The following methodology for teaching the art of Programming in Computing Science is based on guidance from Building the Curriculum 5 and CfE briefing 2.

“In the words of the HMIE publication *Improving Scottish Education:* *ICT in Learning and Teaching* (2007),

‘… staff in pre-school centres and in primary schools recognised that learners developed awareness of the world in which they live more effectively when this included engagement with the world through ICT.’

The CfE Technology Principles and Practice paper specifies the following:

Well-designed practical activities in the technologies offer children and young people opportunities to develop:

* curiosity and problem solving skills, a capacity to work with others and take initiative
* planning and organisational skills in a range of contexts
* creativity and innovation, for example though ICT and computer aided design and manufacturing approaches
* skills in using tools, equipment, software and materials
* skills in collaborating, leading and interacting with others
* critical thinking through exploration and discovery within a range of learning contexts
* searching and retrieving information to inform thinking within diverse learning contexts
* making connections between specialist skills developed within learning and skills for work
* evaluating products, systems and services
* presentation skills.

Through the ‘Bearsden Primary Programming Guide and Programme of Work’ and also more directly in the ‘Bearsden Primary Computational Thinking Programme of Work’ documents we aim to develop each of these skills through a wide range of programming tasks which not only develop ICT skills but help to establish fundamental computational thinking skills which will open the door to new ways of solving problems for children in computing science.



The course will cover the 6 concepts and 5 approaches of computational thinking.

## Why is computational thinking important?

Computational thinking and the concepts behind it, form the basis for much of computer science. Computer scientists are interested in finding the most efficient way to solve problems. They want to find the best solution that solves a problem correctly in the fastest way and using the least amount of resources (time / space).

* Is this the most efficient way to solve the problem?
* Is this the fastest way?
* Does it require the least amount of resources?
* Does it solve the problem and give the right answer?
* Can it be used to solve other problems?

### What can you do with computational thinking?

Although computational thinking describes the sort of thinking that computer scientists and software developers engage in, plenty of other people think in this way too, and not just when it comes to using computers. The thinking processes and approaches that help with computing are really useful in many other domains too.

For example, the way a team of software engineers go about creating a new computer game, video editor or social networking platform is really not that different from how you and your colleagues might work together to put on a school play, or to organise an educational visit.

In each case:

* you take a complex problem and break it down into smaller problems
* it’s necessary to work out the steps or rules for getting things done
* the complexity of the task needs to be managed, typically by focusing on the key details
* the way previous projects have been accomplished can help.

## What does computational thinking look like in class?

There are many ways to develop computational thinking in school beyond the technology outcomes, but as pupils learn to use these in their computing work, you should find that they become better at applying them to other work too.

You will already use computational thinking in many different ways across your school work.

* When your pupils write stories, you encourage them to plan first: to think about the main events and identify the settings and the characters.
* In art, music or design and technology, you will ask pupils to think about what they are going to create and how they will work through the steps necessary for this, by breaking down a complex process into a number of planned phases.
* In maths, pupils will identify the key information in a problem before they go on to solve it.

[](http://barefootcas.org.uk/wp-content/uploads/2014/04/Henley-Primary-School-Unplugged-Building.jpg)

Pupils use a great deal of computational thinking as they consider the best way to build a robot.

**THE 6 CONCEPTS OF COMPUTATIONAL THINKING**

1. **– Logic**

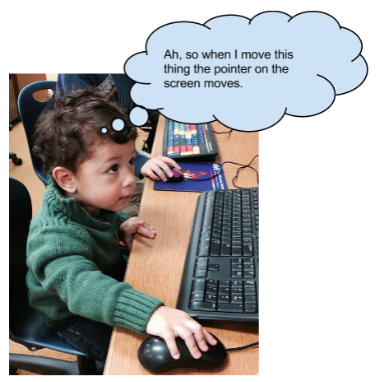
**What is logical reasoning?**

Logical reasoning helps us explain why something happens.

If you set up two computers in the same way, give them the same instructions (the [program](http://barefootcas.org.uk/barefoot-primary-computing-resources/concepts/programming/)) and the same [input](http://barefootcas.org.uk/programme-of-study/work-various-forms-input/inputs/), you can pretty much guarantee the same [output](http://barefootcas.org.uk/programme-of-study/work-various-forms-output/outputs/).

Computers don’t make things up as they go along or work differently depending on how they happen to be feeling at the time. This means that they are predictable. Because of this we can use logical reasoning to work out exactly what a program or computer system will do.

Children quickly pick this up for themselves: the experience of watching others and experimenting for themselves allows even very young children to develop a mental model of how technology works. A child learns that clicking the big round button brings up a list of different games to play, or that tapping here or stroking there on the screen produces a reliably predictable response.

[](http://barefootcas.org.uk/wp-content/uploads/2015/01/Image_Logic_pupil-mouse.png)

Pupils use logical reasoning to develop their understanding of how technology works.

This process of using existing knowledge of a system to make reliable predictions about its future behaviour is one part of logical reasoning. At its heart, logical reasoning is about being able to explain why something is the way it is. It’s also a way to work out why something isn’t quite as it should be.

**Why is logical reasoning important?**

Logic is fundamental to how computers work: deep inside the [computer’s](http://barefootcas.org.uk/barefoot-primary-computing-resources/concepts/computer-systems/) central processing unit (CPU), every operation the computer performs is reduced to logical operations carried out using electrical signals. It’s because everything a computer does is controlled by logic that we can use logic to reason about program behaviour.

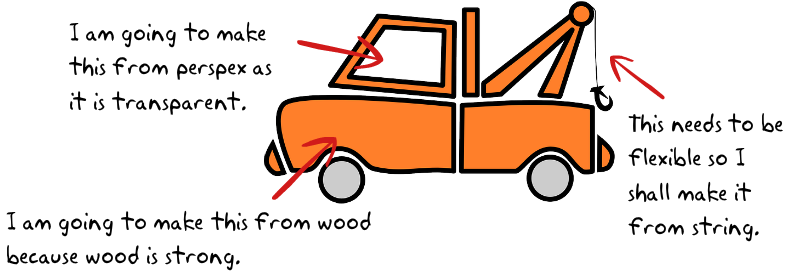
Software engineers use logical reasoning all the time in their work. They draw on their internal mental models of how computer hardware, the operating system (such as Windows 8, OS X) and the programming language they’re using all work, in order to develop new code that will work as they intend. They’ll also rely on logical reasoning when testing new software and when searching for and fixing the ‘bugs’ (mistakes) in their thinking (known as [debugging](http://barefootcas.org.uk/barefoot-primary-computing-resources/computational-thinking-approaches/debugging/)) or their coding when these tests fail.

**What does logic look like in class?**

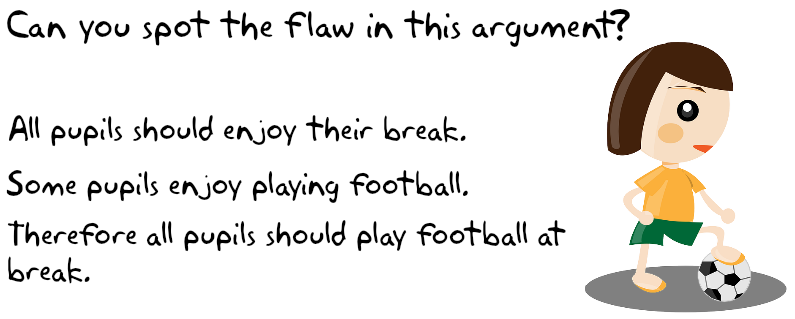
Logical reasoning across the curriculum

There are many ways that children will already use logical reasoning in their computing lessons and across the wider curriculum.

* In English, pupils might explain what they think a character will do next in a novel, or explain the character’s actions in the story so far.
* In science, pupils should explain how they have arrived at their conclusions from the results of their experiments.
* In history, pupils should discuss the logical connections between cause and effect; they should understand how historical knowledge is constructed from a variety of sources.
* In design and technology, pupils reason what material is best suited to each part of their projects.
* In philosophy for children sessions, pupils can use logical reasoning to analyse arguments.

[](http://barefootcas.org.uk/wp-content/uploads/2014/06/Image_logic_DT.png)

Pupils explain reasons for their choice of materials in design and technology projects.

[](http://barefootcas.org.uk/wp-content/uploads/2014/06/Image_logic_P4C.png)

Pupils can use logical reasoning to analyse arguments in philosophy for children sessions.

Specific tasks to improve skills in Logical reasoning can be found here:

<http://barefootcas.org.uk/barefoot-primary-computing-resources/concepts/computational-thinking/>

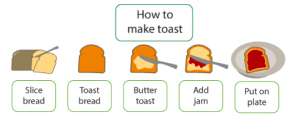
**2. Algorithms**

**What are algorithms?**

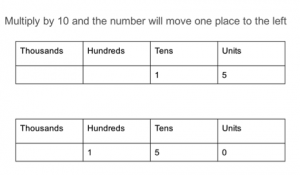
An algorithm is a sequence of instructions or a set of rules to get something done.

You probably know the fastest route from school to home, for example, turn left, drive for five miles, turn right. You can think of this as an ‘algorithm’ – as a sequence of instructions to get you to your chosen destination. There are plenty of algorithms (i.e. routes) that will accomplish the same goal; in this case, there are even algorithms (such as in your satnav) for working out the shortest or fastest route.

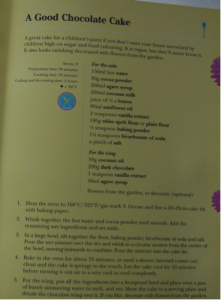
Algorithms are written for a human, rather than for a computer to understand. In this way algorithms differ from programs.

[](http://barefootcas.org.uk/wp-content/uploads/2014/06/making-toast-algorithm.png)

A sequence of instructions – an algorithm – for how to make toast

[](http://barefootcas.org.uk/wp-content/uploads/2014/06/multiply-by-10-algorithm.png)

A set of rules, or an algorithm for multiplying by 10

[](http://barefootcas.org.uk/wp-content/uploads/2014/06/chocolate-cake-algorithm.png)

In the task of how to make a cake, a recipe gives a human a step-by-step sequence of instructions for how to make and bake the cake.

