Computing at School
Assessment Working Group

Algorithms, Flowcharts and Pseudocode

An overview for key stage 3 and 4 teachers

There are three associated concepts: algorithms, flowcharts and pseudocode. They present both opportunities and challenges in the teaching of programming. The purpose of this paper is to identify the key issues and the role of flowcharts and pseudocode in:

- the summative assessment of the understanding of algorithms and programming; and
- their role in formative/diagnostic assessment

It also identifies key teaching strategies, teaching resources and the challenges associated with using pseudocode and flowcharts in the teaching and assessment of computing at key stage 3 and 4.

The report is divided into:

and commentary prepared by members of the CAS Assessment working group on:

Defining terms: working definitions

A computer program follows an **algorithm** (a sequence of activities) which can be represented using **pseudocode** (code-like English text statements) or a **flowchart** (labelled symbols connected to show the flow of action).

Pseudocode and flowcharts are used to help programmers plan and describe their proposed program.

Pseudocode and flowcharts are used in assessments to determine if learners can follow the underlying algorithm or describe a system in terms of an algorithm.

CAS Assessment wg (January 2019) This document arose from discussion at a meeting of the Assessment working group in October 22019 and then subsequently online.
Editor **Andrew Csizmadia** with Phil Bagge, Pete Dring, Cynthia Selby and John Woollard
The importance of unnamed contributors is acknowledged.

Defining terms: what is pseudocode?

There is much debate around the term pseudocode and its role in the curriculum and in assessment.

The term pseudocode is a requirement of the computer science GCSE subject content, “take a systematic approach to problem solving including the use of decomposition and abstraction, and make use of conventions including pseudocode and flowcharts” (DfE, 2015) and is therefore a necessary part of the experience of students. Confusion has arisen because the awarding organisations have used the term pseudocode to describe the syntax and format that examiners will use in the examination papers to present algorithms.

CAS recommends that the term pseudocode is ONLY used in association with freely written, no fixed format or syntax, text used by students (and computer programmers) to construct or communicate the algorithm of their planned program.

There is no definitive pseudocode

Pseudocode is whatever the learner or whatever the programmer makes up to help them conceptualise the algorithm/program for themselves OR to express the algorithm/program to someone else (a known audience)***.***

Considering a reference language

There are good reasons for examiners, resource authors and teachers being consistent in the presentation of algorithms. Learners will focus on the constructs of the algorithm rather than having first to decipher the presentation. There are good reasons for establishing a single standard of presentation of algorithms across all awarding organisations (continuity, standardisation, wider use of resources). To enable examiners to provide a fair and consistent format for ***setting questions,*** each awarding organisations has chosen to create their own format and syntax of presentation. Confusingly, that has been called pseudocode - *it might better be referred to as an examination reference language.*

A strong argument is made in favour of only using correct computer programming code in the examination papers to represent algorithms. That is, it is unnecessary to introduce an examination reference language. This would avoid students having to unnecessarily learn the extra language. The examination papers will need to include the algorithms in the different programming languages specified or accepted by the awarding organisations. The alternative view is that examination papers should be programming language neutral and questions should examine a candidates understanding of the constructs and functions of any programming language. Within CAS there is NOT a consensus over whether the examination papers should have algorithms presented in an examination reference language or presented as correct programming language code. It is suggested that the existence of a reference language in the examination specification will add to the learners' burden of having to learn another language which cannot be tested by executing the text. It is acknowledged that the awarding organisations will interpret the DfE and Ofqual requirements in the best interests of their examinations.

The specifications set by awarding organisations in England describe how "pseudocode" extracts that are in examination questions will appear. The awarding organisations do NOT specify how a candidate needs to answer an examination question. Currently, the awarding organisations assure that a candidate's answers do not have to be in a particular format and with a particular syntax. By encouraging candidates to respond in a similar style of pseudocode, they may find learning pseudocode more straightforward due to repetition and consistency of style.

