



0 x = } 0 1 0 : ) 1 0

( y = 0 \* > 0 1 : 1 @ [ / 

1 @ 1  0 : < / 0 \* 0 1 # %

# COMPUTER CODING

**FOR KIDS**

0  1 : / y # }

x 0 @ ( 1 0  1

 1 }  0 \ = \* 0 :

A UNIQUE STEP-BY-STEP VISUAL GUIDE,  
FROM BINARY CODE TO BUILDING GAMES

CAROL VORDERMAN





# COMPUTER CODING

FOR KIDS







# COMPUTER CODING

FOR KIDS

A UNIQUE STEP-BY-STEP VISUAL GUIDE,  
FROM BINARY CODE TO BUILDING GAMES





LONDON, NEW YORK, MELBOURNE,  
MUNICH, AND DELHI

**DK LONDON**

**Editor** Sam Priddy  
**Designer** Fiona Macdonald  
**Additional editors** Sam Atkinson,  
Lizzie Davey, Daniel Mills, Ben Morgan  
**Additional designer** Simon Murrell  
**Managing editor** Paula Regan  
**Managing art editor** Owen Peyton Jones  
**Senior producer, pre-production** Ben Marcus  
**Senior producer** Mary Slater  
**Jacket editor** Maud Whatley  
**Jacket designer** Laura Brim  
**Jacket design development manager** Sophia MTT  
**Publisher** Sarah Larter  
**Art director** Phil Ormerod  
**Associate publishing director** Liz Wheeler  
**Publishing director** Jonathan Metcalf

**DK INDIA**

**Senior art editor** Devika Dwarkadas  
**Editors** Suefa Lee, Neha Pande  
**Art editors** Sanjay Chauhan,  
Shreya Anand Virmani  
**Assistant art editor** Vanya Mittal  
**DTP designer** Sachin Gupta  
**Managing editor** Rohan Sinha  
**Deputy managing art editor** Sudakshina Basu  
**Pre-production manager** Balwant Singh  
**Jacket designer** Suhita Dharamjit  
**Senior DTP designer** Harish Aggarwal

First published in Great Britain in 2014 by Dorling Kindersley Limited  
80 Strand, London WC2R 0RL  
A Penguin Random House Company  
Copyright © 2014 Dorling Kindersley Limited  
2 4 6 8 10 9 7 5 3 1  
001 – 192672 – Jun/2014

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior written permission of the copyright owner.

A CIP catalogue record for this book is available from the British Library.  
ISBN: 978-1-4093-4701-9

Printed and bound in China by South China Printing Company.

See our complete catalogue at  
[www.dk.com](http://www.dk.com)



**CAROL VORDERMAN MA(CANTAB), MBE** is one of Britain's best-loved TV presenters and is renowned for her skills in mathematics. She has a degree in Engineering from the University of Cambridge. Carol has a keen interest in coding, and feels strongly that every child should have the chance to learn such a valuable skill. She has hosted numerous TV shows on science and technology, such as *Tomorrow's World* and *How 2*, as well as *The Pride of Britain Awards*, on the BBC, ITV, and Channel 4. Whether co-hosting Channel 4's *Countdown* for 26 years, becoming the second best selling female non-fiction author of the noughties decade in the UK, or advising British Prime Minister David Cameron on the future of potential mathematics education in the UK, Carol has a passion and devotion to explaining mathematics, science, and technology in an exciting and easily understandable way.



**DR JON WOODCOCK MA(OXON)** has a degree in Physics from the University of Oxford and a PhD in Computational Astrophysics from the University of London. He started coding at the age of eight and has programmed all kinds of computers from single-chip microcontrollers to world-class supercomputers. His many projects include giant space simulations, research in high-tech companies, and intelligent robots made from junk. Jon has a passion for science and technology education, giving talks on space and running computer programming clubs in schools. He has worked on numerous science and technology books as a contributor and consultant.



**SEAN McMANUS** learned to program when he was nine. His first programming language was Logo. Today he is an expert technology author and journalist. His other books include *Scratch Programming in Easy Steps*, *Web Design in Easy Steps*, and *Raspberry Pi For Dummies*. Visit his website at [www.sean.co.uk](http://www.sean.co.uk) for Scratch games and tutorials.



