



## repetition structures

Repetition control structures are used to execute a code block over and over again until the stopping condition is fulfill. It means you don't have to copy and paste your code many times in the file just use a right loop statement.

```
<?php
    echo "1";
    echo "2";
    echo "3";
    echo "4";
    echo "5";
    echo "6";
    echo "7";
    echo "8";
    echo "9";
    echo "10";
?>
```

As you can see the above statements are written without repetition control structure. The program tries to display a set of numbers from 1 to 10. The programmer needs to copy, paste and modify the echo command in order to print the ten numbers. This program will be a whole lot simpler if the programmer choose to use repetition structure, as below.

```
<?php
    $i=1;
    do {
        echo $i;
        $i++;
    }while ($i<=10);
?>
```

There are many ways of doing repetition. We will discuss three of the most popular repetition structure in PHP.


1. for
2. while and do ... while
3. foreach

## for

General format of *for* structure.

```
for (initialize a counter; conditional statement; increment the  
counter) {  
    codes;  
}
```

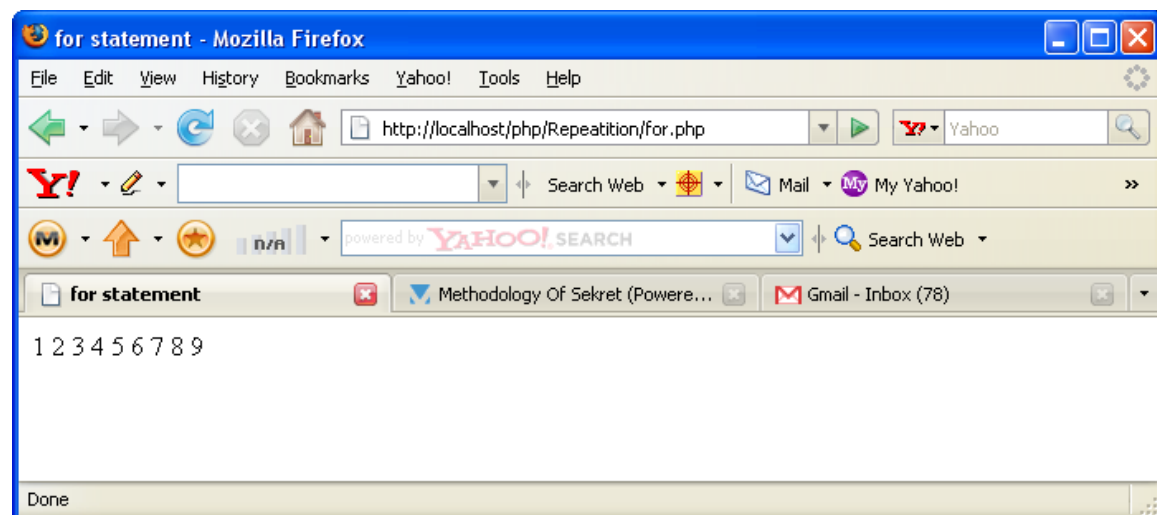
Example using the *for* loop:



```
7  
8 <body>  
9  
10 <?php  
11 for ($i=1;$i<=9;$i++){  
12 echo " $i ";  
13 }  
14 ?>  
15  
16 </body>  
17 </html>  
18 |
```

Well as you can see both example above will given the same output:

Output:



\*As you can see the solution with loop is much better. It is shorter, easier to understand. Besides this in most cases you don't know during the coding how many times the code block needs to be executed.

While Loop

## while

General format of *while* structure.

```
while (condition)
code to be executed;
```

