

## PHP puro

Lenguaje PHP .....	3
¿Que necesito para ejecutar un archivo? .php .....	4
Instalación y configuración del servidor local LARAGON:.....	5
Instalación de php my admin .....	6
Aquí podemos observar la versión de php que nuestro servidor está ejecutando.....	9
Crear un archivo .php y hacer un hola mundo .....	10
Incrustar o combinar código html con código php:.....	12
Comentarios en php:.....	12
Tipos de datos en PHP .....	13
Ejemplos de tipos de datos en código:.....	13
¿Cómo saber que tipo de dato es?:.....	14
Ejemplo de declarar, inicializar y mostrar una variable: .....	16
Estándares de nomenclatura:.....	17
Nombres de variables que no podemos utilizar porque ya están predefinidas para algo. ....	18
Constantes en php: .....	18
Ejemplo en código de declarar una variable (manera antigua) con define:.....	19
Nombres de constantes que no debemos de ocupar: .....	20
Arrays o arreglos en php a partir de la versión 7: .....	21
Tipos de arrays .....	21
Ejemplo de arreglos escalares: .....	22
Ejemplo de arreglos asociativos: .....	23
Ejemplo de arreglos multidimensionales: .....	24
Contar elementos almacenados en un array:.....	25
Ver información de tu php: .....	27
Como concatenar string o cadenas de texto: .....	27
Operadores aritméticos o matemáticos: .....	28
Shortcuts u operadores de asignación: .....	30
Asignación de valor por referencia a una variable:.....	31
Operadores lógicos (and or not etc) .....	33
Operadores de incremento y decremento .....	35
Estructura condicionar simple (IF) .....	36

IF doble (IF else) .....	39
Operador ternario (IF de una sola línea) .....	41
IF multiple (IF-ELSEIF-ELSE).....	42
If anidados (if dentro de otro if) .....	43
Switch y Match (se parecen un poco).....	44
Ciclos en php, bucles o en ingles loops .....	46
Ciclo while .....	46
Ciclo DO-WHILE .....	48
Ciclo FOR .....	49
Ciclo Foreach .....	50
Como detener un ciclo: .....	54
INCLUDE Y REQUIERE para incluir archivos con codigo a otro archivo .....	57
Funciones propias .....	60
Como incluir y llamar una función desde otro archivo php: .....	61
Funciones predefinidas de PHP para Convertir una cadena de texto String a MAYUSCULAS o minúsculas: .....	63
Funciones predefinidas de PHP para contar caracteres o palabras de un String: .....	65
Como convertir un String en un Array en php: .....	66
Funciones matemáticas en php:.....	70
Formatear números con un formato específico en php (cantidad de decimales, separadores de decimales, separadores de miles): .....	73
Obtener fecha y hora actual de cualquier lugar con la función date: .....	75
Obtener fecha y hora actual (EN ESPAÑOL) de cualquier lugar con la función date: .....	78
Como encriptar una contraseña o clave en php con hash (modos antiguos y no tan recomendables): .....	79
Como encriptar una contraseña o clave en php con hash (la manera más recomendable): .....	83
Como enviar formularios con el método get y post en php: .....	85
DIFERENCIA entre el método GET y POST .....	89
Select y Checkbox múltiples en formularios php para mandar multiples datos: .....	90
Como saber si una variable esta vacía o definida):.....	93
Como subir o enviar archivos a servidor php con formularios: .....	96
Como limitar el tipo de archivo a subir o enviar al servidor php con formularios: .....	101
Como limitar el peso del archivo que queremos subir: .....	103
Ejemplo de manejo de cookies en php (crear y eliminar): .....	104

Manejo de sesiones en php con variables de \$_SESSION:.....	105
Si yo quiero que el nombre y la clave sean diferentes .....	107
Como eliminar una sesión: .....	111
Cookies vs Sesiones:.....	113
Como redireccionar al usuario a otra página usando php:.....	113

## Lenguaje PHP

### ¿Qué es PHP?

PHP es un lenguaje de programación de **código abierto** que permite el **desarrollo web o aplicaciones web dinámicas**, el cual es apto para incrustar el lenguaje HTML



## Contenido del curso

1. Conceptos básicos
2. Estructuras condicionales
3. Ciclos o bucles
4. Envió de datos mediante formularios con PHP
5. Conexión a base de datos MySQL
6. CRUD
7. Manejo de sesiones
8. Y mucho mas...

## Requisitos

1. Un editor de código
2. Servidor local (XAMPP, LARAGON u otro)
3. Navegador web

¿Que necesito para ejecutar un archivo? .php con código php dentro además de un navegador y el editor de texto? Necesitas un entorno de servidor web que pueda interpretar y ejecutar el código PHP. Aquí hay dos opciones comunes:

### 1. Servidor web local:

Puedes instalar un servidor web local en tu computadora. Algunas opciones populares son Apache, Nginx o servidores web integrados como XAMPP, WampServer o **Laragon**. Estos paquetes incluyen Apache, PHP y MySQL (o MariaDB) preconfigurados para ejecutar aplicaciones web en tu computadora.

Después de instalar y configurar tu servidor web local, puedes colocar tu archivo PHP en el directorio raíz del servidor (por lo general, htdocs en XAMPP o WampServer, o www en **Laragon**) y acceder a él a través de tu navegador web usando una URL como [http://localhost/tu\\_archivo.php](http://localhost/tu_archivo.php)

