Présentation rapide du langage Python

Principales caractéristiques du langage Python

(cf chap1 Bob Cordeau & Laurent Pointal « **Une introduction à Python 3 »)**

— Historique

—1991 : Guido van Rossum travaille aux Pays-Bas sur le projet AMOEBA, un système d'exploitation distribué. Il conçoit Python à partir du langage ABC et publie la version 0.9.0 sur un forum Usenet
—1996 : sortie de Numerical Python, ancêtre de numpy
— 2001 : naissance de la PSF (Python Software Fundation)
— Les versions se succèdent… Un grand choix de modules est disponible, des colloques annuels sont organisés, Python est enseigné dans plusieurs universités et est utilisé en entreprise…
— 2006: première sortie de IPython
— Fin 2008 : sorties simultanées de Python 2.6 et de Python 3.0
— 2013 : versions en cours des branches 2 et 3 : v2.7.3 et v3.3.0

— Langage Open Source

— Licence Open Source CNRI, compatible GPL, mais sans la restriction copyleft.
Donc Python est libre et gratuit même pour les usages commerciaux
— GvR (Guido van Rossum) est le « BDFL » (dictateur bénévole à vie !)
— Importante communauté de développeurs
— Nombreux outils standard disponibles : Ba • eries included

— Travail interactif

— Nombreux interpréteurs interactifs disponibles (notamment IPython)
— Importantes documentations en ligne
— Développement rapide et incrémentiel
— Tests et débogage outillés
— Analyse interactive de données

— Langage interprété rapide

— Interprétation du bytecode compilé
— De nombreux modules sont disponibles à partir de bibliothèques optimisées (souvent écrites en C ou C++)

— Simplicité du langage

— Syntaxe claire et cohérente
— Indentation significative
— Gestion automatique de la mémoire (garbage collector)
— Typage dynamique fort : pas de déclaration

— Orientation objet

— Modèle objet puissant mais pas obligatoire
— Structuration multi-fichier aisée des applications : facilite les modifications et les extensions
— Les classes, les fonctions et les méthodes sont des objets dits de première classe.
Ces objets sont traités comme tous les autres (on peut les affecter, les passer en paramètre)

— Ouverture au monde

— Interfaçable avec C/C++/FORTRAN
— Langage de script de plusieurs applications importantes
— Excellente portabilité

— Disponibilité de bibliothèques

— Plusieurs milliers de packages sont disponibles dans tous les domaines

Points forts de Python

— langage de très haut niveau ;
— sa lisibilité ;

Langages

Des langages de différents niveaux

— Chaque processeur possède un langage propre, directement exécutable : le langage machine. Il est formé de 0 et de 1 et n'est pas portable, c'est le seul que l'ordinateur puisse utiliser ;
— le langage d'assemblage est un codage alphanumérique du langage machine. Il est plus lisible que le langage machine, mais n'est toujours pas portable. On le traduit en langage machine par un assembleur ;
— les langages de haut niveau. Souvent normalisés, ils permettent le portage d'une machine à l'autre. Ils sont traduits en langage machine par un compilateur ou un interpréteur.

Bref Historique des langages

— Années 50 (approches expérimentales) : FORTRAN, LISP, COBOL, ALGOL…
— Années 60 (langages universels) : PL/□, Simula, Smalltalk, Basic…
— Années 70 (génie logiciel) : C, PASCAL, ADA, MODULA-…
— Années 80 (programmation objet) : C++, LabView, Eiffel, Perl, VisualBasic…
— Années 90 (langages interprétés objet) : Java, tcl/Tk, Ruby, Python…
— Années 2000 (langages commerciaux propriétaires) : C#, VB.NET…

Des centaines de langages ont été créés, mais l'industrie n'en utilise qu'une minorité.

Production des programmes

Deux techniques de production des programmes

La compilation est la traduction du *source* en langage *objet*. Elle comprend au moins quatre phases (trois phases d'analyse — lexicale, syntaxique et sémantique — et une phase de production de code objet). Pour générer le langage machine il faut encore une phase particulière : l'édition de liens. La compilation est contraignante mais offre au final une grande vitesse d'exécution.

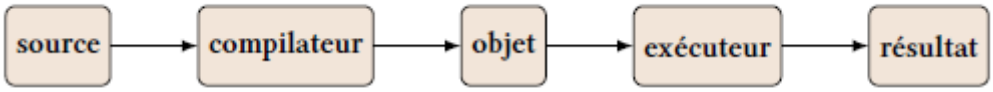

FIGURE 1.1 – Chaîne de compilation

Dans la technique de l'interprétation chaque ligne du *source* analysé est traduite au fur et à mesure en instructions directement exécutées. Aucun programme objet n'est généré. Ce□e technique est très souple mais les codes générés sont peu performants : l'interpréteur doit être utilisé à chaque nouvelle exécution…

FIGURE 1.2 – Chaîne d'interprétation

Technique de production de Python

— Technique mixte : l'interprétation du bytecode compilé. Bon compromis entre la facilité de développement et la rapidité d'exécution ;
— le bytecode (forme intermédiaire) est portable sur tout ordinateur muni de la machine virtuelle Python.

