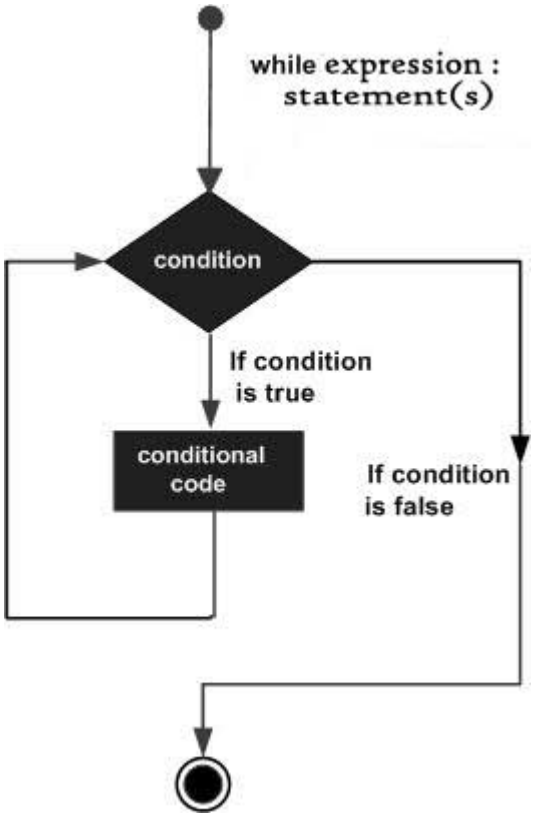
PHP - While Loop

The easiest way to create a loop in a PHP script is with the **while** construct. The syntax of **while** loop in PHP is similar to that in C language. The loop body block will be repeatedly executed as long as the Boolean expression in the while statement is true.

The following flowchart helps in understanding how the **while** loop in PHP works –



The value of the expression is checked each time at the beginning of the loop. If the **while** expression evaluates to false from the very beginning, the loop won't even be run once. Even if the expression becomes false during the execution of the block, the execution will not stop until the end of the iteration.

The syntax of **while** loop can be expressed as follows –

```
while (expr){
   statements
}
```

Example

The following code shows a simple example of how the **while** loop works in PHP. Th
variable **$x** is initialized to 1 before the loop begins. The loop body is asked to execute as

long as it is less than or equal to 10. The **echo** statement in the loop body prints the current iteration number, and increments the value of **x**, so that the condition will turn false eventually.

```php
<?php
   $x = 1;

   while ($x<=10) {
      echo "Iteration No. $x \n";
      $x++;
   }
?>
```

Open Compiler

It will produce the following **output** –

```
Iteration No. 1
Iteration No. 2
Iteration No. 3
Iteration No. 4
Iteration No. 5
Iteration No. 6
Iteration No. 7
Iteration No. 8
Iteration No. 9
Iteration No. 10
```

Note that the test condition is checked at the beginning of each iteration. Even if the condition turns false inside the loop, the execution will not stop until the end of the iteration.

Example

In the following example, "x" is incremented by 3 in each iteration. On the third iteration, "x" becomes 9. Since the test condition is still true, the next round takes place in which "x" becomes 12. As the condition turns false, the loop stops.

Open Compiler

```php
<?php
   $x = 0;
   while ($x<=10){
      $x+=3;
      echo "Iteration No. $x \n";
   }
?>
```

It will produce the following **output** –

```
Iteration No. 3
Iteration No. 6
Iteration No. 9
Iteration No. 12
```


Example

It is not always necessary to have the looping variable incrementing. If the initial value of the loop variable is greater than the value at which the loop is supposed to end, then it must be decremented.

</> Open Compiler

```php
<?php
   $x = 5;
   while ($x>0) {
      echo "Iteration No. $x \n";
      $x--;
   }
?>
```

It will produce the following **output** –

```
Iteration No. 5
Iteration No. 4
Iteration No. 3
```

```
Iteration No. 2
Iteration No. 1
```

Iterating an Array with "while"

An indexed array in PHP is a collection of elements, each of which is identified by an incrementing index starting from 0.

You can traverse an array by constituting a **while** loop by repeatedly accessing the element at the xth index till "x" reaches the length of the array. Here, "x" is a counter variable, incremented with each iteration. We also need a count() function that returns the size of the array.

Example

Take a look at the following example –

</> Open Compiler

```
<?php
   $numbers = array(10, 20, 30, 40, 50);
   $size = count($numbers);
   $x=0;

   while ($x<$size) {
      echo "Number at index $x is $numbers[$x] \n";
      $x++;
   }
?>
```

It will produce the following **output** –

```
Number at index 0 is 10
Number at index 1 is 20
Number at index 2 is 30
Number at index 3 is 40
Number at index 4 is 50
```

Nested "while" Loops

You may include a **while** loop inside another **while** loop. Both the outer and inner while loops are controlled by two separate variables, incremented after each iteration.

Example

</> Open Compiler

```
<?php
   $i=1;
   $j=1;

   while ($i<=3){
      while ($j<=3){
         echo "i= $i j= $j \n";
         $j++;
      }
      $j=1;
      $i++;
   }
?>
```

It will produce the following **output** –

```
i= 1 j= 1
i= 1 j= 2
i= 1 j= 3
i= 2 j= 1
i= 2 j= 2
i= 2 j= 3
i= 3 j= 1
i= 3 j= 2
i= 3 j= 3
```

Note that "j" which is the counter variable for the inner **while** loop is re-initialized to 1 after it takes all the values so that for the next value of "i", "j" again starts from 1.

Traversing the Characters in a String

In PHP, a string can be considered as an indexed collection of characters. Hence, a while loop with a counter variable going from "0" to the length of string can be used to fetch one character at a time.

Example

The following example counts number of vowels in a given string. We use **strlen()** to obtain the length and **str_contains()** to check if the character is one of the vowels.

</> Open Compiler

```
<?php
   $line = "PHP is a popular general-purpose scripting language that is especially
   $vowels="aeiou";
   $size = strlen($line);
   $i=0;
   $count=0;

   while ($i<$size){
      if (str_contains($vowels, $line[$i])) {
         $count++;
      }
      $i++;
   }
   echo "Number of vowels = $count";
?>
```

It will produce the following **output** –

```
Number of vowels = 32
```

Using the "endwhile" Statement

PHP also lets you use an alternative syntax for the **while** loop. Instead of clubbing more than one statement in curly brackets, the loop body is marked with a ":" (colon) symbol after the condition and the **endwhile** statement at the end.

Example

</> Open Compiler

```
<?php
   $x = 1;
```

```
   while ($x<=10):
      echo "Iteration No. $x \n";
      $x++;
   endwhile;
?>
```

It will produce the following **output** −

```
Iteration No. 1
Iteration No. 2
Iteration No. 3
Iteration No. 4
Iteration No. 5
Iteration No. 6
Iteration No. 7
Iteration No. 8
Iteration No. 9
Iteration No. 10
```

Note that the **endwhile** statement ends with a semicolon.