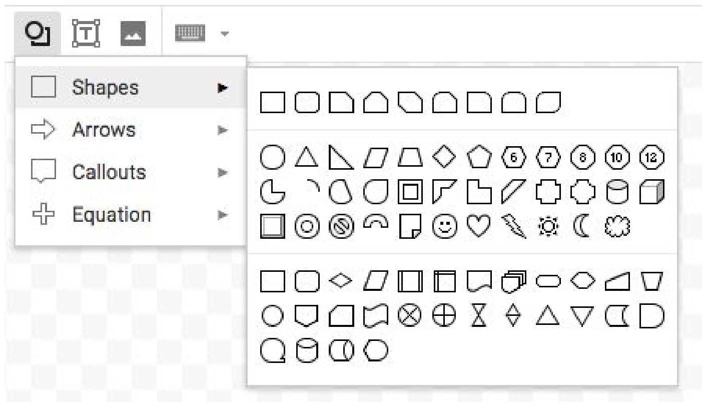
The key points relating to pseudocode and an examination reference language are:

- The word pseudocode should never be used with regard to the examination process – pupils should know of it and use it within their classroom/programming experience.
- Awarding organisations should specify the presentation (format and syntax) of algorithms in examination papers but NOT use the term pseudocode.
- Awarding organisations might specify a reference language or might use one or more programming languages when writing examination papers.
- CAS recommends that candidates' examination answers do not have to be in a particular format or with a particular syntax to obtain full marks; it should be the meaning/logic of a candidate's answer that gains the mark.
- There is NOT a consensus whether algorithms on the examination papers should be presented in an examination reference language or presented as correct programming language code.

Defining terms: what is a flowchart?

There is no single definitive flowchart specification.

Flowcharts are visual representations of algorithms. Many symbols are common and represent the constructs of computer of programming of sequence, selection and iteration with input, output and process and representations of data storage. Most word processors and presentation software contain the standard shapes. For example, Google Docs



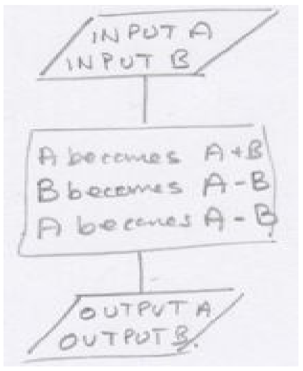
Examination questions might contain flowcharts or the questions might require flowcharts to be drawn as part of the answer. Plastic flowchart templates are available and make for faster drawing when designing algorithms. Not all Awarding Organisations allow them to be taken into examinations. Some specifications include the symbols that they expect candidates to use, for example, Eduqas, OCR p40,

Example pseudocode

```
REPEAT
  OUTPUT 'What is the best subject you take?'
  INPUT user inputs the best subject they take
  STORE the user's input in the answer variable
  IF answer = 'Computer Science' THEN
          OUTPUT 'Of course it is!'
  ELSE
          OUTPUT 'Try again!'
UNTIL answer = 'Computer Science'
```

(BBC Bitesize, https://www.bbc.com/bitesize/guides/z3bq7ty/revision/2)

Example flowchart



Pseudocode and flowcharts can be used in four important ways in supporting the teaching of programming.

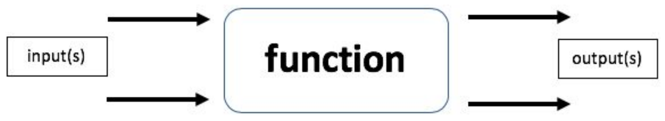
1. Pseudocode and flowcharts can be used to express algorithms (including sorting and searching) in a common format. Either, the algorithms will be more easily understood because they are being represented in another familiar format OR flowcharts and pseudocode will become more familiar through use with algorithms the learners already know.
2. Pseudocode and flowcharts can be used to support the learner's design and solution to a problem. They are more formal than "talking through their idea" but less formal that specifying or producing the exact code.
3. Pseudocode and flowcharts can be used to describe programs that learners are to write in computer code. They provide a level of abstraction closer to the computer code than, say, the spoken word, for example, "write a program that inputs a birthdate and outputs an age" or a scenario, for example, "write a program to test reaction time of someone arrested for driving under the influence".
4. Pseudocode and flowcharts can be used as part of a learner's programmer's toolkit when debugging and refining a program which does not produce the expected results.The learner may need to examine carefully the pseudocode and/or flowchart they created to check that they express a solution correctly. It may be a case that pseudocode and/or flowchart may need to be amended. The process of cross-checking between the computer code and the

pseudocode/flowchart can help the programmer identify errors in a similar way to the rubber duck debugging technique.

Teaching techniques

The following approaches can be adopted to develop learners' confidence, capability and proficiency in creating algorithms using either flowcharts or pseudocodes.

Input-Process-Output (IPO) Model is a graphical representation of the processes involved in a task.



It can be used to remove superficial detail in a problem statement to that the essential information remains with regard to what input(s) is/are required to generate a specific output and the processes required.