### 2. Hospedaje web:

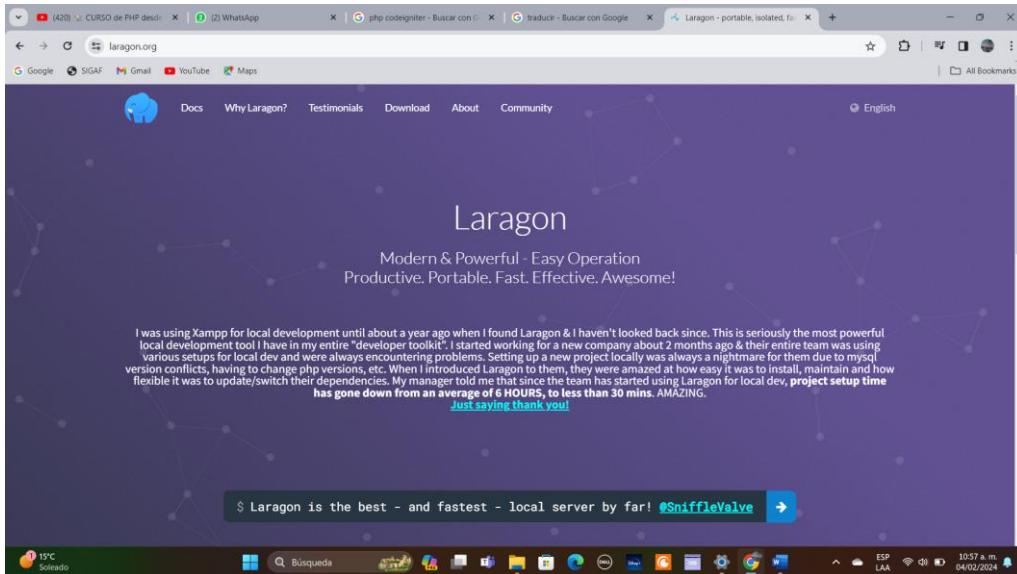
Si tienes acceso a un servidor web remoto (por ejemplo, a través de un servicio de alojamiento web), puedes cargar tu archivo PHP en el servidor y acceder a él a través de tu navegador utilizando la URL del sitio web.

Para subir archivos a un servidor web remoto, generalmente necesitarás un cliente FTP (Protocolo de Transferencia de Archivos) como FileZilla o utilizar las herramientas de administración de archivos proporcionadas por tu proveedor de alojamiento web.

Para este curso todo lo haremos con un servidor web local.

## Instalación y configuración del servidor local LARAGON:

Como editor de código podemos trabajar con Visual Studio Code, Servidor Local ocuparemos Laragon para este curso así que comenzamos con la instalación de LARAGON, primero nos dirigimos a su sitio oficial y damos clic en download



Damos clic aquí para descargar la versión completa:

### Edition

[Download Laragon - Full \(173 MB\)](#)

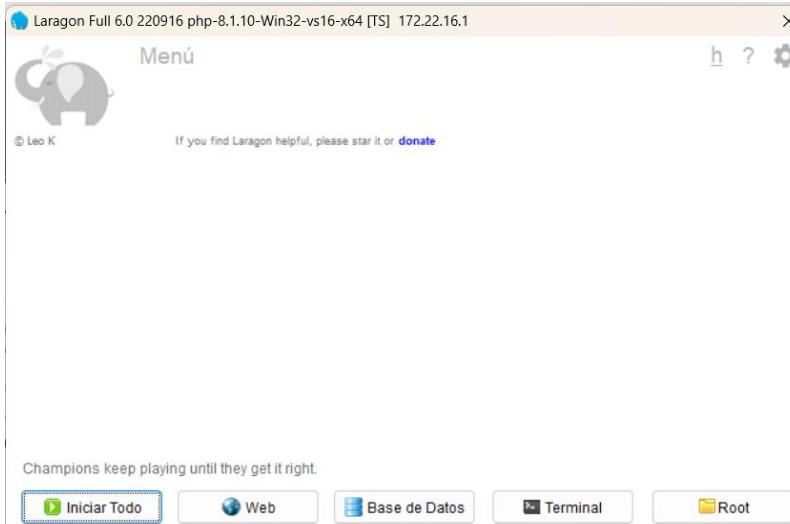
install Rails

og with Rails

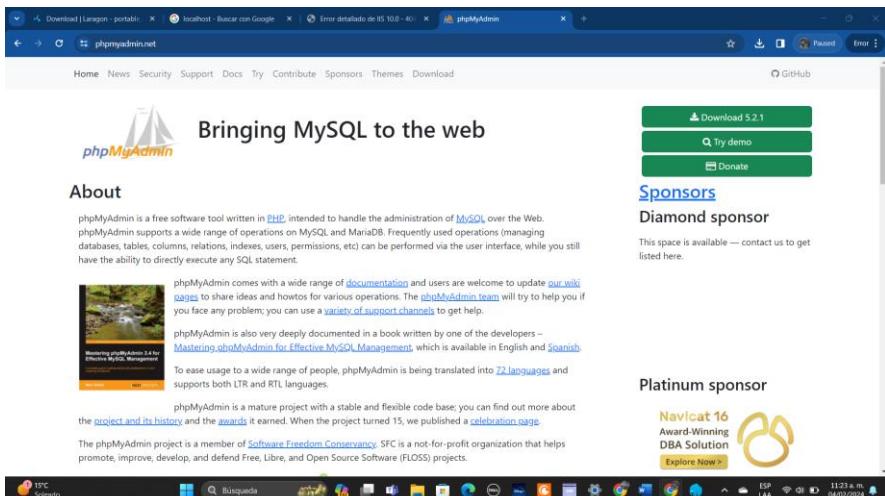
install Django

[Download Laragon - Portable \(20 MB\)](#)

Simplemente damos doble clic al archivo que se descarga, elegimos idioma español, desmarcamos los 3 checkbox que nos aparecen, y finalizamos. Abrimos el servidor y debería mostrarnos una pantalla asi:

