Computational thinking

In this topic, you will be introduced to some of the key concepts of Computer Science that will help you to think like a programmer and solve problems using code. You'll learn to write, **analyse and modify** code, while developing your **computational thinking skills**.

You will understand how to use your Integrated Development Environment (**IDE**) to develop code, as well as how to **write and express algorithms** in the Python Programming Language Subset (**PLS**) and in visual forms (**flowcharts**).

You'll also learn about different aspects of algorithms and programs, such as subprograms, lists, variables and different data types. Through worked examples and practical programming activities, you will have plenty of practice applying the knowledge and skills you learn in this topic to solve real world problems, in order to help you succeed in the Paper 2 exam.

1.2 Algorithms and programs

Learning outcomes

By the end of this section you should be able to:

- describe what an algorithm is
- explain what algorithms are used for
- describe what a program is
- explain what programs are used for
- express algorithms as flowcharts and written descriptions
- translate algorithms to program code
- use variables in program code
- use and describe the purpose of arithmetic operators
- use order of precedence with arithmetic operators.

Understanding algorithms and programs

An example of an algorithm

An interactive map is a useful way to find a route between two locations. This image shows a route between two cities that was calculated by a mapping program.

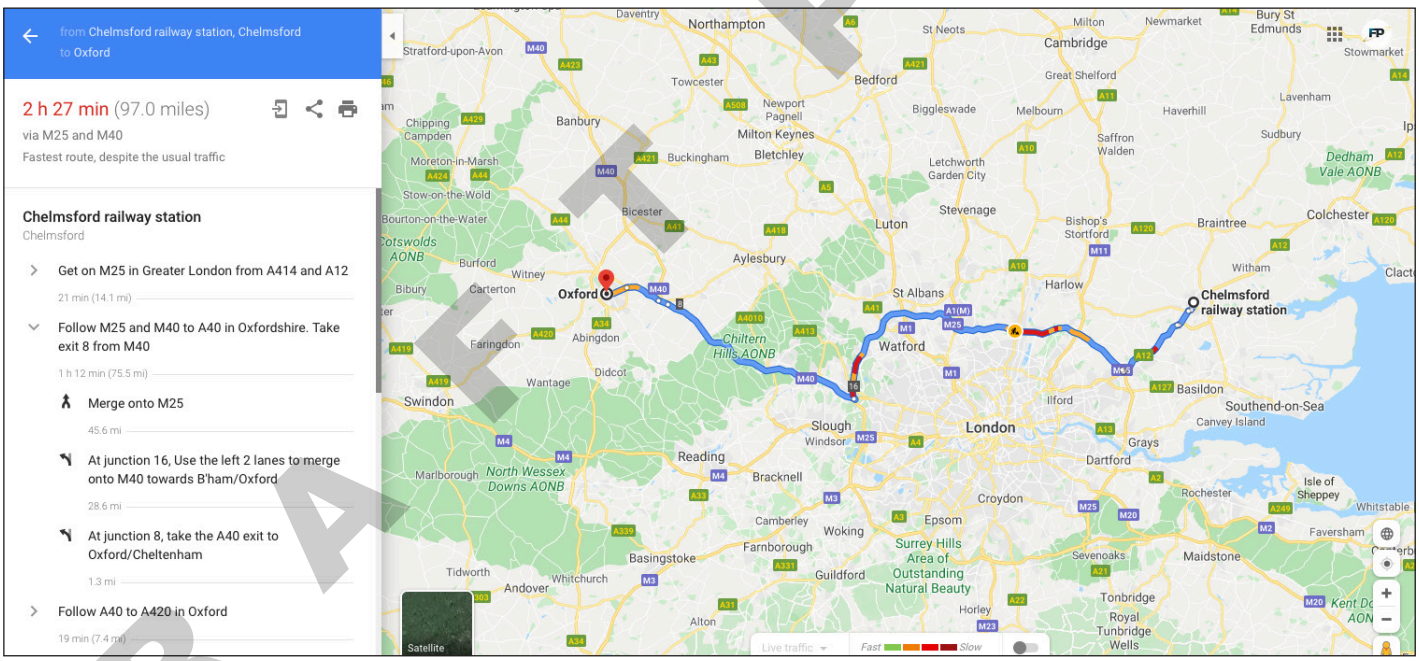


Figure 1.2.1 Written instructions to the driver are given at the left of the map

The written instructions for the driver:

- are unambiguous in telling the driver exactly what to do, like 'turn left', 'turn right' or 'continue straight'.
- are a **sequence** of steps.
- can be used again and will always provide the same result.
- provide a solution to a problem, in this case how to get from Chelmsford to Oxford.