**CRAIG STEELE** is a specialist in Computing Science education. He is Project Manager for CoderDojo Scotland, which runs free coding clubs for young people. Craig has previously worked for the Scottish Qualification Authority, Glasgow Science Centre, and the University of Glasgow. Craig's first computer was a ZX Spectrum.



**CLAIRE QUIGLEY** studied Computing Science at Glasgow University where she obtained a BSc and a PhD. She has worked in the Computer Laboratory at Cambridge University and on a project that aimed to develop computational thinking skills in primary school pupils. She is a mentor at Coderdojo Scotland, a coding club for young people.



**DANIEL McCAFFERTY** holds a degree in Computer Science from the University of Strathclyde. Since graduating, he has been developing software for some of the world's largest investment banks. In his spare time, Daniel is a mentor at CoderDojo Scotland, a coding club for young people.

# Contents

8 **FOREWORD** by Carol Vorderman

10 **HOW THIS BOOK WORKS**

## 1 **WHAT IS CODING?**

14 What is a computer program?

16 Thinking like a computer

18 Becoming a coder

## 2 **STARTING FROM SCRATCH**

22 What is Scratch?

24 Installing Scratch

26 Scratch interface

28 Sprites

30 Coloured blocks and scripts

32 **Project 1: Escape the dragon!**

38 Making things moves

40 Costumes

42 Hide and seek

44 Events

46 Simple loops

48 Pens and turtles

50 Variables

52 Maths

54 Strings and lists

56 Co-ordinates

58 Make some noise

60 **Project 2: Roll the dice**

62 True or false?

64 Decisions and branches

66 Sensing and detecting

68 Complex loops

70 Sending messages

72 Creating blocks

74 **Project 3: Monkey mayhem**

82 Time to experiment

## 3 **PLAYING WITH PYTHON**

86 What is Python?

88 Installing Python

92 Introducing IDLE

94 Errors

96 **Project 4: Ghost game**

98 Ghost game decoded

100 Program flow

102 Simple commands

104 Harder commands

106 Which window?

108 Variables in Python

110 Types of data

112 Maths in Python

114 Strings in Python

116 Input and output

118 Making decisions

120 Branching



- 122 Loops in Python
- 124 While loops
- 126 Escaping loops
- 128 Lists
- 130 Functions
- 132 **Project 5: Silly sentences**
- 134 Tuples and dictionaries
- 136 Lists in variables
- 138 Variables and functions
- 140 **Project 6: Drawing machine**
- 148 Bugs and debugging
- 150 Algorithms
- 152 Libraries
- 154 Making windows
- 156 Colour and co-ordinates
- 158 Making shapes
- 160 Changing things
- 162 Reacting to events
- 164 **Project 7: Bubble blaster**
- 176 What next?

## 4 INSIDE COMPUTERS

- 180 Inside a computer
- 182 Binary and bases
- 184 Symbols and codes
- 186 Logic gates

- 188 Processors and memory
- 190 Essential programs
- 192 Storing data in files
- 194 The Internet

## 5 PROGRAMMING IN THE REAL WORLD

- 198 Computer languages
- 200 Coding stars
- 202 Busy programs
- 204 Computer games
- 206 Making apps
- 208 Programming for the Internet
- 210 Using JavaScript
- 212 Bad programs
- 214 Mini computers
- 216 Becoming a master programmer
  
- 218 Glossary
- 220 Index
- 224 Acknowledgements

Find out more at:

[www.dk.com/computercoding](http://www.dk.com/computercoding)



# Foreword

---



Just a few years ago, computer coding seemed like a mysterious skill that could only be practised by specialists. To many people, the idea that coding could be fun was a strange one. But then the world changed. In the space of a few years, the Internet, email, social networks, smartphones, and apps hit us like a tornado, transforming the way we live.

