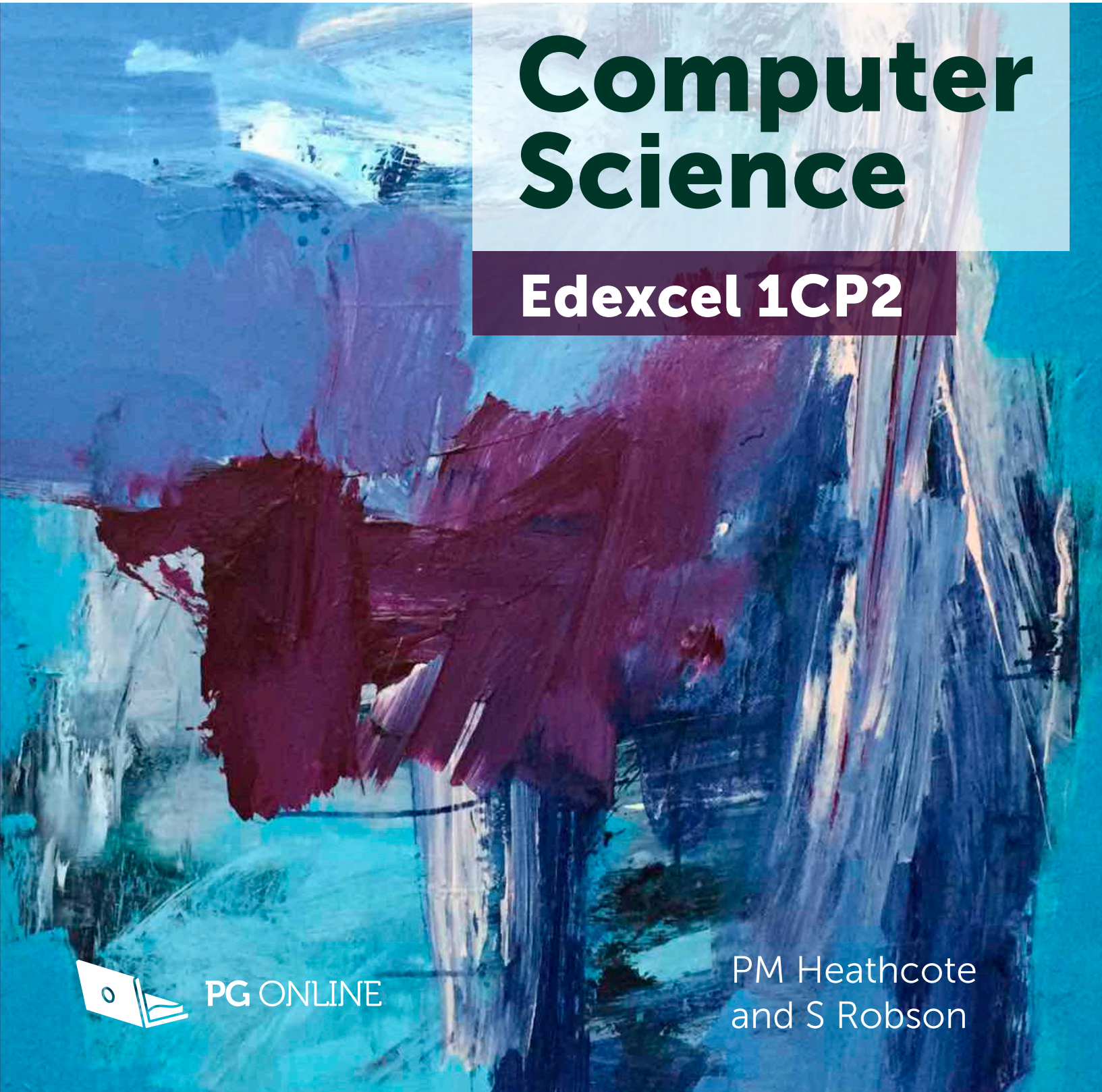
Endorsed for
Pearson Edexcel
Qualifications

GCSE (9-1)

Computer Science

Edexcel 1CP2



PM Heathcote
and S Robson

PG ONLINE

Edexcel GCSE (9-1) 1CP2

Computer Science

P.M. Heathcote
S. Robson

Published by
PG Online Limited
The Old Coach House
35 Main Road
Tolpuddle
Dorset
DT2 7EW
United Kingdom
sales@pgonline.co.uk
www.pgonline.co.uk
2024

PG ONLINE

Acknowledgements

The answers in the Teacher's Supplement are the sole responsibility of the authors and have neither been provided nor approved by the examination boards.

