



Introducción al lenguaje de programación

PHP

<https://www.php.net>



Juan Vladimir
@juanvladimir13

Agenda

1. Que es PHP
2. Comandos de PHP
3. Tipos de dato
4. Variables y constantes
5. Funciones
6. Parámetros y valor de retorno
7. Operadores aritméticos, lógicos y comparación
8. Estructuras de control
9. Operadores de asignación avanzado

PHP

PHP, acrónimo de "PHP: Hypertext Preprocessor" de código abierto que está especialmente pensado para el ***desarrollo web***.

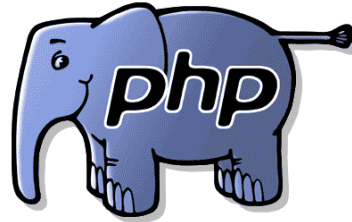
Etiqueta de apertura y cierre

`<?php ?>`

Shortcut etiqueta de apertura y cierre

Imprime el resultado de una *sentencia de una sola linea*

`<?= ?>`



Características de PHP

- ✓ De propósito general
- ✓ Es un lenguaje de '*scripting*'
- ✓ Puede ser embebido(incrustado) en páginas HTML
- ✓ Scripts del lado del servidor
- ✓ Scripts desde la línea de comandos
- ✓ Escribir aplicaciones de escritorio
- ✓ Escribir aplicaciones móviles

Crear archivo de texto plano PHP

Ingresar al ***CMD*** o ***Terminator***

Crear un ***archivo de texto plano*** con la extensión ****.php***

Ejecutar un archivo PHP en un Servidor Web

Ejecutar el comando `php -S localhost:8080 -t .`

Abrir un ***navegador web***

Ingresar al enlace **`localhost:8080`**

Ejecucion de un archivo script PHP

Ejecutar el comando `php index.php`

Tipos de datos

Números enteros (positivos, negativos)

temperatura = -13

anio = 2025

Números con decimales (positivos, negativos)

peso = 60.48

latitud = -18.755121545

Texto / Cadenas / Caracteres / String

nombres = 'Juan Vladimir'

carnet = '12345678'

Booleanos

perteneceAlCurso = true

esMayorDeEdad = false

Variables, constantes y asignación

Variable

```
$edad = 36;           // variable de tipo de dato numero  
$nombre = 'juanvladimir13'; // variable de tipo de dato string  
$isDeveloper = false; // variable de tipo de dato boolean
```

Constante

```
const FECHA_NACIMIENTO = '26-12-1987';  
const CANTIDAD_BOMBAS = 20;
```

Asignación

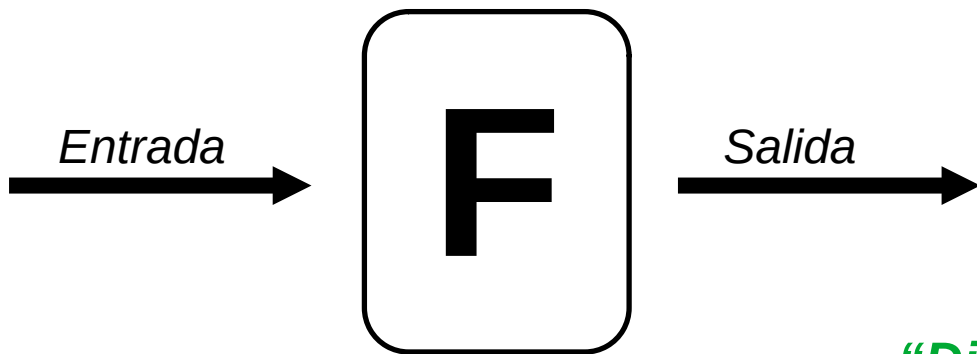
```
$edad = 36;  
$nombre = 'Ing. Juan Vladimir';  
$isDeveloper = true;
```

Función

Es un **bloque de código** reutilizable que realiza una tarea específica.

Las funciones permiten **estructurar el código en partes más pequeñas y manejables**, lo que facilita su **mantenimiento y reutilización**.

