Repetition

CHAPTER 2

Definition

Loop: A sequence of instructions that are repeated.

Language of Loops

Looping is not the only word used for this concept. Other popular ones include iteration and repetition.

Everyday Repeating Actions

Pupils already have a basic knowledge of everyday repetition. This may be the lyrics of a chorus in a popular song or the actions in a dance craze. Then there are the many mundane tasks that involve repetition: washing up, cleaning, decorating, to name a few. Making these links with pupils' established schema of understanding grounds our new interpretation of repetition to established knowledge.

Programming Loops

Count-controlled loop

A count-controlled loop is one that has a number that controls how many times the loops will run for. This loop might have one or many instructions that are repeated.

Count-controlled loop in algorithm

Fill glass with water
Loop 5 times
    Drink water
    pause

Count-controlled loop in code

Act Two Times Same as Act, Act

Helping pupils see that a sequence of the same commands can be replaced with a

count-controlled loop and vice versa is important for understanding both why we might choose to loop (less instructions, more elegant code) and how it works.

Jump
Jump
Jump

Same as

Loop 3 times
    Jump

Do Twice
    Stand
    Sit

Same as

Stand
Sit
Stand
Sit

Role-playing simple count-controlled loops in this manner before writing their own to test their understanding, followed by converting a simple sequence to a loop and vice versa, are useful steps in comprehending how a simple count-controlled loop works.

Count-controlled loops have a definite end. Once the number of loops defined by the number is complete they end.

Indefinite Loops

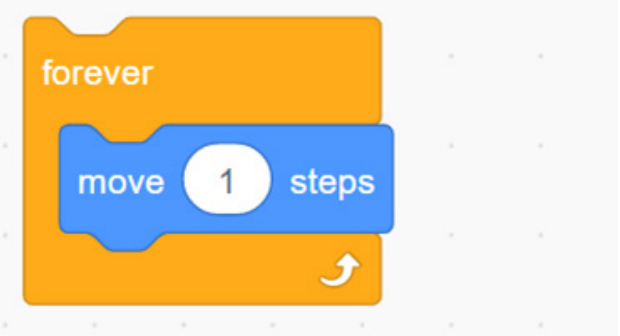
Indefinite loops are ones where we don't know when they are going to end or how many loops they will complete.

The simplest indefinite loop in Scratch is the forever or infinite loop.

Infinite loop in algorithm

Loop always
    Check your phone

Infinite loop in code



Unlike a count-controlled loop, the infinite loop has no ending, which means that no programming structure can be built after it. Most programming languages do not have infinite loops, but it is a very useful stepping stone towards greater complexity.

Condition-ends-loop

A condition-ends-loop is also an indefinite loop in that we do not know when it will end or how many times it will repeat.

Unlike our infinite forever loop there is a way of ending the loop using a condition.

Condition-ends-loop in algorithm

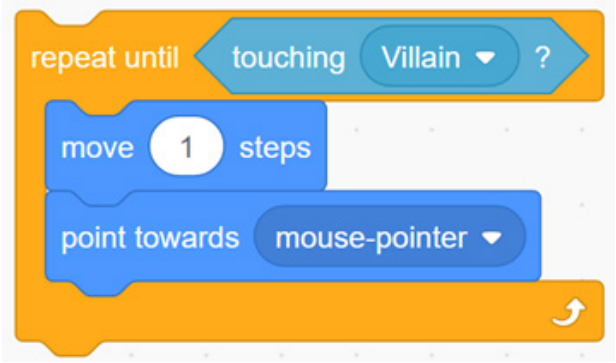
Start eating food

Loop until full up

- Eat

You might have noticed that I have chosen to indent each action inside a loop algorithm. This helps the reader to know what is inside the loop. An algorithm can be written in any way the writer chooses, and so it is possible to choose some other way to show what is inside a loop, such as bullet points, as long as it is clear to another human reading it.

