

Jean-Paul Roy

Python

Apprentissage actif

pour l'étudiant et le futur enseignant



petite musique japonaise
atmosphère zen...

Introduction

Le langage Python

Les programmeurs ont coutume de dire que les langages de programmation sont *universels*. Chaque langage contient les constructions linguistiques suffisantes pour exprimer n'importe quel calcul, et si les professionnels acceptent de passer par les fourches caudines de langages industriels complexes, efficaces et sécurisés, le choix d'un *petit langage* suffisamment performant à enseigner aux grands débutants s'est posé dès le début. En 1964 naquit le langage **BASIC**¹, comme simplification de son grand aîné **FORTRAN**² (1955). BASIC a été le plus populaire en son temps, accompagnant la naissance de la micro-informatique et des premiers ordinateurs personnels. Le langage interactif **LISP**³ (1958) fut au cœur de l'apparition de l'Intelligence Artificielle, et sa simplification **Logo** fut aussi utilisée à l'école dès la fin des années 1960 dans le cadre de recherches sur le développement cognitif de l'enfant, puis de l'enseignement des bases de la programmation. La maturation des langages industriels continuait d'être mise à profit par des chercheurs souhaitant disposer de langages de script permettant de piloter de gros systèmes logiciels, mais aussi par des universitaires soucieux de langages à vocation pédagogique pour remplacer ce BASIC un peu spartiate.

Le langage ABC développé à l'institut CWI⁴ d'Amsterdam visait ces deux objectifs. C'est Guido van Rossum, chercheur entré au CWI en 1983 et influencé par les idées et les types de données présents dans ABC, qui définit le langage **Python** en 1989, un petit langage bientôt appelé à jouer dans la cour des grands.

L'ambition de son créateur (qui se définissait lui-même comme *dictateur bénévole à vie* jusqu'à ce qu'il se retire en 2018 du comité décisionnel de la *Python Software Foundation*) consistait à créer un langage interactif à objets le plus simple possible, basé sur le langage C⁵. Il s'appuyait entre autres sur une phrase

-
1. **B**eginner's **A**ll-purpose **S**ymbolic **I**nstruction **C**ode, par John Kemeny et Thomas Kurtz.
 2. **F**ORmula **T**RANslator, par John Backus.
 3. **L**ISt **P**rocessing, par John McCarthy.
 4. **C**entrum **W**iskunde & **I**nformatica, où *Wiskunde* signifie *Mathématiques* en néerlandais.
 5. L'implémentation de Python par la maison-mère www.python.org se nomme CPython, c'est elle que nous utiliserons majoritairement dans ce livre. Mais nous rencontrerons aussi Jython qui est basé sur Java, ainsi que le compilateur Cython pour les programmeurs C.

d'Albert Einstein : *Les choses doivent être aussi simples que possible, mais pas plus simples.* Le but fut assez remarquablement atteint : la syntaxe est légère, toute valeur est un *objet* (les types sont les classes) que l'on peut ranger dans une variable, passer en argument à une fonction, *etc.*

Certains ont noté une ressemblance avec le langage LISP, mais Guido van Rossum n'a jamais été un grand partisan de la programmation fonctionnelle, bien qu'ayant doté Python des *lambda-fonctions* de LISP (dans une version très limitée) et de primitives comme `map` ou `filter`, voire `reduce`⁶. Ce que van Rossum n'a d'ailleurs jamais voulu reprendre de LISP, et que l'on peut regretter, c'est l'élimination automatique de la récurrence terminale, pour des raisons liées à la mise au point. De son point de vue, la programmation par récurrence doit être évitée. C'est une opinion qui n'engage que lui, mais de fait la programmation en Python utilise majoritairement des itérations et des constructions fonctionnelles comme les *compréhensions de listes*. Les parenthèses de LISP ou les accolades de C/Java, qui structurent les phrases d'un programme, sont remplacées par une *indentation* (distance à la marge) *obligatoire*. Python reste assez solitaire dans cette voie qui s'avère pourtant un bon atout pédagogique d'après mon expérience d'enseignant. Le langage ABC, ou Occam au même moment, tous deux oubliés, en faisaient autant.

