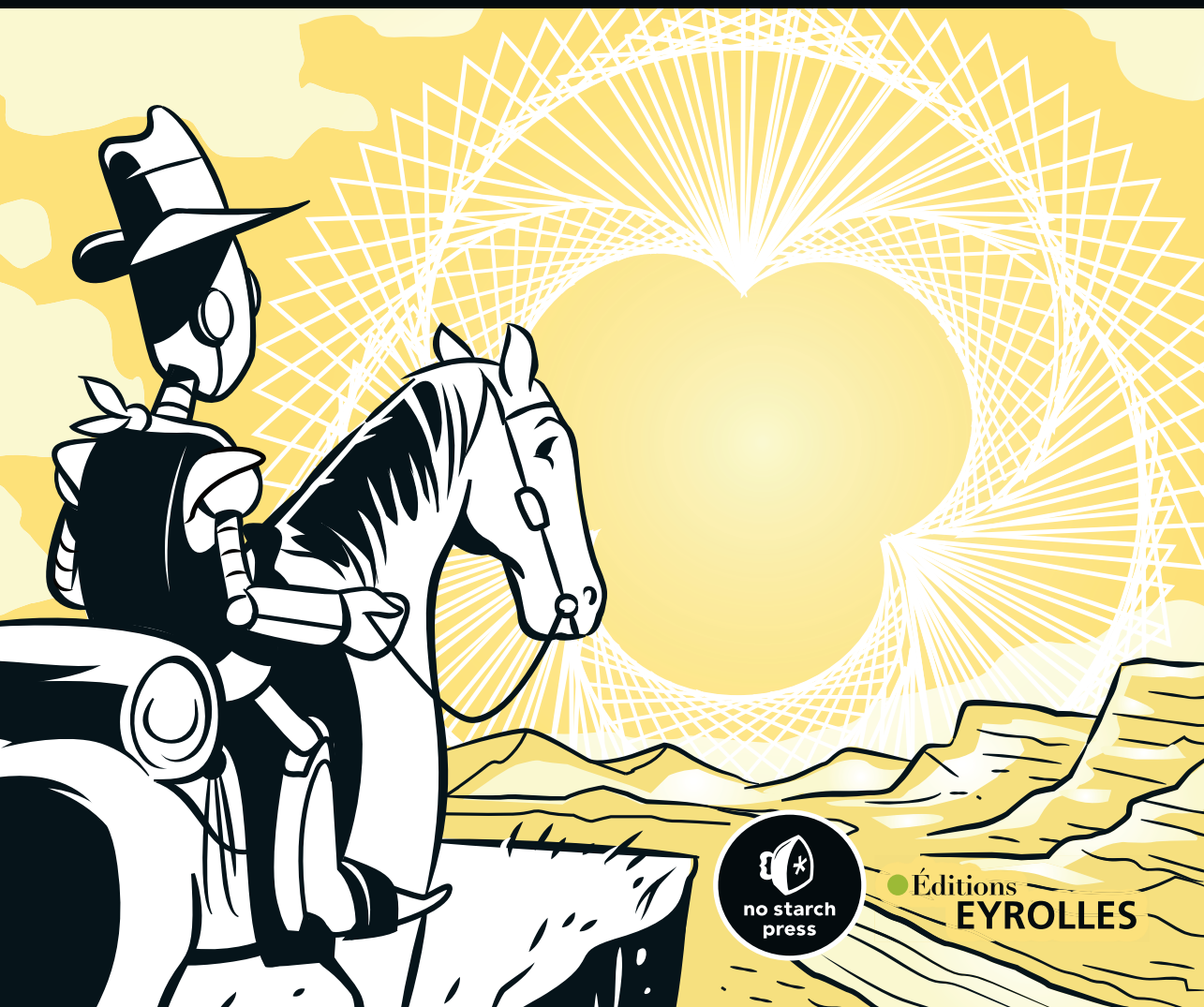
PETER FARRELL

À LA CONQUÊTE DES MATHS AVEC PYTHON

DÈS 14 ANS



no starch press

Éditions EYROLLES

LES MATHS DEVIENNENT LUDIQUES AVEC PYTHON !

COUVRE LE PROGRAMME DU LYCÉE

Cet ouvrage d'une grande pédagogie exploite la puissance de la programmation informatique pour rendre les mathématiques digestes et amusantes. Avec l'aide du langage Python, vous découvrirez ainsi comment visualiser les solutions de différents problèmes, dans des domaines aussi variés que l'algèbre, la trigonométrie ou les matrices.

Sous l'angle passionnant des mathématiques, vous manipulerez les principaux concepts de la programmation Python (boucles, variables, classes...) pour résoudre des équations, rechercher des racines carrées, dessiner et manipuler des formes, ou encore créer des ondes sinusoïdales. Illustré par de nombreux exemples et exercices dont les solutions sont disponibles en ligne, ce livre vous apprendra en outre à coder vos propres programmes et vous montrera à quel point les mathématiques peuvent être pétillantes !

Ce livre vous expliquera notamment à :

- dessiner et transformer des graphiques 2D et 3D à l'aide de matrices
- créer des motifs colorés comme les ensembles Mandelbrot et Julia avec des nombres complexes
- utiliser la récursivité pour produire des fractales
- générer des moutons virtuels qui se multiplient de manière autonome

À QUI S'ADRESSE CE LIVRE ?

- À tous les lycéens souhaitant progresser en maths et en Python
- Aux enseignants, associations, parents

À PROPOS DE L'AUTEUR

Professeur de mathématiques et d'informatique, **Peter Farrell** est passionné par l'apprentissage des sciences grâce à la technologie.

no starch press

www.editions-eyrolles.com

PETER FARRELL

À LA CONQUÊTE DES MATHS AVEC PYTHON

no starch press

Éditions EYROLLES

Éditions Eyrolles
61, bd Saint-Germain
75240 Paris Cedex 05
www.editions-eyrolles.com

Adapté de l'anglais par Paul Bizouard.
Mise en pages : Sandrine Escobar

Figure 10-2 : © Acadac / Avsa (CC-BY-SA-3.0)
Figure 11-19 : © Fabienne Serrière (https://knityak.com)

ISBN : 978-2-212-67957-1

Ce livre est dédié à tous mes étudiants, qui m'ont tant appris.

