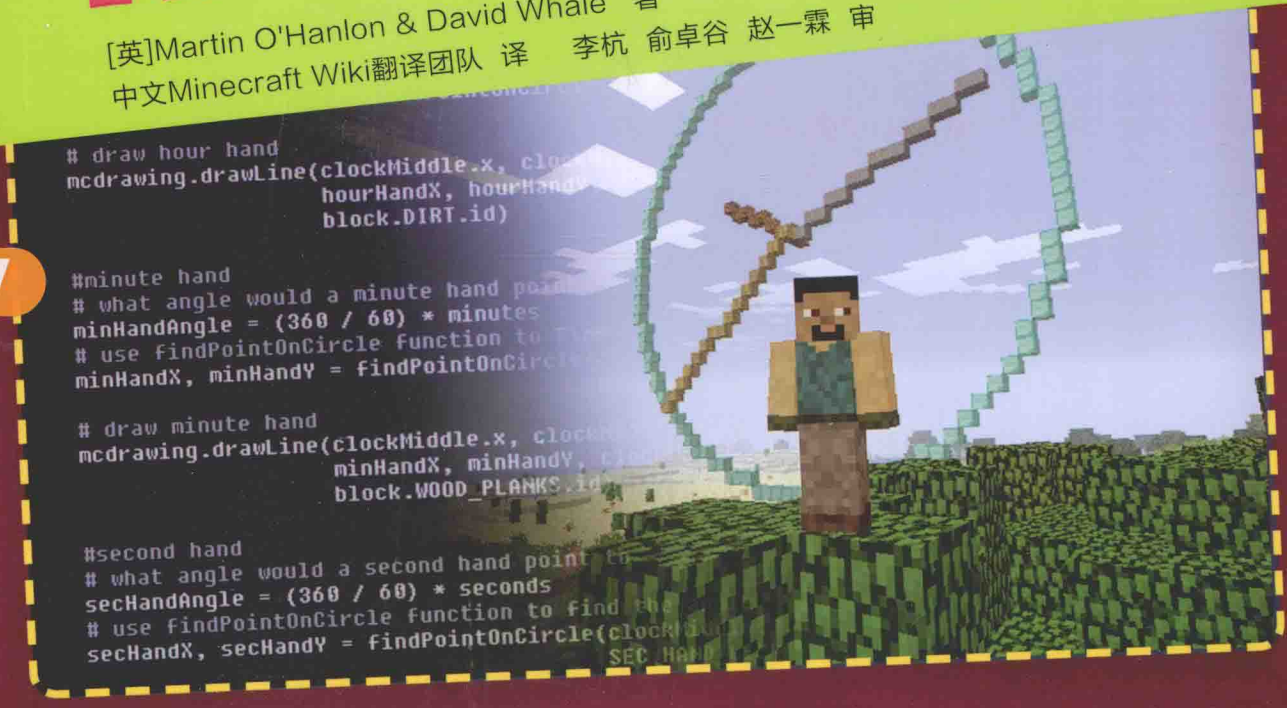
I'M MAKER 创客 WILEY

初学Minecraft®编程就上手
最有趣的Minecraft®编程学习入门书
包含为游戏爱好者特别设计的9个超棒实例

零基础学 Adventures in Minecraft®

Minecraft®编程（图文版）

[英]Martin O'Hanlon & David Whale 著
中文Minecraft Wiki翻译团队 译　李杭 俞卓谷 赵一霖 审



中国工信出版集团　人民邮电出版社 POSTS & TELECOM PRESS

最有趣的Minecraft®编程学习入门书！

在你体验Minecraft冒险的同时，学习宝贵的编程技能！

如果你很喜欢玩Minecraft，却被游戏中的建造耗费大量时间而困扰，并且你想要对游戏添加一些改动，那么本书就是为你而设计的。在游戏中，你可以学习许多Python编程技能，在PC、Mac或树莓派上与游戏进行互动。这些冒险不仅局限在虚拟世界——你也将会学习如何将Minecraft与电子元件连接起来，这样你的Minecraft世界就能够感知和控制真实世界中的物品。通过本书你将学到许多编程技能，将来在其他设备和学习项目中都必不可少，你的Minecraft世界最终会是这个星球上独一无二的！

本书作者Martin O'Hanlon & David Whale将手把手地教会你下面的知识：

- 在你的PC、Mac或树莓派上，使用Python编写Minecraft程序
- 设计房屋结构及制作3D打印机
- 使用简单电路搭建个性化游戏控制面板
- 建造智能物件并编写外星入侵程序
- 建造巨型的2D和3D建筑，如球体或金字塔
- 设计并编写一个完整的交互性竞技游戏

作者：Martin O'Hanlon，一个深度极客和业余滑雪爱好者，对编程及帮助他人学习编程有极大的热情。他经常为程序员、教师和年轻人举办关于Minecraft编程的演讲及讲习班，激励他们尝试新鲜事物，同时也提升编程的乐趣。

David Whale，非常喜欢编写计算机软件及帮助他人学习编程，他举办周末的计算机俱乐部，在那里孩子们可以学习Minecraft编程和树莓派等。David是*Adventures in Raspberry Pi*（《零基础学Raspberry Pi（图文版）》）一书的技术编辑。

译者：中文Minecraft Wiki翻译团队，访问http://minecraft-zh.gamepedia.com了解更多。

本书含配套网站，提供所有代码文件、参考表、词汇表，额外的项目实例，以及可供收集的成就徽章。书中每个冒险项目都附有在线视频内容。访问www.wiley.com/go/adventuresinminecraft来下载启动工具包和项目代码。

扫描二维码，访问Minecraft我的世界中文论坛，查看中文版相关在线内容。

本书特别感谢 MINECRAFT 我的世界中文论坛

封面设计：金蕊

分类建议：计算机与互联网/电子技术/编程语言与程序设计
人民邮电出版社网址：www.ptpress.com.cn

ISBN 978-7-115-39897-0

ISBN 978-7-115-39897-0
定价:79.00 元

零基础学

Minecraft® 编程（图文版）

[英]Martin O'Hanlon & David Whale 著

中文Minecraft Wiki翻译团队 译

李杭 俞卓谷 赵一霖 审

人民邮电出版社

北京

图书在版编目（CIP）数据

零基础学Minecraft编程 : 图文版 / (英) 惠尔(Whale,D.), (英) 汉隆 (Hanlon,M.O.) 著 ; 中文Minecraft Wiki翻译团队译. -- 北京 : 人民邮电出版社, 2015.9
(i创客)
ISBN 978-7-115-39897-0

Ⅰ. ①零… Ⅱ. ①惠… ②汉… ③中… Ⅲ. ①软件工具－程序设计 Ⅳ. ①TP311.56

中国版本图书馆CIP数据核字(2015)第172502号

版权声明

内 容 提 要

本书是非常有趣的Minecraft编程入门学习书。在Minecraft游戏中，你可以学习许多Python编程技能，在PC、Mac或树莓派上与游戏进行互动，更可将Minecraft与电子元件连接起来，使你的Minecraft世界能够感知和控制真实世界中的物品。本书适合所有对Minecraft游戏感兴趣的人，以及游戏开发者、Python编程初学者。

◆ 著 [英] David Whale Martin O'Hanlon
译 中文 Minecraft Wiki 翻译团队
责任编辑 李 健
执行编辑 周 璇
责任印制 周昇亮
◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路 11 号
邮编 100164 电子邮件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京市雅迪彩色印刷有限公司印刷
◆ 开本：800×1000 1/16
印张：15.5 2015 年 9 月第 1 版
字数：422 千字 2015 年 9 月北京第 1 次印刷
著作权合同登记号 图字：01-2015-0753 号

定价：79.00 元

读者服务热线：(010)81055339 印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京崇工商广字第 0021 号

献给我的妻子 Leonie，没有你的帮助，我不可能会走到现在。

——Martin

献给我的妻子 Gail，感谢你能忍受我没完没了地玩 Minecraft。

——David

出版致谢

以下人员参与了出版发行的工作：

编辑人员

社长：Barry Pruett
副社长：Jim Minatel
特约执行编辑：Craig Smith
采编：Aaron Black
项目编辑：Sydney Argenta
文字编辑：Grace Fairley
技术编辑：Cliff O'Reilly
编辑经理：Mary Beth Wakefield
项目编辑：Sara Shlaer
编辑助理：Jessie Phelps
插画师：Sarah Wright

市场人员

市场经理：Lorna Mein
市场助理：Polly Thomas

Minecraft 顾问

Zachary Igielman
Lauren Trussler
Sam Whale
Ben Foden
Ben Ramachandra
Ria Parish

关于作者

MARTIN O'HANLON 在整个成年后的时间里，一直都在设计和编写计算机系统。他对于编程及帮助他人学习的热情，指引着他创建了名为 <Stuff about="code" /> 的博客（www.stuffaboutcode.com），在博客里分享了他的经验、技能和想法。Martin 经常向程序员、教师和年轻人发表关于 Minecraft 编程的演讲，并举办相关讲习班，激励他们尝试新鲜事物，同时提升编程的乐趣。

DAVID WHALE 在为一些你无法预想其中包含计算机的设备编写程序。当他在还是一个 11 岁的在校学生时就对计算机产生了浓厚的兴趣，现在他依然非常喜欢编写计算机软件以及帮助他人学习编程。他在艾塞克斯郡经营软件咨询相关业务，但也经常性地志愿参加工程与技术协会 (The IET) 的各类校园活动，如举办周末的计算机俱乐部，评判校内竞赛，以及在全英国的社区活动中为年轻人开办编程讲习班。你可以从他的博客（blog.whaleygeek.co.uk）中了解到他的更多经历。

致谢

许多人都参与到这本书的诞生过程中，在这片有限的空间里我想要说的实在太多。我们都希望将这份特别的感谢献给下面的这些人。

- Mojang 的员工，感谢他们设计了如此美妙的游戏，感谢你们在游戏制作时将其设计为可编程的，在这点上，你们展现了天才和远见。没有这份远见，本书永远不可能诞生。
- 树莓派基金会和开源社区，没有它们就不会有树莓派或是 Bukkit 服务端，这两者都是这本书所面向的广大受众的重要平台。
- 我们的测试人员以及年轻的 Minecraft 专家——Zachary Igielman、Lauren Trussler、Sam Whale、Ben Foden、Ria Parish 感谢他们测试了我们的程序并提供了极其有用的反馈，没有他们的帮助，我们就不能了解到这本书是否适合于目标年龄的读者。
- S.K.Pang 先生，感谢他的建议为我们的项目选择合适的电子元件，以及为我们从 PC 或 Mac 上轻松便捷地控制这些电子线路时提供的帮助。
- Cliff O'Reilly，感谢他保证了本书在技术上的正确无误，感谢他为了我们整整进行了 3 遍测试（分别在 3 种不同的计算机平台上）。
- Sarah Wright, 感谢她为本书所画的美丽插画。这些插画都是赏心悦目的视觉艺术，极佳地表现出了每一个冒险项目的核心。
- Ben Ramachandra，在伦敦帝国学院 2013 圣诞节的 Fire Tech Camp 活动上的小伙子。他决意要在 Minecraft 里应用 Python，正是那个时刻促使我们编写本书的想法闪现并成为现实。
- Roma Agrawal，英国碎片大厦的结构工程师，感谢她的建议和发来的链接，这激发了我们在冒险 4 及附加冒险中添加高耸建筑的想法——让我们期待来自读者们的创意！
- 最后，但是相当重要地，我们想要感谢 Carrie Anne Philbin，感谢她的眼光和毅力来完成她的第一本书——*Adventures in Raspberry Pi*（《零基础学 Raspberry Pi（图文版）》），没有她的这本书，冒险系列丛书就不复存在——现在，看看你给我们带来了一个怎样的开始，Carrie-Anne？

译者序

《Minecraft》（我的世界）诞生于 2009 年的瑞典。自诞生之日起，由于其超越以往所有游戏的自由度与创造性，以及对于社区创意作品的开放态度，短短几年时间里便在欧美国家形成了庞大的玩家团体，完成了 PC/Mac、Xbox、PlayStation、Android、iOS、Windows Phone 与树莓派的全平台覆盖，并在各个平台上创造了一个又一个的销售神话。2010 年，这股风潮也传到了中国。以 Minecraft 中文论坛为代表的玩家社区先后建立，在新老玩家的共同推动下，以服务器、模组与玩家创意作品为核心的国内的 Minecraft 相关产业也达到了相当的规模。

本书是一本定位不同于通常的游戏攻略类书籍的作品，并不是 Minecraft 游戏教程或游戏技巧讲解，而更像一本编程与数电教程，注重寓教于乐，将诸多由易到难的 Python 编程实例经过精心设计，体现于一个又一个的 Minecraft 世界的“冒险”中。本书同时涉及程序设计与数字电路，跨 PC、Mac 与树莓派三个平台，也正是得益于 Minecraft 的开放性。

2015 年 1 月，我收到了人民邮电出版社的合作意向，并以中文 Minecraft Wiki 的名义公开招募翻译团队成员。完成对译者的遴选后，我们在 2015 年初春伊始完成了全书的初翻、统稿与初校。参与翻译的译者均非专业翻译人员，但都具备相当程度的 Minecraft 游戏经验，在计算机科学与电子学等专业领域也有所涉猎，部分成员在 Minecraft Wiki 也具备编辑经验。由于团体的业余性，我们虽然可以在专业领域力保准确，但具体到“信达雅”的翻译技巧就难免生疏。对此我们向为本书最终出版付出汗水的人民邮电出版社的编辑们表示感谢。

参与本书翻译与校对工作的有（排名不分先后）：李润恒、刘彦良、周天澜、朱伊琳、阮威、黄一天、徐正一、夏晏。在此感谢所有辛勤付出的翻译人员，因为有你们本书才能呈现于国内读者面前。

本书是第一本在国内市场出版的外国 Minecraft 相关译著。我们期待着本书能够使得更多的青少年喜欢上编程，培养良好的 Minecraft 游戏环境，也希望 Minecraft 在被微软公司收购后能够有更广阔的发展前景。

李杭
与中文 Minecraft Wiki 翻译团队
2015 年 6 月于上海

前言

那是又一个忙碌的校内 IT 午间聚会，我总是很乐意放弃我的午间休息，来给赶作业、搜索资料、给教师发邮件和打印作业的学生们一个机会。当我走过一排又一排计算机时，我看到了正在进行中的“艰苦工作”：五颜六色的小鸟被弹射到高塔上，外星人被射杀直至投降，一个学生在停车场尝试倒车——至少其中之一应该会有教育意义。

我的人生哲学告诉我，孩子们在计算机上花费越多的时间，计算机对于他们而言就越自然，甚至游戏也会提升手眼协调以及对于键盘的熟悉度。他们习惯于登录，打开浏览器，并寻找到搜索引擎上关键术语的最短数字。他们能在我打开门让 IT 实验室敞开的这段时间里，迅速进入房间并登录游戏。

当我在实验室的走道中巡查时，总是渴望着在学生之中发现一个在做有实际产出事情的学生，后来，我发现一个学生在游戏窗口中，兢兢业业地在搭一种看上去像是 3D 积木的东西。这个学生看上去深深地被吸引着，而当我驻足，好奇为什么这种相对简单的努力却引起了这个孩子的专注，我才发现，一座房屋在那一堆方块中拔地而起。

这是我了解 Minecraft 的过程：一个正在建造中的小小建筑师，一个在午餐时间置身于 IT 实验室喧嚣之外的学生，踏入了他自己所创造的世界之中——一座由他自己设计、创造并探索的虚拟房屋。孩子们的游戏是极具传播性的小事，一旦传播到一定的规模，就足够实现一种想法，他们就都会去参与。这引发了一种趋势，剩下的人就需要参与其中来跟上其他人，等到了夏天的时候，孩子们就都在玩 Minecraft 了。

然而，与过去那些流行不同的是，我很快发现这个游戏有很多独特之处。我发现学生们能够自主地进行建造。通过对房屋进行排线，他们可以创建他们自己的照明系统以及电梯，也可以设计隐藏在墙壁内部的楼梯。实际上，他们可以建造出任何能够想象到的创意发明。历史课延续到了午间俱乐部里，因为在游戏中他们建造了在历史课堂上刚学习过的城堡，生物课的学习则让他们明白这是在竞争中打造最好的骷髅，而地理课则会让他们了解，当游戏中的模拟火山喷发时，遍地的岩浆会让过去所有的艰辛劳作毁于一旦。

当我们学校购置了 MinecraftEdu 账号后，我的学生们便可以协力完成工作了。他们制作了巨大无比的学校徽标，还设计了一个迷宫并举行了一次迷宫逃脱的校园竞赛。他们测量了教室的尺寸，并在 Minecraft 中建造了校园模型。

更重要的是，这给我提供了一个虚拟的教室。当我的角色从学生们的角色头顶飞过时，教导他们关于逻辑的细节知识，这还是件诡异的事。我使用这个世界的方块，从“异或门”开始讲起，之后还给他们分配了一次团队工作，通过排布电路来建造一个计算器。

让我高兴的是，我后来了解到 Minecraft 在树莓派上同样是可用的。并且更重要的是，它可以使用 Python 语言进行编程。于是，我的学生们立刻开始行动了。通过使用 for 循环和放置方块，我们建造了站台、蹦床、火球和拉力赛——这是一种游戏中的游戏，甚至游戏的初始设计者也未必预料过，他们创造了属于他们自己的世界。

我的一个学生，基于他自己的想法，创造了一个自动建造房屋的脚本，可以让用户在身边自动

生成一个家；另外一个学生创造了一个能够自动清除的拼图。编写程序突然成为了一种他们迫切渴望去学习的事情，因为他们都受到了强烈的愿望的驱动——他们希望在 Minecraft 世界内建造物件，让更大更复杂的工程自动化进行，以及追赶并超越他们的伙伴们的新发明。

现在，我们学校搭建了一个以目的为导向的 Minecraft 网络服务器，由学生建造，为学生服务。通过这个网络，以和作业评估完全相同的方式，他们可以上交他们的作品来得到评分。他们可以相互竞争来编写更大的世界中独立的部分，并在之后一起探索这个世界。

这本书是踏进这个不可思议的世界的第一步。它会教你编程，使用这个星球上最流行的游戏之一来教你编程！孩子们会逐渐从 Minecraft 世界中脱离出来，从而踏入计算机编程的世界。他们拥有对于学习编程的真正目标，那就是他们渴望着比他们的朋友更快地学习如何将一些想法变成现实。

正是因为他们所写的代码的效果是即时可视的，这才使它的效果如此强烈。编程对于学生们而言可能是枯燥乏味的，因为当代码的结果只是以屏幕上的文本形式呈现时，他们便不能将之与代码的效果联系到一起。Minecraft 能够让他们以一种能够理解和欣赏的方式，看到这些代码的效果。

我相信，Minecraft 具备影响整个时代的力量，鼓励为了编程所应该成为的那样——有趣并非常值得去编程。所以，不要浪费任何时间，在 Martin 和 David 的指导下，开始你自己的 Minecraft 世界的冒险之旅吧。保持创造力、保持兴奋感，建造一些绝妙的东西，并在旅途中学习如何编写程序！

Ben Smith BSc (Hons)
Head of Computing,
ArnoldKeqms, Lytham St. Annes.

目录

CONTENT

冒险 2
追踪玩家移动 29

冒险 3
建筑自动化 51

冒险 4 与方块交互 75

冒险 5 与电路互动 97

概 述

你是一个冒险家吗？你喜欢尝试新事物或学习新技能吗？你是 Minecraft 的忠实粉丝吗？你想通过计算机编程与游戏互动，来拓宽你在 Minecraft 中的能力，让你的朋友惊异于你的创造力与“魔法”吗？如果答案为“是！”，那么此书就是为你准备的。

Minecraft 是什么

Minecraft 是一个“独立沙盒游戏”（sandbox indie game），你可以在其中建造建筑、收集物品、挖掘矿物，以及为了生存而与怪物战斗。游戏展现给你一个由不同方块组成的 3D 虚拟世界，在虚拟世界中，每一个方块都有它自己在网格里的位置。图 1 所示为一个 Minecraft 世界的例子。

图 1　Minecraft 的世界

虚拟世界

在一个沙盒游戏（这是一个十分宽广的沙盒，就像是填满沙子的游戏区域）中，你是一个在虚拟世界里的玩家。与预先设置好关卡的游戏不同，你可以在虚拟世界中漫游，自行选择你想完成什么目标，并且自己决定如何完成。因为你从一开始就自己决定来做什么，所以沙盒游戏有近乎无限的可能性。你创造你自己的故事并在 3D 世界中漫游，还可以通过偶然及实验学习新技能与新功能。

在 Minecraft 中，你的玩家——也可以称为化身，叫作“Steve”。你引导 Steve 在沙盒虚拟世界中穿梭，来完成你所计划的目标。第一个晚上，如果你成功地在与怪物的战斗中存活下来，你可以遵循着你自己那令人着迷的任务来与其他的玩家进行交互，建造超大型的建筑物。只有想不到，没有做不到！

沙盒游戏允许你，即玩家，来自己决定如何玩游戏，而不是被迫去走游戏设计师设计的特定路径。你可以访问 http://zh.wikipedia.org/wiki/ 开放世界，了解更多关于这种游戏设计的信息。为什么玩家叫作 Steve？这是一个小秘密，但你可以访问 http://minecraft-zh.gamepedia.com/ 玩家了解更多。

Minecraft 是如何诞生的

独立游戏，全称叫“独立视频游戏”（independent video games），是由个人或小团队制作的。它们通常没有游戏发行商的任何资金或是支持。但由于它们的独立性质，独立游戏往往比主流的游戏更加有创新性。依据维基百科所述，Minecraft 的创始人是瑞典计算机程序员 Markus Persson，他在游戏内名为“Notch”。在 2009 年，Notch 展示了 Minecraft 的早期版本，而第一个官方正式版是在 2011 年发布的。Notch 创立了一家名为 Mojang AB 的瑞典公司，继续在不同平台上开发 Minecraft，这些平台包括 PC、Mac、树莓派、Linux、iOS、Android、Windows Phone、Xbox 与 Playstation。

你可以在一个名为《Minecraft：Mojang 故事》（http://en.wikipedia.org/wiki/Minecraft:_The_Story_of_Mojang）的纪录片中找到更多关于 Minecraft 的故事。

Minecraft 编程是什么

本书内容主要关于计算机编程——采用 Minecraft 游戏的方式来教你编程。如果你正在寻找关于如何建造建筑与战斗的技巧，附录 A 中列出的一些书可能会对你有帮助。

通过对 Minecraft 编程，你可以获得更加刺激、更富创造力且更具个性化的游戏体验。当你在正常地玩游戏时，你遵守着 Minecraft 游戏设计师制定的基本规则。但通过编写程序与 Minecraft 游戏世界交互，你可以使复杂与重复的任务——如建造大型的房子或建筑物，变得自动化。你可以使游戏和游戏中的物体有新的特性，甚至发明游戏创始人都没有想到过的新物品。但最重要的是，你可以学习一项通用的技能——如何使用 Pyhton 语言进行编程。在这之后，你就能够把它应用于各种各样的事情，而不仅仅是在 Minecraft 中。如图 2 所示为一个由简短的 Python 程序自动建造的巨大房屋。

在一个最近的关于为什么所有的孩子都要学习编程的视频中，引用了 Will.i.am 的话：“顶尖的程序员就是今日的摇滚巨星。”你在本书接下来的冒险中学到的新技能，将会让你的 Minecraft 经历更具个性化、更富创造力、更富雄心。你在编程时所使用的新“魔法”会让你的朋友和其他玩家感到惊讶，并使他们来询问你使用了何种“魔法”来实现这种惊人的壮举。答案是理所当然的——是计算机编程的魔力。

图 2　仅用 20 行的 Python 程序建造出的巨大房屋

这本书是给谁写的

本书是写给任何喜欢玩 Minecraft 且想要学习编程并用它来做更多的事情的青少年及爱好者的。本系列图书主要针对 11 ~ 15 岁的青少年读者，但书后面的一些更具挑战性的冒险可能适合年龄更大的读者。本书前面的章节也适合只有 8 岁的读者。

也许你已经是一名 Minecraft 的专家了，但却发现花费很长时间来建造大型建筑让人十分头疼。也许你会想要找到拓展游戏的方式，给游戏世界增加一些额外的智能化与自动化功能。无论你的理由是什么，本书始终会是你在 Minecraft 编程之旅中的指南。而每个冒险者都知道，指南永远是你背包中最重要的物品。你的长途旅行会从简单的项目开始，例如，在 Minecraft 的聊天区域发出一句话。通过在 Minecraft 中学习一些基本的 Python 语言，你可以探索如何使用新的计算机编程技能在 Minecraft 中制作出激动人心的游戏。在冒险的最后，你学到所有的技能将会使你成为一个 Minecraft 编程的先行者！

你将会学到什么

你将了解 Minecraft 的很多方面，以及如何通过 Python 语言来与 Minecraft 的功能进行交互。你将探索在 3D 世界中如何使用坐标来定位方块，如何获得玩家所处位置，如何在 Minecraft 世界中增删方块，以及如何判断方块是否已被玩家击中。

如果你使用的是树莓派，你会学习如何把 Minecraft 编程接口和 Minecraft 树莓派版游戏绑定在一起。如果你使用的是 PC 或 Mac，你将学习如何用社区开发的 CraftBukkit 服务端建立和运行自己的本地 Minecraft 服务端，并学习如何通过安装 RaspberryJuice 插件来使用和树莓派相同的编程接口。

你将学习如何使用 Python 编程语言编写程序，从最基础的“你好，Minecraft 世界”程序开始，到创建巨大的 3D 物体并与其互动。由于你学会的新的 Python 编程技能，你的游戏将会更加个性化。

使用免费的 MinecraftStuff 模块——预先编写的 Python helper 代码，你就能提高你的能力来使用方块、线、多边形和文本来创造 2 D 和 3 D 等对象。

你的冒险并不仅限于 Minecraft 的虚拟世界！我们将向你介绍将 Minecraft 连接到电子元件的方法，这意味着你的 Minecraft 世界能够感觉和控制现实世界中的对象。因此，我们给了你一个有价值的秘密，即如何打破虚拟沙盒世界的界限！

Minecraft 有两个主要的运行模式：生存和创造模式。在本书中，我们将使用创造模式。我们不会涉及生存模式（主要因为当你看着你的程序运行时，你被爬行者杀死是件令人很沮丧的事情）。市场上已经有很多好书解释如何在 Minecraft 中生存，我们在本书最后的附录 A 中给出相关链接及其他资源。然而，你在创造模式里写的程序也是可以在生存模式里运行的。

我们认为你应该知道的

因为这本书是关于在 Minecraft 中编程的，并且我们想让你主要关注编程，所以有些东西是你必须要知道的。

1. 你有一台计算机（指装有 Raspbian 的树莓派、装有微软 Windows 的 PC 或装有 Mac OS X 的 Mac），能达到运行 Minecraft 的最低标准，已经被配置好并能工作。
2. 你知道如何基本地进行计算机操作，使用系统菜单来启动程序，并使用应用菜单比如文件 ⇨ 新建 ⇨ 保存。
3. 你的计算机连接到了互联网，并且你可以用网页浏览器从网上下载文件。
4. 如果你使用的是 PC 或 Mac，你需要一个 Minecraft 账号并已经在计算机上安装了 Minecraft。
5. 你知道如何玩 Minecraft，比如如何启动游戏、如何到处移动、如何在物品栏中选择物品，以及如何放置与破坏方块。

因为本书是关于编写 Minecraft 中程序的，我们并不需要你有编程的知识。在你进行冒险的同时，我们会教你如何编程。

在项目中你需要

这本书适合现在的 3 种常见的平台：装有 Raspbian 的树莓派、装有微软 Windows 的 PC 或装有 Mac OS X 的 Mac。Minecraft 支持其他一些平台，比如 Linux，但我们不介绍在这些平台下的设置。

为了使各部分的安装更简单，我们准备了三个初学者工具包，每个支持的平台各有一个。你可以从配套资源网站上下载初学者工具包，并且我们在冒险 1 中提供了一步一步的操作指南，包含如何下载与安装这些工具包并让它们工作。初学者工具包提供了所有你需要的东西，除了 Minecraft 游戏本身。你可以在任何时间启动并运行它们！

当你下载初学者工具包时，你将需要一个互联网连接。几乎所有你在冒险中需要的东西都包括在初学者工具包中。一些冒险有特殊的需求，所以我们会在冒险的开头的时候提醒你，你就可以在开始前把一切

准备好。

在冒险 5 和 9 中，我们向你展示如何连接小型电子电路使 Minecraft 虚拟世界与现实世界连接在一起。为此你需要买一些小的电子元件，你可以在大多数的电子元件零售商买到（在附录 A 我们提供了一些链接）。

树莓派有内建的输入 / 输出引脚，所以你可以把你的电路直接连接到这些引脚上。因为 PC 和 Mac 并不包含输入 / 输出引脚，我们选择了一个你能购买的小插件板，它可以用 USB 和你的计算机连接。同样，附录 A 中有链接，你可以买这个。

在旅行中最重要的东西是你自己对 Minecraft 的兴趣、热情和一些好奇心，并愿意尝试自己的想法来拓宽你的视野！

给家长和老师的提示

我们已经把这本书分成几个独立的冒险，你可以当作一个个独立的项目，其中每个项目都关注 Minecraft 编程的一个具体的功能。在每个冒险中，我们逐渐介绍 Python 语言。早期冒险主要针对初学者，后来冒险变得更具挑战性并引入了更多的 Python 知识，用来更多地扩展读者的能力。

每个冒险都展示了一个实用的项目，以描述性的风格提供一步一步的指示（读者可以跟着它们完成），非常像一个注释得很好的程序清单。学生可以稍后阅读出现在“深入代码”栏目的详细说明，这意味着他们把精力集中在输入代码和调试程序上，不会偏离主要的进展。

每个冒险可能需要多个课时才能完成，但它们都被分成了很多节，每小节在逻辑上独立，可以作为单独一节课提供一个目标，或者在不同的章节中可以作为通用的一个知识点。

Python 语言使用程序左边的缩进表示代码结构，并且它是区分大小写的语言。成年人需要对年轻读者进行提示，以确保他们正确使用大小写和缩进，从而避免将错误引入到他们的项目的可能性。所有的项目可以从配套资源网站上下载，所以如果你有缩进问题，你可以检查我们的版本的程序，看你在哪里出了差错。

很多学校已经安装了 Python 版本 3。在撰写本文时，Mojang 没有发布使用 Python 版本 3 的 Minecraft 的编程接口，所以就像在冒险 1 中解释的那样，你应该使用 Python 版本 2。

这本书的组织结构是怎样的

这本书的每一章都是一个独立的冒险，在你编程与测试项目时会教你新技能和概念。这本书以每个冒险是一个独立的项目的方式来组织，但你会发现按照顺序工作会更加容易，这会在整本书中逐步建立你对编程概念的理解。

冒险 1 是在你做其他事情前至关重要的。这是因为它向你展示了如何下载和安装你需要的一切，并检查所有东西是否工作正常。在这个冒险中，我们介绍一些你需要知道的在其他所有冒险中的基本步骤，但在前面的冒险中，在你开始时也会有些提醒。

前三个冒险是写给很少或根本没有编程知识的初学者的。我们会解释所有的术语和概念使你能解决它们。在冒险 2、3、4 中，你会学习如何修改 Minecraft 游戏的核心部分。这包含探测 Minecraft 世界中发生了什么，使用简单的数学计算一些东西，并使你的程序的行为会有所不同，例如在聊天窗口显示一条消息或自动创建方块。你将通过本书中的探测、计算和行为这三个概念来制造大的和激动人心的 Minecraft

程序！

冒险 5、6 是建立在前面的基础上，并探索一些把 Minecraft 世界和显示世界连接起来的方法。你将实践一些搭建小型电路来进行实体交互的激动人心的主题，它能控制在 Minecraft 中发生的一些事情，并且会响应发生在 Minecraft 中的事。它们可以以此为基础创建丰富的、令人兴奋的想法和游戏！冒险 6 寻找一种方法，可以从数据文件中读取大量的数据并用一个 3D“复制机”复制大型的建筑。

冒险 7、8 介绍自由的 MinecraftStuff 模块，这使得建造线、圆和其他 2D 形状，以及一些惊人的 3D 球体和金字塔变为可能。这可以成为那些很难手工建造的建筑的雏形。冒险 8 介绍了如何给移动的物体以个性来使它们有智能。使用这些技术，你可以编写一些激动人心的“游戏内的游戏”，让你的朋友啧啧称奇。

冒险 9 利用所有早些时候冒险时学到的编程概念和技能，创建一个最后的大项目—— 一个特别神奇的游戏。这个游戏有积分，有移动的物体，物体分为你必须躲开的或随身携带的两种。在这个冒险中，你也可以选择通过电子元件来搭建物理组件，允许你在现实世界中按按钮来操控游戏。

附录 A（“接下来去哪？”）显示一系列的资源，你可以用来扩展和增强你的冒险，学习更多关于 Python 编程和根据你在这本书中所学到的东西，创建更神奇的 Minecraft 程序。

附录 B（“快速参考”）有一个关于在书中使用的编程特性的全面参考指南，以及在 Minecraft 中的特定的编程语句，还有一个你可以建造的方块类型表。你会发现对你自己的项目和发明，这将是一个宝贵的参考部分！

术语表提供了一个我们在书中介绍的所有术语和术语的方便快速的参考，并且集合了所有冒险中的定义。

本书的配套资源网站

在这本书中，你可以找到很多对配套资源网站 www.wiley.com/go/adventuresinminecraft 的引用。在这个网址中，你可以找到初学者工具包用来开始对 Minecraft 编程，如果你在某个地方卡住了，也有一系列的视频教程来帮助你。大一点的工程的代码文件也可以在网页上找到。

你还会发现配套资源网站上一个完整的额外奖励冒险！在这个冒险里，你将在 Minecraft 里建造一个功能齐全的电梯，并能够乘坐它上下驰骋在你的世界中。这个额外奖励冒险非常具有挑战性，你需要运用所有你学过的技能，包括用电子电路控制电梯。

本书在配套资源网站上提供一个你可以下载的 PDF 格式的参考附录。把它放在你的旁边作为在 Minecraft 中冒险的参考。你也可以作为参考，把它用在未来的任何编程项目中。Wiley 网站还包括一个术语表。虽然定义都已经在冒险中包含了，但如果你想要查一个单词，你可以上网。

其他的帮助来源

计算机是非常复杂的机器，而操作系统和软件总是在变换。为了使你和你的冒险尽量避免因为未来的软件升级而造成的不必要的麻烦，我们在冒险 1 中提供了你可以下载的初学者工具包。这里面提供了你所需要的大多数东西。不过，如果你遇到了问题或者是需要特别的帮助，这里有一些你可以去的网站：

注册用户 ID 并下载、安装 Minecraft：http://minecraft.net

如何玩 Minecraft：http://minecraft.gamepedia.com/Minecraft_Wiki

http://minecraft-zh.gamepedia.com/Minecraft_Wiki（中文 Wiki 网站）

树莓派官网：www.raspberrypi.org
微软 Windows 操作系统：http://support.microsoft.com
Apple Mac 和 Mac OS X：www.apple.com/support
Python 语言官网：www.python.org
IDLE 集成开发环境：https://docs.python.org/2/library/idle.html
Minecraft 树莓派版：http://pi.minecraft.net
CraftBukkit 服务端：http://wiki.bukkit.org
Raspberry juice bukkit 插件：http://dev.bukkit.org/bukkit-plugins/raspberryjuice
（由于受美国数字千年版权法案影响，Craftbukkit 官方网站目前已被迫停止下载服务。我们提供的初学者工具包依然包含 Craftbukkit，或读者也可以在 www.spigotmc.org 找到与 Craftbukkit 理念类似的服务端 Spigot——译者注）

一些约定

你会注意到书中有一些特殊的栏目，它们会帮助并指引你。如下列举了一些：

这个版块会给你介绍一些你不熟悉的概念。

这个版块会提供使你的计算机编程经历更加容易的提示。

这个版块包含一些重要的警告，在一些操作的时候保护你和你的计算机的安全。

这个版块会快速地测试你的理解是否正确或让你更好地理解一个主题。

这个版块作者会解释东西或讲解一些有用的知识。

这个版块指引你观看支持页面上的视频，这些视频会指导你一步一步完成任务。

你还会发现在书中有两种栏目。如果你想要项目走得更远一点，“挑战”栏目给你可以接受的额外任务，也许你可以通过更改或添加新功能来完成它们。“深入代码”栏目会解释代码中的细节或者新功能，让你更好地理解 Python 这门语言。这些侧边栏意味着你可以先在代码上集中精力，然后再来扩展它们的功能，并学习它们是如何工作的。

当你遵循我们的步骤使用代码或指令时，你输入的代码应该与书上的代码完全一致。在Python语言中，每一行开头的空白（缩进）对于代码的意义是很重要的，所以你要额外注意你是否在每行开头输入了足够的空格。我们的代码版块是彩色的，让你更容易看到每一行需要缩进多少。不要过于担心，我们会在早期冒险中你第一次需要使用它时向你解释。

有时你会需要输入很长很长的代码，长到本书的一行不足以放下它们。当你在一行代码的末尾看到 ↩ 符号时，这意味着这行代码和它下面的一行代码是一起的，你需要把它们输在一行里，而不是单独的两行。比如下面的代码应该被输在一行里，而不是两行：

```
print("Welcome to Adventures in Minecraft by ↩
Martin O'Hanlon and David Whale")
```

如果你在电子阅读器上看这本书，为了确保你输入程序的准确性，请把你的阅读器字体设置得小一些。这样，程序就不会在页面边缘产生不必要的换行，带入你的程序中导致错误。

大多数冒险的最后均包含一个快速参考表，表内总结了这个冒险涉及的主要编程语句或概念。当你需要参考时，你可以参考这些指南。附录 B 中也有一个参考部分，它会告诉你在 Minecraft 和 Python 中的重要的编程语句。我们希望你会发现身处冒险时这些指南的便利性。

当你完成一个冒险，你可以解锁一个成就并收集新的徽章。你可以在 Minecraft 大冒险的配套资源网站收集徽章来代表这些成就（www.wiley.com/go/adventuresinminecraft）。

与作者接触

在附录 A 中你会找到一些拓展你的 Minecraft 编程知识的方法，其中有一系列的网址、组织、视频和其他的资源。很多的资源包含论坛，这些论坛可以让你问问题或与其他人交流关于 Minecraft 编程的事。

你可以登录我们的网站，通过发信息来与作者沟通：

Martin：www.stuffaboutcode.com

David：http://blog.whaleygeek.co.uk

是时候开始你的冒险了！

冒险 1

你好，Minecraft 世界

在本书中，你将学习如何通过编写程序来与 Minecraft 世界互动，并做一些令人非常兴奋的事情。你将使用名为 Python 的语言来完成这件事。用 Python 来控制 Minecraft 世界的方法原本是在特别为树莓派设计的《Minecraft: 树莓派版》上创立的，如果你没有树莓派，但你有 Windows 或 Apple Mac，也没有问题——只需要在开始前做一点额外的设置工作。我们会一步步教你。

> Python 是在本书中使用的编程语言。
>
> 
>

本书中准备了各种冒险来教你如何为 Minecraft 编写程序。这里充满了各种你可以用来招待你的朋友的事情，让游戏变得更加好玩。你会发现一些很炫的方式来让你的玩家到处走动，当过了不久之后，你会发现你已经可以很容易地建造从未为人所知的城市或 Minecraft 作品。

与 Python 编程语言配合工作的还有名为 IDLE 的代码编辑器，你会用它来建立、编辑并运行你在冒险中创建的程序。

Python 语言在全世界被用于商业和教育事业。它非常强大，并且易学。你可以在 www.python.org 找到更多有关 Python 的信息。

当编程员学习一种新的编程语言，或者一种做事的新方法时，他们总是以一个“hello world”程序开始。在屏幕上显示“hello world”是非常简单的程序，这可以确保所有东西安装完毕，且工作正常。

在第一个冒险中，你将设置你的计算机以允许你编写一个可以在 Minecraft 聊天窗口显示“Hello Minecraft World”的程序（见图 1-1）。

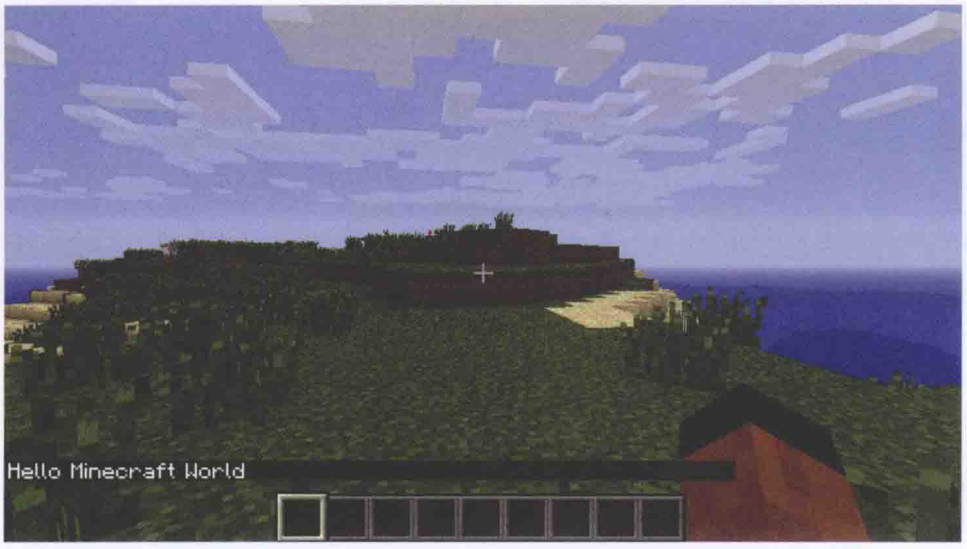


图 1-1　Hello Minecraft World

在本书中，要为 Minecraft 编程，你需要有以下三种计算机之一：一台装有 Raspbian 的树莓派、装有微软 Windows 的 PC 或装有 Mac OS X 的 Mac。虽然在不同平台上的设置方法有所不同，但在你设置完毕之后，对 Minecraft 编程的方式是完全一样的。为了让设置更加方便，你可以在本书的配套资源网站下载初学者工具包（www.wiley.com/go/adventuresinminecraft）。初学者工具包已经被仔细测试过，确保本书中的所有冒险都能正常运行。你会看到初学者工具包包含一个名为 README 的文件，你必须仔细阅读它。它描述了这个初学者工具包包含的内容以及建立过程。你可以借助这些信息从零开始设置你的计算机，但我们并不推荐这么做。本书中的指导内容更为丰富。

正确设置你的计算机十分重要，否则你会陷入困境。所以你要确保非常仔细地跟着指导来设置。

在树莓派上开始你的编程之旅

如果你正在使用树莓派，在你开始在 Minecraft 上编写你的第一个程序之前有以下两步要做。

1. 下载并安装《Minecraft：树莓派版》——专门为树莓派设计的 Minecraft 版本。
2. 下载并解压树莓派初学者工具包。在文件夹 MyAdventure 中，包含了所有为了完成本书中的冒险而需要的东西，这里也是你需要把你的 Minecraft 程序保存的地方。

访问我们配套资源网站 www.wiley.com/go/adventuresinminecraft 观看视频来了解更多关于设置树莓派的方法。

本书中使用树莓派的图形用户界面（GUI），也被叫作 X windows。GUI 被安装在树莓派上，但这依赖于你如何设置树莓派，当系统引导时并不一定会加载 GUI。你可能以一个登录和命令提示符开始。

如果你的树莓派以命令提示符启动，你需要登录并在命令提示符出现时输入 `startx`，按下回车键来加载 GUI。

请使用刚刚安装好的 Raspbian 来开始 Minecraft 冒险之旅，这样你可以保证树莓派已被正确设置。访问 www.raspberrypi.org/help/ 来获取关于树莓派与安装 Raspbian 的信息。

在树莓派上安装 Minecraft

当你的树莓派已经引导完毕并且 GUI 已经启动，你就可以安装《Minecraft：树莓派版》了。《Minecraft：树莓派版》的安装包可以从 http://pi.minecraft.net 下载，你也可以在那里获知从文件中提取游戏并运行的方法。

请参考下面的步骤下载、提取《Minecraft：树莓派版》。

1. 双击桌面上的 LXTerminal 图标（它看起来像一个黑色的计算机屏幕）。这样你就可以在打开的 LXTerminal 窗口中打字。
2. 把下面的指令输入 LXTerminal，一次一行，在输入每个指令后按回车键来下载《Minecraft：树莓派版》的安装文件：

```
cd ~
wget http://s3.amazonaws.com/assets.minecraft.net/pi↩
/minecraft-pi-0.1.1.tar.gz
```

屏幕上会显示下载的进度条。

3. 为了解压《Minecraft：树莓派版》，输入：

```
tar -zxvf minecraft-pi-0.1.1.tar.gz
```

《Minecraft：树莓派版》会被解压到名为 mcpi 的文件夹。

下面你需要下载树莓派初学者工具包并提取名为 `MyAdventures` 的文件夹。请参考下面的步骤。

1. 打开网页浏览器（比如：Midori），访问配套资源网站（www.wiley.com/go/adventuresinminecraft），并下载树莓派初学者工具包。

2. 一个询问你是要打开还是保存的窗口打开了。单击“打开”来启动名为 Xarchiver 的程序，它会提取文件。

3. 在 Xarchiver 菜单中单击 Action⇨Extract。

4. 在 Extract to：文本框中输入 /home/pi（见图 1-2）。

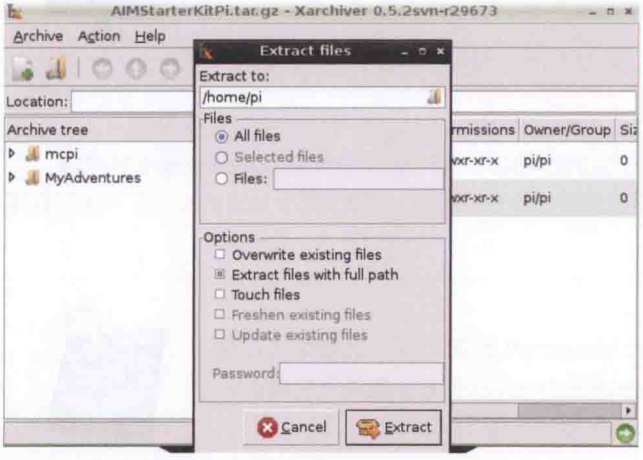


图 1-2　选择路径并解压树莓派初学者工具包

5. 单击 Extract。你的文件会被解压到文件夹 `/home/pi`。你可以在任何时间访问并提取它们。

在树莓派上启动 Minecraft

现在你已经下载并安装了《Minecraft：树莓派版》。运行游戏，并在开始编写你的第一个程序前到处走走。

在今后的冒险中，说明会告诉你何时启动 Minecraft。如果你要回顾如何启动 Minecraft，回顾本节。

跟着下面的步骤来启动 Minecraft。

1. 双击桌面上的 LXTerminal 图标（它看起来像一个黑色的计算机屏幕）。这样你就可以在打开的 LXTerminal 窗口中打字。

2. 输入下面的指令来启动 Minecraft（见图 1-3）：

```
cd ~
cd mcpi
./minecraft-pi
```

图 1-3　输入一些简单的指令来启动 Minecraft

3. 好了！现在你可以开始玩 Minecraft 了。单击 Start Game⇨Create New 来新建一个 Minecraft 世界。

主菜单里有两个选项："Start Game"来新建或进入一个已经存在的世界和"Join Game"来加入其他玩家的《Minecraft：树莓派版》的世界。

当你设置好你的树莓派，并且你的 Minecraft 已经在运行，你可以跳过下一个部分（除非你想在 PC 或 Mac 设置 Minecraft，就像在树莓派上一样）并直接到本章后面名为"创建程序"的章节。

在 Windows PC 或 Apple Mac 上开始你的编程之旅

无论你在用 Windows PC 还是 Mac，你都需要确保 Minecraft 已经在你的计算机上安装并正常工作。如果你没有 Minecraft，访问 www.minecraft.net 来购买游戏。如果在安装、运行或游戏中有任何困难，帮助就在身边——访问 https://help.mojang.com 即可。

为了用同种方式对在 Windows PC 或 Mac 上的完整版 Minecraft 进行编程，就像在树莓派上一样，你需要使用开源的 Minecraft 服务端——Bukkit（http://bukkit.org）和 RaspberryJuice 插件（http://dev.bukkit.org/ bukkit-plugins/raspberryjuice）。

Bukkit 是可以通过创建插件来让游戏变得不同的一个第三方 Minecraft 服务端程序。Bukkit 上的 RaspberryJuice 插件（plugin）允许你用和树莓派一样的方式通过编写程序改变 Minecraft。你将使用 Bukkit、RaspberryJuice 和 Python 编程语言来建立你的 Minecraft 程序。

Bukkit 是一个允许人们通过插件来更改游戏的第三方 Minecraft 服务端程序。（Bukkit 实质上指的是这种服务端程序使用的应用编程接口。准确地说，这个服务端的名称应当是 CraftBukkit ——译者注）

插件（plugin）是在 Bukkit 中运行，并允许你改变游戏本身的程序。

你需要下载 Python 编程语言并把它安装在你的计算机上。虽然最新的版本为 Python 版本 3，但在 Minecraft 大冒险中，你会使用 Python 版本 2，这是因为 Mojang 提供的 Python 库（用来与 Minecraft 建立连接）只能在版本 2 工作。本书中的所有程序已经被测试过，能够在 Python 版本 2.7.6 工作。虽然你并不一定使用这个版本，但你一定要使用 Python 版本 2。

访问 www.python.org 了解更多关于 Python 的知识。你可以在 www.python.org/download 下载 Python。Python Wiki 位于 wiki.python.org，这里包含了信息、教程和到 Python 配套资源网站的链接。

下面的步骤告诉你如何在 Windows PC 或 Apple Mac 上建立你的第一个 Minecraft 程序：

1. 下载并提取 PC 或 Apple Mac 初学者工具包，它包含了已经设置好的包含 RaspberryJuice 插件的 Bukkit 服务端和一个用来保存你的程序的名为 `MyAdventures` 的文件夹。
2. 下载并安装 Python 编程语言。
3. 把 Minecraft 设为和 Bukkit 相同的版本，并将其连接到 Bukkit。

访问我们配套资源网站 www.wiley.com/go/adventuresinminecraft 观看如何在 Windows PC 和 Apple Mac 上进行设置的视频。

在 Windows PC 上安装初学者工具包和 Python

如果你正在使用 Windows PC，你可以用下面的方法来设置你的计算机：

1. 下载 Windows PC 初学者工具包并解压到桌面上。
2. 下载并把 Python 编程语言安装到你的计算机上。

下载并解压初学者工具包

参考下面的步骤下载 Windows PC 初学者工具包并把它的内容放在桌面上。这样做可以在你需要的时候快速找到它们：

1. 打开 PC 的网页浏览器（例如：Internet Explorer、Chrome），访问配套资源网站（www.wiley.com/go/adventuresinminecraft），并下载 Windows PC 初学者工具包。

2. 当初学者工具包下载完成后，你可以通过直接打开文件或者打开它所在的文件夹并双击 `AIMStarterKitPC.zip` 来打开它。

3. Zip 文件只包含了一个文件夹，名为 `AdventuresInMinecraft`。单击这个文件夹，按住鼠标并把它拖到桌面上来将其复制到桌面（见图 1-4）。

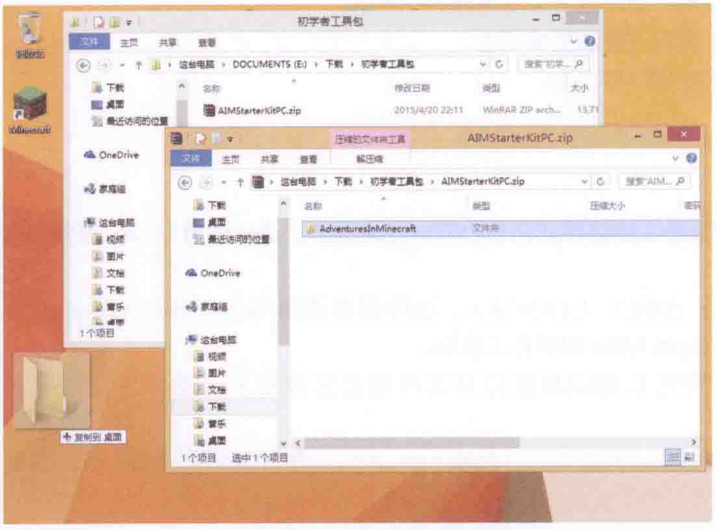


图 1-4　把 `AdventuresInMinecraft` 复制到桌面上以便于你能方便地找到它

下载并安装 Python

你将会使用Python编程，因此你现在需要参考下面的步骤安装Python编程语言和IDLE代码编辑器：

如果你没有计算机的管理员账户，你需要一个有管理员账户的人为你输入管理员密码来安装 Python。

1. 打开 PC 的网页浏览器（例如：Internet Explorer、Chrome），访问 www.python.org/download/releases/2.7.6 并单击 Windows x86 MSI Installer（2.7.6）链接来安装它。

2. 当 `python-2.7.6.msi` 下载完毕后，你可以通过直接打开文件或者打开它所在的文件夹并双击它来运行。

3. 这时可能会出现一个安全警告对话框：“你确定要运行此程序吗？”单击“运行”。

4. 单击 Next 来开始安装。

5. 你会被询问要把 Python 安装到哪里，你最好选择默认路径，单击 Next，把这个路径记下来以方便你在需要的时候找到 Python 文件。

6. 不要改变在“Customize Python”窗口中的设置，你只需单击 Next。
7. 用户账户控制可能在这时向你索取执行初始化程序的权限，单击“是”。
8. 等待程序完成安装并单击 Finish。

在 Apple Mac 上安装初学者工具包和 Python

如果你正在使用 Apple Mac，你可以用下面的方法来设置你的计算机：

1. 下载用于 Apple Mac 的初学者工具包并解压到桌面上。
2. 下载并把 Python 编程语言安装到你的 Mac 上。

下载并解压初学者工具包

参考下面的步骤下载 Apple Mac 初学者工具包并把它的内容放在桌面上。这样做可以在你需要的时候快速找到它们：

1. 打开 Mac 的网页浏览器（例如：Safari、Chrome），访问配套资源网站（www.wiley.com/go/adventuresinminecraft），并下载 Apple Mac 初学者工具包。

2. 当初学者工具包下载完成后，你可以通过直接打开文件或者打开它所在的文件夹并双击 `AIMStarterKitMac.zip` 来打开它。

3. Zip 文件只包含了一个文件夹，名为 `AdventuresInMinecraft`。单击这个文件夹，按住鼠标并把它拖到桌面上来将其复制到桌面（见图 1-5）。

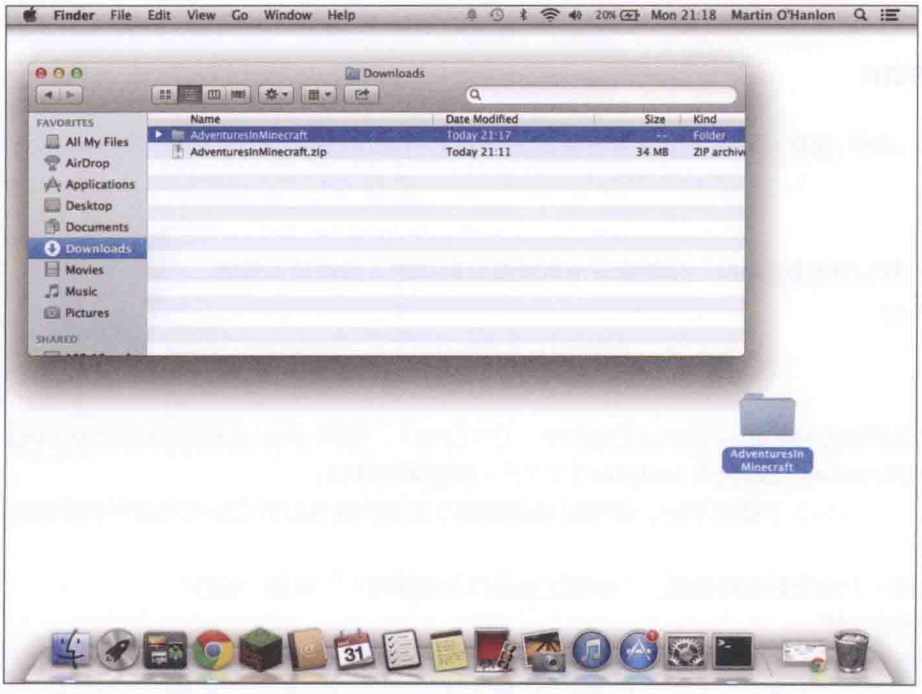


图 1-5 把 `AdventuresInMinecraft` 复制到桌面上

下载并安装 Python

你将会使用Python编程，因此你现在需要参考下面的步骤以安装Python编程语言和IDLE代码编辑器:

根据计算机的设置不同，你可能需要输入你的 Apple 密码，或让一个有管理员账户的人在你安装 Python 前输入密码。

1. 打开 Apple Mac 的浏览器访问 www.python.org/download/releases/2.7.6 并单击 Mac OS X 64-bit/32-bit x86-64/i386 Installer (2.7.6) for Mac OS X 10.6 and later 链接来下载它。
2. 当 `python-2.7.6-macosx10.6.dmg` 下载完毕后，单击它或在 Finder 里打开它下载到的文件夹，并双击来打开文件。
3. 找到名为 `Python.mpkg` 的文件。这是 Python 的安装程序。右键单击它，并单击打开方式 ⇨ 安装来开始 Python 的安装。

在 OS X 10.8+ 上你可能会看到 Python 不能被安装，因为它来自于“未知的开发者”。当警告出现时，不要理它！这只是因为 Python 不与在 OS X 10.8 中被引入的 Gatekeeper 安全特征兼容，但你安装它并没有什么问题，这不会给你的计算机带来安全问题。你只要单击“打开”就可以了。

4. 此时会出现关于 Python 安装包的介绍屏幕。单击 Continue。
5. ReadMe 屏幕会显示关于 Python 的重要信息。单击 Continue。
6. 许可屏幕会显示软件许可协议。单击 Continue 并单击 Agree。
7. “安装类型”屏幕会显示将会占用多少磁盘空间，单击 Install。如果有需要，你也可以改变安装位置。
8. 输入 Apple 密码并单击 Install Software 来开始安装，安装的进度会显示在屏幕上。等待 Python 完成安装。
9. 当安装程序显示安装成功时，单击 Close。

在 Windows PC 或 Apple Mac 上启动 Minecraft

在今后的冒险中，说明会告诉你何时运行 Minecraft。如果你要回顾如何在 Windows PC 或 Apple Mac 上启动 Bukkit 并把 Minecraft 与之连接，回顾本节。

你现在已经安装了所有你在 Windows PC 或 Apple Mac 上所需要的软件。但现在还没有准备好，

首先你需要参考下面的步骤以启动 Bukkit，并把 Minecraft 与之连接：

1. 双击桌面上的 `AdventuresInMinecraft` 文件夹来打开它。
2. 双击 StartBukkit 程序来运行它。这会打开 Bukkit 命令窗口。

如果你正在使用 Apple Mac，那么依赖于计算机的设置，你可能会收到一个内容为“‘StartBukkit.command’不能被打开，因为它来自于未知的开发者”的信息。如果这样的话，右键单击 StartBukkit 并选择“用终端打开”。

3. 按键盘上的任意键来启动 Bukkit。当 Bukkit 启动时，屏幕会显示信息来让你知道最新的进展。

第一次运行 Bukkit 时，你的计算机可能会显示信息来向你询问运行程序的许可，或允许 Bukkit 访问网络。如果这发生了，单击同意来给予权限 / 访问。

4. 当 Bukkit 加载完毕后，“Done（完成）”信息会出现（见图 1-6）。

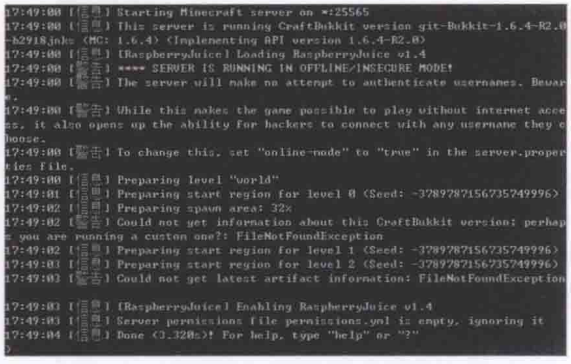


图 1-6　Bukkit 命令窗口在准备好使用时会显示“Done”信息

在本书中的所有冒险中，使用的 Minecraft 和 Bukkit 版本是 1.6.4。这意味着你需要更改 Minecraft 启动器的设置来运行 1.6.4 版本的游戏。只有在 1.6.4 版本中，我们才能保证代码和程序能够正常工作。你只需要设置一次，Minecraft 启动器就会记住你使用的版本。

如果你想要使用更新版本的 Minecraft，初学者工具包中的 `README` 文件会告诉你如何创建你自己的 Minecraft 初学者工具包。但这仅限于那些知道如何设置计算机，有设置配置文件并运行 Java 程序经历的高级用户。

参考下面的步骤以把 Minecraft 启动器版本设置为 1.6.4：

1. 启动 Minecraft。
2. 当 Minecraft 启动器窗口显示时，单击在屏幕左侧，你的用户名下方的 Edit Profile 按钮。
3. 在 Use Version 下拉菜单中，选择 release 1.6.4，如图 1-7 所示。
4. 单击 Save Profile。

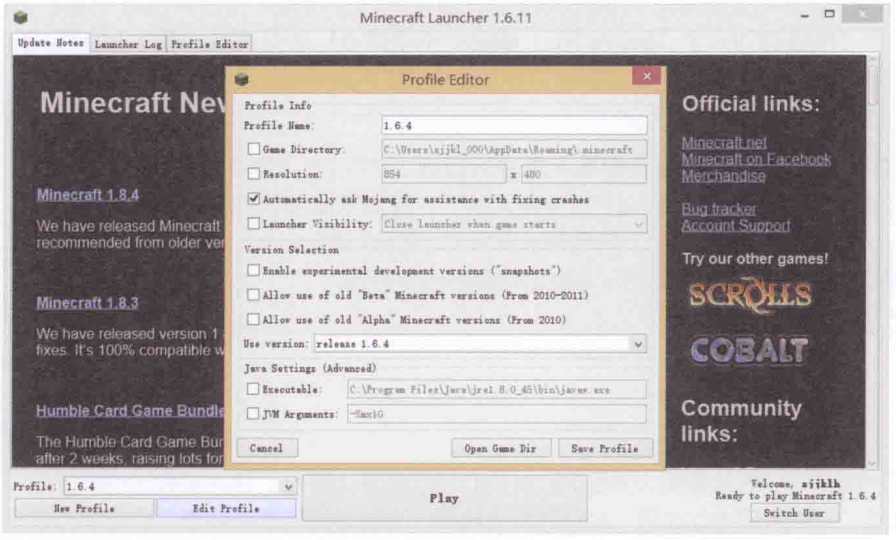


图 1-7　选择使用的 Minecraft 版本

现在把 Minecraft 连接到 Bukkit 服务器：

1. 在 Minecraft 启动器中单击 Play 来启动游戏。
2. 在 Minecraft 菜单中，选择多人游戏。
3. 在多人游戏菜单中，选择直接连接。
4. 在服务器地址中输入 `localhost` 并单击“连接”（见图 1-8）。好了！你现在应该进入了 Bukkit 服务端所创建的 Minecraft 世界。你终于来了！

