PHP

Introduction to PHP

What is PHP?

- PHP is a recursive acronym for "PHP: Hypertext Preprocessor". It is a server side scripting language to make dynamic and interactive web pages.
- PHP is a open source language
- PHP is a easy to learn and runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- PHP is a widely-used, free, and efficient alternative to competitors such as Microsoft's ASP
- PHP is a server-side scripting language designed for web development but also used as a general-purpose programming language.

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
- PHP can be used to include files
- PHP can be used to validate form data
- PHP can be used to filter data
- PHP can be used to send and receive emails

What is a PHP File?

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

PHP Syntax

```php
<?php
// PHP code goes here
?>
```

Statement terminator

```php
<?php
    echo "Hello World";
?>
```

Line breaks

- PHP ignores line breaks
- PHP ignores white spaces
- PHP ignores tabs

Way to break a line in output

```php
<?php
    echo "Hello World<br>";
?>
```

Comments

- Single line comments
- Multi line comments
- Doc comments

Example

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

# This is also a single line comment

/* This is a multi line comment
   This is the second line of the comment
   This is the third line of the comment */

/**
 * This is a doc comment
 */
?>
```

Output in PHP

- echo
- print
- print_r
- var_dump

echo

- echo is a language construct, not a function
- echo can take multiple parameters
- echo has no return value
- echo is marginally faster than print

echo example

```
<?php
    echo "Hello World";
    echo "Hello World", "Hello World";
?>
```

print

- print is a function, not a language construct
- print can take one argument
- print has a return value of 1 so it can be used in expressions
- print is slower than echo

print example

```
<?php
    print "Hello World";
?>
```

print_r

- print_r is a function, not a language construct
- print_r can take one argument

print_r example

```
<?php
    print_r("Hello World");
?>
```

var_dump

- var_dump is a function, not a language construct

- var_dump can take one argument

var_dump example

```
<?php
    var_dump("Hello World");
?>
```

Data types

- String
- Integer
- Float
- Boolean
- Array
- Object
- NULL
- Resource

String

- A string is a sequence of characters, like "Hello world!"
- A string can be any text inside quotes
- You can use single or double quotes for strings

String example

```
<?php
    $x = "Hello world!";
    $y = 'Hello world!';
?>
```

Integer

- An integer is a number without a decimal point, like 4195
- Integers can be specified in: decimal (base 10), hexadecimal (base 16), octal (base 8), or binary (base 2) notation

Integer example

```php
<?php
    $x = 5985;
    var_dump($x);
    echo "<br>";
    $x = -345; // negative number
    var_dump($x);
    echo "<br>";
    $x = 0x8C; // hexadecimal number
    var_dump($x);
    echo "<br>";
    $x = 047; // octal number
    var_dump($x);
?>
```

Float

- A float (floating point number) is a number with a decimal point or a number in exponential form
- Floats can also be specified in scientific notation (for example: 2.0e3 or 7E-10)

Float example

```php
<?php
    $x = 10.365;
    var_dump($x);
    echo "<br>";
    $x = 2.4e3;
    var_dump($x);
    echo "<br>";
    $x = 8E-5;
    var_dump($x);
?>
```

Boolean

- A Boolean represents two possible states: TRUE or FALSE

Boolean example

```php
<?php
    $x = true;
    $y = false;
?>
```

Array

- An array stores multiple values in one single variable

- An array is a special variable, which can hold more than one value at a time
- An array can hold many values under a single name, and you can access the values by referring to an index number

Array example

```php
<?php
    $cars = array("Volvo", "BMW", "Toyota");
    echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

Object

- An object is a data type which stores data and information on how to process that data
- An object is a combination of variables and functions
- An object is like a variable, but it can also have functions
- Objects are created from classes, and they are the most important aspect of object-oriented programming

Object example

```php
<?php
    class Car {
        function Car() {
            $this->model = "VW";
        }
    }
    // create an object
    $herbie = new Car();
    // show object properties
    echo $herbie->model;
?>
```

NULL

- NULL is a special data type which can have only one value: NULL
- A variable of data type NULL is a variable that has no value assigned to it
- NULL is the only possible value of type NULL
- NULL is case-sensitive