Computers are a huge part of life that we all now take for granted. Instead of calling someone on the phone, we send a text message or use social media. From shopping and entertainment to news and games, we guzzle on everything computers have to offer. But we can do more than just use this technology, we can create it. If we can learn to code, we can make our own digital masterpieces.

Everything computers do is controlled by lines of code that someone has typed out on a keyboard. It might look like a foreign language, but it's a language anybody can pick up quite quickly. Many would argue that coding has become one of the most important skills you can learn in the 21st century.

---

Learning to code is tremendous fun as you can get instant results, no matter how much more you have to learn. In fact, it's such fun creating games and programs that it feels effortless once you're hooked. It's also creative – perhaps the first science that combines art, logic, storytelling, and business.

Not only that, coding is a fantastic skill for life. It strengthens logical thinking and problem-solving skills – vital in many different areas of life, from science and engineering to medicine and law. The number of jobs that require coding is set to increase dramatically in the future, and there's already a shortage of good coders. Learn to code, and the digital world is yours for the taking!

*Carol Vorderman*

CAROL VORDERMAN



# How this book works

This book introduces all the essential concepts needed to understand computer coding. Fun projects throughout put these ideas into practice. Everything is broken down into small chunks so that it's easy to follow and understand.

Each topic is described in detail, with examples and exercises

"See also" boxes list other subjects that are linked to the topic



170 PLAYING WITH PYTHON

## BUBBLE BLASTER

**Working out the distance**  
In this game, and lots of others, it is between two objects. Here's how to formula to have the computer work

11 This function calculates the distance between two objects. Add this bit of code direct the code you wrote in step 9.

```
from math import sqrt
def distance(id1, id2):
    x1, y1 = get_coords(id1)
    x2, y2 = get_coords(id2)
    return sqrt((x2 - x1)
```

42 STARTING FROM SCRATCH

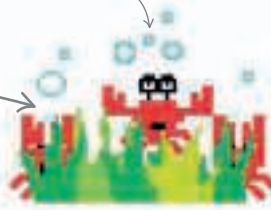
## Hide and seek

Welcome to the special effects studio! Using the purple "Looks" blocks, find out how to make sprites vanish and reappear, grow and shrink, and fade in and out.

### Hiding sprites

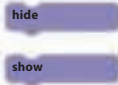
To make a sprite disappear, use the "hide" block. The sprite is still on the stage, and it can still move around, but it can't be seen unless the "show" block is used to make it visible again.

Use the "hide" block to make sprites disappear in games



#### Hide and show

To make a sprite vanish, use the "hide" block. When you're ready for it to be seen again, use the "show" block. These blocks are found in the "Looks" section of the blocks palette.



#### Disappearing cat

Try this script using the cat sprite. It disappears and reappears but it keeps moving, even when you can't see it.

This block hides the cat

This block rotates the cat clockwise

The cat still moves even when hidden

This block shows the cat again

### SEE ALSO

(38-39 Making things move) Sending (70-71) messages

### Sizes and effects

Scripts can be used to change the size of a sprite and add special effects to it.

change size by (10)

Type in positive numbers to make sprites bigger and negative numbers to make them smaller

set size to (100) %

Higher numbers make sprites bigger and lower numbers make them smaller. 100 is normal size

#### Changing a sprite's size

These two blocks can be used to make a sprite bigger or smaller, either by a set amount or by a percentage of its size.

Resets all the effects

△ Addir The gap used to c or distort experie

### Using effects to teleport

Add a ghost sprite from the "Fantasy" category of the sprite library, and create the script shown below. It makes the ghost appear to teleport when clicked.

The "ghost" effect makes the fade slightly; by repeating this block 20 times the sprite fades away completely

This "Oper" selects a random horizontal

Using this block makes the sprite fade back in

Colourful illustrations highlight different programming concepts

Programming scripts and code are explained line by line

Instructions show what to click, drag, or select

Labels help explain each step

### EXPERT TIPS

#### Showing sprites

Select a sprite in the sprite list. Click the "H" button on it to open the information panel. There you can also use the "show" tick box to show or hide a sprite.