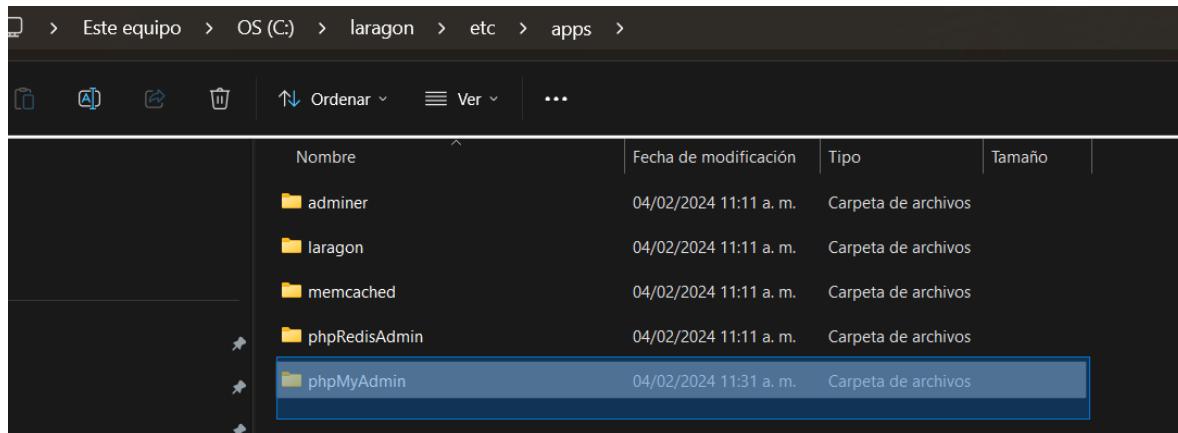


# Instalación de php my admin



The screenshot shows the phpMyAdmin website's "About" page. At the top, there's a logo and the tagline "Bringing MySQL to the web". Below that, there's a section titled "About" with a "Download" button. To the right, there are sections for "Sponsors" (Diamond sponsor) and "Platinum sponsor" (Navicat 16). The page also contains text about the project's documentation, history, and translations.

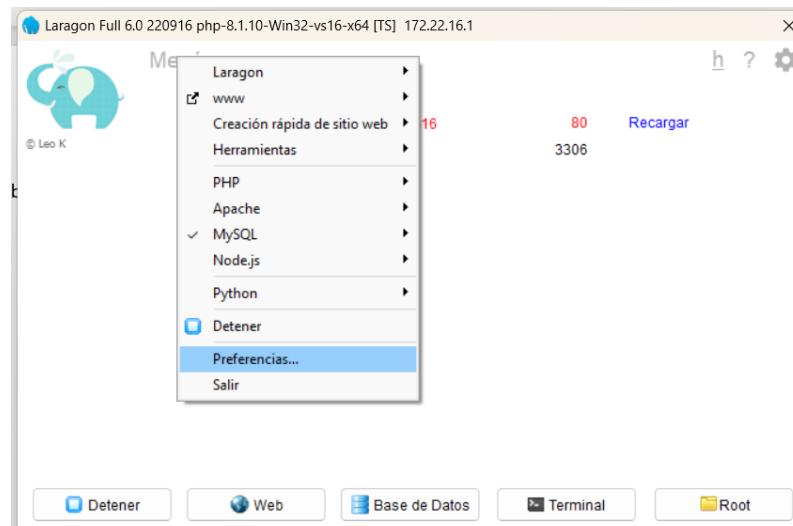
Después extraemos la carpeta del archivo rar y la ponemos en esta dirección:



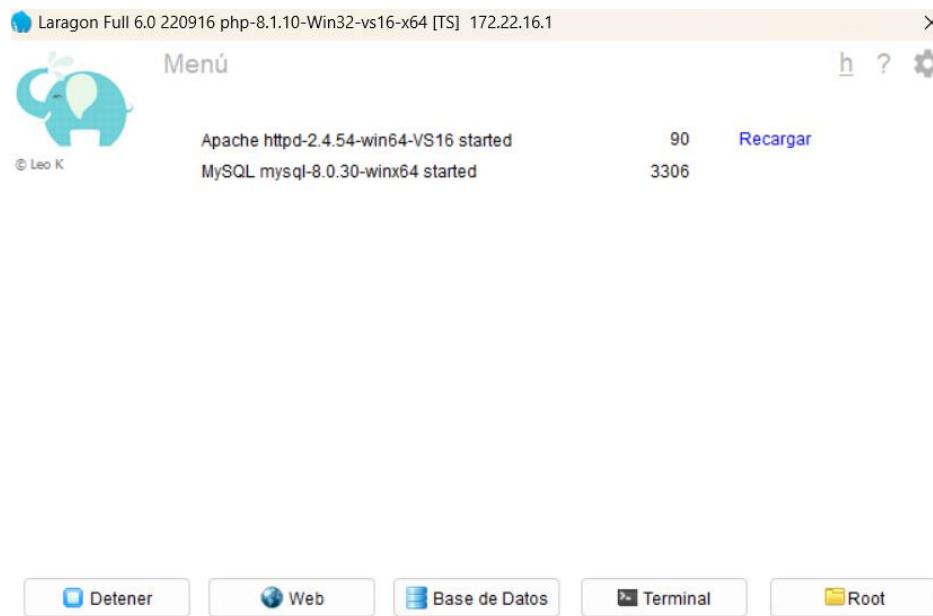
Al tratar de iniciar el servidor me dice que mi puerto 80 el cual es el que se suele ocupar para iniciar un servidor, lo tengo ocupado