REMERCIEMENTS

J'aimerais remercier Don « the Mathman » Cohen pour m'avoir montré à quel point l'apprentissage des mathématiques peut être amusant et stimulant ; Seymour Papert pour avoir prouvé que l'informatique a sa place dans les cours de mathématiques ; Mark Miller pour m'avoir donné une chance de transformer mes idées en actions ; Hansel Lynn et Wayne Teng de theCoderSchool, qui me laissent prendre plaisir à coder avec mes élèves ; et Ken Hawthorn pour mettre à disposition mes projets à son école. Je remercie également mes éditeurs de No Starch, Annie Choi, Liz Chadwick et Meg Sneeringer pour leur aide précieuse dans l'écriture de ce livre, et Paddy Gaunt, dont la marque est visible tout au long de cet ouvrage. Ce livre n'existerait pas sans vous tous. Merci à tous ceux qui m'ont dit non — vous m'avez donné l'énergie de continuer. Enfin, je voudrais remercier Lucy qui a toujours cru en moi.

TABLE DES MATIÈRES

PARTIE 2
CHEVAUCHEZ EN TERRITOIRE MATHÉMATIQUE

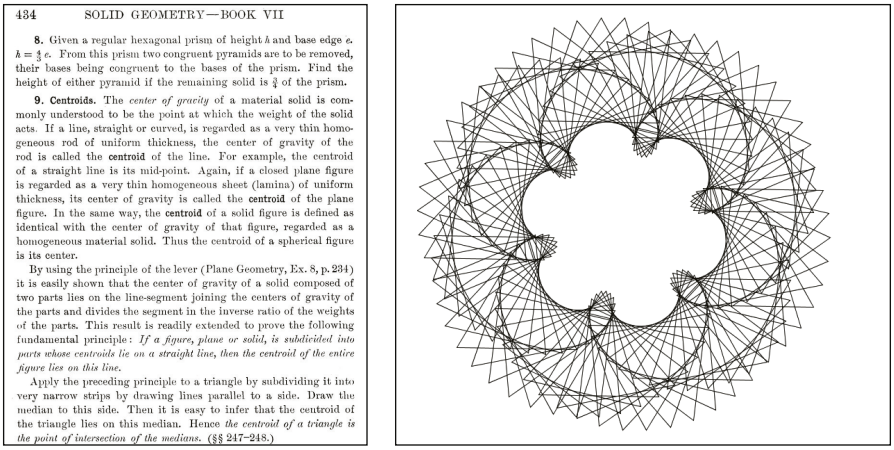
4
TRANSFORMER ET MÉMORISER DES NOMBRES GRÂCE À L'ALGÈBRE 63

5
TRANSFORMER DES FORMES AVEC LA GÉOMÉTRIE 91

PARTIE 3
TRACEZ VOTRE PROPRE PISTE

INTRODUCTION

Laquelle des approches illustrées sur la figure I-1 préféreriez-vous ? À votre gauche, vous avez l'exemple d'une approche d'enseignement traditionnelle des mathématiques, incluant des définitions, des propositions et des preuves. Cette méthode demande beaucoup de lecture et implique des symboles étranges. Vous auriez du mal à deviner qu'il y est question de figures géométriques. Et pourtant, ce texte explique comment trouver le *barycentre*, ou centre, d'un triangle. Mais les approches traditionnelles telles que celle-ci ne nous expliquent jamais *pourquoi* il peut être intéressant de trouver le centre d'un triangle.



434 SOLID GEOMETRY—BOOK VII

8. Given a regular hexagonal prism of height h and base edge e. $h = \frac{4}{5}e$. From this prism two congruent pyramids are to be removed, their bases being congruent to the bases of the prism. Find the height of either pyramid if the remaining solid is $\frac{3}{4}$ of the prism.

9. Centroids. The *center of gravity* of a material solid is commonly understood to be the point at which the weight of the solid acts. If a line, straight or curved, is regarded as a very thin homogeneous rod of uniform thickness, the center of gravity of the rod is called the **centroid** of the line. For example, the centroid of a straight line is its mid-point. Again, if a closed plane figure is regarded as a very thin homogeneous sheet (lamina) of uniform thickness, its center of gravity is called the **centroid** of the plane figure. In the same way, the **centroid** of a solid figure is defined as identical with the center of gravity of that figure, regarded as a homogeneous material solid. Thus the centroid of a spherical figure is its center.

By using the principle of the lever (Plane Geometry, Ex. 8, p. 234) it is easily shown that the center of gravity of a solid composed of two parts lies on the line-segment joining the centers of gravity of the parts and divides the segment in the inverse ratio of the weights of the parts. This result is readily extended to prove the following fundamental principle: *If a figure, plane or solid, is subdivided into parts whose centroids lie on a straight line, then the centroid of the entire figure lies on this line.*

Apply the preceding principle to a triangle by subdividing it into very narrow strips by drawing lines parallel to a side. Draw the median to this side. Then it is easy to infer that the centroid of the triangle lies on this median. Hence *the centroid of a triangle is the point of intersection of the medians.* (§§ 247–248.)

Figure I-1. Deux approches d'enseignements à propos du barycentre

À droite du texte, vous pouvez voir une figure dynamique contenant une centaine de triangles en rotation. Elle est le résultat final d'un projet de programmation difficile mais stimulant pour lequel il est nécessaire de trouver le barycentre d'un triangle afin de faire tourner les triangles de manière correcte (et que ce soit joli). Dans beaucoup de cas, créer des graphiques intéressants est impossible sans connaître la théorie mathématique en jeu. Vous verrez au cours de ce livre que le fait de connaître quelques outils mathématiques, tels que le barycentre, nous aidera beaucoup à dessiner nos figures. Un élève qui connaît un peu plus profondément les mathématiques et qui sait comment dessiner des graphiques intéressants sera plus enclin à affiner ses connaissances géométriques par lui-même et à utiliser des racines carrées ou fonctions trigonométriques, ci et là, en tant qu'outils. À l'inverse, un élève qui ne comprendrait pas la raison de cet apprentissage et se contenterait de résoudre des exercices issus d'un livre scolaire serait probablement moins motivé à en apprendre davantage sur la géométrie.