NULL example

```php
<?php
    $x = "Hello world!";
    $x = null;
    var_dump($x);
?>
```

Resource

- A resource is a special variable, holding a reference to an external resource
- A resource is created and used by a special function

Resource example

```php
<?php
    $file = fopen("welcome.txt", "r");
?>
```

Variables

- Local variables
- Global variables
- Static variables

Local variables

- Local variables are the variables that are declared inside a function
- Local variables are only accessible within the function
- Local variables are destroyed when the function ends

Local variables example

```php
<?php
    function myFunction() {
        $x = 5; // local scope
        echo "<p>Variable x inside function is: $x</p>";
    }
    myFunction();
    // using x outside the function will generate an error
    echo "<p>Variable x outside function is: $x</p>";
?>
```

Global variables

- Global variables are declared outside a function
- Global variables are accessible inside a function
- Global variables are accessible from anywhere in the script

Global variables example

```php
<?php
    $x = 5; // global scope
    function myFunction() {
        // using x inside this function will generate an error
        echo "<p>Variable x inside function is: $x</p>";
    }
    myFunction();
    echo "<p>Variable x outside function is: $x</p>";
?>
```

Static variables

- Static variables are declared inside a function
- Static variables are only accessible within the function
- Static variables are not destroyed when the function ends

Static variables example

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

Constants

- Predefined constants
- User-defined constants

Predefined constants

- Predefined constants are constants that are built into PHP

- Predefined constants are automatically defined by PHP
- Predefined constants are always available in all scopes

Predefined constants example

```
<?php
    echo PHP_VERSION;
    echo "<br>";
    echo PHP_OS;
    echo "<br>";
    echo __LINE__;
    echo "<br>";
    echo __FILE__;
    echo "<br>";
    echo __DIR__;
?>
```

User-defined constants

- User-defined constants are constants that you yourself create
- User-defined constants are created with the define() function
- User-defined constants are always global

User-defined constants example

```
<?php
    define("GREETING", "Welcome to W3Schools.com!");
    echo GREETING;
?>
```

Operators

- Arithmetic operators
- Assignment operators
- Comparison operators
- Increment/Decrement operators
- Logical operators
- String operators
- Array operators
- Conditional assignment operators

Arithmetic operators

- Arithmetic operators are used with numeric values to perform common mathematical operations, such as addition, subtraction, multiplication etc.

Operator	Name	Example
+	Addition	`$x + $y`
-	Subtraction	`$x - $y`
*	Multiplication	`$x * $y`
/	Division	`$x / $y`
%	Modulus	`$x % $y`
**	Exponentiation	`$x ** $y`

Assignment operators

- Assignment operators are used with numeric values to write a value to a variable

Operator	Name	Example
=	Simple assignment	`$x = $y`
+=	Addition assignment	`$x += $y`
-=	Subtraction assignment	`$x -= $y`
*=	Multiplication assignment	`$x *= $y`
/=	Division assignment	`$x /= $y`
%=	Modulus assignment	`$x %= $y`
.=	Concatenation assignment	`$x .= $y`

Comparison operators

- Comparison operators are used to compare two values (number or string)

Operator	Name	Example
==	Equal	`$x == $y`
===	Identical	`$x === $y`

Operator	Name	Example
!=	Not equal	`$x != $y`
<>	Not equal	`$x <> $y`
!==	Not identical	`$x !== $y`
>	Greater than	`$x > $y`
<	Less than	`$x < $y`
>=	Greater than or equal to	`$x >= $y`
<=	Less than or equal to	`$x <= $y`
<=>	Spaceship	`$x <=> $y`

Increment/Decrement operators

- Increment operators increase a variable's value by one
- Decrement operators decrease a variable's value by one

Operator	Name	Example
++	Pre-increment	`++$x`
++	Post-increment	`$x++`
--	Pre-decrement	`--$x`
--	Post-decrement	`$x--`

Logical operators

- Logical operators are used to combine conditional statements

Operator	Name	Example
and	And	`$x and $y`
or	Or	`$x or $y`
xor	Xor	`$x xor $y`
&&	And	`$x && $y`

