Éléments de base du langage Python

BENMANSOUR Asma

Avril 2018
1.0

Table des matières

Objectifs

Le cours programmation python vise à :

- Susciter l'intérêt de l'apprenant pour un sujet aussi ardu que la programmation à travers l'utilisation d'un langage moins contraignant que les autres (ex : langage C).
- Mettre en évidence les invariants de la programmation.
- Identifier les points forts et les points faible du langage python
- Situer le langage python parmi les autres langage de programmations avant la résolution d'un problème
- Mémoriser la syntaxe du langage python.
- Mettre en pratique cette syntaxe lors de l'écriture d'un programme représentant la résolution d'un problème spécifique.
- Traduire un algorithme pré-existant en programme une utilisant la syntaxe du langage python.

Objectifs spécifiques du chapitre

A l'issu de ce chapitre l'apprenant sera capable de :

- Mémoriser les caractéristiques principale du langage
- Différencier entre un algorithme et un programme lors de la conception de la solution à un problème
- Comprendre la technique de production et la technique de construction d'un programme python
- Distinguer entre les différentes versions du langage crées à ce jour.
- Mettre en pratique les divers modes d'exécutions disponibles du langage.

Pourquoi Python ?

II

Définition

Python est un langage portable, dynamique, extensible, gratuit, qui permet sans l'imposer une approche modulaire et orientée objet de la programmation. Python est développé depuis 1989 par Guido van Rossum et de nombreux contributeurs bénévoles[1] *p.23* . Python est un langage de programmation généraliste, facile à apprendre et rapide à mettre en œuvre [2].

Python est généraliste car, selon les multiples réalisations qu'il a à son actif, il peut être utilisé dans tous les domaines : écriture d'applications pour le Web (serveur d'application Zope, *framework* Django), programmes de calculs mathématiques (biblithèque SciPy), interfaces graphiques (il existe des supports de Python pour les systèmes d'interface graphique GTK, *p.22* Qt, TK, wxWidget), programmation de scripts systèmes, etc [2].

De ce fait, Python dispose d'une très large bibliothèque standard qui offre au programmeur des outils très divers pour : la gestion réseau (librairie socket), la manipulation du format xml, l'accès aux protocoles d'Internet (protocoles des services courriel, divers protocoles web), l'accès aux éléments du système d'exploitation sous-jacent (accès aux fichiers et répertoires, gestion des processus), l'écriture d'interfaces graphiques (librairie Tkinter), l'accès aux bases de données relationnelles,etc [2].

Il est aussi possible d'étendre Python en intégrant de nouveaux modules. Par exemple, *la librairie PIL* * permet de traiter des images [2].

Python est facile à apprendre car de nombreuses opérations dévolues au programmeur dans les langages classiques comme le langage C, par exemple la gestion de la mémoire, sont prises en charge par l'interpréteur Python. De même, Python gère dynamiquement les variables et libère le programmeur des déclarations de type. De plus Python impose d'écrire les blocs d'instructions de manière indentée, ce qui favorise grandement la lecture des programmes [2].

Enfin, en tant que langage interprété, Python est rapide à mettre en œuvre. Il suffit de lancer la console Python pour avoir sous la main de quoi tester directement des commandes et des structures de données. Cela est un gain de temps pour le programmeur par rapport au cycle compilation/édition de liens du langage C [2].

Python est souvent considéré comme un langage de scripts, c'est à dire destiné à l'écriture de petits programmes utilitaires servant dans des contextes particuliers. En tant que langage dynamique interprété, Python doit réaliser un nombre d'opérations bien plus grand que les lignes écrite par le programmeur dans son programme. En effet, il faut que Python garde en mémoire et gère des

descripteurs pour chacun des symboles utilisés dynamiquement et qu'il gère les réservations et libérations de mémoire correspondantes [2].

L'exécution d'un programme Python sera donc toujours plus lente que celle d'un programme compilé. Ainsi un gros logiciel diffusé à grande échelle (prenons l'exemple d'un logiciel de traitement de texte ou d'un gestionnaire de bases de données) sera plus probablement écrit en langage C ou en C++ [2].

Cf. "Python présentation Youtube"

Attention

Même s'il est vrai que Python peut être utilisé dans de nombreux domaines, cela ne signifie par que Python est le bon choix dans tous les cas !

Mais pour de nombreux usages, quand la rapidité d'exécution n'est pas le critère déterminant, Python est un bon choix.

