SIMON MONK

Programmez un Raspberry Pi™

Initiation avec Python

Traduit de l'américain
par Dominique Maniez

Copyright © 2014 Dunod.

DUNOD

L'édition originale de cet ouvrage a été publiée en anglais par McGraw-Hill Global Education Holdings, LLC, sous le titre *Programming the Raspberry Pi™, Getting Started with Python*,
ISBN : 978-0-07-180783-7.

Copyright © 2013 by The McGraw Hill Companies. All rights reserved.

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

Le pictogramme qui figure ci-contre mérite une explication. Son objet est d'alerter le lecteur sur la menace que représente pour l'avenir de l'écrit, particulièrement dans le domaine de l'édition technique et universitaire, le développement massif du photocopillage.
Le Code de la propriété intellectuelle du 1er juillet 1992 interdit en effet expressément la photocopie à usage collectif sans autorisation des ayants droit. Or, cette pratique s'est généralisée dans les établissements d'enseignement supérieur, provoquant une baisse brutale des achats de livres et de revues, au point que la possibilité même pour les auteurs de créer des œuvres nouvelles et de les faire éditer correctement est aujourd'hui menacée.
Nous rappelons donc que toute reproduction, partielle ou totale, de la présente publication est interdite sans autorisation de l'auteur, de son éditeur ou du Centre français d'exploitation du droit de copie (CFC, 20, rue des Grands-Augustins, 75006 Paris).



© Dunod, Paris, 2014 pour la traduction française
ISBN 978-2-10-070867-3

Le Code de la propriété intellectuelle n'autorisant, aux termes de l'article L. 122-5, 2° et 3° a), d'une part, que les « copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective » et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, « toute représentation ou reproduction intégrale ou partielle faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause est illicite » (art. L. 122-4).
Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles L. 335-2 et suivants du Code de la propriété intellectuelle.

Copyright © 2014 Dunod.

TABLE DES MATIÈRES

CHAPITRE		PAGE
	Avant-propos	**1**
	Remerciements	**2**
1	**Introduction**	**3**
	1.1 Qu'est-ce que le Raspberry Pi ?	4
	1.2 Que peut-on faire avec un Raspberry Pi ?	5
	1.3 Tour d'horizon du Raspberry Pi	5
	1.4 Installation du Raspberry Pi	7
	1.5 Démarrer le système	14
	1.6 Résumé	15
2	**Prise en main**	**17**
	2.1 Linux	18
	2.2 Le bureau	18
	2.3 Internet	20
	2.4 Ligne de commande	21
	2.5 Applications	23
	2.6 Ressources Internet	25
	2.7 Résumé	25
3	**Bases de Python**	**27**
	3.1 IDLE	28
	3.2 Nombres	32
	3.3 Variables	32
	3.4 Boucles for	34
	3.5 Simulation de lancers de dés	35
	3.6 IF	37
	3.7 While	41
	3.8 Résumé	42
4	**Chaînes, listes et dictionnaires**	**43**
	4.1 Théorie des chaînes	44
	4.2 Listes	46
	4.3 Fonctions	49

Copyright © 2014 Dunod.

4.4	Jeu du pendu	51
4.5	Dictionnaires	58
4.6	Tuples	59
4.7	Exceptions	61
4.8	Résumé de fonctions	62
4.9	Résumé	68
5	**Modules, classes et méthodes**	**69**
5.1	Modules	70
5.2	Orientation objet	74
5.3	Définition des classes	74
5.4	Héritage	76
5.5	Résumé	79
6	**Fichiers et Internet**	**81**
6.1	Fichiers	82
6.2	Pickling	87
6.3	Internet	88
6.4	Résumé	90
7	**Interfaces utilisateur graphiques**	**91**
7.1	Tkinter	92
7.2	Hello World	92
7.3	Convertisseur de température	93
7.4	Autres contrôles d'interface utilisateur	97
7.5	Boîtes de dialogue	104
7.6	Menus	106
7.7	Le Canvas	107
7.8	Résumé	108
8	**Programmation de jeux**	**109**
8.1	Pygame	110
8.2	Hello Pygame	110
8.3	Un jeu avec des framboises	112
8.4	Résumé	121
9	**Interfaçage du matériel**	**123**
9.1	Connexions des broches GPIO	124
9.2	Connexion directe aux broches GPIO	125
9.3	Cartes extension	126
9.3	Cartes de prototypage	130
9.4	Arduino et le Pi	132
9.5	Résumé	135

Copyright © 2014 Dunod.

10	**Projet de prototypage d'une horloge**	**137**
	10.1 Matériel nécessaire	138
	10.2 Assemblage du matériel	139
	10.3 Logiciel	141
	10.4 Phase 2	143
	10.5 Résumé	146
11	**Le RaspiRobot**	**147**
	11.1 Matériel nécessaire	148
	11.2 Phase 1 : création d'un robot basique	149
	11.3 Phase 2 : ajout d'un télémètre et d'un écran	155
	11.4 Résumé	160
12	**Pour aller plus loin**	**161**
	12.1 Ressources Linux	162
	12.2 Ressources Python	162
	12.3 Ressources Raspberry Pi	163
	12.4 Autres langages de programmation	164
	12.5 Applications et projets	166
	12.6 Résumé	166
	Annexe : Liste des principales fonctions de Pygame	**167**
	Index	**177**

Copyright © 2014 Dunod.

Copyright © 2014 Dunod.

AVANT-PROPOS

À mes frères, Andrew et Tim Monk, pour leur amour et leur sagesse.

Le Raspberry Pi est vite devenu un phénomène mondial. Les gens commencent à découvrir le potentiel d'un ordinateur polyvalent à 30 euros qui peut servir de station de travail, de *media center*, ou bien encore de contrôleur d'un système domotique.

Cet ouvrage explique en termes simples, à la fois aux non-programmeurs et aux programmeurs qui découvrent le Raspberry Pi, comment écrire des programmes pour le Pi à l'aide du langage populaire Python. En outre, il vous apprend à créer une interface utilisateur graphique et des jeux simples avec le module Pygame.

Les programmes de ce livre sont principalement écrits en Python 3, mais Python 2 est parfois utilisé quand cela est nécessaire pour la disponibilité des modules. Dans cet ouvrage, nous utilisons la distribution Raspbian Wheezy qui est recommandée par la Fondation Raspberry Pi.

Ce livre débute par une introduction au Raspberry Pi qui couvre l'achat des accessoires et l'installation du matériel et du logiciel. Vient ensuite une introduction à la programmation qui permet au fil des chapitres de progresser graduellement. Les concepts sont illustrés par des exemples d'applications qui vous permettent de commencer à programmer votre Raspberry Pi. Trois chapitres sont consacrés à la programmation et à l'utilisation du connecteur GPIO du Raspberry Pi, qui permet à l'appareil d'être relié à des périphériques électroniques externes. Ces chapitres comprennent deux projets : la réalisation d'une horloge à LED et la construction d'un robot complet avec un télémètre à ultrasons. Enfin, l'annexe présente la liste des principales fonctions de Pygame.

Voici une liste des principaux sujets qui sont traités dans cet ouvrage :

- Nombres, variables et concepts de base du langage Python
- Chaînes, listes, dictionnaires et structures de données en Python

Copyright © 2014 Dunod.

- Modules et orientation objet
- Fichiers et Internet
- Interface utilisateur graphique avec Tkinter
- Programmation des jeux avec Pygame
- Interfaçage avec le matériel *via* le connecteur GPIO
- Exemples de projets impliquant du matériel

L'intégralité des listings de code de cet ouvrage est disponible sur http://www.dunod.com/contenus-complementaires/9782100706594.

REMERCIEMENTS

Comme toujours, je remercie Linda pour sa patience et son soutien.

Je souhaite aussi remercier Andrew Robinson et mon fils, Matthew Monk, pour leur relecture technique d'une grande partie de l'ouvrage. Surveillez la sortie du livre d'Andrew sur le Raspberry Pi car je suis certain qu'il sera excellent.

Copyright © 2014 Dunod.

CHAPITRE PAGE

1 INTRODUCTION

1	Introduction	3

1.1 Qu'est-ce que le Raspberry Pi ? 4
1.2 Que peut-on faire avec un Raspberry Pi ? 5
1.3 Tour d'horizon du Raspberry Pi 5
1.4 Installation du Raspberry Pi 7
1.5 Démarrer le système 14
1.6 Résumé 15

2	Prise en main	17
3	Bases de Python	27
4	Chaînes, listes et dictionnaires	43
5	Modules, classes et méthodes	69
6	Fichiers et Internet	81
7	Interfaces utilisateur graphiques	91
8	Programmation de jeux	109
9	Interfaçage du matériel	123
10	Projet de prototypage d'une horloge	137
11	Le RaspiRobot	147
12	Pour aller plus loin	161

Copyright © 2014 Dunod.

Le Raspberry Pi a débuté sa commercialisation à la fin du mois de février 2012 et les sites web des différents fournisseurs où l'on pouvait passer commande ont été immédiatement pris d'assaut. Qu'est-ce que ce petit appareil a de si spécial et pourquoi donc suscite-t-il autant d'intérêt ?

1.1 QU'EST-CE QUE LE RASPBERRY PI ?

Le Raspberry Pi, illustré à la figure 1.1, est un ordinateur qui s'exécute sous le système d'exploitation Linux. Il possède deux ports USB auxquels vous pouvez raccorder un clavier et une souris, et une sortie vidéo HDMI (*High-Definition Multimedia Interface*) à laquelle vous pouvez connecter un téléviseur ou un moniteur. De nombreux moniteurs ne possèdent qu'un connecteur VGA et le Raspberry Pi ne fonctionnera pas avec ce type de sortie. En revanche, si votre moniteur a un connecteur DVI, il existe des adaptateurs HDMI-DVI bon marché.

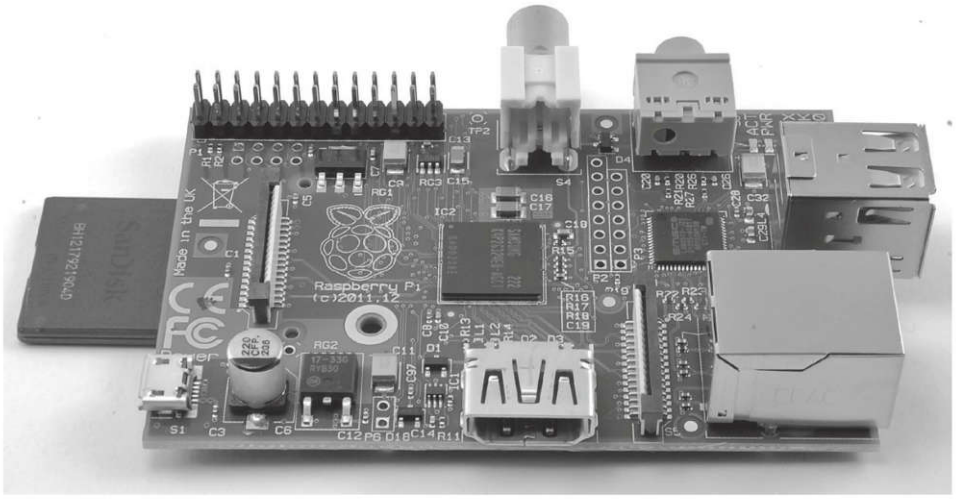


Figure 1.1
Raspberry Pi.

Quand le Raspberry Pi démarre, on obtient le bureau Linux qui est illustré à la figure 1.2. Il s'agit vraiment d'un ordinateur complet, avec une suite bureautique, des fonctionnalités de lecture vidéo, des jeux, etc. Mais il ne s'exécute pas sous Microsoft Windows ; à la place, on a le concurrent open source de Windows, Linux (une distribution Debian), et un environnement de fenêtrage qui s'appelle LXDE.

Copyright © 2014 Dunod.

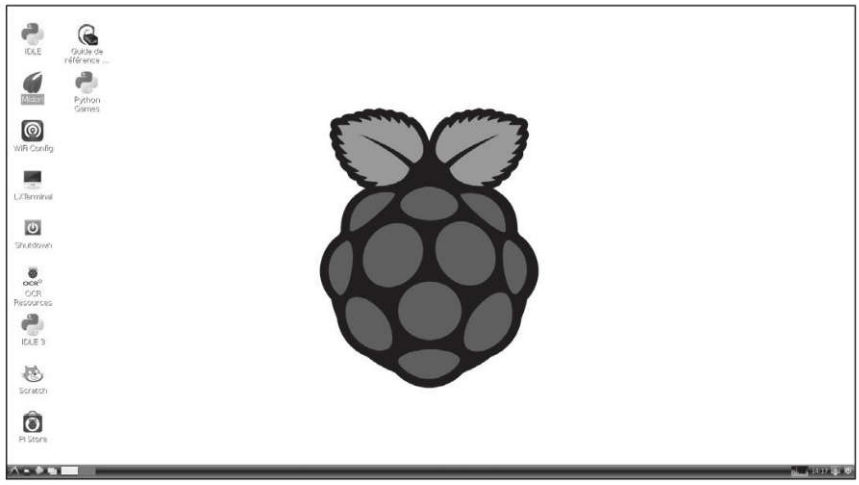


Figure 1.2
Bureau du Raspberry Pi.

Le Raspberry Pi est petit (de la taille d'une carte de crédit) et d'un prix parfaitement abordable (le prix de départ est à moins de 30 euros). Son prix modique s'explique par le fait que certains composants ne sont pas inclus avec la carte ou bien sont optionnels (par exemple, la carte est livrée nue, sans boîtier de protection). Il n'est pas non plus livré avec une alimentation et vous devez donc trouver un chargeur micro-USB de 5 volts (comme celui de votre téléphone, mais sans doute avec une alimentation plus puissante). Une alimentation USB et un câble micro-USB sont souvent utilisés pour cela.

1.2 QUE PEUT-ON FAIRE AVEC UN RASPBERRY PI ?

On peut pratiquement faire avec un Raspberry Pi tout ce que l'on peut faire avec un ordinateur de bureau sous Linux, à quelques exceptions près. Le Raspberry Pi utilise une carte SD à la place d'un disque dur, bien que l'on puisse le brancher à un disque dur USB. On peut modifier des documents bureautiques, surfer sur Internet, et jouer à des jeux (même à des jeux qui ont des graphismes évolués, comme *Quake*). Le prix bas du Raspberry Pi le destine particulièrement à être utilisé comme *media center*. Il peut lire des vidéos et on peut l'alimenter à partir du port USB que l'on trouve sur de nombreux téléviseurs.

1.3 TOUR D'HORIZON DU RASPBERRY PI

La figure 1.3 inventorie les différentes parties d'un Raspberry Pi. La figure illustre un Raspberry Pi modèle B, qui se différencie du modèle A par la présence d'un connecteur RJ-45, ce qui lui permet de se connecter à un réseau.

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

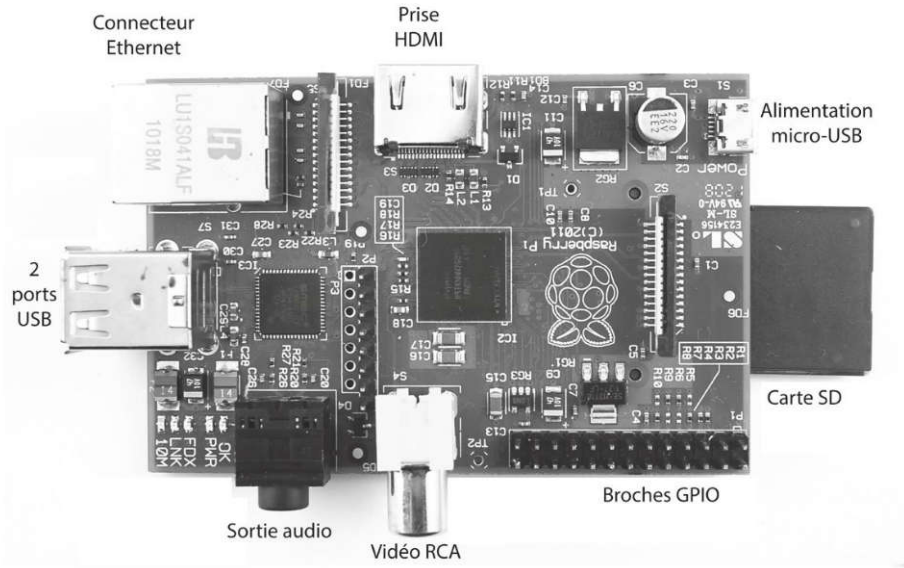


Figure 1.3
Anatomie d'un Raspberry Pi.

Le connecteur Ethernet RJ-45 est illustré dans le coin supérieur gauche de la figure. Si vous disposez d'un hub à la maison, vous pouvez brancher directement le Raspberry Pi sur votre réseau local. À ce sujet, il faut noter que le Raspberry Pi n'a pas le Wi-Fi intégré. Pour disposer d'un réseau sans fil, vous devez brancher un adaptateur USB sans fil qui peut nécessiter l'installation de pilotes supplémentaires.

Sous la prise Ethernet, vous trouverez deux ports USB, l'un au-dessus de l'autre. Vous pouvez brancher un clavier, une souris, ou des disques durs externes, mais vous allez vite être à court de prises. C'est la raison pour laquelle de nombreuses personnes utilisent un hub USB.

Dans le coin inférieur gauche de la figure, vous trouverez un port audio qui fournit un signal analogique stéréo pour un casque ou des haut-parleurs amplifiés. Le connecteur HDMI permet aussi de transférer le son.

À côté de la prise audio se trouve un connecteur vidéo RCA. Vous ne vous servirez probablement pas de ce connecteur, à moins que vous n'utilisiez votre Raspberry Pi avec un vieux téléviseur. Vous emploierez plutôt le connecteur HDMI qui est à l'opposé (au sommet de la figure). La prise HDMI est de meilleure qualité, transporte le son, et peut être connectée à un écran avec une prise DVI à l'aide d'un adaptateur bon marché.

À la droite de la fiche jaune RCA, il y a deux rangées de broches que l'on nomme broches GPIO (*General Purpose Input/Output*, soit entrées / sorties à usage général) et qui permettent au Raspberry Pi

Copyright © 2014 Dunod.

de se connecter à des composants électroniques. Les utilisateurs d'Arduino et des autres microcontrôleurs ne seront pas dépaysés par l'idée de broches GPIO. Au chapitre 11, nous utiliserons ces broches pour transformer le Raspberry Pi en « cerveau » d'un petit robot itinérant pour contrôler ses moteurs. Au chapitre 10, nous utiliserons le Raspberry Pi pour fabriquer une horloge à LED.

Le Raspberry Pi a un emplacement pour carte SD sous le circuit. Cette carte SD doit avoir une taille minimum de 2 Go. Elle contient le système d'exploitation de l'ordinateur ainsi que le système de fichiers qui permet de stocker les documents que l'on crée. La carte SD est une option quand on achète un Raspberry Pi. La préparation de sa propre carte SD s'avère un peu complexe à réaliser et des fournisseurs comme SK Pang, Farnell et RS Components vendent tous des cartes SD prêtes à l'emploi. Comme il n'y a aucun disque intégré au Raspberry Pi, cette carte SD constitue votre véritable ordinateur et vous pouvez donc la retirer et l'insérer dans un autre Raspberry Pi pour retrouver tout votre environnement de travail.

Au-dessus de la carte SD, il y a un port micro-USB qui n'est utilisé que pour fournir l'alimentation du Raspberry Pi. Cela signifie que vous aurez besoin d'une alimentation avec un connecteur micro-USB à son extrémité. Il s'agit du même type de connecteur que celui qui est utilisé par de nombreux téléphones mobiles, notamment la plupart des téléphones Android. Il faut cependant vérifier que votre alimentation peut fournir au moins un courant de 700 mA, sinon votre Raspberry Pi pourra se comporter de façon erratique.

Pour ceux que les spécifications techniques intéressent, la grosse puce carrée au centre de la carte est l'emplacement du cœur du système : il s'agit d'un SOC (*System On a Chip*, ou système sur une puce) de chez Broadcom qui comprend 256 Mo de mémoire ainsi que différents processeurs (graphique et à usage général) qui pilotent le Raspberry Pi.

Vous aurez aussi peut-être remarqué des connecteurs avec un câble plat à côté de la carte SD et entre les prises Ethernet et HDMI. On les utilise respectivement pour un afficheur LCD et une caméra. Il existe déjà un module pour une caméra et on devrait trouver dans un avenir proche un module pour afficheur LCD.

1.4 INSTALLATION DU RASPBERRY PI

Vous vous faciliterez la vie en achetant une carte SD prête à l'emploi et une alimentation quand vous ferez l'acquisition de votre Raspberry Pi, et par la même occasion également un clavier

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

et une souris USB (à moins que vous n'en ayez chez vous qui traînent). Commençons le processus d'installation en examinant ce dont nous aurons besoin et où se le procurer.

Acquisition du matériel nécessaire

Le tableau 1.1 liste ce qu'il vous faut pour obtenir un système Raspberry Pi totalement fonctionnel.

Tableau 1.1
Kit Raspberry Pi.

Article	Distributeur et n° d'article	Informations complémentaires
Raspberry Pi, Modèle A ou B	Farnell (www.farnell.com) Newark (www.newark.com) RS Components (www.rs-components.com)	La différence entre les deux modèles est que le modèle B a une connexion réseau.
Chargeur USB	Farnell : 1734526	Chargeur USB 5 V USB. Doit pouvoir fournir une alimentation de 700 mA (3 W), mais 1 A (5 W) est préférable.
Câble micro-USB	RadioShack : 55048949 Farnell : 2115733 Adafruit : PID 592	
Clavier et souris	Magasin d'informatique	Tout clavier USB fera l'affaire. Les claviers et les souris sans fil qui sont fournis avec leur propre adaptateur USB fonctionnent également.
Télé/moniteur avec prise HDMI	Magasin d'informatique ou d'électronique	
Câble HDMI	Magasin d'informatique ou d'électronique	
Carte SD (préparée)	SK Pang : RSP-2GBSD Newark : 96T7436 Farnell : 2113756	
Adaptateur Wi-Fi*	http://elinux.org/RPi_VerifiedPeripherals#USB_WiFi_Adapters	Elinux.org fournit une liste à jour des adaptateurs Wi-Fi.
Hub USB*	Magasin d'informatique	
Adaptateur HDMI vers DVI*	Newark : 74M6204 Maplins : N24CJ Farnell : 1428271	
Câble Ethernet*	Magasin d'informatique	
Boîtier*	Adafruit, SK Pang, ou Alliedelec.com	
		* Ces articles sont optionnels.

Copyright © 2014 Dunod.

Bloc d'alimentation

La figure 1.4 illustre un bloc d'alimentation USB classique avec un câble USB-A vers micro-USB.



Figure 1.4
Bloc d'alimentation USB.

Vous pouvez utiliser une alimentation d'un ancien lecteur MP3 ou quelque chose de similaire, tant qu'elle a une tension de 5 volts et peut fournir suffisamment de courant. Il est important de ne pas surcharger l'alimentation car elle pourrait chauffer et tomber en panne (voire provoquer un incendie). Par conséquent, l'alimentation doit être capable de fournir au moins 700 mA, mais 1 A donne au Raspberry Pi un peu de marge qui sera utilisée par les périphériques connectés à ses ports USB. Si vous regardez attentivement les spécifications de l'alimentation, vous devriez être capable de déterminer sa capacité. Il arrive que la capacité soit exprimée en watts (W) ; si tel est le cas, elle doit être au moins de 3 W. Si elle indique 5 W, cela correspond à 1 A.

Clavier et souris

Le Raspberry Pi fonctionne avec pratiquement n'importe quels claviers et souris USB. Vous pouvez aussi utiliser la plupart des claviers et des souris sans fil (ceux qui marchent avec un dongle que l'on branche sur un port USB). C'est plutôt une bonne idée, particulièrement si le clavier et la souris fonctionnent sur le même dongle, ce qui permet de n'utiliser qu'un seul port USB. Cela se révélera aussi assez pratique au chapitre 10 quand on utilisera un clavier sans fil pour contrôler un robot piloté par un Raspberry Pi.

Affichage

La présence d'une sortie vidéo RCA sur le Raspberry Pi est, pour être honnête, un peu surprenante car la plupart des utilisateurs vont privilégier le connecteur HDMI qui est plus moderne. Un téléviseur LCD de 22 pouces bon marché fera parfaitement l'affaire pour afficher le contenu du Pi. En fait, vous pouvez vous contenter d'utiliser le téléviseur familial en connectant le Pi à la télé quand vous en avez

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

besoin. Si vous disposez d'un écran d'ordinateur qui n'a qu'un seul connecteur VGA, vous ne pourrez pas l'utiliser sans un convertisseur qui coûte cher. En revanche, si votre moniteur a une sortie DVI, un adaptateur bon marché fera parfaitement l'affaire.

Carte SD

Vous pouvez utiliser votre propre carte SD dans le Raspberry Pi, mais elle devra être préparée avec une image disque d'un système d'exploitation. Cette opération est un peu délicate si bien qu'il est souhaitable de dépenser quelques euros de plus pour acquérir une carte SD qui est déjà prête à l'emploi. Vous pouvez aussi trouver des gens lors des rencontres consacrées au Raspberry Pi qui seront heureux de vous aider à préparer votre carte SD. Les cartes SD préparées par Farnell et RS Components sont d'un prix trop élevé et il est préférable de chercher sur Internet des fournisseurs qui vendent des cartes préparées avec la dernière distribution du système d'exploitation, pour un prix inférieur à ce que vous payeriez pour une carte SD dans une grande surface. Si vous tenez vraiment à préparer vous-même votre carte SD, suivez les instructions disponibles à *www.raspberrypi.org/downloads*. Pour préparer votre propre carte, vous devez avoir un autre ordinateur avec un lecteur de carte SD. La procédure dépend du système d'exploitation de votre machine : Windows, Mac ou Linux. Il existe cependant plusieurs outils pratiques qui automatisent autant que possible le processus. Si vous décidez de préparer vous-même votre carte, suivez soigneusement les instructions de ces outils car il est assez facile de reformater accidentellement le disque dur de votre ordinateur si l'utilitaire le confond avec la carte SD ! Heureusement, ce processus s'améliore au fil des versions de ces différents outils. Le grand avantage de la création de vos propres cartes SD est que vous pouvez en fait choisir parmi une grande variété de distributions Linux. Le tableau 1.2 liste les distributions les plus populaires qui sont disponibles lors de la rédaction de cet ouvrage. Vérifiez sur le site web de la Fondation Raspberry Pi la mise à jour des nouvelles distributions.

Tableau 1.2
Distributions Linux Raspberry Pi.

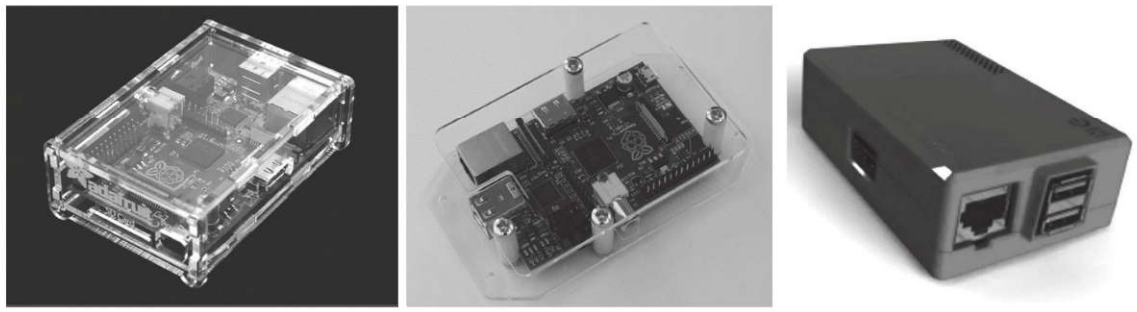
Distribution	Remarques
Raspbian Wheezy	Il s'agit du système d'exploitation « standard » du Raspberry Pi et c'est celui-ci que nous utiliserons dans cet ouvrage. Il utilise le bureau LXDE.
Arch Linux ARM	Cette distribution convient aux experts Linux.
QtonPi	Cette distribution est conçue pour les gens qui développent des programmes graphiques à l'aide du framework Qt5.
Occidentalis	Cette distribution est élaborée par Adafruit et basée sur la Raspbian Wheezy, mais elle intègre des améliorations conçues pour les bidouilleurs électroniciens.

Copyright © 2014 Dunod.

Bien entendu, rien ne vous empêche d'acheter quelques cartes SD et de tester différentes distributions pour voir celle que vous préférez. Néanmoins, si vous débutez en Linux, il est préférable de vous en tenir à la distribution standard Raspbian Wheezy.

Boîtier

Le Raspberry Pi est livré nu, sans protection. Cela permet d'abaisser son prix, mais cela le rend vulnérable à la casse. Il est par conséquent préférable d'acheter un boîtier dès que vous le pouvez ou bien d'en réaliser un vous-même. La figure 1.5 illustre quelques boîtiers prêts à l'emploi qui sont commercialisés.



(a) (b) (c)

Figure 1.5
Boîtiers Raspberry Pi.

Les boîtiers illustrés sont fournis par (a) Adafruit (*www.adafruit.com*), (b) SK Pang (*www.skpang.co.uk*), et (c) ModMyPi (*www.modmypi.com*). Le choix du boîtier dépend de ce vous envisagez de faire avec votre Raspberry Pi. Si vous avez accès à une imprimante 3D, vous pouvez aussi utiliser les plans open source suivants :

- *www.thingiverse.com/thing:23446*
- *www.thingiverse.com/thing:24721*

Vous trouverez également un modèle de boîtier en carton pliable, surnommé le Raspberry Punnet, à l'adresse suivante :

www.raspberrypi.org/archives/1310

Beaucoup de gens s'amusent à construire leur propre boîtier pour leur Raspberry Pi en recyclant toutes sortes de conteneurs, comme de vieux ordinateurs ou des consoles de jeux. On peut même construire un boîtier en Lego. Mon premier boîtier de Raspberry Pi a été créé en perçant des trous dans une boîte conçue pour contenir des cartes de visite (voir la figure 1.6).

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

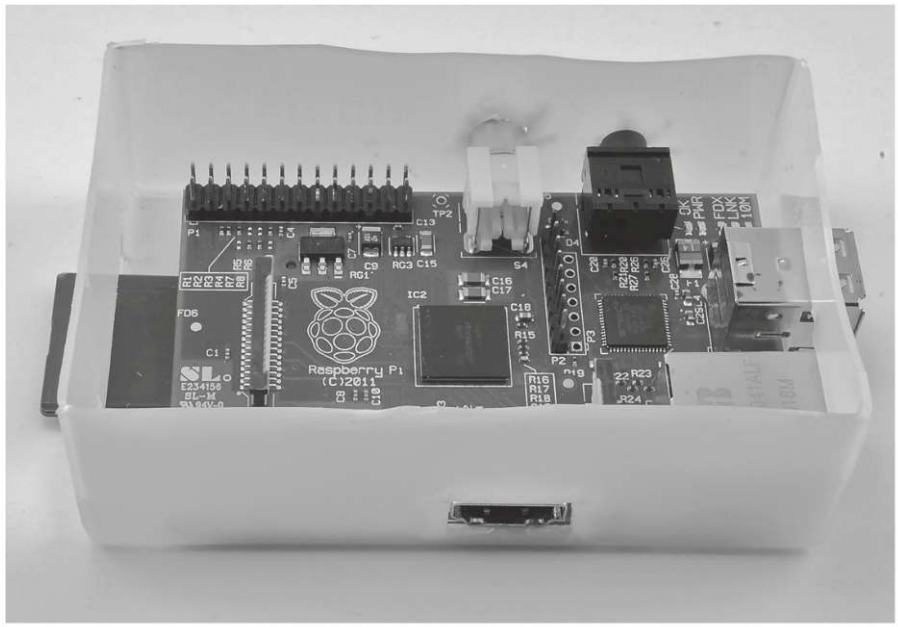


Figure 1.6
Boîtier de Raspberry Pi fait main.

Wi-Fi

Aucun des deux modèles de Raspberry Pi ne prend en charge le Wi-Fi. Si vous souhaitez connecter en Wi-Fi votre Raspberry Pi au réseau, vous avez deux possibilités. La première est d'utiliser un adaptateur sans fil USB qui se branche sur un port USB (voir la figure 1.7). Avec un peu de chance, Linux devrait le reconnaître et permettre de vous connecter immédiatement (ou vous indiquer ce que vous devez faire pour vous connecter).

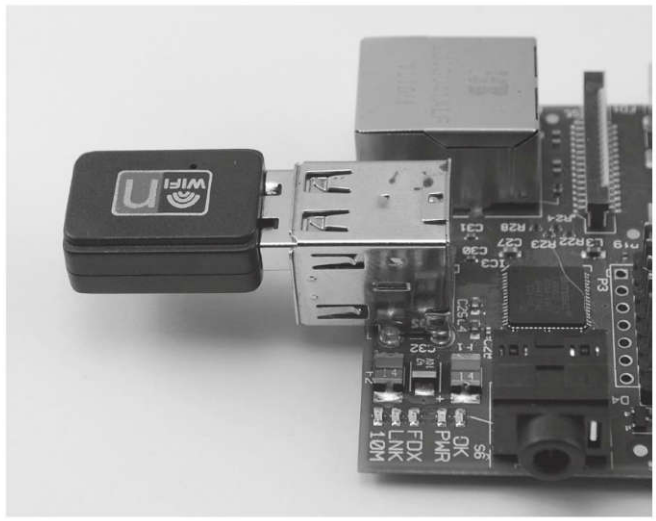


Figure 1.7
Adaptateur Wi-Fi.

Les adaptateurs Wi-Fi référencés dans le tableau 1.1 sont censés fonctionner avec le Raspberry Pi. Cependant, il y a parfois des problèmes avec les pilotes Wi-Fi, si bien qu'il est nécessaire de

Copyright © 2014 Dunod.

vérifier sur le forum et le wiki Raspberry Pi la compatibilité des périphériques. L'autre possibilité consiste à utiliser un pont Wi-Fi avec un Raspberry Pi modèle B. Ces appareils sont en général alimentés par une prise USB et se branchent dans le port Ethernet du Raspberry Pi. Ils sont souvent utilisés par les possesseurs de consoles de jeux qui ont un port Ethernet, mais pas de connexion Wi-Fi. Cette configuration a l'avantage que le Raspberry Pi ne nécessite pas d'équipement spécial.

Hub USB

Comme le Raspberry Pi n'a que deux ports USB disponibles, vous serez rapidement à court de prises USB. Afin d'avoir plus de ports USB, vous devez employer un hub USB (voir la figure 1.8).



Figure 1.8
Hub USB.

Ces hubs sont disponibles dans n'importe quel magasin d'informatique et ils peuvent avoir de trois à huit ports. Assurez-vous que les ports prennent bien en charge la norme USB 2. C'est également une bonne idée d'utiliser un hub USB alimenté de telle sorte que vous ne tirez pas trop de courant du Raspberry Pi.

Connexion de l'ensemble des éléments

Une fois que vous avez tous les éléments dont vous avez besoin, vous pouvez tous les brancher et démarrer votre Raspberry Pi pour la première fois. La figure 1.9 illustre le branchement de chacun des éléments.

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

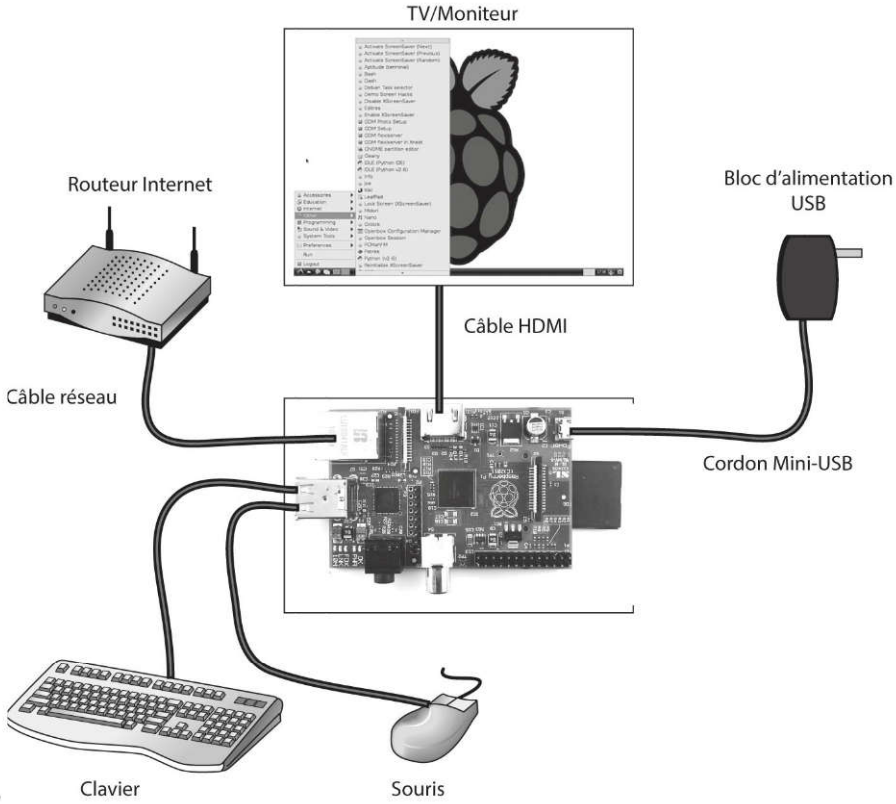


Figure 1.9
Système Raspberry Pi.

Insérez la carte SD, connectez le clavier, la souris et le moniteur au Pi, branchez l'alimentation et vous êtes prêt à démarrer.

1.5 DÉMARRER LE SYSTÈME

La première fois que vous démarrez votre Raspberry Pi, il ne va pas immédiatement démarrer dans un environnement graphique auquel vous êtes habitué (par exemple, celui d'un ordinateur sous Windows). Au lieu de cela, il va s'arrêter pour vous permettre de le configurer (voir la figure 1.10). Il est souhaitable de réaliser un certain nombre de modifications dans la configuration, comme nous allons vous le montrer.

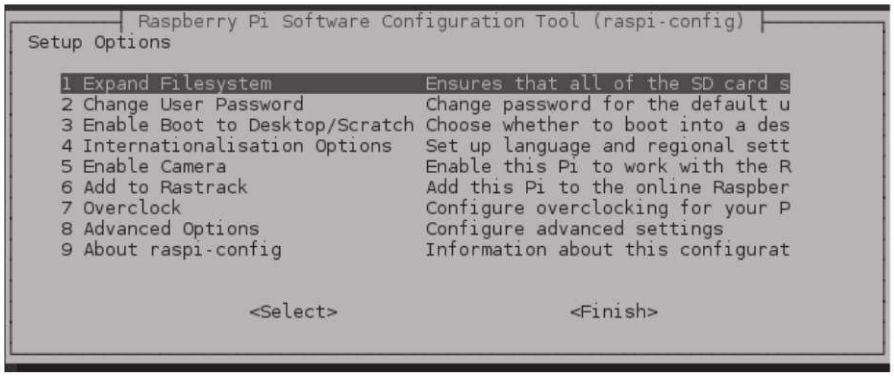


Figure 1.10
Écran de configuration.

Copyright © 2014 Dunod.

Tout d'abord, si votre carte SD a une taille supérieure à 2 Go, le Raspberry Pi n'utilisera que les deux premiers Go sauf si vous sélectionnez l'option **Expand Filesystem** (Étendre le système de fichiers).

Sélectionnez cette option en utilisant les touches de déplacement du curseur et validez en appuyant sur la touche Entrée. Il est aussi intéressant de modifier l'option **Enable Boot to Desktop**. Si elle n'est pas définie à **Desktop Log in as user 'pi' at the graphical desktop** (Se connecter au Bureau en tant qu'utilisateur 'pi' dans l'environnement graphique), vous serez forcé de vous connecter et de démarrer l'environnement de fenêtrage manuellement chaque fois que vous allumez votre Raspberry Pi.

Il est enfin nécessaire de modifier l'option **Internationalisation Options** qui va vous permettre de modifier la langue du système d'exploitation, les paramètres régionaux et la disposition du clavier (voir la figure 1.11).

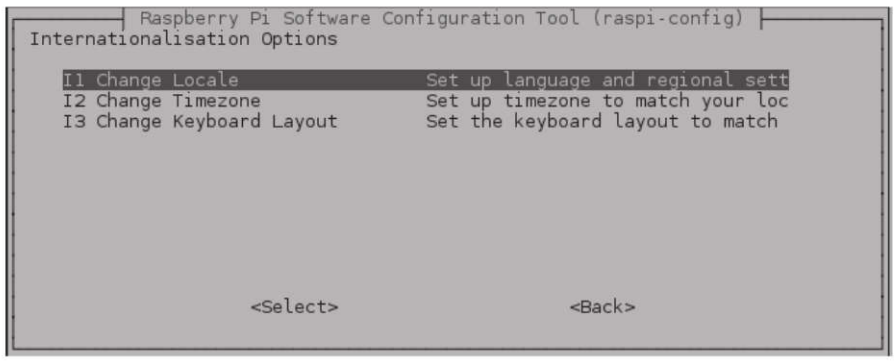


Figure 1.11
Réglages des paramètres pour disposer d'une version en français.

Dans l'option **Change Locale**, choisissez **fr_FR.UTF-8 UTF 8**. Dans l'option **Change Keyboard Layout**, sélectionnez **French**. Si vous voulez modifier les paramètres d'installation ultérieurement, vous pouvez relancer l'utilitaire de configuration du système en saisissant la commande `sudo raspi-config` dans le terminal (voir le chapitre suivant pour l'utilisation du terminal).

1.6 RÉSUMÉ

Maintenant que nous avons installé notre Raspberry Pi et qu'il est prêt à l'emploi, nous pouvons commencer à explorer ses caractéristiques et les bases de Linux.

Copyright © 2014 Dunod.

Copyright © 2014 Dunod.

CHAPITRE PAGE

2 PRISE EN MAIN

1	Introduction	3
2	Prise en main	17

2.1 Linux **18**
2.2 Le bureau **18**
2.3 Internet **20**
2.4 Ligne de commande **21**
2.5 Applications **23**
2.6 Ressources Internet **25**
2.7 Résumé **25**

3	Bases de Python	27
4	Chaînes, listes et dictionnaires	43
5	Modules, classes et méthodes	69
6	Fichiers et Internet	81
7	Interfaces utilisateur graphiques	91
8	Programmation de jeux	109
9	Interfaçage du matériel	123
10	Projet de prototypage d'une horloge	137
11	Le RaspiRobot	147
12	Pour aller plus loin	161

Copyright © 2014 Dunod.

Le Raspberry Pi utilise Linux comme système d'exploitation. Ce chapitre présente Linux et montre comment utiliser le bureau et la ligne de commande.

2.1 LINUX

Linux est un système d'exploitation open source. Ce logiciel a été écrit dans le cadre d'un projet communautaire pour ceux qui recherchaient une alternative au monopole de Microsoft Windows et d'Apple OS X. C'est un système d'exploitation complet, basé sur les mêmes concepts robustes d'UNIX qui est apparu au début de l'informatique. Il a de nombreux partisans fidèles et serviables et s'est transformé en un système d'exploitation puissant et facile à utiliser.

Bien que le système d'exploitation s'appelle Linux, plusieurs distributions (appelées aussi *distros*) Linux différentes ont été produites. Celles-ci comprennent le même système d'exploitation de base, mais sont livrées avec des applications ou des systèmes de fenêtrage différents. Bien qu'il existe de nombreuses distributions, c'est la Wheezy Raspbian qui est recommandée par la Fondation Raspberry Pi.

Si vous n'avez jamais utilisé un autre système que Microsoft Windows, attendez-vous à éprouver quelques frustrations au fur et à mesure de l'étude d'un nouveau système d'exploitation. En effet, les choses fonctionnent un peu différemment sous Linux. Vous pouvez pratiquement tout modifier sous Linux. Le système est ouvert et entièrement sous votre contrôle. Cependant, comme il est dit dans *Spiderman*, de grands pouvoirs impliquent une grande responsabilité. Cela signifie que si vous ne faites pas attention, vous pouvez arriver à endommager l'installation de votre système d'exploitation.

2.2 LE BUREAU

À la fin du premier chapitre, nous nous sommes contentés de démarrer notre Raspberry Pi, de nous identifier et de lancer le système de fenêtrage. La figure 2.1 illustre l'apparence du bureau du Raspberry.

Copyright © 2014 Dunod.

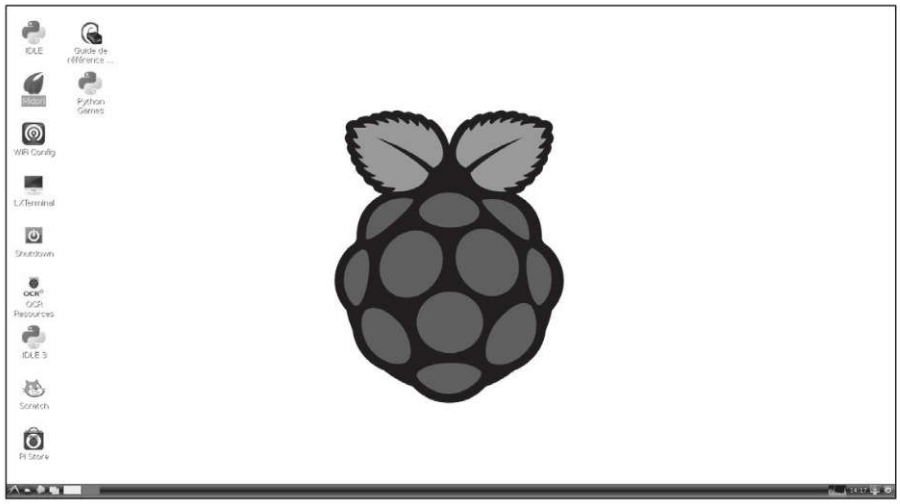


Figure 2.1
Bureau du Raspberry Pi.

Si vous êtes un utilisateur d'ordinateurs sous Windows ou de Mac, vous êtes familiarisé avec le concept de bureau qui représente un dossier au sein d'un système de fichiers et qui agit comme une métaphore de tout ce que vous faites sur l'ordinateur.