Durant mes huit années d'expérience en tant que professeur de mathématiques et trois années en tant que professeur d'informatique, j'ai rencontré beaucoup d'étudiants qui préféraient une approche visuelle à l'approche académique. Lorsqu'on s'applique à créer quelque chose d'intéressant nécessitant l'utilisation des mathématiques, on en vient à réaliser qu'elles ne se résument pas seulement à suivre une méthode pour résoudre des équations. On comprend alors qu'explorer les mathématiques *via* l'informatique permet de découvrir une multitude de façons différentes de résoudre des problèmes captivants tout en trébuchant sur des erreurs inattendues au cours du processus et autant d'opportunités de s'améliorer.

Ceci est la différence entre les mathématiques enseignées à l'école et les vraies mathématiques.

LE PROBLÈME DES MATHÉMATIQUES SCOLAIRES

Qu'est-ce que je veux dire lorsque j'écris « mathématiques scolaires », exactement ? Aux États-Unis, durant les années 1860, les mathématiques scolaires consistaient à vous préparer pour des tâches correspondant à un travail de bureau comme additionner des tableaux de nombres à la main. Aujourd'hui, le travail est différent et on doit donc se préparer différemment.

On apprend plus rapidement par la pratique. Mais le système scolaire ne prend que rarement ce fait en compte, ce qui favorise un apprentissage passif chez les élèves. « La pratique » en cours de français ou d'histoire, par exemple, signifierait écrire des dissertations ou faire des présentations, et dans les domaines scientifiques, faire des expériences. Mais qu'en est-il pour les mathématiques ? Jusqu'à récemment, « pratiquer » en mathématiques signifiait résoudre des équations, factoriser des polynômes ou dessiner des fonctions. Mais désormais, les ordinateurs peuvent effectuer ces tâches à notre place ; et ces exercices sont de moins en moins appropriés.

Le fait d'apprendre comment résoudre, factoriser ou tracer automatiquement une fonction n'est pas le but final. Dès lors qu'un étudiant sait comment automatiser une méthode, il a la possibilité d'en découvrir davantage sur le sujet ; ce qu'il n'aurait pas pu faire auparavant.

La figure I-2 illustre un problème mathématique typique des manuels scolaires qui consiste à définir une fonction `f(x)`, et à l'évaluer à différents points.

Exercice 1-22 : Calculer la valeur de la fonction aux différents points indiqués.

$$f(x) = \sqrt{x+3} - x + 1$$

$$g(t) = t^2 - 1$$

$$h(x) = x^2 + \frac{1}{x} + 2$$

1. *f(0)*
2. *f(1)*
3. *f*($\sqrt{2}$)
4. *f*($\sqrt{2} - 1$)

Figure I-2. Une approche traditionnelle pour enseigner les fonctions

Le même format d'exercice est utilisé pour 18 autres questions ! Ce type de problème est trivial lorsqu'on utilise un langage de programmation tel que Python. Il suffirait simplement de définir la fonction `f(x)`, puis d'y insérer les valeurs depuis une liste comme suit.

```
import math

def f(x):

    return math.sqrt(x + 3) - x + 1
#liste des valeurs à évaluer
for x in [0,1,math.sqrt(2),math.sqrt(2)-1]:
    print("f({:.3f}) = {:.3f}".format(x,f(x)))
```

La dernière ligne sert simplement à rendre l'affichage du résultat plus esthétique en arrondissant les solutions à trois décimales après la virgule :

```
f(0.000) = 2.732
f(1.000) = 2.000
f(1.414) = 1.687
(0.414) = 2.434
```

Dans les langages de programmation comme Python, JavaScript et Java, les fonctions sont des outils d'une importance capitale pour transformer les nombres et d'autres objets – et parfois même d'autres fonctions ! Lorsque vous utilisez Python, il est possible de nommer des fonctions afin de simplifier la compréhension du code. Par exemple, vous pouvez nommer la fonction qui calcule l'aire d'un rectangle en l'appelant `calculateArea()` :

```
def calculateArea(width,height):
```

Un manuel de mathématiques publié au XXI^{e} siècle, plusieurs décennies après que Benoît Mandelbrot a pour la première fois généré une fractale depuis son ordinateur alors qu'il travaillait pour IBM, montre une illustration de l'ensemble de Mandelbrot, mettant en avant cette découverte. Le manuel décrit l'ensemble de Mandelbrot (figure I-3), comme « un objet mathématique fascinant issu des nombres complexes. Son contour magnifique illustre un comportement chaotique. »

Le livre conduit ensuite le lecteur au travers d'une méticuleuse « exploration », lui montrant comment effectuer la transformation d'un point dans le plan complexe. Cependant, il n'est expliqué comment le faire que depuis une calculatrice, limitant à deux le nombre de points transformés (pour sept itérations) en raison de la faible puissance de calcul de la machine. Seulement deux points.

Dans ce livre, vous apprendrez comment faire de même avec Python, ce qui vous permettra de transformer des centaines de milliers de points automatiquement, et même de recréer l'ensemble de Mandelbrot illustré à la figure I-3 !

