Python 3

Chez le même éditeur

Python précis et concis
5e édition
Mark Lutz
272 pages
Dunod, 2017

Python pour le data scientist
Emmanuel Jakobowicz
304 pages
Dunod, 2018

Python 3

Apprendre à programmer dans l'écosystème Python

Bob Cordeau
Ancien ingénieur d'études à l'Onera
Ancien enseignant à l'Université Paris-Saclay

Laurent Pointal
Informaticien au LIMSI/CNRS
Chargé de cours à l'Université Paris-Saclay – IUT d'Orsay

Préface de **Gérard Swinnen**

2e édition

DUNOD

Toutes les marques citées dans cet ouvrage
sont des marques déposées par leurs propriétaires respectifs.

Illustrations intérieures :
© Hélène Cordeau

Illustration de couverture :
© Rachid Maraï

© Dunod, 2017, 2020
11 rue Paul Bert, 92240 Malakoff
www.dunod.com
ISBN 978-2-10-080914-1

Préface

Professeur de sciences désormais retraité de l'enseignement secondaire belge, je fus de ces aventuriers qui se lancèrent à la découverte des premiers micro-ordinateurs *grand public* à la fin des années 1970. Il fallait être un peu fou, à cette époque, pour investir des sommes plutôt rondelettes dans ces machines bricolées, au comportement assez capricieux, dont on fantasmait de tirer tôt ou tard des applications extraordinaires, mais souvent sans trop savoir au juste lesquelles, et encore moins comment on pourrait y arriver.

Il n'était évidemment pas question d'Internet en ce temps-là. Trouver de la documentation était une tâche ardue. Les rares documents que l'on parvenait à trouver (*via* les clubs de radio-amateurs, principalement) traitaient davantage d'électronique que de programmation. Et c'était bien nécessaire, il valait mieux être capable de manier le fer à souder ou de trouver l'un ou l'autre copain technicien dans un laboratoire disposant d'un programmateur d'EPROM [1].

C'est dans ce contexte que je découvris en autodidacte (inutile de dire qu'aucune formation n'était encore organisée à l'époque) mes premiers langages de programmation. Sur le TRS-80 de mes débuts, on disposait seulement d'un Basic sommaire et d'un Assembleur. Il fallait s'accrocher. La mise au point d'un tout petit programme pouvait prendre des heures, et même sa sauvegarde (sur cassette à bande magnétique !) pouvait se révéler problématique. Pas question en tout cas d'imaginer une seule seconde enseigner ce genre de choses à mes jeunes élèves.

Mon activité de programmation en ces années-là se focalisa alors sur le développement de simulations expérimentales. Sur le modèle anglo-saxon des années 1960, je souhaitais centrer mon enseignement scientifique sur la découverte et l'investigation personnelle des élèves, et j'organisais donc un maximum de séances de travaux pratiques. La simulation me permettait d'étendre cette méthodologie à des expérimentations cruciales pour la compréhension de principes fondamentaux (en physique ou en biologie, par exemple), mais irréalisables dans le cadre scolaire ordinaire pour des raisons diverses. Avec un programme de simulation d'expérience bien conçu, l'élève peut se trouver plongé dans une situation de travail très proche de celle d'un laboratoire. Je trouvais particulièrement intéressante, sur le plan pédagogique, l'idée qu'en procédant de la sorte j'instaurais pour l'étudiant un véritable droit à l'erreur : en simulation, il peut en effet décider lui-même sa stratégie expérimentale, procéder par tâtonnements, se tromper, recommencer éventuellement un grand nombre de fois ses tentatives, sans qu'il en résulte un coût excessif en temps ou en ressources matérielles.

Je progressais ainsi dans ma connaissance de la programmation, sans aucune intention de l'enseigner un jour, en m'adaptant au fil des années aux évolutions du matériel et des langages, jusqu'à ce jour de 1998 où l'on me demanda de participer à l'élaboration de cursus pour une nouvelle filière d'enseignement secondaire qui serait centrée sur l'apprentissage de l'informatique.

1. La mémoire EPROM (*Erasable Programmable Read-Only Memory*) est un type de mémoire morte reprogrammable. Pour effectuer cette (re)programmation, il faut en général retirer l'EPROM de son support et la placer dans un appareil dédié à cet effet.