LISP (1958)	Python (1989)	Java (1995)
<pre>(if (> x 0) (print x) (print y))</pre>	<pre>if x > 0: print(x) else: print(y)</pre>	<pre>if (x > 0) { System.out.println(x); } else { System.out.println(y); }</pre>

Comme LISP et contrairement à C ou Java, le langage Python est *dynamiquement typé* : les variables sont simplement des noms qui font référence à des objets en mémoire. Comme les autres langages dynamiques, la vérification du typage se fait à l'exécution et non lors d'une phase de compilation.

Les programmes sont organisés en *modules*, permettant de développer ou faire appel à des fonctionnalités distinctes dans des fichiers séparés. Une large bibliothèque de modules est fournie en standard et nous aurons l'occasion d'en explorer plusieurs, ainsi que certains en provenance de sources extérieures.

J'ai choisi la distribution **Anaconda**, car elle contient le langage Python mais aussi un grand nombre de logiciels scientifiques parfois pénibles à installer individuellement à cause de leurs dépendances. Les systèmes Anaconda et Jupyter mériteraient un livre à eux seuls, j'en dirai le minimum pour les rendre vite opérationnels, en privilégiant le fond sur la forme.

6. L'algorithme qui fit la gloire de Google se nomme *MapReduce* en l'honneur des fonctions `map` et `reduce` mises en avant par LISP.

Après quelques hésitations, il m'a paru intéressant de présenter aussi le système *Processing*, destiné aux artistes numériques et organisé autour du langage Java. Depuis peu, il existe un mode Python à l'intérieur de *Processing*, ce dernier prenant alors le nom de **Processing.py**. Nous aurons l'occasion d'y faire bien entendu du graphisme, mais aussi d'entr'ouvrir la porte de la musique électronique (avec la librairie *Beads* de Java). Nous verrons à cette occasion une autre implémentation de Python qui se nomme *Jython* et qui permet l'accès en Python 2 à des modules programmés en Java. Hélas, à ce jour nous serons en effet quelques rares fois obligés d'utiliser du Python dans son ancienne version 2 et devons nous adapter. Je n'ai pas voulu masquer la réalité des faits, qui sont têtus car il peut s'avérer nécessaire d'utiliser d'autres langages voire d'autres versions de Python, cela fait partie de la vie du programmeur. Nous pénétrerons aussi dans les grosses bibliothèques mathématiques **numpy** et **sympy** essentielles au scientifique travaillant en Python. Nous n'y passerons pas un temps démesuré car il existe des ouvrages spécialisés pour un travail plus approfondi. J'ai été guidé par un souci constant de vous *mettre le pied à l'étrier* et de vous laisser lors des exercices consulter toutes les ressources du Web, notamment les documentations en ligne.

Guide de lecture

La première partie est là pour éclaircir le langage lui-même, en détaillant ses constructions linguistiques et ses tournures idiomatiques les plus fréquentes, sans vouloir faire un manuel de référence complet que l'on peut trouver par ailleurs sur <https://docs.python.org/fr/3/>. Ayant observé que les étudiants lisaient de moins en moins les livres de A à Z, chapitre après chapitre (moi aussi je l'avoue), j'ai décidé de structurer le texte comme une FAQ sur Internet (*Frequently Asked Questions*) en essayant d'insister sur les points qui cristallisaient la plus forte demande d'éclaircissements tout en gardant une progression suffisamment linéaire, sans m'y plier complètement, pour le lecteur préférant un apprentissage plus traditionnel. Vous téléchargerez les solutions en Python sur la page Web de ce livre aux Editions Ellipses (www.editions-ellipses.fr) ainsi que sur :

<http://jean.paul.roy.free.fr>

Qui dit programmation dit algorithmes, mais ce livre traite de la pratique du codage. Ce n'est ni un cours ni un ouvrage d'algorithmique qui explorerait toutes les structures de données. Il vous faudrait pour ces dernières par exemple l'indispensable [COR]⁷ qui fait à juste titre abstraction du langage de programmation, ou des ouvrages comme [DUR] s'appuyant explicitement sur Python.