图 1-8　把 Minecraft 连接到 Bukkit

停止 Bukkit

当你结束一天的冒险时，你需要与 Bukkit 断开连接。你可以通过按下 Esc 键来打开菜单，并单击“断开连接”。现在你可以安全地通过在 Bukkit 命令窗口中输入 `stop` 并按下回车来关闭 Bukkit。

在屏幕上，你会看到许多 Bukkit 的信息，告诉你关闭 Bukkit 时发生了什么。当它告诉你“Closing listening threads”后，你可能会看到一个“SEVERE”信息。不用担心！这很正常，这并不会影响你的计算机或 Minecraft 程序。

你可以在 Bukkit 控制台中输入指令来控制 Minecraft 游戏，比如时间或是天气。例如，如果在 Minecraft 世界里天黑了，但你不准备等待天亮，输入 `set time 1` 来把时间设回早晨。如果开始下雨了，你可以用 `weather clear` 来重置天气。你会通过输入“help”来找到完整的指令列表。

创建程序

恭喜！你现在已经成功设置了你的计算机，并且 Minecraft 已经在其上运行。你可能会觉得设置的过程单调乏味，但你已经完成了，并且只有你在另一台计算机上再试一次时才需要重新设置。现在是时候来做一些有趣的事了——编写你的第一个程序，“Hello Minecraft World”。

在今后的冒险中，说明会告诉你何时启动 IDLE。如果你要回顾如何启动 IDLE，回顾本节。

首先，你需要跟随下面的步骤启动 Python 并打开 IDLE：

在树莓派上：双击桌面上的 IDLE（不是 IDLE3）图标。

在 PC 上：单击“开始”按钮 ⇨ 所有程序 ⇨Python 2.7⇨IDLE（Python GUI）。

在 PC 上（Windows 8）：单击“开始”按钮，并单击 IDLE（Python GUI）或是使用搜索来查找 IDLE。

在 Mac 上：单击 Finder 菜单上的 Go 按钮 ⇨ 应用程序 ⇨Python 2.7。双击 IDLE。

现在 Python Shell 窗口会出现。

如果你使用树莓派，Python Shell 窗口会在你双击图标几秒后才出现。

第一次运行 IDLE 时，你的计算机可能会显示信息来向你询问运行程序的许可或允许 IDLE 访问网络。如果这发生了，单击“同意”来给予权限 / 访问。

当 Python Shell 窗口出现时，你可以用 IDLE 建立一个新的程序。这个程序不会做什么花哨的事。就像一个计算机编程员总是以“hello world”程序开始，你现在要建立一个 Hello Minecraft World 程序来检查你计算机中的所有东西已经安装好了。下面告诉你如何来做：

在今后的冒险中，说明会告诉你何时创建程序并保存到 `MyAdventures` 文件夹。如果你要回顾如何创建程序，只需回顾本节。

1. 单击 IDLE 菜单上的 File⇨New File。（注意在树莓派上，“New File”可能叫作“New Window”。）
2. 单击 IDLE 编辑器菜单上的 File⇨Save 来把文件保存到 `MyAdventures` 文件夹。
3. 选择 `MyAdventures` 文件夹：

在树莓派上：在路径选择下拉菜单中选择 `/home/pi`，并双击文件浏览器中的 `MyAdventures`。

在 PC 上：单击左侧面板上的“桌面”图标，双击文件浏览器中的 `AdventuresInMinecraft` 文件夹，并双击 `MyAdventures`。

在 Mac 上：在文件浏览器中单击“桌面”，单击 `AdventuresInMinecraft`，然后单击 `MyAdventures`。

4. 接下来你可以给你的文件取名。输入 `HelloMinecraftWorld.py` 并单击“保存”。文件名结尾处的 `.py` 告诉计算机这是一个 Python 程序。

把程序保存在 `MyAdventures` 中很重要。你会把你的所有程序都保存在这里，因为这里包含了使你程序运行的所有东西。

5. 是时候开始编程了！在 IDLE 编辑器中输入下列代码来开始编写 Hello Minecraft World 程序。确保字母的大小写是正确的，因为 Python 是大小写敏感（case-sensitive）的。

```
import mcpi.minecraft as minecraft
mc = minecraft.Minecraft.create()
mc.postToChat("Hello Minecraft World")
```

Python 是大小写敏感（case-sensitive）的，这意味着你输入字符的大小写必须正确。比如，Python 会把 Minecraft（大写）和 minecraft（小写）看成是不同的。如果你的输入是混淆的，那么这会导致错误，这时你必须返回查找你哪里有错。

在下一章中，你会了解更多关于代码的意义。现在，你只需要保持简单并使“Hello Minecraft World”显示在屏幕上以确保一切正常工作。

6. 单击 IDLE 编辑器菜单中的 File⇨Save 来保存你的程序。

运行程序

在今后的冒险中，说明会告诉你何时运行程序。如果你要回顾如何运行 Python 程序，只需回顾本节。

你已经建立了一个程序！现在是时候来试试它了。要开始你的程序，需要用下面的步骤告诉 IDLE 来运行程序：

1. 在你开始运行程序前，Minecraft 必须已经启动 并已经在游戏中。如果还没有的话，参考前面的指南来启动它。

2. 调整 Minecraft 窗口的大小，使你能同时看到 Minecraft 和 IDLE 窗口（如果你在使用 PC 或 Mac，并且 Minecraft 处于全屏状态，按 F11 来退出全屏，这样你就可以看到在窗口里的 Minecraft 了）。你必须打开下列窗口：

在树莓派上，你需要打开三个窗口：Python Shell，带有你的 HelloMinecraftWorld.py 程序的 IDLE 代码编辑器和 Minecraft（见图 1-9）。

在 PC 上，你需要打开四个窗口：Python Shell，显示着你的 `HelloMinecraftWorld.py` 程序的 IDLE 代码编辑器，Bukkit 指令窗口和 Minecraft（见图 1-10）。

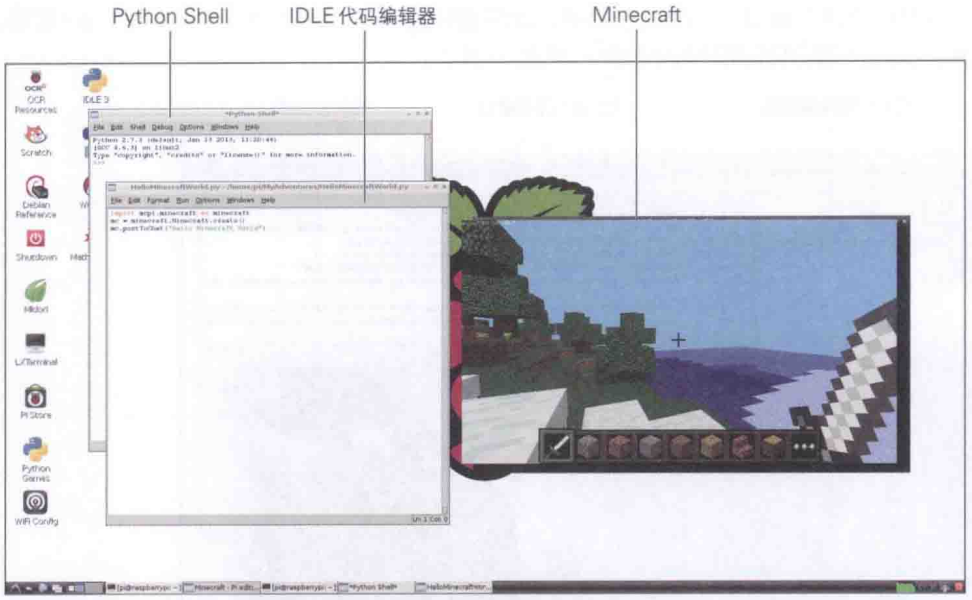


图 1-9　准备好执行你的程序的树莓派

图 1-10　准备好执行你的程序的 PC

在 Mac 上，你需要打开四个窗口：Python Shell，显示着你的 HelloMinecraftWorld.py 程序的 IDLE 代码编辑器，Bukkit 指令窗口和 Minecraft（见图 1-11）。

图 1-11　准备好执行你的程序的 Mac

3. 按 Esc 键来打开 Minecraft 菜单。用你的鼠标指针来选定 IDLE 代码编辑器。

4. 在 IDLE 菜单中选择 Run⇨Run Module（启动模块），或按下 F5 来执行 HelloMinecraftWorld.py。

5. IDLE 会自动切换到 Python Shell 窗口并运行程序。如果有任何错误，你会看到它们以红色的字显示出来。如果出现错误，请在 IDLE 中仔细检查你先前输入的代码并与上面的代码进行比较，来寻找哪里出错了。

6. 把 Minecraft 窗口放到最前面。发现了什么吗？你所有的努力得到了收获，并且“Hello Minecraft World”现在被显示在了聊天窗口（这看起来像图 1-1）。

如果一切正常，“Hello Minecraft World”现在被显示在了聊天窗口（见图 1-1）。但如果没有，或者在 Python Shell 中有错误显示，根据冒险前面的指示再仔细做一遍。你的初始设置的正确性是很重要的，如果设置不正确，在今后的冒险中，你的程序也不会运行。

停止程序

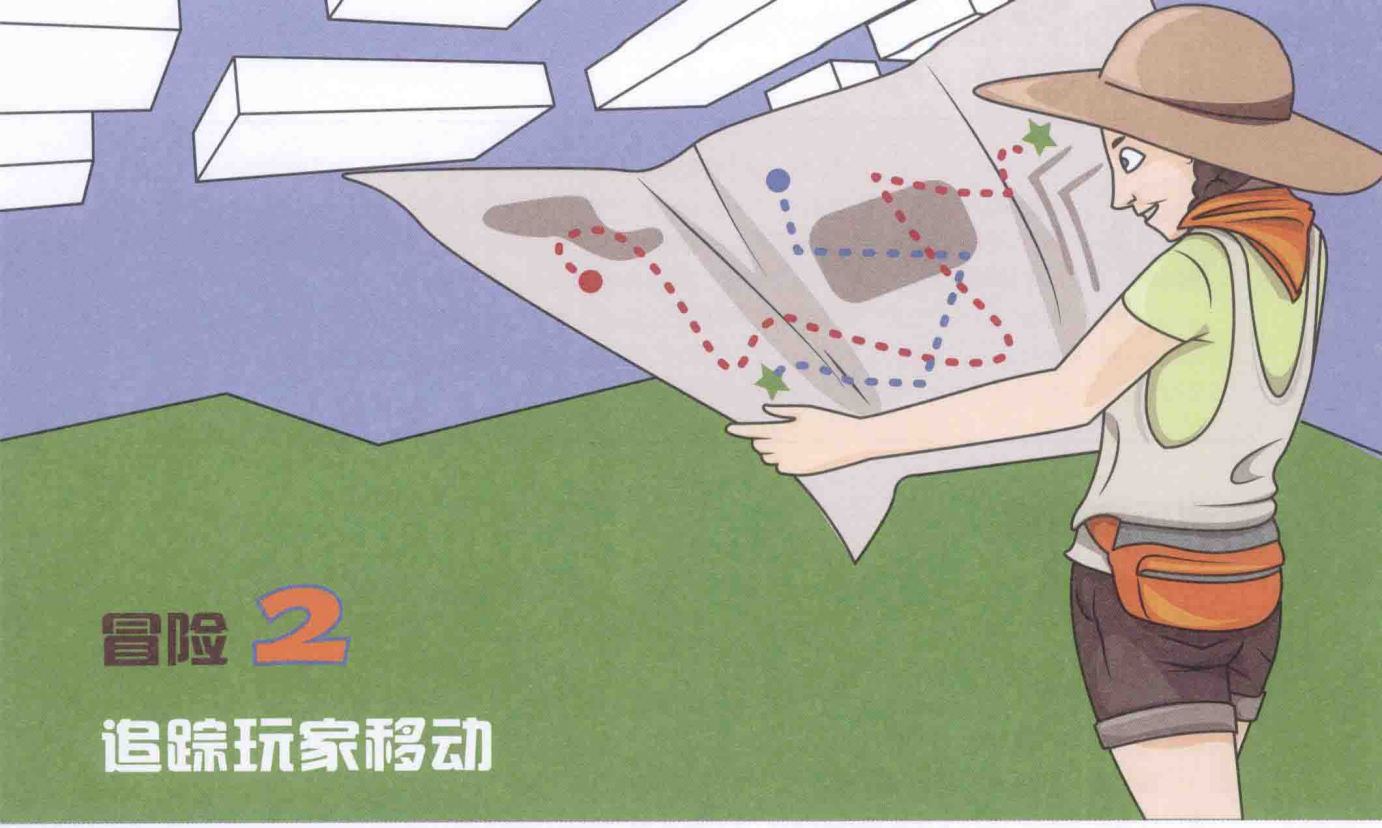
在今后的冒险中，“说明”会告诉你何时停止程序。如果你要回顾如何停止 Python 程序，只需回顾本节。

“你好，Minecraft 世界”程序运行了，在屏幕上显示了信息，随后停止。了解如何停止程序是很重要的，因为在之后的冒险中，在你告诉你的程序要停止之前，它是不会停止的！如果你要停止一个 Python 程序，选中 Python Shell 窗口，然后单击“菜单”中的 Shell⇨Restart Shell（重启 Shell），或在按下 Ctrl 键的同时按下 C。

解锁成就：最困难的部分已经过去了！你已经获取了 Minecraft 并使它运行，而且还编写了你的第一个程序，向这个世界（World）说“你好（Hello）”。

在下一次冒险中……

在冒险 2 中，你将会用 Minecraft 和 Python 学习一些简单的编程技巧，你可以用它们来定位玩家。你将了解“游戏内的游戏”的概念，并编写一些简单的游戏，这些游戏的行为会随着玩家在 Minecraft 世界中的位置而改变。



冒险 2

追踪玩家移动

当你打开 Minecraft，你所体验的是一个由别人为你设计好的游戏。Minecraft 世界充满了乐趣，但如果能够使它随你心意而变，你将体会到更多的乐趣。在使用 Python 语言编写 Minecraft 程序时，一个完整的编程环境将盈于你的指尖，你可以创造并控制你能想象到的任何东西。本次冒险将为你介绍可以用 Minecraft 和 Python 编程语言创造的一些有趣的事物。

在游戏中创造游戏，在 Minecraft 编程中是一件真正的趣事。Minecraft 是一个“世界”，而你的程序将使这个世界以崭新的方式运行。你会发现，大部分 Minecraft 程序拥有三个使它们变得更刺激有趣的功能：检测世界的某些信息，如玩家的位置；计算某个新的数据，如分数；以及完成某种行为，如将玩家移动到一个新的位置。

在本次冒险中，你将学习如何感知玩家的位置，并随着玩家的移动在游戏中触发各种不同的事件。在“欢迎回家”游戏中，你将建造一块魔法门垫：当玩家踏上门垫时，它将向你问好。我们将从实体的 Minecraft 围栏入手，引入一种名为“区域限定”（geo-fencing）的技术，而你的游戏将给你自己和你的朋友们一个挑战，要求你们相互比拼、在尽可能短的时间内收集物体。最后，你将学习如何移动玩家，如何在玩家未从指定区域及时离开时，将他弹射到天空中！

程序（program）是用特定编程语言输入、并由计算机自动执行的一系列指令或语句。你必须使用与所采用的编程语言相对应的、正确种类的指令。在本书中，你将使用的是 Python 编程语言。

语句（statement）是计算机学科的一个综合性术语，通常指的是给予计算机的一条完整的指令，例如某个计算机程序中的某一行：`print("Hello Steve")`。有些人也会称之为命令（command），对于那些在命令提示符中向计算机输入的指令，用“命令”来称呼更为确切。

Python 本身对于语句这个词有着非常精准的解释，但就本书而言，我所说的语句指的是 Python 程序中一条独立的指令或一行。如果你想了解更多有关 Python 语言的内容，你可以访问 https://docs.python.org/2/ 阅读在线参考文档。

检测玩家位置

在学习检测玩家位置之前，你需要稍微了解一下 Minecraft 世界是如何组织的。在 Minecraft 中，任何物体都是一个方块大小，而一个方块代表的是一个单位立方体（也即长、宽、高皆为一米的立方体——译者注）。当你的玩家——Steve，在 Minecraft 世界中四处移动时，Minecraft 游戏通过一系列的坐标来记录他的位置。整个 Minecraft 世界是由空方块构成的，而这些空方块则由通常占据单位立方体体积的材料所填充。这条规则也有一些例外，诸如地毯、水和岩浆之类的液体所占据的体积就比看上去更小一些，但它们不能与其他材料结合来共同填满小于一个单位立方体的空间。

通过在 Minecraft 世界中检测玩家位置，你可以使你的游戏根据玩家在世界中的移动而作出智能的回应，如在指定的位置显示信息，随玩家移动自动建造建筑以及当玩家走到指定位置时移动玩家等。通过本书，你将掌握实现所有这些功能的方法，甚至就在本次冒险中，你就将学会其中的许多内容！

坐标（coordinate）是指唯一代表位置的一系列数字。Minecraft 使用 3D 坐标代表 Minecraft 三维世界中的确切位置，每个坐标分别包含三个数字。访问 http://zh.wikipedia.org/wiki/ 坐标系，了解更多有关坐标的信息，或 http://minecraft-zh.gamepedia.com/ 坐标，了解 Minecraft 如何应用世界坐标。

这些坐标变量分别被称为 x、y 和 z。在 Minecraft 中谈及坐标时需要当心：如果你使用了诸如“左”和“右”之类的词汇，要知道是你在 Minecraft 世界中面朝的方向决定了物体是在左边还是右边。一种更好的方式是将 Minecraft 坐标联想成指南针上的不同方向。图 2-1 展现了 x、y 及 z 坐标的变化是如何与 Minecraft 世界中的移动相对应的。

在游戏中进行实际操作将帮助你更轻松地理解 Minecraft 中玩家移动时的坐标变换。在树莓派平台下，当你在世界中移动玩家时，屏幕左上角将会显示他的 x、y 和 z 坐标。在 PC 或 Mac 平台下，你可以按下 F3 键来显示坐标。你可以访问 http://minecraft-zh.gamepedia.com/ 控制，来学习所有关于 Minecraft 高级控制的知识。

- 当玩家向东移动时，x 坐标变大；向西移动时，x 坐标变小。

- 向天空上升时，y 坐标变大；向地面下落时，y 坐标变小。
- 当玩家向南移动时，z 坐标变大；向北移动时，z 坐标变小。

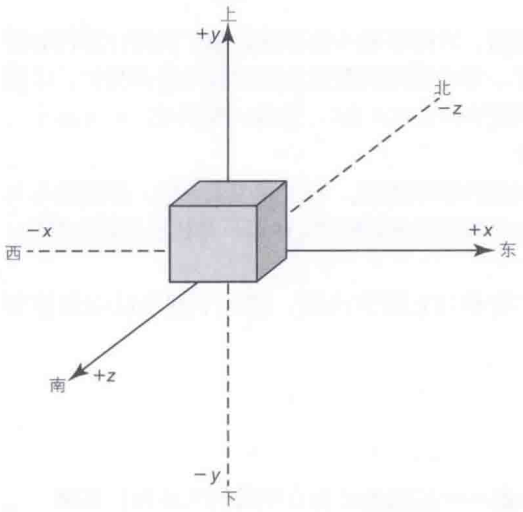


图 2-1　Minecraft 中的 x、y 和 z 坐标与指南针的朝向呈现这样的对应关系

准备启程

在冒险 1 中，你已经完成了“Hello Minecraft World”程序，现在是时候编写你自己的程序，与 Minecraft 连接了！

冒险 1 详尽地介绍了如何让程序顺利运行，而在接下来的冒险中，你还将不断完善这些步骤。所有启动配置及编写程序所需要的准备工作已在冒险 1 中作了充分阐述，但由于你尚在学习的过程之中，我们在前几次冒险时仍将给你一些提示。

如果需要关于如何在特定计算机上正确运行 Minecraft 的提醒指南，请访问配套资源网站 www.wiley.com/go/adventuresinminecraft，并选择视频“Adventure 1”。

1. 运行 Minecraft 和 IDLE，如果你是在 PC 或 Mac 上工作的话，请将服务端一并打开。目前你应该已经有了一些配置启动环境的实践经验了，但如果你需要提醒指南的话，可以随时回顾冒险 1。

2. 打开 Python 集成开发环境（Integrated Development Environment，IDE）——IDLE，就像你在冒险 1 中所做的那样。它将是你了解 Python 编程语言的窗口，并成为你输入和运行所有 Python 程序的平台。

显示玩家位置

理解坐标概念的最佳途径就是编写一个小型的实验程序，这也正是你即将要完成的任务。这个程序将

会展示你的玩家的 3D 坐标，而当玩家移动时，你将能够观察到坐标的变换。

1. 从 Python Shell 中选择 File⇨New File 来创建一个新的程序。一个新的未命名窗口将会打开，你将在这个窗口中输入你的新程序。

2. 在开始编写程序之前，先保存文件是一个较好的做法，这样你就不会在需要匆忙离开计算机的时候紧张兮兮地给程序命名了——你只要选择“保存”就行了。专业程序员甚至会随时随地保存文件，以保证代码不会丢失。保存方法是从 Python Shell 菜单中选择 File⇨Save As，并键入文件名 `whereAmI.py`。请确保你的程序是存放在 `MyAdventures` 文件夹下的。

3. 你的程序将通过 Minecraft API 与 Minecraft 游戏进行直接通信。为了使用该 API，你需要导入 `minecraft` 模块，赋予你的 Python 程序访问所有 Minecraft 游戏资源的权限。使用下列代码来导入模块：

```
import mcpi.minecraft as minecraft
```

4. 要与一个运行中的 Minecraft 游戏实现通信，你需要与它建立连接。键入下列代码以连接到 Minecraft 游戏：

```
mc = minecraft.Minecraft.create()
```

请注意 Python 语言是大小写敏感的——因此务必当心字母的大小写！正确键入大写或小写字母是很重要的。在你刚刚输入的语句中，第二个 Minecraft 必须以大写字母 M 开头，这样程序才能正常工作。

5. 接着，利用 `getTilePos()` 来向 Minecraft 游戏获取玩家的位置：

```
pos = mc.player.getTilePos()
```

6. 最后，让 Minecraft 游戏把玩家位置坐标显示出来。`print()` 会在程序运行时将这些坐标显示在 Python Shell 上：

```
print(pos.x)
print(pos.y)
print(pos.z)
```

7. 从编辑菜单中选择 File⇨Save 以保存文件。

8. 从编辑菜单中选择 Run⇨Run Module 以运行程序。

现在你应该能够看到 Python Shell 上显示的玩家位置坐标（见图 2-2）。稍后，在本次冒险中，你将用这些坐标去感知玩家所站立的位置，并建造一块魔法门垫——当你的玩家站在门垫上时，它将欢迎你回家！

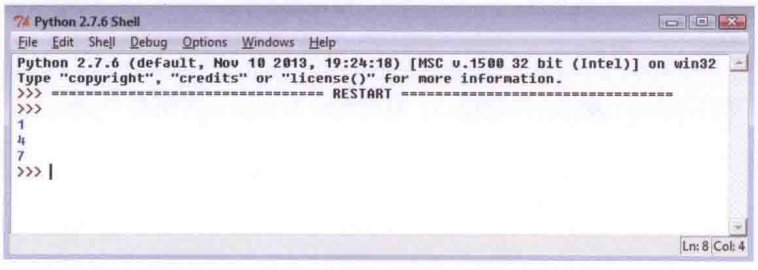


图 2-2　利用 `getTilePos` 在 Python Shell 上显示玩家位置

API 代表应用程序接口。API 为你提供了从你自己的程序中安全访问某应用程序的各个部分的方法。Minecraft API 就为你提供了从 Python 程序中访问 Minecraft 游戏的方法。你必须先运行 Minecraft 游戏，才能通过该 API 从 Python 程序连接到它。在 `import mcpi.minecraft as minecraft` 这一行中，`mcpi` 代表的是 Minecraft Pi，因为最初第一版 Minecraft API 只能在树莓派上运行。

接口（interface）是解释作为程序员应如何访问计算机系统某些其他部分的一系列规则。而 API，则是解释如何与一个正在运行的程序——在这里我们指的是 Minecraft 游戏——进行通信的一系列规则。所有 Minecraft 程序将通过 Minecraft API 来访问运行中的 Minecraft 游戏。

你所创建的所有 Python 程序都要访问 Minecraft API。为使程序正常工作，你的程序必须能够访问 `mcpi.minecraft` 模块，而它被存储在 `MyAdventures` 文件夹下的 `mcpi` 文件夹下。为保证正常访问，最简单的方法就是每次都在 `MyAdventures` 文件夹下创建 Python 程序。

深入代码

恭喜——你刚刚在你的程序中使用了一个变量！`pos.x` 是一个变量，`pos.y` 和 `pos.z` 也同样是。

变量只是对计算机内存空间中某个位置的一个命名。当你将变量名放在等号左边时，计算机就会在该变量中存储一个值，例如：

```
a = 10
```

当你在 `print` 语句中使用变量时，程序将会把这个变量的值显示在 Python Shell 上。因此，如果 `a = 10`，则下列语句

```
print(a)
```

将会打印 10，作为变量 `a` 的值。

在 `pos = mc.player.getTilePos()` 中用到的 `pos` 变量是一种特殊的变量。你可以把它想象成一个被分割成 *x*、*y* 和 *z* 三个子隔间的盒子：

```
print(pos.x)
```

试着将你的玩家移到一个新的位置，然后重新运行程序。检查一下 Python Shell 上显示的坐标——它们现在应该有所变化，已经对应到玩家的新位置了。你可以随时使用这个小程序来确定 Minecraft 世界中某个位置的坐标。

变量（variable）是对计算机内存空间中某个位置的命名。在你的程序中，你可以随时向变量中存入新的值，也可以从变量中读回存储的值并显示，或将其用于算式中。变量有点类似于计算器上的存储按钮。

简化位置显示

在继续丰富你的 Minecraft 程序之前，你可以改进玩家位置的显示方式以便理解。每次显示玩家位置时分别显示三行输出是非常占地方的。更好的方法是将所有三部分的玩家位置信息显示在同一行里，就像这样：

```
x=10 y=2 z=20
```

实现以上效果需要完成这些步骤：

1. 修改你的 `whereAmI.py` 程序，删去三条 `print()` 语句并用下面这一行代码代替：

```
print("x="+str(pos.x) +" y="+str(pos.y) +" z="+str(pos.z))
```

2. 从编辑菜单中选择 File⇨Save 再一次保存你的程序，然后单击 Run⇨Run Module 运行它。看看输出是如何得到改善的——现在信息都显示在同一行里了！

深入代码

在新一版本的 `whereAmI.py` 程序中，更多的一些“魔法”出现了，让我来解释一下它们是如何产生的。

在 Python 语言中，`print()` 会显示任何置于括号之间的内容，而输出将被显示在 Python Shell 上。你已经学会了用 `print()` 显示信息的多种不同方式。而现在，你可以更深入地审察这些用法。看看这个例子：

```
print("hello")
```

正如“你好 Minecraft 世界”冒险中所展现的那样，`print()` 将显示 hello 这个词语。词语两边的引号告诉 Python 你想要原封不动地显示所给的词语 hello。如果你在引号之间键入了空格，这些空格也将被显示出来。位于引号之间的文本被称作字符串。

这是 `print()` 的另一种用法：

```
print(pos.x)
```

`print()` 将显示变量 `pos.x` 内存储的值。这个值是一个数字，但在每次执行 `print()` 语句时，这个数字可能随着存于变量中的值的改变而有所不同。

最后，你需要理解 `str()` 在 `print()` 语句中的作用。Python 中的 `print()` 为你提供了混合数字与字符串的多种不同的方法。当你，如下列代码所示，在 Python 中混合数字和字符串时，你必须告诉 Python 让它把数字转换成字符串，以便将其附加到需要被显示的字符串中剩下的字母上。`str()` 是 Python 的内置方法，它会将括号内的变量或值转换成字符串。加号（+）告诉 Python 将“`x=.`”和变量 `pos.x` 中的值联结成一个大字符串，再将其打印到 Python Shell 窗口上。

```
print("x="+str(pos.x))
```

为什么不试试在这一行中把 `str()` 去掉，再看看会发生什么呢：

```
print("x="+ pos.x)
```

字符串（string）是一种可以存储一系列字母、数字及符号的变量。它被称作“字符串”（原意为“线”——译者注），你可以将它想象成一段串有珠子的长长的线，比如项链或手环。每个“珠子”上可以印制一个不同的数字、字母或符号。而线（字符串）则替你将它们以固定的顺序排列好。

你可以用字符串完成许多事情，比如存储你的名字、一个地址或需要显示在 Minecraft 聊天窗口的一条信息。

利用 postToChat 改变位置信息显示方式

在 whereAmI.py 程序中，你在 Minecraft 窗口中进行游戏，但你的位置信息却显示在 Python Shell 窗口中，这是非常别扭的。然而你可以借用一种技巧很轻松地解决这个问题——那就是你已经在冒险 1 中使用过的 postToChat！以下是具体方法：

1. 将 print() 语句改成 mc.postToChat()，就像这样：

```
print("x="+str(pos.x) +" y="+str(pos.y) +" z="+str(pos.z))
mc.postToChat("x="+str(pos.x) +" y="+str(pos.y) +
  " z="+str(pos.z))
```

2. 单击 File⇨Save 保存你的程序，然后从编辑菜单中选择 Run⇨Run Module 重新运行程序。

干得好！现在你的玩家位置应该已经显示在 Minecraft 聊天窗口中了，这样更方便你在运行游戏的时候查看。

引入游戏循环

每当你想要知道玩家位置时，你都需要运行一次 whereAmI.py 程序，这可能会带来一些不便。幸运的是，你可以对你的程序稍作修改，通过添加一个游戏循环来避免这个问题。本书中几乎所有其他的程序都需要添加一个游戏循环来保证程序持续运行，并且始终能够与 Minecraft 游戏交互。在大部分游戏中，游戏本身会一直运行，直到玩家将它们关闭，这也是一个很有用的技巧。在计算机科学中，这样的结构被称为死循环，因为它将永远循环下去。

死循环（infinite loop）是指永远不会终止的循环——它将一直循环到永远。跳出死循环的唯一方法就是终止 Python 程序。你可以在 Python Shell 菜单上选择 Shell⇨Restart Shell，或在 Python Shell 界面按住键盘上的 Ctrl 键和 C 键来终止 IDLE 中的程序。

你可以在已有的程序上稍作修改来添加游戏循环。新程序与原来的程序非常相似，所以这样对你来说会比较节省时间。请务必将新文件以不同的文件名存储，以免原来的程序丢失。

1. 首先，从编辑菜单中选择 File⇨Save As 来将你的程序存储为一个名为 whereAmI2.py 的新文

件。请确认你是否将文件存储在了 MyAdventures 文件夹下。如果没有，程序将不会正常运行。

2. 现在你需要导入一个新的模块，它将允许你插入时间延迟。你必须进行延时操作，因为如果你不将循环的节奏放慢，你的聊天窗口将会被洪水般的信息淹没，以至于你根本看不到当前执行的任务。在你现有的程序中加入下列**加粗**的一行代码：

```
import mcpi.minecraft as minecraft
import time
```

3. 接着，将下列**加粗**标记的几行代码加入程序主体部分，请注意，你需要对 while True 下面的几行代码做缩进排版，告诉 Python 哪些指令是需要重复执行的。需要缩进的具体原因参见“深入代码”栏目：

```
while True:
  time.sleep(1)
  pos = mc.player.getTilePos()
  mc.postToChat("x="+str(pos.x) +" y="+str(pos.y) +
  " z="+str(pos.z))
```

4. 从编辑菜单中选择 File⇨Save 保存所作的修改。

5. 现在你可以从编辑菜单中选择 Run⇨Run Module 来运行你的程序了。在 Minecraft 世界中散散步吧，你可以注意到，每隔一秒，你的玩家所处的位置坐标就会被展示在聊天窗口中。

深入代码

缩进在所有编程语言中都占有极其重要的地位，因为它展现了程序被构建的方式、以及哪些语句从属于其他语句等。在 Python 编程语言中，缩进显得尤为重要，因为 Python 是通过缩进来理解程序语义的（不像一些其他语言，如 C 语言，会用 { } 括号来给语句分组）。在使用 while 循环和其他语句的时候，缩进非常重要，因为它说明了哪些语句属于循环体，因而将被一遍又一遍地重复。在你的游戏循环中，所有位于 while True: 下方的程序语句都需要做缩进排版，因为它们都属于循环体，需要在每次循环过程中被重复执行。

在下列范例中，你可以看到“hello”只被打印了一次，而接着单词“tick”每一秒都被打印一次。time.sleep() 和 print() 都属于循环体，因为它们都做了缩进排版：

```
print ( "hello" )
while True: # 这是循环的开始
  time.sleep ( 1 )
  print ( "tick" )
```

缩进在 Python 语言中比在其他编程语言中更加重要，因为缩进的多少（缩进级数）将确切地告诉 Python 哪些语句是属于循环体的。如果你错误地处理了缩进，整个程序的语义就会发生改变。

缩进（indentation）是指程序每一行左侧的空格数。缩进是用来展示程序结构，并为循环下的程序语句和一些其他语句分组的。在 Python 中，缩进因为能够改变程序语义而起到重要作用。

正确处理缩进是很重要的。Python 可以识别使用空格和使用制表符（tab）的缩进方式，但如果你在同一个程序中混合使用空格和制表符，Python 很容易弄混你想表达的意思，所以在对程序做缩进排版时最好采用同一方式。大多数人偏好在每一级缩进时按一次 Tab 键，因为这样比打大量空格要快很多，不过一旦你开始使用 Tab 键，你就应该始终保持一致，避免将制表符与空格混在一起。

既然你的程序是在一个死循环中运行的，那么它将永远不会停止！在你运行下一个程序之前，从 Python Shell 菜单选择 Shell⇨Restart Shell，或在键盘上按下 Ctrl+C 来终止这个正在运行的程序。

创建“欢迎回家”游戏

是时候将游戏循环投入实践，开发一个简单的应用了！当你的玩家在 Minecraft 世界中四处移动时，“欢迎回家”游戏将利用感应技术（追踪玩家位置）对他进行跟踪。你还将学会利用 Python 中的 `if` 语句来实现更加复杂的感应功能。你将建造一块魔法门垫，当你站立在门垫上时，Minecraft 聊天窗口将弹出“欢迎回家”的信息。

如果你需要一份编写和运行“欢迎回家”游戏的指南，请访问配套资源网站 www.wiley.com/go/adventuresinminecraft 并选择视频“Adventure 2”。

利用 if 语句建造魔法门垫

为了判断你的玩家是否正站立在门垫上，你将使用一条 `if` 语句来比较玩家的位置和门垫的位置。当这两个位置相同时，你的玩家就应该已经到家了。

首先，让我们来看看 `if` 语句是如何工作的。在 Python Shell 上亲身一试将帮助你理解 `if` 语句：

1. 单击 Python Shell。请确认你单击的是 >>> 提示符的正右方，确保你是在正确的位置键入字符。

2. 将下列代码键入 Python Shell，接着只要你按下回车键，Python 便会立刻运行它。目前它还不会显示任何信息，因为这行代码所做的唯一一件事就是将数字 20 存入名为 a 的变量里：

```
a = 20
```

3. 向 Python Shell 键入下列代码以检查变量 a 中存储的值是否正确：

```
print(a)
```

屏幕上应该显示数字 20。

4. 试试使用 if 语句来判断是否变量 a 中存储的值大于 10。请注意 if 语句的末尾需要加上冒号。当你在第一行的末尾按下回车时，Python 会自动帮你完成下一行的缩进排版：

```
if a>10:
  print("big")
```

5. 现在，再一次按下回车，来告诉 Python Shell 你已经完成了 if 语句的输入。你应该能看到“big”这个词出现在 Python Shell 上。演示结果如图 2-3 所示。

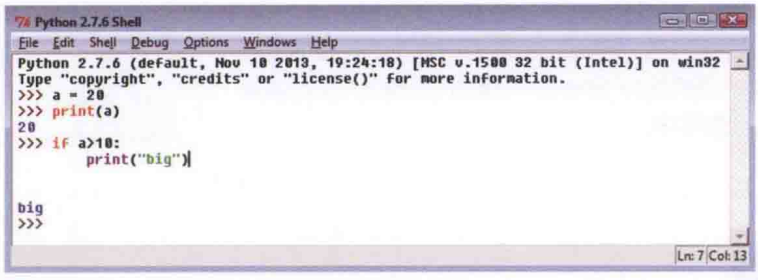


图 2-3　在 Python Shell 中交互式使用 if 语句

if 语句有两种结果，如上述代码 if a>10 的结果可以是 True（为真，a 比 10 大）或 False（为假，a 比 10 小）。If 语句也只能有两种结果，在稍后的冒险中，你将再次接触到 True 和 False 这两个值。

检查玩家是否在指定位置

为了能够判断出玩家是否正站在你的魔法门垫上，你的程序至少需要检查坐标中的两个部分。检查门垫的 x 和 z 坐标与玩家位置坐标中的 x 和 z 是否相同，是侦测玩家是否站立在门垫上的一个非常合理的方法。要实现这样的检测功能，你可以在 if 语句中使用关键词 and 来检查一个条件以及另一个条件是否成立。y 坐标在这里不是那么重要，而正因为你在这个程序中没有检查 y 坐标，所以哪怕你的玩家在门垫上空盘旋，你也仍将看到“欢迎回家”的信息。

首先在 Python Shell 中交互尝试一番，以确定代码能够正确运行：

1. 在 Python Shell 的 >>> 提示符旁，键入下列代码来设置一个变量：

```
a = 8
```

2. 现在你需要输入一条使用 and 的 if 语句。这条语句将会检查变量 a 的值是否在两个数字之间。当你第一次键入这些代码时，你将会看到单词“between”显示在 Python Shell 上，因为 8 比 5 大且比 10 小。记得在完成键入 if 的缩进部分时需要第二次按下回车：

```
if a>5 and a<10:
  print("between")
```

3. 现在将 a 的值改成一个小于 5 的数字，看看会发生什么（见图 2-4）：

```
a = 3
if a>5 and a<10:
  print("between")
```

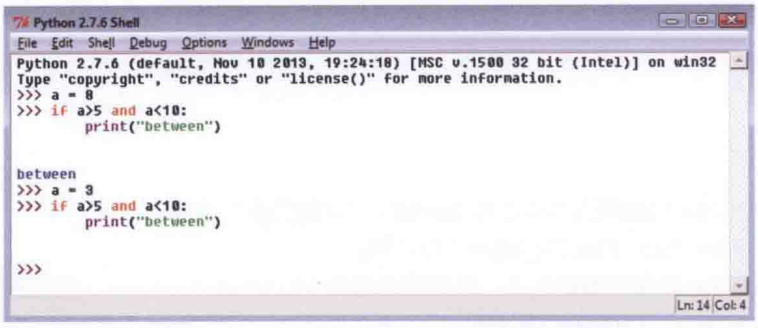


图 2-4　使用带有 and 的 if 语句检查两个或多个条件是否成立

挑战

尝试在上述程序中使用不同的数字，尤其是接近 5 和 10 的数字。哪些数字能使“between”显示在屏幕上呢？

当你测试程序时，用大量不同的数据来测试是非常有效的。有一种方法可以把可能无穷多的测试数量缩小到可控范围——那就是观察 if 语句的判断条件，而且只在这些临界数字的附近范围进行测试。在上一个例子中，我会用 4、5、6 和 9、10、11 进行测试。如果程序在给定这些数字时输出正确，那么就一定可以假设，对于所有其他的数字，程序也可以正确运行。

建造魔法门垫

在编写游戏之前，你首先需要在 Minecraft 世界中建造一些物件来为你的程序提供交互对象。而你唯一真正需要的，就是一块门垫。不过作为一切的开始，请你打开 Minecraft 并载入你已经在使用的世界。

1. 在地板上放置一块门垫，需要你从物品栏中选择一种物品，并对着面前单击右键来将它放置在地板上。用羊毛来制作门垫或许是一个不错的选择。

2. 为了获取在 Minecraft 世界中你的门垫的坐标，请再次运行你的 `whereAmI.py` 程序，然后站到你的门垫上。记下当前的 *x*、*y* 和 *z* 坐标，因为当你开始编写新程序的时候你会用到它们，这样你就可以在 Minecraft 世界中定位你的门垫了。

你可能已经等不及想建造巨型建筑了吧！在冒险 3 中，你将学会如何通过编写 Python 程序来自动建造巨型建筑，但现在，还是应该先建造一些简单的东西，专注于使程序正确运行。你也许想在你的门垫周围建造一栋简单的房屋，但一定要确保门垫位于房屋的入口处。

编写“欢迎回家”游戏

既然你已经理解了 `if` 语句的原理，你可以按照以下这些步骤来编写“欢迎回家”游戏：

1. 从 Python Shell 菜单中选择 File⇨New File 来创建一个新程序。

2. 从编辑菜单中选择 File⇨Save As 来保存你的程序，并将它命名为 `welcomeHome.py`。切记要将你的程序保存在 `MyAdventures` 文件夹下，以使它正确运行。

3. 键入下列代码来导入这个程序所需要的模块：

```
import mcpi.minecraft as minecraft
import time
```

4. 连接到 Minecraft 游戏，记得检查单词 `Minecraft` 的首字母是否大写：

```
mc = minecraft.Minecraft.create()
```

5. 加入检测玩家位置的游戏主循环，其中包含延时操作，以免循环运行过快：

```
while True:
  time.sleep(1)
  pos = mc.player.getTilePos()
```

6. 添加检测玩家是否站立在门垫上的 `if` 语句。这里使用了带有 `and` 的 `if` 语句来检查两个条件是否同时成立。如果要使程序得到玩家正站在门垫上的判断结果，门垫的 *x* 和 *z* 坐标必须与玩家位置的 *x* 与 *z* 坐标相一致。还留着你先前记下的门垫的 *x* 和 *z* 坐标吗？在这里将坐标键入，这样程序就可以知道门垫所在的确切位置了：

```
  if pos.x == 10 and pos.z == 12:
    mc.postToChat("welcome home")
```

是时候看看你的程序能否正确运行了！在编辑菜单中选择 Run⇨Run Module 后，你就可以在 Minecraft 世界中四处移动了。当你的玩家站在门垫上时，程序就会对你说“welcome home”（意为“欢迎回家”——译者注），如图 2-5 所示。酷！

图 2-5　玩家在门垫上行走时将会显示“welcome home”信息

你注意到在 `if` 语句中是如何使用 `==`（双等号）的了吗？这里一个常见的错误是使用 `=`（单等号）。试一试，看看会发生什么。你应该会看到运行程序时出现了一个错误。Python 用单等号（`=`）来表示“将语句右边的值存入语句左边的变量”，例如 `a = 3`。Python 用 `==`（双等号）来表示“比较右边和左边的值”，例如 `if a == 3:`。

在为“欢迎回家”游戏编写代码时，请确保你的缩进格式是正确的。属于 `while True` 循环的代码应该缩进一格。而属于 `if` 语句的 `mc.postToChat` 应该缩进两格。如果你以错误的方式缩进排版，程序将会完成错误的行为。

信不信由你，你已经学会了在其他冒险中将用到的所有基础知识。每个 Minecraft 程序都必须导入模块、连接游戏、进行游戏循环、感应某事件的发生，因而作出不同的响应。这些简单的程序将埋下收获神奇的种子！

挑战

你是否认为检查 *y* 坐标（指示上下方向的坐标）将会改进对于玩家是否站在门垫上的检测机制？试一试。修改你的程序，使之检查门垫所有三个方向的坐标，并观察该应用是否有更好的表现。

深入代码

有时你可能会输入一些错误的代码，这时你也许会在 Python Shell 上看到一条红色的错误信息。看一看你可能会犯的各种错误是一个好主意，这样你在偶尔遇到它们的时候就不会惊慌失措了。

如果你键入了错误的符号或以错误的顺序输入了代码，你可能会看到一个语法错误。如果你漏写了一个符号或是在不恰当的位置键入了一个变量名，你会收到一个语法错误报告。

举个例子，如果你将下面一行代码键入 Python Shell，你会看到如图 2-6 所示的语法错误，因为等号是不应该单独出现的。通常它是和变量及其他值一同被使用的：

```
=
```

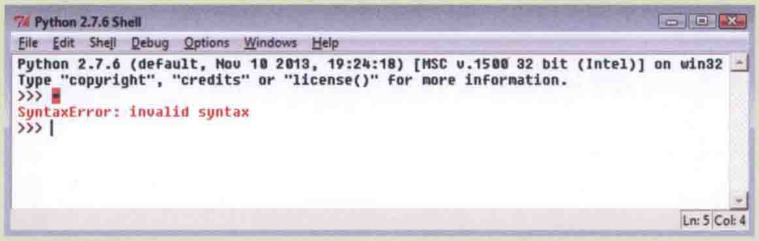


图 2-6　Python 显示你的代码中有一个语法错误

现在将这些代码键入 Python Shell：

```
print（zz）
```

你会看到 Python 显示了一个名字错误（如图 2-7 所示），因为你还没有在变量名 `zz` 中存储一个值，因此 Python 还不知道有一个叫 `zz` 的变量。

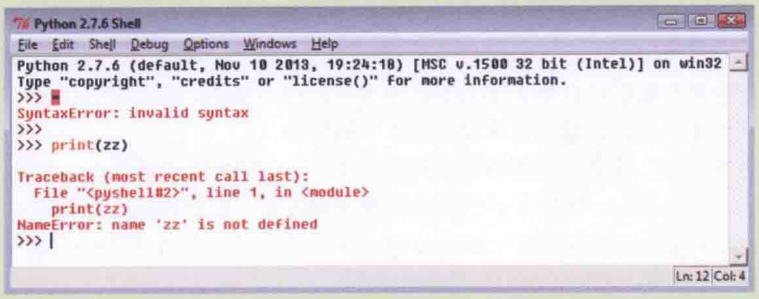


图 2-7 Python 显示关于名字错误的信息

试试键入一个 `while` 循环，但省去最后最重要的那个冒号（`:`）：

```
while True
```

你将再次收到一个语法错误报告，因为你没有使用 Python 所预期的那个冒号。

在你看到程序返回一个错误信息的时候，请不要惊慌。仔细检查它所指出的那一行，看看是否能够发现你的输入错误。有时你会因为程序中前文某一行的错误而收到一个语法错误报告，所以请检查错误信息附近的所有代码，看看是否能够发现问题所在。如果以上方法全都失败了，那么请委托一个朋友来替你仔细检查程序。相比自己的程序，人们总是更容易发现其他人的程序中的错误！当 Python Shell 报告错误的时候，出现错误的行数也包含在错误信息中。你可以看看 IDLE 窗口的右下角来判断你的光标是在哪一行，或在 IDLE 菜单中使用 Edit⇨Go To Line 跳转到指定行。

语法（syntax，原意为“句法”——译者注）指的是语言（在这里指 Python 语言）的规则，主要是指与键入代码顺序相关的一些规则。

利用区域限定（Geo-Fencing）收取租金

现在你将编写一个新的游戏，名为 `rent.py`。在这个游戏中，将有一块被围栏围住的场地。程序将检测你的玩家是否站在场地内，而当玩家位于场地内部时，它将一直向玩家收取租金。你的玩家所面临的挑战则是把这块场地内的任何物体尽可能快地移除，这样就可以缴纳尽可能少的租金。

你的 `welcomeHome.py` 程序利用了 `if` 语句来检查玩家坐标与门垫坐标是否一致。要使程序判断结果为“一致”，你的玩家必须站在门垫的正上方。在这一点上，你可以利用区域限定技术来作一些改进，这也是你即将在这个新游戏中使用到的技术。对于玩家位置的检测有所改进，因为程序所检查的是玩家是否站立在 Minecraft 世界中的某一区域内，而并非限定于一个具体的坐标。这将允许你检测玩家是否站在一片开阔区域中的任意位置上，因此站立在方块上时，位置也不需要那么精确。

区域限定（geo-fencing）指的是在地图上根据坐标建造虚拟围栏的一种综合性技术。当任意物体进入这片被虚拟围栏围住的区域时，将发生一定的事件响应。“任意物体”指的可以是 Minecraft 世界中的一个玩家、真实世界中的一个人、或如割草机之类的机器、甚至真实世界中的动物。

在很多情况下，区域限定是在牲口或交通工具等离开预定区域时给使用者发送警报的。你可以访问 http://en.wikipedia.org/wiki/Geo-fence 了解更多区域限定在真实世界中的应用情况，并访问 http://www.cbsnews.com/news/kenya-uses-text-messages-to-track-elephant/ 来了解在真实生活中，人们是如何利用区域限定来追踪野生大象的。

实现区域限定需要两样东西：一是需要被追踪的物体或区域，二是该物体四周的“围栏”的限定坐标范围。现在你要用真实的 Minecraft 围栏在物体周围构造场地、并记录该场地四个角落的坐标，从而学习实现区域限定的方法。

如果只是要使程序正常运行，你不一定非要建造围栏，但建造围栏会使这个游戏更加有趣，因为当你需要进出场地的时候，你需要跃过围栏或是助跑一段距离来穿过围栏的间隙。在游戏中添加障碍物会使游戏更富于挑战性，且更令人兴奋。这些 Minecraft 围栏将与区域限定的虚拟围栏对齐，并为程序提供了一种非常简单的方式来对“玩家是在场地内吗？”这个问题做出回答。

建造完成的场地和围栏看上去应该如图 2-8 所示的那样：

图 2-8　一块被围栏围住的场地，这块场地是 10×10 个方块大小

找出场地角落坐标

为了用程序对场地施加区域限定，你需要掌握场地角落坐标的精确数据，这样你才能将这些数据输入你的 Python 程序中。

要完成这个任务，最简单的方法就是运行先前的 whereAmI.py 程序，然后操作玩家分别跑到场地的四个角落，记录下每个角落的 *x*、*y* 和 *z* 坐标。找一张纸，画一张你所建造的场地的草图，并在图上标注角落坐标。编写程序时，你将用到它们。要记得当玩家向南移动时 *z* 坐标变大，而向北移动时 *z* 坐标变小，正如你在图 2-1 中所看到的那样。

在图 2-9 中，你可以看到我在编写本书时所建造的场地的角落坐标。我记下了所有四个角落的坐标，因为我需要通过它们来判断 *x* 和 *z* 方向的最小和最大坐标。注意最小坐标出现在场地的左上角。

区域限定程序需要四个数字。首先，你需要知道 *x* 方向的最小和最大坐标，这些数据可以从你的图表中找出。在前文的范例中，最小 *x* 坐标是 10，而最大 *x* 坐标是 20。我把最小 x 坐标称作 *X*1，最大 x 坐标称作 *X*2。

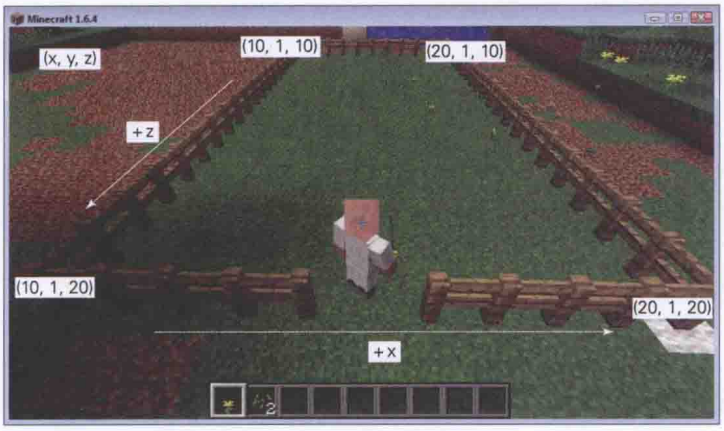


图 2-9　有了场地角落坐标，就可以实现区域限定了

在 Minecraft 世界中，随着玩家站立位置的不同，一些坐标可能是负数（如 x=-5，y=0，z=-2），这可能导致计算操作变得有些棘手。你可能会发现，首先移动到一个 Minecraft 世界中坐标皆为正数（如 x=5，y=2，z=4）的地方会使一切变得更加简单。你可以通过观察 Minecraft 屏幕的左上角（在树莓派平台上）或按下 F3（在 PC 或 Mac 平台上）来检查玩家站立的位置。

接着，你需要找出 z 方向的最小和最大坐标。在这个例子中，最小 z 坐标是 10，而最大 z 坐标是 20。我把最小 z 坐标称作 Z1，最大 z 坐标称作 Z2：

```
X1 = 10
Z1 = 10
X2 = 20
Z2 = 20
```

将你的最大和最小坐标以相同方式命名，因为你将在下一步——编写程序中用到它们。

编写区域限定程序

最后一步就是编写程序添加区域限定，并且当玩家站立在场地内时向其收取租金。这个程序的结构与 whereAmI.py 程序类似。

在这个新游戏中，你还将使用常量，以便将来进行移动场地的操作。这样的话，如果你想要移动场地，也不需要在程序中苦苦寻找每一个需要更改的坐标值了。

常量（constant）是对计算机内存中某个部分的命名（就像变量一样），你可以在常量中存储程序运行时通常不会改变的值。

Python 语言不像其他的编程语言那样拥有一套处理常量的特殊机制。Python 程序员通常遵循这样的编程习惯（也即普遍的工作方式）——用大写字母命名常量。这使得常量在编程过程中显得更加突出，而一旦给常量赋予了初始值，你就不应该改变它。其他编程语言各有自己的常量处理机制，在这些语言中，如果你尝试改变常量的值，程序将会报错。Python 没有这样的特征。

按照以下步骤编写程序：

1. 首先，从菜单中选择 File⇨New File，在 IDLE 中打开一个新窗口。
2. 从“菜单”中选择 File⇨Save As，并将文件命名为 `rent.py`。
3. 导入该程序所需要的模块：

```
import mcpi.minecraft as minecraft
import time
```

4. 连接到 Minecraft 游戏：

```
mc = minecraft.Minecraft.create()
```

5. 现在你需要为限定区域的四个坐标定义一些常量。请确保你使用的是先前得到的坐标。这里我用的是我自己的程序中所用到的坐标，而你的坐标可能有所不同：

```
X1 = 10
Z1 = 10
X2 = 20
Z2 = 20
```

6. 接着，创建一个变量以动态记录玩家被收缴的租金数量。游戏刚开始时还未收取任何租金，因此创建一个名为 `rent`（意为“租金、地租”——译者注）的新变量并将其赋值为零：

```
rent = 0
```

7. 用 `while` 语句创建游戏主循环。你的程序应该每秒循环一次，因此需要使用一条 `sleep` 语句来使程序在每次循环时延迟 1 秒。每次循环时延迟一小段时间不仅放慢了程序节奏，还为你提供了计算玩家在场地内停留时间的简便方法。稍后你将在每次游戏循环中对 rent 变量加 1，而且由于你已经在循环体的起始处添加了一秒的延迟，这就意味着租金将每秒增加 1。请确保目前的代码缩进处理是正确的：

```
while True:
  time.sleep(1)
```

8. 为了判断你的玩家是否在限定的区域内，你需要知道玩家的位置。就像在之前的程序中所做的一样，利用 `player.getTilePos()` 来获取玩家位置：

```
  pos = mc.player.getTilePos()
```

9. 现在你来到了程序最重要的部分。在这里，你将引导程序使用你先前得到的四个坐标来判断出玩

家是否站在场地内，如果是，则向他收取租金，并将这些信息显示在聊天窗口来告知玩家。键入下列代码以实现上述功能：

```
if pos.x>X1 and pos.x<X2 and pos.z>Z1 and pos.z<Z2:
  rent = rent+1
  mc.postToChat("You owe rent:"+str(rent))
```

程序进展到这个阶段，在缩进的正确性上，你必须十分小心。Python 是根据缩进来判断哪些程序语句是属于同一模块的。你可以看到所有在 `while True:` 下的语句都采用了一级缩进，这样 Python 能够知道所有这些语句都是属于 `while` 循环的。但是请注意观察 `if` 语句，在它之后的两条语句又缩进了一级。这意味着这两条语句是 `if` 语句的一部分，因此只有当你的玩家站在场地之内时，这两条语句才会执行。

再次保存程序，并从“菜单”中选择 Run⇨Run Module 来运行程序。

好好享受一番操纵玩家进出场地的乐趣吧。当玩家位于场地内时，每隔一秒，聊天窗口就应出现一条信息，表明玩家已经被收缴了多少租金。当玩家离开场地时，他将不会被收取租金。你的程序还将记录下玩家已经被收缴的租金金额，因此当玩家重新进入场地时，程序不会忽略玩家已经欠下的租金金额。

挑战

为了增加游戏的趣味性，在玩家停留在场地内时，不如给他分配一些任务。在场地内部各个位置随机放置一些方块，然后再次运行你的程序。玩家面临的挑战即是在缴纳尽可能少的租金的情况下收集所有的方块。敲击并移除方块就可以算是收集方块。虽然程序无法检测该事件的发生，但是加入这样的事件会使你的游戏变得更加有趣。你可以将各种各样的物体分散放置在场地的各个位置，然后向你的朋友们发出收集物体的挑战，而收集完物体时缴纳租金最少的玩家就是挑战的胜利者！

移动你的玩家

还有一种方式可以使你的游戏变得更富于挑战且更令人兴奋，它将涉及 Minecraft API 的另一个在游戏创作中非常有用的特性——那就是在 Minecraft 世界中将你的玩家移到另一个不同的位置！为了实现该功能，你要在已有的 `rent.py` 程序上稍作修改，如果玩家在场地内停留超过 3 秒，他就将被弹射到天空中并弹出场地之外，因而不得不重新回到场地。

首先你需要找到一个指定场地之外的地点，如果玩家在场地内停留时间过长，该地点将是你希望玩家弹飞后落地的位置。最简单的方法也许是找到场地的最大 *X* 坐标和 *Z* 坐标，然后分别加上 2 得到新坐标。为了创造戏剧性效果，你需要选择一个位于天空中的 *Y* 坐标。实际的数字选择将取决于你所建造的场地在 Minecraft 世界中处于多高的位置，但如果你是在 `y=0` 处建造的场地，那么你可以选择 10 左右的数字作为 *y* 的值。这样，你的玩家将被弹射到天空中，而重力将使他重新落向地面。

现在你要按照下列步骤修改你的 `rent.py` 程序，当玩家在场地内停留超过三秒时，他将被弹出场地：

1. 在程序起始处，设置三个新常量来记录中心位置。该中心位置应该刚好在场地之外，而当你的玩家被弹出场地时，他将被移动到这些坐标所对应的位置。需要添加的代码已加粗标注：

```
X1 = 10
Z1 = 10
X2 = 20
Z2 = 20

HOME_X = X2 + 2
HOME_Y = 10
HOME_Z = Z2 + 2
```

2. 现在你要测定玩家在场地内停留的时间，为此你需要另一个变量，你可以将它称作 `inField`。它将存储玩家在场地内已经停留的秒数，并用于判断从何时起玩家在场地中停留时间过长！同样，你只需要添加加粗部分的代码：

```
rent = 0
inField = 0
```

3. 接着，你需要加上一行代码，这样当玩家处于场地之内时，`inField` 变量的值将增加 1。添加加粗部分的代码来实现该功能：

```
if pos.x>X1 and pos.x<X2 and pos.z>Z1 and pos.z<Z2:
  rent = rent+1
  mc.postToChat("You owe rent:"+str(rent))
  inField = inField+1
```

4. 现在你要添加一条 `else` 语句，如果玩家不在场地之内时，程序会将计时器清零。请确保你的 `else` 语句进行了正确的缩放排版，使 Python 能够确切理解你的程序语义。很快你就将了解到有关 `else` 语句的知识以及 # 符号的作用，但现在，请直接键入加粗的代码：

```
  mc.postToChat("You owe rent:"+str(rent))
  inField = inField+1
else: # 不在区域内
  inField = 0
```

5. 最后，你需要在程序末尾追加一些代码，这样当玩家在场地内停留超过三秒时将他弹射到天空中并弹出场地。重力会使他落向地面，然后他将不得不重新跑入场地。这些代码应添加在程序的最后。`if` 应缩进一级，因为它是 `while True` 循环体的一部分，而从属于 `if` 的语句应从左侧缩进两级。键入以下代码：

```
if inField>3:
  mc.postToChat("Too slow!")
  mc.player.setPos(HOME_X, HOME_Y, HOME_Z)
```

6. 保存程序，并从编辑菜单中选择 Run⇨Run Module 来运行程序。

现在你应该觉得这个游戏非常有趣了，因为你必须精心规划取物策略，不时进出场地来避免玩家被弹射到天空中！现在，就像你以前曾经做过的那样，选择各种各样的物体并将它们随机分散放置在场地内。给自己和朋友们一个挑战，尝试“收集”（也即摧毁）尽量多的方块，同时使得租金账单上累计的金额最小（见图 2-10）。

图 2-10　当你的玩家在场地内停留时间过长时，他将被弹射到天空中

深入代码

你已经用到了 Python 语言的两个新组件，这里需要更深入地解释一下。

首先，你用到了 `else` 语句。`else` 是 `if` 语句的一个可选部分。它应永远与对应的 `if` 具有相同的缩进级数，而从属于 `else` 的语句块也应该进行恰当的缩进排版。这里有一个例子：

```
if a>3:
  print("big")
else:
  print("small")
```

如果变量 `a` 的值大于 3，Python 将会在 Python Shell 上打印“big”，否则（也就是如果 a 的值不比 3 大）它将在 Python Shell 上打印“small”。换句话说，如果 a>3 为真，Python 将打印“big”，否则如果 a>3 为假，则打印“small”。

一条 `if` 语句不一定要有 `else`（它是可选的），但有时你会希望在程序中对条件为真和条件为假的情况作不同的处理。

你所用到的另一组件是注释（comment）。你会发现，对于你自己和任何阅读代码的其他人来说，注释是在程序中留下小提示的一种非常有用的方式。

注释以井字符开始，就像这样：

```
# 这是一条注释
```

你可以在一行中的任意位置添加注释。无论你在哪里添加了井字符，从井字符到那一行末尾之间的所有文字都将被 Python 忽略。

关于玩家追踪的更多冒险

在本次冒险中，你学会了游戏循环的基础知识并将其投入了实践，同时辅以检测技术和区域限定技术，创建一个有趣的游戏。当你制作游戏时，让朋友和家人试玩并将他们喜欢或不喜欢的部分告诉你是一个很不错的主意。这是获取反馈，从而帮助你改进游戏的一种很好的方式。

- “欢迎回家”游戏中存在一个问题：当你的玩家站在门垫上时，程序将不停地显示“welcome home”并填满整个聊天窗口。你能想出一个修改方案，使它在你进屋时只显示一次“welcome home”吗？
- 要想使游戏变得更加精彩，有一种寻找灵感的绝佳方式，那就是去接触其他游戏，并观察它们是怎样吸引玩家的。你可以接触大量其他计算机游戏，并整理出一张关于游戏可以如何处理玩家位置改变的清单，内容越多越好。
- 上网搜索关于地理围栏的资料，看看你是否能够找到一些在工业中卓有成效的应用方式。如果其中任何一个方面引起了你的兴趣，那么你可以尝试从该方面入手编写一个 Minecraft 程序——基于你在本次冒险中所学的知识，编写一个属于你自己的简单游戏。

快速参考表

导入 Minecraft API	与 Minecraft 建立连接
`import mcpi.minecraft as minecraft`	`mc = minecraft.Minecraft.create()`
获取玩家位置	**在 Minecraft 聊天窗口显示信息**

`pos = mc.player.getTilePos()`
`x = pos.x`
`y = pos.y`
`z = pos.z`
	`mc.postToChat("Hello Minecraft")`

