Python en signal-image: éléments sur le langage, les outils, l'architecture, les fonctionnalités et la communauté

Jean-Baptiste FASQUEL[1], Christian JEANGUILLAUME[1,2]

[1]LARIS (Laboratoire Angevin de Recherche en Ingénierie des Systèmes), 62 avenue Notre Dame du Lac, 49000 Angers, France

[2]C.H.U. d'Angers - Pôle Imagerie - Service de médecine nucléaire et biophysique, 4 Rue Larrey, 49000 Angers, France

Jean-Baptiste.Fasquel@univ-angers.fr, christian.jeanguillaume@univ-angers.fr

Résumé – Le langage généraliste et opensource Python, couplé à ses librairies scientifiques, constitue, pour le développement d'applications scientifiques, une alternative aux langages ou environnements tels que Matlab, Java et C++. Dans ce contexte, nous présentons les caractéristiques principales du langage, des outils associés, l'architecture et la diversité de son environnement scientifique. Nous illustrons son potentiel avec 4 applications simples couvrant un large spectre: l'imagerie médicale, la réalité augmentée, l'informatique embarquée et la programmation depuis un navigateur web. Nous donnons également quelques informations marquantes relatives à la communauté (classement du langage, nombre de librairies, usage en ingénierie, recherche et enseignement).

Abstract – The multi-purpose opensource Python language, together with related scientific packages, is, for developping scientific applications, an interesting alternative to languages or environments such as Matlab, Java and C++. We describe main langage features, related development tools, as well as the architecture and the diversity of its scientific environment. We illustrate its potential with 4 basic applications covering a wide range of possibilities: medical imaging, augmented reality, embedded computing and web-based scientific programming. We also provide some information concerning its community (ranking, number of librairies, usage for engineering, research and teaching purposes).

1 Introduction

En calcul scientifique et plus particulièrement en traitement du signal et de l'image, un enjeu est de faciliter le développement d'algorithmes (bas-niveau) et l'intégration de fonctionnalités hétérogènes au niveau logiciel (haut-niveau). Ainsi, à bas-niveau, il est nécessaire que le code des algorithmes soit générique par rapport à la nature du signal ou de l'image considérée pour faciliter sa réutilisation : type (e.g. entier, flottant), dimension (1D, 2D, nD), et éventuellement la méthode de parcours pour le traitement de régions d'intérêt [3]. A haut-niveau (niveau logiciel), les fonctionnalités peuvent être très diverses : interface graphique 2D, visualisation 3D, périphérique d'entrée/sortie (e.g. Leapmotion, Kinect, caméra video), système de traitement (e.g. Raspberry Pi, GPU), architecture logicielle spécifique (client lourd, application web, application distribuée), format de données (e.g. image DICOM, fichier video, son, bases de données, données de type "tableur"). Dans ce cas, l'objectif est de pouvoir aisément réutiliser des programmes existants, et de les assembler de manière modulaire en respectant les bonnes pratiques du génie logiciel ("design patterns", "architectural patterns" [4]).

Dans ce contexte, le langage Python, couplé à son écosystème de librairies scientifiques, est de plus en plus utilisé dans la communauté scientifique [8, 7, 2].

L'objectif est ici de présenter les caractéristiques principales de ce langage ainsi que son environnement, en terme de fonctionnalités, de structure mais également de communauté.

La première partie porte sur les caractéristiques principales, hors contexte scientifique, de ce langage et de son environnement (domaines d'application et outils de développement).

La seconde partie porte sur l'environnement scientifique en traitement du signal et de l'image. Nous présentons tout d'abord quelques librairies majeures, ainsi que la manière dont elles sont structurées (architecture) les unes par rapport aux autres. Afin d'illustrer la diversité des applications envisageables, nous présentons succinctement quelques applications réalisées relatives à l'imagerie médicale, à l'informatique embarqué, à l'interface homme-machine, et à la réalité augmentée. Enfin, nous discutons du positionnement de cet environnement dans la communauté scientifique.