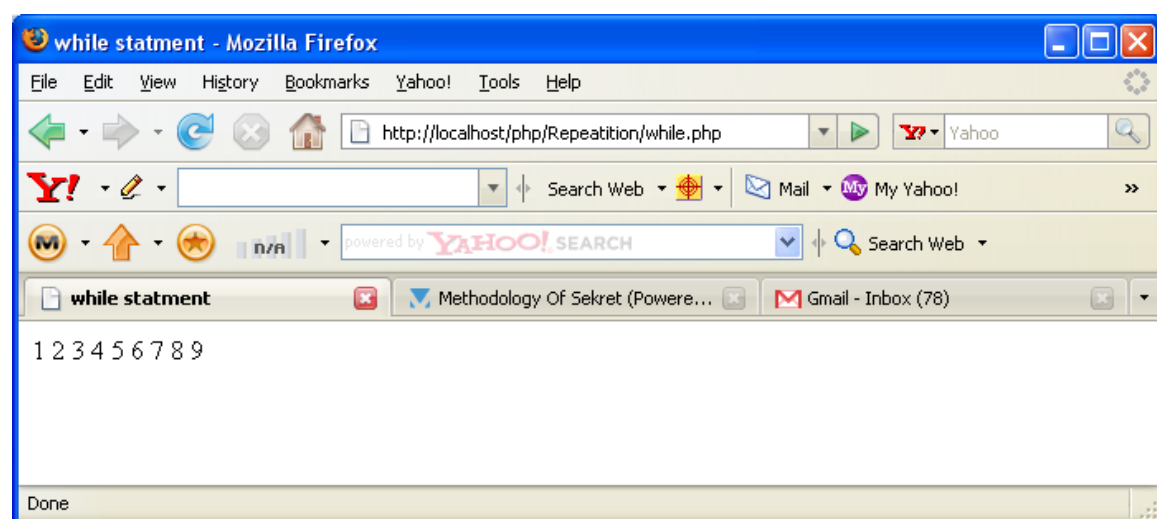
Example using the while loop

```
7
8 |
9 <body>
10 <?php
11     $i=1;
12     while ($i<=9){
13         echo " $i ";
14         $i++;
15     }
16 ?>
17 </body>
18 </html>
19
```

The code block will be executed until the condition is true. It means that it can happen that it will never execute. To implement our first example with while loop looks like this:

The output will be exactly the same as before. However if you initialize \$i variable with 10 (\$i=10) then nothing will be displayed. If you forgot to increment the variable it will result an endless loop as the condition will be never changed and it is always true.

Output:



Another while example:

Imagine that you are running an art supply store. You would like to print out the price chart for number of brushes and total cost. You sell brushes at a flat rate, but would like to display how much different quantities would cost. This will save your customers from having to do the mental math themselves.

You know that a while loop would be perfect for this repetitive and boring task. Here is how to go about doing it.

```

8  <body>
9  <?php
10     $brush_price = 5;
11     $counter = 10;
12
13     echo "<table border='1' align='center'>";
14     echo "<tr><th>Quantity</th>";
15     echo "<th>Price</th></tr>";
16     while ( $counter <= 100 ) {
17         echo "<tr><td>";
18         echo $counter;
19         echo "</td><td>";
20         echo $brush_price * $counter;
21         echo "</td></tr>";
22         $counter = $counter + 10;
23     }
24     echo "</table>";
25 ?>
26 </body>
27 </html>

```

Output:

Quantity	Price
10	50
20	100
30	150
40	200
50	250
60	300
70	350
80	400
90	450

Pretty cool, huh? The loop created a new table row and its respective entries for each quantity, until our counter variable grew past the size of 100. When it grew past 100 our conditional statement failed and the loop stopped being used. Let's review what is going on.

1. We first made a \$brush\_price and \$counter variable and set them equal to our desired values.
2. The table was set up with the beginning table tag and the table headers.
3. The while loop *conditional statement* was checked, and \$counter (10) was indeed smaller or equal to 100.
4. The code inside the while loop was executed, creating a new table row for the price of 10 brushes.
5. We then added 10 to \$counter to bring the value to 20.
6. The loop started over again at step 3, until \$counter grew larger than 100.
7. After the loop had completed, we ended the table.

