Préface

"Python has the philosophy of making sensible compromises that make the easy things very easy, and don't preclude too many hard things."

« La philosophie de Python consiste à faire les compromis raisonnables qui rendent les choses faciles, très faciles, et n'empêchent pas de faire trop de choses difficiles. »

Peter Norvig, Directeur de la Recherche, Google Inc.

Python n'est pas un langage de niche.

Conçu au départ comme langage de script système, Python est à présent, en fait depuis au moins une dizaine d'années, un langage universel qui sait couvrir des besoins aussi variés que le scripting système ou de codes complexes de calcul numérique, le développement rapide d'applications desktop ou d'applications web, l'intégration applicative, les tests, etc.

De ses racines dans l'enseignement de la programmation, via sa filiation avec le langage ABC, Python a conservé résolument comme principe de se focaliser sur la simplicité et la cohérence conceptuelle ainsi que la lisibilité du code. On parle avec Python de « pseudo-code exécutable ». En effet, ce langage permet de se concentrer sur l'essentiel – les structures de données et les algorithmes – avec une syntaxe claire et expressive, comme c'est le cas des extraits de pseudo-code que l'on peut lire dans les ouvrages d'algorithmique.

Cette lisibilité de Python, combinée à son expressivité et ses structures de données de haut niveau, font que les développeurs Python sont le plus souvent capables de produire un code 3 à 5 fois plus dense, et en même temps au moins aussi clair et lisible,

que du code Java ou C++. La productivité initiale, mais aussi la maintenabilité des logiciels ainsi produits, est ainsi augmentée, selon les métriques éprouvées, dans des proportions encore plus importantes.

En fait, la proximité syntaxique et sémantique de Python avec C++, Java ou C# fait que les développeurs chevronnés de ces langages se sentent immédiatement à l'aise quand il s'agit de passer à Python. Réciproquement, il n'est pas difficile à un programmeur Python de lire la littérature consacrée à ces langages, il est vrai actuellement plus populaires, et d'appliquer en Python ce qu'il y aura appris.

Langage pragmatique, Python sait s'adapter aux différents styles de programmation, y compris en changeant son propre modèle objet via des constructions avancées comme les métaclasses. Il sait aussi s'interfacer, avec souplesse, grâce à son typage dynamique, avec les autres langages et les bibliothèques tierces.

Langage agile, il permet de construire les logiciels par itérations successives, depuis le script de quelques dizaines de lignes où la plupart des paramètres sont codés en dur, jusqu'au framework extensible complet, et s'est vu adopter avec enthousiasme par la communauté du test logiciel et du développement dirigé par les tests.

Ces qualités sont le résultat de 15 années d'évolution contrôlée, selon un mode de développement ouvert et communautaire. Chaque évolution du langage a fait l'objet de discussions approfondies et a été évaluée en fonction de son apport, bien sûr, mais aussi de l'impact qu'elle peut avoir sur les qualités du langage.

Le rythme choisi est un compromis, entre des évolutions rapides qui suivent sans ligne directrice les dernières modes en matière de langages de programmation, et une sclérose du langage. L'utilisation de Python par des milliers de développeurs open source, mais aussi, à un niveau critique, dans des entreprises comme Google, Yahoo!, Apple ou Microsoft, est un gage de stabilité et de pérennité du langage.

L'importance de la communauté Python est évidemment à souligner : l'équipe de développement est à remercier et à féliciter pour la façon dont elle a su faire évoluer les spécifications du langage, sa bibliothèque standard, et la multitude de bibliothèques tierces. Elle maintient aussi l'implémentation de référence du langage, y compris ses multiples portages sur différentes plates-formes (C multi-plates-formes, JVM, CLI, .NET). Parmi celles-ci, on peut souligner le projet PyPy, financé par l'Union Européenne : ce projet, qui vise à implémenter un compilateur Python en Python (!), a le potentiel de propulser Python à un nouveau supérieur de performance et de souplesse, en appliquant les principes du développement agile aux implémentations-mêmes du langage, avec comme objectif de simplifier et donc de dynamiser le travail sur l'interpréteur standard, mais également de viser des cibles multiples pour l'environnement d'exécution : code objet natif ou bytecodes de différentes machines virtuelles.

Il faut enfin parler du soutien indéfectible de plusieurs piliers de la communauté Python (connus pour certains par des surnoms étranges, « Timbot », « Effbot », « Martellibot ») aux débutants et experts moins avancés qu'eux sur la voie de la Révélation.

Le livre que vous tenez entre les mains est un des tout premiers ouvrages édités directement en français sur le langage Python. Tarek Ziadé y présente bien sûr les bases du langage et de sa bibliothèque standard, avec des exemples tirés des principaux domaines d'application.

Cependant, l'originalité de Tarek est de nous présenter ce qui distingue le programmeur Python averti du débutant : les conventions de codage, les meilleures pratiques de design objet, le développement dirigé par les tests. En un mot, il nous aide à distinguer ce qui est pythonique de ce qui ne l'est pas.

Avec Python, la programmation redevient plaisante et les projets sont livrés à temps et pour moins cher. De quoi faire plaisir aux développeurs, à leurs managers et à leurs clients.

Stefane Fermigier
Utilisateur de Python depuis 1995
PDG et fondateur, Nuxeo SAS