Show a hidden sprite

Seven projects build up coding skills. Project pages are highlighted with a blue band

Simple step-by-step instructions guide you through each project

**BUBBLE BLASTER 171**

13 Now update the main game loop to use the functions you have just created. Remember that the order is important, so make sure you put everything in the right place. Then run the code. Bubbles should burst when they hit the sub. Check the shell window to see the score.

```

score = 0
#MAIN GAME LOOP
while True:
    if randint(1, BUB_CHANCE) == 1:
        create_bubble()
    move_bubbles()
    clean_up_bubs()
    score += collision()
    print(score)
    window.update()
    sleep(0.01)

```

Each line of code is clearly labelled so you can't go wrong

**EXPERT TIPS**  
**Python shortcut**  
The code "score += collision()" is a shortcut for writing "score = score + collision()". It adds the collision score to the total score, then updates the total score. Code like this is common, so a shortcut is useful. You can also do the same thing using the "-" symbol. For example, "score -= 10" is the same as "score = score - 10".

Don't forget to save your work

Each line of code is clearly labelled so you can't go wrong

This icon indicates that the project continues on the next page

Boxes give extra information: tips, definitions, and things to remember

**HIDE AND SEEK 43**

Change the numbers in the blocks to set how strong the effect is

pixelate effect by 25

effect to 0

Each colour is represented by a number. Change the number to set the colour

random position

This block selects a random vertical position

This block makes the ghost move slowly, hidden from view

**EXPERT TIPS**

**When to save**

This save icon appears on the project spreads. It reminds you when to save the work you've done, so that nothing is lost if the computer crashes. Remember to always save your work frequently.

Don't forget to save your work



**T**

# What is coding?



# What is a computer program?

A computer program is a set of instructions that a computer follows to complete a task. “Coding”, or “programming”, means writing the step-by-step instructions that tell the computer what to do.

## Computer programs are everywhere

We are surrounded by computer programs. Many of the devices and gadgets we use each day are controlled by them. These machines all follow step-by-step instructions written by a computer programmer.

### SEE ALSO

Thinking like **16–17** ›  
a computer

Becoming **18–19** ›  
a coder



#### ◁ Mobile phones

Programs allow you to make a phone call or send text messages. When you search for a contact, a program finds the correct phone number.



#### △ Computer software

Everything a computer does, from browsing the Internet to writing documents or playing music, works because of code written by a computer programmer.



#### △ Washing machines

Washing machines are programmed to follow different cycles. Computer code controls how hot the water is and how long the wash takes.



#### ◁ Games

Consoles are just another type of computer, and all the games that run on them are programs. All the graphics, sounds, and controls are written in computer code.

#### ▷ Cars

In some cars, computer programs monitor the speed, temperature, and amount of fuel in the tank. Computer programs can even help control the brakes to keep people safe.





## How computer programs work

Computers might seem very smart, but they are actually just boxes that follow instructions very quickly and accurately. As intelligent humans, we can get them to carry out different tasks by writing programs, or lists of instructions.

### 1 Computers can't think

A computer won't do anything by itself. It's up to the computer programmer to give it instructions.



Without instructions a computer is clueless

### 2 Write a program

You can tell a computer what to do by writing a set of very detailed instructions called a program. Each instruction has to be small enough that the computer can understand it. If the instructions are incorrect, the computer won't behave the way you want it to.

This is a computer program counting down to launch

```
for count in range(10, 0, -1):
    print("Counting down", count)
```

### 3 Programming languages

Computers can only follow instructions in a language they understand. It's up to the programmer to choose which language is best for the task.

```
for count in range(10, 0, -1):
    print("Counting down", count)
```

All programs are finally converted into "binary code", a basic computer language that uses only ones and zeroes

```
0000 0000 1000 1100
1000 0100 0100 1001
0100 1001 0000 0101
```

# BLAST OFF!

### LINGO

#### Hardware and software

"Hardware" means the physical parts of the computer that you can see or touch (all the wires, the circuits, the keyboard, the display screen, and so on). "Software" means the programs that run on the computer and control how it works. Software and hardware work together to make computers do useful things.

# Think like a computer

A programmer must learn to think like a computer. All tasks must be broken down into small chunks so they are easy to follow, and impossible to get wrong.

