**Computing fundamentals  
Lesson 4: Programming 2**

**Introduction**

In this lesson students develop their understanding of algorithms and programming by experimenting with iteration and selection and graphical and text-based programming languages.

**Learning objectives**

* To understand iteration and selection and why they are used
* To develop algorithms and programs using iteration and selection
* To experiment with graphical and text-based programming languages

**You will need**

Lesson plan, lesson guide, paper, [micro:bit MakeCode editor](https://makecode.microbit.org)

**Lesson summary**

1. Write the algorithm (5 minutes)
2. Iteration and selection (10 minutes)
3. Using iteration (15 minutes)
4. Using selection (15 minutes)
5. Experimenting and sharing (10 minutes)
6. Wrap up (5 minutes)

**1. Introduction: Write the algorithm (5 minutes)**

* Give out paper to pairs and show the program on **slide 2**. Ask students to write the algorithm for the program using pseudocode.
* As a class write the algorithm and address any misconceptions (click for example pseudocode).

**2. Iteration and selection (10 minutes)**

* Explain today’s lesson will focus on key concepts in algorithms and programming, sharing the learning objectives on **slide 3** if you wish.
* Invite students to share their current understanding of iteration and selection (**slide 4**).
* Highlight the repetition used in the introductory activity (**slide 5**) and as a class discuss and show how the algorithm and program could be amended to include selection (**slide 6**).
* Invite students to think/pair/share why we use selection and iteration in algorithms & programs (**slide 7**).

**3. Using iteration (15 minutes)**

* Explain you would like students to experiment with using iteration. Firstly, designing an algorithm using iteration, then using their algorithm to write a program using the MakeCode editor (**slide 8**).
* Give out paper and suggest students take 5 minutes to write their algorithm, then 5 minutes to program, testing and debugging as they go (you can set a timer to give prompts if you wish).
* As they are writing their algorithms and programs, look for good examples and when the time is up, invite students to share these with the class and others to share their learning.

**4. Using selection (15 minutes)**

* Repeat the above for selection (**slide 9**), either extending the current algorithm and program, or creating another.

**5. Graphical and text programming languages (10 minutes)**

* Invite students to recap what kind of programming language the MakeCode editor is (graphical) and what other type there is (text-based), using **slide 10** to recap and illustrate.
* Show students how to access the JavaScript version of the program they have created and ask them to view and experiment with the JavaScript version of their program, going between the graphical and text-based language to make changes and view the impact.
* Discuss as a class, inviting students to share their observations and insights.

**6. Wrap up (5 minutes)**

* Show the key words on **slide 11** and give students 1 minute each to discuss in their pairs what they now know about each one (you can set a timer for a minute each, or swap the starting person for each word to ensure equal participation).
* If you have time, share as a class and review the learning objectives if you wish on **slide 12**.

**Extension ideas:**

* You could introduce the [micro:bit Python editor](https://python.microbit.org/v/1.1) and get students to experiment with using that (this could also be a stretch and challenge activity).
* Students could swap their algorithms to test and debug and/or program, giving feedback to each other.
* Students could record a short video/screen recording to talk through their algorithms and code, highlighting where they have used selection and iteration to explain their understanding, or they could print and annotate their code.

**Differentiation**

**Support:**

* Ask students to focus on simple iteration and selection, giving further examples and support if needed.
* Through discrete questioning and observation, ensure a clear understanding of the basic concepts and confident use in algorithms and programs before moving to more complex use.

**Stretch & challenge:**

* Encourage students to create algorithms and programs with more complex use of iteration and selection (and combining both).
* Also see extension activities.

**Opportunities for assessment:**

* Informal observation of students’ during activities, discussions and wrap up.
* More formal assessment of students’ algorithms and programs.