Programmation en Python (VI)

Akka Zemmari
Hervé Hocquard
herve.hocquard@u-bordeaux.fr

LaBRI, Université de Bordeaux - CNRS

17 septembre, 2023

université de BORDEAUX

Plan

Ce que nous verrons dans ce cours :

- Opérateurs de base
- Types de variables
- Nombres
- Chaines de caractères
- Listes
- Tuples
- Dictionnaire
- Branchements conditionnels
- Boucles
- Fonctions
- Modules
- Fichiers I/O
- Exceptions
- Classes et Objets

université de BORDEAUX

Les classes

- Une classe = type permettant de regrouper dans la même structure :
  - les informations (champs, propriétés, attributs) relatives à une entité,
  - les procédures et fonctions permettant de les manipuler (méthodes).
- Jargon :
  - Classe est la structure
  - Objet est une instance de la classe
  - Instanciation corréspond à la création d'un objet

université de BORDEAUX

Les classes : définition et implémentation

```python
#Il est préférable de déclarer la classe dans
#un module (ModuleEmplye.py par exmple)
class Employe:
    """Classe Employé """

    #Constructeur
    def __init__(self):
        #Les champs de la classe
        self.matricule = '0000'
        self.nom = ''
        self.age = 0

        #Puis les méthodes
        #...
```

- `class` est un mot clé permettant de définir la structure
- `Employe` est le nom de la classe ici `"""Classe Emplyé"""` sert à documenter la classes
- `__init__` est une méthode standard appelée constructeur, automatiquement appelée lors de l'instaniation
- `self` représente l'instance elle-même, elle doit apparaître en première position dans la définition de toutes les méthodes

Instanciation et utilisation dans le programme principal

```
1  #On charge le module
2  import ModuleEmployes as em
3  #instanciation
4  e = em.EmployePersonne()
5  #affiche tous les membres de e
6  print(dir(e))
7
8  #affectation aux champs
9  e.matricule = input("Matricule :")
10 e.nom = input("Nom : ")
11 p.age = int(input("Age : "))
12
13 #affichage
14 print(e.matricule, ", ", e.nom,", ", e.age)
```

- Il faut d'abord importer le module contenant la classe, nous lui attribuons l'alias `em` ici
- Pour la création de l'instance `e`, nous spécifions le module puis le nom de la classe
- ligne 4, Il y a `()` parce que c'est bien une méthode que nous appelons à la création : le constructeur
- Le paramètre `self` du constructeur n'est pas à spécifier sur l'instance

université de BORDEAUX

Les classes : Ajout de méthodes supplémentaires

```python
#Il est préférable de déclarer la classe dans
#un module (ModuleEmplye.py par exmple)
class Employe:
    """Classe Employé """

    #Constructeur
    ...
    #saisie des infos
    def saisie(self):
        self.matricule = input("Matricule : "))
        self.nom = input("Nom : ")
        self.age = int(input("Age : "))
    #fin saisie

    #affichage des infos
    def affichage(self):
        print("Matricule : ", self.matricule)
        print("Nom ", self.nom)
        print("Age : ", self.age)
    #fin affichage
```

Les classes : Ajout de méthodes supplémentaires

```python
#Il est préférable de déclarer la classe dans
#un module (ModuleEmplye.py par exmple)
class Employe:
    """Classe Employé """

    #Constructeur
    ...
    #saisie des infos
    def saisie(self):
        self.matricule = input("Matricule : "))
        self.nom = input("Nom : ")
        self.age = int(input("Age : "))
    #fin saisie

    #affichage des infos
    def affichage(self):
        print("Matricule : ", self.matricule)
        print("Nom ", self.nom)
        print("Age : ", self.age)
    #fin affichage
```

```python
#On charge le module
import ModuleEmploye as em
#instanciation
e = em.Employe()
#saisie
e.saisie()
#méthode affichage
e.affichage()
```

Les classes : Ajout de méthodes paramétrées

Rajouter la méthode `retraite()` qui calcule le nombre d'années avant l'âge limite de la retraite.

```
1  #Il est préférable de déclarer la classe dans
2  #un module (ModuleEmplye.py par exmple)
3  class Employe:
4      """Classe Employé """
5      ...
6      #reste avant retraite
7      def retraite(self, limite):
8          reste = limite - self.age
9          if (reste < 0):
10             print("Vous êtes à la retraite")
11         else:
12             print("Il vous reste
13                     %s années" % (reste))
14     #fin retraite
```

