Langage Python

Cours 3/5 : Paradigmes de programmation

Hubert Godfroy

12 novembre 2015

La dernière fois…

- Utilisation comme langage de script (séance 1)
- Structures de données pour différents usages (séance 2)

La dernière fois…

- Utilisation comme langage de script (séance 1)
- Structures de données pour différents usages (séance 2)

⇝ Quels sont les autres usages de Python ?

Quesako ?

Un paradigme de programmation est un style fondamental de programmation informatique qui traite de la manière dont les solutions aux problèmes doivent être formulées dans un langage de programmation.

Wikipedia

Plan

Programmation impérative structurée

Programmation orientée objets

Programmation fonctionnelle

Plan

Paradigme déjà vu

- Celui des premiers langages de programmation (C, Basic, Pascal, …)
- Le code est structuré en fonctions.
- Les boucles de contrôles sont `if`, `for`, `while`.
- Les `goto` sont prohibés.

Go To Statement Considered Harmful

Schéma de base

Un programme écrit dans ce paradigme suit le modèle suivant.

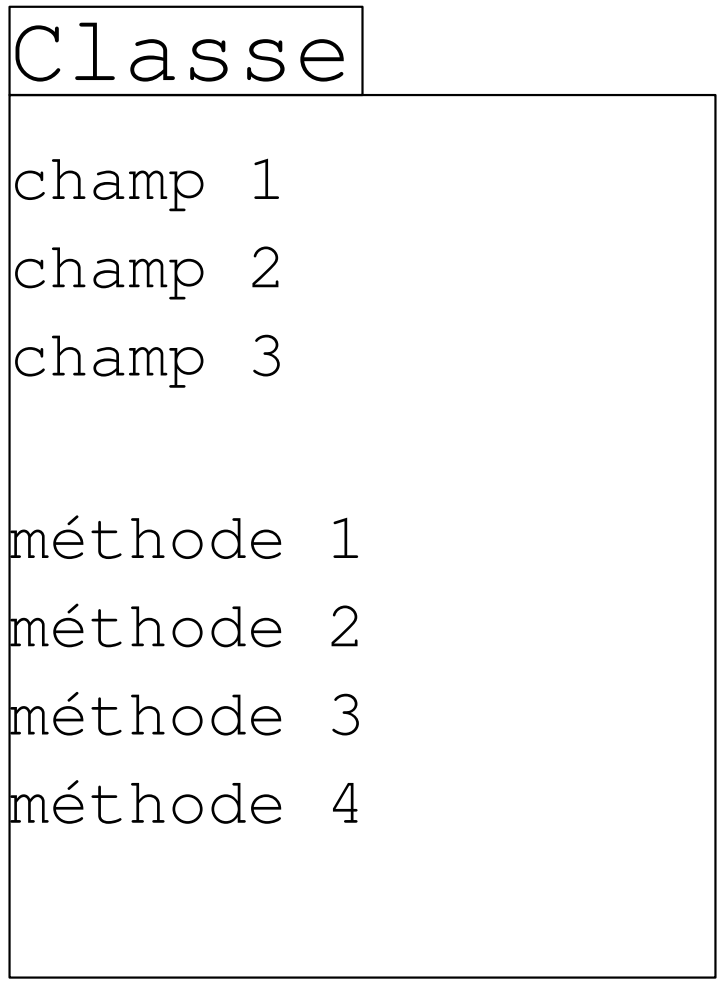
```
def fun1(...):
  ...
def fun2(...):
  ...
def fun3(...):
  ...
def main(...):
  ...
main(...)
```

Plan

POO en général

- Moyen pour stratifier les définitions de fonctions
- Chaque objet dispose de ses propres fonctions.
- Permet de cacher les détails d'implémentation
- On se concentre sur les relations entre les objets.

Une classe en général



POO en Python

- La définition d'une classe suit le modèle

```
class Name:
  field1 = v1
  field2 = v2
  field3 = v3

  def methods1(self):
    ...
  def methods2(self, ...):
    ...
```

- Les champs et les méthodes sont appelées avec "`.`" : si `aName` est un objet de la classe `Name`, on peut faire :

```
var1 = aName.field1
aName.field2 = var2
aName.methods1()
```

POO en Python

- La définition d'une classe suit le modèle

```
class Name:
  field1 = v1
  field2 = v2
  field3 = v3

  def methods1(self):
    ...
  def methods2(self, ...):
    ...
```

- Les champs et les méthodes sont appelées avec "`.`" : si `aName` est un objet de la classe `Name`, on peut faire :

```
var1 = aName.field1
aName.field2 = var2
aName.methods1()

aName.field4 = var4
```