7. Cette notation [COR] renvoie au livre de Cormen & al. cité en bibliographie.

La seconde partie consiste en une sélection de petits exercices ou problèmes de programmation. En plus de la partie algorithmique de base, essentielle, j'ai dû faire des choix et en ai profité pour vous emmener dans des zones moins enseignées comme le graphisme, l'analyse syntaxique, l'intelligence artificielle ou la musique électronique. Je ne saurais trop insister sur l'**apprentissage actif** du lecteur. *Hands on!* comme disent les anglo-saxons, mettez les mains sur le clavier. Mais rappelez-vous : *le plus simple possible, mais pas plus simple*. Ce sera donc un peu délicat par endroits. Lisez et programmez à votre rythme, consultez les références et aides en ligne, cherchez des compléments sur le Web, travaillez en groupe. L'important consiste à passer des obstacles, à avoir le sentiment que l'on a gravi une marche de plus : il vous faut sentir la progression. Si vous résolvez vite tous les problèmes en tapant sur le clavier avec le seul petit doigt de la main gauche, ce livre vous aura été sans doute inutile.

Étudiant en Faculté des Sciences, en Ecole d'Ingénieurs, en IUT, peut-être futur enseignant des lycées préparant le CAPES d'Informatique ou déjà chargé de l'option informatique et utilisant sans doute Python, j'espère que ce livre vous plaira et que la partie *Exercices* vous donnera envie d'aller plus loin. J'ai humblement essayé de faire le livre que j'aurais voulu tenir entre les mains en étant étudiant.

Et si vous êtes simplement un amateur de programmation, sachez que le contenu utilise un peu de mathématiques sans en faire un point de mire. Sa lecture ne requiert jamais plus qu'un niveau de classe terminale scientifique ou exceptionnellement de première année post-Bac. Pour le lecteur à la recherche de sensations plus fortes, il existe des ouvrages de programmation scientifique comme [KAR] ou [LAN], ou vraiment axés sur les mathématiques : [ZIM] ou [BAR] avec le logiciel SAGE souplement basé sur Python.

La bibliographie contient aussi des références en anglais lorsque je n'ai pas trouvé de livres équivalents en français. J'en suis désolé mais à ma décharge, la pratique de l'anglais est devenue indispensable dans toute activité scientifique.

Remerciements

Cet ouvrage est en majeure partie issu de cours donnés en Licence au Département Informatique de l'UFR Sciences de l'Université Nice Sophia-Antipolis. Je remercie les nombreux enseignants et moniteurs ayant collaboré à ces enseignements et qui m'ont fait des retours instructifs. Merci aussi à celles et ceux qui ont relu certaines parties de ce livre. S'il reste des erreurs, il va de soi que j'en suis l'unique responsable.

Table des matières

I	L'essentiel du langage	1
1	Environnement de programmation	3
1.1	Installation de la distribution Anaconda	3
1.2	L'éditeur IDLE et sa console	4
1.3	Le gestionnaire de paquets <code>conda</code>	5
1.4	L'interface graphique : Anaconda Navigator	9
1.5	Un autre IDE pour Anaconda : Spyder	10
1.5.1	L'éditeur de Spyder	11
1.5.2	La console de Spyder	11
1.6	Le projet Jupyter	12
1.6.1	Les <i>notebooks</i> de Jupyter	13
1.6.2	Jupyter Lab	15
1.7	Conclusion	15
2	Variables, nombres et fonctions	17
2.1	Qu'est-ce qu'une variable ?	17
2.2	Qu'est-ce qu'un type ?	18
2.3	Le type <code>int</code> des nombres entiers	20
2.4	Les nombres flottants	20
2.5	Les nombres complexes	22
2.6	Qu'est-ce qu'un module en Python ?	22
2.7	Donc pas de nombres rationnels en Python ?	24
2.8	L'égalité est-elle fiable sur les nombres flottants ?	24
2.9	L'infini existe-t-il en programmation ?	25
2.10	Qu'est-ce que la priorité d'un opérateur ?	25
2.11	Comment construire couples, triplets, <i>etc.</i> ?	26
2.12	Quelle différence entre expression et instruction ?	27
2.13	Les expressions booléennes	28

