**PHP variables**

most important things to know about variables in PHP.

* All variables in PHP are denoted with a leading dollar sign ($).
* The value of a variable is the value of its most recent assignment.
* Variables are assigned with the = operator, with the variable on the left-hand side and the expression to be evaluated on the right.
* Variables can, but do not need, to be declared before assignment.
* Variables in PHP do not have intrinsic types - a variable does not know in advance whether it will be used to store a number or a string of characters.
* Variables used before they are assigned have default values.
* PHP does a good job of automatically converting types from one to another when necessary.
* PHP variables are Perl-like.

PHP has a total of **eight data types**which we use to construct our variables −

* **Integers** − are whole numbers, without a decimal point, like 4195.
* **Doubles** − are floating-point numbers, like 3.14159 or 49.1.
* **Booleans** − have only two possible values either true or false.
* **NULL** − is a special type that only has one value: NULL.
* **Strings** − are sequences of characters, like 'PHP supports string operations.'
* **Arrays** − are named and indexed collections of other values.
* **Objects** − are instances of programmer-defined classes, which can package up both other kinds of values and functions that are specific to the class.
* **Resources** − are special variables that hold references to resources external to PHP (such as database connections).

The first five are *simple types*, and the next two (arrays and objects) are compound - the compound types can package up other arbitrary values of arbitrary type, whereas the simple types cannot.

We will explain only simple data type in this chapters. Array and Objects will be explained separately.

Integers

They are whole numbers, without a decimal point, like 4195. They are the simplest type .they correspond to simple whole numbers, both positive and negative. Integers can be assigned to variables, or they can be used in expressions, like so −

$int\_var = 12345;

$another\_int = -12345 + 12345;

Integer can be in decimal (base 10), octal (base 8), and hexadecimal (base 16) format. Decimal format is the default, octal integers are specified with a leading 0, and hexadecimals have a leading 0x.

For most common platforms, the largest integer is (2\*\*31 . 1) (or 2,147,483,647), and the smallest (most negative) integer is . (2\*\*31 . 1) (or .2,147,483,647).

Doubles

They like 3.14159 or 49.1. By default, doubles print with the minimum number of decimal places needed. For example, the code −

<?php

 $many =2.2888800;

 $many\_2 =2.2111200;

 $few = $many + $many\_2;

print("$many + $many\_2 = $few <br>");

?>

It produces the following browser output −

2.28888 + 2.21112 = 4.5

Boolean

They have only two possible values either true or false. PHP provides a couple of constants especially for use as Booleans: TRUE and FALSE, which can be used like so −

if(TRUE)

print("This will always print<br>");

else

print("This will never print<br>");

Interpreting other types as Booleans

Here are the rules for determine the "truth" of any value not already of the Boolean type −

* If the value is a number, it is false if exactly equal to zero and true otherwise.
* If the value is a string, it is false if the string is empty (has zero characters) or is the string "0", and is true otherwise.
* Values of type NULL are always false.
* If the value is an array, it is false if it contains no other values, and it is true otherwise. For an object, containing a value means having a member variable that has been assigned a value.
* Valid resources are true (although some functions that return resources when they are successful will return FALSE when unsuccessful).
* Don't use double as Booleans.

Each of the following variables has the truth value embedded in its name when it is used in a Boolean context.

$true\_num=3+0.14159;

$true\_str="Tried and true"

$true\_array[49]="An array element";

$false\_array=array();

$false\_null= NULL;

$false\_num=999-999;

$false\_str="";

NULL

NULL is a special type that only has one value: NULL. To give a variable the NULL value, simply assign it like this −

$my\_var= NULL;

The special constant NULL is capitalized by convention, but actually it is case insensitive; you could just as well have typed −

$my\_var=null;

A variable that has been assigned NULL has the following properties −

* It evaluates to FALSE in a Boolean context.
* It returns FALSE when tested with IsSet() function.