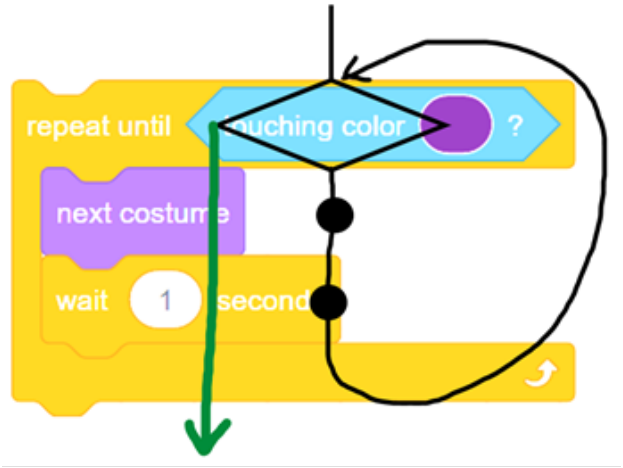
Condition-ends-loop in code



Loop Ended by a Condition

Condition-ends-loop flow of control

Many pupils believe that as soon as the condition is met the loop will end. Whilst this is generally true, it actually only ends if the condition is true at the moment it is checked in the flow of control.



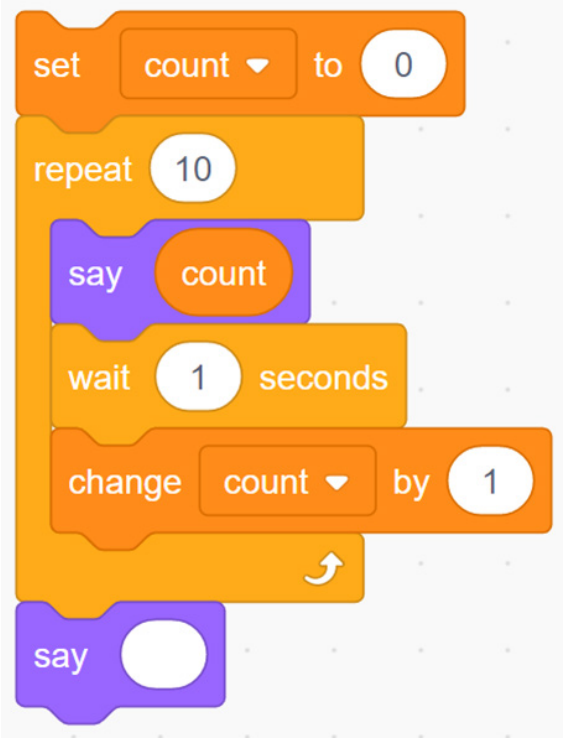
Flow of control in a condition-ends-loop

In our flow of control example above, if our sprite was touching the colour purple while waiting one second but was not touching purple when the condition is checked, the loop would not end.

The condition is checked represented by the diamond, as we have not touched the colour purple we proceed down the black line to run next costume (dot) and wait (dot). The loop then goes back to check the condition (diamond) if we are touching purple we would exit the loop via the green line. (To help you understand this more, read chapter 27 on flow of control.)

Cumulative effect

Repetitive tasks such as washing up, cleaning and decorating, all have a cumulative effect, they all build on the previous action to achieve a greater purpose be that clean dishes, clean home or fresh painted wall. Some, but not all, programming uses of repetition will share this attribute. In this counting program a count-controlled loop has a clear cumulative affect, as the number increases from 0 to 9.



(You might want to read chapter 4 on variables, to help you understand this more.)

Variables Used to Control Loops

We have already seen that a variable can be used to create a cumulative affect and it can also be used to control the number of times a loop repeats.

In this example the user is asked how many sides they would like the shape to have. Their response is assigned to a variable called **no_sides**, which is short for number of sides. The value is then used to set how many times the count-controlled loop repeats and to divide 360 by the number assigned to the **no_sides** variable.

So if 3 is input by the user then the loop will repeat 3 times and turn 360 divided by 3, which makes 120 degrees, drawing a triangle.