Les classes : Ajout de méthodes paramétrées

Rajouter la méthode `retraite()` qui calcule le nombre d'années avant l'âge limite de la retraite.

```
#Il est préférable de déclarer la classe dans
#un module (ModuleEmplye.py par exmple)
class Employe:
    """Classe Employé """
    ...
    #reste avant retraite
    def retraite(self, limite):
        reste = limite - self.age
        if (reste < 0):
            print("Vous êtes à la retraite")
        else:
            print("Il vous reste
                    %s années" % (reste))
    #fin retraite
```

Programme principal :

```
#On charge le module
import ModuleEmploye as em
#instanciation
e = em.Employe()
#saisie
e.saisie()
#méthode affichage
e.affichage()
#reste avant retaite
e.retraite(62)
```

Les classes : Ajout de méthodes paramétrées

Rajouter la méthode `retraite()` qui calcule le nombre d'années avant l'âge limite de la retraite.

```
#Il est préférable de déclarer la classe dans
#un module (ModuleEmplye.py par exmple)
class Employe:
    """Classe Employé """
    ...
    #reste avant retraite
    def retraite(self, limite):
        reste = limite - self.age
        if (reste < 0):
            print("Vous êtes à la retraite")
        else:
            print("Il vous reste
                  %s années" % (reste))
    #fin retraite
```

Programme principal :

```
#On charge le module
import ModuleEmploye as em
#instanciation
e = em.Employe()
#saisie
e.saisie()
#méthode affichage
e.affichage()
#reste avant retaite
e.retraite(62)
```

A l'exécution :

```
Matricule : AP001
Nom : Durand
Age : 51
Il vous reste 11 années
```

Collection d'objets

Nous avons déjà vu que Python propose des outils pour la gestion des collections d'objets hétérogènes (tuples, listes, dictionnaires). On peut alors les utiliser pour les instances des classes :

```
1  #On charge le module
2  import ModuleEmploye as em
3  #Liste vide
4  liste = []
5  n = int(input('Nombre d'employés :'))
6  #On saisie la liste
7  for i in range(0:n):
8      e = em.Employe()
9      e.saisie()
10     liste.append(e)
11 #On affiche la liste
12 for e in liste:
13     print('--------------------')
14     e.affichage()
```

université de BORDEAUX

Collection d'objets

Nous avons déjà vu que Python propose des outils pour la gestion des collections d'objets hétérogènes (tuples, listes, dictionnaires). On peut alors les utiliser pour les instances des classes :

```
#On charge le module
import ModuleEmploye as em
#Liste vide
liste = []
n = int(input('Nombre d'employés :'))
#On saisie la liste
for i in range(0:n):
    e = em.Employe()
    e.saisie()
    liste.append(e)
#On affiche la liste
for e in liste:
    print('--------------------')
    e.affichage()
```

A l'exécution :

```
Nombre d'employés : 2
Matricule : AP001
Nom : Durand
Age : 51
Matricule : AP002
Nom : Dupont
Age : 35
--------------------
Matricule : AP001
Nom : Durand
Age : 51
--------------------
Matricule : AP002
Nom : Dupont
Age : 35
```

UX

Collection d'objets : Accès indicé et modification

```
1  #accès par numéro
2  numero = int(input("Numéro de l'employé
3              à traiter :"))
4  if (numero < len(liste)):
5      e = liste[numero]
6      e.age = e.age + 1
7      #affichage de nouveau
8      print("---------- de^but affichage 2")
9      for e in liste:
10         print("-----------------")
11         e.affichage()
12 else:
13     print("indice non valable")
```

Collection d'objets : Accès indicé et modification

```
1  #accès par numéro
2  numero = int(input("Numéro de l'employé
3              à traiter :"))
4  if (numero < len(liste)):
5      e = liste[numero]
6      e.age = e.age + 1
7      #affichage de nouveau
8      print("---------- de^but affichage 2")
9      for e in liste:
10         print("-----------------")
11         e.affichage()
12 else:
13     print("indice non valable")
```

A l'exécution :

```
1  Numéro de l'employé à traiter : 1
2  ---------- début affichage 2
3  -----------------
4  Matricule : AP001
5  Nom : Durand
6  Age : 52
7  -----------------
8  Matricule : AP002
9  Nom : Dupont
10 Age : 35
```