Introduction à l'Objet en Python

Eric BERTHOMIER

eric.berthomier@free.fr

30 septembre 2017

Ce cours est une adaptation libre du tutoriel
Object Oriented programming in Python (Universität Tübingen) [1]

1. http://abi.inf.uni-tuebingen.de/Teaching/Old/SS11/BILW/handouts-1/ObjectOrientedprogramminginPython.pdf

Classe - Class

Définition

On appelle classe la structure d'un objet, c'est-à-dire, la déclaration de l'ensemble des entités qui composeront un objet.
Source : http://www.commentcamarche.net

Snake.py

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

class veryBigSnake:

    # Quelques methodes bien utiles
    def manger (self):
        # Le code viendra par la suite
        pass

    def dormir(self):
        # Le code viendra par la suite
        pass

    def mourir(self):
        # Le code viendra par la suite
        pass
```

Création d'un objet

Définition

Un objet est une instance d'une classe.
La classe est le moule qui permet la fabrication des objets.

objSnake.py

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

# On importe la classe de l'objet
from Snake import veryBigSnake

# On cree un objet de type VeryBigSnake
python = veryBigSnake()

print (type (python))
```

./objSnake.py

```
<class 'Snake.veryBigSnake'>
```

Plusieurs instances

objSnakes.py

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

from Snake import veryBigSnake

kaa = veryBigSnake()
persifleur = veryBigSnake()
dregs= veryBigSnake()

print (type (kaa))
print (type (persifleur))
print (type (dregs))
```

./objSnakes.py

```
<class 'Snake.veryBigSnake'>
<class 'Snake.veryBigSnake'>
<class 'Snake.veryBigSnake'>
```

Méthodes (1/2)

Définition

Une méthode est une fonction associée à un objet.

Snake2.py

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

class veryBigSnake:

    # Quelques methodes bien utiles
    def manger (self):
        print ("Je mange (Miam)")

    def dormir(self):
        print ("Je dors (Dodo)")

    def mourir(self):
        print ("Arghhh ! Je me meurs !")
```

Méthodes (2/2)

objSnakes2.py

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

from Snake2 import veryBigSnake

kaa = veryBigSnake()
persifleur = veryBigSnake()
dregs= veryBigSnake()

kaa.manger()
persifleur.dormir()
dregs.mourir()
```

./objSnakes2.py

```
Je mange (Miam)
Je dors (Dodo)
Arghhh ! Je me meurs !
```

Attributs (1/2)

Définition

Un attribut est une caractéristique d'un objet.

Snake3.py

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

class veryBigSnake:

    def set_snake_name (self, name):
        self.name = name

    def manger (self):
        print ( self.name + "mange (Miam)")

    def dormir(self):
        print (self.name + "dort (Dodo)")

    def mourir(self):
        print ("Arghhh ! " + self.name + "se meurt !")
```

Attributs (2/2)

objSnakes3.py

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

from Snake3 import veryBigSnake

persifleur = veryBigSnake()
kaa = veryBigSnake()

kaa.set_snake_name("Kaa")
persifleur.set_snake_name("Persifleur")

print ( kaa.name + " " + persifleur.name )
```

./objSnakes3.py

```
Kaa Persifleur
```

Constructeur (1/2)

Définition

Le constructeur est utilisé pour initialiser des variables ou exécuter des méthodes au moment de la création de l'objet.

Ajout dans Snake3.py ⇒ Snake4.py

```
class veryBigSnake:

    # constructeur
    def __init__(self):
        # initialize properties
        self.name = "Unnamed Python"
        self.type = "python"
        print ("New snake in the house!")

    def set_snake_name(self, name):
        self.name = name

    def set_snake_type(self, type):
        self.type = type

    def who_am_i(self):
        print ("My name is " + self.name + ", I'm a " + self.type + " and I'm perfect for you!
               Take me home today!")
```

Constructeur (2/2)

objSnakes4.py

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

from Snake4 import veryBigSnake

persifleur = veryBigSnake()

persifleur.who_am_i()
```

./objSnakes4.py

```
New snake in the house!
My name is Unnamed Python, I'm a python and I'm perfect for you! Take me home today!
```

Destructeur

Définition

Le destructeur est très peu utilisé en Python mais il existe. Ce dernier sert à exécuter des commandes juste avant la destruction d'un objet. [a]

a. Notamment utile si vous mélangez les langages

Ajout dans Snake4.py

```
    # destructor
    def __del__(self):
        print ("Just killed the snake named " + self.name + "!")
```

./objSnakes4.py

```
New snake in the house!
My name is Unnamed Python, I'm a python and I'm perfect for you! Take me home today!
Just killed the snake named Unnamed Python!
```

Héritage (1/4)

Définition

L'héritage permet d'utiliser une classe existante pour la compléter et/ou la modifier.

