

Al Sweigart

Apprendre à coder des jeux vidéo en PYTHON

Dès 10 ans



La programmation
accessible
aux enfants !



● Éditions
EYROLLES

Apprendre à coder des jeux vidéo en PYTHON

Ne vous contentez pas de jouer à des jeux, créez-les !

Cet ouvrage a pour ambition de vous initier au développement de jeux vidéo grâce au populaire langage Python, et ce, même si n'avez encore jamais programmé de votre vie !

Vous développerez d'abord des jeux classiques comme le Pendu, Devinez le nombre ou le Morpion, avant de vous attaquer à la conception de jeux plus avancés techniquement qui intègrent, entre autres, du texte, des animations graphiques et du son.

Par la même occasion, vous apprendrez les concepts de base de la programmation et des mathématiques pour amener vos compétences en codage de jeux vidéo à un autre niveau. Tous les projets de cet ouvrage sont basés sur la dernière version (3) de Python.

Au cours de votre lecture, vous allez acquérir des bases solides en matière de programmation Python. Quel nouveau jeu allez-vous ensuite pouvoir créer à l'aide de la puissance de Python ?

Tout au long de cette aventure en programmation, vous apprendrez à :

- choisir le bon type de structure de données pour faire le travail, comme des listes, dictionnaires ou tuples ;
- ajouter des illustrations et des animations dans votre jeu à l'aide du module pygame ;
- interagir avec le clavier et la souris ;
- programmer une intelligence artificielle suffisamment simple pour jouer contre l'ordinateur ;
- utiliser la cryptographie pour convertir des messages texte en codes secrets ;
- déboguer vos programmes et identifier les erreurs les plus communes.

À propos de l'auteur

Al Sweigart est un développeur de logiciels qui enseigne la programmation à la fois aux adultes et aux enfants. Il anime le blog inventwithpython.com qui propose de nombreux tutoriels de programmation.

Dans la même collection



www.editions-eyrolles.com



Apprendre à coder
des jeux vidéo en
PYTHON

Al Sweigart

Apprendre à coder
des jeux vidéo en
PYTHON



● Éditions
EYROLLES

ÉDITIONS EYROLLES
61, bd Saint-Germain
75240 Paris Cedex 05
www.editions-eyrolles.com

Traduction autorisée de l'ouvrage en langue anglaise intitulé
Invent Your Own Computer Games with Python, 4th Edition
de Al Sweigart (ISBN : 978-1-59327-795-6),
publié par No Starch Press.

All Rights Reserved.

En application de la loi du 11 mars 1957, il est interdit de reproduire intégralement ou partiellement le présent ouvrage, sur quelque support que ce soit, sans l'autorisation de l'Éditeur ou du Centre Français d'exploitation du droit de copie, 20, rue des Grands Augustins, 75006 Paris.

© 2016 by Al Sweigart / No Starch Press pour l'édition en langue anglaise

© Éditions Eyrolles, 2019, pour la présente édition, ISBN : 978-2-212-67757-7

© Traduction française : Paul Durand Degranges

TABLES DES MATIÈRES

Introduction	1
À qui s'adresse ce livre ?	2
À propos de cet ouvrage	3
Comment utiliser ce livre	5
Numéro de ligne et indentation	5
Longues lignes de code	6
Télécharger et installer Python	6
Windows	7
Mac OS	7
Ubuntu	7
Démarrer IDLE	8
Obtenir de l'aide en ligne	9
1	
Le shell interactif	11
Sujets traités dans ce chapitre	11
Calculs mathématiques simples	12
Entiers et décimaux	13
Expressions	13
Évaluer les expressions	14
Erreurs de syntaxe	15
Stocker les valeurs dans des variables	16
En résumé	20
2	
Écrire des programmes	21
Sujets traités dans ce chapitre	22
Chaînes de caractères	22
Concaténation de chaînes de caractères	23
Écrire des programmes dans l'éditeur de texte d'IDLE	23
Créer le programme Bonjour	24
Enregistrer le programme	25
Exécuter votre programme	26
Comment fonctionne le programme Bonjour	27
Nommer les variables	30
En résumé	31

3

Devinez le nombre 33

Sujets traités dans ce chapitre	34
Le jeu Devinez le nombre en action	34
Code source pour Devinez le nombre	34
Importer le module random	36
Générer un nombre aléatoire avec la fonction random.randint()	37
Accueillir le joueur	38
Instruction de contrôle de flux	38
Utiliser des boucles pour répéter le code	39
Grouper par blocs	39
Boucler avec l'instruction for	40
Obtenir la réponse du joueur	42
Convertir des valeurs avec les fonctions int(), float() et str()	42
Les données de type booléen	44
Opérateurs de comparaison	44
Connaître le vrai du faux avec les conditions	45
Expérimenter les booléens, les opérateurs de comparaison et les conditions	45
Différence entre = et ==	46
L'instruction if	46
Sortir d'une boucle à l'aide de l'instruction break	47
Vérifier si le joueur a gagné	47
Vérifier si le joueur a perdu	48
En résumé	49

4

Un programme de blagues 51

Sujets traités dans ce chapitre	52
Un exemple d'exécution des blagues	52
Source du programme	52
Fonctionnement du code	53
Caractère d'échappement	53
Guillemets anglais simples et doubles	55
La fonction print() et le paramètre clavier	56
En résumé	57