Un **algoritmo** se resuelve con *una o muchas funciones*



“Divide y vencerás”

Partes de una Función

Nombre: Se usa para "llamarla" o invocarla. Este nombre debe ser descriptivo para indicar qué hace la función.

Parámetros: Son **cero, uno o más valores de entrada**, que la función utiliza para realizar su tarea.

Cuerpo: Es el conjunto de instrucciones que se ejecutan cuando la función es llamada.

Valor de retorno: Valor como resultado de la tarea. Este valor puede ser utilizado por el código que llamó a la función.

Llamada a la función: Para usar una función, se "llama" por su nombre seguido de paréntesis. Si la función requiere parámetros, estos se colocan dentro de los paréntesis.

```
function calcularEdadDeEstudiante( string $nombre, int $anioNac ):int {  
    // Cuerpo (sentencias)  
    return 0;  
}  
$edad = calcularEdadDeEstudiante( 'juanvladimir', 2009 );
```

Parámetros y valor de retorno

Números

```
function sumar(int $numX , int $numY): int { return 0; }
```

```
function promediar(float $numX , float $numY): float { return 0.1; }
```

Cadenas

```
function saludar(string $usuario ): string { return 'Hi'; }
```

Booleanos

```
function hasGirlFriend (bool $state) : bool { return true; }
```

Tipo de dato de retorno void

```
function destruirElMundo(int $cantidadBombas): void { }
```

Operadores Aritméticos

`$sueldo = 3200;`

Suma	+	<code>\$sueldo = \$sueldo + 5;</code>
Resta	-	<code>\$sueldo = \$sueldo - 5;</code>
Multiplicación	*	<code>\$sueldo = \$sueldo * 5;</code>
División	/	<code>\$sueldo = \$sueldo / 3;</code>
Residuo	%	<code>\$sueldo = \$sueldo % 3;</code>
Exponente	**	<code>\$sueldo = \$sueldo ** 3;</code>
Incremento	++	<code>\$sueldo++ ;</code>
Decremento	--	<code>\$sueldo--;</code>

Operadores Lógicos

Conjunción: AND &&

Disyunción: OR ||

Negación: NOT !

`$seComportanBien = true;`

`$hacenTareas = true;`

`$hayChurrasco = false;`

`hayChurrasco = seComportanBien && hacenTareas;`

`hayChurrasco = seComportanBien || hacenTareas;`

`hayChurrasco = !hayChurrasco;`

Operadores de comparación

```
$sueldoGerente = 3200;  
$sueldoEmpleado = 5500;
```

Igual	==	<code>\$sueldoGerente == \$sueldoEmpleado;</code>
Diferente	!=	<code>\$sueldoGerente != \$sueldoEmpleado;</code>
Igual estricto	===	<code>\$sueldoGerente === \$sueldoEmpleado;</code>
Diferente estricto	!==	<code>\$sueldoGerente !== \$sueldoEmpleado;</code>
Mayor que	>	<code>\$sueldoGerente > \$sueldoEmpleado;</code>
Menor que	<	<code>\$sueldoGerente < \$sueldoEmpleado;</code>
Mayor igual	>=	<code>\$sueldoGerente >= \$sueldoEmpleado;</code>
Menor igual	<=	<code>\$sueldoGerente <= \$sueldoEmpleado;</code>

El resultado de una **comparación** siempre es una **proposición** *false* o *true*

Expresión

VAR : variable

FUN : función

OPE : operador/conector

CON : constante

Expresión unaria

OPE { **VAR** | **FUN** | **CON** }

Ejemplos:

\$a = ~\$hayChurrasco;

\$res = calcularSuma(8, 5);

\$curso = 'Quinto D';

Expresiones

{ **VAR** | **CON** | **FUN** } **OPE** { **VAR** | **CON** | **FUN** }

Ejemplos:

\$x = 8 + 5;

\$y = \$sueldoGerente > \$sueldoEmpleado;

\$z = calcularSuma(5,5) * 0.3;