Strings

They are sequences of characters, like "PHP supports string operations". Following are valid examples of string

$string\_1 ="This is a string in double quotes";

$string\_2 ='This is a somewhat longer, singly quoted string';

$string\_39 ="This string has thirty-nine characters";

$string\_0 ="";// a string with zero characters

Singly quoted strings are treated almost literally, whereas doubly quoted strings replace variables with their values as well as specially interpreting certain character sequences.

<?php

 $variable ="name";

 $literally ='My $variable will not print!';

print($literally);

print"<br>";

 $literally ="My $variable will print!";

print($literally);

?>

This will produce following result −

My $variable will not print!

My name will print

There are no artificial limits on string length - within the bounds of available memory, you ought to be able to make arbitrarily long strings.

Strings that are delimited by double quotes (as in "this") are preprocessed in both the following two ways by PHP −

* Certain character sequences beginning with backslash (\) are replaced with special characters
* Variable names (starting with $) are replaced with string representations of their values.

The escape-sequence replacements are −

* \n is replaced by the newline character
* \r is replaced by the carriage-return character
* \t is replaced by the tab character
* \$ is replaced by the dollar sign itself ($)
* \" is replaced by a single double-quote (")
* \\ is replaced by a single backslash (\)

Here Document

You can assign multiple lines to a single string variable using here document −

<?php

 $channel =<<<\_XML\_

<channel>

<title>What's For Dinner</title>

<link>http://menu.example.com/ </link>

<description>Choose what to eat tonight.</description>

</channel>

 \_XML\_;

 echo <<<END

 This uses the "here document" syntax to output multiple lines with variable

 interpolation. Note that the here document terminator must appear on a line with

 just a semicolon. no extra whitespace!

 END;

 print $channel;

?>

This will produce following result −

This uses the "here document" syntax to output

multiple lines with variable interpolation. Note

that the here document terminator must appear on a

line with just a semicolon. no extra whitespace!

<channel>

<title>What's For Dinner<title>

<link>http://menu.example.com/<link>

<description>Choose what to eat tonight.</description>

Variable Scope

Scope can be defined as the range of availability a variable has to the program in which it is declared. PHP variables can be one of four scope types −

* [Local variables](https://www.tutorialspoint.com/php/php_local_variables.htm)
* [Function parameters](https://www.tutorialspoint.com/php/php_function_parameters.htm)
* [Global variables](https://www.tutorialspoint.com/php/php_global_variables.htm)
* [Static variables](https://www.tutorialspoint.com/php/php_static_variables.htm)

Variable Naming

Rules for naming a variable is −

* Variable names must begin with a letter or underscore character.
* A variable name can consist of numbers, letters, underscores but you cannot use characters like + , - , % , ( , ) . & , etc

There is no size limit for variables.

A constant is a name or an identifier for a simple value. A constant value cannot change during the execution of the script. By default, a constant is case-sensitive. By convention, constant identifiers are always uppercase. A constant name starts with a letter or underscore, followed by any number of letters, numbers, or underscores. If you have defined a constant, it can never be changed or undefined.

To define a constant you have to use define() function and to retrieve the value of a constant, you have to simply specifying its name. Unlike with variables, you do not need to have a constant with a $. You can also use the function constant() to read a constant's value if you wish to obtain the constant's name dynamically.

constant() function

As indicated by the name, this function will return the value of the constant.

This is useful when you want to retrieve value of a constant, but you do not know its name, i.e. It is stored in a variable or returned by a function.

constant() example

<?php

define("MINSIZE",50);

 echo MINSIZE;

 echo constant("MINSIZE");// same thing as the previous line

?>

Only scalar data (boolean, integer, float and string) can be contained in constants.

Differences between constants and variables are

* There is no need to write a dollar sign ($) before a constant, where as in Variable one has to write a dollar sign.
* Constants cannot be defined by simple assignment, they may only be defined using the define() function.
* Constants may be defined and accessed anywhere without regard to variable scoping rules.
* Once the Constants have been set, may not be redefined or undefined.

Valid and invalid constant names

// Valid constant names

define("ONE", "first thing");

define("TWO2", "second thing");

define("THREE\_3", "third thing");

define("\_\_THREE\_\_", "third value");

// Invalid constant names

define("2TWO", "second thing");

PHP Magic constants

PHP provides a large number of predefined constants to any script which it runs.