**Why are algorithms important?**

Computer scientists strive to find the most effective and efficient algorithms, that is those that solve a problem in the quickest time, using the least resources (memory or time) or in the most effective way (getting the correct or closest to the correct answer).

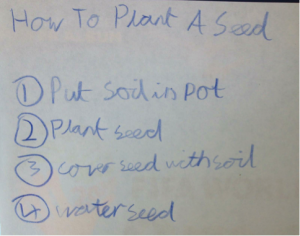
Search engines such as Bing or Google use algorithms to put a set of search results into order, so that more often than not, the result we’re looking for is at the top of the front page. Your Facebook news feed is derived from your friends’ status updates and other activity, but it only shows that activity which the algorithm (EdgeRank) thinks you’ll be most interested in seeing.

The recommendations you get from Amazon, Netflix and eBay are algorithmically generated, based in part on what other people are interested in.

**What do algorithms look like in the class?**

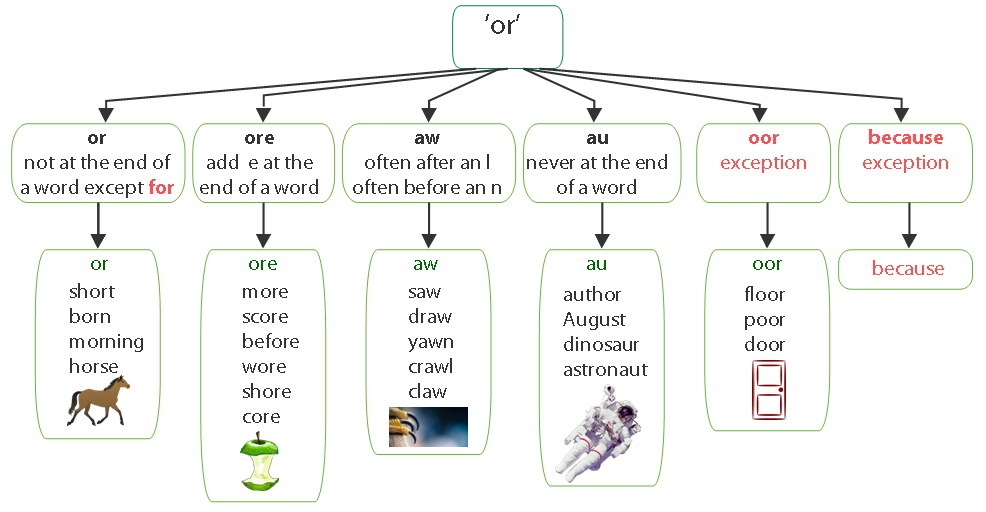
Helping pupils to get an idea of what an algorithm is needn’t be confined to computing lessons. You and your pupils will already use algorithms in many different ways across the school.

* + A lesson plan can be regarded as an algorithm for teaching a lesson.
  + There will be a sequence of steps pupils follow for many activities, such as getting ready for lunch or going to PE.
  + In cookery, we can think of a recipe as an algorithm.
  + In English, we can think of instructional writing as a form of algorithm.
  + In science, we might talk about the method of an experiment as an algorithm.
  + In maths, your approach to mental arithmetic (or many computer-based educational games) might be an implementation of a simple algorithm.
  + An example of this might be:
    - repeat 10 times
      * ask a question
      * wait for a response
      * provide feedback on whether the response was right or wrong.

[](http://barefootcas.org.uk/wp-content/uploads/2014/06/growing-seed-algorithm.png)

A set of instructions for how to plant a seed

[](http://barefootcas.org.uk/wp-content/uploads/2014/06/storyboard-algorithm.png)A storyboard is a sequence of instructions for writing a story or making a film

[](http://barefootcas.org.uk/wp-content/uploads/2014/09/activity_spelling_rules_1_or_diagram.png)Spelling rules for the ‘or’ phoneme. Images from pixabay Public Domain CC0

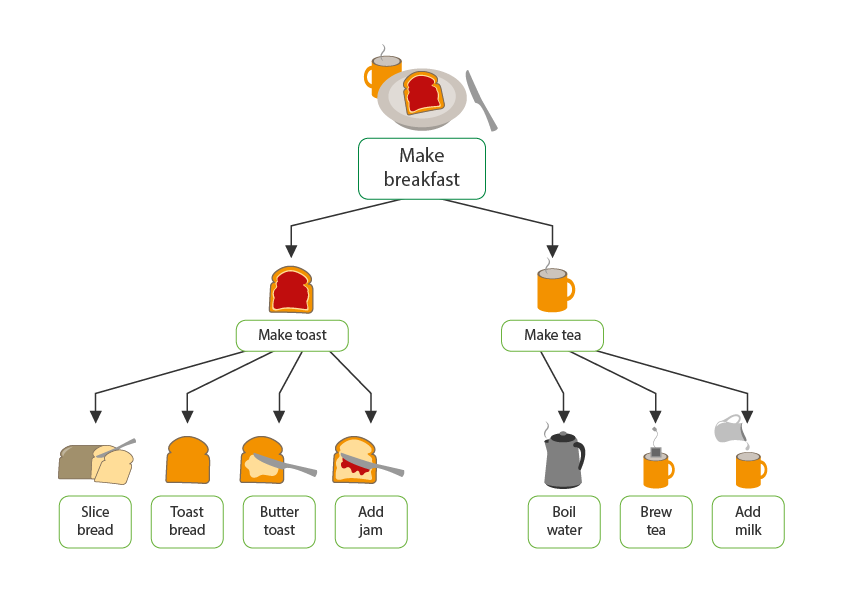
1. **Decomposition.**

## What is decomposition?

The process of breaking down a problem into smaller manageable parts is known as decomposition. Decomposition helps us solve complex problems and manage large projects.

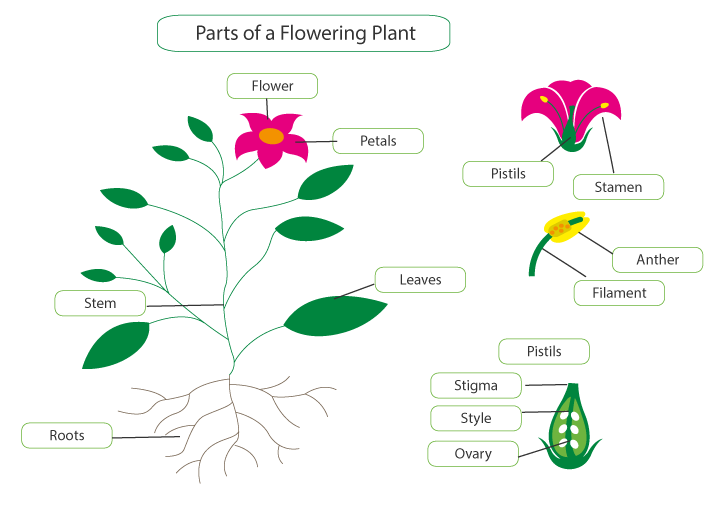
This approach has many advantages. It makes the process a manageable and achievable one – large problems are daunting, but a set of smaller, related tasks are much easier to take on. It also means that the task can be tackled by a team working together, each bringing their own insights, experience and skills to the task.

The problem of making breakfast can be decomposed into a number of tasks:

[](http://barefootcas.org.uk/wp-content/uploads/2014/06/Breakfast.png)

Two people could make this breakfast at the same time, one could make tea and one could make toast.

Decomposition is particularly important if we are trying to understand things that are complex. Sometimes we break parts down further.

[](http://barefootcas.org.uk/wp-content/uploads/2014/06/Plant-Labelled-Diagram.png)

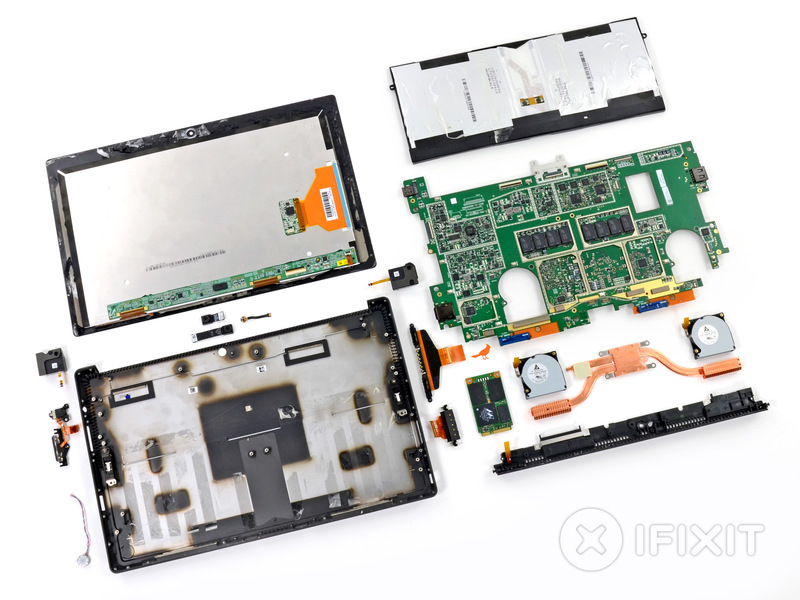
A labelled diagram of a flowering plant. We find out more as we decompose.

## Why is decomposition important?

Decomposing problems into their smaller parts is not unique to computing: it’s pretty standard in engineering, design and project management.

Software development is a complex process, and so being able to break down a large project into its component parts is essential – think of all the different elements that need to be combined to produce a program, like PowerPoint.

The same is true of computer hardware: a smartphone or a laptop computer is itself composed of many components, often produced independently by specialist manufacturers and assembled to make the finished product, each under the control of the operating system and applications.

