Introduction to Loops

by Sophia

WHAT'S COVERED

In this lesson, you will learn about the basics of loops. Specifically, this lesson covers:

1. Iteration

Computers are often used to automate repetitive tasks. Repeating identical or similar tasks without making errors is something that computers do well, and people do poorly. Iteration is one of the most important concepts in looping.

KEY CONCEPT

Iteration is the repetition of a sequence of steps.

Iteration is so common that Java provides several language features to make iteration easier to use. Loops allow us to repeat code multiple times efficiently and without unduly repetitious code. A programming feature that implements iteration well is called a loop.

In programming, there are two types of iteration with loops, indefinite and definite iteration.

1a. Indefinite Iteration

When using **indefinite iteration**, users do not specify the number of times that the loop is meant to run in advance. Instead, the loop would run repeatedly, as long as a condition is met. This is very similar to a selection statement that is checked each time. Think about a video game menu that keeps repeating the selections until the user quits the game by choosing the option to exit. In Java, indefinite loops are created using any loop. This tutorial will focus on the use of the while loop.

TERM TO KNOW

Indefinite Iteration
A loop that does not specify, in advance, how many times it is to be run. It repeats as long as a condition is met. In Java, indefinite loops are typically created using the while loop.

1b. Definite Iteration

When using **definite iteration**, the number of times the block of code runs and the maximum number of times that the block of code is allowed to run should be explicitly defined when the loop starts. This is typically implemented using the for loop. The for loop has a specific start and endpoint.

KEY CONCEPT

When working with any kind of loop, it is important the determine the following:

- Which steps in the process are not repeated and need to happen before the loop starts repeating a sequence of steps. This sets up the context in which the loop will run.
- Which steps need to be repeated, and thus should be in the body of the loop.
- Which steps in the process happen only once after the end of the loop (i.e., after the loop is done repeating).

TERM TO KNOW

Definite Iteration
Definite iteration identifies the number of times, or the maximum number of times, the block of code runs. This should be explicitly defined when the loop actually starts. Typically, this is implemented using the for loop. The for loop has a specific start and endpoint.

2. Basics of a while Loop

The **while loop** is one of the most commonly used loops. A while loop continually evaluates a condition looking for a `true` or `false` value. It keeps looping as long as the evaluated condition is `true`. A while loop is used when the number of times the loop will run, or the maximum number of times the loop will run, is not known at programming time. This is an example of indefinite iteration.

Let’s first look at the structure of the while loop. The format of a while loop looks like the following:

EXAMPLE

```
while(<expression>) {
 <statement(s)>
}
```

As you learned in the tutorial about selection statements, if the body of the loop consists of just one statement, the curly brackets are not required. However, if the code is modified so that the body of the loop consists of more than one statement, and the curly brackets are not added, the code will behave in unexpected ways. As a general rule, it is best to put in the curly brackets in all cases, since using them is never wrong, and they just take up a small amount of space on the screen. They make the structure of the program clearer (especially for beginners).

In the example above, the `<expression>` and `<statement(s)>` are just for information purposes; these are not keywords or actual code. These are just to explain what goes into each of the parts of a while loop.

The `<statements(s)>` term represents the block of code that should be executed repeatedly. This is also called the **body of the loop**. This is notated in the same way as a conditional statement is, with an indent. All of the iteration features like the while and for loop (for loop coming up later) use the same style of indentation as used previously, with the if/else/elif conditional statements by Java to define the code blocks.

The `<expression>` term typically is based on one or more variables that are initialized outside of the loop and then modified within the body of the loop. The `<expression>` represents the evaluated condition. Java is looking for a Boolean True or False value from this condition.

2a. How Does a while Loop Work?

During execution, or runtime, when a while loop is reached first, the program will evaluate the `<expression>`. This includes determining what conditions are in the expression. If the result is `true`, the body of the loop will execute. Once all of the statements in the body of the loop are executed, the `<expression>` is checked again. If it is still `true`, the body of the loop executes again. This is where the term “loop” comes from, because the last statement in the body of the loop, loops back around to the top. We call each time we execute the body of the loop an iteration. This process keeps running until the `<expression>` becomes `false`. When that occurs, the program moves on to the first statement that’s outside of the body of the loop.

