PHP básico

Índice

[PHP 3](#_Toc71992059)

[Variables 3](#_Toc71992060)

[Enteros (Integer) 4](#_Toc71992061)

[Numero de coma flotante (double) 4](#_Toc71992062)

[Cadena de caracteres 4](#_Toc71992063)

[Valores binarios (Boolean) 4](#_Toc71992064)

[Valores nulos (NULL) 5](#_Toc71992065)

[Arrays 5](#_Toc71992066)

[In\_array () 6](#_Toc71992067)

[List () 6](#_Toc71992068)

[Variables 6](#_Toc71992069)

[Funciones de variables 6](#_Toc71992070)

[Constante 7](#_Toc71992071)

[Constantes predefinidas 8](#_Toc71992072)

[Defined () 8](#_Toc71992073)

[Operadores 8](#_Toc71992074)

[Operadores aritméticos 9](#_Toc71992075)

[Operadores de comparación 9](#_Toc71992076)

[Operadores lógicos 10](#_Toc71992077)

[Estructuras de control 11](#_Toc71992078)

[If 12](#_Toc71992079)

[Else if 12](#_Toc71992080)

[Switch 12](#_Toc71992081)

[While 13](#_Toc71992082)

[Do-While 14](#_Toc71992083)

[For 15](#_Toc71992084)

[ForEach 18](#_Toc71992085)

[Break 19](#_Toc71992086)

[Continue 20](#_Toc71992087)

# PHP

## Variables

En PHP las variables se representan con un signo de dólar seguido por el nombre de la variable. El nombre de la variable es sensible a minúsculas y mayúsculas.

Los nombres de variables siguen las mismas reglas que otras etiquetas en PHP. Un nombre de variable válido tiene que empezar con una letra o un carácter de subrayado (underscore), seguido de cualquier número de letras, números y caracteres de subrayado. Como expresión regular se podría expresar como: '[a-zA-Z\_\x7f-\xff][a-zA-Z0-9\_\x7f-\xff]\*'

## Enteros (Integer)

Los Integer pueden especificarse mediante notación decimal (base 10), hexadecimal (base 16), octal (base 8) o binaria (base 2), opcionalmente precedidos por un signo (- o +).

Los literales de tipo Integer binarios están disponibles desde PHP 5.4.0.

Para utilizar la notación octal, se antepone al número un 0 (cero). Para utilizar la notación hexadecimal, se antepone al número 0x. Para utilizar la notación binaria, se antepone al número 0b.

## Numero de coma flotante (double)

Formalmente a partir de PHP 7.4.0 (anteriormente, no se permitían los guiones bajos):

LNUM [0-9]+(\_ [0-9]+) \*

DNUM ([0-9]\*(\_ [0-9]+) \*[\.]{LNUM}) | ({LNUM}[\.][0-9]\*(\_[0-9]+)\*)

EXPONENT\_DNUM (({LNUM} | {DNUM}) [eE] [+-]? {LNUM})

El tamaño de un 'float' depende de la plataforma, aunque un valor común consiste en un máximo de aproximadamente 1.8e308 con una precisión cercana a los 14 dígitos decimales (el formato de 64 bit del IEEE).

## Cadena de caracteres

Un String, o cadena, es una serie de caracteres donde cada carácter es lo mismo que un byte. Esto significa que PHP solo admite un conjunto de 256 caracteres, y de ahí que no ofrezca soporte nativo para Unicode. Véanse los [detalles del tipo String](https://www.php.net/manual/es/language.types.string.php#language.types.string.details).

## Valores binarios (Boolean)

Una variable de tipo boolean solo puede contener dos valores: Verdadero o Falso. Se suelen utilizar para comprobar si se cumple una condición en el programa.

## Valores nulos (NULL)

Es un tipo especial de datos que solo contiene el valor NULL. Aunque se suele utilizar por norma en mayúscula, PHP lo acepta escrito de diversas formas: NULL, null, NuLl.