5

L'antre du dragon 59

Jouer à l'antre du dragon	59
Sujets traités dans ce chapitre	60

Exemple d'exécution	60
Organigramme du programme.	61
Code source pour l'antre du dragon	62
Importer les modules random et time	63
Fonctions dans le programme	63
Déclaration def	63
Appeler une fonction	64
Où placer les définitions de fonctions ?	64
Chaîne de caractères sur plusieurs lignes.	65
Boucler avec l'instruction while	66
Les opérateurs booléen	66
L'opérateur and	67
L'opérateur or	68
L'opérateur not	68
Évaluer les opérateurs booléens.	69
Valeurs renvoyées.	70
Portée globale et locale	71
Paramètres des fonctions	72
Afficher le résultat du jeu.	73
Choisir où se trouve le gentil dragon	74
Boucler le jeu.	75
Appeler les fonctions dans le programme	75
Demander de rejouer.	76
En résumé	77

6

Utiliser le débogueur 79

Sujets traités dans ce chapitre.	80
Types de bogues	80
Le débogueur.	81
Démarrer le débogueur.	81
Avancer dans le programme avec le débogueur	83
Zone Globals	83
Zone Locals	84
Les boutons Go et Quit	84
Step into, over et out.	84
Trouver le bogue	86
Configurer des points d'arrêt	89
Utiliser les points d'arrêt	90
En résumé	92

7

L'organigramme du jeu du pendu 93

Sujets traités dans ce chapitre	93
Jouer au pendu	94
Exemple de fonctionnement du jeu	94
Art ASCII	95
Concevoir le programme à l'aide d'un organigramme	96
Créer l'organigramme	97
Les branches de l'organigramme	98
Finir ou redémarrer le jeu	100
Proposer une autre lettre	101
Donner des infos au joueur	102
En résumé	103

8

Écrire le code du pendu 105

Sujets traités dans ce chapitre	106
Le code source du pendu	106
Importer le module random	109
Constantes	110
Le type de données liste	110
Accéder aux éléments avec des index	111
Erreur d'index	112
Modifier un élément avec l'index	112
Concaténation de listes	112
L'opérateur in	113
Appeler une méthode	113
Les méthodes reverse() et append()	113
La méthode split()	114
Choisir un mot dans la liste	115
Afficher le jeu	116
Les fonctions list() et range()	117
Découpage de liste et de chaîne	118
Afficher le mot avec des tirets	118
Demander une lettre	120
Les méthodes lower() et upper()	121
Sortir de la boucle while	122
Instruction elif	122
Vérifier la proposition du joueur	123
Demander au joueur s'il rejoue	124
Les fonctions dans le jeu	124
La boucle du programme	125

Appel de la fonction <code>displayBoard()</code>	125
Permettre la saisie d'une proposition.	126
Vérifier si la lettre est bonne	126
Vérifier si le joueur a gagné	126
Gérer une mauvaise réponse	127
Vérifier si le joueur a perdu	128
Arrêter ou recommencer le jeu	128
En résumé	129

9

Améliorer le jeu du pendu 131

Sujets traités dans ce chapitre.	132
Augmenter le nombre de propositions.	132
Le type de données dictionnaire.	132
Connaître la taille du dictionnaire avec <code>len()</code>	134
Différences entre dictionnaires et listes	134
Les méthodes <code>keys()</code> et <code>values()</code> des dictionnaires	135
Utiliser les dictionnaires pour le jeu du pendu	135
Choisir aléatoirement une liste	136
Supprimer des éléments d'une liste.	138
Affectation multiple	139
Afficher la catégorie de mots	140
En résumé	140

10

Le jeu du morpion 143

Sujets traités dans ce chapitre.	144
Exécution du jeu	144
Code source du morpion	145
Concevoir le programme.	149
Représenter le plateau sous la forme de données	151
Établir la stratégie de l'IA	151
Importation du module <code>random</code>	152
Afficher le plateau de jeu	153
Laisser le joueur choisir X ou O	154
Choisir le joueur qui démarre	155
Placer un pion sur le plateau	155
Références à des listes	156
Utiliser la référence aux listes dans <code>makeMove()</code>	159
Vérifier si le joueur a gagné	159
Dupliquer les données du plateau	162
Vérifier si une case est libre.	162

Permettre au joueur de placer sa marque	162
Court-circuiter l'évaluation	163
Choisir un emplacement	165
La valeur None	166
Créer l'IA de l'ordinateur	167
Vérifier si l'ordinateur peut gagner en un coup	167
Vérifier si le joueur gagne en un coup	168
Vérifier les coins, le centre et les côtés (dans cet ordre)	169
Vérifier si le plateau est plein	169
La boucle du jeu	170
Choisir la marque du joueur et décider qui commence	170
Exécuter le tour du joueur	171
Exécuter le tour de l'ordinateur	172
Proposer une nouvelle partie	173
En résumé	173

11

Le jeu

Pico, fermi, bagels

175

Sujets traités dans ce chapitre	176
Exemple d'exécution de Bagels	176
Code source de Pico, fermi, bagels	177
Organigramme de Bagels	179
Importer random et définir getSecretNum()	179
Créer un jeu de chiffres uniques	180
Changer l'ordre de la liste à l'aide de la fonction random.shuffle()	180
Obtenir le nombre à l'aide de chiffres mélangés	181
Opérateur d'affectation augmentée	181
Calculer l'indice à fournir	182
La méthode sort()	183
La méthode join()	184
Vérifier qu'une chaîne contient uniquement des chiffres	184
Démarrer le jeu	185
Interpolation de chaîne	185
La boucle du jeu	187
Obtenir les propositions du joueur	187
Obtenir les indices pour le joueur	188
Vérifier si le joueur a gagné ou perdu	188
Proposer de rejouer	188
En résumé	189