Conseil

Pour l'apprentissage de la programmation, c'est un bon choix également car cela permet une entrée en matière plus rapide et simplifiée. Il faut cependant avoir conscience que les facilités offertes par Python ou autres langages dynamiques cachent de nombreux détails au programmeur et pourraient donner une impression erronée au débutant en programmation. En effet la gestion mémoire d'un tableau de nombres entiers, par exemple, n'est pas une chose « allant de soi » et le programmeur en assembleur ou en C (ou dans le cas de Python : l'interpréteur) doit contrôler ce qui se passe en mémoire. Cela est en quelque sorte plus pédagogique [2]. *p.23*

Principales caractéristiques du langage Python

III

1. Historique

- 1991 : Guido van Rossum publie Python au CWI (Pays-Bas) à partir du langage ABC et du projet AMOEBA (système d'exploitation distribué) [3].
- 1996 : sortie de Numerical Python [3].
- 2001 : naissance de de la PSF [3]. *p.22*
- Les versions se succèdent... Un grand choix de modules disponibles, des colloques annuels sont organisés, Python est enseigné dans plusieurs universités et est utilisé en entreprise...[3].
- Fin 2008 : sorties simultanées de Python 2.6 et de Python 3.0 [3].
- 2010 : versions en cours 1 : v2.7 et v3.1.2 [3].

2. Langage open source

- Licence *Open Source* CNRI *p.22* , compatible GPL *p.22* , mais sans la restriction *copyleft* . Python est libre et gratuit même pour les usages commerciaux [3].
- Importante communauté de développeurs [3].
- Nombreux outils standard disponibles : *Batteries included* [3].

3. Travail interactif

- Nombreux interpréteurs interactifs disponibles [3].
- Importantes documentations en ligne [3].
- Développement rapide et incrémentiel [3].
- Tests et débogage faciles [3].
- Analyse interactive de données [3].

4. Langage interprété rapide

- Interprétation du *bytecode* compilé [3].
- De nombreux modules sont disponibles :
  à partir de bibliothèques optimisées écrites en C, C++ ou FORTRAN [3].

5. Simplicité du langage

- Syntaxe claire et cohérente [3].
- Indentation significative [3].
- Gestion automatique de la mémoire *(garbage collector)* [3].
- Typage dynamique fort : pas de déclaration [3].

6. Orientation objet

- Modèle orienté objet p.21 puissant mais pas obligatoire [3].
- Structuration multi-fichier très facile des applications :
  facilite les modifications et les extensions [3].
- Les classes, les fonctions et les méthodes sont des objets dits de première classe [3].
- Ces objets sont traités comme tous les autres :
  on peut les affecter, les passer en paramètre [3].

7. Ouverture au monde

- Interfaçable avec C/C++/FORTRAN [3].
- Langage de script de plusieurs applications importantes [3].
- Excellente portabilité [3].

8. Disponibilité de bibliothèques

- Plusieurs milliers de *packages* sont disponibles dans tous les domaines [3].

Exercice : Python pour les débutants en programmation ?

IV

[solution n°1 p.20]

Pourquoi python est il un bon choix pour les débutants en programmation :

- ☐ Il est orienté objet
- ☐ Pas de déclaration de type
- ☐ Syntaxe clair
- ☐ Exécution rapide
- ☐ Importante documentation en ligne

Algorithme et programme

V

1. Notion d'algorithme

Définition : Qu'est ce qu'un algorithme ?

Ensemble d'étapes écrites dans un ordre précis permettant d'atteindre un but en répétant un nombre fini de fois un nombre fini d'instructions. Un algorithme se termine en un temps fini [3].

2. La présentation des programmes

Définition : Qu'est ce qu'un 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 [3].

Un programme source est destiné à l'être humain. Pour en faciliter la lecture, il doit être judicieusement 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 [3].

La production des programmes python

VI

1. Les techniques existantes de production des programmes

Il existe 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 [3].

Chaine 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. Cette technique est très souple mais les codes générés sont peu performants : l'interpréteur doit être utilisé à chaque exécution...[3].

Technique de l'interprétation

2. La technique de production de Python

Technique mixte : l'interprétation du *bytecode* compilé. Bon compromis entre la facilité de développement et la rapidité d'exécution [3]. le *bytecode* (forme intermédiaire) est portable sur tout ordinateur muni de la machine virtuelle Python [3].