Mon expérience et mes contacts m'avaient entre-temps fait prendre conscience de la problématique de la liberté logicielle. J'étais opportunément en train de découvrir l'une des premières distributions *crédibles* de Linux (l'une des premières Red Hat), et je me suis immédiatement persuadé que si l'on voulait effectivement inculquer une saine compréhension de ce que sont l'informatique et ses enjeux à des étudiants aussi jeunes, on se devait de le faire sur la base de logiciels libres.

L'un des cours à mettre en place devait être une initiation à la programmation. J'avais une certaine expérience en la matière, et c'était pour cela qu'on sollicitait mon avis, mais tous les outils que j'avais utilisés personnellement jusque-là étaient des langages propriétaires (Basic, Delphi, Clarion...), et je ne voulais être le démarcheur d'aucun d'entre eux. C'est donc dans cet esprit que je me suis mis à la recherche de ce que je craignais être la quadrature du cercle : un langage de programmation qui soit à la fois libre, multi-plateformes, polyvalent, assez facile à apprendre, avec lequel il soit possible d'aborder un maximum de concepts, tant sur les paradigmes de programmation que sur les structures de données, qui soit surtout de haut niveau et très lisible (je m'imaginais à l'avance le casse-tête que constituerait pour les professeurs le travail de correction d'un programme mal écrit par un élève à l'esprit tordu, dans un langage proche de la machine et à la syntaxe alambiquée…).

Le miracle a eu lieu : j'ai découvert Python. Ses qualités sont décrites dans les pages qui suivent. Restait le problème de l'enseigner à des jeunes de 16-18 ans. En l'occurrence je souhaitais aussi valider autant que possible la stratégie pédagogique d'apprentissage par investigation libre que j'avais développée pour mes cours de sciences, et aucun cours de programmation satisfaisant à mes critères n'existait à l'époque, du moins en français. L'essentiel de la documentation de Python lui-même n'existait d'ailleurs qu'en anglais (on en était à la version 1.5).

Je me suis donc lancé le défi – encore une fois un peu fou – de rédiger mon propre manuel de cours. La suite est connue : bien conscient de mes limitations d'autodidacte, j'ai tout de suite mis mes notes à la disposition de tout le monde sur l'Internet, et j'ai ainsi pu récolter de nombreux avis et conseils, grâce auxquels le texte s'est amélioré au fil du temps et a fini par paraître aussi en version imprimée, distribuée en librairies.

C'est au cours de cette saga que j'ai eu la chance de faire la connaissance de Bob Cordeau, qui m'a gentiment rendu le service de relire mes 430 pages pour y débusquer coquilles et étourderies. Au cours de cet important travail, il a donc eu tout le loisir de constater tous les défauts de mon texte : imprécisions diverses, structuration fantaisiste, concepts omis ou traités de manière triviale…

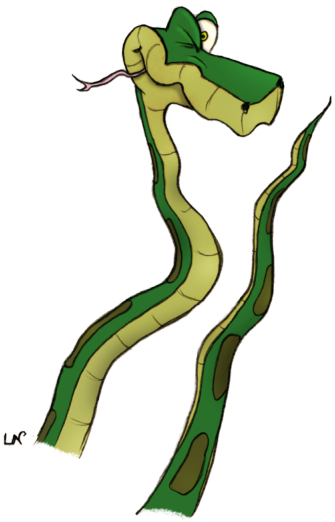
Il ne m'en a rien dit pour ne pas me faire de la peine, mais il s'est courageusement mis à l'ouvrage pour rédiger son propre texte, que vous aurez le plaisir de découvrir dans les pages qui suivent. Là où je m'étais contenté d'une ébauche brouillonne, Bob et Laurent ont réalisé un vrai travail de « pro » : un des meilleurs textes de référence sur ce merveilleux outil qu'est Python.

Bonne lecture, donc.

Gérard Swinnen [1]

1. Auteur d'*Apprendre à programmer avec Python 3*, paru aux éditions Eyrolles, et disponible également en téléchargement libre (`https://inforef.be/swi/python.htm`).

Table des matières



Avant-propos

En se partageant, le savoir ne se divise pas,

il se multiplie.

À qui s'adresse ce livre ?