Debo dar clic en Menú, después en preferencias

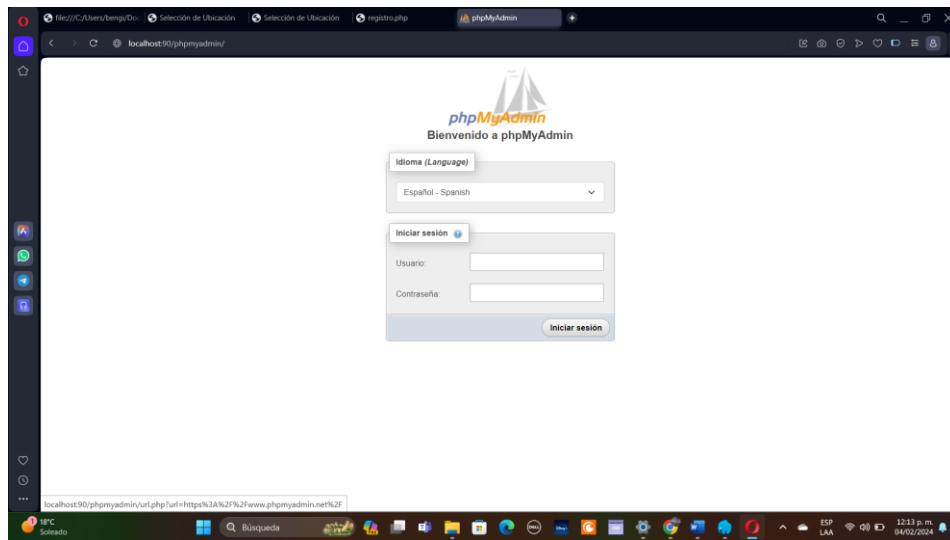


Servicios y puertos, y cambio el puerto a uno que tenga desocupado. En este caso ocupe el puerto 90.

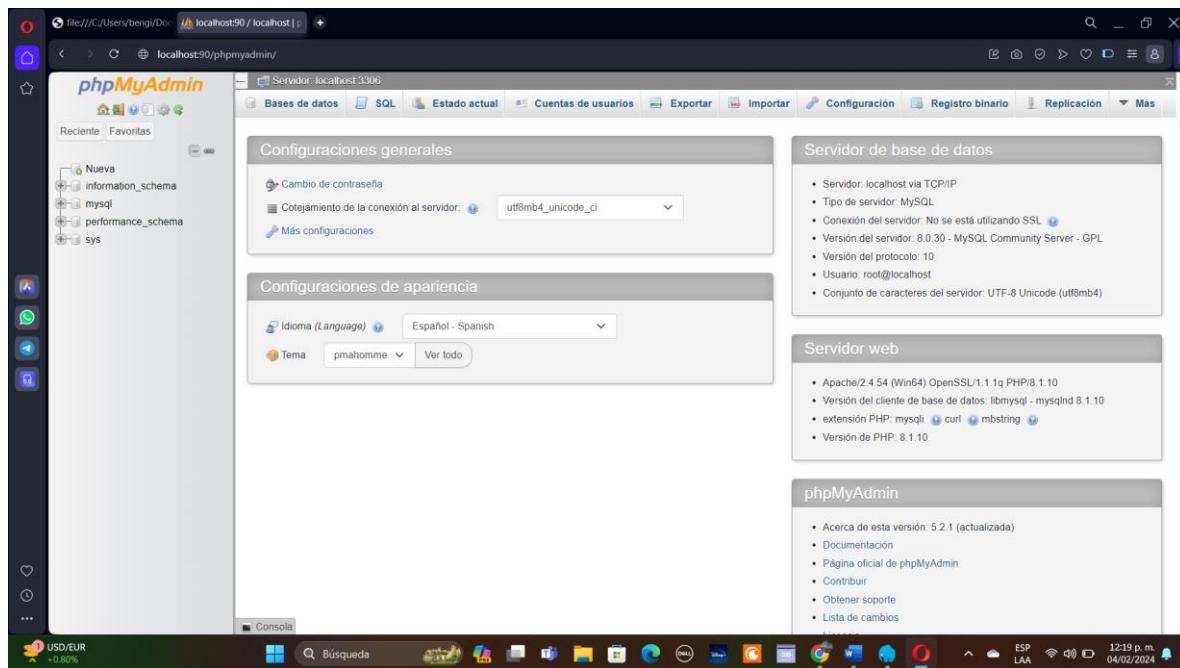


Ahora una vez que ya nos dejó iniciar todo, abriremos un navegador y colocare esta ruta  
<http://localhost:90/phpmyadmin>

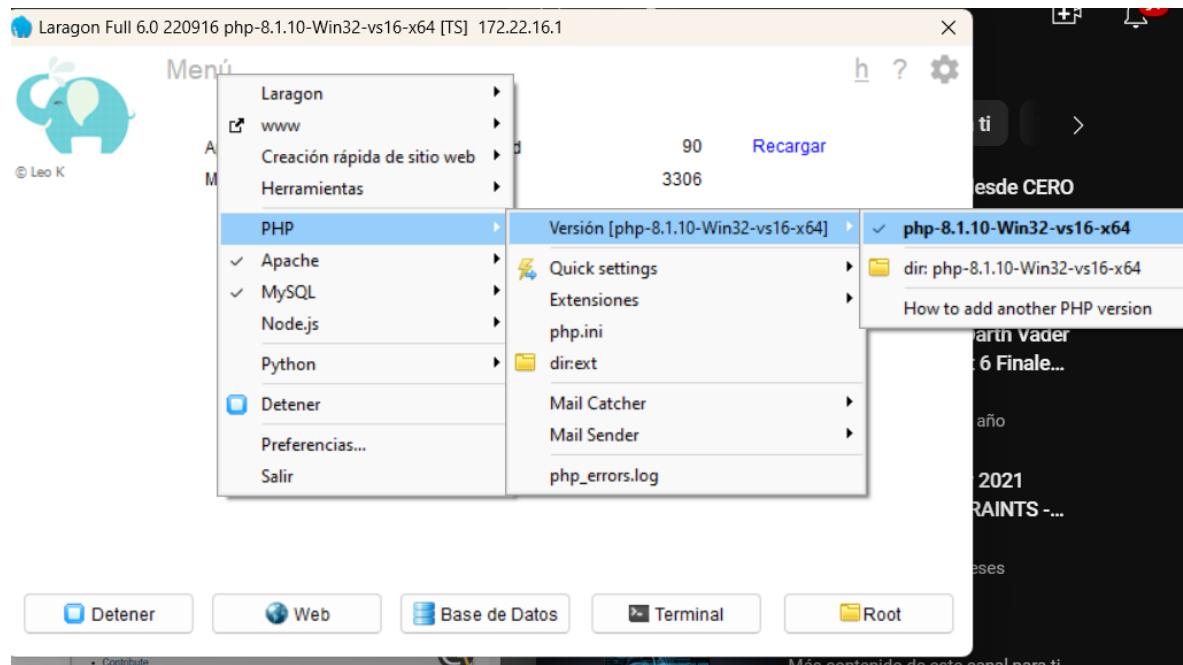
En esa dirección pongo el puerto 90 porque fue el que configure anteriormente para apache. Luego en este login coloco **root** en usuario, y contraseña lo dejo vacío e inicio sesión, si no llega a funcionar detengo laragon, vuelvo a iniciar y refresco esa página.



