qui affiche convenablement une personne

on peut aussi imaginer que les opérateurs `>` et `<` puissent comparer des personnes (par rapport à leur date de naissance)

Les variables `toi` et `moi` seraient des *instances de la classe* *Personne*, des **objets**

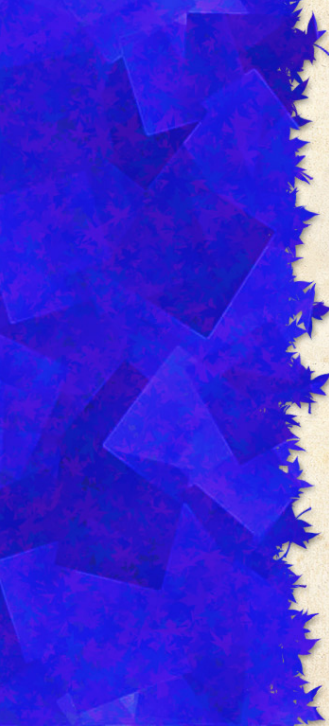
Une fois la classe `Personne` déclarée, on pourra écrire

```
toi = Personne( "Dupont", "andré",  
               (22,11,1987), "rue_de_la_pompe", 75016,  
               Pairs" )  
age = toi.calculeAge()  
print toi.affichePersonne()
```

La création d'un objet de la classe `Personne` se fera avec le nom de la classe suivie (entre parenthèse) des arguments nécessaires. L'appel de méthodes sur l'objet `toi` se fait comme tout appel de méthode sur un objet, avec la notation `.`

De même, on peut accéder aux différents champs (attributs) avec la notation `.`

```
print moi.nom  
toi.naissance = (22,11,1988)
```

A decorative border on the left side of the page, featuring a blue background with overlapping geometric shapes (squares, triangles) and a white star-like pattern.

Création de classe

En Python, une classe se déclare avec le mot-clé `class`

```
class Personne:  
  
    """La classe Personne permet de définir une personne,  
    avec son nom, prénom, date de naissance et adresse"""  
  
    ...
```

Il faut maintenant la remplir.

Cela consiste en définir les différentes *méthodes* de classes, qui seront vues *comme* des fonctions définies à l'intérieur de la classe

Constructeur

Il y a une méthode *particulière* qu'il faut forcément écrire. Il s'agit du **constructeur** qui est appelé au moment où l'on construit l'objet. On veut par exemple construire une *Personne* à partir de 3 chaînes de caractères (nom, prénom, adresse) et un tuple (date de naissance)

```
class Personne:
    """La classe Personne...."""
    def __init__( self, unNom, unPrenom,
                  uneDate, uneAdresse)
        """Constructeur de la classe Personne"""
        self.nom = unNom
        self.prenom = unPrenom
        self.naissance = uneDate
        self.adresse = uneAdresse
```

`self` n'est **pas** un mot-clé de python, **mais** une convention.

Chaque méthode devra avoir un 1^{er} élément correspondant à l'objet sur lequel la méthode s'applique. On le note habituellement `self`

```
class Personne:  
    ...  
    def calculeAge( self):  
        """calcule_l'âge_d'une_personne"""  
        return 2015 - self.naissance[2]
```

A decorative border on the left side of the page, featuring a blue background with overlapping geometric shapes (squares and rectangles) and a dark blue, star-like or floral pattern along the right edge.

Exemple

Exemple

On va écrire une classe pour représenter et utiliser des polynômes. Un polynôme sera représenté par une liste de réels (le 1er terme correspondant au coefficient de plus grand degré). Par exemple $[1, 2, 3]$ correspondra au polynôme $x^2 + 2x + 3$.

En plus du constructeur, on écrira les méthodes

- pour afficher un polynôme
- pour connaître son degré
- pour calculer un polynôme en un point
- pour additionner deux polynômes
- pour obtenir son dérivé

Exemple

On commence par écrire le constructeur

```
class Polynome:
    """Classe représentant un polynôme.
    Un polynôme est représenté par la liste
    de ces coefficients (le 1er terme
    correspond au coefficient du degré le plus
    haut) """

    def __init__( self, listeCoefs ):
        """Constructeur du polynôme
        listeCoefs est une liste de
        coefficients (1er terme correspondant au
        terme de plus haut degré) """
        self.coefs = listeCoefs
```

Exemple

Pour le calcul du degré du polynôme, c'est assez simple.
On écrit une méthode qui renvoie le degré d'un polynôme

```
def degre( self):  
    """Calcule et renvoie le degré d'un  
        polynôme"""  
    return len(self.coefs) - 1
```

Exemple

Pour l'affichage d'un polynôme, on utilise une autre méthode particulière (commence et finit par `__`) : la méthode `__str__()`.

C'est elle qui est appelée quand on veut convertir un objet quelconque en chaîne de caractères (ce qui est le cas quand on veut afficher un polynôme avec `print`).

`__str__` prend comme paramètre l'objet en question, et renvoie une chaîne de caractères

Exemple

```
def __str__( self):
    """Transforme un objet de la classe
       polynôme en chaîne de caractères
       pour l'afficher"""
    s = "P(x)_"
    d = self.degre()
    for i in range(d+1):
        s = s + "%fx^%d" % (self.coefs[i], d-i)
        if i<d:
            s = s + "_+"
    return s
```

Exemple

On écrit une méthode qui évalue un polynôme en un point x

```
def evaluate( self, x):  
    """Évalue un polynôme en un point x  
    """  
    res = 0  
    for c in self.coefs:  
        res = res*x + c  
    return res
```

Exemple

Pour le calcul et renvoie le dérivé du polynôme.

```
def derive( self):  
    """Retourne le polynôme dérivé"""  
    d = self.degree()  
    L = list( self.coefs )  
    L.pop()  
    for i in range(d):  
        L[i] = L[i] * (d-i)  
  
    return Polynome(L)
```

Exemple

Enfin, on peut utiliser un (des) objets de la classe polynome

```
P = Polynome( [1,2,3,4] )  
print( P )  
print( "P(1)=" + str( P.evaluate(1) ) )  
print( "P(2)=" + str( P.evaluate(2) ) )  
Q = P.derive()
```

$$P(x) = 1.000000x^3 + 2.000000x^2 + 3.000000x^1 + 4.000000x^0$$
$$P(1) = 10$$
$$P(2) = 26$$
$$P(x) = 3.000000x^2 + 4.000000x^1 + 3.000000x^0$$