Operator	Name	Example
!	Not	`!$x`

String operators

- String operators are used to concatenate two strings

Operator	Name	Example
.	Concatenation	`$txt1 . $txt2`
.=	Concatenation assignment	`$txt1 .= $txt2`

Array operators

- Array operators are used to compare arrays

Operator	Name	Example
+	Union	`$x + $y`
==	Equality	`$x == $y`
===	Identity	`$x === $y`
!=	Inequality	`$x != $y`
<>	Inequality	`$x <> $y`
!==	Non-identity	`$x !== $y`

Conditional assignment operators

- Conditional assignment operators are used to assign a value to a variable based on a condition

Operator	Name	Example
??	Null coalescing	`$x ?? $y`
?:	Ternary	`$x ?: $y`

Conditional statements

- if statement
- if...else statement
- if...elseif...else statement
- switch statement

if statement

- The if statement executes some code if one condition is true
- Syntax

```
if (condition) {
    code to be executed if condition is true;
}
```

if statement example

```
<?php
    $t = date("H");
    if ($t < "20") {
        echo "Have a good day!";
    }
?>
```

if...else statement

- The if...else statement executes some code if a condition is true and another code if that condition is false
- Syntax

```
if (condition) {
    code to be executed if condition is true;
} else {
    code to be executed if condition is false;
}
```

if...else statement example

```php
<?php
    $t = date("H");
    if ($t < "20") {
        echo "Have a good day!";
    } else {
        echo "Have a good night!";
    }
?>
```

if...elseif...else statement

- The if...elseif...else statement executes different codes for more than two conditions
- Syntax

```php
if (condition) {
    code to be executed if this condition is true;
} elseif (condition) {
    code to be executed if first condition is false and this condition is true;
} else {
    code to be executed if all conditions are false;
}
```

if...elseif...else statement example

```php
<?php
    $t = date("H");
    if ($t < "10") {
        echo "Have a good morning!";
    } elseif ($t < "20") {
        echo "Have a good day!";
    } else {
        echo "Have a good night!";
    }
?>
```

switch statement

- The switch statement is used to perform different actions based on different conditions
- Syntax

```
switch (n) {
    case label1:
        code to be executed if n=label1;
        break;
    case label2:
        code to be executed if n=label2;
        break;
    case label3:
        code to be executed if n=label3;
        break;
    ...
    default:
        code to be executed if n is different from all labels;
}
```

switch statement example

```
<?php
    $favcolor = "red";
    switch ($favcolor) {
        case "red":
            echo "Your favorite color is red!";
            break;
        case "blue":
            echo "Your favorite color is blue!";
            break;
        case "green":
            echo "Your favorite color is green!";
            break;
        default:
            echo "Your favorite color is neither red, blue, nor green!";
    }
?>
```

Loops

- while loop
- do...while loop
- for loop
- foreach loop

while loop

- The while loop executes a block of code as long as the specified condition is true
- Syntax

```
while (condition is true) {
    code to be executed;
}
```

while loop example

```
<?php
    $x = 1;
    while($x <= 5) {
        echo "The number is: $x <br>";
        $x++;
    }
?>
```

do...while loop

- The do...while loop will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true.
- Syntax

```
do {
    code to be executed;
} while (condition is true);
```

do...while loop example

```
<?php
    $x = 1;
    do {
        echo "The number is: $x <br>";
        $x++;
    } while ($x <= 5);
?>
```

for loop

- The for loop is used when you know in advance how many times the script should run.
- Syntax

```
for (init counter; test counter; increment counter) {
    code to be executed;
}
```

for loop example

```php
<?php
    for ($x = 0; $x <= 10; $x++) {
        echo "The number is: $x <br>";
    }
?>
```

foreach loop

- The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.
- Syntax

```php
foreach ($array as $value) {
    code to be executed;
}
```

foreach loop example

```php
<?php
    $colors = array("red", "green", "blue", "yellow");
    foreach ($colors as $value) {
        echo "$value <br>";
    }
?>
```

Functions

- User-defined functions
- Built-in functions

User-defined functions

- A function is a block of statements that can be used repeatedly in a program.
- A function will not execute automatically when a page loads.
- A function will be executed by a call to the function.
- Syntax

```php
function functionName() {
    code to be executed;
}
```