Parson's Problems requires learners to sort either blocks or individual lines of pseudocode into the correct order. It is an effective collaborative learning technique or a way of identifying a learner's misconceptions. The algorithm can be simple but it requires learners to think about the algorithm - the sequence. More complex problems can include repetition (loops) and selection (IF statements). Parson's problems can be based on computer code or pseudocode. In a similar way, flowcharts can be dissected and then learners reassemble to form the correct algorithm.

Peer Instruction (PI) is an active-learning technique that utilises Multiple Choice Single Answer (MCSA) questions. It highlights individual learner's alternative conceptions and misconceptions. Peer instruction provides learners with the opportunity to practice talking about computer science and it is regarded as an efficient way to draw out learner's misconceptions, misalignments or alternative concepts.

PREP is an approach used when developing functions. The acronym stands for:

Parameters	Define input(s) with meaningful name(s)
Return	Output(s)
Example	Test run your function with an example
Pseudocode	Describe the logic of the internal works of the function

Guidance and resources for using peer instruction can be found in the peer instruction blog and for computer science in particular at Plan-C and Peer Instruction for Computer Science.

Plan first - code later is a principle advocated by many teachers. It agrees with approaches the pupils adopt in other spheres of learning.

Paper and pencil first is a principle advocated by many teachers. Before ever approaching the programming environment, the learner must first express their ideas and understanding on paper followed by describing their potential solution to another (rubber ducking).

Typing pseudocode first in the programming environment is alternatively advocated. The pupil uses the IDE as a text editor to describe the algorithm of their potential program. The statements then become the annotation of their code. Some examples,

```
// input 2 values
// check validity
// calculate the product
// output to screen
```

```
% INPUT NAME
% LOAD CSV
% SEARCH FOR NAME
% OUTPUT FULL DETAILS
```

```
# repeat 10 times
# present question
# check response
# give score
```

Puzzle-based Learning is an approach in which learners are presented with either a puzzle or puzzle-based problem to solve using algorithmic thinking. This approach supports the learners' design and analysis of algorithms.

Unplugged Computing are kinesthetic activities, for example, learners provide the instructions to perform a task such as Phil Bagge's Jam Sandwich Robot or record the steps to perform a particular task illustrated by various dance routines AlgoRythmics.

Visualisation of Algorithms is an interactive and kinaesthetic approach that provides learners with an opportunity to interrogate and step through an algorithm. Further examples include:

- **Jam Sandwich Robot** http://code-it.co.uk/unplugged/jamsandwich
- **AlgoRythmics** https://www.youtube.com/user/AlgoRythmics
- **Algorithms Animations** https://www.toptal.com/developers/sorting-algorithms
- **VisuAlgo** https://visualgo.net/en
- **Just Dance** http://www.digitalschoolhouse.org.uk/workshop/just-dance-algorithm
- **Get with the Algo-rhythm** http://www.digitalschoolhouse.org.uk/workshops/get-algo-rhythm

Starting with everyday experience where possible. Pupils need to relate programming concepts to things in their current scheme of understanding. Where there is this cognitive overlap, comprehension of new ideas is increased. Writing and role playing everyday examples of conditional selection set out in a pseudocode layout increases pupils' understanding of the ideas in subsequent pseudocode planning and programming activities. For example,

```
If you are a boy then
   Floss
Else
    Dab
```

(Floss and dab are popular dances)

Pupils, as early as Year 4 can be given points for writing a correct expression, extra points for putting each element on a different line and extra points for correct indentation. This aligns with English grammar and syntax expectations. When planning programming projects they can use the same pseudocode notation.

Tracing through algorithms, flowcharts and pseudocode helps develop learners' understanding of and fluency in creating - it is the 'reading' complement to 'writing'.

Tools and resources

The following tools and resources will help learners to develop confidence and competence in creating algorithms using flowcharts and pseudocode. The resources have been recommended by teachers but many hundreds are available.

The close relationship between pseudocode and program code is demonstrated using resources produced by David Whale. The Python to AQA-style pseudocode convertor can be used to demonstrate the relationship between computer code and pseudocode.
https://github.com/whaleygeek/pc_parser

Knowledge Organiser is not only a tool for revising for examinations but it can also be used to define the minimum (breadth and depth) a learner needs to know.
https://www.tes.com/teaching-resource/gcse-computer-science-module-quiz-programming-basics-pseudocode-linked-to-knowledge-organiser-11865453

Pseudocode Challenges are a set of scaffolded tasks; learners are required to create an algorithmic solution either visually as a flowchart or textually using pseudocode. Examples include: pseudocode challenges, Edexcel Programming Challenges, STEM pseudocode challenges, OCR GCSE Computer Science Algorithm Challenge Booklet.