There are five magical constants that change depending on where they are used. For example, the value of \_\_LINE\_\_ depends on the line that it's used on in your script. These special constants are case-insensitive and are as follows −

A few "magical" PHP constants are given below −

|  |  |
| --- | --- |
| **Sr.No** | **Name & Description** |
| 1 | **\_\_LINE\_\_**The current line number of the file. |
| 2 | **\_\_FILE\_\_**The full path and filename of the file. If used inside an include,the name of the included file is returned. Since PHP 4.0.2, **\_\_FILE\_\_** always contains an absolute path whereas in older versions it contained relative path under some circumstances. |
| 3 | **\_\_FUNCTION\_\_**The function name. (Added in PHP 4.3.0) As of PHP 5 this constant returns the function name as it was declared (case-sensitive). In PHP 4 its value is always lowercased. |
| 4 | **\_\_CLASS\_\_**The class name. (Added in PHP 4.3.0) As of PHP 5 this constant returns the class name as it was declared (case-sensitive). In PHP 4 its value is always lowercased. |
| 5 | **\_\_METHOD\_\_**The class method name. (Added in PHP 5.0.0) The method name is returned as it was declared (case-sensitive). |

**PHP Operators:**

**What is Operator?** Simple answer can be given using expression *4 + 5 is equal to 9*. Here 4 and 5 are called operands and + is called operator. PHP language supports following type of operators.

* Arithmetic Operators
* Comparison Operators
* Logical (or Relational) Operators
* Assignment Operators
* Conditional (or ternary) Operators

Lets have a look on all operators one by one.

Arithmetic Operators

There are following arithmetic operators supported by PHP language −

Assume variable A holds 10 and variable B holds 20 then −

[Show Examples](https://www.tutorialspoint.com/php/php_arithmatic_operators_examples.htm)

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| + | Adds two operands | A + B will give 30 |
| - | Subtracts second operand from the first | A - B will give -10 |
| \* | Multiply both operands | A \* B will give 200 |
| / | Divide numerator by de-numerator | B / A will give 2 |
| % | Modulus Operator and remainder of after an integer division | B % A will give 0 |
| ++ | Increment operator, increases integer value by one | A++ will give 11 |
| -- | Decrement operator, decreases integer value by one | A-- will give 9 |

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| == | Checks if the value of two operands are equal or not, if yes then condition becomes true. | (A == B) is not true. |
| != | Checks if the value of two operands are equal or not, if values are not equal then condition becomes true. | (A != B) is true. |
| > | Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true. | (A > B) is not true. |
| < | Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true. | (A < B) is true. |
| >= | Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true. | (A >= B) is not true. |
| <= | Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true. | (A <= B) is true. |

Comparison Operators

There are following comparison operators supported by PHP language

Assume variable A holds 10 and variable B holds 20 then −

[Show Examples](https://www.tutorialspoint.com/php/php_comparison_operators_examples.htm)

Logical Operators

There are following logical operators supported by PHP language

Assume variable A holds 10 and variable B holds 20 then −

[Show Examples](https://www.tutorialspoint.com/php/php_logical_operators_examples.htm)

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| and | Called Logical AND operator. If both the operands are true then condition becomes true. | (A and B) is true. |
| or | Called Logical OR Operator. If any of the two operands are non zero then condition becomes true. | (A or B) is true. |
| && | Called Logical AND operator. If both the operands are non zero then condition becomes true. | (A && B) is true. |
| || | Called Logical OR Operator. If any of the two operands are non zero then condition becomes true. | (A || B) is true. |
| ! | Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false. | !(A && B) is false. |

Assignment Operators

There are following assignment operators supported by PHP language −

[Show Examples](https://www.tutorialspoint.com/php/php_assignment_operators_examples.htm)

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| = | Simple assignment operator, Assigns values from right side operands to left side operand | C = A + B will assign value of A + B into C |
| += | Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand | C += A is equivalent to C = C + A |
| -= | Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand | C -= A is equivalent to C = C - A |
| \*= | Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand | C \*= A is equivalent to C = C \* A |
| /= | Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand | C /= A is equivalent to C = C / A |
| %= | Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand | C %= A is equivalent to C = C % A |