Sur le côté gauche de l'écran, vous voyez des icônes qui lancent des applications. En cliquant sur l'icône située à l'extrême gauche de la barre en bas de l'écran, vous faites apparaître des applications et des outils installés sur le Raspberry Pi (un peu comme le menu Démarrer de Microsoft Windows). Nous allons commencer avec le gestionnaire de fichiers qui se trouve dans les accessoires.

Le gestionnaire de fichiers est similaire à l'Explorateur de fichiers de Windows ou au Finder sur Mac. Il permet d'explorer le système de fichiers, de copier et de déplacer des fichiers, et aussi de lancer des fichiers exécutables (applications).

Lorsqu'il démarre, le gestionnaire de fichiers affiche le contenu de votre répertoire personnel (appelé home). Par défaut, dans la distribution que nous utilisons, vous êtes connecté automatiquement sous le nom d'utilisateur **pi** et le mot de passe de cet utilisateur est **raspberry**. La racine de votre répertoire personnel sera par conséquent /home/pi. Notez que comme Mac OS X, Linux utilise des slashs (/) pour séparer les parties d'un nom de répertoire. Par conséquent, / est appelé le répertoire *racine* et /home/ est un répertoire qui contient d'autres répertoires, chaque utilisateur ayant un répertoire à son nom. Comme notre Raspberry Pi ne va avoir qu'un seul utilisateur (appelé pi), le répertoire home ne contiendra donc qu'un seul répertoire appelé pi. Le répertoire en cours est affiché dans la barre d'adresse en haut et vous pouvez saisir directement le chemin d'un autre répertoire pour modifier le répertoire en cours de visualisation, ou bien utiliser la barre de navigation sur le côté. Le contenu du répertoire /home/pi comprend uniquement les répertoires Desktop et python_games.

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

En faisant un double-clic sur Desktop, on ouvre le répertoire Desktop, mais cela n'a pas grand intérêt car il contient juste les raccourcis situés sur le côté gauche du bureau. Si vous ouvrez python_games, vous pourrez voir des jeux que vous pouvez essayer (voir la figure 2.2).

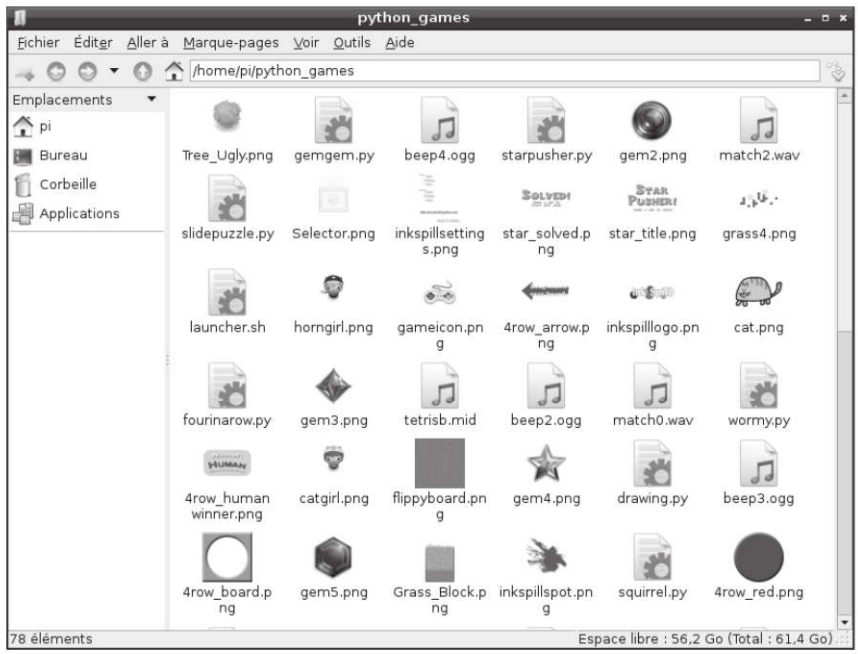


Figure 2.2
Contenu du répertoire python_games affiché dans le gestionnaire de fichiers.

Vous ne devriez pas avoir souvent besoin d'utiliser le système de fichiers en dehors de votre répertoire personnel. Vous devez conserver tous vos documents, vos fichiers musicaux, etc. dans des dossiers de votre répertoire personnel ou bien sur une clé USB.

2.3 INTERNET

Si vous avez un routeur Internet à votre domicile sur lequel vous pouvez brancher n'importe quel appareil relié à Internet à l'aide d'un câble Ethernet, vous ne devriez avoir aucun problème à connecter à Internet votre Raspberry Pi. Votre routeur devrait automatiquement attribuer au Raspberry Pi une adresse IP et lui permettre de se connecter au réseau.

Le Raspberry Pi est livré avec un navigateur web appelé Midori, que vous trouverez dans la section Internet du menu Démarrer. Vous pouvez vérifier que votre connexion fonctionne correctement en démarrant Midori et en allant sur un site web (voir la figure 2.3).

Copyright © 2014 Dunod.



Figure 2.3
Navigateur web Midori.

2.4 LIGNE DE COMMANDE

Si vous êtes un utilisateur de Windows ou de Mac, vous n'avez peut-être jamais utilisé la ligne de commande. En revanche, si vous êtes un utilisateur de Linux, c'est certainement déjà le cas. En fait, si vous êtes un utilisateur de Linux, vous vous êtes sans doute déjà rendu compte qu'il est inutile de lire ce chapitre parce qu'il est un peu trop basique pour vous.

Bien qu'il soit possible d'utiliser un système Linux entièrement via l'interface graphique, on a en général besoin d'entrer des commandes dans l'invite de commandes.

On réalise cela notamment pour installer de nouvelles applications et configurer le Raspberry Pi. Depuis le bouton lanceur (en bas à gauche), ouvrez LXTerminal qui est illustré à la figure 2.4.

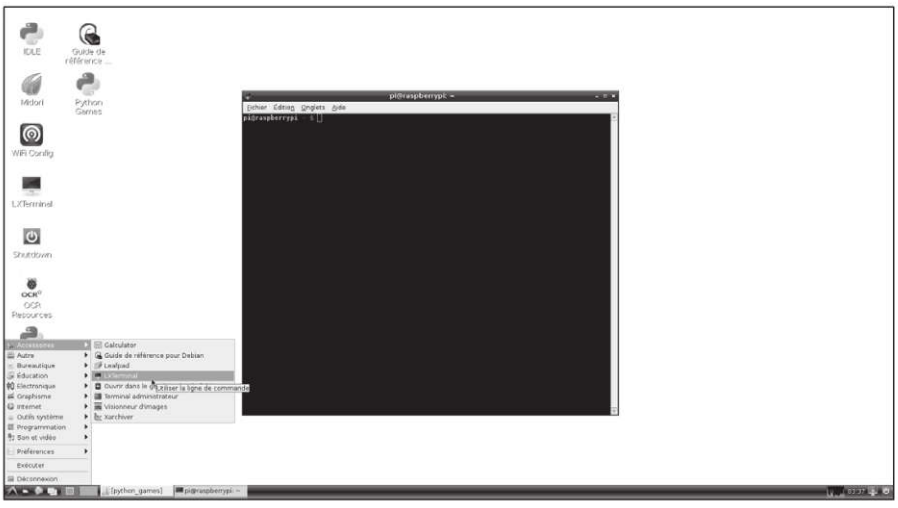


Figure 2.4
Utilitaire de ligne de commande LXTerminal.

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2014 Dunod.

Naviguer avec le Terminal

Quand vous utiliserez la ligne de commande, vous emploierez souvent trois commandes principales. La première commande est `pwd` (abréviation *print working directory*) qui affiche le répertoire de travail dans lequel vous êtes actuellement. Pour tester cette commande, après le signe $ dans la fenêtre de terminal, saisissez **pwd** et appuyez sur la touche Entrée, comme cela est illustré à la figure 2.5.

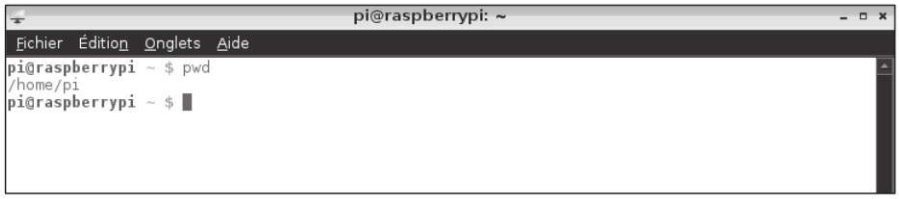


Figure 2.5
Commande pwd.

Comme vous pouvez le constater, nous sommes actuellement dans le répertoire /home/pi. Plutôt que d'imprimer une copie d'écran pour tout ce que nous allons saisir dans le terminal, j'utiliserai la convention suivante : tout ce que vous devez saisir sera précédé par un signe $, comme dans l'exemple suivant :

```
$pwd
```

Les réponses du terminal ne comporteront pas de signe $ en entête. Par conséquent, tout le processus d'exécution de la commande `pwd` ressemblera à ceci :

```
$pwd
/home/pi
```

La prochaine commande courante que nous allons évoquer est `ls` (abréviation de *list*) qui affiche une liste des fichiers et des répertoires du répertoire de travail. Essayez les opérations suivantes :

```
$ls
Desktop
```

Cela nous apprend que le seul élément dans /home/pi est le répertoire Desktop. La dernière commande que nous allons étudier est `cd` (qui est l'abréviation de *change directory*). Cette commande permet de modifier le répertoire de travail en cours. Elle peut changer le répertoire par rapport à l'ancien répertoire de travail ou bien changer complètement de répertoire si vous spécifiez un répertoire entier en commençant par /. Par exemple, la commande suivante change le répertoire de travail actuel en /home/pi/Desktop :

Copyright © 2014 Dunod.

```
$pwd
/home/pi
$cd Desktop
```

Vous pourriez faire la même chose en saisissant ceci :

```
cd /home/pi/Desktop
```

Notez que lorsque vous entrez un répertoire ou un nom de fichier, vous n'avez pas à saisir tout ça. Au lieu de cela, à tout moment, après avoir saisi une partie du nom, vous pouvez appuyer sur la touche TAB. Si le nom du fichier est unique, il sera alors complété automatiquement.

sudo

`sudo` (abréviation de *super-user do*) est une autre commande que vous utiliserez probablement beaucoup. Elle exécute la commande que vous avez saisie comme si vous étiez un super-utilisateur. Vous vous demandez sans doute pourquoi, en tant que seul utilisateur de cet ordinateur, vous n'êtes pas automatiquement super-utilisateur. La réponse est que, par défaut, votre compte d'utilisateur normal (nom d'utilisateur : pi, mot de passe : raspberry) n'a pas de privilèges vous permettant d'atteindre les parties essentielles du système d'exploitation, notamment pour supprimer des fichiers. Pour exécuter de telles commandes, vous devez les faire précéder par `sudo`. Cela ne fait qu'ajouter une couche de protection contre les accidents. Pour les trois commandes que nous avons déjà vues, il n'est pas nécessaire de les préfixer par `sudo`. Cependant, à titre de test, essayez de saisir la commande suivante :

```
sudo ls
```

Cela fonctionne de la même manière que pour la simple commande `ls` ; vous êtes toujours dans le même répertoire de travail. La seule différence est que l'on vous demandera votre mot de passe la première fois que vous utiliserez `sudo`.

2.5 APPLICATIONS

La distribution Raspbian Wheezy pour Raspberry Pi comporte assez peu d'applications, mais vous pouvez en installer autant que vous le voulez. L'installation de nouvelles applications exige à nouveau l'utilisation de la ligne de commande. La commande `apt-get` est utilisée à la fois pour installer et désinstaller des applications. Comme l'installation d'une application requiert souvent des privilèges de super-utilisateur, vous ferez précéder les commandes `apt-get` par `sudo`.

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

La commande `apt-get` utilise une base de données des paquets disponibles qui est mise à jour par Internet, si bien que la première commande `apt-get` que vous devez utiliser est :

```
sudo apt-get update
```

Cette commande met à jour la base de données des paquets. Pour qu'elle fonctionne, vous devez bien entendu être connecté à Internet. Pour installer un paquet particulier, la seule chose que vous devez savoir est le nom de son gestionnaire de paquet. Par exemple, pour installer l'application de traitement de texte Abiword, vous devez saisir :

```
sudo apt-get install abiword
```

Le téléchargement et l'installation peuvent prendre un certain temps, mais à la fin du processus, vous constaterez que vous avez un nouveau dossier dans le menu Démarrer, qui s'appelle Office et qui contient l'application Abiword (voir la figure 2.6).

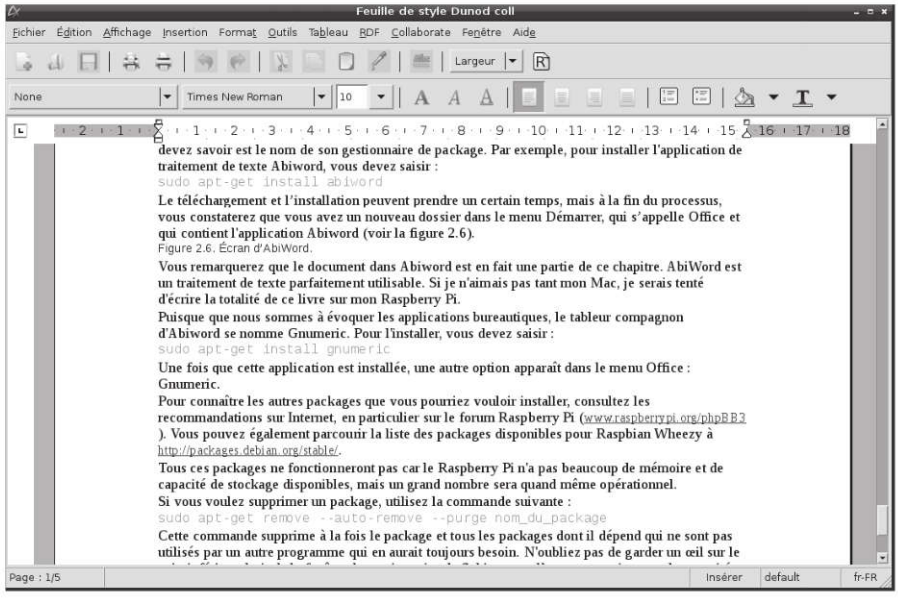


Figure 2.6
Écran d'AbiWord.

Vous remarquerez que le document dans Abiword est en fait une partie de ce chapitre. AbiWord est un traitement de texte parfaitement utilisable. Si je n'aimais pas tant mon Mac, je serais tenté d'écrire la totalité de ce livre sur mon Raspberry Pi.

Puisque nous en sommes à évoquer les applications bureautiques, le tableur compagnon d'Abiword se nomme Gnumeric. Pour l'installer, vous devez saisir :

```
sudo apt-get install gnumeric
```

Une fois que cette application est installée, une autre option apparaît dans le menu Office : Gnumeric.

Copyright © 2014 Dunod.

Pour connaître les autres paquets que vous pourriez vouloir installer, consultez les recommandations sur Internet, en particulier sur le forum Raspberry Pi (*www.raspberrypi.org/phpBB3*). Vous pouvez également parcourir la liste des paquets disponibles pour Raspbian Wheezy à *http://packages.debian.org/stable/*.

Tous ces paquets ne fonctionneront pas car le Raspberry Pi n'a pas beaucoup de mémoire et de capacité de stockage disponibles, mais un grand nombre sera quand même opérationnel.

Si vous voulez supprimer un paquet, utilisez la commande suivante :

```
sudo apt-get remove --auto-remove --purge nom_du_paquet
```

Cette commande supprime à la fois le paquet et tous les paquets dont il dépend qui ne sont pas utilisés par un autre programme qui en aurait toujours besoin. N'oubliez pas de garder un œil sur le coin inférieur droit de la fenêtre du gestionnaire de fichiers car elle vous renseigne sur la quantité d'espace libre disponible.

2.6 RESSOURCES INTERNET

Vous disposez maintenant d'un ordinateur fonctionnel que vous avez sans doute envie d'explorer. Pour vous aider dans cette tâche, il existe de nombreux sites Internet où vous pouvez obtenir des conseils et des recommandations afin de tirer le meilleur parti de votre Raspberry Pi. Le tableau 2.1 énumère certains des sites les plus utiles traitant du Raspberry Pi. Votre moteur de recherche pourra vous en montrer aussi beaucoup d'autres.

Tableau 2.1 **Ressources Internet pour le Raspberry Pi.**

Site
www.raspberrypi.org Page d'accueil de la Fondation Raspberry Pi. Consultez le forum et les FAQ.
www.raspberrypi-spy.co.uk Blog avec des tutoriels utiles.
http://elinux.org/RaspberryPiBoard Principal wiki sur Raspberry Pi. Contient beaucoup d'informations sur le Raspberry Pi, notamment une liste des périphériques compatibles (http://elinux.org/RPi_VerifiedPeripherals).

2.7 RÉSUMÉ

Maintenant que tout est installé et que votre Raspberry Pi est prêt à fonctionner, il est temps de commencer à programmer en Python.

Copyright © 2014 Dunod.

Copyright © 2014 Dunod.

CHAPITRE PAGE

3 BASES DE PYTHON

1	Introduction	3
2	Prise en main	17
3	Bases de Python	27

3.1 IDLE **28**
3.2 Nombres **32**
3.3 Variables **32**
3.4 Boucles for **34**
3.5 Simulation de lancers de dés **35**
3.6 IF **37**
3.7 While **41**
3.8 Résumé **42**

4	Chaînes, listes et dictionnaires	43
5	Modules, classes et méthodes	69
6	Fichiers et Internet	81
7	Interfaces utilisateur graphiques	91
8	Programmation de jeux	109
9	Interfaçage du matériel	123
10	Projet de prototypage d'une horloge	137
11	Le RaspiRobot	147
12	Pour aller plus loin	161

Copyright © 2014 Dunod.

Le moment est venu de commencer à créer nos propres programmes pour le Raspberry Pi. Le langage que nous allons utiliser s'appelle Python. Il a le grand avantage d'être facile à apprendre tout en étant assez puissant pour créer des programmes intéressants, y compris des jeux simples et des programmes qui utilisent des graphiques.

Comme pour la plupart des choses dans la vie, il est nécessaire d'apprendre à marcher avant de pouvoir courir, et nous allons donc commencer par les bases du langage Python.

Un langage de programmation est un langage pour écrire des programmes informatiques. Mais pourquoi est-il nécessaire d'utiliser un langage spécial pour cela ? Ne pouvons-nous pas nous contenter d'utiliser un langage humain ? Comment l'ordinateur utilise-t-il les choses que nous écrivons dans ce langage ?

La raison pour laquelle nous n'utilisons pas le français ou une autre langue humaine vient du fait que les langues humaines sont imprécises et ambiguës. Les langages informatiques utilisent des mots anglais et des symboles, mais de façon très structurée.

3.1 IDLE

La meilleure façon d'apprendre une nouvelle langue est de commencer à l'utiliser tout de suite. Nous allons donc lancer le programme que nous allons utiliser pour nous aider à écrire en Python. Ce programme s'appelle IDLE, et vous le trouverez dans la section Programmation du menu Démarrer. En fait, vous y trouverez plusieurs entrées IDLE. Sélectionnez celle qui s'intitule « IDLE 3 ». La figure 3.1 illustre IDLE et le shell Python.

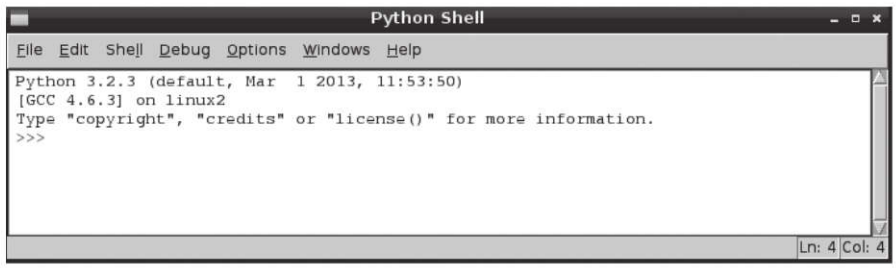


Figure 3.1
IDLE et le shell Python.

Versions de Python

Python 3 a apporté des modifications majeures par rapport à Python 2. Ce livre est basé sur Python 3.1, mais au fur et à mesure que vous progresserez dans l'apprentissage de Python, vous

Copyright © 2014 Dunod.

constaterez que certains modules que vous souhaitez utiliser ne sont pas disponibles pour Python 3.

Shell Python

La figure 3.1 illustre le shell Python. Il s'agit de la fenêtre où vous saisissez les commandes Python et voyez ce qu'elles produisent. Cela est très utile pour réaliser des tests, notamment au cours de votre apprentissage de Python.

Comme dans l'invite de commandes, vous pouvez saisir des commandes après l'invite (dans ce cas, >>>) et la console Python vous montrera ce que cela produit sur la ligne en dessous.

L'arithmétique est quelque chose de naturel pour tous les langages de programmation, et Python ne fait pas exception à cette règle. Pour illustrer notre propos, saisissez 2 + 2 après l'invite du shell Python ; vous devriez voir le résultat (4) sur la ligne en dessous, tel que cela est illustré à la figure 3.2.

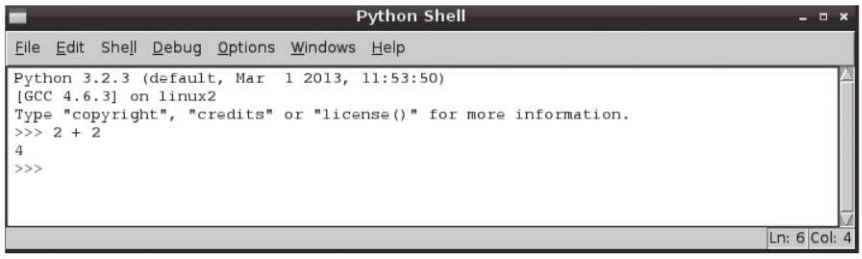


Figure 3.2
Calcul dans le shell Python.

Éditeur

Le shell Python est l'endroit idéal pour expérimenter, mais ce n'est pas le bon emplacement pour écrire un programme. Les programmes Python sont stockés dans des fichiers afin que vous n'ayez pas à les retaper. Un fichier peut contenir une longue liste de commandes d'un langage de programmation, et lorsque vous souhaitez exécuter toutes ces commandes, il faut en fait exécuter ce fichier. La barre de menu au sommet du programme IDLE permet de créer un nouveau fichier. Sélectionnez File, puis New Window dans la barre de menus. La figure 3.3 illustre l'éditeur IDLE avec une nouvelle fenêtre ouverte.

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

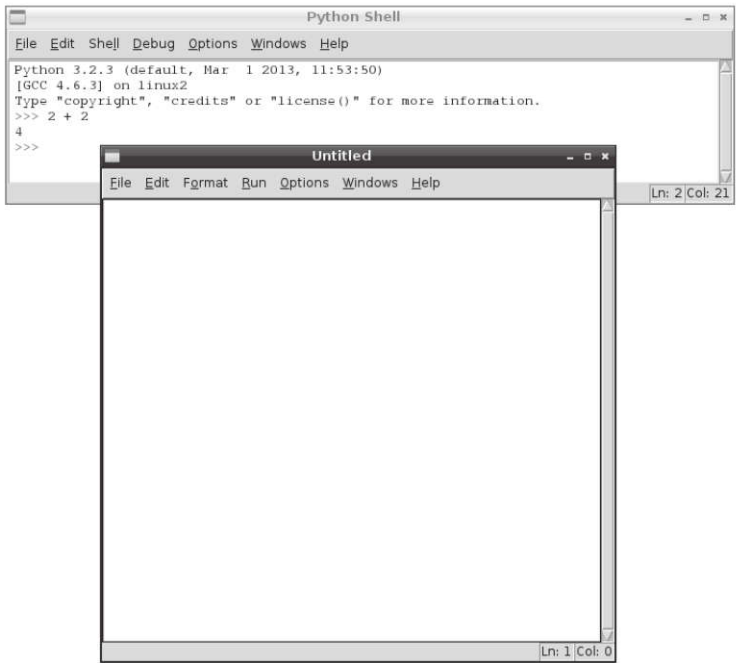


Figure 3.3
Éditeur IDLE.

- Saisissez les deux lignes suivantes de code dans IDLE :

```
print('Hello')
print('World')
```

Vous remarquerez que l'éditeur n'a pas le symbole d'invite >>>. Cela est dû au fait que ce que nous écrivons ici ne sera pas exécuté immédiatement ; au lieu de cela, ce sera simplement stocké dans un fichier jusqu'à ce que nous décidions de l'exécuter. Si vous le souhaitez, vous pouvez utiliser un autre éditeur de texte pour écrire vos programmes, mais l'éditeur IDLE s'intègre bien avec Python. Il possède également une certaine connaissance du langage Python, ce qui peut servir d'aide-mémoire lorsque vous saisissez des programmes.

Nous avons besoin d'un bon emplacement pour conserver tous les programmes Python que nous allons écrire, et nous allons donc ouvrir le gestionnaire de fichiers à partir du menu Démarrer (dans la rubrique Accessoires, sélectionnez Ouvrir dans le gestionnaire de fichiers). Faites un clic droit sur la zone principale et sélectionnez Créer un nouveau, puis Dossier dans le menu contextuel (voir la figure 3.4). Entrez comme nom de dossier **Python**, et appuyez sur la touche Entrée.

Copyright © 2014 Dunod.

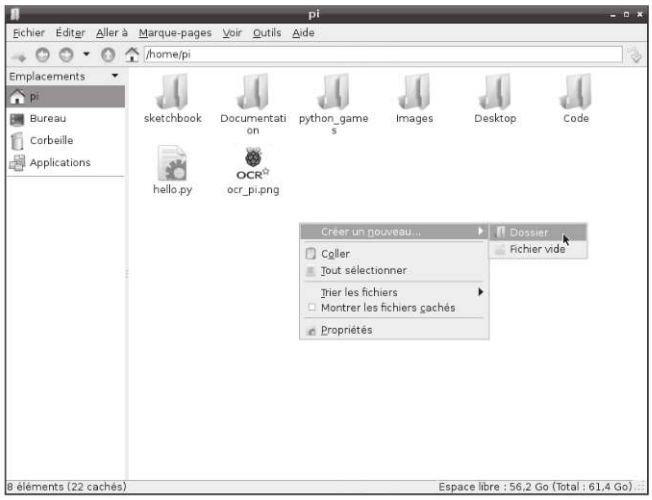


Figure 3.4
Création d'un dossier Python.

Nous devons ensuite revenir à la fenêtre de l'éditeur, puis enregistrer le fichier en utilisant le menu File. Désignez le nouveau répertoire Python et nommez fichier **hello.py**, comme cela est illustré à la figure 3.5.

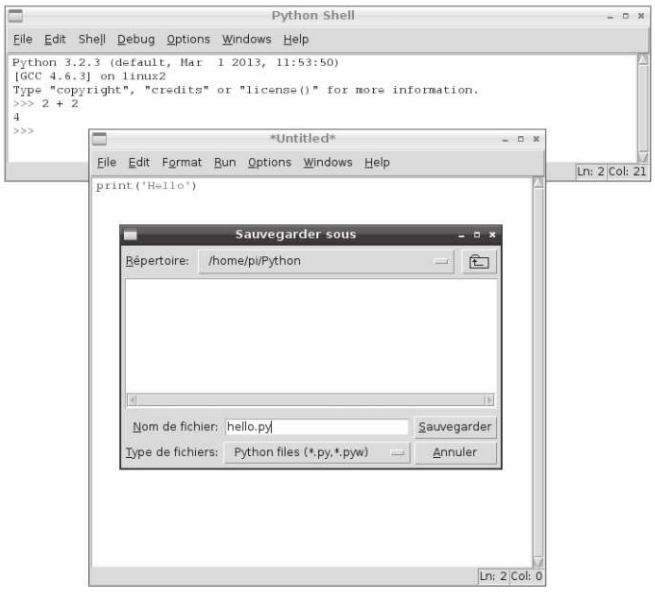


Figure 3.5
Enregistrement du programme.

Pour exécuter le programme et voir ce qu'il fait, allez dans le menu Run et sélectionnez Run Module. Vous devriez voir les résultats de l'exécution du programme dans le shell Python. C'est sans grande surprise que le programme affiche les deux mots *Hello* et *World*, chacun sur sa propre ligne.

Ce que vous saisissez dans le shell Python n'est enregistré nulle part, si bien que lorsque vous quittez IDLE et le redémarrez, tout ce que vous avez saisi dans le shell Python est perdu. Cependant, comme nous avons enregistré notre fichier dans l'éditeur, nous pouvons le charger à tout moment à partir du menu File.

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Note : Pour que ce livre ne se transforme pas en une longue succession de copies d'écran, quand je voudrai que vous saisissiez quelque chose dans le Shell Python, je préfixerai les commandes par >>>. Les résultats apparaîtront alors sur les lignes en dessous.

3.2 NOMBRES

Les nombres jouent un rôle primordial dans la programmation, et l'arithmétique est un des domaines où les ordinateurs excellent. Nous commencerons en faisant des tests avec des nombres, et le meilleur endroit pour ce genre d'expérience est le shell Python. Saisissez ce qui suit dans le Shell Python :

```
>>> 20 * 9 / 5 + 32
68.0
```

Ceci n'est pas beaucoup plus évolué que l'exemple 2 + 2 que nous avons testé précédemment. Toutefois, cet exemple nous enseigne quelques petites choses :

- * signifie multiplier.
- / signifie diviser.
- Python fait la multiplication avant la division, et il fait la division avant l'addition.

Si vous le souhaitez, vous pouvez ajouter quelques parenthèses pour garantir que tout se passe dans le bon ordre :

```
>>> (20 * 9 / 5) + 32
68.0
```

Ces nombres sont tous des nombres entiers (les programmeurs les désignent aussi sous le nom de *integer*). Nous pouvons aussi utiliser un point décimal si nous voulons utiliser des nombres décimaux. En programmation, ces nombres sont appelés *floats*, qui est l'abréviation de *floating point* (virgule flottante).

3.3 VARIABLES

Abordons à présent les variables. Vous pouvez vous représenter une variable comme quelque chose qui a une valeur. C'est un peu comme quand on utilise des lettres à la place des nombres en algèbre. Pour commencer, saisissez le texte suivant :

Copyright © 2014 Dunod.

```
>>> k = 9.0 / 5.0
```

Le signe «égal» assigne une valeur à une variable. La variable doit être du côté gauche et doit être composée d'un seul mot (sans espace) ; toutefois, elle peut être aussi longue que vous le souhaitez et contenir des chiffres et le caractère de soulignement (_). En outre, les caractères peuvent être en majuscules et en minuscules. Ce sont les règles de nommage des variables, mais il y a également des conventions. La différence est que si vous enfreignez les règles, Python se plaindra, alors que si vous ne respectez pas les conventions, les autres programmeurs risquent de se moquer de vous.

Les conventions pour les variables stipulent qu'elles doivent commencer par une lettre minuscule et comporter un trait de soulignement entre les mots (par exemple, nombre_de_poulets). Les exemples du tableau 3.1 vous donnent une idée de ce qui est légal et de ce qui est conventionnel.

Tableau 3.1 **Nommage des variables.**

Nom de la variable	Légal	Conventionnel
x	Oui	Oui
X	Oui	Non
nombre_de_poulets	Oui	Oui
nombre de poulets	Non	Non
nombreDePoulets	Oui	Non
NombreDePoulets	Oui	Non
2beOrNot2b	Non	Non
toBeOrNot2b	Oui	Non

Beaucoup d'autres langages utilisent une convention différente pour le nommage des variables appelée *camel-case* (l'alternance entre les majuscules et les minuscules fait penser aux bosses d'un chameau), où les mots sont séparés en mettant l'initiale de chaque mot (sauf la première) en majuscules (par exemple, nombreDePoulets). Vous verrez parfois cette convention appliquée dans certains exemples de code Python. En fin de compte, si le code est destiné uniquement à votre propre usage, la manière dont la variable est écrite n'a pas vraiment d'importance ; en revanche, si votre code doit être lu par d'autres programmeurs, c'est une bonne idée de s'en tenir aux conventions.

En adhérant aux conventions de nommage des variables, vous facilitez le travail des autres programmeurs Python qui lisent votre code.

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Si vous faites quelque chose que Python n'aime pas ou ne comprend pas, vous obtiendrez un message d'erreur. Essayez d'entrer le texte suivant :

```
>>> 2beOrNot2b = 1
SyntaxError: invalid syntax
```

Il s'agit d'une erreur car vous essayez de définir une variable qui commence par un chiffre, ce qui n'est pas autorisé.

Dans un exemple précédent, nous avons assigné une valeur à la variable `k`. On peut voir quelle valeur elle a en saisissant simplement `k` :

```
>>> k
1.8
```

Python se souvient de la valeur de k, si bien que nous pouvons maintenant l'utiliser dans d'autres expressions. Si l'on reprend notre première expression, nous pourrions saisir ce qui suit :

```
>>> 20 * k + 32
68.0
```

3.4 BOUCLES FOR

L'arithmétique, c'est bien joli, mais cela ne rend pas un programme très excitant. Par conséquent, dans cette section, nous allons étudier les *boucles* qui permettent à Python d'accomplir une tâche plusieurs fois au lieu d'une seule. Dans l'exemple suivant, vous devrez saisir plusieurs lignes de Python. Lorsque vous appuyez sur la touche Entrée et passez à la deuxième ligne, vous remarquerez que Python est en attente. Il n'a pas immédiatement exécuté ce que vous avez saisi parce qu'il sait que vous n'avez pas encore terminé. Le caractère « : » à la fin de la ligne signifie qu'il y a encore quelque chose à faire.

Chacune de ces tâches supplémentaires doit figurer sur une ligne indentée. Par conséquent, dans le programme suivant, au début de la deuxième ligne, vous devez appuyer sur la touche TAB une fois, puis saisir `print(x)`. Pour faire exécuter ce programme de deux lignes, appuyez sur la touche Entrée deux fois après la saisie de la deuxième ligne.

Copyright © 2014 Dunod.

```
>>> for x in range(1, 10):
        print(x)
1
2
3
4
5
6
7
8
9
>>>
```

Ce programme affiche les nombres de 1 à 9 au lieu des nombres de 1 à 10. La commande `range` n'inclut pas le dernier nombre de la plage, mais elle comprend le premier.

Vous pouvez vérifier cela simplement en demandant au programme d'afficher des valeurs sous la forme d'une liste :

```
>>> list(range(1, 10))
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

La ponctuation nécessite ici une petite explication. Les parenthèses servent à délimiter ce que l'on appelle des *paramètres*. Dans cet exemple, `range` comporte deux paramètres : `from(1)` et `to(10)`, séparés par une virgule.

La commande `for in` comporte deux parties. Après le mot `for`, il doit y avoir un nom de variable. Cette variable recevra une nouvelle valeur à chaque tour de boucle. Par conséquent, la première fois elle recevra la valeur 1, puis au tour suivant la valeur 2 et ainsi de suite. Après le mot `in`, Python s'attend à voir quelque chose qui équivaut à une liste d'éléments. Dans notre exemple, il s'agit d'une liste de nombres de 1 à 9.

La commande `print` accepte également un argument qui affiche les valeurs dans le shell Python. À chaque tour de boucle, la valeur suivante de `x` est affichée.

3.5 SIMULATION DE LANCERS DE DÉS

En nous basant sur ce que nous venons d'apprendre sur les boucles, nous allons écrire un programme qui simule dix lancers d'un dé.

Pour ce faire, vous aurez besoin de savoir comment générer un nombre aléatoire. Testez le code suivant dans le shell :

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

```
>>> import random
>>> random.randint(1,6)
2
```

Essayez de saisir la deuxième ligne plusieurs fois, et vous verrez que vous obtiendrez différents nombres aléatoires compris entre 1 et 6. La première ligne importe une bibliothèque qui indique à Python comment générer des nombres. Vous en apprendrez beaucoup plus sur les bibliothèques plus loin dans ce livre, mais pour l'instant, vous avez juste besoin de savoir que nous devons exécuter cette commande avant de commencer à utiliser la commande `randint` qui renvoie un nombre aléatoire.

Note : Je suis assez laxiste sur l'utilisation du terme commande ici. À proprement parler, les éléments comme `randint` sont en fait des fonctions, et non pas des commandes, mais nous y reviendrons plus tard.

Maintenant que vous savez générer un nombre aléatoire unique, vous devez combiner cela avec vos connaissances sur les boucles pour afficher 10 nombres aléatoires en une seule fois. Comme cela dépasse ce qui peut être raisonnablement saisi dans le Shell Python, nous allons donc utiliser l'éditeur IDLE.

Vous pouvez soit saisir les exemples imprimés dans ce livre, soit télécharger tous les exemples Python sur le site des éditions Dunod (www.dunod.com) sur la page consacrée à cet ouvrage. Chaque exemple de programmation comporte un numéro. Ainsi, ce programme est stocké dans le fichier 3_1_de.py, qui peut être chargé dans l'éditeur IDLE.

À ce stade, vous devez saisir les exemples pour vous aider à assimiler les concepts. Ouvrez une nouvelle fenêtre de l'éditeur IDLE, saisissez ce qui suit, puis enregistrez votre travail :

```
#03_01_de
import random
for x in range(1, 11):
    nombre_aleatoire = random.randint(1, 6)
    print(nombre_aleatoire)
```

La première ligne commence par un caractère #. Cela indique que la ligne entière n'est pas du code, mais simplement un commentaire destiné à renseigner ceux qui lisent le programme. Les commentaires comme celui-ci constituent un moyen utile d'ajouter des informations supplémentaires à un programme, sans interférer avec le bon fonctionnement du programme. En d'autres termes, Python ignorera toute ligne commençant par le caractère #.

Copyright © 2014 Dunod.

Maintenant, dans le menu Run, sélectionnez Run Module. Le résultat devrait ressembler à ce qui est illustré à la figure 3.6, où vous pouvez voir le résultat dans le shell Python derrière la fenêtre de l'éditeur.

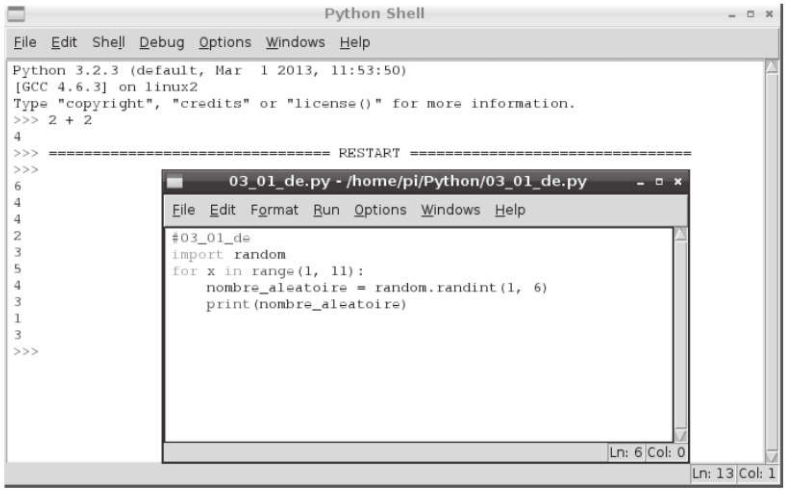


Figure 3.6
Simulation de lancers d'un dé.

3.6 IF

Nous allons à présent améliorer notre programme en lançant deux dés à la fois ; quand les deux dés sont jetés, si nous obtenons un total de 7 ou de 11 ou n'importe quel double, nous affichons un message après le lancer. Saisissez ou chargez le programme suivant dans l'éditeur IDLE :

```
#03_02_double_de
import random
for x in range(1, 11):
    lancer_1 = random.randint(1, 6)
    lancer_2 = random.randint(1, 6)
    total = lancer_1 + lancer_2
    print(total)
    if total == 7:
       print('Sept!')
    if total == 11:
       print('Onze!')
    if lancer_1 == lancer_2:
       print('Vous avez fait un double !')
```

Lorsque vous exécutez ce programme, vous devriez voir quelque chose qui ressemble à ceci :

```
7
Sept!
8
```

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

```
5
4
Vous avez fait un double !
7
Sept!
7
Sept!
12
Vous avez fait un double !
6
8
6
```

La première chose à remarquer dans ce programme, c'est que maintenant deux nombres aléatoires compris entre 1 et 6 sont générés (un pour chacun des deux dés). Une nouvelle variable, `total`, est affectée à la somme des deux lancers.

Vient ensuite la partie intéressante : la commande `if`. La commande `if` est immédiatement suivie par une condition (dans le premier cas, `total == 7`). Il y a ensuite le caractère deux points (:), et les lignes suivantes ne seront exécutées par Python que si la condition est vraie. À première vue, vous pourriez penser qu'il y a une erreur dans la condition car on utilise == au lieu de =. Le double signe égal est utilisé lors de la comparaison de deux éléments pour voir s'ils sont égaux, alors que le signe égal tout seul est utilisé lors de l'assignation d'une valeur à une variable.

Le deuxième `if` n'est pas indenté et il sera donc exécuté, que la condition du premier `if` soit vraie ou fausse. Ce deuxième `if` est comme le premier, sauf que nous recherchons un total de 11. Le dernier `if` est un peu différent car il compare deux variables (throw_1 et throw_2) pour voir si elles sont identiques, ce qui indique qu'un double a été fait.

Maintenant, la prochaine fois que vous allez jouer au *Monopoly* et qu'il manquera les dés, vous savez quoi faire : démarrer votre Raspberry Pi et écrire un petit programme.

Mise en garde : On appelle indentation du code le fait que le texte des programmes soit décalé vers la droite par l'appui sur la touche tabulation dans l'éditeur de programmes. En Python, l'indentation du code ne sert pas qu'à faire joli : elle a une signification bien précise et une mauvaise indentation de votre code peut donc engendrer des erreurs de programmation. Si vous voulez approfondir les règles de syntaxe de Python, vous pouvez consulter le tutoriel suivant : http://python.developpez.com/cours/TutoSwinnen/?page=page_5.

Copyright © 2014 Dunod.

Comparaisons

Pour tester si deux valeurs sont identiques, nous utilisons ==. C'est ce qu'on appelle un *opérateur de comparaison*. Nous pouvons utiliser les opérateurs de comparaison qui sont listés dans le tableau 3.2.

Tableau 3.2 **Opérateurs de comparaison.**

Comparaison	Description	Exemple
==	Égalité	total == 11
!=	Non-égalité	total != 11
>	Supérieur à	total > 10
<	Inférieur à	total < 3
>=	Supérieur ou égal à	total >= 11
<=	Inférieur ou égal à	total <= 2

Vous pouvez tester ces opérateurs de comparaison dans le shell Python. Voici un exemple :

```
>>> 10 > 9
True
```

Dans ce cas, nous avons dit fondamentalement à Python, « est-ce que 10 est supérieur à 9? ». Python a répondu, « True » (vrai). Maintenant demandons à Python si 10 est inférieur à 9 :

```
>>> 10 > 9
False
```

Question de logique

On ne peut pas prendre en faute la logique. Quand Python nous dit « True » ou « False », il n'affiche pas simplement un message à notre attention. `True` et `False` sont des valeurs spéciales appelées *valeurs logiques*. Toute condition utilisée avec une instruction `if` est transformée en une valeur logique par Python lorsqu'il décide s'il faut exécuter la ligne suivante.

Ces valeurs logiques peuvent être combinées un peu de la même façon que l'on effectue des opérations arithmétiques avec les opérateurs plus et moins. Cela n'a aucun sens d'ajouter les valeurs logiques `True` et `True`, mais cela a parfois du sens de dire `True and True`. Par exemple, si nous voulons afficher un message chaque fois que la somme du lancer de nos deux dés est comprise entre 5 et 9, nous pouvons écrire quelque chose comme ceci :

```
if total >= 5 and total <= 9:
      print('Pas mal !')
```

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Outre l'opérateur logique `and`, nous pouvons utiliser `or`. Nous pouvons également utiliser `not` pour transformer la valeur `True` en `False` et vice versa, comme cela est illustré par le code suivant :

```
>>> not True
False
```

Ainsi, une autre façon de dire la même chose serait d'écrire ce qui suit :

```
if not (total < 5 or total > 9):
      print('Pas mal !')
```

Exercice

Essayez d'intégrer le test précédent dans le programme de simulation de lancer de dés. Pendant que vous y êtes, ajoutez deux instructions supplémentaires `if` : une qui affiche « Bon lancer ! » si le lancer est supérieur à 10 et l'autre qui affiche « Malchanceux ! », si le lancer est inférieur à 4. Testez votre programme. Si vous êtes coincé, vous pouvez consulter la solution qui se trouve dans le fichier 3_3_double_de_solution.py.

Else

Dans l'exemple précédent, vous constaterez que certains lancers peuvent générer plusieurs messages. N'importe laquelle des lignes contenant une instruction `if` peut afficher un message supplémentaire si la condition est vraie. Parfois, on souhaite un comportement légèrement différent de telle sorte que si la condition est vraie, on fasse une chose, et dans le cas contraire, on fasse une autre chose. En Python, on utilise l'instruction `else` pour y parvenir :

```
>>> a = 7
>>> if a > 7:
      print('a est grand')
else:
      print('a est petit')
a est petit
>>>
```

Dans cet exemple, un seul des deux messages sera affiché.

Il existe une autre variante qui est `elif` (abréviation de *else if*). Nous pouvons ainsi modifier l'exemple précédent pour qu'il y ait trois clauses qui s'excluent mutuellement :

Copyright © 2014 Dunod.