12

Le système de coordonnées cartésiennes 191

Sujets traités dans ce chapitre	192
Grilles et coordonnées cartésiennes	192
Nombres négatifs	193
Le système de coordonnées de l'écran	195
Astuces mathématiques	196
Astuce 1 : un moins « mange » le signe plus sur sa gauche	196
Astuce 2 : deux moins donnent un plus	196
Astuce 3 : deux chiffres qui s'additionnent peuvent s'intervertir	197
Valeur absolue et la fonction abs()	198
En résumé	198

13

La chasse au trésor 199

Sujets traités dans ce chapitre	200
Exemple d'exécution du jeu	201
Code source du jeu	203
Concevoir le programme	208
Importer les modules random, sys et math	208
Créer le plateau de jeu	209
Dessiner le plateau de jeu	210
Dessiner les coordonnées en x en haut du plateau	211
Dessiner l'océan	212
Afficher une ligne dans l'océan	212
Afficher les coordonnées en x en bas de l'écran	213
Créer les trésors aléatoirement	213
Déterminer si un mouvement est valide	214
Placer un sonar sur le jeu	214
Trouver le coffre le plus proche	215
Supprimer des valeurs à l'aide de la méthode de liste remove()	217
Obtenir le choix du joueur	219
Afficher les instructions	220
La boucle du jeu	221
Afficher l'état du jeu	222
Gérer le choix du joueur	222
Trouver un coffre	223
Vérifier si le joueur a gagné	223
Vérifier si le joueur a perdu	224
Terminer le programme avec la fonction sys.exit	224
En résumé	224

14

Le chiffrement de César 227

Sujets traités dans ce chapitre	228
Cryptographie et chiffrement	228
Fonctionnement du chiffrement de César	228
Exemple de fonctionnement du chiffrement de César	230
Code source pour le chiffrement de César	231
Configurer la longueur maximale de la clé	232
Choisir de chiffrer ou de déchiffrer	233
Obtenir le message du joueur	233
Obtenir la clé du joueur	233
Chiffrer ou déchiffrer le message	234
Trouver la chaîne passée avec la méthode <code>string()</code>	234
Chiffrer ou déchiffrer chaque lettre	235
Démarrer le programme	236
La technique force brute	237
Ajouter le mode force brute	237
En résumé	239

15

Le jeu d'Othello 241

Sujets traités dans ce chapitre	242
Comment jouer à Othello	242
Exemple d'exécution d'Othello	245
Code source d'Othello	247
Importer les modules et configurer les constantes	253
Structure de données du plateau de jeu	253
Afficher le plateau	254
Créer une nouvelle structure de données pour le plateau	255
Vérifier si un choix est valide	255
Vérifier les huit directions	256
Vérifier s'il y a des pions à retourner	257
Vérifier la validité des coordonnées	258
Obtenir une liste de tous les choix valides	258
Appeler la fonction <code>bool()</code>	259
Obtenir le score du plateau de jeu	260
Demander au joueur de choisir une « couleur » de pion	261
Déterminer qui commence	261
Placer un pion sur le plateau	262
Copier la structure du plateau	262
Déterminer si l'espace est dans un coin	263
Obtenir le choix du joueur	263

Obtenir le choix de l'ordinateur	265
Mouvement dans les coins	265
Obtenir la liste des choix donnant le plus de points	266
Afficher les scores à l'écran	267
Démarrer le jeu	267
Vérifier si le jeu est dans une impasse.	268
Le tour du joueur.	268
Le tour de l'ordinateur	270
La boucle du jeu	270
Demander au joueur s'il rejoue	271
En résumé	272

16

Simulation avec l'IA 273

Sujets traités dans ce chapitre	274
Faire jouer l'ordinateur contre lui-même	274
Exemple d'exécution de la simulation 1.	275
Code source pour la simulation 1	275
Supprimer l'invite et ajouter un ordinateur comme joueur	277
Faire jouer l'ordinateur plusieurs fois	278
Exemple d'exécution de la simulation 2.	278
Code source pour la simulation 2	278
Conserver une trace des jeux	279
Mettre en commentaires les instructions print().	280
Utiliser les pourcentages pour comparer les IA	281
Divisions et décimales.	281
La fonction round()	282
Comparer différents algorithmes	283
Code source pour la simulation 3	283
Fonctionnement de l'IA dans la simulation 3	284
Comparer les IA	287
Coins et côtés contre coins seuls.	289
En résumé	290

17

Créer des graphiques 291

Sujets traités dans ce chapitre	292
Installer pygame	292
Bonjour avec pygame	293
Exécution du programme Bonjour	293
Code source pour Bonjour version pygame	294
Importer le module pygame	295

