while Loops

This lesson introduces the while loops in PHP. It uses coding examples to show their implementation and explains how they work.
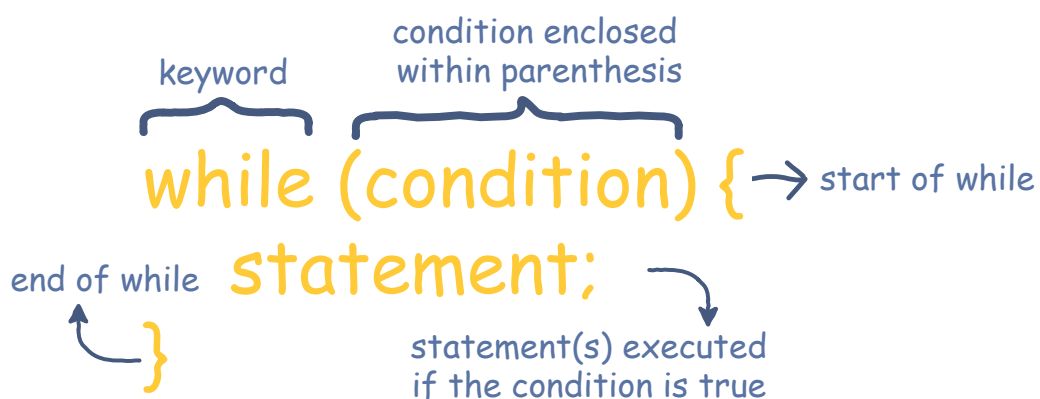
What Is the While Loop?

The `while` *loop* is really the only necessary repetition construct. The `for` loop, coming up, can be duplicated using a `while` loop, and with more control.

So basically, the while loop consists of a condition and while that condition is true, a set of statements is executed repeatedly. The syntax is as follows:

Again, the **curly braces** surrounding the *body* of the `while` *loop* indicate that *multiple* statements can be executed as part of this *loop*.
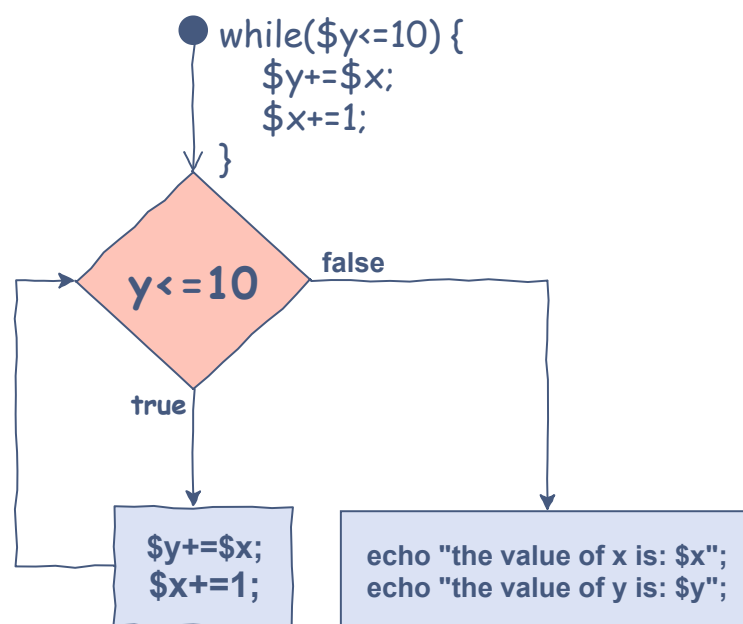
Example

Here's a look at the `while` loop code:

```php
<?php
$x = 4;
$y = 0;
while ($y <= 10)
{ // the while loop will run till y<=10 condition is being met
    $y += $x;
    $x += 1;
} //the loop will iterate 3 times
echo "the value of x is: $x\n";
echo "the value of y is: $y\n";
?>
```

Explanation

Below is an *illustration* of the code above to help you better understand the logic.

Flow Chart For While Loop Code

Infinite Loops

If the `while` *loop* code looked like this instead:

```php
while ( $y <= 10 ) { //since y is not being changed inside the while loop you will get stuck
    $x += 1;         // in an infinte loop as the condiiton will always be met
}
$y += $x;
```

There would be a problem.

There would be a problem.

According to what is written, even though the *second* line after the `while` was *incremented*, only the *first* line corresponds to the `while` *loop*. This is a huge problem because the variable involved in the condition `(x)` does **not** change, so it will always evaluate to *true*, making this an **infinite** loop. This could be alleviated by containing all statements intended to be a part of the loop body in `{ }`. You will learn more about Infinite loops later in the chapter.

In the next lesson, we will discuss a slight variation of the while loop – the do... while loop.