## SEE ALSO

◀ 14–15 What is a computer program?

Becoming 18–19 ▶  
a coder

## Thinking like a robot

Imagine a café where the waiter is a robot. The robot has a simple computer brain, and needs to be told how to get from the café kitchen to serve food to diners seated at tables. First the process has to be broken down into simple tasks the computer can understand.

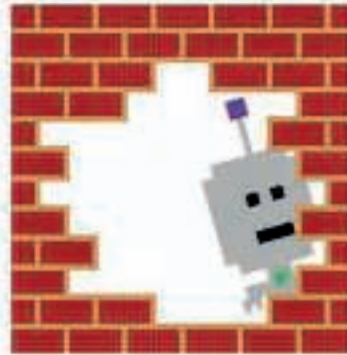
### 1 Waiter robot program 1

Using this program the robot grabs the food from the plate, crashes straight through the kitchen wall into the dining area, and puts the food on the floor. This algorithm wasn't detailed enough.

1. Pick up food

2. Move from kitchen to diner's table

3. Put food down



### ◁ Disaster!

The instructions weren't clear: we forgot to tell the robot to use the door. It might seem obvious to humans but computers can't think for themselves.

### 2 Waiter robot program 2

This time we've told the robot waiter to use the kitchen door. It makes it through the door, but then hits the café cat, trips, and smashes the plate on the floor.

1. Pick up a plate with food on it

2. Move from kitchen to diner's table by:

Move to door between kitchen and dining area

Move from door to the table

3. Put plate down on the table in front of the diner



### △ Still not perfect

The robot doesn't know how to deal with obstacles like the cat. The program needs to give the robot even more detailed instructions so it can move around safely.

**3 Waiter robot program 3**

In this version of the program, the robot successfully delivers the food to the diner avoiding any obstacles. But after putting the plate down, the robot remains standing at the table while food piles up in the kitchen.

1. Pick up a plate with food on it holding it level at all times

2. Move from kitchen to diner's table by:

Move to door between kitchen and dining area

checking for obstacles and steering around them

Move from door to the table

checking for obstacles and steering around them

3. Put plate down on the table in front of the diner

**△ Success at last?**

Finally the robot can deliver the food safely. But we forgot to give it instructions to go back to the kitchen and get the next plate.

**Real-world example**

The waiter robot might be imaginary, but algorithms like this are in action all around us. For example, a computer-controlled lift faces the same sort of problems. Should it go up or down? Which floor should it go to next?



1. Wait until doors are closed

2. Wait for button to be pressed

If button pressed is higher than current floor:

Move lift upwards

If button pressed is lower than current floor:

Move lift downwards

3. Wait until current floor equals button pressed

4. Open doors

**◁ Lift program**

For the lift to work correctly and safely, every step has to be precise, clear, and cover every possibility. The programmers have to make sure they create a suitable algorithm.

# Becoming a coder

Coders are the people who write the programs behind everything we see and do on a computer. You can create your own programs by learning a programming language.

## SEE ALSO

What is **22–23** ›  
Scratch?

What is **86–87** ›  
Python?

## Programming languages

There are a huge range of programming languages to choose from. Each one can be used for different tasks. Here are some of the most popular languages and what they are often used for:

**C** A powerful language for building computer operating systems.

**Ada** Used to control spacecraft, satellites, and aeroplanes.

**Java** Works on computers, mobile phones, and tablets.

**MATLAB** Ideal for programs that need to carry out lots of calculations.

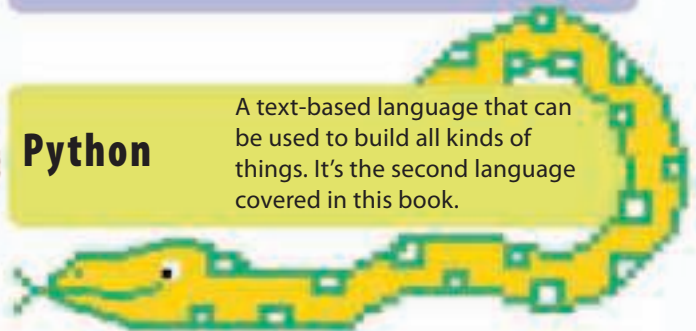
**Ruby** Automatically turns lots of information into web pages.