```
>>> a = 7
>>> if a > 9:
      print('a est très grand')
elif a > 7:
      print('a est assez grand')
else:
      print('a est petit')
a est petit
>>>
```

3.7 WHILE

`while` est une autre commande de boucle qui fonctionne un peu différemment de la boucle `for`. La commande `while` ressemble un peu à une commande `if` en ce sens où elle est immédiatement suivie par une condition. Dans ce cas, la condition sert à rester dans la boucle. En d'autres termes, le code à l'intérieur de la boucle sera exécuté jusqu'à ce que la condition ne soit plus vraie. Cela signifie que vous devez veiller à ce que la condition à un moment donné devienne fausse ; dans le cas contraire, la boucle continue indéfiniment et votre programme semblera planté.

Pour illustrer l'utilisation de `while`, le programme de lancer de dés a été modifié de telle sorte qu'il continue de tourner jusqu'à ce que l'on fasse un double six :

```
#03_04_double_de_while
import random
lancer_1 = random.randint(1, 6)
lancer_2 = random.randint(1, 6)
while not (lancer_1 == 6 and lancer_2 == 6):
   total = lancer_1 + lancer_2
   print(total)
   lancer_1 = random.randint(1, 6)
   lancer_2 = random.randint(1, 6)
print('Vous avez fait un double six !')
```

Si vous testez ce programme, vous verrez qu'il fonctionne. Toutefois, il est un peu plus grand qu'il ne devrait l'être. Nous devons répéter les lignes suivantes deux fois, une fois avant le début de la boucle et une fois à l'intérieur de la boucle :

```
lancer_1 = random.randint(1, 6)
lancer_2 = random.randint(1, 6)
```

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Un principe bien connu de la programmation est que l'on ne se répète pas. Bien que cela ne soit pas bien grave dans un petit programme comme celui-ci, quand les programmes deviennent plus complexes, vous devez éviter la situation où le même code est utilisé à plusieurs endroits car cela rend le programme difficile à maintenir.

Nous pouvons utiliser la commande `break` pour raccourcir le code et le rendre un peu moins répétitif. Lorsque Python rencontre la commande `break`, il interrompt la boucle. Voici une autre version du programme qui utilise cette fois la commande `break` :

```
#03_05_double_de_while_break
import random
while True:
    lancer_1 = random.randint(1, 6)
    lancer_2 = random.randint(1, 6)
    total = lancer_1 + lancer_2
    print(total)
    if lancer_1 == 6 and lancer_2 == 6:
        break
print('Vous avez fait un double six !')
```

La condition pour rester dans la boucle est définie de manière permanente à la valeur `True`. La boucle continue jusqu'à ce qu'elle rencontre la commande `break` qui ne se produit que lorsqu'on a lancé un double six.

3.8 RÉSUMÉ

Vous pouvez à présent vous amuser avec IDLE et tester des choses dans le shell Python. Je vous recommande vivement de tenter de modifier les exemples de ce chapitre, de changer le code pour voir comment cela affecte le cours des programmes. Dans le prochain chapitre, nous examinerons d'autres types de données que les nombres que vous pouvez exploiter en Python.

Copyright © 2014 Dunod.

CHAPITRE PAGE

4 CHAÎNES, LISTES ET DICTIONNAIRES

1	Introduction	3
2	Prise en main	17
3	Bases de Python	27
4	Chaînes, listes et dictionnaires	43

4.1 Théorie des chaînes **44**
4.2 Listes **46**
4.3 Fonctions **49**
4.4 Jeu du pendu **51**
4.5 Dictionnaires **58**
4.6 Tuples **59**
4.7 Exceptions **61**
4.8 Résumé de fonctions **62**
4.9 Résumé **68**

5	Modules, classes et méthodes	69
6	Fichiers et Internet	81
7	Interfaces utilisateur graphiques	91
8	Programmation de jeux	109
9	Interfaçage du matériel	123
10	Projet de prototypage d'une horloge	137
11	Le RaspiRobot	147
12	Pour aller plus loin	161

Copyright © 2014 Dunod.

On aurait pu rajouter les fonctions au titre de ce chapitre, mais il était déjà assez long comme ça. Dans ce chapitre, nous allons d'abord étudier et manipuler les différentes manières de représenter les données et d'ajouter une structure à vos programmes Python. Nous rassemblerons ensuite toutes les connaissances apprises pour créer un jeu simple de pendu où vous devez deviner un mot choisi au hasard en demandant si ce mot contient une lettre particulière.

Le chapitre se termine par une section de référence qui vous dit tout ce qu'il faut savoir sur les fonctions intégrées les plus utiles dans le domaine des mathématiques, des chaînes, des listes et des dictionnaires.

4.1 THÉORIE DES CHAÎNES

En programmation, une *chaîne* (qui est la traduction du terme anglais *string*) est une séquence de caractères que l'on utilise dans un programme. En Python, pour créer une variable contenant une chaîne, vous pouvez simplement utiliser l'opérateur = pour faire l'assignation, mais au lieu d'assigner à la variable une valeur numérique, on lui attribue une valeur de chaîne en mettant cette valeur entre guillemets, comme dans l'exemple suivant :

```
>>> nom_livre = 'Programmer le Raspberry Pi'
```

Si vous voulez voir le contenu d'une variable, vous pouvez saisir simplement le nom de la variable dans le shell Python ou bien exécuter la commande `print`, tout comme nous l'avons fait avec les variables qui contiennent une valeur numérique :

```
>>> nom_livre
'Programmer le Raspberry Pi'
>>> print(nom_livre)
Programmer le Raspberry Pi
```

Il y a une subtile différence entre les résultats de chacune de ces méthodes. Si vous entrez simplement le nom de la variable, Python encadre sa valeur par des guillemets simples afin de signifier qu'il s'agit d'une chaîne. En revanche, lorsque vous utilisez `print`, Python affiche simplement la valeur.

Note : Vous pouvez également utiliser des guillemets doubles pour définir une chaîne, mais la convention est d'utiliser des guillemets simples, à moins d'avoir une raison pour employer des guillemets doubles (par exemple, si la chaîne que vous souhaitez créer comporte une apostrophe).

Copyright © 2014 Dunod.

On détermine le nombre de caractères d'une chaîne grâce à la fonction suivante :

```
>>> len(nom_livre)
26
```

Vous pouvez trouver le caractère situé à un endroit particulier de la chaîne de la manière suivante :

```
>>> nom_livre[1]
'r'
```

Deux choses sont à noter ici : tout d'abord, l'utilisation des crochets à la place des parenthèses qui sont employées pour les paramètres et, deuxièmement, la position qui commence à 0 et non pas à 1. Pour trouver la première lettre de la chaîne, vous devez procéder de la sorte :

```
>>> nom_livre[0]
'P'
```

Si vous mettez un numéro qui est trop grand pour la longueur de la chaîne, vous verrez ceci :

```
>>> nom_livre[100]
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
IndexError: string index out of range
>>>
```

Il s'agit d'une erreur, et c'est un moyen qu'emploie Python pour nous dire que nous avons fait quelque chose de mal. Plus précisément, la partie «string index out of range» (indice de chaîne en dehors de la plage) du message indique que nous avons essayé d'accéder à quelque chose que nous ne pouvons pas atteindre. Dans cet exemple, il s'agit du centième élément d'une chaîne qui compte seulement 24 caractères.

Vous pouvez extraire des fragments d'une grande chaîne dans une chaîne plus petite :

```
>>> nom_livre[0:10]
'Programmer'
```

Le premier nombre entre crochets est la position de départ de la chaîne que nous voulons extraire et le deuxième nombre n'est pas, comme vous vous en doutez, la position du dernier caractère que vous souhaitez, mais la position du dernier caractère plus 1.

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

À titre d'expérience, essayez d'extraire le mot *Raspberry* du titre du livre. Si vous ne spécifiez pas le deuxième numéro, vous obtiendrez par défaut la fin de la chaîne :

```
>>> nom_livre[14:]
'Raspberry Pi'
```

De la même manière, si vous ne spécifiez pas le premier nombre, il sera égal par défaut à 0.

Enfin, vous pouvez également regrouper des chaînes grâce à l'opérateur +. Voici un exemple :

```
>>> nom_livre + ' de Simon Monk'
'Programmer le Raspberry Pi de Simon Monk'
```

4.2 LISTES

Quand nous avons étudié précédemment dans cet ouvrage les nombres, nous avons vu qu'une variable ne pouvait contenir qu'un seul nombre. Il est cependant parfois utile qu'une variable puisse contenir une liste de nombres ou chaînes, voire un mélange des deux, ou même une liste de listes. La figure 4.1 vous aidera à visualiser ce qui se passe lorsqu'une variable est une liste.

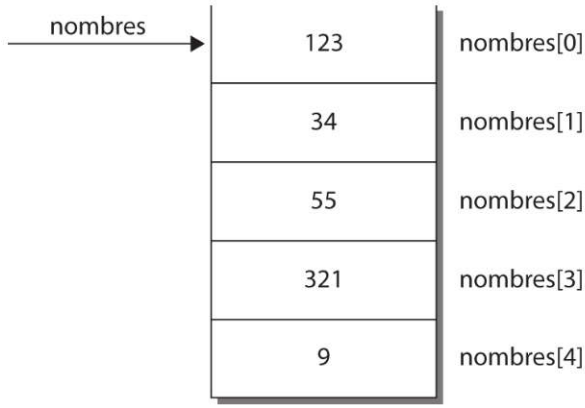


Figure 4.1
Liste.

Note : On emploie aussi le terme « tableau » (en anglais, *array*) à la place de liste.

Les listes se comportent plutôt comme des chaînes. Après tout, une chaîne n'est qu'une liste de caractères. L'exemple suivant vous montre comment créer une liste. Vous noterez la manière dont `len` fonctionne sur les listes, mais aussi sur les chaînes :

Copyright © 2014 Dunod.

```
>>> nombres = [123, 34, 55, 321, 9]
>>> len(nombres)
5
```

Des crochets sont utilisés pour indiquer une liste, et comme avec les chaînes nous pouvons employer des crochets pour trouver un élément individuel dans une liste ou pour extraire une liste à partir d'une liste plus grande :

```
>>> nombres[0]
123
>>> nombres[1:3]
[34, 55]
```

Vous pouvez utiliser = pour assigner une nouvelle valeur à l'un des éléments de la liste :

```
>>> nombres[0] = 1
>>> nombres
[1, 34, 55, 321, 9]
```

Ceci modifie le premier élément de la liste (élément 0) qui passe de la valeur 123 à la valeur 1. Comme avec des chaînes, vous pouvez regrouper deux listes en utilisant l'opérateur + :

```
>>> plus_de_nombres = [5, 66, 44]
>>> nombres + plus_de_nombres
[1, 34, 55, 321, 9, 5, 66, 44]
```

Si vous souhaitez trier la liste, vous pouvez procéder de la manière suivante :

```
>>> nombres.sort()
>>> nombres
[1, 9, 34, 55, 321]
```

Pour supprimer un élément d'une liste, on utilise la commande `pop`. Si vous ne spécifiez pas l'argument de `pop`, elle se contente d'enlever le dernier élément de la liste et de le retourner.

```
>>> nombres
[1, 9, 34, 55, 321]
>>> nombres.pop()
321
>>> nombres
[1, 9, 34, 55]
```

Si vous spécifiez un nombre comme argument de la commande `pop`, il s'agit de la position de l'élément à supprimer. Voici un exemple :

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

```
>>> nombres
[1, 9, 34, 55]
>>> nombres.pop(1)
9
>>> nombres
[1, 34, 55]
```

Vous pouvez également insérer un élément dans une liste à une position particulière. La fonction `insert` accepte deux arguments. Le premier est la position à laquelle on insère l'élément, et le second argument est l'élément à insérer.

```
>>> nombres
[1, 34, 55]
>>> nombres.insert(1, 66)
>>> nombres
[1, 66, 34, 55]
```

Lorsque vous souhaitez connaître la longueur d'une liste, vous utilisez `len(nombres)`, mais lorsque vous souhaitez trier la liste ou extraire un élément de la liste, vous mettez un point après la variable contenant la liste puis exécutez la commande de la manière suivante :

```
nombres.sort()
```

Ces deux styles différents sont le résultat de ce que l'on appelle l'*orientation objet* que nous allons étudier dans le prochain chapitre. Les listes peuvent être transformées en des structures assez complexes qui contiennent d'autres listes et un mélange de différents types (nombres, chaînes et valeurs logiques). La figure 4.2 montre la structure de la liste qui résulte de la ligne de code suivante :

```
>>> grande_liste = [123, 'hello', ['liste interne', 2, True]]
>>> grande_liste
[123, 'hello', ['liste interne', 2, True]]
```

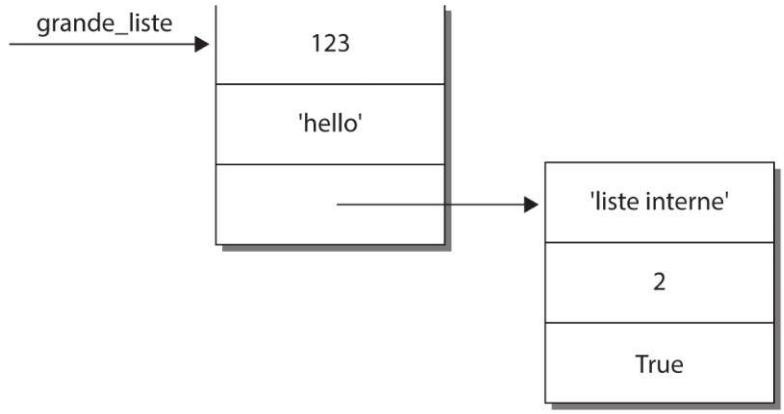


Figure 4.2
Liste complexe.

Copyright © 2014 Dunod.

Vous pouvez combiner ce que vous savez des listes avec des boucles et écrire un programme court qui crée une liste et affiche ensuite chaque élément de la liste sur une ligne distincte :

```
#04_01_liste_et_for
liste = [1, 'un', 2, True]
for element in liste:
    print(element)
```

Voici le résultat de ce programme :

```
1
un
2
True
```

4.3 FONCTIONS

Lorsque vous écrivez des petits programmes comme ceux que nous avons conçus jusqu'à présent, ils ne remplissent pas de véritable fonction, et il n'y a donc pas besoin de les décomposer. Il est en effet assez simple de voir ce qu'ils tentent de réaliser. Au fur et à mesure que les programmes grossissent, les choses deviennent cependant plus compliquées et il devient nécessaire de découper vos programmes en unités appelées *fonctions*. Quand nous serons encore plus avancés dans l'étude de la programmation, nous nous pencherons sur les moyens encore meilleurs de structurer nos programmes à l'aide des classes et des modules.

Beaucoup de choses que j'ai désignées comme des *commandes* sont en réalité des fonctions intégrées à Python, comme `range` et `print`.

Le plus gros problème dans le développement de logiciels, quel que soit leur type, est de gérer la complexité. Les meilleurs programmeurs écrivent des logiciels qui sont faciles à étudier et à comprendre et nécessitent très peu d'explications supplémentaires. Les fonctions sont un outil essentiel dans la création de programmes faciles à comprendre qui peuvent ainsi être modifiés sans difficulté ni risque de semer la pagaille.

Une fonction est un peu comme un programme à l'intérieur d'un programme. Nous pouvons l'utiliser pour englober une séquence de commandes que nous voulons réaliser. Une fonction qui a été définie peut être appelée depuis n'importe quel endroit d'un programme et contient ses propres variables et sa propre liste de commandes. Lorsque les commandes ont été exécutées, on retourne juste après l'emplacement du code à partir duquel la

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

fonction a été appelée. À titre d'exemple, nous allons créer une fonction qui prend une chaîne comme argument et ajoute le mot *SVP* à la fin de la chaîne. Chargez le fichier suivant (ou mieux encore, saisissez-le dans une nouvelle fenêtre de l'éditeur), puis exécutez-le pour voir ce qui se passe :

```
#04_02_fonction_politesse
def parler_poliment(phrase):
    phrase_polie = phrase + ' SVP'
    return phrase_polie

print(parler_poliment('Passez-moi le sel'))
```

La fonction commence par le mot-clé `def` qui est suivi du nom de la fonction; cette dernière respecte les mêmes conventions de nommage que les variables. Après cela, on trouve à l'intérieur des parenthèses les paramètres qui sont séparés par des virgules s'il y en a plusieurs. La première ligne doit se terminer par le signe deux-points.

À l'intérieur de la fonction, on utilise une nouvelle variable appelée `phrase_polie` qui prend le paramètre passé à la fonction et ajoute « SVP » (y compris l'espace à gauche). Cette variable ne peut servir qu'à l'intérieur de la fonction.

La dernière ligne de la fonction est une commande `return`. Cela spécifie la valeur que la fonction doit retourner au code qui l'a appelée. C'est comme les fonctions trigonométriques comme `sin`, où vous passez en paramètre un angle et récupérez en retour un nombre. Dans ce cas, c'est la valeur de la variable `phrase_polie` qui est retournée.

Pour utiliser la fonction, on a juste besoin de spécifier son nom et de lui fournir les arguments appropriés. Une valeur de retour n'est pas obligatoire, et certaines fonctions vont se contenter de faire quelque chose au lieu de faire un calcul. Par exemple, nous pourrions écrire une fonction plutôt absurde qui afficherait « Hello » un certain nombre de fois :

```
#04_03_bonjour_n
def dire_bonjour(n):
    for x in range(0, n):
        print('Bonjour')

dire_bonjour(5)
```

Vous connaissez à présent les bases de ce dont nous aurons besoin pour écrire notre jeu du pendu. Il vous reste encore d'autres choses à apprendre, mais nous y reviendrons plus tard.

Copyright © 2014 Dunod.

4.4 JEU DU PENDU

Le pendu est un jeu où l'on doit deviner des mots et auquel on joue généralement avec du papier et un crayon. Un joueur choisit un mot et trace un tiret pour chaque lettre du mot, l'autre joueur devant deviner le mot. Le joueur qui doit deviner le mot choisit une lettre, et si cette lettre ne fait pas partie du mot, il perd une vie et une partie de la potence est dessinée. Si la lettre choisie figure dans le mot, toutes les occurrences de la lettre sont inscrites à la place des tirets.

Nous allons laisser Python choisir un mot et nous devrons le deviner. Au lieu de dessiner une potence, Python va se contenter de nous indiquer le nombre de vies qu'il nous reste.

Vous allez commencer par fournir une liste de mots à Python dans laquelle il devra choisir. Cela ressemble à un travail pour une liste de chaînes :

```
mots = ['poulet', 'chien', 'chat', 'souris', 'grenouille']
```

Le programme doit ensuite sélectionner un de ces mots au hasard. Nous pouvons écrire une fonction qui réalise cela et écrire un programme pour la tester :

```
#04_04_mots_pendu
import random

mots = ['poulet', 'chien', 'chat', 'souris', 'grenouille']

def choisir_un_mot():
    mot_position = random.randint(0, len(mots) - 1)
    return mots[mot_position]

print(choisir_un_mot())
```

Exécutez ce programme plusieurs fois pour vérifier qu'il sélectionne des mots différents dans la liste.

Il s'agit d'un bon début, mais cette fonction doit s'intégrer dans la structure du jeu. Il faut ensuite définir une nouvelle variable appelée `vies_restantes`. Il s'agit d'un entier que nous pouvons initialiser à 14 et diminuer d'une unité à chaque fois qu'une supposition erronée est réalisée par le joueur. Ce type de variable est appelé variable *globale*, car contrairement aux variables définies dans les fonctions, nous pouvons y accéder à partir de n'importe quel endroit du programme.

Il nous faut aussi écrire une fonction, appelée `jeu`, qui contrôle le déroulement du jeu. Nous savons ce que la fonction `jeu` doit réa-

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

liser, mais nous n'avons pas encore tous les détails. Par conséquent, nous pouvons écrire la fonction `jeu` puis les autres fonctions qui la composent et qu'elle appellera, comme `obtenir_supposition` et `traiter_supposition`, mais aussi utiliser la fonction `choisir_un_mot` que nous venons de créer. Voici la structure générale du jeu :

```
def jouer():
    mot = choisir_un_mot()
    while True:
        supposition = obtenir_supposition(mot)
        if traiter_supposition(supposition, mot):
            print('Vous avez gagné ! Bravo !')
            break
        if vies_restantes == 0:
            print('Vous êtes pendu !')
            print('Le mot était : ' + mot)
            break
```

Le jeu du pendu implique tout d'abord que l'on choisisse un mot. Il y a ensuite une boucle qui se poursuit jusqu'à ce que le mot soit deviné (`traiter_supposition` renvoie `True`) ou que le nombre de `vies_restantes` soit réduit à zéro. À chaque tour de boucle, on demande à l'utilisateur une autre supposition.

Nous ne pouvons pas exécuter ce programme pour le moment parce que les fonctions `obtenir_supposition` et `traiter_supposition` n'existent pas encore. Cependant, on peut écrire ce qu'on appelle leurs *stubs* ce qui nous permettra au moins de tester notre fonction `play`. Les stubs ne sont que des squelettes de fonctions qui ne font pas grand-chose, mais qui permettent de faire tourner un programme en attendant que les versions complètes des fonctions soient rédigées.

```
def obtenir_supposition(mot):
    return 'a'

def traiter_supposition(supposition, mot):
    global vies_restantes
    vies_restantes = vies_restantes -1
    return False
```

Le stub de la fonction `obtenir_supposition` simule simplement le fait que le joueur propose toujours la lettre *a* et le stub de la fonction `traiter_supposition` suppose toujours que le joueur s'est trompé si bien que la variable `vies_restantes` diminue de 1 et la fonction retourne `False` pour indiquer que le joueur n'a pas gagné.

Copyright © 2014 Dunod.

Le stub de `traiter_supposition` est un peu plus compliqué. La première ligne indique à Python que la variable `vies_restantes` est une variable globale. Sans cette ligne, Python supposerait qu'il s'agit d'une nouvelle variable qui serait locale à la fonction. Le stub diminue ensuite le nombre de vies restantes et retourne `False` pour indiquer que le joueur n'a pas encore gagné. Pour finir, nous ajouterons des vérifications pour voir si le joueur a deviné toutes les lettres du mot.

Ouvrez le fichier 4_5_jeu_pendu.py et exécutez-le. Vous obtiendrez un résultat similaire à ceci :

```
Vous êtes pendu !
Le mot était : chien
```

Les 14 suppositions ont été épuisées très rapidement et Python nous a dit le mot qu'il fallait deviner et que nous avions perdu.

Pour terminer le programme, il nous reste à remplacer les stubs des fonctions par des fonctions réelles, et nous allons commencer avec la fonction `obtenir_supposition` :

```
def obtenir_supposition(mot):
    afficher_mot_avec_tirets(mot)
    print('Vies restantes : ' + str(vies_restantes))
    supposition = input(' Devinez une lettre ou le mot entier ? ')
    return supposition
```

La première tâche de la fonction `obtenir_supposition` est d'indiquer au joueur l'état actuel de ses suppositions (quelque chose qui ressemble à «g-e-----e») à l'aide de la fonction `afficher_mot_avec_tirets`. Pour le moment, nous utiliserons un autre stub de fonction. On dit ensuite au joueur le nombre de vies qu'il lui reste. Vous noterez que comme nous voulons ajouter un nombre (vies_restantes) à la chaîne « Vies restantes », la variable numérique doit être convertie en une chaîne à l'aide de la fonction intégrée `str`.

La fonction intégrée `input` affiche le message qui lui est passé en paramètre sous la forme d'une invite de commande, puis retourne ce que l'utilisateur a saisi. Notez qu'en Python 2, la fonction `input` est appelée `raw_input`. Par conséquent, si vous décidez d'utiliser Python 2, modifiez cette fonction en employant `raw_input`.

Enfin, la fonction `obtenir_supposition` retourne tout ce que l'utilisateur a saisi. Le stub de la fonction `afficher_mot_avec_tirets` nous rappelle simplement que nous avons quelque chose à écrire ultérieurement :

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

```
def afficher_mot_avec_tirets(mot):
    print('afficher_mot_avec_tirets : pas encore fait')
```

Ouvrez le fichier 4_6_pendu_obtenir_supposition.py et exécutez-le. Vous obtiendrez un résultat similaire à ceci :

```
afficher_mot_avec_tirets : pas encore fait
Vies restantes : 14
Devinez une lettre ou le mot entier ? x
afficher_mot_avec_tirets : pas encore fait
Vies restantes : 13
Devinez une lettre ou le mot entier ? y
afficher_mot_avec_tirets : pas encore fait
Vies restantes : 12
Devinez une lettre ou le mot entier ?
```

Entrez les suppositions jusqu'à ce que toutes vos vies soient perdues pour vérifier que vous obtenez le message « Vous êtes pendu ! ».

Ensuite, nous pouvons écrire la version correcte de la fonction `afficher_mot_avec_tirets`. Cette fonction doit afficher quelque chose comme «g-e-----e» si bien que l'on a besoin de savoir quelles lettres le joueur a devinées. Pour ce faire, on utilise une nouvelle variable globale (une chaîne cette fois-ci) qui contient toutes les lettres devinées. Chaque fois qu'une lettre est devinée, elle est ajoutée à cette chaîne :

```
lettres_devinees = ''
```

Voici la fonction :

```
def afficher_mot_avec_tirets(mot):
    afficher_mot = ''
    for lettre in mot:
        if lettres_devinees.find(lettre) > -1:
            # lettre trouvée
            afficher_mot = afficher_mot + lettre
        else:
            # lettre non trouvée
            afficher_mot = afficher_mot + '-'
    print(afficher_mot)
```

Cette fonction commence par une chaîne vide puis parcourt chaque lettre du mot. Si la lettre est une des lettres que le joueur a déjà devinées, elle est ajoutée à `afficher_mot` ; dans le cas contraire, un tiret (-) est ajouté. La fonction intégrée `find` est utilisée pour vérifier si la lettre figure dans `lettres_devinees`. La fonction `find` renvoie -1 si la lettre est absente ; sinon, elle retourne la position de la lettre. La seule chose qui nous importe

Copyright © 2014 Dunod.

vraiment est de savoir si la lettre est présente et on se contente de vérifier que le résultat est supérieur à -1. Pour finir, le mot est affiché.

Actuellement, chaque fois que la fonction `traiter_supposition` est appelée, elle ne fait rien de la supposition car c'est toujours un stub pour le moment. Nous pouvons la rendre moins squelettique en lui faisant ajouter la lettre devinée à `lettres_devinees`, de la manière suivante :

```
def traiter_supposition(supposition, mot):
    global vies_restantes
    global lettres_devinees
    vies_restantes = vies_restantes -1
    lettres_devinees = lettres_devinees + supposition
    return False
```

Ouvrez le fichier 4_7_pendu_afficher_mot.py et exécutez-le. Vous obtiendrez un résultat similaire à ceci :

```
-------
Vies restantes : 14
Devinez une lettre ou le mot entier ? e
--e------e
Vies restantes : 13
Devinez une lettre ou le mot entier ? g
g-e------e
Vies restantes : 12
Devinez une lettre ou le mot entier ?
```

Ça commence à ressembler au véritable jeu maintenant. Toutefois, il reste toujours le stub de la fonction `traiter_supposition` à compléter, ce que nous allons faire maintenant :

```
def traiter_supposition(supposition, mot):
    if len(supposition) > 1:
        return supposition_mot_entier(supposition, mot)
    else:
        return supposition_lettre_unique(supposition, mot)
```

Quand le joueur saisit une hypothèse, il a deux possibilités : soit il saisit une seule lettre, soit il essaye de deviner le mot en entier. Dans cette méthode, on se contente de décider de quel type de supposition il s'agit et d'appeler la fonction `supposition_mot_entier` ou `supposition_lettre_unique`. Comme ces fonctions sont toutes les deux assez simples, nous allons les mettre en œuvre directement, plutôt que de passer par l'intermédiaire de stubs :

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

```
def supposition_lettre_unique(supposition, mot):
    global lettres_devinees
    global vies_restantes
    if mot.find(supposition) == -1:
        # supposition incorrecte
        vies_restantes = vies_restantes - 1
    lettres_devinees = lettres_devinees + supposition
    if toutes_lettres_trouvees(mot):
        return True
    return False

def toutes_lettres_trouvees(mot):
    for letter in mot:
        if lettres_devinees.find(letter) == -1:
            return False
        return True
```

La fonction `supposition_mot_entier` est effectivement plus simple que la fonction `supposition_lettre_unique` :

```
def supposition_mot_entier(supposition, mot):
    global vies_restantes
    if supposition.lower() == mot.lower():
        return True
    else:
        vies_restantes = vies_restantes - 1
    return False
```

On se contente ici de comparer la supposition et le mot à trouver pour voir s'ils sont identiques quand ils sont tous les deux convertis en minuscules. S'ils ne sont pas identiques, une vie est perdue. La fonction retourne `True` si la supposition était exacte ; dans le cas contraire, elle retourne `False`.

Le programme est à présent terminé. Ouvrez 4_8_pendu_complet.py dans l'éditeur IDLE et exécutez-le. Voici le listing complet du programme :

```
#04_08_pendu_complet
import random

mots = ['poulet', 'chien', 'chat', 'souris', 'grenouille']
vies_restantes = 14
lettres_devinees = ''

def jouer():
    mot = choisir_un_mot()
    while True:
        supposition = obtenir_supposition(mot)
        if traiter_supposition(supposition, mot):
```

Copyright © 2014 Dunod.

```
            print('Vous avez gagné ! Bravo !')
            break
        if vies_restantes == 0:
            print('Vous êtes pendu !')
            print('Le mot était : ' + mot)
            break

def choisir_un_mot():
    mot_position = random.randint(0, len(mots) - 1)
    return mots[mot_position]

def obtenir_supposition(mot):
    afficher_mot_avec_tirets(mot)
    print('Vies restantes : ' + str(vies_restantes))
    supposition = input(' Devinez une lettre ou le mot entier ? ')
    return supposition

def afficher_mot_avec_tirets(mot):
    afficher_mot = ''
    for lettre in mot:
        if lettres_devinees.find(lettre) > -1:
            # lettre trouvée
            afficher_mot = afficher_mot + lettre
        else:
            # lettre non trouvée
            afficher_mot = afficher_mot + '-'
    print(afficher_mot)

def traiter_supposition(supposition, mot):
    if len(supposition) > 1:
        return supposition_mot_entier(supposition, mot)
    else:
        return supposition_lettre_unique(supposition, mot)

def supposition_mot_entier(supposition, mot):
    global vies_restantes
    if supposition == mot:
        return True
    else:
        vies_restantes = vies_restantes - 1
        return False

def supposition_lettre_unique(supposition, mot):
    global lettres_devinees
    global vies_restantes
    if mot.find(supposition) == -1:
        # supposition incorrecte
        vies_restantes = vies_restantes - 1
    lettres_devinees = lettres_devinees + supposition
    if toutes_lettres_trouvees(mot):
```

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

```
            return True
        return False

def toutes_lettres_trouvees(mot):
    for letter in mot:
        if lettres_devinees.find(letter) == -1:
            return False
return True

jouer()
```

Le jeu tel qu'il est conçu comporte quelques limitations. Tout d'abord, il fait la différence entre les minuscules et les majuscules, si bien que vous devez saisir vos suppositions en minuscules, de la même manière que les mots sont écrits dans la liste `mots`. Deuxièmement, si vous saisissez accidentellement **aa** au lieu de la lettre **a** quand vous faites votre supposition, cela sera traité comme une proposition de mot entier, même si c'est trop court pour représenter un mot en entier. Le jeu devrait sans doute détecter cela et ne tenir compte que des suppositions de la même longueur du mot secret pour les considérer comme des suppositions du mot entier.

À titre d'exercice, vous pouvez essayer de corriger ces problèmes. Pour le problème du respect de la casse, nous vous conseillons d'expérimenter la fonction intégrée `lower`. La correction de cet exercice se trouve dans le fichier 4_8_pendu_complet_solution.py.

4.5 DICTIONNAIRES

Les listes sont parfaites lorsque vous souhaitez accéder à vos données en partant du début et parcourir toute la liste, mais elles peuvent se révéler lentes et inefficaces lorsqu'elles grossissent et que vous avez beaucoup de données à traiter (par exemple, lors de la recherche d'une entrée particulière). C'est un peu comme si vous devez faire une recherche dans un livre qui ne possède aucun index ni table des matières. Pour trouver ce que vous recherchez, vous devez lire l'intégralité de l'ouvrage. Les dictionnaires, comme vous pouvez le deviner, fournissent un moyen plus efficace d'accéder à une structure de données lorsque vous souhaitez aller directement à un élément particulier. Lorsque l'on utilise un dictionnaire, on associe une valeur à une clé. Chaque fois que l'on veut cette valeur, on la demande à l'aide de la clé. C'est un peu comme l'association d'un nom de variable à une valeur ; cependant, la différence est qu'avec un dictionnaire, les clés et valeurs sont créées pendant l'exécution du programme. Examinons un exemple :

Copyright © 2014 Dunod.

```
>>> oeufs_par_semaine = {'Penny': 7, 'Amy': 6, 'Bernadette': 0}
>>> oeufs_par_semaine['Penny']
7
>>> oeufs_par_semaine['Penny'] = 5
>>> oeufs_par_semaine
{'Amy': 6, 'Bernadette': 0, 'Penny': 5}
>>>
```

Cet exemple enregistre le nombre d'œufs que chacune de mes poules pond. Le nombre des œufs pondus par semaine est associé au nom de chaque poule. Quand on veut récupérer la valeur d'une des poules (par exemple, Penny), on utilise ce nom entre crochets au lieu du numéro d'indice que l'on utiliserait pour une liste. Nous pouvons utiliser la même syntaxe dans les affectations pour modifier l'une des valeurs.

Par exemple, si Bernadette devait pondre un œuf, nous pourrions mettre à jour les enregistrements de la manière suivante :

```
oeufs_par_semaine['Bernadette'] = 1
```

Vous avez peut-être remarqué que lorsque le dictionnaire est affiché, les éléments ne sont pas dans le même ordre que celui dans lequel nous les avons définis. Le dictionnaire ne garde pas la trace de l'ordre dans lequel les éléments ont été définis. Notez également que même si nous avons utilisé une chaîne comme clé et un nombre comme valeur, la clé aurait pu être une chaîne, un nombre ou un tuple (voir la section suivante), alors que la valeur pourrait être n'importe quoi, y compris une liste ou un autre dictionnaire.

4.6 TUPLES

À première vue, les tuples ressemblent aux listes, mais sans les crochets. Par conséquent, nous pouvons définir et accéder à un tuple de la manière suivante :

```
>>> tuple = 1, 2, 3
>>> tuple
(1, 2, 3)
>>> tuple[0]
1
```

Cependant, si nous essayons de modifier un élément d'un tuple, on obtient un message d'erreur :

```
>>> tuple[0] = 6
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Ce message d'erreur s'explique par le fait que les tuples sont *immuables*, ce qui signifie que vous ne pouvez pas les modifier. Les chaînes et les nombres sont identiques. Bien que l'on puisse modifier une variable pour faire référence à une autre chaîne, à un autre nombre ou à un autre tuple, vous ne pouvez pas modifier le nombre lui-même. En revanche, si la variable fait référence à une liste, vous pouvez modifier cette liste en y ajoutant des éléments, en les supprimant ou en les modifiant.

Si un tuple n'est rien qu'une liste avec laquelle on ne peut pas faire grand-chose, vous vous demandez sans doute pourquoi on souhaiterait l'utiliser. La réponse est que les tuples offrent un moyen utile de créer une collection temporaire d'éléments. Python vous permet de programmer élégamment quelques astuces à l'aide des tuples, comme nous allons le voir dans les deux prochains paragraphes.

Affectation multiple

Pour assigner une valeur à une variable, on utilise simplement l'opérateur =, comme ceci :

```
a = 1
```

Python vous permet également de faire plusieurs affectations en une seule ligne :

```
>>> a, b, c = 1, 2, 3
>>> a
1
>>> b
2
>>> c
3
```

Plusieurs valeurs de retour

On souhaite parfois dans une fonction retourner plusieurs valeurs à la fois. Par exemple, imaginez une fonction qui prenne en paramètre une liste de nombres et retourne le minimum et le maximum. Voici un exemple :

```
#04_09_stats
def stats(nombres):
    nombres.sort()
    return (nombres[0], nombres[-1])
```

Copyright © 2014 Dunod.

```
list = [5, 45, 12, 1, 78]
min, max = stats(list)
print(min)
print(max)
```

Cette méthode pour trouver le minimum et le maximum n'est pas très efficace, mais il s'agit d'un simple exemple. La liste est triée puis on prend le premier et le dernier nombres. Notez que `nombres[-1]` retourne le dernier nombre parce que lorsque vous fournissez un indice négatif à une liste ou à une chaîne, Python compte en arrière à partir de la fin de la liste ou de la chaîne. Par conséquent, la position –1 indique le dernier élément, –2 l'avant dernier et ainsi de suite.

4.7 EXCEPTIONS

Python utilise des exceptions pour signaler que quelque chose a mal tourné dans votre programme. Les erreurs peuvent se manifester sous de très nombreuses formes pendant l'exécution de votre programme. Une erreur fréquente que nous avons déjà rencontrée se produit lorsque vous essayez d'accéder à un élément d'une liste ou d'une chaîne qui se trouve en dehors de la plage autorisée. Voici un exemple :

```
>>> list = [1, 2, 3, 4]
>>> list[4]
Traceback (most recent call last):
File «<stdin>», line 1, in <module>
IndexError: list index out of range
```

Si quelqu'un reçoit un tel message d'erreur alors qu'il exécute votre programme, il trouvera cela pour le moins déroutant. Par conséquent, Python fournit un mécanisme pour intercepter ces erreurs, ce qui vous permet de les gérer à votre convenance :

```
try:
      list = [1, 2, 3, 4]
      list[4]
except IndexError:
      print('Désolé')
```

Nous reparlerons des exceptions dans le prochain chapitre, où vous apprendrez la hiérarchie des différents types d'erreur qui peuvent être interceptés.

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

4.8 RÉSUMÉ DE FONCTIONS

Ce chapitre a été écrit pour vous aider à assimiler rapidement les caractéristiques les plus importantes de Python. Par nécessité, nous avons passé sous silence certaines choses et laissé de côté quelques petits détails. Cette section fournit par conséquent une référence sur les principales caractéristiques et fonctions disponibles pour les principaux types que nous avons étudiés. Considérez-la comme un aide-mémoire auquel vous pourrez vous reporter au fur et à mesure que vous progressez dans la lecture de ce livre ; n'oubliez pas de tester certaines fonctions pour voir comment elles s'utilisent. Il n'est pas nécessaire de lire en entier cette section, et il suffit de savoir où elle se trouve quand vous en aurez besoin. N'oubliez pas que le shell Python est votre ami.

Pour des informations détaillées sur Python, reportez-vous au site *http://docs.python.org/py3k*.

Nombres

Le tableau 4.1 liste certaines fonctions que vous pouvez utiliser avec des nombres.

Tableau 4.1
Fonctions permettant de gérer des nombres.

Fonction/Description/Exemple

abs(x)
Retourne la valeur absolue (supprime le signe –).
`>>>abs(-12.3)`
`12.3`

bin(x)
Utilisée pour convertir en une chaîne binaire.
`>>> bin(23)`
`'0b10111'`

complex(r,i)
Crée un nombre complexe avec des composants réels et imaginaires. Utilisée en science et en ingénierie.
`>>> complex(2,3)`
`(2+3j)`

hex(x)
Utilisée pour convertir en une chaîne hexadécimale.
`>>> hex(255)`
`'0xff'`

oct(x)
Utilisée pour convertir en une chaîne octale.
`>>> oct(9)`
`'0o11'`

Copyright © 2014 Dunod.

round(x, n)
Arrondit x avec n décimales.
`>>> round(1.111111, 2)`
`1.11`

math.factorial(n)
Fonction factorielle (comme dans 4 x 3 x 2 x 1).
`>>> math.factorial`
`(4)24`

math.log(x)
Logarithme naturel.
`>>> math.log(10)`
`2.302585092994046`

math.pow(x, y)
Élève x à la puissance de y (on peut aussi utiliser la notation x ** y).
`>>> math.pow(2, 8)`
`256.0`

math.sqrt(x)
Racine carrée.
`>>> math.sqrt(16)`
`4.0`

math.sin, cos, tan, asin, acos, atan
Fonctions trigonométriques (radians).
`>>> math.sin(math.pi / 2)`
`1.0`

Chaînes

Les constantes de chaîne peuvent être encadrées par des guillemets simples (cas le plus fréquent) ou par guillemets doubles. Les guillemets doubles sont utiles si vous souhaitez inclure des guillemets simples dans la chaîne, comme dans l'exemple suivant :

```
s = "Aujourd'hui, nous sommes mardi"
```

Dans certaines occasions, vous aurez besoin d'inclure des caractères spéciaux tels que caractères de fin de ligne ou tabulations dans une chaîne. Pour ce faire, on utilise ce qu'on appelle des *caractères d'échappement*, qui commencent par slash (\). Voici les caractères dont vous êtes susceptible d'avoir besoin :

- `\t` Caractère de tabulation
- `\n` Caractère de nouvelle ligne

Le tableau 4.2 liste certaines fonctions que vous pouvez utiliser avec des chaînes.

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Tableau 4.2
Fonctions de chaînes.

Fonction/Description/Exemple

s.capitalize()
Met l'initiale en majuscules et le reste en minuscules.
`>>> 'aBc'.capitalize()`
`'Abc'`

s.center(width)
Comble la chaîne avec des espaces pour la centrer. Un paramètre supplémentaire optionnel permet de définir le caractère de remplissage.
`>>> 'abc'.center(10, '-')`
`'---abc----'`

s.endswith(str)
Retourne `True` si la fin de la chaîne correspond.
`>>> 'abcdef'`
`.endswith('def')`
`True`

s.find(str)
Retourne la position d'une sous-chaîne. Deux paramètres supplémentaires optionnels indiquant les positions de départ et de fin peuvent être utilisés pour limiter la recherche.
`>>> 'abcdef'.find('de')`
`3`

s.format(args)
Met en forme une chaîne à l'aide de marqueurs modèles avec {}.
`>>> «Its {0} pm».format('12')`
`«Its 12 pm»`

s.isalnum()
Retourne `True` si tous les caractères de la chaîne sont des lettres ou des chiffres.
`>>> '123abc'.isalnum()`
`True`

s.isalpha()
Retourne `True` si tous les caractères sont alphabétiques.
`>>> '123abc'.isalpha()`
`False`

s.isspace()
Retourne `True` si le caractère est un espace, une tabulation ou un autre caractère espace.
`>>> ' \t'.isspace()`
`True`

s.ljust(width)
Comme center(), mais justifié à gauche.
`>>> 'abc'.ljust(10, '-')`
`'abc-------'`

s.lower()
Convertit une chaîne en minuscules.
`>>> 'AbCdE'.lower()`
`'abcde'`

Copyright © 2014 Dunod.

s.replace(ancien, nouveau)
Remplace toutes les occurrences du paramètre ancien par le paramètre nouveau.
`>>> 'hello world'`
`.replace('world', 'there') 'hello there'`

s.split()
Retourne une liste de tous les mots de la chaîne séparés par des espaces. Un paramètre optionnel permet d'indiquer un autre caractère de séparation. Le caractère de fin de ligne (\n) est souvent utilisé.
`>>> 'abc def'.split()`
`['abc', 'def']`

s.splitlines() Découpe la chaîne en se basant sur le caractère de nouvelle ligne.

s.strip()
Supprime les espaces aux deux extrémités de la chaine.
`>>> ' a b '.strip()`
`'a b'`

s.upper() Convertit la chaîne en majuscules.

Listes

Nous avons déjà examiné la plupart des fonctionnalités des listes qui sont résumées dans le tableau 4.3.

Tableau 4.3 **Fonctions de gestion de listes.**

Fonction/Description/Exemple

del(a[i:j])
Supprime les éléments de la liste, de l'élément i à l'élément j – 1.
`>>> a = ['a', 'b', 'c']`
`>>> del(a[1:2])`
`>>> a`
`['a', 'c']`

a.append(x)
Ajoute un élément à la fin de la liste.
`>>> a = ['a', 'b', 'c']`
`>>> a.append('d')`
`>>> a`
`['a', 'b', 'c', 'd']`

a.count(x)
Compte les occurrences d'un élément particulier.
`>>> a = ['a', 'b', 'a']`
`>>> a.count('a')`
`2`

a.index(x)
Retourne la position d'indice de la première occurrence de x dans a. Des paramètres optionnels peuvent être utilisés pour les indices de début et de fin.
`>>> a = ['a', 'b', 'c']`
`>>> a.index('b')`
`1`

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

a.insert(i, x)
Insère x à la position i de la liste.

```
>>> a = ['a', 'c']
>>> a.insert(1, 'b')
>>> a
['a', 'b', 'c']
```

a.pop()
Retourne le dernier élément de la liste et le supprime. Un paramètre optionnel permet de spécifier une autre position d'indice pour la suppression.

```
>>> ['a', 'b', 'c']
>>> a.pop(1)
'b'
>>> a
['a', 'c']
```

a.remove(x)
Supprime l'élément spécifié.

```
>>> a = ['a', 'b', 'c']
>>> a.remove('c')
>>> a
['a', 'b']
```

a.reverse()
Inverse la liste.

```
>>> a = ['a', 'b', 'c']
>>> a.reverse()
>>> a
['c', 'b', 'a']
```

a.sort()
Trie la liste. Des options avancées sont disponibles quand on trie des listes d'objets. Reportez-vous au chapitre suivant pour de plus amples informations.

Dictionnaires

Le tableau 4.4 détaille les informations que vous devez connaître sur les dictionnaires.

Tableau 4.4
Fonctions de gestion des dictionnaires.

Fonction/Description/Exemple

len(d)
Retourne le nombre d'éléments d'un dictionnaire.
`>>> d = {'a':1, 'b':2}`
`>>> len(d)`
`2`

del(d[key])
Supprime un élément du dictionnaire.
`>>> d = {'a':1, 'b':2}`
`>>> del(d['a'])`
`>>> d`
`{'b': 2}`

Copyright © 2014 Dunod.

clé in d
Retourne `True` si le dictionnaire (d) contient la clé.
`>>> d = {'a':1, 'b':2}`
`>>> 'a' in d`
`True`

d.clear()
Supprime tous les éléments du dictionnaire.
`>>> d = {'a':1, 'b':2}`
`>>> d.clear()`
`>>> d`
`{}`

get(clé,valeur par défaut)
Retourne la valeur de la clé ou la clé par défaut si la clé est absente.
`>>> d = {'a':1, 'b':2}`
`>>> d.get('c', 'c')`
`'c'`

Conversions de type

Nous avons déjà discuté de la situation où l'on veut convertir un nombre en une chaîne, afin de pouvoir l'ajouter à une autre chaîne. Python contient quelques fonctions intégrées permettant de convertir des éléments d'un type vers un autre type, comme cela est indiqué dans le tableau 4.5.