[](http://barefootcas.org.uk/wp-content/uploads/2014/06/decomposed-surface-pro-image.jpg)

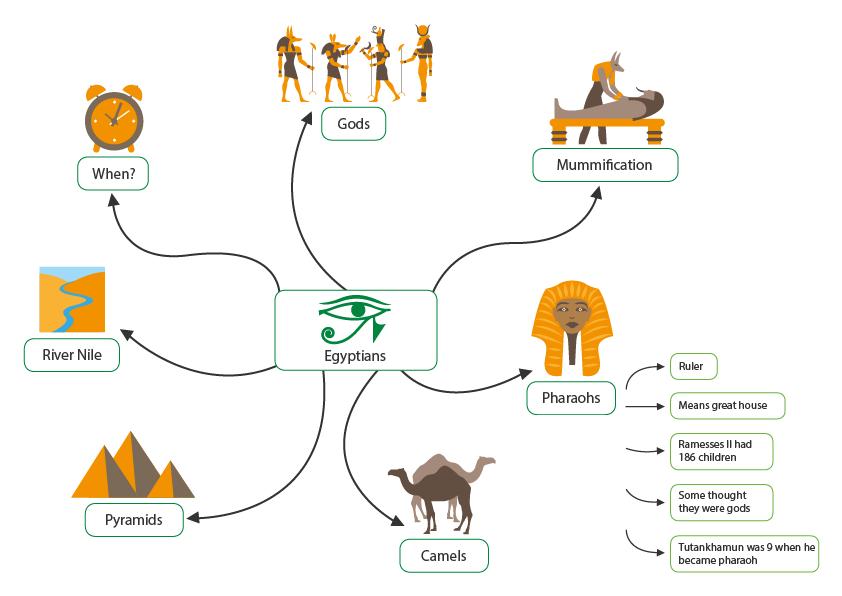
A tablet can be broken down (decomposed) into smaller components.  
With thanks to iFixit.com

## What does decomposition look like in the class?

Decomposition is everywhere in primary practice. We are always asking pupils to find out more, tell us more.

Whenever pupils are labelling, adding detail to concept maps, or creating instructions, life cycles, and timelines they are breaking something down, and thinking about detail, and so developing their decomposition skills.

If pupils undertake any kind of project or task, such as: putting on a school play, organising a cake sale, creating a news report, working out how to solve a maths problem, making a sandwich or getting dressed for PE, they will need to break the task up into manageable tasks or parts. That is decomposition.

[](http://barefootcas.org.uk/wp-content/uploads/2014/06/Egyptians.png)

Pupils add detail to their knowledge and understanding of the Egyptians in concept maps.

In primary settings, as pupils learn to decompose, they might create only a partial decomposition. That is, they might not include all aspects or parts of a topic, but only the things they currently know about. As they progress they should check that they create a complete decomposition and do not miss any part of the whole. Also, as they progress, they can further decompose each part into sub-parts and so on.

If making a computer game, a pupil might decompose the game into: plot, characters and setting. They might then further decompose the characters into actions and appearance. The setting might be decomposed into obstacles, scoring objects and background. In developing a robotic toy, pupils would need to consider the hardware components, both individually and as a system, the algorithms they’ll need to control their toy and how to write those as code. In general, technology = hardware + [algorithms](http://barefootcas.org.uk/sample-resources/algorithms/) + code.

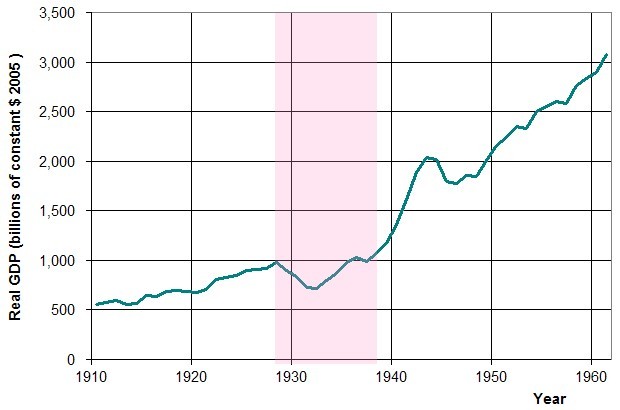
1. **Patterns**

## What is pattern?

Patterns are everywhere, for example, we use weather patterns to create weather forecasts; children might notice patterns in how teachers react  to their behaviour to work out how to behave next time. By identifying patterns we can make predictions, create rules and solve more general problems. In computing, the method of looking for a general approach to a class of problems is called generalisation.

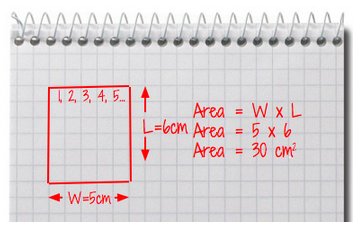
[](http://barefootcas.org.uk/wp-content/uploads/2014/06/glowing-253676_1920.jpg)

Red sky at night farmers delight – an adage based on weather patterns. Images from Pixabay Creative Commons Deed CC0

[](http://barefootcas.org.uk/wp-content/uploads/2014/06/US_GDP_10-60.jpg)

Analysts might attempt to predict future trends based on historical patterns. By Lawrencekhoo (Own work) [Public domain], via [Wikimedia Commons.](http://commons.wikimedia.org/wiki/File%3AUS_GDP_10-60.jpg%C2%A0http:/en.wikipedia.org/wiki/Causes_of_the_Great_Depression)

In learning about area, pupils could find the area of a particular rectangle by counting the centimetre squares on the grid on which it’s drawn. But a better solution would be to multiply the length by the width: not only is this quicker, it’s also a method that will work on all rectangles, including really small ones and really large ones. Although it takes a while for pupils to understand this formula, once they do it’s so much faster than counting squares.

[](http://barefootcas.org.uk/wp-content/uploads/2014/06/maths-generalisation-pattern.png)

Pupils learn about formula’s in maths: these are generalisations.

## Why is pattern important?

Computer scientists strive to solve problems quickly and efficiently and to re-use previously created methods. If they see a common pattern across a problem or program they will look to create a single common solution or module to reuse many times.  This will mean they only have to design and build the common module once, rather than designing and building many versions. Common modules of code are sometimes called procedures or functions; many programming languages have shared libraries of these functions.

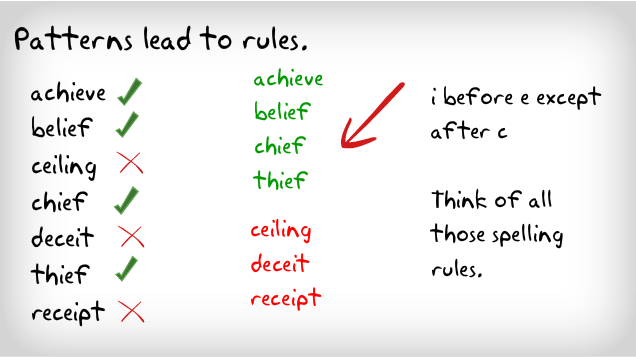
Machine learning is an important application of computer science, in which computers are programmed to recognise patterns in input data, such as automatic number plate recognition systems, facial recognition systems and algorithmic trading on stock markets.

## What does pattern look like in the class?

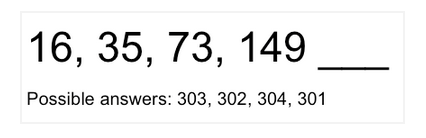
We ask pupils to notice, look for and learn from patterns to help them better understand the world. The word pattern is used across subjects such as mathematics, science, literacy, geography and modern languages.

Pupils are likely to encounter the idea of generalising patterns in many areas of the curriculum.

* From an early age, they’ll become familiar with repeated phrases in nursery rhymes and stories; later on they’ll notice repeated narrative structures in traditional tales or other genres.
* In music, children will learn to recognise repeating melodies or bass lines in many musical forms.
* In English, pupils might notice common rules for spellings, and their exceptions.
* In maths, pupils typically undertake investigations in which they spot patterns and deduce generalised results.



Patterns reveal spelling rules

[](http://barefootcas.org.uk/wp-content/uploads/2014/06/number-sequence-pattern.png)

Can pupils spot the pattern to reveal the number sequence rule?

1. **Abstraction**

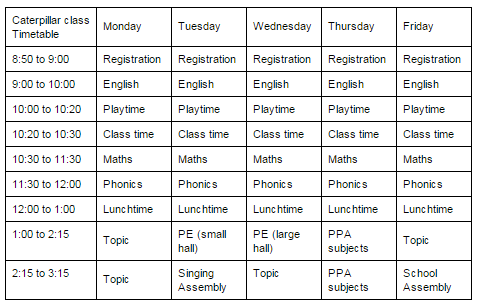
## What is abstraction?

For American computer scientist Jeanette Wing, credited with coining the term, abstraction lies at the heart of computational thinking:

*The abstraction process – deciding what details we need to highlight and what details we can ignore – underlies computational thinking.’ Computational thinking and thinking about computing (The Royal Society, 2008)*

Abstraction is about simplifying things; identifying what is important without worrying too much about the detail. Abstraction allows us to manage complexity.

We use abstractions to manage the complexity of life in schools. For example, the school timetable is an abstraction of what happens in a typical week: it captures key information such as who is taught what subject where and by whom, but leaves to one side further layers of complexity, such as the learning objectives and activities planned in any individual lesson.

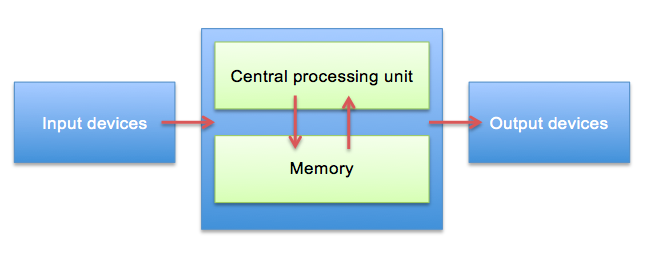
[](http://barefootcas.org.uk/wp-content/uploads/2014/07/class_timetable.png)

A key stage 1 class timetable is an abstraction of the school day for one class, much is omitted to provide a simplified summary.