Interprétation du bytecode compilé

Pour exécuter un programme, Python charge le fichier source *.py* (ou *.pyw*) en mémoire vive, en fait l'analyse syntaxique, produit le *bytecode* et enfin l'exécute. Pour chaque module importé par le programme, Python vérifie d'abord s'il existe une version pré-compilée du *bytecode* (dans un fichier *.pyo* ou *.pyc*) dont la date correspond au fichier *.py*. S'il y en a un, Python l'utilise, sinon il fait une analyse syntaxique du module *.py*, et utilise le *bytecode* qu'il vient de générer [3].

Fondamental : Pas besoin de compiler en Python !

En pratique, il n'est pas nécessaire de compiler explicitement un module, Python gère ce mécanisme de façon transparente.

La construction des programmes Python

VII

Le génie logiciel étudie les méthodes de construction des programmes. Plusieurs modèles sont envisageables, python offre les deux techniques procédurale et objet décrites ci-dessous:

1. 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 [3]. p.23
2. la méthodologie *objet* :
   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 [3].

Les implémentations de Python

VIII

- CPython p.22 : codé en C, portable sur différents systèmes [3]
- Python3000 : Python 3, la nouvelle implémentation de CPython [3].
- Jython : ciblé pour la *Java Virtual Machine* p.21 (utilise le *bytecode* de JAVA) [3].
- IronPython :
  Python.NET, écrit en C#, utilise le MicroSoft Intermediate Language [3].
- Stackless Python :
  élimine l'utilisation de la pile du langage C (permet de récurser tant que l'on veut) [3].
- Pypy : projet de recherche européen d'un interpréteur Python écrit en Python [3].

Les modes d'exécution de Python

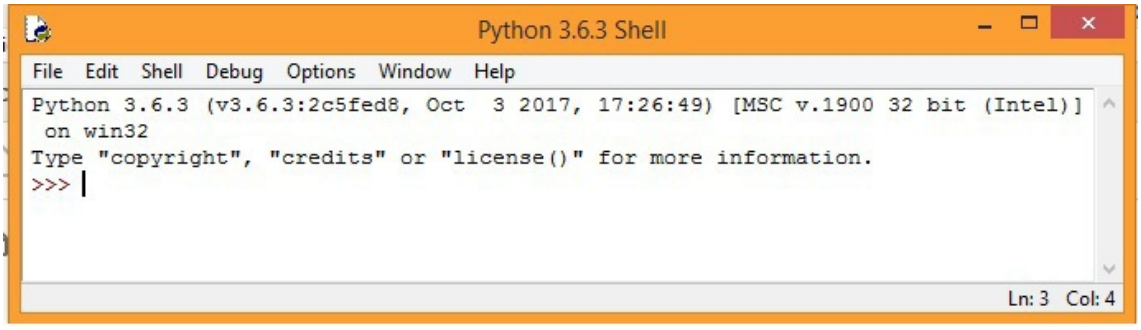
IX

Python présente la particularité de pouvoir être utilise de deux manières différentes c'est à dire qu'il existe deux modes d'exécution d'un code Python:

1) Vous allez d'abord l'utiliser en *mode interactif* via un *interpréteur* (par exemple IDLE p.21 p.22 p.21) permettant d'obtenir un résultat immédiat, c'est-à-dire d'une manière telle que vous pourrez dialoguer avec python directement depuis le clavier. Cela vous permettra de découvrir très vite un grand nombre de fonctionnalités du langage [1].

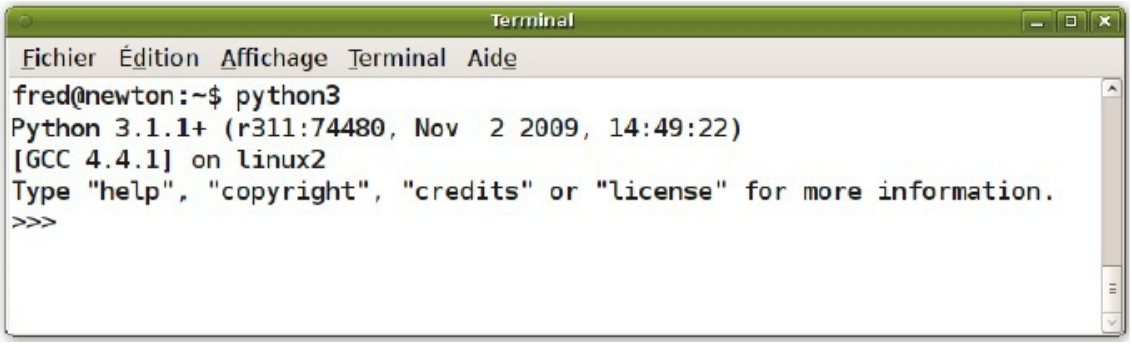
L'interpréteur peut être lancé directement depuis la ligne de commande (dans un "*shell*" Linux, ou bien dans une fenêtre DOS sous Windows) : il suffit d'y taper la commande python3 (en supposant que le logiciel lui-même ait été correctement installé, et qu'il s'agisse d'une des dernières versions de Python), ou d'y taper python (si la version de Python installée sur votre ordinateur est antérieure a la version 3.0) [1].