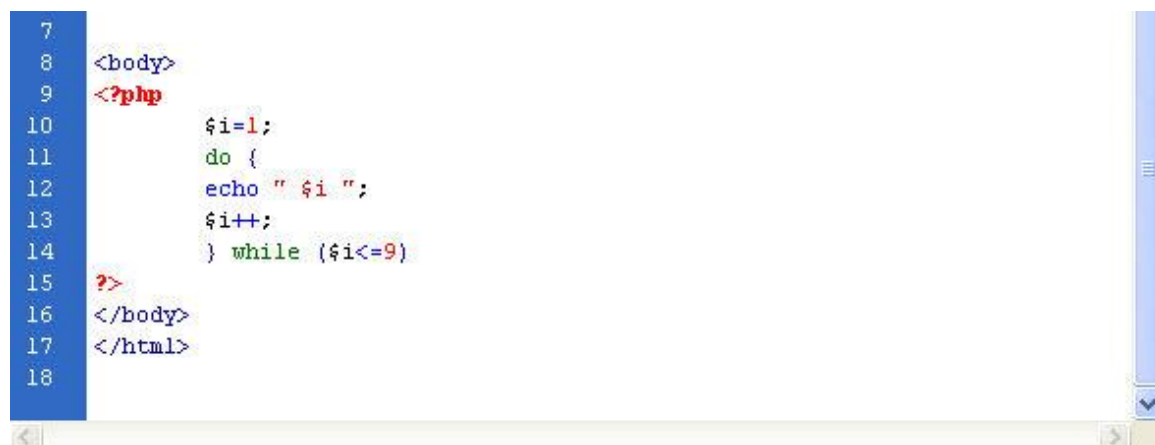
### do...while

This loop variant is very similar to the while loop. However there is one important difference. With do while loop the code block will be executed at least once. This is because in case of a do while loop PHP checks the condition only after the first iteration. It is clearly visible from the syntax:

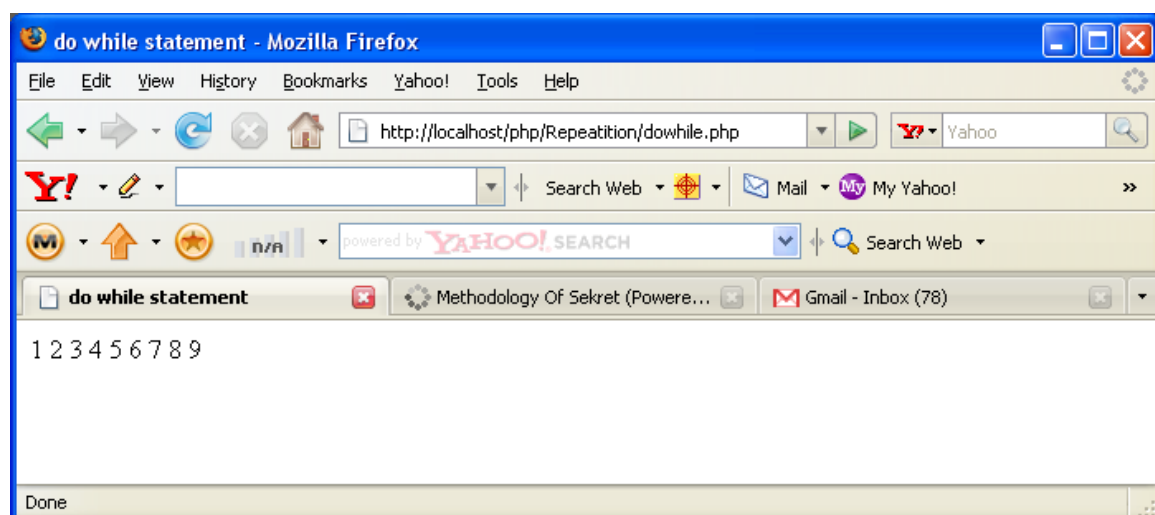
General format of *for* structure.

```
do {
    code block
} while (condition)
```

Besides this there is no more difference to the basic while loop. Here is the example how to use it:



Output:



The last loop structure in PHP is the foreach. This is a special loop as you can use it only for arrays. The goal of the foreach loop to iterate over each element of an array. If you try to use it with a normal variable you will get an error.

## **foreach**

General format of foreach structure.