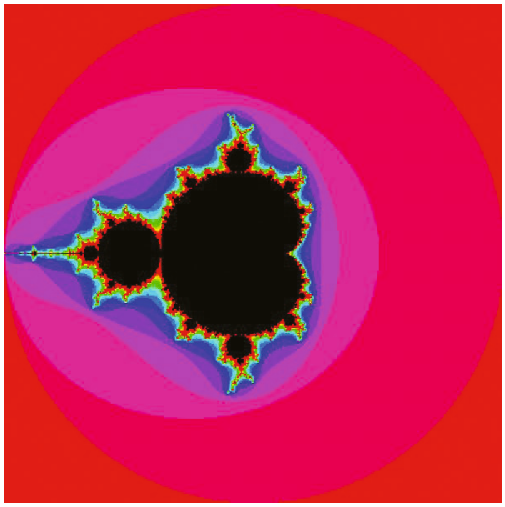


Figure I-3. L'ensemble de Mandelbrot

À PROPOS DE CE LIVRE

Le but de cet ouvrage est d'utiliser l'outil informatique afin de rendre l'apprentissage des mathématiques plus amusant et pertinent. Vous tracerez des graphiques montrant tous les résultats possibles d'une fonction. Vous dessinerez des figures dynamiques et interactives. Vous apprendrez même à programmer un écosystème composé de moutons qui se déplacent et broutent de l'herbe ou encore des organismes essayant de trouver le chemin le plus rapide entre différentes destinations pendant que vous les regarderez faire !

Vous ferez tout cela en utilisant Python et Processing afin d'aller au-delà du niveau de vos cours de mathématiques. L'objet de ce livre n'est pas de supprimer l'utilisation des mathématiques, mais plutôt d'apprendre à utiliser des outils plus excitants et au goût du jour afin de les appliquer au service de la créativité et de développer de vraies compétences informatiques tout en découvrant les liens entre les mathématiques, l'art, les sciences et la technologie. Processing nous apportera les graphiques, formes, mouvements et couleurs alors que nous utiliserons Python pour tout ce qui relève des calculs et méthodes en amont.

Pour chacun des projets de ce livre, vous serez amené à écrire le code par vous-même, du début à la fin, étape par étape depuis un fichier vierge jusqu'au programme final. À force de faire des erreurs dans vos programmes et de les déboguer, vous apprendrez de mieux en mieux comment utiliser chaque bloc de code.

À QUI EST-IL DESTINÉ ?

Ce livre est destiné à quiconque désire en apprendre plus sur les mathématiques et voudrait utiliser des outils plus modernes pour la compréhension de certains domaines des mathématiques tels que la trigonométrie ou l'algèbre. Si vous vous intéressez déjà à Python, vous pouvez utiliser ce livre pour appliquer vos compétences informatiques en cours d'apprentissage au travers de projets intéressants comme les automates cellulaires, les algorithmes génétiques et l'art numérique.

Les enseignants peuvent s'inspirer des projets présentés dans ce livre afin de créer des exercices intéressants pour leurs étudiants ou pour rendre les mathématiques plus abordables et pertinentes. Quelle meilleure manière d'enseigner les matrices que d'insérer une liste de points dans une matrice et de les utiliser pour dessiner une figure en 3D ? Python permet cela et bien plus encore.

QUE CONTIENT-IL ?

Ce livre commence par trois chapitres couvrant les concepts de base de Python, nécessaires à une exploration plus poussée des mathématiques. Les neufs chapitres suivants explorent des concepts et problèmes mathématiques pouvant être visualisés et résolus à l'aide de Python et Processing. Vous pouvez vous essayer aux exercices parsemés tout au long de ce livre afin d'appliquer ce que vous venez d'apprendre et vous mettre à l'épreuve.

- **Chapitre 1 – Tracer des polygones avec le module Turtle** vous apprend les concepts de base de la programmation tels que les boucles, les variables et les fonctions à l'aide du module intégré nommé Turtle.
- **Chapitre 2 – Rendre l'arithmétique amusante à l'aide des listes et des boucles** vous permet de découvrir plus profondément les concepts de programmation tels que les listes et les booléens.
- **Chapitre 3 – Tenter de deviner et vérifier grâce aux conditions** vous permet d'appliquer vos nouvelles compétences en programmation Python à des problèmes comme factoriser des nombres et créer un jeu de devinettes interactif.
- **Chapitre 4 – Transformer et mémoriser des nombres grâce à l'algèbre** vous apprend comment résoudre des équations du troisième degré numériquement, puis de manière graphique.
- **Chapitre 5 – Transformer des formes avec la géométrie** vous montre comment créer des formes puis les multiplier, les pivoter ou les étendre sur tout votre écran.

- **Chapitre 6 – Créer des oscillations avec la trigonométrie** vous apprend, au-delà des angles droits, à créer des ondes et autres formes oscillantes.
- **Chapitre 7 – Utiliser les nombres complexes** vous apprend comment utiliser les nombres complexes pour déplacer des points sur votre écran et créer des figures telles que celle de l'ensemble de Mandelbrot.
- **Chapitre 8 – Utiliser les matrices pour les graphiques informatiques et les systèmes d'équations** vous emmène dans la troisième dimension où vous déplacerez et ferez pivoter des formes en 3D et où vous résoudrez d'immenses systèmes d'équations en un seul programme.
- **Chapitre 9 – Créer des objets à l'aide des classes** vous montre comment créer un objet, ou autant que votre ordinateur pourra en générer, au travers d'une bataille pour la survie entre des moutons et de l'herbe succulente.
- **Chapitre 10 – Créer des fractales en utilisant la récursion** vous montre comment la récursion peut être utilisée comme une toute nouvelle méthode pour mesurer les distances et créer des figures inattendues.
- **Chapitre 11 – Programmer des automates cellulaires** vous apprend à générer et programmer des automates cellulaires qui suivent vos instructions.
- **Chapitre 12 – Résoudre des problèmes à l'aide des algorithmes génétiques** vous montre comment apprivoiser la théorie de la sélection naturelle pour résoudre des problèmes sans laquelle la résolution prendrait des millions d'années.

Le code des programmes des chapitres et les solutions des exercices sont disponibles à l'adresse *https://www.editions-eyrolles.com/dl/0067957*.

