

How to code in

Python

GCSE, iGCSE, National 4/5 and Higher

Greg Reid

Every effort has been made to trace all copyright holders, but if any have been inadvertently overlooked, the Publishers will be pleased to make the necessary arrangements at the first opportunity.

Although every effort has been made to ensure that website addresses are correct at time of going to press, Hodder Education cannot be held responsible for the content of any website mentioned in this book. It is sometimes possible to find a relocated web page by typing in the address of the home page for a website in the URL window of your browser.

Hachette UK's policy is to use papers that are natural, renewable and recyclable products and made from wood grown in well-managed forests and other controlled sources. The logging and manufacturing processes are expected to conform to the environmental regulations of the country of origin.

Orders: please contact Bookpoint Ltd, 130 Park Drive, Milton Park, Abingdon, Oxon OX14 4SE.

Telephone: +44 (0)1235 827827. Fax: +44 (0)1235 400401. Email education@bookpoint.co.uk. Lines are open from 9 a.m. to 5 p.m., Monday to Saturday, with a 24-hour message answering service. Visit our website: www.hoddereducation.co.uk

ISBN: 978 1 5104 6182 6

© Greg Reid 2020

First published in 2020 by

Hodder Education

An Hachette UK Company,

Carmelite House, 50 Victoria Embankment

London EC4Y 0LS

Impression number 5 4 3 2 1

Year 2024 2023 2022 2021 2020

All rights reserved. Apart from any use permitted under UK copyright law, no part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, or held within any information storage and retrieval system, without permission in writing from the publisher or under licence from the Copyright Licensing Agency Limited. Further details of such licences (for reprographic reproduction) may be obtained from the Copyright Licensing Agency Limited, www.cla.co.uk

Cover photo © AndSus/stock.Adobe.com

Illustrations by Aptara Inc.

Typeset in India by Aptara Inc.

Printed in Spain.

A catalogue record for this title is available from the British Library.



Contents

■ Chapter 1 – Introduction	1
Section 1 – Input, output and simple calculations	4
■ Chapter 2 – Examples of input, output and simple calculations	4
■ Chapter 3 – Computational thinking puzzles (input and output)	8
■ Chapter 4 – Examples of simple calculations	10
■ Chapter 5 – Computational thinking puzzles (simple calculations)	12
■ Chapter 6 – Examples of simple string functions	14
■ Chapter 7 – Computational thinking puzzles (string functions)	16
■ Chapter 8 – Examples of simple mathematical functions	19
■ Chapter 9 – Computational thinking puzzles (mathematical functions)	21
■ Chapter 10 – Programming challenges for Section 1	22
Section 2 – Selection (if) statements	30
■ Chapter 11 – Examples of selection (if) statements	31
■ Chapter 12 – Computational thinking puzzles (if statements)	36
■ Chapter 13 – Programming challenges for Section 2	40
Section 3 – Repetition (loop) statements	43
■ Chapter 14 – Examples of repetition (loop) statements	44
■ Chapter 15 – Computational thinking puzzles (loops)	48
■ Chapter 16 – Programming challenges for Section 3	55

Section 4 – Storing multiple values using lists	63
■ Chapter 17 – Examples of lists	66
■ Chapter 18 – Computational thinking puzzles (lists)	72
■ Chapter 19 – Programming challenges for Section 4	80
Section 5 – Predefined functions	86
■ Chapter 20 – Examples of predefined functions	86
■ Chapter 21 – Computational thinking puzzles (predefined functions)	92
■ Chapter 22 – Programming challenges for Section 5	96
Section 6 – Modular programming	100
■ Chapter 23 – Examples of modular programming	101
■ Chapter 24 – Computational thinking puzzles (modular programming)	106
■ Chapter 25 – Programming challenges for Section 6	109
Section 7 – File handling	112
■ Chapter 26 – Examples of file handling	113
■ Chapter 27 – Computational thinking puzzles (file handling)	117
■ Chapter 28 – Programming challenges for Section 7	120
Section 8 – Standard algorithms	123
Section 9 – Large project tasks	136

Section 1 – Input, output and simple calculations

Almost all computer programs are written to **input**, **process** and **output** data. For example:

- A calculator takes numbers and instructions from a keypad (input), performs the calculation requested (process) and displays the answer on the calculator's small screen (output).
- A washing machine senses the weight of clothes (input) and the selected wash cycle from the machine's control panel (input). The amount of water required and length of the wash cycle are calculated (process). Electrical signals are sent to pumps, heaters and motors to control the wash cycle (output).