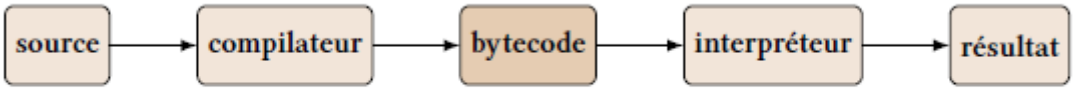

FIGURE 1.3 – Interprétation du bytecode compilé

Pour `exécuter un programme`, Python charge le fichier source .py en mémoire vive, en fait l'analyse (lexicale, syntaxique et sémantique), produit le bytecode et enfin l'exécute.
Afin de ne pas refaire inutilement toute la phase d'analyse et de production, Python sauvegarde le bytecode produit (dans un fichier .pyo ou .pyc) et recharge simplement le fichier bytecode s'il est plus récent que le fichier source dont il est issu.
En pratique, il n'est pas nécessaire de compiler explicitement un module, Python gère ce mécanisme de façon transparente.

Construction des programmes

Le génie logiciel étudie les méthodes de construction des programmes. Plusieurs modèles sont envisageables,entre autres :

— la méthodologie procédurale. On emploie l'analyse descendante (division des problèmes) et remontante (réutilisation d'un maximum de sous-algorithmes). On s'efforce ainsi de décomposer un problème complexe en sous-programmes plus simples. Ce modèle structure d'abord les actions ;
— la méthodologie objet. Centrée sur les données, elle est considérée plus stable dans le temps et meilleure dans sa conception. On conçoit des fabriques (*classes*) qui servent à produire des composants (*objets*) qui contiennent des données (*attributs*) et des actions (*méthodes*). Les classes dérivent (*héritage* et *polymorphisme*) de classes de base dans une construction hiérarchique.
Python offre les *deux* techniques.

1.4 Présentation du langage Python 17

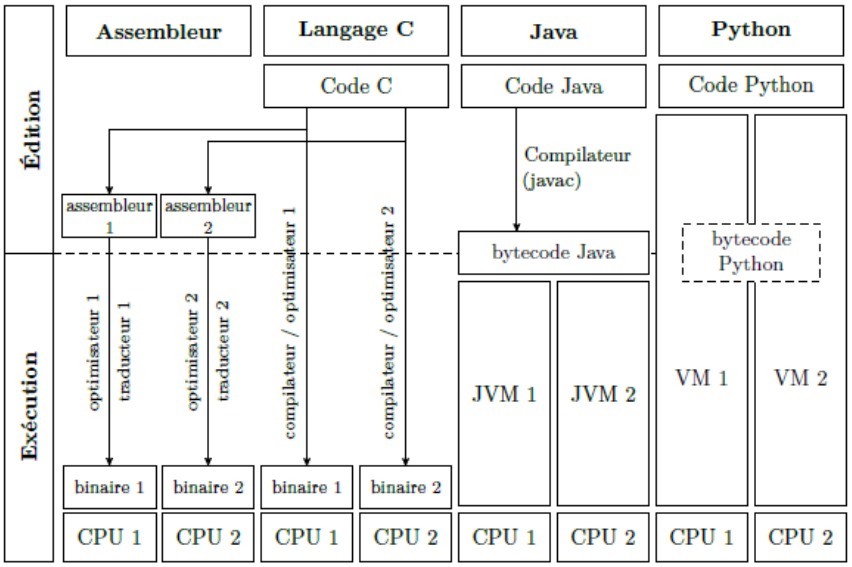

FIGURE 1.1 – Comparaison entre différents langages

Algorithme et programme

Définitions

Définition

Algorithme : ensemble des étapes permeant d'aeindre un but en répétant un nombre fini de fois un nombre fini d'instructions.
Donc un algorithme se termine en un temps fini.

Définition

Programme : un programme est la traduction d'un algorithme en un langage compilable ou interprétable par un ordinateur.
Il est souvent écrit en plusieurs parties dont une qui *pilote* les autres : le programme principal.

Présentation des programmes

Un programme *source* est destiné à l'être humain. Pour en faciliter la lecture, il doit être judicieusement *présenté* et *commenté*.
La signification de parties non triviales (et uniquement celles-ci) doit être expliquée par un commentaire.

En Python, un commentaire commence par le caractère # et s'étend jusqu'à la fin de la ligne :

```
#^^^^^^^^^^^^^^^^^^^^^^
# Voici un commentaire
#^^^^^^^^^^^^^^^^^^^^^^
9+2 # en voici un autre
```

Implémentations de Python

Remarque

Une « implémentation » signifie une « mise en oeuvre ».

— CPython : *Classic Python*, codé en C, implémentation portable sur différents systèmes
— Jython : ciblé pour la JVM (utilise le bytecode de JAVA)
— IronPython : *Python.NET*, écrit en C#, utilise le MSIL (*MicroSo・ Intermediate Language*)
— Sta・less Python : élimine l'utilisation de la pile du langage C (permet de récurser tant que l'on veut)
— Pypy : projet de recherche européen d'un interpréteur Python écrit en Python