TÉLÉCHARGEMENT ET INSTALLATION DE PYTHON

La façon la plus simple de démarrer est d'utiliser le logiciel de distribution Python 3. Python est devenu l'un des langages de programmation les plus populaires du monde. Il est utilisé pour créer des sites web tels que Google, YouTube ou Instagram et des chercheurs du monde entier s'en servent pour leurs analyses dans des domaines variés, depuis l'astronomie jusqu'à la zoologie. Au moment où ces lignes sont composées, la version la plus récente est Python 3.8.2. Vous pouvez la télécharger gratuitement à l'adresse *https://www.python.org/downloads/* (figure I-4).

Figure I-4. Le site web officiel de la Python Software Foundation

Choisissez la version correspondant à votre système d'exploitation. Le site a détecté automatiquement que j'utilise Windows. Cliquez sur le fichier lorsque le téléchargement est terminé (figure I-5).

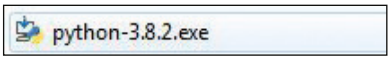


Figure I-5. Cliquez sur le fichier téléchargé pour démarrer l'installation.

Suivez les instructions et sélectionnez toujours l'option par défaut. L'installation peut prendre quelques minutes. Après cela, cherchez « IDLE » dans vos programmes. C'est l'IDE de Python, ou *Integrated Development Environment,* qui vous servira à écrire votre code. Pourquoi « IDLE » ? Le langage Python tient son nom de la troupe de comédiens Monty Python et l'un de ses membres s'appelle Eric Idle.

DÉMARRAGE DE L'IDLE

Trouvez l'IDLE dans vos programmes et démarrez-le.

Figure I-6. Ouvrir l'IDLE dans Windows

Une fenêtre appelée « shell » va apparaître. Vous pouvez l'utiliser comme environnement de codage interactif mais vous préférerez souvent pouvoir sauvegarder votre code. Cliquez sur **File>New File** ou appuyez sur les touches **CTRL+N** pour ouvrir un nouveau fichier (figure I-7).

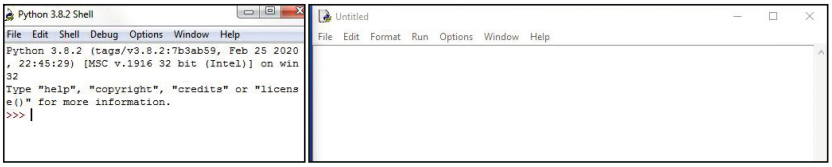


Figure I-7. Le shell interactif Python (à gauche) et un nouveau fichier (à droite), prêt à coder !

C'est ici que vous allez écrire votre code. Nous aurons aussi besoin du logiciel Processing alors autant vous montrer dès maintenant comment le télécharger et l'installer.

INSTALLATION DE PROCESSING

Il est possible de faire énormément de choses avec Python et nous utiliserons très souvent l'IDLE. Mais pour les codes qui demandent plus de compétences graphiques, Processing sera notre outil. Processing est une bibliothèque utilisée par des codeurs professionnels et des artistes pour créer des graphiques ou des œuvres interactives et dynamiques.

Allez à l'adresse *https://www.processing.org/download/* et choisissez votre système d'exploitation comme montré sur la figure I-8.

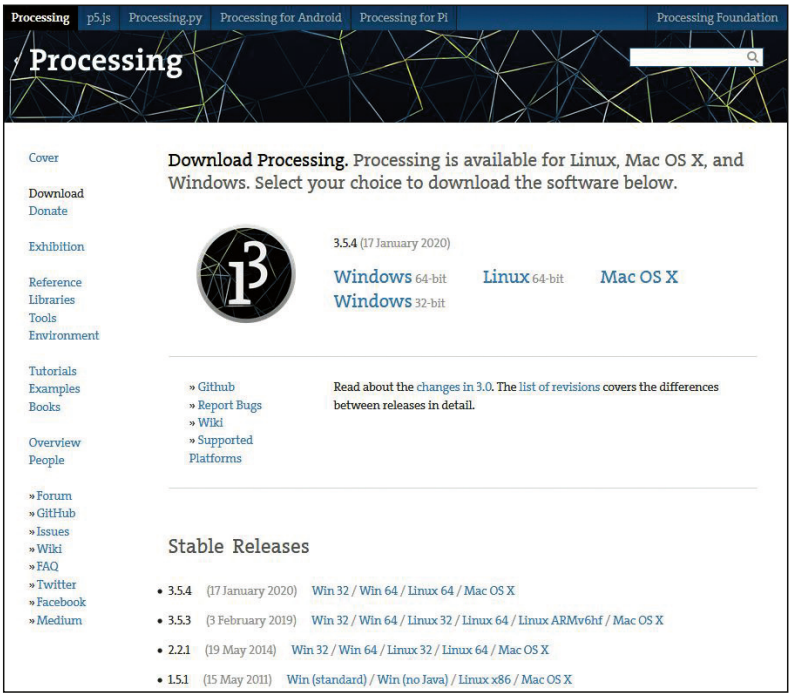


Figure I-8. Site web de Processing

Téléchargez l'installeur en double-cliquant sur le lien correspondant à votre système d'exploitation, puis désippez-le. Double-cliquez sur son icône pour démarrer Processing. Par défaut, le programme est en mode Java. Cliquez sur Java pour ouvrir le menu déroulant comme sur la figure I-9, puis cliquez sur **Ajoutez un mode**.

Sélectionnez **Python Mode for Processing 3** et cliquez sur **Install**. Cela devrait prendre une minute ou deux mais, une fois terminé, vous pourrez codez en Python sur Processing.

Maintenant que Python et Processing sont installés, vous voici prêt pour la conquête des mathématiques !

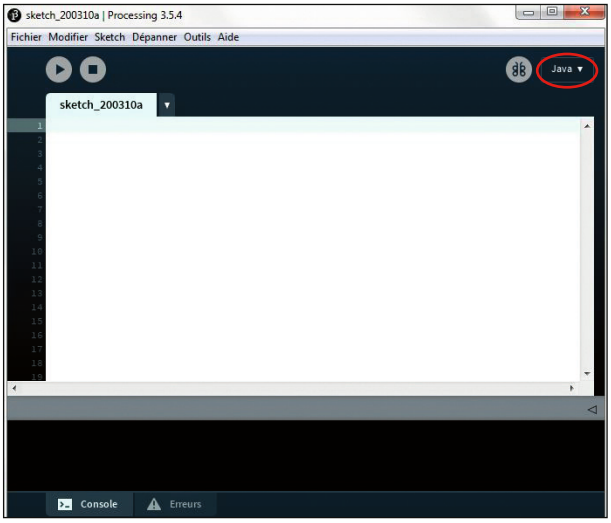


Figure I-9. Le menu où trouver les autres modes de Processing, notamment le mode Python que nous allons utiliser.