While it would be great to write program code that reads data from sensors or controls a water pump, the first steps in learning to program usually involve writing simple code that

- allows the user to enter text or numbers using the keyboard (input)
- changes the text or performs simple mathematical calculations with numbers (process)
- displays text or numbers on the user's screen (output).

In programming, the user is the person who will use the executing (running) program.

Chapter 2 – Examples of input, output and simple calculations

Some examples of Python 3 input and output instructions are shown below. Try typing each of these programs into your Python editor. **Execute** (run) the code and observe what happens.

Example 1 – Output using a simple print() statement

To display text or numbers on the user's screen we use the **print()** statement. Notice that displaying text requires quotation marks " " around the words, while displaying numbers does not.

```
▼ Program Code
print("Hello World")
print("I am")
print(44)
print("years old.")
```

```
Output (as seen on the user's screen)
Hello World
I am
44
years old.
```

Each print() statement will display its output on a new line.

Example 2 – Output using a complex print() statement

A print() statement can be made up of several parts, with each part separated by a comma.

Note that Python automatically replaces the comma with a space when the output is displayed.

▼ Program Code

```
print("Hello World")
print("I am",44,"years old.")
```

Output

```
Hello World
I am 44 years old.
```

Example 3 – Keyboard input using a simple input() statement

Input statements are used to ask the user to type in text or numbers.

▼ Program Code

```
print("Please enter your name.")
userName = input( )
```

Input (typed by the user) & output

```
Please enter your name.
Greg
```

When an **input()** statement is used, the program must store whatever data the user enters.

Variables

The second line of the code in Example 3 above creates a **variable** (or storage location) called "userName". When the user types the name (enters text) and then presses enter, the keyboard input "Greg" is stored in the variable.

You might want to imagine variables as boxes that your program temporarily stores data in while it's running.

A variable can be given almost any name made up of letters and numbers. Some exceptions to this are:

- You cannot use two or more separate words as a variable name.
Note that this book uses camel case (using capital letters in the middle of a word) in place of two words. For example, "userName" instead of "user name".
- In Python, variable names should always start with a lower case letter. This is because capital letters are reserved for something else.
- A variable cannot be called any of the statement words of the Python language. For example, "print" or "input" would not be valid variable names.



Chapter 3 – Computational thinking puzzles (input and output)

Any good programmer is able to predict what their code will do. Learning to predict the effect of code will improve your ability to read code, write code and solve problems. This skill is known as 'computational thinking'.

The puzzles throughout this book will improve your computational thinking skills. Each puzzle consists of a short bit of code. You will be asked to predict the output produced when each short program is executed.

For example, the following code would produce the output shown below.

Program code (question)

```
street = "Dover Drive"
donation1 = 120
donation2 = 80
donation3 = 60
print("The total donations for",street,"are:",donation1, donation2, donation3)
```

Output from program (answer)

The total donations for Dover Drive are: 120 80 60

Puzzle set 1 – Input and output

For each of the following puzzles, think through the code and write down the exact output produced by the code (including any spaces). The puzzles deliberately get harder.

1 `productName = "Bicycle Chain"`
`print(productName)`

Output

2 `dogBreed = "Labradoodle"`
`dogAge = "Two"`
`print(dogBreed, dogAge)`

Output

3 `name = "Scott"`
`age = 23`
`print(name, "aged", age)`

Output

4 `coffee = "Lava Java"`
`print(coffee + coffee)`

Output

5 State the output if the user enters "video".

```
game = input("Please enter a word")
print(game, game + game)
```

Output

6 `var1 = "Going"`
`var2 = "Gone"`
`print(var1, "Going" + var2)`

Output

7 `print("This, "+" is a plus +","symbol")`

Output

8 `textOne = "Be kind"`
`textTwo = "possible"`
`textThree = textOne + " whenever"`
`print(textThree, textTwo + ".")`

Output

9 `con1 = "c"`
`vow1 = "e"`
`con2 = "m"`
`print("energy(" + vow1 + ") =", con2 + con1, "squared")`

Output

10 For this puzzle, state the two inputs that would produce the output shown below.

Output

Love the life you live. Live the life you love.

`word1 = input("Please enter the first word")`
`word2 = "the " + word1 + " you"`
`word3 = input("Please enter the first word")`
`print("Love", word2, word3 + ". Live the", word1, "you love.")`

Input