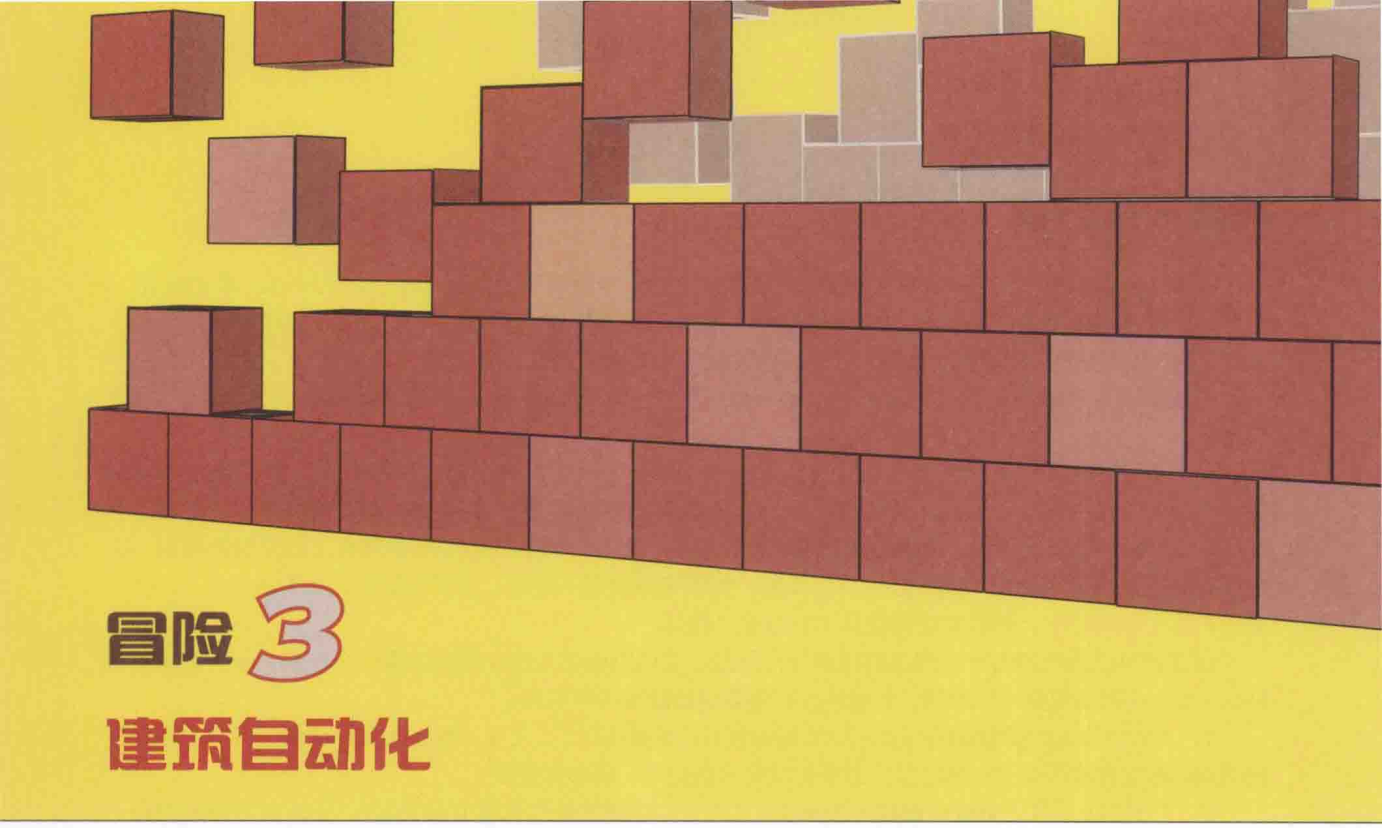
设置玩家位置	

`x = 5`
`y = 3`
`z = 7`
`mc.player.setTilePos(x, y, z)`	

解锁成就：在 Minecraft 中创建了一个响应玩家移动的精彩游戏。

在下一次冒险中……

在冒险 3 中，你将学会如何利用 Minecraft 方块及 Python 循环来使房屋等大型建筑的建造自动化。利用 Python 程序，你能够以远快于徒手的速度建造巨型建筑。你将能够建造一整座 Minecraft 城镇，所需要的时间甚至比朋友们一起手工搭建的时间还要少。



冒险 3
建筑自动化

在 Minecraft 中，建造是一件非常有趣的事情。你可以造出几乎任何你能想象到的东西，这一切只受限于 Minecraft 世界的大小和你的想象力。你可以建造房子、城堡、地下水道、带泳池的多层旅馆，甚至一个完整的城镇。但是用不同种类的方块建立复杂的建筑，尤其是当建造包含大量重复结构的建筑时，会花费大量的时间和精力。如果你能使一些建筑任务变得自动化，将会如何呢？难道对你的朋友来说，这不是一件魔术一般的事情吗？

诸如 Python 等的编程语言对于自动完成复杂的任务大有裨益。在本次冒险中，你可以学习一些用于在 Minecraft 世界中自动建造大量的方块，然后通过不断重复执行这些指令来建造大量类似的建筑的神奇方法，例如，一个布满了房子的街道（见图 3-1）。当你的朋友在房子与房子之间走过的时候，他们会惊异于这些宏伟的建筑，同时疑惑你是如何建造出如此复杂的建筑的。你也会学习到如何让程序在运行时从键盘读入数值，这样你的用户就可以直接使用你的程序，而

图 3-1　一个用 Minecraft 方块建造的房子

不需要先修改你的源代码。

建造一个方块

在 Minecraft 世界中，每个不同的方块都有它自己的方块编号。如果你是一名 Minecraft 老玩家，你可能已经知道许多方块的编号。然而，Minecraft 的程序接口也允许你使用方块的名称代替方块编号，这样可以让你更加容易地编写代码。方块名字只是一个常量（你已经在冒险 2 中见过），它们代表了不同的方块类型。在附录 B 的引用章节里，有一张表，它包含了大多数常见方块类型以及它们对应的名字和编号。

Minecraft 世界可划分为多个虚拟的立方空间（每一个空间都有它唯一的坐标），即使这个空间包含的方块类型是空气（AIR），计算机也会一直记住它的编号。作为一名 Minecraft 的程序员，你可以向 Minecraft 程序接口询问每个坐标上的方块的具体类型，同样你也可以改变任意坐标上的方块的类型。这种操作会在游戏中呈现出不同的方块，比如说，如果你想建造一座桥，你可以把方块类型从空气（AIR）变为石头（STONE），你将会在冒险 4 中完成这个任务。

现在你可以开始准备一个简单的程序作为示范，它可以自动在玩家面前放置一个方块。一旦你成功在 Minecraft 中自动放置一个方块，你就可以在里面自动建造任何东西。

1. 启动 Minecraft 游戏和 IDLE，如果你在 PC 或者 Mac 上工作，那么同时也启动服务端。你现在应该对启动这些应用有了一些经验，如果你需要一些提示，请参考冒险 1。

2. 打开 IDLE 和 Python 集成开发环境（IDE），这里就是你编写和运行你的 Python 程序的窗口。

3. 单击菜单中 File⇨New File，新建一个程序，然后单击“文件 -> 另存为”来保存你的程序，命名这个程序为 `blocck.py`。记得把你的程序保存在 `MyAdventures` 文件夹中，否则它们将不能被正常运行。

4. 现在导入你将会用到的模块，这里你会使用到一个额外的模块叫 `block`，它包含了所有的常数，这些常数代表了 Minecraft 支持的所有的方块类型。在窗口中输入：

```
import mcpi.minecraft as minecraft
import mcpi.block as block
```

5. 输入如下语句用于连接 Minecraft 游戏：

```
mc = minecraft.Minecraft.create()
```

6. 获取玩家的位置，并且存入变量 `pos` 中，你将会使用这个位置来计算你面前那个空间的坐标，并在那里放置一个新的方块：

```
pos = mc.player.getTilePos()
```

7. 现在要在与玩家当前所在位置相对的那个坐标上放置一个新的方块。在这一节最后，你可以了解到什么是“相对坐标”，通过使用 `pos+3` 来保证这个方块不会出现在你的正上方，在窗口中输入：

```
mc.setBlock(pos.x+3, pos.y, pos.z, block.STONE.id)
```

8. 单击File⇨Save来保存你的程序，然后单击“编辑菜单”中的Run⇨Run Module来运行这个程序。

现在你应该看到离你非常近的地方出现了一块石头。这样你已经通过编程实现了在玩家面前放置一个方块的任务。以这个简单的实例作为起点，你现在可以通过建筑自动化建造一些真正有趣的建筑。

挑战

石头（STONE）并不是一种能让人提起兴趣的方块。浏览一下在附录 B 中的方块目录，然后在实验中尝试使用其他类型的方块，一些方块并不适用于所有平台，所以附录 B 中同样列举了这些方块适用的平台的类型。

比如说，GLOWING_OBSIDIAN 可以在树莓派平台上运行，但不能在 PC 或 Mac 平台上运行，一些方块是受重力影响的，所以用水（WATER）和沙子（SAND）做实验会引发一些有趣的效果。

相对坐标（relative coordinates）是指相对于其他点的坐标位置的坐标（这些点可能是 Minecraft 世界中玩家所处的位置），比如说，pos.x,pos.y+10,pos.z+3 代表了在 Minecraft 世界中距离玩家向上 10 个方块长度，向南 3 个方块长度的位置，换句话说，这个坐标和玩家所在的位置关联。当玩家移动后，这个相对坐标也会变化。

绝对坐标（absolute coordinates）是指使用固定数字来代表坐标位置的坐标（它可能是 Minecraft 世界中是一个方块），比如说，坐标 x=10,y=10,z=15 就是一个绝对坐标——每次你使用这个坐标，它都代表 Minecraft 中坐标为（10，10，15）的那个方块，永远不会改变。

挑战

如果你已经完成了冒险 2，加载并且运行 whereAmI.py 程序，然后在 Minecraft 世界中将人物移动到一个相对空旷的地方。记录下你当前所处位置的坐标，修改你的 block.py 程序中的 setBlock() 函数，使用记录下来的绝对坐标，然后运行这个程序并看它的运行效果。思考一下相对坐标和绝对坐标对你编写程序会起到什么样的帮助呢？

建造更多的方块

以上文的内容作为基础，你可以开始建造任何东西。现在通过扩展你的程序，建造更多的方块。当你建造更多方块的时候，你唯一需要记住的事情是，你需要使用一些简单的数学知识来计算出你想要建造的方块的坐标。

现在你将要扩展你的 build.py 程序，通过对每个你想要放置的方块使用 setBlock() 函数，使这个程序能够在玩家面前放置另外 5 个方块。你即将完成的程序将会建造一个建筑，它看起来就像骰子上的点，为了实现这个，请按照如下步骤操作：

1. 在“编辑菜单”中选择 File⇨Save As，然后将你的程序命名为 dice.py；
2. 然后在程序的末尾添加以下代码：

```
mc.setBlock(pos.x+3, pos.y+2, pos.z,   block.STONE.id)
mc.setBlock(pos.x+3, pos.y+4, pos.z,   block.STONE.id)
mc.setBlock(pos.x+3, pos.y,   pos.z+4, block.STONE.id)
mc.setBlock(pos.x+3, pos.y+2, pos.z+4, block.STONE.id)
mc.setBlock(pos.x+3, pos.y+4, pos.z+4, block.STONE.id)
```

3. 保存程序然后运行它，你可以看到一个简单的建筑出现在你面前，它看上去就像是骰子上数字六的那面（见图 3-2）。

图 3-2　在玩家面前建造一个建筑，它看起来像骰子上数字六的那面

挑战

修改 dice.py 程序，让六个点附近的空间都被白色方块填满，这样它看起来更像是骰子的一个面。实验通过设置不同的方块，比如石头或者空气，建造骰子的其他面。记住，在附录 B 中有一个非常详细的关于方块属性的列表。

使用 for 循环

通过每次建造一个方块，你可以建造任何你想要的建筑，但是这就像是用手在计算机屏幕上画一个复杂的画，每次只能画一个点。你需要的是某种复制方块的方法，这样你就可以在不增加程序大小的情况下，建造更大的建筑。

用 for 循环建立多个方块

幸运的是，和其他程序语言一样，Python 也有循环语句。你在冒险 2 中已经见过循环语句：`While True:` 游戏循环。在 Python 中，循环是用来多次复制东西的方法。

你在这里使用的循环叫作 `for` 循环，有时也被称为计数循环（counted loop），因为它使用了一个固定的循环次数。

> for 循环被称为计数循环（counted loop）或者“数字控制循环”，因为它总是循环固定的次数，你在 range() 语句中决定了它循环几次。for 循环语句在 Python 中非常特殊，因为它能计数的东西不仅是数字，你将会在冒险 6 中找到它更加高级的用法。
>
> 
>

就像你在冒险 2 中使用 `while` 循环语句来建立你的游戏循环一样，你必须缩进那些属于 `for` 循环的语句，这样 Python 语言才会知道只有这些语句才需要被重复地执行。

为了明白这是怎么实现的，按照以下的步骤，使用 Python Shell 来实现一个 `for` 循环：

1. 单击 Python Shell 窗口，然后把光标移动到最后一个标识符（>>>）的右侧。
2. 输入以下语句，然后按键盘上的回车键：

```
for a in range(10):
```

Python 现在还不会做任何事，因为它正在等待接下来属于这个循环的语句（这些语句通常被称为循环体）。

3. Python Shell 会把接下来的一行自动缩进一级，这样 Python 就会知道你的 `print()` 语句属于这个 `for` 循环，现在键入 `print()` 语句：

```
  print(a)
```

4. 按下两次回车键，第一次按下回车键表示 `print()` 语句的结束，第二次按下回车键告诉 Python Shell 这个循环已经结束了。

现在你可以看到数字 0 ~ 9 依次出现在 Python Shell 的屏幕上。

深入代码

你刚才使用的 for 循环有一些有趣的特征，

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

实例中，变量 a 被称为“循环控制变量”，它意味着每次循环执行的时候，它的值都是在变化的。第一次运行的时候，变量 a 的值是 0，第二次是 1，以此类推。

语句 `range(10)` 告诉 Python 你想要通过这个 `for` 循环来计数，生成一个从 0 到 9 的数字序列。

末尾的冒号（`:`）就和你在之前冒险中使用的 `while` 循环和 `if` 语句中的冒号一样。冒号标记了循环体的起始位置。循环体就是（在这个实例中）用来被重复执行 10 次的。

任何在循环中的 Python 语句（换句话说，就是那些从 for 语句开始，都缩进一级的语句）都可以调用变量 a，并且这个变量每次循环都会代表不同的值。

不管你是出于什么目的，你都可以在循环体的任何地方调用循环控制变量 a。你不必每次都申明变量 a。同样你可以把它命名为任何名字，使用更具表述力的变量名称是很有实践意义的，这样其他阅读你的代码的人可以更轻易地理解它，在这个实例中，一个可以代替 a 的更好的名称是 count 或者 number_of_times。

使用 for 循环建立一座巨塔

你想要在 Minecraft 中建立一座巨大的石头塔吗？既然你已经知道了所有关于 for 循环的内容，你已经可以实现它了。参照以下的步骤：

1. 在“编辑菜单”中选择 File⇨New File，新建一个程序，选择 File⇨Save As 并将它命名为 tower.py。

2. 和往常一样，导入你需要的模组：

```
import mcpi.minecraft as minecraft
import mcpi.block as block
```

3. 连接 Minecraft 游戏：

```
mc = minecraft.Minecraft.create()
```

4. 如果你在玩家周围建造一个塔，你就可以更加轻易地发现它，因此你现在要做的第一件事情是识别玩家当前所在的位置：

```
pos = mc.player.getTilePos()
```

5. 你准备建造一个 50 个方块高度的塔，因此使用一个循环 50 次的 for 循环来开始建造，不要忘记行末的冒号：

```
for a in range(50):
```

6. 接下来一行是缩进的，因为它属于 for 循环体。在 Minecraft 世界中，*y* 坐标代表了高度，因此把变量 a 和玩家的 *y* 坐标相加，这个表示在 Minecraft 世界中，每次循环上升了一个方块的高度：

```
    mc.setBlock(pos.x+3, pos.y+a, pos.z, block.STONE.id)
```

7. 保存这个程序并运行它，它正常运行了吗？你的 for 循环应该在你的面前建立了一座巨大的塔，和图 3-3 中的类似，让我们数数，它有 50 个方块高吗？

在以上代码中，for 循环执行了 50 次，并且在每次循环里，执行了一次缩进的语句，这样变量 a 在每次执行后，都会在原来的基础上加 1，然后把变量 a（循环控制变量）和变量 pos.y（玩家所在高度）相加，这样才能够建立一个 50 个方块高度的塔（见图 3-3）。

图 3-3　在 Minecraft 中使用 for 循环建造的巨塔

挑战

在 `for` 循环里修改 `range()` 函数来建立更高的塔，你需要建立多高的塔，才能看不到它的顶端呢？

清理一些空间

有时，在 Minecraft 世界中，找到足够的空间来建造你的建筑会有一点困难。在玩家的周围通常会有大量的树和山，这就意味着一般情况下周围没有足够的空间来建造大型建筑。你可以通过编写一个程序来解决这个问题，清理出更多的空间来建造任何你想要的东西。

调用 setBlocks 函数进行更快速的建造

在 Minecraft 世界中，所有的方块都有它的 ID 编码，你面前的空的空间也不例外，它被称为 `block.AIR.id`。因此，如果你把一个大区域里所有的方块都设置成 `block.AIR.id`，这样就可以清除出一片巨大的空间，在附录 B 中列举了绝大多数你常用的方块（就是之前我们讨论过的表格）。比如，`block.AIR.id` 的值为 0 和 `block.STONE.id` 的值为 1。记住，在 Minecraft 世界中你所见到的“空”的地方就是一个方块 ID 为 `block.AIR.id` 的方块，它占 1 个方块的大小，并且充满空气。

计算机运行是非常快的，但是你在 `for` 循环中使用的语句越多，你的程序运行得就越慢。设置一个方块的指令，修改方块类型的指令以及更新屏幕上的打印内容的指令，这些指令 Minecraft 游戏都需要花费一些时间来解析。你可以编写一个大的循环来把你面前数百个方块都设置成 `block.AIR.id` 来清理一些空间，这样是可以的，但是这真的很慢。幸运的是，有个更好的方法可以一次性设置很多方块，它的名字叫 `setBlocks()`（注意这个名字最后的 s），`setBlock()` 设置一个方块，`setBlocks()` 可以一次性设置一个 3D 区域内的所有方块。

Minecraft 程序接口中有一个 `setBlocks()` 语句，它可以给一个 3D 区域中所有的方块设置一个相同的方块 id。因为在 Minecraft 游戏中这只是一个简单的请求，所以 Minecraft 可以优化它的执行，这样它将比你在上文中的 `for` 循环中使用 `setBlock()` 函数快很多。

因为 `setBlocks()` 适用于一个 3D 区域，因此它需要两组坐标，一个是 3D 空间的一个角坐标，另一个是它的对角的坐标。因为每个 3D 坐标都有 3 个数字，因此你一共需要 6 个数字来向 Minecraft 描述一个 3D 的空间。

你现在将会编写一段实用的程序，你可以在任何时候使用它来清理出一片空间，然后就可以随意在这片空间里建造东西了。

1. 新建一个程序，选择“菜单”里 File⇨New File；
2. 保存这个程序，选择 File⇨Save As，命名为 `clearSpace.py`；
3. 导入你需要使用的模块：

```
import mcpi.minecraft as minecraft
import mcpi.block as block
```

4. 连接 Minecraft 游戏：

```
mc = minecraft.Minecraft.create()
```

5. 如果你想要清理玩家面前的一片空间，你需要使用相对坐标来实现这个功能。首先你要获取玩家所在的位置：

```
pos = mc.player.getTilePos()
```

6. 现在，清理出一片 50 × 50 × 50 的区域。这片区域的左下角坐标就是玩家所在的位置，右上角的坐标将分别距离 *x*、*y*、*z* 坐标 50 个方块的长度：

```
mc.setBlocks(pos.x, pos.y, pos.z, pos.x+50, pos.y+50, pos. ↩
  z+50, block.AIR.id)
```

7. 保存程序。

令人感到惊奇的是，你刚才所编写的程序，可以让你在 Minecraft 中的任何地方、任何时间清理出一片 50 × 50 × 50 的区域。现在运行这个程序，选择 Run⇨Run Module，然后所有的树、山以及其他任何附近的东西，都会从你的眼前消失。

现在去环游 Minecraft 世界，不断运行你的程序来清理出更多的空间。

从键盘获取输入

你可以做的另外一个事情是，让你的 `clearSpace.py` 程序可以轻易地调整所要清除出的这片空间的大小。这样，如果你要建立一个小的建筑，你只需要清理一小片地方，如果你知道你即将建立一个大建筑，那就先清理出一大片区域。

为了实现这个功能，你需要使用一个新的 Python 语句，叫作 `raw_input()`，它可以从键盘上读入一个数字。然后你可以前往 Minecraft 世界中任何一个地方，运行你的程序，然后输入一个数字，它代表你想要清理的空间的大小，你的程序就会为你清理出这片区域。这意味着当你想在 Minecraft 世界中清理不同大小的区域时，你不需要每次都修改你的程序。

Python 语言有两个主要的版本——Python2 和 Python3。Minecraft 程序接口是用 Python2 来实现的，所以你需要在本书的冒险中使用 Python2。使用 Python3 作为 Minecraft Python 语言也是可以的，但是你需要更新你的 mcpi 模组。如果你曾经在网上看到过 Python3 版本的程序，它们可能使用 `input()` 而不是 `raw_input()` 来从键盘读入内容。在编写的时候，最新、最精确的版本号是 2.7.6 和 3.3.3。在 Python2 和 Python3 之间存在重要的差别，这也就是为什么使用 Python2 实现本书中的例子是很重要的。

现在你已经了解了 `raw_input()` 能做什么，那么该是扩展你程序的时候了。为了实现这个功能，你仅仅需要在你的 `clearSpace.py` 程序中修改两处地方，参照以下步骤：

1. 首先，使用 `raw_input()` 让用户输入他要清理的空间的大小，然后把这个数值存储到变量 `size` 中。就像冒险 2 中你使用 `str()` 把一些内容转化为字符串，这里你使用 `int()` 来把从 `raw_input()` 中读入的字符串转化为数值，这样你就可以对这个值进行一些运算。尝试一下如果没有使用 `int()` 函数会发生什么事情，现在在代码中添加下面标记出来的那一行新内容：

```
pos = mc.player.getTilePos()
size = int(raw_input("size of area to clear? "))
```

2. 接下来，修改 `setBlocks()` 那一行，这样他就不会使用数值 50，而是使用你的新变量 `size`：

```
mc.setBlocks(pos.x, pos.y, pos.z, pos.x+size, pos.y+size, ↩
  pos.z+size, block.AIR.id)
```

3. 保存你的程序，然后从“菜单”里选择 Run⇨Run Module 运行它。

现在，在 Minecraft 世界中，把游戏人物移动到一个有很多树和山地方，然后在 Python Shell 里输入一个数值，可以是 100 或者其他数字，然后按下回车键。所有你周围的树和山都会神奇般地消失，如图 3-4 所示，你将会拥有一个非常好的建造建筑的地方。

保留这个小而实用的程序，因为如果你以后要在 Minecraft 世界中清理出一片空间，它会非常有用。

图 3-4　在运行 `clearSpace.py` 程序以后，注意有多少树叶从树干上被锯下来了

挑战

`clearSpace.py` 程序在玩家周围清理出了一片立方体形状的空间，它使用玩家的位置作为这个空间的左下角坐标，但是如果玩家背对着这片空间，他将无法第一时间发现这片空间。如果它能以玩家所在的位置为中心清理出一片空间，它会变得更加方便。如果是这样，你需要修改 `setBlocks()` 语句中的相对坐标，然后计算出这个 3D 框架的两个角的坐标，将玩家置于这个空间的中间位置。同样，你也需要通过减少 *y* 坐标让 `clearSpace.py` 能够向下挖掘一些方块，让你能够拥有一些空间来放置地基、草地或者其他土地方块。尝试一下，看看你能不能研究出怎么实现这个功能，这会让这个程序变得更有用。

建造一个房子

当你处于 Minecraft 的生存模式中，你需要做的第一件事情就是建立一个庇护所来保护自己，这样可以避免在游戏中天黑后的各种潜在的威胁。试问，如何仅仅按一个键就可以建一个房子呢？幸运的是，当你在进行 Minecraft 编程的时候，你可以把复杂的任务简化为——仅按一个键。

在 `clearSpace.py` 程序中，你学到了怎么通过仅仅使用一行程序语句，把大量的方块设置成同一个类型。现在你将会学到如何使用相同的技术来快速建造一个房子。

有一种建造房子的方法，就是使用 `setBlocks()` 建造每一面墙，这样，你就需要使用四次独立的 `setBlocks()` 函数，它需要进行大量的数学计算，得出每面墙的每个角的坐标。幸运的是，我们有一

个更快的方法来建造空心的立方体建筑，你要做的事情是，建造一个巨大的立方体空间，然后再次使用 `setBlocks()`，通过对坐标的微调把里面的实体方块变成空气。

在动手之前，你需要花一些时间来了解你即将建造的房子的数学模型，因为你必须在程序中使用正确的坐标。图 3-5 显示了一个房子的设计草图，标记出了所有在建造过程是可能使用到的重要的坐标。当你需要建造一个复杂的建筑时，在草稿纸上画出草图然后计算出重要的坐标是非常重要的。这是一个非常特殊的房子，因为我们设计并计算了它的大小，以及门和窗户的位置。在后面的冒险中你就知道为什么它这么重要了。

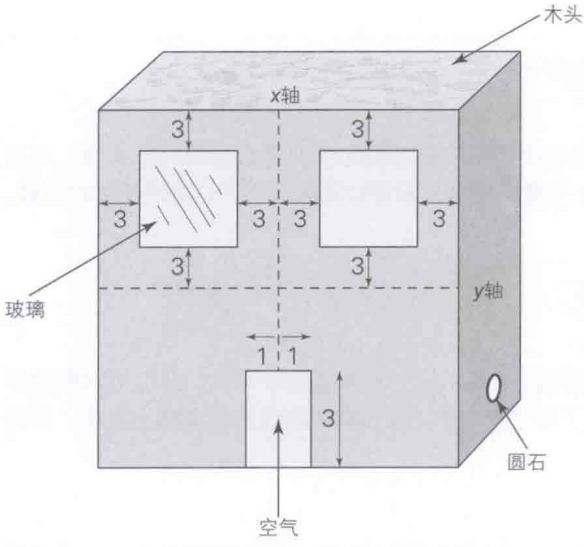


图 3-5　一个房子的设计草图，它包含了所有重要的坐标

如果你需要观看关于怎么建造房子的辅导视频，访问 www.wiley.com/gp/adventuresinminecraft 了解更多关于冒险 3 的内容。

现在你已经设计好了你的房子，那么按照下面的步骤尝试建造它：

1. 在“菜单”中选择 File⇨New FIle，新建一个程序。
2. 在“菜单”中选择 File⇨Save As，保存这个程序，并命名为 `buildHouse.py`。
3. 导入需要的模块：

```
import mcpi.minecraft as minecraft
import mcpi.block as block
```

4. 连接 Minecraft 游戏：

```
mc = minecraft.Minecraft.create()
```

5. 使用一个常量来存储你房子的大小。通常的话，在你建造房子的代码中，我们使用常量 `SIZE`。在

这里使用一个常量，而不是使用一个数值 20，可以让你以后修改你的房子的大小变得更加简单：

```
SIZE = 20
```

记住，把常量定义放在程序的顶端是程序员的惯例，在 Python 语言里，并没有强行规定一定要这样做，但是这种习惯能帮助你记住这是一个常量，并且你不想让它在程序运行的过程中被改变。

6. 获取玩家的位置，这样就可以在玩家附件建造房子：

```
pos = mc.player.getTilePos()
```

7. 把 *x*、*y* 和 *z* 的坐标存储在新的变量中，这会让接下来的程序语句变得更容易编写和阅读，并且对你的房子以后做一些更有创造力的改造是有帮助的。变量 *x* 将会被设置成距离玩家 2 个方块长度的位置，这样房子就不会建造在玩家的正右方：

```
x = pos.x+2
y = pos.y
z = pos.z
```

8. 计算两个变量 midx 和 midy，它们代表了你的房子在 *x* 和 *y* 方向上的中间点。它们可以帮助你更加容易地计算出窗户和门的坐标，因此如果你修改了房子的大小，窗户和门的位置也会随之变化，显得更加合适：

```
midx = x + SIZE/2
midy = y + SIZE/2
```

9. 建造一个巨大的 3D 空间作为你房子的外壳，你可以选择任意的方块类型，但是圆石是一个很好的选择，这让它看起来更加古老：

```
mc.setBlocks(x, y, z, x+SIZE, y+SIZE, z+SIZE,  ↩
  block.COBBLESTONE.id)
```

10. 现在挖空房子的内部，让它充满空气。记住你在设计阶段时用来计算房子内部空气的坐标方法，它们是房子外壳四个角的相对坐标：

```
mc.setBlocks(x+1, y, z+1, x+SIZE-2, y+SIZE-1, z+SIZE-2, ↩
  block.AIR.id)
```

11. 同样用空气（AIR）类型的方块在墙上挖空一个位置作为门，你不会使用一个 Minecraft 中的门，因为这是一个巨大的房子。相反，你会建造一个 3 个方块长度高 2 个方块长度宽的巨大的门，你的门需要处于房子正面的中间，而 midx 就代表了 *x* 坐标的中间位置：

```
mc.setBlocks(midx-1, y, z, midx+1, y+3, z, block.AIR.id)
```

12. 用玻璃（GLASS）类型的方块在墙上挖出两个窗户。因为这是一个巨大的房子，你需要在距离房子外侧 3 个方块长度，距离中心点 3 个方块长度的位置建造窗户。如果你改变了你的常量 SIZE 的值，并且重新运行这个程序，这个程序中所有的值都会重新计算，门和窗户也会出现在合适的地方，这样它看起来仍然像一个房子：

```
mc.setBlocks(x+3, y+SIZE-3, z, midx-3, midy+3, z, ↩
  block.GLASS.id)
mc.setBlocks(midx+3, y+SIZE-3, z, x+SIZE-3, midy+3, z, ↩
  block.GLASS.id)
```

13. 增加一个木制的屋顶：

```
mc.setBlocks(x, y+SIZE-1, z, x+SIZE, y+SIZE-1, z+SIZE, ↩
  block.WOOD.id)
```

14. 现在，增加一个羊毛地毯：

```
mc.setBlocks(x+1, y-1, z+1, x+SIZE-2, y-1, z+SIZE-2, ↩
  block.WOOL.id, 14)
```

在 `setBlocks()` 末尾你看到了一个额外的数字，这个数字代表了地毯的颜色，在这个例子中，数值 14 代表的红色，在接下来的“深入代码”栏目中，这个数字会得到详细的解释。

保存你的程序，然后在 Minecraft 世界中把人物移动到一个相对空旷的位置，通过在“菜单”中选择 Run⇨Run Module 来运行你的程序。你将会看到你的房子奇迹般地出现在你的眼前。走进去并且探索它，不管是站在屋顶上看，或者从窗户向里面看，你都会惊奇于能够这么迅速按照草稿建造了这个房子（见图 3-6）。

图 3-6　这个房子是用 Python 程序自动建造的

不要忘记 `buildHouse.py` 总是根据你的玩家所在的位置来建造房子。在 Minecraft 世界中移动，然后重新运行 `buildHouse.py` 来建造另外一个房子，你可以在 Minecraft 中任何地方建造你的房子，这是多么酷的事情！

深入代码

当你放置你的地毯，在 `setBlocks()` 中你使用了一个额外的数值：

```
mc.setBlocks(x1, y1, z1, x2, y2, z2, block.WOOL.id,14)
```

让我们深入探讨一下，然后研究出它表示什么？

羊毛（`WOOL`）是一个非常有趣的方块类型，因为它包含了“附加数据”（extra data），这意味着你不仅可以把一个方块变成羊毛，也可以使用附加数据来改变羊毛的外貌。在 Minecraft 世界中，有很多其他的方块也包含附加数据，并且每个方块都有不同的使用方法。对于羊毛来说，这个“附加数据”代表了羊毛的颜色，这个有用的特征使得羊毛成为一个多功能的方块，因为你可以在调色板中选择任何你想要的颜色，让你能够把你的设计变得更加富有颜色，并且更加真实。

在本次冒险的末尾，你可以找到一张表格，它列举了很多数字和它们对应的羊毛的颜色。附录 B 同样列举了很多其他类型的方块，它们都拥有可以让你改变的附加数据。

如果你在互联网上看到了其他的 Minecraft 程序，你可能会疑惑为什么在本书中，方块总是被表示为 `block.WOOL.id`，为什么总是把 `.id` 放在末尾。为什么网上的其他程序不是这么做的？这是因为当你的额外的属性被使用的时候，`setBlock()` 和 `setBlocks()` 有时不会工作，此时你只能把 `.id` 放在方块的末尾。因此，为了保持所有的东西简单且一致，我们决定总是使用 `.id`，这样每次你使用它的时候，它看起来都是一致的。你并不是一定需要把 `.id` 放在末尾的，但是如果你这么做，你就不需要去记住你何时需要这么做，何时又不需要这么做。

尝试把你程序中的常量 `SIZE` 调整为一个更大的数值，比如说 50，然后再次运行这个程序。如果你这么做了，你的房子会变成什么样？然后改常量 `SIZE` 为一个小的数值，比如 10，现在房子又会变成什么样呢？想一想为什么当常量 `SIZE` 变小之后，房子会变成这个样子。

建造更多的房子

建造一个房子是非常有趣的，但为什么要止步于此呢？回想一下本次冒险的前半部分里编写的 `tower.py` 程序，通过 `for` 循环你可以很简单地把一个程序语句运行固定的次数。因此，建造一个街道的房子，甚至一个小镇，是一定有可能的，只要重复运行建造房子的程序就可以了。

在你动手之前，你需要优化一下建造房子的代码，确保它不会变得太大、太复杂，以至于不容易管理。

调用 Python 函数

接下来你要做的一件事是建造一个包含许多不同设计的房屋的城镇。用于建造城镇的程序会变得非常

大而复杂，幸运的是，Python 语言里有一个特性，它可以帮助你把复杂的代码封装到一个个小的程序块里，并且可以重复使用。它被称为“函数”（function）。

通过 Python 函数（function），你可以把一些相关的程序语句整合在一起，并对它们命名。当你想运行那些作为一个整体的程序语句，你只要通过调用它们的名字，然后加上一对括号即可。

你可能还不理解它，但你已经通过本书使用了各种类型的函数，早到你在第一次冒险中运用 `mc.postToChat("hello")` 发送信息给 Minecraft。`PostToChat()` 是一个函数，它包含了一些相关的程序语句，你可以在你的程序中使用 `PostToChat()` 来调用它。

在你准备把你的 `buildhouse.py` 程序改写成函数形式之前，在 Python Shell 中试用一些函数，确保你理解了它们：

1. 单击 Python Shell 窗口，把它置顶。
2. 输入下面的代码，它将定义一个名叫 `myname` 的新函数：

```
def myname():
```

`def` 的意思是“定义一个新函数并给它命名”。就像早期 `while`、`if` 和 `for` 语句，你必须在句尾添加一个冒号（:），这样 Python 会知道你会提供额外程序语句作为函数体。

3. Python Shell 会自动为你缩进下一行，因为 Python 知道它们是函数的一部分。输入一些新的 `print` 语句，它们会打印出你的名字和信息。不用惊讶，当你输入每一行时什么都不会发生，这是正常的。耐心点，一会儿一切都会清楚：

```
  print("my name is David")
  print("I am a computer programmer")
  print("I love Minecraft programming")
```

4. 最后，连续按两下键盘上的回车键，Python Shell 能识别出你已经输入结束，同时终止语句缩进。
5. 现在通过输入函数的名字，并在它后面加上括号，让 Python Shell 调用它：

```
myname()
```

6. 尝试多次输入 `myname()`，然后观察会发生什么。

开始可能有点奇怪，因为当你在 Python Shell 中输入指令后，什么都没发生。一般情况下，当你在 Python Shell 中输入内容，一按回车键后，就会有情况发生，但是为什么这次什么都没发生？因为这一次事情有点不一样，你“定义”一个新函数并命名为 `myname`，然后要求 Python 记住三个属于这个函数的 `print` 语句。Python 会把这些 `print` 语句存储在计算机内存里，而不是直接执行它们。现在，无论何时你输入 `myname()`，它都会执行存储中的语句，然后把三行文本打印到屏幕上。

函数是非常强大的，它允许你整合任意数量的 Python 程序语句到一个简单的名字里，有点像一个迷你程序。无论何时你想要执行这些程序语句，你只需输入函数的名字。

让我们好好利用一下这个特性，定义一个函数，让它为你建造一个房子。然后，无论何时你想要一个由圆石制成的房子，你只需要输入 house()，它会自动为你建造。

1. 你不用中断正在运行的 buildhouse.py 程序，从“编辑菜单”中选择 File⇨Save As，然后把新的程序命名为 buildHouse2.py；

2. 在程序的顶端，import 语句之后，定义一个新的函数，命名为 house：

```
def house():
```

3. 现在移动 midx、midy 和 setblocks() 语句，把它们都缩进到 def house(): 下面。下面就是你的程序现在应该看起来的样子。要确保缩进是正确的：

```
import mcpi.minecraft as minecraft
import mcpi.block as block

mc = minecraft.Minecraft.create()

SIZE = 20

def house():
  midx = x + SIZE/2
  midy = y + SIZE/2
  mc.setBlocks(x, y, z, x+SIZE, y+SIZE, z+SIZE, ↩
  block.COBBLESTONE.id)
  mc.setBlocks(x+1, y+1, z+1, x+SIZE-2, y+SIZE-2, ↩
     z+SIZE-2, block.AIR.id)
  mc.setBlocks(x+3, y+SIZE-3, z, midx-3, midy+3, z, ↩
  block.GLASS.id)
  mc.setBlocks(midx+3, y+SIZE-3, z, x+SIZE-3, midy-3, z, ↩
     block.GLASS.id)
  mc.setBlocks(x, y+SIZE, z, x+SIZE, y+SIZE, z+SIZE, ↩
     block.SLATE.id)
  mc.setBlocks(x+1, y+1, z+1, x+SIZE-1, y+1, z+SIZE-1, ↩
     block.WOOL.id, 7)

pos = mc.player.getTilePos()
x = pos.x
y = pos.y
z = pos.z

house()
```

4. 注意程序代码的最后一行，你已经添加了代码 house()。这行程序将会调用已经存储在计算机内存里，通过 def house(): 语言定义的代码。

5. 保存程序，把人物移动到一个新的位置然后运行程序。你会发现有一个房子出现在你的面前。

你可能在想着“然后呢？”。你修改了你的程序，把一些语句转移了位置，但是它仍然在做着同样的事。有时，在编写程序的时候，在你能够用现有的代码做一些令人惊奇的事情之前，你有必要先改进程序的框架和结构。这就是你刚才做的事情。你已经稍微调整了你的程序，这样我们可以更加容易地重复使用你建造的房子的代码。

挑战

这个 `house()` 函数总是根据保存在变量 `x`、`y` 和 `z` 中的数值来建造房子。在你的程序末尾添加一些额外的语句来改变变量 `x`、`y` 和 `z` 的值，然后添加另一个 `house()` 语句，之后看看会发生什么。你觉得你能用这种方法建造多少个房子？

深入代码

在你的新程序里，通过把所有 `setBlocks()` 中的语句添加到 `house()` 函数中，就会发生一件神奇的事情。

变量 `x`、`y` 和 `z` 和常量的 `SIZE` 都被称为“全局（global）变量”。它们是全局的，因为它们在主程序中（非缩进的部分）被赋值了。因为它们是全局的，意味着你能在程序的任何地方调用它们，包括在 `house()` 函数里。所以，如果你做过一些实验，想要在主要程序里通过改变 `x`、`y` 和 `z` 的值建造一些不一样的房子，因为这些变量是全局的并且能在任意地方被调用，所以这样是可行的。

在冒险 6 中，你会知道全局变量在一个大程序中出错的时候，变得很难调试。有一个更好的方法让函数与函数共享信息。然而现在，使用全局变量已经足够了，因为你的程序很小，它不会出现问题。

一个全局（global）变量是一个能在程序中的任何地方被调用的变量。任何变量（或者常量），如果它们定义的时候没有左缩进，它们都是全局变量，你能在程序的任何地方访问它。因此，如果你用 `a=1`，并且没有左缩进，那这个变量就能在程序中任何地方访问。

然而，所有在定义的时候左缩进的变量都不是全局变量（它们通常被称为“局部变量”）。所以，如果你在 def 语句下面（也就是在函数里面），在缩进区域里面定义 `a=1`，那这个变量仅能在这个函数的缩进区域里访问，其他地方将不能被访问。关于全局变量你还有很多要学的，但目前为止你只需要了解这么多。

用一个 for 循环建造一条满是房子的街道

现在你将要用你在这次冒险中学到的所有东西来建造一个满是房子的街道。如果你手动从物品栏中选择方块来建造所有的房子，这需要花费你数小时的时间。在建造过程中，你也可能偶尔犯错误，把它们建的或小或大。通过 Minecraft 编程，你可以自动化建造房子和其他建筑，并且会做得非常迅速和精确。

为了建造大量的房子，你需要给你的程序添加一个 `for` 循环，如下：

1. 你不用删除你已存在的 `buildhouse2.py` 程序，从“菜单”中选择 File⇨Save As，然后将新的程序命名为 `buildhouse2.py`。

2. 在程序的末尾，在 `house()` 上一行增加一个 `for` 循环，同时改变房子建造时所使用的 `x` 的位置。这样每个房子与房子之间的距离都是 `SIZE` 大小。新增如下加粗的代码：

```
for h in range(5):
  house()
  x = x + SIZE
```

保存你的程序，然后在 Minecraft 世界中，把人物移动到一个相对空旷的地方，并运行这个程序。见图 3-7，在你的视野之内出现了一个包含了 5 所房子的街区！操作人物，进入每一个房子，检查每一所房子是否和预想中的一样。

图 3-7　一条用 Python 程序自动建成的包含 5 个一样的房子的街区

挑战

想一下，如果你不是在一条线上建造房子，而是要在竖直方向上建造一个巨大的塔。尝试一下。把你用于建造巨塔的代码放入函数 `tower()` 中，然后使用一个循环，在 Minecraft 世界中建造更多的塔。

当你建造你的房子时，会受限于玩家所在的位置和周围的地势，例如树和山。一些有趣的事情会发生在你的房子上。有一些房子可能吃掉了半棵树，或者镶嵌在山里面。如果你的房子建在了海上，它可能会被淹没。因此，你可能需要修改你的 `house()` 函数，在房子的下面建造一层基岩，这样可以保证房子总是被建造在实地上。

添加随机地毯

到目前为止，你已经在 Minecraft 世界中拥有了大量的房子，这看起来非常了不起。你仅仅使用了短小的 Python 代码就建造了一些大的工程结构。

然而，你可能开始觉得一条街的房子看起来一模一样，会有点无聊。那么，你可以做些什么细微的调整，使你每个房子看起来有些差别？

一种方法就是写一些和 `house()` 不一样的函数，如 `cottage()`、`townhouse()` 和 `maisonette()`，然后修改你的程序，让它通过调用不同的函数给你的街道增加一些多样性。

另一个使你的建筑变得更有趣的方法就是改变它们的部分特征，比如说地毯，让它在你每次运行程序的时候都不一样。如果你使用这个方法，即使是程序员也不会知道它到底会建造什么，除非他把每个房子都探索一遍。

生成随机数

计算机是十分精准的机器。在日常生活中的每个方面，我们都依赖计算机的可预见性，每个程序总是在做着同样的事情。当你给你的银行账户存入 10 元，你需要确认 10 元已经被存入计算机中代表你账户余额的存储模块中，每次如此，不会出错。因此在如此精确的系统中随机的概念可能看起来不太寻常。

然而，随机在游戏设计领域是非常重要的。如果游戏每次玩起来都一样，那么它就会显得简单——没有乐趣或挑战。几乎每个你玩的计算机游戏都会包含一些随机性来使游戏每次玩起来都不一样，这样能吸引你的注意力。

幸运的是，Python 语言有一个内嵌的模块能给你生成随机数（random numbers），因而你不用自己写编码来实现它，只需要用这个嵌入的随机数生成器即可。

为使你了解这些随机数看起来是怎样的，在 Python Shell 中输入如下内容：

1. 单击 Python Shell 窗口，把它置顶。
2. 导入随机模块，这样你就可以使用内嵌的随机函数，输入：

```
import random
```

3. 让程序生成一个 1 到 100 之间的随机数，并且打印到屏幕上：

```
print(random.randint(1,100))
```

你可以看到一个 1 到 100 之间的数字出现在屏幕上。再次输入 `print` 语句，这次会出现什么数字？

4. 现在使用一个 `for` 循环来打印大量的随机数。确保第二行缩进，这样 Python 才会知道 `print` 语句是 `for` 循环的一部分：

```
for n in range(50):
  print(random.randint(1,100))
```

在 randint() 函数括号里的两个数表示你希望产生随机数的范围；1 是你希望产生的最小的数，100 则是最大值。

一个随机数（random number）通常产生于一个随机数列——不包含任何明显的模式或重复序列的数字序列。

计算机是非常精确的机器，并不会产生真正的随机数字；事实上，他们产生的是伪随机数字。这些数字看起来是随机数列的一部分，但是它们确实是有规律的。你可以访问 http://en.wikipedia.org/wiki/Random_number_generation，或者访问 www.random.org 以了解更多关于真正随机数字的内容。

放置地毯

在本次冒险的前面，你在 setblocks() 函数使用了一个额外的数字使羊毛地毯显示为红色，这个额外的数字被设置为羊毛方块的附加数据。羊毛方块的附加数据的范围是 0 ~ 15——换句话说，有 16 种颜色你可以选择。现在通过如下步骤来修改建造房子的程序，让它生成随机的数字，并且把它们作为羊毛的颜色，也就是房子里地毯的颜色：

1. 选择“菜单”上的 File⇨Save As 来保存你的 buildStreet.py，把它重新命名为 buildstreet2.py；
2. 在程序的上方增加这个导入语句来获得权限使用随机数生成器：

```
import random
```

3. 修改你的 house() 函数，使它在每次使用时生成一个随机的地毯颜色。因为羊毛方块可以使用一个 0 ~ 15 的附加数据，所以这就是你要产生的随机数的范围。把这个随机数存入变量 c 中，这样这个地毯颜色生成语句就不会变得过长、难以阅读。确保这些语句都被正确的缩进了，因为它们都是 house() 函数的部分：

```
  c = random.randint(0, 15)
  mc.setBlocks(x+1, y+1, z+1, x+SIZE-1, y+1, z+SIZE-1, ↩
    block.WOOL.id, c)
```

4. 保存你的程序。

现在你需要在你的 Minecraft 世界中，找到一个有足够空间的地方，这样可以建造更多的房子。移动你的人物，找到一个合适的地方，然后运行你的新程序。它能建造另一个街道，但这次，当你探索完每个房子内部之后，你会发现每一个房子的毛毯都是不一样的，是一个随机的颜色。在图 3-8 中你可以看到这些新房子的样子，但你需要真正进入每一个房子并检查所有的地毯，确保它们是不是确实都不一样！

图 3-8　每个房子有一个随机颜色的地毯

挑战

定义一个 `house2()` 函数，让它建造一个不同类型的房子。在主要的 `for` 循环里使用随机数来建造其中一所房子，这样你建造的每所房子都将是随机的。扩展这个程序，让它建造三个不同类型的房子。你建造的房子类型越多，你的街道将变得越有趣。你甚至能用负的坐标来实验，如 `y=-5`，来建造一个地基或者在房子里建造一个游泳池。

很大程度上，在 Minecraft 世界中建造大型的建筑仅仅限制于你的想象力。当然，Minecraft 世界中有高度限制，它会限制你建造的建筑的最高高度。现实生活中的一些建筑物包含倾斜或弯曲的结构。比如说，碎片大厦的设计，它是目前英国最高的建筑。这个令人惊异的建筑也能在 Minecraft 世界建造，但因为它有斜棱，这会变得更加困难。

保持耐心，以一些简单的矩形建筑物作为起点。在冒险 7 里，Martin 将给你介绍一些其他更有帮助的函数，它们能够建造折线和曲线。之后，在 Minecraft 世界里，你所能建造的建筑将不再受到限制。

在 Minecraft 世界中，能建造多大的建筑还受限于两点——计算机内存的大小和人物的视距。越大的建筑需要越大的计算机内存，因此足够的计算机内存是很重要的。第二，玩家只能看到一定的距离，特别是在树莓派平台上，它只有很短的视距。

快速参考表	
导入并使用方块 ID 编号常量	**设置 / 改变某个位置的方块**
`import mcpi.block as block` `b = block.DIRT.id`	`mc.setBlock(5, 3, 2, block.DIRT.id)`
一次性设置 / 改变很多方块	**对于建造房子有用的方块类型**

`mc.setBlocks(0,0,0,5,5,5,block.DIRT.id)`	`block.AIR.id`
	`block.STONE.id`
	`block.COBBLESTONE.id`
	`block.GLASS.id`
	`block.WOOD.id`
	`block.WOOL.id`
	`block.SLATE.id`

所有在这次冒险里你学到的新 Python 和 Minecraft 语句都会列在附录 B 的参考章节里。

访问中文 MinecraftWiKi 百科：http://minecraft-zh.gamepedia.com 方块，了解完整的关于方块 ID 的信息列表。访问 http://minecraft-zh.gamepedia.com/ 数据值，了解更多关于 Minecraft 中方块的附加数据值。

当你在街道里的房子中放置随机的地毯的时候，羊毛是一个非常有用的方块类型。这里有个参考，它包含了方块类型 `block.WOOL.id` 能使用的所有的颜色。回头看看你的 `buildstreet2.py` 程序，了解怎样给 Minecraft API 提供额外的数据。

0 white	1 orange	2 magenta	3 light blue
4 yellow	5 lime	6 pink	7 grey
8 light grey	9 cyan	10 purple	11 blue
12 brown	13 green	14 red	15 black

在建造方面进一步冒险

在这次冒险中，你已经学到了在 Minecraft 世界里怎样使用一个单独的 Python 程序建造单独方块和一整个区域的方块。你已经建造了一些相对不错的建筑。你已经学会使用函数把你的程序分解成很小的逻辑单元，并且用 `for` 循环，不断重复再重复。有了这些知识，你可以建造几乎所有你能想象出的建筑。

- 使用在这次冒险中学到的技术，为骰子的每一个面写一个函数。用一个随机次数的循环，每次循环展示骰子不同的面。使用 `random.randint()` 来停止循环，然后你和你的小伙伴可以尝试并猜想下一个 Minecraft 骰子会停在哪里。

- 制作一个包含其他随机属性的街道的列表。你可以走在你自己的街道上，看看每个房子和其他的有什么不同之处。定义更多关于房子的函数，每个函数代表了一个房子，尝试建造更多拥有不同房子类型的更复杂的街道。
- 和朋友一起把你建造房子的程序加入到一个更大的程序中。使用它来建造一个拥有各种类型的房子的街区。当你走在你的新镇上，你会觉得非常惊奇，房屋的设计竟然有这么多种类！

解锁成就：在 Minecraft 中，卓越建筑物的设计师和令人惊讶的巨型建筑物的建造者！

在下一次冒险中……

在冒险 4 中，你将学如何检测并和方块交互，包括如何判断你所站的那个方块的类型，如何判断方块是否被玩家击中。你将使用所有新的知识一起去设计一个有趣的寻宝游戏！

冒险 4

与方块交互

一个让你的 Minecraft 之旅更有趣的方法就是，让这个游戏基于周围玩家的行为做出改变。当你在游戏的世界里移动时，你面对的选择都依赖于你之前的操作，使你每次玩游戏时游戏都会变得略有不同。Minecraft 的 API 能让你获取你脚下的方块类型，或者检测到你用剑击打的方块，以此来完成你与方块的交互行为。

在这次冒险中，你首先将会学习编写一个魔法桥梁程序，这是一个基本的方块交互程序。这个桥的特殊之处在于，当你在水上或者空中行走时，一座桥将会神奇地在你面前出现来保证你的安全。你的 Minecraft 世界很快就会充满了桥梁。然后，魔法桥梁建造者的第二版将会使用一个 Python 列表保存你建造过的桥梁，当你安全返回地面之后，所有的桥梁都会直接在你眼前消失！

最后，你将会学习如何检测到一个方块被击打的事件，并制作一个相当带劲的寻宝游戏。你要通过你的魔法桥梁来寻找和收集随机出现在天空中的宝藏，包括导航信标和分数功能。

你在这次冒险中会像一名真正的软件工程师那样，每次只编写一个程序的一个功能，最终将所有模块组合在一起，完成一个大型的程序。系好你的安全带，小心前方的提示，我们即将踏上一段激动人心的空中之旅！

检测你站在什么方块上

冒险 2 中已经提到，我们可以用 `getTilePos()` 读取玩家的坐标，这样就可以追踪玩家的位置。这里的坐标代表着玩家 Steve 当前所在位置的 *x*、*y* 和 *z* 坐标。你通过这个坐标就可以检测 Steve 是否站在一个魔毯上，或者通过区域限定的方式，检测他是否在某一个特定区域。

然而，除非你的程序自己维护一个详细地图，记录世界里每一个坐标的具体方块类型，否则仅仅通过坐标来检测是很不灵活的方法，因为这会使你的程序变得非常复杂。你可能会一直自己维护这个细节地图，

但是，Minecraft 为了在屏幕上显示这个三维世界，一定已经在计算机内存里存储了所有的地形信息。既然如此，我们为何还要自寻烦恼呢？

幸运的是，Minecraft 的 API 同样包括了一个 `getBlock()` 函数。这个函数提供了访问 Minecraft 内存里地图信息的全部权限。通过坐标，你可以知道所有方块的信息，不仅仅是 Steve 所在位置的方块，而是整个世界里的任意方块。

在冒险 3 中我们提到，可以通过方块类型来改变 Minecraft 世界里的任何方块。同样很幸运，`setBlock()` 和 `getBlock()` 是一对函数，所以如果你用 `setBlock()` 设置了一个方块的 id，然后立刻用 `getBlock()` 来获取它，你将会获取到相同的方块 id。

你很快就要在 Minecraft 里编写另一个令人兴奋的游戏了，但是你编写出来的程序会有些庞大。编写一个庞大程序的最好方法就是先编写一个个小规模的程序，在确定这些小程序能够运行之后，将所有小程序组合起来。我们先从一个简单的程序开始学习这种思想，这个程序会告诉你玩家是站在安全的位置还是不安全的位置。

检测你是否站在地面上

启动 Minecraft 与 IDLE，如果你是在 PC 或者 Mac 上运行，也打开服务端。你之前应该已经有了运行它们的经验，如果有问题可以参考冒险 1。你即将编写一个告诉玩家位置的重要信息的程序。在你能建立空中魔法桥梁之前，你需要检查玩家的脚是否站在地面上。

1. 单击 File⇨New File 新建一个窗口，把这个新程序保存为 `safeFeet.py`，记住将这个程序保存在 `MyAdventures` 目录里，否则程序可能不工作。

2. 导入必需的模块，你需要导入一个标准的 `Minecraft` 模块。并且，由于你需要和方块交互，所以也要导入 `block` 模块：

```
import mcpi.minecraft as minecraft
import mcpi.block as block
import time
```

3. 连接到 Minecraft 游戏：

```
mc = minecraft.Minecraft.create()
```

4. 因为现在这个程序将来会是另一个较大程序的一部分，所以需要把所有代码写在一个 `safeFeet()` 函数里，这样会使重用这段代码变得更简单。这个函数做的第一件事是，获取玩家 Steve 的位置：

```
def safeFeet():
  pos = mc.player.getTilePos()
```

5. `getBlock()` 能获取到你提供的坐标处方块的 id，因为 `pos.x`、`pos.y` 和 `pos.z` 是玩家的坐标，所以你必须用 `pos.y-1` 作为玩家脚下的方块坐标：

```
  b = mc.getBlock(pos.x, pos.y-1, pos.z)
  # 注意 : pos.y-1 很重要
```

6. 现在要用一个简单的方法实现检测玩家是否在安全的位置，如果玩家站在空气上或水上，就是“不安全”的，否则就是“安全”的。显然，检查玩家是否“不安全”更简单，因为如果判断玩家是否“安全”，你就要检查上百种方块的情况，这样代码就会很长，而判断不安全就只要一行代码：

```
  if b == block.AIR.id or b == block.WATER_STATIONARY.id ↩
or b == block.WATER_FLOWING.id:
```

7. 如果方块是不安全的，就在Minecraft聊天窗口发送一条告诉玩家不安全的消息，否则就是安全的。

> 请保证你的程序代码缩进都正确，否则程序将无法工作。所有函数内的语句都要缩进一级，而 `if` 和 `else` 内的语句要再缩进一级。`if` 和 `else` 语句则有相同的缩进，因为它们互相关联并在逻辑上处于同级。
>
> 
>

```
    mc.postToChat("not safe")
  else:
    mc.postToChat("safe")
```

8. `safeFeet()` 函数已经完成，在后面留一个空行来表示函数结尾，然后开始一个没有任何缩进的 `while` 循环，一开始你可以加入一个延时，以后的版本会去掉这个延时：

```
while True:
  time.sleep(0.5)
  safeFeet()
```

单击 File⇨Save 将程序保存。然后在编辑器的“菜单”单击 Run⇨Run Module。

此时玩家在游戏里移动会发生什么？你应该在聊天窗口看见 safe 和 not safe，见图 4-1，你可以飞到空中或者在海里游泳，看看会发生什么。

图 4-1　当你在空中时，safeFeet 程序会提醒你不安全

在你的 safeFeet 程序中，和其他程序一样，你在主循环中加入了一个短暂的延时。延时在程序里一般不是必需的，稍后你会看到程序会被延时减慢速度，以至于不能正常工作。但是在向 Minecraft 聊天窗口发送消息时，慢一点会更好，否则消息会滚动得太快。你可以减少延时的时间或者干脆去掉延时，看看会发生什么。

建造魔法桥梁

在之前的程序里，你实现了直接检查玩家脚下的方块，并且向玩家的聊天窗口发送消息，这是完成一个大程序的一小步。但现在我们看看是否可以再进一步，将我们在冒险 3 里学到的一些知识和 `setBlock()` 函数组合在一起。

只需要对 `safeFeet.py` 做一些小改动，它就可以变成一个魔法桥梁程序。玩家走路时，这个程序会在玩家的脚下放置玻璃桥梁，这样玩家永远不会掉到海里或者从空中摔下来了！在以后的冒险里，我们还会再次用到这个函数，所以你需要先把程序的名字修改正确。

1. 单击 File⇨SaveAs 并将 `safeFeet.py` 改成 `magicBridge.py`。
2. 将 `safeFeet()` 函数改成 `buildBridge()` 函数，然后修改下面加粗的 `if/else` 语句，将 `mc.postToChat()` 替换成 `mc.setBlock()`。这样，无论什么时候玩家处在“不安全”状态，这个函数都会在玩家脚下建立起魔法桥梁方块。注意要仔细检查 `if` 语句里的这一长行代码：

```
def buildBridge():
  pos = mc.player.getTilePos()
  b = mc.getBlock(pos.x, pos.y-1, pos.z)
  if b == block.AIR.id or b == block.WATER_FLOWING.id ↩
or b==block.WATER_STATIONARY.id:
    mc.setBlock(pos.x, pos.y-1, pos.z, block.GLASS.id)
```

3. 在第 2 步中可以看到，`else` 语句和 `mc.postToChat()` 已经被删除了，并且以后也不再需要了。
4. 如果程序放置方块的速度太慢，玩家就会掉落下去，所以现在你应该删掉主循环里的延时了，而且要确保你调用的是新的 `buildBridge()` 函数：

```
while True:
  buildBridge()
```

5. 在“编辑器菜单栏”单击 File⇨Save 保存程序。

运行程序，并在游戏里走一走，跳到天上或者水中。玩家无论在游戏里的何处，只要处于不安全的位置，一个魔法玻璃桥梁就会出现防止玩家摔落，如图 4-2 所示，现在玩家都学会水上漂了，真是奇迹！

这段程序在树莓派上能够很好地运行，但是在 PC 或者 Mac 上，程序的响应时间会比较慢，以至于玩家仍然会摔落，因为放置方块的速度不够快。但是别紧张，别在游戏里跑得太快，这个程序不是总能保证你的安全。为了程序工作得更好，你可以尝试更快或更慢的程序延时，或者你可以试试进入潜行模式（在移动时按住 Shift 键）让自己走得慢一点。

图 4-2　magicBridge.py 程序建立了一座玻璃桥梁使玩家能在水上行走

在 magicBridge.py 程序里，你删掉了主循环里的延时。如果再把这个延时添加回来会发生什么？体验一下不同延时的区别，你可以把延时设置为 0.1 ~ 2 秒的一个值，然后观察这会对你的魔法桥梁的可用性造成什么影响。看看哪个延时值是最好的。

深入代码

让我们休息一会儿，因为还有些关于函数的问题没有解释清楚。

在这本书的一开始，我们就使用函数，mc.postToChat() 是个函数，mc.getTilePos() 也是，它们都是 Minecraft API 的一部分。你也曾经用 def 关键字定义自己的函数，比如在冒险 2 里，你定义了一个 house() 函数。但是，当你在代码里使用函数的时候，究竟发生了什么事情？

之前，我一直在说“使用这个函数”，但是当你在代码里使用 house() 或者 buildBridge() 函数时，这个操作的正确说法应该是调用（call）函数。

在 Python 程序运行的过程中，计算机会记住当前运行到的语句，这有点像是有一根看不见的手指一直在指向正在运行的那行代码。一般情况下，这根手指会从顶部向底部移动。当你使用了循环时，这根手指会跳转到循环的开头，将循环里的代码重新执行一定次数。

当你调用函数时，额外还会发生一些神奇的事情。Python 记住你调用函数时那根看不见的手指的位置（你可以想象成在程序的那个位置贴了一张有颜色的标签），然后跳到定义函数的位置，比如 `buildBridge()` 函数。当它到达 `buildBridge()` 函数的结尾时，它就会跳回到之前贴标签的位置，然后继续执行下去。

出于很多原因，写程序时应用函数很有用，其中两个最重要的原因是：

- 你可以把一大段程序分割成小段程序
- 你可以在相同程序里的不同位置多次重新使用函数里的代码

由于这两个原因，函数使我们的程序更加易于阅读与修改。

当你调用（call）一个函数时，Python 会记住程序执行到的位置，并临时跳转到你用 `def` 关键字定义函数的地方。当函数执行完毕，Python 会跳回到之前跳转到函数的地方。

使用 Python 列表作为魔法存储器

在所有之前你写过的程序中，你用过某些形式的变量来存储在程序运行时改变的数据。每个变量有一个名字和一个值，如果你想存储多个数据，你只能使用多个变量。

然而，这些变量都只有确定的存储空间，并且也只能存储一个数据（比如一个数字或一个字符串）。作为 Minecraft 程序开发者，你以后可能会写很多程序，它们也许需要存储不确定个数的数据，你不会总知道你到底需要在程序里存储多少数据。

还好，作为一个现代编程语言，Python 有一个叫做列表（list）的特性，列表能在程序运行时存储可变数量的数据。

列表（list）是编程语言里的一种变量，它能存储任意数量的数据。你可以向列表添加新数据，计算列表里数据的数量，通过提供详细的位置来访问数据，从列表的任意位置删除数据以及很多其他的功能。列表是存储一系列相似数据的有用方式，比如需要被排序的一列数字、一个用户组里的成员，甚至是 Minecraft 里的一系列方块。

