Chris Roffey

Cambridge IGCSE® and O Level Computer Science

Programming Book

For Python

CAMBRIDGE UNIVERSITY PRESS

CAMBRIDGE
UNIVERSITY PRESS

University Printing House, Cambridge CB2 8BS, United Kingdom

One Liberty Plaza, 20th Floor, New York, NY 10006, USA

477 Williamstown Road, Port Melbourne, VIC 3207, Australia

4843/24, 2nd Floor, Ansari Road, Daryaganj, Delhi – 110002, India

79 Anson Road, #06 -04/06, Singapore 079906

Cambridge University Press is part of the University of Cambridge.

It furthers the University's mission by disseminating knowledge in the pursuit of education, learning and research at the highest international levels of excellence.

Information on this title: www.cambridge.org

First published 2017

20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

Printed in Spain by GraphyCems

A catalogue record for this publication is available from the British Library

ISBN 978-1-316-61782-3 Paperback

Additional resources for this publication at www.cambridge.org

IGCSE is the registered trademark of Cambridge International Examinations

All examination-style questions, sample mark schemes, solutions and/or comments that appear in this book were written by the author. In examination, the way marks would be awarded to answers like these may be different.

Contents

Introduction

When Richard Morgan wrote the Visual Basic edition of this book he had two aims in mind. The first was to provide a programming book that specifically covered the material relevant to the Cambridge IGCSE® and O Level Computer Science syllabuses (0478/2210). The second, and perhaps more important, aim was to provide the student with a start to the exciting and rewarding process of being able to create their own computer programs. These are admirable aims that I hope have not been lost in this derivative translation into Python 3.

There are a few subtle changes to the flow diagrams and the pseudocode in this Python edition but fundamentally, wherever possible, the algorithms used are the same as those in the Visual Basic book. This has the exciting outcome that students can be taught the same material in a Cambridge IGCSE and O Level Computer Science class with a mixture of the two books. They can work on solutions in groups and then go and write the code for working implementations of their algorithms in either language.

Python and Visual Basic have different strengths and weaknesses and so they lend themselves to slightly different approaches. For this reason, the chapters have been slightly reordered in this book. The original Chapter 1 has been split: only text-based programming is introduced in Chapter 1 while how to produce GUIs has been moved to the optional Chapter 5. There is also an additional chapter on preparing for the pre-release task. All other chapter titles remain the same so easy comparison should be possible.

Language

The syntax and structures used to implement programming techniques will vary across different languages. This book is entirely based around Python 3, one of the three recommended languages for the Cambridge International AS and A Level syllabus. Python has, at its core, the principle that code should be easy to read. This means that in many ways it is very close to pseudocode.

The pseudocode structure used in the Cambridge IGCSE and O Level Computer Science examination papers uses a language neutral style. Although students are expected to be familiar with this and be able to read and follow the logic easily, they are not expected to produce their own pseudocode in exactly this style. Pseudocode is meant to be a way of expressing clearly the logic of a program, free from the worries of syntax.

Python also has a recommended style guide that can be found at https://www.python.org/dev/peps/pep-0008/.

Here, for example, it is recommended that Python programmers name functions and variables with descriptive all lower case characters separated by underscores, for example `my_variable`. This style is never used in Cambridge IGCSE and O Level Computer Science pseudocode; however, students should not be marked down for doing so in their own pseudocode. As it could be very confusing to keep swapping naming conventions, this book assumes that students are going to stick, wherever possible, to the correct Python style but be flexible enough thinkers to be able to read other pseudocode styles. It is recommended that when preparing for exams, students ensure they are aware of the exam board variable naming style. Chapter 13 in this book provides some examination style questions.

Examination focused

The Cambridge IGCSE and O Level Computer Science course will test computational thinking independent of any specific programming language. It will do this through the use of program design tools such as structure diagrams and flowcharts. It will also make use of pseudocode, a structured method for describing the logic of computer programs.

It is crucial that the student becomes familiar with these techniques. Throughout this book all the programming techniques are demonstrated in the non-language-specific format required, with the exception of variable and function naming. This will help to prepare the student to answer the types of question they will meet in their studies.

To support learning, many of the chapters include examination-style tasks. Chapter 14 has examples of appropriate code solutions that show how to turn logical ideas into actual programs. There is also a series of examination-style questions in Chapter 13, which has a sample mark scheme giving possible solutions and showing where the marks might be awarded.

Developing programming skills

One of the advantages of Python is that it provides a language that encourages the student to program solutions making use of the basic programming constructs: sequence, selection and iteration. Although the language does have access to many powerful pre-written code libraries, they are not generally used in this book.

Computational thinking is the ability to break down a problem into its constituent parts and to provide a logical and efficient coded solution. Experience shows that knowing how to think computationally relies much more on an understanding of the underlying programming concepts than on the ability to learn a few shortcut library routines.

This book is aimed at teaching those underlying skills which can be applied to the languages of the future. It is without doubt that programming languages will develop over the coming years but the ability to think computationally will remain a constant.