Tableau 4.5 **Fonctions de conversion de types.**

Fonction/Description/Exemple

float(x)
Convertit x en un nombre à virgule flottante.
`>>> float('12.34')`
`12.34`
`>>> float(12)`
`12.0`

int(x)
Argument optionnel utilisé pour spécifier la base du nombre.
`>>> int(12.34)`
`12`
`>>> int('FF', 16)`
`255`

list(x)
Convertit x en une liste. C'est aussi un moyen pratique d'obtenir une liste des clés d'un dictionnaire.
`>>> list('abc')`
`['a', 'b', 'c']`
`>>> d = {'a':1, 'b':2}`
`>>> list(d)`
`['a', 'b']`

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

4.9 RÉSUMÉ

Votre apprentissage de Python va se dérouler progressivement et vous n'aurez donc pas à assimiler toutes ces commandes d'un coup. Quand vous aurez besoin d'utiliser une fonction, n'hésitez pas à retourner à ces tableaux pour consulter sa syntaxe.

Dans le chapitre suivant, nous allons voir comment Python gère l'orientation objet.

Copyright © 2014 Dunod.

CHAPITRE PAGE

5 MODULES, CLASSES ET MÉTHODES

1	Introduction	3
2	Prise en main	17
3	Bases de Python	27
4	Chaînes, listes et dictionnaires	43
5	Modules, classes et méthodes	69

5.1 Modules **70**
5.2 Orientation objet **74**
5.3 Définition des classes **74**
5.4 Héritage **76**
5.5 Résumé **79**

6	Fichiers et Internet	81
7	Interfaces utilisateur graphiques	91
8	Programmation de jeux	109
9	Interfaçage du matériel	123
10	Projet de prototypage d'une horloge	137
11	Le RaspiRobot	147
12	Pour aller plus loin	161

Copyright © 2014 Dunod.

Dans ce chapitre, nous allons voir comment créer et utiliser nos propres modules, comme le module `random`, que nous avons utilisé au chapitre 3. Nous étudierons également la manière dont Python implémente l'orientation objet, qui permet aux programmes d'être structurés en classes, chaque classe étant responsable de son propre comportement. Cela permet de garder le contrôle sur la complexité de nos programmes, ce qui en général facilite leur gestion. Les classes et les méthodes sont les principaux mécanismes pour arriver à ce résultat. Vous avez déjà utilisé des méthodes et des classes intégrées dans les chapitres précédents sans forcément le savoir.

5.1 MODULES

La plupart des langages informatiques ont un concept équivalent aux modules qui permettent de créer un groupe de fonctions ; cette structure se révèle pratique, pour vous-même ou pour les autres, pour utiliser vos programmes dans différents projets.

Python réalise ce regroupement de fonctions d'une façon très simple et très élégante. Par essence, n'importe quel fichier de code Python peut être considéré comme un module portant le même nom que le fichier. Toutefois, avant d'écrire nos propres modules, nous allons étudier la manière d'utiliser les modules déjà installés avec Python.

Utilisation des modules

Lorsque nous avons utilisé le module `random` précédemment, nous avons fait quelque chose équivalent à :

```
>>> import random
>>> random.randint(1, 6)
6
```

La première chose que l'on réalise ici est de dire à Python que nous voulons utiliser le module `random` en employant la commande `import`. Quelque part dans l'installation de Python, il y a un fichier nommé random.py qui contient une fonction `randint` ainsi que quelques autres fonctions.

Avec tous les modules dont nous disposons, il y a un réel risque pour que différents modules puissent contenir des fonctions du même nom. Dans ce cas, comment Python peut-il savoir celle qu'il faut utiliser ? Heureusement, nous n'avons pas à nous soucier de cela parce que nous avons importé le module et qu'aucune des fonctions du module n'est visible, à moins que l'on ne préfixe son

Copyright © 2014 Dunod.

nom par le nom du module suivi d'un point. Essayez d'omettre le nom du module :

```
>>> import random
>>> randint(1, 6)
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
NameError: name 'randint' is not defined
```

Le fait d'avoir à indiquer le nom du module devant chaque appel de fonction, si elle est utilisée un grand nombre de fois, peut rapidement devenir fastidieux. Heureusement, nous pouvons simplifier les choses en invoquant la commande `import` de la manière suivante :

```
>>> import random as r
>>> r.randint(1,6)
2
```

Cela attribue au module un nom local au sein de notre programme et permet d'utiliser simplement `r` à la place de `random`, ce qui économise la frappe de quelques touches.

Si vous êtes certain qu'une fonction que vous souhaitez utiliser à partir d'une bibliothèque ne va pas rentrer en conflit avec un autre élément de votre programme, vous pouvez encore améliorer les choses de la manière suivante :

```
>>> from random import randint
>>> randint(1, 6)
5
```

Pour aller encore plus loin, vous pouvez importer la totalité d'un module d'un seul coup. À moins que vous ne sachiez exactement ce qui se trouve dans le module, ce n'est en général pas une très bonne idée, mais vous pouvez réaliser cela de la manière suivante :

```
>>> from random import *
>>> randint(1, 6)
2
```

Dans ce cas, l'astérisque (*) signifie que l'on importe « tout ».

Bibliothèques Python utiles

Jusqu'à présent, nous avons utilisé le module `random`, mais il existe d'autres modules inclus dans Python. Ces modules sont

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

souvent appelés la *bibliothèque standard* de Python. Ces modules sont trop nombreux pour qu'on les répertorie intégralement. Toutefois, vous trouverez une liste complète des modules Python à *http://docs.python.org/release/3.1.5/library/index.html.* Voici quelques-uns des modules les plus utiles que vous devriez étudier :

- **string** fonctions de gestion des chaînes
- **datetime** fonctions pour manipuler des dates et des heures
- **math** fonctions mathématiques (sin, cos et ainsi de suite)
- **pickle** fonctions pour sauvegarder ou restaurer des structures de données dans des fichiers (voir le chapitre 6)
- **urllib.request** fonctions pour la lecture des pages web (voir le chapitre 6)
- **tkinter** fonctions pour créer des interfaces utilisateur graphiques (voir le chapitre 7)

Installation de nouveaux modules

Outre les modules de la bibliothèque standard, des milliers de modules ont été écrits par la communauté Python. Il existe notamment un module très populaire qui se nomme `pygame`, et que nous utiliserons au chapitre 8. Il est souvent disponible sous la forme d'un paquet binaire, et vous pouvez l'installer en saisissant simplement la commande suivante :

```
sudo apt-get install python-pygame
```

Ce n'est cependant pas aussi facile pour tous les autres modules, et vous devrez parfois consacrer plus d'efforts à l'installation de certains modules.

Tous les modules suffisamment bons pour pouvoir être utilisés sont empaquetés selon la méthode standard de Python. Cela signifie que pour les installer, vous devez télécharger un fichier compressé contenant le répertoire du module. Prenons l'exemple du module `RPi.GPIO` que nous utiliserons au chapitre 11. Pour installer ce module, vous irez d'abord sur le site web du module, trouverez la section de téléchargements et récupérerez l'archive du module (voir la figure 5.1). L'étape suivante consiste à enregistrer le fichier dans un répertoire approprié (pour cet exemple, utilisez le répertoire Python que nous avons créé au chapitre 3).

Copyright © 2014 Dunod.

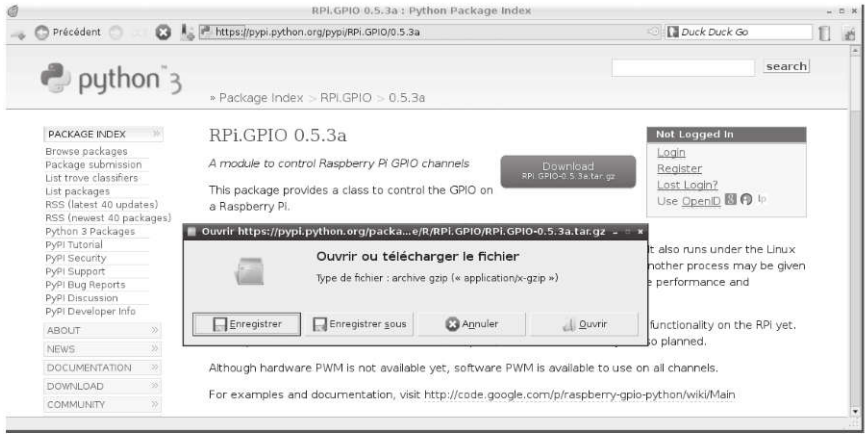


Figure 5.1
Téléchargement du module RPi.GPIO.

Une fois que le fichier a été sauvegardé, ouvrez LXTerminal et utilisez la commande suivante pour vous rendre dans le répertoire Python :

```
pi@raspberrypi:~ cd Python
pi@raspberrypi:~/Python$ ls
RPi.GPIO-0.3.1a.tar.gz
```

Ensuite, vous devez décompresser le répertoire de l'archive en saisissant la commande suivante :

```
pi@raspberrypi:~/Python$ tar -xzf RPi.GPIO-0.3.1a.tar.gz
pi@raspberrypi:~/Python$ ls
RPi.GPIO-0.3.1a RPi.GPIO-0.3.1a.tar.gz
```

Maintenant que vous avez un nouveau dossier pour le module, vous devez vous déplacer dedans à l'aide de la commande `cd` puis exécuter la commande `install`. Il est toutefois toujours utile de vérifier les instructions au préalable pour voir s'il y a autre chose que vous devez faire. Pour obtenir les instructions, saisissez `more INSTALL.txt`.

C'est une bonne idée d'avoir pensé à vérifier car les instructions disent que vous devez exécuter la commande suivante :

```
sudo apt-get install python3-dev
```

Vous êtes enfin prêt à exécuter le programme d'installation du module :

```
pi@raspberrypi:~/Python$ cd RPi.GPIO-0.3.1a
pi@raspberrypi:~/Python/RPi.GPIO-0.3.1a$ sudo python3 setup.py
install
```

Une fois que le module est installé, vous serez en mesure de l'importer à partir du shell Python.

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

5.2 ORIENTATION OBJET

L'orientation objet a beaucoup de traits communs avec les modules. Elle partage les mêmes objectifs d'essayer de regrouper les éléments associés, afin qu'ils soient plus faciles à maintenir et à localiser. Comme son nom l'indique, l'orientation objet a un rapport avec les objets. Nous avons déjà utilisé des objets, même si nous n'en avions pas conscience. Par exemple, une chaîne est un objet. Ainsi, quand nous saisissons :

```
>>> 'abc'.upper()
```

nous disons à la chaîne `'abc'` que nous en voulons une copie, mais en majuscules. En termes d'orientation objet, `abc` est une *instance* de la classe intégrée `str` et `upper` est une *méthode* de la classe `str`.

En fait, nous pouvons déterminer la classe d'un objet de la manière suivante (notez bien le double trait de soulignement avant et après le mot `class`) :

```
>>> 'abc'.__class__
<class 'str'>
>>> [1].__class__
<class 'list'>
>>> 12.34.__class__
<class 'float'>
```

5.3 DÉFINITION DES CLASSES

Nous en avons assez d'utiliser les classes des autres et nous allons à présent créer les nôtres. Nous allons commencer par créer une classe qui réalise la conversion de mesures d'une unité dans une autre en multipliant une valeur par un facteur d'échelle.

Nous donnerons à la classe le nom ConvertisseurEchelle. Voici le listing de l'ensemble de la classe, ainsi que quelques lignes de code pour la tester :

```
#05_01_convertisseur
class ConvertisseurEchelle:

    def __init__(self, unites_depart, unites_arrivee, facteur):
        self.unites_depart = unites_depart
        self.unites_arrivee = unites_arrivee
        self.facteur = facteur
```

Copyright © 2014 Dunod.

```
    def description(self):

return 'Convertir ' + self.unites_depart + ' en ' + self.unites_
arrivee

    def convertir(self, value):
        return value * self.facteur

c1 = ConvertisseurEchelle('pouces', 'mm', 25)
print(c1.description())
print('conversion de 2 pouces')
print(str(c1.convertir(2)) + c1.unites_arrivee)
```

Cela nécessite quelques explications. La première ligne, qui est assez évidente, stipule que nous commençons la définition d'une classe appelée ConvertisseurEchelle. Le signe deux-points (:) à la fin indique que tout ce qui suit fait partie de la définition de la classe, jusqu'à ce que l'on revienne au niveau d'indentation de la marge gauche.

À l'intérieur de `ConvertisseurEchelle`, nous pouvons voir ce qui ressemble à trois définitions de fonctions. Ces fonctions, qui appartiennent à la classe, ne peuvent pas être utilisées autrement que par une instance de la classe. On appelle *méthodes* ces types de fonctions qui appartiennent à une classe.

La première méthode, `__init__`, semble un peu étrange car son nom a deux caractères de soulignement de chaque côté. Lorsque Python crée une nouvelle instance d'une classe, il appelle automatiquement la méthode `__init__`. Le nombre de paramètres que doit accepter `__init__` dépend du nombre de paramètres fournis lorsque l'on crée une instance de la classe. Pour comprendre cela, il faut se pencher sur cette ligne à la fin du fichier :

```
c1 = ConvertisseurEchelle('pouces', 'mm', 25)
```

Cette ligne crée une nouvelle instance de `ConvertisseurEchelle`, en précisant les unités à convertir, ainsi que le facteur d'échelle. La méthode `__init__` doit comporter tous ces paramètres, mais elle doit aussi avoir un paramètre appelé `self` qui est le premier paramètre :

```
def __init__(self, unites_depart, unites_arrivee, facteur):
```

Le paramètre `self` fait référence à l'objet lui-même. Maintenant, si l'on examine le corps de la méthode `__init__`, on peut voir certaines affectations :

```
self.unites_depart = unites_depart
self.unites_arrivee = unites_arrivee
self.facteur = facteur
```

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Chacune de ces affectations crée une variable qui appartient à l'objet et reçoit sa valeur initiale des paramètres passés dans `__init__`. Pour résumer, lorsque nous créons une nouvelle instance de `ConvertisseurEchelle` en saisissant :

```
c1 = ConvertisseurEchelle('pouces', 'mm', 25)
```

Python crée une nouvelle instance de `ConvertisseurEchelle` et assigne les valeurs « pouces », « mm » et 25 à ses trois variables `self.unites_depart`, `self.unites_arrivee` et `self.facteur`. Le terme *encapsulation* est souvent utilisé quand on évoque les classes. C'est le travail d'une classe d'encapsuler tout ce qui a à voir avec la classe. Cela comprend le stockage des données (comme les trois variables) et toutes les choses que vous pouvez réaliser avec les données grâce aux méthodes `description` et `convertir`.

La première méthode (`description`) prend les informations du convertisseur à propos de ses unités et crée une chaîne qui les décrit. Comme avec `__init__`, toutes les méthodes doivent avoir un premier paramètre nommé `self`. La méthode aura probablement besoin d'accéder aux données de la classe à laquelle elle appartient.

Essayez vous-même en exécutant le programme 05_01_convertisseur.py et en saisissant les commandes suivantes dans le shell Python :

```
>>> convertisseur_idiot = ConvertisseurEchelle('pommes', 'raisins',
74)
>>> convertisseur_idiot.description()
'Convertir pommes en raisons'
```

La méthode `convertir` a deux paramètres : le paramètre obligatoire `self` et un paramètre appelé `valeur`. La méthode retourne simplement le résultat de la multiplication de la valeur transmise par `self.factor` :

```
>>> convertisseur_idiot.convertir(3)
222
```

5.4 HÉRITAGE

La classe `ConvertisseurEchelle` convient pour les unités de longueur et les choses du même genre, mais elle ne fonctionnerait pas, par exemple, pour une conversion de température de degrés Celsius (C) en degrés Fahrenheit (F). La formule de conversion est F = C * 1,8 + 32. Il y a à la fois un facteur d'échelle (1,8) et un décalage (32).

Copyright © 2014 Dunod.

Nous allons créer une classe appelée `ConvertisseurEchelleEtDecalage` qui est comparable à `ConvertisseurEchelle`, mais avec un facteur et aussi un décalage. On pourrait réaliser cela en copiant tout simplement l'ensemble du code de `ConvertisseurEchelle` et en le modifiant légèrement en ajoutant une variable supplémentaire. Cela pourrait ressembler à ceci :

```
#05_02_convertisseur_decalage_mauvais
class ConvertisseurEchelleEtDecalage:

   def __init__(self, unites_depart, unites_arrivee, facteur,
decalage):
      self.unites_depart = unites_depart
      self.unites_arrivee = unites_arrivee
      self.facteur = facteur
      self.decalage = decalage

   def description(self):

return 'Convertir ' + self.unites_depart + ' en ' + self.unites_
arrivee

   def convertir(self, value):
      return value * self.facteur + self.decalage

c2 = ConvertisseurEchelleEtDecalage('C', 'F', 1.8, 32)
print(c2.description())
print('conversion de 20C')
print(str(c2.convertir(20)) + c2.unites_arrivee)
```

Si l'on souhaite que les deux types de convertisseurs soient disponibles dans le programme que nous sommes en train d'écrire, il ne faut pas procéder de la sorte. Cette façon de faire est mauvaise parce que nous répétons le code. La méthode `description` est réellement identique et `__init__` est très similaire. Une bien meilleure façon de faire consiste à utiliser ce qu'on appelle *l'héritage*.

Derrière le concept d'héritage des classes, il y a l'idée que quand vous voulez une version spécialisée d'une classe qui existe déjà, vous héritez de toutes les variables et méthodes de la classe parente et que vous vous contentez d'en ajouter de nouvelles ou bien vous remplacez celles qui sont différentes. La figure 5.2 illustre un diagramme de classes des deux classes, qui indique la manière dont `ConvertisseurEchelleEtDecalage` hérite de `ConvertisseurEchelle`, ajoute une nouvelle variable (`decalage`) et substitue la méthode `convertir` (parce qu'elle fonctionne un peu différemment).

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

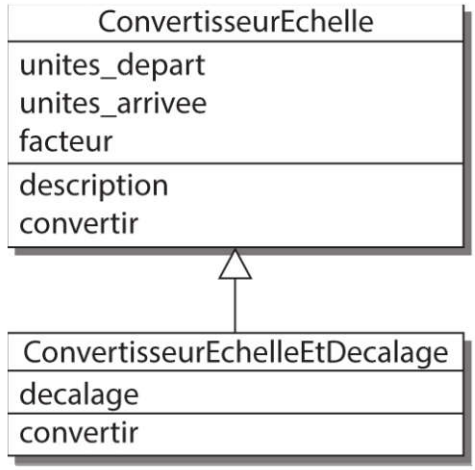


Figure 5.2
Exemple d'utilisation de l'héritage.

Voici la définition de classe de `ConvertisseurEchelleEtDecalage` qui utilise l'héritage :

```
class ConvertisseurEchelleEtDecalage(ConvertisseurEchelle):

   def __init__(self, unites_depart, unites_arrivee, facteur,
decalage):
      ConvertisseurEchelle.__init__(self, unites_depart, unites_
arrivee, facteur)
      self.decalage = decalage

   def convertir(self, valeur):
      return valeur * self.facteur + self.decalage
```

La première chose à noter est que la définition de classe de `ConvertisseurEchelleEtDecalage` est suivie immédiatement de `ConvertisseurEchelle` entre parenthèses. C'est la manière dont on spécifie la classe parente d'une classe.

La méthode `__init__` de la nouvelle « sous-classe » de `ConvertisseurEchelle` appelle tout d'abord la méthode `__init__` de `ConvertisseurEchelle` avant de définir la nouvelle variable `decalage`. La méthode `convertir` va remplacer la méthode `convertir` de la classe parente parce qu'il faut ajouter le décalage pour ce type de convertisseur. Vous pouvez tester les deux classes ensemble en exécutant le fichier 05_03_convertisseurs_final.py :

```
>>> c1 = ConvertisseurEchelle('pouces', 'mm', 25)
>>> print(c1.description())
Convertir pouces en mm
>>> print('conversion de 2 pouces')
```

Copyright © 2014 Dunod.

```
conversion de 2 pouces
>>> print(str(c1.convertir(2)) + c1.unites_arrivee)
50mm
>>> c2 = ConvertisseurEchelleEtDecalage('C', 'F', 1.8, 32)
>>> print(c2.description())
Convertir C en F
>>> print('conversion de 20C')
conversion de 20C
>>> print(str(c2.convertir(20)) + c2.unites_arrivee)
68.0F
```

Il est facile de convertir ces deux classes en un module que l'on pourra utiliser dans d'autres programmes. En fait, nous allons utiliser ce module au chapitre 7, où nous lui grefferons une interface utilisateur graphique.

Pour transformer ce fichier en un module, nous devons d'abord supprimer le code de test puis donner au fichier un nom plus explicite. Appelons-le convertisseurs.py. Vous trouverez ce fichier dans les compléments électroniques de ce livre. Le module doit se situer dans le même répertoire que le programme qui souhaite l'utiliser. Pour utiliser le module à présent, il suffit d'agir de la manière suivante :

```
>>> import convertisseurs
>>> c1 = ConvertisseurEchelle('pouces', 'mm', 25)
>>> print(c1.description())
Convertir pouces en mm
>>> print('conversion de 2 pouces')
conversion de 2 pouces
>>> print(str(c1.convertir(2)) + c1.unites_arrivee)
50mm
```

5.5 RÉSUMÉ

Il y a de nombreux modules disponibles pour Python, et certains ont été créés spécifiquement pour le Raspberry Pi, comme la bibliothèque RPi.GPIO pour contrôler les broches GPIO. Au fur et à mesure que vous avancerez dans la lecture de cet ouvrage, vous rencontrerez différents modules. Vous trouverez également qu'à mesure que vos programmes deviennent plus complexes, l'approche orientée objet dans la conception et le codage de vos projets de codage en facilite la gestion.

Dans le prochain chapitre, nous allons étudier l'utilisation des fichiers et d'Internet.

Copyright © 2014 Dunod.

Copyright © 2014 Dunod.

CHAPITRE PAGE

6 FICHIERS ET INTERNET

1	Introduction	3
2	Prise en main	17
3	Bases de Python	27
4	Chaînes, listes et dictionnaires	43
5	Modules, classes et méthodes	69
6	Fichiers et Internet	81

6.1 Fichiers **82**
6.2 Pickling **87**
6.3 Internet **88**
6.4 Résumé **90**

7	Interfaces utilisateur graphiques	91
8	Programmation de jeux	109
9	Interfaçage du matériel	123
10	Projet de prototypage d'une horloge	137
11	Le RaspiRobot	147
12	Pour aller plus loin	161

Copyright © 2014 Dunod.

Avec Python, il est facile d'utiliser des fichiers et de se connecter à Internet. Vous pouvez lire des données stockées dans des fichiers, écrire des données dans des fichiers et extraire du contenu à partir d'Internet. Vous pouvez même par programmation vérifier l'arrivée des nouveaux courriels et des tweets.

6.1 FICHIERS

Lorsque vous exécutez un programme Python, toutes les valeurs stockées dans des variables seront perdues quand le programme se terminera. Les fichiers offrent la possibilité de conserver les données.

Lecture des fichiers

Avec Python, il est extrêmement facile de lire le contenu d'un fichier. À titre d'exemple, nous pouvons convertir le programme de pendu du chapitre 4 pour lire la liste des mots à partir d'un fichier au lieu qu'ils soient figés dans le programme.

Tout d'abord, commencez un nouveau fichier dans IDLE et placez-y quelques mots en écrivant un mot par ligne. Enregistrez ensuite le fichier sous le nom mots_pendu.txt dans le même répertoire que le programme de pendu du chapitre 4 (04_08_pendu_complet.py). Notez que dans la boîte de dialogue vous devrez changer le type de fichier en .txt (voir la figure 6.1).

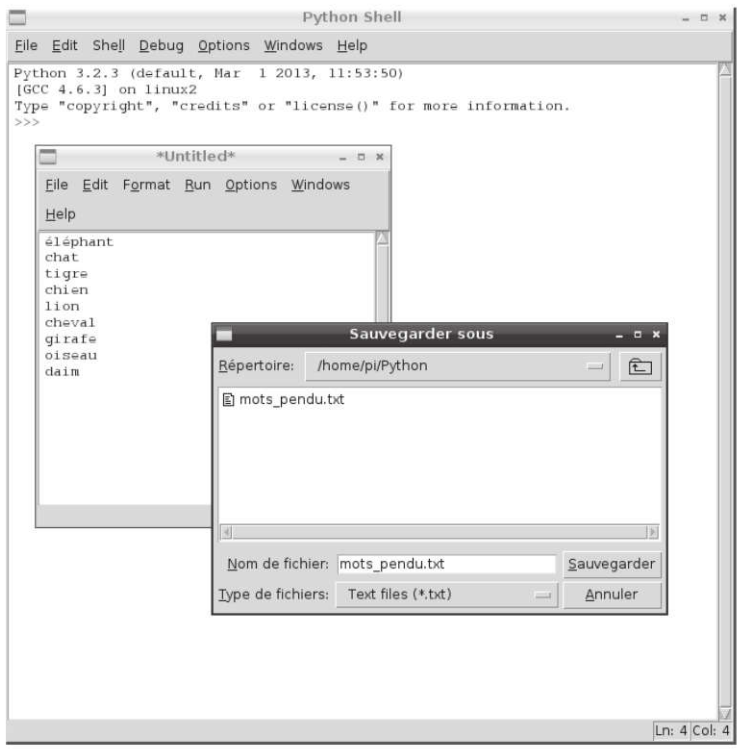


Figure 6.1
Création d'un fichier texte dans IDLE.

Copyright © 2014 Dunod.

Avant de modifier le programme de pendu lui-même, nous pouvons tester la lecture du fichier dans la console Python. Entrez l'instruction suivante dans la console :

```
>>> f = open('Python/mots_pendu.txt')
```

Notez que la console Python a comme répertoire en cours /home/pi, si bien que le répertoire Python (ou quel que soit l'emplacement où vous avez enregistré le fichier) doit être valide. Saisissez ensuite ce qui suit dans la console Python :

```
>>> mots = f.read()
>>> mots
'éléphant\nchat\ntigre\nchien\nlion\ncheval\ngirafe\noiseau\
ndaim\n'
>>> mots.splitlines()
['éléphant', 'chat', 'tigre', 'chien', 'lion', 'cheval', 'girafe',
'oiseau', 'daim']
>>>
```

Je vous avais bien dit que c'était facile ! La seule chose à faire pour ajouter ce fichier au programme de pendu consiste à remplacer la ligne

```
mots = ['poulet', 'chien', 'chat', 'souris', 'grenouille']
```

par les lignes suivantes :

```
f = open('mots_pendu.txt')
mots = f.read().splitlines()
f.close()
```

La ligne `f.close()` a été ajoutée car vous devez toujours appeler la commande `close` lorsque vous en avez terminé avec un fichier afin de libérer les ressources du système d'exploitation. Le fait de laisser un fichier ouvert peut poser des problèmes.

Le programme complet figure dans le fichier 06_01_fichier_pendu.py, et il y a une liste de noms d'animaux dans le fichier mots_pendu.txt. Ce programme ne fait rien pour vérifier que le fichier existe avant de tenter de le lire ce qui signifie que si le fichier est absent, on obtient l'erreur suivante :

```
Traceback (most recent call last):
File «06_01_fichier_pendu.py», line 4, in <module>
f = open(mots_pendu.txt')
IOError: [Errno 2] No such file or directory: 'mots_pendu.txt'
```

Pour que cela soit un peu plus convivial, le code de lecture du fichier doit se trouver à l'intérieur d'une commande `try` :

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

```
try:
    f = open('mots_pendu.txt')
    mots = f.read().splitlines()
    f.close()
except IOError:
    print(«Impossible de trouver le fichier 'mots_pendu.txt'»)
    exit()
```

Python va essayer d'ouvrir le fichier, mais comme le fichier est absent, il ne sera pas en mesure de le faire. Par conséquent, la partie `except` du programme s'appliquera et un message plus convivial sera affiché. Comme nous ne pouvons rien faire sans liste de mots à deviner, il est inutile de continuer si bien que la commande `exit` est utilisée pour quitter le programme.

Pour l'écriture du message d'erreur, nous avons répété le nom du fichier. Si l'on veut s'en tenir strictement au principe de ne pas se répéter, le nom du fichier doit être stocké dans une variable, comme cela est illustré ci-dessous. De cette façon, si nous décidons d'utiliser un autre fichier, il suffit de modifier le code en un seul endroit.

```
fichier_mots = 'mots_pendu.txt'
try:
    f = open(fichier_mots)
    mots = f.read().splitlines()
    f.close()
except IOError:
    print("Impossible de trouver : " + fichier_mots)
    exit()
```

Vous trouverez une version modifiée du jeu du pendu avec ce code dans le fichier 06_02_fichier_pendu_try.py.

Lecture de gros fichiers

La façon dont nous avons procédé dans la section précédente marche très bien pour les petits fichiers ne contenant que quelques mots. Cependant, si nous lisons un fichier vraiment énorme (par exemple, de plusieurs mégaoctets), deux choses se produiront. Tout d'abord, il faudra à Python beaucoup de temps pour lire toutes les données. Deuxièmement, comme toutes les données sont lues en même temps, à peu près autant de mémoire que la taille du fichier sera utilisée et, dans le cas de fichiers vraiment volumineux, Python pourra se retrouver à court de mémoire.

Si vous vous trouvez dans cette situation, vous devez penser à la manière dont vous allez gérer cela. Par exemple, si vous recher-

Copyright © 2014 Dunod.

chez une chaîne particulière dans un fichier, vous pouvez vous contenter de lire le fichier ligne à ligne :

```
#06_03_fichier_readline
fichier_mots = 'mots_pendu.txt'
try:
    f = open(fichier_mots)
    ligne = f.readline()
    while ligne != '':
        if ligne == 'tigre\n':
            print('Il y a un tigre dans le fichier')
            break
        ligne = f.readline()
    f.close()
except IOError:
    print("Impossible de trouver : " + fichier_mots)
```

Lorsque la fonction `readline` atteint la dernière ligne du fichier, elle retourne une chaîne vide (''). Sinon, elle retourne le contenu de la ligne, y compris le caractère de fin de ligne (`\n`). Si elle lit une ligne vide qui est en fait juste un trou entre deux lignes et non pas la fin du fichier, elle retourne simplement le caractère de fin de ligne (`\n`). Grâce au programme de lecture ligne à ligne, la mémoire utilisée ne dépasse jamais l'équivalent d'une ligne complète. Si le fichier n'est pas décomposé en lignes pratiques, vous pouvez spécifier un argument dans `read` qui limite le nombre de caractères lus. L'exemple suivant permet de ne lire que les 20 premiers caractères d'un fichier :

```
>>> f = open('mots_pendu.txt')
>>> f.read(20)
'éléphant\nchat\ntigre\n'
>>> f.close()
```

Écriture de fichiers

Il est presque aussi simple d'écrire des fichiers. Lorsqu'un fichier est ouvert (on spécifie le nom du fichier à ouvrir), vous pouvez également préciser le mode d'ouverture du fichier. Le mode est représenté par un caractère, et si aucun mode n'est spécifié, on suppose qu'il est en lecture. Voici les modes disponibles :

- `r (read)`.
- `w (write)` remplace le contenu d'un fichier existant portant le même nom.
- `a (append)` écrit quelque chose à la fin d'un fichier existant.
- `r+` ouvre le fichier pour la lecture et l'écriture (pas souvent utilisé).

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Pour écrire un fichier, il faut l'ouvrir avec un deuxième paramètre qui prendra la valeur `'w'`, `'a'` ou `'r+'`. Voici un exemple :

```
>>> f = open('test.txt', 'w')
>>> f.write('Ce fichier n'est pas vide')
>>> f.close()
```

Le système de fichiers

Occasionnellement, vous aurez besoin d'effectuer quelques opérations sur le système de fichiers (déplacement, copie, etc.). Python utilise Linux pour effectuer ces actions, mais il offre un moyen élégant de les accomplir, dans le style de Python. Plusieurs de ces fonctions figurent dans le paquet `shutil` (utilitaire shell). Il y a un certain nombre de variantes subtiles dans les fonctionnalités de copie de base et de déplacement qui ont trait aux autorisations de fichiers et aux métadonnées. Dans cette section, nous traiterons seulement des opérations de base. Vous pouvez consulter la documentation officielle de Python pour toutes les autres fonctions à l'adresse suivante :

http://docs.python.org/release/3.1.5/library

Voici comment faire pour copier un fichier :

```
>>> import shutil
>>> shutil.copy('test.txt', 'test_copy.txt')
```

Pour déplacer un fichier, vous pouvez changer son nom ou bien le déplacer dans un répertoire différent :

```
shutil.move('test_copy.txt', 'test_dup.txt')
```

Cela fonctionne sur les répertoires ainsi que sur les fichiers. Si vous souhaitez copier le contenu d'un dossier entier (y compris ses sous-dossiers), vous pouvez utiliser la fonction `copytree`. Il existe aussi la fonction `rmtree` qui est plutôt dangereuse car elle supprime de manière récursive un répertoire et tout son contenu. Soyez donc extrêmement prudent si vous l'utilisez !

La façon la plus simple de découvrir ce qui se trouve dans un répertoire consiste à utiliser *globbing*. Le paquet `glob` permet de créer une liste des fichiers d'un répertoire en spécifiant un caractère générique (*). Voici un exemple :

```
>>> import glob
glob.glob('*.txt')
['mots_pendu.txt', 'test.txt', 'test_dup.txt']
```

Copyright © 2014 Dunod.

Si vous souhaitez simplement tous les fichiers d'un dossier, il suffit de spécifier :

```
glob.glob('*')
```

6.2 PICKLING

Le *pickling* consiste à sauvegarder le contenu d'une variable dans un fichier de telle sorte que le fichier puisse être chargé plus tard pour récupérer la valeur d'origine. La raison la plus courante de cette opération est la volonté d'enregistrer les données entre les différentes exécutions d'un programme. À titre d'exemple, nous pouvons créer une liste complexe contenant une autre liste et divers autres objets de données, puis sauvegarder cette liste dans un fichier appelé maliste.pickle de la manière suivante :

```
>>> maliste = ['a', 123, [4, 5, True]]
>>> maliste
['a', 123, [4, 5, True]]
>>> import pickle
>>> f = open('maliste.pickle', 'w')
>>> pickle.dump(maliste, f)
>>> f.close()
```

Si vous recherchez le fichier et l'ouvrez dans un éditeur pour l'examiner, vous verrez quelque chose d'énigmatique qui ressemble à ceci :

```
(lp0
S'a'
p1
aI123
a(lp2
I4
aI5
aI01
a - a.
```

Cela était à prévoir ; il s'agit de texte, mais il n'est pas censé être sous une forme lisible par un être humain. Pour reconstruire un fichier pickle dans un objet, voici ce qu'il faut faire :

```
>>> f = open('maliste.pickle')
>>> autre_tableau = pickle.load(f)
>>> f.close()
>>> autre_tableau
['a', 123, [4, 5, True]]
```

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

6.3 INTERNET

La plupart des applications utilisent Internet d'une manière ou d'une autre, même si c'est juste pour vérifier si une nouvelle version de l'application est disponible. On communique avec un serveur web en lui envoyant des requêtes HTTP (*Hypertext Transfer Protocol*). Le serveur web envoie alors sa réponse sous la forme d'un flux de texte. Ce texte sera en HTML (*Hypertext Markup Language*), le langage utilisé pour créer des pages web.

Essayez de saisir le code suivant dans la console Python :

```
>>> import urllib.request
>>> u = 'http://www.amazon.com/s/ref=nb_sb_noss?field-
keywords=raspberry+pi'
>>> f = urllib.request.urlopen(u)
>>> contents = f.read()
>>> contents
… vous obtiendrez beaucoup de code HTML
>>> f.close()
```

Notez que vous devez exécuter la ligne `read` dès que possible après l'ouverture de l'URL. Vous avez envoyé une requête web au site www.amazon.com, en lui demandant d'effectuer une recherche sur « raspberry pi ». Cela a renvoyé le code HTML de la page web d'Amazon qui s'afficherait si vous utilisiez un navigateur et vous obtenez ainsi la liste des résultats de la recherche.

Si vous regardez attentivement la structure de cette page web, vous constaterez que vous pouvez l'utiliser pour fournir une liste de produits trouvés par Amazon et en lien avec le Raspberry Pi. Si vous faites défiler ce texte, vous trouverez quelques lignes comme celles-ci :

```
<div class="productTitle"><a href="http://www.amazon.com/Raspberry-
User-Guide
-Gareth-Halfacree/dp/111846446X"> Raspberry Pi User Guide</a> <span
class="ptBrand">by <a href="/Gareth-Halfacree/e/B0088CA5ZM">Gareth
Halfacree</a> and Eben Upton</span><span class="binding"> (<span
class
="format">Paperback</span> - Nov. 13, 2012)</span></div>
```

La chose importante à noter est `<div class=»productTitle»>`. Il y a une seule instance de cette balise avant chacun des résultats de la recherche (si vous voulez faire une comparaison, il est plus facile d'ouvrir la même page web dans un navigateur). Si vous voulez copier uniquement le texte du titre de l'article, il faut trouver la position du texte `productTitle`, compter deux caractères

Copyright © 2014 Dunod.

>, puis extraire le texte à partir de cette position jusqu'au prochain caractère < :

```
#06_04_recherche_amazon
import urllib.request

u = 'http://www.amazon.com/s/ref=nb_sb_noss?field-
keywords=raspberry+pi'
f = urllib.request.urlopen(u)
contenu = str(f.read())
f.close()
i = 0
while True:
    i = contenu.find('productTitle', i)
    if i == -1:
        break
    # Trouve les 2 prochains '>' après 'productTitle'
    i = contenu.find('>', i+1)
    i = contenu.find('>', i+1)
    # Trouve le premier '<' après les 2 '>'
    j = contenu.find('<', i+1)
    titre = contenu[i+2:j]
    print(titre)
```

Lorsque vous exécutez ce programme, vous obtiendrez généralement une liste de produits. Si vous avez vraiment envie de vous lancer dans ce genre de choses, recherchez sur Internet « expressions régulières en Python ». Les expressions régulières, qui sont presque un langage à part entière, sont utilisées pour faire des recherches complexes et des validations de texte. Elles ne sont pas faciles à apprendre ni à utiliser, mais elles peuvent simplifier des tâches comme celle-ci.

La technique que nous avons employée n'est cependant pas idéale. Tout d'abord, les entreprises n'aiment pas beaucoup que l'on vienne extraire des données de leurs pages web avec des programmes automatisés. Vous pouvez par conséquent obtenir un avertissement, voire être banni de certains sites. En second lieu, cette action est très dépendante de la structure de la page web si bien qu'un petit changement du site web pourrait empêcher le programme de fonctionner. Il est donc préférable de chercher à utiliser le service web officiel du site. Au lieu de retourner les données en HTML, ces services renvoient des données beaucoup plus faciles à traiter, souvent au format XML ou JSON.

Si vous voulez en savoir plus sur la façon de réaliser ce genre de chose, recherchez sur Internet « services web en Python ».

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

6.4 RÉSUMÉ

Ce chapitre vous a fourni les bases de l'utilisation des fichiers et de l'accès aux pages web en Python. Il y a beaucoup d'autres choses à réaliser en Python si l'on souhaite exploiter Internet, notamment l'accès au courrier électronique et aux autres protocoles Internet. Pour plus d'informations sur ce sujet, reportez-vous à la documentation de Python qui est accessible à l'adresse suivante :

http://docs.python.org/Release/3.1.5/Library/Internet.html

Copyright © 2014 Dunod.

CHAPITRE PAGE

7 INTERFACES UTILISATEUR GRAPHIQUES

1	Introduction	3
2	Prise en main	17
3	Bases de Python	27
4	Chaînes, listes et dictionnaires	43
5	Modules, classes et méthodes	69
6	Fichiers et Internet	81
7	Interfaces utilisateur graphiques	91

7.1 Tkinter **92**
7.2 Hello World **92**
7.3 Convertisseur de température **93**
7.4 Autres contrôles d'interface utilisateur **97**
7.5 Boîtes de dialogue **104**
7.6 Menus **106**
7.7 Le Canvas **107**
7.8 Résumé **108**

8	Programmation de jeux	109
9	Interfaçage du matériel	123
10	Projet de prototypage d'une horloge	137
11	Le RaspiRobot	147
12	Pour aller plus loin	161

Copyright © 2014 Dunod.

Pour l'instant, nous n'avons fait que manipuler du texte et notre jeu de pendu aurait pu trouver sa place sur un ordinateur domestique des années 1980. Ce chapitre va vous montrer comment créer des applications avec une interface utilisateur graphique (que l'on abrège en anglais par GUI pour *graphical user interface*).

7.1 TKINTER

Tkinter est l'interface Python du système GUI Tk. Tk n'est pas spécifique à Python et il existe des interfaces pour Tk dans de nombreux langages différents, Tk fonctionnant sur pratiquement n'importe quel système d'exploitation, y compris Linux. Tkinter est livré avec Python et il n'y a donc pas besoin d'installer quoi que ce soit. C'est aussi l'outil le plus couramment utilisé pour la création d'une interface graphique en Python.

7.2 HELLO WORLD

La tradition veut que le premier programme que l'on écrive avec un nouveau langage doive faire quelque chose de trivial, juste pour s'assurer que ça marche ! En général, on se contente d'écrire un programme qui affiche le message « Hello World ». Comme vous vous en souvenez peut-être, nous avons déjà fait cela pour Python au chapitre 3, et je vais donc sacrifier à la tradition pour notre premier programme utilisant une interface utilisateur graphique :

```
#07_01_hello.py

from tkinter import *
root = Tk()
Label(root, text='Hello World').pack()
root.mainloop()
```

La figure 7.1 montre une application qui ne paye pas de mine.



Figure 7.1
Programme Hello World en Tkinter.

Vous n'avez pas à vous soucier de la manière dont tout cela fonctionne. Vous devez cependant savoir qu'il faut attribuer une variable à l'objet `Tk`. Ici, nous appelons cette variable `root`, ce

Copyright © 2014 Dunod.

qui est une convention usuelle. Ensuite, nous créons une instance de la classe `Label`, dont le premier argument est `root`. Cela indique à Tkinter que l'étiquette (en anglais *label*) lui appartient. Le deuxième argument spécifie le texte à afficher dans l'étiquette. Enfin, la méthode `pack` est appelée sur l'étiquette. Cela indique à l'étiquette d'occuper l'espace disponible. La méthode `pack` contrôle l'affichage des éléments dans la fenêtre. Sous peu, nous allons utiliser une grille pour mettre en page des composants.

7.3 CONVERTISSEUR DE TEMPÉRATURE

Pour débuter avec Tkinter, vous allez créer pas à pas une application simple qui fournit une interface graphique à une application de conversion de température (voir la figure 7.2). Cette application utilise le module `convertisseurs`, que nous avons créé au chapitre 5 pour réaliser la conversion.



Figure 7.2
Application de conversion de température.

Notre application Hello World, bien qu'elle soit simple, n'est pas bien structurée et ne se prêterait pas bien à un exemple plus complexe. Il est normal quand on crée une interface graphique avec Tkinter d'utiliser une classe pour représenter chaque fenêtre de l'application. Par conséquent, notre première étape consiste à créer un cadre pouvant accueillir l'application ; on commence donc par une fenêtre avec le titre « Convertisseur de température » et une étiquette unique :

```
#07_02_temp_framework.py

from tkinter import *

class App:

    def __init__(self, master):
        frame = Frame(master)
        frame.pack()
        Label(frame, text='degré C').grid(row=0, column=0)
        button = Button(frame, text='Convertir', command=self.
convert)
        button.grid(row=1)
```

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

```
    def convert(self):
        print('Non implémenté')

root = Tk()
root.wm_title('Convertisseur de température')
app = App(root)
root.mainloop()
```

Nous avons ajouté une classe au programme que nous avons appelée `App`. Elle a une méthode `__init__` qui est utilisée lorsqu'une nouvelle instance de l'application est créée dans la ligne suivante :

```
app = App(root)
```

Nous passons l'objet root `Tk` à `__init__` à l'endroit où l'interface utilisateur est construite.

Comme dans l'exemple Hello World, nous utilisons un objet `Label`, mais cette fois-ci, au lieu d'ajouter l'étiquette à l'objet `Tk` root, nous l'ajoutons à un objet `Frame` qui contient l'étiquette et d'autres éléments qui constitueront en fin de compte la fenêtre de notre application. La structure de l'interface utilisateur est illustrée à la figure 7.3. Finalement, elle contiendra tous les éléments indiqués.

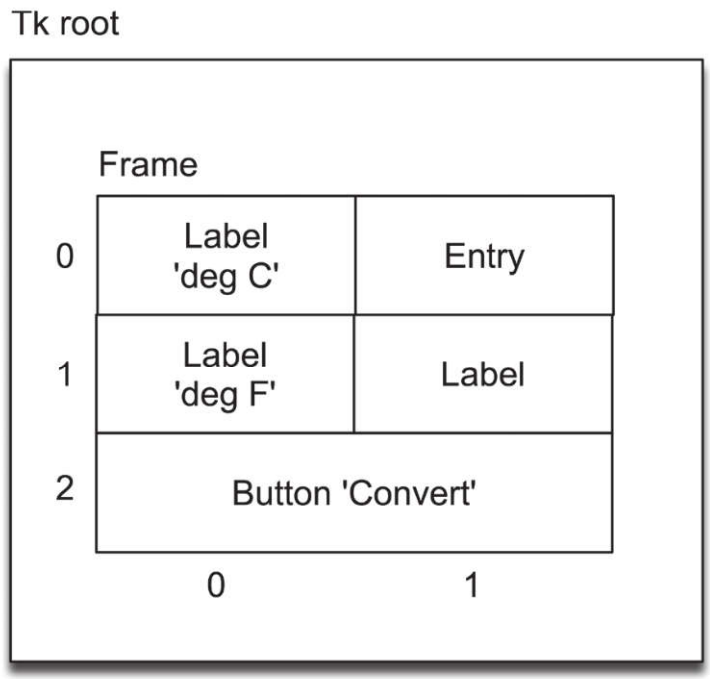


Figure 7.3
Structure de l'interface utilisateur.

Copyright © 2014 Dunod.

L'objet Frame (cadre) est « emballé » dans l'objet root, mais cette fois-ci, quand on ajoute l'étiquette, on utilise la méthode `grid` au lieu de `pack`. Cela nous permet de spécifier une présentation en forme de grille des parties de notre interface utilisateur. Le champ passe à la position 0, 0 sur la grille, et l'objet bouton qui est créé à la ligne suivante se positionne sur la deuxième ligne de la grille (ligne 1). La définition du bouton spécifie également une « commande » à exécuter lorsque l'on clique sur le bouton. Pour le moment, c'est juste un stub qui affiche le message « Non implémenté ».