PARTIE 1

MONTEZ DANS LE WAGON PYTHON

1

TRACER DES POLYGONES AVEC LE MODULE TURTLE

Il y a quelques siècles de cela, un homme occidental entendit un sage hindou dire que la Terre reposait sur le dos d'une tortue. Lorsqu'il lui demanda sur quoi reposait la tortue, l'hindou répondit : « Sur des tortues, à l'infini. »

Avant de vous appuyer sur les mathématiques pour réaliser tous les projets amusants de ce livre, vous allez apprendre à donner des instructions à votre ordinateur en utilisant le langage de programmation appelé Python. Dans ce chapitre, vous vous familiariserez avec des concepts de base de la programmation comme les boucles, les variables et les fonctions en traçant différentes figures à l'aide du module `turtle` intégré à Python. Comme vous le constaterez, utiliser le module `turtle` permet d'apprendre les fonctionnalités de base de Python de façon amusante et d'avoir un premier aperçu des possibilités qu'offre la programmation.

LE MODULE TURTLE DE PYTHON

Le nom de l'outil turtle sur Python vient de la fonctionnalité du même nom disponible dans le langage de programmation Logo, inventé dans les années 1960 dans le but de rendre la programmation plus accessible au grand public. L'environnement graphique de Logo a permis de rendre les interactions avec la machine plus visuelles et intéressantes. (Je vous invite à lire le livre remarquable de Seymour Paper, *Mindstorms*, et ainsi, découvrir de très nombreuses idées géniales pour apprendre les mathématiques avec les tortues virtuelles de Logo.) Les créateurs de Python ont tant aimé cette fonctionnalité de Logo qu'ils ont décidé d'écrire un module similaire sur Python et de l'appeler `turtle`.

Le module `turtle` de Python vous permet de contrôler une petite image en forme de tortue, comme dans un jeu vidéo. Pour déplacer la tortue sur votre écran, vous devez lui donner des instructions précises. Et, comme la tortue laisse une trace derrière elle, vous pouvez vous en servir pour tracer des formes.

Commençons par importer le module `turtle` !

IMPORTATION DU MODULE TURTLE

Ouvrez un nouveau fichier Python sur l'IDLE et enregistrez-le sous le nom `myturtle.py` dans le dossier `Python`. Vous devriez voir une page vierge s'afficher. Pour utiliser les tortues sur Python, vous devez d'abord importer les fonctions depuis le module `turtle`.

Une *fonction* est un ensemble de lignes de code exécutant une action spécifique et que l'on peut réutiliser sur demande. Il existe une multitude de fonctions pré-intégrées sur Python mais il est possible d'écrire vos propres fonctions (vous apprendrez comment les programmer un peu plus tard dans ce chapitre).

Un *module* (ou *bibliothèque*) est un fichier qui contient des fonctions et des instructions prédéfinies et utilisables dans d'autres programmes. Par exemple, le module `turtle` contient énormément de code très utile téléchargé automatiquement lors de l'installation de Python.

Bien qu'il existe plusieurs façons différentes d'importer un module, nous n'en utiliserons ici qu'une seule, qui a l'avantage d'être simple. En première ligne du fichier `myturtle.py` que vous venez de créer, tapez la ligne suivante :

```
from turtle import *
```

La commande `from` indique que nous importons du code depuis un autre fichier que le nôtre. On donne alors le nom du module que l'on souhaite importer, `turtle` dans notre cas. Nous utilisons le mot-clé `import` pour importer le code depuis le module `turtle`. Enfin, nous

utilisons ici l'astérisque (*), qui correspond à une commande générique signifiant « importer tout ce que contient le module ». Assurez-vous de bien ajouter une espace entre `import` et l'astérisque.

Enregistrez le fichier modifié en vous assurant qu'il se situe bien dans le dossier `Python` ; sinon le programme affichera une erreur.

ATTENTION *Ne pas sauvegarder le fichier sous le nom* `turtle.py`. *Ce fichier existant déjà, cela causerait un conflit lors de l'importation depuis le module* `turtle` *! N'importe quel autre nom fonctionne :* `myturtle.py`, `turtle2.py`, `mondayturtle.py`, *etc.*

DÉPLACEMENT DE VOTRE TORTUE

Maintenant que vous avez importé le module `turtle`, vous êtes prêt à écrire les instructions qui feront se déplacer votre tortue. Nous utiliserons la fonction `forward()` (abrégée `fd`) afin de faire avancer la tortue de quelques pas tout en laissant une trace derrière elle. Notez que la fonction `forward()` est une de celles que nous venons d'importer avec le module `turtle`. Tapez la ligne suivante pour faire avancer votre tortue :

```
forward(100)
```

Nous utilisons ici la fonction `forward()` avec le nombre `100` entre les parenthèses pour indiquer le nombre de pas que la tortue doit effectuer. Dans ce cas, on dit que 100 est l'*argument* que nous donnons à la fonction `forward()`. Chaque fonction accepte un ou plusieurs arguments. N'hésitez pas à essayer la fonction avec d'autres nombres. Lorsque vous pressez la touche **F5** pour démarrer le programme, une nouvelle fenêtre contenant une flèche en son milieu s'ouvre comme sur la figure 1-1.