Let’s look at a simple example to demonstrate how this works. This program keeps prompting the user to enter a number as long as the entry is greater than 0.

When the user enters a 0 (or a negative number), the program ends as seen below:

```
import java.util.Scanner;

class WhileLoop {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    System.out.println("This loop will run as long as the input is > 0.");
```

```
        // Initialize loop variable so the loop will run (> 0)
        int entry = 1;
        while(entry > 0) {
          System.out.print("Enter a number (0 or less to quit): ");
          entry = input.nextInt();
        }
        System.out.println("=== End of Loop ===\n");
      }
    }
```

Here is the output from a sample run of the program:

```
This loop will run as long as the input is > 0.
Enter a number (0 or less to quit): 3
Enter a number (0 or less to quit): 1
Enter a number (0 or less to quit): 0

=== End of Loop ===
```

The following table describes the iterations in further detail:

Iteration	Description
First Iteration	Because the variable `entry` is initialized to 1, the loop's condition (entry > 0) evaluates to `true`, so the loop begins the first iteration. During this iteration, the user is prompted to enter a number, and 3 is entered. The flow of the program returns to the top of the loop.
Second Iteration	In the sample run, the value of `entry` is 3, which is greater than 0, so the loop's condition evaluates to `true`. Once again, the code in the body of the loop prompts the user to enter a number, and the user enters 1, which now becomes the value of `entry`. The program returns to the top of the loop.
Third Iteration	Since the value of `entry` is 1, which is greater than 0, the loop is executed again. The code in the body of the loop prompts the user for the input of a number, and the user enters 0. This value is assigned to `entry`. After the third iteration, the value of `entry` is 0, so the loop's condition evaluates to `false`. The loop does not run again. Program flow resumes after the end of the body of the loop (the closing curly bracket). The "End of Loop" message is displayed.

Since the expression in the while loop is tested first, it’s possible that it could have been false to begin with, which means the body of the loop would never have run at all. To avoid this problem, the variable entry was initialized to a value greater than 0 so the condition would be true.

KEY CONCEPT

The body of the loop should change the value of one or more variables so that eventually the condition becomes false and the loop terminates. The variable that changes each time the loop executes and controls when the loop finishes is referred to as the **iteration variable**. If there is no iteration variable, the loop will repeat forever, and results in an **infinite loop**. An infinite loop is a loop in which the terminating condition is never satisfied or for which there is no terminating condition.

In the previous example, entry acted as the iteration variable. Each iteration, its value was changed. This was based on user input. This ensured that at some point, the condition would be evaluated as false, thus ending the loop.

In the example below, we're using the same code, but the variable entry is first set to 0. Since 0 > 0 returns false, the body of the loop never executes:

```
import java.util.Scanner;

class WhileLoop {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    System.out.println("This loop will run as long as the input is > 0.");
    // Initialize loop variable so the loop will run (> 0)
    int entry = 0;
    while(entry > 0) {
      System.out.print("Enter a number (0 or less to quit): ");
      entry = input.nextInt();
    }
    System.out.println("=== End of Loop ===\n");
  }
}
```

The results should look like this:

```
This loop will run as long as the input is > 0.

=== End of Loop ===
```

THINK ABOUT IT

During runtime, the while loop condition was checked, found to be false, and so program flow passed out of the loop to the next line of code, which happened to be the final `println() method`.

TERMS TO KNOW

while Loop
The while loop is one of the most commonly used loops. It keeps going as long as some condition is true.

Body of the Loop
The body of the loop represents the indented block of code that should be executed repeatedly within the loop during each loop iteration.

Iteration Variable
The variable that changes each time the loop executes and controls when the loop finishes.

Infinite Loop
An infinite loop is a loop in which the terminating condition is never satisfied or for which there is no terminating condition.