Key term

Sequence: an ordered set of instructions.

A solution to a problem with these characteristics is called an **algorithm**. Most problems have more than one solution so different algorithms can be created for the same problem.

Successful algorithms

There are three criteria for deciding whether an algorithm is successful.

- **Accuracy** – it must lead to the expected outcome (e.g. create a route from Chelmsford to Oxford).
- **Consistency** – it must produce the same result, for the same input, each time it is run.
- **Efficiency** – it must solve the problem in the shortest possible time using as few computer resources as possible. In this example, the mapping software is replacing a manual method, and if it were no faster than looking in an atlas, it would not be an improvement on the older method. Later in the topic there is a section on algorithms used to sort and search data. Some of these algorithms are more efficient than others and will sort the data far more quickly.

The relationship between algorithms and programs

Algorithms and programs are closely related, but they are not the same. An algorithm is a detailed design for a solution; a program is the implementation of that design. A **program** is an algorithm that has been converted into program code so that it can be **executed** by a computer. A well-written algorithm should be free of logical errors and easy to code in any high-level language.

As part of this course, you will learn to write programs in Python. All high-level programming languages resemble natural human languages, which makes them easier for humans to read and write but impossible for computers to understand without the help of a translator. You will learn more about how a program written in a high-level language is translated into machine code – the language of computers – in Topic 3.

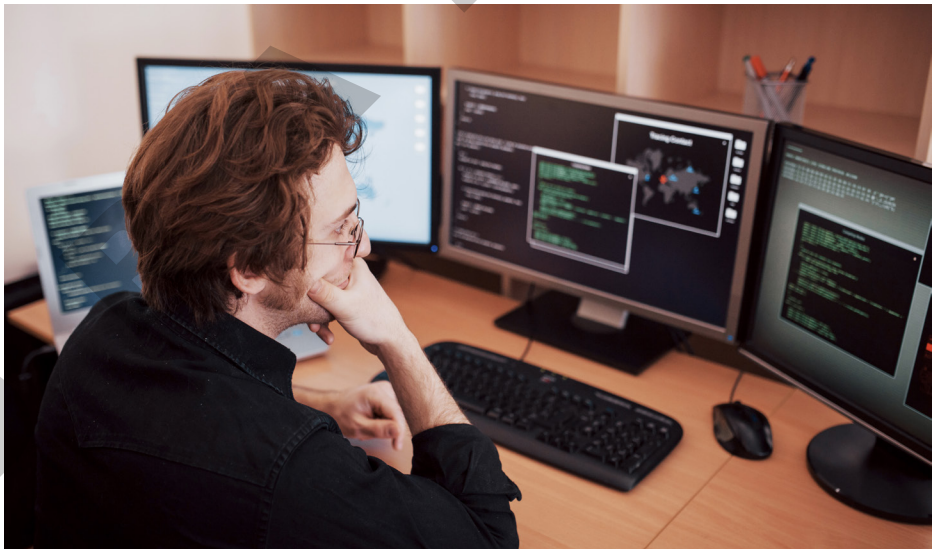
Key term

Algorithm: a precise method for solving a problem. It consists of a sequence of step-by-step instructions that solve a specific problem.

Key terms

Program: an algorithm that has been converted into program code so that it can be executed by a computer.

Execution: the process by which a computer carries out the instructions of a program.



▶ A computer programmer at work writing code

Worked example

Algorithm for making a cup of instant coffee

Fill kettle with water.

Turn on kettle.

Place coffee in cup.

Wait for water to boil.

Pour water into cup.

Add milk and sugar.

Stir.

