**Difference Between exit and break in PHP**

This article is created to differentiate [exit()](https://codescracker.com/php/php-exit-function.htm) and [break](https://codescracker.com/php/php-break-continue-keyword.htm) in PHP.

The PHP **exit()** terminates the execution of the current PHP script, whereas the **break** terminates the execution of the current [loop](https://codescracker.com/php/php-loops.htm) or [switch](https://codescracker.com/php/php-switch.htm) structure. For example, let me use **break** first:

<?php

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

{

if($i==4)

break;

echo "The value of \$i is $i";

echo "<BR>";

}

echo "<HR>";

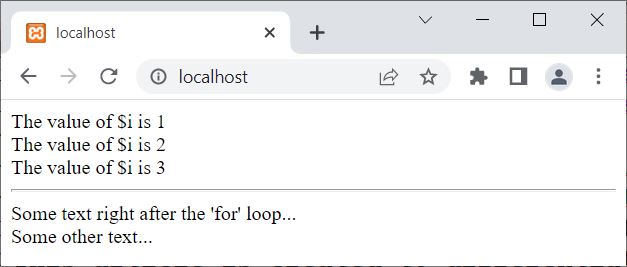
echo "Some text right after the 'for' loop...";

echo "<BR>";

echo "Some other text...";

?>

The output of this PHP example is:



And let me create the same example, using **exit()**:

<?php

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

{

if($i==4)

exit();

echo "The value of \$i is $i";

echo "<BR>";

}

echo "<HR>";

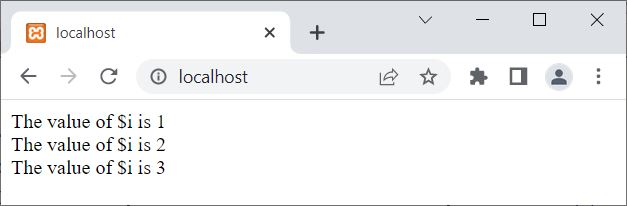
echo "Some text right after the 'for' loop...";

echo "<BR>";

echo "Some other text...";

?>

Now the output is:



That is, the **break** is used when we need to exit from a loop, whereas the **exit** is used when we need to exit from the whole PHP script/program.

# What is the difference between echo and print\_r in PHP?

**echo:** echo is not a function rather it is described as a language construct. It accepts an list of argument (multiple arguments can be passed) and returns no value or returns void. It cannot be used as a variable function in PHP. It is used to display the output of parameters that is passed to it. It display the outputs one or more strings separated by commas.

**Example:**

|  |
| --- |
| <?php    // PHP program to illustrate echo    // Declare variable and initialize it.  $x = "Geeks ";  $y = "Computer science portal";    // Display the value of $x  echo $x, $y;  ?> |

**Output:**

Geeks Computer science portal

**print\_r():** print\_r() is a regular function. It outputs the detailed information about the parameter in a format with its type (of an array or an object), which can be easily understandable by humans. In this function the output get stored on the internal buffer when the return parameter is passed. If pass the return parameter to TRUE, print\_r() would return the complete information rather than just print it. During walk-through this function helps in identifying any of the glitches while executing the program. It is more similar to the var\_dump() function.  
**Example:**

|  |
| --- |
| <?php    // PHP program to illustrate echo    // Declare an array  $arr = **array**('0' => "GeeksforGeeks",               '1' => "Computer",               '2' => "Science",               '3' => "Portal");    // Display the value of $x  print\_r($arr);  ?> |

**Output:**

Array

(

[0] => GeeksforGeeks

[1] => Computer

[2] => Science

[3] => Portal

)

**Example:**

|  |
| --- |
| <?php    $a = "GeeksforGeeks";  $b = **array**('0' => "Geeks", '1' => "for", '2' => "Geeks");  $c = 3.14;  $d = 7;    // Single argument  print "\n$a\n";    // Multiple argument  echo $c + $d . "\n";    // Return with internal output buffering  print\_r($b);  ?> |

**Output:**

GeeksforGeeks

10.14

Array

(

[0] => Geeks

[1] => for

[2] => Geeks

)