---

3. Basics of a for Loop

In Java, definite iteration loops are generally implemented using **for loops**. There are a couple types of for loops. This tutorial is focused on the basic for loop. "Enhanced" for loops will be addressed in a future tutorial.

The pattern for a for loop is:

EXAMPLE

```
for(<initialization>; <condition>; <increment/decrement>){
 <statement(s)>
}
```

In the example above, the `<initialization>`, `<condition>`, `<increment/decrement>` and `<statement(s)>` are terms with angle brackets that are just for information purposes. These are not keywords or actual code. These explain what goes into each of these parts of a for loop.

The `<initialization>` may declare a numeric variable (`int` or `long`) that is used as the loop counter and sets its initial value. If the variable has been declared before the loop, the `<initialization>` just sets the initial value. If the variable is declared and initialized here, the variable is only accessible in the body of the loop. If the variable is declared before the loop, its value is accessible outside of the loop (as well as inside the body of the loop). It is important to remember that this initialization happens only once at the start of the loop.

The `<condition>` is a boolean expression. It evaluates true or false based on the result of comparison. The loop executes as long as this expression evaluates as true. The loop stops when this expression becomes false.

HINT

There is a semicolon after the `<initialization>` and the `<condition>`.

The `<increment/decrement>` is an expression that defines how the value of the loop variable is modified at the end of each iteration. The most common expression used is the loop variable followed by the increment operator (++), so the loop variable's value is incremented by one after each iteration. It is also possible to change the variable by other values or decrement the variable, but incrementing by one is the most common.

HINT

The `<statement(s)>` are the statements that will execute each time the loop runs. Consider the following sample program:

```
class ForLoop {
  public static void main(String[] args) {
    System.out.println("This for loop will count to 5: ");
    /* The loop variable, count, starts with a value of 1.
       The loop keeps running as long as count <= 5.
       After each iteration - before returning to the top of the loop
       count is incremented by 1.
    */
    for(int count = 1; count <= 5; count++) {
      System.out.println(count);
    }
    System.out.println("The loop is done.");
  }
}
```

The results should look like this:

```
This for loop will count to 5:
1
2
3
4
5
The loop is done.
```

TERM TO KNOW

for Loop

In Java, the definite iteration loops are generally called for loops.

SUMMARY

In this lesson, you learned about the basics of **iteration**. You learned that **indefinite iteration** is a loop that runs as long as a condition is true and will exit from the loop when that condition becomes false. You also learned that **definite iteration** relies on defined start and stop values. You explored the **basic properties of a while loop** and that these loops are generally used for indefinite conditions where the number of times the loop should be run is not known. Finally, you learned about the **basics of a for loop** and that they are used for definite iterations where we know ahead of time how many times the loop should run or the maximum number of times the loop should run.

Source: This content and supplemental material has been adapted from Java, Java, Java: Object-Oriented Problem Solving. Source cs.trincoll.edu/~ram/jjj/jjj-os-20170625.pdf

It has also been adapted from “Python for Everybody” By Dr. Charles R. Severance. Source py4e.com/html3/

TERMS TO KNOW

Body of the Loop
The body of the loop represents the indented block of code that should be executed repeatedly within the loop during each loop iteration.

Definite Iteration
Definite iteration identifies the number of times, or the maximum number of times, the block of code runs. This should be explicitly defined when the loop actually starts. Typically, this is implemented using the for loop. The for loop has a specific start and endpoint.

Indefinite Iteration
A loop that does not specify, in advance, how many times it is to be run. It repeats as long as a condition is met. In Java, indefinite loops are typically created using the while loop.

Infinite Loop
An infinite loop is a loop in which the terminating condition is never satisfied or for which there is no terminating condition.

Iteration Variable
The variable that changes each time the loop executes and controls when the loop finishes.

Loop
In Java, the definite iteration loops are generally called for loops.

while Loop
The while loop is one of the most commonly used loops. It keeps going as long as some condition is true.