Initialiser pygame	296
Configurer la fenêtre pygame	296
Tuple	297
Objet Surface	297
Configurer les variables de couleur	298
Écrire du texte dans la fenêtre pygame	299
Définir le style de police	299
Rendu d'un objet Font	300
Configurer l'emplacement du texte et les attributs de Rect	300
Remplir un objet Surface avec une couleur	302
Les fonctions de dessin de pygame	303
Dessiner un polygone	303
Dessiner une ligne	304
Dessiner un cercle	305
Dessiner une ellipse	305
Dessiner un rectangle	306
Colorer les pixels	306
La méthode blit() pour les objets Surface	307
Dessiner les objets Surface à l'écran	308
Les événements et la boucle du programme	308
Obtenir les objets Event	308
Quitter le programme	309
En résumé	309

18

Animer des graphiques 311

Sujets traités dans ce chapitre	311
Exemple d'exécution du programme Animation	312
Code source du programme Animation	312
Déplacer et faire rebondir les boîtes	314
Configurer les constantes	315
Constantes pour les directions	316
Constantes pour les couleurs	317
Configurer la structure de données des boîtes	317
La boucle principale	318
Gérer l'arrêt du programme	318
Déplacer les boîtes	319
Faire rebondir une boîte	320
Dessiner les boîtes à leur nouvelle position	321
Dessiner la fenêtre à l'écran	321
En résumé	322

19

Détection de collision

323

Sujets traités dans ce chapitre	324
Exemple d'exécution du programme de détection de collision	324
Code source du programme de détection de collision	325
Importer les modules	327
Utiliser une horloge pour rythmer le jeu	328
Configurer la fenêtre et la structure des données	328
Configurer les variables de suivi des mouvements	330
Gérer les événements	330
Gérer l'événement KEYDOWN	331
Gérer l'événement KEYUP	333
Téléporter le joueur	334
Ajouter des carrés de nourriture	334
Déplacer le joueur dans la fenêtre	335
Dessiner la boîte à l'écran	336
Détection des collisions	336
Dessiner les carrés de nourriture dans la fenêtre	337
En résumé	338

20

Utiliser des sons et des images

339

Sujets traités dans ce chapitre	340
Ajouter des images avec des sprites	340
Fichiers sons et images	341
Exemple d'exécution du programme Sprites et sons	341
Code source pour le programme Sprites et sons	342
Configurer la fenêtre et la structure de données	345
Ajouter un sprite	345
Configurer la musique et les sons	346
Ajouter des fichiers audio	346
Activer et désactiver les sons	347
Dessiner le joueur dans la fenêtre	348
Détection des collisions	348
Dessiner les cerises dans la fenêtre	349
En résumé	349

Un jeu d'esquive avec images et sons 351

Sujets traités dans ce chapitre	352
Les types de données de base de pygame	352
Exemple d'exécution de Dodger	353
Code source de Dodger	354
Importer les modules	358
Configurer les constantes	359
Définir les fonctions	360
Arrêter et mettre en pause le jeu	360
Conserver la trace des collisions	361
Dessiner du texte dans la fenêtre.	361
Initialiser pygame et configurer la fenêtre	362
Configurer les objets police, sons et images	363
Afficher l'écran de départ	364
Démarrer le jeu	365
La boucle du jeu	367
Gérer les événements du clavier	367
Gérer les mouvements de la souris	368
Ajouter des méchants	369
Déplacer les méchants et le personnage	370
Implémenter les codes de triche.	372
Supprimer les méchants.	372
Dessiner la fenêtre	373
Dessiner le score du joueur.	373
Dessiner le personnage et les méchants	373
Détection de collisions	374
L'écran de fin.	375
Modifier le jeu Dodger	375
En résumé	376



INTRODUCTION



Lorsque j'ai commencé à jouer aux jeux vidéo en tant qu'enfant, j'ai été « scotché ». Cependant, je ne voulais pas uniquement jouer ; je voulais aussi créer mes jeux. J'ai alors trouvé un livre comme celui-ci pour apprendre à réaliser mes premiers programmes. C'était simple et amusant. Mes premiers jeux étaient semblables à ceux de cet ouvrage. Ils n'étaient pas du niveau de ceux de Nintendo que mes parents m'achetaient, mais c'est moi qui les avais conçus.

Maintenant que je suis adulte, j'ai toujours plaisir à programmer et je suis payé pour cela. Même si vous ne souhaitez pas en faire votre métier, il est bon de savoir programmer. Cela entraîne le cerveau à penser de façon logique, à concevoir des plans et à reconsidérer ses idées face à des erreurs dans le code.

De nombreux livres de programmation pour débutants tombent dans deux catégories. Dans la première, on trouve des ouvrages qui présentent la programmation de façon tellement

simplifiée que ce n'en est plus. Dans l'autre catégorie, on trouve des livres qui enseignent la programmation comme s'il s'agissait de mathématiques, avec tous les principes et les concepts, mais sans lien avec le monde réel. Le livre que vous lisez tente une approche différente et vous indique comment programmer en réalisant des jeux vidéo ; je montre le code source des jeux puis j'explique les principes à partir d'exemples. C'est ainsi que j'ai étudié la programmation. Plus j'ai appris sur la manière de travailler des autres, plus j'ai eu d'idées pour mes propres codes.

Tout ce dont vous avez besoin, c'est d'un ordinateur, d'un logiciel gratuit appelé un interpréteur Python et de cet ouvrage. Une fois que vous aurez appris à créer des jeux à l'aide de ce livre, vous serez en mesure d'écrire vos propres programmes.