## Arrays

Un array en PHP es en realidad un mapa ordenado. Un mapa es un tipo de datos que asocia valores con claves. Este tipo se optimiza para varios usos diferentes; se puede emplear como un array, lista (vector), tabla asociativa (tabla hash - una implementación de un mapa), diccionario, colección, pila, cola, y posiblemente más. Ya que los valores de un array pueden ser otros arrays, también son posibles árboles y arrays multidimensionales.

Una explicación sobre tales estructuras de datos está fuera del alcance de este manual, aunque se proporciona al menos un ejemplo de cada uno de ellos. Para más información, consulte la extensa literatura que existe sobre este amplio tema.

### In\_array ()

Busca dentro de un array un valor pasado como parámetro, y si lo encuentra devuelve el valor TRUE, si no, devuelve FALSE. Toma dos argumentos, el valor a buscar y el array donde buscar.

### List ()

Asigna a las variables indicadas como parámetros los valores del array al que se iguala.

## Variables

En PHP las variables se representan con un signo de dólar seguido por el nombre de la variable. El nombre de la variable es sensible a minúsculas y mayúsculas.

Los nombres de variables siguen las mismas reglas que otras etiquetas en PHP. Un nombre de variable válido tiene que empezar con una letra o un carácter de subrayado (underscore), seguido de cualquier número de letras, números y caracteres de subrayado. Como expresión regular se podría expresar como: '[a-zA-Z\_\x7f-\xff][a-zA-Z0-9\_\x7f-\xff]\*'

### Funciones de variables

#### Isset () y unset ()

La función Isset () nos indica si la variable existe y para eliminarla de la memoria necesitamos utilizar unset ().

#### Is\_integer (), is\_double (), is\_string ()

Estas funciones devuelven TRUE si la variable pasada coincide con el tipo que indica la función.

#### Intval (), doubleval (), strval ()

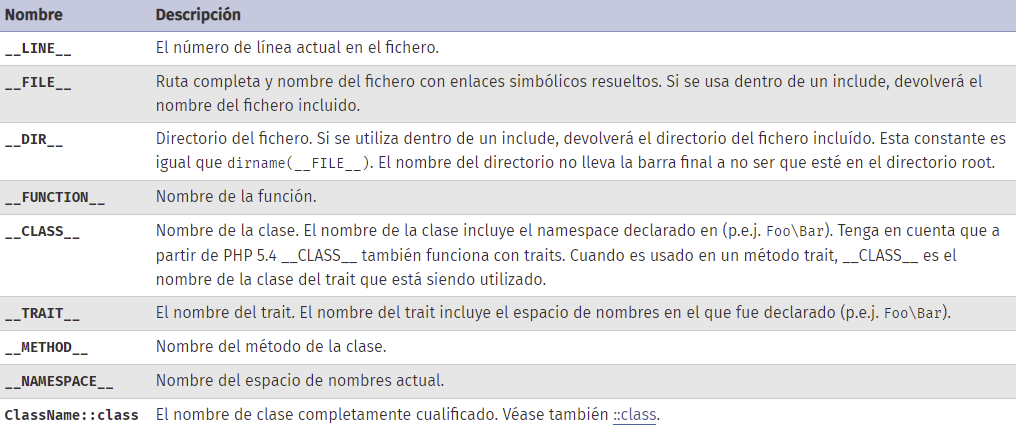
Convierte el valor de una variable al tipo indicado en la función. Estas funciones nos permiten la conversión a los tipos Object o array.

## Constante

Una constante es un identificador (nombre) para un valor simple. Como el nombre sugiere, este valor no puede variar durante la ejecución del script (a excepción de las [constantes mágicas](https://www.php.net/manual/es/language.constants.predefined.php), que en realidad no son constantes). Por defecto, una constante distingue mayúsculas y minúsculas. Por convención, los identificadores de constantes siempre se declaran en mayúsculas.