We would also like to thank the following for permission to reproduce copyright photographs:

Page 2: London Underground passengers photo © Matthew Ashmore

Page 2: Angel of the North © Ron Ellis / Shutterstock

Page 4: London Underground map © Thinglass / Shutterstock.com

Color Picker screenshot © Adobe Inc.

Other photographic images © Shutterstock

Cover picture: 'Antibes' 2017
Acrylic on linen, 100 cm x 100 cm
© Deborah Lanyon
www.deborahlanyon.co.uk

Cover artwork, graphics and typesetting by PG Online Ltd

First edition 2024 10 9 8 7 6 5 4 3 2 1

A catalogue entry for this book is available from the British Library

ISBN: 978-1-916518-14-8

This product is made of material from well-managed FSC® certified forests and from recycled materials.

Printed by Bell & Bain Ltd, Glasgow, UK.

Preface

Aimed at GCSE students, this book provides comprehensive yet concise coverage of all the topics covered in the Edexcel GCSE (9-1) Computer Science (1CP2) specification, written and presented in a way that is accessible to teenagers. It can be used as a course text and as assessment preparation for students nearing the end of their course.

It is divided into seven sections covering every element of the specification. Sections 6A and 6B of the textbook cover programming concepts with a practical approach to provide students with experience of understanding, writing and adapting programs that solve problems.

Each chapter contains exercises and exam-style questions. Answers to all of these are available to teachers only in a free Teachers' Supplement which can be ordered from our website **www.pgonline.co.uk**.

Endorsement Statement

In order to ensure that this resource offers high-quality support for the associated Pearson qualification, it has been through a review process by the awarding body. This process confirms that this resource fully covers the teaching and learning content of the specification or part of a specification at which it is aimed. It also confirms that it demonstrates an appropriate balance between the development of subject skills, knowledge and understanding, in addition to preparation for assessment.

Endorsement does not cover any guidance on assessment activities or processes (e.g. practice questions or advice on how to answer assessment questions), included in the resource nor does it prescribe any particular approach to the teaching or delivery of a related course.

While the publishers have made every attempt to ensure that advice on the qualification and its assessment is accurate, the official specification and associated assessment guidance materials are the only authoritative source of information and should always be referred to for definitive guidance.

Pearson examiners have not contributed to any sections in this resource relevant to examination papers for which they have responsibility.

Examiners will not use endorsed resources as a source of material for any assessment set by Pearson.

Endorsement of a resource does not mean that the resource is required to achieve this Pearson qualification, nor does it mean that it is the only suitable material available to support the qualification, and any resource lists produced by the awarding body shall include this and other appropriate resources.

This resource was designed using the most up to date information from the specification. Specifications are updated over time which means there may be contradictions between the resource and the specification, therefore please use the information on the latest specification and Sample Assessment Materials at all times when ensuring students are fully prepared for their assessments.

Contents

Paper 1 – Computer systems

Edexcel GCSE 1CP2 (9-1) Specification map

		Section 1	Section 2	Section 3	Section 4	Section 5	Section 6A & 6B
1	**Computational thinking**						
1.1	Decomposition and abstraction	✓					
1.2	Algorithms	✓					
1.3	Truth tables	✓					
2	**Data**						
2.1	Binary		✓				
2.2	Data representation		✓				
2.3	Data storage		✓				
3	**Computers**						
3.1	Hardware			✓			
3.2	Software			✓			
3.3	Programming languages			✓			
4	**Networks**						
4.1	Networks				✓		
4.2	Network security				✓		
5	**Issues and impact**						
5.1	Environmental					✓	
5.2	Ethical and legal					✓	
5.3	Cybersecurity					✓	
6	**Programming**						
6.1	Develop code						✓
6.2	Constructs						✓
6.3	Data types and structures						✓
6.4	Input/output						✓
6.5	Operators						✓
6.4	Subprograms						✓

