

Julien Guillod

Programmation Python par la pratique

2^e édition

DUNOD

Graphisme de couverture : Elizabeth Riba

© Dunod, 2021, 2024
11 rue Paul Bert, 92240 Malakoff
www.dunod.com
ISBN 978-2-10-087515-3



Table des matières

Chapitre 1 Introduction	1
1.1 Remerciements	1
1.2 Pourquoi Python ?	2
1.3 Prérequis	2
1.4 Documentation	3
1.5 Installation	3
1.6 Lancement de Jupyter Lab	6
1.7 Utilisation de Jupyter Lab	6
1.8 Utilisation avancée de Jupyter Lab	8
Chapitre 2 Structures de données	11
Exercice 2.1. Listes	11
Exercice 2.2. Tuples	13
Exercice 2.3. Ensembles	14
Exercice 2.4. Dictionnaires	15
Solution 2.1. Listes	16
Solution 2.2. Tuples	18
Solution 2.3. Ensembles	19
Solution 2.4. Dictionnaires	19
Chapitre 3 Structures homogènes	21
Exercice 3.1. Introduction à Numpy	21
Exercice 3.2. Opérations sur les tableaux	23
Exercice 3.3. Matrice de Vandermonde	24
Exercice 3.4. Indexage de tableaux (!)	24
Solution 3.1. Introduction à Numpy	25
Solution 3.2. Opérations sur les tableaux	26

Solution 3.3. Matrice de Vandermonde	26
Solution 3.4. Indexage de tableaux (!)	28
Chapitre 4 Représentations graphiques	29
Exercice 4.1. Représentations graphiques	29
Exercice 4.2. Chaos déterministe	31
Exercice 4.3. Ensemble de Mandelbrot	34
Exercice 4.4. Représentations graphiques avancées (!)	34
Solution 4.1. Représentations graphiques	36
Solution 4.2. Chaos déterministe	37
Solution 4.3. Ensemble de Mandelbrot	42
Solution 4.4. Représentations graphiques avancées (!)	45
Chapitre 5 Intégration	51
Exercice 5.1. Méthode des rectangles	51
Exercice 5.2. Méthode des trapèzes	52
Exercice 5.3. Méthode de Monte-Carlo	53
Exercice 5.4. Méthode de Simpson (!)	54
Exercice 5.5. Intégration avec Scipy (!!)	54
Solution 5.1. Méthode des rectangles	55
Solution 5.2. Méthode des trapèzes	59
Solution 5.3. Méthode de Monte-Carlo	62
Solution 5.4. Méthode de Simpson (!)	66
Solution 5.5. Intégration avec Scipy (!!)	69
Chapitre 6 Algèbre	71
Exercice 6.1. Décomposition LU	71
Exercice 6.2. Méthode de la puissance itérée	72
Exercice 6.3. Exponentielle de matrices	73
Exercice 6.4. Groupes de permutations	74
Solution 6.1. Décomposition LU	75
Solution 6.2. Méthode de la puissance itérée	78

Solution 6.3. Exponentielle de matrices	80
Solution 6.4. Groupes de permutations	84
Chapitre 7 Théorie des graphes	89
Exercice 7.1. Graphes comme dictionnaires	89
Exercice 7.2. Triangles dans un graphe	90
Exercice 7.3. Module NetworkX (!!)	91
Solution 7.1. Graphes comme dictionnaires	91
Solution 7.2. Triangles dans un graphe	94
Solution 7.3. Module NetworkX (!!)	96
Chapitre 8 Calcul symbolique	101
Exercice 8.1. Introduction à Sympy	101
Exercice 8.2. Applications	103
Exercice 8.3. Conjecture due à Euler	104
Exercice 8.4. Fonction pathologique	105
Exercice 8.5. Fonction de Green du laplacien (!)	106
Solution 8.1. Introduction à Sympy	107
Solution 8.2. Applications	108
Solution 8.3. Conjecture due à Euler	110
Solution 8.4. Fonction pathologique	113
Solution 8.5. Fonction de Green du laplacien (!)	115
Chapitre 9 Zéro de fonctions	121
Exercice 9.1. Méthode de Newton en une dimension	121
Exercice 9.2. Méthode de Newton en plusieurs dimensions	122
Exercice 9.3. Attracteur de la méthode de Newton	123
Exercice 9.4. Équation différentielle non linéaire (!!)	124
Solution 9.1. Méthode de Newton en une dimension	125
Solution 9.2. Méthode de Newton en plusieurs dimensions	126