**Javascript** A language used to build interactive websites.

**Scratch** A visual language that's ideal for learning programming. This is the first language covered in this book.

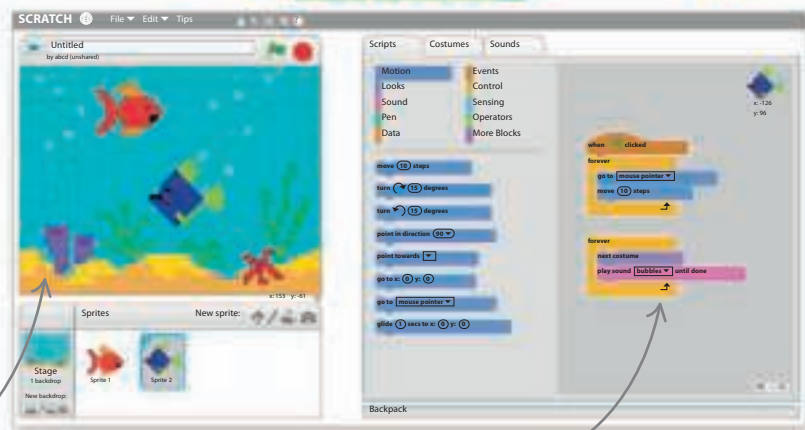


**Python** A text-based language that can be used to build all kinds of things. It's the second language covered in this book.



## What is Scratch?

Scratch is a great way to start coding. Programs are created by connecting together blocks of code, instead of typing it out. Scratch is quick and easy to use, and also teaches you the key ideas you need to use other programming languages.



The program appears on this side of the screen

Code is made by connecting coloured blocks together



## What is Python?

People around the world use Python to build games, tools, and websites. It's a great language to master as it can help you build all kinds of different programs. Python looks like a mixture of recognizable words and characters, so it can be easily read and understood by humans.

A program written  
in Python

```
IDLE  File  Edit  Shell  Debug  Window  Help
ghostgame
# Ghost Game
from random import randint
print('Ghost Game')
feeling_brave = True
score = 0
while feeling_brave:
    ghost_door = randint(1, 3)
    print('Three doors ahead...')
```