Section 1

Computational thinking

Objectives

- understand the benefit of using decomposition and abstraction to model aspects of the real world and analyse, understand and solve problems
- understand the benefits of using subprograms
- be able to follow and write algorithms including flowcharts, pseudocode and program code
- be able to use sequence, selection and repetition
- know how to use both count-controlled and condition-controlled repetition
- be able to use iteration over every item in a data structure
- be able to use input, processing and output to solve problems
- understand how to follow and write algorithms that use variables and constants
- be able to use one- and two-dimensional data structures, including strings, records and arrays
- be able to follow and write algorithms that use the arithmetic operators addition, subtraction, division, multiplication, modulus, integer division and exponentiation
- understand how to follow and write algorithms that use the relational operators equal to, less than, greater than, not equal to, less than or equal to, greater than or equal to
- be able to follow and write algorithms that use the logical operators AND, OR and NOT
- determine the correct output of an algorithm for a given set of data
- be able to use a trace table to determine the value variables will hold at a given point in an algorithm
- understand the errors that occur in programs including syntax, logic and runtime
- be able to identify and correct logic errors in algorithms
- understand how the linear search and binary search algorithms work
- understand how the bubble sort and merge sort algorithms work
- be able to use logical reasoning and test data to evaluate an algorithm's fitness for purpose and efficiency
- understand that efficiency may be considered in terms of the number of comparisons, number of passes through a loop or the use of memory
- be able to apply the logical operators AND, OR and NOT in truth tables with up to three inputs to solve problems

1.1 – Algorithms, decomposition and abstraction

Computational thinking

Computer Science is all about studying problems and working out how to solve them. The problem might be a mathematical one such as adding the numbers 1 to 100, or finding all the prime numbers less than a million. It might be something less well-defined, such as getting a computer to recognise when the platform of an underground railway is becoming dangerously full. It could be a problem that a virus-checker attempts to solve – detecting when your computer has a virus.

A human being looking at a CCTV of an underground platform would be able to tell quite easily if it was too crowded, and no more people should be allowed through the barrier, but how do we get a computer to recognise that situation?

Some of the key concepts in computational thinking include:

- abstraction
- decomposition
- algorithmic thinking

Abstraction

Abstraction involves removing unnecessary details from a problem in order to solve it. We are all familiar with the idea of abstracting away details from abstract paintings and statues; think of the famous statue 'Angel of the North' by Antony Gormley, dominating the skyline near the A1 at Gateshead.

2.1 – Storage units and binary numbers

Computers are made up of complicated hardware that stores and processes data. If you break a computer down into its most basic components you have millions of circuits that either allow electricity to flow, or not. Imagine a whole row of light switches that you can switch on and off in different combinations to mean different things. Each switch is either on or off. It has only two states, which can be represented by 1 or 0. This is called **binary**.

All data, such as numbers, program instructions, text, sound and graphics are stored and processed as binary.

A single **1** or **0** is a **bi**nary digi**t**, or a **bit** for short. A group of eight bits is called a **byte**. Imagine you've taken a small bite out of an apple, you might call that a nibble. So four bits, half a byte, is called a **nibble**.

Units

A byte is the smallest addressable unit of memory in a computer. Just as a kilometre is 1000 metres, we can group together 1000 bytes to make a **kilobyte**. In Computer Science, memory is often considered using powers of 2. So, a **kibibyte** is 2^{10} = 1,024 bytes, a **mebibyte** is 2^{20} = 1,048,576 bytes, and so on.

Memory size is measured in the following multiples:

Unit	Number of bytes	Number of bytes	Equivalent to
Kibibyte (kiB)	2^{10}	1,024 bytes	1,024 bytes
Mebibyte (MiB)	2^{20}	1,048,576 bytes	1,024 kiB
Gibibyte (GiB)	2^{30}	1,073,741,824 bytes	1,024 MiB
Tebibyte (TiB)	2^{40}	1,099,511,627,776 bytes	1,024 GiB

(**Note:** An alternative system of units makes use of standard prefixes, with 1 kilobyte = 1000 bytes, 1 megabyte = 1,000,000 bytes, 1 gigabyte = 1,000,000,000 bytes, 1 terabyte = 1,000,000,000,000 bytes.)

Q1

(a) Convert 2,048 MiB into GiB

(b) Convert 1 TiB into GiB

Q2

(a) How much storage is needed for a typical photograph file taken from a mobile phone camera?

(b) A music streaming service has 1 million tracks. A typical track requires 3 MiB. How much storage will be needed to store all the tracks?

Computers use the binary system to store numbers and perform binary arithmetic and logic operations. The storage capacities of computer systems have grown, and the memory size of secondary storage systems is now commonly measured in tebibytes (TiB).

Binary

Binary data uses only two digits, 0 and 1. Our denary system uses ten digits, 0 to 9. The number 75, for example, is 7 tens plus 5 units.

Imagine you are back in primary school, learning to add again. 7 + 5 = 12, so you write down the 2 units but carry the group of 10. 23 would be 2 groups of 10 and 3 units.