## Why is abstraction important?

Abstractions are sometimes represented as layers or hierarchies, allowing us to view things at different degrees of detail. The nature of being able to hide complexity within boxes within boxes makes abstraction a powerful tool as we do not need to worry about the technical detail of what goes on inside each box.

In computer science, abstraction is used to manage the complexity of much of what is designed and created. Computer hardware is seen as components or black boxes. Software is built of layers each hiding the complexity of the next successive layer.

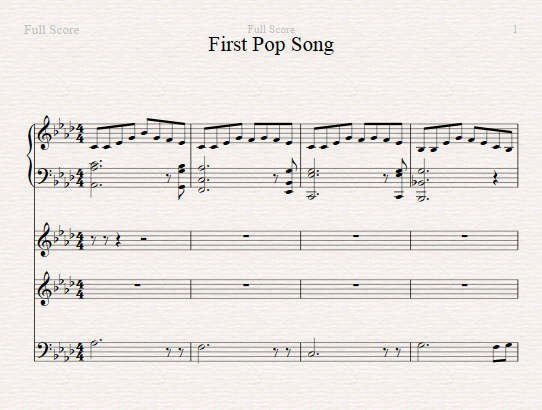
[](http://barefootcas.org.uk/wp-content/uploads/2014/07/abstraction_11_computer_system.png)

The abstraction above represents a computer. It shows the names of the components and how they interact with each other but hides the complexity of each type of component.

## What does abstraction look like in the class?

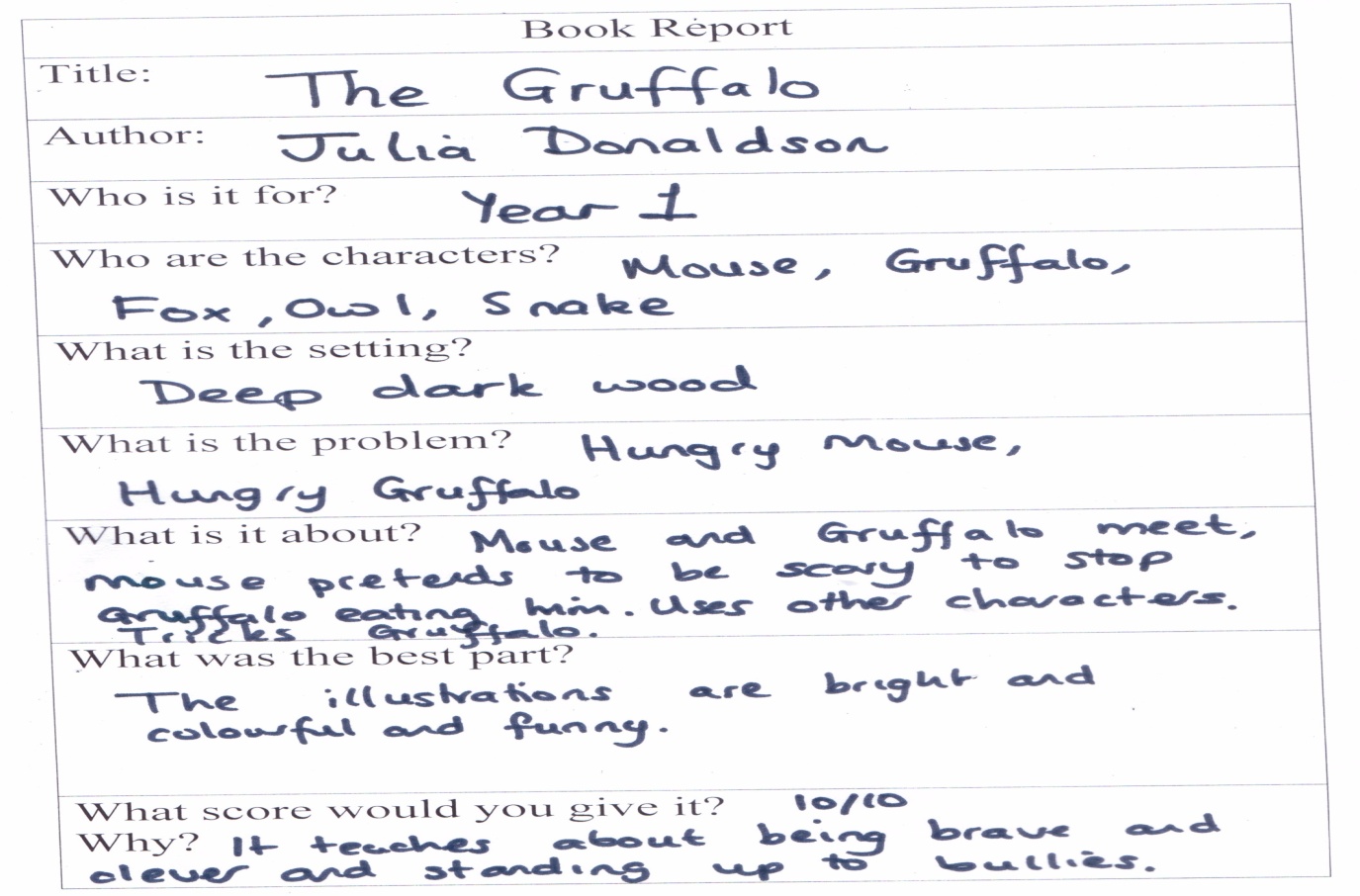
Abstraction is such a powerful way of thinking about systems and problems that it seems worth introducing pupils to this whilst they’re still at primary school. This doesn’t have to be just in computing lessons.

* In maths, working with ‘word problems’ often involves a process of identifying the key information and establishing how to represent the problem in the more abstract language of arithmetic, algebra or geometry.
* In geography, pupils can be helped to see a map as an abstraction of the complexity of the environment, with maps of different scales providing some sense of the layered nature of abstraction in computing.
* In music, the piano score of a pop song might be thought of as an abstraction for that piece of music.

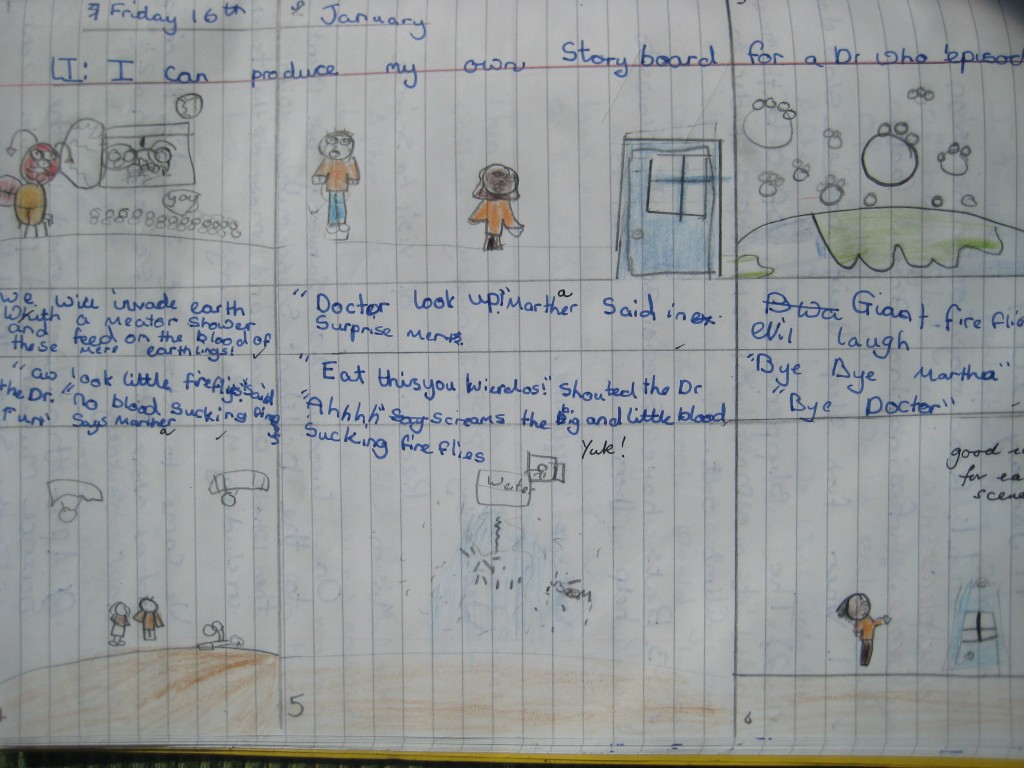
[](http://barefootcas.org.uk/wp-content/uploads/2014/07/abstraction_7_music1.png)

Music is abstracted to musical notation.

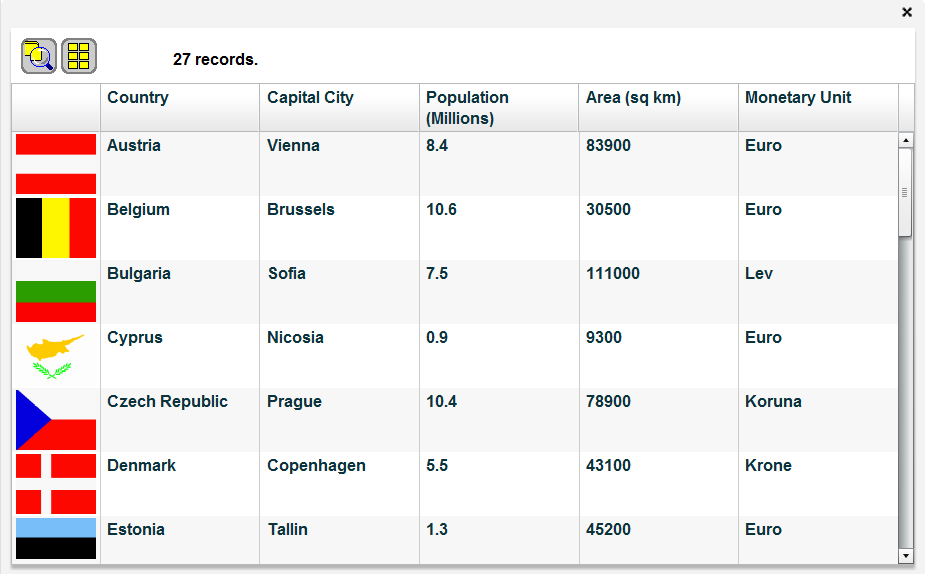
When creating a story plan, a summary, or working out a mind map pupils are abstracting, as they are leaving to one side the detail they do not need at that time.