La fonction `wm_title` définit le titre de la fenêtre. La figure 7.4 montre l'aspect de l'interface utilisateur à ce stade.



Figure 7.4
Interface utilisateur de base pour l'application de convertisseur de température.

L'étape suivante consiste à remplir le reste de l'interface utilisateur. Nous avons besoin d'une « entrée » dans laquelle on puisse saisir une valeur en degrés C et de deux étiquettes supplémentaires. Nous avons besoin d'une étiquette permanente qui se contente d'afficher « degré F » et d'une étiquette à sa droite où s'affiche la température convertie.

Tkinter a une façon spéciale de lier les champs de l'interface utilisateur avec les valeurs qu'ils contiennent. Quand on a besoin de récupérer ou de définir ces valeurs, on crée une instance d'une variable objet spéciale. Il existe plusieurs versions de cette variable objet et la plus courante est StringVar. Cependant, comme nous saisissons et affichons des nombres, nous allons utiliser DoubleVar. *Double* signifie qu'il s'agit d'un nombre à virgule flottante en double précision. C'est comme un nombre en virgule flottante, mais encore plus précis.

Après avoir ajouté le reste des contrôles d'interface utilisateur et les variables permettant d'interagir avec les contrôles, le programme ressemblera à ceci :

```
#07_03_temp_ui.py

from tkinter import *
```

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

```
class App:

    def __init__(self, master):
     frame = Frame(master)
     frame.pack()
     Label(frame, text='degré C').grid(row=0, column=0)
     self.c_var = DoubleVar()
     Entry(frame, textvariable=self.c_var).grid(row=0, column=1)
     Label(frame, text='degré F').grid(row=1, column=0)
     self.result_var = DoubleVar()
     Label(frame, textvariable=self.result_var).grid(row=1,
column=1)
     button = Button(frame, text='Convertir', command=self.convert)
     button.grid(row=2, columnspan=2)

    def convert(self):
     print('Non implémenté')

root = Tk()
root.wm_title('Convertisseur de température')
app = App(root)
root.mainloop()
```

Le premier objet DoubleVar (`c_var`) est affecté au champ de saisie avec une propriété `textvariable`. Cela signifie que l'entrée va afficher le contenu de DoubleVar, et que si la valeur de DoubleVar est modifiée, l'affichage du champ se met automatiquement à jour pour refléter la nouvelle valeur. De plus, lorsque l'utilisateur saisit quelque chose dans le champ, la valeur de DoubleVar est modifiée. Notez qu'une nouvelle étiquette mentionnant « degré F » a également été ajoutée.

Le deuxième objet DoubleVar est lié à une autre étiquette qui va afficher par la suite le résultat du calcul. Nous avons ajouté un autre attribut à la commande `grid` qui affiche le bouton. Comme nous spécifions `columnspan = 2`, le bouton occupera les deux colonnes.

Si vous exécutez le programme, il affiche l'interface utilisateur finalisée, mais lorsque vous cliquez sur le bouton Convertir, le message « Non implémenté » s'affiche sur la console Python.

La dernière étape consiste à remplacer le stub de la méthode `convertir` par une véritable méthode qui utilise le module `convertisseurs` du chapitre 5. Pour ce faire, nous devons importer le module. Afin de réduire notre travail de saisie, nous allons tout importer de la manière suivante :

```
from convertisseurs import *
```

Copyright © 2014 Dunod.

Par souci d'efficacité, il vaut mieux, si nous créons un seul « convertisseur » dans `__init__`, utiliser le même chaque fois que l'on clique sur le bouton. On crée par conséquent une variable nommée `self.t_conv` pour faire référence au convertisseur. La méthode `convertir` se réduit finalement à ceci :

```
def convertir(self):
    c = self.c_var.get()
    self.result_var.set(self.t_conv.convertir(c))
```

Voici le listing complet du programme :

```
#07_04_temp_final.py

from tkinter import *
from convertisseurs import *

class App:

    def __init__(self, master):
     self.t_conv = ConvertisseurEchelleEtDecalage('C', 'F', 1.8, 32)
     frame = Frame(master)
     frame.pack()
     Label(frame, text='degré C').grid(row=0, column=0)
     self.c_var = DoubleVar()
     Entry(frame, textvariable=self.c_var).grid(row=0, column=1)
     Label(frame, text='degré F').grid(row=1, column=0)
     self.result_var = DoubleVar()
     Label(frame, textvariable=self.result_var).grid(row=1,
column=1)
     button = Button(frame, text='Convertir', command=self.conver-
tir)
     button.grid(row=2, columnspan=2)

    def convertir(self):
     c = self.c_var.get()
     self.result_var.set(self.t_conv.convertir(c))

root = Tk()
root.wm_title('Convertisseur de température')
app = App(root)
root.mainloop()
```

7.4 AUTRES CONTRÔLES D'INTERFACE UTILISATEUR

Dans le convertisseur de température, nous n'avons utilisé que des champs texte (classe `Entry`) et des étiquettes (classe `Label`). Comme vous vous en doutez, vous pouvez employer beaucoup

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2014 Dunod.

d'autres contrôles d'interface utilisateur dans votre application. La figure 7.5 présente l'écran principal d'une application de test qui illustre la plupart des contrôles utilisables dans Tkinter. Ce programme est disponible dans le fichier 07_05_controles.py.



Figure 7.5
Application illustrant les contrôles disponibles dans Tkinter.

Checkbutton

Le contrôle Checkbutton (case à cocher qui figure à la première colonne, deuxième ligne de la figure 7.5) est créé de la manière suivante :

```
Checkbutton(frame, text='Checkbutton')
```

Cette ligne de code crée juste un objet Checkbutton avec une étiquette à côté. Si nous avons fait l'effort de placer une case à cocher sur la fenêtre, nous devons également trouver un moyen de savoir si elle est cochée.

Pour réaliser cela, on utilise une « variable » spéciale comme nous l'avons fait dans l'exemple du convertisseur de température. Dans l'exemple suivant, on emploie une StringVar, mais si les valeurs de `onvalue` et `offvalue` avaient été des nombres, on aurait pu utiliser une IntVar à la place.

```
check_var = StringVar()
check = Checkbutton(frame, text='Checkbutton', variable=check_var,
onvalue='Y', offvalue='N')
check.grid(row=1, column=0)
```

Listbox

Pour afficher une liste d'articles dans laquelle on peut sélectionner un ou plusieurs éléments, un contrôle Listbox est utilisé (voir le centre de la figure 7.5). Voici un exemple :

```
listbox = Listbox(frame, height=3, selectmode=BROWSE)
for item in ['rouge', 'vert', 'bleu', 'jaune', 'rose']:
   listbox.insert(END, item)
listbox.grid(row=1, column=1)
```

Copyright © 2014 Dunod.

Dans ce cas, il se contente d'afficher une liste de couleurs. Chaque chaîne doit être ajoutée à la liste individuellement. Le mot `END` indique que l'élément doit se placer à la fin de la liste.

Vous pouvez contrôler la façon dont les sélections sont faites dans la liste à l'aide de la propriété `selectmode`, qui peut prendre les options suivantes :

- `SINGLE` Une seule sélection à la fois.
- `BROWSE` Similaire à SINGLE, mais permet de sélectionner à l'aide de la souris. Il semblerait que cette différence ne fonctionne pas avec l'implémentation de Tkinter sur le Raspberry Pi.
- `MULTIPLE` MAJ-clic pour sélectionner plusieurs lignes.
- `EXTENDED` Similaire à MULTIPLE, mais permet également la sélection CTRL-MAJ-clic sur des plages.

Contrairement aux autres contrôles qui utilisent une StringVar ou un autre type de variable spéciale pour lire des valeurs, il faut employer la méthode `curselection` pour connaître les éléments de la liste qui sont sélectionnés. Cette méthode renvoie une collection des indices sélectionnés. Si le premier, le deuxième et le quatrième élément de la liste sont sélectionnés, vous obtiendrez donc la liste suivante :

```
[0, 1, 3]
```

Lorsque `selectmode` a la valeur `SINGLE`, vous obtenez toujours une liste, mais elle ne contient qu'une seule valeur.

Spinbox

Le contrôle Spinbox fournit un autre moyen de faire une sélection unique dans une liste :

```
Spinbox(frame, values=('a','b','c')).grid(row=3)
```

La méthode `get` retourne l'élément actuellement affiché dans le contrôle, mais pas son indice de sélection.

Mise en page

Disposer les différentes parties de votre application pour que tout s'affiche correctement, même lorsque vous redimensionnez la fenêtre, est une des parties les plus difficiles de la création d'une interface graphique.

Il vous arrivera souvent de placer un type de mise en page à l'intérieur d'une autre mise en page. Par exemple, la forme globale de notre application de démonstration des contrôles est une grille de

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

3 × 3, mais au sein de cette grille il y a un autre cadre pour deux boutons radio :

```
radio_frame = Frame(frame)
radio_selection = StringVar()
b1 = Radiobutton(radio_frame, text='portrait',
   variable=radio_selection, value='P')
b1.pack(side=LEFT)
b2 = Radiobutton(radio_frame, text='paysage',
   variable=radio_selection, value='L')
b2.pack(side=LEFT)
radio_frame.grid(row=1, column=2)
```

Cette approche est assez fréquente, et c'est une bonne idée de faire une ébauche de la disposition de vos contrôles sur papier avant de commencer à écrire le code.

Un problème particulier se pose lors de la création d'une interface graphique : comment contrôler ce qui se passe lorsque la fenêtre est redimensionnée ? Normalement, il est souhaitable de garder certains contrôles au même endroit et de conserver leur taille, tout en permettant à d'autres contrôles de s'agrandir.

Pour prendre un exemple, nous pouvons créer une fenêtre simple comme celle qui est illustrée à la figure 7.6, qui a une zone de liste (à gauche) qui conserve sa taille et une zone de message extensible (à droite) qui s'agrandit au fur et à mesure que la fenêtre est redimensionnée.

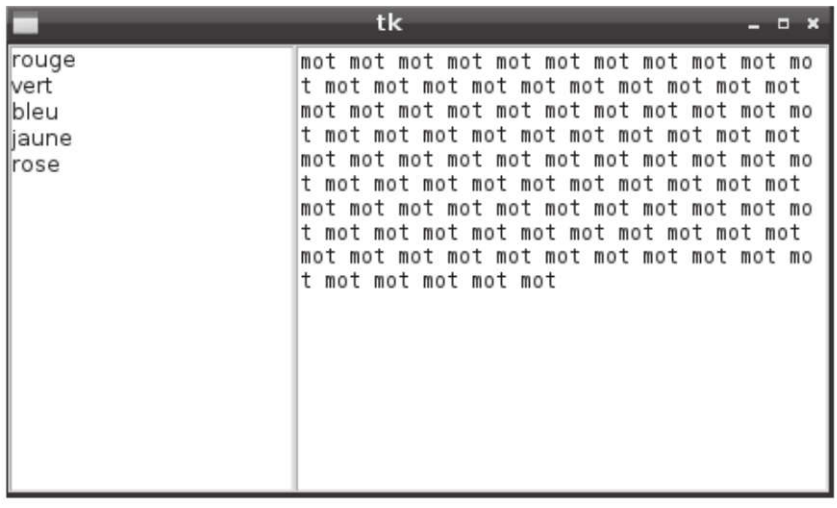


Figure 7.6
Exemple de redimensionnement d'une fenêtre.

Voici le code qui réalise cela :

```
#07_06_redimensionner.py

from tkinter import *
```

Copyright © 2014 Dunod.

```
class App:

    def __init__(self, master):
     frame = Frame(master)
     frame.pack(fill=BOTH, expand=1)
     #Listbox
     listbox = Listbox(frame)
     for item in ['rouge', 'vert', 'bleu', 'jaune', 'rose']:
        listbox.insert(END, item)
     listbox.grid(row=0, column=0, sticky=W+E+N+S)

     #Message
     text = Text(frame, relief=SUNKEN)
     text.grid(row=0, column=1, sticky=W+E+N+S)
     text.insert(END, 'mot ' * 100)
     frame.columnconfigure(1, weight=1)
     frame.rowconfigure(0, weight=1)
root = Tk()
app = App(root)
root.geometry(«400x300+0+0»)
root.mainloop()
```

La clé pour comprendre ces mises en page est l'utilisation des attributs `sticky` des composants : vous devez décider sur quelles bordures de leur cellule de la grille ils doivent adhérer. Pour contrôler les colonnes et les lignes qui doivent s'agrandir lorsque la fenêtre est redimensionnée, vous utilisez les commandes `columnconfigure` et `rowconfigure`. La figure 7.7 montre la disposition des contrôles qui composent cette fenêtre. Les lignes indiquent l'emplacement où le bord des contrôles doit « coller » à la bordure de leur cellule.

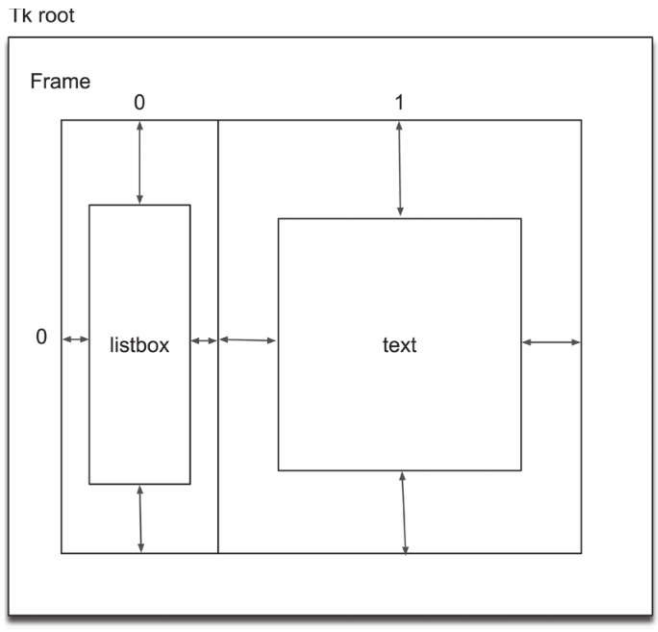


Figure 7.7
Exemple de disposition pour un redimensionnement de fenêtre.

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Examinons le code de cet exemple pour que les choses commencent à faire sens. Tout d'abord, la ligne

```
frame.pack(fill=BOTH, expand=1)
```

veille à ce que le cadre remplisse la fenêtre root de telle sorte que le cadre reflète les changements de taille de la fenêtre.

Après avoir créé la zone de liste, nous l'ajoutons à la disposition en grille du cadre à l'aide de la ligne suivante :

```
listbox.grid(row=0, column=0, sticky=W+E+N+S)
```

Cela spécifie que la Listbox doit se positionner en ligne 0, colonne 0, mais l'attribut `sticky` précise que les bords ouest (W), est (E), nord (N) et sud (S) de la Listbox doivent rester connectés à la grille. Les constantes W, E, N et S sont des constantes numériques qui peuvent être employées dans n'importe quel ordre. Le contrôle Text est ajouté à la grille de la même manière, et son contenu est initialisé avec le mot *mot* répété 100 fois.

La dernière partie du puzzle consiste à programmer le comportement de redimensionnement que nous souhaitons pour la zone de texte qui s'étend vers la droite et à faire en sorte que la zone de liste ne bouge pas. Pour ce faire, nous utilisons les méthodes `columnconfigure` et `rowconfigure` :

```
frame.columnconfigure(1, weight=1)
frame.rowconfigure(0, weight=1)
```

Par défaut, les lignes et les colonnes ne s'agrandissent pas du tout lorsque leur contrôle est redimensionné. Nous ne voulons pas que la colonne 0 s'agrandisse et nous pouvons donc laisser les choses en l'état. En revanche, nous souhaitons que la colonne 1 s'élargisse vers la droite, et nous voulons que la ligne 0 (seulement la ligne) soit en mesure de s'étendre vers le bas. On réalise cela en leur attribuant un coefficient à l'aide des méthodes `columnconfigure` et `rowconfigure`. Si, par exemple, nous avons plusieurs colonnes que nous voulons étendre uniformément, on leur donne le même coefficient (il s'agit de l'attribut `weight` qui est généralement à 1). Cependant, si nous voulons qu'une des colonnes s'élargisse deux fois plus que l'autre, on lui donne un coefficient double. Dans notre exemple, nous avons seulement une colonne et une ligne qui ont besoin d'expansion, si bien qu'elles peuvent toutes les deux avoir un coefficient de 1.

Copyright © 2014 Dunod.

Scrollbar

Si vous rétrécissez la fenêtre du programme 07_06_redimensionner.py, vous remarquerez qu'il n'y a aucune barre de défilement pour accéder au texte qui est caché. Vous pouvez toujours voir le texte, mais une barre de défilement faciliterait franchement les choses.

Les barres de défilement sont des contrôles (de type Scrollbar) autonomes et l'astuce pour les faire fonctionner avec un autre contrôle (par exemple Text, Message ou Listbox) est de les disposer côte à côte puis de les lier.

La figure 7.8 illustre un contrôle Text avec une barre de défilement.

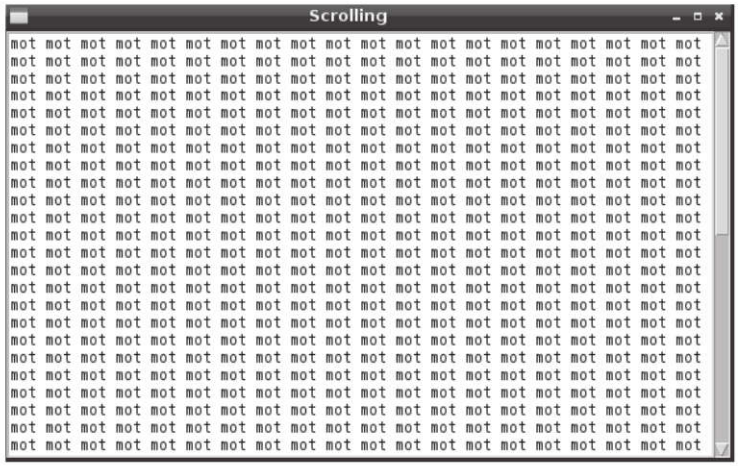


Figure 7.8
Contrôle Text avec une barre de défilement.

Voici le code pour arriver à ce résultat :

```
#07_07_defilement.py

from tkinter import *

class App:

    def __init__(self, master):
     scrollbar = Scrollbar(master)
     scrollbar.pack(side=RIGHT, fill=Y)
     text = Text(master, yscrollcommand=scrollbar.set)
     text.pack(side=LEFT, fill=BOTH)
     text.insert(END, 'mot ' * 1000)
     scrollbar.config(command=text.yview)

root = Tk()
root.wm_title('Scrolling')
app = App(root)
root.mainloop()
```

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Dans cet exemple, nous utilisons la disposition `pack`, en positionnant la barre de défilement sur la droite et la zone de texte sur la gauche. L'attribut `fill` spécifie que le contrôle Text est autorisé à utiliser tout l'espace libre dans *les deux* dimensions X et Y. Pour lier le contrôle Scrollbar au contrôle Text, nous affectons à la propriété `yscrollcommand` du contrôle Text la méthode `set` de la barre de défilement. De la même manière, l'attribut `command` de la barre de défilement est défini à `text.yview`.

7.5 BOÎTES DE DIALOGUE

Il est parfois utile d'afficher une petite fenêtre avec un message et de demander à l'utilisateur de cliquer sur OK avant qu'il ne puisse faire autre chose (voir la figure 7.9). Ces fenêtres sont appelées des *boîtes de dialogue modales* et Tkinter en a toute une série dans le paquet `tkinter.messagebox`.

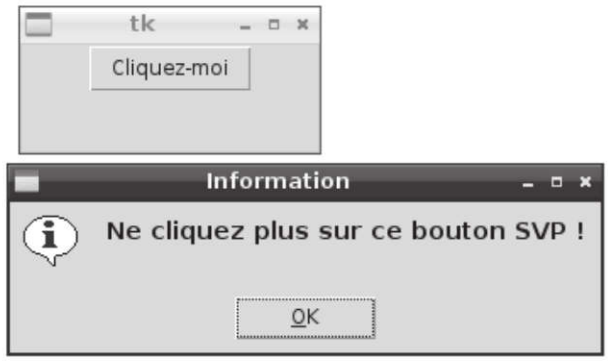


Figure 7.9
Boîte de dialogue d'alerte.

L'exemple suivant montre comment afficher une telle alerte. Outre `showinfo`, `tkinter.messagebox` a aussi les fonctions `showwarning` et `showerror` qui fonctionnent de manière identique, mais qui affichent un symbole différent dans la fenêtre.

```
#07_08_dialogue.py

from tkinter import *
import tkinter.messagebox as mb

class App:

    def __init__(self, master):
     b=Button(master, text='Cliquez-moi', command=self.info).pack()

    def info(self):
     mb.showinfo('Information', "Ne cliquez plus sur ce bouton SVP !")
```

Copyright © 2014 Dunod.

```
root = Tk()
app = App(root)
root.mainloop()
```

On trouvera d'autres types de boîtes de dialogue dans les paquets `tkinter.colorchooser` et `tkinter.filedialog`.

Sélecteur de couleurs

Le sélecteur de couleurs retourne une couleur sous la forme de composants RVB séparés, mais aussi une chaîne de couleurs hexadécimale standard (voir la figure 7.10).

```
#07_09_choix_couleur.py

from tkinter import *
import tkinter.colorchooser as cc

class App:

    def __init__(self, master):
     b=Button(master, text='Couleur..', command=self.ask_color).
pack()

    def ask_color(self):
     (rgb, hx) = cc.askcolor()
     print("rgb=" + str(rgb) + "hx=" + hx)

root = Tk()
app = App(root)
root.mainloop()
```

Le code ci-dessus renvoie quelque chose comme ceci :

```
rgb=(255.99609375, 92.359375, 116.453125) hx=#ff5c74
```

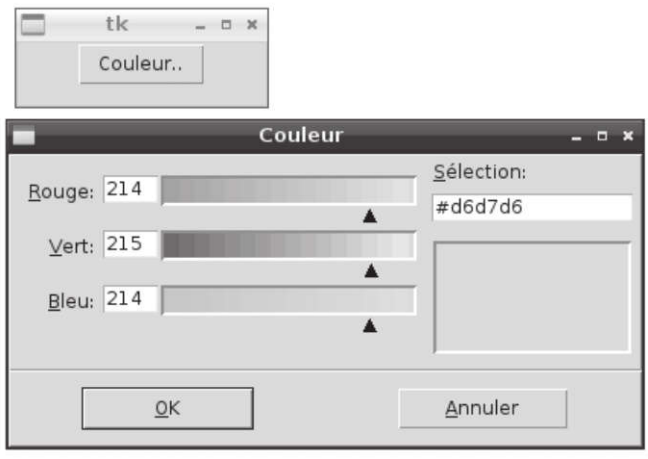


Figure 7.10
Sélecteur de couleurs.

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Sélecteur de fichiers

Les sélecteurs de fichiers se trouvent dans le paquet `tkinter.filedialog`. Ils suivent exactement le même modèle que les autres boîtes de dialogue que nous avons examinées.

7.6 MENUS

Vous pouvez doter vos applications de menus. À titre d'exemple, nous pouvons créer une application très simple avec un champ de saisie et deux options de menu (voir la figure 7.11).

```
#07_10_menus.py

from tkinter import *

class App:

    def __init__(self, master):
     self.entry_text = StringVar()
     Entry(master, textvariable=self.entry_text).pack()

     menubar = Menu(root)

     filemenu = Menu(menubar, tearoff=0)
     filemenu.add_command(label='Quitter', command=exit)
     menubar.add_cascade(label='Fichier', menu=filemenu)

     editmenu = Menu(menubar, tearoff=0)
     editmenu.add_command(label='Remplir', command=self.fill)
     menubar.add_cascade(label='Editer', menu=editmenu)

     master.config(menu=menubar)

    def fill(self):
     self.entry_text.set('abc')

root = Tk()
app = App(root)

root.mainloop()
```



Figure 7.11
Menus.

Copyright © 2014 Dunod.

La première étape consiste à créer un `Menu` racine. C'est l'objet qui contiendra tous les menus (Fichier et Edition, dans notre exemple, ainsi que toutes les options de menu).

Pour créer le menu Fichier, avec sa seule option, Quitter, on crée d'abord une autre instance de `Menu` puis on ajoute une commande pour quitter et enfin on ajoute le menu Fichier au `Menu` racine :

```
filemenu = Menu(menubar, tearoff=0)
filemenu.add_command(label='Quitter', command=exit)
menubar.add_cascade(label='Fichier', menu=filemenu)
```

Le menu Edition est créé de la même façon. Pour faire apparaître les menus sur la fenêtre, il faut utiliser la commande suivante :

```
master.config(menu=menubar)
```

7.7 LE CANVAS

Au chapitre suivant, vous bénéficierez d'une brève introduction à la programmation des jeux à l'aide de Pygame qui permet toutes sortes de jolis effets graphiques. Toutefois, si vous avez juste besoin de créer des graphiques simples, tels que le dessin de formes ou le traçage de graphiques à l'écran, vous pouvez utiliser l'interface Canvas de Tkinter à la place (voir la figure 7.12).

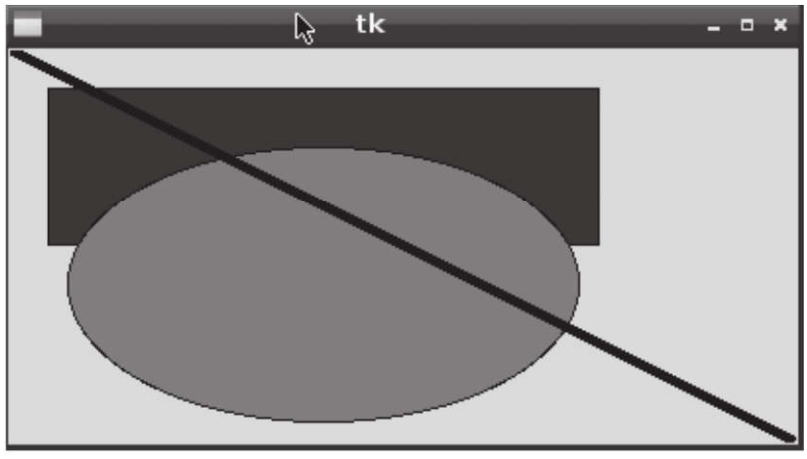


Figure 7.12
Contrôle Canvas.

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Le Canvas est comme n'importe quel autre contrôle que vous pouvez ajouter à une fenêtre. L'exemple suivant indique comment dessiner des lignes, des rectangles et ovales :

```
#07_11_canvas.py

from tkinter import *

class App:

    def __init__(self, master):
     canvas = Canvas(master, width=400, height=200)
     canvas.pack()
     canvas.create_rectangle(20, 20, 300, 100, fill='blue')
     canvas.create_oval(30, 50, 290, 190, fill='#ff2277')
     canvas.create_line(0, 0, 400, 200, fill='black', width=5)
root = Tk()
app = App(root)
root.mainloop()
```

Vous pouvez dessiner des arcs, des images, des polygones et du texte de manière similaire. Reportez-vous à la référence de Tkinter en ligne à http://infohost.nmt.edu/tcc/help/pubs/tkinter/ pour plus d'informations.

Note : L'origine des coordonnées est le coin supérieur gauche de la fenêtre, et les coordonnées sont en pixels.

7.8 RÉSUMÉ

Dans un livre de cette taille, il faut parfois se contenter d'une présentation sommaire de certains sujets qui vous indiquent seulement la voie à suivre. Une fois que vous avez suivi les exemples de ce chapitre, exécutez-les, modifiez-les et analysez ce qui se passe, et vous serez bientôt avide d'informations supplémentaires. Vous avez passé le stade où l'on doit vous tenir la main et vous avez à présent une idée précise des programmes que vous voulez écrire, mais aucun livre ne vous dira exactement comment construire le projet que vous avez en tête. C'est là où Internet devient très utile. Les références en ligne suivantes vous permettront de poursuivre votre apprentissage :

- *www.pythonware.com/bibliothèque/tkinter/introduction/*
- *http://infohost.nmt.edu/tcc/help/pubs/tkinter/*

Copyright © 2014 Dunod.

CHAPITRE PAGE

8 PROGRAMMATION DE JEUX

1	Introduction	3
2	Prise en main	17
3	Bases de Python	27
4	Chaînes, listes et dictionnaires	43
5	Modules, classes et méthodes	69
6	Fichiers et Internet	81
7	Interfaces utilisateur graphiques	91
8	Programmation de jeux	109

8.1 Pygame **110**
8.2 Hello Pygame **110**
8.3 Un jeu avec des framboises **112**
8.4 Résumé **121**

9	Interfaçage du matériel	123
10	Projet de prototypage d'une horloge	137
11	Le RaspiRobot	147
12	Pour aller plus loin	161

Copyright © 2014 Dunod.

Il est évident qu'un chapitre unique sur la programmation des jeux ne fera pas de vous un expert du domaine. Si vous voulez creuser la question, il existe un certain nombre de bons livres qui sont consacrés spécifiquement à la programmation en Python, notamment *Beginning Game Development with Python and Pygame* de Will McGugan. Ce chapitre vous présente une bibliothèque très pratique, appelée pygame, et vous initie à son utilisation pour créer un jeu simple.

8.1 PYGAME

Pygame est une bibliothèque qui facilite l'écriture des jeux pour le Raspberry Pi (ou plus généralement pour les ordinateurs qui exécutent le langage Python). Une bibliothèque est utile parce que la plupart des jeux ont certains éléments en commun et que l'on rencontre les mêmes difficultés lors de l'écriture des jeux. Une bibliothèque comme pygame allège ces difficultés et permet notamment :

- de dessiner des graphiques qui ne scintillent pas.
- de contrôler l'animation de telle sorte qu'elle fonctionne à la même vitesse qu'on l'exécute sur un Raspberry Pi ou sur un PC de jeu surpuissant.
- d'intercepter les événements de clavier et souris pour contrôler le jeu.

La distribution Raspbian Wheezy est livrée avec deux versions de Python : Python 2 et Python 3. C'est la raison pour laquelle deux raccourcis vers IDLE apparaissent sur le bureau. Jusqu'à présent, dans ce livre, nous avons utilisé la version 3 de IDLE et par conséquent, Python 3. Avec la distribution Raspbian Wheezy, l'installation de Python 3 n'inclut pas pygame, alors que c'est le cas avec l'installation de Python 2.

Au lieu d'installer pygame dans Python 3 (ce qui est un peu délicat), nous allons utiliser Python 2 dans ce chapitre. Ne vous inquiétez pas car tout le code que nous allons écrire devrait fonctionner avec Python 3. Il faut simplement ne pas oublier de lancer IDLE au lieu de IDLE 3.

8.2 HELLO PYGAME

Vous pouvez également avoir un raccourci sur votre bureau appelé Python Games. Ce raccourci exécute un programme qui vous permet d'exécuter certains jeux écrits en Python. Si vous utilisez l'Explorateur de fichiers, vous trouverez également un dossier dans votre répertoire racine appelé python_games. Si vous

Copyright © 2014 Dunod.

regardez son contenu, vous pourrez y voir les fichiers .py des jeux, et vous pourrez ouvrir ces fichiers dans IDLE pour regarder comment ces jeux ont été écrits. La figure 8.1 montre à quoi ressemble une application de type Hello World en pygame. Voici le code de cette application :

```
#08_01_hello_pygame.py

import pygame

pygame.init()

screen = pygame.display.set_mode((200, 200))
screen.fill((255, 255, 255))
pygame.display.set_caption('Hello Pygame')

raspberry = pygame.image.load('raspberry.jpg').convert()
screen.blit(raspberry, (100, 100))

pygame.display.update()
```



Figure 8.1
Programme Hello Pygame.

Il s'agit d'un exemple très rudimentaire qui n'a aucun moyen de quitter proprement l'application. Le fait de fermer la console Python à partir de laquelle ce programme a été lancé devrait mettre fin à son exécution au bout de quelques secondes.

En regardant le code de cet exemple, vous pouvez constater que la première chose réalisée est l'importation de la bibliothèque pygame. La méthode `init` (abréviation de *initialisation*) est ensuite exécutée pour que pygame soit prêt à être utilisé. On initialise ensuite une variable appelée `screen` à l'aide de la ligne suivante :

```
screen = pygame.display.set_mode((200, 200))
```

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Cela crée une nouvelle fenêtre de 200 x 200 pixels que l'on remplit ensuite avec du blanc (la couleur 255, 255, 255) sur la ligne suivante avant de définir la légende de la fenêtre à « Hello Pygame ».

Les jeux utilisent des graphismes, c'est-à-dire en général des images. Dans cet exemple, nous lisons un fichier image à l'aide de pygame :

```
raspberry = pygame.image.load('raspberry.jpg').convert()
```

Dans cet exemple, l'image est un fichier appelé raspberry.jpg, qui est inclus avec les programmes de ce livre. L'appel à `convert()` à la fin de la ligne est important, car cela convertit l'image en une représentation interne qui lui permet d'être affichée très rapidement, ce qui est essentiel lorsque nous commencerons à faire se déplacer l'image dans la fenêtre.

Ensuite, on affiche l'image de la framboise sur l'écran aux coordonnées 100, 100 à l'aide de la commande `blit`. Comme avec l'objet Canvas de Tkinter que vous avez manipulé dans le chapitre précédent, les coordonnées commencent à 0, 0 dans le coin supérieur gauche de l'écran.

Enfin, la dernière commande dit à pygame de mettre à jour l'affichage de telle sorte que l'on puisse voir l'image.

8.3 UN JEU AVEC DES FRAMBOISES

Afin de vous montrer comment pygame peut être utilisé pour créer un jeu simple, nous allons pas à pas construire un jeu où l'on doit attraper des framboises qui tombent de l'écran à l'aide d'une cuillère. Les framboises tombent à des vitesses différentes et doivent être récupérées dans le creux de la cuillère avant de toucher le sol. La figure 8.2 montre le jeu finalisé en pleine action. C'est rudimentaire, mais fonctionnel.

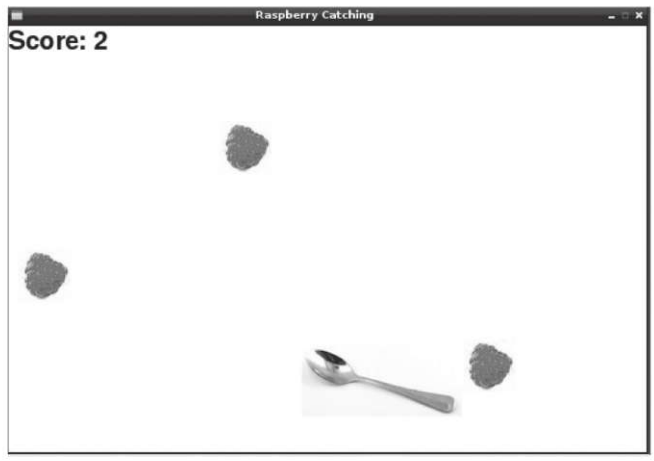


Figure 8.2
Le jeu des framboises.

Copyright © 2014 Dunod.

Suivre la souris

Nous allons commencer par développer le jeu en créant l'écran principal avec l'affichage d'une cuillère (en anglais *spoon*) qui suit les mouvements de la souris de gauche à droite. Chargez le programme suivant dans IDLE :

```
#08_02_rasp_game_mouse

import pygame
from pygame.locals import *
from sys import exit

spoon_x = 300
spoon_y = 300

pygame.init()

screen = pygame.display.set_mode((600, 400))
pygame.display.set_caption('Raspberry Catching')

spoon = pygame.image.load('spoon.jpg').convert()

while True:

    for event in pygame.event.get():
        if event.type == QUIT:
            exit()

    screen.fill((255, 255, 255))
    spoon_x, ignore = pygame.mouse.get_pos()
    screen.blit(spoon, (spoon_x, spoon_y))

    pygame.display.update()
```

La structure de base de notre programme Hello World est toujours là, mais vous avez de nouvelles choses à découvrir. Tout d'abord, il y a quelques importations supplémentaires. L'importation de `pygame.locals` donne accès aux constantes utiles, comme `QUIT`, que nous utiliserons pour détecter quand le jeu est sur le point de se terminer. L'importation de `exit` à partir de `sys` permet de quitter le programme proprement.

Nous avons ajouté deux variables (`spoon_x` et `spoon_y`) pour stocker la position de la cuillère. Comme la cuillère va seulement se déplacer de gauche à droite, `spoon_y` ne changera jamais.

À la fin du programme, il y a une boucle `while`. À chaque tour de la boucle, on vérifie tout d'abord la présence d'un événement QUIT en provenance du système pygame. Les événements se produisent

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

chaque fois que le joueur déplace la souris ou appuie sur une touche ou la relâche. Dans cet exemple, on ne s'intéresse qu'à l'événement `QUIT`, qui est provoqué quand l'utilisateur clique sur l'icône de fermeture de fenêtre dans le coin supérieur droit de la fenêtre. On aurait pu ici choisir de ne pas quitter immédiatement et de demander au joueur s'il voulait vraiment quitter le jeu. La ligne suivante efface l'écran en le remplissant avec la couleur blanche.

Vient ensuite une affectation dans laquelle on définit `spoon_x` avec la valeur de la position x de la souris. Notez que bien qu'il s'agisse d'une double affectation, on ne se soucie pas de la position y de la souris, si bien que l'on ignore la deuxième valeur de retour en lui assignant une variable appelée `ignore` que nous ignorons par la suite. On affiche ensuite la cuillère à l'écran et on met à jour l'affichage.

Exécutez ce programme et vous devriez voir la cuillère suivre la souris.

Une framboise

L'étape suivante de la construction du jeu consiste à ajouter une framboise. Par la suite, nous allons étendre le jeu afin qu'il y ait trois framboises qui tombent à la fois, mais nous allons commencer avec une seule car c'est plus facile. Le code de ce programme se trouve dans le fichier 08_03_rasp_game_one.py.

Voici les changements par rapport à la version précédente :

- Ajout de variables globales pour la position de la framboise (`raspberry_x` et `raspberry_y`).
- Chargement et conversion de l'image raspberry.jpg.
- Séparation de la mise à jour de la cuillère dans sa propre fonction.
- Ajout d'une nouvelle fonction appelée `update_raspberry`.
- Mise à jour de la boucle principale afin d'utiliser les nouvelles fonctions.

Vous devriez déjà être familier avec les deux premiers éléments de ce listing et nous allons donc commencer par les nouvelles fonctions :

```
def update_spoon():
    global spoon_x
    global spoon_y
    spoon_x, ignore = pygame.mouse.get_pos()
    screen.blit(spoon, (spoon_x, spoon_y))
```

Copyright © 2014 Dunod.

La fonction `update_spoon` se contente de reprendre le code que nous avions dans la boucle principale du fichier 08_02_rasp_game_mouse et le place dans une fonction autonome. Cela réduit la taille de la boucle principale de telle sorte que l'on sait plus facilement ce qui s'y passe.

```
def update_raspberry():
    global raspberry_x
    global raspberry_y
    raspberry_y += 5
    if raspberry_y > spoon_y:
        raspberry_y = 0
        raspberry_x = random.randint(10, screen_width)
    raspberry_x += random.randint(-5, 5)
    if raspberry_x < 10:
        raspberry_x = 10
    if raspberry_x > screen_width - 20:
        raspberry_x = screen_width - 20
    screen.blit(raspberry, (raspberry_x, raspberry_y)
```

La fonction `update_raspberry` modifie les valeurs de `raspberry_x` et `raspberry_y`. Elle ajoute 5 à la position *y* pour déplacer la framboise vers le bas de l'écran et déplace la position *x* d'un nombre aléatoire compris entre –5 et +5. Cela rend la trajectoire des framboises imprévisibles au cours de leur descente. Toutefois, les framboises finissent par tomber au bas de l'écran, si bien qu'une fois que la position *y* est supérieure à la position de la cuillère, la fonction la replace en haut de l'écran à une nouvelle position *x* aléatoire.

Il y a aussi un risque que les framboises disparaissent du cadre de l'écran, à gauche ou à droite. Par conséquent, il y a deux tests qui vérifient que les framboises ne sont pas trop proches du bord de l'écran, et si tel est le cas, elles ne sont alors pas autorisées à aller plus loin à gauche ou à droite. Voici la nouvelle boucle principale qui appelle ces nouvelles fonctions :

```
while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            exit()

    screen.fill((255, 255, 255))
    update_raspberry()
    update_spoon()
    pygame.display.update()
```

Testez le fichier 08_03_rasp_game_one et vous verrez un programme fonctionnel qui ressemble à une véritable partie. Toutefois, il ne se passe rien lorsque vous attrapez une framboise.

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Détection de l'interception et comptabilisation du score

Nous allons maintenant ajouter une zone de message pour afficher le score (c'est-à-dire le nombre de framboises récupérées). Pour ce faire, nous devons être capables de détecter que nous avons pris une framboise. Le programme amélioré qui réalise cela se trouve dans le fichier 08_04_rasp_py_game_scoring.py.

Les principaux changements de cette version sont les deux nouvelles fonctions `check_for_catch` et `display` :

```
def check_for_catch():
    global score
    if raspberry_y >= spoon_y and raspberry_x >= spoon_x and \
      raspberry_x < spoon_x + 50:
      score += 1
    display(«Score: « + str(score))
```

Vous noterez que comme la condition de l'instruction if est très longue, nous avons utilisé la commande de continuation de ligne (\) pour décomposer la ligne de code en deux lignes.

La fonction `check_for_catch` ajoute 1 au score si la framboise est tombée aussi bas que la cuillère (`raspberry_y > = spoon_y`) et que la position *x* de la framboise se situe entre la position *x* (gauche) de la cuillère et la position *x* de la cuillère plus 50 (à peu près la largeur du creux de la cuillère).

Indépendamment de savoir si la framboise est interceptée, le score est affiché en utilisant la fonction `display`. La fonction `check_for_catch` est également ajoutée dans la boucle principale comme une chose supplémentaire à effectuer à chaque tour de boucle. La fonction d'affichage est chargée d'afficher un message à l'écran :

```
def display(message):
    font = pygame.font.Font(None, 36)
    text = font.render(message, 1, (10, 10, 10))
    screen.blit(text, (0, 0))
```

On écrit du texte à l'écran en pygame en créant une police (dans cet exemple, on ne précise aucune famille de police spécifique, mais simplement une taille de 36 points) puis on crée un objet `text` en affichant le contenu de la chaîne `message` dans la police définie. La valeur (10, 10, 10) représente la couleur du texte. Le résultat final contenu dans la variable `text` peut alors être affiché à l'écran de la façon habituelle.

Copyright © 2014 Dunod.

Gestion du temps

Vous avez peut-être remarqué que rien dans ce programme ne contrôle la vitesse à laquelle les framboises tombent du ciel. Nous avons la chance qu'elles tombent à la bonne vitesse sur un Raspberry Pi. Cependant, si nous devions exécuter ce jeu sur un ordinateur plus rapide, elles tomberaient probablement beaucoup trop vite pour qu'on les attrape.

Pour gérer la vitesse, pygame a une horloge interne qui permet de ralentir notre boucle principale de manière suffisamment précise pour permettre d'effectuer un certain nombre d'actualisations par seconde. Malheureusement, on ne peut rien faire pour accélérer notre boucle principale. Cette horloge est très facile à utiliser et il suffit de placer la ligne suivante quelque part avant la boucle principale :

```
clock = pygame.time.Clock()
```

Cela crée une instance de l'horloge. Pour accomplir le ralentissement nécessaire de la boucle principale, placez la ligne suivante quelque part dans la boucle (habituellement à la fin) :

```
clock.tick(30)
```

Dans ce cas, nous utilisons une valeur de 30, ce qui signifie une cadence de 30 images par seconde. Vous pouvez mettre une valeur différente ici, mais l'œil humain (et le cerveau) n'enregistre pas d'amélioration de la qualité au-delà de 30 images par seconde.

Un maximum de framboises

Notre programme commence à être un peu plus complexe. Si nous devions rajouter la gestion de plusieurs framboises à ce stade, il deviendrait alors encore plus difficile de voir ce qui se passe. Nous allons donc effectuer une *refactorisation*, ce qui signifie que nous allons changer un programme qui tourne parfaitement bien en modifiant sa structure sans modifier ses fonctionnalités. Nous allons réaliser cela en créant une classe appelée `Raspberry` pour accomplir toutes les choses que doit faire une framboise. Pour l'instant, cela ne fonctionne toujours qu'avec une seule framboise, mais il sera plus facile de gérer plusieurs framboises ultérieurement. Le code de cette étape se trouve dans le fichier 08_05_rasp_game_refactored.py. Voici la définition de la classe :

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

```
class Raspberry:
    x = 0
    y = 0

    def __init__(self):
        self.x = random.randint(10, screen_width)
        self.y = 0

    def update(self):
        self.y += 5
        if self.y > spoon_y:
            self.y = 0
            self.x = random.randint(10, screen_width)
        self.x += random.randint(-5, 5)
        if self.x < 10:
            self.x = 10
        if self.x > screen_width - 20:
            self.x = screen_width - 20
        screen.blit(raspberry_image, (self.x, self.y))

    def is_caught(self):
        return self.y >= spoon_y and self.x >= spoon_x and \

            self.x < spoon_x + 50
```

Les variables `raspberry_x` et `raspberry_y` sont devenues des variables de la nouvelle classe `Raspberry`. En outre, lorsqu'une instance de framboise est créée, sa position x sera définie de façon aléatoire. L'ancienne fonction `update_raspberry` est devenue une méthode de la classe `Raspberry` appelée simplement `update`. De la même manière, la fonction `check_for_catch` demande maintenant à la framboise si elle a été interceptée.