WaterSnake.py

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

from Snake4 import veryBigSnake

class waterSnake(veryBigSnake):

    # function to swim
    def swim (self):
        print (self.name + " swim")
```

Héritage (2/4)

objWaterSnake.py

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

from WaterSnake import waterSnake

persifleur = waterSnake("persifleur")

persifleur.swim()

persifleur.who_am_i()
```

./objWaterSnake.py

```
New snake in the house!
persifleur swim
My name is persifleur, I'm a python and I'm perfect for you! Take me home today!
Just killed the snake named persifleur!
```

Héritage (3/4)

Définition

Le mot clé super() permet de faire appel à la classe Parent.

SpecialWaterSnake.py

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

from Snake4 import veryBigSnake

class SpecialWaterSnake(veryBigSnake):

    def __init__ (self, status="Aucun"):
        super().__init__ ("goldorak")
        self.status = status

    def who_am_i(self):
        super().who_am_i()
        print ("Mon status est : " + self.status)
```

Héritage (4/4)

objSpecialWaterSnake.py

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

from SpecialWaterSnake import SpecialWaterSnake

goldorak = SpecialWaterSnake("Mode combat")
goldorak.set_snake_type ("robot")
goldorak.who_am_i()
```

./objSpecialWaterSnake.py

```
New snake in the house!
My name is goldorak, I'm a robot and I'm perfect for you! Take me home today!
Mon status est : Mode combat
Just killed the snake named goldorak!
```

Surchage (1/2)

Définition

La surcharge consiste à écraser une méthode héritée par une nouvelle.

WaterSnake2.py

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

from Snake4 import veryBigSnake

class waterSnake(veryBigSnake):

    # function to swim
    def swim (self):
        print (self.name + " swim")

    # function to display name and type
    def who_am_i(self):
        print ("My name is " + self.name + ", I'm a " + self.type + " and I'm perfect for you!
               Take me home today!")
        print ("I can swim !")
```

Surcharge (2/2)

objWaterSnake2.py

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

from WaterSnake2 import waterSnake

persifleur = waterSnake()

persifleur.swim()

persifleur.who_am_i()
```

./objWaterSnake2.py

```
New snake in the house!
Unnamed Python swim
My name is Unnamed Python, I'm a python and I'm perfect for you! Take me home today!
I can swim !
Just killed the snake named Unnamed Python!
```

Propriétés d'un objet : dir

Définition

La fonction dir permet de connaître les propriétés et les méthodes d'un objet.

dirWaterSnake2.py

```
#!/usr/bin/python3
# -*- coding: UTF-8 -*-

from WaterSnake2 import waterSnake

persifleur = waterSnake()

print (dir (persifleur))
```

./dirWaterSnake2.py

```
New snake in the house!
['__class__', '__del__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__
      ', '__ge__', '__getattribute__', '__gt__', '__hash__', '__init__', '__le__', '__lt__', '
      __module__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__
      ', '__sizeof__', '__str__', '__subclasshook__', '__weakref__', 'name', 'set_snake_name', '
      set_snake_type', 'swim', 'type', 'who_am_i']
Just killed the snake named Unnamed Python!
```

Un mot sur Import

Import

En programmation objet, on utilise (**doit utiliser**)

```
import NomModule
```

Constructeur

Pour indiquer où se trouve la définition d'un objet, on utilise :

```
MonInstance = NomModule.Classe()
```

Exemple

CChanson.py

```
class Chanson(object):

    def __init__(self, chant):
        self.chant = chant

    def chante_moi_une_chanson(self):
        for line in self.chant:
            print (line)
```

chanson.py

```
#!/usr/bin/python3

import CChanson

chanson = CChanson.Chanson ("chant")
chanson.chante_moi_une_chanson()
```

Nomenclature évoluée

Définition

_ peut être utilisé comme variable de substitution.

underscore.py

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-

n = 42
for _ in range(n):
    print ("coucou")
```

Définition

_ indique que la fonction ou la variable est **privée**.

__ est transformé en **_classname___...** .

À vous de jouer !