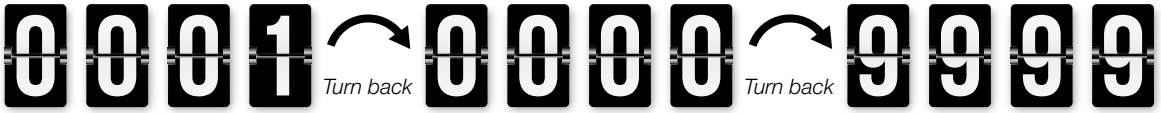
2.3 – Two's complement and binary shifts

Two's complement

Previously you have looked at binary to denary conversions that use positive integers. These are known as **unsigned** integers as there is no + or – sign at the start of the number.

When storing numbers, only binary 1s and 0s can be used. A system known as two's complement is used to store both negative and positive numbers using one bit pattern.

Imagine an analogue counter. As you turn it back, it goes from 0001 to 0000 to 9999.



The 9999 then represents the number -1. 9998 would represent -2.

Two's complement numbers work in the same way using binary numbers.

Two's complement	Denary
1000 0000	-128
1000 0001	-127
…	
1111 1110	-2
1111 1111	-1
0000 0000	0
0000 0001	1
0000 0010	2
…	
0111 1101	125
0111 1110	126
0111 1111	127

Q12 Give the 8-bit two's complement binary that represents the number +3.
Use the table above to help you.

Q13 Give the 8-bit two's complement binary that represents the number -3.
Use the table above to help you.

Exercises

1. (a) Add the following two 8-bit binary numbers. [2]

```
0 0 1 1 0 1 0 1
1 0 0 1 1 1 0 1
```

(b) Explain what is meant by an overflow error when adding two 8-bit binary numbers. [2]

2. The number 73 could be a denary number or a hex number.

(a) If 73 is a hex number, calculate its value as a denary number.

You **must** show your working. [2]

(b) If 73 is a denary number, calculate its value as a hex number.

You **must** show your working. [2]

3. (a) Convert the binary number `10110110` to denary, showing your working. [2]

(b) Convert the denary number 175 to binary, showing your working. [2]

4. (a) Convert the denary number -32 to two's complement. [3]

(b) Convert the two's complement binary number `11011011` to denary. [3]

5. An arithmetic right shift of one place is applied to the number `10110100`.

(a) State the result after the arithmetic right shift has been applied. [1]

(b) State the result which would have occurred if a logical right shift had been used. [1]

6. (a) Explain why data is stored in computers in a binary format. [2]

(b) In the ASCII character set, the character codes for three capital letters are given below.

Letter	ASCII character code
A	65
N	78
R	82

(i) Explain how the ASCII character set is used to represent text in a computer. [2]

(ii) Convert the word RAN into binary using the ASCII character set. [4]

7. A digital camera stores image files that require 3 mebibytes each.

Calculate the amount of storage space that would be required to store 1024 images. Give your answer in gibibytes.

You must show your working. [2]

3.4 – Operating systems

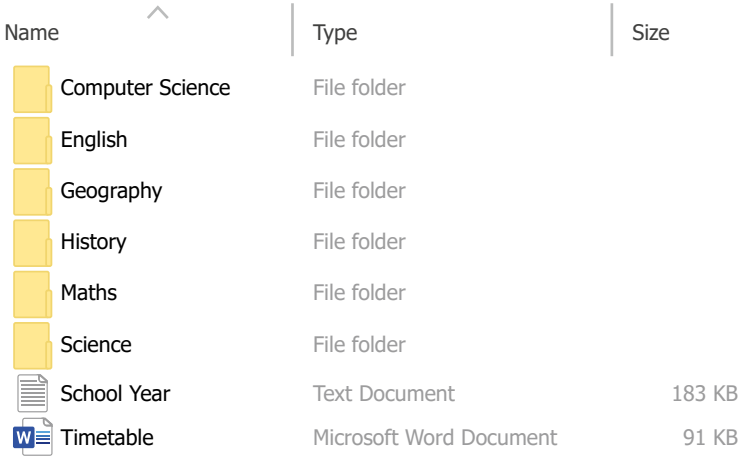
The operating system is a group of programs that manages the computer's resources. One feature of most operating systems that you will already be familiar with is the graphical user interface (GUI). This is the part of the operating system that allows the user to interact with it through the use of icons, windows, menus and pointers.

Operating systems are also responsible for other functions including:

- File management
- User management
- Process management
- Peripheral management

File management

A file system is normally organised into folders and subfolders for easy user navigation and usage. These folders contain files which could be software programs, databases, documents and many other types of file.