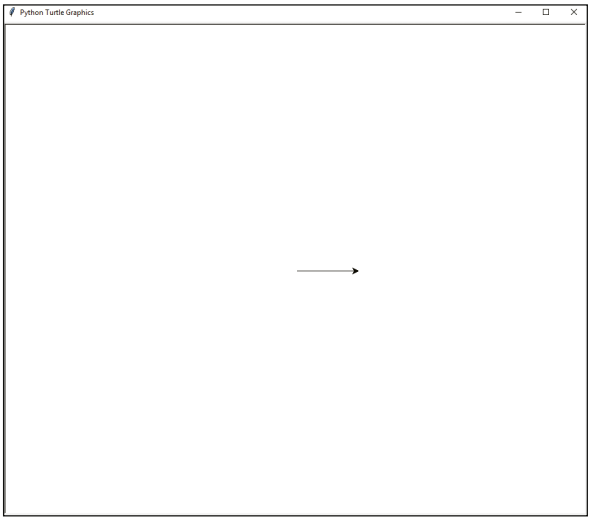


Figure 1-1. Exécution de votre première ligne de code !

Comme vous pouvez le constater, la tortue a commencé au milieu de l'écran et a effectué 100 pas (c'est-à-dire qu'elle a avancé de 100 pixels). Notez que la forme par défaut est une flèche, non pas une tortue, et que la direction par défaut est vers la droite. Pour transformer la flèche en tortue, changez votre code afin qu'il ressemble à ceci :

myturtle.py

```
from turtle import *
forward(100)
shape('turtle')
```

Comme vous l'aurez sûrement deviné, `shape()` est une autre fonction intégrée au module `turtle`. Elle vous permet de changer la flèche par défaut en une autre forme, un cercle ou un carré par exemple. La fonction `shape()` prend ici la chaîne de caractères `'turtle'` en argument et non pas un nombre. (Vous en apprendrez plus sur les chaînes de caractères et les autres types de données dans le chapitre suivant.) Sauvegardez votre fichier `myturtle.py` puis lancez-le une nouvelle fois. Vous devriez obtenir quelque chose ressemblant à la figure 1-2.

Figure 1-2. Changement de la flèche en une tortue !

Votre flèche a désormais la forme d'une petite tortue.

CHANGEMENT DE DIRECTION

La tortue ne peut se diriger que droit devant elle. Pour changer sa direction, vous devrez d'abord la faire pivoter selon un certain angle (exprimé en degrés) à l'aide des fonctions `right()` ou `left()` avant de la faire avancer. Mettez à jour votre fonction en ajoutant les deux dernières lignes de code comme suit :

myturtle.py

```
from turtle import *
forward(100)
shape('turtle')
right(45)
forward(150)
```

Ici, nous allons utiliser la fonction `right()` (ou `rt()` pour abréger) pour faire pivoter la tortue de 45 degrés avant de la faire avancer de 150 pas. Lorsque vous lancez ce code, le résultat devrait ressembler à celui représenté à la figure 1-3.

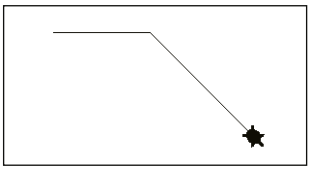


Figure 1-3. Changement de direction de la tortue

Comme vous pouvez le voir, la tortue a démarré sa course au milieu de l'écran, a parcouru 100 pas, pivoté de 45 degrés puis elle a parcouru 150 pas de plus. Notez que Python exécute chaque ligne de code dans l'ordre, du haut vers le bas.

EXERCICE 1-1 : DANSE EN CARRÉ

Reprenez le programme `myturtle.py`. Votre premier défi sera de modifier le code et de n'utiliser que les fonctions `forward` et `right` afin que votre tortue dessine un carré.

RÉPÉTITION DU CODE À L'AIDE DES BOUCLES

Dans chaque langage de programmation, il est possible de répéter automatiquement une série de commandes un certain nombre de fois. Cette fonctionnalité est très utile puisqu'elle nous évite de réécrire le même code, encore et encore, et permet ainsi de désencombrer notre programme. Cela évite également les fautes de frappe qui empêcheraient votre programme de bien s'exécuter.

UTILISATION DE LA BOUCLE FOR

Avec Python, nous utilisons la boucle `for` pour répéter le code. Nous utilisons aussi le mot-clé range pour spécifier le nombre d'itérations de notre boucle. Ouvrez un nouveau fichier dans l'IDLE et enregistrez-le sous le nom `for_loop.py`, puis saisissez les lignes suivantes :

for_loop.py

```
for i in range(2):
    print('hello')
```

Ici, la fonction `range()` crée la variable `i`, appelée *itérateur*, pour chaque boucle. L'itérateur est une valeur qui augmente de un à chaque fois qu'elle est utilisée. Le nombre `2` entre parenthèses est l'argument que nous donnons à la fonction pour contrôler son comportement. Ceci ressemble à la manière de donner différentes valeurs aux fonctions `forward()` et `right()` dans la section précédente.

Dans notre cas, `range(2)` crée une liste de deux nombres, 0 et 1. Pour chacun de ces nombres, la commande `for` exécute les actions situées après les deux points ; ici, afficher (`print`) le mot « hello ».

Faites bien attention à indenter toutes les lignes de code qui doivent être répétées par la boucle en utilisant la touche **Tab** qui correspond à un alinéa de quatre espaces. L'indentation permet à Python de savoir quelles lignes sont contenues dans la boucle afin de savoir exactement quelles instructions répéter. N'oubliez pas, non plus, les deux points à la fin de la ligne ; ils définissent où commence la boucle. Lorsque vous lancez ce programme, vous devez voir le texte suivant s'afficher dans votre interface :

```
hello
hello
```

Comme vous le voyez, le programme affiche `hello` deux fois en raison de la commande `range(2)` qui crée une liste contenant deux valeurs, 0 et 1. Cela signifie que la commande `for` parcourt la liste, réitérant pour chaque valeur l'instruction d'afficher `hello`. Changeons le nombre indiqué entre parenthèses comme ceci :

for_loop.py

```
for i in range(10):
    print('hello')
```

Lorsque vous exécutez ce programme, vous devez voir s'afficher le message `hello` dix fois :

```
hello
hello
hello
hello
hello
```

```
hello
hello
hello
hello
hello
```