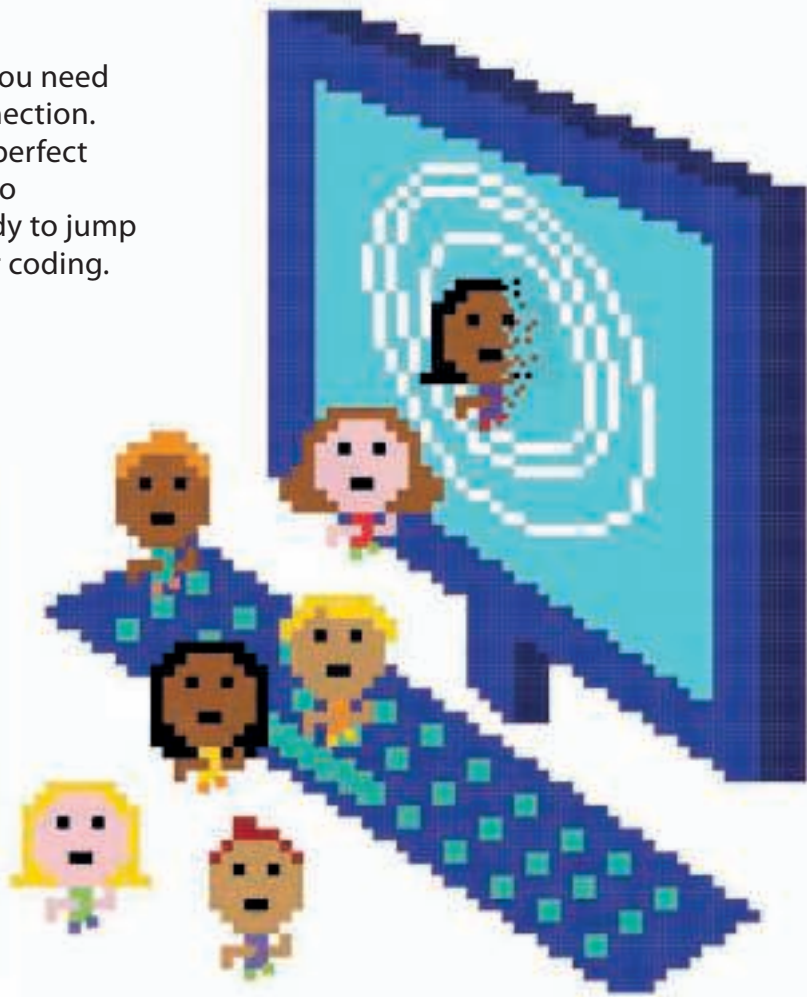
## Getting started

It's time to start programming. All you need is a computer with an Internet connection. This book starts with Scratch – the perfect language to help you on your way to becoming a coding expert. Get ready to jump into the exciting world of computer coding.

### EXPERT TIPS

#### Enjoy experimenting

As a programmer you should experiment with the code and programs you make. One of the best ways to learn programming is to play about and see what happens when you change different parts of the code. By tinkering and fiddling, you'll discover new ways of doing things. You'll learn much more about computer programming and have even more fun.





# Starting from Scratch



# What is Scratch?

Scratch is a visual programming language that makes coding simple. It can be used to make all sorts of fun and interesting programs.

## SEE ALSO

Installing and launching Scratch **24–25** ›

Scratch interface **26–27** ›

Coloured blocks and scripts **30–31** ›

## Understanding Scratch

Scratch is perfect for making games and animations. It has large collections (or “libraries”) of cool graphics and sounds that you can play around with.

### 1 Start programming

Scratch is a programming language. There’s not much typing, and it’s easy to get started.

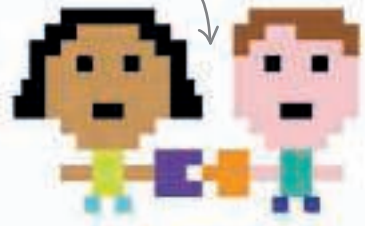
Write your first program in Scratch!



### 2 Put together programming blocks

Scratch uses coloured blocks of code. Blocks are selected and joined together to make a script, which is a set of instructions.

Blocks lock together like jigsaw pieces



### 3 Make sprites move and speak

Objects such as people, vehicles, and animals can be added to a program. These objects are called sprites. Scripts make them move and speak.

Sprites can be programmed to walk, run, and dance



Sprites like me can be programmed to talk in speech bubbles.



## LINGO

### Why is it called Scratch?

“Scratching” is a way of mixing different sounds to make new music. The Scratch programming language enables you to mix pictures, sounds, and scripts to make new computer programs.





- [Dragonfire for free](#)
- [Hip Chick's Guide to Macrobiotics pdf, azw \(kindle\)](#)
- [\*\*click Diagram Geometry: Related to Classical Groups and Buildings \(Ergebnisse der Mathematik und ihrer Grenzgebiete. 3. Folge / A Series of Modern Surveys in Mathematics\) online\*\*](#)
- [Sherry: A Modern Guide to the Wine World's Best-Kept Secret, with Cocktails and Recipes pdf, azw \(kindle\), epub, doc, mobi](#)
- [download Understanding Media: The Extensions of Man](#)
  
- <http://xn--d1aboelcb1f.xn--p1ai/lib/Dragonfire.pdf>
- <http://musor.ruspb.info/?library/The-Gatekeeper--A-Memoir.pdf>
- <http://nexson.arzamashev.com/library/The-Turner-House.pdf>
- <http://www.celebritychat.in/?ebooks/Epiphany-of-the-Long-Sun--The-Book-of-the-Long-Sun--Books-3-4-.pdf>
- <http://www.netc-bd.com/ebooks/Art-Worlds.pdf>