Flowgorithm is a graphical authoring tool which allows learners to create and execute programs using flowcharts. The approach is to emphasis the algorithm rather than the syntax of a specific programming language. http://www.flowgorithm.org

Flowol is a popular application for developing flowcharts within interesting scenarios (mimics) of a control technology nature. Free trial from http://www.flowol.com/Flowol4Trial.aspx

Pseudocode Challenge: Edexcel version and OCR version are self-marking exercises to check and challenge comprehension of concepts, keywords and stepping through algorithms in pseudocode.

Challenges and teaching advice

Many of the challenges, reported in the classroom and detailed below, are not issues with the pseudocode itself. The structures associated with sequence, repetition and selection in programming languages differ from each other. Conflict may occur when students believe that either a) the rules of pseudocode are exactly the same as the programming language that they learn or b) their pseudocode will be universally understood when using language specific constructs. For example, the rules around interpreting pseudocode are similar to those associated with strongly typed, structured, and procedural programming languages. The rules associated with Python, a popular language used in schools, are not necessarily the same in all aspects.

Variables present a challenge In Scratch, Blockly, and variants, such as Crumble, a variable is set or changed.

Set VAR friendsname to John
Say VAR friendsname, how are you today? produces “John, how are you today?”

Set VAR a to 3

Care is needed in introducing particular syntax/keywords into pseudocode writing. There are different ways in which, for example, assignment is written. It is best to adopt the style the candidates will meet in the examination. That style varies between Awarding Organisations:

```
a ← 3 (AQA)
SET a TO 3 (Edexcel)
a = 3 (OCR)
```

Pseudocode, Python, VB.NET Quick Reference is a resource that makes the connection between code written in Python and VB.NET and the different flavours of pseudocode described by the Awarding Organisations https://community.computingatschool.org.uk/resources/5255/single.

- Teaching advice
  Adopt the relevant Awarding Organisation's syntax when presenting algorithms to the students.

Conditional (selection) statements also present a particular challenge for some learners when writing and then interpreting their pseudocode. For example,

```
IF X=1 THEN OUTPUT="one"
IF X=2 THEN OUTPUT="two"
ELSE OUTPUT="other"
PRINT OUTPUT
```

appears straightforward to understand but the subsequent computer coding can be difficult; in many languages, if X is 1 then "other" would be printed. A programming language has rules about parsing these constructs. How pseudocode is parsed (understood) depends upon the reader as much as the writer.

- Teaching advice
  Always present algorithms using a consistent syntax to encourage the students to write consistently.

Pseudocode is richer in constructs than some programming languages used in the classroom. For example, pseudocode can represent a repeat/until or do/while construct. For example:

```
REPEAT
   x  = x  + 1
   PRINT x
UNTIL x > 10
```

keep printing the counter until the counter is bigger than 10

```
DO
   x = x + 1
   PRINT x
WHILE x < 10
```

keep adding to x and printing while x is less than 10

These constructs are not supported in, for example, Python. This means that pupils with no experience of pseudocode representation of algorithms may not have the opportunity to see these constructs in solutions or trace their behaviours in a machine. However, they can be traced by hand by going through the pseudocode.

- Teaching advice
  Be careful when presenting algorithms that they do not contain constructs that are not available in the programming language being used by the learners.

If teaching programming constructs, be careful to identify those constructs available to them in the language they are/will be using.

Review the Awarding Organisations specification for the programming techniques they are expected to learn - and introduce these techniques to candidates first. Then investigate whether these constructs are supported in the programming language you are using.

Keyword blocking. Many languages use keywords at the start and end of blocks, for example,

```
FOR i = 1 TO 10
NEXT i

IF X = 2 THEN PRINT X
ELSE PRINT 3
END IF
```