File management systems:

- enable a user to create, name, save, modify, copy, delete and move files and folders
- enable a user to search for a particular file
- keep track of location of files on the storage device so that they can be retrieved when needed
- keep track of the free space available where files can be stored
- enable users to restore deleted files
- prevent conflicts when two users on a network attempt to modify the same file
- maintain access rights to files

User management

User management enables a network administrator to allocate accounts and set different access rights for different users. Administration software can also identify all the users currently on the network, manually log out users and monitor when and for how long each user is logged in over a period of time.

The user management system allows the administrator to set default logout times based on inactivity.

5.1 – Environmental issues

Environmental issues associated with digital devices

The use of digital devices has many associated environmental issues. These include:

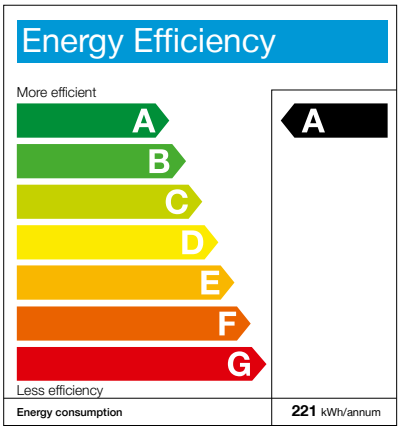
- Energy consumption
- Manufacture
- Replacement cycle
- Disposal

Energy consumption

All digital devices require energy to use. Desktop computers may typically consume 200 watts of power when running. Meanwhile, a more efficient laptop might require only 50 watts. Even more efficient tablet computers may require just 20 watts.

This energy usage almost always comes from the National Grid and therefore will contribute to the use of oil, gas and nuclear materials.

Most people are aware that computers and devices use energy, and as such they may choose more energy efficient devices and appliances. Domestic appliances, in particular, make use of energy efficiency labels to help inform customers and motivate them to buy more energy efficient models.



Manufacture

There are also significant amounts of energy required to manufacture electronic products. For instance, a desktop computer's manufacture could require the same amount of energy as running the computer in an office for over three years.

Computers and other digital devices make use of many different minerals such as iron, aluminium, copper, cobalt and cadmium. Even gold is used for the contacts on CPUs. These materials are not renewable, and if they cannot be recovered in recycling then they deplete reserves. The mining process itself may be harmful to the environment, wildlife and people.

Replacement cycle

The replacement cycle is the amount of time it takes from purchasing an electronic product to when it is replaced. Think about the energy use in the manufacture of a desktop computer. If the computer is used for just three years this would be significant. However, if the computer is used for nine years, only one third of the energy would have been needed for each year. The same can be said for all the raw materials that went into its manufacture.

Python

The programming language used for Edexcel Paper 2 is **Python** (version 3). The exam board produces a **Programming Language Subset** (**PLS**) which you should use as a reference when programming. This covers all the different parts of Python that could be used to solve problems and answer questions in the Paper 2 exam.

Python is a free programming language and may be downloaded from **https://www.python.org/downloads/**

6A.1 – Data types and operators

Variables, data types and constants

Any data that a program uses must be stored in main memory locations, each given its own identifier while the program is running. As the program runs, the values in these locations might change, which is why they are called **variables**. For example: a variable called `total` might change several times as many numbers are added to it. A variable called `surname` may change as the program processes a list of customer orders.

Each variable has an **identifier** (a unique name) that refers to a location in memory where the data item will be stored. Each variable also has a **data type** that defines the type of data that will be stored at the memory location and therefore the operations that can be performed on it (for example, you can multiply two numbers but you can't multiply two words).

Data types

Variables need to be declared before they are used. In some languages, such as C# or Java, you need to state what type the variable will be. In Python, the variable type is determined by the value it is first given. So if you write `a = 23`, `b = "Fred"`, `a` will be stored as an integer (whole number) and `b` will be stored as a string (text).

The table below shows the different data types that are used in Python.

Data type	Data type in Python	Meaning	Example of the data type being used
Integer	int	A whole number, positive or negative.	`age = 15`
Floating point number	float	A number with a decimal point.	`height = 1.73`
Boolean	bool	Holds the value True or False.	`gameOver = True`
String	str	A sequence of one or more letters.	`city = "Leeds"`
Character	str	One letter. In Python this is held in a string.	`firstInitial = "S"`

6A.5 – Records and files

Previously we looked at arrays. An array is a collection of data items stored under one identifier so that the data items can be processed easily. When we group data items together so they can be treated as a set of data, we refer to this as a **data structure**.