Al iniciar sesión nos mostrara esa pagina:



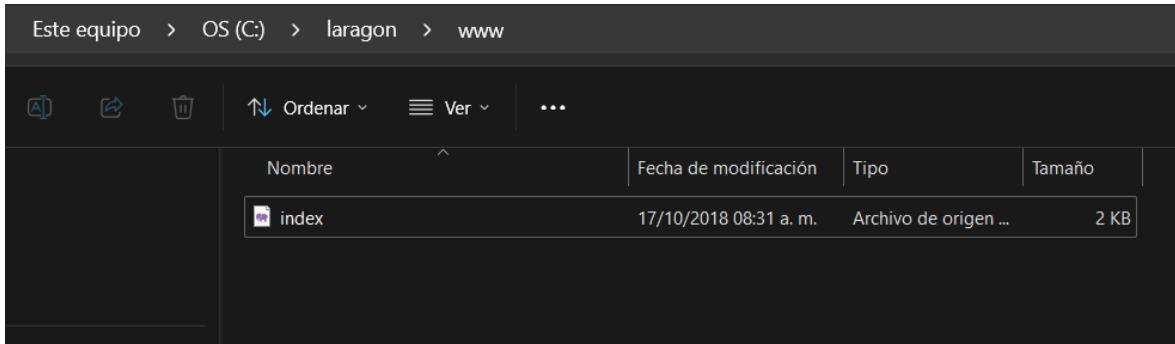
Aquí podemos observar la versión de php que nuestro servidor está ejecutando en el menú de laragon:



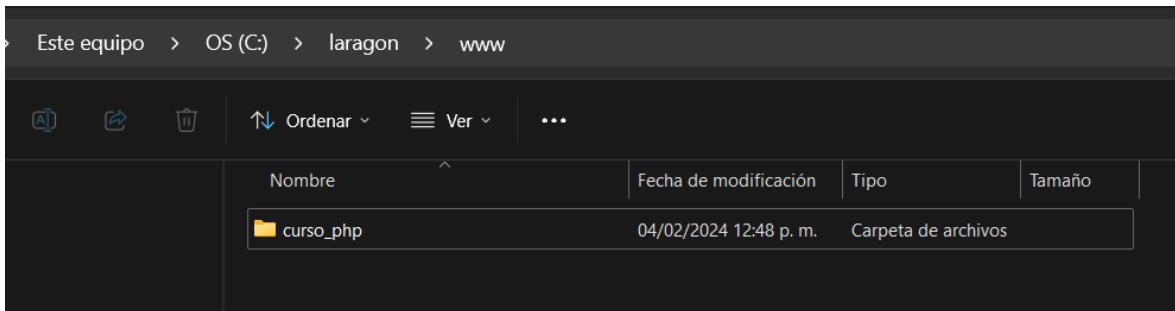
Para actualizar la versión tenemos que ir a la página oficial de laragon, podemos visitar este video [https://youtu.be/YqSB8WSIb2o?si=Apweov\\_D-6TLBSky](https://youtu.be/YqSB8WSIb2o?si=Apweov_D-6TLBSky) Y adelantar al minuto 8:50.

## Crear un archivo .php y hacer un hola mundo

Si yo entro en mi navegador a <http://localhost:90> observare que se está ejecutando el siguiente index que ya viene creado:



Eliminamos este index, y aquí podemos crear una carpeta para un proyecto de una aplicación, en este caso cree la siguiente carpeta para el curso de php:



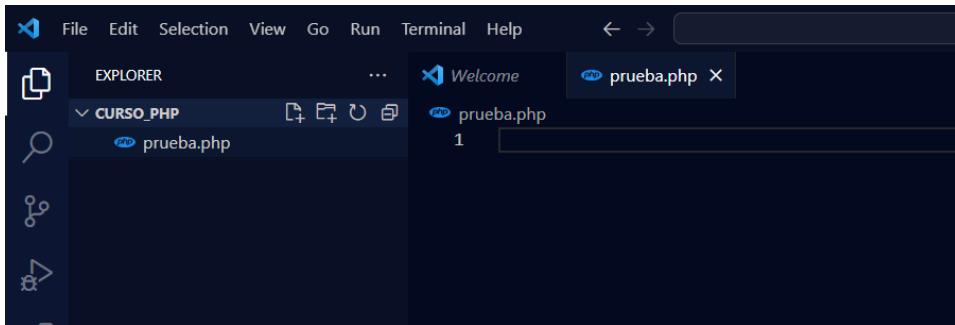
Y al entrar nuevamente a <http://localhost:90> observare las carpetas que yo vaya creando, y puedo entrar a ellas dando doble clic:



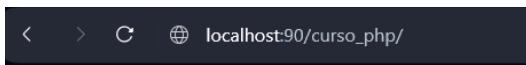
## Index of /

- [curso\\_php/](#)

Ahora para hacer nuestro primer programa, abro visual studio code, abro la carpeta que acabo de crear en la dirección C:\laragon\www de mis documentos, llamada curso\_php y dentro creo un archivo llamado prueba.php



En mi navegador puedo observar que ya aparece igual ese archivo que cree, si le doy doble clic se ejecutara ese archivo en mi navegador



## Index of /curso\_php

- [Parent Directory](#)
- [prueba.php](#)