Issu d'un cours pour les étudiants du département « Mesures physiques » de l'IUT d'Orsay, ce livre s'adresse en premier lieu aux étudiants débutant en programmation, issus des IUT, des BTS, des licences pro et scientifiques, des écoles d'ingénieurs, aux élèves des classes préparatoires scientifiques, aux enseignants du secondaire (et peut-être à leurs élèves les plus motivés) et plus généralement à tout autodidacte désireux d'apprendre Python en tant que premier langage de programmation.

Le langage Python est préconisé pour l'apprentissage de la programmation par l'Éducation nationale pour les classes de lycée. La lecture du thème « Numérique et sciences informatiques » du Conseil supérieur des programmes introduit un vocabulaire et des concepts dont nous avons tenu compte (cf. l'index).

Prenant la programmation à la base, cet ouvrage se veut pédagogique sans cesser d'être pratique. D'une part en fournissant de très nombreux exemples, des exercices corrigés dans le texte et en ligne, et d'autre part en évitant d'être un catalogue exhaustif sur le langage d'apprentissage en offrant d'abord une introduction aux principaux concepts nécessaires à la programmation et en regroupant les éléments plus techniques liés au langage choisi dans les derniers chapitres.

Pour permettre néanmoins d'approfondir ces détails, il propose plusieurs moyens de navigation : une table des matières détaillée en début et, en annexe, des résumés syntaxiques et fonctionnels complets, un glossaire bilingue et un index. De plus, l'ouvrage offre, sur les pages intérieures de la couverture, le mémento Python 3.

Au-delà de l'apprentissage scolaire de la programmation grâce au langage Python, ce livre aborde des aspects souvent négligés, à savoir : le processus de réflexion utilisé dans la phase d'analyse préalable, la façon de passer de l'analyse à l'écriture du programme, de découper proprement celui-ci en fichiers modules réutilisables et ensuite d'utiliser les outils et techniques pour corriger les erreurs.

Nos choix

Cet ouvrage repose sur quelques partis pris :
— la version 3 du langage Python [1] ;
— le choix de logiciels libres [2] : la distribution Python scientifique `Pyzo`, `miniconda3`, `Jupyter Notebook` et des outils *open source* de production de documents (XeLaTeX) ;
— une introduction à la programmation, mais aussi à de nombreux à-côtés souvent oubliés et que l'on ne découvre qu'avec l'expérience.

Les exercices

Parmi les exercices proposés à la fin de chaque chapitre (exercices simples ✓, moins simples ✓✓, voire plus difficiles ✓✓✓) ceux marqués du logo 💡 sont corrigés en fin d'ouvrage. Tous les exercices du livre, ainsi que **125 exercices corrigés supplémentaires** au format *notebook*, accompagnent ce cours sur la page web dédiée à l'ouvrage (https://www.dunod.com/EAN/9782100809141) et sur GitHub (https://github.com/lpointal/appbclp).

Présentation des codes

Un code Python *interprété* sera présenté sous la forme :

```
>>> len("abcde")      # Longueur (nombre de caractères dans la chaîne)
5
>>> "abc" + "defg"    # Concaténation (mise bout à bout de deux chaînes ou plus)
'abcdefg'
>>> "Fi! " * 3        # Répétition (avec * entre une chaîne et un entier)
'Fi! Fi! Fi! '
```

Un *script* complet ou un fragment de script Python sera présenté sous la forme :

```
# coding: utf8
""" Jeu de dés"""

# Programme principal ========================================================
n = int(input("Entrez un entier [2 .. 12] : "))
while not(n >= 2 and n <= 12):
    n = int(input("Entrez un entier [2 .. 12], s.v.p. : "))
s = 0
for i in range(1, 7):
    for j in range(1, 7):
        if i+j == n:
            s += 1
print(f"=> Il y a {s:d} façon(s) de faire {n:d} avec deux dés.")
```

Les commandes textuelles, résultats d'exécution ou fichiers texte seront présentés sous la forme :

```
Une commande ou un fichier "texte".
```

1. Version qui abolit la compatibilité descendante avec les versions antérieures. C'est une grave décision, mûrement réfléchie : « *Un langage qui bouge peu permet une industrie qui bouge beaucoup* » (Bertrand Meyer).
2. Voir la bibliographie (☞ p. 266, § E).

Mise en lumière des éléments importants

Exemple d'encart de définition :

Définition

Une **expression** est une portion de code que l'interpréteur Python peut **évaluer** pour obtenir une **valeur**. Les expressions peuvent être simples ou complexes.

