A

L'histoire de Python

Le langage Python a été créé à la fin des années 1980 à l'Institut national de recherches mathématiques et informatiques de Hollande (le CWI) par Guido van Rossum. Par commodité, nous utiliserons le raccourci GvR pour nommer ce dernier dans la suite de cette annexe.

Le langage ABC

GvR a rejoint le CWI en 1983 dans l'équipe en charge du développement du langage ABC, sur lequel il a travaillé pendant 3 ans. Cette période a fortement influencé GvR sur la conception de Python, qui hérite de certains des concepts d'ABC.

Le langage ABC est un langage de programmation interactif fortement typé, qui a été pensé pour remplacer le Basic, largement répandu à l'époque, en fournissant un environnement particulier ainsi que d'autres caractéristiques notables, comme le typage spécifique des données et la syntaxe par indentation.

Environnement de développement

La particularité de l'environnement d'ABC est qu'il n'est pas nécessaire de sauvegarder fonctions et procédures dans des fichiers sources : une fois entrées dans l'environnement interactif, leur saisie dans l'invite de commande (le prompt, symbolisé sous ABC par >>>) les conservent automatiquement d'une exécution à l'autre.

Un système de complétion de code permet en outre de faciliter la saisie des commandes. Enfin, un historique autorise de revenir en arrière sans limite.

Types de données

ABC fournit 5 types, qui permettent d'exprimer toutes formes de structures de données :

- le type `nombre`, pour les entiers et les réels, sans aucune limite de taille, hormis la mémoire physique disponible de la machine ;
- le type `text`, pour les chaînes de caractères ;
- le type `list`, pour manipuler des collections d'éléments ordonnés ;
- le type `compound`, équivalent au type `list` mais non modifiable. C'est une sorte de `recordset` sans étiquette ;
- le type `table`, qui définit un certain nombre de clés uniques, et associe une valeur à chacune d'entre elles. Ce type est comparable à une combinaison de deux instances de type `list` : les clés et les valeurs.

Exemple de manipulation de table sous ABC

```
>>> PUT {} IN distance_paris
>>> PUT 300 IN distance_paris["Dijon"]
>>> PUT 220 IN distance_paris["Lille"]
>>> PUT 770 IN distance_paris["Marseille"]
>>> WRITE distance_paris["Dijon"]
300
>>> WRITE distance_paris
{["Dijon"]: 300; ["Lille"]: 220; ["Marseille"]: 770}
```

Il n'est pas nécessaire ici de signaler que la variable `distance_paris` est de type `table`, ABC le fait automatiquement lors de la première affectation.

Indentation du code

L'imbrication de code ABC n'est pas faite comme en C ou en Pascal par des accolades ou des délimiteurs `begin..end`. C'est l'indentation des lignes qui détermine le niveau d'imbrication du code.

Exemple de définition de la fonction message

```
HOW TO DISPLAY message:
    FOR line IN message:
        WRITE line /
```

```
>>> DISPLAY "ABC est l'ancêtre de Python"
ABC est l'ancêtre de Python
```

EN SAVOIR PLUS Le langage ABC

Pour plus d'informations sur le langage ABC, Le lecteur interessé peut se référer à l'ouvrage *The ABC Programmer's Handbook* (Geurts, Meertens, Pemberton, aux Éditions Prentice-Hall).

Le projet ABC n'a malheureusement pas eu le succès escompté en dehors du cercle du CWI et est resté relativement confidentiel.

Le projet Amoeba

GvR a rejoint en 1986 le projet Amoeba, un système d'exploitation distribué. Il a été chargé dans ce cadre de créer un langage de script pour manipuler le système plus facilement. Les contraintes du projet étaient relativement souples pour laisser GvR, fort de son expérience passée, mettre au point une première version de ce qui allait devenir le langage Python.

GvR implémenta ce langage de script en essayant de supprimer toutes les contraintes et frustrations qu'il avait vécues avec ABC.

Par exemple, ABC ne permettait pas de lire et écrire dans un fichier, et cette fonctionnalité ne pouvait pas être ajoutée facilement au langage, dénué de tout concept de bibliothèque ou de tout système de programmation d'entrée/sortie souple.