尝试使用列表

理解列表的最好方式就是在 Python Shell 里试验一下。

在这一节，请仔细查看括号。这里使用两种括号——圆括号 () 和方括号 [ ]。不过不必过于担心，当你学习完这一节，你应该能清楚分辨两种括号的区别。

1. 单击 Python Shell 窗口让它显示在前面，在末尾的 >>> 提示符右边单击鼠标，你将在这里输入你的交互命令。别忘了你必须先选择 Shell⇨Restart Shell 菜单项，或者同时按住 Ctrl 键和 C 键以停止正在运行的程序。因为如果你之前的程序仍在运行，之后的操作将不起作用！

2. 新建一个空列表，并显示列表里的内容：

```
a = [] # 一个空列表
print(a)
```

3. 向列表里添加一项元素，再显示列表里的内容：

```
a.append("hello")
print(a)
```

4. 向列表添加第二项元素，并再次显示内容：

```
a.append("minecraft")
print(a)
```

5. 查看列表的长度：

```
print(len(a))
```

6. 查看列表里特定位置的项目内容，这里的位置被称为“索引”：

```
print(a[0]) # the [0] 是列表中第一项的索引
print(a[1]) # the [1] 是列表中第二项的索引
```

7. 从列表末尾移除一个词，并显示这个词和列表的剩余部分：

```
word = a.pop()
print(word)
print(a)
```

8. 查看列表和这个词的长度：

```
print(len(a))
print(len(word))
```

9. 从列表里移除最后一项元素，显示这一项的内容和列表的内容，并显示列表的长度：

```
word = a.pop()
print(word)
print(a)
print(len(a))
```

图 4-3 展示了在 Python Shell 里试验时的输出结果。

索引（index）是指定要访问列表里哪一项的一个数字。索引要在方括号里指定，就像 `a[0]` 这样是表示列表里第一个数据，而 `a[1]` 表示第二个。Python 里的索引都是从 0 开始的，所以 0 总是列表里的第一位。在刚完成的步骤里，你“索引”了这个列表。

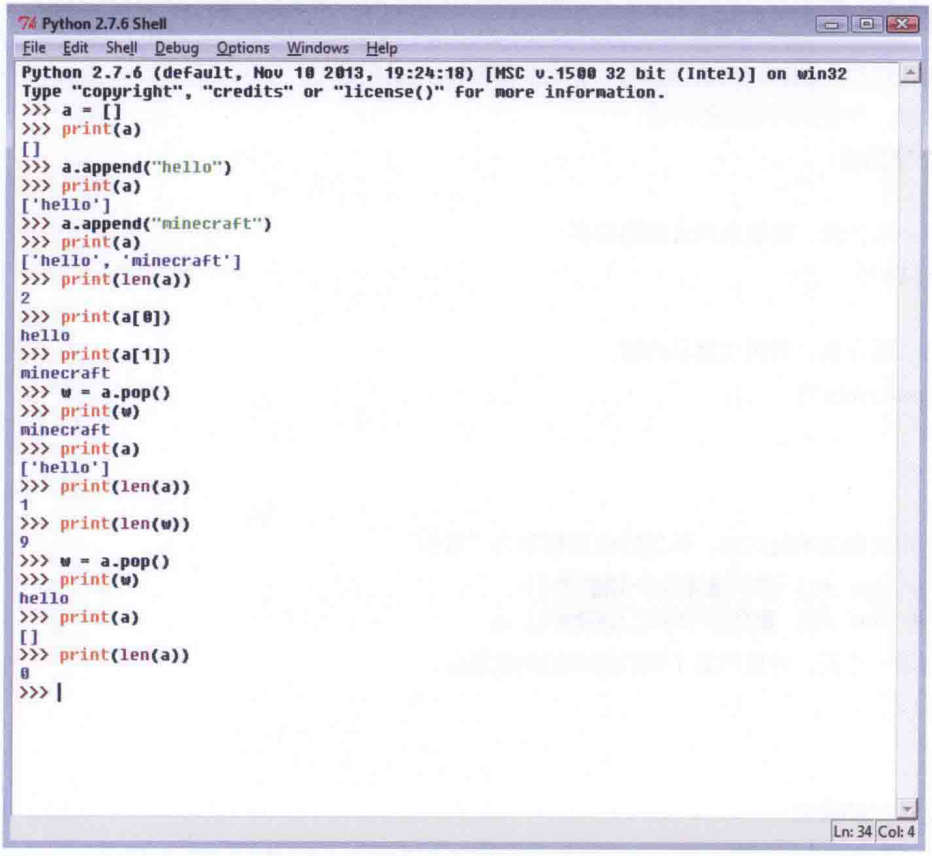


图 4-3　在 Python Shell 里试验时的输出结果

列表的长度会在你向列表添加项或者从列表移除项时改变，如果你试图访问一个列表里不存在的数据会发生什么？在 PythonShell 里试验这些语句，看看会发生什么：

```
b = []
print(b[26])
```

你会如何防止你的程序发生这样的情况？（注意这个问题不只有一个答案。你可以在互联网上做一些研究，找出能在程序里防止这个问题的不同方法。）

Python 列表是一个万能的变量类型，因为它可以存储任意类型的数据。但它不是 Python 里唯一能存储多种类型数据的变量类型。对你而言，列表最重要的一点是你不需要在程序运行之前知道列表的大小。列表会在你用 append() 添加项目或用 pop() 弹出项目时自动伸长或缩短。

从一个列表弹出（pop）项目，就是删除列表的最后一项。

用 Python 列表建造隐藏桥梁

现在你将要运用你学到的关于列表的新知识来编写一个桥梁建造程序。桥梁将会在玩家重新安全返回地面时隐藏。这个程序类似于 magicBridge.py，所以你可以把它保存为一个新名字并编辑这个文件，但是这里仍然会给出全部的程序，这样更容易解释每一个步骤。如果你想节省一点打字的时间，那就复制你的 magicBridge.py 程序并粘贴到新文件里。

如果你想观看关于“如何建立并参与隐藏桥梁的游戏”的视频，访问 www.wiley.com/go/adventuresinminecraft 并选择冒险 4 的视频。

1. 单击 File⇨New File 新建一个文件，在“编辑器菜单”里选择 File⇨Save As，保存为 vanishing Bridge.py。

2. 导入必需的库：

```
import mcpi.minecraft as minecraft
import mcpi.block as block
import time
```

3. 连接到 Minecraft 游戏：

```
mc = minecraft.Minecraft.create()
```

4. 创建一个空的桥梁列表。目前你的桥梁不包含任何方块，所以现在列表还是空的。

```
bridge = []
```

5. 定义一个 buildBridge() 函数用于建造桥梁。在本章最终的程序中，你将会使用这个函数，所以请保证函数的名字是正确的。函数开头的大多数代码与 magicBridge.py 是相同的，所以如果你想节

省一点时间，你可以将原来的代码复制过来。注意保持正确的缩进：

```
def buildBridge():
  pos = mc.player.getTilePos()
  b = mc.getBlock(pos.x, pos.y-1, pos.z)
  if b == block.AIR.id or b == block.WATER_FLOWING.id ↩
or b == block.WATER_STATIONARY.id:
    mc.setBlock(pos.x, pos.y-1, pos.z, block.GLASS.id)
```

6. 当建造桥梁的一部分时，你需要记录建造的方块的位置，这样方块才能在以后玩家回到地面时被移除。你需要另一个列表来记住方块的三维坐标，并将这个列表添加到桥梁列表。你可以阅读下面的“深入代码”栏目来获取比较完整的说明：

```
    coordinate = [pos.x, pos.y-1, pos.z]
    bridge.append(coordinate)
```

7. 为了能在玩家离开桥梁之后将桥梁隐藏，你需要一个 `else` 语句来判断玩家是否站在玻璃上。如果不在，程序就开始移除组成桥梁的方块。程序必须检查是否还有桥梁方块没有被移除，否则如果试图从一个空列表弹出元素，程序将会触发一个错误。`elif` 是 `else if` 的缩写，`!=` 的意思是不等于。注意这里代码的缩进：`elif` 缩进一级，因为它是 `buildBridge()` 函数的一部分，而下一个 `if` 语句缩进两级，因为它是 `elif` 语句的一部分：

```
  elif b != block.GLASS.id:
    if len(bridge) > 0:
```

8. 以下的代码需要缩进三级，因为这些代码属于上面的 if 语句的一部分，这个 if 语句是 elif 语句的一部分，这个 `elif` 语句又是 `buildBridge()` 函数的一部分！嗬！

还记得之前我们向桥梁列表里添加一个包含三维坐标的列表吗？在这里，你需要用 `coordinate[0]` 来表示 *x* 坐标，用 `coordinate[1]` 来表示 *y* 坐标，用 `coordinate[2]` 来表示 *z* 坐标。添加一句 `time.sleep()` 延时能让桥梁隐藏的速度慢一点，这样就可以看见隐藏的过程：

```
      coordinate = bridge.pop()
      mc.setBlock(coordinate[0], coordinate[1], ↩
      coordinate[2], block.AIR.id)
      time.sleep(0.25)
```

9. 最后，编写主游戏循环。在之前的试验中，你可能已经尝试过在游戏循环里添加不同时间的延时来增强程序的可用性。记住游戏循环是主程序的一部分（`while True`: 完全不缩进），所以还是要检查缩进是否正确：

```
while True:
  time.sleep(0.25)
  buildBridge()
```

单击 File⇨Save 保存程序，单击 Run⇨Run Module 运行程序。在游戏里走一走，试着跳跃到天上或者走到水中，然后返回到安全的地面上。会发生什么？图 4-4 展示了当 Steve 返回地面时，桥逐渐消失的场景。

图 4-4　当玩家返回地面时，桥逐渐消失

深入代码

之前，当你试验 Python 列表时，你仅仅将字符串添加到列表里。而在 `vanishingBridge.py` 程序里，还有一些关于 Python 的神奇特性需要解释一下。

Python 列表能存储任意类型的数据，例如字符串、数字，甚至是列表。大概你还没有意识到，其实你已经在程序里运用过这个特性了。

这行代码创建了一个包含三个元素的列表（方块的 *x*、*y*、*z* 坐标）：

```
coordinate = [pos.x,pos.y-1,pos.z]
```

通过将值写在方括号 `[ ]` 之间，你创建的是一个已经包含元素的列表，而不是一个空列表。这和以下代码的效果相同：

```
coordinate = []
coordinate.append(pos.x)
coordinate.append(pos.y-1)
coordinate.append(pos.z)
```

然后，当程序获取需要移除的方块的坐标时，桥梁列表里会弹出一个坐标列表：

```
coordinate = bridge.pop()
```

存储坐标的列表包含三个元素：`coordinate[0]` 是方块的 *x* 坐标，`coordinate[1]` 是方块的 *y* 坐标，`coordinate[2]` 是方块的 *z* 坐标。

图 4-5 展示了这个“列表中的列表”实际看起来的样子。

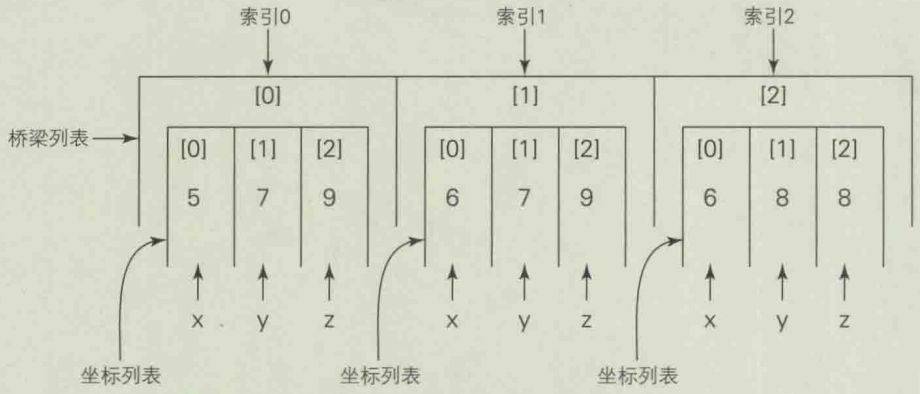


图 4-5　桥列表是一系列的坐标，每一个坐标也是一个列表

Python 实际上有很多不同类型的变量来存储一系列元素，这些类型是你用过的列表的增强形式。专业的 Python 程序员可能会在程序里使用一种叫作元组（tuple）的特性来表示坐标。我不打算在这里深入介绍关于元组的概念，如果想深入了解元组，你可以在互联网上搜索相关资料，找出元组与列表的不同点，以及元组给 Python 编程带来的好处。

检测一个方块被击打

在这次冒险中，你的工具包里还需要最后一项检测能力，就是检测玩家击打一个方块的能力。方块击打检测可以让你创建一些精彩的游戏和程序，因为这能让玩家和 Minecraft 里的每一个方块直接交互。

开始一个新程序，一开始这只是一个小的、独立的试验，但是稍后它就会变成本次冒险最终的游戏程序。

1. 在“编辑器菜单”选择 File⇨New File 新建一个文件，单击 File⇨Save As 将文件保存为 `blockHit.py`。

2. 导入必需的库：

```
import mcpi.minecraft as minecraft
import mcpi.block as block
import time
```

3. 连接到 Minecraft 游戏：

```
mc = minecraft.Minecraft.create()
```

4. 获取玩家的坐标，然后向一边略微移动一点。将这个坐标作为你即将创建的钻石块——你的神秘宝藏的坐标：

```
diamond_pos = mc.player.getTilePos()
diamond_pos.x = diamond_pos.x + 1
mc.setBlock(diamond_pos.x, diamond_pos.y, diamond_pos.z, ↩
  block.DIAMOND_BLOCK.id)
```

5. 定义一个 `checkHit()` 函数，这个函数会在最终的程序中使用，所以请保证函数名正确：

```
def checkHit():
```

6. 向 Minecraft API 请求一个触发的事件列表。返回的是一个普通的列表，就像你之前在 `vanishingBridge.py` 程序里使用的那样：

```
  events = mc.events.pollBlockHits()
```

7. 用 for 循环依次处理每一个事件。阅读下面的深入挖掘代码部分获取更多关于 for 循环的细节解释：

```
  for e in events:
    pos = e.pos
```

8. 让你的程序检查玩家击打的方块位置是否与钻石块的位置相同。如果相同，就让程序在聊天窗口输出一条消息：

```
    if pos.x == diamond_pos.x and pos.y == diamond_pos.y ↩
and pos.z == diamond_pos.z:
      mc.postToChat("HIT")
```

9. 最后，编写主游戏循环。目前，你可以使用一个 1 秒的延时，以此限制程序向聊天窗口输出消息的速度，但你也可以试验其他的延时数值，让程序更加好用。同时依旧要仔细检查代码缩进：

```
while True:
  time.sleep(1)
  checkHit()
```

单击 File⇨SaveAs 保存文件并选择 Run⇨RunModule 运行程序。

移动玩家直到你能看见那块钻石块。现在，用剑击打钻石块的每一面。发生了什么？如图 4-6 所示，当你击打钻石块时，聊天窗口就会出现一条“HIT”消息。

图 4-6　一个方块被击打

如果你不小心按错了鼠标按钮将宝藏破坏掉，你将无法再击打它！你只能重新运行程序来创造一个新宝藏，或者手动在相同的位置放置一个钻石块。因为空气方块不会触发击打事件。

挑战

修改 `blockHit.py` 程序，让它也能读取 `for` 循环里事件返回的 `e.face` 变量。`e.face` 变量是一个数字，它的值取决于方块的六个面中哪个面被击打。

先在聊天窗口中输出 `e.face` 的值，然后在游戏里击打方块的六个面，看看哪个值代表哪个面。最后，修改你的程序，让它在你击打钻石块的不同表面时输出不同的消息。

深入代码

在方块击打检测程序中，你使用了 for 循环，但是你以前从来没用过这种形式的 for 循环。我们来看看这个神奇的形式，这是 Python 里非常有用的一个特性。

之前你使用的一般 `for` 循环看起来是这个样子：

```
for i in range(6):
  print(i)
```

`range(6)` 实际上是一个生成一个数字列表 `[0, 1, 2, 3, 4, 5]` 的函数，你可以在 Python Shell 里输入这句代码来验证这一点：

```
print(range(6))
```

你将会看到如下输出结果：

```
[0,1,2,3,4,5]
```

看起来有点像，不是吗？实际上 `range()` 函数就是用来生成一个新的数字列表的。

Python 里的 `for/in` 语句会在一个列表里遍历所有元素。它先将列表里的首项元素存储在循环控制变量里（例如这里的 `i`），然后执行循环体，再把下一项元素存储在循环控制变量里，如此循环，直到列表里所有元素都被遍历一次。

这意味着，只要你有一个列表，无论里面的元素是什么，你就可以用同样的方式遍历这个列表。在 PythonShell 里尝试执行这些代码：

```
for name in["David","Gail","Janet","Peter"]:
  print("hello " + name)
```

你也可以遍历一个字符串中的每一个字符，就像这样：

```
name = "David"
for ch in name:
  print(ch)
```

编写一个寻宝游戏

在这次冒险中，你一直在学习技能、编写和测试代码片段以及试验 Minecraft 的各种感应特性。现在，是时候将所有代码组合在一起，编写成一个完整游戏了。游戏的名字叫“空中寻宝”，这是一个寻宝游戏，你要使用一个导航信标来寻找随机出现在天上的钻石块，然后击打钻石块获取分数。

尽管这个游戏有点特殊——你每走一步都会留下金块组成的踪迹，每消耗一个金块都会扣掉你一分。如果你在寻找宝藏时只是毫无目标地跑来跑去，你的分数就会迅速减少，甚至可以是负数！你需要应用你的 Minecraft 导航技巧来快速寻找钻石块，并且尽可能在获取宝藏时消耗较少的步数。

当你找到所有的宝藏之后，金块踪迹就会自动消失（可能会在地面上留下洞穴，所以注意脚下！）。

这个程序主要由在本次冒险中编写过的小段实验代码中可重复利用的部分构成。你可以复制并粘贴一些程序的代码，然后在此基础上修改，这样可以节省一些打字时间。但我也会提供全部的源码，让你了解哪些代码是需要的。

专业的软件工程师通常会先制作一个简单的框架程序，里面仅仅包含一些输出语句，用于测试程序结构是否正确，然后逐渐添加并测试新特性。在本节中，你将会体验作为一名真正的软件工程师，一步步编写并测试程序的过程。首先，编写一个游戏循环的框架和一些模拟的函数，这些函数将会在编写程序的过程中逐渐被完善。

编写函数和主循环

1. 单击 File⇨NewFile 新建一个文件，单击 File⇨SaveAs 将文件保存为 `skyHunt.py`。
2. 导入必需的库：

```
import mcpi.minecraft as minecraft
import mcpi.block as block
import time
import random
```

3. 连接到 Minecraft 游戏：

```
mc = minecraft.Minecraft.create()
```

4. 定义一个 score 变量跟踪玩家的分数，再定义一个 `RANGE` 常量，通过设置玩家周围随机出现宝藏的范围来控制难度。一开始测试时，要把这个值设置成一个比较小的数，当程序完成时，可以把这个数换成一个较大的数：

```
score = 0
RANGE = 5
```

5. 因为程序需要一步一步地开发与测试，所以首先给程序的功能编写几个模拟的函数。Python 的函数里至少要有一条语句，所以你可以使用 Python 的 `print` 语句来输出一些消息。这仅仅是临时用来占位的代码，稍后将会详细实现这些函数：

```
treasure_x = None # 宝藏 x 的坐标

def placeTreasure():
  print("placeTreasure")
```

```
def checkHit():
  print("checkHit")

def homingBeacon():
  print("homingBeacon")

bridge = []

def buildBridge():
  print("buildBridge")
```

6. 现在编写主游戏循环。当游戏运行时，这个循环会运行得非常快（大约每秒执行 10 次），所以金块踪迹会非常准确地出现在天空中，但是，在程序测试阶段，先将这个速度降到每秒仅执行 1 次。在游戏循环里调用稍后会完善的模拟函数：

```
while True:
  time.sleep(1)

  if treasure_x == None and len(bridge) == 0:
    placeTreasure()

  checkHit()
  homingBeacon()
  buildBridge()
```

7. 单击 File⇨Save 保存文件，然后单击 Run⇨RunModule 运行代码。程序应该没有任何错误，并且目前程序的效果应该是每秒在 PythonShell 输出一些消息，现在程序的框架已经做好了，你可以一点点向其中添加新的代码。

在空中放置宝藏

第一个需要写的函数是在空中的随机位置放置宝藏。你要使用三个全局变量来存储宝藏的坐标，并且变量被初始化为 `None`。`None` 是 Python 里的一个特殊的值，表示变量已经被加载到内存里，但是还没有被赋值。你需要在主循环里用它检查是否需要建立一个新的宝藏。

1. 创建三个全局变量用于跟踪宝藏的坐标，将加粗的代码添加到文件里：

```
treasure_x = None
treasure_y = None
treasure_z = None
```

2. 补充 `placeTreasure()` 函数（并记得删掉之前添加的 `print` 语句）：

```
def placeTreasure():
  global treasure_x, treasure_y, treasure_z
  pos = mc.player.getTilePos()
```

3. 用 `random` 函数在玩家周围不超过 `RANGE` 范围的区域生成宝藏，但是更改 y 坐标，使宝藏生成在玩家上方（也许会是在空中）：

```
treasure_x = random.randint(pos.x,   pos.x+RANGE)
treasure_y = random.randint(pos.y+2, pos.y+RANGE)
treasure_z = random.randint(pos.z,   pos.z+RANGE)
mc.setBlock(treasure_x, treasure_y, treasure_z,
block.DIAMOND_BLOCK.id)
```

运行并测试程序，检查玩家附近的天空中是否有一块宝藏。

在击打时收集宝藏

现在需要稍微修改之前的 `blockHit.py` 程序用于检测宝藏何时被玩家的剑击打。

1. 删除 `checkHit()` 函数中的 `print` 语句，并用如下所示的代码代替它。`score` 和 `treasure_x` 变量必须声明为全局变量，因为 `checkHit()` 函数需要修改它们的值。Python 要求在函数里列出所有需要在函数里被修改的全局变量。如果不这么做，程序将不能正常运行：

```
def checkHit():
  global score
  global treasure_x
```

2. 读取所有方块击打事件，并检查被击打的方块是否是宝藏方块：

```
  events = mc.events.pollBlockHits()
  for e in events:
    pos = e.pos
    if pos.x == treasure_x and pos.y == treasure_y ↩
    pos.z == treasure_z:
      mc.postToChat("HIT!")
```

3. 如果被击打的方块是宝藏方块，就告诉程序向 score 变量添加分数，并移除宝藏方块。最后不要忘记将 `treasure_x` 变量设置为 `None`（这会让 `placeTreasure()` 函数能生成一个新的随机宝藏）。注意这里的代码缩进，因为这里的代码是 `if` 语句的一部分：

```
      score = score + 10
      mc.setBlock(treasure_x, treasure_y, treasure_z, ↩
      block.AIR.id)
      treasure_x = None
```

保存并运行程序，检查玩家击打宝藏时，宝藏方块是否消失。并且在宝藏消失之后，一个新的随机宝藏又会在玩家附近生成。

添加一个导航信标

导航信标每秒都会在 Minecraft 聊天窗口显示当前分数和玩家与宝藏之间的大概距离。

1. 新建一个 `timer` 变量。因为主游戏循环会每秒循环 10 次，所以你在这里需要每秒计数 10 次。`timer` 就是用于计数的变量。如果你改变了主循环的速度，你也需要在这里更改 `TIMEOUT` 的值。确认下面这些代码直接放在 `homingBeacon()` 函数上方（没有任何缩进）：

```
TIMEOUT = 10
timer = TIMEOUT
```

2. 删除 homingBeacon() 函数里的 print 语句，然后将 timer 声明为一个全局变量，因为此函数将会改变这个变量的值：

```
def homingBeacon():
  global timer
```

3. 如果 treasure_x 的变量有值，宝藏就会在空中出现。你需要检测宝藏是否已经生成了，如果宝藏存在，你就会在聊天窗口收到导航信标的信息：

```
  if treasure_x != None:
```

4. 这个函数每秒会被主循环调用 10 次，所以你需要每执行 10 次函数才更新导航信标数据：

```
    timer = timer - 1
    if timer == 0:
      timer = TIMEOUT
```

5. 每次 timer 超时（也就是每 10 次函数调用，或者说每秒），函数会计算出一个比较粗略的数字，告诉玩家与宝藏之间的距离。abs() 函数会返回两个位置之间距离的绝对值（一个正数）。把所有的距离加在一起，你会得到另一个值，这个值在玩家距离宝藏较远时会偏大，在玩家距离宝藏较近时会偏小。注意这里代码的缩进，这里的所有代码都是 if 语句的一部分：

```
      pos = mc.player.getTilePos()
      diffx = abs(pos.x - treasure_x)
      diffy = abs(pos.y - treasure_y)
      diffz = abs(pos.z - treasure_z)
      diff = diffx + diffy + diffz
      mc.postToChat("score:" + str(score) + " treasure:" + ↩
                    str(diff))
```

保存并运行程序，确保导航信标和分数会在 Minecraft 聊天窗口上显示。由于目前还在程序的开发调试阶段，并且主游戏循环运行 10 次的速度要比正常慢，所以你看到的效果应该是每 10 秒收到一条消息。你应该明白这是因为函数每计数 10 次才会进行操作。另外，你仍然会在 PythonShell 里看到一些模拟函数的输出内容，因为程序还没有编写完成。

添加桥梁建造者

现在需要将之前的 vanishingBridge.py 的桥梁建造者加入到程序里。你只需要改动一点点，仅仅是检查玩家是否站在金块上，如果不是，就生成金块轨迹。

1. 让你的 buildBridge() 看起来如下所示，与 vanishingBridge.py 的程序不同的重要代码行已经被加粗：

```
bridge = []

def buildBridge():
  global score
  pos = mc.player.getTilePos()
```

```
    b = mc.getBlock(pos.x, pos.y-1, pos.z)

    if treasure_x == None:
      if len(bridge) > 0:
        coordinate = bridge.pop()
        mc.setBlock(coordinate[0], coordinate[1],
        coordinate[2], block.AIR.id)
        mc.postToChat("bridge:" + str(len(bridge)))
        time.sleep(0.25)
    elif b != block.GOLD_BLOCK.id:
      mc.setBlock(pos.x, pos.y-1, pos.z, block.GOLD_BLOCK.id)
      coordinate = [pos.x, pos.y-1, pos.z]
      bridge.append(coordinate)
      score = score - 1
```

2. 恭喜你！你已经完成了整个程序的编写。因为程序已经完全准备好正式运行了，所以现在需要将 `time.sleep(1)` 的延时修改为 0.1 秒。这会让程序的主循环每秒运行 10 次。因为 `homingBeacon()` 函数里的 timer 变量每计数 10 次才会执行一次，所以现在聊天窗口里应该每秒都会显示导航信标数据了。

保存并运行程序。检查你收集到宝藏之后，金块轨迹是否消失，以及每消耗一个金块时，你的分数是否减少。

现在你可以尽情玩游戏了！看看通过收集宝藏来获取高分有多困难？

图 4-7 展示了 Minecraft 聊天窗口里的分数和导航信标数据。

图 4-7　导航信标显示了玩“空中寻宝”游戏时的近似统计数据

快速参考表

获取某坐标处方块的类型	获取被击打的方块的坐标

`b = mc.getBlock(10,5,2)`	`hits = mc.events.pollBlockHits()`
	`for hit in hits:`
	`  pos = hit.pos`
	`  print(pos.x)`

创建一个列表	**向列表末尾添加一个元素**
`a = [] # 一个空列表` `a = [1,2,3] # 一个已初始化的列表`	`a.append("hello")`
输出列表的内容	**获取列表的大小**
`print(a)`	`print(len(a))`
通过索引访问列表里的元素	**访问列表的最后一项元素**
`print(a[0]) # 0= 第一项 ,1= 第二项`	`print(a[-1])`
移除列表的最后一项元素	**遍历列表的每一项元素**
`word = a.pop() # 移除末尾的元素` `print(word) # 已移除的元素`	`for item in a:` `  print(item)`

方块交互的扩展冒险

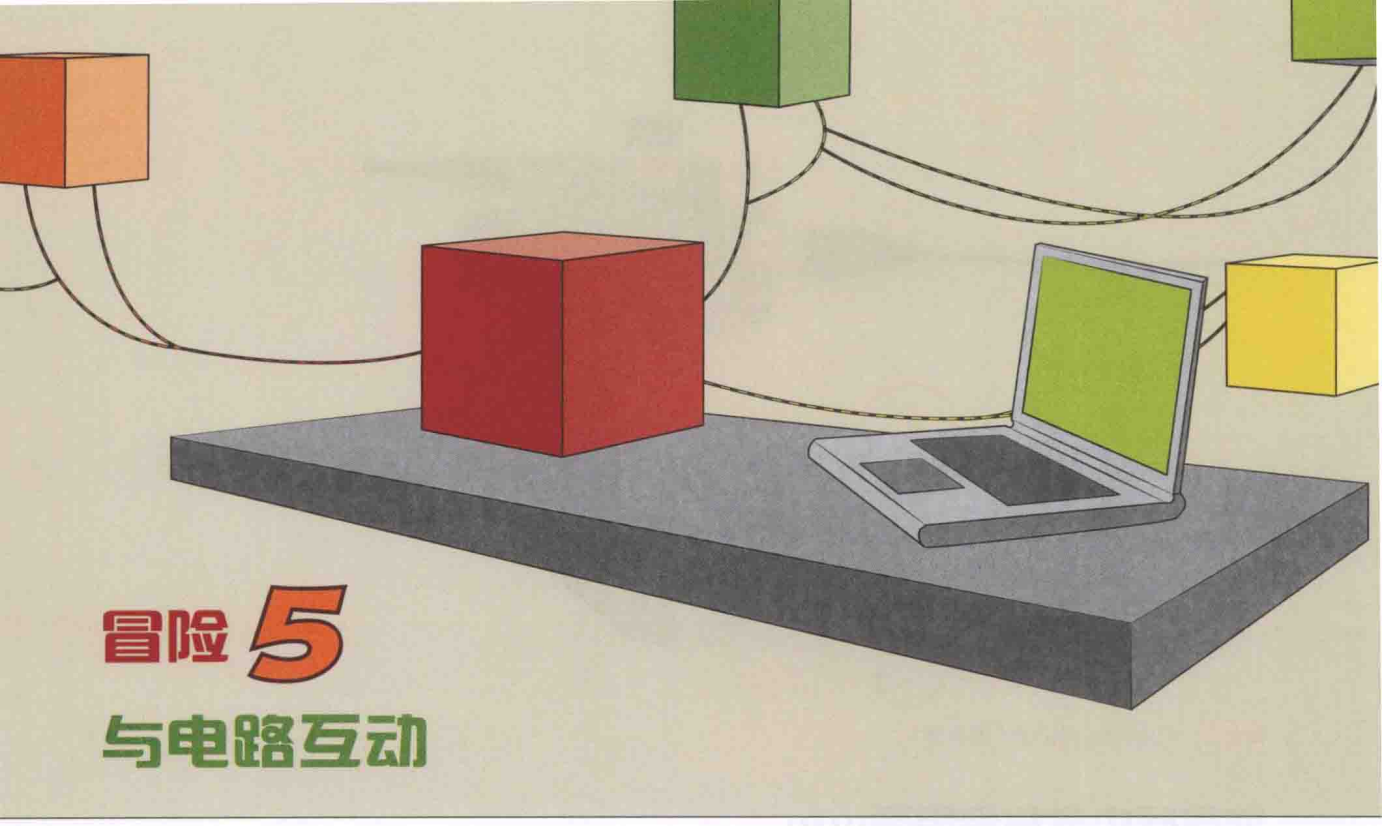
在本次冒险中，你学会了使用 `getBlock()` 函数检测玩家站在什么方块上，以及如何使用 `events.pollBlockHits()` 响应玩家击打方块表面的事件。你也在 Minecraft 里编写了一个奇妙的完整游戏，包括分数功能！

- 将 `RANGE` 常量改为一个更大的数，让宝藏生成得距离玩家远一些。这样游戏会更困难。试着添加一些新的游戏特性，比如在聊天窗口显示“冷”、“暖”或“热”等信息，以此来表示玩家距离宝藏的远近。
- 设计更好的计分系统，让玩家在正常玩游戏时有可能获取到一个较好的正数得分。多做一些实验，找出最好的收集宝藏得分数和消耗金块扣分数。
- 在互联网上研究毕达哥拉斯定理（我国一般称作勾股定理——译者注），看看能不能让 `homingBeacon()` 函数里的估算距离更准确。（嘘！ Martin 在冒险 7 中提到了一些相关内容，所以你可以偷看一下冒险 7，看看能不能自己研究明白！）

解锁成就：反重力定律的专家、水上漂——在一次冒险中获得两个成就！

在下一次冒险中……

在冒险5中，你将要学到如何跳出Minecraft虚拟游戏的框架，甚至不再拘泥于计算机的框框。你将会把 Minecraft 和真实世界的对象连接起来，你将会使用电子元器件制作你自己的交互式游戏控制器，以及游戏里的另一个完整游戏！



冒险 5
与电路互动

在玩 Minecraft 时，无论你是否在使用编程界面，都是处于一个虚拟世界之中。而与游戏互动的唯一方式便是通过工程师提前设计好的方式，用鼠标和键盘来控制。

但也有其他和 Minecraft 互动的方法可以打破沙盒游戏和现实世界的藩篱。在此你将会发现虚拟与现实之间的界线变得模糊了，这也让你的游戏经历变得富有创造性，引人入胜。

在这个冒险中你将会学习如何用 API 将 Minecraft 与小型电路连接起来。在最开始，你会制作一个当你在游戏中进入房子时会闪烁的灯，然后你将添加一个称作“七段数码管”的特殊显示方式，你可以用它来在你和 Minecraft 互动时显示倒计时或是其他游戏信息。你将要添加一个按钮，通过按按钮可以在 Minecraft 中触发各种有趣的动作。最后你将要把这些像图 5-1 那样组合起来，制作一个红色的大按钮来启动引爆倒计时。从此你在清理建筑用地时将不会有任何难题，而你的小伙伴们也将会对你的新花样感到惊奇，并要求你为他们也做一个！

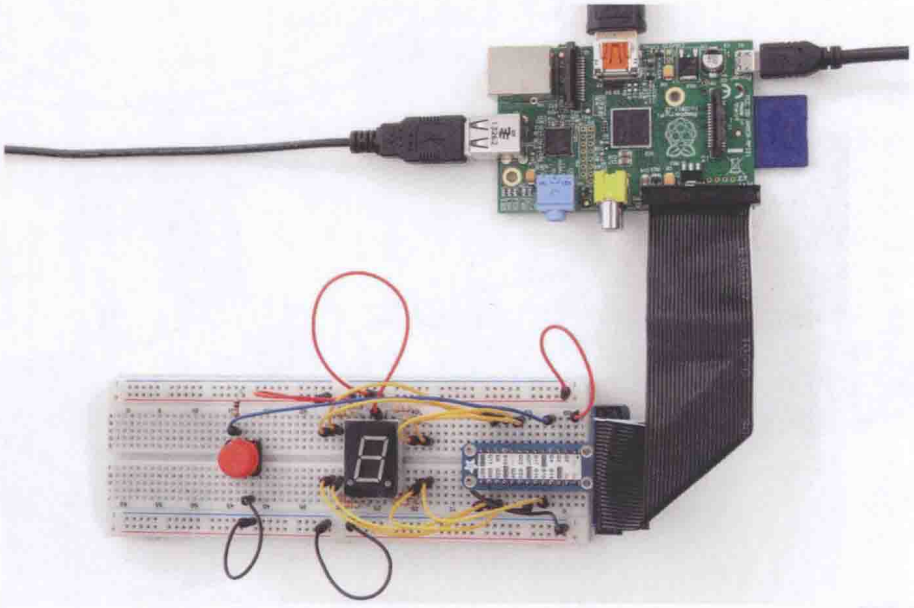


图 5-1　使用树莓派制作的引爆按钮

在冒险中你将会用到……

下面是在这个冒险中你所需要的所有东西——你将会需要一些电子部件，以及一些附加部件来把你的计算机和它们连接起来。所有你要用的部件都在图 5-2 和图 5-3 中：

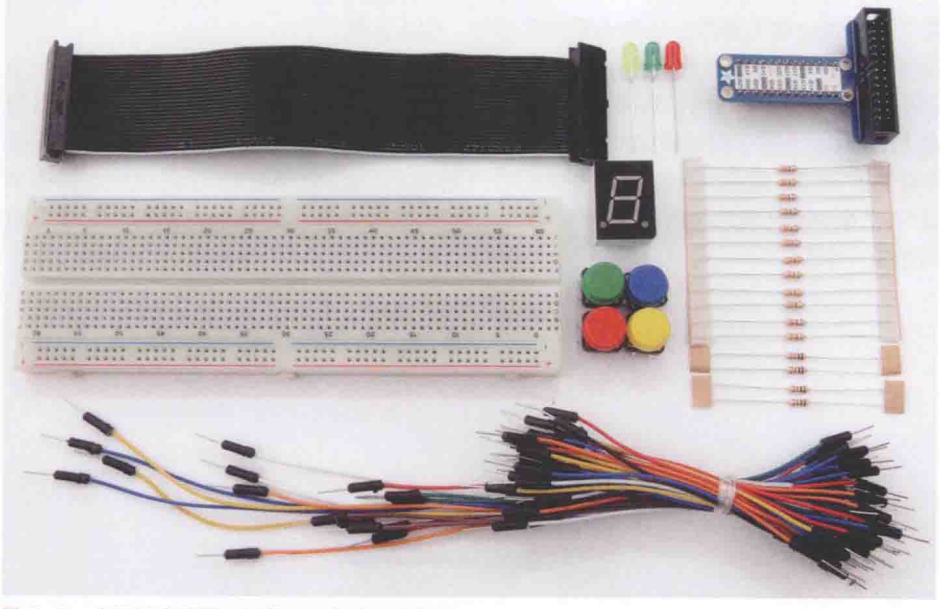


图 5-2　本冒险中需要和树莓派配合使用的部件

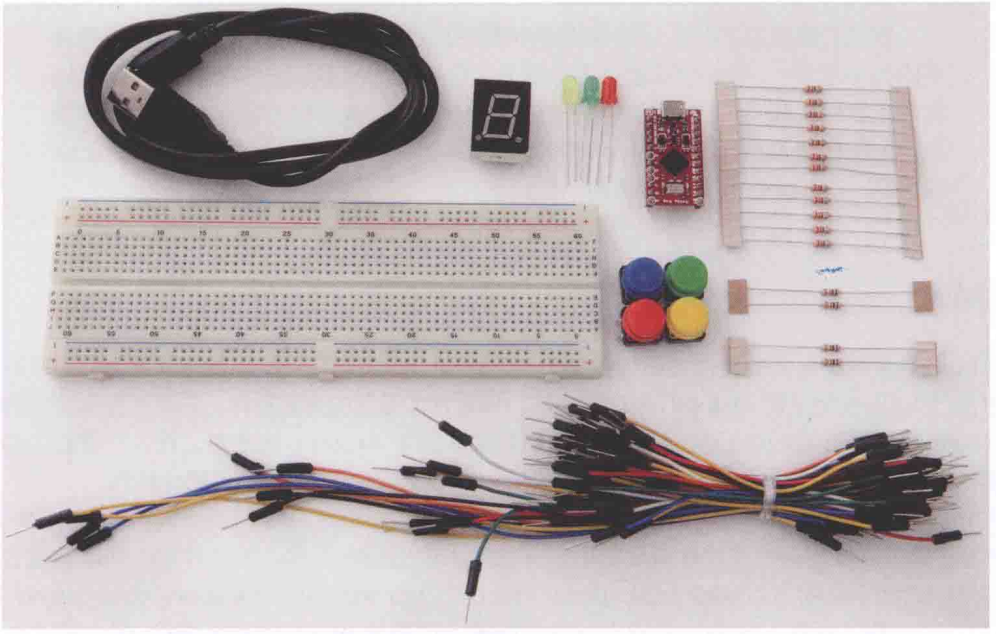


图 5-3　本冒险中需要和 PC 或 Mac 配合使用的部件

- 一块面包板
- 至少四个按钮式常闭开关
- 至少一个 LED，任意颜色（但颜色越多越好玩！）
- 一个七段数码管
- 十个以上的 330 欧姆电阻器，用来配合你的数码管和 LED 使用
- 四个以上的 10 千欧姆电阻器，用来配合你的按钮使用
- 至少 20 根两端是针脚的接线，或者一些细的实心电线和剥线钳
- 一个小的电池盒和两个 5 号或 7 号电池

网络上有大量的供应商，你可以从他们那里买到所有部件。例如 www.skpang.co.uk，www.sparkfun.co.uk，www.coolcomponents.com 以及 www.protopic.com，还有例如 Maplin Electronics 的商业大街电子商店中的绝大多数。许多商店销售成包的部件，例如一些 LED 和电阻或是接线包。如果你很难找到两端是针脚的接线，你也可以买一段实心电线，然后用剥线钳简便地制作一些接线。冒险中默认使用接线，但用实心电线的方法是完全相同的。

对于树莓派

你可以用一端是针脚，一端是接口的接线来直接把面包板上的孔连接到树莓派。这有时可能很繁琐，而且插入所有的针脚需要一定技巧。在冒险中我使用了一个预焊接的 Adafruit Pi-T-Cobbler，它是一个连接树莓派的排线，带标记的针脚可以直接插入面包板。如果你会焊接，你也可以买一个 Pi-T-Cobbler 并把它焊在一起，这样要便宜得多。图 5-2 展示了在冒险中使用树莓派所需的所有部件。

你可以从 www.adafruit.com 或 www.skpang.co.uk 上购买 Pi-T-Cobbler。切记要买已经预装好的 Pi-T-Cobbler（除非你想同时练习焊接！）。

在我们编写这本书时，B+ 型号的树莓派已经发布了。所有书中的冒险都是使用的树莓派 B+，但你需要额外为你的 Pi-T-Cobbler 购买一根接线以正确连接树莓派 B+ 的 40 路插头。接线需要有一个 40 针接口来连接树莓派 B+，另一端则是一个用来插入 Pi-T-Cobbler 的 26 路插头。你可以在从 www.skpang.co.uk 购买 Pi-T-Cobbler 的同时得到一根。

对于 PC 和 Mac

由于 PC 和 Mac 并没有树莓派的硬件插口，我基于一个 Arduino 组装了一个小的硬件平台，可以在这些计算机上使用。你可以使用任何 Arduino 来做这件事，只要预加载我的开源软件，它就能像树莓派的输入输出插口一样使用。你可以在 http://arduino.cc 了解到更多关于 Arduino 的信息。在这个冒险中你将会用到一个叫作 Arduino Pro Micro 的版本，同时你需要一根 USB 线把它连接到你的计算机。

你可以在 https://www.sparkfun.com/products/12587 购买 Arduino Pro Micro（产品号 DEV-12587）。但你需要焊接以及导入特殊软件来让它可以作为输入输出板来工作。为了节省时间以及省去麻烦，你也可以从 http://skpang.co.uk/catalog/pro-micro-33v8mhz-with-headers-and-anyio-firmware-p-1327.html 购买预焊、预测试并导入软件的版本（产品号 AR-PROMICRO-ANYIO）。你在此之后需要做的只是将电路板插入你的面包板，并用 USB 线连接你的 PC 或 Mac，它就会像一块输入输出板一样工作了。图 5-3 展示了用 PC 或 Mac 开始冒险所有你需要的部件。

如果你有一台 Mac，它至少需要 OS X 10.6 版本来让 Arduino 正常工作。

你需要一些软件的源文件来完成此次冒险。所有文件都在初学者工具包中，可以从本书的相关网站下载。我在此之后也提供了链接，以便你或你的朋友可以从此下载最新版的源文件，以供在 Minecraft 以外的其他工程和实验中使用。

一旦你集齐了所有需要的零部件，我们就可以开始建造你的第一个电路了！

用面包板制造电子原型机

当一个电子工程师为一个新产品设计电路时，他们总是会在开始制造、一下生产几千个之前，先设计一个原型机（prototype）。

一个原型机（prototype）是物体的小型模型，以供测试物体是否正常工作。修复单独一个物体的问题，要远比制造了成千上万个之后再修复简单得多。

定义解释

在网上有很多树莓派计算机原型机的精彩图片，这些原型机由树莓派基金会的 Eben Upton 于 2006 年所造。绝大多数复杂的电路板最初都是像这样的原型机，然后可能经过了许多次的调整和修改，才变成了现在你面前的这一块。你可以在这里看到图片：http://www.raspberrypi.org/raspberry-pi-2006-edition/。

作者提醒

大多数的电路都被焊接在一起以提供牢固和耐久的连接。焊接电路虽然提供了长效的连接，但对你的元件试验以及设计和测试电路没有益处，因为你将会花费大量时间来拆焊和重焊元件。由于这个原因，工程师们通常会使用一个免焊接的设备——面包板（breadboard）。

面包板（breadboard）是一种可重复使用的设备，不用焊接元件就可以制造电路。面包板有一系列的洞，可以将电线或电子元件插进去来制造电路。位于顶部和底部的各两排洞用作电源。一般会有一条红色的线，用做正极连接（例如 3.3 伏特，有时标为 + 或 VCC），和一条蓝色或黑色的线，用作 0 伏特（或接地线，有时标为 − 或 GND）。

定义解释

图 5-4 展示了两块面包板，左边的面包板已经插上了一些元件来组成一个电路，右边则是面包板内部的样子。仔细看的话能够看到连接用的金属条。理解面包板内部是如何连线的十分重要，因为由此你可以知道当你插入元件时，它们是如何连接起来构成电路的。

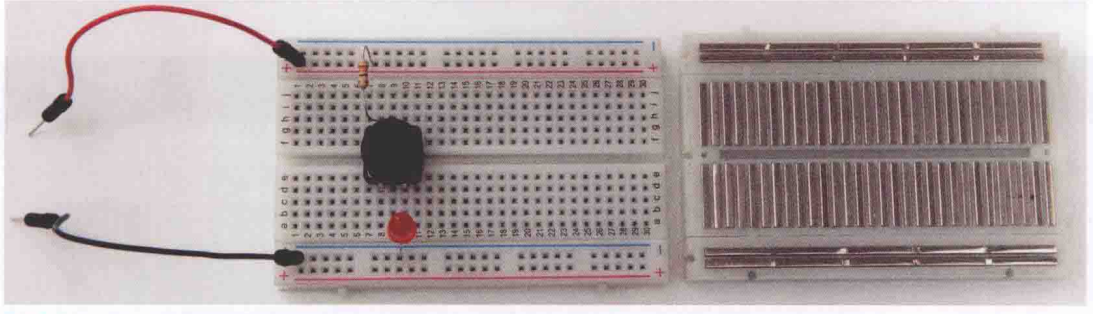


图 5-4　左：一块面包板；右：面包板内部及其内部的连线情况

建造点亮 LED 的电路

图 5-4 是一个简单的电路，可以用来测试元件。任何电路的工作都需要加上电压（voltage），在这

里需要将一个 3 伏特的电池连接到面包板顶部和底部的条上，这些条被称为“电源导轨”。这些导轨在面包板全长上都是连通的，这方便了同时为大量元件提供电力。红色元件是一个发光二极管（LED），一种之后你会用到的灯。上面有彩色条带的元件是一个电阻器（resistor），而位于它下面的黑色装置是一个按钮。如果你从电池开始，用电线把所有元件顺序连接起来，最后连回到电池上，这就是一个完整的电路了。如果你按下按钮，电流（current）就会开始通过，LED 就会点亮。

电压（voltage），又称电势差，是电路中两点间电势（原文为电能 electrical energy，电势更为准确——译者注）的差值，它在电学中相当于管道中的水压，是迫使电流通过电路的压力。电压的单位是伏特（V，简称伏）。

发光二极管（Light-emitting diode，LED）会在有电流经过时点亮。二极管只允许电流单向通过，一个 LED 因此只会在正确方向通过电流时点亮。LED 有各种不同颜色，但总有一根“短腿”（阴极或负极）和一根“长腿”（阳极或正极），由此你可以判断为了让电流通过，应该以哪个方向放入电路中。

电阻器（resistor）是一种在电路中阻碍电流通过的元件。举例来说，LED 会被过大的电流损坏，但如果你加入合适的电阻器来限制流过 LED 的电流，就可以保护 LED。电阻值的单位是欧姆，通常用希腊字母 Ω 来表示（简称欧）。你需要选择合适电阻值的电阻器来限制电路中的电流，电阻值可以通过从左到右读取上面的彩色条带来获知。访问 http://zh.wikipedia.org/wiki/ 电阻色码，了解更多关于电阻器颜色编码的信息。

深入电路

在图 5-4 的电路中使用了一个电阻器。LED 是一种精细的元件，只能通过很小的电流，通常不超过 20 毫安（即 0.02 安培，是很小的电流）。如果把 LED 和电池直接连接，电路中会通过过大的电流而损坏 LED。为了阻止这种现象，一个电阻器被连接进电路来限制电流。电阻器的电阻值由其上的彩色条带注明。

如果使用了一个电压更高的电池，那么就需要使用更高电阻值的电阻器来限制电路中的电流。我已经为你们计算好了所需的电阻值，但也有许多的网站可以帮助你重新计算不同电压的电池所需要的电阻值，例如 www.ohmslawcalculator.com/led_resistor_calculator.php。

电流（current）是电路中电荷（原文为电能 electrical energy，电荷更为准确——译者注）流过一点的速率，在电学中相当于管道中水的流速。电流的单位是安培（A），通常也简写作安，微小的电流也使用毫安（mA）作为单位。访问 www.allaboutcircuits.com/vol_1/chpt_1/7.html 了解更多关于电路中电流实际流向的有趣理论。

挑战

尝试用你的电子元件建造图 5-4 的电路，在你按按钮时，LED 应该会点亮。你会用到一个小电池盒和两节 5 号或 7 号电池。确保将 LED 按正确的方向放入，如果 LED 不亮的话，仔细检查接线并确认 LED 的方向正确。

将电路和你的计算机连接

很快你就将把 LED 连接到你的计算机并用一个 Python 程序来控制。当你想要用计算机控制电子元件时，你的计算机需要一种连接和控制这些电路的方法，这就是通用输入输出（GPIOs）。

通用输入输出（GPIOs）是连接到计算机系统中央处理器的信号。它们并不被设计用于特殊用途，而是用于通用用途。也可以被设计者指派用来使用电路，通常是用来监控外部电路。

在一些计算机（例如树莓派）上，GPIO 线路被引出至计算机侧面的特殊针脚。一个 GPIO 针脚既可以作为输入设备来探测外部电压，也可以被设为输出设备以控制外部电压。树莓派或是 Arduino 板使用的是 3.3 伏特的电压，虽然很小，但这些针脚足以提供点亮 LED 或是感应一个按钮的电流。作为比较，英国的主电力供应是 240 伏特，非常高而且危险。而美国的输电铁塔通过的电压甚至高达 230 000 伏特！和这些相比，计算机电路中使用的电压非常微小。

计算机是数字化的机器，也就是说它内部使用的是数字。所有的现代计算机都是数字化的，使用数字 1 和 0。当使用计算机的 GPIO 针脚时，0 一般代表无电压（关），而 1 则代表产生了一个电压（在此是 3.3 伏特）。当你使用 GPIO 针脚来探测外部电压时，计算机内部只能把针脚上的电压转换成数字 0 或是 1。接近 0 伏特的电压被作为 0，接近 3.3 伏特的电压被作为 1。中间区域类似于“灰色区域”，可能被计算机视为 0 或 1。

设置 PC 或 Mac 来控制电路

如果你拥有一个树莓派，由于树莓派已经有预置的 GPIO 针脚而成为控制电路的绝佳计算机，因此你就可以跳过这一节，直接前往下一节“控制一个 LED”。

PC 和 Mac 并没有预置的 GPIO 针脚，因此你需要用另外一块电路板添加一个。做这件事有许多方法，但在本书中我选择使用一块小型的、预编程的 Arduino 板。这本书不会教任何关于 Arduino 的内容，但如果你想要了解更多这个小计算机，去看书后附录的“接下来去哪？”章节中提供的链接。

在之后的教学中，我默认你们已经购买了在本冒险开头列出的预焊、预编程的 Arduino 板。如果你决定购买空板并自己焊接的话，你需要遵循以下网站上的接线教程：http://www.sparkfun.com/products/12587，并加载我写的特殊开源 GPIO 软件。这个软件以及初学者工具包中的一个叫做 anyio 的 Python 模块，让 Arduino 板可以像树莓派上的 GPIO 针脚一样工作。

也就是说，在这个冒险中，之后的代码只需要在 Python 程序中改变一行就可以正常工作。如果你需要自己编程 Arduino 板的话，需要加载的软件储存在初学者工具包的 anyio/arduino/firmware 文件夹中。你和你的朋友也可以在我的 github 上找到最新版的 anyio 软件包以用于 Minecraft 以外的工程及试验：https://github.com/whalegeek/anyio

配置驱动程序

当设备被插入计算机时，计算机需要安装特殊的软件来与设备交流，这个软件名叫“设备驱动程序”。在此之前你可能已经在插入打印机或其他设备时安装过它的驱动程序，在此你只需要做一样的工作。

1. 将 USB 线一端插入 Arduino，另一端插入你的计算机。
2. 遵循屏幕上的指导和提示（不幸的是，在不同的操作系统上不一样），逐步安装驱动程序，这样你便可以在计算机上使用 Pro Micro。

在 Mac 上：

你会看到一个弹出消息说添加了一个新设备，单击红色的“叉”关掉消息框即可，在 Mac 上的安装就完成了！

在 PC 上：

你会被要求下载一个驱动程序，但你并不用真的下载，因为 Windows 包含了所有必要的驱动程序。但 Windows 并不擅长识别未知设备，因此我在初学者工具包中提供了一个文件来帮助 Windows 正常检测到设备。这个文件叫作 `ProMicro.inf`，储存在初学者工具包的 `anyio/arduino/firmware` 文件夹中。工具包位于 MyAdventures 文件夹。单击“从磁盘安装…”按钮，选择并单击“确定”，现在 Windows 应该就能将这个设备识别为一个 USB 串行插口了。

如果在这一步你的计算机不能识别设备，打开下面的 sparkfun 网页并遵循其上的连接教程，教程很详细，并也会随着操作系统的版本而更新：https://www.sparkfun.com/products/12587。你也可以去看看 SKPang 的 ProMicro 的产品页面，他在那里有更多的教程来帮助你：http://skpang.co.uk/catalog/pro-micro-33v8mhz-with-headers-and-anyio-firmware-p-1327.html。

找到串行端口号

刚刚连接到计算机的 Arduino 会在计算机上显示为一个叫作“串行端口”的设备。本书不会介绍其意义，但你需要确认 anyio 软件包选择了正确的端口，这样它才能像一个 GPIO 扩展器一样运行。

我已经为你写好了一个小程序来做这件事，程序可以在 anyio 软件包里找到。图 5-5 展示了一个使用此程序的会话实例，这样你就知道正常情况下应该会是什么样。

为了确认已经选择了正确的端口：

1. 打开 IDLE 并且单击 File⇨Open 来浏览已存在的 Python 程序。
2. 打开 MyAdventures 文件夹中文件名为 findPort.py 的文件，这样它就会被载入 Python 的编辑窗口。
3. 选择“菜单”中的 Run⇨Run 运行程序，你会在屏幕上得到一些指导。
4. 拔出连接 Arduino 的 USB 线，然后按回车键来开始扫描。这一步将所有 USB 线脱离你的计算机，这样程序就可以扫描计算机的所有端口，并列出所有那些没有连接设备的端口。
5. 然后，你将会被要求重新插入设备，程序将会再次扫描所有可用的端口，这样就可以得出插入了什么新设备。再次插入你的 USB 线，等几秒，然后再次按下回车键。
6. 如果这些都做成功了，你应该会得到一条关于找到的端口名称或是号码的信息，并会询问是否想要记住这些信息。输入 Y 并按回车键。findPort.py 程序（见图 5-5）将会生成一个名叫 findPort.cache 的文件，记录了 Arduino 所对应的端口号。之后当你使用任何 GPIO 功能时，anyio 都能正确地找到你的 Arduino 端口。

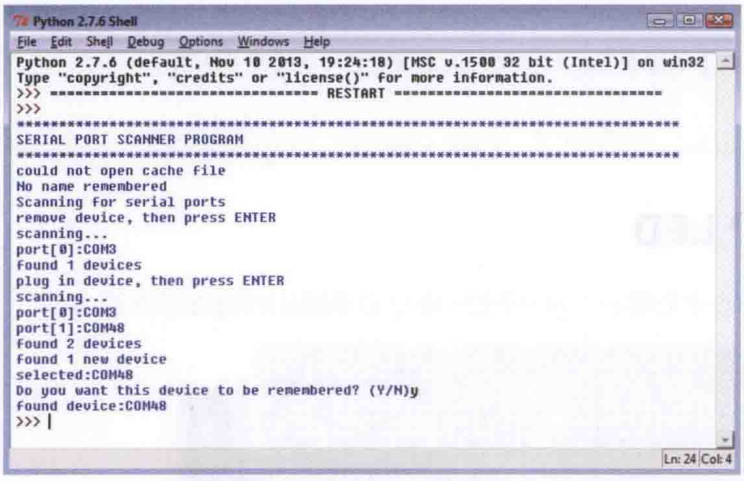


图 5-5　来自 findPort.py 程序的输出

如果你向计算机插入了其他 USB 设备，你可能会发现在你下次打开计算机并插入 Arduino 板时端口号改变，程序因而停止工作。如果发生这种情况，只需再次运行 findPort.py 程序，或是删除 findPort.cache 文件后运行程序来重新进行端口扫描。

编写一个能在任何可能的操作系统版本下运行的软件包是极其困难的，并且在将来还会有更新的操作系统。如果你的操作系统不能正常地安装软件或找到端口，去看看配套资源网站（www.wiley.com/go/adventuresinminecraft）上是否有一个升级版本的程序包，你也随时可以在我的 github 上找到最新的消息和程序包更新：https://github.com/whalegeek/anyio。

控制一个 LED

在冒险 1 中你编写了一个名叫 `helloMinecraftWorld.py` 的程序，它在 Minecraft 聊天中输出一条信息。当你控制电子器件时，你也可以写出一个相当于“hello world”的程序。这个程序并不会在屏幕上显示信息，而是会使一个 LED 闪烁。如果你在设置好所有部件之后能够编程来让一个 LED 闪烁起来，那么你就知道了所有部件都能正常工作，就可以拓展到更大更激动人心的工程了。

这其实是一个特殊的 LED，因为它被连接到了 Minecraft 游戏之中。你将会扩展在冒险 2 中所编写的“魔法门垫”实验中的一个设想。在这一次，门垫将会被连接到你的电路板上，而 LED 将会在玩家站在门垫上时闪烁。不仅是 LED，电路也可以被连接到任何东西上。想象一下，你可以编程来让玩家在站上门垫时点亮你真实卧室中的灯，而且远远不止如此！

在这个工程中，如果你在使用 PC 或 Mac，你需要用到 anyio 软件包。这个软件包在初学者工具包中提供，也可以从本冒险开头的链接中下载。

用你的计算机点亮一个 LED

首先你将在面包板上连接电路。图 5-6 和图 5-7 是关于如何将 LED 连接至计算机的示意图。

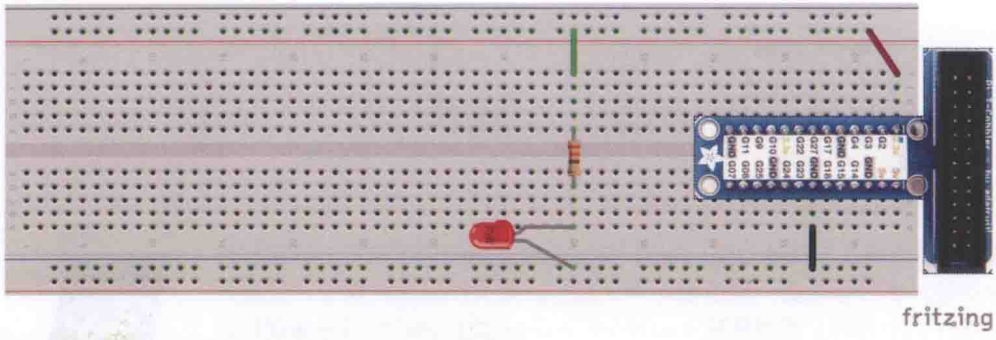


图 5-6　一个 LED 和一个电阻器被连接到一台树莓派上

1. 观察 LED 并找到较短的针脚，这是 LED 的负极（阴极）。将它连接到面包板底部的 0 伏特电源导轨。

2. LED 的另一个针脚更长，是正极（阳极）。将它和电阻器的一个针脚插入面包板的同一竖条中，并将电阻器的另一个针脚插在面包板的上半边。切记需要使用电阻器以限制通过 LED 的电流大小。

3. 用导线把上半部的电阻器针脚和面包板顶部的正极电源导轨相连，这样你就可以先测试 LED 了。这条线在此之后会被移动。

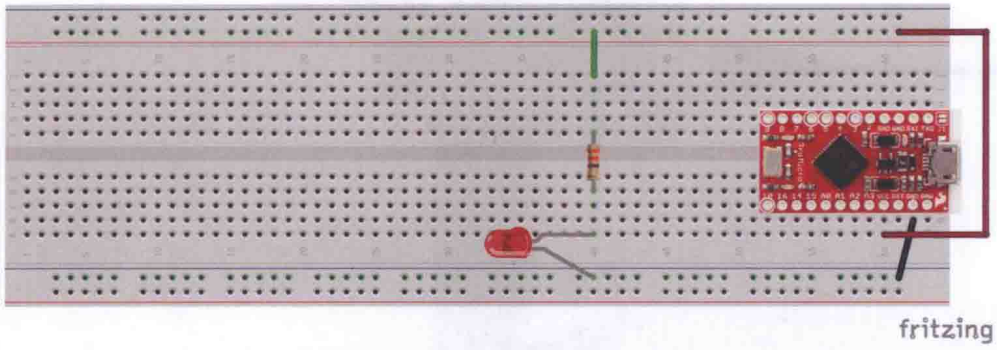


图 5-7　一个 LED 和一个电阻器被连接到一台 Arduino 上

基于树莓派和基于 PC/Mac（使用 Arduino）的教程稍有不同，因此请阅读和你所使用的设备相适合的部分。

在树莓派上：

1. 如果你有一个为 Pi-T-Cobbler 所做的黏性标签，把它贴上，如图 5-8 所示。这会帮助你更容易地找到正确的 GPIO 针脚。将 Pi-T-Cobbler 插入面包板，使得黑色连接器刚好处于面包板边缘外。把 Pi-T-Cobbler 插进面包板的右端，这样它的针脚会与最右边的一些插孔对应，一半针脚应该在面包板上半边，一半在下半边。用力下压来保证针脚入位。

2. 用排线连接 Pi-T-Cobbler 和树莓派。排线插头上有一个突起，而 Pi-T-Cobbler 的插口上有一个凹槽，由此保证插线方向正确。当用排线连接树莓派时，要注意要让塑料接线片处于远离树莓派电路板的一端，否则所有针脚都会插反，电路也不会工作！图 5-8 所示是连接后的样子。

