Espaces de noms / Portée des noms

Builtins

directement dans Python

abs apply **bool** buffer callable **chr** classmethod cmp coerce compile complex delattr **dict dir divmod** Ellipsis **enumerate** eval execfile **False** filter getattr **globals** hasattr hash **help hex id input int** intern isinstance issubclass iter **len list locals long** map **max min None** object **oct open ord pow** property **range raw_input** reduce reload **repr** reversed **round set** setattr slice sorted staticmethod **str sum** super **True tuple type** unichr unicode vars **xrange** zip
+ les exceptions standard

Globales

au niveau d'un module

```
from math import pi,sin,cos                  # Noms importés
import random

MAX_POINTS = 5000                            # Constante
index_courant = 0                            # Variable globale

def calcul_index(table) :                    # Fonction
    ...
class Point(object) :
    compteur = 0                             # Attribut de classe
    def __init__(self,x,y) :
        self.x,self.y = x,y                  # Attributs d'instance
    def distance(self,autre) :               # Méthode
        ...
    ...
def calcul_offset(table,reference) :
```

Locales

au niveau d'une fonction/méthode

```
# Les paramètres table et reference sont des locales.
global index_courant    # Pour pouvoir le modifier.
ofs = 0
for i in len(table) :
    p = Point(table[i][0],table[i][1])
    ...
index_courant = index_courant + 1
...
# Dans le cas d'une méthode, le premier paramètre self est une locale
# référençant l'objet auquel on applique la méthode.
```

Quand on utilise directement un nom pour trouver la valeur associée, il est recherché dans les locales puis dans les globales puis dans les builtins. S'il n'est trouvé nulle part, une exception `NameError` est levée.
Quand on utilise directement un nom dans une fonction/méthode pour fixer la valeur associée, il est défini par défaut dans les locales - a moins d'avoir précisé « `global nom` » auparavant.
Il est possible d'accéder à certains noms par indirections :

```
self.x  Point.compteur
```

Dans une méthode, on trouve aussi les valeurs des attributs parmi ceux de la classe :

```
print self.compteur
```