POO en Python

- La définition d'une classe suit le modèle

```
class Name:
  field1 = v1
  field2 = v2
  field3 = v3

  def methods1(self):
    ...
  def methods2(self, ...):
    ...
```

- Les champs et les méthodes sont appelées avec "`.`" : si `aName` est un objet de la classe `Name`, on peut faire :

```
var1 = aName.field1
aName.field2 = var2
aName.methods1()

aName.field4 = var4
```

- On l'objet fait référence à lui même avec le mot clef `self`

Construction d'objets

- Construction d'un objet de la classe `Name` :

```
aName = Name()
```

Construction d'objets

- Construction d'un objet de la classe `Name` :

```
aName = Name()
```

- La fonction `Name()` est un constructeur.
- Il est généré automatiquement par défaut.

Construction d'objets

- Construction d'un objet de la classe `Name` :

```
aName = Name()
```

- La fonction `Name()` est un constructeur.
- Il est généré automatiquement par défaut.
- On peut changer sa valeur par défaut :

```
class Name:
  __init__(self, a, b):
    ...

aName = Name(a, b)
```

Héritage

- Minimiser la quantité de code
- Programmation générique
- Classification/stratification des objets

Héritage en Python

```
class Base:
  ...
class Derive(Base):
  ...
```

- La classe `Derive` hérite de toutes les méthodes et champs de la classe `Base`.
- La classe `Derive` peut redéfinir une méthode de la classe `Base` : c'est la surcharge

Surcharge

Exemple

```
class Base:
    def __init__(self):
        print("init Base")
    def affiche(self):
        print("affiche dans Base")
    def initial(self):
        print("initial dans Base")
```

Surcharge

Exemple

```
class Base:
    def __init__(self):
        print("init Base")
    def affiche(self):
        print("affiche dans Base")
    def initial(self):
        print("initial dans Base")

class Derive(Base):
    def __init__(self):
        print("init Derive")
    def affiche(self):
        print("affiche dans Derive")
    def terminal(self):
        print("terminal dans Derive")
```

Surcharge

- Comment faire appel à la méthode `affiche()` à partir de la classe `Derive` ?

Surcharge

- Comment faire appel à la méthode `affiche()` à partir de la classe `Derive` ?
- On utilise le nom de la classe parente :

```
class Derive(Base):
  def affiche(self):
    print("affiche dans Derive et je sais
        utiliser la methode afficher() de la
        classe parent :")
    Base.affiche(self)
```

Méthodes statiques

On utilise `@staticmethod`.

```
class Name:
  @staticmethod
  def static_method(x):
    ...

Name.static_method(x)
```

Plan

Paradigme fonctionnel

- On voit les fonctions comme des objets.
- Fonctions anonymes au même titre que les valeurs anonymes (`2`, `"Hello World!"`)
- Peut être passé en paramètre d'autres fonctions (fonctions d'ordre supérieur)

Fonctions anonymes

Deux façon de définir la fonction $f : x \mapsto 2x$

```
def f2(x):
    return x*2

f1 = lambda x: x * 2
```

Compréhension

Génération intelligente et concise de listes

- `range(n)` génère une liste d'entiers de 0 à $n-1$.

Compréhension

Génération intelligente et concise de listes

- `range(n)` génère une liste d'entiers de 0 à $n-1$.
- En maths : $\{i \mid i \in [\![0, n-1]\!]\}$
- En python : `[i for i in range(10)]`

Autres exemples

Mathématiques	Python
$\{i * 2 \mid i \in [\![1, 10]\!]\}$ $\{i \mid i \in [\![1, 10]\!] \wedge i \equiv 0[2]\}$	

Compréhension

Génération intelligente et concise de listes

- `range(n)` génère une liste d'entiers de 0 à $n-1$.
- En maths : $\{i \mid i \in [\![0, n-1]\!]\}$
- En python : `[i for i in range(10)]`

Autres exemples

Mathématiques	Python
$\{i*2 \mid i \in [\![1,10]\!]\}$ $\{i \mid i \in [\![1,10]\!] \wedge i \equiv 0[2]\}$	`[i*2 for i in range(10)]`

Compréhension

Génération intelligente et concise de listes

- `range(n)` génère une liste d'entiers de 0 à $n-1$.
- En maths : $\{i \mid i \in [\![0, n-1]\!]\}$
- En python : `[i for i in range(10)]`

Autres exemples

Mathématiques	Python
$\{i*2 \mid i \in [\![1,10]\!]\}$	`[i*2 for i in range(10)]`
$\{i \mid i \in [\![1,10]\!] \wedge i \equiv 0[2]\}$	`[i for i in range(10) if i % 2 == 0]`