Records

Most languages will allow arrays to be defined quite easily but sometimes we want to define our own data structures. Imagine a program for a car sales showroom. If your program is going to process details about cars, it will be easier to create a record structure to hold all of the car details rather than storing them as one long string of text or lots of separate variables. We cannot put them in an array because the separate data items we need to store about each car are not all of the same data type.

Here is a text file with some car details in it:

```
RE05 HSD, Ford, 2005, 97500, 650
SW12 SDF, Vauxhall, 2012, 59650, 2500
BN64 WJR, Nissan, 2014, 39900, 18000
```

We could process this file as lines of text but it would be easier it we defined our own data type that gave this line of text some structure. Delphi is a high-level programming language. A Delphi programmer could define this record type as follows:

```
type TCar = record
   registration:  string;
   make:          string;
   year:          integer;
   mileage:       integer;
   price:         integer;
end;
```

Individual data items within a record are called fields.

Table: CarTable

registration	make	year	mileage	price
AV60 HES	Peugeot	2010	33156	£5,500
GF56 RTE	Toyota	2006	26875	£8,500
FD02 YOU	Hyundai	2002	85300	£3,499
AD62 HGF	Peugeot	2012	50887	£7,649
AF63 HTE	Peugeot	2013	45860	£6,780
GF64 NGB	Renault	2014	38665	£6,199
GR11 JUL	Renault	2011	90760	£2,999

Code readability

It is important for code to be readable as this helps both you and other programmers to understand and maintain it.

To make code more easily readable, make use of the following techniques:

- Use meaningful identifiers – for example, a list named `testScores` is far easier to understand than a list named `ts`.
- Code layout – declaring variables at the start of the program is one way to improve the layout. Having blank lines between code blocks also helps to make it easier to read and understand.
- Comments – comments are useful for helping to understand code. They don't need to be on every line as this is often unhelpful. They should explain difficult to understand lines of code or what the purpose of a function, procedure or block of code is.
- White space – using whitespace includes blank lines to separate code blocks, indentation of code blocks and using additional white space around operators – for example:

```
average = sum / total
```

is easier to read than:

```
average=sum/total
```

Q13 Look at the following code which calculates the maximum of two numbers. Rewrite the code to improve its readability.

```
def max(n1,n2):
    if n1>n2:
        return n1
    else:
        return n2
a=int(input("Enter first number: "))
b=int(input("Enter second number: "))
print("Larger number is: "+str(max(a,b)))
```

Errors and testing

When you write a program in a high-level programming language, a **translator** (**compiler** or **interpreter**, will scan each line of code and convert it into machine code. As you will already have found out, programming is not as easy as it looks.

- Firstly, it is very easy to make mistakes typing in the code, for example typing "prnt" instead of print. These are **syntax errors**.
- Secondly, once you have corrected all the syntax errors, the code may run but not do what you want. This means there are **logic errors** in your program.
- Thirdly, code may run, but create an error that crashes the program when it is running. These are known as **runtime errors**. For example, if a program tries to divide by zero.

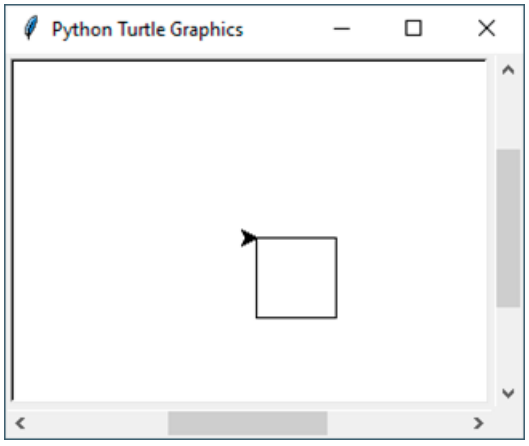
Syntax errors

The translator expects commands to have a certain format, called syntax, just like a sentence in English has grammar rules. Syntax is a set of rules which defines the format of each command.

Example 12

The following example shows how a square could be drawn with the turtle.

Program	Output

`import turtle`
`screen = turtle.Screen()`
`screen.setup(800,400)`
`turtle.screensize(800,400)`
`turtle.speed(0)`
`terry = turtle.Turtle()`
`terry.forward(50)`
`terry.right(90)`
`terry.forward(50)`
`terry.right(90)`
`terry.forward(50)`
`terry.right(90)`
`terry.forward(50)`
`terry.right(90)`
`turtle.done()`
	

Q21 Improve the code so that the square is drawn using a FOR loop.

