Iteration in programming

When designing programs, there may be some instructions that need repeating. This is known as iteration, and is implemented in programming using FOR and WHILE statements.

Iteration

An algorithm is a plan, a set of step-by-step instructions designed to solve a problem. There are three basic building blocks (constructs) to use when designing algorithms:

- sequencing
- selection
- iteration

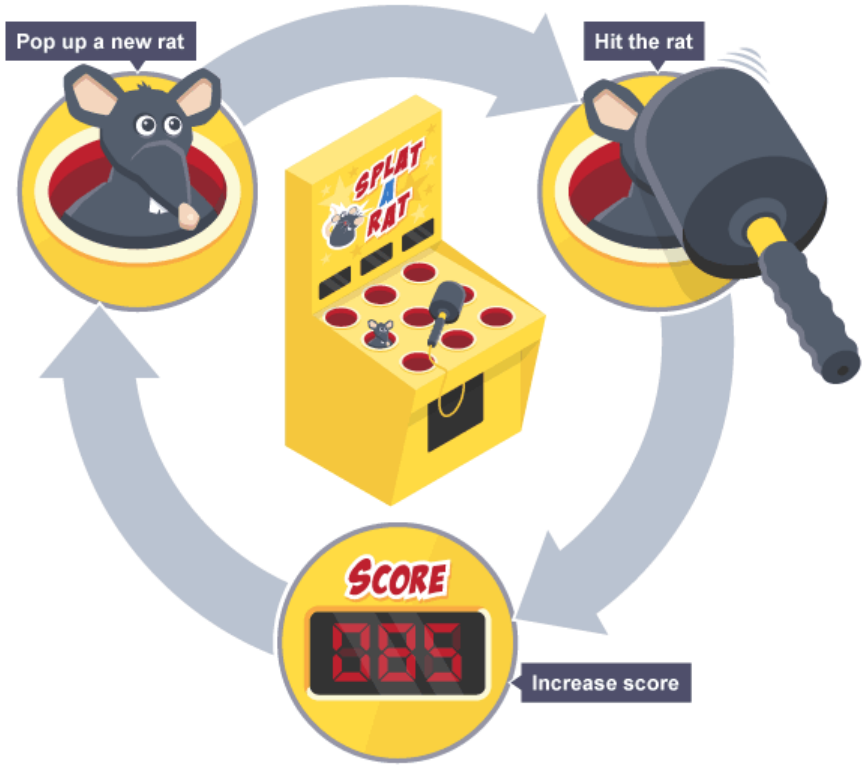
Algorithms are used to help design programs that perform particular tasks.

What is iteration?

An explanation of iteration, as used in algorithms and programming

Algorithms consist of steps that are carried out (performed) one after another. Sometimes an algorithm needs to repeat certain steps until told to stop or until a particular condition has been met.

Iteration is the process of repeating steps.



For example, a very simple algorithm for eating breakfast cereal might consist of these steps:

1. put cereal in bowl
2. add milk to cereal
3. spoon cereal and milk into mouth
4. repeat step 3 until all cereal and milk is eaten
5. rinse bowl and spoon

The algorithm will repeat steps 3 and 4 until all the cereal and milk has been eaten.

Why is iteration important?

Iteration allows us to simplify our algorithm by stating that we will repeat certain steps until told otherwise.

This makes designing algorithms quicker and simpler because they don't have to include lots of unnecessary steps.

Iteration in programming

Once an algorithm has been designed and perfected, it must be translated – or programmed – into code that a computer can read.

We create programs to implement algorithms. Algorithms consist of steps. Programs consist of statements. A statement is a single instruction - in other words, a single step.

Iteration is implemented in programming using **FOR** and **WHILE** statements.

In programming, iteration is often referred to as 'looping', because when a program iterates it 'loops' to an earlier step.

Count-controlled loops

There are two ways in which programs can iterate or 'loop':

- **count-controlled** loops
- **condition-controlled** loops

Each type of loop works in a slightly different way and produces different results.