Mi primer hola mundo

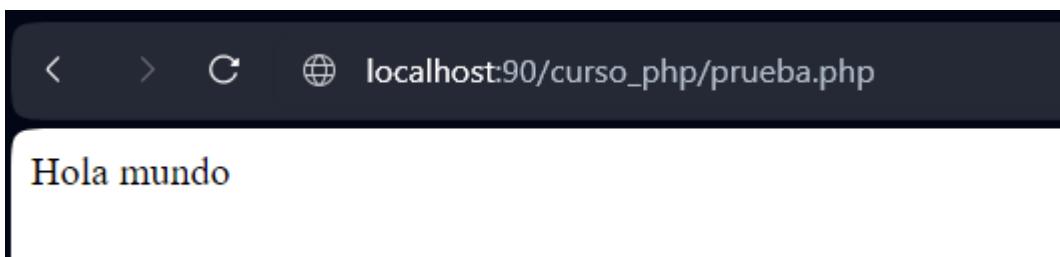
Al escribir código php necesito poner la etiqueta de apertura `<?php` y de salida `?>` y todo lo que escriba adentro es mi código, en este caso un hola mundo.

A screenshot of the Visual Studio Code interface. The editor shows the following PHP code:

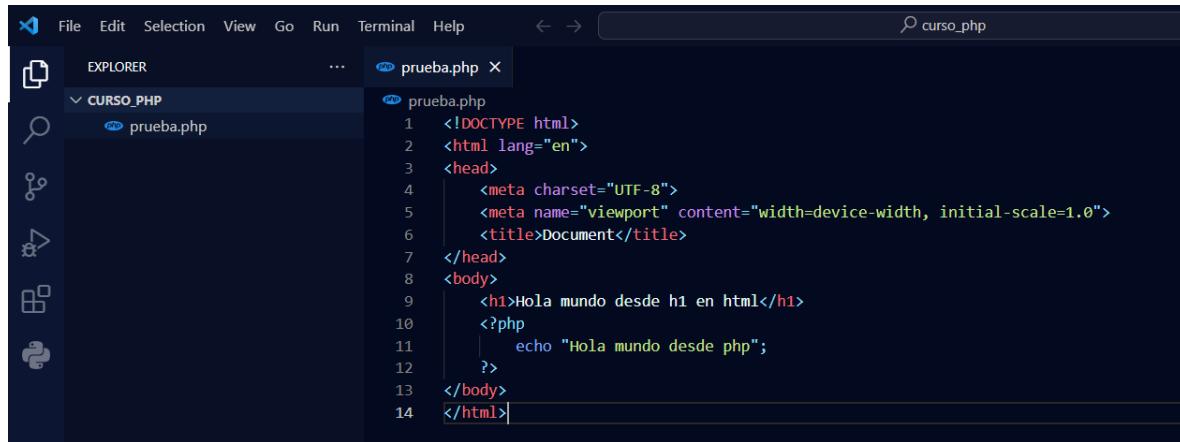
```
1 <?php
2 echo "Hola mundo";
3 ?>
```

The status bar at the bottom indicates the file path as 'C:\laragon\www\curso\_php\prueba.php'.

Y así se ve al abrir <http://localhost:90> y entrar a mi archivo en el navegador o entrar directamente a mi archivo con la ruta del navegador [http://localhost:90/curso\\_php/prueba.php](http://localhost:90/curso_php/prueba.php)



## Incrustar o combinar código html con código php:



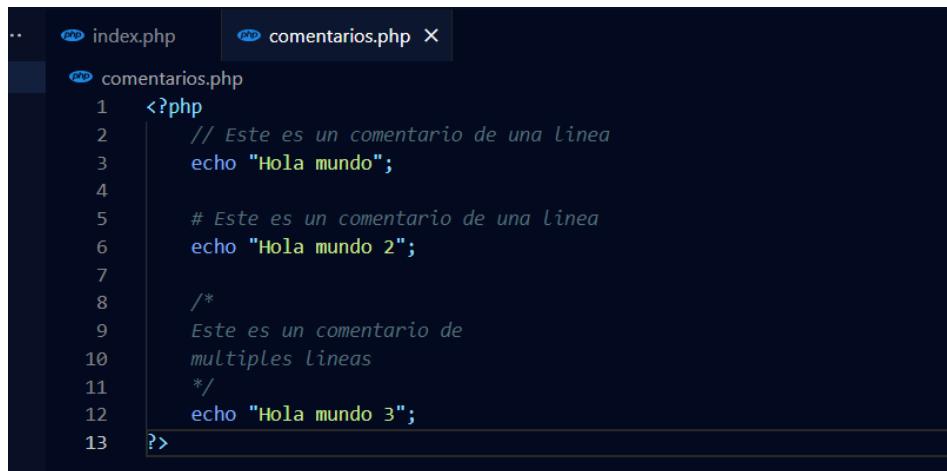
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>Hola mundo desde h1 en html</h1>
    <?php
        echo "Hola mundo desde php";
    ?>
</body>
</html>
```

Resultado:



## Comentarios en php:

Hay 2 formas de poner comentarios de una sola línea, y una forma para comentarios de líneas múltiples.



```
<?php
// Este es un comentario de una linea
echo "Hola mundo";

# Este es un comentario de una linea
echo "Hola mundo 2";

/*
Este es un comentario de
multiples lineas
*/
echo "Hola mundo 3";
?>
```

## Tipos de datos en PHP

### Tipos de datos en PHP

1. **Booleanos (boolean)**: TRUE (1), FALSE (0)
2. **Enteros (integer)**: 10,7, -20, -9
3. **Flotantes (double)**: 7.90, 0.20, -12.75, 3.1416
4. **Cadenas (string)**: "Cadenas de texto", 'Cadena de texto 2'



Ejemplos de tipos de datos en código:

Booleano