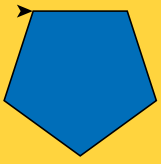
Filling shapes

It is possible to have the turtle fill a shape with colour once it has been drawn.

The following subprograms are used:

Subprogram	Meaning
`terry.fillcolor("red")`	Change the fill colour to red. Alternatively, RGB colours can be used. For example (1, 0, 0) or "#FF0000" will also be red.
`terry.begin_fill()`	Start filling the shape.
`terry.end_fill()`	End filling the shape.

Q22 Create a program to draw a pentagon (five sided shape) that is filled with blue



Turtle colours

Some predefined turtle colours include:

Blue, black, green, yellow, orange, red, pink, purple, indigo, olive, lime, navy, orchid, salmon, peru, sienna, white, cyan, silver and gold.

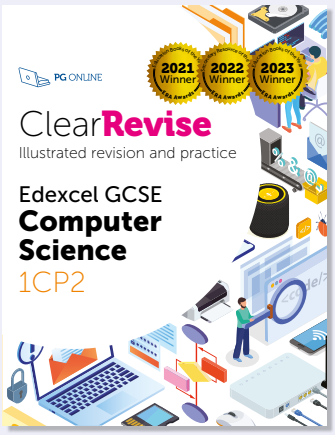
Index

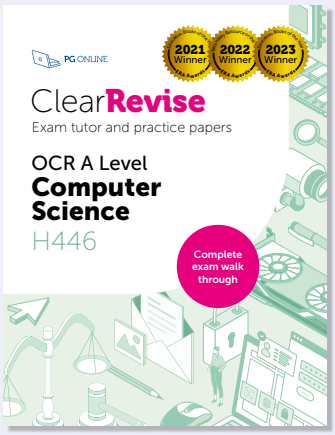
New titles **coming soon!**

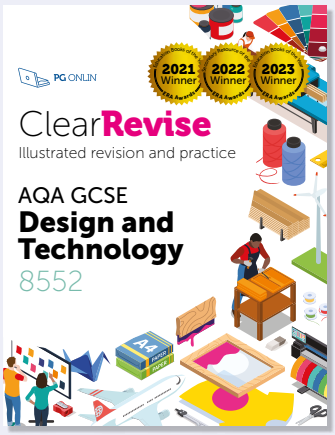
ClearRevise Guides

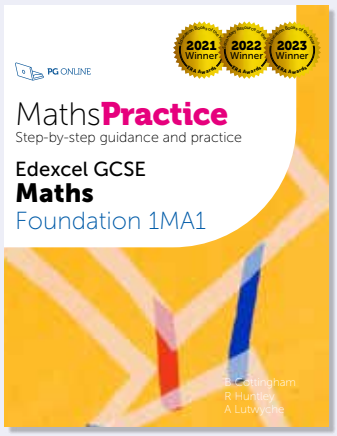
Multi-award-winning revision series

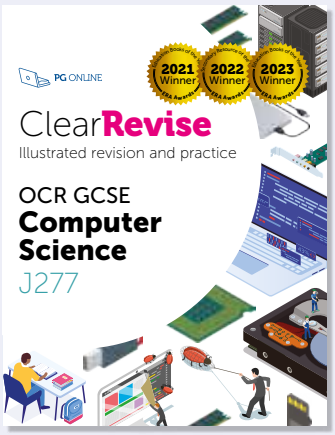
- Hundreds of marks worth of examination style questions
- Answers provided for all questions within the books
- Illustrated topics to improve memory and recall
- Specification references for every topic
- Examination tips and techniques
- Free Python solutions pack (CS Only)