[](http://barefootcas.org.uk/wp-content/uploads/2014/07/abstraction_13_book_report.jpg)

A book report provides a summary, a pupil’s abstraction or view of the book.

[](http://barefootcas.org.uk/wp-content/uploads/2014/07/abstraction_14_story_plan.jpg)

A story plan summaries a story, providing an abstraction of the story showing just the key features.

[](http://barefootcas.org.uk/wp-content/uploads/2014/07/abstraction_15_database.gif)

A database of country data only holds selected data, such as population, area. [Image by kind permission of Simon Haughton.](http://simonhaughton.typepad.com/.a/6a0120a530561e970b014e8bf6e6b4970d-pi)

Simulations and models are abstractions: these are used across the curriculum to explain ideas. For example, science simulations might help to teach about gravity, history simulations about the Roman invasion of Britain, physical geography models give insights about how fossilization occurs, art simulations might show how to work with clay, data handling models allow exploration of what if scenarios and so on. Also beyond the curriculum simulations and models are the basis of most computer games.

In computing lessons, pupils can learn about the process of abstraction from playing computer games, particularly those that involve interactive simulations of real world systems as they appreciate that they are based on, but simpler than real life. Encourage pupils’ curiosity about how things work, helping them to think about what happens inside the computer or on the internet as they use software or browse the web.

1. **Evaluation**

## What is evaluation?

Evaluation is about making judgements, in an objective and systematic way where possible.

Evaluation is something we do every day – we make judgements about what to do and what we think based on a range of factors.

For example, when considering a new [digital device](http://barefootcas.org.uk/barefoot-primary-computing-resources/concepts/computer-systems/) for use in the classroom, there would be a number of criteria that would be considered; for example, operating system, portability, memory, screen size, ease of use and warranty.

[](http://barefootcas.org.uk/wp-content/uploads/2014/09/evaluation-image-1-OS-monitor-1.png)

After evaluation OS X operating systems may be judged to be most suitable. Image public domain CCO http://pixabay.com/en/monitor-display-screen-computer-149362/

[](http://barefootcas.org.uk/wp-content/uploads/2014/09/evaluation-image-2-laptop.png)

Or in some cases, after evaluating the options, laptops may be judged as the most suitable device for a classroom or school. Image public domain CCO http://pixabay.com/en/laptop-black-blue-screen-monitor-33521/

## Why is evaluation important?

In computer science, evaluation is systematic and rigorous; it is about judging the quality, effectiveness and efficiency of solutions, systems, products and processes. Evaluation checks that solutions do the job they are designed to do and are fit for purpose.

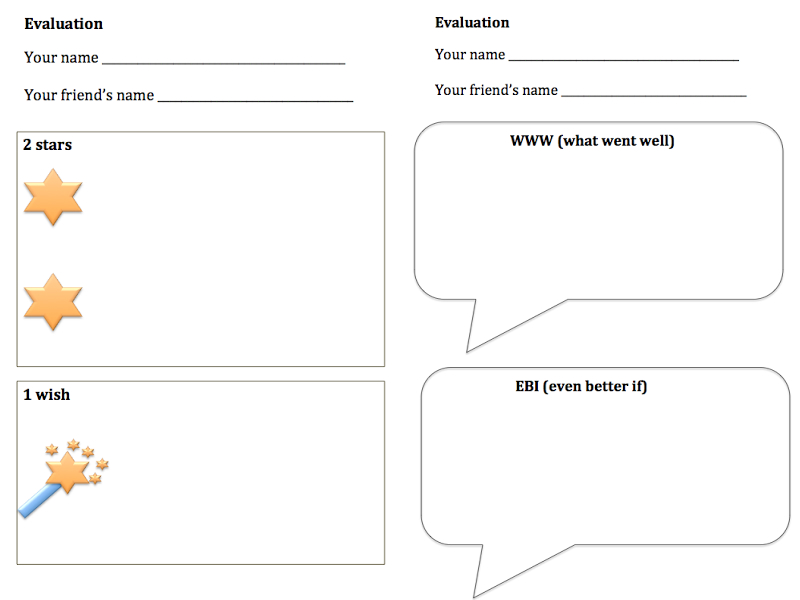
One approach to systematic evaluation could be to use:

* specific criteria (for example a design goal or specification)
* user needs (considering who the users are and what they need from the final design)

## What does evaluation look like in the class?

Evaluation is something that occurs everyday in schools; pupils evaluate their work, teachers evaluate lessons that they deliver and pupils’ learning and progress is evaluated.

For example, self and peer assessment can help to develop pupils’ evaluation skills as they become used to making judgements about their own and others’ work, evaluating against the success criteria and suggesting how the work could be improved.

[](http://barefootcas.org.uk/wp-content/uploads/2014/09/evaluation-examples-barefoot-computing.jpg)

Example peer-evaluations – 2 stars and a wish and www/EBI

## Evaluation across the curriculum

### English

In English, pupils may know the success criteria for writing. For example, they may need to remember to include capital letters, full stops, use adjectives, synonyms and adverbs.

As pupils progress through key stage 1 they start to be able to express their choices and preferences more readily and clearly. For example, they may recommend a book to a friend, making judgements about what books they think are most suitable and explaining why they think their friend will enjoy it.

### Design and technology

The [design and technology curriculum](https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/239041/PRIMARY_national_curriculum_-_Design_and_technology.pdf) makes particular use of evaluation as pupils work through the design, make and evaluate cycle. In key stage 1 pupils are expected to evaluate against criteria and in key stage 2 also consider standards and the needs of others in their designs and products.

### PE

In gymnastics pupils may have a list of ‘good’ things they are looking for, such as landing on two feet from a jump, using different rolls in a routine that uses the ball and always being in control.

**THE 5 APPROACHES TO COMPUTATIONAL THINKING**

1. **Tinkering**

## What is tinkering?

Tinkering means trying things out. For young children, this is the vital play based, exploration and experimentation phase of learning about something. For older pupils and adults it is purposeful exploration and making, often through trial and improvement.

We often tinker when we encounter something new to find out what it does and how it works, for example when given a new mobile phone, we might try out all the features and then more purposefully work out what is useful for us.

Having the freedom to explore through playful challenges in a risk free environment engenders confidence and a let’s have a go attitude. Open ended questions encourage creativity and diverse ideas. Tinkering should be fun, free, creative and full of questions and surprises.

[](http://barefootcas.org.uk/wp-content/uploads/2014/09/Tinkering_image-1_girl-tinkering-with-ScratchJr.jpg)

Tinkering with Scratch Junior.

When tinkering, we try things and and may do things in many different ways. Tinkering builds [perseverance](http://barefootcas.org.uk/barefoot-primary-computing-resources/computational-thinking-approaches/persevering/). Tinkering helps us to see our use of technology as being about developing our understanding rather than getting the ‘right answer’. When tinkering we look at things from many different angles. We try ideas that seem wrong just to see have happens. When we use technology we’ve tinkered with, we are more likely to try innovative and novel solutions.

[](http://barefootcas.org.uk/wp-content/uploads/2014/09/Tinkering_image-2_boy-playing-with-mouse.jpg)

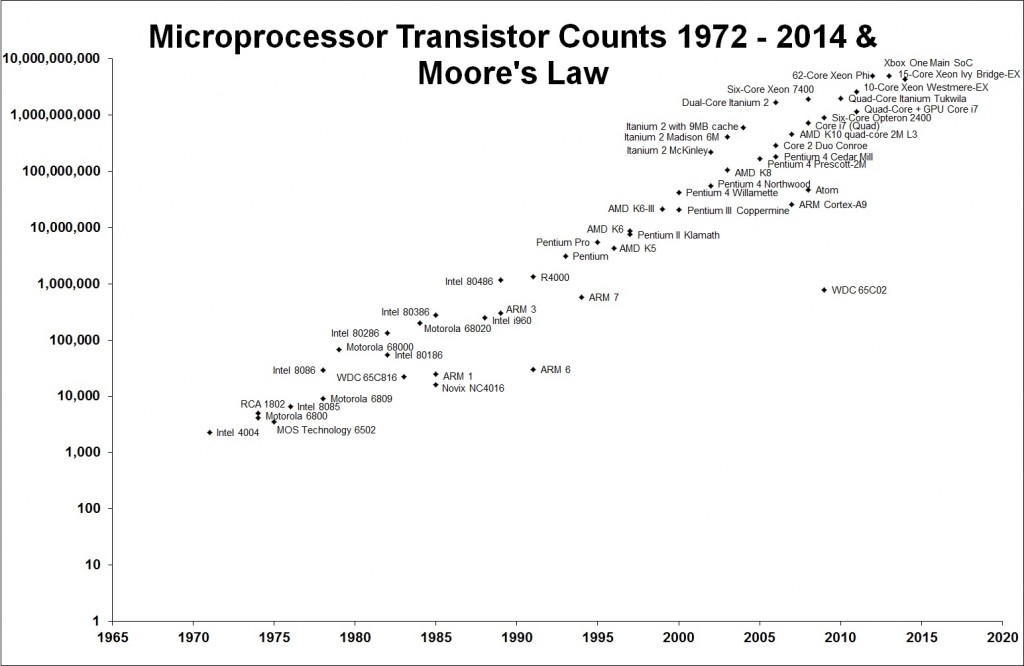
Finding out what it does.  
Image from www.pixabay.com Creative Commons Deed CC0

Tinkering is closely associated with[logical reasoning](http://barefootcas.org.uk/barefoot-primary-computing-resources/concepts/logic/), as pupils try things out they start to build up experiences of cause and effect. “If I move this, then this happens.”