Après avoir défini une classe `Raspberry`, on en crée une instance de la manière suivante :

```
r = Raspberry()
```

Ainsi, lorsque nous voulons vérifier l'interception d'une framboise, la fonction `check_for_catch` interroge simplement la framboise de la sorte :

```
def check_for_catch():
    global score
    if r.is_caught():
        score += 1
```

L'appel pour afficher le score a aussi été déplacé en dehors de la fonction `check_for_catch` et se trouve dans la boucle principale. Comme tout fonctionne à présent, il est temps d'ajouter plus de

Copyright © 2014 Dunod.

framboises. La version finale du jeu se trouve dans le fichier 08_06_rasp_game_final.py. Voici l'intégralité du listing :

```
#08_06_rasp_game_final

import pygame
from pygame.locals import *
from sys import exit
import random

score = 0

screen_width = 600
screen_height = 400

spoon_x = 300
spoon_y = screen_height - 100

class Raspberry:
    x = 0
    y = 0
    dy = 0

    def __init__(self):
     self.x = random.randint(10, screen_width)
     self.y = 0
     self.dy = random.randint(3, 10)

    def update(self):
     self.y += self.dy
     if self.y > spoon_y:
        self.y = 0
        self.x = random.randint(10, screen_width)
     self.x += random.randint(-5, 5)
     if self.x < 10:
        self.x = 10
     if self.x > screen_width - 20:
        self.x = screen_width - 20
     screen.blit(raspberry_image, (self.x, self.y))

    def is_caught(self):

return self.y >= spoon_y and self.x >= spoon_x and self.x < spoon_x +
50

clock = pygame.time.Clock()
rasps = [Raspberry(), Raspberry(), Raspberry()]

pygame.init()
```

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2014 Dunod.

```
screen = pygame.display.set_mode((screen_width, screen_height))
pygame.display.set_caption('Raspberry Catching')

spoon = pygame.image.load('spoon.jpg').convert()
raspberry_image = pygame.image.load('raspberry.jpg').convert()

def update_spoon():
    global spoon_x
    global spoon_y
    spoon_x, ignore = pygame.mouse.get_pos()
    screen.blit(spoon, (spoon_x, spoon_y))

def check_for_catch():
    global score
    for r in rasps:
     if r.is_caught():
        score += 1

def display(message):
    font = pygame.font.Font(None, 36)
    text = font.render(message, 1, (10, 10, 10))
    screen.blit(text, (0, 0))

while True:
    for event in pygame.event.get():
     if event.type == QUIT:
        exit()

    screen.fill((255, 255, 255))
    for r in rasps:
     r.update()
    update_spoon()
    check_for_catch()
    display(«Score: « + str(score))
    pygame.display.update()
    clock.tick(30)
```

Pour créer plusieurs framboises, la variable `r` unique a été remplacée par une collection appelée `rasps` :

```
rasps = [Raspberry(), Raspberry(), Raspberry()]
```

Cela crée trois framboises ; on pourrait changer cela dynamiquement pendant l'exécution du programme en ajoutant de nouvelles framboises à la liste (ou bien en en supprimant).

Nous devons maintenant faire quelques autres modifications pour gérer plus d'une framboise. Tout d'abord, dans la fonction `check_for_catch`, nous avons besoin de boucler sur toutes les framboises et de demander à chacune d'elle si elle a été interceptée (au lieu de demander à une seule framboise). Deuxièmement,

Copyright © 2014 Dunod.

dans la boucle principale, nous avons besoin d'afficher tous les framboises en bouclant sur chacune d'entre elles et en demandant sa mise à jour.

8.4 RÉSUMÉ

Vous pourrez en apprendre beaucoup plus sur pygame. Le site officiel dont l'adresse est www.pygame.org contient de nombreuses ressources et des exemples de jeux que vous pouvez modifier.

Copyright © 2014 Dunod.

Copyright © 2014 Dunod.

CHAPITRE PAGE

9 INTERFAÇAGE DU MATÉRIEL

1	Introduction	3
2	Prise en main	17
3	Bases de Python	27
4	Chaînes, listes et dictionnaires	43
5	Modules, classes et méthodes	69
6	Fichiers et Internet	81
7	Interfaces utilisateur graphiques	91
8	Programmation de jeux	109
9	Interfaçage du matériel	123

9.1 Connexions des broches GPIO **124**
9.2 Connexion directe aux broches GPIO **125**
9.3 Cartes extension **126**
9.4 Cartes de prototypage **130**
9.5 Arduino et le Pi **132**
9.6 Résumé **135**

10	Projet de prototypage d'une horloge	137
11	Le RaspiRobot	147
12	Pour aller plus loin	161

Copyright © 2014 Dunod.

Le Raspberry Pi a une double rangée de broches sur l'un de ses côtés. Ces broches, que l'on appelle connecteur GPIO (*General Purpose Input/Output*, entrées / sorties à usage général), permettent de connecter du matériel électronique au Pi sans passer par le port USB.

Les communautés des enseignants et des électroniciens ont déjà commencé à produire des cartes d'extension et des cartes de prototypage que vous pouvez relier à votre Pi de manière à pouvoir ajouter vos propres appareils électroniques. Cela comprend de nombreux périphériques qui vont des simples capteurs de température jusqu'aux relais. Vous pouvez même transformer vos Raspberry Pi en contrôleur de robot.

Dans ce chapitre, nous allons examiner les différentes façons de brancher le Pi à des appareils électroniques qui utilisent le GPIO. Nous allons utiliser les premiers produits qui sont commercialisés à cet effet. Parce qu'il s'agit d'un domaine où les évolutions sont rapides, il est certain que de nouveaux produits seront sortis une fois que vous lirez cet ouvrage ; vous devez donc vérifier sur Internet la sortie de nouveaux produits dans ce secteur d'activité. J'ai tenté de sélectionner un ensemble représentatif de différentes approches pour l'interfaçage du matériel. Cela signifie que, même si les versions identiques ne sont pas disponibles, vous aurez au moins une idée de ce qui existe et de la manière de l'utiliser.

Les produits qui vous serviront à relier des composants électroniques à votre Pi peuvent être classés en deux catégories : les outils de prototypage et les cartes d'extension. Avant de nous pencher sur chacune de ces catégories, nous allons examiner en détail le connecteur GPIO.

9.1 CONNEXIONS DES BROCHES GPIO

La figure 9.1 illustre les connexions disponibles sur le connecteur GPIO du Raspberry Pi. Les broches marquées GPIO peuvent être utilisées comme broches d'entrée/sortie. En d'autres termes, n'importe quelle broche peut être programmée comme une entrée ou une sortie. Si la broche est configurée pour être une entrée, vous pouvez ensuite la tester pour voir si elle est à « 1 » (au-dessus de 1,7 V) ou à « 0 » (en dessous de 1,7 V). Notez que toutes les broches GPIO sont en 3,3 V et le fait de les connecter à des tensions supérieures pourrait endommager votre Raspberry Pi.

Copyright © 2014 Dunod.

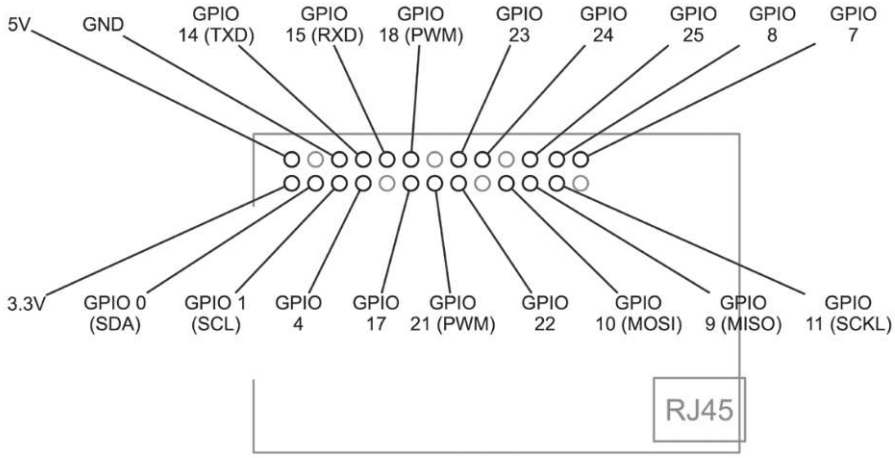


Figure 9.1
Connexions des broches GPIO.

Si vous définissez une broche en sortie, la broche peut être à 0 V ou à 3,3 V (valeur logique 0 ou 1). Les broches ne peuvent fournir qu'une petite quantité de courant (5 mA paraît être une valeur raisonnable), si bien qu'elles peuvent allumer seulement une LED si vous utilisez une résistance de valeur élevée (par exemple, 1 kΩ). Vous remarquerez que certaines broches GPIO ont des lettres entre parenthèses après leur nom. Ces broches ont une fonction spéciale. Par exemple, GPIO 0 et 1 sont les autres noms de SDA et SCL. Il s'agit, respectivement, de l'horloge et des lignes de données d'un type de bus série appelé I2C, qui est populaire pour communiquer avec les périphériques comme des capteurs de température, des écrans LCD, etc. Ce bus I2C est utilisé par le Pi Face et le Slice of PI/O que nous allons étudier dans les sections suivantes.

Les broches GPIO 14 et 15 dupliquent également les broches Rx et Tx (réception et transmission) du port série du Raspberry Pi. La communication série est aussi possible par le biais des broches GPIO 9 à 11 (MOSI, MISO et SCLK). Ce type d'interface série est appelé SPI.

Enfin, les broches GPIO 18 et GPIO 21 comportent l'appellation PWM (*Pulse Width Modulation*), ce qui signifie qu'elles sont capables de modulation de largeur d'impulsion. Cette technique permet de contrôler l'alimentation des moteurs, des LED etc. en faisant varier la largeur des impulsions générées à un rythme constant.

9.2 CONNEXION DIRECTE AUX BROCHES GPIO

Avec précaution, il est possible de relier des composants électroniques simples, comme des LED, directement aux broches GPIO ; vous devez cependant ne le faire que si vous savez parfaitement ce que vous faites car vous pouvez endommager facilement votre Raspberry Pi. En fait, c'est plus ou moins ce que nous ferons dans la section « Cartes de prototypage ».

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

9.3 CARTES EXTENSION

Les cartes d'extension ont généralement des borniers à vis et une certaine quantité d'électronique intégrée. Cela les rend bien adaptées aussi bien pour ceux qui ont des visées éducatives que pour ceux qui ne veulent pas rentrer dans le détail des aspects électroniques. En général, ces cartes ne nécessitent aucune soudure. Elles servent généralement de « tampon » à toutes les connexions du Raspberry Pi, ce qui signifie que le Raspberry Pi est protégé s'il se produit quelque chose de fâcheux sur la carte d'extension. Par exemple, un court-circuit à travers une sortie risquerait d'endommager la carte d'extension, mais aucun dégât n'endommagera votre précieux Pi.

Les sections qui suivent détaillent quelques-unes des cartes les plus populaires, expliquent leurs caractéristiques et illustrent la manière dont vous pouvez les utiliser. Parmi toutes ces cartes, la RaspiRobotBoard nous servira à créer un robot simple au chapitre 11.

Pi Face

Le Pi Face, illustré à la figure 9.2, est une carte destinée principalement à des fins éducatives. Il a été développé par l'Université de Manchester au Royaume-Uni. Tout en fournissant une plateforme matérielle utile, il propose également une bibliothèque Python facile à utiliser et une intégration avec l'environnement de programmation Scratch.

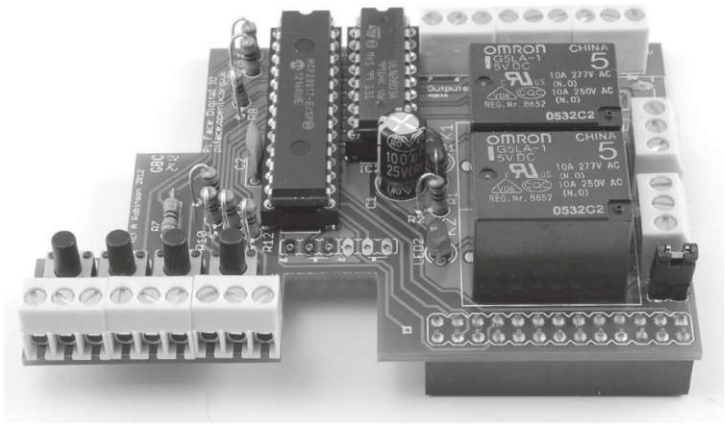


Figure 9.2
Carte d'extension Pi Face.

Le Pi Face s'installe au sommet du Raspberry Pi et fournit des borniers à vis pratiques pour connecter des appareils. Il n'utilise pas les broches GPIO directement, mais utilise à la place une puce expandeur de ports MCP23S17 avec laquelle il communique par le biais de l'interface série I2C. Ceci fournit 8 entrées et 8 sorties

Copyright © 2014 Dunod.

sur la carte d'extension, mais seules les deux broches I2C du connecteur GPIO du Raspberry Pi sont utilisées. Les sorties sont fournies avec une amplification de courant utilisant un circuit Darlington qui peut délivrer jusqu'à 500 ma pour chaque sortie (ce qui est plus que suffisant pour piloter directement un relais ou une LED haute puissance de 1 W).

Les périphériques de sortie sur la carte comprennent deux relais qui peuvent être utilisés pour commuter des courants de charge élevée. Chaque relais possède également une LED qui s'allume lorsque le relais est activé. Il y a aussi deux LED qui peuvent être contrôlées indépendamment. Quatre des entrées possèdent des interrupteurs poussoirs.

Le Pi Face a son propre module Python qui simplifie l'utilisation de la carte. L'exemple suivant, saisi dans la console Python, montre comment lire l'entrée numérique 2 :

```
>>> import piface.pfio as pfio
>>> pfio.init()
>>> pfio.digital_read(2)
0
```

Voici la commande pour activer la sortie numérique 2 :

```
>>> pfio.digital_write(2, 1)
```

Les LED et les relais ont leurs propres fonctions de contrôle. L'exemple suivant allume la LED 1, l'éteint ensuite, puis active le relais 1 :

```
>>> led1 = pfio.LED(1)
>>> led1.turn_on()
>>> led1.turn_off()
>>> relay = pfio.Relay(1)
>>> relay.turn_on()
```

La bibliothèque doit être téléchargée et installée. Pour les téléchargements, la documentation et quelques exemples de projets, visitez la page hébergeant le code du projet à *https://github.com/thomasmacpherson/piface*. Vous trouverez également plus d'informations sur ce projet à *http://pi.cs.Man.AC.uk/interface.htm.*

Slice of PI/O

Le Slice of PI/O, illustré à la figure 9.3, est une petite carte bon marché qui fournit huit entrées et huit sorties bufferisées qui utilisent le même expandeur de ports MCP23S17 que celui du Pi Face. Il n'a pas, cependant, le circuit Darlington du Pi Face et, par

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

conséquent, ne peut pas piloter des charges de forte puissance. La charge maximale délivrée directement à partir du MCP23S17 est de 25 mA, ce qui est suffisant pour piloter une LED avec une résistance en série appropriée, mais pas assez pour piloter un relais directement.

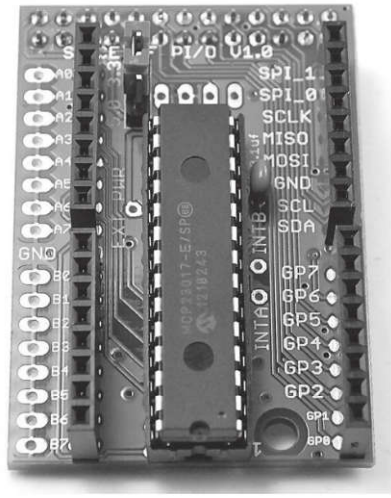


Figure 9.3
Slice of PI/O.

La carte occupe toutes les broches d'entrées/sorties jusqu'aux connecteurs du bord, et chacune des 16 broches d'entrées/sorties peut être configurée comme une entrée ou une sortie. Voici une liste des principales fonctionnalités :

- Seize connexions d'entrées/sorties bidirectionnelles amplifiées
- Fonctionne en 3,3 V ou 5 V (sélection par cavalier)
- Connexions série I2C et SPI du Raspberry Pi reportées (attention : elles ne sont pas bufferisées)
- Broches GPIO 0 à 7 du Raspberry Pi reportées (attention : elles ne sont pas bufferisées)

Pour l'instant, la carte n'est pas fournie avec un module Python, mais cette situation est susceptible de changer grâce aux efforts du fournisseur ou de la communauté Raspberry Pi.

RaspiRobotBoard

Je dois vous avouer mon intérêt personnel pour la carte RaspiRobotBoard qui est illustrée à la figure 9.4, dans la mesure où c'est moi qui l'ai conçue. L'objectif de cette carte est d'utiliser le Raspberry Pi comme contrôleur de robot. C'est la raison pour laquelle la carte est dotée d'un contrôleur de moteurs qui permet de contrôler la direction de deux moteurs (généralement reliés aux roues).

Copyright © 2014 Dunod.

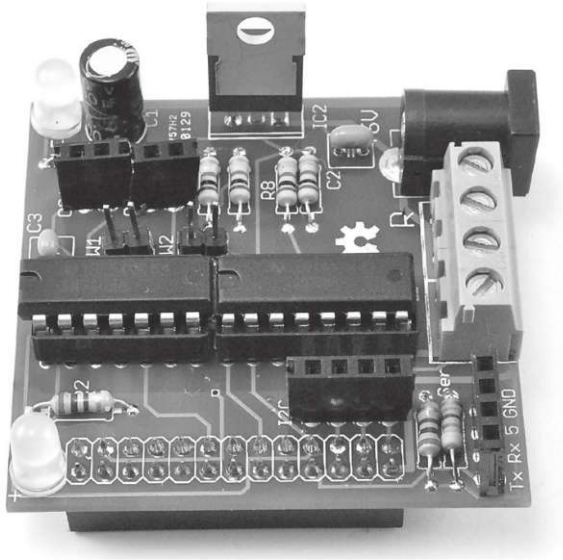


Figure 9.4
RaspiRobotBoard.

Cette carte possède également un régulateur de tension qui alimente le Raspberry Pi à l'aide de toute source d'énergie comprise entre 6 V et 9 V, comme quatre piles AA. Le RaspiRobotBoard dispose aussi de connecteurs pour deux types différents de port série, dont un est destiné à supporter une carte adaptateur de module de télémètre à ultrasons. La carte a également deux entrées dotées d'un interrupteur, deux LED et deux autres sorties bufferisées qui peuvent être utilisées pour piloter d'autres LED ou des charges de courant faible. Cette carte est utilisée au chapitre 11 pour construire un petit robot qui se déplace.

Gertboard

Le Gertboard est conçu par Gert van Loo de Broadcom et constitue donc la carte d'extension Raspberry Pi la plus officielle (voir la figure 9.5).

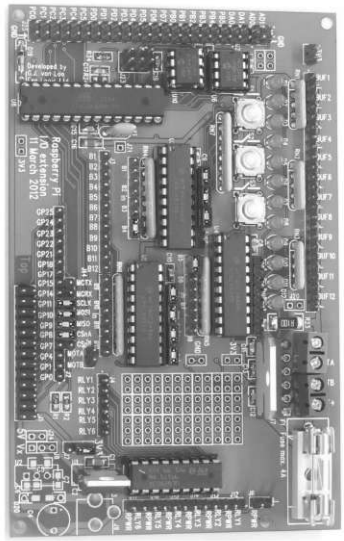


Figure 9.5
Carte d'extension Gertboard pour le Pi.

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

La Gertboard est vraiment le couteau suisse des cartes d'extension. Voici ses principales caractéristiques :

- Zone de connexions où les broches GPIO peuvent être connectées à différents modules
- Microcontrôleur ATmega (comme sur l'Arduino)
- Convertisseurs SPI analogique-numérique et numérique-analogique
- Contrôleur de moteur (comme le RaspiRobotBoard)
- Sorties de collecteur ouvert de 500 mA (comme sur le Pi Face)
- Douze LED et trois boutons-poussoirs

9.3 CARTES DE PROTOTYPAGE

Contrairement aux cartes d'extension, les cartes de prototypage nécessitent pour la plupart l'utilisation d'un fer à souder et de bonnes connaissances en électronique. Elles se connectent aussi directement à la puce principale du Raspberry Pi, ce qui signifie que si vous commettez une erreur, vous pouvez facilement endommager votre Raspberry Pi. Ces cartes sont faites pour les électroniciens amateurs expérimentés ou très prudents (ou pour ceux qui sont très téméraires et n'ont pas peur de griller leur Raspberry Pi).

Parmi ces cartes de prototypage, le Cobbler n'est pas réellement une carte, mais plutôt un connecteur qui permet de relier les connexions GPIO à une platine de test sans soudure sur laquelle vous pouvez ajouter vos propres appareils électroniques. Nous utiliserons cette carte dans le prochain chapitre.

Pi Cobbler

Le Pi Cobbler de chez Adafruit (*http://www.adafruit.com/products/914*) est livré en kit et nécessite d'être soudé. La soudure est assez simple, et une fois que tout est assemblé, vous aurez une carte avec 26 broches qui peut être reliée à une platine de test sans soudure (voir la figure 9.6). Sur le dessus de la carte, il y a un connecteur à 26 broches qui permet de relier, grâce à un câble (également fourni), le connecteur GPIO du Raspberry Pi au Pi Cobbler.

Copyright © 2014 Dunod.

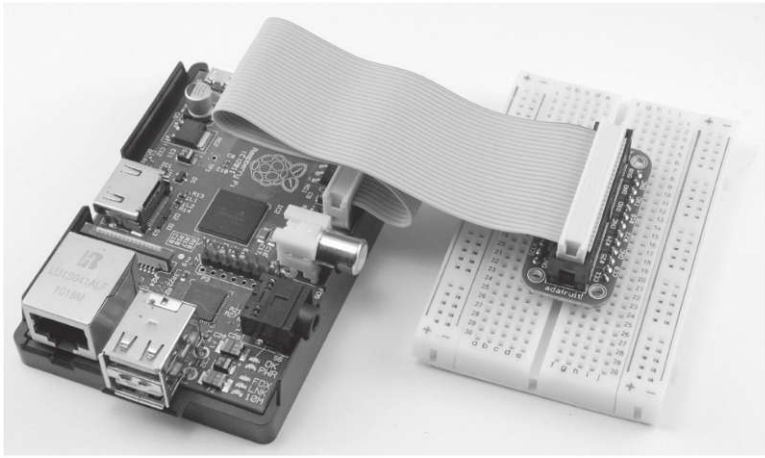


Figure 9.6
Pi Cobbler de chez Adafruit.

PI Plate

Le Pi Plate, illustré à la figure 9.7, est un autre produit de chez Adafruit (*https://www.adafruit.com/products/801*). Il s'agit d'une carte de prototypage qui dispose d'une grande surface au centre sur laquelle on peut souder des composants. Des bornes à vis sont situées tout autour du bord de la carte si bien que vous pouvez relier des câbles à des composants externes qui ne tiennent pas sur la carte, comme des moteurs. Dans un coin de la carte, il y a une zone où un circuit intégré peut être soudé. Les broches situées à côté déportent les broches du circuit qui sont difficiles à utiliser.

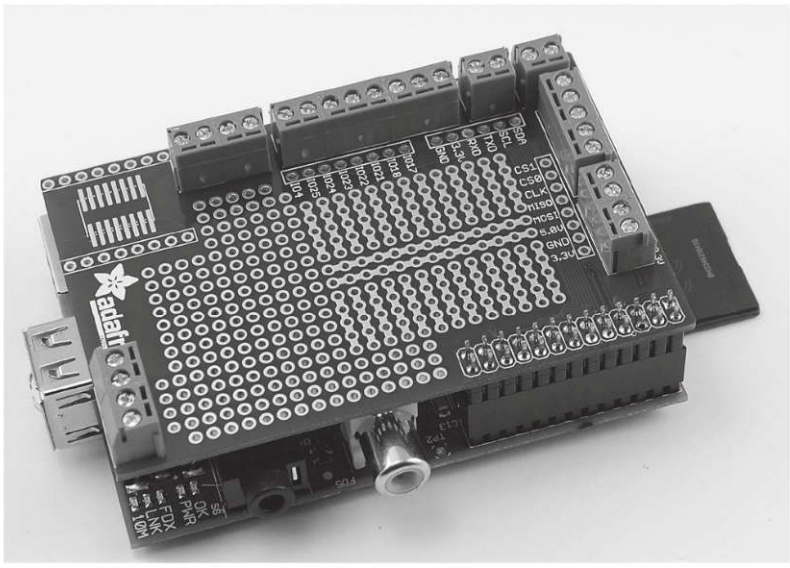


Figure 9.7
Pi Plate de chez Adafruit.

Humble Pi

Le Humble Pi, illustré à la figure 9.8, est assez similaire au Pi Plate, mais il lui manque la zone de prototypage de montage en surface. Toutefois, il comble cette lacune en fournissant un espace où vous pouvez ajouter votre propre régulateur de tension et une prise

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

d'alimentation, ce qui permet de fournir du 5 V au PI grâce à des piles ou un chargeur externe. Le régulateur de tension et les condensateurs associés ne sont pas fournis, mais Ciseco les commercialise dans un kit de composants.

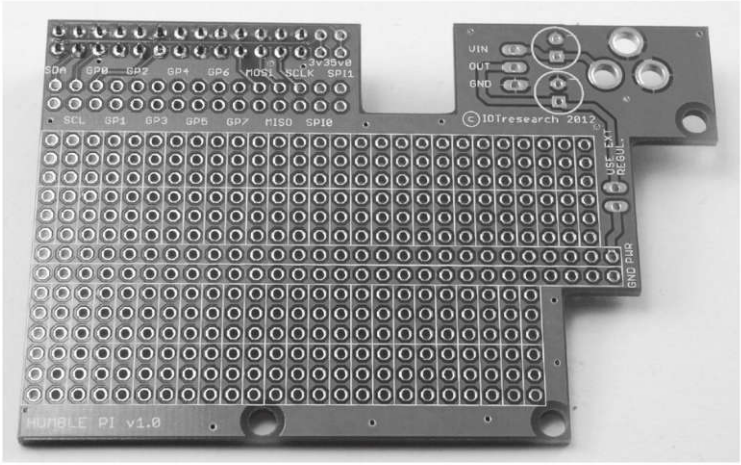


Figure 9.8
Humble Pi.

9.4 ARDUINO ET LE PI

Bien que le Raspberry Pi puisse être utilisé comme microcontrôleur pour piloter des moteurs, il n'a pas vraiment été conçu pour ça. En tant que telles, les broches GPIO ne peuvent pas fournir beaucoup de courant, sont un peu délicates et ne tolèrent pas les surcharges électriques. Il s'agit, après tout, du rôle dévolu aux cartes d'extension décrites dans la section précédente.

Les cartes Arduino, en revanche, sont beaucoup plus robustes et conçues pour être utilisées pour contrôler les appareils électroniques (voir la figure 9.9). Qui plus est, elles ont des entrées analogiques qui peuvent mesurer une tension, comme celle d'un capteur de température.

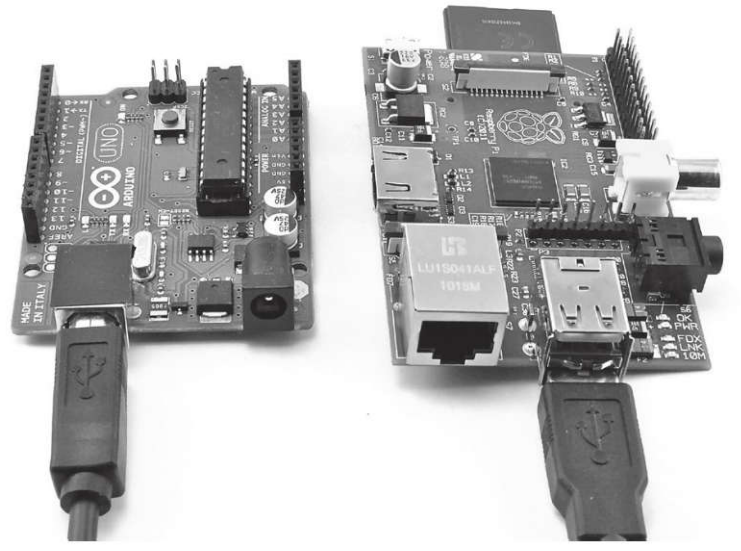


Figure 9.9
Arduino connecté à un Raspberry Pi.

Copyright © 2014 Dunod.

Les cartes Arduino sont conçues pour permettre la communication avec un ordinateur hôte via un port USB, et il n'y a aucune raison pour que cet hôte ne soit pas un Raspberry Pi. Cela signifie que l'Arduino prend en charge toute l'électronique et que le Raspberry Pi envoie des commandes ou écoute les requêtes entrantes de l'Arduino.

Si vous avez un Arduino, vous pouvez tester l'exemple simple suivant qui envoie des messages à l'Arduino pour faire clignoter sa LED intégrée, tout en recevant les messages entrants de l'Arduino. Une fois que vous savez faire cela, il est facile d'adapter le sketch Arduino ou le programme Python du Raspberry Pi pour s'acquitter de tâches plus complexes.

Cet exemple suppose que vous êtes familiarisé avec l'Arduino. Si vous ne l'êtes pas, vous pouvez lire les ouvrages publiés chez Dunod sur l'Arduino.

Communication entre l'Arduino et le Pi

Pour faire communiquer l'Arduino et le Pi, nous allons les relier à l'un des ports USB du Raspberry Pi. Comme l'Arduino ne consomme que 50 mA et qu'il n'a dans cet exemple aucun périphérique qui lui soit relié, il peut être alimenté par le Pi.

Logiciel Arduino

La seule chose que vous ayez à faire consiste à charger le sketch (nom des programmes dans l'environnement Arduino) Arduino suivant sur l'Arduino. Il est préférable de réaliser cela depuis votre ordinateur habituel, car il n'existe pour l'instant qu'une version ancienne du logiciel Arduino disponible pour le Raspberry Pi. Le sketch suivant, qui est disponible dans les compléments électroniques de cet ouvrage, se nomme PiTest.ino :

```
// Pi et Arduino

const int ledPin = 13;

void setup()
{
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  Serial.println(«Hello Pi»);
```

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

```
  if (Serial.available())
  {
    flash(Serial.read() - '0');
  }
  delay(1000);
}

void flash(int n)
{
  for (int i = 0; i < n; i++)
  {
   digitalWrite(ledPin, HIGH);
   delay(100);
   digitalWrite(ledPin, LOW);
   delay(100);
  }
}
```

Ce sketch très simple ne contient que trois fonctions. La fonction `setup` initialise la communication série et définit la broche 13 de la LED comme une sortie. Cette broche est reliée à la LED intégrée à l'Arduino. La fonction `loop` est appelée de façon continue jusqu'à ce que l'Arduino soit mis hors tension. Elle envoie d'abord le message « Hello Pi » au Raspberry Pi, puis vérifie s'il y a une communication entrante en provenance du Pi. Si tel est le cas (on attend un seul chiffre), le sketch fait clignoter la LED autant de fois que le chiffre reçu en utilisant la fonction `flash`.

Logiciel Raspberry Pi

Le code Python pour communiquer avec l'Arduino est encore plus simple et peut se contenter d'être saisi dans la console Python. Mais vous devez d'abord installer le paquet PySerial pour permettre la communication. On réalise cela de la même manière que pour les autres paquets que nous avons installés ; il faut aller récupérer le fichier tar compressé à l'adresse suivante :

http://sourceforge.net/projects/pyserial/files/latest/download?source=files

Ensuite, il faut extraire le répertoire de l'archive en saisissant la commande suivante :

```
tar -xzf pyserial-2.5.tar.gz
```

Vous avez à présent un dossier pour le module, et il faut vous positionner dans ce dossier à l'aide de la commande `cd`, puis exécuter la commande `install` (il est cependant préférable de vérifier les instructions pour voir s'il faut faire quelque chose d'autre

Copyright © 2014 Dunod.

au préalable). Vous êtes maintenant prêt à exécuter le programme d'installation du module :

```
cd pyserial-2.5
sudo python setup.py install
```

Après l'installation, vous serez en mesure d'importer le module à partir du shell Python. Basculez maintenant du terminal Linux à la console Python et saisissez la commande suivante :

```
import serial
ser = serial.Serial('/dev/ttyACM0', 9600)
```

Cela ouvre la connexion série USB avec l'Arduino à une vitesse de 9600 bauds. Vous devez à présent exécuter une boucle qui écoute les messages de l'Arduino :

```
while 1 :
   ser.readline()
```

Il faudra appuyer deux fois sur la touche Entrée après avoir saisi la deuxième ligne. Les messages devraient maintenant commencer à apparaître ! Vous pouvez constater quand l'Arduino communique avec le Pi car les messages sont écrits en bleu ; appuyez ensuite sur CTRL + C pour interrompre les messages en provenance de l'Arduino.

Maintenant, saisissez ce qui suit dans la console Python :

```
ser.write('5')
```

Cela devrait faire clignoter la LED 5 fois.

9.5 RÉSUMÉ

Dans ce chapitre, nous avons examiné les nombreuses façons d'ajouter de l'électronique à nos projets Raspberry Pi. Dans les deux chapitres suivants, nous allons réaliser des projets en utilisant deux approches différentes : tout d'abord nous emploierons le Cobbler de chez Adafruit ainsi qu'une platine de test, puis nous nous servirons du RaspiRobotBoard comme base d'un petit robot qui se déplace.

Copyright © 2014 Dunod.

Copyright © 2014 Dunod.

CHAPITRE PAGE

10 PROJET DE PROTOTYPAGE D'UNE HORLOGE

1	Introduction	3
2	Prise en main	17
3	Bases de Python	27
4	Chaînes, listes et dictionnaires	43
5	Modules, classes et méthodes	69
6	Fichiers et Internet	81
7	Interfaces utilisateur graphiques	91
8	Programmation de jeux	109
9	Interfaçage du matériel	123
10	Projet de prototypage d'une horloge	137

10.1 Matériel nécessaire **138**
10.2 Assemblage du matériel **139**
10.3 Logiciel **141**
10.4 Phase 2 **143**
10.5 Résumé **146**

11	Le RaspiRobot	147
12	Pour aller plus loin	161

Copyright © 2014 Dunod.

Dans ce chapitre, nous allons construire quelque chose qui ne peut être considéré que comme une horloge numérique à LED exagérément sophistiquée. Nous utiliserons un Raspberry Pi, le Cobbler de chez Adafruit, une platine de test et un afficheur à LED à quatre chiffres (voir la figure 10.1).

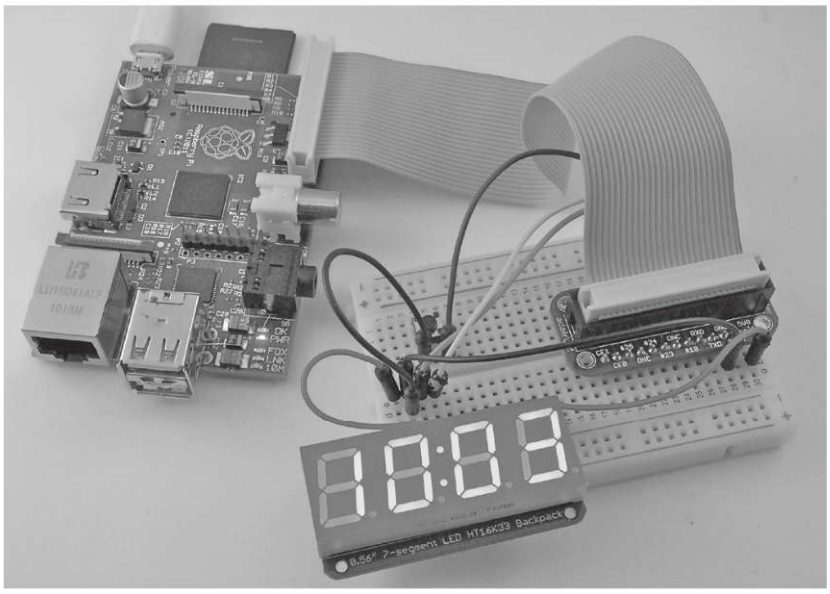


Figure 10.1
Horloge à LED utilisant le Raspberry Pi.

Dans la première phase de conception, le projet se contente d'afficher l'heure, mais dans une seconde phase, nous allons l'améliorer en ajoutant un bouton qui, lorsque l'on appuie dessus, bascule entre différents modes d'affichage (heures/minutes, secondes, et date).

10.1 MATÉRIEL NÉCESSAIRE

Pour créer ce projet, il vous faut les pièces suivantes. J'ai indiqué quelques fournisseurs possibles, mais vous pouvez également trouver ces articles ailleurs sur Internet.

Tableau 10.1

Article	Fournisseurs	Prix indicatif (en dollars)
Raspberry Pi	Farnell, RS Components	35 $
Pi Cobbler	Adafruit (Produit 914)	8 $
Afficheur à quatre chiffres I2C à sept segments de chez Adafruit	Adafruit (Produit 880)	10 $

Copyright © 2014 Dunod.

Platine de test sans soudure	Adafruit (Produit 64), SparkFun (SKU PRT-00112), Maplin (AG09K)	5 $
Assortiment de câbles de liaison (mâle-mâle)	Adafruit (Produit 758), SparkFun (SKU PRT-08431), Maplin (FS66W)	8 $
Bouton-poussoir*	Adafruit (Produit 367), SparkFun (SKU COM-00097), Maplin (KR92A)	2 $
** Optionnel. Nécessaire uniquement pour la deuxième phase.*		

10.2 ASSEMBLAGE DU MATÉRIEL

Le Pi Cobbler et les modules d'affichage de chez Adafruit sont livrés en kit et ils doivent être soudés avant de pouvoir être utilisés. Ils sont tous les deux assez faciles à souder, et on trouvera des instructions de montage détaillées sur le site web d'Adafruit. Chaque module possède des broches qu'il suffit d'enfoncer dans les trous de la platine de test.

L'afficheur comporte seulement quatre broches (VCC, GND, SDA et SCL) lorsqu'il est monté sur la platine de test ; alignez-le afin que la broche VCC soit sur la ligne 1 de la platine.

Le Cobbler a 26 broches, mais nous n'en utiliserons que quelques-unes. Il doit être inséré à l'autre bout de la platine, ou du moins assez loin pour qu'aucune des broches ne se chevauche avec les mêmes lignes que celles de l'afficheur. Le connecteur du Pi Cobbler a une découpe sur un côté pour s'assurer que la nappe ne peut être introduite que dans un seul sens. Cette découpe doit se trouver vers le haut de la platine, comme cela est illustré à la figure 10.2.

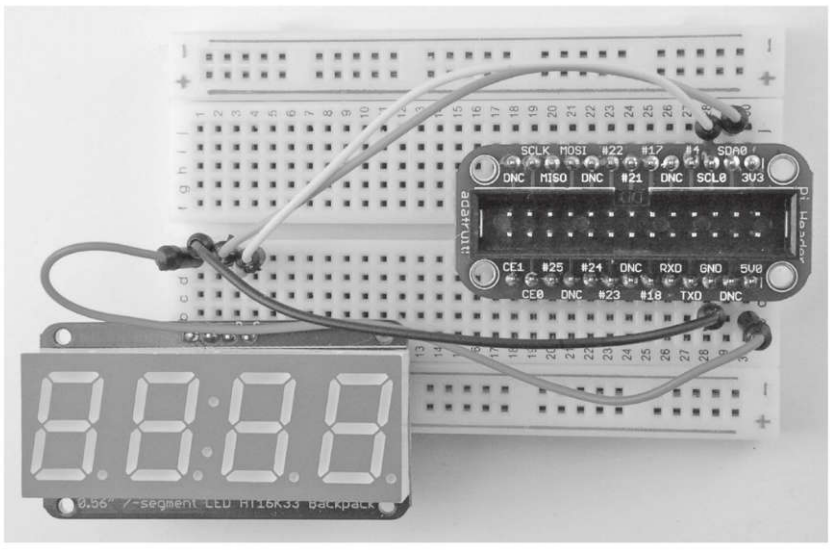


Figure 10.2
Disposition de la platine de test.

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2014 Dunod.

Sous les trous de la platine sans soudure, il y a des bandes de connecteurs, qui relient les cinq trous de chaque ligne. Vous noterez que c'est parce que la carte est sur le côté que les lignes sont en fait à la verticale sur la figure 10.2.

La figure 10.2 illustre la platine de test sans soudure avec les quatre broches de l'afficheur à une extrémité de la platine et le Pi Cobbler à l'autre extrémité. Lorsque vous suivrez les instructions de ce chapitre, cela facilitera les choses si vous montez vos modules de la même manière que cela est illustré à la figure 10.2.

Note : Il est plus facile de relier les câbles de raccordement sur la platine de test avant que la nappe du Cobbler ne soit mise en place.

Voici les connexions qu'il faut établir (tableau10.2).

Tableau 10.2

Suggestion de couleur du câble	De	Vers
Noir	Masse du Cobbler (GND)	Masse de l'afficheur (deuxième broche en partant de la gauche)
Rouge	Broche 5 Volts du Cobbler	Broche VCC de l'afficheur (extrême gauche)
Orange	Broche SDA0 du Cobbler	Broche SDA de l'afficheur (troisième broche en partant de la gauche)
Jaune	Broche SCL0 du Cobbler	Broche SCL de l'afficheur (extrême droite)

Les couleurs indiquées dans le tableau 10.2 ne sont qu'une suggestion, mais on a coutume d'utiliser le rouge pour le pôle positif et le noir ou le bleu pour la connexion à la masse.

Mise en garde : Dans ce projet, on connecte un module d'affichage de 5 V à un Raspberry Pi qui utilise généralement du 3,3 V. Nous ne pouvons le faire en toute sécurité que parce que le module d'affichage utilisé ici agit seulement en tant que périphérique « esclave » si bien qu'il n'écoute que sur les lignes SDA et SCL. D'autres périphériques I2C peuvent agir en tant que périphérique maître et s'ils sont en 5 V, il y a de grands risques d'endommager votre Pi. Par conséquent, avant de connecter n'importe quel périphérique I2C à votre Raspberry Pi, assurez-vous que vous comprenez ce que vous faites.

Copyright © 2014 Dunod.

Nous pouvons maintenant relier le Cobbler au Raspberry Pi en utilisant le câble fourni avec le Pi Cobbler. Le Raspberry Pi doit être hors tension quand vous réalisez cela. Le câble ne rentrera que dans un seul sens dans le Pi Cobbler, mais aucune protection de cette sorte n'est fournie sur le Raspberry Pi. Assurez-vous par conséquent que la ligne rouge du câble est à l'extérieur du Raspberry Pi, comme cela est illustré à la figure 10.1.

Allumez le Raspberry Pi. Si les voyants habituels ne s'allument pas, éteignez-le immédiatement et vérifiez tous les câbles.

10.3 LOGICIEL

Tout est connecté à présent et le Raspberry Pi a démarré. Toutefois, l'afficheur est toujours vide parce que nous n'avons pas encore écrit de code pour l'utiliser. Nous allons commencer par une horloge simple qui se contente d'afficher l'heure système du Raspberry Pi. Le Raspberry Pi n'a pas d'horloge temps réel pour lui indiquer l'heure, mais il ira la chercher automatiquement sur un serveur de temps s'il est connecté à Internet.

Le Raspberry Pi affiche l'heure dans le coin inférieur droit de l'écran. Si le Pi n'est pas connecté à Internet, vous pouvez régler l'heure manuellement à l'aide de la commande suivante :

```
sudo date -s «Jan 24 12:15»
```

Toutefois, vous devrez faire cela chaque fois que vous redémarrez. Il est par conséquent beaucoup mieux que votre Raspberry Pi soit connecté à Internet.

Si vous utilisez un serveur de temps Internet, il se peut que les minutes soient correctes, mais pas l'heure. Cela signifie probablement que votre Raspberry Pi ne sait pas dans quel fuseau horaire il se trouve. Ceci peut être corrigé en utilisant la commande suivante, qui ouvre une fenêtre où vous pouvez sélectionner votre continent et ensuite la ville du fuseau horaire dont vous avez besoin :

```
sudo dpkg-reconfigure tzdata
```

Au moment de la rédaction de ce livre, la distribution Raspbian Wheezy nécessitait que l'on exécute quelques commandes pour rendre le bus I2C accessible au programme Python que nous allons écrire, afin d'utiliser le bus I2C dont se sert l'afficheur. Il est probable que les versions ultérieures de la distribution Raspbian (ainsi que les autres distributions) régleront ce problème de telle sorte que les commandes suivantes ne seront plus nécessaires. Toutefois, voici ce que vous devez faire pour l'instant :

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

```
sudo apt-get install python-smbus
sudo modprobe i2c-dev
sudo modprobe i2c-bcm2708
```

Note : Il est possible que vous deviez exécuter les deux dernières commandes chaque fois que vous redémarrez le Raspberry Pi.

Maintenant que le Raspberry Pi sait l'heure exacte et que le bus I2C est disponible, nous pouvons écrire un programme Python qui envoie l'heure à l'afficheur. Afin de simplifier ce processus, j'ai écrit un module de bibliothèque Python spécifiquement pour ce type d'affichage. Vous pouvez le télécharger à l'adresse suivante :

http://code.google.com/p/i2c7segment/downloads/list

Comme pour les autres modules que vous avez installés, vous devez récupérer le fichier, le décompresser dans un emplacement qui convient (à l'aide de `tar -xzf`) puis lancer la commande suivante pour l'installer sous Python 2 :

```
sudo python setup.py install
```

Le véritable programme d'horloge, dont vous trouverez le listing ci-dessous, se nomme 10_01_horloge.py :

```
import i2c7segment as display
import time

disp = display.Adafruit7Segment()

while True:
    h = time.localtime().tm_hour
    m = time.localtime().tm_min
    disp.print_int(h * 100 + m)
    disp.draw_colon(True)
    disp.write_display()
    time.sleep(0.5)
    disp.draw_colon(False)
    disp.write_display()
    time.sleep(0.5)
```

Le programme est élégant et simple. La boucle s'exécute en permanence et calcule l'heure et les minutes et en les positionnant aux bons endroits sur l'afficheur ; les heures sont multipliées par 100 pour les décaler dans la partie gauche des chiffres et puis on ajoute les minutes qui seront ainsi affichées sur la droite.

La bibliothèque i2c7segment fait l'essentiel du travail à notre place. Cette bibliothèque est utilisée en définissant d'abord ce qui

Copyright © 2014 Dunod.

doit être affiché à l'aide de `print_int` ou de `draw_colon`, puis en employant `write_display` pour mettre à jour ce qui s'affiche.