2 Considérations générales

2.1 Le langage

Python est un langage opensource multiplateforme interprété, à typage dynamique fort et disposant d'une librairie standard riche (structures de données, mutithreading, fichiers, réseau, ...). Le fait que le langage soit interprété permet par exemple de s'affranchir de la compilation, nécessaire avec des langages compilés tels que le C++, avec les inconvénients associés (e.g. lenteur, cross-compilation sur des systèmes embarqués). Le typage dynamique permet une grande généricité aux programmes,

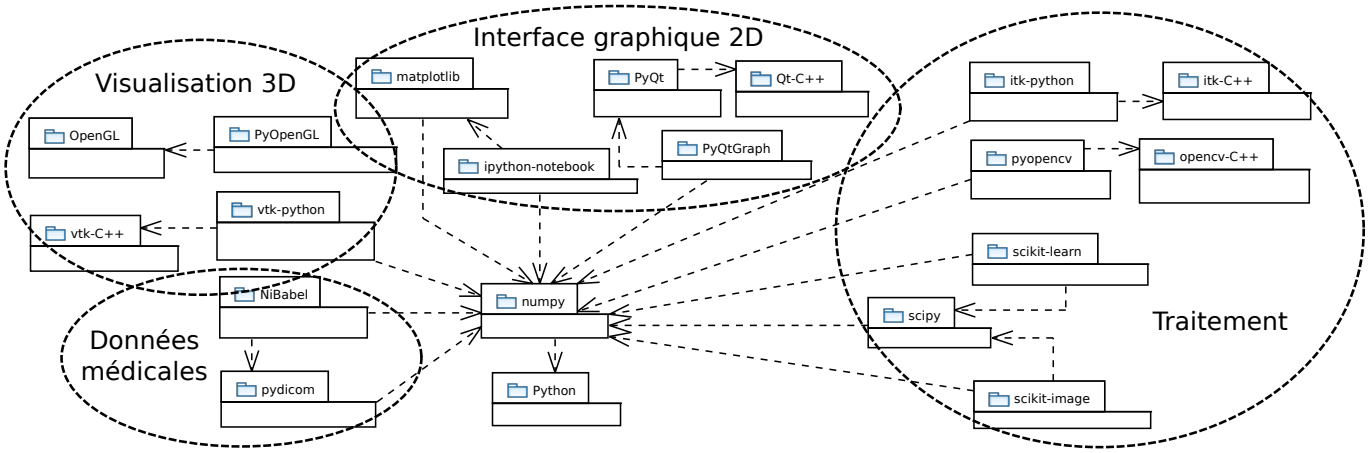

FIGURE 1 – Diagramme UML de paquetage impliquant quelques librairies scientifiques de l'écosystème Python.

qui peut s'adapter, à l'exécution, aux types des données fournies (e.g. problématique du type et de la dimension mentionnées dans l'introduction). Ceci est à comparer à la généricité seulement statique des langages tels que C++ ou Java avec le mécanisme des templates (polymorphisme de compilation).

Python est un langage multi-paradigmes [1]. En particulier, il permet la programmation procédurale et orientée objet (sans pour autant l'imposer comme Java). Python supporte l'introspection, permettant par exemple de dynamiquement de consulter (voire modifier dans certains cas) la composition d'une classe, d'un objet ou encore d'un fichier de code. Ceci facilite la programmation selon une approche orientée composants [5], permettant de développer des logiciels modulaires où les modules peuvent être inspectés et chargés dynamiquement (e.g. cas de plugins [4]).

2.2 L'environnement

Python est utilisé dans de nombreux domaines [1] tels que, par exemple, le calcul scientifique, l'administration système et le développement d'applications web (e.g. voir la récente étude comparative intégrant le framework Python Django [6]). Ainsi, le dépôt de référence des librairies Python recense 56415 librairies (PyPI [2]). Le classement TIOBE [3] place Python en 8ième position (6ième en excluant les langages propriétaires C# et Objective-C), derrière le Java et le C mais bien devant les outils purement scientifiques tels que Matlab et R par exemple (respectivement 19ième et 20ième positions).

Comme tout langage généraliste, l'environnement Python dispose de tous les outils d'industrialisation des développements, facilitant ainsi un prototypage logiciel de qualité : environnement de programmation (e.g. généraliste tel que le plugin eclipse PyDev, spécialisé tel que spyder pour le calcul scientifique), génération de la documentation et exécution des tutoriaux (e.g. sphynx [4]), tests unitaires, gestion des dépendances et des distributions (e.g. pip, plus général que les distributions telles que PythonXY) et de déploiement [5] (i.e. génération d'une application exécutable sur un ordinateur ne disposant ni de Python, ni d'environnement de développement).