Tinkering is a big part of independent learning: pupils try things out and learn by themselves without the lead of someone else.

## Why is tinkering important?

Software and hardware change at what seems an ever increasing rate. Whether we are a user of technology or a creator we need to be open to these rapid and frequent developments. Increased speed and capacity reveal new opportunities and drives change. To keep up to date in computing we need to be able to learn new skills, appreciate differences and similarities and work out how we might harness new opportunities.

[](http://barefootcas.org.uk/wp-content/uploads/2014/09/Tinkering_image_3_Mooreslaw.jpg)

Graph showing Moore’s Law.

In 1965, Gordon Moore, co-founder of Intel, observed the doubling in processing power every two years. He predicted that it would continue (Moore’s Law). In fact the rate of change has increased to doubling every eighteen months. In the past, this improvement in performance has been aligned to the increased density of transistors on microprocessors. Physical limits of current electronic components indicate that Moore’s Law may not hold true soon. However, computer scientists are looking at alternative ways of designing components, how they are implemented and even new ways of programming to maintain increases in processing power.

Often programmers first explore a new technology to familiarise themselves with how it works and to get ideas about how they might exploit it before moving onto a more systematic, rigorous, purposeful use. Being confident at tinkering helps us see change as an opportunity rather than a risk, enabling us to keep up to date and harness new technologies.

## What does tinkering look like in the class?

When introducing any new digital device, programming language or software environment, start by tinkering. For example, if using Scratch for the first time, pupils might play some existing Scratch games and then try some of the features of the programming environment. You might set them the challenge to try three new things or to make something unexpected happen.

Pupils’ confidence to tinker will develop through opportunities to explore technologies. Some children will have little home access to digital devices and so may be hesitant at tinkering with school equipment. These pupils may need extra encouragement to learn how to tinker!

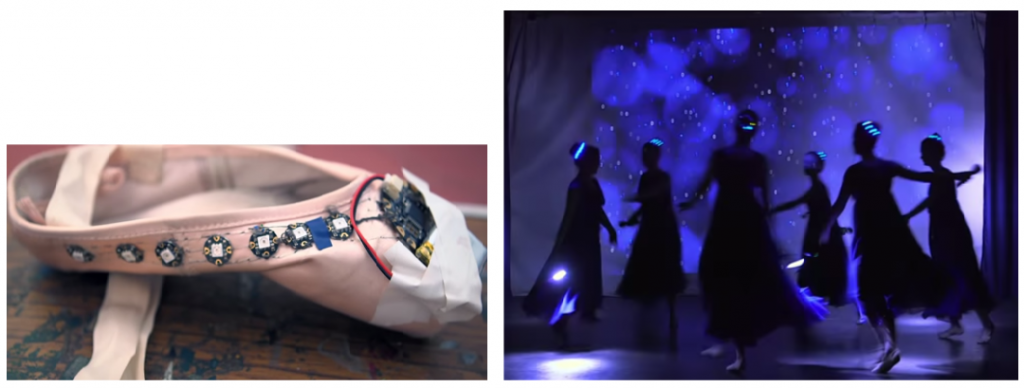
## Creating

## What is creating?

Creating is about planning, making and evaluating things; e.g. animations, games or robots.

Programming is a creative process. Creative work involves both originality and making something of value: typically something that is useful or at least fit for the purpose intended.

Sometimes the things we create will be products which fulfil particular needs, sometimes we will work on more imaginative endeavours in which various media provide outlets for creative expression. The things made do not need to be tangible, physical artefacts – software and digital media allow scope for creativity as do hardware or more complex systems.

[](http://barefootcas.org.uk/wp-content/uploads/2015/02/dance.png)

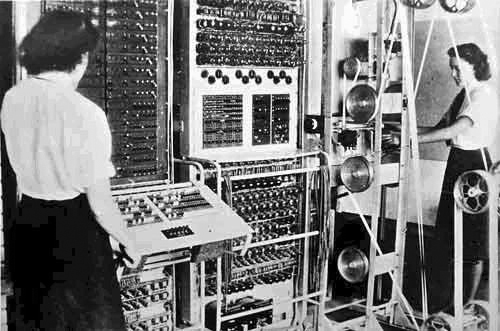
A pressure sensitive ballet shoe that lights up. In the performance (second image) animations of the dancers movements are displayed. The animation program uses dancer movement data, captured using a Kinect. Images by kind permission from <http://www.readysaltedcode.org/>

Sir Ken Robinson defines creativity as ‘the process of having original ideas that have value’. Creative work should be original, and this should at least mean that it is our own work, not something we have simply filled in a blank of or copied. Creative work should also be of value: at the very least to the person creating it, but also to a wider audience. As well as originality and value, creative work also implies that something has been made. An emphasis on creativity recognises how powerful the process of making things for others is as a means to learn. By becoming masters of software tools and digital devices we develop confidence, competence and independence that we can use playfully, experimentally and purposefully, as a way to express our insights and ideas.

## Why is creating important?

Computer Science is not just an academic subject, it is a practical applied engineering discipline that creates solutions to real world problems and provides opportunities for endeavours across the arts.

Since the early pioneers of computing when wartime codes were cracked, the world has seen an explosion of computer use. The range and complexity of created systems is astonishing, number crunching accounting systems, telephone systems, personal computing devices, human genome mapping systems, space shuttles, smart cities, art installations, gaming environments, online spaces each system created by skilled computer scientists.

[](http://barefootcas.org.uk/wp-content/uploads/2015/02/Colossus.jpg)

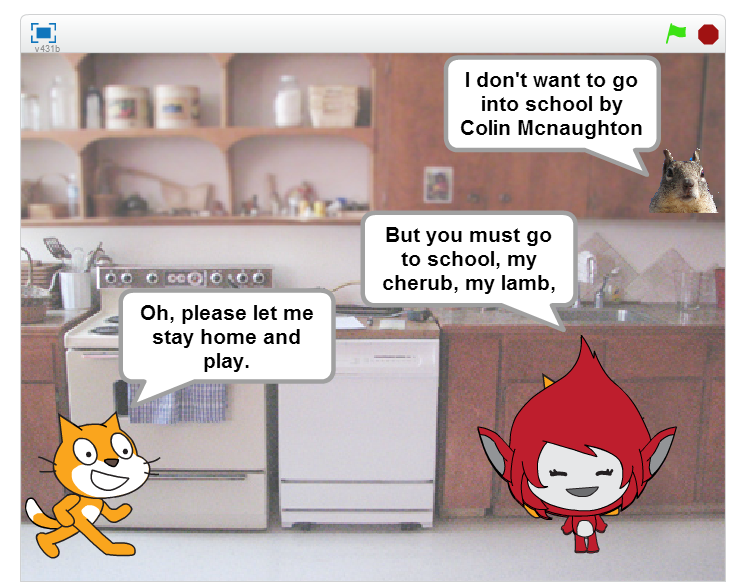
1943 Colossus code breaking computer.  
[Public domain], via Wikimedia Commons http://commons.wikimedia.org/wiki/File%3AColossus.jpg  
http://en.wikipedia.org/wiki/The\_National\_Archives\_(United\_Kingdom)

In computing, we create systems and programs that solve problems or exploit opportunities. We have ideas about what we would like to make, we analyse the problem or systems involved, [design](http://barefootcas.org.uk/sample-resources/algorithms/), [write](http://barefootcas.org.uk/barefoot-primary-computing-resources/concepts/programming/)and [debug](http://barefootcas.org.uk/barefoot-primary-computing-resources/computational-thinking-approaches/debugging/)code and [evaluate](http://barefootcas.org.uk/barefoot-primary-computing-resources/concepts/evaluation/)what we have created. Throughout this process we use computational thinking to help us focus on what is important ([abstract](http://barefootcas.org.uk/barefoot-primary-computing-resources/concepts/abstraction/)), break down problems or systems into parts ([decompose](http://barefootcas.org.uk/sample-resources/decomposition/)), think [logically](http://barefootcas.org.uk/barefoot-primary-computing-resources/concepts/logic/), spot [patterns](http://barefootcas.org.uk/barefoot-primary-computing-resources/concepts/patterns/) and look for opportunities to reuse components.

## What does creating look like in the class?

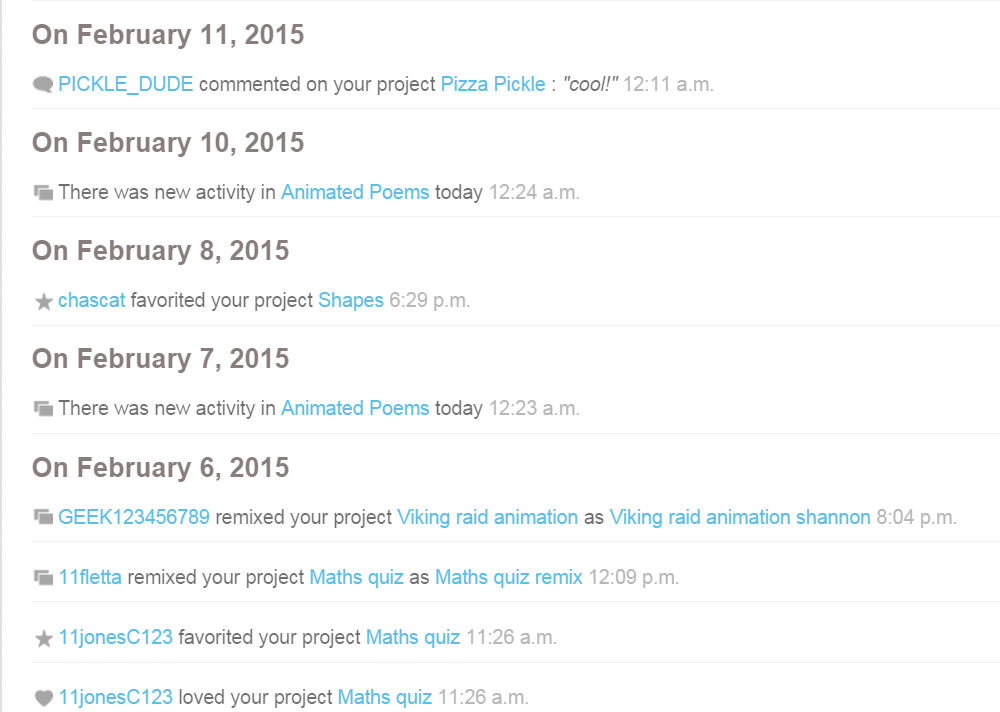
Through the programme of study for Computing Science, Bearsden Primary pupils will become skilled at creating high quality products and content using digital technology including programming.