Displaying an algorithm

We carry out many everyday tasks using algorithms because we are following a set of instructions to achieve an expected result, for example making a cup of coffee. If we have performed the task many times before, we usually carry out the instructions without thinking, but if we are doing something unfamiliar, such as putting together a flat-pack chest of drawers, then we follow the instructions very carefully.

An algorithm can be expressed in different ways.

Written descriptions

A written description is the simplest way of expressing an algorithm. Here is an algorithm describing the everyday task of making a cup of instant coffee.

Practical activity

Produce a written description of an algorithm for getting to school. It should start with leaving home and end with arriving at school..

Key terms

Flowchart: a graphical representation of an algorithm. Each step in the algorithm is represented by a symbol. Symbols are linked together with arrows showing the order in which steps are executed.

Flowcharts

Flowcharts can be used to represent an algorithm graphically. They provide a more visual display.

There are formal symbols that must be used in a flowchart – you can't just make up your own because nobody else would be able to follow your algorithm.

Here are the symbols that should be used. There should only be a single start and a single stop symbol. The other symbols can be used as many times as required to solve the problem.

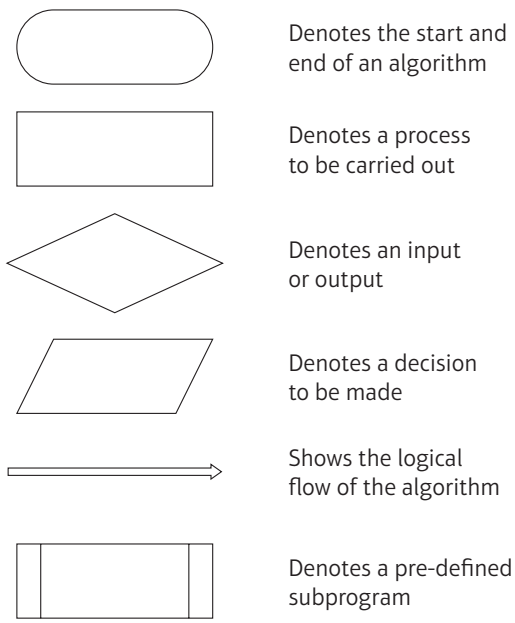
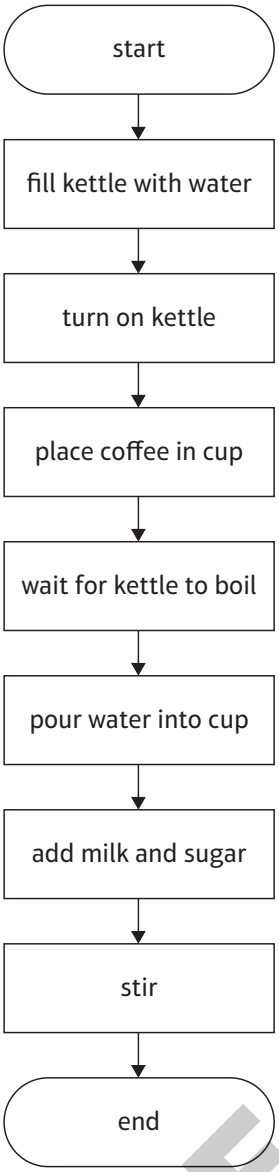


Figure 1.2.2 Flowchart symbols



This is a process (an action that has to be performed

This instruction is unambiguous – the water must be boiling

Stating 'wait for the water to heat' would be ambiguous. How hot?

Figure 1.2.3 Flowchart of an algorithm to make a cup of coffee

.This flowchart is an alternative way of depicting the algorithm that was expressed above as a written description.

Practical Activity 2

Display the 'journey to school' algorithm that you created in the previous activity as a flowchart.

The algorithms you have looked at so far are designed for humans to follow. Algorithms also form the basis of computer programs. A computer is a senseless machine that simply does exactly what it is told and follows a set of instructions, but computers can carry out these instructions far more quickly than humans. That is why they are so useful.

Computer Science in action: Chess algorithms

Algorithms for playing chess are used widely. After four moves by each opponent, there are over 288 billion possible further moves – far too many for a human to consider, but within the range of computers. This is what makes it possible for a top-level computer program to defeat a chess grandmaster.

