



# *For and While Loops*

Robotics Curriculum  
IT Adventures



# *Lesson Overview*

- Learn about while and for loops
- Learn how to create loops in micro:bit code
- Combine loops with previous lessons (inputs and Boolean logic) to control the RVR



# Loops... and Loops... and Loops... and Loops...

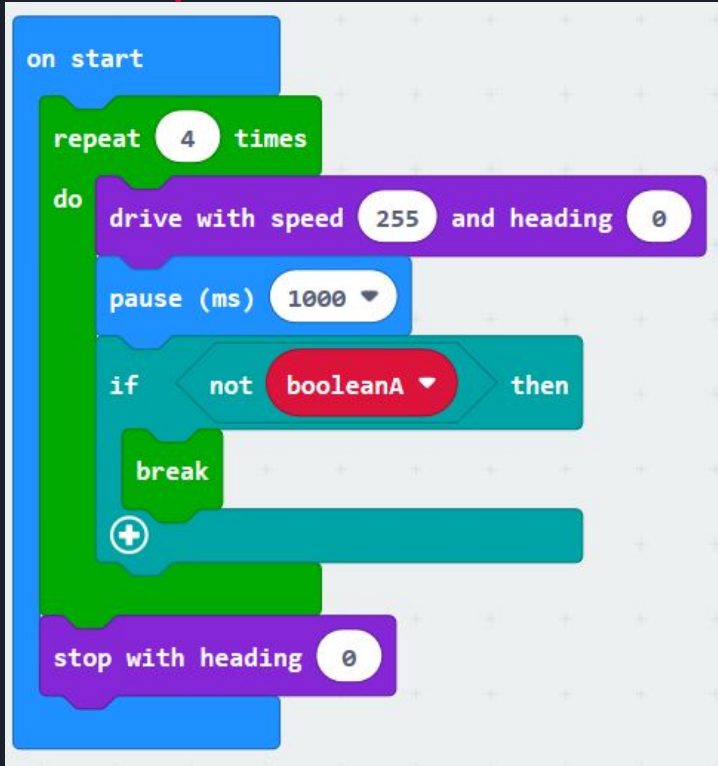
Loops are essentially as they sound: they will repeat the code that they are given until a certain condition is met. There are two *main* types of loops:

For loops run *for* a certain number of sequences or through a certain set of events/variables. These are especially powerful with arrays, which we'll dive into during the next lesson. For now, know that they can be used to iterate a certain number of predefined times.

While loops will run *while* a certain condition is met. While this can be more flexible, it is also worth noting that if the condition is never met (or is impossible to meet), the code *will* run forever. On the flipside, if the condition is not met upon entering the loop for the first time, the code is skipped entirely.

Note: Loops analyze their condition only at the beginning of each loop, so if the condition is not met for a brief period of time while running the loop, but is back to being true by the time the loop is about to start over, it will run the code again. This is especially important to keep in mind if the program is looking for something, such as a line. If the loop is too long, or the code isn't correct, it could see the line but skip over it.

# Code Example



```
on start
  repeat 4 times
    do
      drive with speed 255 and heading 0
      pause (ms) 1000
      if not booleanA then
        break
      +
    stop with heading 0
```

The image shows a Scratch code editor with the following blocks: an 'on start' block, a 'repeat 4 times' loop block, a 'do' block containing 'drive with speed 255 and heading 0', 'pause (ms) 1000', an 'if not booleanA then' block containing a 'break' block, and a 'stop with heading 0' block. A plus sign is visible at the end of the 'if' block's 'then' clause, indicating a continuation point.

In the code to the left, an extremely simple loop will run 4 times. This code highlights how simple loops can be. If you explore under the “Loops” tab in MakeCode, you will see that this is simply the first example of a loop, with *while* and *for* loops also being present. Try out some of them! As a part of the code, though, there is an *If* statement with a *break* in it.

*break* will break out of the loop immediately, instead of waiting for the looping to end. On the other hand, there is a *continue* statement as well, which will skip the rest of the code in the current loop, and start back at the top.




# Primary Learning Challenge: Dizzying Heights

The primary challenge for today is to make the RVR run in a circle... or rather, to make it run in a circle *until* a button is pressed. It is completely up to you to define the speed of the RVR, what button needs to be pressed, and more. You are limited in two ways:

- 1) You may only use one loop (think carefully about which loop would apply in this situation)
- 2) You may only use one move command

The only other thing to keep in mind is speed - while you can make it faster, remember that the faster it goes, the harder it is to press the button!

You can optionally add on to this challenge by having it so that when the button is pressed a second time, it will resume running in a circle.



## Secondary Learning Challenge: Spiraling In

Similar to the primary challenge, your goal is to only use a single loop and move command to drive the RVR. However, your goal this time is to drive the RVR in a spiral inwards. Think carefully about what type of loop works for this challenge, what kinds of limits you may need to place, and more. Good luck!