On fait clignoter le caractère deux-points en l'allumant, puis en attendant une demi-seconde, puis en l'éteignant à nouveau. L'accès au port I2C est uniquement disponible pour les super-utilisateurs, de telle sorte que vous devez exécuter la commande en tant que super-utilisateur en saisissant le texte suivant :

```
sudo python 10_01_horloge.py
```

Si tout fonctionne bien, l'afficheur devrait indiquer l'heure.

10.4 PHASE 2

Après avoir fait fonctionner l'affichage de base, nous allons étendre le matériel et le logiciel de notre projet en ajoutant un bouton qui modifie le mode d'affichage (on affiche successivement l'heure avec les minutes, les secondes, puis la date). La figure 10.3 illustre la platine de test avec l'ajout du bouton ainsi que deux nouveaux fils de liaison. Vous noterez que rien n'a changé par rapport à la première phase de conception, si ce n'est l'ajout du bouton.

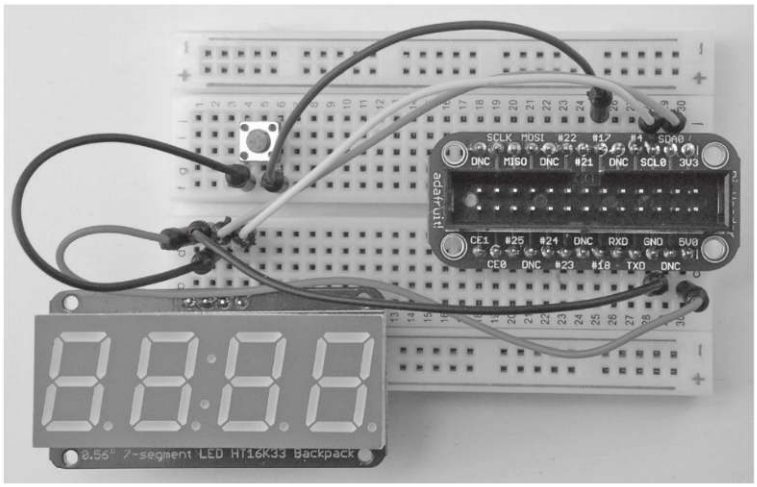


Figure 10.3
Ajout d'un bouton à la conception.

Note : Arrêtez et éteignez votre Pi avant de commencer à faire les modifications sur la platine de test.

Le bouton présente quatre conducteurs et doit être placé dans la bonne position ; sinon, le commutateur apparaîtra comme étant tout le temps fermé. Les câbles doivent sortir des côtés vers le haut et le bas de la figure 10.3. Ne vous inquiétez pas si vous avez positionné l'interrupteur dans le mauvais sens ; rien ne sera endommagé, mais l'affichage changera continuellement de mode sans que l'on n'appuie sur le bouton.

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Deux nouveaux câbles sont nécessaires pour connecter le commutateur. L'un part d'un fil de l'interrupteur (voir la figure 10.3) vers la masse de l'afficheur. L'autre câble va sur la connexion numérotée 17 du Pi Cobbler. L'effet est que chaque fois que vous appuyez sur le bouton du commutateur, la broche GPIO 17 du Raspberry Pi est reliée à la masse.

Voici le logiciel mis à jour que vous trouverez dans le fichier 10_02_horloge2.py :

```
import i2c7segment as display
import time
import RPi.GPIO as io

switch_pin = 17
io.setmode(io.BCM)
io.setup(switch_pin, io.IN, pull_up_down=io.PUD_UP)
disp = display.Adafruit7Segment()

time_mode, seconds_mode, date_mode = range(3)
disp_mode = time_mode

def display_time():
    h = time.localtime().tm_hour
    m = time.localtime().tm_min
    disp.print_int(h * 100 + m)
    disp.draw_colon(True)
    disp.write_display()
    time.sleep(0.5)
    disp.draw_colon(False)
    disp.write_display()
    time.sleep(0.5)

def display_date():
    d = time.localtime().tm_mday
    m = time.localtime().tm_mon
    disp.print_int(d * 100 + m) # format mondial
    #disp.print_int(m * 100 + d)  # format américain
    disp.draw_colon(True)
    disp.write_display()
    time.sleep(0.5)

def display_seconds():
    s = time.localtime().tm_sec
    disp.print_str('----')
    disp.print_int(s)
    disp.draw_colon(True)
    disp.write_display()
    time.sleep(0.5)
```

Copyright © 2014 Dunod.

```
while True:
    key_pressed = not io.input(switch_pin)
    if key_pressed:
     disp_mode = disp_mode + 1
     if disp_mode > date_mode:
        disp_mode = time_mode
    if disp_mode == time_mode:
     display_time()
    elif disp_mode == seconds_mode:
     display_seconds()
    elif disp_mode == date_mode:
     display_date()
```

Comme nous avons besoin d'accéder à la broche GPIO 17 pour voir si le bouton est enfoncé, il faut utiliser la bibliothèque `RPi.GPIO`. Au chapitre 5, nous avons vu comment installer un module et c'est cette bibliothèque qui nous a servi d'exemple. Si vous n'avez pas encore installé `RPi.GPIO`, reportez-vous au chapitre 5 et faites-le maintenant.

Nous avons défini la broche du commutateur comme une entrée en utilisant la commande suivante :

```
io.setup(switch_pin, io.IN, pull_up_down=io.PUD_UP)
```

Cette commande active également une résistance pull-up interne qui garantit que l'entrée est toujours à 3,3 V (valeur HIGH) sauf si l'on appuie sur l'interrupteur (la valeur passe alors à LOW).

La plupart du code qui se trouvait dans la boucle a été subdivisé en une fonction appelée `display_time`. En outre, deux nouvelles fonctions ont été ajoutées : `display_seconds` et `display_date`, qui, comme leur nom l'indique, affichent respectivement les secondes et la date.

Il est intéressant de noter que display_date affiche la date au format mondial. Si vous voulez changer cela et afficher la date au format américain, modifiez la ligne qui commence par `disp.print_int` de façon appropriée (voir les commentaires dans le code). Pour garder une trace du mode dans lequel nous sommes, nous avons ajouté de nouvelles variables dans les lignes suivantes :

```
time_mode, seconds_mode, date_mode = range(3)
disp_mode = time_mode
```

La première ligne attribue à chacune des trois variables un nombre différent. La deuxième ligne définit la variable `disp_mode` avec la valeur de `time_mode`, que nous utiliserons plus tard dans la boucle principale.

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

La boucle principale a été modifiée afin de déterminer si le bouton est enfoncé. Si c'est le cas, la valeur 1 est ajoutée à `disp_mode` pour changer le mode d'affichage. Si le mode d'affichage a atteint la fin, il est réinitialisé à `time_mode`.

Pour finir, les blocs `if` qui suivent sélectionnent la fonction d'affichage appropriée, en fonction du mode, puis appellent les fonctions.

10.5 RÉSUMÉ

Le matériel de ce projet peut être facilement adapté à d'autres usages. Vous pourriez, par exemple, présenter toutes sortes de choses sur l'afficheur en modifiant le programme. Voici quelques idées :

- Votre vitesse de connexion à Internet
- Le nombre de courriels dans votre boîte de réception
- Un compte à rebours des jours restant dans l'année
- Le nombre de visiteurs d'un site web

Dans le chapitre suivant, nous allons réaliser un autre projet : un robot qui se déplace dont le Raspberry Pi sera le cerveau.

Copyright © 2014 Dunod.

CHAPITRE PAGE

11 LE RASPIROBOT

1	Introduction	3
2	Prise en main	17
3	Bases de Python	27
4	Chaînes, listes et dictionnaires	43
5	Modules, classes et méthodes	69
6	Fichiers et Internet	81
7	Interfaces utilisateur graphiques	91
8	Programmation de jeux	109
9	Interfaçage du matériel	123
10	Projet de prototypage d'une horloge	137
11	Le RaspiRobot	147

11.1 Matériel nécessaire **148**
11.2 Phase 1 : création d'un robot basique **149**
11.3 Phase 2 : ajout d'un télémètre et d'un écran **155**
11.4 Résumé **160**

12	Pour aller plus loin	161

Copyright © 2014 Dunod.

Dans ce chapitre, vous allez apprendre à programmer le Raspberry Pi pour lui faire jouer le rôle de cerveau d'un robot d'exploration (en anglais *rover*) simplifié, qui est illustré à la figure 11.1. Le Pi recevra les commandes à partir d'un clavier sans fil et contrôlera la puissance des moteurs reliés à un kit de châssis de robot. Le robot sera également doté (en option) d'un télémètre à ultrasons qui lui indique à quelle distance se trouvent les obstacles et d'un écran LCD qui affiche les informations du télémètre.

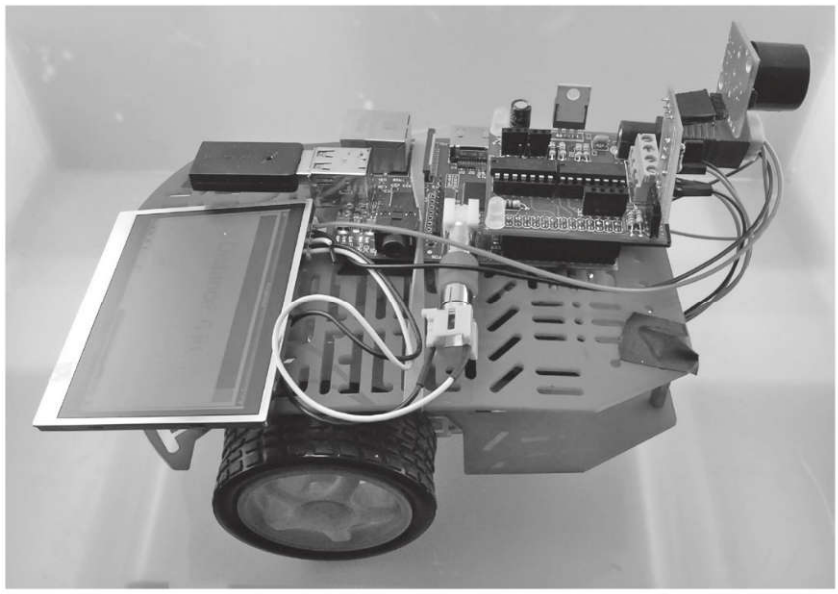


Figure 11.1
RaspiRobot.

Comme le projet du chapitre précédent, ce projet est divisé en deux phases. Dans la première phase, on crée un robot de base que vous pouvez piloter avec un clavier sans fil ; dans la deuxième phase, on ajoute l'écran et le télémètre.

Mise en garde : Si les piles sont reliées à la RaspiRobotBoard, elles alimenteront le Raspberry Pi. Attention, n'alimentez jamais votre Raspberry Pi avec son bloc d'alimentation et la RaspiRobotBoard en même temps ! Vous pouvez laisser la RaspiRobotBoard connectée à votre Raspberry Pi, mais ne fixez pas sur le Raspberry Pi les moteurs ou les batteries.

11.1 MATÉRIEL NÉCESSAIRE

Pour réaliser ce projet, vous aurez besoin des articles suivants. J'ai indiqué quelques fournisseurs possibles, mais vous pouvez également trouver ces articles ailleurs sur Internet.

Copyright © 2014 Dunod.

Tableau 11.1

Article	Fournisseurs	Prix indicatif (en dollars)
Raspberry Pi	Farnell, RS Components, CPC, Newark	35 $
RaspiRobotBoard	SparkFun	30 $
Adaptateur série pour le télémètre*	www.raspirobot.com	5 $
Télémètre Maxbotix LV-EZ1*	SparkFun (SEN-00639), Adafruit (Product 172)	25 $
Écran LCD 3,5 pouces*	Adafruit (Product 913)	45 $
Adaptateur mâle-mâle RCA pour l'écran*	Adafruit (Product 951)	2 $
Magician Chassis	SparkFun (ROB-10825)	15 $
Fiche cc 5,5 x 2,1 mm mâle vers connexion à vis **	Adafruit (Product 369), SparkFun (PRT-10288)	2 $
Coupleur six piles AA	Adafruit (Product 248)**, Newark (63J6606), Maplins (HQ01B)	5 $
Contact à pression pile 9 V**	RadioShack (270-324), Maplins (NE19V)	2 $
Six piles AA (rechargeables ou alcalines)		
Clavier USB sans fil	Magasin d'informatique	10 $
* *Uniquement pour la phase 2.*		
**C'est un boîtier de piles différent de celui que j'ai utilisé. Il se termine par une fiche de 2,1 mm, si bien que le contact à pression n'est pas nécessaire si vous utilisez ce boîtier. Si vous vous contentez de la première phase du projet et que vous utilisez le boîtier Adafruit, vous n'avez pas besoin de la fiche cc 5,5 * 2,1, ni du contact à pression.		

11.2 PHASE 1 : CRÉATION D'UN ROBOT BASIQUE

La figure 11.2 illustre le robot de base. La base de ce robot d'exploration est le kit qui se nomme Magician Chassis. Ce kit se compose d'un châssis en plastique, de moteurs à engrenages, de roues et de toutes les pièces nécessaires pour assembler le châssis. Il comprend également un boîtier où peuvent se loger quatre piles AA, mais dans ce projet, ce boîtier sera remplacé par un autre qui accepte six piles AA.

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

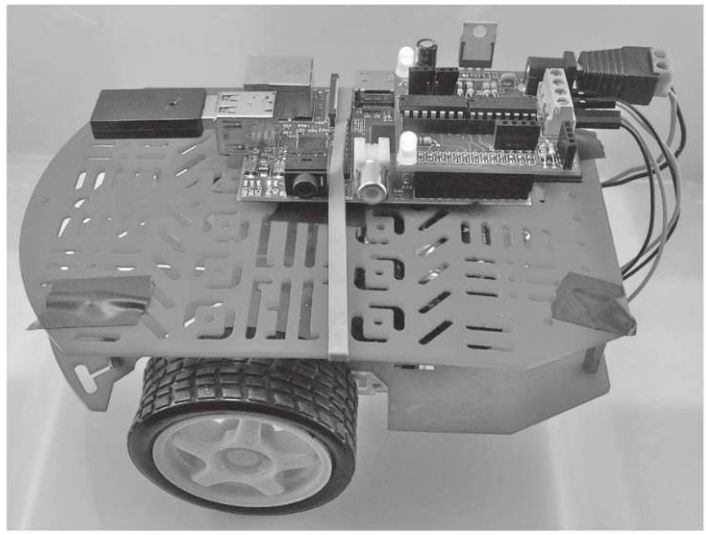


Figure 11.2
Rover de base.

Assemblage du matériel

Ce projet est assemblé à partir de kits de pièces différents. Si vous recherchez sur Internet, vous pourrez trouver des kits déjà assemblés quand vous achetez la RaspiRobotBoard et le télémètre, ce qui signifie que l'ensemble du projet peut être réalisé sans aucune soudure (l'outil le plus difficile à utiliser ne sera donc qu'un tournevis).

Étape 1 : assemblage du châssis

Le Magician Chassis est livré en pièces détachées qui doivent être assemblées. Des instructions de montage détaillées sont incluses dans le kit. Lors de l'assemblage du châssis, vous devrez remplacer le boîtier de piles fourni (quatre piles AA) par un boîtier permettant d'accueillir six piles AA (voir la figure 11.3). Si votre boîtier est du genre à contenir deux rangées de trois piles, vous constaterez qu'il peut tenir sur la plaque inférieure du châssis.

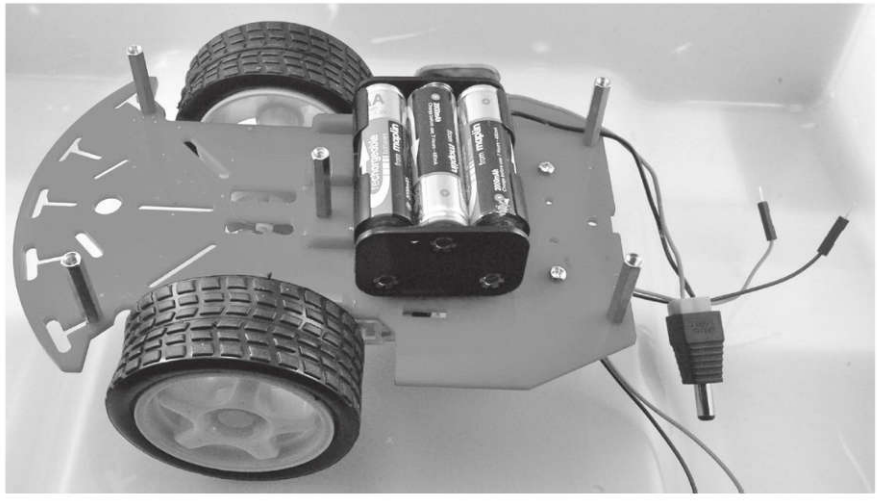


Figure 11.3
Remplacement du boîtier de piles.

Copyright © 2014 Dunod.

Si votre boîtier contient toutes les piles en une seule rangée, vous devrez probablement utiliser les vis fournies avec le Magician Chassis pour son boîtier de piles pour fixer votre propre boîtier solidement sur le châssis.

Fixez le contact à pression au boîtier de piles et les fils du contact dans le bornier de l'adaptateur. Veillez à bien respecter la bonne polarité (le rouge sur le plus) !

Avant de fixer la plaque supérieure du châssis, placez un élastique autour de la surface supérieure. Cela servira à maintenir le Raspberry Pi en place (voir la figure 11.2).

Étape 2 : assemblage de la RaspiRobotBoard

Vous pouvez vous procurer la RaspiRobotBoard en kit auprès de Sparkfun (*https://www.sparkfun.com/products/11561*). Les instructions de montage du kit sont disponibles à :

https://github.com/simonmonk/raspirobotboard/wiki

dans la rubrique Construction. Une fois assemblée, la carte devrait ressembler à celle qui est illustrée à la figure 11.4.

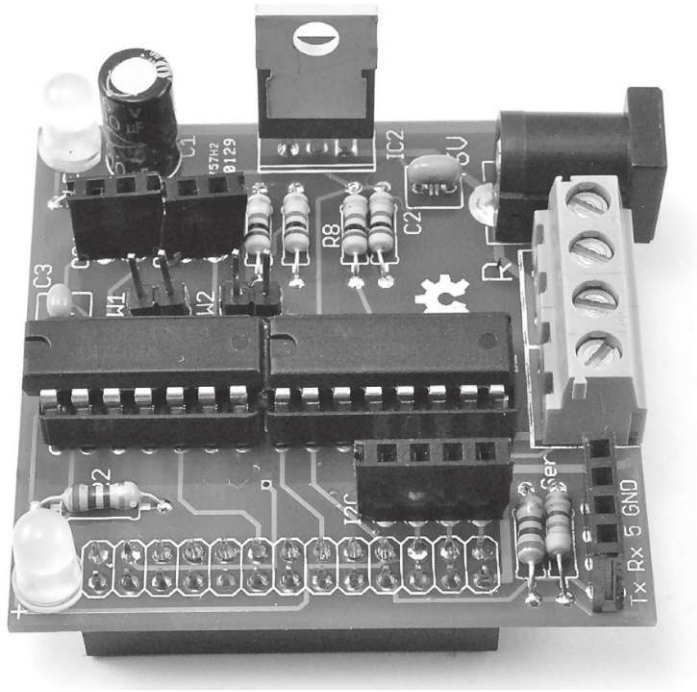


Figure 11.4
RaspiRobotBoard.

Toutes les connexions qui nous intéressent se trouvent sur le côté droit de la figure 11.4. En haut, il y a la prise de courant, et dessous les bornes à vis pour les moteurs gauche et droit.

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Étape 3 : installation du logiciel sur le Raspberry Pi

Pour contrôler le robot, nous allons écrire un programme en Python qui détecte quand on appuie sur des touches et se sert de ces informations pour l'alimentation des moteurs du robot. Pour ce faire, nous utiliserons pygame qui permet de savoir facilement si des touches sont enfoncées.

Il est beaucoup plus simple de configurer le programme avant de relier le Raspberry Pi au châssis. Dans ces conditions, reliez la RaspiRobotBoard au Raspberry Pi, mais débranchez les moteurs et les piles, puis démarrez le Pi à partir de son bloc d'alimentation USB normal.

La RaspiRobotBoard possède sa propre bibliothèque Python, mais elle utilise également d'autres bibliothèques qui doivent être installées. Tout d'abord, elle nécessite la bibliothèque `RPi.GPIO` que nous avons employée au chapitre 5, puis au chapitre 10. Si vous ne l'avez pas déjà fait, installez la bibliothèque `RPi.GPIO`. Vous devrez aussi installer la bibliothèque `PySerial`. Pour ce faire, reportez-vous aux instructions de la section sur Arduino vers la fin du chapitre 9.

La bibliothèque de la RaspiRobotBoard peut être installée à partir du site web suivant :

http://code.google.com/p/raspirobotboard/downloads/list

L'installation est la même que pour n'importe quel autre paquet Python. Comme nous utilisons Python 2 dans ce projet, la bibliothèque doit être installée à l'aide de la commande suivante :

```
tar -xzf raspirobotboard-1.0.tar.gz
cd raspirobotboard-1.0
sudo python setup.py install
```

Le programme Python de cette version du robot se trouve dans le fichier 11_01_rover_basic.py, qui doit être exécuté en tant que super-utilisateur. À titre de test (les moteurs sont toujours déconnectés), exécutez le programme en vous positionnant dans le répertoire du programme et en saisissant la commande suivante dans le terminal :

```
sudo python 11_01_rover_basic.py
```

Une fenêtre vide pygame doit apparaître et les deux LED doivent s'éteindre. Nous pouvons tester le programme sans les moteurs car le programme contrôle les moteurs, mais également les LED. Si vous appuyez sur la flèche vers le haut, les deux LED doivent se rallumer. Appuyez sur la barre d'espace pour les éteindre à nou-

Copyright © 2014 Dunod.

veau. Puis essayez les touches fléchées gauche et droite ; chaque flèche allume une LED.

Nous n'allons pas pouvoir relier notre robot à un écran et une souris, si bien que nous devons prendre des dispositions pour que ce programme démarre automatiquement lorsque notre Raspberry Pi a fini son processus de démarrage. Pour ce faire, nous devons placer le fichier `raspirobot_basic.desktop` (inclus dans le répertoire du code) dans un répertoire nommé `/home/pi/.config/autostart`. Vous pouvez faire tout cela avec le gestionnaire de fichiers. Il suffit de saisir `/home/pi/.config` dans la barre d'adresse en haut de l'écran. Notez que les répertoires qui commencent par un point sont cachés, et vous ne pouvez donc pas vous positionner dedans à partir du gestionnaire de fichiers simplement à l'aide de la souris.

S'il n'y a aucun répertoire appelé `autostart` dans le dossier `.config`, il faut en créer un et copier dedans le fichier `raspirobot_basic.desktop`. Nous pouvons nous assurer que notre programme de démarrage automatique fonctionne bien en redémarrant le Pi. Si tout va bien, la fenêtre de pygame apparaîtra automatiquement.

Nous reviendrons plus tard sur l'étude du code de ce projet, mais pour l'instant, nous voulons voir si tout fonctionne.

Étape 4 : connexion des moteurs

Éteignez et débranchez le Raspberry Pi de son alimentation. N'oubliez pas de la ranger, ce qui vous évitera de la brancher accidentellement en même temps que les piles. Mettez les piles dans le compartiment et fixez la plaque supérieure du châssis. Couvrez les vis métalliques avec un peu de chatterton ou un scotch isolant pour empêcher tout court-circuit accidentel avec le Raspberry Pi, puis glissez le Pi sous l'élastique. Reliez ensuite les fils des moteurs au bornier, comme cela est illustré à la figure 11.5.

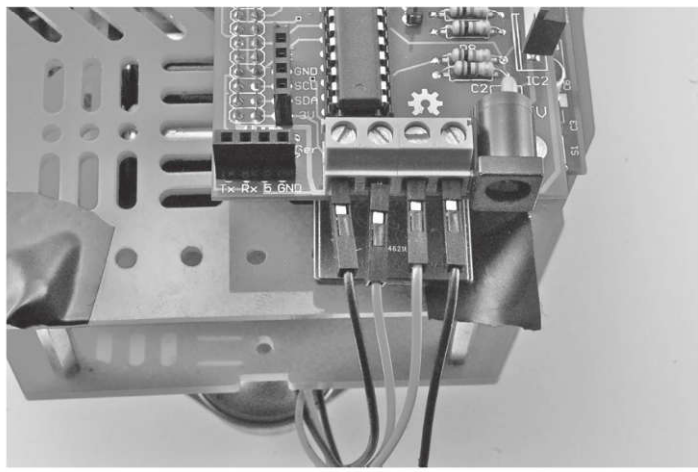


Figure 11.5
Connexion des moteurs.

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Chaque moteur a un fil rouge et un fil noir. Trouvez les fils du moteur gauche et fixez le fil noir à la borne la plus à gauche de la figure 11.5. Connectez le fil rouge du même moteur à la deuxième borne en partant de la gauche du bornier. Pour l'autre moteur, mettez le fil rouge dans la troisième borne en partant de la gauche et le fil noir dans la borne restante.

Étape 5 : test

C'est terminé et nous sommes prêts à tester le robot ! Connectez le dongle USB du clavier sans fil au Pi et fixez ensuite la fiche d'alimentation de la batterie dans la prise située sur la RaspiRobotBoard. Les LED du Raspberry Pi devraient scintiller quand il commence à démarrer. Si ce n'est pas le cas, débranchez immédiatement la batterie et vérifiez vos branchements.

Au départ, les LED du RaspiRobotBoard devraient toutes les deux être allumées, mais lorsque le programme Python s'exécute, elles devraient s'éteindre. Attendez deux secondes pour que le programme démarre correctement et appuyez ensuite sur les touches fléchées et la barre d'espacement du clavier. Votre RaspiRobot devrait commencer à se déplacer !

Logiciel

Le logiciel de la première phase est listé ci-dessous :

```
from raspirobotboard import *
import pygame
import sys
from pygame.locals import *

rr = RaspiRobot()

pygame.init()
screen = pygame.display.set_mode((640, 480))

pygame.display.set_caption('RaspiRobot')
pygame.mouse.set_visible(0)


while True:
    for event in pygame.event.get():
     if event.type == QUIT:
        sys.exit()
     if event.type == KEYDOWN:
        if event.key == K_UP:
         rr.forward()
         rr.set_led1(True)
```

Copyright © 2014 Dunod.

```
        rr.set_led2(True)
    elif event.key == K_DOWN:
        rr.set_led1(True)
        rr.set_led2(True)
        rr.reverse()
    elif event.key == K_RIGHT:
        rr.set_led1(False)
        rr.set_led2(True)
        rr.right()
    elif event.key == K_LEFT:
        rr.set_led1(True)
        rr.set_led2(False)
        rr.left()
    elif event.key == K_SPACE:
        rr.stop()
        rr.set_led1(False)
        rr.set_led2(False)
```

Note : Si vous avez sauté le chapitre 8 sur pygame, c'est peut-être le moment d'y consacrer un peu de temps.

Le programme démarre en important les modules des bibliothèques dont il a besoin. Ensuite, il crée une instance de la classe `RaspiRobot` et l'assigne à la variable `rr`. La boucle principale commence par vérifier la présence d'un événement `QUIT` et si tel est le cas, le programme se termine. Le reste de la boucle vérifie toutes les autres touches et exécute les commandes appropriées si une touche est enfoncée. Par exemple, si vous appuyez sur la touche flèche vers le haut, le RaspiRobot envoie la commande `forward` qui dit aux deux moteurs d'avancer et d'allumer les LED.

11.3 PHASE 2 : AJOUT D'UN TÉLÉMÈTRE ET D'UN ÉCRAN

Lorsque vous aurez terminé la phase 2, votre RaspiRobot ressemblera à celui qui est illustré à la figure 11.1. Déconnectez les piles de la RaspiRobotBoard afin de réaliser les changements nécessaires.

Étape 1 : assemblage de l'adaptateur série du télémètre

Le module série du télémètre, illustré à la figure 11.6, génère un signal inversé si bien qu'une petite carte avec un transistor et une résistance doit être utilisée pour inverser le signal de retour. Des instructions complètes pour le montage de ce petit adaptateur se trouvent à l'adresse suivante :

https://github.com/simonmonk/raspirobotboard/wiki/Rangefinder-Adapter

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

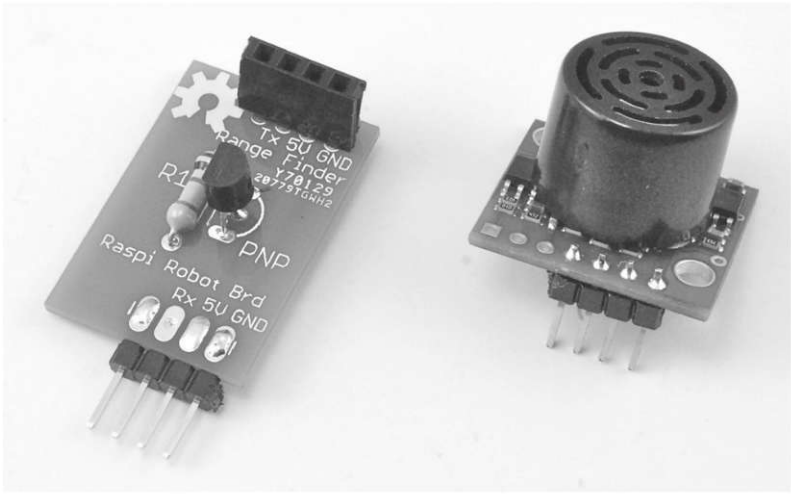


Figure 11.6
Adaptateur série du télémètre et module de télémètre.

Le module du télémètre se branche sur le dessus de l'adaptateur, et la partie inférieure de l'adaptateur s'insère dans la prise série (voir la figure 11.7).

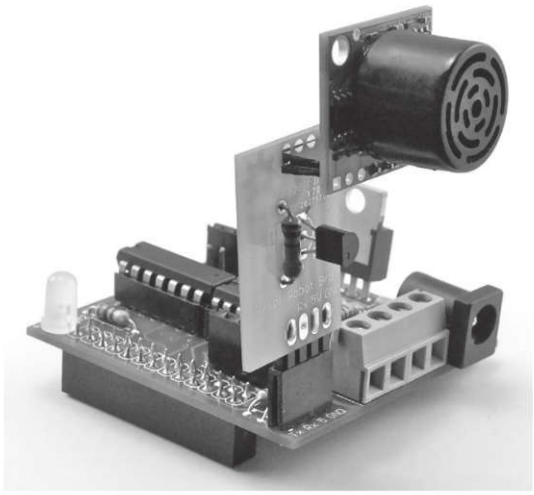


Figure 11.7
Assemblage du télémètre et de son adaptateur.

Étape 2 : connexion de l'écran

L'écran LCD est livré en deux parties : l'écran lui-même et la carte du pilote. Ces deux parties sont reliées par une nappe assez délicate. J'ai utilisé un autocollant matelassé pour solidariser l'ensemble. Soyez prudent lorsque vous fixez l'écran et manipulez-le délicatement.

L'écran est livré avec des câbles d'alimentation (rouge et noir) ainsi que deux prises RCA. Pour que l'installation soit plus propre, j'ai coupé une des fiches RCA (celle qui est reliée aux câbles du milieu dans la prise du connecteur blanc ; voir la figure 11.8). Si cela vous semble trop radical, vous pouvez aussi fixer les fils avec un serre-câble afin qu'ils n'encombrent pas.

Copyright © 2014 Dunod.

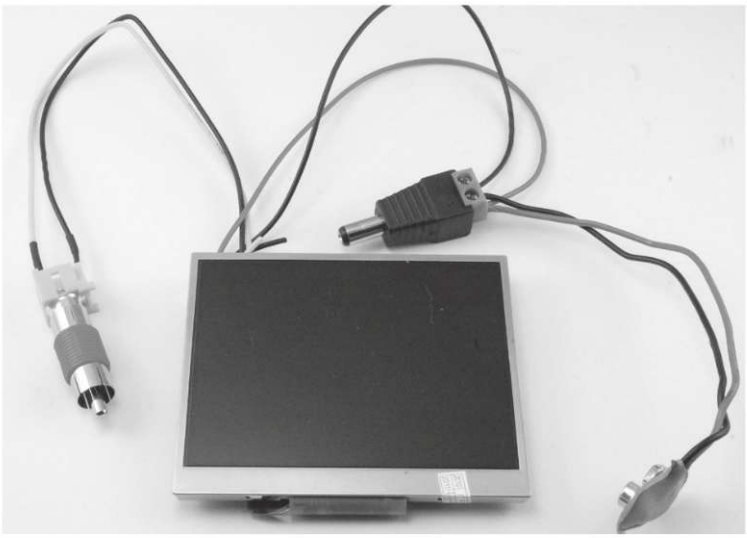


Figure 11.8
Câblage de l'afficheur.

Pour la fiche RCA restante, j'y ai connecté l'adaptateur RCA mâle-mâle. Les fils d'alimentation sont ensuite tressés sur les câbles d'alimentation de la même couleur du boîtier de piles, puis insérés dans les bornes à vis de l'adaptateur. Si votre boîtier de piles se termine par une fiche, vous pouvez couper la fiche et dénuder les fils. Ces fils peuvent alors servir comme s'il s'agissait des fils du boîtier de piles. Quoi qu'il en soit, le câblage du projet est résumé à la figure 11.9.

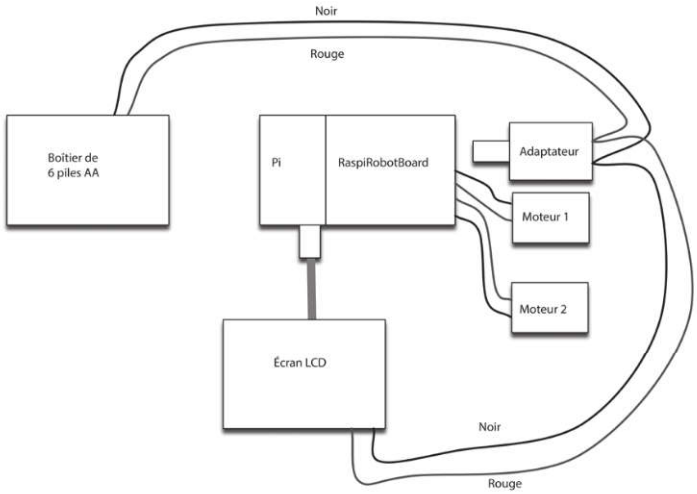


Figure 11.9
Schéma de câblage.

Ce montage a pour conséquence que l'écran sera toujours connecté à la batterie, même si vous débranchez le courant de la RaspiRobotBoard. C'est pour cette raison que vous devez utiliser le boîtier de piles pour allumer et éteindre le robot.

Note : Les lecteurs plus intrépides se paieront le luxe d'ajouter un interrupteur marche/arrêt.

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

L'écran est fixé au châssis au moyen de mastic adhésif. Ce n'est pas une solution pérenne et une sorte de crochet en plastique serait préférable.

Étape 3 : mise à jour du logiciel

La modification du matériel nécessite une mise à jour du logiciel. Le programme mis à jour se trouve dans le fichier 11_02_rover_plus.py. Vous devrez également modifier le programme qui se lance automatiquement au démarrage. Pour ce faire, copiez le fichier `raspirobot_plus.desktop` dans le répertoire `/home/pi/.config/autostart` et supprimez le fichier `raspirobot_basic.desktop` de ce dossier ; si vous ne le faites pas, les deux programmes démarreront automatiquement.

Vous noterez que comme dans cette phase du projet, le Raspberry Pi dispose d'un écran et d'un clavier (certes très petits), il est possible de faire les changements décrits ici avec le nouvel écran LCD installé, mais vous devrez alors utiliser un écran minuscule. Si cela s'avère trop difficile, débranchez la batterie et déconnectez les moteurs, puis démarrez le Raspberry Pi à partir de son bloc d'alimentation USB avec un moniteur normal, un vrai clavier et une souris.

Étape 4 : test final

Ça y est ! Le projet est prêt à fonctionner. Comme toujours, si les LED du Raspberry Pi ne s'allument pas tout de suite, débranchez les piles et cherchez le problème. Le Raspberry Pi consomme pas mal d'énergie pour un appareil alimenté par des piles. L'écran utilise aussi pas mal de courant si bien que pour éviter de trop recharger les batteries, vous devez débrancher ces appareils quand vous ne vous en servez pas.

Logiciel révisé

Le nouveau programme est plus grand que l'ancien, et il n'est pas reproduit ici dans son intégralité. Vous pouvez l'ouvrir dans IDLE pour y jeter un coup d'œil. Les principales différences sont, comme vous vous en doutez, la détection de la distance et l'affichage. La fonction `get_range` est listée ci-dessous :

```
def get_range():
    try:
     dist = rr.get_range_inch()
    except:
     dist = 0
    return dist
```

Copyright © 2014 Dunod.

Cette fonction encapsule un appel à `get_range_inch` qui se trouve dans le module RaspiRobot. On a ajouté une gestion des exceptions, car si le télémètre ne fonctionne pas, pour une quelconque raison (par exemple, s'il n'est pas branché), des exceptions seront déclenchées. Cette fonction intercepte toutes les exceptions et retourne une distance de 0 en cas d'exception.

La fonction `update_distance` récupère d'abord la distance puis l'affiche en même temps qu'une indication graphique de la proximité des obstacles, comme cela est illustré à la figure 11.10.

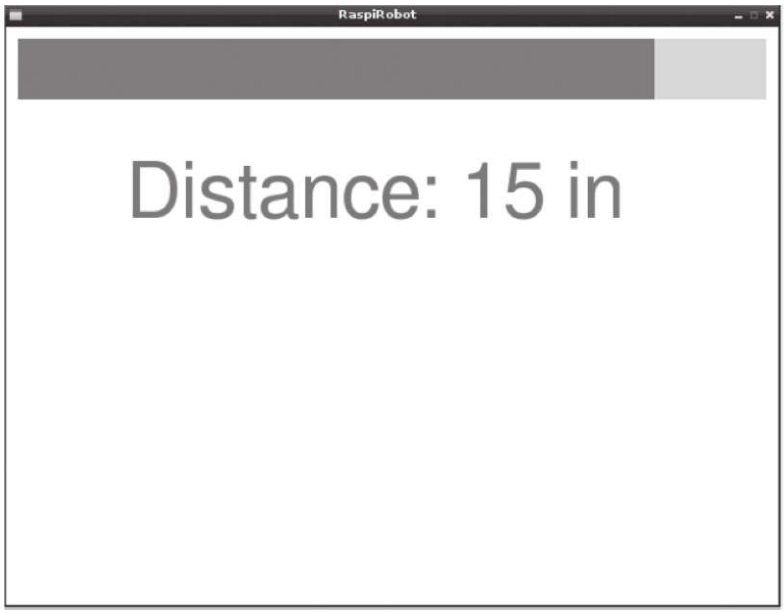


Figure 11.10
Affichage du RaspiRobot.

Voici le code de cette fonction :

```
def update_distance():
    dist = get_range()
    if dist == 0:
     return
    message = 'Distance: ' + str(dist) + ' pouces'
    text_surface = font.render(message, True, (127, 127, 127))
    screen.fill((255, 255, 255))
    screen.blit(text_surface, (100, 100))

    w = screen.get_width() - 20
    proximity = ((100 - dist) / 100.0) * w
    if proximity < 0:
     proximity = 0
    pygame.draw.rect(screen, (0, 255, 0), Rect((10, 10),(w, 50)))
    pygame.draw.rect(screen, (255, 0, 0), Rect((10, 10),(proximity,
50)))
    pygame.display.update()
```

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

La distance est mesurée et un message est composé sur une surface qui est ensuite affichée à l'écran. La représentation graphique est créée en dessinant un rectangle vert de taille fixe, puis on trace un rectangle rouge par-dessus, dont la largeur est calculée en fonction de la distance mesurée par le télémètre.

11.4 RÉSUMÉ

Ce projet peut être considéré comme étant le point de départ de vos propres projets robotiques. La RaspiRobotBoard dispose de deux sorties supplémentaires qui pourraient servir à contrôler d'autres appareils électroniques. Une autre extension intéressante du projet serait d'écrire un programme qui permette au robot de tourner sur place et d'utiliser le télémètre pour créer un graphique de la pièce, comme ceux que l'on réalise à l'aide d'un sonar. Avec le module de caméra du Raspberry Pi et un dongle Wi-Fi, il y a énormément d'autres possibilités pour créer des appareils de télésurveillance.

Le dernier chapitre de cet ouvrage vous propose d'explorer d'autres pistes pour aller encore plus loin avec votre Raspberry Pi, et fournit des références utiles vers d'autres ressources Raspberry Pi.

Copyright © 2014 Dunod.

CHAPITRE PAGE

12 POUR ALLER PLUS LOIN

1	Introduction	3
2	Prise en main	17
3	Bases de Python	27
4	Chaînes, listes et dictionnaires	43
5	Modules, classes et méthodes	69
6	Fichiers et Internet	81
7	Interfaces utilisateur graphiques	91
8	Programmation de jeux	109
9	Interfaçage du matériel	123
10	Projet de prototypage d'une horloge	137
11	Le RaspiRobot	147
12	Pour aller plus loin	161

12.1 Ressources Linux **162**
12.2 Ressources Python **162**
12.3 Ressources Raspberry Pi **163**
12.4 Autres langages de programmation **164**
12.5 Applications et projets **166**
12.6 Résumé **166**

Copyright © 2014 Dunod.

Le Raspberry Pi est un appareil extrêmement souple que vous pouvez utiliser dans toutes sortes de situations, que ce soit pour remplacer votre ordinateur de bureau, profiter d'un *media center* ou contrôler un système à l'aide d'un ordinateur embarqué.

Ce chapitre fournit quelques conseils sur les différentes manières d'utiliser votre Raspberry Pi et recense quelques ressources sur la programmation du Raspberry Pi et son usage au sein d'une maison.

12.1 RESSOURCES LINUX

Le Raspberry Pi est, bien sûr, l'un des nombreux ordinateurs qui s'exécutent sous Linux. Vous trouverez des informations utiles dans la plupart des livres sur Linux ; cherchez en particulier des livres qui traitent de la distribution que vous utilisez, en l'occurrence Debian pour la Raspbian.

Mis à part le gestionnaire de fichiers et les applications qui nécessitent de plus amples explications, vous aurez sans doute envie d'en apprendre plus sur l'utilisation du Terminal et la configuration de Linux. Nous vous conseillons dans ce domaine l'ouvrage de William E. Shotts, Jr, *The Linux Command Line: A Complete Introduction*. On trouve aussi de bonnes ressources pour apprendre Linux sur l'Internet, et votre moteur de recherche pourra vous guider.

12.2 RESSOURCES PYTHON

Python n'est pas spécifique au Raspberry Pi, et vous pouvez trouver de nombreux livres et des ressources Internet qui lui sont consacrés. Pour une introduction en douceur à Python, vous pouvez lire *Python: Visual QuickStart Guide*, de Toby Donaldson. Il a un style similaire à ce livre, mais offre une perspective différente. En outre, il est écrit d'une manière conviviale qui rassure. Si vous voulez quelque chose d'un peu plus roboratif, mais qui s'adresse encore au débutant, vous pouvez opter pour *Python Programming: An Introduction to Computer Science*, de John Zelle.

Si vous voulez mieux connaître Pygame, vous trouverez très utile la lecture de *Beginning Game Development with Python and Pygame*, de Will McGugan.

Pour finir, voici quelques bonnes ressources web sur Python que vous aurez probablement envie d'ajouter à la liste des favoris de votre navigateur :

Copyright © 2014 Dunod.

- *http://docs.python.org/py3k/* Le site officiel de Python, complet avec des tutoriels utiles et des documents de référence.
- *www.pythonware.com/library/tkinter/introduction/* Une référence utile sur Tkinter.
- *http://zetcode.com/gui/tkinter/layout/* Ce tutoriel indispensable explique comment positionner les contrôles dans Tkinter.
- *www.pygame.org* Le site officiel de Pygame. Il contient des actualités, des tutoriels, des documents de référence et des exemples de code.

12.3 RESSOURCES RASPBERRY PI

Le site officiel de la Fondation Raspberry Pi est *www.raspberrypi.org*. Ce site contient une foule de renseignements utiles, et c'est l'endroit pour trouver des annonces sur les événements de l'univers Raspberry Pi.

Les forums sont particulièrement utiles lorsque vous cherchez la réponse à un problème épineux. Vous pouvez rechercher dans le forum d'autres personnes qui ont déjà tenté de réaliser ce que vous essayez de faire ; vous pouvez aussi poser des questions ou simplement montrer à la communauté ce que vous réalisez. Lorsque vous cherchez à mettre à jour l'image de votre distribution Raspberry Pi, c'est probablement le meilleur endroit pour trouver votre bonheur. La page de téléchargements répertorie les distributions actuellement en vogue.

Le Raspberry Pi a même son propre magazine en ligne, nommé avec humour *The MagPi*. C'est un fichier PDF que l'on peut télécharger gratuitement (*www.themagpi.com*) et qui contient à la fois des articles sur les fonctionnalités et des tutoriels qui vous donneront envie de faire de grandes choses avec votre Pi.

Pour de plus amples informations sur le côté matériel de l'utilisation du Raspberry Pi, vous trouverez les liens suivants utiles :

- *http://elinux.org/RPi_VerifiedPeripherals* Une liste des périphériques vérifiés comme fonctionnant avec le Raspberry Pi.
- *http://elinux.org/RPi_Low-level_peripherals* Une liste des périphériques à interfacer avec le connecteur GPIO.
- *www.element14.com/community/docs/DOC-43016/* Une documentation technique de la puce Broadcom qui est le cœur du Raspberry Pi.

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Si vous voulez acheter des modules complémentaires et des composants pour votre Raspberry Pi, Adafruit a une section entière consacrée au Raspberry Pi. SparkFun vend également des modules et des cartes additionnelles pour le Raspberry Pi.

12.4 AUTRES LANGAGES DE PROGRAMMATION

Dans cet ouvrage, nous avons étudié exclusivement la programmation du Raspberry Pi en Python, en raison du fait que Python est un langage populaire qui offre un bon compromis entre facilité d'utilisation et puissance. Cependant, Python n'est en aucun cas le seul langage de programmation du Raspberry Pi. La distribution Raspbian Wheezy inclut notamment plusieurs autres langages.