whereas some programming languages are blocked by indentation, for example,

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
  if x == "banana":
    break
```

When writing pseudocode, programmers need not use the ending block keywords because they ‘know’ where the end occurs. As a learner’s familiarity with the coding process develops then the consistency of their pseudocode, perhaps to reflect a particular programming language, will develop. If consistent indentation is used, then the reader may be able to better understand the meaning. If there is inconsistent indentation and missing keywords, then the intent becomes less clear to a reader.

- Teaching advice
  When presenting algorithms, support the learners by using a format that is consistent with the programming language in which they are learning to code.

Formative, diagnostic and self assessment

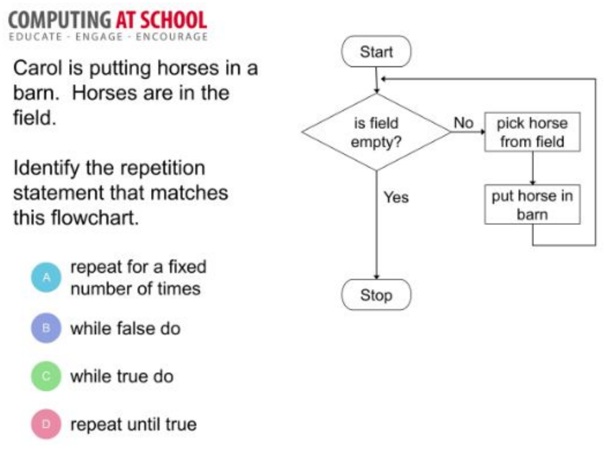
Flowcharts and pseudocode are used as part of the assessment tool for assessing an understanding of algorithms and therefore computational thinking. The ability to trace (or dry run) through a flowchart or pseudocode is an indicator of being able to follow an algorithm. If the assessment is used to assess algorithmic thinking then consistency of presentation in formative assessments is important.

The Awarding Organisations specifications can guide the presentation of the pseudocode, even for key stage 3 activities. Alternatively, where the resources are also to encourage or develop the concept that there are no fixed rules to pseudocode, then presenting the learners with a range of different presentation styles is beneficial. Learners should be encouraged to design and express algorithms in any way they see fit.

Formative assessment resources for flowcharts and pseudocode include:

Flowgorithm can be used as a self-assessment tool for flowcharts as it can run through the flowchart automatically step by step with test data. This eliminates the need for the learner to carry out a 'dry run'. http://www.flowgorithm.org

Project Quantum https://community.computingatschool.org.uk/resources/4382 has a developing set of quizzes. Questions are based on flowcharts or pseudocode. An example from the Year 8 baseline collection is shown here:



COMPUTING AT SCHOOL
EDUCATE · ENGAGE · ENCOURAGE

Carol is putting horses in a barn. Horses are in the field.

Identify the repetition statement that matches this flowchart.

- A repeat for a fixed number of times
- B while false do
- C while true do
- D repeat until true

Pete Dring's online exercises http://tools.withcode.uk/ks4pseudo give valuable feedback about those in the class that can read pseudocode algorithms in a format like,

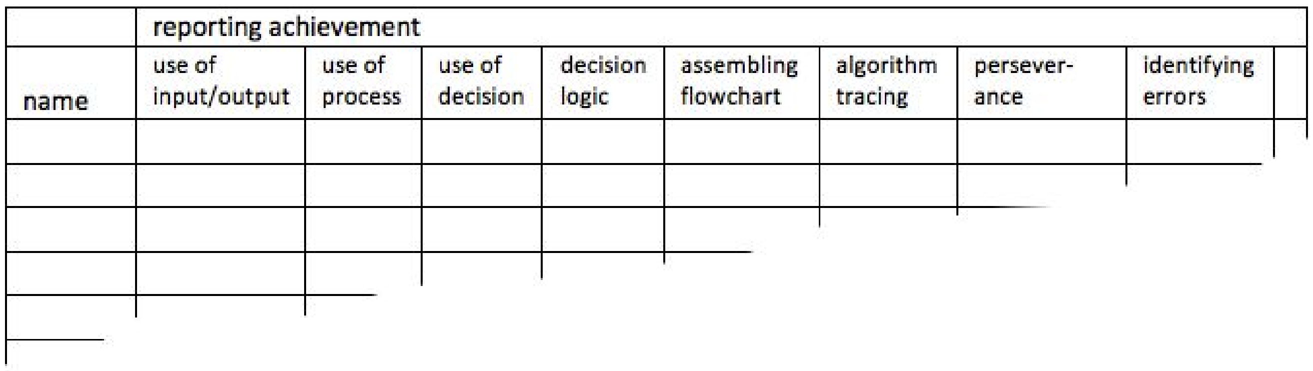
```
SET hungry TO false
SET exercise TO 2 * 6
IF exercise > 10 THEN
   SET hungry TO true
END IF
SET food TO "toast"
IF hungry = true THEN
   SET food TO "pasta"
END IF
SEND food TO DISPLAY
```

Students work sequentially through variables, iteration, selection, subroutines, arrays and string handling.

Flowol can be used to set open-ended tasks:

- ○ divide the class into 2s or 3s, each with a computer;

- ○ give them a simple mimic to explore;
- ○ circulate around the room answering questions and offering advice;
- ○ use an achievement sheet to record successes.
- ○ after the activity, analyse the sheet to determine the focus of future lesson activities.



	reporting achievement								
name	use of input/output	use of process	use of decision	decision logic	assembling flowchart	algorithm tracing	persever-ance	identifying errors	

Summative assessment

The use of flowcharts and pseudocode is relevant and necessary in the formal assessments of the understanding of algorithms and computer code. The documents made available by the Awarding Organisations are intended to:

- direct examiners when writing questions;