User-defined functions example

```php
<?php
    function writeMsg() {
        echo "Hello world!";
    }
    writeMsg();
?>
```

Built-in functions

- A function is a block of statements that can be used repeatedly in a program.
- Syntax

```
function functionName() {
    code to be executed;
}
```

Built-in functions example

```php
<?php
    echo strlen("Hello world!");
?>
```

Function arguments

- Arguments are specified after the function name, inside the parentheses.
- You can add as many arguments as you want, just separate them with a comma.

function arguments example

```php
<?php
    function familyName($fname) {
        echo "$fname Refsnes.<br>";
    }
    familyName("Jani");
    familyName("Hege");
    familyName("Stale");
    familyName("Kai Jim");
    familyName("Borge");
?>
```

Default argument value

- If we call the function setHeight() without arguments it will use the default value as argument.

default argument value example

```
<?php
    function setHeight($minheight = 50) {
        echo "The height is : $minheight <br>";
    }
    setHeight(350);
    setHeight(); // will use the default value of 50
    setHeight(135);
    setHeight(80);
?>
```

Returning values

- To let a function return a value, use the return statement.

returning values example

```
<?php
    function sum($x, $y) {
        $z = $x + $y;
        return $z;
    }
    echo "5 + 10 = " . sum(5,10) . "<br>";
    echo "7 + 13 = " . sum(7,13) . "<br>";
    echo "2 + 4 = " . sum(2,4);
?>
```

Classes and objects

- Class
- Object

Class

- A class is a template for objects, and objects are instances of a class.
- A class is a user-defined data type, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class. A class is like a blueprint while an object is like a house built using the blueprint.

- Syntax

```
class ClassName {
    // properties and methods goes here
}
```

Class example

```
<?php
    class Car {
        function Car() {
            $this->model = "VW";
        }
    }
    $herbie = new Car();
    echo $herbie->model;
?>
```

Object

- An object is an instance of a class.
- An object is a basic unit of Object Oriented Programming and represents the real life entities.
- Object is a real-world entity such as pen, bike, chair, table, keyboard, mouse, etc.
- Syntax

```
class ClassName {
    // properties and methods goes here
}
```

Object example

```
<?php
    class Car {
        function Car() {
            $this->model = "VW";
        }
    }
    $herbie = new Car();
    echo $herbie->model;
?>
```

Break and continue

- break
- continue

break

- The break statement can be used to jump out of a loop.
- Syntax

```
break;
```

break example

```
<?php
    for ($x = 0; $x < 10; $x++) {
        if ($x == 4) {
            break;
        }
        echo "The number is: $x <br>";
    }
?>
```

continue

- The continue statement can be used to skip the current iteration in a loop.
- Syntax

```
continue;
```

continue example

```
<?php
    for ($x = 0; $x < 10; $x++) {
        if ($x == 4) {
            continue;
        }
        echo "The number is: $x <br>";
    }
?>
```

Strings in PHP

- strlen()

- str_word_count()
- strrev()
- strpos()
- str_replace()

Function	Description	Example
strlen()	Returns the length of a string	`echo strlen("Hello world!");`
str_word_count()	Counts the number of words in a string	`echo str_word_count("Hello world!");`
strrev()	Reverses a string	`echo strrev("Hello world!");`
strpos()	Searches for a specific text within a string	`echo strpos("Hello world!", "world");`
str_replace()	Replaces some characters with some other characters in a string	`echo str_replace("world", "Dolly", "Hello world!");`

Arrays in PHP

- An array is a special variable, which can hold more than one value at a time. it is a container which can hold multiple values at the same time.
- An array is a data structure which stores values of same data type.
- Array is a collection of elements which are accessed using an index.

Creating an array

- There are three ways to create an array in PHP.