El nombre de una constante sigue las mismas reglas que cualquier otra etiqueta de PHP. Un nombre de constante válido empieza por una letra o guion bajo, seguido por cualquier número de letras, números o guiones bajos. Usando una expresión regular, se representaría de la siguiente manera: [a-zA-Z\_\x7f-\xff][a-zA-Z0-9\_\x7f-\xff]\*

### Constantes predefinidas



### Defined ()

La función defined () comprueba si una constante está definida o no.

## Operadores

Un operador es algo que toma uno más valores (o expresiones, en la jerga de programación) y produce otro valor (de modo que la construcción en si misma se convierte en una expresión).

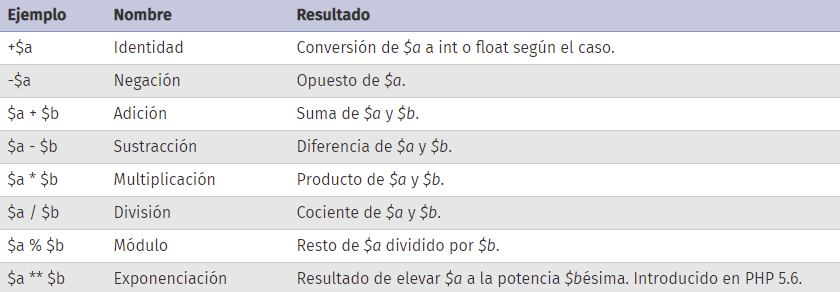
Los operadores se pueden agrupar de acuerdo con el número de valores que toman. ¡Los operadores unarios toman sólo un valor, por ejemplo! (el [operador lógico de negación](https://www.php.net/manual/es/language.operators.logical.php)) o ++ (el [operador de incremento](https://www.php.net/manual/es/language.operators.increment.php)). Los operadores binarios toman dos valores, como los familiares [operadores aritméticos](https://www.php.net/manual/es/language.operators.arithmetic.php) + (suma) y - (resta), y la mayoría de los operadores de PHP entran en esta categoría. Finalmente, hay sólo un [operador ternario](https://www.php.net/manual/es/language.operators.comparison.php#language.operators.comparison.ternary),? :, el cual toma tres valores; usualmente a este se le refiere simplemente como "el operador ternario" (aunque podría tal vez llamarse más correctamente como el operador condicional).

Una lista completa de operadores de PHP sigue en la sección [Precedencia de Operadores](https://www.php.net/manual/es/language.operators.precedence.php). La sección también explica la precedencia y asociatividad de los operadores, las cuales gobiernan exactamente cómo son evaluadas expresiones que contienen varios diferentes operadores.

### Operadores aritméticos

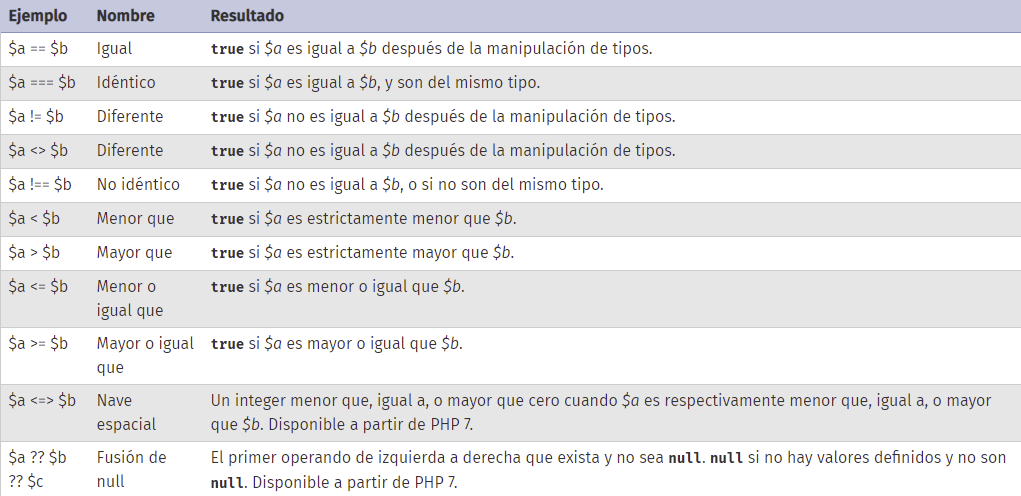
El operador de división ("/") devuelve un valor flotante a menos que los dos operandos sean integers (o String que se conviertan a integers) y los números sean divisibles, en cuyo caso será devuelto un valor integer.