Les ordinateurs sont des machines incroyables et, contrairement à ce que pensent beaucoup de personnes, ce n'est pas très compliqué de les commander. Un programme informatique est un ensemble d'instructions que l'ordinateur comprend, de la même manière qu'un roman est un ensemble de mots que le lecteur comprend. Pour donner des instructions à un ordinateur, on utilise un langage de programmation. Dans cet ouvrage, nous utilisons Python. Il existe de nombreux autres langages, comme Java, JavaScript, PHP ou C++.

Quand j'étais enfant, j'ai appris le BASIC, mais de nouveaux langages comme Python sont plus faciles à comprendre. Python est utilisé par des professionnels pour leur travail, mais aussi pour le divertissement. Ce langage est totalement gratuit et il suffit d'une connexion à Internet pour le télécharger.

Comme les jeux vidéo ne sont rien d'autre que des programmes, ils sont également constitués d'instructions. Les jeux que vous allez créer à partir de cet ouvrage semblent simples comparés à ceux pour Xbox, Playstation ou Nintendo. Ils ne présentent pas de jolis graphiques parce que le but est de vous apprendre les bases de la programmation. Ils sont volontairement épurés pour que vous vous concentriez sur votre apprentissage. De plus, les jeux n'ont pas à être complexes pour être divertissants.

À qui s'adresse ce livre ?

La programmation n'est pas compliquée. En revanche, il est compliqué de trouver des sujets pour l'enseigner de façon intéressante. Les ouvrages traitent souvent de nombreux sujets inutiles aux débutants. Ce livre va vous apprendre des techniques utiles en réalisant des jeux amusants. Il s'adresse aux personnes suivantes :

- débutants qui veulent apprendre la programmation, même s'ils n'ont aucune expérience ;
- enfants et adolescents qui veulent apprendre la programmation en créant des jeux ;
- adultes et professeurs qui souhaitent enseigner la programmation ;
- toute personne, jeune ou moins jeune, qui souhaite approfondir sa connaissance de la programmation avec un langage professionnel.

À propos de cet ouvrage

Dans la plupart des chapitres de ce livre, un jeu simple est présenté et expliqué. Quelques chapitres traitent de sujet utiles et complémentaires (comment déboguer un programme, par exemple). Les nouveaux concepts de programmation sont expliqués quand les jeux y font appel et les chapitres sont conçus pour être lus dans l'ordre.

- **Chapitre 1 – le shell interactif**

Servez-vous du shell interactif de Python pour tester du code ligne après ligne.

- **Chapitre 2 – écrire des programmes**

Utilisez l'éditeur de code de Python pour écrire des programmes.

- **Chapitre 3 – devinez le nombre**

Écrivez votre premier jeu, qui demande au joueur de deviner un nombre et lui indique si sa proposition est supérieure ou inférieure. Maniez les entrées-sorties et les comparaisons de base.

- **Chapitre 4 – un programme de blagues**

Manipulez les chaînes de caractères dans un programme simple qui raconte des blagues à l'utilisateur.

- **Chapitre 5 – l'autre du dragon**

Définissez des fonctions. Le joueur doit choisir entre deux grottes : l'une d'elles est occupée par un gentil dragon et l'autre par un dragon affamé.

- **Chapitre 6 – utiliser le débogueur**

Découvrez un outil très utile pour résoudre des problèmes dans votre programme.

- **Chapitre 7 – l’organigramme du pendu**
Prenez l’habitude d’écrire des organigrammes pour planifier les longs programmes, comme celui du jeu du pendu.
- **Chapitre 8 – écrire le code du pendu**
Découvrez les méthodes et les listes pour deviner un mot en dessinant un élément du pendu à chaque mauvaise réponse.
- **Chapitre 9 – améliorer le jeu du pendu**
Ajoutez de nouvelles fonctionnalités au jeu précédent à l’aide du type de données dictionnaire de Python.
- **Chapitre 10 – le jeu du morpion**
Programmez une intelligence artificielle (IA) pour faire jouer l’humain contre l’ordinateur.
- **Chapitre 11 – le jeu pico, fermi, bagels**
Mélangez, triez et complétez des listes pour un jeu de déduction dans lequel le joueur doit devenir un nombre, guidé par des indices.
- **Chapitre 12 – le système de coordonnées cartésiennes**
(Re)découvrez le système de coordonnées qui vous sera utile dans les chapitres suivants.
- **Chapitre 13 – la chasse au trésor**
Définissez des structures de données complexes pour représenter un système de coordonnées. Appliquez-les dans un jeu consistant à trouver des trésors enfouis dans l’océan.
- **Chapitre 14 – le chiffrement de César**
Allez plus loin dans la manipulation de chaînes de caractères pour chiffrer et déchiffrer des messages.
- **Chapitre 15 – Le jeu d’Othello**
Codez un jeu ordinateur-contre-humain avancé, pratiquement imbattable, capable de simuler un choix et d’en évaluer les conséquences.
- **Chapitre 16 – simulation avec l’IA**
Testez différents algorithmes en transformant le jeu d’Othello pour que plusieurs IA s’affrontent.