3 Le contexte signal-image

3.1 Architecture : quelques éléments

Dans le domaine des sciences et plus particulièrement du traitement du signal et des images, Python dispose d'une grande diversité de librairies scientifiques opensources : par exemple, PyPI recense 3617 packages dans la catégorie Scientific/Engineering. Du fait de cette grande variété et du manque de cohésion dans le développement de ces libraires, une difficulté est de disposer d'une vision claire des librairies fondamentales et de leurs interpendances (ce point étant bien géré par Matlab avec une politique maîtrisée de "toolboxes"). Ce dernier point est essentiel pour le développement modulaire selon une approche par composant où les composants (e.g. "plugins") sont par définition [5] des modules présentant notamment des dépendances identifiées et limitées. Une vision claire de l'architecture peut également aider dans le choix des librairies.

Pour apporter quelques éléments de réponse, la figure 1 fournit un diagramme ULM de paquetages intégrant quelques librairies que nous avons pratiquées, et qui nous semblent incontournables (e.g. richesse des fonctionnalités, de la documentation, de la communauté). On constate que la librairie fondamentale est *numpy*, dont (presque) tous les autres packages dépendent, du fait de sa structure de tableau générique (typage dynamique) n-dimensionnel, permettant de dynamiquement gérer les signaux et images quelque soit leur type et leur dimension (1D, 2D+t, 2D couleur, 3D,...).

Ce diagramme indique également les librairies développées

1. https ://wiki.python.org/moin/OrganizationsUsingPython
2. Python Package Index : https ://pypi.python.org/pypi
3. http ://www.tiobe.com/index.php/content/paperinfo/tpci/index.html
4. http ://sphinx-doc.org/
5. http ://docs.python-guide.org/en/latest/shipping/freezing/

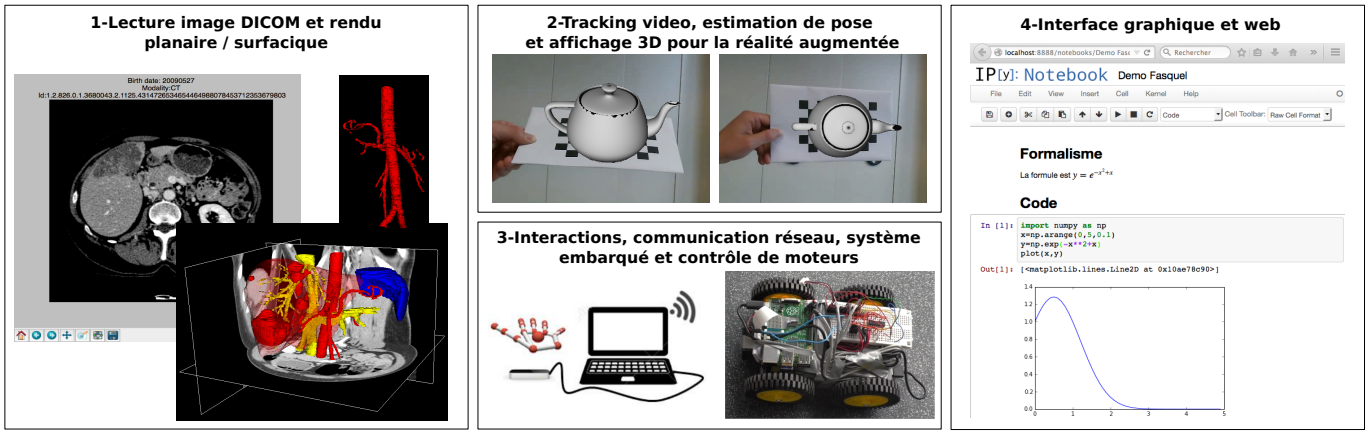

FIGURE 2 – Diversité de l'écosystème scientifique de Python : quelques exemples de réalisations.

spécifiquement pour Python (e.g. les scikits) ainsi que celles obtenues par surcouche d'une librairie écrite dans un autre langage (souvent en "C++"). Ceci est une force de Python (notion courante de multi-langage) qui offre l'accès à des librairies de référence (anciennes par rapport à Python) souvent écrites en C++. Il est à noter que ces librairies ont été réparties spatialement par catégorie : les interfaces graphique 2D, la visualisation 3D, les données (limité au cas des données médicales) et le traitement (scipy, scikit-learn, scikit-image, opencv, itk). Ceci illustre la richesse des fonctionnalités disponibles (ceci n'étant qu'un aperçu). A noter que beaucoup de librairies sont pluridisciplinaires (e.g. opencv permet de lire de flux video, numpy fournit beaucoup d'algorithmes relatifs à l'algèbre linéaire, aux statistiques,...), ceci n'étant pas indiqué sur le diagramme pour en simplifier la représentation. Dans un souci de clarté, certaines dépendances mineures ont été ignorées (e.g. les "scikits" dépendent de matplotlib).