Los operandos del módulo se convierten en integers (por extracción de la parte decimal) antes del procesamiento.



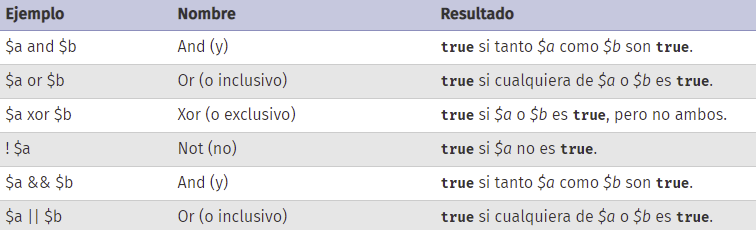
### Operadores de comparación

Los operadores de comparación, como su nombre lo indica, permiten comparar dos valores. Puede también estar interesado en ver [las tablas de comparación de tipos](https://www.php.net/manual/es/types.comparisons.php), ya que muestran ejemplos de las varias comparaciones relacionadas con tipos. Si se compara un número con un String o la comparación implica String numéricos, entonces cada String es [convertido en un número](https://www.php.net/manual/es/language.types.string.php#language.types.string.conversion) y la comparación realizada numéricamente. Estas reglas también se aplican a la sentencia [switch](https://www.php.net/manual/es/control-structures.switch.php). ¡La conversión de tipo no tiene lugar cuando la comparación es === o! == ya que esto involucra comparar el tipo, así como el valor.



### Operadores lógicos

La razón para tener las dos variaciones diferentes de los operadores "and" y "or" es que ellos operan con precedencias diferentes.

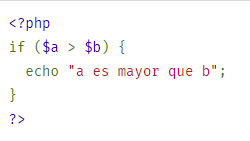


## Estructuras de control

Todo script PHP está construido según una serie de sentencias. Una sentencia puede ser una asignación, una llamada de función, un ciclo, una sentencia condicional o incluso una sentencia que no hace nada (una sentencia vacía). Las sentencias generalmente finalizan con un punto y coma. Adicionalmente, las sentencias pueden agruparse en un conjunto de sentencias, encapsulándolas entre corchetes.

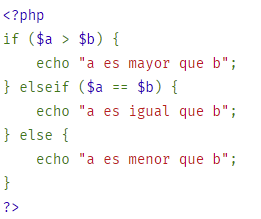
### If

El constructor if es una de las características más importantes de muchos lenguajes, incluido PHP. Permite la ejecución condicional de fragmentos de código.



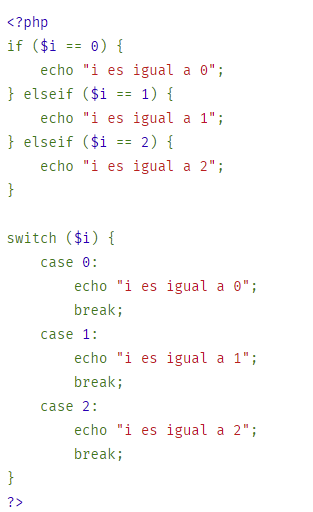
### Else if

elseif, como su nombre lo sugiere, es una combinación de if y else. Del mismo modo que else, extiende una sentencia if para ejecutar una sentencia diferente en caso de que la expresión if original se evalúe como false. Sin embargo, a diferencia de else, esa expresión alternativa sólo se ejecutará si la expresión condicional del elseif se evalúa como true.