Key terms

Program code: the implementation of an algorithm, in a human-readable form, that can be translated to a form that can be executed on a computer.

Program code

In addition to flowcharts and written descriptions, algorithms can also be expressed in **program code**. Both flowcharts and written descriptions can be translated into programming languages.

There are steps to follow that will help produce programs that execute on a computer. Depending on the complexity of the original problem, some of these may be omitted, but for most problems, these steps show a sensible approach.

Analyse – understand the problem that needs to be solved and how to determine success.

Design – develop a solution to the problem and express the solution in a written description and/or a flowchart.

Implement – translate the description of the algorithm to a programming language.

Debug and test – using tools, execute, find errors, fix errors and test different inputs.

Evaluate – make judgements about the solution based on the original requirements of the problem and consider changes related to efficiency.

Example of a simple algorithm

Here is an example of a simple algorithm. It sets two numbers, calculates the total by adding them together, and then displays the total on the screen. This algorithm does not need the user to type in any numbers.
Three different versions of this algorithm are shown: a written description; a flowchart; and program code.

Written description

Worked example

Below is an algorithm for adding two numbers.

Set the first number to 11.
Set the second number to 22.
Calculate total by adding first and second numbers.
Output total.

Flowchart

Worked example

Algorithm for adding two numbers

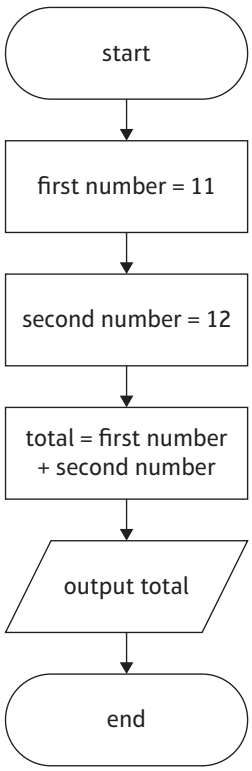


Figure 1.2.4 Flowchart showing the adding of two numbers

Program code

Worked example

Algorithm for adding two numbers

```
1   # ----------------------------------------------
2   # Global variables
3   # ----------------------------------------------
4   numFirst = 11
5   numSecond = 22
6   total = 0
7   # ----------------------------------------------
8   # Main program
9   # ----------------------------------------------
10  total = numFirst + numSecond
11  print (total)
```

In this program code, the + symbol is an **arithmetic operator** that denotes addition. The = symbol is an **assignment** operator. The **identifier** on the left is the name of a **variable**. The result of the expression on the right of the = operator is placed into the variable on the left.

Key terms

Arithmetic operator: used to perform a calculation on two numbers.

Assignment: the act of storing a value in a variable. Uses the '=' symbol. The value on the right is stored in the variable on the left.

Identifier: a unique name given to a variable or a constant. Using descriptive names for variables and constants makes code much easier to read.

Variables: name associated with containers which programmers use to hold data. The contents can be changed during execution.

Worked example

Arithmetic operators

Operator	Function	Example	Result if known
+	Addition: add the values together	`8 + 5` `myScore1 + myScore2`	13
–	Subtraction: subtract the second value from the first	`17 - 4` `myScore1 - myScore2`	13
*	Multiplication: multiply the values together	`6 * 9` `numberBought * price`	54
/	Division: divide the first value by the second value and return the result including decimal places	`13 / 4` `totalMarks / numberTests`	3.25
//	Integer division: divide the first value by the second value and return only the whole number (**integer**) part of the result	`13 // 4` `totalMarks // numberTests`	3
%	Modulus: divide the first value by the second value and return the remainder	`13 % 4` `score % buckets`	1
**	Exponentiation: raise the first value to the power of the second value	`3**4` `num1 ** num2`	

Key terms

Integer: whole numbers, both positive, negative and 0 without a fractional part, e.g. –12, 0, 1002.

Precedence: the default order in which operations are carried out. For logical operators, the order is `( )`, `not`, `and`, `or`.

Worked example

In computer programming the order of precedence (the order in which you do each calculation) is the same as in mathematics and science – **BIDMAS**.

This is how $3^2 \times 9 + (5 - 2)$ would be evaluated.