1. array() function
2. array keyword
3. short array syntax

array() function

- Syntax

```
array(value1, value2, value3, ...)
```

array() function example

```
<?php
    $cars = array("Volvo", "BMW", "Toyota");
    echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

array keyword

- Syntax

```
array = [value1, value2, value3, ...]
```

array keyword example

```
<?php
    $cars = ["Volvo", "BMW", "Toyota"];
    echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

short array syntax

- Syntax

```
array = [value1, value2, value3, ...]
```

short array syntax example

```
<?php
    $cars = ["Volvo", "BMW", "Toyota"];
    echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

Types of arrays

- Indexed arrays

- Associative arrays
- Multidimensional arrays

Indexed arrays

- An indexed array is an array that has a numeric index.
- Syntax

```
array = [value1, value2, value3, ...]
```

Indexed arrays example

```
<?php
    $cars = ["Volvo", "BMW", "Toyota"];
    echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

Associative arrays

- An associative array is an array that uses named keys that you assign to it.
- Syntax

```
array = [key1 => value1, key2 => value2, key3 => value3, ...]
```

Associative arrays example

```
<?php
    $age = ["Peter" => "35", "Ben" => "37", "Joe" => "43"];
    echo "Peter is " . $age['Peter'] . " years old.";
?>
```

Multidimensional arrays

- A multidimensional array is an array containing one or more arrays.
- Syntax

```
array = [
    [value1, value2, value3, ...]
    [value1, value2, value3, ...]
    [value1, value2, value3, ...]
];
```

Multidimensional arrays example

```php
<?php
    $cars = [
        ["Volvo", 22, 18],
        ["BMW", 15, 13],
        ["Saab", 5, 2],
        ["Land Rover", 17, 15]
    ];
    echo $cars[0][0] . ": In stock: " . $cars[0][1] . ", sold: " . $cars[0][2] . ".<br>";
    echo $cars[1][0] . ": In stock: " . $cars[1][1] . ", sold: " . $cars[1][2] . ".<br>";
    echo $cars[2][0] . ": In stock: " . $cars[2][1] . ", sold: " . $cars[2][2] . ".<br>";
    echo $cars[3][0] . ": In stock: " . $cars[3][1] . ", sold: " . $cars[3][2] . ".<br>";
?>
```

Array functions

Function	Description	Example
count()	Counts the number of elements in an array	`echo count($cars);`
sort()	Sorts an array in ascending order	`sort($cars);`
rsort()	Sorts an array in descending order	`rsort($cars);`
asort()	Sorts an associative array in ascending order, according to the value	`asort($age);`
ksort()	Sorts an associative array in ascending order, according to the key	`ksort($age);`
arsort()	Sorts an associative array in descending order, according to the value	`arsort($age);`
krsort()	Sorts an associative array in descending order, according to the key	`krsort($age);`

Accessing HTML data using get and post method

- The HTML form data can be accessed using the $_GET and $_POST variables.
- The $_GET variable is used to collect form data after submitting an HTML form with method="get".

- The $_POST variable is used to collect form data after submitting an HTML form with method="post".

get method

- The $_GET variable is an associative array that contains the data sent using the GET method.
- The data sent using the GET method is visible to everyone.
- The data sent using the GET method is visible in the URL.
- The data sent using the GET method is limited to 1024 characters.
- The data sent using the GET method is not secure.
- The data sent using the GET method is not recommended for sensitive data.
- The data sent using the GET method is not recommended for sending passwords or credit card numbers.
- The data sent using the GET method is not recommended for sending large amounts of data.
- The data sent using the GET method is not recommended for sending binary data, such as images or files.

get method example

```php
<?php
    echo "Study " . $_GET['subject'] . " at " . $_GET['web'];
?>
```

- subject and web are the names of the input fields in the HTML form.

post method

- The $_POST variable is an associative array that contains the data sent using the POST method.
- The data sent using the POST method is not visible to everyone.
- The data sent using the POST method is not visible in the URL.
- The data sent using the POST method is not limited to 1024 characters.
- The data sent using the POST method is secure.
- The data sent using the POST method is recommended for sensitive data.
- The data sent using the POST method is recommended for sending passwords or credit card numbers.
- The data sent using the POST method is recommended for sending large amounts of data.
- The data sent using the POST method is recommended for sending binary data, such as images or files.

post method example

```php
<?php
    echo "Study " . $_POST['subject'] . " at " . $_POST['web'];
?>
```

- subject and web are the names of the input fields in the HTML form.