- inform book and teaching resource publishers; and
- guide teaching when talking about, demonstrating and creating resources that use pseudocode.

The Awarding Organisations in England have developed pseudocode guidance independently. It is therefore important that pupils are presented with pseudocode that is similar to the pseudocode they will meet in the examinations and in the textbooks associated with the course they are following. The sources are:

AQA – Teaching Guide – Pseudo-code
"*It is not the intention that candidates should use this style of pseudo-code in their own work*"
https://filestore.aqa.org.uk/resources/computing/AQA-8520-TG-PC.PDF

Edexcel Pseudo-code command set
"*students may answer questions using any valid method*"
http://tools.withcode.uk/ks4pseudo/media/edexcel_pseudocode.pdf

OCR GCSE Computer Science Pseudocode Guide
"*the following guide shows the format pseudocode will appear in the examined components*"
https://www.ocr.org.uk/Images/202654-pseudocode-guide.pdf

OCR AS and A Level Computer Science Pseudocode Guide
"*learners are not expected to memorise the syntax of this pseudocode and when asked may provide answers in any style of pseudocode they choose*"
https://www.ocr.org.uk/Images/260952-pseudocode-guide.pdf

By using the pseudocode syntax and keywords of the Awarding Organisations, the learning resources are better preparing the learners to be able to read and understand the examination questions.

In Scotland, the examinations are set by a single body, the Scottish Qualifications Authority (SQA). The code cited in examination questions is written in the SQA Reference Language (also called Haggis). The correctness of code can be verified by using Richard Connor's SQA RL checker. This helps to avoid papers which include SQA RL examples containing errors where the fragment of code published does not correspond with the formally defined language. SQA RL is as a single coherent programming language, with a formal imperative specification and can be used to exemplify procedural and object-oriented styles.

Professional Learning

Many of the resources used with learners are also useful for professional learning. The CAS Tenderfoot materials are purposely designed to support key stage 3 teachers.

The following resources are helpful in developing understanding of algorithms, flowcharts and pseudocode:

CAS Tenderfoot Unit 1: Laying Firm Foundations: A Conceptual Approach to Programming
https://www.computingatschool.org.uk/custom_pages/295-unit_01

CAS Tenderfoot Unit 2: How Computer Do Stuff: Algorithms + Data Structures = Programs
https://www.computingatschool.org.uk/custom_pages/296-unit_02

CAS Tenderfoot Unit 3: Clever Stuff for Common Problems: Going Beyond Simple Algorithms
https://www.computingatschool.org.uk/custom_pages/297-unit_03

Pete Dring's guide to pseudocode and programming constructs is a useful starting point for Python and VB.NET users http://tools.withcode.uk/quickref

CraignDave's Guide to pseudocode for the GCSE NEA (now 'Programming Project')
http://www.ccsict.com/nea/ssp/Guide to pseudocode.pdf

Khan Academy – Introduction to Algorithms
https://www.khanacademy.org/computing/computer-science/algorithms

Quickstart – How do we think about problems so that computers can help?
https://community.computingatschool.org.uk/files/8261/original.pdf

A fun way of introducing pseudocode with MINIONS created by Pam Jones
https://community.computingatschool.org.uk/resources/2321

Michael Jones series of short videos
https://community.computingatschool.org.uk/resources/5892

Algorithms in pseudocode and flowcharts - a video resource
https://www.youtube.com/watch?v=-TSOy_-8h3k 4:11 mins

David Malan explains what's an algorithm?
https://www.youtube.com/watch?v=6hfOvs8pY1k 4:57 mins

CraignDave
https://www.youtube.com/watch?v=wvAblqsEj54 11:37 mins