Conditional Operator

There is one more operator called conditional operator. This first evaluates an expression for a true or false value and then execute one of the two given statements depending upon the result of the evaluation. The conditional operator has this syntax −

[Show Examples](https://www.tutorialspoint.com/php/php_conditional_operator_examples.htm)

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| ? : | Conditional Expression | If Condition is true ? Then value X : Otherwise value Y |

Operators Categories

All the operators we have discussed above can be categorised into following categories −

* Unary prefix operators, which precede a single operand.
* Binary operators, which take two operands and perform a variety of arithmetic and logical operations.
* The conditional operator (a ternary operator), which takes three operands and evaluates either the second or third expression, depending on the evaluation of the first expression.
* Assignment operators, which assign a value to a variable.

Precedence of PHP Operators

Operator precedence determines the grouping of terms in an expression. This affects how an expression is evaluated. Certain operators have higher precedence than others; for example, the multiplication operator has higher precedence than the addition operator −

For example x = 7 + 3 \* 2; Here x is assigned 13, not 20 because operator \* has higher precedence than + so it first get multiplied with 3\*2 and then adds into 7.

|  |  |  |
| --- | --- | --- |
| **Category** | **Operator** | **Associativity** |
| Unary | ! ++ -- | Right to left |
| Multiplicative | \* / % | Left to right |
| Additive | + - | Left to right |
| Relational | <<= >>= | Left to right |
| Equality | == != | Left to right |
| Logical AND | && | Left to right |
| Logical OR | || | Left to right |
| Conditional | ?: | Right to left |
| Assignment | = += -= \*= /= %= | Right to left |

Here operators with the highest precedence appear at the top of the table, those with the lowest appear at the bottom. Within an expression, higher precedence operators will be evaluated first.

**Unit -2**

**PHP(this will cover the introduction, advantages,characterstic of PHP, how to install ,basic php syntax, comments, and variables used in java)**

The **PHP Hypertext Preprocessor (PHP)** is a programming language that allows web developers to create dynamic content that interacts with databases. PHP is basically used for developing web based software applications.

**Rasmus Lerdorf unleashed the first version of PHP way back in 1994**

**key advantages of learning PHP**:

* PHP is a recursive acronym for "PHP: Hypertext Preprocessor".
* PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
* It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
* PHP is pleasingly zippy in its execution, especially when compiled as an Apache module on the Unix side. The MySQL server, once started, executes even very complex queries with huge result sets in record-setting time.
* PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.
* PHP is forgiving: PHP language tries to be as forgiving as possible.
* PHP Syntax is C-Like.

Characteristics of PHP

Five important characteristics make PHP's practical nature possible −

* Simplicity
* Efficiency
* Security
* Flexibility
* Familiarity

What is a PHP File?

* PHP files can contain text, HTML, CSS, JavaScript, and PHP code
* PHP code is executed on the server, and the result is returned to the browser as plain HTML
* PHP files have extension ".php"

What Can PHP Do?

* PHP can generate dynamic page content
* PHP can create, open, read, write, delete, and close files on the server
* PHP can collect form data
* PHP can send and receive cookies
* PHP can add, delete, modify data in your database
* PHP can be used to control user-access
* PHP can encrypt data

Why PHP?

* PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
* PHP is compatible with almost all servers used today (Apache, IIS, etc.)
* PHP supports a wide range of databases
* PHP is free. Download it from the official PHP resource: [www.php.net](http://www.php.net/)
* PHP is easy to learn and runs efficiently on the server side

To start using PHP, you can:

* Find a web host with PHP and MySQL support
* Install a web server on your own PC, and then install PHP and MySQL

Use a Web Host With PHP Support

If your server has activated support for PHP you do not need to do anything.

Just create some .php files, place them in your web directory, and the server will automatically parse them for you.

You do not need to compile anything or install any extra tools.

Because PHP is free, most web hosts offer PHP support.

