Introduction to Loops

> In programming, repetition of a line or a block of code is also known as **iteration**. **A loop is an algorithm that executes a block of code multiple times till the time a specified condition is met.** Therefore, we can say that a loop iterates a block of code multiple times till the time mentioned condition is satisfied.

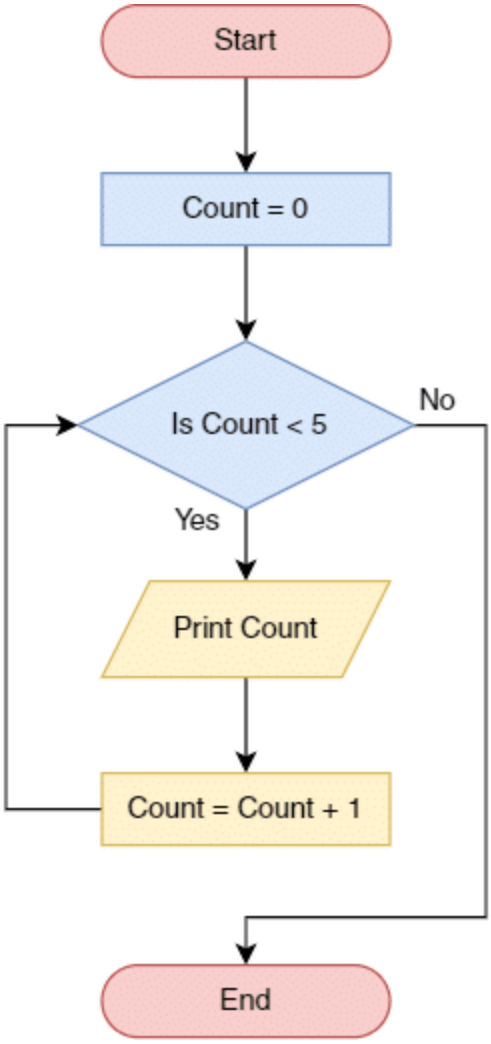
Consider a requirement where we need to print numbers in incremental order from 1 to 1000. Although it is possible to print it with each line of code written, it will get a very tedious and lengthy process. This is where loops come into the picture to make this task easier. You can use the concept of loops and get the desired output by writing just a few lines of code.

Increment Loops

Loops provide the facility to execute a block of code repetitively, based on a condition.

To run a block of code in a loop, one needs to set a condition and set its number of iterations. Each time the condition is true, and the block of code executes once, it is counted to be one iteration. Before moving to the next iteration, one needs to increase the count of iterations to two. This is called incrementing a loop.

For example, if you need to print numbers 0 to 4, you will execute a block of code with the Print statement in five iterations. With each passing iteration, you will increment the count by one.



Let us understand loops with a flowchart:

Here **every time the condition (Count < 5) is true, “Print count” gets executed**. So, we do not have to write the “Print” statement multiple times. The loop takes care of that.

What is important to note is **every loop must have an exit condition**. In our example, the exit condition is (Count < 5). **The loop will exit when the condition becomes false.**

Also, most loops will have **a variable which in programming terms is called a counter variable**. **The counter variable keeps track of how many times the loop is executed**. In this example, the “count” variable is our counter.

Benefits of Loops

Below are the two important benefits of loops:

Reduces lines of code
Code becomes easier to understand

Different types of loops

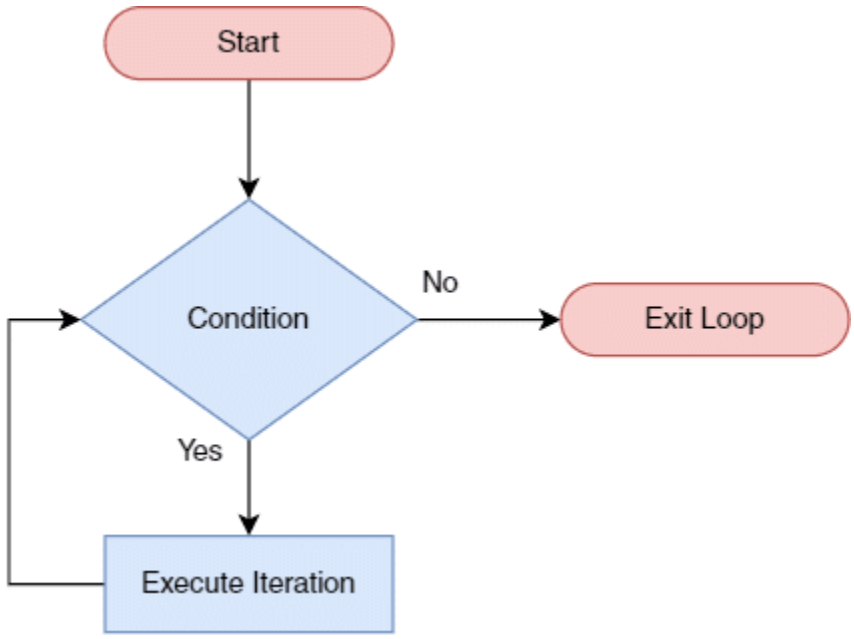
Loops make our code more manageable and organized. Let us now see what the different types of loops are:

While Loop
For Loop
Nested Loop
If Loop

While Loop

The While loop can execute a set of commands till the condition is true. While Loops are also called conditional loops. Once the condition is met then the loop is finished.