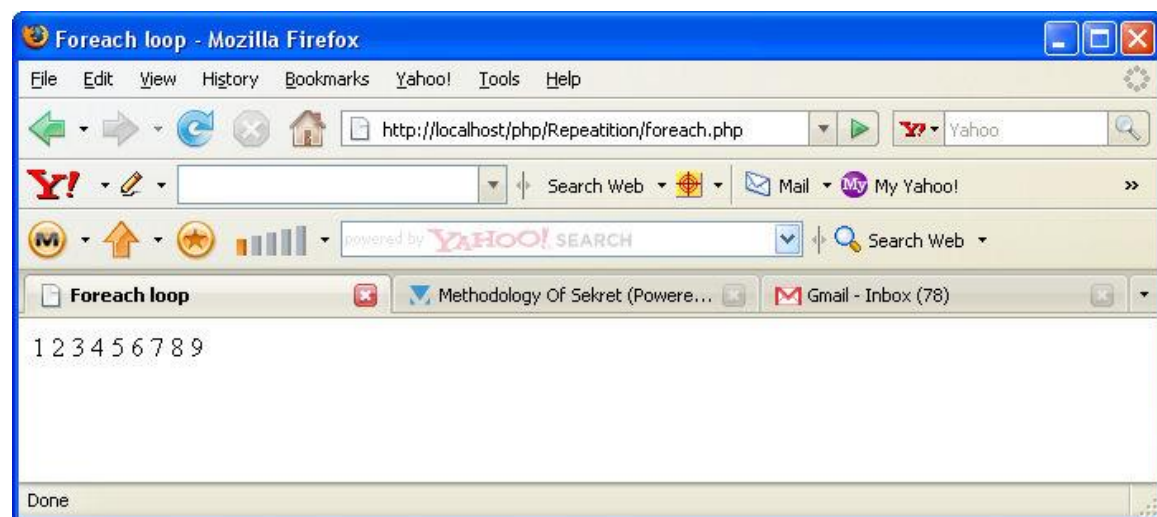
```
foreach (array as $value)
code executed;
```

It means that in each iteration the actual array value will be copied to the \$value variable and you can use it in the code executed.

Example using the for each loop:

```
7
8 <body>
9 <?php
10     $myList = array(1,2,3,4,5,6,7,8,9);
11
12     foreach ($myList as $value) {
13         echo " $value ";
14     }
15 ?>
16 </body>
17 </html>
18
```

Output:

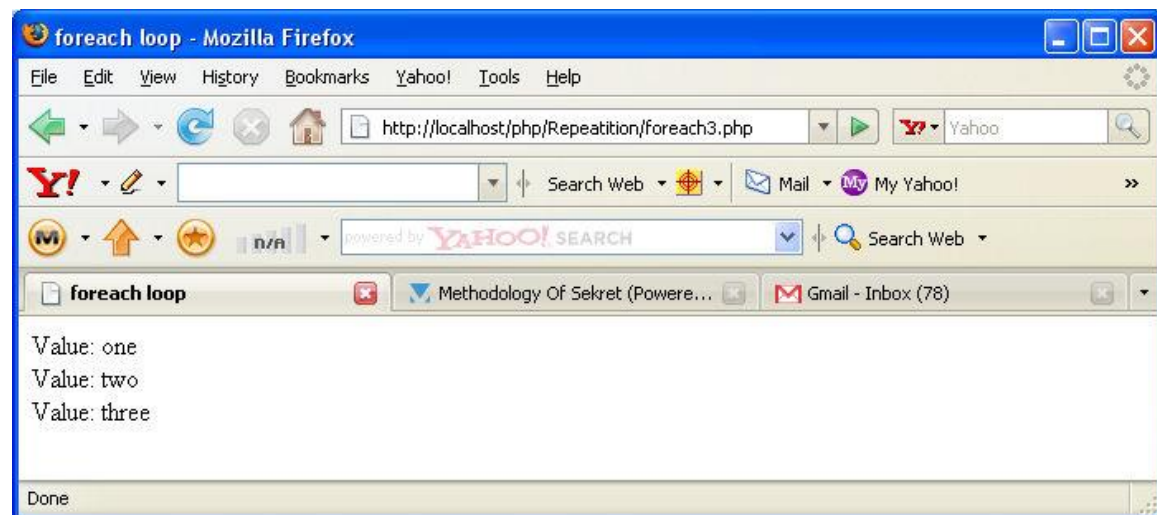


Another simple example for the *foreach* loop:

```
7
8 <body>
9 <?php
10     $arr=array("one", "two", "three");
11
12     foreach ($arr as $value)
13     {
14         echo "Value: " . $value . "<br />";
15     }
16 ?>
17 </body>
18 </html>
19
```

Well as we can see here we declared \$arr as our test variable. And we declared the value of the variable in the array. And it is "one", "two", and "three". So from the code also we can see that the array that contain in the variable \$arr is being copied into the variable \$value using the foreach loop ( \$arr as \$value ).And the last thing we see here it echo the variable \$echo and the output will be :