Scratch

Scratch est un langage de programmation visuelle développé par le MIT. Il est devenu populaire dans les milieux éducatifs comme moyen d'encourager des jeunes à apprendre la programmation. Scratch inclut son propre environnement de développement, comme IDLE pour Python, mais la programmation s'effectue en glissant et en déposant des structures de programmation au lieu de saisir simplement du texte.

La figure 12.1 illustre l'un des exemples de programmes fournis avec Scratch pour le jeu *Pong*, où une balle rebondit sur une raquette.

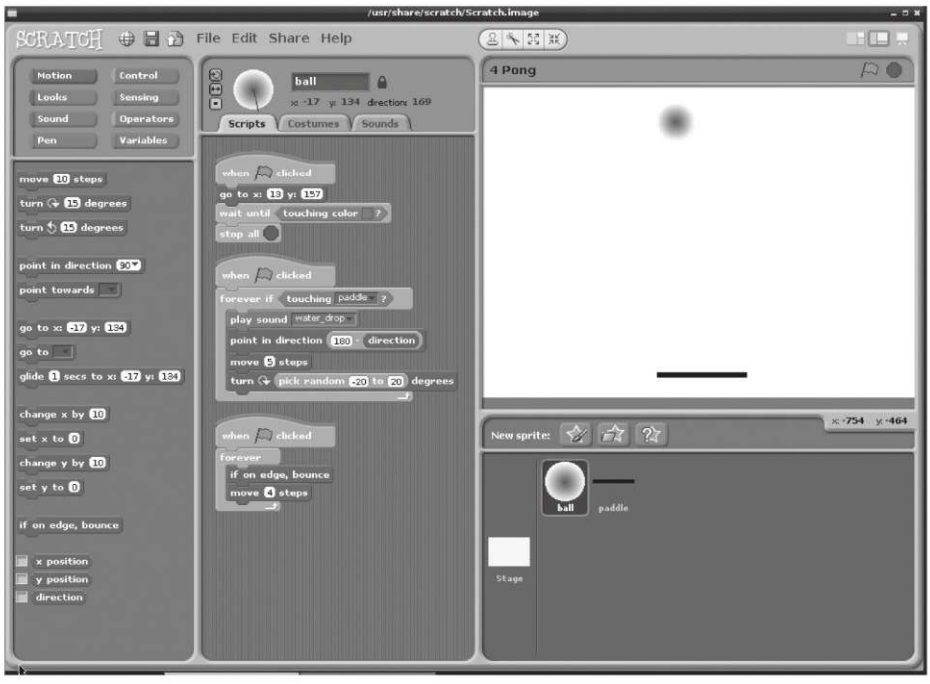


Figure 12.1
Édition d'un programme en Scratch.

Copyright © 2014 Dunod.

C

Le langage de programmation C est le langage utilisé pour programmer Linux, et le compilateur C GNU est inclus dans la distribution Raspbian Wheezy.

Pour tester un petit « Hello World » programmé en C, IDLE permet de créer un fichier avec le contenu suivant :

```
#include<stdio.h>
main()
{
   printf(«\n\nHello World\n\n»);
}
```

Enregistrez le fichier sous le nom de hello.c. Ensuite, depuis le même répertoire où ce fichier est enregistré, saisissez la commande suivante dans le terminal :

```
gcc hello.c -o hello
```

Cela exécute le compilateur C (gcc), qui va convertir le fichier hello.c en un programme exécutable appelé simplement `hello`. Vous pouvez l'exécuter depuis la ligne de commande en saisissant :

```
./hello
```

La fenêtre de l'éditeur IDLE et la ligne de commande sont illustrées à la figure 12.2, où vous pouvez également voir le résultat du programme. Vous remarquerez que les caractères `\n` créent des lignes vides autour du message.

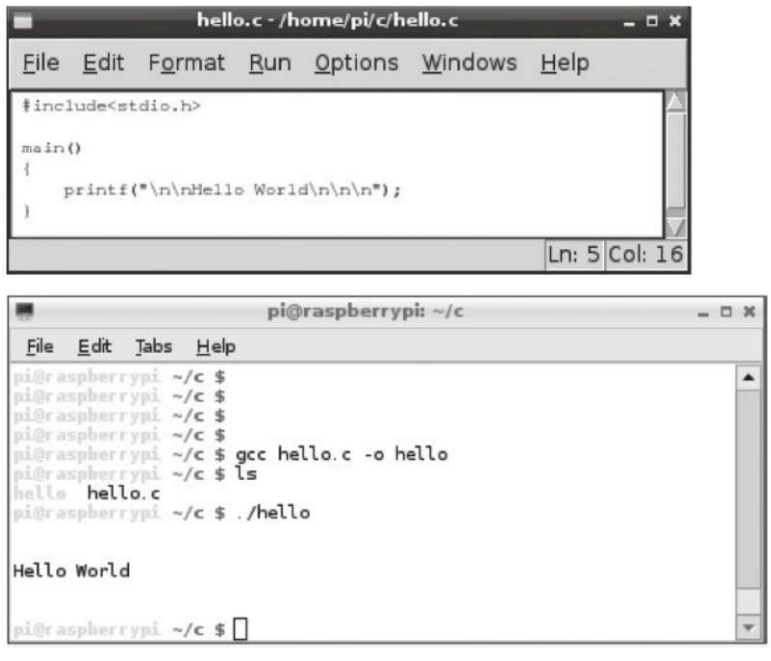


Figure 12.2
Compilation d'un programme C.

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2014 Dunod.

12.5 APPLICATIONS ET PROJETS

Toute nouvelle technologie comme le Raspberry Pi attire une communauté d'amateurs innovants qui trouvent des utilisations intéressantes au Raspberry Pi. Nous détaillons dans les sections suivantes deux projets dignes d'intérêt.

Media Center (Raspbmc)

Raspbmc est une distribution pour le Raspberry Pi qui le transforme en *media center*, que vous pouvez utiliser pour lire des films et des fichiers audio stockés sur un support USB connecté au Pi ; vous pouvez aussi lire des fichiers audio et vidéo à partir d'autres appareils comme les iPad qui sont connectés à votre réseau domestique. Raspbmc est issu du projet XBMC qui a remporté un grand succès et qui a débuté comme un projet pour utiliser la Xbox de Microsoft comme *media center*. Toutefois, il est disponible sur un large éventail de plates-formes.

Étant donné le coût peu élevé du Raspberry Pi, il est probable que beaucoup d'entre eux se retrouvent dans de petits boîtiers à côté du téléviseur, surtout depuis que la plupart des téléviseurs ont un port USB qui peut alimenter le Pi.

Vous pouvez trouver plus d'informations sur Raspbmc à *www.raspbmc.com/about/* et vous pourrez découvrir le projet XBMC à *www.xbmc.org*. Tous ces logiciels sont bien sûr open source.

Domotique

De nombreux projets à petite échelle utilisent le Raspberry Pi pour la domotique. La facilité avec laquelle on peut y connecter des capteurs et des actionneurs, directement ou via un Arduino, fait du Pi un appareil parfaitement adapté pour agir comme centre de contrôle.

Dans la plupart des approches, le Raspberry Pi héberge un serveur web sur le réseau local afin qu'on puisse depuis un navigateur Internet contrôler les différentes fonctions de la maison, comme allumer / éteindre les lumières ou contrôler le thermostat.

12.6 RÉSUMÉ

Le Raspberry Pi est un appareil très souple et peu coûteux pour lequel vous pourrez trouver de nombreux usages. Même comme simple ordinateur pour surfer sur le web à partir de l'écran de la télévision, il remplit parfaitement son rôle (et reste beaucoup moins cher que la plupart des autres méthodes). Vous finirez probablement par acheter vous-même plusieurs Raspberry Pi que vous intégrerez dans différents projets domotiques.

Copyright © 2014 Dunod.

ANNEXE

LISTE DES PRINCIPALES FONCTIONS DE PYGAME

- **pygame.Color :** objets pygame pour les représentations de la couleur.
  - Color.a : lit ou écrit la valeur alpha de la couleur.
  - Color.b : lit ou écrit la valeur bleu de la couleur.
  - Color.cmy : lit ou écrit la représentation CMY de la couleur.
  - Color.correct_gamma : applique une valeur gamma à la couleur.
  - Color.g : lit ou écrit la valeur verte de la couleur.
  - Color.hsla : lit ou écrit la représentation HSLA de la couleur.
  - Color.hsva : lit ou écrit la représentation HSVA de la couleur.
  - Color.i1i2i3 : lit ou écrit la représentation I1I2I3 de la couleur.
  - Color.normalize : retourne les valeurs normalisées RGBA de la couleur.
  - Color.r : lit ou écrit la valeur rouge de la couleur.
  - Color.set_length : écrit le nombre d'éléments de la couleur (1, 2, 3 ou 4).
- **pygame.Overlay :** objet pygame pour la gestion des overlays.
  - Overlay.display : écrit les données des pixels de l'overlay.
  - Overlay.get_hardware : teste l'accélération matérielle de l'overlay.
  - Overlay.set_location : contrôle l'emplacement de l'affichage de l'overlay.
- **pygame.PixelArray :** objet pygame pour l'accès direct des pixels des surfaces.
  - PixelArray.compare : compare un PixelArray à un autre PixelArray.
  - PixelArray.extract : extrait la couleur passée dans le PixelArray.
  - PixelArray.make_surface : crée une nouvelle surface à partir du PixelArray en cours.
  - PixelArray.replace : remplace la couleur passée du PixelArray.
  - PixelArray.surface : lit la surface utilisée par le PixelArray.
- **pygame.Rect :** objet pygame pour stocker des coordonnées rectangulaires.
  - Rect.clamp : déplace le rectangle à l'intérieur d'un autre rectangle.
  - Rect.clamp_ip : version « in place » de Rect.clamp.
  - Rect.clip : rogne un rectangle à l'intérieur d'un autre rectangle.
  - Rect.collidedict : teste si un rectangle d'un dictionnaire rentre en collision.
  - Rect.collidedictall : teste si tous les rectangles d'un dictionnaire rentrent en collision.
  - Rect.collidelist : teste si un rectangle d'une liste rentre en collision.
  - Rect.collidelistall : teste si tous les rectangles d'une liste rentrent en collision.
  - Rect.collidepoint : teste si un point est à l'intérieur d'un rectangle.

Copyright © 2014 Dunod.

- Rect.colliderect : teste si deux rectangles se recouvrent.
- Rect.contains : teste si un rectangle est à l'intérieur d'un autre rectangle.
- Rect.copy : copie le rectangle.
- Rect.fit : redimensionne et déplace un rectangle en gardant les proportions.
- Rect.inflate : agrandit ou diminue la taille d'un rectangle.
- Rect.inflate_ip : version « in place » de Rect.inflate.
- Rect.move : déplace le rectangle.
- Rect.move_ip : version « in place » de Rect.move.
- Rect.normalize : corrige les tailles négatives.
- Rect.union : rassemble deux rectangles en un seul.
- Rect.union_ip : version « in place » de Rect.union.
- Rect.unionall : union de plusieurs rectangles.
- Rect.unionall_ip : version « in place » de Rect.unionall.

- **pygame.Surface :** objet pygame pour la représentation des images.
  - Surface.blit : trace une image sur une autre image.
  - Surface.convert : change le format d'une image.
  - Surface.convert_alpha : change le format d'une image en incluant les alphas par pixel.
  - Surface.copy : crée une nouvelle copie d'un objet Surface.
  - Surface.fill : remplit la Surface avec une couleur solide.
  - Surface.get_abs_offset : trouve la position absolue d'une sous-surface enfant à l'intérieur de son parent de niveau supérieur.
  - Surface.get_abs_parent : trouve le parent de niveau supérieur d'une sous-surface.
  - Surface.get_alpha : lit la valeur de transparence de la Surface en cours.
  - Surface.get_at : lit la valeur de couleur d'un seul pixel.
  - Surface.get_bitsize : lit la profondeur de bit du format de pixel d'une Surface.
  - Surface.get_bounding_rect : trouve le plus petit rectangle contenant des données.
  - Surface.get_buffer : acquiert un objet buffer pour les pixels d'une Surface.
  - Surface.get_bytesize : lit les octets utilisés par pixel de Surface.
  - Surface.get_clip : lit la zone de clipping en cours de la Surface.
  - Surface.get_colorkey : lit l'attribut colorkey de transparence en cours.
  - Surface.get_flags : lit les paramètres supplémentaires utilisés par la Surface.
  - Surface.get_height : lit la hauteur de la Surface.
  - Surface.get_locked : teste si la Surface est verrouillée.
  - Surface.get_locks : lit les verrous de la Surface.
  - Surface.get_offset : trouve la position d'une sous-surface enfant dans un surface parent.
  - Surface.get_palette : lit la palette des couleurs d'une Surface 8-bits.
  - Surface.get_palette_at : lit la couleur d'une entrée unique d'une palette.
  - Surface.get_parent : trouve le parent d'une sous-surface.
  - Surface.get_pitch : lit le nombre d'octets utilisés par ligne de Surface.
  - Surface.get_rect : lit la zone rectangulaire de la Surface.
  - Surface.get_size : lit les dimensions de la Surface.
  - Surface.get_width : lit la largeur de la Surface.
  - Surface.lock : verrouille la mémoire de la Surface.
  - Surface.map_rgb : convertit une couleur en une valeur de couleur mappée.
  - Surface.mustlock : teste si la Surface doit être verrouillée.

Copyright © 2014 Dunod.

- Surface.scroll : décale l'image de la Surface.
- Surface.set_alpha : écrit la valeur alpha de l'image complète de la Surface.
- Surface.set_at : écrit la valeur de couleur d'un pixel unique.
- Surface.set_clip : écrit la zone de clipping en cours de la Surface.
- Surface.set_colorkey : définit l'attribut colorkey de transparence.
- Surface.set_palette : définit la palette de couleurs d'une Surface 8-bits.
- Surface.subsurface : crée une nouvelle surface qui référence son parent.
- Surface.unlock : déverrouille la mémoire de la Surface.

- **pygame.cursors :** module pygame pour la gestion des curseurs.
  - pygame.cursors.compile : crée des données binaires de curseur à partir de simples chaînes.
  - pygame.cursors.load_xbm : charge des données de curseur à partir d'un fichier xbm.
- **pygame.display :** module pygame pour contrôler l'écran et la fenêtre d'affichage.
  - pygame.display.Info : créer un objet d'information d'affichage vidéo.
  - pygame.display.flip : met à jour l'affichage complet de la Surface à l'écran.
  - pygame.display.get_active : vrai quand l'affichage est actif.
  - pygame.display.get_caption : lit le titre de la fenêtre.
  - pygame.display.get_driver : lit le nom du backend.
  - pygame.display.get_init : vrai si le module d'affichage est initialisé.
  - pygame.display.get_surface : lit une référence de l'affichage en cours.
  - pygame.display.get_wm_info : lit des informations sur le système de fenêtrage en cours.
  - pygame.display.gl_get_attribute : lit la valeur d'un attribut opengl de l'affichage.
  - pygame.display.gl_set_attribute : demande un attribut opengl du mode d'affichage.
  - pygame.display.iconify : met en icône l'affichage.
  - pygame.display.init : initialise le module d'affichage.
  - pygame.display.list_modes : liste des modes d'affichage plein écran disponibles.
  - pygame.display.mode_ok : sélectionne la meilleure profondeur de couleur d'un mode d'affichage.
  - pygame.display.quit : quitte le module d'affichage.
  - pygame.display.set_caption : définit le titre de la fenêtre en cours.
  - pygame.display.set_icon : modifie l'image système de la fenêtre d'affichage.
  - pygame.display.set_mode : initialise une fenêtre ou un écran.
  - pygame.display.set_palette : définit la palette de couleurs.
  - pygame.display.toggle_fullscreen : bascule entre l'affichage plein écran et le mode fenêtre.
  - pygame.display.update : met à jour les portions de l'écran pour les affichages logiciels.
- **pygame.draw :** module pygame pour tracer des formes.
  - pygame.draw.aaline : trace des lignes anticrénelées.
  - pygame.draw.aalines : trace une suite de lignes anticrénelées.
  - pygame.draw.arc : trace une section partielle d'ellipse.
  - pygame.draw.circle : trace un cercle autour d'un point.
  - pygame.draw.ellipse : trace une forme arrondie à l'intérieur d'un rectangle.
  - pygame.draw.line : trace un segment de ligne droite.
  - pygame.draw.lines : trace plusieurs segments de lignes continues.

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

- pygame.draw.polygon : trace une forme ayant plusieurs côtés.
- pygame.draw.rect : trace un rectangle.

- **pygame.error :** exception pygame standard.
- **pygame.event :** module pygame pour communiquer avec les événements et les files d'attente.
  - pygame.event.Event : crée un nouvel objet événement.
  - pygame.event.clear : supprime tous les événements de la file d'attente.
  - pygame.event.event_name : lit le nom d'un événement à partir de son ID.
  - pygame.event.get : lit les événements dans la file d'attente.
  - pygame.event.get_blocked : teste si un type d'événement est bloqué dans la file d'attente.
  - pygame.event.get_grab : teste si le programme partage des périphériques d'entrée.
  - pygame.event.peek : teste si des types d'événements sont dans la file d'attente.
  - pygame.event.poll : lit un événement dans la file d'attente.
  - pygame.event.post : place un nouvel événement dans la file d'attente.
  - pygame.event.pump : traite en interne les gestionnaires d'événements pygame.
  - pygame.event.set_allowed : contrôle les événements autorisés dans la file d'attente
  - pygame.event.set_blocked : contrôle les événements non autorisés dans la file d'attente.
  - pygame.event.set_grab : contrôle le partage des périphériques d'entrée par les autres applications.
  - pygame.event.wait : attend un événement unique dans la file d'attente.
- **pygame.examples :** module d'exemples de programmes.
  - pygame.examples.aliens.main : exemple complet d'aliens.
  - pygame.examples.arraydemo.main : montre différents effets surfarray.
  - pygame.examples.blend_fill.main : options blend de la méthode surface.fill.
  - pygame.examples.chimp.main : exemple de singe qui se déplace
  - pygame.examples.cursors.main : affiche deux curseurs personnalisés.
  - pygame.examples.eventlist.main : affiche les événements pygame.
  - pygame.examples.fastevents.main : test du module fastevents.
  - pygame.examples.fonty.main : exécute un exemple d'affichage de police.
  - pygame.examples.glcube.main : affiche un cube animé en 3D avec OpenGL.
  - pygame.examples.headless_no_windows_needed.main : écrit un fichier image qui est une copie mise à l'échelle d'un fichier passé en paramètre.
  - pygame.examples.liquid.main : affiche un effet liquide animé.
  - pygame.examples.mask.main : exemple de détection de collision.
  - pygame.examples.midi.main : exécute un exemple midi.
  - pygame.examples.moveit.main : affiche des objets animés à l'écran.
  - pygame.examples.movieplayer.main : joue un film mpeg.
  - pygame.examples.oldalien.main : exécute l'exemple original des aliens.
  - pygame.examples.overlay.main : joue une vidéo .pgm en utilisant des overlays.

Copyright © 2014 Dunod.

- pygame.examples.pixelarray.main : affiche différents effets générés avec des tableaux de pixels.
- pygame.examples.scaletest.main : met à l'échelle de manière interactive une image.
- pygame.examples.scrap_clipboard.main : accède au presse-papiers.
- pygame.examples.scroll.main : exemple de Surface.scroll qui illustre une image agrandie.
- pygame.examples.sound.main : charge et joue un son.
- pygame.examples.sound_array_demos.main : exécute différents effets sndarray.
- pygame.examples.testsprite.main : déplacement de nombreux sprites.
- pygame.examples.vgrade.main : affiche un gradient vertical.

- **pygame.font :** module pygame pour le chargement et l'affichage des polices.
  - pygame.font.Font : crée un nouvel objet Font à partir d'un fichier.
    - Font.get_ascent : retourne en pixels la hauteur du jambage ascendant de la police.
    - Font.get_bold : vérifie si le texte affiché est en gras.
    - Font.get_descent : retourne en pixels la hauteur du jambage descendant de la police.
    - Font.get_height : lit la hauteur de la police.
    - Font.get_italic : vérifie si le texte affiché est en italique.
    - Font.get_linesize : lit l'interligne du texte de la police.
    - Font.get_underline : vérifie si le texte affiché est souligné.
    - Font.metrics : lit la métrique de chaque caractère de la chaîne passée en paramètre.
    - Font.render : affiche le texte sur une nouvelle Surface.
    - Font.set_bold : active un pseudo affichage du texte en gras.
    - Font.set_italic : active un pseudo affichage du texte en italique.
    - Font.set_underline : contrôle si le texte est affiché en souligné.
    - Font.size : détermine l'espace nécessaire pour afficher le texte.
  - pygame.font.SysFont : crée un objet Font à partir des polices système.
  - pygame.font.get_default_font : le lit nom du fichier de la police par défaut.
  - pygame.font.get_fonts : lit toutes les polices disponibles.
  - pygame.font.get_init : vrai si le module font est initialisé.
  - pygame.font.init : initialise le module font.
  - pygame.font.match_font : trouve une police spécifique sur le système.
  - pygame.font.quit : quitte le module font.
- **pygame.get_error :** lit le message d'erreur en cours.
- **pygame.get_sdl_version :** retourne le numéro de version de SDL.
- **pygame.gfxdraw :** module pygame pour le traçage de formes.
  - pygame.gfxdraw.aacircle : trace un cercle anticrénelé.
  - pygame.gfxdraw.aaellipse : trace une ellipse anticrénelée.
  - pygame.gfxdraw.aapolygon : trace un polygone anticrénelé.
  - pygame.gfxdraw.aatrigon : trace un triangle anticrénelé.
  - pygame.gfxdraw.arc : trace un arc.
  - pygame.gfxdraw.bezier : trace une courbe de Bézier.
  - pygame.gfxdraw.box : trace une boîte.
  - pygame.gfxdraw.circle : trace un cercle.
  - pygame.gfxdraw.ellipse : trace une ellipse.
  - pygame.gfxdraw.filled_circle : trace un cercle plein.
  - pygame.gfxdraw.filled_ellipse : trace une ellipse pleine.
  - pygame.gfxdraw.filled_polygon : trace un polygone plein.
  - pygame.gfxdraw.filled_trigon : trace un trigone plein.

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

- pygame.gfxdraw.hline : trace une ligne horizontale.
- pygame.gfxdraw.line : trace une ligne.
- pygame.gfxdraw.pixel : place un pixel.
- pygame.gfxdraw.polygon : trace un polygone.
- pygame.gfxdraw.rectangle : trace un rectangle.
- pygame.gfxdraw.textured_polygon : trace un polygone avec une texture.
- pygame.gfxdraw.trigon : trace un triangle.
- pygame.gfxdraw.vline : trace une ligne verticale.

- **pygame.image :** module pygame pour le transfert des images.
  - pygame.image.frombuffer : crée une nouvelle Surface qui partage les données dans un buffer.
  - pygame.image.fromstring : créer une nouvelle Surface à partir d'un buffer.
  - pygame.image.get_extended : teste si les formats étendus d'images peuvent être chargés.
  - pygame.image.load : charge une nouvelle image à partir d'un fichier.
  - pygame.image.save : enregistre une image sur le disque.
  - pygame.image.tostring : transfère une image dans un buffer.
- **pygame.init :** initialise tous les modules pygame importés.
- **pygame.key :** module pygame pour gérer le clavier.
  - pygame.key.get_focused : vrai si l'affichage reçoit une entrée clavier à partir du système.
  - pygame.key.get_mods : détermine quelle touche modificatrice (Maj, Verr Num…) est enfoncée.
  - pygame.key.get_pressed : lit l'état de toutes les touches du clavier.
  - pygame.key.get_repeat : lit le délai de répétition des touches.
  - pygame.key.name : lit le nom d'un identificateur de touche.
  - pygame.key.set_mods : définit temporairement quelle touche modificatrice est enfoncée.
  - pygame.key.set_repeat : contrôle le délai de répétition des touches.
- **pygame.locals :** constantes pygame.
- **pygame.mask :** module pygame de gestion des masques d'image.
  - pygame.mask.Mask : objet pygame pour la représentation des masques en 2D.
    - Mask.angle : retourne l'orientation des pixels.
    - Mask.centroid : retourne le centroïde des pixels d'un objet Mask.
    - Mask.clear : définit tous les bits à 0.
    - Mask.connected_component : retourne un masque d'une région connectée de pixels.
    - Mask.connected_components : retourne une liste des masques des régions connectées de pixels.
    - Mask.count : retourne le nombre de pixels définis.
    - Mask.draw : trace un masque sur un autre masque.
    - Mask.erase : efface un masque.
    - Mask.fill : définit tous les bits à 1.
    - Mask.get_at : retourne un entier différent de zéro si le bit à (x,y) est défini.
    - Mask.get_size : retourne la taille du masque.
    - Mask.invert : inverse les bits d'un masque.
    - Mask.outline : liste des points encadrant un objet.
    - Mask.overlap : retourne le point d'intersection si les masques se superposent.
    - Mask.overlap_area : retourne le nombre de pixels se superposant.
    - Mask.overlap_mask : retourne un masque des pixels se superposant.

Copyright © 2014 Dunod.

        - Mask.scale : redimensionne un masque.
        - Mask.set_at : définit la position d'un masque.
    - pygame.mask.from_surface : retourne un masque à partir d'une surface donnée.
- **pygame.mixer :** module pygame pour la gestion des sons.
    - pygame.mixer.Channel : crée un objet Channel pour contrôler la lecture
        - Channel.fadeout : arrête la lecture en diminuant progressivement le son.
        - Channel.get_busy : vérifie si l'objet Channel est actif.
        - Channel.get_endevent : lit l'événement envoyé par un canal quand la lecture s'arrête
        - Channel.get_queue : retourne un son mis en file d'attente.
        - Channel.get_sound : retourne l'objet Sound qui est joué.
        - Channel.get_volume : retourne le volume du canal en cours de lecture.
        - Channel.pause : met en pause temporairement la lecture d'un canal.
        - Channel.play : joue un son sur un canal spécifique.
        - Channel.queue : met en file d'attente un objet Sound juste après l'objet en cours de lecture.
        - Channel.set_volume : définit le volume du canal.
        - Channel.stop : arrête la lecture sur un canal.
        - Channel.unpause : reprend la lecture d'un canal après sa mise en pause.
    - pygame.mixer.Sound : crée un nouvel objet Sound à partir d'un fichier.
        - Sound.fadeout : arrête la lecture d'un son en baissant progressivement le volume.
        - Sound.get_length : lit la longueur du son.
        - Sound.get_num_channels : compte le nombre de lectures du son.
        - Sound.get_volume : lit le volume de lecture.
        - Sound.play : commence la lecture du son.
        - Sound.set_volume : définit le volume de lecture du son.
        - Sound.stop : arrête la lecture du son.
    - pygame.mixer.fadeout : diminue le volume de tous les sons avant de l'arrêter.
    - pygame.mixer.find_channel : trouve un canal non utilisé.
    - pygame.mixer.get_busy : teste si un son est joué sur le mixer.
    - pygame.mixer.get_init : teste si le mixer est initialisé.
    - pygame.mixer.get_num_channels : retourne le nombre total de canaux de lecture.
    - pygame.mixer.init : initialise le module de mixer.
    - pygame.mixer.music : module pygame pour le contrôle des sons en streaming.
        - pygame.mixer.music.fadeout : arrête progressivement la lecture de la musique.
        - pygame.mixer.music.get_busy : vérifie si la musique est en cours de lecture.
        - pygame.mixer.music.get_pos : retourne le temps de lecture de la musique
        - pygame.mixer.music.get_volume : retourne le volume de la musique.
        - pygame.mixer.music.load : charge un fichier de musique à lire.
        - pygame.mixer.music.pause : arrête temporairement la lecture de la musique.
        - pygame.mixer.music.play : démarre la lecture du flux de musique.

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

        - pygame.mixer.music.queue : met en file d'attente un fichier de musique.
        - pygame.mixer.music.rewind : redémarre la lecture de la musique depuis le début.
        - pygame.mixer.music.set_volume : définit le volume de la musique.
        - pygame.mixer.music.stop : arrête la lecture de la musique.
        - pygame.mixer.music.unpause : reprend la lecture de la musique mise en pause.
      - pygame.mixer.pause : arrête temporairement la lecture de tous les canaux.
      - pygame.mixer.pre_init : prédéfinit les arguments init du mixer.
      - pygame.mixer.quit : quitte le mixer.
      - pygame.mixer.set_num_channels : définit le nombre total de canaux de lecture.
      - pygame.mixer.set_reserved : réserve des canaux pour qu'ils ne soient pas lus automatiquement.
      - pygame.mixer.stop : arrête la lecture de tous les canaux.
      - pygame.mixer.unpause : reprend la lecture de tous les canaux mis en pause.
  - **pygame.mouse :** module pygame pour gérer la souris.
      - pygame.mouse.get_cursor : retourne l'image du curseur système de la souris.
      - pygame.mouse.get_focused : vérifie si l'affichage reçoit l'entrée de la souris.
      - pygame.mouse.get_pos : lit la position du curseur de la souris.
      - pygame.mouse.get_pressed : lit l'état des boutons de la souris.
      - pygame.mouse.get_rel : lit l'amplitude du mouvement de la souris.
      - pygame.mouse.set_cursor : définit l'image du curseur système de la souris.
      - pygame.mouse.set_pos : définit la position du curseur de la souris.
      - pygame.mouse.set_visible : masque ou affiche le curseur de la souris.
  - **pygame.movie :** module pygame pour la lecture des vidéos au format mpeg.
      - pygame.movie.Movie : charge un fichier mpeg.
        - Movie.get_busy : vérifie si le film est en cours de lecture.
        - Movie.get_frame : retourne le numéro du frame en cours de lecture.
        - Movie.get_length : longueur totale du film en secondes.
        - Movie.get_size : retourne la résolution de la vidéo.
        - Movie.has_audio : vérifie si le fichier du film contient du son.
        - Movie.has_video : vérifie si le fichier du film contient de la vidéo.
        - Movie.pause : arrête temporairement et reprend la lecture.
        - Movie.play : démarre la lecture d'un film.
        - Movie.render_frame : définit le frame en cours de lecture.
        - Movie.rewind : redémarre la lecture du film.
        - Movie.set_display : définit la Surface cible de la vidéo.
        - Movie.set_volume : définit le volume audio de lecture.
        - Movie.skip : avance la position de lecture du film.
        - Movie.stop : arrête la lecture du film.
  - **pygame.quit :** quitte tous les modules pygame.
  - **pygame.register_quit :** enregistre une fonction à appeler quand pygame se termine.
  - **pygame.scrap :** module pygame pour la prise en charge du presse-papiers.
      - pygame.scrap.contains : vérifie si un certain type est disponible dans le presse-papiers.

Copyright © 2014 Dunod.

- pygame.scrap.get : retourne les données du type spécifié à partir du presse-papiers.
- pygame.scrap.get_types : retourne une liste des types de presse-papiers disponibles.
- pygame.scrap.init : initialise le module scrap.
- pygame.scrap.lost : vérifie si le presse-papiers est actuellement détenu par une application.
- pygame.scrap.put : place des données dans le presse-papiers.
- pygame.scrap.set_mode : définit le mode d'accès du presse-papiers.

- **pygame.set_error :** définit le message d'erreur en cours.
- **pygame.sprite :** module pygame de gestion des classes d'objets basiques de jeu.
  - pygame.sprite.DirtySprite : sous-classe de sprites avec plus d'attributs.
  - pygame.sprite.Group : classe conteneur pour gérer de nombreux sprites.
    - Group.add : ajoute des sprites à un groupe.
    - Group.clear : trace un arrière-plan pour les sprites.
    - Group.copy : duplique le groupe.
    - Group.draw : trace les images du sprite.
    - Group.empty : supprime tous les sprites.
    - Group.has : teste si un groupe contient des sprites.
    - Group.remove : supprime les sprites du groupe.
    - Group.sprites : liste les sprites que le groupe contient.
    - Group.update : appelle la méthode update sur les sprites du groupe.
  - pygame.sprite.Sprite : classe de base pour gérer les objets visibles des jeux.
    - Sprite.add : ajoute le sprite aux groupes.
    - Sprite.alive : est-ce que le sprite appartient à un groupe ?
    - Sprite.groups : liste des groupes qui contiennent ce sprite.
    - Sprite.kill : supprime le sprite de tous les groupes
    - Sprite.remove : supprime le sprite des groupes.
    - Sprite.update : méthode pour contrôler le comportement du sprite.
  - pygame.sprite.collide_circle : détection de collision entre deux sprites.
  - pygame.sprite.collide_circle_ratio : détection de collision entre deux sprites.
  - pygame.sprite.collide_mask : détection de collision entre deux sprites.
  - pygame.sprite.collide_rect : détection de collision entre deux sprites.
  - pygame.sprite.collide_rect_ratio : détection de collision entre deux sprites.
  - pygame.sprite.groupcollide : trouve tous les sprites qui entre en collision entre deux groupes.
  - pygame.sprite.spritecollide : trouve les sprites d'un groupe qui a une intersection avec un autre sprite.
  - pygame.sprite.spritecollideany : simple test pour voir si un sprite a une intersection avec un groupe.
- **pygame.surfarray :** module pygame pour accéder aux pixels d'une surface avec des tableaux.
  - pygame.surfarray.array2d : copie des pixels dans un tableau 2D.
  - pygame.surfarray.array3d : copie des pixels dans un tableau 3D.
  - pygame.surfarray.array_alpha : copie des alphas de pixels dans un tableau 2D.
  - pygame.surfarray.array_colorkey : copie des valeurs colorkey dans un tableau 2D.
  - pygame.surfarray.blit_array : trace directement à partir des valeurs d'un tableau.

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

- pygame.surfarray.get_arraytype : retourne le type de tableau actif en cours.
- pygame.surfarray.get_arraytypes : retourne les types système de tableaux actuellement pris en charge.
- pygame.surfarray.make_surface : copie un tableau dans une nouvelle surface.
- pygame.surfarray.map_array : mappe un tableau 3D dans un tableau 2D.
- pygame.surfarray.pixels2d : référence les pixels dans un tableau 2D.
- pygame.surfarray.pixels3d : référence les pixels dans un tableau 3D.
- pygame.surfarray.pixels_alpha : référence les alphas des pixels dans un tableau 2D.
- pygame.surfarray.use_arraytype : définit le tableau à utiliser par le système pour les tableaux des surfaces.

- **pygame.tests :** ensemble de tests pour Pygame.
  - pygame.tests.run : exécute une suite de tests.
- **pygame.time :** module pygame pour la gestion du temps.
  - pygame.time.Clock : crée un objet pour faciliter le suivi du temps.
    - Clock.get_fps : calcule le nombre de frames par seconde.
    - Clock.get_rawtime : temps écoulé depuis le tic précédent.
    - Clock.get_time : millisecondes écoulées depuis le tic précédent.
    - Clock.tick : met à jour l'horloge.
    - Clock.tick_busy_loop : met à jour l'horloge.
  - pygame.time.delay : met en pause le programme pendant un certain temps.
  - pygame.time.get_ticks : temps écoulé en millisecondes.
  - pygame.time.set_timer : crée de manière répétée un événement dans la file d'attente des événements.
  - pygame.time.wait : met en pause le programme pendant un certain temps.
- **pygame.transform :** module pygame pour transformer les surfaces.
  - pygame.transform.average_color : trouve la couleur moyenne d'une surface.
  - pygame.transform.average_surfaces : trouve la surface moyenne de plusieurs surfaces.
  - pygame.transform.chop : retourne une copie d'une image en supprimant une zone intérieure.
  - pygame.transform.flip : retourne verticalement et horizontalement.
  - pygame.transform.laplacian : trouve les bords d'une surface.
  - pygame.transform.rotate : fait tourner une image.
  - pygame.transform.rotozoom : transformation par rotation et mise à l'échelle.
  - pygame.transform.scale : redimensionne avec une nouvelle résolution.
  - pygame.transform.scale2x : double la taille d'une image.
  - pygame.transform.threshold : trouve les pixels d'une surface qui sont dans une plage de couleur.
- **pygame.version :** module contenant des informations sur la version de Pygame.
  - pygame.version.ver : numéro de version sous la forme d'une chaîne.

Copyright © 2014 Dunod.

INDEX

Symbols

== 38

A

Abiword 24
abs 62
Adafruit 130
adaptateur HDMI
  DVI 8
adaptateur Wi-Fi 8
  compatibilité 12
affectation
  multiple 60
affichage 9
  commande 35
afficheur
  LCD 139
  module de bibliothèque 142
aléatoire
  nombre 36
  tirage 51
alerte 104
alimentation 7
  mise en garde 140
  puissance 9
  USB 9
append 65
application
  installation 23
  sortie 113
  suppression 25
apt-get
  installation d'application 23
  update 24
Arch Linux ARM 10
Arduino 132
  communication avec le Pi 133
array *Voir* tableau

B

barre de défilement 103
bibliothèque
  écriture de jeux 110
bibliothèque standard
  Python 72
bin 62
blit 112
boîte de dialogue
  modale 104
boîtier 8, 11
  plans 11
boîtier de piles 150
boucle 34, 41
  interruption 41
bouton
  ajout à l'horloge 143
break 41
broche
  entrée 124
  quantité de courant 125
  sortie 125
bureau 18

Copyright © 2014 Dunod.

C

C 165
câble micro-USB 8
cadre 100
camel-case 33
Canvas 107
capitalize 64
caractère(s)
  chaîne 44
  d'échappement 63
carte de prototypage 130
carte d'extension 126
carte SD 7, 10
  fournisseurs 7
  préparation 10
  prête à l'emploi 8
  taille supérieure à 2 Go 15
case à cocher 98
cd
  changement de répertoire 22
center 64
chaîne 44
  concaténation 46
  extraction de caractères 45
  longueur 45
chargeur USB 8
Checkbutton 98
classe 74
  conversion en module 79
  définition 74
  exemple de création 117
  héritage 77
  parente 78
clavier 8
  disposition 15
clear 67
close 83
code
  indentation 38
  saisie 36
  téléchargement 36
collection 120
colonne
  largeur 102
columnconfigure 101
commentaire 36
comparaison 38
  opérateurs 38
complex 62
condition 38
  alternative 40
configuration 14
  utilitaire 15
continuation de ligne 116
contrôle 98
  disposition 99
convert 112
coordonnées 108
copytree 86
couleur
  sélecteur 105
count 65
curselection 99

D

datetime 72
Debian 4
def 50
del 65, 66
démarrage 14
dictionnaire 58
distribution
  Linux 10, 18
distro *Voir* distribution
domotique 166
dossier
  affichage de la liste 22
DoubleVar 95, 96

E

écran 8
écran LCD 156
éditeur de programme 29
elif 40
Elinux.org 8
else 40
encapsulation 76
endswith 64
entrée 95

Copyright © 2014 Dunod.

entrée/sortie 124
erreur
  traitement 61, 83
ethernet
  prise RJ-45 6
événement 113
except 84
exception 61
exit 84
expandeur de ports 126, 127
expression régulière 89

F

False 39
FAQ 25
fenêtre
  redimensionnement 100
fichier
  affichage de la liste 22
  copie 86
  déplacement 86
  écriture 85
  fermeture 83
  lecture 82
  lecture ligne à ligne 84
  modes d'écriture 85
  sélecteur 106
fill 104
find 54, 64
float 32, 67
fonction(s) 49
  conventions de nommage 50
  conversion de types 67
  définition 50
  gestion de chaînes 63
  gestion de dictionnaires 66
  gestion de listes 65
  liste 62
  numériques 62
  paramètres 50
  plusieurs valeurs de retour 60
  trigonométriques 63
  valeur de retour 50
for
  différences avec while 41
for in 35
format 64
forum 25
Frame 94
fuseau horaire 141

G

General Purpose Input/Output *Voir* GPIO
Gertboard 129
gestionnaire de fichiers 19
get 67
glob 86
globbing 86
Gnumeric 24
GPIO 6, 72, 124
  connexion des broches 124
graphique 107
grid 95
grille 99
GUI *Voir* interface graphique
guillemet
  double 44

H

HDMI 9
  adaptateur DVI 6
  câble 8
  connecteur 6
héritage 76, 77
hex 62
home
  répertoire personnel 19
horloge
  création 138
  matériel nécessaire 138
  mode d'affichage 143
  temps réel 141
horloge interne 117
hub USB 6, 8, 13
Humble Pi 131

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

I

I2C 125, 126
  accessibilité du bus 141
i2c7segment 142
IDLE 28
  versions 28
if 37, 38
image
  affichage 112
import 70
in 67
indentation 38
index 65
init 75, 111
input 53
insert 48, 66
instance 74
int 67
integer 32
interface graphique 92
interface utilisateur 92
  contrôles 98
Internet 20, 88
invite de commande 21
isalnum 64
isalpha 64
isspace 64

J

jeu
  création 112
  programmation 110
jeu du pendu 51
JSON 89

L

Label 94
langage de programmation 28
langue 15
lecture
  fichier 83
len 45, 66
ligne
  traçage 107
ligne de commande 21
ligne de programme
  continuation 116
Linux 4, 18
  distributions 18
  distributions pour le Pi 10
list 67
Listbox 98
  éléments sélectionnés 99
  type de sélection 99
liste 46
  assignation d'une valeur 47
  de listes 48
  insertion d'un élément 48
  longueur 46
  suppression d'un élément 47
  tri 47
ljust 64
logique
  valeurs 39
lower 58, 64
ls
  liste des fichiers 22
LXTerminal 21

M

Magician Chassis 149, 150
math 72
math.factorial 63
math.log 63
math.pow 63
math.sqrt 63
MCP23S17 126, 127
media center 166
menu 106
  création 107
méthode 74, 75
Midori
  navigateur Internet 20
minuscules
  conversion 58
MISO 125
module 70
  création 79
  décompression 73

Copyright © 2014 Dunod.

importation 70
importation totale 71
installation 72, 134
nom local 71
téléchargement 72
visibilité 70
moniteur 8
MOSI 125
mot de passe
utilisateur 19
moteur
connexion 153

N

navigateur 20
nombre 32
aléatoire 35
entier 32
virgule flottante 32
nom d'utilisateur 19
nouvelle ligne 63

O

objet
orientation 74
Occidentalis 10
oct 62
open 83
opérateurs mathématiques
priorité 32
orientation objet 74
ovale
traçage 107

P

pack 93, 104
paquet 24
base de données 24
suppression 25
paramètre 35
paramètres régionaux 15
périphérique
esclave 140
Pi Cobbler 130, 139
Pi Face 126
module Python 127
Pi Plate 131
pickling 87
pop 47, 66
print 35
programmation
jeux 110
programme
commentaire 36
démarrage automatique 153
éditeur 28
enregistrement 31
exécution 31
sortie 84
pwd
affichage du répertoire courant 22
PWM 125
pygame 110
site officiel 121
PySerial 134, 152
Python 28
bibliothèque standard 72
console 29
invite de commandes 29
modules utiles 72
ressources 162
shell 29
versions 28, 110
Python 2 110
python_games 110

Q

QtonPi 10
QUIT 114

R

randint 36
random 36
range 35
Raspberry Pi
acquisition 8
adaptateur Wi-Fi 12
alimentation 7, 148
applications 23
applications disponibles 25

Copyright © 2014 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Arduino 132
boîtier 8, 11
caractéristiques 4
clavier 9
clavier sans fil 9
configuration 14
connexion des éléments 13
démarrage 14
description 5
distributions Linux 10
écran 9
fondation 10, 25
forum 25
fournisseurs 8
installation 7
interfaçage du matériel 124
langages de programmation 164
magazine 163
mise en garde alimentation 140
modèles 8
prix 5
processeur 7
ressources 162, 163
ressources Internet 25
sortie 4
souris 9
souris sans fil 9
usages 5
Raspbian Wheezy 10
paquets disponibles 25
Raspbmc 166
raspi-config 15
RaspiRobot 148
RaspiRobotBoard 128
alimentation Raspberry Pi 148
bibliothèque Python 152
fournisseur 151
RCA 9
connecteur 6
read 83, 85
readline 85
rectangle
traçage 107
redimensionnement
fenêtre 100
refactorisation 117
remove 66
répertoire
affichage de la liste 22
racine 19
répertoire de travail
changement 22
répertoire personnel
home 19
replace 65
requête web 88
résistance pull-up interne 145
return 50
reverse 66
rmtree 86
robot 148
assemblage 149
matériel nécessaire 148
round 63
rover *Voir* robot
rowconfigure 101
RPi.GPIO 145, 152
téléchargement 72

S

saisie 53, 95
SCL 125
SCLK 125
Scratch 164
Scrollbar 103
SDA 125
sélecteur de couleurs 105
sélecteur de fichiers 106
selectmode 99
self 75
serveur de temps 141
shell
Python 29
showerror 104
showinfo 104
showwarning 104
shutil 86
site web
interrogation 88
Slice of PI/O 127
sort 47, 66

Copyright © 2014 Dunod.

souris 8
  suivi des mouvements 113
sous-classe 78
SPI 125
Spinbox 99
split 65
splitlines 65
sticky 101, 102
string 72 *Voir* chaîne
StringVar 95
strip 65
stub 52
sudo 23
super-utilisateur 23
système de fichiers 86

T

tableau *Voir* liste
tableur
  installation 24
tabulation 38, 63
télémètre 155
téléviseur 8
temps
  gestion 117
terminal 21
test conditionnel 37
The MagPi 163
tirage aléatoire 51
Tk
  objet 92
Tkinter 92
  application graphique 93
  coordonnées 108
  référence 108
tkinter.filedialog 106
tkinter.messagebox 104
traitement de texte
  installation 24
trigonométrie 63
True 39
try 61, 83
tuple 59
  différences avec une liste 60
type
  conversion 67

U

upper 65
urllib.request 72
USB
  câble micro 8
  hub 6
  ports 6
utilisateur par défaut 19

V

valeur
  comparaison 38
  logique 39
  saisie 95
valeur de retour
  multiple 60
variable 32
  affichage de la valeur 34
  assignation d'une valeur 33
  chaîne 44
  conventions de nommage 33
  exemples de noms 33
  globale 51
  objet 95
  règles de nommage 33
  sauvegarde dans un fichier 87
VGA 10

W

while 41
  différences avec for 41
widget *Voir* contrôle
Wi-Fi
  adaptateur 12

X

XBMC 166
XML 89

Z

zone de liste 98