Solution 9.3. Attracteur de la méthode de Newton	127
Solution 9.4. Équation différentielle non linéaire (!!)	129
Chapitre 10 Probabilités et statistiques	133
Exercice 10.1. Série harmonique de signe aléatoire	133
Exercice 10.2. Ruine du joueur	134
Exercice 10.3. Urnes de Polya	134
Exercice 10.4. Théorème central limite	136
Exercice 10.5. Génération aléatoire de vecteurs unitaires	138
Exercice 10.6. Percolation (!!)	139
Solution 10.1. Série harmonique de signe aléatoire	141
Solution 10.2. Ruine du joueur	142
Solution 10.3. Urnes de Polya	146
Solution 10.4. Théorème central limite	149
Solution 10.5. Génération aléatoire de vecteurs unitaires	152
Solution 10.6. Percolation (!!)	154
Chapitre 11 Équations différentielles	159
Exercice 11.1. Méthodes d'Euler	159
Exercice 11.2. Méthodes de Runge-Kutta	160
Exercice 11.3. Mouvement d'une planète	162
Exercice 11.4. Attracteur de Lorenz	163
Exercice 11.5. Équation des ondes cubique (!!)	163
Exercice 11.6. Méthodes de Bogacki-Shampine (!!!)	164
Solution 11.1. Méthodes d'Euler	165
Solution 11.2. Méthodes de Runge-Kutta	169
Solution 11.3. Mouvement d'une planète	172
Solution 11.4. Attracteur de Lorenz	175
Solution 11.5. Équation des ondes cubique (!!)	178

Chapitre 12 Science des données	181
Exercice 12.1. Introduction à Pandas	181
Exercice 12.2. Loi de Benford	184
Exercice 12.3. Méthode des moindres carrés	185
Exercice 12.4. Reconnaissance de chiffres manuscrits	185
Exercice 12.5. Différentiation automatique (!)	187
Exercice 12.6. Réseau de neurones (!)	190
Solution 12.1. Introduction à Pandas	191
Solution 12.2. Loi de Benford	193
Solution 12.3. Méthode des moindres carrés	199
Solution 12.4. Reconnaissance de chiffres manuscrits	201
Solution 12.5. Différentiation automatique (!)	202
Solution 12.6. Réseau de neurones (!)	206
Chapitre 13 Cryptographie	211
Exercice 13.1. Code de Vigenère	211
Exercice 13.2. Casser le code de Vigenère (!)	212
Exercice 13.3. Générer des nombres premiers	213
Exercice 13.4. Générer des nombres pseudo-premiers	213
Exercice 13.5. Chiffrement RSA	214
Exercice 13.6. Casser le chiffrement RSA (!!!)	216
Solution 13.1. Code de Vigenère	216
Solution 13.2. Casser le code de Vigenère (!)	217
Solution 13.3. Générer des nombres premiers	220
Solution 13.4. Générer des nombres pseudo-premiers	221
Solution 13.5. Chiffrement RSA	223
Index	227

Introduction

Python est un langage de programmation phare dans le monde scientifique. Il est parfaitement adapté pour programmer des problèmes mathématiques. Cet ouvrage propose de se focaliser sur l'utilisation pratique du langage Python dans différents domaines des mathématiques : les suites, l'algèbre linéaire, l'intégration, la théorie des graphes, la recherche de zéros de fonctions, les probabilités, les statistiques, les équations différentielles, le calcul symbolique, et la théorie des nombres. À travers 40 exercices de difficulté croissante, et corrigés en détails, il permet d'avoir une bonne vision d'ensemble des possibilités d'utilisation de la programmation dans les mathématiques et d'être à même de résoudre des problèmes mathématiques complexes.

Il n'est pas nécessaire de faire les exercices dans l'ordre proposé, même si certains exercices font parfois appel à des notions vues dans des exercices précédents. Les exercices plus difficiles sont indiqués par des points d'exclamation :

- ! : plus long ou plus difficile ;
- !! : passablement long et complexe ;
- !!! : défi proposé sans correction.

Le séparateur décimal utilisé dans cet ouvrage est le point et non pas la virgule, afin de se conformer avec l'usage international employé par Python et éviter les confusions. Ces exercices servent de base aux travaux pratiques donnés à Sorbonne Université dans le cadre de la licence de mathématiques.

L'ensemble des codes sources de l'ouvrage est disponible en ligne à l'adresse : <https://python.guillod.org/>. Ce site est mis à jour régulièrement, aussi il se peut qu'il diffère du présent ouvrage dans le futur.

1.1 Remerciements

Merci à Marie Postel et Nicolas Lantos pour leurs relectures attentives de la première version de ce recueil et pour les nombreuses corrections et suggestions. Pour cette seconde édition, un merci particulier à Johann Faouzi et Louis Thiry.

Merci également aux membres des équipes pédagogiques de Sorbonne Université ayant utilisé ces exercices pour leurs retours et contributions : Mathieu Barré, Constantin Bône, Jules Bonnard, Cédric Boutillier, Thibault Cimic, Jeanne Decayeux, Cécile Della Valle, Guillaume Duboc, Jean-Jil Duchamps, Jean-Merwan Godon, Jean-Merwan Godon, Elise Grosjean, Cindy Guichard, Nicolas Lantos, Mathieu Mari, David Michel, Leo Miolane, Anouk Nicolopoulos, Arnaud Padrol, Diane Peurichard, Marie Postel, Xavier Poulot-Cazajous, Alexandre Rege, Othmane Safsafi, Emmanuel Schertzer, Agustín

Somacal, Didier Smets, Robin Strudel, Gauthier Tallec, Nicolas Thomas, Paul Vernhet, Jules Vidal et Raphaël Zanella.