- **Chapitre 17 – créer des graphiques**
Découvrez le module `pygame` de Python et son utilisation pour dessiner des graphiques en 2D.
- **Chapitre 18 – animer des graphiques**
Animez des formes géométriques simples avec `pygame`.
- **Chapitre 19 – détection de collision**
Déterminez et traitez la collision d'objets dans les graphiques 2D.
- **Chapitre 20 – utiliser les sons et les images**
Améliorez vos jeux en y intégrant des sons et des images.
- **Chapitre 21 – un jeu d'esquive avec images et sons**
Combinez les concepts des chapitres 17 à 20 pour concevoir un programme d'évitement : Dodger.



1

LE SHELL INTERACTIF



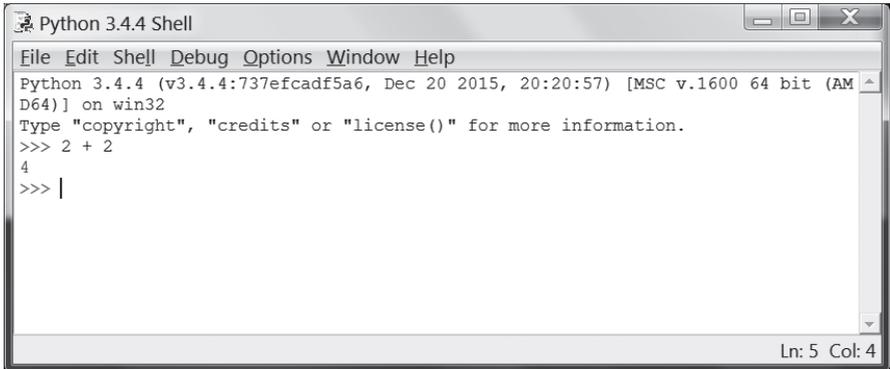
Avant de créer des jeux, vous devez connaître les bases de la programmation. Dans ce chapitre, vous allez apprendre à utiliser le shell interactif de Python et effectuer des calculs de base.

Sujets traités dans ce chapitre

- Opérateurs
- Chiffres entiers et décimaux
- Valeurs
- Expressions
- Erreurs de syntaxe
- Stocker des valeurs dans des variables

Calculs mathématiques simples

Lancez IDLE en suivant les étapes de « Démarrer IDLE », page 8. Tout d'abord, nous allons utiliser Python pour résoudre quelques calculs mathématiques simples. Le shell interactif peut servir de calculatrice. Saisissez `2+2` à l'invite `>>>` puis appuyez sur **Entrée** (ou **Retour** selon les ordinateurs). La figure 1-1 montre la réponse, qui s'affiche immédiatement.



```
Python 3.4.4 Shell
File Edit Shell Debug Options Window Help
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015, 20:20:57) [MSC v.1600 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 2 + 2
4
>>> |
```

FIGURE 1-1 Saisissez `2+2` dans le shell interactif.

La résolution de ce problème mathématique se code à l'aide de quelques instructions de programmation. Le signe plus `+` indique à l'ordinateur d'ajouter les deux chiffres `2`. L'ordinateur effectue l'opération et donne la réponse en affichant le chiffre `4` sur la ligne suivante. Le tableau 1-1 liste les autres opérateurs mathématiques disponibles dans Python.

TABLEAU 1-1 Opérateurs mathématiques

Opérateur	Opération
<code>+</code>	Addition
<code>-</code>	Soustraction
<code>*</code>	Multiplication
<code>/</code>	Division

Le signe `-` soustrait des nombres, l'astérisque `*` les multiplie et la barre de fraction `/` les divise. Lorsqu'ils sont utilisés pour ces fonctions, ces symboles sont appelés des opérateurs. Les opérateurs indiquent à Python ce qu'il faut faire avec les chiffres qui les entourent.

Entiers et décimaux

Les entiers sont des nombres sans décimales comme 4, 99 ou 0. Les décimaux sont des nombres avec au moins une décimale comme 3.5, 42.1 ou 5.0. Dans Python, 5 est un nombre entier mais 5.0 est un nombre à décimale. Notez qu'en programmation on utilise la notation anglo-saxonne : le séparateur décimal est un point et non pas une virgule.

Ces nombres sont appelés des valeurs (par la suite, vous découvrirez d'autres valeurs en plus des nombres). Dans le problème de mathématique que vous avez saisi dans le shell, 2 et 2 sont des valeurs entières.

Expressions

Le problème mathématique $2+2$ est l'exemple d'une expression. Comme le montre la figure 1-2, les expressions sont composées de valeurs (les nombres) connectées par des opérateurs et produisent une nouvelle valeur (l'expression) que le code peut utiliser. Les ordinateurs savent résoudre des millions d'expressions par seconde.

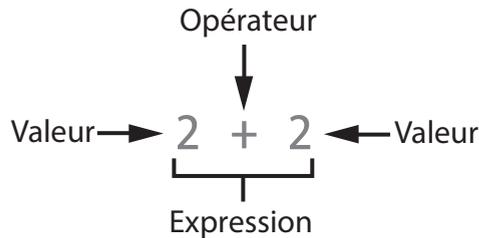


FIGURE 1-2 Une expression est composée de valeurs et d'opérateurs.

Essayez de saisir les expressions suivantes dans le shell interactif, en appuyant sur **Entrée** après chacune des expressions.

```
>>> 2+2+2+2+2
10
>>> 8*6
48
>>> 10-5+6
11
>>> 2 +      2
4
```

Notez les espaces dans le dernier exemple. Avec Python, vous pouvez insérer n'importe quel nombre d'espaces entre les valeurs et les opérateurs. Toutefois, vous devez toujours démarrer l'expression au début de la ligne (sans espace) lorsque vous utilisez le mode shell interactif.

