**Learning to loop**

Years 5–6



**Summary**

Students create algorithms with a condition that tells the computer to repeat a sequence of instructions.

**Suggested steps**

Divide the class into two teams. Invite students to play a simple game involving repeated actions; for example:

1. Run from the centre of the netball court to the goal ring.
2. Collect the netball.
3. Attempt to throw the ball into the hoop.
4. Attempt to throw the ball into the hoop.
5. Attempt to throw the ball into the hoop.
6. Attempt to throw the ball into the hoop.
7. Attempt to throw the ball into the hoop.
8. Count up your score.
9. Run back to the centre of the netball court.
10. Give the ball to the next player.
11. Start again.

The winner is the team that has thrown the most goals after each player has had five attempts.

### Write or draw an algorithm

After playing the game, ask the students to write and/or draw their own version of the instructions (algorithm) to show others how to play. Invite students to share their algorithms and, through discussion, introduce the ‘repeat’ command. Demonstrate how the number of steps can be reduced by including ‘repeat’ in the algorithm. Ask students to consider how the score is counted and the game ends. They may include a command such as ‘repeat until each player has had a turn’; so their algorithm might include two loops; one nested inside the other.

### Variation

Create an algorithm for different game play; for example:

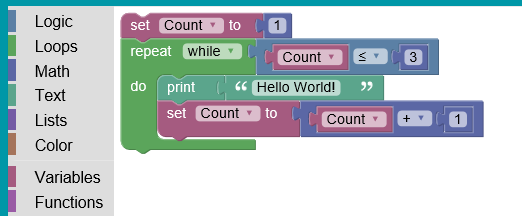
1. The winner is the team that is quickest to have a player score a goal.
2. The winner is the team with the most goals, with each player having only 30 seconds to shoot.

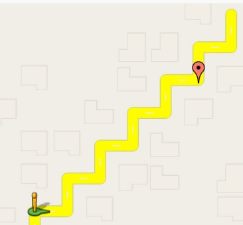
### Examples of repetition

Ask students to share examples of how repetition is used in familiar games or quizzes; for example, what actions are repeated and how do players know when a game has come to an end? How do we replicate this when coding a game or quiz? Explain that a common aspect of repetition is the inclusion of a condition that lets the computer know how many times to repeat a sequence of instructions.

Explain that repetition (iteration) is used to make programs more efficient.

Show students examples of blocks of code in visual programming environments where the repeat loop is used to repeat a single instruction multiple times; for example, environments such as Blockly, Snap or Scratch.







Ask students to work in small groups, and use a visual programming language to create their own simple maze, quiz or game within the context of a learning area. Algorithms must include repeat instructions and structures such as reading user input until a correct answer/successful outcome is given. Students need to consider the sequence of the blocks of code and functional requirements such as navigation buttons.

## Discussion

Encourage students to interact with each other’s games, and to consider the following:

* Where might the program use loops (repetition)?
* What is the benefit of using Repeat blocks?

Reinforce the students’ understanding of the concept of repetition (iteration). Repetition allows us to complete an action multiple times. Explain that when programming we use repetition to solve problems more efficiently.

|  |
| --- |
|  |

## Why is this relevant?

In the Digital Technologies curriculum, students in years 5 and 6 are expected to include repetition (iteration) in their programming solutions, as well as decision-making and user-interface design. When creating their solutions students increase the sophistication of their algorithms by identifying repetition and incorporating repeat instructions.

Repetition is commonly used when creating algorithms and is an important concept of computational thinking. Repetition allows us to complete a common task multiple times, even when the information or context changes. A common component of repetition is a condition that tells the computer how many times to repeat a sequence of instructions.

Examples of repetition in the behaviour of software systems include the algorithm that controls the behaviour of the scanner at the supermarket; and the algorithm that controls the behaviour of robotic devices.

**Peer assessment**

Students interact with each other’s games and evaluate success via a simple checklist, keeping in mind that the key focus is the ability to incorporate repetition within a program.

Checklist questions include:

* Are the instructions easy to follow?
* Can the user interact easily with the game?
* Is the game fun/entertaining?
* Does the game include examples of repetition (looping)?

**Curriculum links**

**Links with the Digital Technologies curriculum area**

|  |  |
| --- | --- |
| **Strand** | **Content description** |
| **Processes and Production Skills** | Design, modify and follow simple algorithms involving sequences of steps, branching, and iteration (repetition) (ACTDIP019) |
| **Processes and Production Skills** | Implement digital solutions as simple visual programs involving branching, iteration (repetition), and user input (ACTDIP020) |

**Links with other curriculum areas**

Students can create games and quizzes within most curriculum areas. For example, in years 5–6:

* HASS (Geography): Students can create a quiz to support people within the community to prepare for a severe weather event
* Civics and Citizenship: Students can create a quiz to test knowledge of the three levels of government in Australia; or the key institutions of Australia’s democratic system of government and how it is based on the Westminster system.