How to use this book: a guided tour

Chapter 8: Designing Algorithms

Learning objectives

By the end of this chapter you will understand:

- that systems are made up of subsytems, which may in turn be made up of further subsytems
- how to apply top-down design and structure diagrams to simplify a complex system
- how to combine the constructs of sequence, selection and iteration to design complex systems
- how to produce effective and efficient solutions to complex tasks.

Learning objectives – are included at the beginning of each chapter and present the learning aims for the unit.

This is very similar to the Cambridge IGCSE and O Level Computer Science pseudocode format:

```
WHILE counter > 0 DO
    //code to be iterated
    counter = counter – 1
ENDWHILE
```

KEY TERMS

WHILE loop: A type of iteration that will repeat a sequence of code while a set of criteria continue to be met. If the criteria are not met at the outset the loop will not run and the code within it will not be executed.

Key terms – provide clear definitions for the most important terms within each unit.

SYLLABUS CHECK

Pseudocode: understand and use pseudocode, using WHILE . . . DO . . . ENDWHILE loop structures.

Syllabus checks – link programming concepts to points on the Cambridge IGCSE syllabus.

Each individual element of the loop performs an important role in achieving the iteration, as shown in Table 7.04.

Table 7.04

Element	Description
`while`	The start of the loop
`counter > 0`	The condition that controls the loop. Each time the iteration is run, the condition is evaluated and if it remains True, the iteration will run. Once the condition is False, execution of the code is directed to the line following the loop. In counter-controlled WHILE loops, it is important that code is included within the loop to increment or decrement the counter. In a FOR loop, the counter is automatically incremented. The same facility does not apply to WHILE loops and, as a result, the programmer must include appropriate code.
end of indented code	The end of the current iteration. Execution of the program returns to `while` so the condition can be re-evaluated and further iterations actioned. Do not forget to add ENDWHILE when writing pseudocode.

TIP

Remember that WHILE loops iterate while the condition evaluates to True. It is possible to create an infinite loop rather easily:

```
>>> while True:
        print('Hello', end='')
```

It is therefore important to know how to break out of infinite loops. To do so, hold down the CTRL key on your keyboard and press C. Try the code above yourself in an interactive session. The optional parameter `end=''` provided in the `print()` function suppresses the default line return.

Tip boxes – offer quick suggestions to remind students about important learning points.

EXTENSION TASK

Task 2 – Discussion Question

a What is the aim of this flowchart?

b What kind of loop is being suggested here?

Extension tasks – build on task exercises to help the student further develop their knowledge and understanding.

11.07 Array Tasks

TASKS

Task 6

a Draw a flowchart and create a pseudocode algorithm that iterates through an array of Integers and outputs the average. Declare and initialise the array with the following set of Integers: 12, 14, 10, 6, 7, 11 and 3.

b Test that your algorithm works by programming and running the code in Python.

Task 7

An algorithm will take an Integer value, *n*. It will call a subroutine to place into an array 12 incremental multiples of *n* (the first array index will hold 1 × *n* and the last index position 12 × *n*). An additional subroutine will allow the user to output all the multiples in order.

a Draw a flowchart and create pseudocode for this algorithm.

b Test that your algorithm works by programming and running the code in Python.

Task 8

The data in Table 11.06 is to be organised in arrays so that the user can search via User ID and the system will display all the data related to that User ID.

Table 11.06

User ID	Age	Gender
112	45	Male
217	16	Female
126	27	Female

a Draw a flowchart and create a pseudocode algorithm that accepts a User ID and displays the related data.

b Test that your algorithm works by programming and running the code in Python.

Tasks – contain exercises for the student to test their knowledge of the topic.

Summary

- An array is a variable that can hold a set of data items, of the same data type, under a single identifier.
- When an array is declared, its size is defined. In Python indexes start from zero.
- Each element or data item in an array can be referenced by its index.
- The index can be used to read or write values in an array.
- A FOR loop can be used to iterate through the index locations in an array. The loop counter is used to identify successive index numbers.
- Holding records which consist of more than one data item can be achieved by the use of multiple arrays. Data for each record is held at the same index position in the different arrays.
- When using Python to implement algorithms involving arrays, a list is used as a substitute for an array.

Summary checklists – are included at the end of each chapter to review what the student has learned.

Acknowledgements

Thanks to the following for permission to reproduce images:

Cover image: Soulart/Shutterstock; Chapter opener 1 isak55/Shutterstock; Chapter opener 2 aimy27feb/Shutterstock; Chapter opener 3 Image Source/Getty Images;Chapter opener 4 Devrimb/iStock/Getty Images;Chapter opener 5 Andrew Brookes/Getty Images; Chapter opener 6 Magictorch/Ikon Images/Getty Images;Chapter opener 7 alexaldo/iStock/Getty Images;Chapter opener 8 Ioana Davies (Drutu)/Shutterstock; Chapter openers 9, 10 Kutay Tanir/Photodisc/Getty Images;Chapter opener 11 ILeysen/Shutterstock; Chapter opener 12 Kamil Krawczyk/E+/Getty Images; Chapter opener 13 Aeriform/Getty Images