General Flow of While Loop



Implementing While Loop in PictoBlox

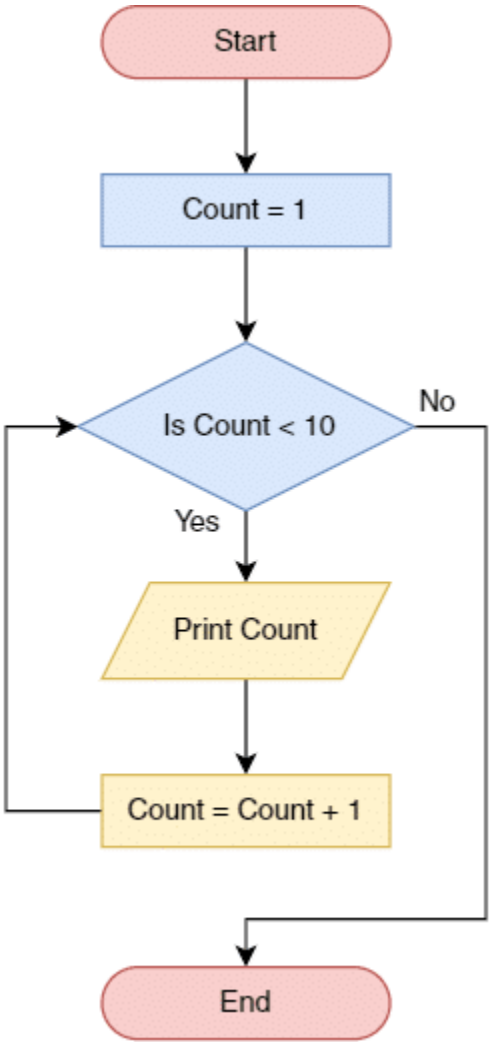
The **Repeat Until ()** block is a **Control** Block and a **C** block. Blocks held inside this block will loop until the specified boolean statement is true, in which case the code beneath the block (if any) will execute. This loop is in similar nature to a while loop.


repeat until

Activity 1

Let’s create a code to print 1-9 numbers

Flow Chart



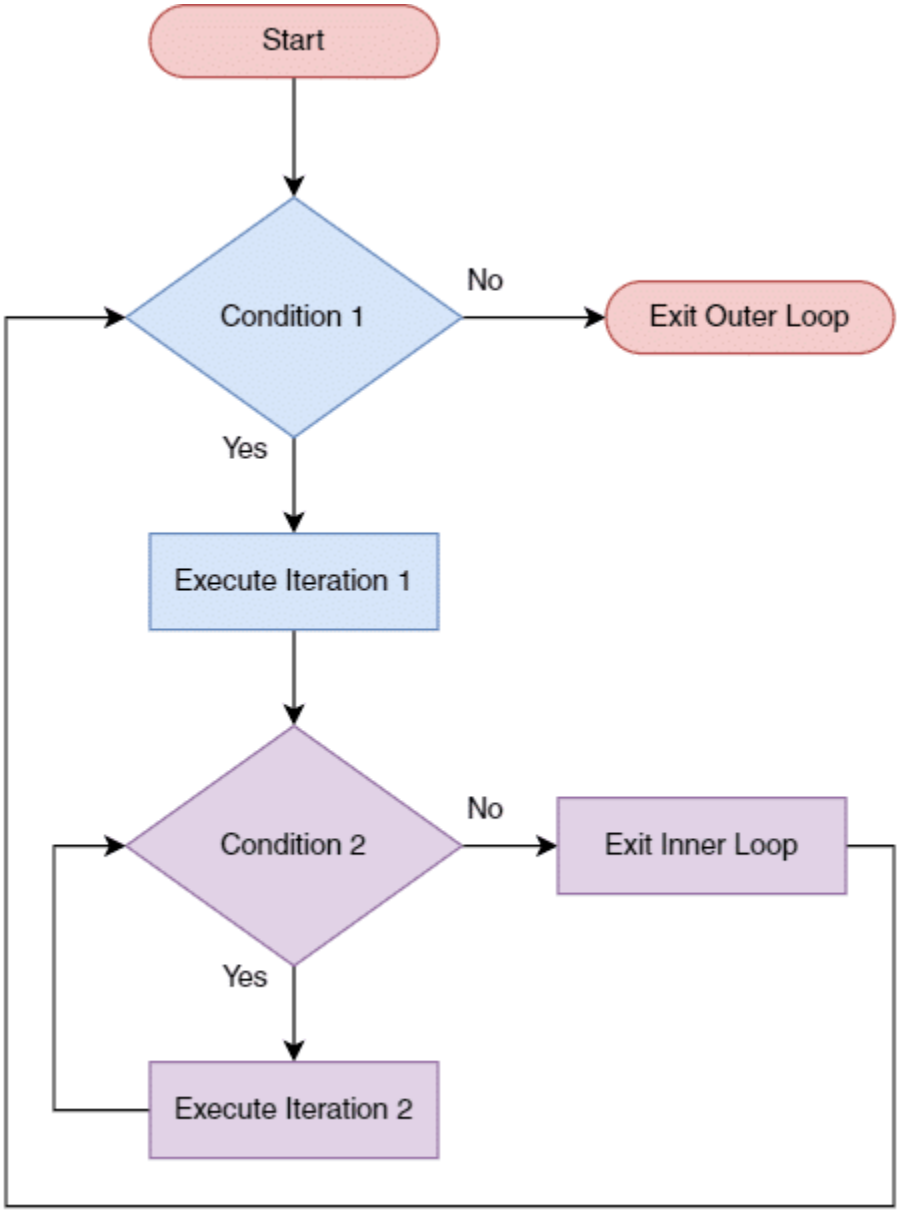
Nested Loop

> Any loop in the program may contain another loop inside it. When there is a loop inside another loop, it is called a nested loop.

How it works is that the first success condition of the outer loop triggers the inner loop which runs and reaches completion. This combination of loops inside the loop is helpful while working with requirements where the user wants to work on multiple conditions simultaneously.

There is no restriction on how many loops can be nested inside a loop.

General Structure of Nested Loop



Example of a Clock