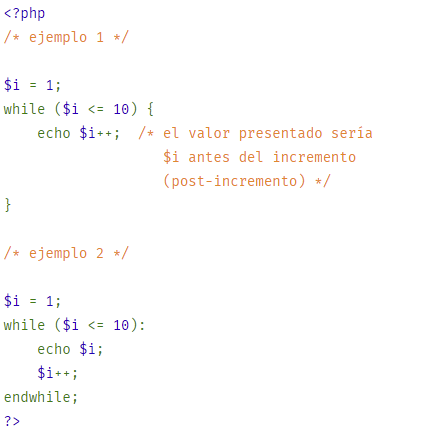
### Switch

La sentencia switch es similar a una serie de sentencias IF en la misma expresión. En muchas ocasiones, es posible que se quiera comparar la misma variable (o expresión) con muchos valores diferentes, y ejecutar una parte de código distinta dependiendo de a qué valor es igual. Para esto es exactamente la expresión switch.

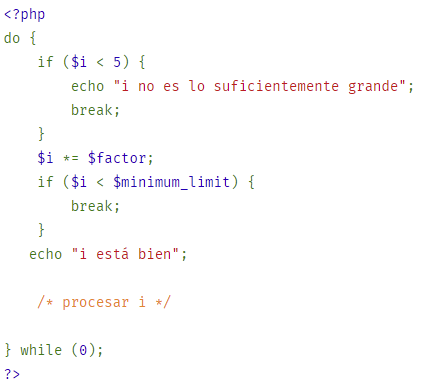


### While

Los bucles while son el tipo más sencillo de bucle en PHP. Se comportan igual que su contrapartida en C. El significado de una sentencia while es simple. Le dice a PHP que ejecute las sentencias anidadas, tanto como la expresión while se evalúe como true. El valor de la expresión es verificado cada vez al inicio del bucle, por lo que incluso si este valor cambia durante la ejecución de las sentencias anidadas, la ejecución no se detendrá hasta el final de la iteración (cada vez que PHP ejecuta las sentencias contenidas en el bucle es una iteración). A veces, si la expresión while se evalúa como false desde el principio, las sentencias anidadas no se ejecutarán ni siquiera una vez.

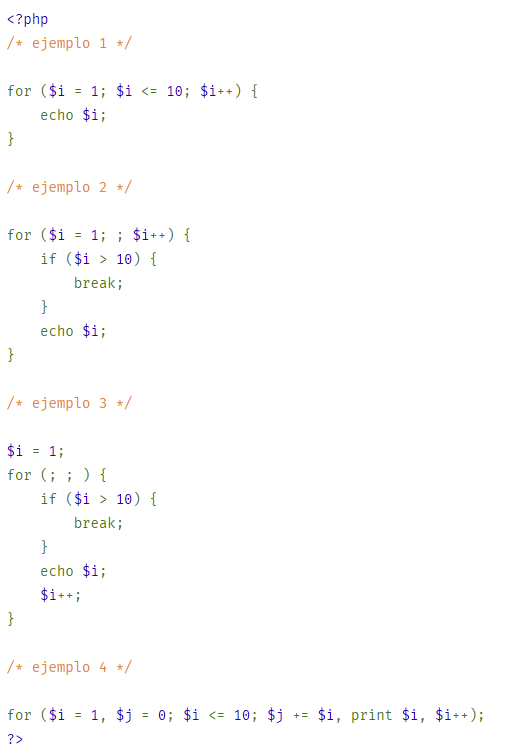


### Do-While

Los bucles do-while son muy similares a los bucles while, excepto que la expresión verdadera es verificada al final de cada iteración en lugar que al principio. La diferencia principal con los bucles while es que está garantizado que corra la primera iteración de un bucle do-while (la expresión verdadera sólo es verificada al final de la iteración), mientras que no necesariamente va a correr con un bucle while regular (la expresión verdadera es verificada al principio de cada iteración, si se evalúa como false justo desde el comienzo, la ejecución del bucle terminaría inmediatamente). 