Count-controlled loops

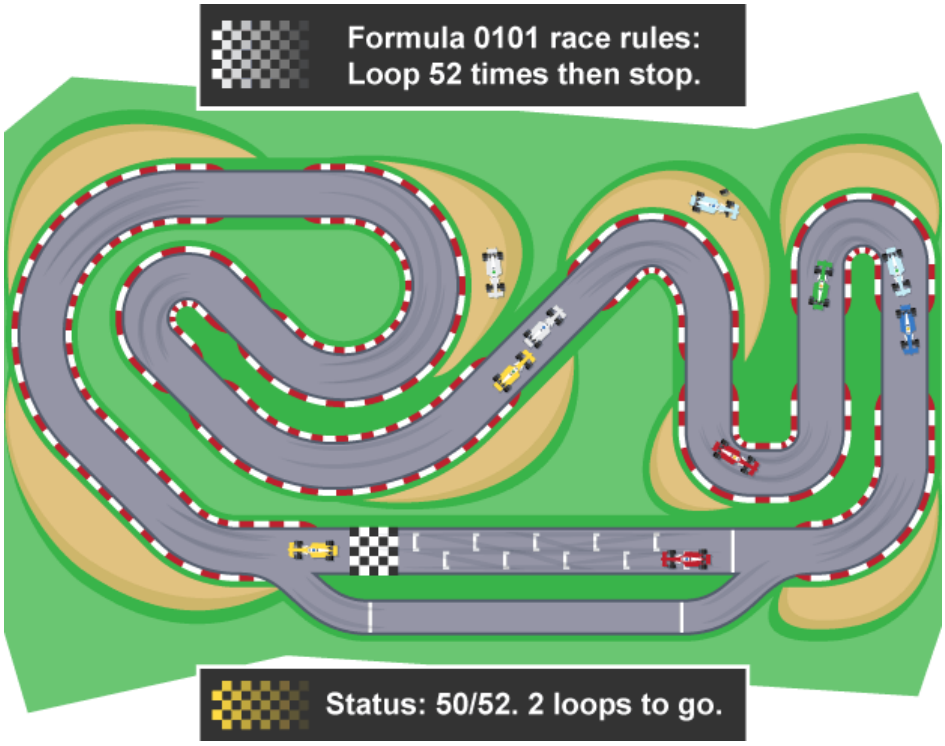
Sometimes it is necessary for steps to iterate a specific number of times.

Consider this simple algorithm for adding up five inputted numbers:

1. set the total to 0
2. repeat this section five times
   - input a number
   - add the number to the total
3. go back to step 2
4. say what the total is

This algorithm would allow five numbers to be inputted and would work out the total. Because it is known in advance how many times the algorithm needs to loop, a count-controlled loop is used.

A count-controlled loop is used when the number of iterations to occur is already known.



A count-controlled loop is so called because it uses a counter to keep track of how many times the algorithm has iterated. The pseudocode for this algorithm might look like this:

