Python: Philosophie et bonnes pratiques

Félix-Antoine Bourbonnais

fbourbonnais@rubico.info

18 novembre 2006

Université Laval,
Québec

Première partie

Introduction

Qu'est-ce que Python

http://www.python.org/

- Le Python
  - Inventé par Guido van Rossum
  - Python Software Foundation (PSF)
  - PSL (Python Software License)
    - Proche de BSD
    - Compatible GPL
  - Fonctionne sous une multitude de plates-formes
    - Linux, BSD, ...
    - Windows
    - etc

Caractéristiques

http://www.python.org/

- Caractéristiques du langage
  - Interprété / interactif
  - **Dynamiquement** typé
  - **Fortement** typé
  - Ramasse-miettes
  - Multiparadigmes
    - Programmation structurée
    - Fonctionnelle
    - **Orientée objets**
    - **Orientée aspects**
  - Haut niveau
  - Forte introspection
  - Minimaliste
    - Beau, Explicite, Simple

La syntaxe

- Syntaxe:
  - claire et dégagée
  - Explicite
- Notes:
  - En python, les méthodes sont aussi des attributs.

```
class MyClass(MyBaseClass):
    """Documentation de la classe"""

    def __init__(self):
        """Documentation de l'init."""
        MyBaseClass.__init__(self)
        self.egg = 1

    def addIt(self, value):
        """Documentation de la méthode"""
        self.egg += value

>> inst = MyClass()
>> inst.addIt(6)
>> inst.egg
7
```

Le langage

Module
On croit à tord que cette notion brise l'encapsulation objet. En fait, on peut voir le module comme une classe avec des attributs et méthodes statiques.

Classe Farm qui **hérite** de la classe FoodProduction

Méthode « **privée** » (par convention)

Documentation (DocString)

Food.py:

```
class Farm(FoodProduction):
    """Une ferme qui a des poules"""
    def __init__(self):
        """Init. la ferme"""
        MyBaseClass.__init__(self)
        self.hen = 0
        self._newHen()
        self.egg = 1

    def _newHen(self):
        """Nouvelle Poule"""
        self.hen += 1

    def addEgg(self, value):
        """La poule a pondu un oeuf"""
        self.egg += value

farms = []
def createFarms():
    farms.append(Farm())
```

Préjugés

- Python brise l'orientation objet
  - Les modules peuvent être vus comme des classes dont les attributs sont statiques.
  - Les fonctions dans « builtins » font appel à des méthodes membres des objets (__cmp__, __len__, ...)
- Python n'est pas typé
  - Au contraire, il est dynamiquement typé mais **fortement typé**.
  - Les conversions de type sont effectuées via des méthodes membres

```
>> a = '1'
>> b = 2

>> a + 2
TypeError: cannot concatenate 'str'
and 'int' objects

>> type(a)
<type 'str'>

>> isinstance(a, int)
False

>> class Egg:
>>     pass

>> a = Egg()
>> a.abc()
AttributeError: Egg instance has no
attribute 'abc'
```

Petits trucs simples

- help(x)
  - Affiche la documentation (docstring)
- dir(obj)
  - Affiche les attributs d'un objet
- Toujours garder un interpréteur en mode interactif ouvert

Les piles et les recharges

- Pour le web:
  - Cherrypy: Framework web HTTP fortement orienté objet
  - Django, Turbogears: Frameworks web haut niveau
- Pour le réseau
  - Twisted: Framework pour concevoir des applications réseau
- Pour développer / déployer:
  - PyDev: Plugin pour Eclipse
  - SQLObject: Interface oritentée objet pour une base de données (Object Relational Mapper)
  - SQLAlchemy
  - easy_install

Première partie

Pourquoi Python ?

Avantages

http://www.python.org/

- Avantages
  - Apprentissage rapide (simplicité)
  - Syntaxe explicite et propre
  - Code facile à maintenir
  - Rapide à coder (but rapidement atteint)
  - Réduction du temps de développement
  - Flexibilité
  - Nombreuses reliures (bindings)
- Bien adapté pour:
  - Les projets en eXtreme Programming ou AGILE
  - Se combine très bien avec le développement piloté par les tests (Test driven development)
  - Projet nécessitant une infrastructure légère, simple mais efficace

Désavantages et éléments neutres

http://www.python.org/

- Désavantages
  - Nécessite de la discipline de la part des programmeurs
  - Le programmeur doit savoir ce qu'il fait...
  - IDE moins performant que Eclipse avec Java
- Neutre (avantage ou non selon le contexte):
  - Dynamiquement typé
  - Dynamiquement interprété

Succès

http://www.python.org/

- Succès
  - Google (plusieurs API, etc)
    - "Python has been an important part of Google since the beginning, and remains so as the system grows and evolves. Today dozens of Google engineers use Python, and we're looking for more people with skills in this language." said Peter Norvig, director of search quality at Google, Inc. [5]
  - NASA
    - "NASA is using Python to implement a CAD/CAE/PDM repository and model management, integration, and transformation system which will be the core infrastructure for its next generation collaborative engineering environment. We chose Python because it provides maximum productivity, code that's clear and easy to maintain, strong [...]" said Steve Waterbury, Software Group Leader, NASA STEP Testbed [5]
  - Mailman
  - Zope
  - ...

Troisième partie

Philosophie

La philosophie Python

http://www.python.org/

- Framework très complet
  - « Python comes with batteries included »
- Les développeurs ne sont pas stupides
  - « [...] bad behavior should be discouraged but not banned. If you really want to change the value of None, you can do it, but don't come running to me when your code is impossible to debug. » [1]
  - Rien n'est fait pour empêcher le programmeur de faire quelque chose. Y compris accéder et modifier les objets système.
- Duck Typing
  - « If it looks like a duck and quacks like a duck, it must be a duck. »
  - On ne veut pas savoir la classe d'une instance mais plutôt si elle a la méthode à appeler

Bonnes pratiques

http://www.python.org/

```
class Spam(object):
  '''
  Contains egg's datas
  '''

  def foo(self):
    'Counts my eggs'
    return self._cnt

  def bar(self, n):
    'Sets some eggs'
    self._cnt = n

  egg=property(foo, bar)

>> help(Spam)
>> Spam().egg = 2
   # bar(2) a été
     exécuté
```

- Respecter la « règle du canard » [4]
  - Éviter `isinstance` au profit de `hasattr`
- Éloigner le réflexe XML [2]
  - Contrairement au Java, XML est utilisé presque uniquement par interopérabilité
  - Il est plus facile et plus performant d'écrire du code Python que de lire du XML
- Éviter les get/set
  - utiliser `property`
- Utiliser les docstrings [3]
  - Génération automatique de la doc.
  - fonction `help(...)`

Zen of Python (PEP 20)

http://www.python.org/

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Readability counts.
- Special cases aren't special enough to break the rules.
- Errors should never pass silently; Unless explicitly silenced.
- In the face of ambiguity, refuse the temptation to guess.
- If the implementation is hard to explain, it's a bad idea.
- If the implementation is easy to explain, it may be a good idea.
- ...

Tim Peters, *The Zen of Python*, PEP20
http://www.python.org/dev/peps/pep-0020/

Annexes

Références

Références

- Python
  - [5] http://python.org/
  - [1] http://en.wikipedia.org/wiki/Python_programming_language
  - http://en.wikipedia.org/wiki/Python_philosophy
  - [4] http://en.wikipedia.org/wiki/Duck_typing
  - [3] http://www.nuxeo.com/publications/slides/meilleures-pratiques-du
  - [2] http://dirtsimple.org/2004/12/python-is-not-java.html
  - http://www.diveintopython.org/