

Getting active

**Lessons:** 5

**Programming languages:** MakeCode

**Target age:** 7-11 yrs

**Subjects & topics:**

* Programming: Variables, Randomisation
* Computational thinking: Algorithms
* Sciences: Health

# Unit of work summary

This series of five lessons is aimed at students aged 10-11 years. They are introduced to variables and develop their understanding of planning, coding and debugging through a mixture of unplugged and practical programming activities.

Students use variables to design and program the micro:bit to be star-jump and step counters. They then use random numbers and selection to code a times table test and an activity selector.

## Overall key learning

* can analyse problems in computational terms, and have repeated practical experience of writing computer programs in order to solve such problems
* are responsible, competent, confident and creative users of information and communication technology

## Additional skills

Creative thinking, collaboration, problem-solving, debugging, evaluation

## Lesson 1: Describing with variables

In this ‘unplugged’ lesson, pupils develop their understanding of variables. They use variable names and values to retell stories and use variables to describe characters before changing variables to describe the number of sides different shapes have.

**Key learning:**

* To know and understand what variables are
* To use variables to describe a character
* To write algorithms that use variables

## Lesson 2: Using variables in programs

Pupils apply their understanding of variables to record star-jumps and write an algorithm to represent this. They explore a program using variables and debug it so the BBC micro:bit can be used to record an activity.

**Key learning:**

* To write algorithms that use variables
* To explain how variables are used in programs
* To debug programs containing variables

## Lesson 3: Programming step-counters

Pupils consider how activity trackers record the number of steps a person takes and write algorithms to represent this. They then use the MakeCode editor to use the BBC micro:bit as a step-counter and evaluate its effectiveness.

**Key learning:**

* To identify the uses of a step-counter
* To write an algorithm for a step-counter
* To program the BBC micro:bit as a step-counter

## Lesson 4: Random activities

Pupils complete a times table test using the BBC micro:bit and identify how variables and random numbers have been used, before modifying it. They consider how variables and random numbers can be used to help a family become more active with micro:bit and write an algorithm to follow next lesson.

**Key learning:**

* To predict how variables will be used in programs
* To understand how a variable can be set to a random number
* To write algorithms that use random number variables

## Lesson 5: Programming an activity picker

Pupils apply their understanding of variables and random numbers by writing, testing and debugging a program to use micro:bit as a family activity selector, following their algorithm. They then evaluate their work.

**Key learning:**

* To debug programs involving random number variables
* To write programs that use random number variables
* To evaluate a solution effectively

# Curriculum links

These lessons are mapped to the following learning objectives and standards for computing and science.

## England National Curriculum

#### KS2 computing curriculum

Curriculum aims:

* can analyse problems in computational terms, and have repeated practical experience of writing computer programs in order to solve such problems
* are responsible, competent, confident and creative users of information and communication technology

Students should be taught to:

* design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts
* use sequence, selection and repetition in programs; work with variables and various forms of input and output
* use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs

[Read the full KS2 computing curriculum](https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/239033/PRIMARY_national_curriculum_-_Computing.pdf)

#### KS2 science curriculum

Humans and other animals (year 6 program of study)

Students should be taught to:

* recognise the impact of diet, exercise, drugs and lifestyle on the way their bodies function.

[Read the full KS2 science curriculum](https://www.gov.uk/government/publications/national-curriculum-in-england-science-programmes-of-study/national-curriculum-in-england-science-programmes-of-study)

#### KS2 DT curriculum

Students should be taught to:

* apply their understanding of computing to program, monitor and control their products

[Read the full KS2 DT curriculum](https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/239041/PRIMARY_national_curriculum_-_Design_and_technology.pdf)

## Scotland Curriculum for Excellence

#### Technologies

* I can extend and enhance my knowledge of digital technologies to collect, analyse ideas, relevant information and organise these in an appropriate way (TCH 2-01a)
* I understand the operation of a process and its outcome. I can structure related items of information (TCH 2-13a)
* I can explain core programming language concepts in appropriate technical language (TCH 2-14a)
* I can create, develop and evaluate computing solutions in response to a design challenge (TCH 2-15a)

[Read the full Curriculum for Excellence: technologies](https://education.gov.scot/Documents/Technologies-es-os.pdf)

#### Sciences

* By researching, I can describe the position and function of the skeleton and major organs of the human body and discuss what I need to do to keep them healthy (SCN 1-12a)

[Read the full Curriculum for Excellence: sciences](https://www.education.gov.scot/Documents/sciences-eo.pdf)

## Northern Ireland Curriculum - Primary

#### Using ICT across the curriculum

Pupils should be taught to:

* explore - investigate, make predictions and solve problems through interaction with digital tools
* evaluate - talk about, review and make improvements to work, reflecting on the process and outcome

#### KS1 - suggested curriculum ideas for the world around us

* design and make simple models
* the variety of living things in the world and how we can take care of them

#### KS2 - suggested curriculum ideas for the world around us

* the effect of adding components to circuits

[Read the full Northern Ireland curriculum - primary](https://ccea.org.uk/downloads/docs/ccea-asset/Curriculum/The%20Northern%20Ireland%20Curriculum%20-%20Primary.pdf)

#### KS1 & 2 - requirements for using ICT

* explore - investigate, make predictions and solve problems through interaction with digital tools

[Read the full KS1 & 2 - requirements for using ICT](https://ccea.org.uk/downloads/docs/ccea-asset/Curriculum/Curriculum%20Requirements%20for%20Using%20ICT.pdf)

#### Primary using ICT - desirable features - computational thinking and coding

**Level 4**

Pupils should:

* create a more sophisticated coding project using a broad range of commands; and/or
* solve a given problem using commands in a programming environment.