> 最后需要检查你的计算机是否可以点亮 LED。在用 Python 程序让它闪烁之前，你将会使用计算机提供的电力来测试 LED。
>
> 
>

3. 用导线连接面包板顶端的正极电源导轨和 Pi-T-Cobbler 标注 3V3 的针脚。

4. 用导线连接面包板底部的负极电源导轨和 Pi-T-Cobbler 标注 0V 的针脚。

只要你的树莓派是通电的，你的 LED 现在就应该点亮，因为它正在被计算机的电源供能。

在 Arduino 上：

1. 将 Arduino 连接在面包板右侧并让银色的 USB 接口位于面包板边缘外。你可能需要用力按压来

让针脚插入面包板，但一定要在电路板上均衡用力以免碎裂。图 5-9 所示是连接后的样子。

2. 用 USB 线将你的计算机连接到 Arduino 上。

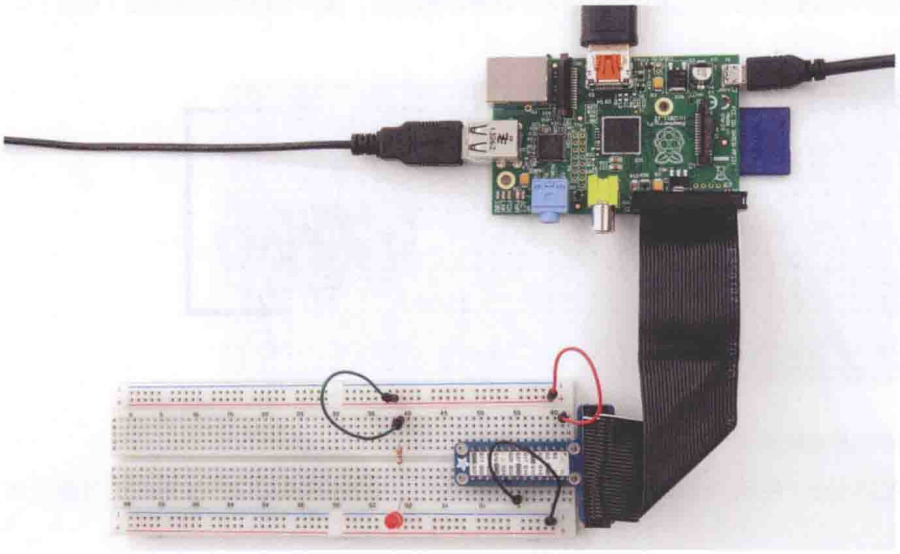


图 5-8 被连接到树莓派的 Pi-T-Cobbler

最后需要检查你的计算机是否可以点亮 LED。在用 Python 程序让它闪烁之前，你将会使用计算机提供的电力来测试 LED。

3. 用导线连接面包板顶端的正极电源导轨和 Arduino 标注 VCC 的针脚（VCC 是一个术语，意为正极电源，在你的 Arduino 上是 3.3 伏特）。

4. 用导线连接面包板底部的负极电源导轨和 Arduino 标注 GND 的针脚（GND 是一个术语，意为 0 伏特的电源连接）。

只要你的 Arduino 是通电的，你的 LED 现在就应该点亮，因为它正在被计算机的电源供能。

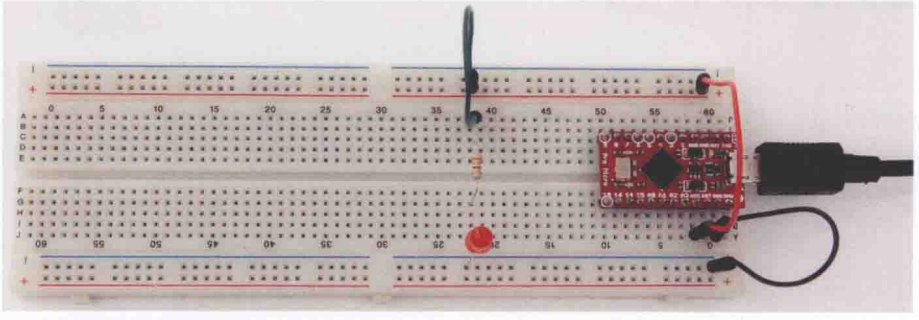


图 5-9 被连接到面包板的 Arduino

如果你的 LED 没有点亮，检查它是否朝向正确，把它的正负针脚互换来观察是否工作。你也可以用面包板内部布线的图片来检查，从 3.3 伏特电源针脚开始一路连接到 0 伏特的针脚，检查是否每个连接都是正确的。LED 不亮的原因通常是它的正负极接反了，或是针脚插入了面包板上错误的孔以致不能连接成完整的回路。

使 LED 闪烁

现在你知道你的 LED 已经被正确连线，并能够正常工作。最后一件事就是用一个 Python 程序来使其闪烁。

1. 启动 IDLE，单击 File⇨New 创建一个新程序，并保存为 `testLED.py`。
2. 导入时间模块以在 LED 两次闪烁之间插入延时。

```
import time
```

在这个冒险的所有程序中，我使用了 `import RPi.GPIO as GPIO` 或 `import anyio.GPIO as GPIO`——这就相当于你之前使用的 `import mcpi.minecraft as minecraft`。其中的 `as` 告诉 Python 重命名模块来使用右侧的名称。这对于一些很长的模块名称来说很有用。这也意味着冒险的教程和代码只需改动一两行就可以在 Arduino 和树莓派上工作。这是 Python 的一个非常巧妙的特性！

3. 导入控制 GPIO 的模块，并为 GPIO 号定义一个常量，它会被连接到你的 LED，代码如下：

在树莓派上：

```
import RPi.GPIO as GPIO
LED = 15
```

在 Arduino 上：

```
import anyio.GPIO as GPIO
LED = 15
```

4. 将 GPIO 设为针脚编号模式（在代码中可以看到字母 BCM，对树莓派来说意为 Broadcom，也就是处理器的名称。这意味着告诉树莓派使用 GPIO 号而不是板上的针脚号）。然后将 GPIO 配置为输出，这样你的程序就可以改变加在 LED 上的电压。

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(LED, GPIO.OUT)
```

5. 现在编写一个让 LED 闪烁一次的函数。函数使用了一个参数 t 以便之后改变 LED 闪烁的速度。`output()` 函数能改变 GPIO 上的电压，设为 `true` 时电压将会提高到 3.3 伏特，设为 `false` 时电压将会降至 0 伏特。这将会开、关 LED。

```
def flash(t):
  GPIO.output(LED, True)
  time.sleep(t)
  GPIO.output(LED, False)
  time.sleep(t)
```

6. 编写一个游戏循环来使 LED 闪烁，并在程序结束时使用 `GPIO.cleanup()` 函数来将 GPIO 针脚在结束时置于安全状态。你之后可以在“深入代码”栏目中读到有关 `try/finally` 的内容。

```
try:
  while True:
    flash(0.5)
finally:
  GPIO.cleanup()
```

7. 为使 LED 闪烁，需要将其连接至计算机上的 GPIO 针脚，需要做这些事：

在树莓派上：

移动连接电阻器和 3.3 伏特电源导轨的导线，把它连接到导轨的那端连到 GPIO 15（Pi-T-Cobbler 标签为 G15）。

在 Arduino 上：

移动连接电阻器和 3.3 伏特电源导轨的导线，把它连接在导轨的那端连到 Arduino 上的 GPIO 15。

Arduino 板本身带有数个 LED，上面的黄色 LED 表明 Arduino 正在从 USB 接收指令。你会看到这个 LED 也在闪烁，但频率是面包板上 LED 的两倍。这是因为所有开启和关闭面包板上 LED 的指令都会使 Arduino 的黄色 LED 闪烁。不要被这个现象所分心，这是正常的！

运行 GPIO 程序

最后，是时候运行你的程序，并看看 LED 是否在向你闪烁了！

在 PC/Mac 上：

你可以和平常一样在 IDLE 上运行程序。Arduino IO 板已经用 USB 连接好，而你的 PC/Mac 拥有对连接的完全访问权限。

在树莓派上：

在树莓派上运行程序略为复杂。虽然它们可以使用 IDLE 来编辑，但使用 GPIO 的程序需要不同的方式来运行，你可以在“深入代码”栏目中读到为什么需要做这些额外的步骤。

1. 用“桌面”或“菜单”打开LXTerminal程序。这会打开一个命令提示符窗口，以供向树莓派输入命令。
2. 输入命令，切换至 `MyAdventures` 文件夹。

```
cd MyAdventures
```

3. 用 sudo 运行你的 Python 程序（在“深入代码”栏目中有相关信息）：

```
sudo python testLED.py
```

你的 LED 在闪烁了吗？这看起来是一件小事，但由此你已经迈出了重大的一步。现在你可以控制电路了，你突破了计算机的界限！

在此你可能会看到一个警告“Channel already in use”。不要担心，这是正常的。在这个冒险之后的部分你会学到如何使用 GPIO.cleanup() 来避免这条警告信息。

最后要做的事就是把这个程序和 Minecraft 世界联系起来，这也是你下一步将要做的。

用 Python 程序使 LED 闪烁，就相当于电子设备中的“Hello World”。当你连接计算机和电路时，在更加深入之前首先试着让这个电路正常工作，这是很重要的。如果你的 LED 不闪烁，仔细检查你的接线，确认你的 LED 和 GPIO 针脚被插入了面包板上正确的孔中。你在之前已经测试了 LED 可以用计算机的 3.3 伏特电源点亮，所以如果它不工作的话，不是你的 Python 程序有错误，就是 LED 和 GPIO 针脚间的连线有错误。

深入代码

你刚才使用到了两种“魔法”，你可能愿意更深入地了解。

首先，`try/finally/GPIO.cleanup()` 是做什么的?

为了避免每次运行程序都出现警告，每次程序结束时都要记得调用 `GPIO.cleanup()`。它将所有 GPIO 针脚重置为输入，并禁用 GPIO 线路。“重置为输入”意味着针脚不会再主动控制电路。如果你在接线或断开时偶然短路了两个针脚，电路可能被破坏。为了防止这类事情发生，让程序调用 `GPIO.cleanup()` 函数是个好习惯，这样改变电路时，偶然的短路就不会破坏你的电路了。

但也有一个问题：你在 Python 程序中使用了 `while True:`，这意味着你的程序进入了一个无限循环，唯一停止程序的办法就是按下 Ctrl+C 或者从 IDLE 菜单停止程序。但如果你如此停止一个程序，它立刻就终止了。

不必担心，有一种方法可以解决这个问题，它使用了 `try/finally`——一种异常处理程序。本书中不会介绍异常处理程序的细节，但你在 GPIO 中总要用到它们！基本上说，当你用 Ctrl+C 或者 IDLE 菜单结束程序时，Python 产生了一个叫作 `KeyboardInterrupt` 的异常来让程序停止。使用 `try/finally` 是一个程序员的小技巧，可以让你的 Python 程序检测到程序已停止。在 finally 语句后面你可以放置需要在结束后执行用来清理的任何 Python 代码。因此当你按下 Ctrl+C 或使用 IDLE 菜单停止程序时，程序跳转到 `finally:` 块中的代码并执行，然后终止。这保证了 `GPIO.cleanup()` 每次都被调用，在程序终止时所有 GPIO 针脚都处在干净、安全的状态。

我不会在书中介绍更多关于程序异常的知识了，访问 https://wiki.python.org/moin/HandlingExceptions 了解更多。

其次，树莓派上的 sudo 是做什么的？为什么需要用 LXTerminal 运行你的 GPIO 程序？

树莓派上的所有硬件都是被保护的，你作为一般的 pi 用户登录是无法访问的，但你可以用超级用户权限来代替。sudo 意为“替换用户并执行”，基本上是把你的用户从 pi 改变为 root（超级用户），然后用 Python 语言运行你的 testLED.py。这意味着你的程序可以访问受保护的 GPIO 硬件。如果你不使用这种办法在树莓派上运行 GPIO 程序，你会收到一条错误。

在撰写这本书时，我曾尝试为 sudo 寻找一个清楚的解释，但有趣的是，所有 Google 搜索都把我带回了我博客上很受欢迎的一页。访问 http://blog.whaleygeek.co.ul/sudo-on-raspberry-why-and-why-you-need-to-ask-kids-too 了解更多关于 sudo 和它的应用的知识。

编写“魔法门垫”LED 程序

关于你的 LED，最后一件要做的事就是把它连接到 Minecraft 游戏。幸运的是，最后一步很简单，所有步骤都是你知道怎么做的。

1. 从“菜单”选择 File⇨Save 来保存已有的 testLED.py，将其另存为 welcomeLED.py。
2. 在文件顶端加上导入所需 Minecraft 模块的代码，将其连接到 Minecraft 游戏：

```
import mcpi.minecraft as minecraft
import mcpi.block as block
mc = minecraft.Minecraft.create()
```

3. 在它下面设置一些常量，这会成为你的“欢迎回家”地毯的坐标。你需要决定地毯在 Minecraft 世界中出现的位置并选择相应的坐标。在树莓派上，你可以在 Minecraft 窗口的左上角看到你的坐标。在 PC/Mac 上，你可以按下 F3 来查看玩家坐标：

```
HOME_X = 0
HOME_Y = 0
HOME_Z = 0
```

4. 在这些常量下面，在地毯的位置创造一个羊毛方块。这就是你的门垫——就像冒险 2 里所创造的一样：

```
mc.setBlock(HOME_X, HOME_Y, HOME_Z, block.WOOL.id, 15)
```

5. 修改你的游戏循环成下面这样，注意缩进。这段代码重复读取玩家的位置，并使用“地理围栏”来判断你是否在门垫上。如果你在上面，LED 会闪烁，让你知道你到家了：

```
try:
  while True:
    pos = mc.player.getTilePos()
    if pos.x == HOME_X and pos.z == HOME_Z:
      flash(0.5)
```

```
finally:
  GPIO.cleanup()
```

遵循之前“运行 GPIO 程序”一节中提到的方法运行程序，并确认当你站在门垫上时，LED 闪烁着欢迎你回家！

多么美妙！你现在拥有了突破 Minecraft 沙盒的虚拟世界并将游戏与现实对象连接所需的全部要素。如果你能够编写程序来使一个 LED 闪烁，你也一样可以控制任何其他电子设备——唯一的限制就是你的想象力和动力了！

挑战

用一些不同颜色的 LED 并连接到计算机的 GPIO 针脚上。使用一个蓝色 LED 以及 getblock() 函数，在 Python 程序加入代码使当你站在水中时点亮 LED。使用一个绿色 LED，在你站在草上时亮起……你可以发明出多少种事物，在 Minecraft 世界中发生事情时点亮 LED？

使用七段数码管

你在这个冒险中的下一个工程将会用到一种非常常见的 LED 数码管，在 Minecraft 中发生事情时显示数字和其他符号。你也将会在书中最后的冒险中使用这种数码管。在你把它连接到 Minecraft 之前，首先要把数码管接电并让其工作。在完成这步之后，即可将其连接到 Minecraft 游戏，就像连接闪烁 LED 一样简单。

在这个工程中，如果你在使用 PC/Mac，你需要使用 anyio 软件包。这个软件包在初学者工具包中提供，也可以在冒险开头列出的链接中下载。

什么是七段数码管

如果你在房子里或是去商店看一看，你会发现数百种使用七段数码管的产品。数码手表、计时器或是微波炉、CD 播放器，甚至是中央供暖控制系统，都在使用七段数码管的图案。图 5-10 展示了七段数码管的独特图案以及其内部的布线方式。

所有的七段数码管都拥有一些 LED，每个组件的 LED 数目相同。最常见的，也是在下一个工程中将要用到的七段数码管中，每个组件含有 8 个 LED，但其中一个是用来在下方表示小数点的。只有七个 LED 用来显示实际图案，如图 5-10 所示，展现出一个“8”的形状。

为了下一个工程，关于七段数码管有两件重要的事情需要记住：

- 由于数码管内部有八个独立的 LED，如果你可以编程来打开和关闭 LED，你就可以轻松地编写程序来控制这个数码管，只需八个独立的 GPIO 连接即可。
- 当你使用七段数码管的图案时，你可以通过不同的 LED 开闭组合来显示许许多多的符号，0 ~ 9 的所有数字都是可以显示的。

图 5-10 中不仅可以看到七段数码管内部的布线，也可以看到哪个针脚和哪个 LED 相连。你可以看到每个 LED 的正极都被连接到同一个针脚（CoM），这也是你将要在面包板上连接到 3.3 伏特电源导轨的针脚。二极管的箭头指明了电流通过的方向，同时也是从正极指向负极。

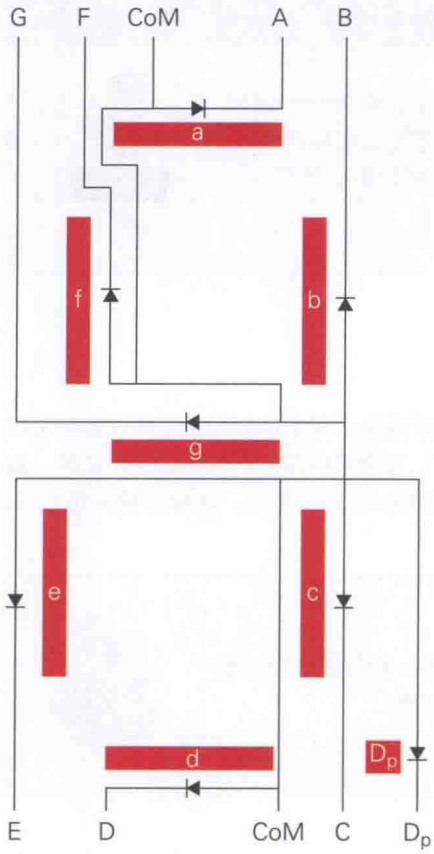


图 5-10　七段的布局以及一个共阳极七段数码管的内部布线

标签 A B C D E F G 是工业上标识七段数码管的通用约定。如果你仔细观察，会发现许多七段数码管的发光段微微向右倾斜，这只是一个许多制造商都在用的设计特性，让七段数码管更具视觉吸引力。

你可以买到很多种七段数码管，它们的连线方式并不相同。有些是共阴极的，（所有 LED 的负极针脚被连在一起）而有些是共阳极的（所有 LED 的正极针脚被连在一起）。你还会发现有些七段 LED 数码管模块为每个 LED 部分设置了不同的针脚。在我们的工程中，使用的是一种特殊的共阳极的七段数码管，你可以在 www.skpang.co.uk 上找到。如果你买了其他的数码管，请在连线之前仔细检查它的布线细节。如果你的数码管是不同的（即共阴极），我已经在所有程序的“ON”处写了注释，这样你就能知道拿到不同的数码管的时候该怎么修改。

为七段数码管接线

现在你已经准备好了将七段数码管连接到面包板上，检查它是否正常工作，步骤如下：

1. 将七段数码管插入面包板，使一半重叠在面包板上半边，另一半重叠在下半边。在顶部和底部各有 5 个针脚，每个针脚要插入面包板上的一个孔。

2. 用导线将上面中间的针脚（公共阳极）连接到面包板顶部的 3.3 伏特电源导轨（如果你在使用共阴极数码管，你需要从下方中间的针脚连一条线到面包板底部的 0 伏特电源导轨）。

3. 由于数码管内有 LED，你需要像之前一样使用电阻器，但有 8 个 LED，因此需要 8 个电阻器。将每个 330 欧姆电阻器（色环是橙、橙、棕）的两个针脚弯折，将一端连接到数码管的一脚，另一端连接到面包板上的一个闲置插孔。图 5-12 展示了数码管需要如何连接。回去看图 5-4 可以帮助你记起面包板内部是如何连接起来的，以免你把所有的电阻器短路在一起！

4. 现在可以测试你的数码管了。用导线临时连接 0 伏特的电源导轨和各个未连接的电阻器针脚（即没连到数码管的那个针脚）。每当你用导线接触一个针脚时，数码管中的一段应该会点亮。将点亮情况和图 5-10（这是我在工程中所用的数码管）进行对比，并确认你的数码管内部布线和我的是否相同。如果不同，记录下你的针脚的连接情况，稍后将会用到。

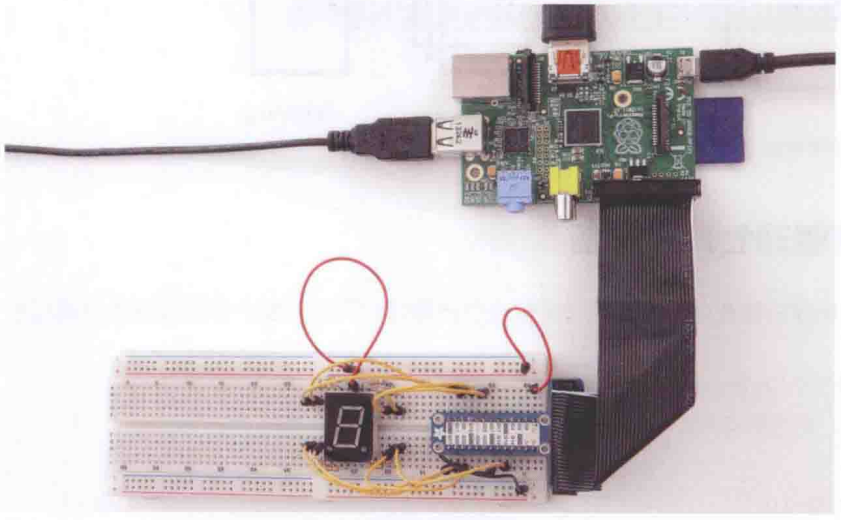


图 5-11　七段数码管连接就绪，准备使用树莓派进行测试

如果你已经测试了你的数码管并确定它正常工作，下一步就是把它连接到你的计算机 GPIO，使用 Python 程序来控制每个 LED。从每个电阻器的未连接针脚引一条线到不同的 GPIO，树莓派如图 5-12 所示，Arduino 如图 5-13 所示。如果在此之前的测试中你的数码管连接方式和我不同，请根据自己数码管的连接情况连接到正确的计算机 GPIO 针脚，否则运行程序时将会得到一些奇怪的图案！

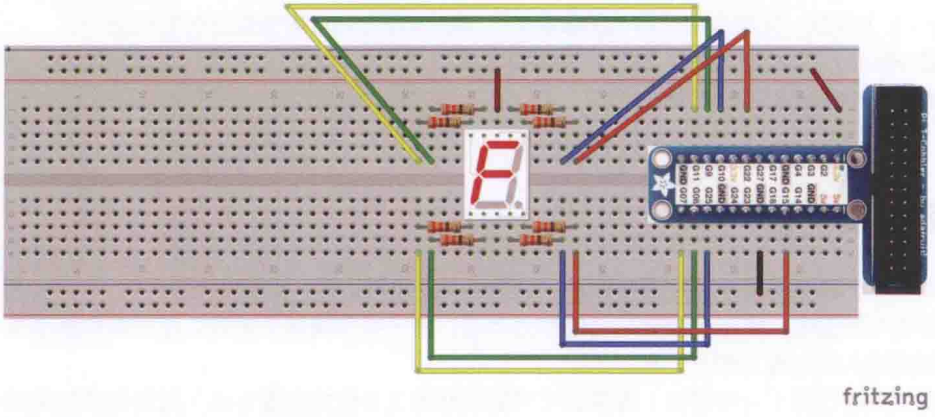


图 5-12　树莓派上的七段数码管接线图

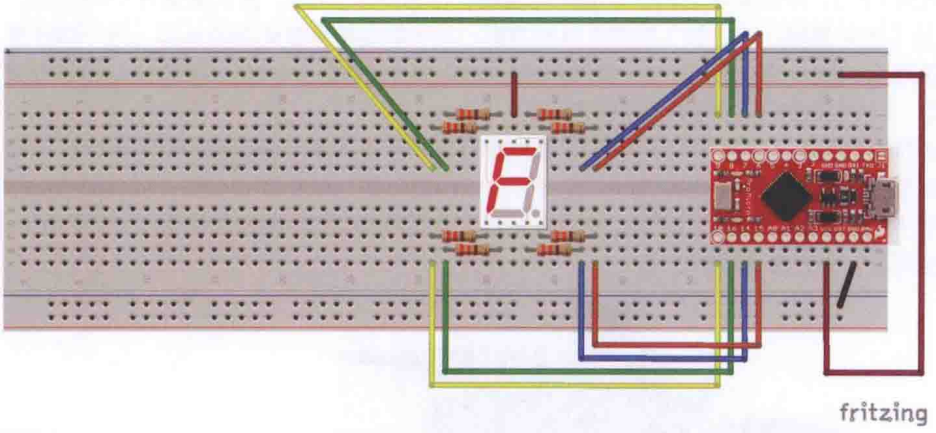


图 5-13　Arduino 上的七段数码管接线图

编写 Python 程序来驱动七段数码管

现在你已经测试了你的七段数码管并且已经连接到所有八个计算机 GPIO 针脚，你已经准备好编写控制它的 Python 程序了。

1. 单击 File⇨New 新建一个 Python 程序，保存为 `testDisplay.py`。
2. 导入所有必要的模块，并为所有 8 个 GPIO 号设置常量。

在树莓派上：

```
import RPi.GPIO as GPIO
LED_PINS = [10,22,25,8,7,9,11,15] # 顺序很重要
```

在 Arduino 上：

```
import anyio.GPIO as GPIO
LED_PINS = [7,6,14,16,10,8,9,15] # 顺序很重要
```

3. 正确设置 GPIO 编号模式：

```
GPIO.setmode(GPIO.BCM)
```

4. 选择输出类型（我在电路中使用的是常见的共阳极数码管）：

```
ON = False # False= 共阳极，      True= 共阴极
```

5. 编写一个循环来将所有 8 个针脚设置为输出：

```
for g in LED_PINS:
  GPIO.setup(g, GPIO.OUT)
```

6. 数码管中所有的 LED 都是非开即关的，因此一个建立图案的好方法就是在一个表中存储 8 个 `True` 或 `False` 值，每个代表数码管中的一个 LED。还记得你在冒险 4 中用到了表和索引吗？实际上和那个完全一样。由于相应的值被设为 `True`，下面的表将会点亮 A、B、C、D、E、F、G 段，但小数点会被关闭，因为最后一个位置储存了一个 `False` 值。表中的索引 `[0]` 对应 A 段，索引 `[1]` 对应 B 段，以此类推。

```
pattern = [True, True, True, True, True, True, True,
  False] # ABCDEFG （没有亮点 DP）
```

7. 编写一个循环来让你所选的图案对应的所有针脚变为打开状态：

```
for g in range(8):
  if pattern[g]:
    GPIO.output(LED_PINS[g], ON)
  else:
    GPIO.output(LED_PINS[g], not ON)
```

8. 现在你的程序需要被改写为等待你输入一个键来结束，否则当程序结束时所有 LED 都会熄灭，你不会看到任何东西。使用 `raw_input()` 函数会让程序等待你键入回车，然后才继续执行下一行：

```
raw_input("finished?")
```

9. 最后，告诉程序在结束前清理 GPIO：

```
GPIO.cleanup()
```

运行程序，观察发生了什么事情。另外记住在树莓派上，你需要从 LXTerminal 窗口用 `sudo python testDisplay.py` 来运行。

挑战

尝试找到数码 0-9 所对应的表中的值，表中的 True/False 值是按照 A、B、C、D、E、F、G、DP 排列的，可以在数码管上试试看。在七段数码管上，你能设计出多少个字母？你可以在这个维基百科条目上找到一些有用的七段数码管图案：http://en.wikipedia.org/wiki/Seven-segment_display。

使用 Python 模块来控制数码管

你可能已经发现了可以在七段数码管上显示许多种不同的图案。为了更简单地使用这些图案，我编写了一个小的 Python 模块名叫 `seg7.py`，包含在初学者工具包中的 `anyio` 文件夹里。你可以任意使用和编辑这个模块，或是把它加入到你自己的游戏或工程中。你可以遵循以下步骤简便地将其连接到一个已有的程序。

1. 单击 File⇨New 新建一个 Python 程序，保存为 `testDisplay2.py`。
2. 导入我的显示模块，以及用添加延时的时间模块。

```
import anyio.seg7 as display
import time
```

3. 从 `testDisplay.py` 中复制从开头到 `ON=False` 的所有行到本程序。
4. 然后建立数码管模块。让它可以访问你的 GPIO 针脚，并给它一张表，对应每段的针脚号：

```
display.setup(GPIO, LED_PINS, ON)
```

5. 编写一个游戏循环，重复地从 0 数到 9。我们把它设为一个循环，这样你就可以让它无限地运行，如果数码管的一些段不工作，你也可以方便调整线路，而不需要一次次地重启程序。

```
try:
  while True:
    for d in range(10):
      display.write(str(d))
      time.sleep(0.5)
finally:
  GPIO.cleanup()
```

保存并运行程序，观察发生了什么事情。数码管应该从 0 数到 9 并不断重复此过程。另外记住在树莓派上，你需要从 LXTerminal 窗口用 `sudo python testDisplay.py` 来运行。

你可以用一个更长的延时（2 秒）来代替上面程序的 `time.sleep(0.5)`，甚至是用一个 `raw_input()`，这样数码管会不变，方便你调整线路到正确的位置。

挑战

打开存储在 `MyAdventures/anyio` 文件夹的 `seg7.py` 程序，遵循文件中的指导来添加新的符号。试试看在这个模块中能编写出多少种在之前的挑战中设计的字母。使用这个数码管你可以写出怎样的单词？你可以逐个字母写出自己的名字吗？字母表中哪些字母不能被打出来？

制作一个引爆器

作为此次冒险的最后一个工程，你将要制作一个又大又红的引爆按钮。当你按下的时候，七段数码管上将会显示一个倒计时来供你逃离爆炸，然后一个大弹坑将会出现在按下按钮时你在游戏中所站的位置。这将为你的工具箱增加一种快速清理一些空间的方法，在你在 Minecraft 世界中游荡和建造时很有用。这个工程将会介绍一种新类型的 GPIO 它输入它就将会感应一个按钮的按下。与此同时，你的七段数码管也会发挥重要作用！

访问我们的配套资源网站 www.wiley.com/go/adventuresinminecraft 中冒险 5 的视频，观看关于如何建造和玩引爆器游戏的教程。

在这个工程中，如果你在使用 PC 或 Mac，需要使用 `anyio` 软件包，这个软件包在新手包中提供。

给一个按钮接线

工程的第一步是给一个按钮接线，这样你可以按下按钮来启动引爆器。使用一个又大又红的按钮可以让它看起来令人兴奋！

你将会用到一个不会锁定的、常闭的按钮。也就是说按钮平时不会连通电路，只有当你按下时才会连通。并且按钮不会保持按下，当你移开手指时电路就会再次断开。

图 5-14 展示了树莓派所用的电路图，图 5-15 是 Arduino 所用的电路图。你可以按照图片或是按照下面的步骤来搭建，两者中的任意一种都是可以的。

1. 将你的按钮压进面包板上的一处空闲位置，我所使用的按钮（见图 5-16）拥有四个针脚，正好可以横跨在面包板中间的槽上。在把按钮针脚按照图 5-14 和图 5-15 所示的方向放置时，按下按钮将会把左边的两个针脚和右边的两个针脚联通。
2. 你需要一个电阻器来将按钮的输入电压“提高”到 3.3 伏特，否则按钮将产生一个虚假的按压，使得引爆器保持常开（阅读“深入电路”栏目以了解为何这个电阻器是必要的）！把 10 千欧姆的上拉电阻器（色环是棕、黑、橙）的一个针脚连接到按钮左侧的一个针脚，另一个针脚连接到面包板顶端的 3.3 伏特电源导轨。
3. 用一条导线的一端连接电阻器和按钮之间的位置，另一端连接 GPIO 的四号针脚。
4. 用导线连接按钮的右下针脚和面包板底部的 0 伏特电源导轨。你可以阅读“深入电路”来更深地了解这条线的作用，所有的按钮都是这样与计算机相连接的。图 5-16 展示了一个连接到 GPIO 针脚的按钮。

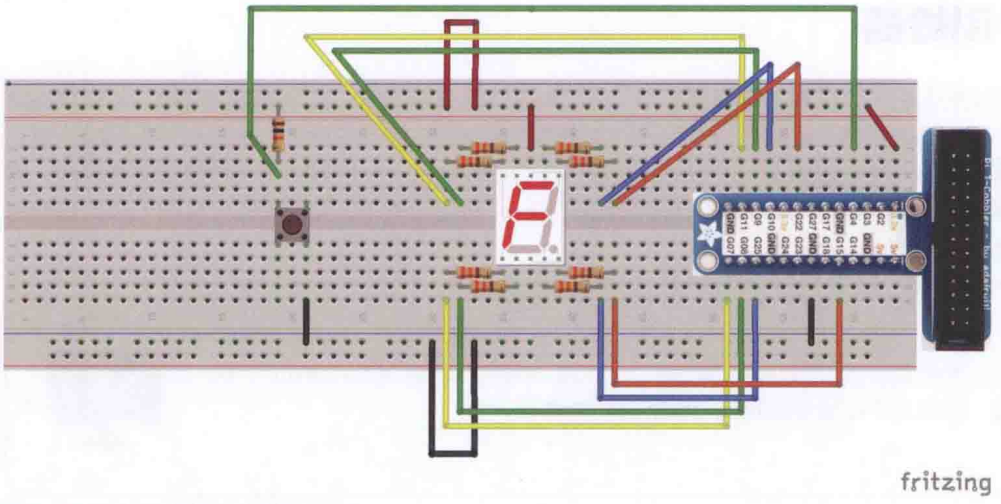


图 5-14　连接到树莓派的按钮电路图

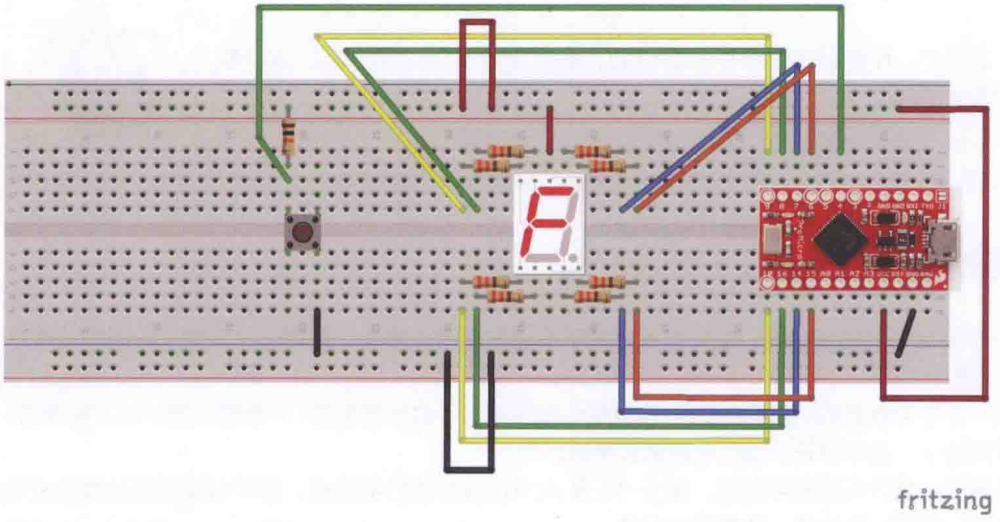


图 5-15　连接到 Arduino 的按钮电路图

有许多种不同的按钮，如果你使用的是和我同一种按钮，你会发现按钮的四个针脚被略微弯曲以顺利插入印刷电路板中以备焊接。当你把按钮插入面包板时，千万小心不要弄断针脚。均匀而稳定地按压整个按钮，应该能够让它正确入位。如果你有一把小镊子，你可以使用它来把针脚完全压扁，这样就可以更轻松地插入面包板。

深入代码

当你为按钮接线时，你将其与一个电阻器相连。这个电阻器被称为“上拉电阻器”，它将按钮针脚上的电压“提高”到 3.3 伏特。

当按钮没有被按下（无连接）时，连接 GPIO 和按钮电阻器连接点的导线处于 3.3 伏特，也就是 GPIO 将会处理为数字“1”的电压。当你按下按钮时，其内部机构将会连接左边和右边的针脚，GPIO 连线的电压会骤降至 0 伏特，GPIO 将其处理为数字“0”。电阻器的电阻值相当高（10 千欧姆，即 10 000 欧姆），否则将会短路 3.3 伏特和 0 伏特的两条电源导轨，你的计算机可能会重启！这是不被推荐的，因为这在某些情况下会损坏你的计算机。“10k”是工程师们对 10 000 欧姆电阻器的简写。

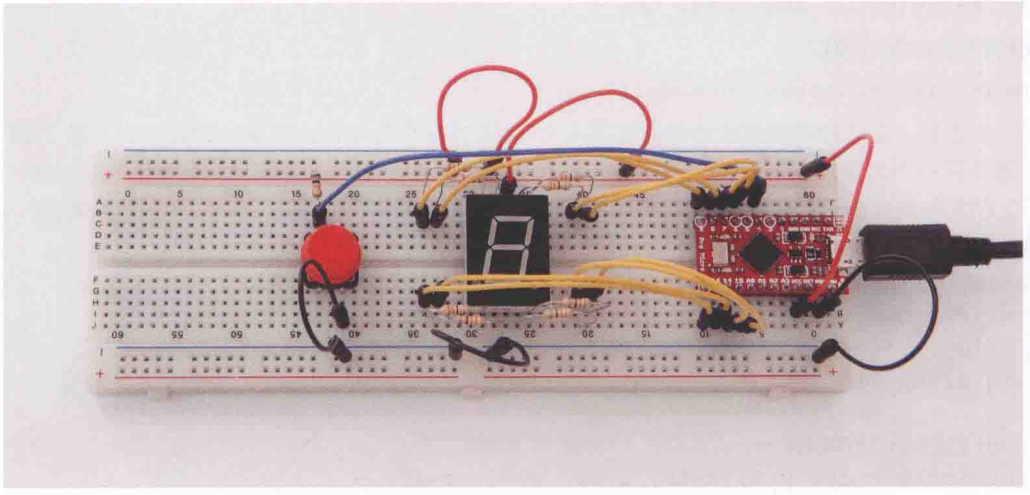


图 5-16　这块面包板上的按钮被连接到一个 Arduino 上的 GPIO，并拥有一个上拉电阻器

编写引爆器程序

现在你已经连接好了你的按钮，最后一步就是编写 Python 程序。这个程序会在每个游戏循环监视按钮的状态，当它在 GPIO 针脚上感应到数字“0”（也就是按钮被按下）时，它会在七段数码管上从 5 倒数至 0，然后在 Minecraft 世界中炸出一个大弹坑。

1. 单击 File⇨New 新建一个 Python 程序，保存为 `detonator.py`。
2. 导入必要的模块：

```
import mcpi.minecraft as minecraft
import mcpi.block as block
import time
import anyio.seg7 as display
```

3. 为你的计算机配置 GPIO：

在树莓派上：

```
import RPi.GPIO as GPIO
BUTTON = 4
LED_PINS = [10,22,25,8,7,9,11,15] # 顺序很重要
```

在 Arduino 上：

```
import anyio.GPIO as GPIO
BUTTON = 4
LED_PINS = [7,6,14,16,10,8,9,15] # 顺序很重要
```

4. 设置按钮所对应的 GPIO，让其成为“输入”，并配置显示用 GPIO。

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(BUTTON, GPIO.IN)
ON = False # False= 共阳极，        True= 共阴极
display.setup(GPIO, LED_PINS, ON)
```

5. 连接到 Minecraft 游戏。

```
mc = minecraft.Minecraft.create()
```

6. 编写一个函数，在炸弹将要爆炸的位置放下一个 TNT 方块作为标识。当倒计时归零时，在 TNT 方块所在的位置炸开一个大弹坑。下面的代码将会在玩家的旁边创造 TNT，以防 TNT 降落在玩家上面！注意弹坑是 20 方块大小的（玩家左面 10 格和右面 10 格），这个计算将在这里由 `setBlocks()` 完成：

```
def bomb(x, y, z):
  mc.setBlock(x+1, y, z+1, block.TNT.id)
  for t in range(6):
    display.write(str(5-t))
    time.sleep(1)

  mc.postToChat("BANG!")
  mc.setBlocks(x-10, y-5, z-10, x+10, y+10, z+10, ↩
  block.AIR.id)
```

7. 编写主要的游戏循环来让它等待按钮按下，然后才启动炸弹。当按钮按下时它会被连接到 0 伏特，因此为了探测按钮按下，你需要将 GPIO 与 `False` 做比较。

```
try:
  while True:
    time.sleep(0.1)
    if GPIO.input(BUTTON) == False:
      pos = mc.player.getTilePos()
      bomb(pos.x, pos.y, pos.z)
finally:
  GPIO.cleanup()
```

保存并运行程序。另外记住在树莓派上，你需要从 LXTerminal 窗口用 `sudo python testDisplay.py` 来运行。

在你的 Minecraft 世界中跑到一个地方，然后按下按钮——赶紧逃离！图 5-17 展示了爆炸后的场面——一个大弹坑。

图 5-17　在 Minecraft 世界中炸出的一个弹坑

弹坑的尺寸可以通过 `setBlocks()` 语句中的数字来改变。你认为改变弹坑尺寸是否是一种好方法呢？你如果想要两倍大的弹坑，需要改变多少程序代码？你是不是认为我有点懒惰？你为什么不自己去试试，完善程序来方便改变弹坑尺寸呢？

挑战

为了让引爆器更加激动人心，试着加入一些区域限定代码——就像冒险 2 中那样来检测玩家是否还在爆炸区内。如果爆炸时你还在爆炸范围内，使用 `mc.player.setTilePos()` 来把玩家弹射到空中。

挑战

为你的电路加入另外 3 个按钮，就像加入第一个按钮那样。编写更多的 Python 代码来让这些按钮做不同的事情，例如在 Minecraft 聊天中发送随机信息，传送玩家到秘密地点，在玩家站立的位置创造各种方块，甚至是在站立位置建造一栋房子。下面的 GPIO 号在加入新按钮时是可以安全使用的。

按钮	树莓派编号	Arduino 编号
BUTTON2	14	5
BUTTON3	23	2
BUTTON4	24	3

在完成冒险后，保存好你布好线的面包板。Martin 将会在最终的冒险 9 中再次使用完全相同的电路布局，我也将会在附加章节中再次使用，你可以在 Wiley 的网站上下载该章节。

快速参考表

配置 GPIO 针脚

```
import RPi.GPIO as GPIO # RaspberryPi
import anyio.GPIO as GPIO # Arduino
GPIO.setmode(GPIO.BCM)
GPIO.setup(5, GPIO.OUT) # 输出
GPIO.setup(5, GPIO.IN) # 输入
```

读取和写入的 GPIO 针脚

```
GPIO.output(5, True) # set on (3.3V)
GPIO.output(5, False) # set off (0V)
if GPIO.input(6) == False:
  print("pressed")
```

使用 GPIO 后安全清理

```
try:
  do_something() # put code here
finally:
  GPIO.cleanup()
```

使用七段数码管模块

```
import anyio.seg7 as display
LED_PINS = [10,22,25,8,7,9,11,15]
ON = False # 共阳级
ON = True # 共阴极
display.setup(GPIO, LED_PINS, ON)
```

向七段数码管输出字符

```
display.write("3")
display.write("A")
display.write("up")
```

其他七段数码管的函数

```
display.setdp(True) # point on
display.setdp(False) # point off
display.clear()
display.pattern([1,1,1,1,1,1,1,1])
```

更多关于电路的冒险

在冒险中，你连接了 Minecraft 世界和现实世界。借助检测和控制现实物品，你将视野扩大到迷人的物理计算世界。使用学到的新知识，你使 LED 闪烁，编写七段数码管图案，并检测按钮的按下。但最重要的是，你已经突破了 Minecraft 沙盒世界的限制，使用学到的新知识，你可以制作自己的游戏控制器和显示设备！

本书没有足够的篇幅来进行太多和电路相关的游戏，但当我开始思考这个理念时，新想法一个接着一个！附加章节中的 Minecraft 电梯是我想让你玩的游戏之一，这就是为什么我把它放到了 Wiley 的配套资源网站上供下载——一定要试试看。我花了数个小时在 Minecraft 世界里乘电梯上上下下，从高远的天空到幽暗的地下，甚至围绕着它建造了一整座高塔！我想要有朝一日围绕着电梯建成一个虚拟的碎片大厦。你能在这方面击败我吗？

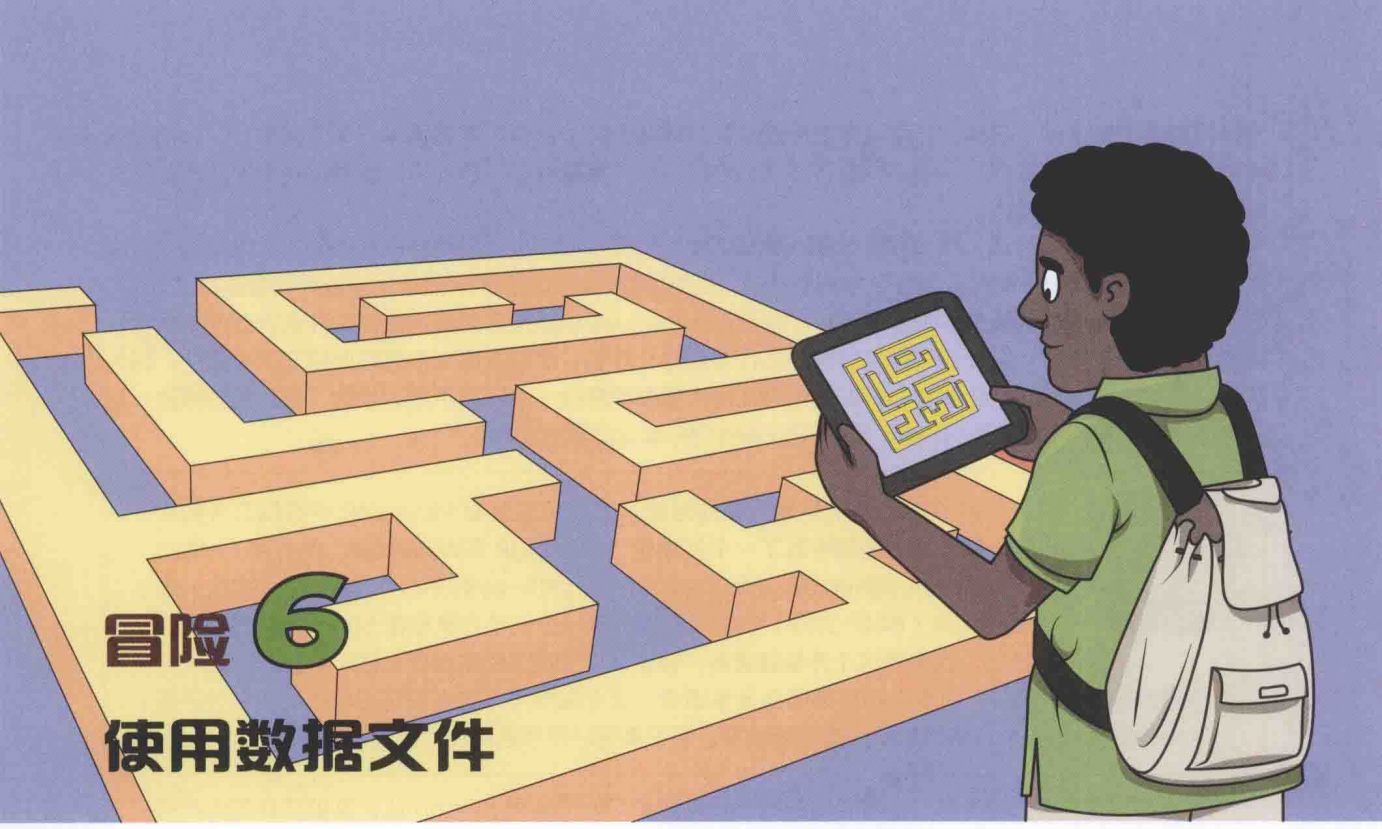
我们树莓派俱乐部的一个孩子有一个很好的想法——在卡片上建造一幅冒险地图，并将 LED 塞入其上的孔洞。想法是在卡片上画出代表他们的 Minecraft 世界的地图，而嵌入卡片中的五个 LED 每个都会在你在游戏中完成一些成就后亮起。我很想看到这个游戏被设计出来，我也希望有一天某个人能把它建造出来——你会成为那个人吗，你会成为那个精彩游戏的设计者吗？

有人发现了如何将加速度计连接计算机来感应他手臂的运动，并将其转化为机械手臂的运动。这里是视频：http://www.youtube.com/watch?v=_KGc9vlOrNk。试试看你能否将加速度计连接到计算机，用挥动手臂来控制 Minecraft 游戏！如果你能做到这个，何不把它作为一个工程来展示，来让更多人学习关于奇妙的 Minecraft 电子世界的知识呢？

解锁成就：你已经突破了 Minecraft 虚拟世界与现实的界限！你已经成为奇妙的 Minecraft 电子新产业中的先驱者。你的游戏体验从此再也没有边界！

下一个冒险……

在冒险 6 中，你将会学习如何从文件中读取数据来自动建造复杂的大型结构，例如 Minecraft 迷宫。你将会使用你的新技能来建造一台 3D 复印机，可以在 Minecraft 世界中复制大型对象。树木再也不会安稳地扎根在地里了！



冒险 6
使用数据文件

当你开始处理更大量的数据时，编程才变得真正激动人心。你的程序会成为众多规则的集合，用来读取、处理以及直观地表达数据——数据在其中占据了极其重要的地位。

在这次冒险中，首先你将会了解如何通过创建文本文件来设计迷宫。之后，迷宫就能在 Minecraft 世界中自动生成，等待着你和你的伙伴将其解决。你一定会好奇，如此精简的一个 Python 程序是怎样做到在 Minecraft 世界里建造这样的巨大物件的。

然后，你会把这个简单的想法拓展，完成复印室设施，即一个用来进行虚拟 3D 扫描和打印的设施。任何你想在复印室中里建造的东西都能以文件形式储存，并在之后调用，传送到 Minecraft 世界的任何一个角落，甚至加载到其他计算机中的 Minecraft 世界里！你会学习到如何建设自己的物品库，并迅速地在 Minecraft 世界中完成打印，你的朋友们一定也会希望拥有这样一个属于他们自己的复印机！

从文件中读取数据

计算机程序普遍还是不够智能的——它们只会重复执行你发送给它们的指令。但如果你不对你程序的输入进行改变，那么在每次运行时，它们只会执行相同的内容。你已经编写好的那些 Minecraft 程序是可以进行交互的，因为它们是根据游戏世界里发生的事，也就是程序的输入来改变它们所要做的事情。

使用数据文件后你能做的趣事

另外一种运行计算机程序的有趣方式，是将输入的数据存储在一个文本文档中。当你启动程序时，让

计算机读取那些文件。这样，你可以在之后创建任意数量的文本文件，甚至编写一个“菜单”，以便根据你所需要程序执行的操作，从中选择打开文件。这被称作“数据驱动”的程序，因为这是由附加的数据文件决定了这个程序的运行。

如果你仔细考虑的话，你会发现你在计算机上使用的许多程序也是使用数据文件的。你用来输入家庭作业的文字处理器将你的作业保存在数据文件里；当你用数码相机拍照时，照片也是被存储在数据文件里；你的照片编辑程序从数据文件里读取照片，甚至连 Minecraft 也是在幕后使用数据文件来完成例如保存和读取世界之类的任务，以及形成构成世界的各种方块的文件包。使用数据文件的程序是非常灵活的，因为这意味着这个程序不必完全依靠你，即使用者在每次想要程序做一些稍微不同的事时，进行手动修改。如果你的朋友们和你有着相同的程序，你也可以和他们分享这些数据文件。

我写过的一个早期的程序范例，可以用来解释 Minecraft 中数据文件的作用。我在我的博客中写了一个名称是“Minecraft BMP 建造器”的程序（http://blog.whaleygeek.co.uk/minecraft-pi-with-python/），这个程序能从一张位图文件（BMP 文件）中读取图像，并一个一个方块地在 Minecraft 世界中建造它。我使用这个方块建造器，建造了一个巨大的树莓派标志，它是如此之大，以至于当你抬头看时，他看上去似乎周身云雾缭绕！这个程序可以读取任何 BMP 位图文件并在 Minecraft 世界中用方块将这个图像建造出来，而不需要对程序进行任何修改。

为了编写数据驱动的程序，你需要做的第一步是要学习如何使用 python 从计算机文件系统中来打开和读取文本文件。

制作一个提示器

为了学习如何打开并读取文本文件，你将要写一个简单的提示器程序。这个程序将会阅读你预先准备好的包含 Minecraft 提示和技巧的文件。每过一段随机的时间间隔，程序会在 Minecraft 聊天栏中显示一则提示信息。你将会需要一个包含提示的文本文件，所以你的第一个工作是创建这个文件。你可以通过普通的 Python 编辑器来创建这个文件，就如同编写你的 Python 程序那样。

打开 Minecraft、IDLE，如果你使用 PC 或是 mac，也需要打开服务端。到了现在，你应该已经对于启动这些比较熟悉了，但如果你需要的话，可以回到冒险 1 中去查看提示。

1. 在 IDLE 中，从“菜单”中选择 File⇨New Flle，来创建一个新的文本文件。将文件保存为 `tips.txt`。

2. 现在，我将列出 4 ~ 5 个提示，每个提示会使用一行文本。下面是一些 Minecraft 提示的例子，但如果你能编写你自己的提示的话，这会变得更为有趣。请务必确保你在每行文本的行末敲入了换行符（newline），这样才能在文件内部创建新的一行（你会在本次冒险中发现更多有关换行符的知识）。

```
Build yourself a house before the mobs come and get you
Use flowing water to fill underwater channels
Build up with sand then knock out the bottom block
Use both first-person and third-person views
Double tap space bar to fly into the sky
```

3. 保存这个文件。你不必运行这个文件，因为这不是 Python 程序而只是一个让 Python 程序使用

的文本文档。

换行符（newline）是一种不可见的特殊字符，在计算机中用来标志一行文本的结束。

有时它也被称作“回车”，在机械打字机时代回车是非常重要的。任何时候你想要输入下一行时，你都需要使用回车来使打字机的滑动架回到页面的最左边并移动到下一行的位置。

不要在 tips.txt 文件中使用逗号，或者任何你曾在 mc.postToChat() 中使用过的信息。在 Minecraft API 中有一个小的 bug（漏洞），在这里任何在信息文本中的逗号都会导致这条信息在该位置处被截断。

现在你有了一个包含提示的数据文件——换句话说，输入数据——那么就到了编写处理输入数据的程序的时候了。这个程序将会随机读取文件中的提示，并在随机的时间间隔后，将提示信息展示在 Minecraft 聊天栏中。之后当你在玩游戏时，有用的提示会在聊天栏中出现。

1. 单击 File⇨New File 来创建一个新的文件，保存为 tipChat.py。
2. 导入必须的模块：

```
import mcpi.minecraft as minecraft
import time
import random
```

3. 连接到 Minecraft 游戏中：

```
mc = minecraft.Minecraft.create()
```

4. 为文件的名称设定一个常量，这样你可以在之后轻松地通过改变常量的值来读取一个不同的文件：

```
FILENAME = "tips.txt"
```

5. 以只读模式打开文件（查看下面的代码挖掘部分，获取对这一行代码更为完整的解释）：

```
f = open(FILENAME, "r")
```

6. 读取文件中所有的文本行并存储至名为 tips 的列表中去。注意你已经在冒险 4（魔法桥梁建造器）和冒险 5（电路）中使用过列表：

```
tips = f.readlines()
```

7. 当你完成后，请关闭所有的文件。在完成使用后记得关闭文件会是个很好的习惯，因为当文件处于打开状态时它就不能再被其他程序使用，比如第一步时你用来输入文本的那个编辑器：

```
f.close()
```

8. 下面将要编写一个循环，它会在 3 ~ 7 秒的范围内等待一个随机的时间间隔：

```
while True:
  time.sleep(random.randint(3,7))
```

9. 现在需要告诉程序来从 tips 列表中随机选择一则信息，并显示在聊天栏中。你需要使用 strip() 函数来排除不需要的换行符。之后，你会需要更多地关注换行符，所以现在不必过多担心是否彻底理解了这些代码。

```
msg = random.choice(tips)
mc.postToChat(msg.strip())
```

保存并运行你的程序。看看发生了什么？当你在 Minecraft 世界里漫步时，每隔一段时间提示会在聊天栏上出现，正如图 6-1 所示的那样。注意一下 Minecraft 聊天栏以怎样的方式将消息展示在屏幕上，一段时间后消息会逐渐消失。

图 6-1　当你在玩 Minecraft 时，随机显示在聊天栏的提示

挑战

基于你的游戏经验，在 tips.txt 文件中添加更多的 Minecraft 提示。把你的 tips.txt 以及 tipChat.pyt 程序发送给你正在学习 Minecraft 的朋友，让他们通过使用你的程序迅速学习如何在游戏中做到那些惊人的事情。你甚至可以编写一个完整系列的 tips.txt 文件，针对玩家的能力涵盖不同的主题以及不同数量的细节。你可以发布在一个小网站上以供其他玩家配合你的 tipChat.py 程序使用。

深入代码

在 tipChat.py 程序中，一点小小的“魔法”在下面这行里发生：

```
f = open(FILENAME,"r")
```

open() 函数打开被命名为 FILENAME 常量的文件——但句末的 "r" 代表的是什么？当你打开一个文件，你必须告诉 open() 函数你希望将哪一级别的权限下放给那个文件。在这个例子里，"r" 意味着你只是希望读取这个文件。如果你恰巧尝试写入这个文件，你会得到一个错误信息。这能防止你的文件被意外篡改。如果你想要打开并写入这个文件，你可以通过使用 "w" 而不是 "r"（或者当你想要同时读取并写入文件时，使用 "rw"）。

其次，什么是 "f"？f 只是另外一个变量，但却储存着一个文件的句柄。一个文件的句柄只是某种“掌控”文件的东西——就像是一种虚拟的对于计算机文件系统中真实文件的指向。无论何时你想要对打开的文件进行操作，使用 f 来指向这个文件。这很方便，正如任何其他之前你使用过的变量，你可以使用多个文件变量来同时打开多个文件，例如下面这样：

```
f1 = open("config.txt","r")
f2 = open("levels.txt","r")
f3 = open("score.txt","rw")
```

f1，f2 和 f3 都可以被使用在之后的程序中，以便正确地表示你想要读取和写入的是哪个文件。

通过数据文件建造迷宫

现在你知道了如何让你的程序读取数据文件，你就已经为之后的深入冒险做好了准备。在之前的冒险中，你知道了在 Minecraft 世界里搭建方块是件非常简单的事情。但如果你能够使用存储在数据文件中的方块进行搭建的话会发生什么呢？这正是下面即将要带领你来做的事情，在 Minecraft 世界建造一个 3D 迷宫，而迷宫的数据是存储在外部文件中的！但首先，你需要决定如何在文件中存储迷宫的数据。

访问网站 www.wiley.com/go/adventuresinminecraft，选择冒险 6 的视频，观看关于如何建立以及体验你的迷宫游戏的教程。

你设计的迷宫会是 3D 的，因为这是建造在 Minecraft 3D 世界里的，但实际上这只是使用 3D 方块建造的 2D 迷宫——对你的迷宫而言，它只有一层。这意味着数据文件只需要存储迷宫中每个方块的 *x* 和 *z* 坐标的数据，所以那会是矩形的。

你需要决定方块自身是如何以数据文件的形式表达的。你将会使用墙和空气。出于简化的目的，我们将使用 1 来代表墙以及 0 来代表空气。你总是能在 Python 程序里改变之后将被用为墙体的方块类型。

理解什么是 CSV 文件

在这个项目中，你会使用到一种特别类型的文本文件——CSV 文件，表示你的迷宫是在计算机文档系统中的文件。

CSV 文件，即是“逗号分隔值文件”（comma-seperated file）。这些值被简单地存在一个文本文件中，使用逗号进行分隔。CSV 文件也可以用来表示一个简单的表格或者数据库。数据库中的每一行记录在文件中使用一行来表达，每一个数据域或数据列在那一行，通过逗号分隔的文字来表示。一些 CSV 文件使用文件的第一行来存储标题或者文件名称。所有常用的数据表以及数据库程序都能以 CSV 的形式来进行数据的导入和导出。

CSV 是一种简单并且广泛使用的文件形式。下面是一个简单的 CSV 文件的范例，存储了从数据库中获取的部分表格，包含了 Minecraft 游戏角色的一些细节。在这个 CSV 文件中，文件的第一行（或列）保存着这个数据域的名字，而剩下的其他行则保存有用逗号进行分隔的数据：

```
Name,Handle,Speciality
David,w_geek,Coding in Python
Roma,physics_gurl,Designing big buildings
Ryan,mr_teck,Minecraft robots
Craig,rrrrrrrr,TNT expert
```

这个 CSV 文件有一个标题行，并且有着三个名为 `Name`、`Handle` 和 `Speciality` 的数据域。它有 4 个数据行，每一行都存有 3 个数据域的数据。

当你在设计复杂的软件结构时，使用其他工具来帮助你有时也是非常有用的。作为一个软件工程师，在我设计和表示数据时，电子表格是经常会用到的工具之一。电子表格提供了一种巧妙的方法来表示在之后的程序中以 CSV 文件形式进行导入和加载的数据表。你也可以试着通过将列变窄，以及使用数学公式来使输入 `0` 时显示一个白色方格，而当输入 `1` 时显示黄色方格，这样你就能在你最喜欢的电子表格程序中完成迷宫的设计。当你以这种方式可视化完成你的迷宫，将它输出保存为 `maze.csv`，运行你的程序来看看你和你的朋友是否能解决这个迷宫游戏。

对于你的迷宫数据文件，你并不需要一个标题行，因为表格中的每一列都代表相同类型的数据；存储的数字 0 代表着一个空位，而数字 1 则代表着墙。你可以从网站下载这个迷宫数据文件范例，但如果你想自己将它们输入则可以参照下面的做法：

1. 单击 File⇨New File 创建一个新的文本文件。单击 File⇨Save As 将这个文件命名为 `maze1.csv`。

2. 仔细地输入下面这几行，确保每一行及每一列的数目是正确的。如果你非常细致地观察这些数据，你应该已经能够得出这个迷宫的结构了！每一行有 16 个数字共计 16 行，因此如果你用铅笔来勾掉已经输入过的行，也许能避免迷失你的位置：

```
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
0,0,0,0,0,0,0,0,0,0,0,0,0,0,1
1,1,1,1,1,1,1,1,1,0,1,0,1,1,0,1
1,0,0,1,0,0,0,0,1,0,1,0,1,0,0,1
1,1,0,1,0,1,1,0,0,0,0,0,1,0,1,1
1,1,0,1,0,1,1,1,1,1,1,1,1,0,1,1
1,1,0,0,0,1,1,1,1,1,0,0,0,0,1,1
1,1,1,1,1,1,0,0,0,0,0,1,1,1,1,1
1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1
1,0,1,1,1,1,0,0,0,0,0,1,1,1,1,1
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1
1,0,1,1,1,1,1,1,1,1,0,1,1,1,1,1
1,0,1,0,0,0,0,0,0,1,0,0,0,0,0,1
1,0,1,0,1,1,1,1,0,1,1,1,1,1,0,1
1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,1
1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1
```

3. 保存这个文件。一旦你写好了读取和处理这个文件中的数据的Python 程序，你马上就需要用到它的。

建造一个迷宫

现在你已经有了描述迷宫的数据文件，最后一步就是编写从文件中读取数据并在 Minecraft 世界里建造迷宫的 Python 程序了。

1. 单击 File⇨New File 来启动一个新的文件，单击 File⇨Save As 将其存储为 `csvBuild.py`。
2. 导入必要的模块：

```
import mcpi.minecraft as minecraft
import mcpi.block as block
```

3. 连接到 Minecraft 游戏：

```
mc = minecraft.Minecraft.create()
```

4. 定义一些常量。`GAP` 常量是在空白处使用的方块属性。这通常会是空气，但你也可以尝试其他不同的方块类型来让迷宫变得更有趣一些。`WALL` 常量是迷宫中作为墙体的方块类型，所以你需要小心地为这个常量选择方块类型，毕竟一些方块的类型是无法作为墙体来使用的。而 `FLOOR` 常量则是被用来建造迷宫的地基层：

```
GAP = block.AIR.id
WALL = block.GOLD_BLOCK.id
FLOOR = block.GRASS.id
```