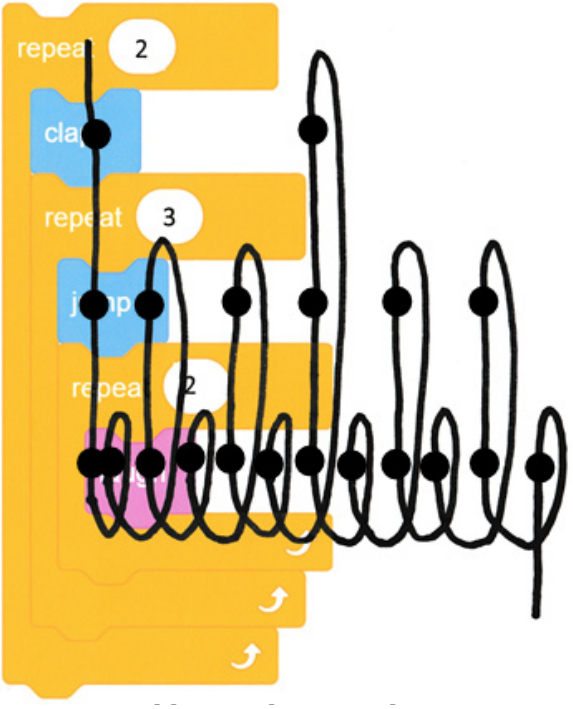
A variable controlling number of repetitions

Loops Can be Nested

The everyday computing team put nested loops as their highest complexity aspect of primary/elementary repetition. You can view their trajectory at the end of this chapter. It is useful to reclassify this as more complex, as it was often taught soon after introducing count controlled loops in many primary/elementary curriculums. Whilst pupils can often nest loops to create wonderful patterns, they are often unable to explain how they work if introduced too early on.

Nested Loops Multiply

Loops that are nested inside other loops have a multiplying effect.



Nested loops shown using one flow of control notation

In this example we can see that the jump command is within two loops, and the repeat 3 loop is inside the outer repeat 2 loop. This means that both loops count values are

multiplied by each other for the jump action, leading to 2×3 = 6 jumps.

Action	Maths	No. loops	Times run
clap	2×	1	2
jump	2×3	2	6
laugh	2×3×2	3	12

Can you calculate how many laughs (bottom purple block) there would be?

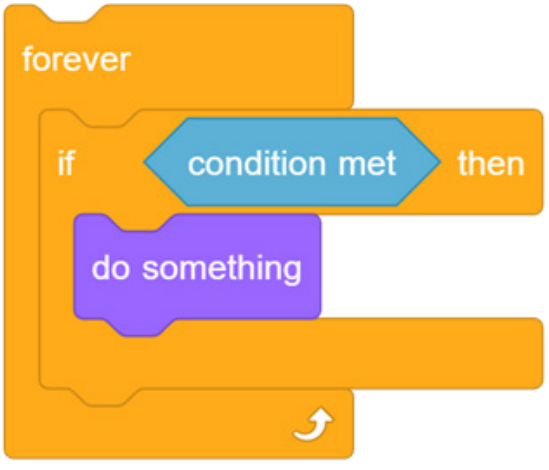
How Does it Stop?

Once you have more than one type of loop in your programming backpack you can ask pupils how the loop stops? This is more than just a formative assessment opportunity as it gets to the heart of what type of loop to use when. Count-controlled loops are useful where you want something to take place for a limited amount of time or distance. Infinite loops are useful if you don't want something to end.

Loops & Conditions

Loops have a very close relationship with conditional selection. They allow conditions to be acted on indefinitely. So much so that early Scratch had a block called **forever-if** that combined both aspects. Once conditions and indefinite loops have been introduced separately, the next step is to combine them to use them in gaming modules of work.

(You can read more about this on the chapter 3 on conditional selection.)



A condition checked inside a loop

Order of Introduction

1. Count-controlled loop
2. Indefinite-loop
3. Loop-ended-by-a-condition
4. Conditions-checked-within-a-loop
5. Loop-controlled-by-variable
6. Nested-loops

This is my research-informed order to introduce loop types. This is based on my interpretation of the repetition trajectory from the Everyday computing team which is printed at the end of this chapter. It is also informed by the complexity of role-playing and writing repetition algorithms and working with flow of control for each of these loop types.

In my opinion, 3 and 4 are of a very similar complexity; if I have reduced time I leave out 3.

Teaching Primary Programming With Scratch

Teacher Book – Research-Informed Approaches

PHIL BAGGE

Published in 2022 by University of Buckingham Press,
an imprint of Legend Times Group
51 Gower Street
London WC1E 6HJ
info@unibuckinghampress.com
www.unibuckinghampress.com

Published by arrangement with Hampshire Inspection and Advisory Service (part of Hampshire County Council)

ISBN 978-1-91505-4-203

CONTENTS

Processes

More Support