Output:





There is an alternative syntax of the foreach loop to handle associative arrays. You can use this if you want to know not only the actual element value but the key as well. The syntax is:

```
foreach (array as $key => $value)
code executed;
```

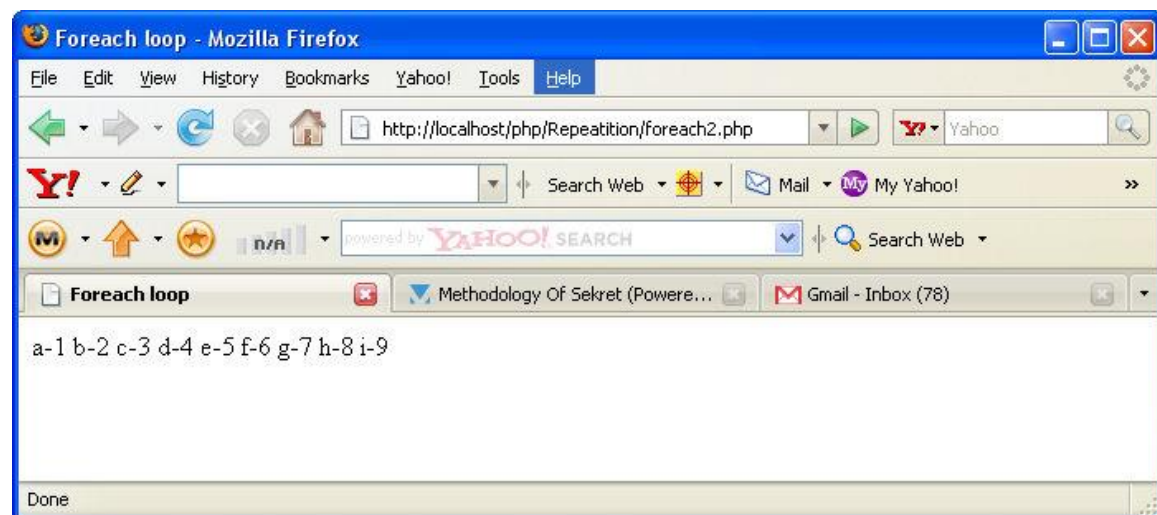
In this case you can use both information in your code execute like this:

```

7
8 <body>
9 <?php
10     $myList = array('a'=>1,'b'=>2,'c'=>3,'d'=>4,'e'=>5,
11     'f'=>6,'g'=>7,'h'=>8,'i'=>9);
12
13     foreach ($myList as $key => $value) {
14         echo " $key-$value ";
15     }
16
17 ?>
18 </body>
19 </html>

```

Output:



**Repetition Applied Example 1 : Generating Date Input using ComboBox**

In this example, we would like to show how to generate option list for date input, consist of day, month and year.

```
<html>
<head>
<title>Input Date</title>
</head>
<body>
<form name="forminsert" method=get action="isodate.php">
Date Input <br>
Day
<select name="day">
<?php
    for ($i=1;$i<=31;$i++){
        echo "<option value='$i'> $i </option>\n";
    }//end for
?>
</select>

Month
<select name="month">
<?php
    for ($i=1;$i<=12;$i++){
        echo "<option value='$i'> $i </option>\n";
    }//end for
?>
</select>

Year
<select name="year">
<?php
    for ($i=1900;$i<=date('Y');$i++){
        echo "<option value='$i'> $i </option>\n";
    }//end for
?>
</select>

<br>
<input type="submit" value="Combine">

</form>
<br>

</body>
</html>
```

From the items selected, generate ISO format for the date (yyyy-mm-dd).

```
<html>
<head>
<title>Date in ISO format</title>
</head>
```