2.14	Les instructions conditionnelles	30
2.15	Comment tirer un nombre au hasard ?	32
2.16	Fonction, paramètres et variables locales	32
2.17	Qu'est-ce qu'une variable globale ?	34
2.18	Les différentes formes de paramètres	36
2.19	Les arguments optionnels de la fonction <code>print</code>	37
2.20	Les lambda-expressions	38
2.21	Comment documenter une fonction ?	40
2.22	Quelle est la différence entre fonction et méthode ?	41
2.23	Qu'est-ce qu'une fonction récursive ?	41
	2.23.1 Suivre une fonction à la trace	43
	2.23.2 Le rôle de la pile de récursion	44
2.24	La boucle <code>while</code>	47
2.25	La boucle <code>for</code>	49
2.26	Comment interrompre brutalement une boucle ?	51
2.27	Le calcul d'une somme en compréhension	51
2.28	La complexité d'un calcul	52
3	Traitement du texte : les bases	55
3.1	Les caractères Unicode	55
3.2	Les textes sont des chaînes de caractères	56
3.3	Comment programmer avec des chaînes ?	59
3.4	Les chaînes formatées	60
3.5	Une chaîne peut-elle contenir du code Python ?	61
4	Dessiner	65
4.1	Le graphisme de la tortue	65
	4.1.1 La fenêtre graphique	65
	4.1.2 Le graphisme cartésien	66
	4.1.3 Le graphisme polaire	69
	4.1.4 Une courbe fractale	70
4.2	Le système <i>Processing.py</i>	72
4.3	Stéganographie avec le module <i>Pillow</i>	75
5	Les collections d'objets	79
5.1	Collections et séquences	79
5.2	Les listes sont des séquences mutables	79
5.3	Les constructions fonctionnelles de listes	81
5.4	Déstructuration d'un objet itérable	83
5.5	Suppression d'objets avec <code>del</code>	84
5.6	Listes et partage de mémoire	84

5.7	Les ensembles	88
5.8	Les dictionnaires	90
5.9	Gestion des erreurs : les exceptions	93
6	Les entrées-sorties	97
6.1	Le système de fichiers et le module <code>os</code>	97
6.2	Écriture dans un fichier texte sur disque (en sortie)	99
6.3	Lecture d'un fichier texte sur disque (en entrée)	100
6.3.1	La construction <code>with</code>	102
6.3.2	L'utilisation de <code>split</code>	103
6.4	Des fichiers en mémoire centrale	103
6.5	Exécution d'un script Python au terminal	104
7	Les objets et les classes	107
7.1	Définition d'une nouvelle classe d'objets	107
7.2	Les attributs de classe	112
7.3	Utilisation des méthodes spéciales	114
7.4	Boucle <code>for</code> et méthodes spéciales : les itérateurs	114
7.5	Les générateurs	116
7.6	Les objets <i>soft</i>	117
7.7	La hiérarchie des classes	118
7.8	Les classes abstraites	121
7.9	Qu'est-ce qu'un décorateur ?	122
8	Traitement avancé du texte	125
8.1	Expressions régulières : le module <code>re</code>	125
8.1.1	Présence d'une sous-chaîne dans une chaîne	126
8.1.2	Les méta-caractères d'un motif	127
8.1.3	Remplacements dans une chaîne	130
8.2	Analyse lexicale avec l'outil <code>lex</code>	131
8.2.1	Installation avec <code>pip</code>	131
8.2.2	Programmation d'un analyseur lexical	132
8.3	Analyse syntaxique avec l'outil <code>yacc</code>	134
8.4	Recherche de texte dans une page Web : <code>requests</code>	138
9	Les interfaces graphiques (GUI)	139
9.1	Le module <code>tkinter</code>	139
9.1.1	La programmation dirigée par les événements	139
9.1.2	La fenêtre du programme	140
9.1.3	Dessiner avec les classes <code>Canvas</code> et <code>Button</code>	141
9.1.4	L'interface graphique comme sous-classe de <code>Tk</code>	144