Set Up PHP on Your Own PC

However, if your server does not support PHP, you must:

* install a web server
* install PHP
* install a database, such as MySQL

The official PHP website (PHP.net) has installation instructions for PHP: <http://php.net/manual/en/install.php>

A PHP script is executed on the server, and the plain HTML result is sent back to the browser.

## Basic PHP Syntax

A PHP script can be placed anywhere in the document.

A PHP script starts with <?php and ends with ?>:

<?php
// PHP code goes here
?>

The default file extension for PHP files is ".php".

A PHP file normally contains HTML tags, and some PHP scripting code.

Below, we have an example of a simple PHP file, with a PHP script that uses a built-in PHP function "echo" to output the text "Hello World!" on a web page:

### Example

<!DOCTYPE html>
<html>
<body>

<h1>My first PHP page</h1>

<?php
echo "Hello World!";
?>

</body>
</html>

[Try it Yourself »](https://www.w3schools.com/php/phptryit.asp?filename=tryphp_syntax)

**Note:** PHP statements end with a semicolon (;).

## PHP Case Sensitivity

In PHP, keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are not case-sensitive.

In the example below, all three echo statements below are equal and legal:

### Example

<!DOCTYPE html>
<html>
<body>

<?php
ECHO "Hello World!<br>";
echo "Hello World!<br>";
EcHo "Hello World!<br>";
?>

</body>
</html>

[Try it Yourself »](https://www.w3schools.com/php/phptryit.asp?filename=tryphp_syntax_case1)

**Note:** However; all variable names are case-sensitive!

Look at the example below; only the first statement will display the value of the $color variable! This is because $color, $COLOR, and $coLOR are treated as three different variables:

### Example

<!DOCTYPE html>
<html>
<body>

<?php
$color = "red";
echo "My car is " . $color . "<br>";
echo "My house is " . $COLOR . "<br>";
echo "My boat is " . $coLOR . "<br>";
?>

</body>
</html>

## Comments in PHP

A comment in PHP code is a line that is not executed as a part of the program. Its only purpose is to be read by someone who is looking at the code.

Comments can be used to:

* Let others understand your code
* Remind yourself of what you did - Most programmers have experienced coming back to their own work a year or two later and having to re-figure out what they did. Comments can remind you of what you were thinking when you wrote the code

PHP supports several ways of commenting:

### Example

Syntax for single-line comments:

<!DOCTYPE html>
<html>
<body>

<?php
// This is a single-line comment

# This is also a single-line comment
?>

</body>
</html>

[Try it Yourself »](https://www.w3schools.com/php/phptryit.asp?filename=tryphp_comments)

### Example

Syntax for multiple-line comments:

<!DOCTYPE html>
<html>
<body>

<?php
/\*
This is a multiple-lines comment block
that spans over multiple
lines
\*/
?>

</body>
</html>

[Try it Yourself »](https://www.w3schools.com/php/phptryit.asp?filename=tryphp_comments2)

### Example

Using comments to leave out parts of the code:

<!DOCTYPE html>
<html>
<body>

<?php
// You can also use comments to leave out parts of a code line
$x = 5 /\* + 15 \*/ + 5;
echo $x;
?>

</body>
</html>

[Try it Yourself »](https://www.w3schools.com/php/phptryit.asp?filename=tryphp_comments3)

## Creating (Declaring) PHP Variables

In PHP, a variable starts with the $ sign, followed by the name of the variable:

### Example

<?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;
?>

[Try it Yourself »](https://www.w3schools.com/php/phptryit.asp?filename=tryphp_var)

After the execution of the statements above, the variable $txt will hold the value Hello world!, the variable $x will hold the value 5, and the variable $y will hold the value 10.5.

**Note:** When you assign a text value to a variable, put quotes around the value.

**Note:** Unlike other programming languages, PHP has no command for declaring a variable. It is created the moment you first assign a value to it.

Think of variables as containers for storing data.

## PHP Variables

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total\_volume).