Finalement merci aux étudiantes et aux étudiants ayant planché sur ces exercices pour leurs retours constructifs qui ont contribué à l'amélioration du présent recueil.

La lectrice attentive ou le lecteur attentif est remercié-e par avance pour tout signalement de coquilles ou autres.

1.2 Pourquoi Python ?

Python est un langage généraliste de programmation interprété qui a la particularité d'être très lisible et pragmatique. Il dispose d'une très grosse base de modules externes, notamment scientifiques, qui le rend particulièrement attractif pour programmer des problèmes mathématiques. Le fait que Python soit un langage interprété le rend plus lent que les langages compilés, il assure en revanche une grande rapidité de développement qui permet à l'humain de travailler un peu moins tandis que l'ordinateur devra travailler un peu plus. Cette particularité fait que Python est devenu l'un des principaux langages de programmation utilisés par les scientifiques.

1.3 Prérequis

Cet ouvrage n'a pas pour but premier d'exposer la syntaxe et les principes du langage Python, aussi les prérequis sont-ils d'en connaître les bases. Il existe de nombreuses ressources pour se mettre à jour en cas de besoin, par exemple :

- le cours en ligne *Python 3 : des fondamentaux aux concepts avancés du langage* d'Arnaud Legout et Thierry Parmentelat à suivre sur le site de France Université Numérique (<https://www.fun-mooc.fr/fr/cours/python-3-des-fondamentaux-aux-concepts-avances-du-langage/>). Les vidéos sont également disponibles sur YouTube (https://www.youtube.com/channel/UCI1UBOXnXjxdjmL_atU53kA)
- l'ouvrage *Programmation en Python pour les sciences de la vie* de Patrick Fuchs et Pierre Poulain (<https://dunod.com/EAN/9782100796021>)
- le cours *1IN001* dispensé à Sorbonne Université (<https://www-licence.ufr-info-p6.jussieu.fr/lmd/licence/2021/ue/LU1IN001-2021oct/>)

Par ailleurs la réalisation des exercices demande d'avoir accès à un ordinateur ou un service en ligne disposant de Python 3.6 (ou plus récent) complété par les modules suivants : Numpy, Scipy, Sympy, Matplotlib, Numba, NetworkX et Pandas. L'emploi d'un éditeur de code permettant l'écriture en Python est aussi vivement conseillé. Il est ici suggéré d'utiliser Jupyter Lab, qui permet à la fois l'écriture des notebooks interactifs et des scripts et également l'ajout de ses propres solutions en dessous des énoncés, ce qui est très pratique. Il n'est pas indispensable d'utiliser Jupyter Lab, d'autres environnements sont aussi adaptés, notamment Spyder ou Jupyter Notebook.

Les sections suivantes décrivent comment installer et lancer l'environnement Python ou l'utiliser en ligne sans installation.

1.4 Documentation

Il n'est généralement pas utile (ni souhaitable) de connaître toutes les fonctions et subtilités du langage Python lors d'une utilisation occasionnelle. Par contre il est indispensable de savoir utiliser la documentation de manière efficace. La documentation officielle est disponible à l'adresse <https://docs.python.org/>. La langue et la version peuvent être sélectionnées en haut à gauche. Il est fortement conseillé de regarder comment la documentation est écrite et d'apprendre à l'utiliser.

1.5 Installation

Les personnes ne pouvant ou ne voulant pas installer Python peuvent directement se rendre à la section 1.6 pour des alternatives disponibles en ligne sans installation.

Il existe essentiellement quatre façons d'installer Python et les modules requis pour réaliser les exercices :

- **Anaconda** est une distribution Python complète, c'est-à-dire qu'elle installe directement une très grande quantité de modules (beaucoup plus que nécessaire pour faire les exercices suivants). L'avantage de cette installation est qu'elle est très simple ; son désavantage est qu'elle prend beaucoup d'espace disque. C'est la méthode à privilégier si vous êtes sous Windows ou MacOS et que vous n'avez pas de problème d'espace disque.
- **Miniconda** est une version légère d'Anaconda, qui installe par défaut uniquement la base. L'avantage est qu'elle prend peu d'espace disque, mais elle requiert une action supplémentaire pour installer les modules requis pour faire les exercices. C'est la méthode à privilégier si vous êtes sous Windows ou MacOS et que vous avez peu d'espace disque disponible.
- **Dépôts Linux** : la plupart des distributions Linux permettent d'installer Python et les modules de base directement à partir des dépôts de paquets qui les accompagnent. C'est la méthode privilégiée sous Linux.
- **Pip** est un gestionnaire de paquets pour Python. C'est la méthode à privilégier pour ajouter un module si Python est déjà installé par votre système d'exploitation, et que ce module n'est pas inclus dans les paquets de votre distribution. Cette méthode permet une gestion plus fine et avancée des modules installés que ce qui est proposé avec les méthodes précédentes.

Installation avec Anaconda : La façon la plus simple d'installer Python 3 et toutes les dépendances nécessaires sous Windows et MacOS est d'installer Anaconda. Le désavantage d'Anaconda est que son installation prend beaucoup d'espace disque car