9.1.5	Les options du gestionnaire <code>pack</code>	145
9.1.6	Gestion des évènements souris et clavier	145
9.1.7	Variables de contrôle	148
9.1.8	Compléments sur les évènements souris	150
9.1.9	Animation en style MVC avec <code>tkinter</code>	152
9.1.10	La tortue dans une interface graphique	155
9.2	Programmer une animation avec <code>pygame</code>	156
10	De Java vers Python	159
10.1	Le typage : fort ou dynamique ?	159
10.2	Les boucles	161
10.3	Les listes	162
10.4	Les classes d'objets	163
11	EXERCICES	167
	Nombres, fonctions, boucles	
11.1	Valeur d'une expression	167
11.2	Quotient et reste. Écriture binaire d'un entier	167
11.3	Écriture binaire d'un nombre flottant	167
11.4	Avec ou sans résultat ?	168
11.5	Heures – minutes – secondes	169
11.6	Décomposition en sous-fonctions	169
11.7	Une fonction, comment ça marche ?	169
11.8	Addition relativiste des vitesses	170
11.9	La formule de Stirling	170
11.10	Payez vos impôts !	170
11.11	Une variable locale dans une lambda ?	171
11.12	Tirage au hasard de nombres entiers	171
11.13	Nombres fractionnaires (rationnels)	172
11.14	Solutions complexes d'une équation de degré 2	172
11.15	Nombres complexes et géométrie plane	172
11.16	La division euclidienne dans les entiers	173
11.17	Travail arithmétique sur les chiffres d'un entier	173
11.18	Boucles <code>for</code> emboîtées	174
11.19	Calcul de x^n par dichotomie	174
11.20	Tester si un nombre entier est premier (naïf)	174
11.21	Nombres premiers entre eux. Le PGCD	175
11.22	Calcul de a^b modulo n dans les grands entiers	176
11.23	Tester si un grand nombre est premier (Fermat)	176
11.24	Calcul des nombres de Fibonacci	177
11.25	Vérification qu'un dé est correctement truqué	178

11.26	Dérivation numérique approchée	179
11.27	Approximation numérique d'un sinus	180
11.28	Valeur approchée d'une intégrale simple	180
11.29	Coût du (temps de) calcul de $n!$	182
11.30	Rectification d'un arc paramétré	183
11.31	Valeur approchée d'une intégrale double	184
11.32	Centre de gravité d'une figure plane	185
11.33	Solution approchée d'une équation (dichotomie)	185
11.34	Solution approchée d'une équation (Newton)	186
11.35	Régime d'un système dépendant du temps	186
11.36	Cryptographie à clé publique (RSA/1)	187

Chaînes de caractères

11.37	Affichage d'une table de multiplication	188
11.38	Construction de l'alphabet minuscule	188
11.39	Les plus longs mots d'une phrase	188
11.40	Travail textuel sur les chiffres d'un entier	189
11.41	Récurrance sur des chaînes	189
11.42	Le codage secret selon César	190
11.43	Les palindromes	190
11.44	Le nombre de Champernowne	191
11.45	Vérification d'un code bancaire IBAN	191
11.46	Les règles de réécriture	192
11.47	L'addition binaire avec retenue	192
11.48	Analyse lexicale à la main	193
11.49	HTML et Unicode	194

Graphisme de la tortue, Processing

11.50	Graphismes polaire et cartésien	194
11.51	Tracé d'un cercle avec centre et rayon	195
11.52	Mangez des oursins!	195
11.53	Un traceur minimal de courbes paramétrées	195
11.54	Dessin d'une courbe en coordonnées polaires	196
11.55	Le jeu du chaos	197
11.56	Une courbe fractale du dragon	197
11.57	La tortue, le hasard et les flaques	198
11.58	Génération d'une fractale par un L-système	199
11.59	Graphisme avec <i>Processing.py</i>	200