不要使用沙子（`SAND`）建造地板层。如果你这么做了，地板层可能会从玩家的脚下坠落，这取决于迷宫所处的不同位置。

5. 打开包含你迷宫数据的文件。FILENAME 常量能帮你轻松地改变需要读取使用的迷宫文件的名字。

```
FILENAME = "maze1.csv"
f = open(FILENAME, "r")
```

6. 获得玩家所在位置，并计算迷宫底部角落的坐标。将 *x* 和 *z* 坐标分别加 1 来保证迷宫不是建造在玩家的头顶。你可以改变 *y* 坐标值将迷宫建造在空中：

```
pos = mc.player.getTilePos()
ORIGIN_X = pos.x+1
ORIGIN_Y = pos.y
ORIGIN_Z = pos.z+1
```

7. *z* 坐标值将会在每一行数据结尾时改变，所以在迷宫初始时先将它设定好。之后 z 值在程序中慢慢变化。

```
z = ORIGIN_Z
```

8. 在文件的每一行中执行循环。详情请见“深入代码”栏目，在那里会解释 readline() 函数的作用。这个 for 循环实际上在文件的每一行处都进行了循环，一个接着一个。每一次进行循环，line 变量存储了读取的文件中的下一行：

```
for line in f.readlines():
```

9. 将每一行分割成部分，即每次出现逗号就进行分隔。“深入代码”栏目里会解释 split() 函数的功能。注意所有 for 循环下的循环体都需要进行一次缩进：

```
  data = line.split(",")
```

10. 细心地执行下面这一步：你现在需要再写一个 for 循环。这叫嵌套循环（nested loop），因为这一个循环是被嵌套在另外一个循环中的。你的程序需要在迷宫中每一行的起始处重置 *x* 坐标的值，因此在读取下一行的 for 循环之前完成这一步。

```
  x = ORIGIN_X

  for cell in data:
```

11. 每一行中每个数字实际上都是以文本（字符）的形式被读取，所以为了让这个程序正确运行，你需要在数字两边放置引号。这个 if/else 语句是基于 CSV 文件中读取的数字来选择是否需要建造一个空档或是墙体。遇到数字 0 则建造一个空档，而任何其他的数字则建造墙体。请务必确保你的缩进是正确的，if 语句需要缩进两次，因为它是在 f.readlines() 循环中 for 循环的一部分。在这里的变量 b 是非常有用的，它使程序简短了许多（你可以尝试重写这部分代码，而不使用变量 b，这样就会明白我所说的是什么意思了！）。

```
    if cell == "0":
      b = GAP
    else:
      b = WALL
    mc.setBlock(x, ORIGIN_Y, z, b)
    mc.setBlock(x, ORIGIN_Y+1, z, b)
    mc.setBlock(x, ORIGIN_Y-1, z, FLOOR)
```

12. 在 `for` 循环末尾，更新 *x* 坐标的值。这里必须进行一次缩进来与之前 `mc.setBlock()` 语句保持对齐，因为这仍然是 for 循环体的一部分。

```
    x = x + 1
```

13. 在处理每一行数据的循环末尾处，更新 *z* 坐标的值。因为这个步骤能使数据的每一行都经过处理（此处只需进行一次缩进，因为这只是在 `f.readlines()` 循环中的 `for` 循环中的循环体）。

```
  z = z + 1
```

保存你的程序，并且再次检查确保你所有的缩进都是正确的。如果缩进出现了错误，那么这个程序就不能够产生正确的效果，而让你得到一些形状非常奇怪的迷宫！

运行你的程序，然后一个不可思议的（并且难以解决的）迷宫就会建造在你面前。在迷宫中漫步，看看你是否能够不借助打破墙体或者飞行，来解决这个迷宫。图 6-2 展示了在地面所在层建造的迷宫的样子。

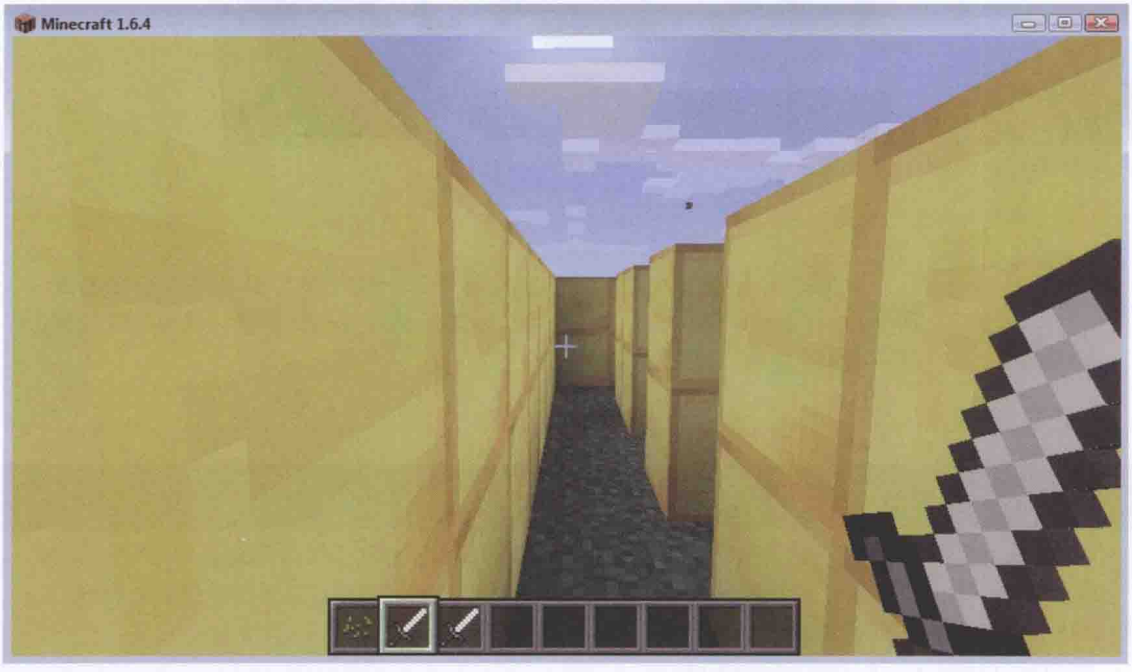


图 6-2　Minecraft 里迷宫的平视图

如果你在某一步卡住了，你总能通过作弊或者飞到空中来得到一个对于迷宫的 3D 鸟俯视图，正如图 6-3 所示的那样。

> 一个嵌套循环（nested loop）是指一个放置在另一个循环中的循环。第一个循环被称为“外循环”，而第二个循环被称为“内循环”。在 Python 中通过对第二个循环进行再次缩进而形成嵌套。你可以进行任意次数的嵌套。
>
> 
>

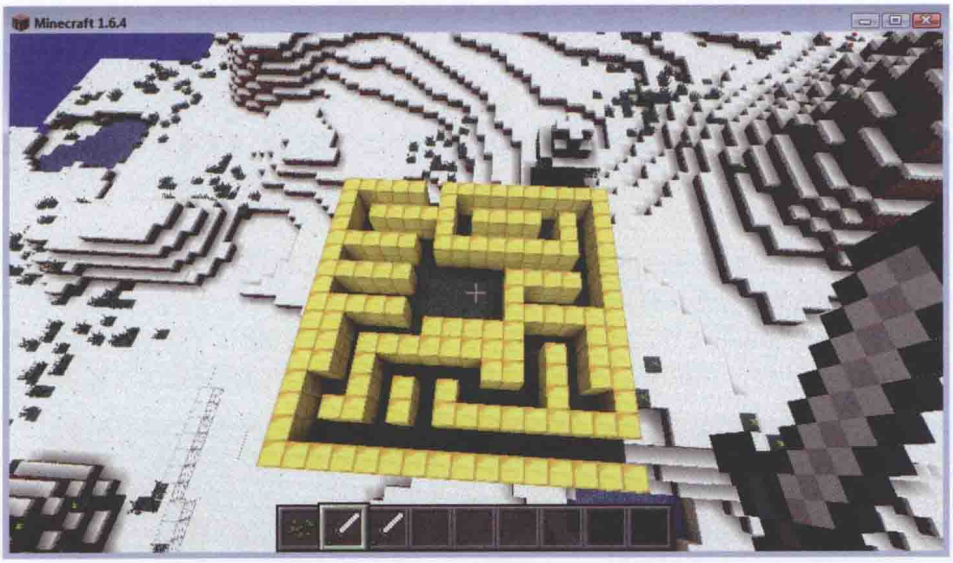


图 6-3　Minecraft 里迷宫的鸟瞰视图

如果你不小心在数据文件的末尾留下了一些空格，那么这些空格会被 Python 程序读取并解读。幸运的是，这并不会破坏你的程序，但你可能会在迷宫结束处发现一些冗余的方块。

深入代码

在之前的冒险中，你已经体验了使用 Python 列表并知道了列表不仅仅只是零个或多个项目的有序集合。许多 Python 内置函数都提供列表形式的数据，你刚在迷宫建造程序中已经遇到了其中的两种。下面的这行属于第一种类型：

```
for line in f.readlines():
```

注意 f 是一个文件句柄——能让你获得打开这个文件的权利。`file` 类型在 Python 里有内置的 `readlines()` 函数，这是非常有用的。这样就能简单地读取文件中的每一行并连接（附加）到列表的末尾。你可能也记得在之前的冒险中，当从在一个列表中取值，`for` 语句会持续执行循环，直至列表中每一项都被取过。

所有的 `for` 语句的作用，都是读取文件中的每一行，存储到一个列表中，并每做一次循环后转到下一行中。每经过一次循环时，行变量（循环控制变量）会获取下一行的数据。我希望你的文件中至少有三行，像这样：

```
line one
line two
line three
```

之后 f.readlines() 会返回一个如下的列表：

```
['line one','line two','line three']
```

在你的程序中，第二个你使用了 Python 列表但却可能被你忽略的地方，是下面这一行：

```
data = line.split(",")
```

行变量中存储了一串字符，这是从 CSV 文件中读取的一行完整的文本。所有的字符串变量都有内置的 `split()` 函数，它会读取这串字符并将其分隔成一个列表。方括号内的引号则告诉 split 函数如何来进行分隔。

想象你现在有一行以逗号形式分隔的包含三个词的变量，像这样：

```
line = "one,two,three"
data = line.split(",")
```

这样，`data` 变量就会包含一个有三个项目的列表，像下面这样：

```
['one','two','three']
```

挑战

使用 CSV 数据文件设计你自己的迷宫，包含许多曲折的通道、死胡同和环形道路来能让你的迷宫更加复杂、难以解决。让你的朋友来挑战一下，寻找迷宫的出口。你也可能在迷宫中或是在迷宫的出口放置一些随机的宝藏（例如钻石方块 `DIAMOND_BLOCK`），这样你的朋友可以用宝藏来证明他们已经穿越了整个迷宫，或者你可以建造一个巨大的能包含几乎整个 Minecraft 世界的迷宫！你会如何保存迷宫文件中记录宝藏的位置呢？

当我在设计这一章的程序时，我不得不稍微地调整程序，避免迷宫过于复杂。比如，当你把建造地板的那一行代码删除，看看会发生什么？尝试一下使用其他方块来建造迷宫的墙壁，如用仙人掌（`CATUS`）或者流水（`WATER_FLOWING`）来建造，看看发生了什么。我试过这个程序的许多版本，最终觉得这种方块建造的带地板层和墙壁最好！

挑战

在这个程序中有个需要你自己来探索的隐藏 bug（漏洞）。如果行末的最后一个数字是 0，出于某种原因，迷宫的建造器会仍然建造一些金方块的墙体。你觉得这个问题是什么？尝试自己解决这个 bug。提示：在 `tipChat.py` 程序中，你已经解决了一个类似的问题了。

在这个冒险中，你已经尝试过了使用 `readlines()` 和 `split()` 函数来处理 CSV 文件。Python 是一种非常大型而健全的编程语言，并有着非常多的内置功能。你可以使用 `import csv` 的方式来查阅内置的 CSV 阅读器模块，毕竟你已经学过这种有效的方式了。然而，我很喜欢以小的步骤来进行编程，因为它能帮助我理解程序是如何运作的，这正是为什么我在本书中使用 `readlines()` 和 `split()` 函数的原因。

建造一个 3D 方块打印机

建造新的迷宫会非常有趣，但你知道了原理之后，你能对数据文件做更多的事。为什么仅仅停留在建造只包含两种方块类型的单层建筑上呢？你现在把迷宫程序作为你自己的 3D 打印机的基础，以此用来复制树、房子——实际上，任何你在 Minecraft 世界里建造的东西，并按下一个按钮，将这些物品打印出来。其实，这是一个方块建造器，但我喜欢将其称为 3D 打印机，因为你能像看到计算机的打印机在纸上每次打印一行那样，或是像看到 3D 打印机每次建造一层结构一样，看到它每次建造一行方块。

你现在已经非常擅长编写程序了，每一个你编写的程序都比上一个长许多。就像在之前冒险中的那样，每当你为了一个功能而编写程序时，你就是以单步骤进行的方式编写程序。

编程的技巧在于，你先编写具有不同功能的函数，再将其整合在一起形成一个大一些的程序。专业的软件工程师们在开发程序时通常会使用这种技巧。这遵循着“大程序是小程序的集合”的原则。这也证明了程序的完整设计从最初就是正确的，而在每个起辅助作用的函数中的细节可在开发和测试过程中被逐渐地完成。我通常喜欢在手边保存一些程序样本，这样如果有人走过来问我：“嘿，你在写什么伟大的程序呢？”我会迅速地给他们看一些有效果的程序。

手动制造一个小型 3D 打印的测试物件

在你的迷宫程序中，数据是被存储在 CSV 文件中的，文件中的每一行对应了 Minecraft 世界里的一行方块。在那行中的每一列（由逗号分隔的）与那一行的方块一一对应。这是一个 2D 结构，因为它存储的只是一个长方形区域中 x 和 z 坐标的值。你制造的迷宫在这里只是 2D 迷宫，因为它们只有一层。它只是在 3DMinecraft 世界里使用 3D 方块建造出来的。

为了建造一个 3D 造型，你也需要设定 y 坐标的值。如果你仅仅把一个 3D 物体想成一个多层或多片的物体，你的 3D 程序就不会和迷宫程序有什么大的不同。你所有需要对 5×5 大小的方块做的，只是存储 5 层的信息。

接下来会遇到一个问题。你的 Python 程序不会知道它究竟需要获取多少行的数据，直到它执行完文件的全部内容。它可以设定一个定值，但它不能设计出一个可变大小的 CSV 文件，以符合实际物件的大小。

为了解决这个问题，你需要在文件的第一行添加一些元数据（metadata）来帮助描述这个物体。

元数据（Metadata）是一种关于数据的数据。如果你文件中的数据是建造的方块类型，那么元数据（关于数据的数据）会是你建造的物体究竟有多大，物体的名字以及设计者的名字。以相似的方式，计算机中的照片的元数据描述的，则是照相的日期、时间以及相机的名字，3D 打印机的元数据则会描述一些有关你的 3D 物体的其他的有用数据。

你现在将要建造一个小型 3D 洞穴的物件范例，这样你可以测试程序的 3D 打印能力 . 你可以从配套的 Wiley 网站下载如图 6-4 所示的范本物体，或者参照之前对迷宫数据所做的那样，简单地进行手动输入。

确保你在每一层物体之间输入了空白行。空白行是为了帮助你看到每一层开始的位置，但如果你遗漏了空白行，你编写的 Python 程序还是会认为它们在这里。

图 6-4　通过编写 CSV 数据文件，你建造了一个小型石窟

尽可能多地在这些步骤中，使用“编辑菜单”中的“复制”和“粘贴”功能，减少输入以及输入错误行号的可能性。

1. 单击 File⇨New File，创建一个新文件，保存为 `object1.csv`。

2. 输入第一个元数据行，用以描述物件的形状。你的测试物件将会是 5×5×5 的大小。第一个数字代表 *x* 值（宽度），第二个数字代表 *y* 值（高度），第三个数字代表 *z* 值（深度）：

```
5,5,5
```

3. 输入测试物件的第一层，确保你在第一行数据结束前输入了空行。你需要所有方块的位置在底层，所以每个数字需要设定成 1：

```
1,1,1,1,1
1,1,1,1,1
1,1,1,1,1
1,1,1,1,1
1,1,1,1,1
```

4. 你在建造的是带有一个开口的中空立方体，所以需要三层同种样式的方块。确保你在这些数字之前放入了一个空白行：

```
1,1,1,1,1
0,0,0,0,1
0,0,0,0,1
0,0,0,0,1
1,1,1,1,1
```

5. 使用复制和粘贴，在下面再放入两层，确保你在每一个方块部分之前至少放入一个空白行。

6. 最后，在物体顶端安上一个固定的顶层。为了实现这个目标，使用复制和粘贴来复制第三步的那些数据，再次确认在这层的第一行数据之前你已经输入了一个空格行。

保存你的文件，但不要运行——你不能运行它，因为它不是 Python 文件，而是被你即将要编写的程序使用的数据文件。

编写 3D 打印机

既然你已经写好了你的测试数据，现在到了编写用来在 Minecraft 世界中建造中空洞穴的 3D 打印机程序的环节了。这个程序和你的迷宫程序非常相似，但这是一个有 3 个嵌套循环的程序，它会对于 *x*，*y*，*z* 坐标分别进行循环。

在整本书中，这可能是你写过的最细致的程序之一。编写这个程序的时候请小心，一步一步地进行下去，最终你会被程序的结果惊艳到！如果你在某一个步骤遇到问题了，请仔细检查你的缩进是否正确。记住，所有在这本书中罗列的程序都能在配套的资源网站下载，访问 www.wiley.com/go/adventuresinminecraft 了解更多。

1. 单击 File⇨New FIle，创建一个新的文件，从“菜单”中单击 File⇨Save As 保存为 `print3D.py`。

2. 导入必要的模块：

```
import mcpi.minecraft as minecraft
import mcpi.block as block
```

3. 连接到 Minecraft 游戏中去：

```
mc = minecraft.Minecraft.create()
```

4. 创建一个用来作为你的数据文件的名字的常量。当你之后想要读取不同文件时，这个常量会使你方便许多：

```
FILENAME = "object1.csv"
```

5. 定义一个 `print3D()` 的函数。之后，在整个项目中，你会再次使用到这个函数，所以确保你对它进行了正确的命名：

```
def print3D(filename, originx, originy, originz):
```

6. 正如之前你对建造迷宫的程序所做的，在读取模式中打开文件并将所有的行读取至 Python 列表中：

```
  f = open(filename, "r")
  lines = f.readlines()
```

7. 列表中的第一项是在索引 0 这一项中。这是存储元数据的文件的第一行。通过使用 `split()` 函数，可以将一行的数据进行分割。然后，将这三个数字存到 `sizex`，`sizey`，`sizez` 这三个变量中。你必须在这里使用 `int()` 函数，因为当你从一个文件中读取行时，它们以字符串的形式被读取，但你必须将它们转变成数字的形式以便在之后对它们进行计算：

```
  coords = lines[0].split(",")
  sizex = int(coords[0])
  sizey = int(coords[1])
  sizez = int(coords[2])
```

8. 创建一个 `lineidx` 的变量来记录在 lines [ ] 列表中你正在处理的行的索引。因为你的程序将会依照行列表扫描并读取数据来建造 3D 物件中的不同层，你不能把它和循环控制变量一样处理：

```
  lineidx = 1
```

9. 第一个 `for` 循环扫描从文件中读取数据的每一个水平面。文件中最开始的几行是为了建造 *y* 坐标值最低的那一层。你可以放置一个 `postToChat` 在这里以便在 Minecraft 世界内部看到打印的过程：

```
  for y in range(sizey):
    mc.postToChat(str(y))
```

10. 通过给 lineidx 变量加 1 来跳过文件每一层中的空白行。记住那些空白行在文件中存在只是因为那会便于让人阅读，但你的程序需要跳过这些空白行。

```
    lineidx = lineidx + 1
```

11. 开始一个用来读取文件下一行的，并在逗号处进行分割的嵌套 `for` 循环。小心对待这里的缩进，for 必须要进行两次缩进（一次是为了函数，另一次是为了 y 循环的 `for`）并且在对 x 进行循环的 for 循环中的代码需要进行三次缩进。

```
    for x in range(sizex):
      line = lines[lineidx]
      lineidx = lineidx + 1
      data = line.split(",")
```

12. 现在你可以开始你的第三个 for 循环，用来扫描读取每一行中的每一个方块，并在 Minecraft 世界中建造它们。这个 for z 的循环是需要进行三次缩进的，循环体需要进行四次缩进，所以务必谨慎地对待缩进，否则你会得到一些形状非常古怪的物件！

```
    for z in range(sizez):
      blockid = int(data[z])
      mc.setBlock(originx+x, originy+y, originz+z, blockid)
```

13. 最终，编写主程序，在这里不再需要缩进了。这些行是为了读取玩家的位置并要求 print3D() 函数来在你的面前打印你的 3D 物件：

```
pos = mc.player.getTilePos()
print3D(FILENAME, pos.x+1, pos.y, pos.z+1)
```

保存并运行你的程序来看看会发生什么结果！在 Minecraft 世界里不同的地点，运行这个程序。每一次你运行这个程序，一个中空的石窟会在玩家的身边生成！图 6-5 向你展示了迅速建造多个石窟也是非常容易的事。

挑战

编写更多包含 3D 物件的 CSV 文件，改变在 print3D.py 程序中的 FILENAME 常量并运行它们，实现在你的 Minecraft 世界中加载这些物体。当你的物件逐渐变得复杂起来，考虑使用一些不同的方式便于你来设计，而不是直接输入数字来创造文件。你也可能找到在表格程序中绘制物件的其他方法。或者也许你可以买一大盒塑料方块，先在物理世界中开始设计，之后按照每一层的方块，将数字输入 CSV 文件中。

图 6-5　在 Minecraft 中 3D 打印了许多石窟

建造一个 3D 方块扫描器

你的 3D 打印机其实已经非常的强大了！你可以为你想建造的不同物体建造一个更大的 CSV 文件库，之后任何你需要的时刻，只需要使用不同的 `FILENAME` 常量来指向你想要的物体并运行你的 3D 打印程序 `print3D.py`，就可以在 Minecraft 世界的任何角落制造这个物体了。

然而，如果你还是指望手动输入那些 CSV 文件中的数字的话，在建造更大和复杂的物体却会遇到困难。Minecraft 自己是最好的建造复杂建筑的程序——所以，如果你能利用 Minecraft 来为你创造 CSV 数据文件的话，会发生什么呢？下面，你可以走到一棵树前抱住它，然后制作这棵树的副本，这样你能在 Minecraft 世界的任何角落将它复制出来。幸运的是，3D 扫描所做的事情和 3D 打印的逆序过程类似，所以可能并不是你想象得那么困难。

1. 单击 File⇨New File，创建一个新的文件，单击 File⇨Save As，将其命名为 `scan3D.py` 程序。
2. 导入必要的模块：

```
import mcpi.minecraft as minecraft
import mcpi.block as block
```

3. 连接到 Minecraft 游戏中：

```
mc = minecraft.Minecraft.create()
```

4. 为你想要读取的 CSV 文件以及读取的范围创建一些常量：

```
FILENAME = "tree.csv"
SIZEX = 5
SIZEY = 5
SIZEZ = 5
```

5. 定义一个 `scan3D()` 函数。你将会在后面的程序中使用到这个函数，所以确保函数名是正确的：

```
def scan3D(filename, originx, originy, originz):
```

6. 打开文件，但这次使用 `open()` 函数的 `"w"` 文件模式来写入：

```
  f = open(filename, "w")
```

7. 文件的第一行必须包含元数据，所以在第一行设定 *x*、*y* 和 *z* 值大小。在“深入代码”栏目查看更多来理解句末的 `"\n"` 所代表的意义以及作用。

```
  f.write(str(SIZEX) + "," + str(SIZEY) + "," + str(SIZEZ) ↩
 + "\n")
```

8. 这个扫描用的程序只是打印程序的逆序版本，再次使用三个嵌套循环。下面是一次性展示完整的程序部分，所以对你而言保证正确的缩进变得容易了许多。查看“深入代码”栏目，获得更多的关于逗号分隔行是如何被创造的相关解释：

```
  for y in range(SIZEY):
    f.write("\n")
    for x in range(SIZEX):
      line = ""
      for z in range(SIZEZ):
        blockid = mc.getBlock(originx+x, originy+y, ↩
        originz+z)
```

```
            if line != "":
              line = line + ","
            line = line + str(blockid)
        f.write(line + "\n")
    f.close()
```

9. 最后这个主程序是完全不需要缩进的。这只是用来读取玩家位置，并计算方块空间以便使玩家处于这个物件的中心，并且要求 `scan3D()` 函数来扫描整个 CSV 文件区域的。

```
pos = mc.player.getTilePos()
scan3D(FILENAME, pos.x-(SIZEX/2), pos.y, pos.z-(SIZEZ/2))
```

保存你的程序。在你运行程序之前，再次检查所有的缩进，确保它们是正确的。

现在走到一棵树前并努力抱住它（站得离树干越靠近越好），运行你的 `scan3D.py` 程序。看看发生了什么？

到底有没有事情发生呢？记住，这个程序只是将一个物件扫描进一个 CSV 文件，所以你必须查看 `tree.csv` 这个 CSV 文件来看看什么数字被存储进来了。通过单击 File⇨Open 来打开 `tree.csv` 文件，这是我使用自己的计算机扫描进去的树！因为扫描区域非常的小，你可能只是得到半棵树的扫描件，但你总能通过改变 `SIZE` 变量来放大扫描区域。

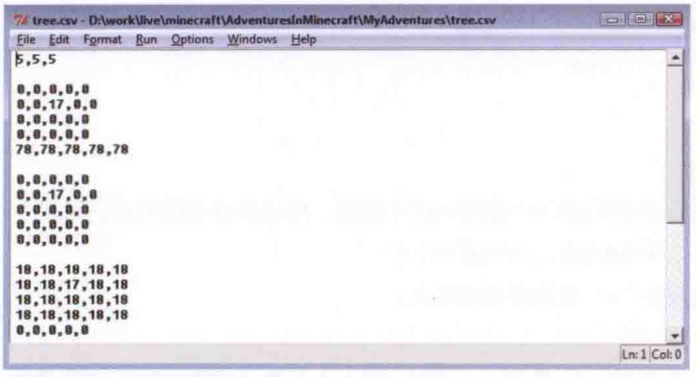


图 6-6　通过抱树及扫描树后得到的 CSV 文件的内容

记住，玩家面朝的方向不一定直接是扫描的方向。所以请确保你环顾了四周。3D 扫描器会从玩家所处的位置开始扫描，并逐渐增加坐标的值。所以，如果站在（0，0，0）位置，3D 扫描器会扫描直至（4，4，4）位置。回顾在冒险 2 中的关于坐标的图表来理解每个坐标的部分分别代表了什么。

IDLE 仅仅在它的打开和保存窗口里展示所有以 `.py` 后缀结尾的文件，但你可以通过选择 Files of Type⇨All Files 来选择你的 `.csv` 文件。

挑战

既然现在你已经拥有了一个 3D 扫描仪和一个 3D 打印机了，修改你的 3D 打印程序来让 FILENAME 常量使用 `tree.csv` 文件。在你的 Minecraft 世界里保持运行 `print3D.py` 程序。你应当可以在任何地方打印这些树。尝试在空中或者水里打印一些树，看看会发生什么。你能以多快的速度用此方式在这里建造一片森林呢？

深入代码

在 `scan3D.py` 程序里，你在 write() 函数里使用了一种特殊的字符，下面将会对这种字符进行一些解释：

```
f.write("\n")
```

这个 `\n` 字符（“反斜杠 n”）是一种特殊的不能被打印的字符，Python 允许你在引号中使用这种字符。它代表的是“换到下一行”。字母 `n` 代表了“新的一行”（newline）。这个反斜杠字符则是为了与这个字符和字母 n 有所区分。

在 `scan3D.py` 程序中，你设计了这个文件的存储形式，因此在物件数据中，每一层之间都有一行空白行，这样你能轻松地看懂并编辑这个 CSV 数据文件。`f.write("\n")` 只是帮助你完成这一行空白行的写入。

之后在 `scan3D.py` 程序中，有一些关于行变量的有趣代码。下面是一些重要的代码片段：

```
for x in range(SIZEX):
  line = ""
  for z in range(SIZEZ):
    blocked = mc.getBlock(origin+x,origin+y,
              originz+z)
    if line ! = "": # 该行非空
      line =line+","
    line = line+str(blockid)
```

加粗的部分是经常会使用到的为了建造逗号分隔值的编程模式的一部分。当 `x` 循环开始，行变量被设定为一串空白的字符。每次 `z` 值增加，首先会对该行进行是否为空的检查，如果该行非空，会增加一个逗号。最终，它会增加这个方块的方块代码。这个 `if` 语句能够有效保证逗号不会出现在第一行的开始即第一个数字之前，不然会引起 `print3D.py` 程序在之后读取时发生混乱。

建造一个复印机

你现在已经得到了用来建造你自己的 3D 复印机程序的基础构件了，这会让你的朋友非常羡慕。有了这个，你将可以在 Minecraft 世界中建造一个神奇的复印室，在那里建造任何你喜欢的物件。之后，你可以将这些物件保存在文件中，在复印室中读取文件来修改或者在 Minecraft 世界的任何地方随心所欲地将它们打印出来。最后，你可以离开 Minecraft 世界并让复印室彻底消失，不留下任何你施展过魔法的痕迹。

正如在早期的冒险中所做过的那样，你将要使用你现存的程序并组合他们成为一个更大的程序。由于这个程序有非常多的特征，你需要添加一个菜单系统来方便管理。

编写复印机的程序框架

第一件要做的事，是编写这个程序的框架，使各个部分都被包括起来。你会从编写一些没有实际作用而仅仅只是打印函数名的哑函数开始，之后再逐渐使用你已经完成的其他程序中的函数将细节添加到现在的哑函数中。你可能还记得吧，这正是你在冒险 4 中编写你的寻宝游戏时使用的编程方式。

1. 单击 File⇨New File 来创建一个新的文件，将名字保存为 `duplicator.py`。
2. 导入必要的模块：

```
import mcpi.minecraft as minecraft
import mcpi.block as block
import glob
import time
import random
```

3. 连接到 Minecraft 游戏中：

```
mc = minecraft.Minecraft.create()
```

4. 设置一些用来设定你的复印机的变量。不要将它们的值设定得过大，不然复印机需要花很多时间来扫描和打印物件。同时也为你的复印室设置一个默认的初始位置：

```
SIZEX = 10
SIZEY = 10
SIZEZ = 10
roomx = 1
roomy = 1
roomz = 1
```

5. 定义所有在这个程序中的哑函数。你很快会将这些细节补充完整：

```
def buildRoom(x, y, z):
  print("buildRoom")

def demolishRoom():
  print("demolishRoom")

def cleanRoom():
  print("cleanRoom")
```

```
def listFiles():
  print("listFiles")

def scan3D(filename, originx, originy, originz):
  print("scan3D")

def print3D(filename, originx, originy, originz):
  print("print3D")
```

6. 这些哑函数是特殊的，因为在函数的末尾会返回一个数值。暂时，你可以先产生一个随机选项然后返回（return），这样就能测试你的程序的早期版本，但不久之后你就可以在这里编写一些合适的“菜单”了。你现在只写了一个“哑菜单”，这样你可以先测试你的程序结构，之后你很快就能把合适的“菜单”内容补充在这里：

```
def menu():
  print("menu")
  time.sleep(1)
  return random.randint(1,7)
```

7. 编写程序用来展示菜单并使用必需的函数的主循环。在“深入代码”栏目查看更多信息关于如何使用 anotherGO 变量：

```
anotherGo = True
while anotherGo:
  choice = menu()

  if choice == 1:
    pos = mc.player.getTilePos()
    buildRoom(pos.x, pos.y, pos.z)

  elif choice == 2:
    listFiles()

  elif choice == 3:
    filename = raw_input("filename?")
    scan3D(filename, roomx+1, roomy+1, roomz+1)

  elif choice == 4:
    filename = raw_input("filename?")
    print3D(filename, roomx+1, roomy+1, roomz+1)

  elif choice == 5:
    scan3D("scantemp", roomx+1, roomy+1, roomz+1)
    pos = mc.player.getTilePos()
    print3D("scantemp", pos.x+1, pos.y, pos.z+1)

  elif choice == 6:
```

```
    cleanRoom()

  elif choice == 7:
    demolishRoom()

  elif choice == 8:
    anotherGo = False
```

保存你的测试程序并运行。看看发生了什么？图片 6-7 显示了在我的计算机上运行时发生的事。你可以看到这个程序正在告诉你它在做什么——它在从“菜单”中随机选择一个项目，然后使用这个处理“菜单”选项的函数。在那个时刻你的程序中的每个函数仅仅会打印它们的函数名，这就是你能够看到的全部内容。你可以在你自己的屏幕上看到和图片 6-7 中看到的不同顺序的内容，因为现在 `menu()` 函数是在每次使用时随机进行选择来决定的。

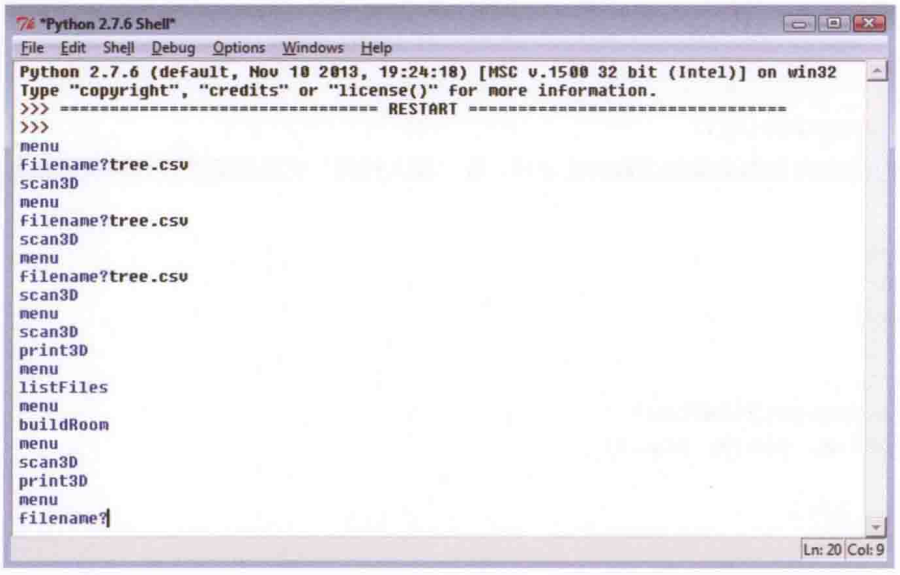


图 6-7　运行测试程序的结果

返回（return）是一种当从函数中回到第一次调用该函数的主程序时，将数值传回来的方法。

举个例子，当你需要一个随机的数字，你使用 `random.randit()` 函数。这个函数返回了一个数值，而你可以在一个变量中存储，如：`a =random.randint(1, 100)`。Python 中的返回仅仅只是允许你使用和刚才相同的技巧，将一个值从你自己的函数中传递回来。

`raw_input()` 函数读取一行从键盘中输入的字符。如果你在引号之间放置一个字符串，这个字符串会作为提示显示，所以 `name = raw_input("What is your name? ")` 会同时提出一个问题并得到答复。

当你使用 `raw_input()` 函数来从键盘读取一些东西，通常它会返回一串字符。如果你想要输入一个数字（比如，在你的主菜单系统中），那么你将需要使用 `int()` 函数来将它转化成一个数字。一些 Python 程序员会喜欢使用其他的方式在一行中解决这个问题，就像这样：`age = int(raw_ input("What is your age? "))`。

在这本书中你使用的是 Python 版本 2。如果你使用其他计算机安装的是 Python 版本 3，两者是存在一些差别的，其中之一是在 Python 版本 3 中，你需要使用 `input()` 代替在 Python 版本 2 中的 `raw_input()`。这本书使用 Python2，因为 Minecraft API 只能和 Python2 匹配，但也只需要进行一些小的改动就可以和 Python3 匹配。

深入代码

你在之前的冒险中已经使用过布尔型变量（boolean）了，但体会一下 anotherGo 变量是如何在 `duplicator.py` 程序中发挥作用还是非常有价值的事。下面是其中一些重要的部分。

```
anotherGo = True
while anotherGo:
  choice = menu()
  if choice == 8:
    anotherGo = False
```

这是对于至少执行一次的循环而言，非常常见的一种编程设计模式，它会询问你是否想要进行下一次的执行，如果不想则退出循环。但在 Python 中也有很多其他方式可以达到相同的目标，但使用布尔型变量来实现是相当好的方法，因为这能清晰地解释这个程序是如何运作的。

你不会在这本书中使用到，但 Python 中有一种被称作 `break` 的语句可以用来跳出循环。你也可以在网络上进行一些搜索，寻找如何使用 `break` 语句代替布尔型变量的方法，重写这段代码并实现相同的功能。

布尔型变量（boolean）是一种仅存放两个值的变量——`True` 或者 `False`。它是以 19 世纪初在形式逻辑领域做出杰出贡献的数学家 George Boole 命名的。访问 http://en.wikipedia.org/wiki/George_Boole 了解更多关于 Boole 的事。

显示菜单

对于一个有许多选项供你选择的程序而言，“菜单系统”是非常有用的功能。这个“菜单系统”会显示所有可用的选项，并且运行下去，等待你输入在正确范围内的一个数字。如果你输入的数字超出范围，那么它会再次显示“菜单”让你重新选择。

修改这些菜单函数，并像下面这样进行替换（请确保你的缩进是正确的）：

```
def menu():
  while True:
    print("DUPLICATOR MENU")
    print(" 1. BUILD the duplicator room")
    print(" 2. LIST files")
    print(" 3. SCAN from duplicator room to file")
    print(" 4. LOAD from file into duplicator room")
    print(" 5. PRINT from duplicator room to player.pos")
    print(" 6. CLEAN the duplicator room")
    print(" 7. DEMOLISH the duplicator room")
    print(" 8. QUIT")

    choice = int(raw_input("please choose: "))
    if choice < 1 or choice > 8:
      print("Sorry, please choose a number between 1 and 8")
    else:
      return choice
```

保存程序并且运行。这次你的程序与之前的“虚拟菜单”有什么不同呢？图 6-8 展示了真实的“菜单”看上去应该是什么样子的。

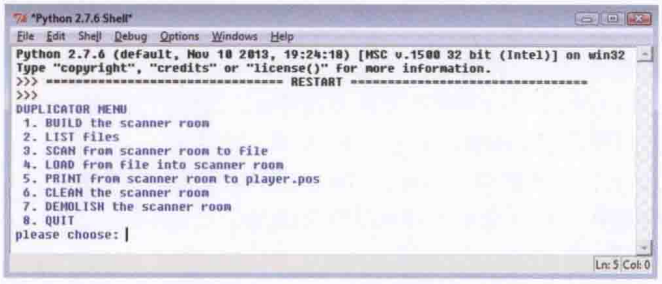


图 6-8　菜单系统

通过“菜单系统”编写程序的框架和哑函数，在之后一个一个填入所有函数的细节并再次测试，你正在以和现在的软件工程师在开发计算机软件时完全相同的方式进行工作。这是很好的练习，以小的步骤编写程序、修改并测试那个修改后的部分，直到你得到了一个完整的程序。这也意味着你永远不会花费超过两分钟的时间，对一个拍着你的肩膀要求你解释一下你的新程序的人进行解释。

建造复印室

复印室将会使用玻璃建造，而它缺失正前方的一面墙壁，这样你的玩家可以轻松地爬进复印室创建或者破坏方块。

使用下面的代码来替代 buildRoom() 函数。小心那些使用 setBlocks() 的长行，但注意我并非在这里使用了换行符，因为 Python 允许你将跨越多行的代码分成几部分（查看“深入代码”栏目，获得更详细的解释——为什么你有时可以进行分割而有时不行）：

```
def buildRoom(x, y, z):
  global roomx, roomy, roomz

  roomx = x
  roomy = y
  roomz = z

  mc.setBlocks(roomx, roomy, roomz,
    roomx+SIZEX+2, roomy+SIZEY+2, roomz+SIZEZ+2,
    block.GLASS.id)

  mc.setBlocks(roomx+1, roomy+1, roomz,
    roomx+SIZEX+1, roomy+SIZEY+1, roomz+SIZEZ+1,
    block.AIR.id)
```

保存你的程序并再次进行测试。测试在“菜单”中的选项 1，确保你可以建造复印室。在 Minecraft 世界中的另一个地点运行，再次选择选项 1 来查看发生了什么！图 6-9 所示为复印室被建造后的场景。

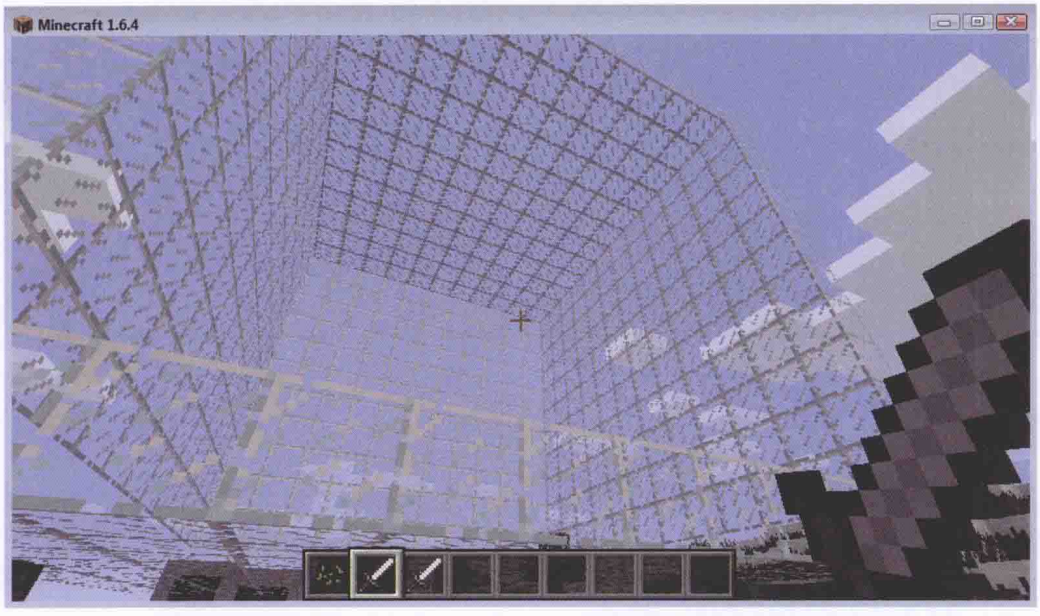


图 6-9　建造在你面前的复印室

深入代码

Python 使用左侧缩进（空格或者制表符），表明哪一些部分的程序语句属于同一组别。当你使用循环、if/else 语句或者函数时，你已经使用过缩进来组合属于一组的程序语句了。Python 使用缩进来组合语句的这种方式是非常独特的，一些其他的程序语言，如 C 或者 C++ 使用特殊的符号如 { and } 来组合这些程序语句。所以，你需要在 python 中非常小心地使用缩进，否则你的程序将不能正常运作！

在大多数时间里，Python 不会允许你将一行代码分割成多行，这就是为什么你经常看到换行箭头来告诉你这是一整行而不应该被分割。

然而，在 python 中你有两种方法来应对过长的一行代码。

首先，你可以使用一个行延长符号——如果这一行最后的一个字符是反斜杠（像这样：\）那么你可以在下一行中继续。

```
a = 1
if a == 1 or \
a == 2:
  print("yes")
```

第二种你有时可以使用的方法是将长的一行在明显可以被 Python 识别到这一行还没有结束的地方进行分割。比如，如果你将要在一个列表中设置初始值，或者使用一个函数，你可以在 Python 从这行代码的其他部分得知这后面必定还有一些后续的代码时，将这一行分割成多行。下面是两个例子，展示 Python 在何时会允许你对较长的一行代码分割成较短的几行，因为这个开放的大括号告诉了 Python 这一行代码必定在未能得到一个对应的大括号之前没有结束：

```
names = ["David",
"Steve",
"Joan",
"Joanne"
]

mc.setBlocks(x1,y1,z1,
x2,y2,z2,block.AIR.id)
```

摧毁复印室

当你多次运行你的复印程序后，你可能会在你的 Minecraft 世界中建造了许许多多的复印室，只有最后一间创建的复印室才是你正在使用的。在一些时间之后，你的世界里到处都是复印室以至于你可能都不清楚哪一间才是真正的。为了解决这个问题，你需要在你的程序中添加一个功能来摧毁复印室，那样你的 Minecraft 世界不至于被废弃的复印室弄得一团混乱。

摧毁复印室就像在冒险 3 中使用你的 clearSpace.py 程序一样，所有需要做的仅仅是知道房间的外坐标。因为房间比以 SIZEX、SIZEY、SIZEZ 定义的复印空间大一个方格，但做一点数学运算是非常简单的。

将 demolishRoom() 函数定义成如下所示。

```
def demolishRoom():
  mc.setBlocks(roomx, roomy, roomz,
    roomx+SIZEX+2, roomy+SIZEY+2, roomz+SIZEZ+2,
    block.AIR.id)
```

保存你的程序并再次运行。现在建造或者摧毁你的复印室是非常简单的工作了。只需要在“菜单”上使用选项 1 即可选择建造，或者使用选项 7 即可选择摧毁它。

在复印室中扫描物件

在之前你已经写过关于这个程序的一部分了——在你的 scan3D.py 程序中，所以你只需要使用那段程序并做一些小的修改即可。

1. 使用下面的代码代替 scan3D() 程序。这些代码和 scan3D.py 程序相比，几乎是相同的，但粗体所示的几行是为了在聊天栏展示扫描过程信息而被添加进来的。扫描一个大的空间可能需要花上较长的时间，所以能够得到一些关于你的程序运行过程的提示是非常好的。你可以使用“复制”和“粘贴”来获取之前的程序片段来节约一些时间：

```
def scan3D(filename, originx, originy, originz):
  f = open(filename, "w")
  f.write(str(SIZEX) + "," + str(SIZEY) + "," + str(SIZEZ)
  + "\n")

  for y in range(SIZEY):
    mc.postToChat("scan:" + str(y))
    f.write("\n")
    for x in range(SIZEX):
      line = ""
      for z in range(SIZEZ):
        blockid = mc.getBlock(originx+x, originy+y,
        originz+z)
        if line != "":
          line = line + ","
        line = line + str(blockid)
      f.write(line + "\n")
  f.close()
```

2. 保存你的程序并通过到复印室里建造一些东西来进行测试，之后从“菜单”中选择选项 3。打开它创建的文件，检查这个物件的扫描结果是否正确。图片 6-10 展示了一部分扫描后生成的文件。注意在这里会有许多的 0，这是因为所有的空气方块也被扫描进来了。

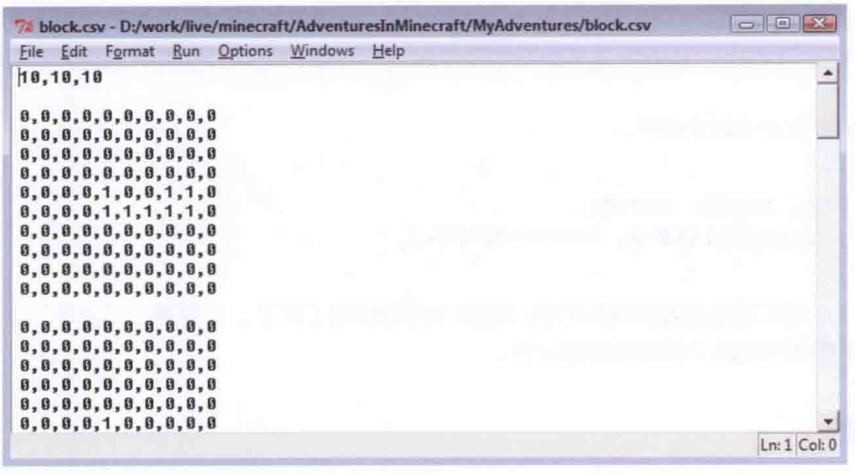


图 6-10　扫描文件的内容

清理复印室

期待着一次彻底的清理？有时拥有一个全新的开始，清理房间是非常有用的。你可以通过摧毁复印室并重新建造实现这个实现清空，但添加一个清空功能也是非常容易的，因为与 `demolishRoom()` 函数相比，它们仅在坐标位置上略有差异。

1. 将 `cleanRoom()` 函数修改成下面所示的代码。注意初始坐标值是如何设定成比房间的边界大的，并且结束的坐标值不是和在 `demolishRoom()` 函数中的一致的。在这个方式下，它清空了房间内所有的内部空间而不摧毁它的墙壁：

```
def cleanRoom():
  mc.setBlocks(roomx+1, roomy+1, roomz+1,
    roomx+SIZEX+1, roomy+SIZEY+1, roomz+SIZEZ+1,
    block.AIR.id)
```

2. 保存你的程序并且再次运行。在房间内创建一些东西并选择选项 6 来看看是否能够进行快速清除。

在复印室中打印

在房间中打印（复印）物件是非常容易的，因为你已经写好了能做到这件事情的 `print3D.py` 程序，在这里是相同的。这里再次重复这段代码只是为了让你在同一个地方看到这些代码：

```
def print3D(filename, originx, originy, originz):
  f = open(filename, "r")
  lines = f.readlines()

  coords = lines[0].split(",")
  sizex = int(coords[0])
```

```
sizey = int(coords[1])
sizez = int(coords[2])

lineidx = 1

for y in range(sizey):
  mc.postToChat("print:" + str(y))
  lineidx = lineidx + 1

  for x in range(sizex):
    line = lines[lineidx]
    lineidx = lineidx + 1
    data = line.split(",")

    for z in range(sizez):
      blockid = int(data[z])
      mc.setBlock(originx+x, originy+y, originz+z,
      blockid)
```

保存程序并再次运行。看看你是否能在房间中建造一些东西，在 Minecraft 世界的某处运行并选择“菜单”选项 5。房间中的物件会被扫描并打印，放置到玩家的位置。你也可以在 Minecraft 世界的任何位置运行和无限次地复印你的物件。尝试在空中或者水里复印物件来看看会发生什么！图 6-11 展示了运行这个程序的结果。

图 6-11　在房间中的一个石制品并在玩家身边打印出来

整理文件目录

你最后的工作是写个有用的小函数来将所有的 CSV 文件添加到你的文件系统列表中。你可以只是使

用文件管理器，但在你的程序中添加这个功能会更好，这样你能够在一个地方完成所有的工作：

1. 修改 `listFiles()` 函数来使用 `glob.glob()` 函数来读取并打印所有文件的列表。查看“深入代码”栏目来获得关于运行原理的更多解释。

```
def listFiles():
  print("\nFILES:")
  files = glob.glob("*.csv")
  for filename in files:
    print(filename)
  print("\n")
```

2. 保存程序并再次运行。扫描一些物件生成 CSV 文件，之后从“菜单”中选择选项 2 来确保它们是在列表中的。图 6-12 展示了在我的计算机上运行这个程序时，我创建的文件。

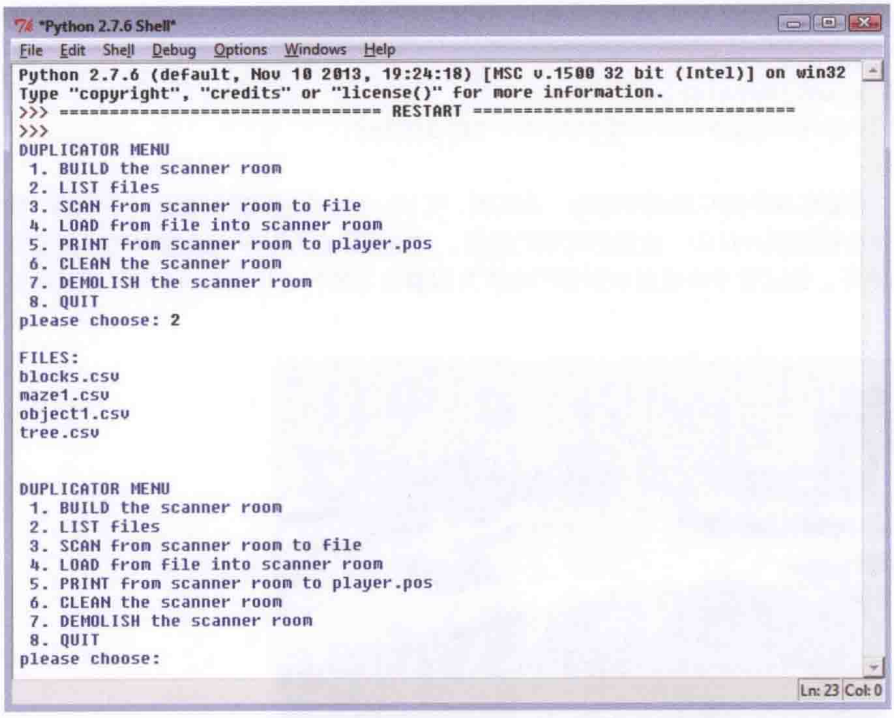


图 6-12　查看你创建的 CSV 文件列表

深入代码

`glob.glob()`——多么有趣的一个函数名称。你需要输入两次 glob，所以这显得很有趣！

`glob` 这个名称在这里使用了两次，因为第一次是代表这个在程序首段中首次被引用的模块（glob 模块）。第二次使用 `glob` 这个名称则是代表 `glob` 模块中的 `glob` 函数。

但 `glob` 函数有什么作用，为什么被命名为 `glob` 呢？

这只是单词“全局命令”（global command）的简写形式，追溯到早期 Unix 操作系统中的设计。访问维基百科 http://en.wikipedia.org/wiki/Glob_(programming) 了解 glob 名称背后的历史。

glob 所做的是收集符合模式（或者通配符）的文件。当你使用 `glob.glob("*.csv")` Python 会在计算机文件系统中的当前目录里进行搜索，并产生一个包含所有以 `".csv"` 的文件列表。

所以，如果你在 `MyAdventures` 文件夹下有 one.csv、two.csv、three.csv 文件的话，使用 `glob.glob("*.csv")` 将会返回像下面这样的一个 Python 列表：

```
['one.csv','two.csv','three.csv']
```

记住，for 循环会持续进行直至所有列表中的项目都被使用过，这也是为什么下面的这几行代码会在实际中很有用：

```
for name in glob.glob("*.csv")
  print(name)
```

通配符（wildcard）是一种用来选择大量相似名称或者词汇的特殊字符。它就像在一堆纸牌中里的鬼牌或者是万能牌一样，它能代表任何你希望它代表的。

在 Python 中，通配符通常被用来在文件系统中选择在文件名称上十分相似的文件，比如“all CSV files”。这能通过使用 `glob.glob("*.csv")` 实现，这个 `*` 字符标记了通配符的位置，并且 `glob.glob()` 会匹配任何在文件系统中的以 `.csv` 结尾的文件。

挑战

如果在你的 `duplicator.py` 程序中，你输入了一个不存在的文件名，程序会因错误而停止并把你的复印室留在 Minecraft 世界中。请你在互联网中搜索检测一个文件存在与否的方法来防止程序出错，加强程序的健壮性。

当 Martin 在为我测试 `duplicator.py` 程序时，他发现了一个 bug（漏洞）——实际房间比预设的还要大一些。这意味着如果你建造了全尺寸的物件，当扫描到文件时，一边的方块会卡住，并且当你在打印时，这边的方块会消失。如果这边是房子或者是其他的什么会造成极大的不便！看看你是否都能确定这种 bug 的原因并自己修复！

快速参考表	
从文件中读取	**在文件中写入**

`f = open("data.txt","r")`
`tips = f.readlines()`
`f.close()`
`for t in tips:`
`  print(t)`
	`f=open("scores.txt","w")`
	`f.write("Victoria:26000\n")`
	`f.write("Amanda:10000\n")`
	`f.write("Ria:32768\n")`
	`f.close()`

获得一个匹配文件名的列表	**移除字符串中不需要的空格**

`import glob`
`names = glob.glob("*.csv")`
`for n in names:`
`  print(n)`
	`a = "\n\n  hello  \n\n"`
	`a = a.strip()`
	`print(a)`

关于数据文件的更多冒险

在这次冒险中，你已经学习了如何读取和写入数据文件。这会极大地便于保存和恢复 Minecraft 世界中的部分，甚至带给你大量从别处获得的真实数据，比如网络上。你建造自己的有着弯曲通道和各种死胡同的 3D 迷宫，完成了建造一个具备完整功能的 3D 扫描和打印机，并且完成了一个完整的“菜单系统”。这种编写“菜单系统”的技巧，会对编写其他程序也颇有裨益的！

- 现在互联网上会有一些“实时数据”。我在写书时找到的数据是诺丁汉市政停车场的数据，可在 http://data.nottinghamtravelwise.org.uk/parking.xml 查看。这些数据记录了所有进出停车场的车辆，每 5 分钟会更新一次。我已经写过一个包含这个数据并打印所有有用信息的 Python 程序，可以在我的 github 页面上查看: ttps://github.com/whaleygeek/ pyparsers。你是否能使用这些数据，在你的 Minecraft 世界里编写一个建造停车场和汽车的 Minecraft 游戏。你的游戏会是一个限时寻找车位的停车挑战！

- 所有你希望了解关于 3D 迷宫的资料可以在这个神奇的页面上找到: http://www.astrolog.org/labymth/algrithm.htm。这个页面中包含了所有以普通文本文件的方式存储的多层 3D 迷宫的例子。你可以试着在这个网站中查找多层迷宫的文件，使用你的 3D 打印机和你学到的技巧，将你的迷宫程序修改成一个更庞大的多层迷宫。你还可能使用这个网站上的一些建议来进行实验，编写一个为你自动生成迷宫的 Python 程序。

解锁成就：破除物理法则的约束，在 Minecraft 世界里你通过按下一个按钮，就能魔幻般地让大型物件出现或是消失。

在下一次冒险中……

在冒险 7 中，你会学习到如何通过编写 Python 程序来建造复杂一些的 2D 和 3D 物件。你会学习如何使用 Minecraft 时钟来保存时间，建造四边形和其他多边形而仅仅使用几行的 Python 代码——并且，准备好一次前往金字塔的激动人心的远征之旅！

冒险 7
建造 2D 和 3D 结构

在 Minecraft 中编程，最棒的事情之一，就是你可以切实地在一个虚拟的 3D 世界里赋予自己的创造发明以生命，同时你也可以在 2D 屏幕上注视着它们。你可以绕着它们走动，进入它们之中或者让它们变得更大。如果你希望的话，甚至可以让它们爆炸！基于最初为了在 2D 屏幕上显示 3D 物体而创新出的想法，你可以在 Minecraft 中编程创造 3D 物体，让平凡不再平凡。

本次冒险中，你将学到如何使用 `minecraftstuff` 模块（module）来创造 2D 直线和形状。这些图形与一些数学知识结合就可以让你通过编程的方式实现一台巨大的时钟，大到当它的指针旋转时你甚至可以站在上面（见图 7-1）。一旦你掌握了 2D 形状的创建，之后你就能学到如何将那些 2D 图形结合起来，在短短几秒钟的时间里创建大型 3D 结构。

图 7-1　一台巨大的 Minecraft 时钟

minecraftstuff 模块

minecraftstuff 是 Minecraft API 的一个扩展模块（extension module），该模块是特意为本书而设计的，包含了所有你所需要绘制的形状和所要控制 3D 对象的代码。其有一系列称为 MinecraftDrawing 的函数，它们可以让你创建直线、圆、球体以及其他形状。正因为这些复杂的代码位于一个独立的模块（module）内，所以编写代码将会变得更加简单，代码也更容易理解。模块是将程序分成小块的一种方式。程序的规模过于庞大时，代码将不便于阅读和理解，查找问题所在也会更久。

minecraftstuff 模块包含在本书的初学者工具包中，访问书后附带的网址 www.wiley.com/go/adventuresinminecraft 了解更多。与 minecraft 以及 block 模块一样，minecraftstuff 模块可以用同样的方式导入 Minecraft 的程序中。

创建直线、圆和球体

当你将一些小而简单的东西结合起来时，不论多大多复杂，你都能创造出你想要的任何东西。本次冒险中，你将使用许多直线、圆、球体以及其他形状的组合来制作一个 Minecraft 中真正的大结构。本次冒险的这一阶段，你将创建一个新程序，然后导入你所需要的模块并启动 minecraft 以及 minecraftdrawing 模块。之后你将使用模块中的函数来绘制直线、圆和球体。

启动 Minecraft 和 IDLE 编辑器。如果你在 PC 或 Mac 上操作，那么请同时开启 Bukkit 服务端。现在你应该已经对一切的启动都有了一定的实践，但如果你还需要提示的话请重新回顾冒险 1。

1. 我们先打开 IDLE 编辑器，然后新建一个文件。文件以 LinesCirclesAndSpheres.py 为名保存到 MyAdventures 文件夹中。
2. 接下来，输入下列代码导入 minecraft、block、minecraftstuff 和 time 模块：

```
import mcpi.minecraft as minecraft
import mcpi.block as block
import mcpi.minecraftstuff as minecraftstuff
import time
```

3. 创建与 Minecraft 的连接：

```
mc = minecraft.Minecraft.create()
```

4. 使用 minecraftstuff 模块，输入下列代码来创造一个 MinecraftDrawing 对象：

```
mcdrawing = minecraftstuff.MinecraftDrawing(mc)
```

深入代码

MinecraftDrawing 是 minecraftstuff 模块中的一个类。类是一种被称为“面向对象”（Object Oriented）的特殊编程方法的一部分。这种特殊的编程方式，将相似的函数和数据聚集起来，对于我们这里的情况就是 Minecraft 的绘图函数和数据。所以我们可以更简单地理解并使用它们。当你使用一个类，然后命名（例如 mcdrawing），那么你就创建了一个对象。这就是所谓的实例化（instantiation）。一个对象类似一个变量，但又不仅仅储存值（或者数据），还带有函数！

面向对象的编程（Object Oriented Programming，OOP）是一门复杂的学科。虽然已经有很多书籍介绍了这门学科的定义以及有效利用的方法，但这里有一篇非常有用的介绍，关于如何在 Python 中使用类。访问 en.wikibooks.org/wiki/A_Beginner's_Python_Tutorial/Classes 了解更多。

你可以用 IDLE 编辑器打开 `MyAdventures/mcpi` 文件夹内的 `minecraftstff.py` 文件来查看 `minecraftstuff` 中的代码。

绘制直线

`MinecraftDrawing` 对象有一个叫作 `drawLine()` 的函数，当其被调用且传递两个位置的（x，y，z）以及方块种类作为参数时，它就会在两位置之间创建一条方块构成的直线。这就像你在冒险 3 中已经学过的 `setBlocks()` 函数一样。

当一个函数需要信息来运行时，比如 `setBlock()` 需要（ ，y，z）和方块种类，那些值就是所谓的参数（parameters）。当一个程序使用那个函数，我们就称之为调用（call）了函数并传递（pass）了参数。

下面的函数创建了一条方块构成的直线，如图 7-2 所示：

```
drawLine(x1, y1, z1, x2, y2, z2, blockType, blockData)
```

图 7-2 `drawLine()` 函数在两组（x，y，z）点之间创造了一条方块构成的直线

现在更新你的程序，这样它就能使用 drawLine() 函数在 Minecraft 里创建三条直线——从玩家的位置开始，一条笔直往上、一条水平、一条斜向。在 LinesCirclesAndSpheres.py 程序的最后加上如下代码：

1. 首先，你需要输入下列代码找到玩家的位置：

```
pos = mc.player.getTilePos()
```

2. 为了绘制一条从玩家位置开始垂直向上的 20 个方块所构成的直线，请输入：

```
mcdrawing.drawLine(pos.x, pos.y, pos.z,
                   pos.x, pos.y + 20, pos.z,
                   block.WOOL.id, 1)
```