**Programmable devices (such as Parrot Drone, micro:bit or Sphere)**

* look at and talk about examples of coding projects, including the use of motion, looks, lights or sounds, sensors, control and events such as ‘if...then’ and ‘loop until’ (or equivalent) that make the code more efficient;
* recognise that these projects are composed of different components and break the task into smaller manageable tasks (decomposition);
* in small groups, plan and storyboard their own coding project, working out what different parts of the program must do, using logical reasoning to discuss and compare the commands that are required for their algorithm;
* use a range of commands to create a project including triggering commands such as ‘if...then’ and ‘loop until’ to facilitate a more efficient method of interaction;
* test and debug at regular intervals and collaborate with others to solve problems as they arise;

**Finally**

* share their work (possibly using digital tools), respond to feedback and comment on others’ work; and
* organise files and export work in an appropriate format so that others may view it.

[Read all Primary using ICT desirable features](https://ccea.org.uk/downloads/docs/ccea-asset/Curriculum/Primary%20Using%20ICT%20Desirable%20Features%20Update%202019.pdf)

## Curriculum for Wales

#### Science and technology

Progression step 2 - design thinking and engineering offer technical and creative ways to meet society’s needs and wants:

* I can safely use a range of tools, materials and equipment to construct for a variety of reasons

Progression step 2 - computation is the foundation for our digital world:

* I can safely use a range of tools, materials and equipment to construct for a variety of reasons
* I can use computational thinking techniques, through unplugged or offline activities
* I can create simple algorithms and am beginning to explain errors
* I can follow algorithms to determine their purpose and predict outcomes
* I can follow instructions to build and control a physical device

Progression step 3 - computation is the foundation for our digital world:

* I can use conditional statements to add control and decision-making to algorithms
* I can identify repeating patterns and use loops to make my algorithms more concise
* I can explain and debug algorithms
* I can use sensors and actuators in systems that gather and process data about the systems’ environment
* I can effectively store and manipulate data to produce and give a visual form to useful information

[Read the full science and technology curriculum](https://hwb.gov.wales/curriculum-for-wales/science-and-technology/descriptions-of-learning/)

#### Digital competence framework

Progression step 1 - data and computational thinking - problem-solving and modelling:

* I can recognise and follow instructions in the appropriate order to perform a task.
* I can organise, select and use simple language to give instructions to others.
* I can control devices giving instructions.

Progression step 2 - data and computational thinking - problem-solving and modelling:

* I can create and record verbal, written and symbolic instructions to test ideas, e.g. the order of waking up through a diagram or flowchart.
* I can change instructions to achieve a different outcome.

Progression step 3 - data and computational thinking - problem-solving and modelling:

* I can understand the importance of the order of statements within algorithms.

Progression step 1 - producing - evaluating and improving digital content:

* I can comment on work in relation to a single success criterion.

Progression step 2 - producing - evaluating and improving digital content:

* I can give an opinion about my own work and suggest improvements based on the success criteria.

[Read the digital competence framework](https://hwb.gov.wales/curriculum-for-wales/cross-curricular-skills-frameworks/digital-competence-framework)

## USA Code.org

#### CS Fundamentals

Course F

Concepts included:

* variables
* algorithms
* programming
* nested conditionals
* product development

[Read the full CS Fundamentals curriculum](https://code.org/educate/curriculum/elementary-school).

## USA CSTA Standards

#### Grades 3-5

* 1B-CS-01 - Describe how internal and external parts of computing devices function to form a system.
* 1B-CS-02 - Model how computer hardware and software work together as a system to accomplish tasks
* 1B-CS-03 - Determine potential solutions to solve simple hardware and software problems using common troubleshooting strategies.
* 1B-AP-08 - Compare and refine multiple algorithms for the same task and determine which is the most appropriate
* 1B-AP-09 - Create programs that use variables to store and modify data.
* 1B-AP-10 - Create programs that include sequences, events, loops, and conditionals.
* 1B-AP-11 - Decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process.
* 1B-AP-12 - Modify, remix, or incorporate portions of an existing program into one's own work, to develop something new or add more advanced features.
* 1B-AP-15 - Test and debug (identify and fix errors) a program or algorithm to ensure it runs as intended.
* 1B-AP-17 - Describe choices made during program development using code comments, presentations, and demonstrations.

[Read the CSTA Standards in full.](https://csteachers.org/k12standards/ )

This content is published under a [Creative Commons Attribution-ShareAlike 4.0 International (CC BY-SA 4.0)](https://creativecommons.org/licenses/by-sa/4.0/) licence.