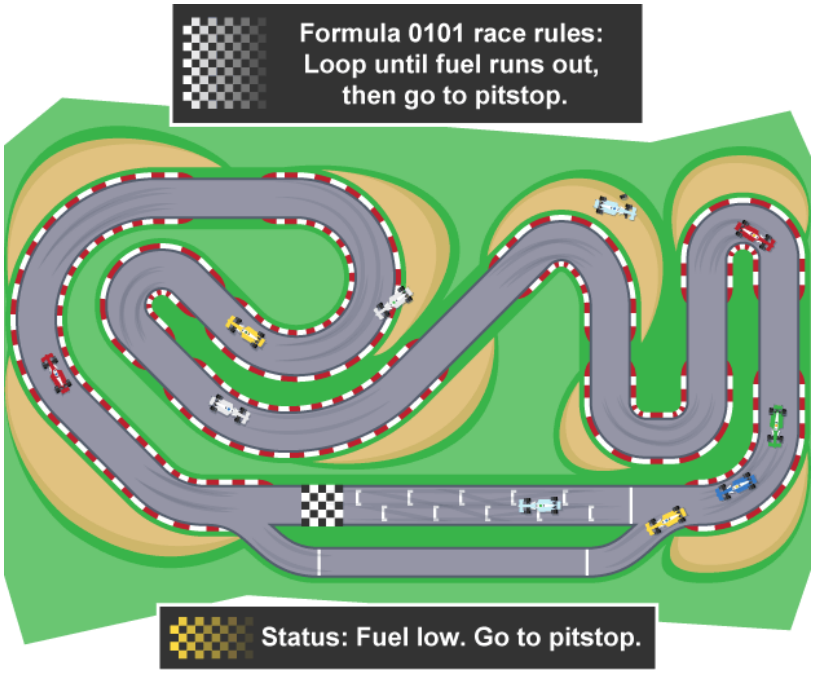
total to 0 count to 1 FOR as long as count is in the range 1 to 5 INPUT user inputs a number STORE the user's input in the number variable total = total + number Add 1 to count OUTPUT "The total is " + total

Steps that are part of the loop are indented. Indentation is used to show which steps are to be iterated.

In this example, the variable ‘count’ is used to keep track of how many times the algorithm has iterated. This variable controls the loop. The algorithm will continue to iterate until the value of count has reached 5. As soon as count reaches 5, the algorithm stops iterating.

Condition-controlled loops

A condition-controlled loop is so called because iteration continues while, or until, a condition is met.



Consider this simple algorithm for entering a correct password:

1. enter password
2. unless password = “ilovecomputing”, go back to step 1
3. say ‘Password correct’

This algorithm would keep iterating until the password is entered correctly. A condition-controlled loop must be used because there is no way of knowing in advance how many times a password will need to be entered before it is correct.

A condition-controlled loop is used when it is not known how many times iteration will need to occur.

The pseudocode for this algorithm might look like this:

password = "blank" WHILE password does not equal "ilovecomputing" INPUT password OUTPUT "Password correct"
Steps that are part of the loop are indented. Indentation is used to show which steps are to be iterated.

In this example, the condition is whether or not the inputted password equals “ilovecomputing”. **The algorithm tests the condition to see if it is true.** If true, the algorithm outputs a message. If false, the algorithm loops back to the beginning and will continue to do so until the tested condition is true.

Count-controlled loops - using FOR

Sometimes an algorithm needs to iterate steps a specific number of times. In programming, count-controlled loops are implemented using **FOR** statements. Python uses the statements **for** and **range** (note the lowercase syntax that Python uses):

- **for** determines the starting point of the iteration
- **range** states how many times the program will iterate

Consider this simple algorithm for adding up five inputted numbers:

1. set the total to 0
2. repeat this section five times
   - input a number
   - add the number to the total
3. go back to step 2
4. say what the total is

This algorithm would allow five numbers to be inputted and would work out the total. A count-controlled loop would be used because it is known, in advance, how many times the algorithm needs to loop.

A count-controlled loop is used when it is known how many times iteration will need to occur.

The Python (3.x) code for this algorithm would look like this:

total = 0 for count in range(5): number = int(input("Type in a number: ")) total = total + number print("The total is: ") print(total)
Steps that are part of the loop are indented. Indentation tells the computer which steps are to be iterated.

The program works like this:

- The program uses the variable ‘count’ to keep track of how many times the iteration has occurred.
- The ‘**for**’ statement is used to specify where the loop starts.
- The ‘**range**’ statement specifies how many times the loop is to happen.
- The variable ‘count’ controls the iteration.
- With each iteration, Python automatically adds 1 to the value of ‘count’.

- The program keeps iterating as long as the value of ‘count’ is in the range specified in the loop (in this case as long as ‘count’ is in the range 0 to 5). Once it reaches the end of the range (5) the iteration stops.

This program iterates five times. To have a different number of iterations, the value ‘5’ would simply have to be changed to the number of times required.