for Loops

In this lesson, you will be introduced to the for loop.

We'll cover the following

Introduction

You have to print **100** copies of a document. Imagine having to print each copy individually and pressing print **100** times. It's a lot more efficient and easier to just tell the printer to print the **100** copies and press print once.

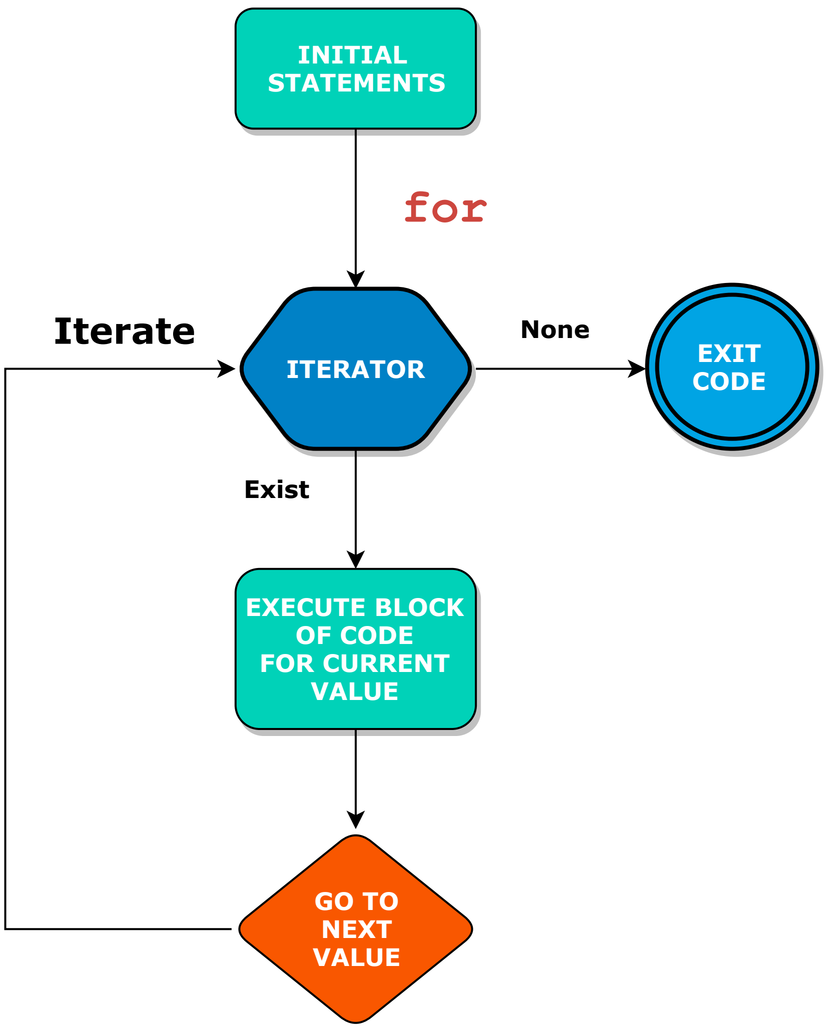
The printer is performing an operation (printing a document) repeatedly in a loop until it reaches its goal (**100** copies).

In computer programming, we sometimes come across scenarios where a block of code needs to be executed multiple times.

Dart provides `for` loops for this exact purpose.

Control Flow

Let's look at the general control flow of a `for` loop below.



The `for` loop allows us to specify a range of numbers over which we want our loop to run.

The **iterator** is responsible for keeping track of the iterations. Initially, its value is the start of the loop range and it changes with each iteration.

Syntax

There is more than one way to write a `for` loop in Dart. Let's try to generalize the syntax below.

```
for (iterator) {
  block of code
}
```

The syntax starts off with the `for` keyword which is followed by the iterator. But how should we define the iterator?

The iterator we will be using is a variable that is assigned a range of numbers. You need to specify three things, the initial value or where the range starts, and the final value, where it ends. You then also need to specify how the iterator will move through the range in every iteration of the `for` loop. For instance, it could be incremented by **1** or decremented by **1**.

All three things are specified within the parentheses and separated by a semicolon (`;`).

Things will become a lot clearer with an example.

`for` Loop in Action

Suppose we want to print something five times. We need to specify three things.

1. The initial value of the iterator: `var i = 0` (initial value is **0**)
2. The final value of the iterator: `i < 5` (up till **5**)
3. How the iterator moves through the specified range: `i++` (increment by **1**)

Let's take a look at the code below:

```
main() {
  for(var i = 0; i < 5; i++){
    print(i);
  }
}
```

The `print` statement is executed five times with `i` being sequentially assigned a new value between **0** and **4** in each iteration. The `for` expression runs for five iterations.

Iterating Through a Collection

We can use `for` loops to perform some operation on every item in a collection.

Let's print every element in a list of colors.

```
main() {
  var colorList = ['blue', 'yellow', 'green', 'red'];
  var colorList = ['blue', 'yellow', 'green', 'red'];
  for(var i = 0; i < colorList.length; i++){
    print(colorList[i]);

  }
}
```

To create our range for the iterator `i`, we use the `length` property. The iterator will stop iterating **1** less than the length of the list. This is because the last index of a list is one less than its length as indexes start from **0** and length starts from **1**.

This is why we have written `i < colorList.length` rather than `i <= colorList.length`.

The For-In Form

For sets and lists, you can use the **for-in** form of iteration.

The `for` loop using *for-in* has the following syntax:

```
for (iterator in collection){
   block of code
}
```

Let's use the same example as above and modify the code so that it uses the new form.

```
main() {
  var colorList = ['blue', 'yellow', 'green', 'red'];

  for(var i in colorList){
    print(i);
  }
}
```

The code works the same way as the one we looked at before. However, this time around our range is the items of a collection. Each item, in turn, is assigned to the iterator `i` and an operation is performed on it. In this case, the print method is called for each item.

called for each item.

Conditions with Loops

Up until now, we have been performing the simple print operation. But there's a lot more you can perform with loops.

Let's incorporate a conditional with a loop.

In our example, we have a list of integers `intList`. We only want to print the even integers from that list.

```
main() {
  var intList = [6,7,3,9,2,5,4];

  for(var i in intList){
    if(i % 2 == 0){
      print(i);
    }
  }
}
```

On **line 4**, we are accessing every item in the list `intList` one after the other. On **line 5**, we are inserting a condition that an item must be even. Only if the condition is `true`, will the code on **line 6** be executed.

In the next lesson, we have a challenge for you to solve for yourself.