**Loops**

A loop is a set of code that can be processed over and over. Loops allow you to use the same code multiple times to process values. Loops need some way of knowing when to stop. This is done with a loop control variable. A loop control variable is some piece of information that will tell us when the loop should stop.

There are some key concepts that all loops share. Knowing these key processes will help make your loops run correctly. Three key processes for every loop:

1. **Identify the loop control variable**. This is the variable or piece of information that we use to control the loop. In essence this variable is checked so the loop knows whether to stop the loop or process the code again.
2. **Check the end condition of the loop control variable**. We check the value of the loop control variable against a condition to see if the loop is supposed to stop or continue to process.
3. **Update the loop control variable inside the body of the loop**. Ideally every time through the loop you should update the value of the loop control variable. If it never changes the loop will never end.

Pre-test versus Post-test loops

Loops can work one of two ways. They can check the end condition of the loop BEFORE they process the body of the loop. This is called a pre-test loop. This means the loop could be started but not do any processing if the end condition is met. Most loops use this model. Pre-test loops include the for loop and the while loop. These are the most common loops used in programming.

Other loops check the condition of the loop AFTER they process the body of the loop. This means they will ALWAYS run the loop code at least one time. These are called post-test loops. The do-while loop is the main example of a post-test loop.

For Loop

This is one of the most common loops used in programming. The PHP for loop looks and acts like most other languages. The for loop is best used when you know how many times you are going to run the loop. The for loop is a counter controlled loop. Usually this means we are going to count the number of times the loop runs and compare it to the number of times it should run.

The for loop builds all three of the key processes of a loop into its format and structure. The parentheses contain an area for the initialization, condition check and the update processes.

The for loops structure is described below.

for ( initialization; check condition; update )

{

*//Body of the loop*

*Code goes here*

}

A working example is shown below.

for($x = 1; $x < 10; $x++ )

{

Echo “<p>The Counter is: $x </p>”;

}

While Loop

While loops are used when we do not how many times the loop will be processed. The loop is controlled with a variable called a ‘sentinel’. While loops are used when you need to run the loop until there is no more data, or until the end of the file or the end of the database result. While loops are often used for input and ask the user if they have more data to enter, or need to continue shopping or wish to add another item, etc.

While loops require more preparation and planning. You have to prepare for a while loop by identifying and setting the loop control variable. The while loop will check the condition. It is critical to remember to update the loop control variable inside the body of the loop.

The general format of a while loop is described below.

while (condition)

{

//Body of the loop

Code goes here

}

A working example of a while loop is shown below.

$numItems = 1; //Identify and define the loop control variable

while ($numItems <= 10) //Run the loop until the row count is > 10

{

echo “<li>Special Item #$numItems</li>”; //Display a list item to the result object

$numItems++; //Update the loop control variable in the body of the loop

Foreach Loop

This loop is designed specifically to work with arrays. In most cases the program will process some or all of the values in an array. You can do this with any of the above loop types. The foreach is offered as a shortcut way to process each of the values of an array.

Like the for loop all the key pieces are contained with the parentheses of the foreach loop. The general format of the foreach loop is shown below.

foreach ($array as $value)

{

*//Body of the loop*

*Code goes here!*

}

A working example of the foreach loop is given below.

foreach($listNames as $outNames)

{

echo “<p>$outNames</p>”;

}

**References:**

W3Schools Loops: <http://www.w3schools.com/php/php_looping.asp>