3. 现在，输入下列代码，就可以绘制一条 20 个方块的水平线笔直穿过之前那条直线：

```
mcdrawing.drawLine(pos.x, pos.y, pos.z,
                   pos.x + 20, pos.y, pos.z,
                   block.WOOL.id, 2)
```

4. 接下来，绘制一条 20 个方块的斜线向上穿过之前两条直线：

```
mcdrawing.drawLine(pos.x, pos.y, pos.z,
                   pos.x + 20, pos.y + 20, pos.z,
                   block.WOOL.id, 3)
```

5. 最后，由于我们也会参与这个程序的运行过程，所以加入一些延迟的话你就能看见到底发生了什么：

```
time.sleep(5)
```

6. 现在保存文件然后运行程序。你应该已经创建了三条方块构成的直线，每一条都使用了不同颜色的羊毛：一条从玩家的位置垂直伸展，第二条水平伸展，第三条在它们之间斜向伸展。

深入代码

DrawLine() 函数使用了 Bresenham 画线算法（Bresenham line algorithm）来创建直线。访问 http://en.wikipedia.org/wiki/Bresenham's_line_algorithm 了解更多关于该算法的知识。

绘制圆

你不必拘泥于直线——你一样可以使用 MinecraftDrawing 来创建圆。使用 drawCircle() 函数并传递圆心和半径以及方块种类即可。你可以使用下面的函数来创造一个圆：

```
drawCircle(x, y, z, radius, blockType, blockData)
```

为了造出图 7-3 中显示的圆，在 LinesCirclesAndSpheres.py 程序最后加上下列代码即可：

1. 首先，输入下列代码获取玩家当前位置：

```
pos = mc.player.getTilePos()
```

2. 现在，输入下列代码从玩家上方 20 个方块处开始，以 20 个方块为半径绘制一个圆：

```
mcdrawing.drawCircle(pos.x, pos.y + 20, pos.z, 20,
                     block.WOOL.id, 4)
```

3. 最后，加上时间延迟。这样你就能看见程序中到底发生了什么，而且你也有机会移动玩家的位置：

```
time.sleep(5)
```

4. 保存文件然后运行程序。

图 7-3 `drawCircle` 使用半径和中心点位置（*x*，*y*，*z*）绘制了一个圆

直线会先绘制出来，然后在圆直接绘制在直线上之前，你有 5 秒时间可以移动玩家。

深入代码

`drawCircle()` 函数使用了“中点圆算法”（mid-point circle algorithm）用方块来创建圆。这是“Bresenham”直线演算法（Bresenham line algorithm）的修改版。想了解更多可以参阅：http://en.wikipedia.org/wiki/Midpoint_circle_algorithm。

绘制球体

`drawSphere()` 函数类似 `drawCircle()` 函数，需要有中心点、半径和方块种类才能起效。你可以像这样用这个函数创建一个球面：

```
drawSphere(x, y, z, radius, blockType, blockData)
```

为了创建图 7-4 中展示的球面，我们可以在 LineCirclesAndSpheres.py 程序的最后加入如下代码：

1. 输入以下代码来获取玩家当前位置：

```
pos = mc.player.getTilePos()
```

2. 为了从玩家上方 20 个方块处以 15 个方块为半径开始绘制球面，请输入：

```
mcdrawing.drawSphere(pos.x, pos.y + 20, pos.z, 15,
                     block.WOOL.id, 5)
```

3. 保存文件然后运行程序。

图 7-4 drawSphere 使用了半径和中心点位置（x，y，z）创造了一个球面

直线和圆将会重新绘制一遍，在球面绘制之前你会有 5 秒的时间以便移动玩家。

你可以从本书的配套资源网站 www.wiley.com/go/adventuresinminecraft 下载完整的绘制直线、圆和球体的代码。

球体适用于在 Minecraft 中创造爆炸。通过绘制空气方块（AIR）的方式构成球体，你可以移除球心周围的所有方块，在世界上创造出一个“洞”。

挑战

现在你已经看到，创建基本图形是多么简单。尝试借助代码，利用直线、圆和球体来创造你自己的 3D 艺术作品吧。

创建一台 Minecraft 时钟

现在你已经知晓如何创造圆和直线了，所以你也许已经明白如何用代码实现图 7-1 中的钟表。表盘是一个巨大的圆，每根指针是直线。那么下面就是难点——解决直线放置的位置以及如何使它们走动起来。

> 访问配套资源网站 www.wiley.com/go/adventuresinminecraft 即可观看教程，学习如何创建 Minecraft 时钟。
>
> 
>

在这部分冒险旅途中，你将会运用三角学（trigonometry）知识，把指针的角度转化为表盘上 x 和 y 的坐标值来解决如何确定指针端点的问题（见图 7-5）。首先用方块绘制直线，然后用空气方块绘制直线来移除它们，之后在新位置绘制直线。通过这种方式使指针看似在走动。

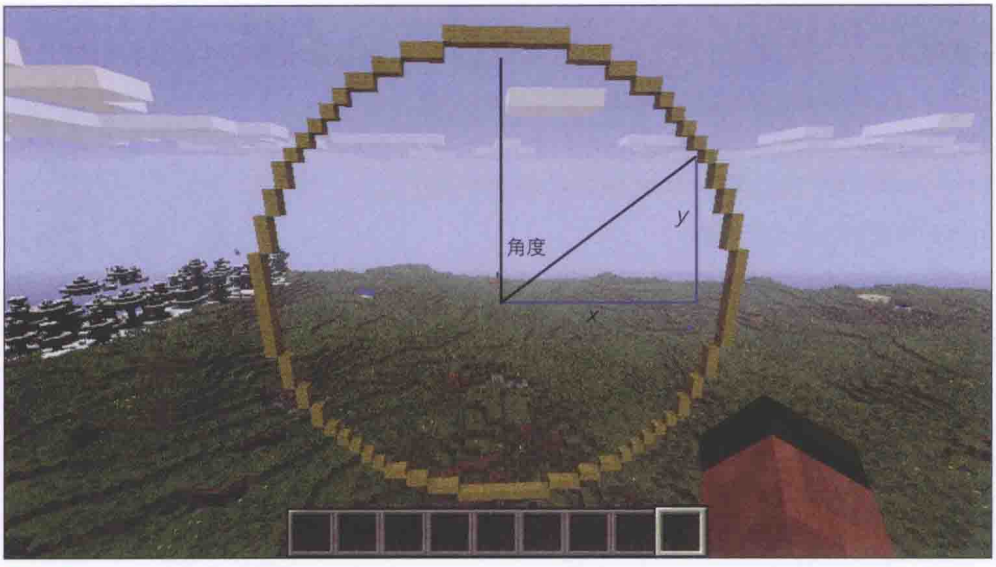


图 7-5　找寻如何确定表盘上指针

> 三角学（trigonometry）是数学的一个分支，其主要解决三角形角度和边长之间的关系。访问 en.wikipedia.org/wiki/Trigonometry 了解更多。
>
> 
>

为了创建你的时钟，请遵循以下步骤：

1. 打开 IDLE 编辑器，为 Minecraft 时钟创建一个新程序，新建一个文件。文件以 `MinecraftClock.`

py 为名保存在 MyAdventures 文件夹内。

2. 输入下列代码导入 minecraft、block、minecraftstuff、time、datetime 以及 math 模块：

```
import mcpi.minecraft as minecraft
import mcpi.block as block
import mcpi.minecraftstuff as minecraftstuff
import time
import datetime
import math
```

3. 创建一个函数 findPointOnCircle()。当你给这个函数传递圆心和时钟指针的角度时，函数会返回时钟指针的位置，就像图 7-5 显示的那样。

```
def findPointOnCircle(cx, cy, radius, angle):
    x = cx + math.sin(math.radians(angle)) * radius
    y = cy + math.cos(math.radians(angle)) * radius
    x = int(round(x, 0))
    y = int(round(y, 0))
    return(x,y)
```

4. 连接 Minecraft 模块并创造对象 MinecraftDrawing：

```
mc = minecraft.Minecraft.create()
mcdrawing = minecraftstuff.MinecraftDrawing(mc)
```

5. 输入下列代码来获取玩家当前位置：

```
pos = mc.player.getTilePos()
```

6. 现在你将为时钟中心（即玩家所在位置上方 25 个方块处）创造一些变量，有表盘半径和指针长度：

```
clockMiddle = pos
clockMiddle.y = clockMiddle.y + 25

CLOCK_RADIUS = 20
HOUR_HAND_LENGTH = 10
MIN_HAND_LENGTH = 18
SEC_HAND_LENGTH = 20
```

7. 接下来，输入下列代码，用 drawCircle() 绘制表盘：

```
mcdrawing.drawCircle(clockMiddle.x, clockMiddle.y,
                     clockMiddle.z,
                     CLOCK_RADIUS, block.DIAMOND_BLOCK.id)
```

8. 启动一个无限循环。这个节点以后的所有代码将被放置于循环内。

```
while True:
```

9. 接下来你需要借助函数 datetime.datetime.now() 看看计算机现在的时间。然后将时间分

成小时、分钟和秒。因为你的时钟是 12 小时制而不是 24 小时制，因此你需要特别注意下午的时间，真正的时间减去 12 小时才是显示的时间（若是 14:00，那么显示 2:00）。输入下面的代码来实现这一切：

```
    timeNow = datetime.datetime.now()
    hours = timeNow.hour
    if hours >= 12:
        hours = timeNow.hour - 12
    minutes = timeNow.minute
    seconds = timeNow.second
```

10. 现在绘制时针。时针的角度将会是 360° 除以 12 小时乘以现在的小时。用 `findPointOnCircle()` 找到位置 *x* 和 *y* 作为时针末端，然后用 `drawLine()` 绘制时针。输入内容如下：

```
    hourHandAngle = (360 / 12) * hours
    hourHandX, hourHandY = findPointOnCircle(
        clockMiddle.x, clockMiddle.y,
        HOUR_HAND_LENGTH, hourHandAngle)
    mcdrawing.drawLine(
        clockMiddle.x, clockMiddle.y, clockMiddle.z,
        hourHandX, hourHandY, clockMiddle.z,
        block.DIRT.id)
```

11. 再下一步，输入下面的代码绘制分针。分针在时针的（z-1）处，即时针的后一个方块的位置。

```
    minHandAngle = (360 / 60) * minutes
    minHandX, minHandY = findPointOnCircle(
        clockMiddle.x, clockMiddle.y,
        MIN_HAND_LENGTH, minHandAngle)
    mcdrawing.drawLine(
        clockMiddle.x, clockMiddle.y, clockMiddle.z-1,
        minHandX, minHandY, clockMiddle.z-1,
        block.WOOD_PLANKS.id)
```

12. 现在加上秒针。秒针在时针的（z+1）处，即时针的前一个方块的位置。

```
    secHandAngle = (360 / 60) * seconds
    secHandX, secHandY = findPointOnCircle(
        clockMiddle.x, clockMiddle.y,
        SEC_HAND_LENGTH, secHandAngle)
    mcdrawing.drawLine(
        clockMiddle.x, clockMiddle.y, clockMiddle.z+1,
        secHandX, secHandY, clockMiddle.z+1,
        block.STONE.id)
```

13. 等待 1 秒：

```
time.sleep(1)
```

14. 现在你需要再次绘制指针来清除显示的时间，只有这时候你才用空气方块进行绘制。请输入：

```
    mcdrawing.drawLine(
        clockMiddle.x, clockMiddle.y, clockMiddle.z,
        hourHandX, hourHandY, clockMiddle.z,
        block.AIR.id)
    mcdrawing.drawLine(
        clockMiddle.x, clockMiddle.y, clockMiddle.z-1,
        minHandX, minHandY, clockMiddle.z-1,
        block.AIR.id)
    mcdrawing.drawLine(
        clockMiddle.x, clockMiddle.y, clockMiddle.z+1,
        secHandX, secHandY, clockMiddle.z+1,
        block.AIR.id)
```

15. 保存文件然后运行程序，看看你努力的结果。你应该会看到 Minecraft 时钟直接出现在玩家上方，而玩家可以向上或者去时钟的一侧看时间。确保玩家在时钟的正面，不然时间将会倒流！

你可以从配套资源网站 www.wiley.com/go/adventuresinminecraft 下载 Minecraft 时钟的完整代码。但我强烈推荐你阅读每一步操作后自己输入代码。那样你将会学到更多！

深入代码

finePointOnCircle() 函数根据你指定的圆心（cx，cy）、半径和角度计算出圆上的一点（*x*，*y*）。

1. 函数定义时包括了 cx、cy、radius 和 angle 作为输入的参数：

```
def findPointOnCircle(cx,cy,radius,angle):
```

2. 圆上的一点（*x*，*y*）经过 math 函数的 sin() 和 cos() 进行计算，然后乘以半径：

```
x = cx + math.sin(math.radians(angle))* radius
y = cy + math.cos(math.radians(angle))* radius
```

math.sin() 和 math.cos() 函数需要使用弧度（radians）来传递参数。弧度是度量角度的另一种方法（不同于常用的 0 ~ 360°），因此我们使用 math.radians() 函数来把角度转化为弧度。

3. *x*、*y* 值计算结果带有小数，但是函数需要整数。因此我们使用 round() 和 int() 函数来取最接近的整数，从而把小数转成整数：

```
x = int(round(x,0))
y = int(round(y,0))
```

4. 然后 *x* 和 *y* 值返回给程序：

```
return(x,y)
```

函数同时返回 *x* 和 *y* 的值，因此调用这个函数的程序必须在调用时提供两个变量。

创建时钟指针你需要一个三步走的过程：

1. 根据 360°除以 12 或 60（取决于时针、分针还是秒针）再乘以现在的小时、分钟或秒，求出指针的角度：

```
hourHandAngle = (360 / 12) * hours
```

2. 用 finePointOnCircle() 函数找到指针末端的坐标（*x*，*y*）：

```
hourHandX,hourHandY = findPointOnCircle(
        clockMiddle.x,clockMiddle.y,
        HOUR_HAND_LENGTH,hourHandAngle)
```

因为这个函数返回两个值，所以调用它时需要提供两个变量（hourHandX，hourHandY）。

3. 从表盘中心到指针末端绘制一条直线：

```
mcdrawing.drawLine(
        clockMiddle.x,clockMiddle.y,clockMiddle.z,
        hourHandX,hourHandY,clockMiddle.z,
        block.DIRT.id)
```

挑战

真实钟表的时针在一小时里的每一分钟都在运动。例如，如果时间是 11:30，时针将会在 11 和 12 中间。而你刚刚创建的 Minecraft 时钟的代码并不是这样运行的——时针在 11:59:59 跳至 12:00:00 之前都将一直指向 11。

思考一下，如果你能够让你的 Minecraft 时钟像现实中的时钟一样运行，你该如何改变计算时针角度的方式？

绘制多边形

多边形是任意的由一些互相连接在一起的边组合起来的 2D 图形。它能够拥有许多条边，至少 3 条（三角形）。正如你从图 7-6 中看到的一样，你能在 Minecraft 中创造出任意多有趣的多边形。

虽然它们都是 2D 图形，但是你会发现多边形对于 3D 图形的制作非常有用，因为实际上你可以把许多多边形联系起来创造任意的 3D 物体。当多边形被一起用来组成 3D 物体时，它们就被称之为表面（Faces）。你可以这样创造一些复杂的结构。见图 7-7，这张图展现了曼哈顿的天际线，而这正是由许多多边形创造出来的（访问 www.stuffaboutcode.com/2013/04/minecraft-pi-edition-manhattanstroll.html 了解更多）。

表面（Faces）即一个更大物体的单个平坦表面。比如，立方体的一个侧面或者鼓的顶部。

图 7-6　Minecraft 中多边形的例子

图 7-7　Minecraft 中纽约曼哈顿的天际线

你可以使用 `MinecraftDrawing` 中的 `drawFace()` 函数来创造多边形（或者表面）。这个函数等待着点（*x*，*y*，*z*）列表的传递，当这些点逐次连接起来以后将会创造出一个完整的多边形。传递一个 `True` 或者 `False` 将会决定表面是否需要填充，而最后一个参数决定了构成表面的方块种类（见图 7-8）：

```
drawFace(points, filled, blockType, blockData)
```

创建一个新程序来用 `drawFace()` 函数进行实验，然后创造图 7-8 所示的三角形。

1. 首先，打开 IDLE 编辑器并创建一个新文件。将其以 `Polygon.py` 为名保存在 `MyAdventures` 文件夹内。

2. 输入下列代码来导入 `minecraft`、`block` 和 `minecraftstuff` 模块：

```
import mcpi.minecraft as minecraft
import mcpi.block as block
import mcpi.minecraftstuff as minecraftstuff
```

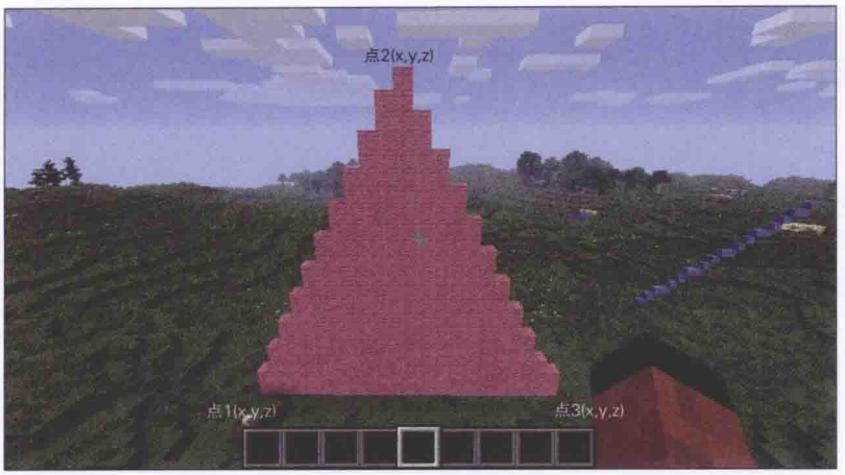


图 7-8 `drawFace()` 被用来根据三个点创建一个三角形

3. 连接 minecraft，然后创建 `MinecraftDrawing` 对象：

```
mc = minecraft.Minecraft.create()
mcdrawing = minecraftstuff.MinecraftDrawing(mc)
```

4. 获取玩家当前位置：

```
pos = mc.player.getTilePos()
```

5. 现在你需要创建一个列表来存放多边形的顶点。请输入：

```
points = []
```

6. 然后将三个点的（x，y，z）加入列表之中。一旦全部加入则会创建出一个三角形：

```
points.append(minecraft.Vec3(pos.x, pos.y, pos.z))
points.append(minecraft.Vec3(pos.x + 20, pos.y, pos.z))
points.append(minecraft.Vec3(pos.x + 10, pos.y + 20,
                             pos.z))
```

7. 使用 `MinecraftDrawing.drawFace` 函数来创造三角形：

```
mcdrawing.drawFace(points, True, block.WOOL.id, 6)
```

8. 保存文件然后运行程序来创建三角形。

深入代码

多边形表面上的顶点使用 `minecraft.Vec3(x, y, z)` 来创建。`minecraft.Vec3()` 是 Minecraft API 中用来将一系列坐标（x，y，z）保存在一起的方式。Vec3 是 3D 向量（vector）的缩写。

使用 `append()` 将表面上的顶点加入顶点列表中。这个函数将会把新的顶点加入列表最后。

挑战

你能用 drawFace() 做些其他什么形状呢？比如五边形？

金字塔

寻找一张金字塔的图片，然后好好观察一下。你注意到了什么呢？它每一面都是什么形状呢？所有的面又都有什么共同点呢？它有几个面呢？

正如你所见，金字塔的每个面（除了底面）都是三角形。埃及的金字塔有四个面（如果包含底面的话是五个面），但金字塔可以有三个或者更多面。你是否也同样注意到任何金字塔的底面正好可以精确地放到一个圆里！看看图 7-9 来试图理解我的意思。

你现在将创建一个程序，使用 drawFace() 和 finePointOnCircle() 函数来制作任意高度、任意面数的金字塔的每一个面：

1. 首先，打开 IDLE 编辑器并创建一个新文件。将其以 MinecraftPyramids.py 为名保存到 MyAdventures 文件夹内。
2. 输入下面的代码导入 minecraft、block、minecraftstuff 和 math 模块：

```
import mcpi.minecraft as minecraft
import mcpi.block as block
import mcpi.minecraftstuff as minecraftstuff
import math
```

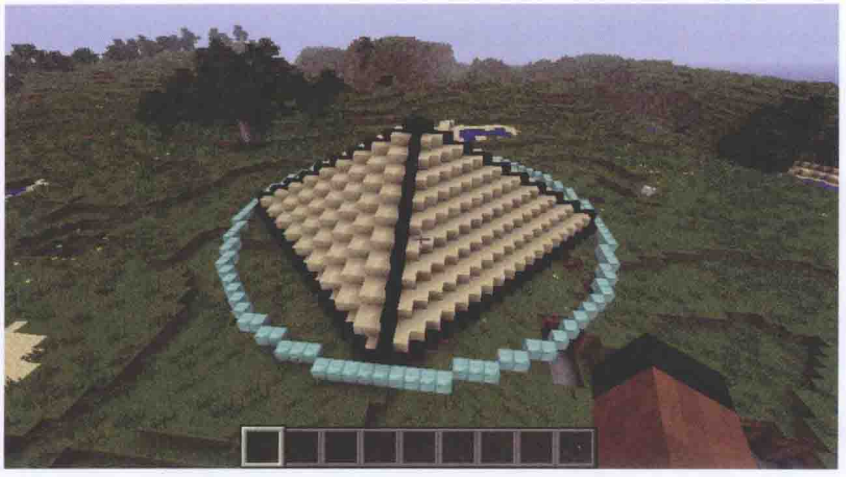


图 7-9　一个用三角形组成的金字塔，它可以精确地放到一个圆里

3. 创建finePointOnCircle()函数。这个函数用于给出用于组成金字塔的三角形将被放置的位置:

```
def findPointOnCircle(cx, cy, radius, angle):
    x = cx + math.sin(math.radians(angle)) * radius
    y = cy + math.cos(math.radians(angle)) * radius
    x = int(round(x, 0))
    y = int(round(y, 0))
    return(x,y)
```

4. 连接到 Minecraft 创建 MinecraftDrawing 对象:

```
mc = minecraft.Minecraft.create()
mcdrawing = minecraftstuff.MinecraftDrawing(mc)
```

5. 现在为你的金字塔建立一个变量。金字塔的中心将会是玩家的位置。那些变量的值将会改变金字塔的大小（半径）、高度和面的个数。请输入:

```
pyramidMiddle = mc.player.getTilePos()

PYRAMID_RADIUS = 20
PYRAMID_HEIGHT = 10
PYRAMID_SIDES = 4
```

6. 金字塔的每一面都要进行循环。从这里开始，之后的所有代码将会被放进 for 循环:

```
for side in range(0, PYRAMID_SIDES):
```

金字塔越大，程序运行的耗时就越长，Minecraft 在游戏中显示这个金字塔的耗时也就越长。如果金字塔过高，那么它同样会突破游戏中的最高高度。所以慢慢来，逐渐增大变量的值。你可以建造巨大的金字塔，但是你可能需要耐心点。那些金字塔真的非常大，可能需要好一会儿才能出现在你的眼前。

7. 对于金字塔的每一面，计算三角形侧面的角度，然后使用 FindPointOnCircle() 来找到（*x*，*y*，*z*）坐标。根据金字塔侧面的个数将 360° 等分，然后乘以所需要绘制的侧面次序，以此来计算角度。这恰如你在图 7-10 看到的一样。请输入下面的代码:

```
  point1Angle = int(round((360 / PYRAMID_SIDES) * side,0))
  point1X, point1Z = findPointOnCircle(
    pyramidMiddle.x, pyramidMiddle.z,
    PYRAMID_RADIUS, point1Angle)
  point2Angle = int(round((360 / PYRAMID_SIDES)
                          * (side + 1),0))
  point2X, point2Z = findPointOnCircle(
    pyramidMiddle.x, pyramidMiddle.z,
    PYRAMID_RADIUS, point2Angle)
```

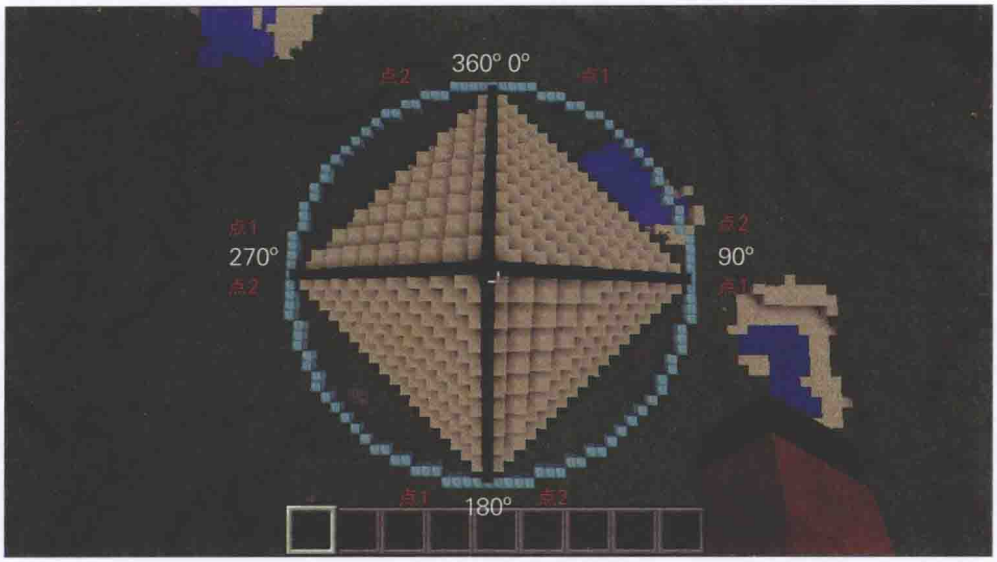


图 7-10　具有四个侧面金字塔的角度

8. 创建三角形的顶点，然后用 `drawFace()` 来创建金字塔的面，如下所示：

```
trianglePoints = []
trianglePoints.append(
  minecraft.Vec3(point1X, pyramidMiddle.y, point1Z))
trianglePoints.append(
  minecraft.Vec3(point2X, pyramidMiddle.y, point2Z))
trianglePoints.append(
  minecraft.Vec3(pyramidMiddle.x,
                 pyramidMiddle.y + PYRAMID_HEIGHT,
                 pyramidMiddle.z))
mcdrawing.drawFace(trianglePoints, True,
                   block.SANDSTONE.id)
```

沙石（`block.SANDSTONE.id`）对于金字塔来说是非常有用的一类方块，因为沙石看起来非常接近沙子，但却有着一个非常有用的特性——它不受重力的影响，即使其紧靠的正下方没有方块提供支撑，沙石也不会下落。如果你想用沙子来完成金字塔，那么玩家将会被掩埋，且需要相当多的时间把自己挖出来。

9. 保存文件然后运行程序。你将会看见在玩家上方出现一个金字塔，然后把他关在了里面！

这个程序能够创建任意大小的有着任意侧面数的金字塔。尝试改变程序中金字塔的变量，然后再次运行程序。图 7-11 展示了两个令人印象深刻的样例。

你可以从本书的配套资源网站 www.wiley.com/go/adventuresinminecraft 处下载 Minecraft 金字塔的完整代码。

挑战

你已经创造的金字塔并没有底面。你能够创造一个吻合金字塔底部的多边形么？对于4个侧面的金字塔来说，如果你编程无误的话，实现并不困难。这对于有5、6甚至7个侧面的金字塔应该同样适用。

图 7-11　Minecraft 金字塔

2D 和 3D 图形的深入冒险

你使用 `drawFace()` 函数就能创造出任何种类的多面体（polyhedron），那些多面体通过平整的表面（就像之前你创建的金字塔）来塑造。那么为什么不创建更多的多面体呢？

你会在下面这些地方发现许多多面体的样例和好点子：

- Maths is Fun（www.mathsisfun.com/geometry/polyhedron.html）
- Kids Math Games（www.kidsmathgamesonline.com/facts/geometry/3dpolyhedronshapes.html）

快速参考表

命令	描述

`mcdrawing.drawLine(0,0,0,`
`10,10,10,`
`block.DIRT.id)`
	在两点之间绘制一条直线。

`mcdrawing.drawCircle(0,0,0,`
`radius,`
`block.DIRT.id)`
	绘制一个圆。

`mcdrawing.drawSphere(0,0,0,`
`radius,`
`block.DIRT.id)`
	绘制一个球体。

`tri = []`
`filled = True`
`tri.append(minecraft.Vec3(0,0,0))`
`tri.append(minecraft.Vec3(10,0,0))`
`tri.append(minecraft.Vec3(5,10,0))`
`mcdrawing.drawFace(tri,`
`filled,`
`block.DIRT.id)`
	绘制一个多边形或表面（例如一个三角形）。

解锁成就：3D 大师、大规模结构创建者、金字塔建筑师太棒了！

在下一次冒险中……

下一次冒险中，你将会学习到如何赋予 Minecraft 对象以它们自己的思想，和一个方块作朋友以及避免外星人入侵。

冒险 8

赋予方块以独立思维

计算机不会思考。它们根本不能拥有任何想法，它们永远只能做那些编程者让它们做的事情。但你可以让计算机看似自己在思考和决策。通过编程来让你的计算机“理解”正在发生的事情，然后给予它规则来“决定”接下来需要做些什么，这样你就打开了通往乐趣的大门。

本次冒险中，你将编写一个只要不是离太远就会伴你左右的方块好友。同样你也将创造一个不追上你并将你吸入自己就誓不罢休的飞碟。

你将学到如何让一个方块沿着它认为是最好的路径移动，以及如何使用 Python 的随机（`random`）模块来使计算机看似正在思考。你还会用 `minecraftstuff` 模块（已在你的初学者工具包中）中的 `MinecraftShape` 函数来创造图形。

你的方块好友

Minecraft 可以成为一个孤独的世界。但是玩家不必如此孤独——你可以创建一个伴玩家左右的方块，与玩家交谈并和玩家做朋友（见图 8-1）。

为了编写一个方块好友，你需要以她的角度来思考！当她靠近玩家时，她会变得开心，所以她想伴随玩家左右并试图坐在玩家身旁。只要玩家足够靠近她，她就会一直保持好心情；如果玩家走开了，她就会跟随玩家。如果玩家离得太远，那么方块好友就会变得沮丧，她会停止跟随，直到玩家返回并给她一个拥抱（站在她边上即可）。

这个方块好友的程序有两个不同部分：

- 方块决定接下来应该如何行动的规则：这部分程序让方块好友做出决策，需要向玩家（即目标）移

动还是停留原地。

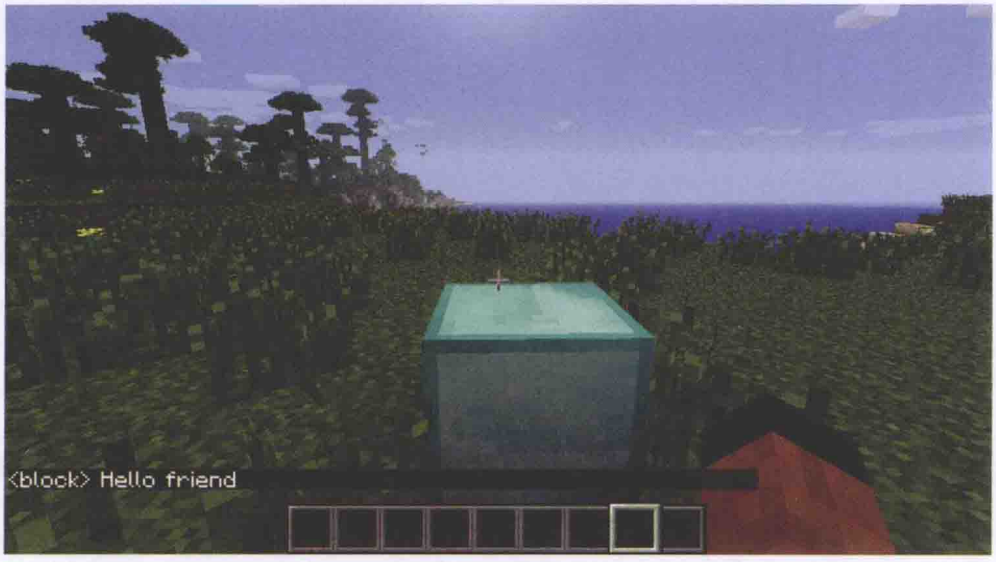


图 8-1　创建一个会打招呼的方块好友，让她在玩家的 Minecraft 冒险中陪伴左右

- 让方块向一个目标移动的代码：一旦方块好友达到了她的目标，那么她就会再次使用那些规则来决定接下来该如何行动。

当方块好友向目标移动时，她应该会在地面“上方”运动，而不是浮在空中或者在地下钻洞！使用 `mc.getHeight(x,z)` 函数并传递一个水平坐标（*x*，*z*）来实现这一点。这个函数将返回从天空向下第一个非空气方块（`AIR`）的垂直位置（y）。

访问配套资源网站 www.wiley.com/go/adventuresinminecraft 观看指导如何创建方块好友的视频。

为方块好友创建一个新程序，我们从这里开始吧：

1. 打开 IDLE 编辑器，单击 File⇨New File 来创建一个新程序，然后将其以 `BlockFriend.py` 为名保存于 `MyAdventures` 文件夹内。

2. 导入 `minecraft`、`block`、`minecraftstuff`、`math` 以及 `time` 模块：

```
import mcpi.minecraft as minecraft
import mcpi.block as block
import mcpi.minecraftstuff as minecraftstuff
import math
import time
```

3. 首先你需要做的就是创建一个函数来计算两点间的距离。这个函数会用来判断方块好友距离玩家有多远：

```
def distanceBetweenPoints(point1, point2):
  xd = point2.x - point1.x
  yd = point2.y - point1.y
  zd = point2.z - point1.z
  return math.sqrt((xd*xd) + (yd*yd) + (zd*zd))
```

4. 现在你要决定当玩家离方块好友多远时她才会停止跟随玩家。创建一个常量来保存你认为是“太远”的距离。我选用了15，这就表示当方块好友和玩家之间距离15个方块时，方块好友就会停止跟随玩家：

```
TOO_FAR_AWAY = 15
```

5. 创建 `Minecraft` 和 `MinecraftDrawing` 对象：

```
mc = minecraft.Minecraft.create()
mcdrawing = minecraftstuff.MinecraftDrawing(mc)
```

6. 创建一个变量来储存方块的心情。对于本次冒险来说，方块不是“高兴”（happy）就是“沮丧”（sad）。我们将其设置为 `"happy"`：

```
blockMood = "happy"
```

7. 获取玩家位置并在 x 方向上的位置加5，然后使用 `getHeight()` 函数来找到 y 方向上的位置。如此，方块就会位于地面上方。通过这种办法，创建一个方块好友，也就是离玩家一段距离的一些方块：

```
friend = mc.player.getTilePos()
friend.x = friend.x + 5
friend.y = mc.getHeight(friend.x, friend.z)
mc.setBlock(friend.x, friend.y, friend.z,
            block.DIAMOND_BLOCK.id)
mc.postToChat("<block> Hello friend")
```

8. 为方块好友创建一个目标。这将会是她移动时前往的地方。一开始，将目标设定在方块好友的当前位置。这样一来，在程序启动时，方块好友不会试图移动到其他地方。

```
target = friend.clone()
```

> 如果想在 Minecraft 中复制位置，你可以使用 `clone()` 函数。这是因为位置是 Python 的对象，由 Minecraft API 返回，它们不同于其他一般的变量。例如，如果你使用代码 `target = friend`，然后改变 `friend` 对象中 `friend.x` 的值，那么 `target` 中 `target.x` 的值也一样会改变。
>
> 
>

9. 启动一个无限循环，如此一来，程序就能一直运行下去（注意这之后的所有代码都在这个循环内部）。

```
while True:
```

10. 获取玩家的位置，然后使用 `distanceBetweenPoints()` 函数计算玩家和方块好友之间的距离：

```
  pos = mc.player.getTilePos()
  distance = distanceBetweenPoints(pos, friend)
```

11. 应用那些你想要方块好友遵循的规则，然后决定她接下来应该做些什么。如果方块好友很“高兴”

（happy），那么让程序比较方块好友和玩家之间的距离以及所谓“太远”（too far away）的常量。如果距离比“太远”小，那么将目标设定为玩家所在的位置。如果距离比“太远”大，那么将方块的心情设置为“sad”（见图 8-2）：

```
if blockMood == "happy":
  if distance < TOO_FAR_AWAY:
    target = pos.clone()
  elif distance >= TOO_FAR_AWAY:
    blockMood = "sad"
    mc.postToChat("<block> Come back. You are too far ↩
                   away. I need a hug!")
```

12. 现在你需要告诉程序，如果是其他情况（也就是距离并非比“太远”的常量小），那么方块好友将会变得 `"sad"`，且她将一直保持这样的状态直到玩家重新回到这个方块的范围内（一个拥抱），这个方块的心情才会变为 `"happy"`：

```
elif blockMood == "sad":
  if distance <= 1:
    blockMood = "happy"
    mc.postToChat("<block> Awww thanks. Let's go.")
```

图 8-2　方块好友现在很伤心

13. 下一部分的代码将方块好友向目标移动：

```
if friend != target:
```

14. 接下来，通过使用 `MinecraftDrawing` 中的 `getLine()` 函数找到所有处于方块好友和她目标之间的方块：

```
  blocksBetween = mcdrawing.getLine(
    friend.x, friend.y, friend.z,
    target.x, target.y, target.z)
```

getLine() 函数和 drawLine() 函数一样使用（见冒险 7），但其返回的是在两组作为参数传递的（x，y，z）坐标点之间，连成一条直线的点（x，y，z）的列表，而不是使用方块绘制直线。

15. 你需要直接在之前的代码下面告诉你的程序，在方块好友和目标之间所有的方块都要经过循环，并且将方块好友移动到她和玩家之间的下一个方块处：

```
    for blockBetween in blocksBetween[:-1]:
      mc.setBlock(friend.x, friend.y, friend.z, block.AIR.id)
      friend = blockBetween.clone()
      friend.y = mc.getHeight(friend.x, friend.z)
      mc.setBlock(friend.x, friend.y, friend.z,
                  block.DIAMOND_BLOCK.id)
      time.sleep(0.25)
    target = friend.clone()
```

当 for 循环结束时，方块好友就到达了她的目标。所以之后要将方块好友的目标设定为她的当前位置。这样她就不会再移动了。

程序是这样移动方块好友的：将她从之前的位置清除［也就是将方块设定成空气（AIR）］，更新直线中方块好友下一个方块位置处，然后在那个位置重新创造出方块好友。

注意 blockBetween 和 blocksBetween 的区别。前者存储了方块好友所在的位置，后者存储了在方块好友从开始运动以及她正在运动的位置之间的方块列表。

方块好友的速度根据程序每次方块移动步骤之间的睡眠时间——time.sleep(0.25) 来设定。如果睡眠时间过短，那么方块就会运动过快，玩家则永远无法远离她。如果睡眠时间过长，方块就会移动过慢。这会让你抓狂！

16. 循环末端输入一个较小的延迟，这样程序就不会因为不停循环，不停请求玩家位置而让 Minecraft 过载：

```
    time.sleep(0.25)
```

17. 运行程序。

方块好友将会出现在玩家身边，然后紧跟其后，直到你们之间的距离过远。发生这种情况时，方块好友将会停止工作，而玩家将不得不往回走，然后站在靠近她的位置，直到她将重新跟随玩家。

访问本书配套资源网站 www.wiley.com/go/adventuresinminecraft 下载方块好友程序的完整代码。

深入代码

为了计算方块和玩家的距离，你创建了一个名为 distanceBetweenPoints(point1,point2) 的函数。其使用了毕达哥拉斯定理来计算两点之间的精确距离。你也许已经在冒险 4 中“添加一盏归航信标”的挑战中研究过毕达哥拉斯定理是如何运用的了。

1. 得到 point1 和 point2 的 x，y，z 的差值：

```
xd = point2.x - point1.x
yd = point2.y - point1.y
zd = point2.z - point1.z
```

2. 计算两点坐标值差值的平方：

```
xd*xd
```

3. 将平方值都加起来：

```
(xd*xd) + (yd*yd) + (zd*zd)
```

4. 返回两点的间隔，即对平方和使用 Python 的函数 mach.sqrt() 算得的平方根：

```
return math.sqrt((xd*xd) + (yd*yd) + (zd*zd))
```

访问 www.mathsisfun.com/algebra/distance-2-points.html 了解更多关于毕达哥拉斯定理以及如何运用它来计算两点间的距离。

方块好友通过找寻她和玩家之间的方块，然后向方块移动，循环往复这个过程直至通过所有的方块：

```
for blockBetween in blocksBetween[:-1]:
```

这里的 [:-1] 告诉 Python 循环 BlocksBetween 列表中的所有方块，直到循环到了最后一个方块。这样的话，当方块好友向玩家移动时，她将会站在玩家边上而不是玩家头顶。

一个数的平方根（square root）即平方根自己乘以自己将会得到的这个数。例如，9 的平方根是 3，因为 3 x 3 = 9。

挑战

1. 方块好友总是以同样的速度移动：移动 1 个方块的距离约 0.25 秒，或者说是每秒 4 个方块。然而，现实的朋友当目标离得较远时将会加速向目标前进。尝试改写程序，让方块好友离玩家越远，移动速度越快。

2. 重新回顾冒险 4 中的“添加一个导航信标”并再次完成挑战。这一次挑战请使用 distanceBetweenPoints() 函数来创建一个更好的估距方式。

使用随机数让你的方块好友更富趣味

你刚才写的方块好友的程序，它的问题在于，所有的行为都是可预测的（predictable），方块好友

总是会做同样的事。当你仅仅运行几次程序后，你就会很清楚地知道接下来她会在什么时候做些什么动作了，这样会很快变得无聊。为了真正赋予我们的方块好友以“思维”，你需要给好友方块一片不可预测的空间。

> 当某件事情是可预测的（predictable），这意味着你能够在事情发生之前预见。如果你希望方块好友表现真实的话，这样并不好。真实的实物不总是可预测的。
>
> 
>

通过使用随机数的方式，你能够数模拟不可预测性——或者说，使某件事情看起来好像是不可预测的。你通过编程，基于概率（probability）选择需要做的事来实现这种不可预测性。比如，你可能会告诉程序每 100 次执行一次一个特定的动作。基于不同概率加入更多规则，通过这种方式，你能够让程序更加难以被预测。在改变方块好友的程序，使用随机数之前，我们先探索一下创造随机数的代码以及概率检验。你将会回忆起冒险 3 中引入的随机数。

> 概率（probability）是指某件事情发生的可能性的度量。例如当我抛掷硬币时，50%（或者二分之一）的机会正面朝上。
>
> 
>

Python 的 `random` 模块包含有 `random.randint(startNumber, endNumber)`，这个函数被用来产生两个特定数字（`startNumber`，`endNumber`）之间的随机数。

下面的代码每次运行都将打印出一个 1 ~ 10 的随机数。如果你想要查看结果，你可以创建一个新的 Python 程序：

```
import random
randomNo = random.randint(1,10)
print randomNo
```

加入 `if` 语句来检查随机数为 10 时，你就创造了一个概率检测器，其大约每 10 次运行中有一次得到的结果为真：

```
import random
if random.randint(1,10) == 10:
  print "This happens about 1 time in 10"
else
  print "This happens about 9 times out of 10"
```

如果你试图运行程序超过 100 次，你预计可以看见“This happens about 1 time in 10”打印了约 10 次（见图 8-3），但你也许只会看见 9 次或 11 次，甚至根本没有。因为这是不可预测的！

如果你试图运行程序 100 次，那么你预计可以看见打印“This happens about 1 time in 10”多少次呢？你预计会看见 10 次，但也许并非如此，你也可能一次也看不到，或者看到 100 次！

如果你在方块好友的程序中使用随机数以及一个概率检测装置，那么你就能让她的行为变得更加难以预测。你甚至能让方块好友对玩家“不友好”！

给方块好友的程序加入一些新规则。这样一来，如果方块好友“沮丧”的话，就有 1% 的机会她会觉得自己受够了等待，且当玩家回来甚至与她拥抱时也不再会跟随玩家：

1. 打开 IDLE 编辑器，然后打开 `MyAdventures` 文件夹中的 `BlockFriend.py` 程序。
2. 单击“文件”（File）⇨ 另存为（Save As），然后以 `BlockFriendRandom.py` 保存文件。

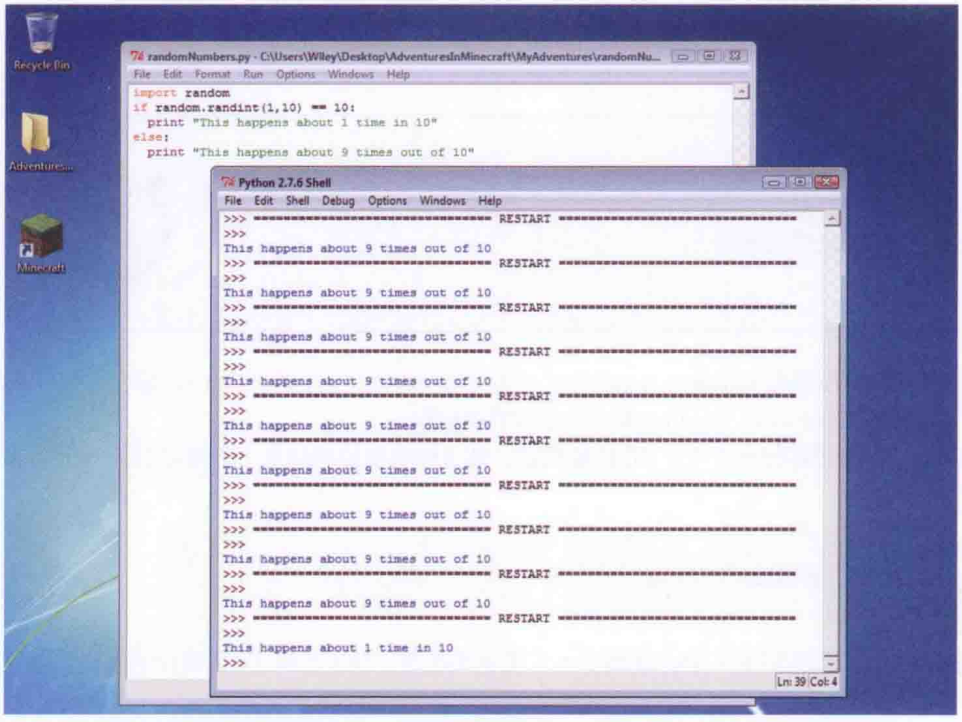


图 8-3　创建随机数来给你的好友方块一片不可预测的空间

3. 将 `random` 模块加入程序开头的 `import` 语句之中（代码已加粗）：

```
import mcpi.minecraft as minecraft
import mcpi.block as block
import mcpi.minecraftstuff as minecraftstuff
import math
import time
import random
```

4. 加入一个 1 到 100 的随机数检测当方块的心情为 "sad" 时转化为 "hadenough"。

```
    elif blockMood == "sad":
      if distance <= 1:
        blockMood = "happy"
        mc.postToChat("<block> Awww thanks. Let's go.")
      if random.randint(1,100) == 100:
        blockMood = "hadenough"
        mc.postToChat("<block> That's it. I have had ↩
                       enough.")
```

5. 运行程序。

当玩家离方块好友太远时，如果你等待足够长的时间（1% 的机会会实现），方块好友就会作出决定，她已经受够了，将不再对玩家“友好”！一旦如此，方块好友就会说：“就这样吧，我已经受够了。”然后永远坐在那里。

挑战

再次修改程序，如果方块好友的心情是 "hadenough"（受够了），那么当玩家给她一个拥抱时，她有 1/50 的机会原谅玩家。

更大的模型

在方块好友的程序中，你让一个方块伴随玩家左右且跟随玩家。但是如果你想要许多方块在玩家身边移动又会如何呢？一个用运动着的方块构成的模型，比如小汽车或者外星人的飞船又如何呢？

这就是难点，因为你需要记录许多方块。每次你想要让这个模型移动时，你都需要将所有的方块设定为空气（AIR），然后在新位置重新创建它们。如果方块非常多，那么模型就不会移动地恰到好处，而且会减缓移动。

minecraftstuff 模块包含了一个叫 MinecraftShape 的函数，这个函数专门为了创建模型以及让模型运动起来而编写。它跟踪记录构成模型的所有方块，通过这种方式来实现功能。模型经过移动后，这个函数仅仅改变那些变化的方块，而非所有。

为了使用 MinecraftShape，你需要创建一个带有模型中方块的列表来告诉函数，这个模型是什么样的。模型中的每一个方块都有一个位置信息（x，y，z）以及方块种类的信息。

图 8-4 展示了用 7 个方块组成的一个简单的模型，以及那些方块的位置信息。如此一来，“位置信息（position）”就和 Minecraft 中的位置信息不一样了。你会看到，木马的中心是（0，0，0），每个方块都与中心的方块有联系，所以中心方块的右边就是（1，0，0）。

图 8-4　Minecraft 木马模型的方块位置

现在你将新建一个程序，使用 MinecraftShape 来创建图 8-4 中的木马并让它移动：

1. 打开 IDLE 编辑器，点击文件（File）⇨ 新建（New File）来创建一个新程序。以 WoodenHorse.py 为名将文件保存在 MyAdventures 文件夹下。

2. 导入 minecraft、block、minecraftstuff 以及 time 模块：

```
import mcpi.minecraft as minecraft
import mcpi.block as block
import mcpi.minecraftstuff as minecraftstuff
import time
```

3. 创建 Minecraft 对象：

```
mc = minecraft.Minecraft.create()
```

4. 用 ShapeBlock 定义每个方块的位置关系和种类来创建方块列表。通过这种方式创建一个木马模型。

```
horseBlocks = [
    minecraftstuff.ShapeBlock(0,0,0,block.WOOD_PLANKS.id),
    minecraftstuff.ShapeBlock(-1,0,0,block.WOOD_PLANKS.id),
    minecraftstuff.ShapeBlock(1,0,0,block.WOOD_PLANKS.id),
    minecraftstuff.ShapeBlock(-1,-1,0,block.WOOD PLANKS.id),
    minecraftstuff.ShapeBlock(1,-1,0,block.WOOD_PLANKS.id),
    minecraftstuff.ShapeBlock(1,1,0,block.WOOD_PLANKS.id),
    minecraftstuff.ShapeBlock(2,1,0,block.WOOD_PLANKS.id)]
```

方块的位置信息与图 8-4 中显示的完全相同。

5. 你需要告诉 MinecraftShape 在 Minecraft 世界中的什么地方创建木马。获取玩家位置然后在 z 和 y 轴方向上加 1，这样一来木马不会直接生成在玩家头顶：

```
horsePos = mc.player.getTilePos()
horsePos.z = horsePos.z + 1
horsePos.y = horsePos.y + 1
```

6. 使用 MinecraftShape 创建木马，且传递 Minecraft 对象、木马模型应该被创建的位置以及 ShapeBlocks 的列表作为参量：

```
horseShape = minecraftstuff.MinecraftShape(mc, horsePos,
                                           horseBlocks)
```

7. 运行程序。看！你应该已经看见玩家身边出现的木马了。
8. 修改 WoodenHorse.py 程序，在程序最后加入下列代码来让木马运动：

```
for count in range(1,10):
  time.sleep(1)
  horseShape.moveBy(1,0,0)
horseShape.clear()
```

moveBy(x, y, z) 函数告诉图形在 *x*、*y*、*z* 方向上“移动”的方块数。所以在这个样例中，horseShape 在 *x* 方向上会移动 1 个方块。clear() 函数移除图形，将所有的方块设定为空气方块（AIR）。

9. 运行程序，然后看木马飞驰！

和 moveBy(x,y,z) 以及 clear() 一样，你也能告诉 move(x,y,z) 让一个模型移动到 Minecraft 中的任意位置。如果图形已经被清除，那么你可以用 draw() 重新创建它。

你可以在本书的配套资源网站 www.wiley.com/go/adventuresinminecraft 下载木马完整的代码。

你将会发现模型相当有用，因为接下来你将要创建一艘外星飞船！并且之后你还会用到模型，用来创建障碍物。

外星人入侵

外星人正打算入侵 Minecraft。一艘宇宙飞船将从天而降至玩家上方，而玩家则身处险境——那些外星人并不友好，而且若不完成它们自己所有的任务就誓不罢休。

下一个程序中，你将使用 MinecraftShape 以及在方块好友的程序中使用到的编程技巧，创建一艘外星飞船（见图 8-5）。这艘飞船将会盘旋在地表上方，追赶玩家并尝试移动到玩家头顶。一旦如此，那么它将传送你上飞船。

你可以使用 MinecraftShape 创建一艘外星飞船，就像在木马模型程序中一样，图形中的每个方块都有自己的相对位置和方块种类。图 8-6 从侧面和上面展示了构成图形的方块位置。

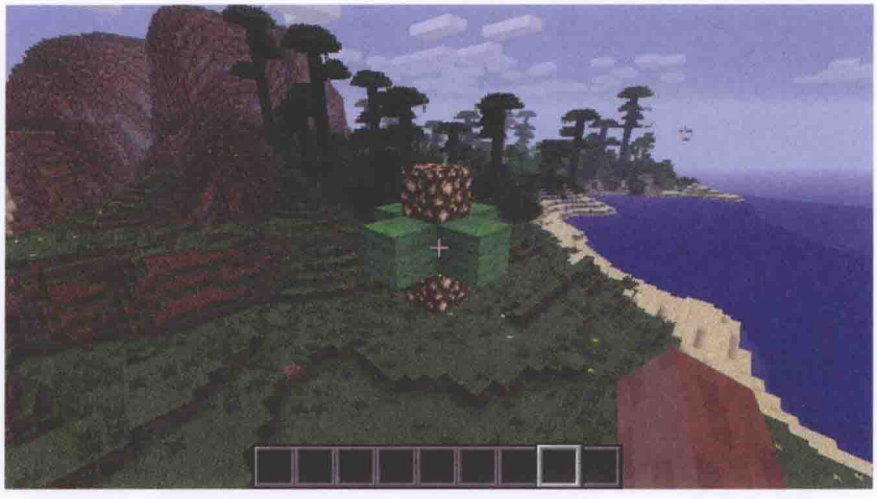


图 8-5　创造一艘外星飞船让你的 Minecraft 游戏更有趣！

图 8-6　外星飞船的方块位置

外星飞船向你展示了如何使用 `MinecraftShape` 对象创造三维模型。模型可大可小，可简单可复杂，只要你喜欢。这给你在 Minecraft 编程过程中使用图形提供了很多选择。

就像方块好友的程序一样，外星人入侵的代码也会有两部分。第一部分即决定飞船接下来该如何行动的规则；第二部分控制外星飞船向玩家移动。

外星飞船追赶玩家时会在对话栏中以消息（比如“你不能逃一辈子”）的方式嘲讽玩家。发送的消息从嘲讽内容的列表中随机选择（见图 8-7）。

决定外星飞船接下来应该做什么的规则将基于三种模式：

着陆模式：程序启动时，这将是飞船的初始模式，飞船将从天而降直至玩家上方。

攻击模式：飞船一着陆就发起攻击，不断追逐玩家直到在玩家正上方，然后用“光线”将玩家吸入。

任务完成模式：玩家已经被飞船用“光线”吸入且外星人打算让他返回地球后，飞船会被设置为该模式。此时程序将终止，而玩家将被用“光线”传送返回。

一旦外星飞船已经捕获玩家，程序将构建一个令人感到忧郁的房间。这个房间会困住玩家，且会改变玩家的位置让他保持在这个空间的内部（见图 8-8）。然后，外星人在将玩家传送回去，也就是让玩家回原先的位置且在清除房间前将给他发送一些消息。

图 8-7　外形飞船追赶玩家

图 8-8　在外星飞船内部

根据如下步骤来创建外星人入侵的程序：

1. 打开 IDLE 编辑器，单击 New⇨New File 并以 AlienInvasion.py 为名保存于 MyAdventures 文件夹内。

2. 导入 minecraft、block、minecraftstuff 以及 time 模块：

```
import mcpi.minecraft as minecraft
import mcpi.block as block
import mcpi.minecraftstuff as minecraftstuff
import time
```

3. 创建 distanceBetweenPoints() 函数：

```
def distanceBetweenPoints(point1, point2):
    xd = point2.x - point1.x
    yd = point2.y - point1.y
    zd = point2.z - point1.z
    return math.sqrt((xd*xd) + (yd*yd) + (zd*zd))
```

4. 为程序创建一些常量。HOVER_HEIGHT 即外星飞船在玩家上方盘旋时距离玩家的方块数；ALIEN_TAUNTS 即讥讽语句的列表，其中的内容将在外星人追逐玩家时显示在对话栏内：

```
HOVER_HEIGHT = 15
ALIEN_TAUNTS = ["<aliens>You cant run forever",
                "<aliens>Resistance is useless",
                "<aliens>We only want to be friends"]
```

你也可以更改外星人的嘲讽内容——看看你有多少创造性了！如果你喜欢的话，也可以加入更多。

5. 创建 Minecraft 以及 MinecraftDrawing 对象：

```
mc = minecraft.Minecraft.create()
mcdrawing = minecraftstuff.MinecraftDrawing(mc)
```

6. 设置外星飞船的初始位置和初始模式，即玩家正上方 50 个方块处以及“着陆模式（landing）”：

```
alienPos = mc.player.getTilePos()
alienPos.y = alienPos.y + 50
mode = "landing"
```

7. 使用 MinecraftShape 创建外星飞船（见图 8-6，它可以提示你这一步是如何完成的）：

```
alienBlocks = [
   minecraftstuff.ShapeBlock(-1,0,0,block.WOOL.id, 5),
minecraftstuff.ShapeBlock(0,0,-1,block.WOOL.id, 5),
minecraftstuff.ShapeBlock(1,0,0,block.WOOL.id, 5),
minecraftstuff.ShapeBlock(0,0,1,block.WOOL.id, 5),
minecraftstuff.ShapeBlock(0,-1,0,
                          block.GLOWSTONE_BLOCK.id),
minecraftstuff.ShapeBlock(0,1,0,
                          block.GLOWSTONE_BLOCK.id)]

alienShape = minecraftstuff.MinecraftShape(mc, alienPos,
  alienBlocks)
```

8. 创建一个 while 循环，当飞船的模式并非任务完成模式（"Missionaccomplished"）时循环将继续，或者换句话说，当模式为“任务完成模式（Missionaccomplished）”时，退出循环：

```
while mode != "missionaccomplished":
```

9. 每次循环都获取玩家位置：

```
  playerPos = mc.player.getTilePos()
```

10. 代码的下一部分关于程序决定飞船后续行为所用的规则——若为着陆模式（"Landing"），则将瞄准目标（即外星飞船将飞往之处）设置为玩家位置的正上方，然后将模式设置为攻击模式（"attack"）：

```
  if mode == "landing":
    mc.postToChat("<aliens> We dont come in peace - please ↩
                   panic")
    alienTarget = playerPos.clone()
    alienTarget.y = alienTarget.y + HOVER_HEIGHT
    mode = "attack"
```

11. 不然的话，如果模式为攻击模式（"attack"），检视外星飞船是否在玩家上方。如果已经在玩家上方，那么就将它传送到飞船内部然后将模式设置为任务完成模式（"missionaccomplished"）。否则，如果玩家已经离开，那么将瞄准目标设定为玩家的当前位置，然后在对话栏中发送一条嘲讽的信息：

```
  elif mode == "attack":
    # 检查飞船是否在玩家正上方
    if alienPos.x == playerPos.x and alienPos.z == ↩
  playerPos.z:
      mc.postToChat("<aliens>We have you now!")

      # 创建一个房间
      mc.setBlocks(0,50,0,6,56,6,block.BEDROCK.id)
      mc.setBlocks(1,51,1,5,55,5,block.AIR.id)
      mc.setBlock(3,55,3,block.GLOWSTONE_BLOCK.id)

      # 传送玩家
      mc.player.setTilePos(3,51,5)
      time.sleep(10)
      mc.postToChat("<aliens>Not very interesting at all - ↩
                     send it back")
      time.sleep(2)

      # 将玩家传送到原来的位置
      mc.player.setTilePos(playerPos.x, playerPos.y,
                           playerPos.z)

      # 清除房间
      mc.setBlocks(0,50,0,6,56,6,block.AIR.id)

      mode = "missionaccomplished"
```

```
    else:
      # 玩家离开
      mc.postToChat(ALIEN_TAUNTS[random.
          randint(0,len(ALIEN_TAUNTS)-1)])
      alienTarget = playerPos.clone()
      alienTarget.y = alienTarget.y + HOVER_HEIGHT
```

当玩家被传送入飞船内部，一个出生点上方 50 个方块处将会生成一间房间，玩家的位置将会被设定在房间内部（就像 Doctor Who 的 Tardis 一样，内部空间比外面看上去的大！）。对话框内随后会呈现消息，直到玩家被重新设置回原先的位置然后房间被清除。

玩家被传送到的房间创建于出生点上方的空中，这在一定程度上是为了方便，但是这同样是因为远离一切会更好。你可以在任何你喜欢的地方创建这个房间，但是因为房间会在玩家离开的同时被清除，而玩家返回与之前相同的 Minecraft 世界中。

12. 如果外星飞船的位置与目标位置不同（在上面的规则中设置），则在程序最后缩进的 `while` 循环内，输入下面的代码来移动外星飞船：

```
  if alienPos != alienTarget:
    blocksBetween = mcdrawing.getLine(
      alienPos.x, alienPos.y, alienPos.z,
      alienTarget.x,alienTarget.y,alienTarget.z)
    for blockBetween in blocksBetween:
      alienShape.move(blockBetween.x, blockBetween.y,
                      blockBetween.z)
      time.sleep(0.25)
    alienPos = alienTarget.clone()
```

13. 到此，程序将会返回 `while` 循环的顶部。当模式被设置为任务完成模式（`"missionaccomplished"`）且 while 循环结束时，程序的最后一行将会让外星飞船消失：

```
alienShape.clear()
```

14. 运行程序，然后注意从天而降直至你头顶的外星人。

你可以从本书的配套资源网站 www.wiley.com/go/adventuresinminecraft 下载外星人入侵的完整代码。

深入代码

每次外星人追逐玩家时，从常量 `ALIEN_TAUNTS` 中会随机选择嘲讽的语句，然后将它显示在对话栏中。

```
mc.postToChat(ALIEN_TAUNTS[random.randint(0,len(ALIEN_
TAUNTS)-1)])
```

ALIEN_TAUNTS是一个字符串的列表; random.randint() 函数被用来从列表中选择一项，随机选择一个介于 0 和 ALIEN_TAUNTS 列表长度减去 1 这个范围内的数。

列表长度要减去 1 是因为当 len() 返回列表中项的实际个数（比如 3）时，你却只能从列表中的第 0 项开始引用（比如 0，1，2）。

挑战

外星飞船真的非常简单。尝试创造一艘惊人的飞船，一艘你真正喜欢的飞船。可以改变飞船，这样的话着陆时它就会切换至“潜伏模式”（lurk）。飞船将待在接近玩家的位置却并不攻击，然后基于一个随机概率，飞船将毫无征兆地切换到“攻击模式”（attack）。

仿真模拟的深入冒险

这次冒险中，你使用了算法和规则来模拟了一个好友和一次外星人入侵——那么更进一步，模拟其他东西如何呢？比如 Minecraft 中的一群鸟（或一群方块）、大海中前进的波浪，又或者是一个巨大的单元系统（cellular system），比如方块组成的生命游戏（Conway's Game of Life）。

访问 en.wikipedia.org/wiki/Conway's_Game_of_Life 了解更多关于生命游戏（Conway's Game of Life）的内容。