Les séquences

11.60	Tirage d'un élément aléatoire dans une séquence	202
11.61	Courbe d'un système dépendant du temps	203

11.62	Inversion des mots d'une phrase	203
11.63	Recherche séquentielle dans une liste non triée	203
11.64	Comptage d'éléments dans une collection	204
11.65	Extraction d'éléments dans une séquence	204
11.66	Recherche dichotomique dans une liste triée	205
11.67	Compactage d'une liste	205
11.68	Le drapeau hollandais de Dijkstra	206
11.69	Les fonctionnelles <code>map</code> et <code>reduce</code>	207
11.70	Vérification des éléments d'une collection	208
11.71	Attention aux mutations dans les listes!	208
11.72	Le triangle de Pascal	209
11.73	Tri stupide d'une liste	209
11.74	Tri d'une liste sur place, par sélection	211
11.75	Chronométrage d'un algorithme sur une longue liste aléatoire	211
11.76	Tri par insertion : récurrence et itération	212
11.77	Tri d'une liste avec copie, par fusion	213
11.78	Tri rapide	214
11.79	Trier sur des critères exotiques	215
11.80	Crible d'Ératosthène pour les nombres premiers	215
11.81	Cryptographie à clé publique (RSA/2)	216
11.82	Unicode et les bizarreries typographiques	217
11.83	Opérations sur des matrices	218
11.84	Construction d'un carré magique d'ordre impair	219
11.85	Message caché dans une image	220
11.86	Automates cellulaires 1-D	220
11.87	Les piles de données	222
11.88	Retour sur la pile de récursion	223
11.89	Polynômes creux	225
11.90	Polynômes de Legendre	227
11.91	Polynôme d'Interpolation de Lagrange	228
11.92	Racines simples d'un polynôme (1)	229
11.93	Racines simples d'un polynôme (2)	231
Itérateurs, ensembles et dictionnaires		
11.94	Fermeture-éclair : l'itérateur <code>zip</code>	232
11.95	Générateur des éléments distincts d'une liste	233
11.96	Réunion et intersection de listes	233
11.97	Produit cartésien d'ensembles	234
11.98	Combinaisons dans un ensemble	234
11.99	Dictionnaires et fréquences	235
11.100	Mémorisation : remplacer le temps par l'espace	236

11.101	Automate cellulaire 2-D : le Jeu de la Vie	237
11.102	Arbres binaires d'expression	239
11.103	La machine à pile VRISC1	242
11.104	La machine VRISC2 : les sauts à étiquette	243
11.105	Parcours itératif d'un arbre binaire	245
11.106	Introduction à la simplification symbolique	246
11.107	L'algorithme de dérivation symbolique	248
11.108	Calcul numérique avec <code>numpy</code>	250
11.109	Ajustement numérique d'une courbe avec <code>scipy</code>	253
Expressions régulières		
11.110	Quelques expressions régulières	255
11.111	Mais tout n'est pas régulier!	256
11.112	Fin de l'analyseur lexical du § 8.2 avec <code>lex</code>	257
11.113	Fin de l'analyseur syntaxique du § 8.3 avec <code>yacc</code>	258
11.114	Une calculette interactive avec <code>lex</code> & <code>yacc</code>	259
	11.114.1 L'analyseur lexical (<code>lexer</code>)	260
	11.114.2 L'analyseur syntaxique (<code>parser</code>)	261
	11.114.3 L'évaluateur	262
11.115	Retour sur les <code>cpbb</code> à dos de <code>yacc</code>	264
Fichiers, Web		
11.116	Écriture dans un fichier texte sur disque	264
11.117	Lecture d'un fichier texte sur disque	265
11.118	Lecture d'un fichier au format CSV	265
11.119	Transformation d'un fichier texte sur disque	266
11.120	Construction d'un graphique Excel en Python	266
11.121	Trouver une heure locale sur Internet	268
11.122	Échange de données au format JSON	269
11.123	Sérialisation des données : <code>json</code> ou <code>pickle</code> ?	270
11.124	Lancer un programme à heure donnée : <code>cron</code>	271
11.125	Dissection d'une page HTML avec Beautiful Soup	272
Objets		
11.126	Classe des matrices $n \times p$ sans <code>numpy</code>	276
11.127	Naissance d'une tortue avec <code>Processing.py</code>	277
11.128	Incursion dans le graphisme 3D	281
11.129	Ajustement par programmation génétique	283
11.130	Introduction au moteur physique <code>pymunk</code>	286
11.131	Moteur physique appliqué : pendule de Newton	292
11.132	Décoration de la récurrence terminale	294
11.133	Neurones : le perceptron	295

11.134	Neurones : la rétro-propagation (théorie)	298
11.135	Neurones : la rétro-propagation (pratique)	301
11.136	Programmation audio (théorie)	304
11.137	Programmation audio avec Beads (pratique)	308
Bibliographie		313
Index		317