User defined functions and there usage

- A function is a block of statements that can be used repeatedly in a program.
- A function will not execute immediately when a page loads.
- A function will be executed by a call to the function.

function syntax

```php
function functionName() {
    // code to be executed
}
```

function example

```php
<?php
    function writeMsg() {
        echo "Hello world!";
    }
    writeMsg(); // call the function
?>
```

Date and Time in PHP

- date()
- time()
- mktime()
- strtotime()

date()

- it is used to format a local time/date.
- Syntax

```
date(format, timestamp)
```

date() example

```
<?php
    echo "Today is " . date("Y/m/d") . "<br>";
    echo "Today is " . date("Y.m.d") . "<br>";
    echo "Today is " . date("Y-m-d") . "<br>";
    echo "Today is " . date("l");
?>
```

time()

- it is used to return the current time in seconds since the Unix Epoch (January 1 1970 00:00:00 GMT).
- Syntax

```
time()
```

time() example

```
<?php
    echo "The time is " . date("h:i:sa");
?>
```

- h is hour in 12-hour format
- i is minutes
- s is seconds
- a is am or pm

mktime()

- it is used to get Unix timestamp for a date.
- Syntax

```
mktime(hour, minute, second, month, day, year)
```

mktime() example

```php
<?php
    $d = mktime(11, 14, 54, 8, 12, 2014);
    echo "Created date is " . date("Y-m-d h:i:sa", $d);
?>
```

strtotime()

- it is used to convert a human readable string to a Unix timestamp.
- Syntax

```
strtotime(time, now)
```

strtotime() example

```php
<?php
    $d=strtotime("10:30pm April 15 2014");
    echo "Created date is " . date("Y-m-d h:i:sa", $d) . "<br>";

    $d=strtotime("tomorrow");
    echo date("Y-m-d h:i:sa", $d) . "<br>";

    $d=strtotime("next Saturday");
    echo date("Y-m-d h:i:sa", $d) . "<br>";

    $d=strtotime("+3 Months");
    echo date("Y-m-d h:i:sa", $d) . "<br>";
?>
```

- Y is year
- m is month
- d is day
- h is hour
- i is minutes
- s is seconds
- a is am or pm

Number in PHP

Function	Description	Example
abs()	Returns the absolute (positive) value of a number	`echo abs(-6.7);`

Function	Description	Example
acos()	Returns the arc cosine of a number	`echo acos(0.64);`
acosh()	Returns the inverse hyperbolic cosine of a number	`echo acosh(64);`
asin()	Returns the arc sine of a number	`echo asin(0.64);`
asinh()	Returns the inverse hyperbolic sine of a number	`echo asinh(64);`
atan()	Returns the arc tangent of a number	`echo atan(0.64);`
atan2()	Returns the arc tangent of two variables	`echo atan2(0.64, 0.64);`
atanh()	Returns the inverse hyperbolic tangent of a number	`echo atanh(0.64);`
base_convert()	Converts a number between arbitrary bases	`echo base_convert("FF", 16, 2);`
bindec()	Converts a binary number to a decimal number	`echo bindec(11111111);`
ceil()	Rounds a number up to the nearest integer	`echo ceil(0.60);`
cos()	Returns the cosine of a number	`echo cos(0.64);`
cosh()	Returns the hyperbolic cosine of a number	`echo cosh(0.64);`
decbin()	Converts a decimal number to a binary number	`echo decbin(255);`
dechex()	Converts a decimal number to a hexadecimal number	`echo dechex(255);`
decoct()	Converts a decimal number to an octal number	`echo decoct(255);`
deg2rad()	Converts the number in degrees to the radian equivalent	`echo deg2rad(45);`