Essayons un autre exemple pour mieux maîtriser la boucle `for` que vous utiliserez énormément dans ce livre :

for_loop.py

```
for i in range(10):
    print(i)
```

Du fait que les listes, en Python, commencent à 0 et non à 1, `for i in range(10)` nous donne les chiffres allant de 0 à 9. Ce bout de code signifie « pour chaque valeur contenue entre 0 et 9, afficher cette valeur ». La boucle `for` répète ensuite le code jusqu'à ce que la liste se termine. Lorsque vous exécutez ce code, vous devez obtenir quelque chose comme ceci :

```
0
1
2
3
4
5
6
7
8
9
```

Par la suite, il faudra toujours se rappeler que `i` commence à 0 et fini au nombre donné en argument de `range()` moins 1. Mais, pour l'instant, il suffit de se rappeler que lorsqu'on veut répéter une instruction quatre fois, il suffit d'écrire :

```
for i in range(4):
```

C'est aussi simple que ça ! Voyons comment on peut l'utiliser à notre avantage.

UTILISATION D'UNE BOUCLE FOR POUR TRACER UN CARRÉ

Dans l'exercice 1-1, votre défi était de tracer un carré en utilisant seulement les fonctions `forward()` et `right()`. Pour ce faire, vous avez dû répéter 4 fois les instructions `forward(100)` et `right(90)`. Ainsi, vous avez dû réécrire quatre fois les mêmes lignes de code, ce qui vous a pris plus de temps et vous a laissé plus de chances de faire des fautes de frappe.

Utilisons une boucle `for` pour éviter de répéter le même code. Ci-dessous se trouve le programme `myturtle.py` utilisant, cette fois-ci, une boucle `for` pour éviter de répéter les fonctions `right()` et `forward()` quatre fois :

myturtle.py

```
from turtle import *
shape('turtle')
for i in range(4):
    forward(100)
    right(90)
```

Notez que l'instruction `shape('turtle')` doit venir juste après avoir importé le module `turtle` et avant de commencer à tracer. Les deux lignes de code dans la boucle `for` indiquent à la tortue d'avancer de 100 pas, puis de pivoter de 90 degrés vers sa droite. (Il faut s'imaginer où regarde la tortue pour savoir où se trouve sa droite !) Puisqu'un carré a quatre côtés, nous utilisons `range(4)` pour répéter ces lignes de code quatre fois. Lancez le programme : vous devriez voir s'afficher quelque chose ressemblant à la figure 1-4.

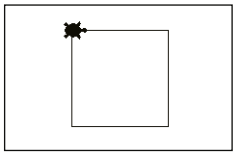


Figure 1-4. Un carré tracé à l'aide de la boucle for

Vous devriez voir la tortue avancer, puis pivoter vers la droite quatre fois et ainsi retourner à sa position d'origine. Vous avez réussi à tracer un carré à l'aide d'une boucle `for` !

CRÉATION DE RACCOURCIS GRÂCE AUX FONCTIONS

Maintenant que vous avons réussi à écrire un code traçant un carré, nous pouvons enregistrer ce code sous un « mot magique » que nous pourrons utiliser à chaque fois que nous voudrons réutiliser le code. Dans chaque langage de programmation, il est possible d'en faire autant. En Python, on appelle cela une *fonction*. C'est la fonctionnalité la plus importante en programmation. Les fonctions permettent de rendre le code plus compacte et plus simple à gérer. Le plus souvent, diviser un problème en plusieurs petites fonctions permet de trouver la méthode la plus efficace pour le résoudre. Jusqu'ici, vous n'avez utilisé que des fonctions intégrées au module `turtle`. Dans cette section, vous allez apprendre à définir vos propres fonctions.

Pour définir une fonction, il faut commencer par lui donner un nom. Celui-ci peut être ce que vous voulez du moment qu'il n'est pas déjà un mot-clé sur Python, tels que le sont les mots `list`, `range`, etc. Lorsque vous nommez des fonctions, il est préférable d'être explicite pour que vous puissiez vous rappeler de leur rôle lorsque vous les utilisez. Appelons notre fonction `square()` étant donné que l'on s'en servira pour tracer des carrés :

myturtle.py

```
def square():
    for i in range(4):
        forward(100)
        right(90)
```

La commande `def` indique à Python que nous sommes en train de définir une fonction et le mot qui la suit sera son nom : `square()`, dans notre exemple. N'oubliez pas de mettre des parenthèses après `square` ! Elles précisent que ce que vous êtes en train de définir est une fonction. Plus tard, nous ajouterons des valeurs entre ces parenthèses mais il faut toujours les écrire même si, pour l'instant, elles ne contiennent aucune valeur afin que Python comprenne qu'il s'agit bien d'une fonction. Aussi, n'oubliez pas le deux-points à la fin de la définition d'une fonction. Notez que nous indentons tout le code qui compose la fonction afin que Python sache que le code fait partie de celle-ci.

Si vous lancez le programme maintenant, rien ne se passera. Vous venez de définir une fonction, mais vous n'avez pas encore demandé au programme de l'exécuter. Pour ce faire, vous allez devoir appeler cette fonction à la fin du fichier `myturtle.py`, après avoir défini la fonction. Tapez le code suivant.

Programme 1-1 : La fonction square() est appelée à la fin du fichier

myturtle.py

```
from turtle import *
shape('turtle')
def square():
    for i in range(4):
        forward(100)
        right(90)
square()
```

Lorsque vous appelez la fonction `square()` en fin de programme comme ici, celui-ci s'exécute correctement. Vous pourrez maintenant appeler la fonction `square()` à chaque fois que vous voudrez tracer un carré dans votre programme.

Vous pouvez aussi utiliser cette fonction dans une boucle afin de tracer quelque chose de plus complexe. Par exemple, si vous voulez tracer un carré, pivoter un petit peu vers la droite, en tracer un autre, puis pivoter encore et ainsi de suite, la meilleure façon sera de placer votre fonction dans une boucle.