Évaluer les expressions

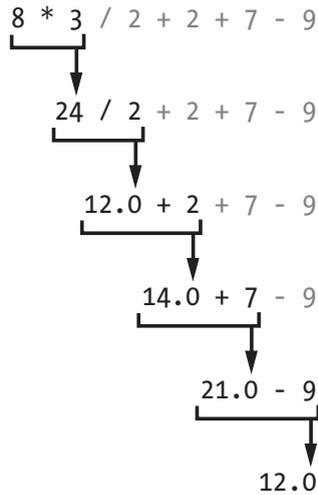
Lorsqu'un ordinateur résout $10+5$ et renvoie la valeur 15 , on dit qu'il a évalué l'expression. Évaluer une expression revient à la réduire à une simple valeur, de la même manière que la résolution d'un problème de mathématique réduit ce dernier à une simple valeur : la réponse. Par exemple, les expressions $10+5$ et $10+3+2$ renvoient la valeur 15 lorsqu'elles sont évaluées.

Lorsque Python évalue une expression, il suit un ordre pour les opérations, de la même manière qu'en mathématiques. Voici quelques règles.

- Les éléments de l'expression se trouvant entre parenthèses sont évalués en premier.
- Les multiplications et les divisions sont réalisées avant les additions et les soustractions.
- L'évaluation s'effectue de gauche à droite.

L'évaluation de l'expression $1+2*3+4$ donne 11 et non pas 13 , parce que l'opération $2*3$ est évaluée en premier. Si l'expression avait été écrite $(1+2)*(3+4)$, le résultat aurait été 21 , parce que les opérations entre parenthèses $(1+2)$ et $(3+4)$ sont évaluées avant la multiplication.

Les expressions peuvent avoir n'importe quelle longueur mais elles ont toujours comme résultat une valeur unique. Même les valeurs uniques sont des expressions. Par exemple, l'évaluation de l'expression 15 donne la valeur 15 . L'expression $8*3/2+2+7-9$ donne la valeur 12.0 à l'aide des étapes suivantes :



Même si l'ordinateur effectue les étapes précédentes, vous ne le voyez pas en mode shell interactif. Ce dernier vous renvoie uniquement le résultat.

```
>>> 8 * 3 / 2 + 2 + 7 - 9
12.0
```

Notez que les expressions contenant l'opérateur de division renvoient toujours une valeur décimale. Par exemple, $24/2$ donne 12.0 . Les opérations mathématiques avec au moins une valeur décimale renvoient également une valeur décimale. Ainsi, $12.0+2$ donne 14.0 .

Erreurs de syntaxe

Si vous saisissez $5+$ dans le shell interactif, vous obtenez le message d'erreur suivant :

```
>>> 5 +
SyntaxError: invalid syntax
```

Cette erreur apparaît parce que $5+$ n'est pas une expression. Les expressions sont composées de valeurs connectées par des opérateurs et l'opérateur $+$ nécessite une valeur avant et une valeur après. Un message d'erreur apparaît lorsqu'une valeur attendue a été oubliée.

`SyntaxError` indique que Python ne comprend pas l'expression parce que vous ne l'avez pas saisie correctement. En programmation, il ne suffit pas de donner des instructions à l'ordinateur ; il est nécessaire de savoir comment les donner correctement.

Ne vous inquiétez pas de faire des erreurs. Celles-ci n'endommageront pas votre ordinateur. Il suffit de saisir à nouveau l'expression dans le mode shell interactif après l'invite.

Stocker les valeurs dans des variables

Lorsqu'une expression renvoie une valeur, vous pouvez utiliser cette dernière par la suite en la stockant dans une variable. Imaginez une variable comme une boîte dans laquelle vous placez la valeur.

Un opérateur d'affectation stocke une valeur dans une variable. Saisissez un nom pour la variable, suivi du signe égal (=), qui est l'opérateur d'affectation. La valeur est alors stockée dans la variable. Par exemple, saisissez ceci dans le shell interactif :

```
>>> spam = 15
>>>
```

La boîte `spam` contient maintenant la valeur `15` comme le montre la figure 1-3.

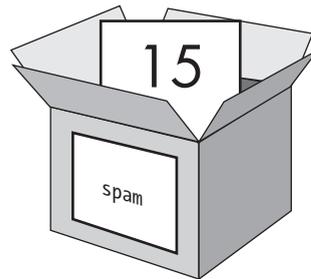


FIGURE 1-3 Les variables sont comme des boîtes qui peuvent contenir des valeurs.

Lorsque vous appuyez sur **Entrée**, vous n'obtenez pas de résultat. Avec Python, vous savez que l'instruction était correcte si aucun message d'erreur ne s'affiche. L'invite apparaît afin de vous indiquer que vous pouvez saisir l'instruction suivante.

Contrairement aux expressions, les affectations sont des instructions qui ne renvoient pas de valeur. C'est pour cela qu'aucune valeur n'apparaît dans le shell interactif lorsque vous saisissez `spam=15`. Si vous avez du mal à faire la différence entre les expressions et les affectations, gardez à l'esprit que les expressions renvoient une valeur. Tout autre type d'instruction est une affectation.

