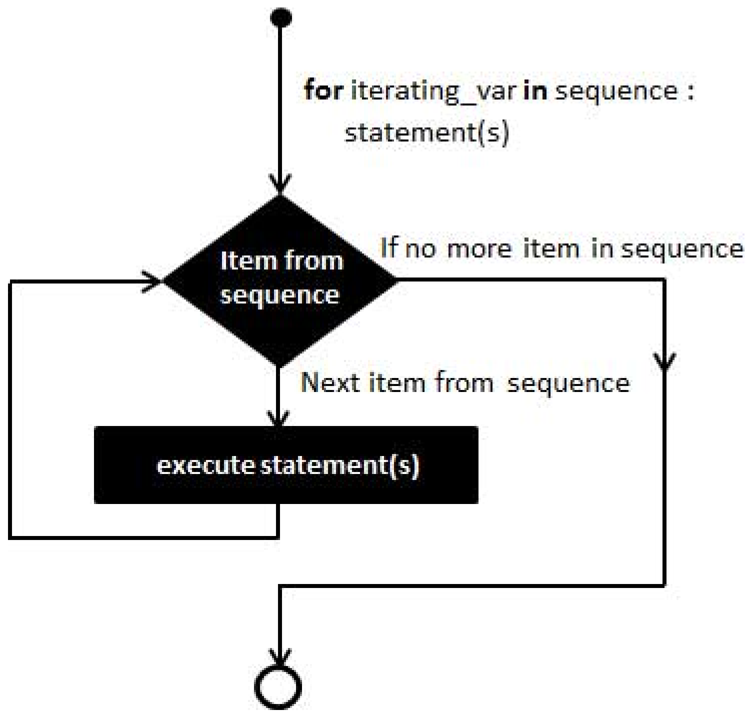
PHP - For Loop

A program by default follows a sequential execution of statements. If the program flow is directed towards any of earlier statements in the program, it constitutes a loop. The **for** statement in PHP is a convenient tool to constitute a loop in a PHP script. In this chapter, we will discuss PHP’s for statement.

Flowchart of "for" Loop

The following flowchart explains how a **for** loop works −



The **for** statement is used when you know how many times you want to execute a statement or a block of statements.

Syntax of "for" Loop

The syntax of **for** statement in PHP is similar to the for statement in C language.

```
for (expr1; expr2; expr3){
   code to be executed;
}
```

The **for** keyword is followed by a parenthesis containing three expressions separated by a semicolon. Each of them may be empty or may contain multiple expressions separated by commas. The parenthesis is followed by one or more statements put inside curly brackets. It forms the body of the loop.

The first expression in the parenthesis is executed only at the start of the loop. It generally acts as the **initializer** used to set the start value for the counter of the number of loop iterations.

In the beginning of each iteration, **expr2** is evaluated. If it evaluates to true, the loop continues and the statements in the body block are executed. If it evaluates to false, the execution of the loop ends. Generally, the **expr2** specifies the final value of the counter.

The **expr3** is executed at the end of each iteration. In most cases, this expression increments the counter variable.

Example

The most general example of a **for** loop is as follows −

Open Compiler

```php
<?php
   for ($i=1; $i<=10; $i++){
      echo "Iteration No: $i \n";
   }
?>
```

Here is its **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
```

An infinite "for" loop

Note that all the three expressions in the parenthesis are optional. A **for** statement with only two semicolons constitutes an infinite loop.

```
for (; ;) {
   Loop body
}
```

To stop the infinite iteration, you need to use a **break** statement inside the body of the loop.

A decrementing "for" loop

You can also form a decrementing **for** loop. To have a **for** loop that goes from 10 to 1, initialize the looping variable with 10, the expression in the middle that is evaluated at the beginning of each iteration checks whether it is greater than 1. The last expression to be executed at the end of each iteration should decrement it by 1.

</> Open Compiler

```php
<?php
   for ($i=10; $i>=1; $i--){
      echo "Iteration No: $i \n";
   }
?>
```

It will produce the following **output** −

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

Using the "for…endfor" construct

You can also use the ":" (colon) symbol to start the looping block and put **endfor** statement at the end of the block.

Open Compiler

```
<?php
   for ($i=1; $i<=10; $i++):
      echo "Iteration No: $i \n";
   endfor;
?>
```

Iterating an indexed array using "for" loop

Each element in the array is identified by an incrementing index starting with "0". If an array of 5 elements is present, its lower bound is 0 and is upper bound is 4 (size of array -1).

To obtain the number of elements in an array, there is a count() function. Hence, we can iterate over an indexed array by using the following **for** statement –

Open Compiler

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

   for ($i=0; $i<count($numbers); $i++){
      echo "numbers[$i] = $numbers[$i] \n";
   }
?>
```

It will produce the following **output** –

```
numbers[0] = 10
numbers[1] = 20
numbers[2] = 30
numbers[3] = 40
numbers[4] = 50
```

Iterating an Associative Array Using "for" Loop

An associative array in PHP is a collection of key-value pairs. An arrow symbol (=>) is used to show the association between the key and its value. We use the **array_keys()** function to obtain array of keys.

The following **for** loop prints the capital of each state from an associative array **$capitals** defined in the code −

Open Compiler

```php
<?php
   $capitals = array(
      "Maharashtra"=>"Mumbai",
      "Telangana"=>"Hyderabad",
      "UP"=>"Lucknow",
      "Tamilnadu"=>"Chennai"
   );
   $keys=array_keys($capitals);

   for ($i=0; $i<count($keys); $i++){
      $cap = $keys[$i];
      echo "Capital of $cap is $capitals[$cap] \n";
   }
?>
```

Here is its **output** −

```
Capital of Maharashtra is Mumbai
Capital of Telangana is Hyderabad
Capital of UP is Lucknow
Capital of Tamilnadu is Chennai
```

Using Nested "for" Loops in PHP

If another **for** loop is used inside the body of an existing loop, the two loops are said to have been nested.

For each value of counter variable of the outer loop, all the iterations of inner loop are completed.

Open Compiler

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

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 a string is a form of an array. The **strlen()** function gives the number of characters in a string.

Example

The following PHP script uses two nested loops to print incrementing number of characters from a string in each line.

Open Compiler

```php
<?php
   $str = "TutorialsPoint";
   for ($i=0; $i<strlen($str); $i++){
      for ($j=0; $j<=$i; $j++){
         echo "$str[$j]";
      }
      echo "\n";
   }
?>
```

It will produce the following **output** −

```
T
Tu
Tut
Tuto
Tutor
Tutori
Tutoria
Tutorial
Tutorials
TutorialsP
TutorialsPo
TutorialsPoi
TutorialsPoin
TutorialsPoint
```