Rules for PHP variables:

* A variable starts with the $ sign, followed by the name of the variable
* A variable name must start with a letter or the underscore character
* A variable name cannot start with a number
* A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_ )
* Variable names are case-sensitive ($age and $AGE are two different variables)

Remember that PHP variable names are case-sensitive!

## Output Variables

The PHP echo statement is often used to output data to the screen.

The following example will show how to output text and a variable:

### Example

<?php
$txt = "my India";
echo "I love $txt!";
?>

[Try it Yourself »](https://www.w3schools.com/php/phptryit.asp?filename=tryphp_var2)

The following example will produce the same output as the example above:

### Example

<?php
$txt = "my India";
echo "I love " . $txt . "!";
?>

[Try it Yourself »](https://www.w3schools.com/php/phptryit.asp?filename=tryphp_var3)

The following example will output the sum of two variables:

### Example

<?php
$x = 5;
$y = 4;
echo $x + $y;
?>

[Try it Yourself »](https://www.w3schools.com/php/phptryit.asp?filename=tryphp_var4)

**Note:** You will learn more about the echo statement and how to output data to the screen in the next chapter.

## PHP is a Loosely Typed Language

In the example above, notice that we did not have to tell PHP which data type the variable is.

PHP automatically associates a data type to the variable, depending on its value. Since the data types are not set in a strict sense, you can do things like adding a string to an integer without causing an error.

In PHP 7, type declarations were added. This gives an option to specify the data type expected when declaring a function, and by enabling the strict requirement, it will throw a "Fatal Error" on a type mismatch.

## PHP Variables Scope

In PHP, variables can be declared anywhere in the script.

The scope of a variable is the part of the script where the variable can be referenced/used.

PHP has three different variable scopes:

* local
* global
* static

## Global and Local Scope

A variable declared **outside** a function has a GLOBAL SCOPE and can only be accessed outside a function:

### Example

Variable with global scope:

<?php
$x = 5; // global scope

function myTest() {
  // using x inside this function will generate an error
  echo "<p>Variable x inside function is: $x</p>";
}
myTest();

echo "<p>Variable x outside function is: $x</p>";
?>

[Try it Yourself »](https://www.w3schools.com/php/phptryit.asp?filename=tryphp_var_global)

A variable declared **within** a function has a LOCAL SCOPE and can only be accessed within that function:

### Example

Variable with local scope:

<?php
function myTest() {
  $x = 5; // local scope
  echo "<p>Variable x inside function is: $x</p>";
}
myTest();

// using x outside the function will generate an error
echo "<p>Variable x outside function is: $x</p>";
?>

[Try it Yourself »](https://www.w3schools.com/php/phptryit.asp?filename=tryphp_var_local)

You can have local variables with the same name in different functions, because local variables are only recognized by the function in which they are declared.

## PHP The global Keyword

The global keyword is used to access a global variable from within a function.

To do this, use the global keyword before the variables (inside the function):

### Example

<?php
$x = 5;
$y = 10;

function myTest() {
  global $x, $y;
  $y = $x + $y;
}

myTest();
echo $y; // outputs 15
?>

[Try it Yourself »](https://www.w3schools.com/php/phptryit.asp?filename=tryphp_var_global_keyword)

PHP also stores all global variables in an array called $GLOBALS[*index*]. The *index* holds the name of the variable. This array is also accessible from within functions and can be used to update global variables directly.

The example above can be rewritten like this:

### Example

<?php
$x = 5;
$y = 10;

function myTest() {
  $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];
}

myTest();
echo $y; // outputs 15
?>

[Try it Yourself »](https://www.w3schools.com/php/phptryit.asp?filename=tryphp_var_globals)

## PHP The static Keyword

Normally, when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job.

To do this, use the static keyword when you first declare the variable:

### Example

<?php
function myTest() {
  static $x = 0;
  echo $x;
  $x++;
}

myTest();
myTest();
myTest();
?>

[Try it Yourself »](https://www.w3schools.com/php/phptryit.asp?filename=tryphp_var_static)

Then, each time the function is called, that variable will still have the information it contained from the last time the function was called.

**Note:** The variable is still local to the function.