Brackets	$3^2 \times 9 + (3)$
Indices	$9 \times 9 + (3)$
Division	
Multiplication	$81 + (3)$
Addition	84
Subtraction	

To calculate $24 \div 3 - 2$, the division would be calculated before the subtraction.

$24 \div 3 = 8$

$8 - 2 = 6$

Naming conventions for variables

It is good practice in programming to adopt a consistent way of writing identifiers, including variable names, throughout programs.

A common convention is to use *camel case* for compound words (e.g. firstName, creditCard) with no space between words and the second word starting with a capital letter.

Computer science in action: Camel case

Formally known as medial capitals, camel case is used in many programming languages to enable easy reading of compound names (two or more words combined). Names of functions and variables cannot contain spaces in most programming languages, so camel case is used when a name needs to contain more than one word. Its earliest technical use is in the labelling of chemical compounds, for example NaCl.

Practical Activity

Here is a written description of an algorithm.

Set base of triangle to 22
Set height of triangle to 44
Set area of triangle to (1/2) * base * height
Output area to display

Express this algorithm in a flowchart.

Computer Science in action: Page rank algorithm

Ever wondered why Google results appear in the order they do?

There's an algorithm for that.

Google uses a selection of algorithms to select the order in which search results are displayed. The original, and best known, is PageRank – named after Google co-founder Larry Page. It works by counting the number and quality of links to a page. The idea is that the most important webpages will have the largest number of other pages that link to it. The algorithm is run a number of times for each search (known as 'iterations' or 'passes') to give the best possible answer.

PRIMM Activity ?

This activity uses the code shown here. Use it for each of the steps shown.

```
1   # -----------------------------------------------
2   # Global variables
3   # -----------------------------------------------
4   b = 22
5   h = 44
6   a = 0
7
8   # -----------------------------------------------
9   # Main program
10  # -----------------------------------------------
11  a = (1/2) * b * h
12  print ("Area is", a)
```

Predict:

What do you think the code will do? What output do you think it will produce?

Run:

Load the code into your coding environment and run it.

Did it do what you thought it would do? Did the output match your prediction? If not, how did it differ?

Investigate:

Remove the line 'b = 22'. Run the code. Carefully read the error message. What is the message trying to tell you? Put the line back in.

Change the calculation to move the '(1/2)' between the 'b' and 'h'. Run the code. What does that tell you about the use of brackets? What does that tell you about the order of arithmetic operators?

Modify:

Be sure to run the code after each change to check it still works.

Change the identifier names (variables) to be more meaningful. Remember to use camel case, if required.

Using the existing output line, add lines to display the values of each variable before the total area.

Make:

A program is needed to calculate the price of an item after adding tax. The program must store the base price of an item, the tax rate (expressed as a decimal), and the final price. The program must output the base price, the tax rate, and the final price. The formula to calculate the final price is base price x (1 + tax rate).

If the base price is 12.30 and the tax rate is 0.20, then the final price is 14.76.

Create a new program file named 'Net_Price.py'. Lay out the sections of code as shown in the examples. Write and test your program

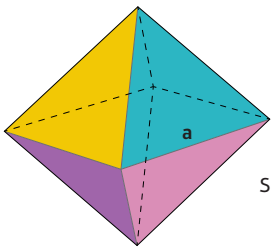
Exam-style question

1 Here is the image of a line from a program. Amend the image with circles to show the order in which the values are evaluated, using the order of precedence rules. (3 marks)

```
answer = 5 + 2 * 3 / 4 - 1
```

2 Compare the modulus (%) operator and the integer division (//) operator. (3 marks)

3 The formula for calculating the surface area of an octahedron is shown below. Construct an expression, using the arithmetic operator symbols, to translate this formula into Python. (5 marks)



$s = 2a^2\sqrt{3}$

Summary

- An algorithm is a precise method for solving a problem.
- Algorithms can be displayed as written descriptions and flowcharts.
- Algorithms can be translated into program code.
- Arithmetic operators are used in calculations and have a precedence order.
- Variables are 'containers' for storing information. The value stored in a variable can change as the program executes.
- Selecting descriptive names for identifiers makes code easier to read