L'extensibilité fut le premier chantier de GvR car il voulait que Python, même si son objectif premier était de fonctionner pour Amoeba, puisse être étendu facilement par des programmeurs tiers en se basant sur un socle commun de primitives et des points d'entrée simples.

L'idée de rendre le langage portable, c'est-à-dire fonctionnel sur plusieurs plates-formes comme Amoeba bien sûr, mais aussi sur MS-Windows, Unix ou Macintosh, était aussi un objectif de GvR.

À un moment de l'histoire de l'informatique où les ordinateurs commençaient à envahir les entreprises et les foyers, le manque d'extensibilité et de portabilité condamnait ABC à un rôle mineur, et GvR, en visionnaire, a su ouvrir les portes de son langage de script.

GvR conçut les premières versions du langage qu'il appela Python, à la gloire des Monty Python dont il était fan. Lorsque la liste de diffusion fut créée plus tard, il n'était pas rare de voir régulièrement des messages de fans des Monty Python, ne pensant pas avoir affaire à un langage de programmation.

Dans les premières versions du langage, le système d'extension qui permettait d'ajouter de nouveaux types d'objets à Python à partir d'un fichier de code Python ou un fichier compilé en C, C++ ou encore en Fortran, a tout de suite été adopté et plébiscité par l'entourage de GvR.

Les versions de Python s'enchaînèrent jusqu'à la version 1.2 en 1995, date à laquelle GvR quitta le CIW pour rejoindre le CNRI (Corporation of National Research Initiatives) à Reston en Virginie (USA) pour continuer ses travaux.

Le CNRI

Cet organisme finança le développement de Python pendant cinq ans, par le biais de fonds de recherche. La *Python Software Activity* (PSA), le Python Consortium et des sociétés privées apportèrent également des fonds pour soutenir l'avancée du langage. Le travail au CNRI a permis de sortir plusieurs versions de Python, de la 1.3 à la 1.6.

En 2000, GvR prit la décision de quitter le CNRI, car les fonds alloués à Python étaient de plus en plus maigres et les développeurs dispatchés sur d'autres projets. De plus, l'organisme ne semblait pas très favorable au logiciel libre.

Ce départ fut relativement tendu et le CNRI insista pour modifier le texte de la licence de Python pour garder une mainmise, en provoquant à l'époque une grande inquiétude de la communauté sur la suite des événements.

Accompagné de 3 autres développeurs du CNRI, GvR fonda le PythonLabs, et rejoignit la startup Californienne BeOpen.com.

PythonLabs et BeOpen.com

Avec l'arrivée à BeOpen.com, l'équipe du PythonLabs passa directement de la version 1.6 à la 2.0, en intégrant des améliorations majeures, comme les *list comprehensions*, le support étendu du XML, un nouveau système de ramasse-miettes cyclique, et une nouvelle licence plus orientée Open Source.

Le projet Python 3000 était lancé en parallèle, pour accueillir la nouvelle version de Python, vouée à contenir des modifications incompatibles avec les versions 2.x, pour corriger des erreurs de conception du langage.

Un système d'avertissement a alors été introduit pour permettre de spécifier les compatibilités ascendantes et descendantes du langage.

En d'autres termes, toute introduction de nouvelle fonctionnalité incompatible avec la version en cours, peut être aperçue et utilisée par le biais du module `__future__`, et toute fonctionnalité qui n'existera plus dans la version suivante affiche un warning lorsqu'elle est utilisée.

Ce système est d'ores et déjà utilisé dans la série des versions 2.x.

Python Software Foundation et Digital Creations

Moins d'un an après l'arrivée à BeOpen.com, l'équipe de PythonLabs déménage une nouvelle fois pour rejoindre Digital Creation, la société qui allait devenir par la suite Zope Corp.

En Mars 2001, la Python Software Foundation voit le jour et remplace la PSA, annoncée par GvR à la neuvième conférence Python, et sponsorisée par les sociétés Digital Creation et ActiveState, contributeurs majeurs de la communauté Python de l'époque.

Le premier comité directeur réunissait des membres de PythonLabs et des responsables des deux sociétés, à savoir : Dick Hardt, David Ascher, Paul Everitt, Fredrik Lundh, Tim Peters, Greg Stein, Guido van Rossum et Thomas Wouters.