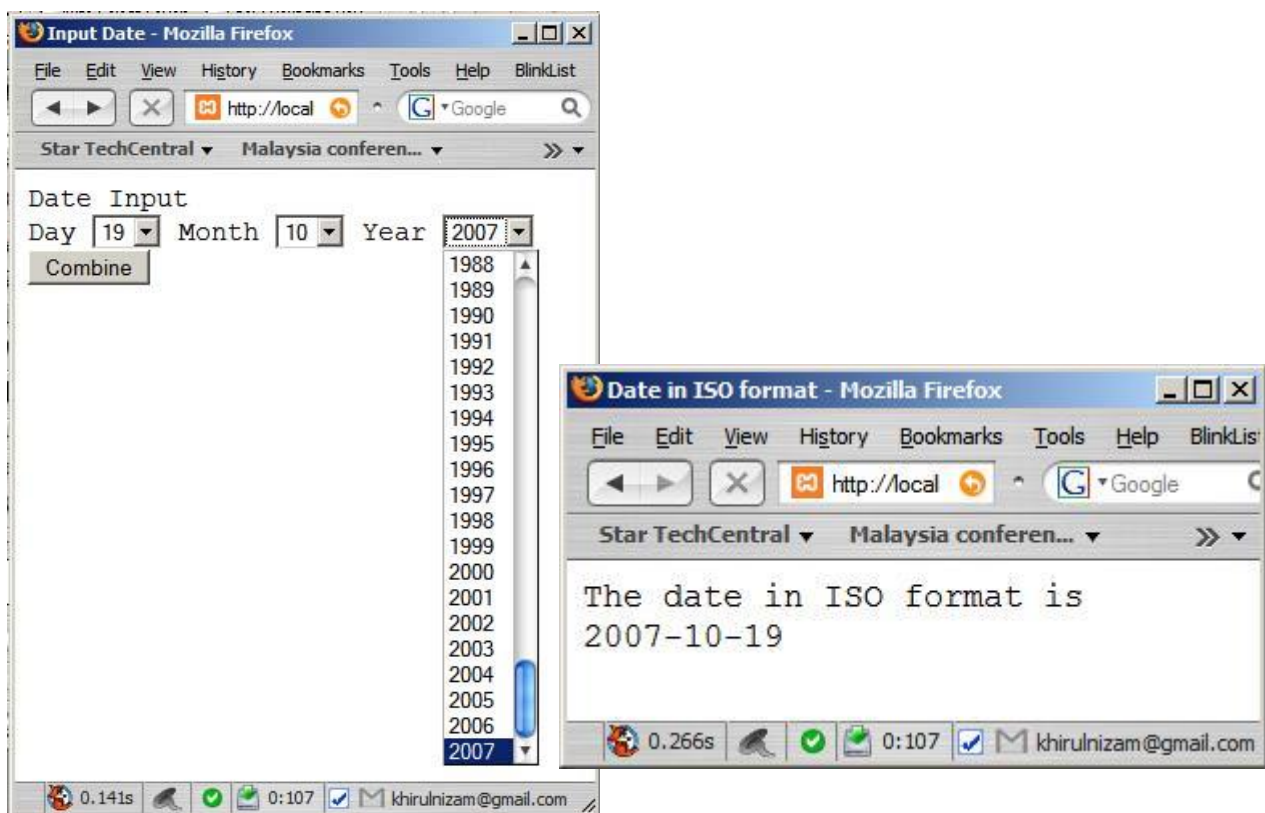
```

<body>
The date in ISO format is <br>
<?php
    $day=$_GET['day'];
    $month=$_GET['month'];
    $year=$_GET['year'];
    $isodate = sprintf("%4d-%2d-%2d", $year, $month, $day);
    echo $isodate;
?>

</body>
</html>

```

\*For further information on sprintf function, go to <http://php.net/sprintf>.



## Exercises for Repetition in PHP

### Question 1

Let say you have this array in your program;

```
$name= array ("Kerul", "Amir", "Luqman", "Muna", "Acik");
```

Create the program in PHP to print all the values in the array by using repetition.

### Question 2

What is the output of the following code execution?

```
<?php
$no=5;
for ($i=0; $i<20; $i++){
    echo (" $no <br>");
    $no--;
}
?>
```

### Question 3

Generate a combo box (selection list) from the array of states in Malaysia. Use for statement to generate the combo box as shown in the picture.

```
$state = array ("Johor","Melaka","N Sembilan", "Selangor", "Perak", "Kedah", "Pulau Pinang",  
"Perlis", "Kelantan", "Terengganu", "Pahang", "Sabah", "Sarawak", "Kuala Lumpur",  
"Labuan", "Putrajaya");
```

This is the sample :