Pour répondre aux problèmes de structuration des librairies Python, on peut mentionner l'initiative des "scikits" (toolkits pour scipy [6]) qui vise à rationnaliser les librairies scientifiques, dans l'esprit des boites à outils de Matlab. Seulement deux modules sont rapportés sur le figure 1, mais des dizaines d'autres existent (e.g. statistiques, optimisation, traitement de la parole...).

Un autre point important concerne les performances en temps de calcul de Python, a priori mauvaises du fait de son niveau d'abstraction. Une récente étude a souligné que l'on peut obtenir des performances similaires à celles obtenues avec un programme écrit en C [2], notamment par conversion en C++ puis compilation à l'exécution. A noter qu'en utilisant des libraires C avec une surcouche en Python, le problème ne se pose pas nécessairement : par exemple, l'exécution d'une opération invoquée en Python sur un tableau NumPy est déléguée à un code compilé écrit en C.

A notre connaissance, il n'existe pas (encore) de logiciel similaire au logiciel Java ImageJ [7] (dédié à l'image) : ceci est un avantage relatif de Java puisque ce type d'outil se concentre plutôt sur la création de plugins (venant étendre les capacités du logiciel) que sur la création d'applications spécifiques. A noter tout de même qu'il existe des quelques logiciels Python dédiés aux sciences, comme Sage [8] (logiciel de Mathématiques) et blender [9] (logiciel de modélisation 3D).

3.2 Applications : quelques exemples

L'objectif est de présenter succinctement quelques exemples de programmes que nous avons réalisés (voir figure 2), et choisis pour couvrir assez largement les possibilités.

La figure 2-1 concerne la lecture d'une image DICOM (données pixels et champs DICOM - librairie PyDicom) et l'affichage à l'aide la librairie matplotlib (seulement 2 lignes de code). Le maillage surfacique est généré et affiché à l'aide de vtk (environ 20 lignes de code).

La figure 2-2 montre un exemple d'application de réalité augmentée (environ 300 lignes de code) : le damier est détecté à chaque image grâce à la librairie OpenCV, permettant d'estimer la pose du damier (estimation des coefficients de la matrice extrinsèque également avec OpenCV). L'affichage est finalement géré avec la librairie PyOpenGL. L'ensemble de cette application fonctionne à cadence vidéo sans latence, les calculs étant effectués de manière sous-jacente par une librairie écrite et compilée en C++ (OpenCV).

La figure 2-3 concerne un système de pilotage à distance, par le périphérique LeapMotion, d'un véhicule motorisé équipé d'une caméra : le langage Python est utilisé pour détecter l'orientation de la main par LeapMotion (marche avant/arrière, orientation des roues) et d'envoyer les instructions au véhicule par communication réseau sans fil (environ 100 lignes de code). Le code embarqué sur le véhicule (Raspberry Pi) est également écrit en Python, reçoit les instructions et envoie les commandes aux moteurs des roues sous forme de signaux PWM (environ 150 lignes de code).

La figure 2-4 montre les possibilités offertes par IPython no-

6. https ://scikits.appspot.com/
7. http ://imagej.nih.gov/ij/
8. http ://www.sagemath.org/fr/
9. blender

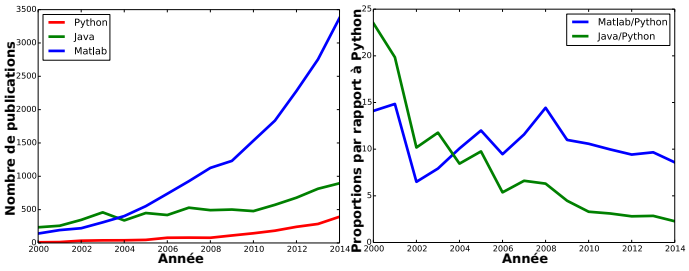

FIGURE 3 – Illustration du référencement de Python, Matlab et Java dans les articles référencés par sciencedirect depuis 2000. Gauche : évolution du nombre de publications. Droite : évolution des rapports Matlab/Python et Java/Python.

tebook [10], permettant d'interagir dynamiquement avec Python et ses librairies scientifiques depuis un navigateur web (assimilé à une interface graphique dans la figure 1). Cet outil n'a initialement pas vocation à développer des applications, mais davantage à documenter, formaliser mathématiquement et programmer des algorithmes. Ainsi ce système supporte, dans un seul environnement (page web d'un navigateur), le traitement de texte, la compilation de formules mathématiques écrites en latex, et l'exécution (déléguée au serveur) de code Python.

