**Introduction to cyber security  
Lesson 3: Making a password generator**

**Introduction**

In this lesson students use their pseudocode algorithm to code their password generator before testing, debugging and evaluating it.

**Time:** @60 minutes

**Learning objectives**

* To follow a pseudocode algorithm to program a password generator using a micro:bit
* To write a program using variables correctly
* To test and debug code to create a working password generator

**Materials needed:** Lesson plan, lesson guide, rough paper, algorithm plans from previous lesson, coding support and password generator evaluation sheets (online or on paper), [MakeCode editor](https://makecode.microbit.org)

**Lesson summary**

1. Recapping challenge (5 minutes)
2. Recapping code (optional: 10 minutes)
3. Coding a password generator (20+ minutes)
4. Sharing, testing and debugging (10 minutes)
5. Evaluating programs (10 minutes)
6. Review & wrap up (10 minutes)

**Introduction: Password generator challenge recap (5 minutes)**

* Invite students to recap the password generator challenge and their tasks for this lesson (**slide 3**).

**Recapping code (10 minutes)**

* If you feel your students would benefit from recapping using selection and variables prior to coding, use **slides 4 and 5.**
* Completing the Dice and Rock and Paper and Scissors tutorials on [MakeCode](https://makecode.microbit.org/) will also help to familiarise them with using variables.

**Coding the password generator (20+ minutes)**

* Ensure students have their plans from the previous lesson, give out the **coding support sheets** as needed, and give students ample time to program their password generator.
* If working in pairs, encourage students to use paired programming (**slide 6**)
* If you have physical micro:bits, once students have a working program, give them time to connect micro:bit, test their password generator and debug as necessary.

**Sharing, testing and debugging (10 minutes)**

* As students finish, invite them to pair up, swap password generators and test each other’s, using **slide 7** to support discussion and debugging as needed.

**Evaluating programs (10 minutes)**

* Give students access to the **password generator evaluation sheet**, ideally on the school network and ask them to evaluate their password generator (**slide 8**).

**Review and wrap up (5 minutes)**

* Review the learning objectives if you wish on **slide 9** and invite students to share their answers to the last 2 questions on the evaluation sheet to review learning.
* An [example password generator program hex file](https://makecode.microbit.org/#pub:_WyUJC7YTg3H2) is also provided.

**Differentiation**

**Support:**

* Provide students with the **coding support sheet** which has starter for the code they will need. They can then edit and add more code to complete their password generator.
* Students could be sensitively paired with a more confident coder who can support them through paired programming.
* Students can be encouraged to make simple evaluative statements in the evaluation activity and given starters to their sentences if needed.

**Stretch & challenge:**

* Challenge the students to use the shake function to create characters and add sound effects for their password generator and make greater use of variables.
* Students can be encouraged to make more insightful, evaluative statements when evaluating their program.
* Students could extend their learning about Cyber Security further by playing the game on the following website: <https://go.joincyberdiscovery.com/tutorial>

**Opportunities for assessment:**

* Informal observation of students during activities.
* Formal assessment of students’ completed programs and evaluation sheets.