快速参考表

命令	描述
`import random`	导入 Phthon 的 random 模块
`random.randint(start,end)`	创建一个介于上下界之间的随机数
`import minecraftstuff`	带任意 minecraftstuff 扩展模块，这个模块已包含在初学者开发包中
`mc.getHeight(x,z)`	获取水平位置（*x*，*z*）处的地面（从天空往下的第一个非空气方块）高度（*y* 坐标值）
`mcdrawing = minecraftstuff. ↩ MinecraftDrawing(mc)`	创建 MinecraftDrawing 对象

续表

快速参考表

命令	描述
`copyOfPosition = position.clone()`	创建 Minecraft 位置的备份（复制）
`mcdrawing.getLine(x1,y1,z1,x2,y2,z2)`	获取两点之间直线上的所有方块

`shapeBlocks = [`
`  minecraftstuff.Sh apeBlock(1,0,0,`
`  block.DIRT.id),`
`  minecraftstuff.Sh apeBlock(0,0,1,`
`  block.DIRT.id)]`
	创建描绘 MinecraftShape 的 shapeBlocks 列表

`shape = minecraftstuff. ↩` `MinecraftShape(mc,pos,shapeBlocks)`	创建传递了特定位置的 shapeBlocks 的 MinecraftShape
`shape.moveBy(x,y,z)`	将一个 MinecraftShape 移动（x，y，z）
`shape.move(x,y,z)`	将一个 MinecraftShape 移动到（x，y，z）
`shape.clear()`	清除 MinecraftShape
`shape.draw()`	绘制 MinecraftShape

解锁成就：被你自己的人工智能诱拐！

在下一次冒险中……

下一次冒险中，你将会使用你至今从冒险过程中学到的所有技能来创造一个游戏。这个游戏中，你将会和时间赛跑，收集钻石。但是请注意——前途布满荆棘，它们都想要阻挡你的脚步。

冒险 9

大冒险：Crafty Crossing 游戏

现在你已来到了最后的大冒险！你将会利用在之前的冒险经历中学到的所有技能，在 Minecraft 世界中创造一个属于你自己的功能丰富的小游戏。你将会意识到 Minecraft 世界的多样性，并且仅仅通过一些简单的命令去获取和设置方块的属性和玩家的位置，就可以创造出非比寻常的游戏。

你也会学到一种新的编程技能，并用线程（threading）来使你的程序能够同时执行多个任务。

这个项目可以延伸到很多方面，并且当你完成它的时候，并不意味着你的冒险已经结束；相反，你只是到达了一个新的起点，会有更多富有创造性的、复杂性的以及挑战性的游戏等着你去创造。

游戏中的游戏

现在你要创造 Crafty Crossing 游戏。游戏目标是在时间结束之前尽量多收集钻石并到达游戏场景另一端的终点，途中一直会有一系列麻烦的障碍物来阻挡你。

每收集一颗钻石你都会得分，并且最终得分是将收集钻石的得分乘上你到达终点时剩余时间的秒数。

为了尽快穿越过游戏场景到达终点，你控制的玩家需要跳上一个河中的移动平台，从一个上下移动的墙下穿过，并且躲开地面上随机出现的陷坑（见图 9-1）。

图 9-1　创造 Crafty Crossing 游戏

访问 www.wiley.com/go/adventuresinminecraft 了解更多关于完整的 Crafty Crossing 游戏的信息。

创造游戏是一个很大的任务。在创造一个富有挑战性的游戏之前，你需要建造、测试并且最终将所有的游戏组成部分组合到一起。为了能让你更容易地做好这个项目，我把这份项目指南分成了四个部分，所以你可以按照每个部分分别开发和测试程序：

- 第一部分：创建程序的主要结构和构架，建造游戏场景。
- 第二部分：编写控制障碍物的程序，在玩家的道路上生成障碍物来阻碍玩家。
- 第三部分：在程序中加入“游戏逻辑”，创造不同难度的关卡，得分，当然还有判定“游戏结束”（Game Over）。
- 第四部分：利用在冒险 5 中学到的技能，重新利用按钮和七段数码管等硬件添加一个开始按钮，一个钻石计数器和一个提示玩家计时终止的指示器。

一旦完成了这个游戏程序，你可以利用任何创意继续开发，利用你自己的创造力，开发出你自己的个性游戏。

冒险中每个部分都有建议的有挑战性的拓展。在你完成整个冒险之前，你不应该去实现它们，否则你可能使任务变得过于复杂。当你完成整个项目后再回来尝试实现它们。

第一部分建造游戏场景

Crafty Crossing 的游戏场景是指所有游戏内容发生的地方。玩家将会以场景的一端为起点，另一端为终点——但是场景中会有障碍物阻碍玩家。

在没有任何障碍物的时候，游戏场景是由草（GRASS）方块组成的一个矩形区域，并且四周围上玻璃（GLASS）构成的墙（见图 9-2）。你将用常量去定义场景的宽、高和长（或者 *x*、*y* 和 *z*）。通过修改这些常量，你可以改变场景的大小和形状，并且障碍物将会自动重新生成在新的场景中。因为河流本身有两个方块深，所以场景的地面需要有三个方块深。

图 9-2　你的游戏场景

还记得在冒险 3 用来建造房子的 `setBlocks()` 函数吗？现在你将用它来建造游戏场景。

启动 Minecraft 与 IDLE，如果你是用 PC 或 Mac 平台，也要启动 Bukkit 服务端。对于这些你应该已经有一些经验了，当然你也可以再回到冒险 1 中查看具体做法。

为 Crafty Crossing 游戏创建一个新程序，首先设置好游戏程序的初始结构：

1. 打开 IDLE，创建一个新文件并在 `MyAdventures` 文件夹里以 `CraftCrossing.py` 文件名保存。
2. 导入需要的模块。你将用到一个新的模块 thread，在第二部分创造障碍物时我们将用到它：

```
import mcpi.minecraft as minecraft
import mcpi.block as block
import mcpi.minecraftstuff as minecraftstuff
import time
import random
import thread
```

3. 为游戏场景创造三个常量，即宽、高和长（*x*、*y* 和 *z*）：

```
ARENAX = 10
ARENAZ = 20
ARENAY = 3
```

4. 现在加入程序中要实现的函数声明。在你完成整个游戏之前，你将一一实现它们：

```
def createArena(pos):
  pass

def theWall(arenaPos, wallZPos):
  pass

def theRiver(arenaPos, riverZPos):
  pass

def theHoles(arenaPos, holesZPos):
  pass

def createDiamonds(arenaPos, number):
  pass
```

pass 语句并不做任何事，它只是作为占位符，用来标记将来此处会被实用的代码取代。

5. 建立和 Minecraft 的连接：

```
mc = minecraft.Minecraft.create()
```

6. 创建一个布尔型变量，它用来指示游戏是否结束。当游戏结束时为 `True`，游戏开始时设为 `False`：

```
gameOver = False
```

接下来你将会向主程序中添加更多的代码或者实现上面已经声明的函数。

此时你已经可以运行这个程序了。尽管没有任何事会发生，但是通过运行此程序，如果 Python Shell 里没有显示错误的话，你可以得知所有的模块已经设置好了。

下一步是来实现 `createArena` 函数，用来在 Minecraft 中创造游戏场景。

1. 在之前的程序中找到如下 `createArena()` 函数的代码：

```
def createArena(pos):
  pass
```

并删除函数体中缩进的 `pass` 语句。

2. 在 `def createArena(pos):` 行下缩进，建立到 Minecraft 的连接：

```
  mc = minecraft.Minecraft.create()
```

3. 现在你可以用传入的 `pos` 参数和常量 `ARENAX`、`ARENAY` 和 `ARENAZ`，利用 setBlocks 函数来创建游戏场景了（图 9-3 显示了这些常量是怎样被加到传入的位置上去创建游戏场景的）：

```
  mc.setBlocks(pos.x - 1 , pos.y, pos.z - 1,
               pos.x + ARENAX + 1, pos.y - 3,
               pos.z + ARENAZ + 1,
               block.GRASS.id)
```

4. 下一步是创建四周的玻璃墙。先创建一个玻璃方块构成的长方体，再在其中创建空气方块使其内部镂空：

```
    mc.setBlocks(pos.x - 1, pos.y + 1, pos.z - 1,
                 pos.x + ARENAX + 1, pos.y + ARENAY,
                 pos.z + ARENAZ + 1,
                 block.GLASS.id)
    mc.setBlocks(pos.x, pos.y + 1, pos.z,
                 pos.x + ARENAX, pos.y + ARENAY,
                 pos.z + ARENAZ,
                 block.AIR.id)
```

5. 现在 `createArena()` 函数已经完成，但是我们仍然需要从主程序中调用它。将下列代码添加到程序的底部，用来获取玩家当前的位置并以一个变量传给 `createArena()` 函数：

```
arenaPos = mc.player.getTilePos()
createArena(arenaPos)
```

6. 现在是时候运行你的程序了！你应该会看到在玩家周围创建出来的游戏场景，如图 9-3 所示。

图 9-3　创建游戏场景

挑战

游戏场景已经创建出来了，但是有点无聊！你能创建一个更好的场景吗？比如说在地面上添加一些指示箭头来指示玩家方向，或者在四周加一些装饰。或者添加一个房顶，并在上面装饰闪耀的火把？

第二部分创建障碍物

好游戏的一个特征是具有挑战性——容易的游戏很快就会让玩家觉得无聊。在接下来这部分冒险中，你将会创建一些障碍物来阻碍玩家，增加到达终点的难度。

墙

你将要创建的第一个障碍物是一堵砖墙，它横穿整个游戏场景并且上下移动。当它移下来时将会挡住道路，玩家不得不等到它移动到上部才能穿过（见图 9-4）。

图 9-4　你创建的墙可以阻挡玩家

一堵墙是一个非常简单而有效的阻挡玩家的方式，但是如果玩家能够有效利用时间，就可以快速从墙下面穿过去迎接后面的挑战。

你将利用 `minecraftstuff` 模块中的 `MinecraftShape`，在冒险 8 中创建 Alien Invasion 时你已经学过它们。

按照以下步骤实现 `theWall()` 函数来创造墙。

1. 在之前的程序中找到如下 `theWall()` 函数的代码：

```
def theWall(arenaPos, wallZPos):
  pass
```

并删除函数体中缩进的 pass 语句。

2. 在 `def theWall(arenaPos,wallZPos):` 行下缩进并建立和 Minecraft 的连接：

```
  mc = minecraft.Minecraft.create()
```

3. `theWall()` 函数需要传入两个参数：`arenaPos`（游戏场景的坐标），以及 `wallZPos`（在游戏场景中用来放置墙的 *Z* 方向的偏移）。利用这两个参数来创建墙的位置（见图 9-5）：

```
wallPos = minecraft.Vec3(arenaPos.x, arenaPos.y + 1,
                         arenaPos.z + wallZPos)
```

图 9-5　创建墙的位置

4. 现在你将要向建立这堵墙模型需要的方块列表里添加方块，利用两个 for 循环——一个沿着 *x* 方向，一个沿着 *y* 方向创建方块，并且将它们加入到 wallBlocks 列表里：

```
wallBlocks = []
for x in range(0, ARENAX + 1):
  for y in range(1, ARENAY):
    wallBlocks.append(minecraftstuff.ShapeBlock(x,
                                                y,
                                                0,
                                  block.BRICK_BLOCK.id))
```

5. 利用第 3 步和第 4 步中的 wallPos 和 wallBlocks 变量创建墙的模型：

```
wallShape = minecraftstuff.MinecraftShape(mc, wallPos,
                                          wallBlocks)
```

6. 墙已经完成——但是等等，它仍然是静态的！为了使墙上下移动，你需要使用 MinecraftShape 的 moveBy() 函数，并在相邻两次调用之间加上很小的时间延迟：

```
while not gameOver:
  wallShape.moveBy(0,1,0)
  time.sleep(1)
  wallShape.moveBy(0,-1,0)
  time.sleep(1)
```

while 循环中的代码会一直运行直到 gameOver 变量被设置为 True（或者说 not gameOver）。当玩家顺利通过游戏或失败从而游戏结束时，这个变量会被设置为 True。

7. `theWall()` 函数已经完成，剩下的就是在主程序中调用它。将如下代码添加到程序底部：

```
WALLZ = 10
theWall(arenaPos, WALLZ)
```

常量 `WALLZ` 存储了墙在场景中的 *Z* 位置。

8. 运行程序，你将会看到墙在场景中央不停地上下移动。

你还没有将 `gameOver` 变量值设为 `True`，所以当你运行程序后它不会停止。你需要单击 Python Shell 里的 Shell⇨Restart Shell 来停止程序运行。

深入代码

当你为墙创建方块时，你用到了两个 `for` 循环——其中一个循环完全位于另外一个循环体内部。这在编程里是一项很有用的技术嵌套，你也可以称之为“嵌套循环”。其中第一个 `for` 循环 `(for x in range(0,ARENAX + 1):)` 对于沿着游戏场景的宽度（*x*）方向每个方块都执行一次。第二个循环 `(for y in range(1,ARENAY + 1):)` 同样对于从 1 到高度（*y*）的每个方块都执行一次。

变量 x 和 y 被用来创建一个 `ShapeBlock`，对于墙体中的每个方块都调用 `minecraftstuff.ShapeBlock(x,y,0,block.BRICK_BLOCK.id)`。

当你使用嵌套 `for` 循环和常量 `ARENAX` 和 `ARENAY` 来编写程序时，这意味着即使游戏场景采用了一个不同的尺寸，墙体也总是会符合场景大小。

启用更多的障碍物

现在你有了一个问题！你写的程序好似卡住了一般，它会一直循环下去，不停地将墙上下移动并且不能干其他的事情。这是因为你写的程序是顺序的（sequential），每个程序中的命令都是依次执行，下一个命令会等待上一个命令执行完毕才会执行。你的程序卡住了是因为它永远不能执行到 `while` 循环后面的命令。

顺序的（sequential）是指按照一定顺序一条条地执行指令。

如果程序只是卡在那里不停地上下移动墙，你怎么才能够创建更多的障碍物并且执行剩余的游戏代码呢？

有一个解决方案，就是利用多线程（multi-threading）！到现在为止你写的所有的程序都是单线程的，换言之，它们都是顺序的，命令一个接一个地运行。

如果你将你的程序想象为一段绳子而你的命令是绳上的结，你会发现你的程序从绳的一端运行到另一端的过程中，每经过一个结相当于执行了一个命令。如果你加入一个循环，你的程序将会回到之前的“结”运行，如果你加入一个 `if` 语句，它可能会让程序跳过一个“结”——但是它仍然只能在同一时间执行一个命令。

如果你使用多线程，实际上你是在让你的程序创建了一个并列的、拥有自己的命令（结）的新绳子（或一段绳子）。它会在你原来程序运行的过程中同时运行。你的程序现在可以同时做两件事而非仅仅一件（见图 9-6）。

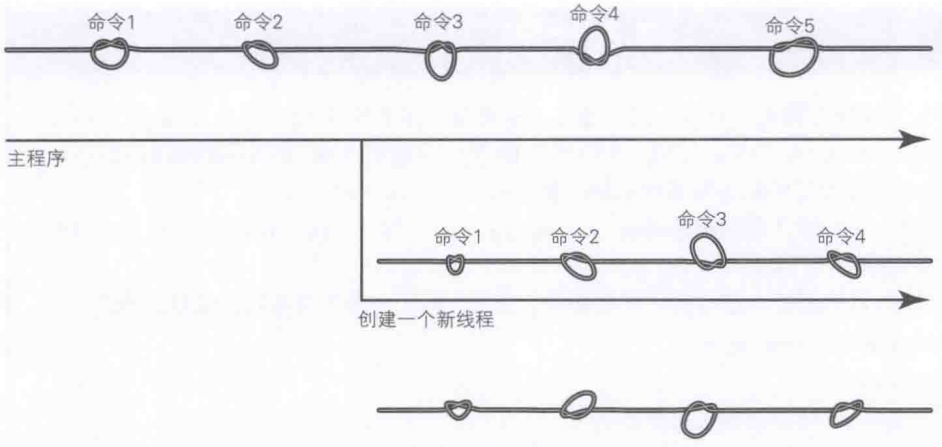


图 9-6　当你创建了多线程，你的程序可以同时做多件事

为了让游戏中所有的障碍物同时起作用，你需要在每个障碍物开始起作用时为它创建一个新线程，这意味着负责每个障碍物的程序都在同时运行着。

多线程在计算机编程中非常难以置信地有用，但是它非常高级而且很容易使写出的代码变得复杂。访问 www.tutorialspoint.com/python/python_multithreading.htm 了解更多关于 Python 多线程编程。

现在回到墙的问题。为了让它在自己的线程里运行，你必须改写调用 `theWall()` 函数的那行程序，并且使用 `thread.start_new_thread()`：

1. 删除程序里调用 `theWall()` 函数的最后一行，如下：

```
theWall(arenaPos, WALLZ)
```

2. 将下面这行代码加入到程序的最后一行来调用 `theWall()` 函数——但是这里会开启一个新线程去调用它：

```
thread.start_new_thread(theWall, (arenaPos, WALLZ))
```

3. 运行程序。

现在你应该能看到程序运行后墙会像以前一样上下运动。唯一不同的是这回它是在自己的线程中运行，因此你可以继续开发游戏中剩余的代码。

如果你想在 IDLE 中停止多线程程序的运行，在 Python Shell 里单击 Shell⇨Restart Shell。如果你在 Python Shell 里按下 Ctrl+C，这只是停止了主线程，并没有停止其他所有的线程，因此你仍然需要通过单击 Shell⇨Restart Shell 来终止主线程和所有其他线程的运行。

深入代码

为了在单独一个线程里调用 `theWall()` 函数，你使用了如下代码 `thread.start_new_thread(theWall, (arena, WALLZ))`。其中传入的第一个参数是需要调用的函数名——`theWall`，第二个参数包含了传给需要调用函数的参数——`arena,WALLZ`。

这些变量（参数）是以放入圆括号的形式 `(arena,WALLZ)` 传入，因为 start_new_thread 函数的第二个参数需要是一个元组 (tuple)。

任何一个 Python 函数都能以这样的形式被调用。例如你写了一个向屏幕输出信息的函数：

```
def printMessage(message)
  print message
```

你可以用如下代码在一个新线程里运行它：

```
thread.start_new_thread(printMessage,
  ("Hello Minecraft World",))
```

`"Hello Minecraft World"` 后面的逗号表明它在 Python 里是一个元组，但是其中只有一个元素。

元组（tuple）和你在之前的冒险中用到的列表很类似——其中关键的不同是一旦元组被创建，你就不能再去修改它，不像列表，你可以向其中添加或删除元素。用编程的术语来说，元组是不可变的（immutable），列表是可变的（mutable）。访问 www.tutorialspoint.com/python/python_tuples.htm 了解更多关于 Python 元组的知识。

创建河流

你的下一个目标就是在场景中沿着宽度方向创建一条河流。玩家并不能直接跳过这条河流，幸运的是河上会有一座桥。然而这座桥会沿着河流来回移动，因此玩家需要看准了跳上这座桥再跳到对岸。

如果玩家掉入了河流，他会被强制回到起点处重来。你将会在第三部分写代码实现让玩家回到起点的功能。现在这个部分只是创建河流并移动桥。

图 9-7　玩家必须越过河流

首先你必须清理掉一部分地面并且放上一层水方块来创建河流。你通过 `MinecraftShape` 来创建小桥，并控制它在两边不停地来回移动，正如你用同样的方法控制墙上下移动一样。

通过如下步骤重写 `theRiver()` 函数来创建河流：

1. 在之前的程序中找到如下 `theRiver()` 函数代码：

```
def theRiver(arenaPos, riverZPos):
  pass
```

并删除函数体中缩进的 `pass` 语句。

2. 在 `deftheRiver(arenaPos, riverZPos):` 行下缩进并建立和 Minecraft 的连接：

```
  mc = minecraft.Minecraft.create()
```

3. 创建两个常量，即河的宽度（`RIVERWIDTH`）和桥的宽度（`BRIDGEWIDTH`）：

```
  RIVERWIDTH = 4
  BRIDGEWIDTH = 2
```

4. 现在利用传入的参数 `arenaPos` 和 `riverZPos`（河流在场景中的 Z 方向坐标偏移）来创建河流：

```
  mc.setBlocks(arenaPos.x,
               arenaPos.y - 2,
               arenaPos.z + riverZPos,
               arenaPos.x + ARENAX,
               arenaPos.y,
               arenaPos.z + riverZPos + RIVERWIDTH - 1,
               block.AIR.id)
```

```
mc.setBlocks(arenaPos.x,
             arenaPos.y - 2,
             arenaPos.z + riverZPos,
             arenaPos.x + ARENAX,
             arenaPos.y - 2,
             arenaPos.z + riverZPos + RIVERWIDTH - 1,
             block.WATER.id)
```

你利用 `setBlocks()` 函数在场景中的地面上创建一片空气方块的区域来清除地面并在其中创造一层水方块来表示河流。

5. 创建桥的位置。它将被放置在河流中央，这样玩家就不能直接从岸上走上去而必须要跳跃：

```
bridgePos = minecraft.Vec3(arenaPos.x, arenaPos.y,
                           arenaPos.z + riverZPos + 1)
```

6. 创建桥的方块列表。这一步和墙类似，利用两个嵌套的 for 循环，一个用于沿着桥的宽度方向（x），另一个沿着河的宽度方向（z）：

```
bridgeBlocks = []
for x in range(0, BRIDGEWIDTH):
  for z in range(0, RIVERWIDTH - 2):
    bridgeBlocks.append(minecraftstuff.ShapeBlock(x,
                                                  0,
                                                  z,
                                block.WOOD_PLANKS.id))
```

在创建桥的时候，从河流宽度 `RIVERWIDTH` 中减去 2，这样在两岸和桥间都形成了一个空隙，玩家必须跳上去或者跳离（见图 9-8）。

图 9-8　由于桥并不是完全连接河流两岸，玩家必须跳跃

7. 利用第 5 步和第 6 步创建的 `bridgePos` 和 `bridgeBlocks` 变量来创建桥的模型：

```
  bridgeShape = minecraftstuff.MinecraftShape(
    mc, bridgePos, bridgeBlocks)
```

8. 为了让桥在河中一次移动一格，首先你要计算桥从一端运动到另一端需要移动的次数，这只需把场景宽度减去桥的宽度：

```
  steps = ARENAX - BRIDGEWIDTH
```

9. 利用两个 `for` 循环来让桥左右移动（一个循环让它向左，另一个向右）。每次移动时利用 `MinecraftShape.moveBy()` 函数来让桥移动一个方块的距离，并在相邻移动之间加上一个小的时间延迟：

```
  while not gameOver:
    for left in range(0, steps):
      bridgeShape.moveBy(1,0,0)
      time.sleep(1)
    for right in range(0, steps):
      bridgeShape.moveBy(-1,0,0)
      time.sleep(1)
```

10. 创建河流的函数现在完成了，现在需要在主程序最后加入如下代码来调用它：

```
RIVERZ = 4
thread.start_new_thread(theRiver, (arenaPos, RIVERZ))
```

11. 是时候来运行程序了！现在你首先应该能看到一条河流，当中有个小桥来回移动，然后是场景中央一堵上下移动的墙。

进入游戏场景并尝试通过小桥。如果你擅于控制玩家的话你会发现要通过河流其实相当容易——但是等等，这一切都要等到你在运动的同时有收集钻石和尽快完成的压力下再说！到时候你很有可能会不停地错过跳上小桥的机会。

创建陷坑

你最后需要在 Crafty Crossing 游戏里创建的障碍物就是陷坑了。这些陷坑在游戏场景中随机生成，持续几秒钟后消失并又随机在其他地方生成（见图 9-9）。

你需要利用 `randint()` 函数来随机为陷坑生成位置。黑色羊毛方块会在随机生成的陷坑打开前先出现，用以给玩家警告，让他们有机会能躲避开。创建陷坑的方法是在场景地面上创建空气方块代替草方块。它将持续几秒时间再变回草方块，然后在别处随机生成新的陷坑。

和河流一样，当玩家不小心掉入陷坑里时，会被送回起点重来，但是这部分代码将在本冒险中第三部分介绍。

现在你需要重写 `theHoles` 函数来创建陷坑：

1. 在之前的程序中找到如下 `theHoles()` 函数代码：

```
def theHoles(arenaPos, holesZPos):
  pass
```

图 9-9　陷坑随机出现，玩家需要要小心不要掉入其中

并删除函数体中缩进的 `pass` 语句。

2. 在 `deftheHoles(arenaPos,holesZPos):` 行下缩进并建立和 Minecraft 的连接：

```
    mc = minecraft.Minecraft.create()
```

3. 创建两个常量，其中一个是将要创建的陷坑数目（`HOLES`），另一个是陷坑生成区域的宽度（见图 9-10）：

```
    HOLES = 15
    HOLESWIDTH = 3
```

4. 加入一个 `while` 循环，一直持续到游戏结束：

```
    while not gameOver:
```

`theHoles` 函数里剩余的代码都会缩进在这个 `while` 循环的循环体内。

图 9-10　创建陷坑生成区域的位置和宽度

5. 利用 `random.randint()` 函数来获得这些陷坑的 x 和 z 坐标位置（y 坐标位置为游戏场景的位置），并把它们加入一个 Python 列表：

```
holes = []
for count in range(0,HOLES):
  x = random.randint(arenaPos.x,
                     arenaPos.x + ARENAX)
  z = random.randint(arenaPos.z + holesZPos,
        arenaPos.z + holesZPos + HOLESWIDTH)
  holes.append(minecraft.Vec3(x, arenaPos.y, z))
```

6. 依次遍历 holes 列表里每个位置坐标，并在相应位置上创建黑色羊毛方块：

```
for hole in holes:
  mc.setBlock(hole.x, hole.y, hole.z,
              block.WOOL.id, 15)
time.sleep(0.25)
```

通过将即将出现陷坑的位置设置为黑色，玩家会获得一定的反应时间，从而有可能避过这些陷坑。

7. 利用 `setBlocks()` 函数将陷坑出现的位置的方块设置为空气方块：

```
for hole in holes:
  mc.setBlocks(hole.x, hole.y, hole.z,
               hole.x, hole.y - 2, hole.z,
               block.AIR.id)
time.sleep(2)
```

每当创建了一些陷坑，紧接着加入一个 2 秒的时间延迟。

8. 利用同样的循环关闭这些陷坑，只不过这一次将方块设置回草方块：

```
for hole in holes:
  mc.setBlocks(hole.x, hole.y, hole.z,
               hole.x, hole.y - 2, hole.z,
               block.GRASS.id)
time.sleep(2)
```

现在你的程序将回返回 `while` 循环开始的地方，重新开始新一轮的陷坑的生成。

9. 你的 `holes` 函数已经完成，接下来将如下代码加到主程序的底部：

```
HOLESZ = 15
thread.start_new_thread(theHoles, (arenaPos, HOLESZ))
```

10. 运行程序。现在你不仅能看到游戏场景里的墙和河流障碍物，还会在相应的位置看到陷坑会不停地随机生成并消失。

试着玩玩，看看你能否在起点和终点之间来回顺利通过，而不会被障碍物阻挡或掉入其中。

你可以按照自己的意愿调整游戏中的一些常量的值。比如将游戏场景变得更长更宽，将障碍物放在不同的位置，或者通过让障碍物运动得更加快速来增加通过它们的难度。

深入代码

这些生成障碍物的函数好像一个个小程序一样，并且由于多线程，它们也是互不干扰地运行着。因此如果你想要更多种类的障碍物，去创造它们是非常容易的。比如说你想要两堵墙，如果你再次调用 `theWall` 函数，并且使用了一个不同的 z 位置，那么就会有一个相同的第二堵墙在场景中生成，不停地上下移动：

```
WALL2Z = 13
thread.start_new_thread(theWall,(arenaPos,WALL2Z))
```

挑战

你仍然可以走得更远。利用创建墙、河流和陷坑的方法，你能否创建一种全新类型的障碍物呢？例如你可以随机生成一些笼子来困住玩家，或者一系列玩家必须跳过的平台。

如果你认为这些障碍物过于容易或困难的话，你可以通过设置常量为不同的值来改变它们的难度。例如，你可以增加或减少时间延迟来让障碍物运动得更慢或更块。比如说为了加快桥的移动，你可以在让桥左右运动的两个 `for` 循环中将 `time.sleep(1)` 改为 `time.sleep(0.5)`。

第三部分游戏逻辑

你的这一部分大冒险中是为游戏加上逻辑。你的目标是将这个有着障碍物的游戏场景真正地转变成一个游戏，这样玩家就愿意不停地玩下去，到达后面的关卡。

为了完成这一目标，这个游戏需要具有挑战性并让人觉得刺激，以及奖励和目标。

游戏的挑战将会是让玩家收集在游戏场景里随机形成的钻石方块，同时在一个设定的时间内尽快通过各个障碍物。作为奖励，依据收集钻石的数量和到达终点的时间，玩家会获得一定的分数。玩家越快到达终点，获得的分数也会越多。游戏的目标就是完成所有的关卡并尽量获得更多的分数。

游戏将会有三关，每一关都比上一关更难，会设置更多的钻石和更少的限制时间。

开始游戏

在这一小节中，你将设置整个游戏，创建两个常量用来表示每关中的钻石数量和时间限制。

游戏程序有两个主循环：

- 游戏（game）循环：这个循环会一直运行直到游戏结束（while not gameOver）。这是所有关卡开始和设置的地方。在每关结束时会计算得分。
- 关卡（level）循环：这个循环会一直运行直到一关结束——因此，或者是一关被完成了或整个游戏结束（while not gameOver and not levelComplete）。如果掉入了河流或陷坑，这个循环同样会将玩家设置到起点，当玩家经过钻石时会清除它们，并且一直检查时间是否用尽。

这一部分冒险中提到的代码均是在游戏（game）循环中或关卡（level）循环中。将代码放入正确的循环体非常重要，否则游戏可能不能正常运行。

首先，按照如下步骤在主程序底部创建游戏的主要框架，即两个主要的循环：

1. 创建三个常量，分别为游戏中的关卡数、将要创造的钻石数量，以及玩家完成关卡的时间限制（单位：秒）：

```
LEVELS = 3
DIAMONDS = [3,5,9]
TIMEOUTS = [30,25,20]
```

DIAMONDS 和 TIMEOUTS 常量均为 Python 列表，它们都有三个元素，分别对应三个关卡；例如在第一关，玩家要收集 3 个钻石，并且需要在 30 秒内完成并到达终点。

2. 创建两个变量，分别用来存储玩家已经获得的分数和当前所在的关卡：

```
level = 0
points = 0
```

3. 创建游戏循环。在它前面添加注释用以提醒你它的位置，因为接下来很多代码需要放置在缩进的循环体中：

```
#game loop
while not gameOver:
```

gameOver 变量在程序开始时被设置为 False；当玩家完成游戏或时间结束时它会被设置为 True。这个变量也被用在和障碍物有关的函数里，当它为 True 时，障碍物就会停止运作。

4. 在游戏循环下缩进，将玩家的位置设置到起点：

```
    mc.player.setPos(arenaPos.x + 1, arenaPos.y + 1,
                     arenaPos.z + 1)
```

5. 获取当前时间并存到一个变量里，用以开始给当前关卡计时：

```
    start = time.time()
```

6. 将关卡完成的标志设置成 False 并创建关卡循环。和游戏循环类似，在这里也添加注释用以标明关卡循环开始的位置，因为后续的代码会添加到缩进的循环体里：

```
    levelComplete = False
    #level loop
    while not gameOver and not levelComplete:
```

7. 在关卡循环下缩进，并添加一个很小的时间延迟。因为游戏运行时，程序一直在执行这里的循环代码，如果不加上这一句，程序将会占用计算机处理器全部资源：

```
    time.sleep(0.1)
```

8. 运行程序。和之前唯一的不同是你会发现玩家被自动放置在了起点，你可以借此来检查是否新加入的所有代码都能正常运行，没有错误发生。

挑战

当玩家被放置在游戏场景的起点时，他实际上是被放在了场景的右边角上。如果把起点设置在中间会不会更好呢？重新修改代码，使得在每关的开始，玩家都是被放在了中间而不是右边角落上。

收集钻石

这个游戏的主要目标是收集钻石。现在你将要写程序，在游戏场景里随机生成这些钻石（见图 9-11）；玩家通过“击打”它们（或者说，拿着剑右键点击）来收集钻石。

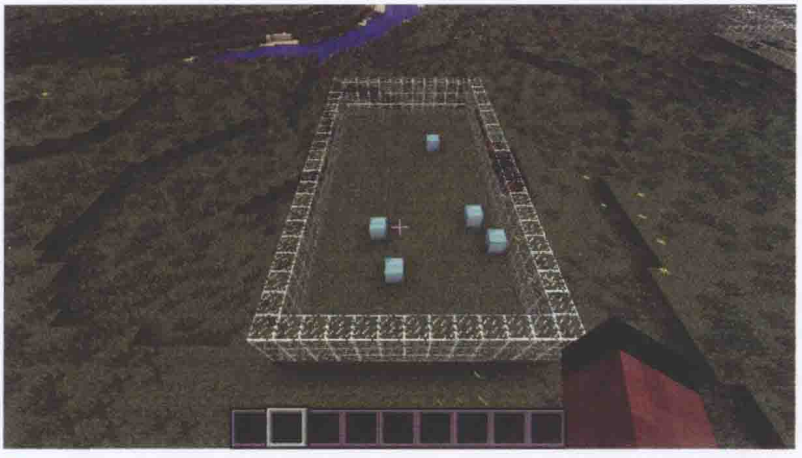


图 9-11　钻石在场景里随机生成

当钻石被收集时，它们会立即消失，并且一旦玩家收集了所有的钻石，他就可以到达终点来结束这一关。按照如下步骤重写 `createDiamonds()` 函数并在游戏循环里调用它：

1. 在之前的程序中找到如下 `createDiamonds()` 函数代码：

```
def createDiamonds(arenaPos, number):
  pass
```

并删除函数体中缩进的 `pass` 语句。

2. 在 `def createDiamonds(arenaPos, number):` 行下缩进并建立和 Minecraft 的连接：

```
mc = minecraft.Minecraft.create()
```

3. 在场景中随机生成 x 和 z 坐标位置并设置该处的方块为钻石（DIAMOND_BLOCK），创建指定数目的钻石：

```
for diamond in range(0, number):
  x = random.randint(arenaPos.x, arenaPos.x + ARENAX)
  z = random.randint(arenaPos.z, arenaPos.z + ARENAZ)
  mc.setBlock(x, arenaPos.y + 1, z,
              block.DIAMOND_BLOCK.id)
```

4. 在游戏循环开始，我们需要调用 createDiamonds() 函数。每当一个新关卡开始时都要创建一系列相应数目的钻石。在游戏循环下缩进，即如下的 while 循环，并加入如下代码：

```
# 游戏循环
while not gameOver:
  createDiamonds(arenaPos, DIAMONDS[level])
  diamondsLeft = DIAMONDS[level]
```

diamondsLeft 变量在此处被创建，用来存储仍需收集的钻石数量。

5. 运行程序。因为是第一关，你会在游戏场景里看见三个随机生成的钻石。

当你创建完钻石后，你可以利用 pollBlockHits 函数（在冒险 4 中学过）加入如下代码去监测玩家的击打事件，并且当玩家击打一个钻石方块的时候，将它变为空气。

向关卡循环里加入如下代码用以将钻石（DIAMOND_BLOCK）变为空气，当玩家击打它时：

1. 在关卡循环下缩进，调用 pollBlockHits 函数来获取方块击打事件：

```
# 关卡循环
while not gameOver and not levelComplete:
  hits = mc.events.pollBlockHits()
```

2. 遍历方块击打事件，获取被击打的方块的种类：

```
  for hit in hits:
    blockHitType = mc.getBlock(hit.pos.x, hit.pos.y,
                               hit.pos.z)
```

3. 检查方块种类是否为钻石（DIAMOND_BLOCK），如果是，将它变为空气方块并且从 diamondsLeft 变量里减去 1：

```
    if blockHitType == block.DIAMOND_BLOCK.id:
      mc.setBlock(hit.pos.x,hit.pos.y, hit.pos.z,
                  block.AIR.id)
      diamondsLeft = diamondsLeft - 1
```

当玩家到达终点时，diamondsLeft 变量会被用来判定是否所有的钻石已经收集完毕。

4. 运行程序。当你击打钻石方块时，它们会变成空气方块从而消失。

> 需要记住的是，你必须持剑并单击鼠标右键来击打方块。
>
> 
>

挑战

你能让钻石更难被收集吗？例如你可以让钻石不停地上下移动，并且玩家只能当钻石在空中时才能收集它们。

超时

如果玩家拥有无限的时间，那么这个游戏会变得很简单（并且无聊）。设定一个时间限制，你可以让你的游戏变得更有挑战性，并且通过让玩家在规定的时间结束之前收集所有的钻石并到达终点，他们也获得了一个明确的游戏目标。

如果时间不够，那么游戏也就结束了。你接下来的任务是在关卡循环里加入代码来检测时间是否结束，如果结束了，那么设置 `gameOver` 变量值为 `True`：

1. 在关卡循环下缩进，编写程序计算当前关卡剩余时间秒数：

```
# 关卡循环
while not gameOver and not levelComplete:

  secondsLeft = TIMEOUTS[level] - (time.time() - start)
```

2. 如果剩余时间小于 0 秒，设置 `gameOver` 变量为 `True` 并且向聊天窗口输出消息：

```
  if secondsLeft < 0:
    gameOver = True
    mc.postToChat("Out of time...")
```

3. 运行程序。过了 30 秒之后（因为是在第一关），程序会终止并输出消息“Out of time…”（见图 9-12）。

深入代码

我们是利用如下代码计算剩余时间的秒数的：

```
secondsLeft = TIMEOUTS[level] - (time.time() - start)
```

设置 `seondsLeft` 变量的方法是首先从 `TIMEOUTS` 常量数组里取得当前关卡的时间限制：

```
TIMEOUTS[level]
```

然后减去当前关卡已经消耗的时间。这个时间是由当前时间减去关卡开始时的时间：

```
(time.time() - start)
```

图 9-12　时间到时通知玩家

追踪玩家位置

当玩家收集了所有的钻石并到达终点的时候，他就完成了当前关卡。当他掉入河流或陷坑的时候，他会被强行返回游戏起点。为了实现这些功能，你的程序需要知道当前玩家的位置。

检测玩家位置之后，程序就可以根据玩家收集钻石的情况或者设置 `levelComplete` 为 `True`，或者将玩家送回起点。

你的下一个任务是在关卡循环中加入追踪玩家位置的代码，并且将玩家设置回起点或者完成当前关：

1. 在关卡循环中缩进，并加入获得玩家位置的代码：

```
# 关卡循环
while not gameOver and not levelComplete:

  pos = mc.player.getTilePos()
```

2. 检测玩家的高度，*y* 是否低于游戏场景的高度。若是，则玩家一定是掉入了河流和陷坑，因此要把他设置回起点：

```
  if pos.y < arenaPos.y:
    mc.player.setPos(arenaPos.x + 1, arenaPos.y + 1,
                     arenaPos.z + 1)
```

3. 检测玩家是否到达终点并且收集到了所有的钻石。这可以通过查看玩家的 *z* 坐标是否和场景终点的 *z* 坐标一致，若是，将 `levelComplete` 设置为 `True`：

```
  if pos.z == arenaPos.z + ARENAZ and diamondsLeft == 0:
    levelComplete = True
```

当 `levelComplete` 为 `True` 时，关卡循环终止，钻石会重新生成并且游戏重新开始。

4. 运行程序。每当玩家收集齐所有的钻石并到达终点后游戏会重新开始；并且如果玩家落入河流或陷坑，他会被送回起点。

此时游戏只能玩一关——并且是最简单的第一关。也许现在你来练习一下这个游戏是个不错的主意，因为在下一部分游戏的难度将会急剧增加！

设置关卡完成和计算得分

当玩家通过一个关卡时，我们的程序需要计算玩家得分并进入下一关。每当玩家通关时，他会相应得分，其中每个钻石计入一分，并且在到达终点时将其乘上剩余时间的秒数。

你需要添加如下的代码来实现它。这些代码需要添加在游戏循环下缩进的循环体中，但是必须放在关卡循环结束之后：

1. 在游戏循环体中关卡循环之后，检查是否是因为关卡完成而退出关卡循环：

```
# 游戏循环
while not gameOver:
  [code]

  # 关卡循环
  while not gameOver and not levelComplete:
    [code]

  if levelComplete:
```

2. 如果关卡完成了，计算得分并加入到 `points` 变量里，然后将得分显示在聊天窗口：

```
    points = points + (DIAMONDS[level] * int(secondsLeft))
    mc.postToChat("Level Complete - Points = " + ↩
                  str(points))
```

3. 将变量 `level` 的值加 1 以进入下一关：

```
    level = level + 1
```

4. 如果这已经是最后一关，将 `gameOver` 变量设为 `True` 并且在聊天窗口显示祝贺的信息：

```
    if level == LEVELS:
      gameOver = True
      mc.postToChat("Congratulations - All levels ↩
                    complete")
```

常量 `LEVELS` 的值为游戏中关卡总数。

5. 运行程序。

你的游戏已经接近完工了！当玩家在时间结束前收集了所有的钻石并到达场景终点时，游戏会自动进

入下一关。每一关都会较之前有更多的钻石和更少的时间限制，因而难度加大。如果玩家成功地通过了所有的关卡，那么游戏结束并显示祝贺信息！

挑战

尝试着添加更多的关卡。你可以让游戏一开始的钻石数目更少，时间限制更多，因而变得更容易，然后逐渐加大难度，让最后几关特别难。

添加游戏结束信息

你需要完成这个游戏的最后一件事就是在游戏结束时添加一条信息，来告诉玩家游戏已经结束，并将最终得分显示出来（见图 9-13）。只需在程序底部简单添加如下代码：

```
mc.postToChat("Game Over - Points = " + str(points))
```

图 9-13　游戏结束时告诉玩家并显示得分

作者提醒

游戏通关——156 分。超过他！

挑战

将每次得分保存到计算机文件中，创建一个排行榜——在每次游戏结束时显示玩家排名。

你可以在 www.wiley.com/go/adventuresinminecraft 网站下载完整的 Craft Crossing 游戏。

游戏完全可以继续下去！这个游戏程序的代码被设计得很容易让你进行继续扩展，例如设置其中的常量值，引入新型的障碍物或宏大的游戏场景。这些完全都是你自己来决定。

第四部分添加按钮和显示

关于 Craft Crossing 游戏仍然有很多的问题。无论玩家有没有准备好，这个游戏都是自动开始的，并且没有任何显示器来显示有用的信息，例如还有多少钻石没有收集，时间是否快要结束了。

在这个冒险的最后一部分，你将重新使用在冒险 5 中创建引爆器用到的硬件来向游戏中添加一个开始游戏的按钮。你也将会用到七段数码管来指示剩余需要收集钻石的数量。当时间只剩下 5 秒时，小数点将会变亮。

你将需要什么

你将需要在冒险 5 中用到的同样的电子元件，当你制造一个引爆器时，用同样的方法在最终练习中连接它们。你应该在一块面包板上将一个七段数码管和一个按钮连接到树莓派（见图 9-14）或者一个 Arduino（见图 9-15）。如果需要的话，回到冒险 5 查看如何连接。

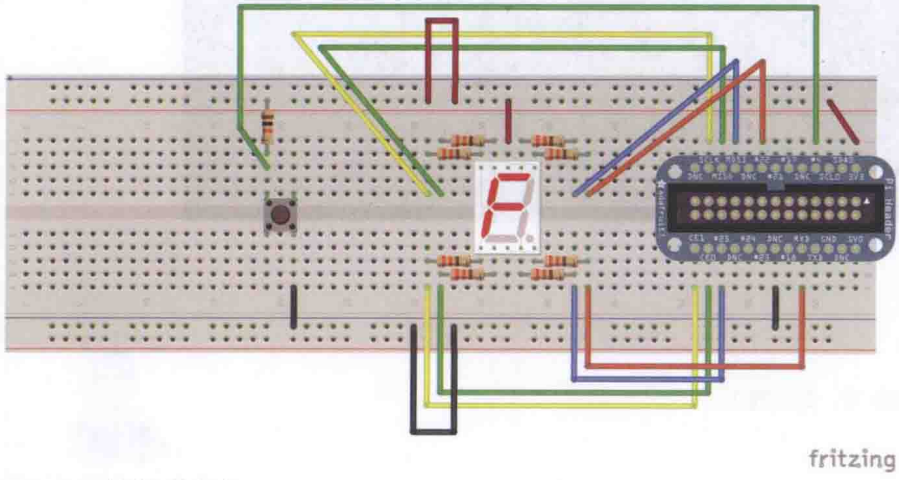


图 9-14　树莓派的电路

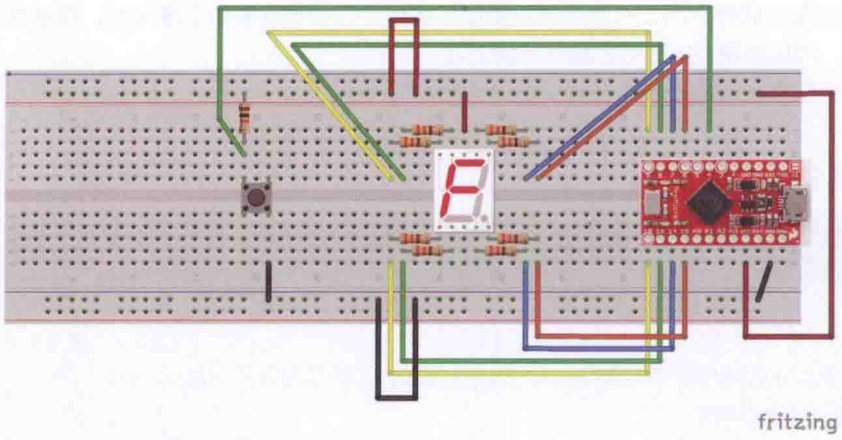


图 9-15　Arduino 的电路

设置硬件

为了添加一个按钮和七段数码管，第一步是在 Python 里导入正确的模块，设置 GPIO 以及为引脚创建常量。

修改你的 Crafty Crossing 游戏程序，设置硬件并等待玩家按下按钮才开始游戏：

1. 在程序里已有的导入语句下继续代码导入七段数码管模块：

```
import anyio.seg7 as display
```

2. 为你的硬件导入 GPIO 模块并为将要用到的引脚建立常量：

在树莓派上：

```
import RPi.GPIO as GPIO
BUTTON = 4
LED_PINS = [10,22,25,8,7,9,11,15]
```

在 Arduino 上：

```
import anyio.GPIO as GPIO
BUTTON = 4
LED_PINS = [7,6,14,16,10,8,9,15]
```

3. 设置硬件：

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(BUTTON, GPIO.IN)
ON = False # 共阳极，                True 为共阴极，
display.setup(GPIO, LED_PINS, ON)
```

如果你使用的是共阴极七段数码管（例如冒险 5 中提到的），设置 `ON = True`。

4. 在游戏循环之前的位置，加上如下代码等待玩家按下按钮：

```
mc.postToChat("Press the button to start")
while GPIO.input(BUTTON):
  time.sleep(0.1)
while not gameOver:
```

现在程序会在创建游戏场景和障碍物之后，一直等待玩家按下按钮。只有当玩家按下按钮时，程序才会退出循环，并进入游戏循环，将玩家设置在起点位置并开始游戏。

5. 正确地结束程序和 GPIO 模块非常重要。在程序结尾添加如下代码：

```
GPIO.cleanup()
```

6. 运行程序并测试新添加的“开始”按钮。

挑战

你能修改你的程序使你能够在游戏结束时通过按下按钮来开始新的游戏，而不是要重新运行程序吗？

钻石计数器

七段数码管将被用来显示剩余需要收集钻石的数量。我们将会利用 `display.write` 函数来更新显示。

为了完成这一目标，你需要修改 Crafty Crossing 游戏程序使在每当场景中生成钻石或者玩家收集到一个钻石的时候，显示会被更新：

1. 在游戏循环中调用 `createDiamonds` 函数生成钻石之后，更新显示：

```
createDiamonds(arenaPos, DIAMONDS[level])
diamondsLeft = DIAMONDS[level]
display.write(str(diamondsLeft))
```

`display.write()` 方法需要一个字符串参数，因此在将剩余钻石的数量传给这个函数之前，利用 `str()` 函数将其转换为字符串。

2. 当玩家收集钻石并且剩余钻石数量减 1 之后，更新显示：

```
    hits = mc.events.pollBlockHits()
    for hit in hits:
      blockHitType = mc.getBlock(hit.pos.x, hit.pos.y,
                                 hit.pos.z)
      if blockHitType == block.DIAMOND_BLOCK.id:
        mc.setBlock(hit.pos.x,hit.pos.y, hit.pos.z,
                    block.AIR.id)
        diamondsLeft = diamondsLeft - 1
        display.write(str(diamondsLeft))
```

3. 在程序最后，你需要清除显示。因此在清理 GPIO 之前，加上如下代码：

```
display.clear()
GPIO.cleanup()
```

4. 运行程序。剩余钻石的数量将会显示出来，并且每当玩家收集一个，它都会减 1 直到 0。

七段数码管只能显示数字 0 ~ 9。如果你修改程序，让某一关有多于 9 个钻石，你应该在程序里添加一个 `if` 语句，只有在钻石数量等于或少于 9 时才更新它的显示。

剩余时间指示器

你在这个冒险里最后一个任务是用显示器指示玩家时间快要结束了。你通过显示小数点来提醒玩家。当时间不足 5 秒时，显示小数点，从而提醒玩家尽快完成当前关卡。

修改 Crafty Crossing 游戏程序，在最后 5 秒显示 LED 上的小数点：

1. 在关卡循环中计算剩余时间的语句之后，检测是否时间已经少于 5 秒。如果是，则显示小数点，否则关掉：

```
secondsLeft = TIMEOUTS[level] - (time.time() - start)
if secondsLeft < 5:
  display.setdp(True)
else:
  display.setdp(False)
```

2. 运行程序。当时间少于 5 秒时小数点会显示。

挑战

在每关开始的时候，先利用七段数码管显示一下当前关卡数，再显示需要收集的钻石数。

快速参考表

命令	描述
`import thread`	导入 Python 的多线程模块
`thread.start_new_thread(function, (variable1,variable2))`	在单独一个新线程里调用函数
`import time`	导入 Python 的时间模块
`timeNow = time.time()`	获取当前时间

在你的 Minecraft 之旅中更多的冒险

Minecraft 给予了你一个奇幻的舞台去展示你的创造力并去冒险。拥有了利用代码控制游戏的能力之后，你的唯一限制就是你的想象力了。所以接下来你会做什么呢?

这里有一些可能给予你灵感的主意或资源:

- 制作游戏是提高你编程能力的一个重要的方法。访问 www.classicgamesarcade.com 了解更多。
- Minecraft 是一个多人游戏。为什么不写程序创造一个可以多人参与的游戏呢?
- 和电子设备交互将 Minecraft 游戏带入了真实的世界。访问 eu.wiley.com/WileyCDA/WileyTitle/productCd-1118948475.html 了解更多关于 Arduino 编程。
- 互联网上有很多开放数据库。可以将 Minecraft 和一些网站结合起来，例如 twitter（dev.twitter.com）或 Met Office 的天气广播（www.metoffi ce.gov.uk/datapoint）。

解锁成就：你的 Minecraft 大项目!

附录 A
接下来去哪

我们希望书中的冒险带给你了许多的设想、代码以及技能，并激励着你进行更深层次的 Minecraft 编程冒险。之后的路，将由你自己来走！如果你还不确定现在该做什么，或是已经有了想法但无从下手，下面为你提供了一些有意思的资源以供浏览。

网站

互联网上有一大批关于 Minecraft 的有用网站，多到甚至无法整理。但这里是一些 Martin 和 David 认为对在 Minecraft 中学习、游玩以及对编程很有用的网站。

Minecraft

www.wiley.com/go/adventuresinminecraft

这是本书的配套资源网站，包括一个完整的附加冒险（Minecraft 电梯），可供下载的参考表、徽章、完整的程序列表，以及书中的每个工程的视频。

www.stuffaboutcode.com

这是很受欢迎的 Martin 的博客，其中一大部分是大量的工程和实验，可以自己尝试去实现。Martin 也编写了最好的 Minecraft API（应用程序编程接口——译者注）参考，网上随处可见。

http://arghbox.wordpress.com

Craig Richardson 经常开发美妙的 Minecraft 编程工程来支持新式计算机课程。强烈建议阅读他的

Minecraft 编程书（开源），以及观看他用真实的水果做的 Minecraft 控制器！

www.minecraftforum.net

这是官方的 Minecraft 论坛，在此你可以针对任何相关问题寻求帮助，例如新特性、建立服务器或是创造模式和生存模式中的小技巧等。

http://minecraft.gamepedia.com/Minecraft_Wiki（http://minecraft-zh.gamepedia.com/ 中文 Minecraft Wiki——译者注）

Minecraft Wiki 是一个收集各种关于 Minecraft 信息的社区。作为社区性质的网站，你也可以作出一份贡献！

www.scarabcoder.com

一个年轻的编程者 Nicolas Harris 的博客。他起初在树莓派上进行 Minecraft 编程，并不断在博客上更新他在编程和技术上的冒险经历。强烈建议去看他的 Bukkit 插件集。

www.reddit.com/r/MCPI

Reddit 是一个社交性的链接网站。在这里人们贴上链接，并投票来决定它的上下次序。大量有趣的 Minecraft 编程工程经常出现在 Reddit 上。

http://mcpipy.wordpress.com 和 https://github.com/brooksc/mcpipy

这是一个 Minecraft 编程资源的合集。虽然你在别的地方也能看到这些资源，但是许多人是通过这个网站以及 Reddit 认识的 Martin O'Hanlon，其中包括 David。

www.minecraftmaps.com

Minecraft 允许玩家读取和存储冒险地图，而一幅地图即是一个 Minecraft 世界的缩影。这个网站上有许多社区制作的地图，下载后便可以在你的 Minecraft 中载入，然后基于这些现成的地图开始你的建造。

http://minecraft.curseforge.com

这是一个开源网站（相当于为 Minecraft 而做的 sourceforge）。工程被托管在这里，并可从 www.curse.com 下载。

http://minecraft-seeds.net

Minecraft 世界由一个程序内置的计算机算法生成。随机数由一个最开始的数字（“种子”）生成。如果你输入已知的种子，你会生成种子所一一对应的那个世界。在这个网站上你可以找到生成各种特定 Minecraft 世界的种子。

www.mcedit.net 和 www.worldpainter.net

这些是可以免费下载的工具。使用这些基于屏幕的编辑工具，你可以设计、描绘你的自定义世界。

http://mcreator.pylo.si

MCreator 是一个游戏模组编辑器。它使用简单的单击界面而不是程序代码。由此你可以制作奇妙的 Minecraft 游戏模组，包括新的方块、生物、盔甲、命令以及许多其他物品。

www.firetechcamp.com

Fire Tech Camp 在每个英国的学校假期为青少年组织编程夏令营。针对对技术感兴趣的青少年，他们开发了一些很好的教授游戏设计、Minecraft 建筑、Python 编程、Arduino 和其他技术的课程。

Python

https://www.python.org 和 https:docs.python.org/2

这些是 Python 编程语言的官方下载和资料网站。

www.codecademy.com/tracks/python

这是一个免费的网上课程，你可以跟随课程进度循序渐进地学习 Python 编程语言。

http://inventwithpython.com/chapters

这是一个免费的在线电子书，用许多不错的 Python 工程来教你如何用它编程。

https://docs.python.org/2/library/idle.html

这是集成开发环境 IDLE 的官方指导，写得非常详细但没有图片。我们更建议使用 Anne Dawson 博士的教程：www.annedawson.net/Python_Editor_IDLE.htm

www.geany.org

IDLE 只是一个小而简单的开发环境，在此之后你可能会想要更全的功能。可以试试 Geany——一个好得多的开发环境。

http://blog.whaleygeek.co.uk

这是 David 的博客，上面有大量的 Python 和树莓派的工程、提示和技巧，甚至还有一些用来记忆 Python 和 Minecraft 的重要语法的小卡片。打印这些卡片，以便在你忘记语法的时候可以看看它们。

Bukkit

http://wiki.bukkit.org/Plugin_Tutorial

书中所用的 Bukkit 服务端是社区开发的而且支持用 java 写的插件。你可以下载关于几乎任何东西的插件，甚至是自己编写一个！在这个网站上有更多相关信息。

http://dev.bukkit.org/bukkit-plugins/raspberryjuice

书中所用的 RaspberryJuice 其实也是一个插件，它把 Minecraft 内部的 API 通过使用树莓派的 MCPI Python 接口表示出来。

http://www.skpang.co.uk

S.K.Pang 先生是一个英国的优质供货商，营业范围主要是电子元件和有趣的工程。他也批发预组装并预编程的 Sparkfin Arduino Pro-Micro，以及一个预组装的 Adafruit T-Cobbler，其上为你标出了所有的 GPIO 引脚。可以考虑直接买这些预组装的产品来更快地开始第 5、9 和附加冒险。

http://www.sparkfun.com

Sparkfun 上有相当多的电子设备出售，还有一些不错的连接设备的教程。你在第 5、9 以及附加冒险里可能使用过的 Arduino Pro-Micro 就是 Sparkfun 设计的。但请注意许多产品并没有预焊以及预编程，你可能需要在使用前做一些处理。

http://www.maplin.co.uk

Maplin Electronics 是一个可靠的零售商。当你在突然有了想法的时候，你可以在它从你脑海里消失之前，就走进它的一家店去购买所需的电子元器件！

其他实现自动化的方法

用 Python 来编程 Minecraft 并不是唯一实现自动化的方法。Minecraft 拥有红石、命令方块和标签。你可以互相组合它们来让你的 Minecraft 世界的各个部分自动进行不同的工作。

http://minecraft.gamepedia.com/Tutorials/Command_Block

命令方块可以被触发以执行自动任务，你可以向它们传输红石信号来触发一条命令。由此你可以造出很多东西，例如教程页上的陷阱和传送器。

www.minecraftforum.net/topic/1969520-17-using-summon-give-datatags-in-map-making-tutorials

这个网站展示了如何用命令来使用二进制命名标签（NBT——许多 minecraft 数据类型的内部储存格式）。输入带有标签的命令，不用编程即可创造和设置各种不同的 Minecraft 对象，例如方块和实体。

www.minecraft101.net/redstone/redstone101.html

红石在 Minecraft 中相当于电。看似很简单，但利用这些简单的电路，你也可以造出极其复杂的装置。

工程和教程

有时实现自己的工程设想是一件很有乐趣的事情，但有时你也需要一点启发来加以实现。下面列举的网站拥有一些很好的范例工程，其中许多是由社区中的爱好者和开发者贡献的。

http://minecraft.gamepedia.com/Tutorials/Advanced_redstone_circuits

这个教程展示了如何使用红石，用一个个小电路来组成相当大的自动结构。跟随教程，你可以一步一步地用红石建造一台完整的计算机！

https://learn.adafruit.com/search?q=minecraft

Adafruit 拥有一些不错的教程，还附有其职员和社区其他人为这些教程写的指导。他们有一个新版块用来展示以电子设备连接的 Minecraft 工程，该版块还在不断扩充之中。

www.stuffaboutcode.com/2013/09/minecraft-ordnance-survey-map-rastrack.html

这是 Martin 和 David 一起做的第一个工程，那时我们甚至还没见过面！这是一个“混搭”（许多项目混在一起）工程。我们从 Ryan Walmsley 的 Rastrack 网站和 Ordnance Survey 的开放地图上获取数据，在英国地图上显示全国的树莓派设备！更多详见网站上的视频。

http://hackaday.com/2013/01/30/controlling-minecraft-with-a-raspberry-pi

使用 Bukkit 和红石是另一种用 Minecraft 控制现实中的硬件的方法。这是一个可以与树莓派通信的 Bukkit 插件，让 Minecraft 中的拉杆和木牌可以监控现实世界中的电子元器件。

www.instructables.com/howto/minecraft

The instuctable 网站有很多极好的教程来一步步地教你建造工程。他们现在有专门的 Minecraft 工程版块。

视频

和许多现代游戏一样，Minecraft 是视觉化的，获得启发和新想法的一大方法便是观察别人在做什么。许多视频来自于博主或者爱好者，而这些作者也因视频而闻名，许多人每周甚至每天都有更新。

http://minecraft.gamepedia.com/Tutorials/Elevators

在本书网站上的附加冒险里你可以编写一个可工作的 Minecraft 电梯。但除此之外还有很多在 Minecraft 中造电梯的方法，都可以在这个 Wiki 上找到。此外还有一些电梯工作的视频。

www.youtube.com/user/sethbling

Seth Bling 拥有一个大而活跃的 YouTube 频道，上面有大量的 Minecraft 工程和设想。强烈建议去看看他的 TNT 奥林匹克运动会！

www.youtube.com/user/stampylonghead

Stampy 每天至少在他的 YouTube 频道上发布一个新的 Minecraft 视频。他有许多易于建造的美观建筑。

www.youtube.com/user/scarabcoder

Nicholas Harris 拥有一个不错的 YouTube 频道，里面有许多有趣的工程。他同时也撰写了一本关于编写 Minecraft 程序的电子书，可以在亚马逊网站上找到。

www.youtube.com/user/SimplySarc

SimplySarc 详细介绍了“原版照相机”（www.youtube.com/watch?v=NyMHCabq_rs）。使用命令方块，它能够来在无 mod 的原版 Minecraft 中拍摄照片！

www.youtube.com/watch？v=7t4bH7Z-Yt4

我们的 Martin O'Hanlon 展示了一个 Python 程序，把你打到的任意方块变成一个数秒后爆炸的炸弹。

www.youtube.com/user/HiFolksImAdam

使用命令方块，Adam 创作了一些视觉错误，他还建造了一个 TARDIS（科幻电视剧 Doctor Who 中的时间机器和宇宙飞船——译者注），其内部空间要比从外面看上去大！可以在 www.youtube.com/watch？ v=3QpqUCaz8fk 这里看到。

www.youtube.com/user/AsdjkeAndBro

Asdjke and Bro 在 Minecraft 中建造了许多的小游戏。我们个人最爱是战舰游戏：www.youtube.com/watch?v=6AbPIT-cAm8

图书

拥有一本实体书籍摊开在手边，其上布满个性化的笔记纸条、铅笔画线以及折角，对你的 Minecraft 冒险来说是一件很美好的事情。Martin 和 David 两个人也都从其他作者学到了很多。与此同时，更多的工程和设想也会让你从本书得到的乐趣和收获更上一层楼。在此特别推荐 Becky Stuart 的《零基础学 Arduino（图文版）》，如果你正在使用 PC 或 Mac，并且喜欢在第 5、9 以及附加冒险中建造电路的话，这本书非常适合作为你的下一步。我们使用的 Arduino Pro Micro 几乎兼容该书中所有的工程，你也可以根据 Sparkfun 网站上的教程重新编程你的 Arduino Pro Micro，实现你想要的任何东西。

Minecraft for Dummies

作者：Jacob Corderio，Wiley 出版社，2013

Minecraft: The Official Beginners'Handbook

Egmont 出版社，2013

Minecraft: The Official Redstone Handbook

Egmont 出版社，2013

Learn to Program with Minecraft Plugin

作者：Andy Hunt，Pragmatic Bookshelf 出版社，2014

Adventures in Raspberry Pi（《零基础学 Raspberry Pi（图文版）》）
作者：Carrie Anne Philbin，Wiley 出版社，2013
Adventures in Arduino（《零基础学 Arduino（图文版）》）
作者：Becky Stewart，Wiley 出版社，2014
Adventures in Python（《零基础学 Python（图文版）》）
作者：Craig Richardson，Wiley 出版社，2014（中文版由人民邮电出版社于 2015 年出版）
Python for Kids
作者：Jason R. Briggs，No Starch 出版社，2012