By engaging in designing and making pupils become active rather than passive learners as they use and applying computational thinking skills to create animations, games, quizzes etc.

[](http://barefootcas.org.uk/wp-content/uploads/2015/02/animated_poem.png)

An animation of a poem.

Projects with a context that is meaningful and relevant to pupils are likely to be particularly engaging as pupils invest their time and effort to create personalised solutions. Programming projects can be shared with peers within a class, and often also with a wider community, by sharing the material they have created pupils gain a sense of pride and satisfaction which can be particularly rewarding and motivating.

[](http://barefootcas.org.uk/wp-content/uploads/2015/02/scratch_remix_favorite_etc.png)

The Scratch Online community gives pupils an opportunity to share what they have created. Work can be ‘remixed, favorited and even loved’.

Often in computing pupils will be learning through making things and it’s helpful to give longer periods of time for such projects in which the processes of planning, implementing, revising and evaluating are fully explored. Working through the stages of a project in detail is good experience for this sort of work elsewhere.

Encourage pupils to reflect on the quality of the work they produce, critiquing their own and others’ projects. The process of always looking for ways to improve on a software project is becoming common practice in software development.

Look for projects in which artistic creativity is emphasised, such as working with digital music, images, animation, virtual environments or even 3D printing, allowing scope for creative expression rather than merely arriving at the right answer.

1. **Debugging**

## What is debugging?

Errors in [algorithms](http://barefootcas.org.uk/sample-resources/algorithms/) and code are called ‘bugs’, and the process of finding and fixing these is called ‘debugging’. Debugging can often take much longer than writing the code in the first place.

In real life, we debug all the time: it’s just spotting mistakes and fixing things. For example we may check a sentence to makes sure it makes sense and then fix it; we try and work out what to do to calm an upset baby and then calm them.

Whilst fixing a program so that it does work can bring a great buzz, staring at code that still won’t work can be the cause of great frustration too: this can be tricky to manage in class.

One way that you can help is to provide a reasonably robust, general set of debugging strategies which pupils can use for any programming bug or when they encounter problems elsewhere.

We suggest a simple sequence of four steps, underpinned by [logical reasoning:](http://barefootcas.org.uk/barefoot-primary-computing-resources/concepts/logic/)

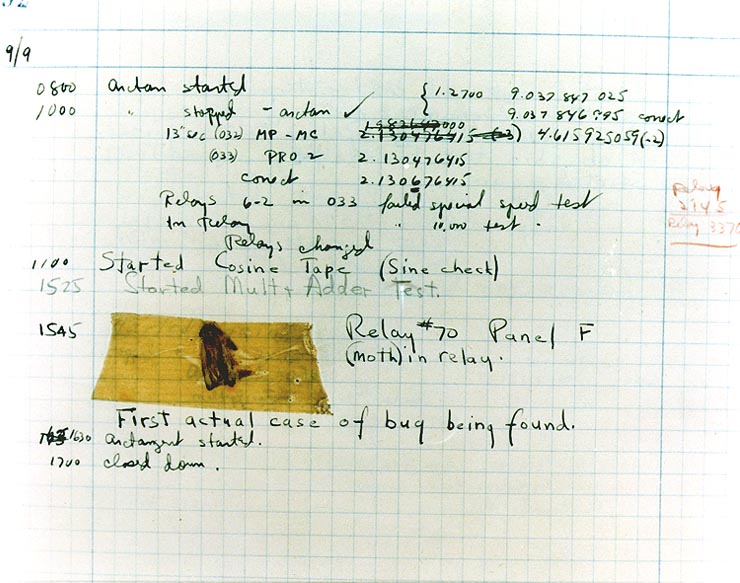
1. Predict what should happen.
2. Find out exactly what happens.
3. Work out where something has gone wrong.
4. Fix it.

Some bugs are logical errors others are coding or syntax errors. Logical errors can be like writing a story where the plot does not make sense.Coding or syntax errors are like writing a story with poor grammar, punctuation or spelling.

## Why is debugging important?

Because of its complexity, the code programmers write often doesn’t work as it’s intended.  There are widely varying statistics and claims related to the amount of effort expended on debugging in commercial projects. In 2002, a [study](http://www.rti.org/newsroom/news.cfm?obj=DA7FBFE6-4A4F-4BFD-B77E0FA3C04D9E22) claimed that software bugs cost the US economy $59.6 billion annually. In Hailpern and Santhanam’s ‘[Software debugging, testing and verification](http://www.cs.uleth.ca/~benkoczi/3720/pres/debug-test-verify_hailpern02.pdf)’ figures of 50 to 75% of project costs being spent on debugging and testing activities are quoted.

The first ‘computer bug’ is said to be a real moth, found in 1947, trapped between points in a relay of a calculator being tested at Harvard University. The operator who found the bug taped it to the computer log, with the entry: “First actual case of bug being found”.

[](http://barefootcas.org.uk/wp-content/uploads/2014/06/Debugging_2e_first_bug.jpg)

The first ‘computer bug’. Image by Courtesy of the Naval Surface Warfare Center, Dahlgren, VA., 1988. [Public domain], via Wikimedia Commons

Although she did not find the famous bug (moth), [Grace Hopper](http://en.wikipedia.org/wiki/Grace_Hopper)made the term debugging popular. Grace was a pioneering computer scientist of her time and a programmer of one of the earliest computers, the Harvard Mark 1. She developed the first compiler, a special type of program that converted the programs we write into the 0s and 1s the computer understands. This paved the way for the development of modern programming languages.

[](http://barefootcas.org.uk/wp-content/uploads/2014/06/Debugging_2d_Grace-Hopper-.jpg)

Grace Hopper made the term debugging popular. Image from U.S. Naval Historical Center Online Library Photograph NH 96919-KN Public Domain.

Another way to help with debugging is to explain one’s algorithm and code to someone or something else. In doing so, it’s quite likely you’ll spot where there’s a problem. The process of describing what a program should do line-by-line is called [rubber duck debugging](http://en.wikipedia.org/wiki/Rubber_duck_debugging%20)!

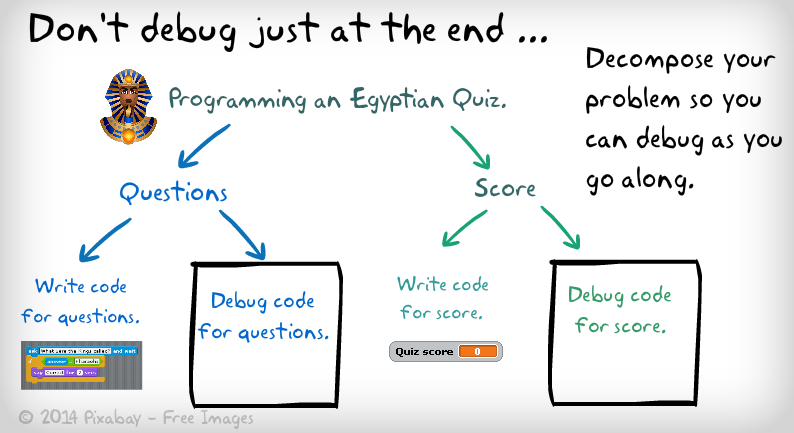
[](http://barefootcas.org.uk/wp-content/uploads/2014/06/Debugging_2g_rubber_duck.jpg)

Pupils can talk to rubber ducks to help them debug.www.pixabay.com. Creative Commons Deed CC0

## What does debugging look like in the primary classroom?

In [programming](http://barefootcas.org.uk/barefoot-primary-computing-resources/concepts/programming/) classes, pupils focused on the task of writing a program for a particular goal might want help from you or others to fix their programs: tempting as this may be, it’s worth you and they remembering that the objective in class is not to get a working program, but to  learn how to program – their ability to debug their own code is a big part of that.

Debugging can be easier if you do it as you go along rather than after you have written lots of code. As such, remember to model how to debug small bite sized portions of code, even when using programmable toys, as this establishes better habits for when pupils have to write more complex code.

[](http://barefootcas.org.uk/wp-content/uploads/2014/06/Debugging_3_quiz.png)

In this example of a plan to create a quiz we are modelling how to decompose the problem into two parts. Then showing how we would write and debug each part. If we plan to debug as we go along this is sometimes called testing.

# Persevering

## What is persevering?

Persevering is never giving up, being determined, resilient and tenacious.

[](http://barefootcas.org.uk/wp-content/uploads/2014/09/Persevering_1_Albert_Einstein_Head1.jpg)

“It’s not that I’m smart it’s just that I stay with problems longer.” Albert Einstein

By Photograph by Oren Jack Turner, Princeton, N.J. Modified with Photoshop by PM\_Poon and later by Dantadd. [Public domain]

Computer programming is hard. This is part of its appeal – writing elegant and effective code is an intellectual challenge requiring not only an understanding of the ideas of the algorithms being coded and the programming language you’re working in, but also a willingness to persevere with something that’s often quite difficult and sometimes very frustrating.