3.3 Communauté : quelques éléments

Comme mentionné précédemment, le classement du langage et le nombre de librairies montrent l'importance de la communauté en terme de développement.

Afin de compléter ceci, il nous est paru intéressant d'observer l'usage des environnements Python, Matlab et Java dans la production scientifique. Ainsi, à titre indicatif, nous avons mesuré le nombre de publications référencées par Science Directe [11] depuis l'année 2000, dans les catégories "computer science", "engineering" et "mathematics", et contenant les mots-clés "Python et Image", "Matlab et Image" et enfin "Java et Image". La figure 3-gauche montre que Matlab domine nettement, Python occupant la troisième place. Ceci provient probablement en partie de l'usage relativement récent de Python en sciences (Matlab est utilisé depuis les années 80) : la production concernée en 2000 n'est que de 10 pour Python (contre 141 Matlab et 235 pour Java). En s'intéressant à l'évolution des rapports (figure 3-droite), on observe que, notamment depuis 2008, Python croît en proportion : facteur 15 en 2008 contre environ 9 en 2014. Le phénomène est davantage marqué pour Java. Il serait intéressant d'affiner cette étude, par exemple en considérant d'autres mots-clés et la base IEEExplore.

Indépendamment du développement et de la production scientifique, on peut noter qu'au niveau des langages enseignés, Python est également largement utilisé : une récente étude de ACM [12] place par exemple Python en première place des langages enseignés dans les universités américaines (devant Matlab et Java).

10. http ://ipython.org/notebook.html
11. http ://www.sciencedirect.com/
12. http ://cacm.acm.org/

4 Conclusion

Python est avant tout un langage généraliste gratuit largement répandu, et disposant de mécanismes et d'outils riches facilitant le développement rapide d'applications modulaires et performantes. Il dispose de nombreuses librairies scientifiques, couvrant énormément de problématiques traitées en traitement du signal et des images, permettant de développer aisément des applications très diverses, comme illustré dans ce document. Il faut retenir que ces librairies sont essentiellement centrées sur la librairie fondamentale numpy, avec son tableau générique n-dimensionnel, tout à fait approprié à la diversité des signaux et images généralement traités. Bien qu'encore sous exploité en sciences (par rapport à Matlab par exemple), la communauté des utilisateurs scientifiques (et non scientifiques) est importante, favorisant la pérenité de cet outil et les échanges entre les acteurs de la recherche. Ainsi, cet outil fournit une alternative intéressante à des solutions propriétaires et non généralistes telles que Matlab, tout en étant plus adaptés que d'autres langages généralistes tels que Java ou C++.

Références

[1] A Martelli, *Python in a Nutshell*, O'Reilly Media, Inc., 2006

[2] J. Akeret and L. Gamper and A. Amara and A. Refregier, *HOPE : A Python just-in-time compiler for astrophysical computations*, Astronomy and Computing, 2015

[3] J.B. Fasquel and V. Agnus and J. Lamy, *An efficient and generic extension to ITK to process arbitrary shaped regions of interest*, Computer Methods and Programs in Biomedecine, 2006

[4] J.-B. Fasquel and Johan Moreau, *A design pattern coupling role and component concepts : Application to medical software*, The Journal of Systems and Software, 84, 847-863, 2011

[5] C. Szyperski, *Component Software : Beyond Object-Oriented Programming*, Addison-Wesley Longman Publishing Co., Inc., 2002

[6] M. Salas-Zarate and G. Alor-Hernandez and R. Valencia-Garcia and L. Rodriguez-Mazahua and A. Rodriguez-Gonzalez and J. Lopez Cuadrado, *Analyzing best practices on Web development frameworks : The lift approach*, Science of Computer Programming,2015

[7] J. M. Perkel, *Programming : Pick up Python*, Nature, 2015

[8] H. P. Langtangen *A Primer on Scientific Programming with Python*, Springer, 2012