Les versions de Python se sont ensuite enchaînées jusqu'à la toute dernière en 2009 au moment de l'écriture de ce livre (3.0).

Python et Zope

Python a joué un rôle fondamental pour le développement du framework Zope, et inversement, Zope a beaucoup contribué au développement du langage.

Le texte ci-dessous est une interview de Paul Everitt, créateur de Digital Creations, l'entreprise qui conçoit Zope, et qui répond à la question suivante :

Quelle a été la place de Python dans l'histoire de Zope et Digital Creations ?

En 1995, la société Digital Creations a été créée pour mettre en ligne des journaux. Nous avons utilisé Python pour concevoir l'architecture de notre plate-forme de journaux en ligne et avons beaucoup participé à la communauté Python.

Jim Fulton (ndlr : directeur technique actuel) a rejoint l'entreprise l'année suivante et a lancé l'idée de publier des objets Python via le Web. Le framework « Bobo » était né et distribué sous licence Open Source.

Une application commerciale nommée Principia et entièrement écrite en Python faisait également partie de nos travaux.

En 1997, nous avons été sortis du consortium des journaux et conservé la propriété intellectuelle. En 1998, Hadar Pedhazur a investi dans l'entreprise, et nous avons concentré nos travaux Python, dans un seul et même produit Open Source : Zope. Une large communauté de développeurs pour la plupart issus de l'Open Source s'est construite autour du projet.

Python était dans notre sang dès le départ. Jim et moi sommes allés à la toute première conférence Python publique (20 personnes). Jim était alors considéré comme un, sinon le principal contributeur du noyau du langage Python.

Grâce à Python, nous étions capables de construire des systèmes web, comme des systèmes de petites annonces électroniques très dynamiques en un temps record, ce qui nous rendait très compétitifs.

Parallèlement, lorsque nous avons conçu Principia, le serveur d'applications propriétaire, nous avions décidé de cacher Python. Cette décision a eu un énorme impact aussi bien positif que négatif, sur le fonctionnement de Zope. Les idées de gérer tout un site à travers des interfaces d'administration en ligne, de stocker des portions de code restreint dans une base de données (ndlr : la ZODB), et d'étendre le serveur par des paquets d'extension, vinrent de cette décision.

Nous avons aussi apporté une nouvelle audience pour le langage Python, puisque beaucoup de gens qui choisissaient Zope, n'avaient jamais fait de Python auparavant (à la première conférence Zope à l'ICP8, la moitié de l'audience n'avait jamais utilisé Python avant Zope).

Malheureusement le choix de cacher Python a également généré une confusion sur ce qu'était Zope. La communauté Python jugeait Zope 2 comme un framework pas très Pythonique. De plus, Zope 2 lui-même vivait une crise d'identité : était-ce un produit destiné aux intégrateurs, ou un produit orienté développeur ?

Zope 3 a résolument pris un tournant en orientant le framework vers un outil pour développeurs.

Des journalistes comme Jon Udell ou Edd Dumbill considèrent que Zope est l'un des frameworks où l'Open Source a réellement vécu des innovations, pour la plupart issues des idées de Jim Fulton. Le langage Python influença beaucoup Jim dans ses idées, et offrit à Zope des fonctionnalités magnifiques : l'idée de publier des objets sur le Web est devenu un sujet informatique d'actualité, 9 ans après que Jim l'ait fait.

Une base de données transactionnelle distribuée d'objets Python, utilisée dans des sites commerciaux énormes, c'est un résultat impressionnant. L'histoire de Zope et Python est maintenant vieille de 10 ans. Place maintenant à un nouveau chapitre :

Zope 3 et son souhait d'être plus pythonique que son prédécesseur et d'intéresser d'avantage la communauté Python.

-- Paul Everitt, fondateur de Digital Creations.

BLOG L'histoire continue...

Il y a quelque temps, Guido van Rossum a initié un blog dédié à l'histoire de Python. Il contient beaucoup plus de détails que cette annexe. Un blog à surveiller donc, pour être au fait des derniers événements liés au langage.

- http://python-history.blogspot.com/