**Scope of Variable:**

Variable scope is known as its boundary within which it can be visible or accessed from code. In other words, it is the context within which a variable is defined. There are only two scopes available in PHP namely local and global scopes.

1. **Local variables (local scope)**
2. **Global variables (special global scope)**
3. **Static variables (local scope)**
4. **Function parameters (local scope)**

When a variable is accessed outside its scope it will cause PHP error Undefined Variable.

**1. Local Scope Variables**

A local scope is a restricted boundary of a variable within which code block it is declared. That block can be a function, class or any conditional span. The variable within this limited local scope is known as the local variable of that specific code block.

The following code block shows a PHP function. We have declared a variable *$count* inside this function. This variable is said to be a local variable of this function and it is in the local scope of the function block.

<?php

function calculate\_count() {

$count = 5;

//will print 5; the value of local variable

echo $count++;

}

?>

Local variables will be destroyed once the end of the code block is reached. Hence the same named variables can be declared within different local scopes.

**2. Global Scope Variables:**

As its name, the global scope provides widespread access to the variable declared in this scope. Variables in global scope can be accessed from anywhere from outside a function or class independent of its boundary.

PHP global variables can be defined by using **global** keyword. If we want to use global variables inside a function, we have to prefix the global keyword with the variable. The following code shows a code block to learn how to use the global keyword with a PHP variable to declared it as a global variable.

<?php

$count = 0;

function calculate\_count() {

global $count;

// will print 0 and increment global variable

echo $count++ . "<br/>";

}

calculate\_count();

echo $count;

?>

PHP has a predefined [superglobal](https://phppot.com/php/php-globals-with-egpcs-information/) variable called **$GLOBALS**. It is an associative array with the name of the variable as key and value as the array element. We can use this array variable to add an array of PHP variables in a global scope.

Let us change the above example with the global keyword by using $GLOBALS superglobal to access the variable in global scope.

<?php

$count = 0;

function calculate\_count() {

// will print 0 and increment global variable declared outside function

echo $GLOBALS["count"]++ . "<br/>";

}

calculate\_count();

echo $count;

?>

**3. Static Variables (local scope)**

A static variable is again a variable with local scope. But the difference with the regular local variable is that it is not destroyed outside the scope boundary. A variable can be defined by using the ‘static’ keyword inside a function. A static variable does not lose its value when the program execution goes past the scope boundary. But it can be accessed only within that boundary. Let me demonstrate it using the following example code,

<?php

function counter()

{

static $count = 0;

echo $count;

$count++;

}

?>

The above counter function has the static variable ‘count’ declared within its local scope. When the function execution is complete, the static count variable still retains its value for further computation. Every time the counter function is called, the value of count is incremented. The count value is initialized only once on the first call.

**4. Function Parameters (Local Scope):**

Function parameters (arguments) are local variables defined within the local scope of the function on which it is used as the argument.

**Scope and File Includes:**

The boundary of file includes does not demarcate the scope of variables. The scope of a variable is only governed by the function block and not based on the file include. A variable can be declared in a PHP file and used in another file by using ‘include’ or ‘require’ that file.

**Function Inside Function or Class:**

Remember that the scope in PHP is governed by a function block. Any new function declared anywhere starts a new scope. If an anonymous function is defined inside another function, the anonymous function has its own local scope.

<?php

function foo() {

$fruit = 'apple';

$bar = function () {

// $fruit cannot be accessed inside here

$animal = 'lion';

};

// $animal cannot be accessed outside here

}

?>

In the above example code, $fruit variable is restricted to the outer function and its scope does not span inside the anonymous inner function. The same way, $animal which is declared inside is not accessible in the outer function as its scope boundary is restricted to the inner function.

### **General Note:**

Whenever you want to use a variable in a different scope, the way in and way out is by passing as an argument. Do not use global scoped variable for such trivial use cases.

**Difference between echo and print:**

## The PHP echo Statement:

The echo statement can output one or more strings. In general terms, the echo statement can display anything that can be displayed to the browser, such as string, numbers, variables values, the results of expressions etc.

Since echo is a language construct not actually a function (like [if](https://www.tutorialrepublic.com/php-tutorial/php-if-else-statements.php) statement), you can use it without parentheses e.g. echo or echo(). However, if you want to pass more than one parameter to echo, the parameters must not be enclosed within parentheses.

## The PHP print Statement

You can also use the print statement (an alternative to echo) to display output to the browser. Like echo the print is also a language construct not a real function. So you can also use it without parentheses like: print or print().

Both echo and print statement works exactly the same way except that the print statement can only output one string, and always returns 1. That's why the echo statement considered marginally faster than the print statement since it doesn't return any value.