### For

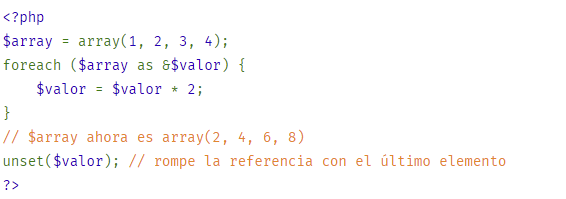
Los bucles for son los más complejos en PHP. Se comportan como sus homólogos en C. La primera expresión (expr1) es evaluada (ejecutada) una vez incondicionalmente al comienzo del bucle. En el comienzo de cada iteración, se evalúa expr2. Si se evalúa como true, el bucle continúa y se ejecutan la/s sentencia/s anidada/s. Si se evalúa como false, finaliza la ejecución del bucle. Al final de cada iteración, se evalúa (ejecuta) expr3.Cada una de las expresiones puede estar vacía o contener múltiples expresiones separadas por comas. En expr2, todas las expresiones separadas por una coma son evaluadas, pero el resultado se toma de la última parte. Que expr2 esté vacía significa que el bucle debería ser corrido indefinidamente (PHP implícitamente lo considera como true, como en C). Esto puede no ser tan inútil como se pudiera pensar, ya que muchas veces se debe terminar el bucle usando una sentencia condicional [break](https://www.php.net/manual/es/control-structures.break.php) en lugar de utilizar la expresión verdadera del for.



### ForEach

El constructor foreach proporciona un modo sencillo de iterar sobre arrays. foreach funciona sólo sobre arrays y objetos, y emitirá un error al intentar usarlo con una variable de un tipo diferente de datos o una variable no inicializada. La primera forma recorre el array dado por expresión\_array. En cada iteración, el valor del elemento actual se asigna a $valor y el puntero interno del array avanza una posición (así en la próxima iteración se estará observando el siguiente elemento).

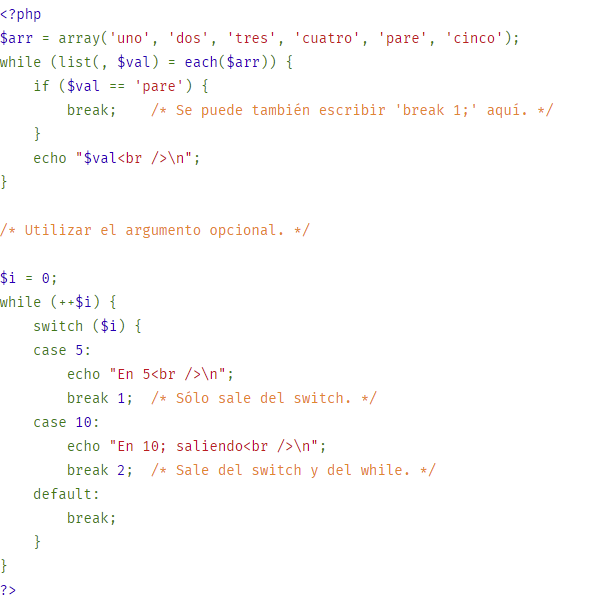
La segunda forma además asigna la clave del elemento actual a la variable $clave en cada iteración.



### Break

break finaliza la ejecución de la estructura for, foreach, while, do-while o switch en curso.

break acepta un argumento numérico opcional que indica de cuántas estructuras anidadas circundantes se debe salir. El valor predeterminado es 1, es decir, solamente se sale de la estructura circundante inmediata.



### Continue

continue se utiliza dentro de las estructuras iterativas para saltar el resto de la iteración actual del bucle y continuar la ejecución en la evaluación de la condición, para luego comenzar la siguiente iteración. continue acepta un argumento numérico opcional, que indica a cuántos niveles de bucles encerrados se ha de saltar al final. El valor por omisión es 1, por lo que salta al final del bucle actual.