\$a = \$haceTareas && \$esLinda;

\$b = haceTareas && (\$sueldo > 3000);

\$c = (\$sueldo + 1500) - \$gastos;

Sentencia

Es una **línea de código** que llevan a cabo una tarea. Las sentencias están formadas por un **conjunto de expresiones** que permiten realizar una acción determinada.

```
$seComportanBien = true;
```

```
$hacenTareas = false;
```

```
$hayChurrasco = false;
```

```
$sueldo = 3200;
```

```
$sueldoGerente = 5000;
```

```
$hayChurrasco = $seComportanBien && $hacenTareas;
```

```
$y = $sueldoGerente > $sueldoEmpleado;
```

```
$z = calcularSuma(5,5) * 0.3;
```

Estructuras de control

Estructura de control condicional (if else)

Ejecuta una **sentencia si una condición específica es evaluada como verdadera**. Si la condición es evaluada como **falsa**, otra sentencia puede ser ejecutada.

```
$sueldo = 3200;
```

```
$hacenTareas = true;
```

Sintaxis

```
if ( $sueldo > 3200 ) {  
    // sentencias o bloque de código  
}
```

```
if ( $hacenTareas == false ) {  
    // sentencias o bloque de código  
}
```

```
if ( $sueldo > 3200 && $hacenTareas ) {  
    // bloque de código por sentencia true  
} else {  
    // bloque de código por sentencia false  
}
```

```
If ( ( $sueldo - 1000 ) > 3200 ) {  
    // sentencias o bloque de código  
}
```

Estructuras de control

Estructura de control repetitiva (while)

Ejecuta una **sentencia especificada mientras cierta condición se evalúe como verdadera**. Dicha condición es evaluada antes de ejecutar la sentencia.

```
$n = 0;
```

```
$estaLloviendo = true;
```

Sintaxis

```
$n = 0;
```

```
while ($n > 13 ) {  
    // sentencias  
    $n++;  
}
```

```
while ( $estaLloviendo ) {  
    // sentencias  
}
```

Estructuras de control

Estructura de control repetitiva (for)

Expresión inicial: Declaración de variable y asignación

Condición: Expresión para ser evaluada antes de cada iteración del bucle. Si esta expresión se evalúa como verdadera, se ejecuta sentencia, si es falsa, la ejecución salta a la siguiente expresión.

Expresión final: Expresión evaluada al final de cada iteración del bucle. Esto ocurre antes de la siguiente evaluación de la condición. Generalmente se usa para actualizar o incrementar la variable contador.

Sintaxis

```
for ($index=0; $index < 13; $index++) {  
    // sentencias  
}
```

Operadores de Asignación Avanzados

```
$sueldo = 3200;
```

```
$hayChurrasco = false;
```

Suma	+=	<code>\$sueldo += 5;</code>
Resta	-=	<code>\$sueldo -= 5;</code>
Multiplicación	*=	<code>\$sueldo *= 5;</code>
División	/=	<code>\$sueldo /= 3;</code>
Residuo	%=	<code>\$sueldo %= 3;</code>
Exponente	**=	<code>\$sueldo **= 3;</code>
AND	&&=	<code>\$hayChurrasco &&= \$sueldo > 3000;</code>
OR	 =	<code>\$hayChurrasco = \$sueldo > 3000;</code>

Tipos de dato array

Array de números

```
$notasDeExamen = [ 51, 80, 33 ];
```

```
function sumar(array $numeros): array { }
```

Array de cadenas

```
$colores = ['rojo' , 'amarillo', 'verde'];
```

```
function saludarUsuarios(array $estudiantes): array { }
```

Array de Arrays

```
$notas = [ [ 50, 70, 80] , [ 84, 51, 100], [ 90, 0, 87] ];
```

Array asociativos

```
$mejoresNotas = [ 'primero' => 100, 'segundo' => 80, 'tercero'=> 73 ];
```

```
$notas = [ [ 'rojo' => 100, 'verde' => 852], [ 'rojo' => 10, 'verde' => 82] ];
```