Exemple d'encart de remarque :

Remarque

Une « implémentation » signifie une « mise en œuvre ».

Exemple d'encart de syntaxe :

Syntaxe

Les méthodes spéciales portent des noms prédéfinis, précédés et suivis de deux caractères de soulignement.

Exemple d'encart d'alerte :

Attention

Toutes les instructions au même niveau d'indentation appartiennent au même bloc.

À propos de la deuxième édition

Cette deuxième édition présente plusieurs nouveautés :

- elle utilise Python 3.8 publié en septembre 2019 ;
- à la fin de chaque chapitre le lecteur trouvera :
  - un résumé des nouveaux acquis sous la forme d'un encadré « Ce Qu'il Faut Retenir »,
  - des exercices d'application (pour la plupart corrigés au chapitre 12), incitant le lecteur à mettre en œuvre ces connaissances ;
- le chapitre 2 indique comment trouver une documentation directement dans le *shell* Python ;
- le chapitre 4 montre l'intérêt des tables de *hash* et leur application aux dictionnaires et aux ensembles ;
- le chapitre 6 donne un exemple de définition et d'usage des packages ;
- le chapitre 7, entièrement réécrit, détaille les fichiers binaires, le gestionnaire de contexte `width`, le module `pathlib`. De plus, deux applications sont développées :
  - une introduction au SGBDR et au langage SQL à travers le module `SQLite`,
  - une introduction rapide à Internet et au web à travers une application micro-serveur web ;
- le chapitre 10 décrit les constructions algorithmiques de base : pile, file, liste chaînée, arbre et graphe ;
- le chapitre 11 présente l'écosystème Python scientifique avec notamment `IPython`, `NumPy`, `Pandas` et `matplotlib`. Le domaine du traitement d'image est abordé avec `scikit-image`. La documentation du code est présentée et les tests sont traités avec `pytest`. Enfin nous évoquerons les microcontrôleurs et les objets connectés ;
- le propos de l'annexe consacrée à l'encodage de caractères a été étendu au codage des nombres.

Installation de la distribution Python

Afin de suivre l'évolution de Python, nous avons choisi de détailler cette installation en dehors du présent ouvrage, sur le site `https://perso.limsi.fr/pointal/python:installation:accueil`.

Le lecteur y trouvera tous les indications nécessaires pour installer Python sur les principales plateformes, notamment en utilisant la distribution `miniconda3`, ainsi que l'installation des environnements de développement utilisés. Une page web annexe présente les bases permettant d'utiliser `conda` et des environnements virtuels.

Les programmes en ligne

L'adresse `https://github.com/lpointal/appbclp` regroupe :

- des documentations générales (abrégé et mémento Python, la documentation Jupyter) ;
- les principaux programmes utilisés pour illustrer les concepts dans le livre ;
- tous les exercices du livre et leur correction ;
- les exercices supplémentaires et leur correction.

Remerciements

Les auteurs remercient vivement toutes les personnes qui les ont aidés dans la réalisation de ce projet, notamment :

- Hélène Cordeau pour ses nouvelles illustrations ; les aventures de *Pythoon* enchantent les têtes de paragraphe ! ;
- Jean-Luc Blanc et Brice Martin, des éditions Dunod, pour leur excellent travail critique de relecture ;
- Cécile Trevian pour son aide professionnelle à la traduction du « Zen de Python » (☞ p. 221, annexe A) ;
- la liste GUTenberg [1] pour ses conseils avisés à un LaTeXien reconnaissant.

Nous tenons également à citer :

- Lucile Roussier, dont la confiance a permis à Laurent de sélectionner et ensuite de promouvoir Python comme langage de script au sein du LURE [2] dès 1995 ;
- nos familles respectives pour avoir supporté nos indisponibilités durant la rédaction de cet ouvrage.

Une pensée spéciale pour Stéphane Barthod : son enseignement didactique auprès des étudiants et son engagement envers ses collègues ne seront pas oubliés.

Enfin il faudrait saluer tous les auteurs butinés sur Internet…

Pour joindre les auteurs

Vous pouvez nous adresser vos remarques aux adresses électroniques suivantes :

@ `pycours@kordeo.eu`
@ `laurent.pointal@laposte.net`

1. Le Groupe francophone des Utilisateurs de TeX (`gut@ens.fr`).
2. Laboratoire pour l'Utilisation du Rayonnement Électromagnétique.