Si vous utilisez une interface graphique telle que Windows, Gnome, WindowMaker ou KDE, vous préférerez vraisemblablement travailler dans une "fenêtre de terminal", ou encore dans un environnement de travail spécialisé tel que IDLE p.22 . Voici par exemple ce qui apparaît dans une fenêtre de terminal Gnome sous Ubuntu Linux [1]:



Fenêtre de terminal Gnome sous Ubuntu Linux

Avec IDLE sous Windows, votre environnement de travail ressemblera à celui-ci :



Fenêtre IDLE sous Windows

Conseil : Testez les commandes sur IDLE et notez les résultats !

Les *trois caractères "supérieur a"* constituent le *signal d'invite*, ou *prompt principal*, lequel vous indique que Python est prêt a exécuter une commande. Par exemple, vous pouvez tout de suite utiliser l'interpréteur comme une simple calculatrice de bureau. Veuillez donc vous-même tester les commandes dans les exemples cités tout au long des prochains chapitres de ce cours. Prenez l'habitude d'utiliser votre cahier d'exercices pour noter les résultats qui apparaissent a l'écran !

2) Dans un second temps, vous apprendrez comment créer vos premiers programmes (*scripts*) et les sauvegarder sur disque. Plus précisément enregistrer un ensemble de commandes Python dans un fichier grâce à un éditeur on parle alors d'un script Python que l'on exécute par une commande ou par une touche du menu de l'éditeur [3].

Exercice : Modes d'exécutions de python

X

[solution n°2 p.20]

Quel mode d'exécution est le plus rapide si vous voulez calculer le résultat de 4^5 +100 x 500^3

- écrire un script et l'enregistrer puis l'exécuter
- la console Python (IDLE)

Solutions des exercices

> **Solution** n°1

Exercice p. 11

Pourquoi python est il un bon choix pour les débutants en programmation :

- ☐ Il est orienté objet
- ☑ Pas de déclaration de type
- ☑ Syntaxe clair
- ☐ Exécution rapide
- ☑ Importante documentation en ligne

> **Solution** n°2

Exercice p. 19

Quel mode d'exécution est le plus rapide si vous voulez calculer le résultat de 4^5 +100 x 500^3

- ○ écrire un script et l'enregistrer puis l'exécuter
- ◉ la console Python (IDLE)

Glossaire

orienté objet

La programmation orientée objet (POO), ou programmation par objet, est un paradigme de programmation informatique élaboré par les Norvégiens Ole-Johan Dahl et Kristen Nygaard au début des années 1960 et poursuivi par les travaux de l'Américain Alan Kay dans les années 1970. Il consiste en la définition et l'interaction de briques logicielles appelées objets ; un objet représente un concept, une idée ou toute entité du monde physique, comme une voiture, une personne ou encore une page d'un livre. Il possède une structure interne et un comportement, et il sait interagir avec ses pairs. Il s'agit donc de représenter ces objets et leurs relations ; l'interaction entre les objets via leurs relations permet de concevoir et réaliser les fonctionnalités attendues, de mieux résoudre le ou les problèmes. Dès lors, l'étape de modélisation revêt une importance majeure et nécessaire pour la POO. C'est elle qui permet de transcrire les éléments du réel sous forme virtuelle.

Abréviations

CNRI : Corporation for National Research Initiatives

CPython : Classic Python

GPL : General Public License

GTK : Gimp ToolKit)

IDLE : Integrated DeveLopment Environment

PIL : Python Standard Library

PSF : Python Software Fundation

Bibliographie

X

[1] Swinnen Gérard, 02/02/2012, "Apprendre à programmer avec python 3", (3e édition), Eyrolles, 435 p, Noire

[2] Jachym, Marc, "Cours de Programmation avec le langage Python: Niveau débutant en programmation",Licence professionnelle – Métrologie dimensionnelle et qualité IUT de St Denis, Université Paris 13.

[3] Cordeau, Bob, Introduction à Python 3, version 2.71828.