Some see playing challenging computer games as a similar experience to coding. When playing a computer game there is a tight feedback loop of cause and effect, similar to when a programmer codes and debugs. Both coders and gamers report experiencing a state of ‘flow’ in which they’re utterly absorbed in and focussed on a single activity. Perhaps all these things help to motivate players and programmers to persevere to finish the game or solve the problem.

## Why is persevering important?

In computing, and many other areas, systems and problems are often large and complex. Problems may be complex to us because they are from unfamiliar contexts or we may need to think about solutions in new ways or try out many options or use technologies new to us. Even the way we think may need to change when dealing with complex problems moving from our normal, [fast and intuitive thinking](http://en.wikipedia.org/wiki/Thinking,_Fast_and_Slow), into a more deliberate, slow, logical way of thinking. As computer scientists we need patience, resilience, a [tolerance of confusion](http://quantumprogress.wordpress.com/2011/04/22/embracing-confusion-as-a-necessary-part-of-learning-part-1/) and endurance when our focused attention is needed over time.

In order to become an expert in anything that is complex we need to persevere and practise. This applies to music, sport, dance, computer gaming, and many other fields of endeavour as well as programming.

It is sometimes quoted that those who aspire to become experts in a particular discipline may need to practise for say [10,000 hours](http://en.wikipedia.org/wiki/10_000_hours_rule) or [10 years](http://norvig.com/21-days.html)to achieve mastery – perhaps implying perseverance and hard work rather than just innate ability makes the difference.

Google’s Peter Norvig thinks it takes 10 years to learn how to program:

*Researchers (*[*Bloom (1985)*](http://www.amazon.com/exec/obidos/ASIN/034531509X/)*,*[*Bryan & Harter (1899)*](http://norvig.com/21-days.html#bh)*,*[*Hayes (1989)*](http://www.amazon.com/exec/obidos/ASIN/0805803092)*, Simmon & Chase (1973)) have shown it takes about ten years to develop expertise in any of a wide variety of areas, including chess playing, music composition, telegraph operation, painting, piano playing, swimming, tennis, and research in neuropsychology and topology. The key is deliberative practice: not just doing it again and again, but challenging yourself with a task that is just beyond your current ability, trying it, analyzing your performance while and after doing it, and correcting any mistakes. Then repeat. And repeat again.*

Norvig, P (2001) [Teach yourself programming in ten years](http://norvig.com/21-days.html).

## What does persevering look like in the class?

Across the curriculum pupils show tenacity and determination as they tackle new, difficult or complex activities. In subjects such as music, sport and dance children practise, train and rehearse to learn and improve skills and techniques.

[](http://barefootcas.org.uk/wp-content/uploads/2014/09/Persevering_4_gymnastics.jpg)

Pupils show perseverance in different ways, some are continually enthusiastic and energetic. Image from [www.pixabay.com](http://barefootcas.org.uk/barefoot-primary-computing-resources/computational-thinking-approaches/persevering/www.pixabay.com) Creative Commons Deed CC0

[](http://barefootcas.org.uk/wp-content/uploads/2014/09/Persevering_5_music.png)

Others demonstrate perseverance through quiet and dogged patience and persistence.

Carol Dweck’s work on ‘[growth mind-sets](http://mindsetonline.com/whatisit/about/)‘ suggests that hard work and a willingness to persevere in the face of difficulties can be key factors in educational outcomes. Dweck’s studies asserted that pupils praised with ‘good job you worked very hard’ were more likely to develop a ‘growth mindset’ than those praised ‘good job, you are clever’.Pupils who relish challenge will be able to stay positive when things don’t work out as expected, seeing it as an opportunity to dig more and find a solution. Those who keep going are likely to learn more as they remain focused and have confidence that they will get through the confusion they feel as they practise, persevere and assimilate their new experiences, understanding that this is a normal part of learning. Pupils who persevere are more likely to find unusual and innovative alternatives.

Encourage pupils to look for strategies they can use when they do encounter difficulties with their programming work, such as working out exactly what the problem is, searching for the solution on Bing or Google (with the safe search mode locked), KidRex or Swiggle, or asking a friend for help.

1. **Collaborating**

## What is collaborating?

Collaborating means working with others to ensure the best result.

It’s hard to think of any job or leisure activity which doesn’t involve some elements of collaboration. For example, as teachers we plan together, team-teach or observe each other in order to develop our practice.

Collaborative group work has long had a place in primary education, and computing should be no different.

When programming, many see ‘pair programming’ as a particularly effective way to write code, with two programmers sharing a screen and a keyboard, working together to create software. Typically one programmer acts as the driver, dealing with the detail of the programming, whilst the other takes on a navigator role, looking at the bigger picture. The two programmers regularly swap roles, so both have a grasp of both detail and big picture.

[](http://barefootcas.org.uk/wp-content/uploads/2014/09/Collaborating_1_boys_working_together.jpg)

Pupils working on a project together.

As well as providing an opportunity to share success, when we collaborate, we are often motivated to keep going on tasks that may seem confusing, or even impossible, if we were working on them by ourselves.  The opportunity to bounce ideas off someone else and to explain something to another person helps pupils to develop their[logical reasoning](http://barefootcas.org.uk/barefoot-primary-computing-resources/concepts/logic/) skills and [perseverance](http://barefootcas.org.uk/barefoot-primary-computing-resources/computational-thinking-approaches/persevering/). If we work independently we may need a duck to explain our program to, although these aren’t normally as effective as another person!

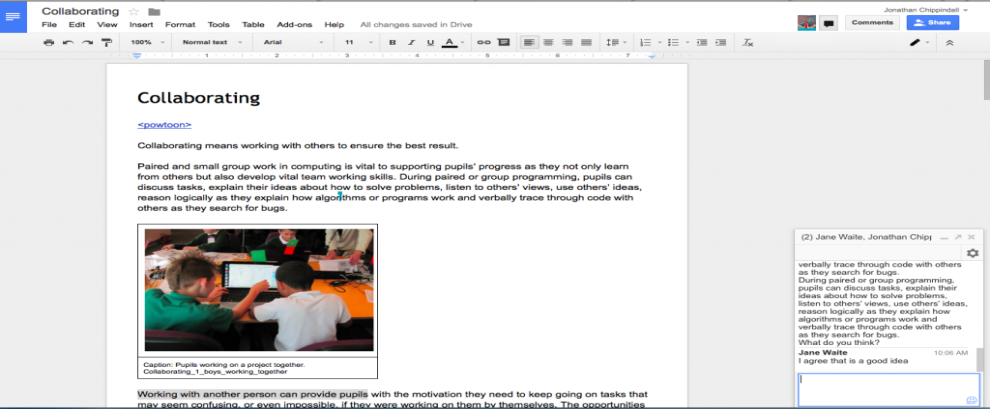
with others.

## Why is collaborating important?

Software is created by teams of programmers and others working together on a shared project. Problems and systems are [decomposed](http://barefootcas.org.uk/sample-resources/decomposition/) into subtasks for different teams with different specialisms. For example, in computer game development, different teams may work on the game design, artwork, animations and programming. Each team will be dependent on and responsible to each other, and so effective communication between each is vital to the success of such projects.

Developers need to communicate with their customers, to find out about requirements and check if a solution works for them, for example. In much modern software development, customers are often seen as collaborators working as part of the development team to create the solution together.

Online collaborative working is a very important part of software development. When collaborating on larger projects, different teams might not all be working in the same geographical location but could be distributed across a country, or the world. The use of online collaboration tools such as[Google Drive](https://drive.google.com/) easily enables people to work together across the internet. Google Docs enables multiple users to simultaneously create and edit content in documents, spreadsheets and presentations, as shown below.

[](http://barefootcas.org.uk/wp-content/uploads/2014/09/Collaborating-Image-3-Google-Docs.png)

A screenshot of Google Docs which allows multiple people to create and edit content whilst chatting in real time using the instant messaging feature.

Computer scientists and software engineers often use others’ work to develop new ideas or improve solutions. Open source software makes it easy to take someone else’s code, look at how it’s been made and then adapt it to your own particular project or purpose. Websites such as[www.github.com](http://www.github.com/) make it easier to share and collaborate on the development of code with others.

## What does collaborating look like in the class?

The internet makes it easy for pupils to work collaboratively online, just as they have always been able to do in class. Web-based platforms such as Google Docs or Office 365 mean that pupils can work on files together, either by inviting comment and review from others, or through real time collaboration.

The efficiency with which joint projects can be undertaken and reviewed can make this a very exciting mode of work. Teachers and pupils alike will be aware of the collaborative nature of Wikipedia. This can provide a good opportunity for pupils to become more discerning in evaluating digital content, and indeed to correct errors or add content to Wikipedia when they can. The Simple English Wikipedia is far less ‘complete’ than the main edition, and so it’s practical for primary classes to ‘adopt’ pages here, editing or monitoring these for other users. Alternatively, teachers can set up their own wiki for their class, using one of a number of online tools  such as [www.pbworks.com](http://www.pbworks.com/). For example, pupils could collaboratively create a wiki on the planets in the solar system or the stages in the water cycle.

As well as wikis, these days, it’s easy for a teacher to set up a class blog, perhaps as open access so that a child’s work can reach an audience, potentially, of close on three billion others. Blogs are also a great way to share what’s happening in your class with your pupils’ parents and with other teachers. Blogs can be used as a basis for partnership projects with another class or group of classes, taking turns to respond to work that’s posted (as in David Mitchell’s [Quadblogging](http://quadblogging.com/" \t "_blank)® projects). However, it’s really important that comments posted to a class or school blog are moderated by a teacher before they’re seen by pupils.

A further opportunity for collaboration can be introduced through [digital leaders](http://www.digitalleadernetwork.co.uk/), here pupils take responsibility for class blogs and equipment, supporting others and representing their class or school in the wider community.

[](http://barefootcas.org.uk/wp-content/uploads/2014/09/Collaborating_4_KS1_beebot_group.jpg)

Three pupils work together to work out a route using a fake bot, arrow cards, and an algorithm sheet to work out their design.