Function	Description	Example
exp()	Returns e raised to the power of a number	`echo exp(0.64);`
expm1()	Returns exp(number) - 1, computed in a way that is accurate even when the value of number is close to zero	`echo expm1(0.64);`
floor()	Rounds a number down to the nearest integer	`echo floor(0.60);`
fmod()	Returns the floating point remainder (modulo) of the division of the arguments	`echo fmod(0.64, 0.64);`
getrandmax()	Returns the largest possible random value	`echo getrandmax();`
hexdec()	Converts a hexadecimal number to a decimal number	`echo hexdec(FF);`
hypot()	Calculates the length of the hypotenuse of a right-angle triangle	`echo hypot(0.64, 0.64);`
is_finite()	Finds whether a value is a legal finite number	`echo is_finite(0.64);`
is_infinite()	Finds whether a value is infinite	`echo is_infinite(0.64);`
is_nan()	Finds whether a value is not a number	`echo is_nan(0.64);`
lcg_value()	Returns a random number from the combined linear congruential generator	`echo lcg_value();`
log()	Returns the natural logarithm of a number	`echo log(0.64);`
log10()	Returns the base-10 logarithm of a number	`echo log10(0.64);`
log1p()	Returns log(1 + number), computed in a way that is accurate even when the value of number is close to zero	`echo log1p(0.64);`
max()	Finds highest value	`echo max(0.64, 0.64);`

Function	Description	Example
min()	Finds lowest value	`echo min(0.64, 0.64);`
mt_getrandmax()	Returns the largest possible random value	`echo mt_getrandmax();`
mt_rand()	Generates a better random value	`echo mt_rand();`
mt_srand()	Seeds the better random number generator	`echo mt_srand();`
octdec()	Converts an octal number to a decimal number	`echo octdec(377);`
pi()	Returns the value of pi	`echo pi();`
pow()	Exponentiation operator	`echo pow(0.64, 0.64);`
rad2deg()	Converts the radian number to the equivalent number in degrees	`echo rad2deg(0.64);`
rand()	Generates a random integer	`echo rand();`
round()	Rounds a number	`echo round(0.60);`
sin()	Returns the sine of a number	`echo sin(0.64);`
sinh()	Returns the hyperbolic sine of a number	`echo sinh(0.64);`
sqrt()	Returns the square root of a number	`echo sqrt(0.64);`
srand()	Seeds the random number generator	`echo srand();`
tan()	Returns the tangent of a number	`echo tan(0.64);`
tanh()	Returns the hyperbolic tangent of a number	`echo tanh(0.64);`

Miscellaneous Library Functions in PHP

Function	Description	Example
constant()	Returns the value of a constant	`echo constant("PHP_VERSION");`

Function	Description	Example
defined()	Checks whether a given named constant exists	`echo defined("PHP_VERSION");`
die()	Prints a message and exits the current script	`echo die("Something went wrong");`
echo()	Outputs one or more strings	`echo echo("Something went wrong");`
empty()	Checks whether a variable is empty	`echo empty($var);`
exit()	Prints a message and exits the current script	`echo exit("Something went wrong");`
eval()	Evaluates a string as PHP code	`echo eval("echo 'Hello World';");`
get_browser()	Returns information about the user's browser	`echo get_browser();`
get_defined_constants()	Returns an array with the names of all the constants and their values	`echo get_defined_constants();`
get_defined_functions()	Returns an array with the names of all the defined functions	`echo get_defined_functions();`
get_defined_vars()	Returns an associative array with the names of all the variables and their values	`echo get_defined_vars();`
get_include_path()	Returns the current include_path configuration option	`echo get_include_path();`

Function	Description	Example
get_included_files()	Returns an array with the names of all the files that have been included, either with require or include	`echo get_included_files();`
get_loaded_extensions()	Returns an array with the names of all the modules compiled and loaded	`echo get_loaded_extensions();`
get_magic_quotes_gpc()	Gets the current configuration setting of magic_quotes_gpc	`echo get_magic_quotes_gpc();`
get_magic_quotes_runtime()	Gets the current active configuration setting of magic_quotes_runtime	`echo get_magic_quotes_runtime();`
getenv()	Gets the value of an environment variable	`echo getenv("PATH");`
gettype()	Gets the type of a variable	`echo gettype($var);`
import_request_variables()	Imports GET/POST/Cookie variables into the global scope	`echo import_request_variables("GPC");`
ini_alter()	Changes the value of a configuration option	`echo ini_alter("display_errors", "On");`
ini_get()	Gets the value of a configuration option	`echo ini_get("display_errors");`