Les variables stockent des valeurs et non pas des expressions. Prenons, par exemple, les expressions `spam=10+5` et `spam=10+7-2`. L'éva-

luation donne le même résultat dans les deux cas et l'opérateur d'affectation stocke la valeur 15 dans la variable `spam`.

Un nom de variable est correct lorsqu'il décrit les données contenues. Imaginez que vous changez de maison et que vous marquez `Trucs` sur tous vos cartons de déménagement. Vous ne retrouverez jamais rien ! Les noms de variables, `spam`, `eggs` et `bacon` sont des noms utilisés pour les variables dans cet ouvrage.

La première fois qu'une variable est utilisée avec un opérateur d'affectation, Python la crée. Pour connaître sa valeur, saisissez son nom dans le shell interactif.

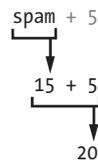
```
>>> spam = 15
>>> spam
15
```

L'expression `spam` est évaluée et on obtient son contenu : 15.

Vous pouvez aussi utiliser des variables dans les expressions. Essayez de saisir ce qui suit dans le shell interactif :

```
>>> spam = 15
>>> spam + 5
20
```

Vous avez indiqué que la valeur de la variable `spam` est 15. Donc, saisir `spam+5` revient à saisir `15+5`. Voici les étapes montrant comment l'expression `spam+5` est évaluée :



Vous ne pouvez pas utiliser une variable avant qu'un opérateur d'affectation ne la crée. Sinon, Python va renvoyer l'erreur `NameError` parce que la variable n'existe pas. Si vous n'orthographiez pas correctement le nom d'une variable, vous obtenez le même message d'erreur :

```
>>> spam = 15
>>> spma
Traceback (most recent call last):
File "<pysshell#8>", line 1, in <module> spma
NameError: name 'spma' is not defined
```

L'erreur s'affiche parce qu'il existe une variable `spam` mais pas de variable `spma`.

Vous modifiez la valeur stockée dans une variable en saisissant un autre opérateur d'affectation. Par exemple, saisissez ceci dans le shell interactif :

```
>>> spam = 15
>>> spam + 5
20
>>> spam = 3
>>> spam + 5
8
```

Lorsque vous saisissez une première fois `spam+5`, le résultat donne `20` parce que vous aviez stocké la valeur `15` dans `spam`. Toutefois, lorsque vous avez saisi `spam=3`, la valeur `15` a été remplacée ou écrasée avec la valeur `3`. Lorsque vous saisissez à nouveau `spam+5`, le résultat est `8`. Lorsque vous donnez une nouvelle valeur à une variable, c'est comme si vous retiriez l'ancienne de la boîte pour en placer une autre, comme le montre la figure 1-4.

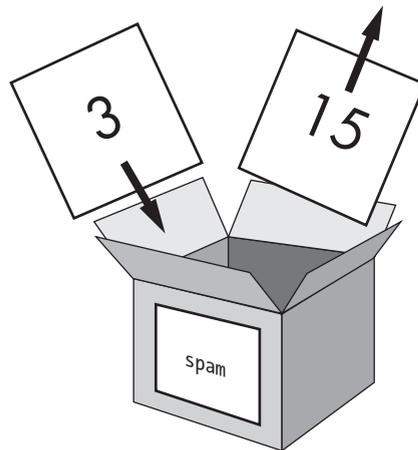


FIGURE 1-4 La valeur 15 est remplacée par la valeur 3.

Vous pouvez même utiliser la valeur contenue dans `spam` pour la modifier :

```
>>> spam = 15
>>> spam = spam + 5
20
```

L'opérateur d'affectation `spam=spam+5` dit « la nouvelle valeur de `spam` sera la valeur actuelle plus cinq ». Pour ajouter plusieurs fois la valeur `5` à `spam`, saisissez ce qui suit dans le shell interactif :

```
>>> spam = 15
>>> spam = spam + 5
>>> spam = spam + 5
>>> spam = spam + 5
>>> spam
30
```

Dans cet exemple, vous donnez initialement à `spam` la valeur 15. À la ligne suivante, vous ajoutez 5 ; `spam` vaut donc maintenant 20. En répétant cela trois fois, `spam` prend 30 comme valeur.

Jusqu'à présent, nous avons utilisé une seule variable, mais vous en créerez autant que nécessaire pour vos programmes. Par exemple, donnons des valeurs différentes à deux variables nommées `truc` et `bidule` :

```
>>> truc = 20
>>> bidule = 15
```

Maintenant, la variable `truc` vaut 20 et la variable `bidule` vaut 15. Chacune des valeurs se trouve dans sa propre boîte, comme le montre la figure 1-5.

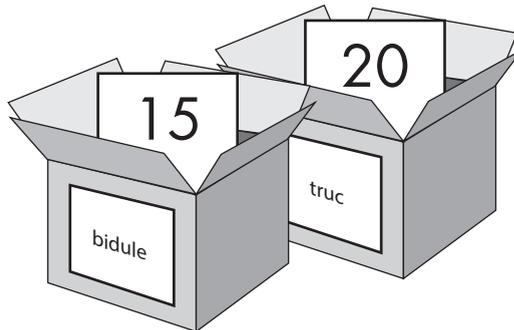


FIGURE 1-5 Les variables `truc` et `bidule` stockent chacune une valeur.

Saisissez `spam=truc+bidule` dans le shell interactif, puis affichez la nouvelle valeur de `spam` :

```
>>> spam = truc + bidule
>>> spam
35
```
