Choisir Python

Par Stephan Richter

J'ai commencé la programmation avec un Commodore 64 (C64), un petit système basé sur le langage de programmation Basic, qui est à la fois simple et puissant. J'ai eu par la suite un PC doté de Borland Pascal. Le système d'aide en ligne de Pascal est très impressionnant : chaque commande et bibliothèque est parfaitement documentée et accompagnée bien souvent d'exemples de code. Ce système permet une maîtrise rapide du langage. De plus, le Pascal permet d'intégrer des séquences d'assembleur, pour programmer par exemple directement la souris et le joystick. Le seul défaut du Pascal est la compilation obligatoire, qui est un peu ennuyeuse pour quelqu'un venant du Basic.

Par la suite, Jason Orendorff, pionnier de la communauté Python et lauréat 2001 du Concours international d'obfuscation de code C (IOCCC) est devenu mon mentor et m'a appris toutes les techniques avancées de programmation, comme la programmation orientée objet par le biais de Java, langage particulièrement ordonné et propre. Mais cette propreté a un prix : l'effort supplémentaire pour écrire un programme Java dans les règles de l'art est trop important. Il faut toujours écrire des classes, et une seule par fichier, puis compiler, etc. Jim Fulton parle de programmation « javiotique » pour décrire ce surcroît d'effort.

Jason m'a alors converti à Python. Après une période d'adaptation, on tombe très vite amoureux de ce langage. Aucune compilation n'est nécessaire et Python est utilisable sur tant de plates-formes qu'il est plus portable que Java. De plus, Python permet de programmer objet mais ne l'impose pas : il reste possible de faire des petits scripts non structurés. Youpi ! Enfin, l'indentation obligatoire du code ne pouvait que satisfaire mes gênes prussiens.

Que peut-on espérer de mieux ? Des fonctionnalités ! Pour un développeur issu du monde Pascal, le passage à des langages comme Java ou C++ est frustrant à cause de

la pauvreté des bibliothèques standards. La philosophie *batteries included* de Python offre tout ce dont un développeur peut rêver.

Un autre avantage de Python est la richesse des bibliothèques tierces. Comme Python est utilisé dans la quasi-totalité des domaines et à tous les niveaux applicatifs, il existe des extensions pour toutes les fonctionnalités que l'on peut imaginer. Vous souhaitez faire du calcul scientifique ? Utilisez l'extension *numeric*. Vous avez du code MatLab à intégrer ? Installez l'extension *matlab* pour pouvoir piloter ce moteur depuis Python. Le langage est aussi utilisé pour les frameworks web comme Zope et Plone, les moteurs de jeu comme *Pygame*, les plug-ins pour Gimp et toute une myriade d'applicatifs. Cette variété prouve la puissance de Python, qui s'adapte aussi bien aux situations où seul un langage de script est nécessaire, que pour des besoins plus complets, faisant appel à la programmation orientée objet.

J'ai découvert par la suite la communauté Python et plus généralement la mouvance open source. Ma première contribution était un correctif dans un exemple pour une bibliothèque d'envois d'e-mails. Guido von Rossum m'a personnellement répondu pour me signaler que mon correctif serait intégré dans la prochaine *release*. L'Open Source, quel bonheur !

Une communauté autour d'une technologie fait toute la différence : le niveau d'assistance est incroyable et les questions obtiennent des réponses en général en quelques heures. Quel logiciel propriétaire offre ce genre de service gratuitement ? Ce système permet d'avancer sans jamais être bloqué, et les développeurs qui acquièrent leur expérience par ce biais renvoient souvent l'ascenseur à la communauté en répondant à leur tour aux questions des autres.

J'ai découvert par la suite Zope, le serveur d'applications écrit en Python. La découverte de Zope provoque le même effet que celle de Python : « wow ! ». Zope offre toutes les fonctionnalités rêvées pour une application web, comme la sécurité et la persistance, ainsi que de nombreuses extensions. Quel plaisir, comparé à des frameworks comme IBM WebSphere et BEA Weblogic.

Durant les quatre dernières années, j'ai fait partie des *core developers* de Zope 3, qui est une réécriture complète de Zope, basée sur l'expérience passée des versions 1 et 2. Ce projet est passé du rang de prototype éducatif à ce qu'il est aujourd'hui : une application utilisée en production par des entreprises pour des projets web critiques. Zope 3, malgré son jeune âge, est considéré comme la plus stable et la plus sure des plates-formes web Open Source disponibles à l'heure actuelle, grâce aux milliers de tests unitaires et fonctionnels qui ont été codés en parallèle de sa conception. Les performances sont également au rendez-vous : Zope 3 peut être configuré pour ne fournir que les services utilisés dans un applicatif donné, et reste très performant comparé aux frameworks capables de fournir la même quantité de fonctionnalités.

Mais que pouvez vous faire avec Zope 3 ? Le premier projet à avoir officiellement utilisé Zope 3 est Schooltool, un outil gratuit de gestion d'école dans lequel je suis également investi. Schooltool fournit de nombreuses fonctionnalités, de la génération de rapports PDF aux calendriers en ligne. Beaucoup d'écoles ont d'ores et déjà adopté Schooltool ainsi que son petit frère SchoolBell, et démontrent le succès de cet outil. Pour l'année à venir, SchoolTool a déjà signé avec de nombreux partenaires du monde de l'éducation, avec pour objectif de remplacer petit à petit les solutions propriétaires, ce qui constitue un premier signe de l'entrée de la solution sur ce marché. Le projet est financé par la *Shuttleworth Foundation*, et Mark Shuttleworth ne risquerait pas un centime sur une technologie qui ne marcherait pas ou ne pourrait pas grandir.

Cela fait maintenant six ans que je gagne ma vie en développant du code Python Open Source et c'est un véritable bonheur ! Je ne voudrais jamais, quelque fût le prix, travailler pour une entreprise qui ne me laisserait pas écrire du code Open Source Python. Dans mon autre vie, je suis un doctorant en physique, et même si les publications de recherche sont ouvertes à tous, le secret qui entoure le travail de recherche m'oppresse souvent, en comparaison à mes travaux dans le monde de l'Open Source.

Merci pour votre lecture et régalez-vous avec ce livre !

Sincèrement,

Stephan

À propos de Stephan Richter

Stephan Richter est étudiant en doctorat de physique à l'université de Tufts (Sommerville, Massachusetts, USA). Il fait partie de la communauté depuis 1999 et a participé à beaucoup de projets communautaires, comme la documentation et l'organisation de la première conférence EuroZope. Stephan a aussi travaillé en tant que consultant pour de nombreuses entreprises travaillant avec Zope, développé beaucoup d'extensions et publié deux livres communautaires sur Zope, et un livre sur Zope 3 (*Zope 3 Developer's Handbook* aux éditions Sams). Depuis son premier sprint Zope 3 en 2002, Stephan participe activement au développement de ce framework et gère de nombreux sous-projets, comme l'internationalisation et la documentation.