```
index.php      comentarios.php      tipos_datos.php X  
tipos_datos.php  
1 <?php  
2 echo TRUE;  
3 ?>
```

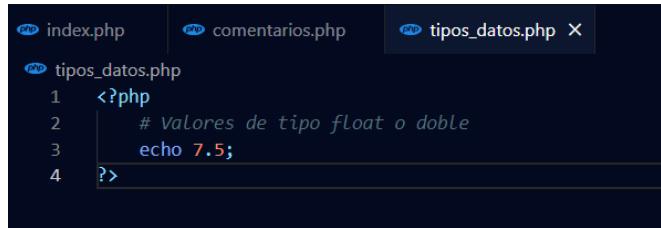
```
< > C   localhost:90/curso_php/tipos_datos.php  
1
```

Entero (también pueden ser negativos):

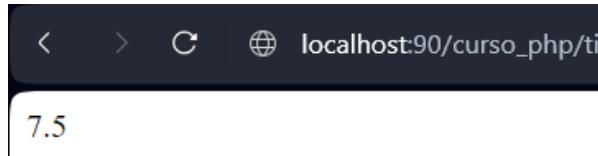
```
index.php      comentarios.php      tipos_datos  
tipos_datos.php  
1 <?php  
2 # Valores enteros  
3 echo -10;  
4 ?>
```

```
< > C   localhost:90/curso_php/tipos_datos.php  
-10
```

Float:



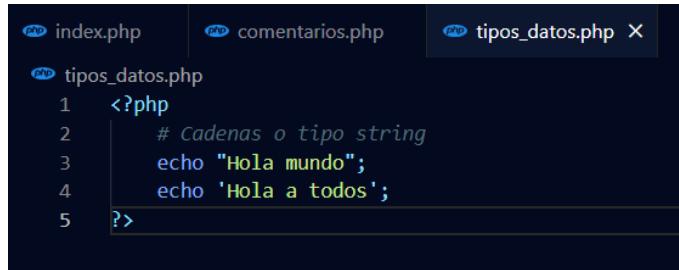
```
index.php | comentarios.php | tipos_datos.php X
tipos_datos.php
1 <?php
2     # Valores de tipo float o doble
3     echo 7.5;
4 ?>
```



localhost:90/curso\_php/tipos\_datos.php

7.5

Cadenas de texto o de tipo String:



```
index.php | comentarios.php | tipos_datos.php X
tipos_datos.php
1 <?php
2     # Cadenas o tipo string
3     echo "Hola mundo";
4     echo 'Hola a todos';
5 ?>
```

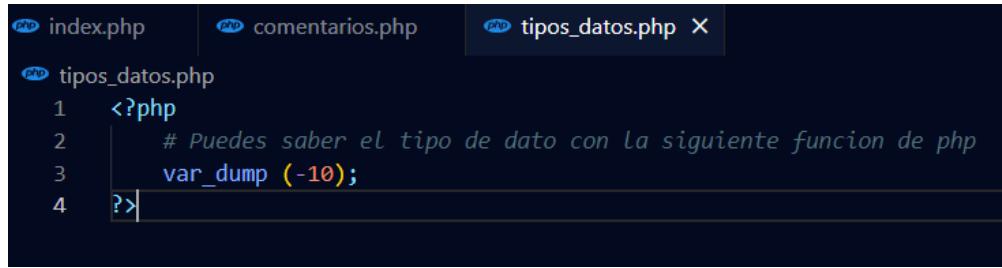


localhost:90/curso\_php/tipos\_datos.php

Hola mundoHola a todos

¿Cómo saber que tipo de dato es?:

Puedes saber el tipo de dato que estas ocupando con la función que tiene php var\_dump



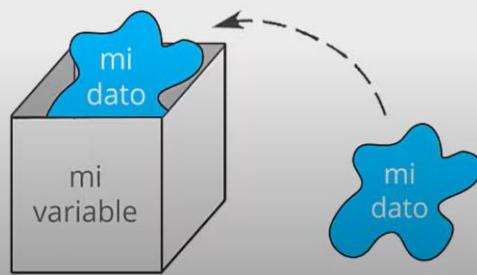
```
index.php | comentarios.php | tipos_datos.php X
tipos_datos.php
1 <?php
2     # Puedes saber el tipo de dato con la siguiente funcion de php
3     var_dump (-10);
4 ?>
```

int(-10)

## ¿Qué es una variable?

Una variable es un espacio de memoria (con un nombre) reservado para almacenar un valor, correspondiente a un tipo de dato (texto, numérico, booleano etc.)

El **valor** de una variable **puede cambiar** durante la ejecución del programa.



## Reglas para definir una variable en PHP

1. En PHP las variables se representan con un **signo de dólar** seguido por el nombre de la variable.
2. Un nombre de variable válido tiene que empezar con una letra o un carácter de subrayado (guion bajo), seguido de cualquier número de letras, números y caracteres de subrayado. **No puede empezar con un numero.**
3. El nombre de la variable es **sensible a minúsculas y mayúsculas**. Se puede usar el estándar de nomenclatura **Camel Case**.

Ejemplo de declarar, inicializar y mostrar una variable:

The screenshot shows a code editor with several tabs at the top: index.php, comentarios.php, tipos\_datos.php, variables.php (which is the active tab), and variables.php X. The code in the editor is as follows:

```
<?php  
# Usando $ declaro mi variable:  
$nombre;  
  
# Inicializo:  
$nombre = "Benjamin";  
  
# Muestro el valor de la variable:  
echo $nombre;  
?>
```

O también puede ser así:

The screenshot shows a code editor with the variables.php tab active. The code is identical to the one above:

```
<?php  
# Declaro e inicio:  
$nombre = "Benjamin";  
  