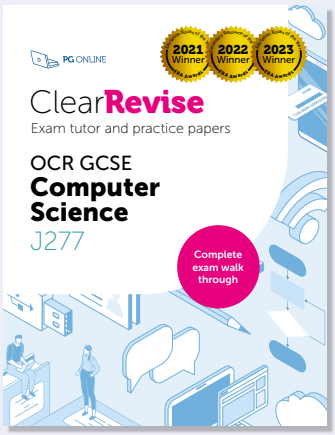


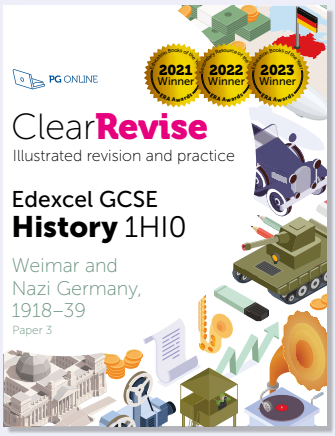


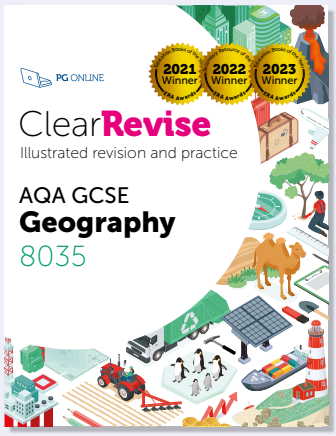


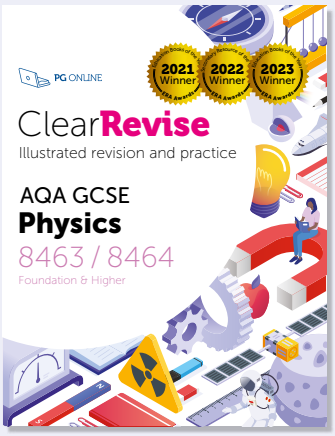


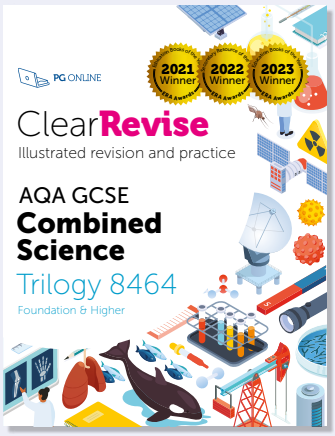


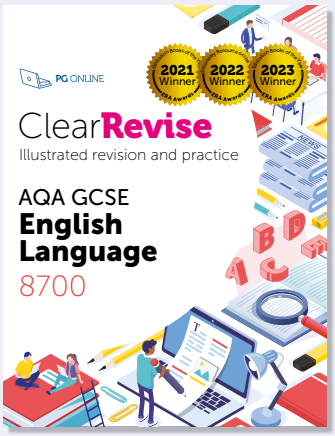


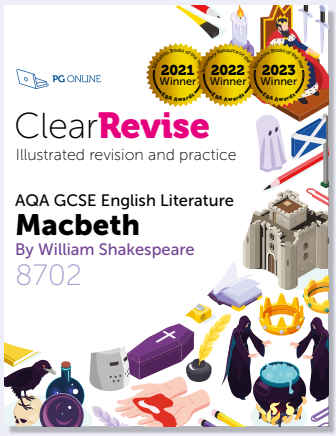


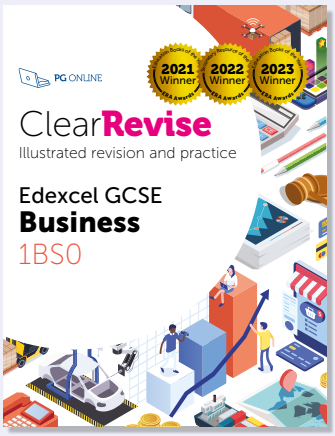












Explore the series and add to your collection at **www.clearrevise.com**

Available from all good book shops

amazon

@pgonlinepub

Edexcel GCSE 1CP2

Computer Science

The aim of this book is to provide an accessible text for students, covering each of the elements in the Edexcel 1CP2 Computer Science GCSE (9-1) specification. It can be used both as a course text and as a revision guide for students nearing the end of their course. It is divided into eight sections, each broken down into manageable chapters of roughly one lesson.

The second section of the textbook covers algorithms and programming concepts with a practical approach to provide students with experience of writing, tracing and debugging Python programs.

Each section contains in-text questions and practice exercises, which can be set as homework. Answers to all of these are available to teachers only, in a free Teachers' Supplement, which can be ordered from our website **www.pgonline.co.uk**

About the authors

Pat Heathcote is a well-known and successful author of Computer Science textbooks. She has spent many years as a teacher of A Level Computing courses with significant examining experience. She has also worked as a programmer and systems analyst, and was Managing Director of Payne-Gallway Publishers until 2005.

Susan Robson worked for International Computers Ltd after graduating from Manchester University with a degree in Computer Science. She spent the following 12 years in technical pre-sales for ECI Telecom, before moving into teaching. As a Head of Computer Science, she gained years of experience teaching GCSE and A Level Computing and has written successful textbooks and teaching materials.

Cover picture:

'Antibes'
Acrylic on linen,
100 x 100 cm
© Deborah Lanyon
www.deborahlanyon.co.uk

This book has been endorsed by Edexcel.

ISBN 9781916518148

PG ONLINE