# Muestro el valor de la variable:  
echo $nombre;  
?>
```

localhost:90/curso\_php/variables.php

Benjamin

Estándares de nomenclatura:

## Estándares de nomenclatura

**Es un conjunto de normas** (un estándar de nomenclatura) para un lenguaje de programación que se recomienda usar como **buenas prácticas** para facilitar la **lectura del código** y este sea fácilmente más entendible y mantenable.

**1. Camel Case:** El nombre viene porque se asemeja a las dos jorobas de un camello, y se puede dividir en dos tipos:

– **Upper Camel Case:** Cuando la primera letra de cada una de las palabras es mayúscula. También denominado **Pascal Case**. Ejemplo: EjemploDeNomenclatura.

– **Lower Camel Case:** Igual que la anterior con la excepción de que la primera letra es minúscula. Ejemplo: ejemploDeNomenclatura.

Nombres de variables que no podemos utilizar porque ya están predefinidas para algo.

## Variables predefinidas

1. **`$GLOBALS`** – Hace referencia a todas las variables disponibles en el ámbito global
2. **`$_SERVER`** – Información del entorno del servidor y de ejecución
3. **`$_GET`** – Variables HTTP GET
4. **`$_POST`** – Variables POST de HTTP
5. **`$_FILES`** – Variables de subida de ficheros HTTP
6. **`$_REQUEST`** – Variables HTTP Request
7. **`$_SESSION`** – Variables de sesión
8. **`$_ENV`** – Variables de entorno
9. **`$_COOKIE`** – Cookies HTTP
10. **`$php_errormsg`** – El mensaje de error anterior
11. **`$HTTP_RAW_POST_DATA`** – Datos POST sin tratar
12. **`$http_response_header`** – Encabezados de respuesta HTTP
13. **`$argc`** – El número de argumentos pasados a un script
14. **`$argv`** – Array de argumentos pasados a un script
15. **`$this`** – Palabra reservada utilizada en programación orientada a objetos

Constantes en php:

## ¿Qué es una constante?

En programación, una constante es un valor que **no puede ser alterado/modificado** durante la ejecución de un programa, únicamente puede ser leído.

Una constante corresponde a una longitud fija de un área reservada en la memoria principal del ordenador, donde el programa almacena valores fijos.

Por ejemplo:

- El valor de PI = 3,1416
- DNI



## Reglas para definir una constante en PHP

1. El nombre de una constante sigue las mismas reglas que cualquier variable de PHP. Un nombre de constante válido empieza por una letra o guion bajo, seguido por cualquier número de letras, números o guiones bajos.
2. Por defecto, una constante distingue mayúsculas y minúsculas.
3. Por convención, los identificadores de constantes siempre se declaran en mayúsculas.

### Sintaxis

```
define("NOMBRE", valor);
```

A partir de PHP 5.3.0

```
const NOMBRE = valor;
```

Ejemplo en código de declarar una variable (manera antigua) con define:

```
variables.php constantes.php X
constantes.php
1 <?php
2 # Defino mi constante llamada EDAD entre comillas " "
3 define("EDAD", 11);
4
5 # Muestro mi constante
6 echo EDAD;
7 ?>
```

```
localhost:90/curso_php/constantes.php
11
```

Si intento cambiarle el primer valor que le coloque al inicio del programa cuando declare la variable, me dice que la constante ya se encuentra definida, es decir que ya almacena un valor por lo tanto no se puede modificar debido a que es una constante NO una variable.

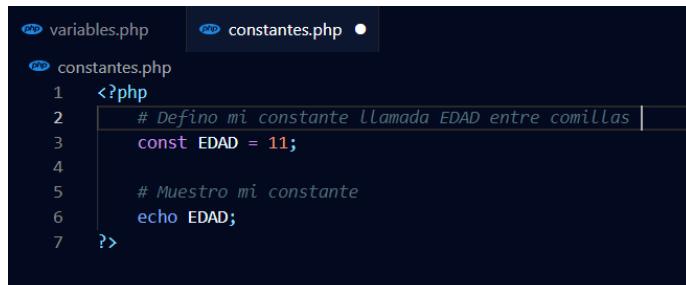
```
variables.php constantes.php
constantes.php
1 <?php
2 # Defino mi constante llamada EDAD entre comillas " "
3 define("EDAD", 11);
4
5 # Intento cambiarle el valor
6 define("EDAD", "Tengo 11");
7
8 # Muestro mi constante
9 echo EDAD;
10 ?>
```



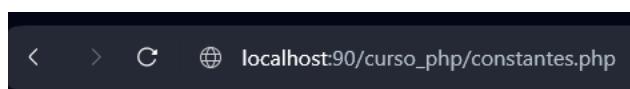
```
variables.php constantes.php
constantes.php
1 <?php
2     # Defino mi constante Llamada EDAD entre comillas |
3     const EDAD = 11;
4
5     # Muestro mi constante
6     echo EDAD;
7 ?>
```

Warning: Constant EDAD already defined in C:\laragon\www\curso\_php\constantes.php on line 5  
11

También se puede definir una variable con la (manera más nueva) usando const:



```
variables.php constantes.php
constantes.php
1 <?php
2     # Defino mi constante Llamada EDAD entre comillas |
3     const EDAD = 11;
4
5     # Muestro mi constante
6     echo EDAD;
7 ?>
```



```
< > C localhost:90/curso_php/constantes.php
```

11

Nombres de constantes que no debemos de ocupar:

## Constantes predefinidas

Varias constantes "mágicas" de PHP

Nombre	Descripción
<code>__LINE__</code>	El número de línea actual en el fichero.
<code>__FILE__</code>	Ruta completa y nombre del fichero con enlaces simbólicos resueltos. Si se usa dentro de un include, devolverá el nombre del fichero incluido.
<code>__DIR__</code>	Directorio del fichero. Si se utiliza dentro de un include, devolverá el directorio del fichero incluido. Esta constante es igual que <code>dirname(__FILE__)</code> . El nombre del directorio no lleva la barra final a no ser que esté en el directorio root.
<code>__FUNCTION__</code>	Nombre de la función.
<code>__CLASS__</code>	Nombre de la clase. El nombre de la clase incluye el namespace declarado en (p.ej. <code>Foo\Bar</code> ). Tenga en cuenta que a partir de PHP 5.4 <code>__CLASS__</code> también funciona con traits. Cuando es usado en un método trait, <code>__CLASS__</code> es el nombre de la clase del trait que está siendo utilizado.
<code>__TRAIT__</code>	El nombre del trait. El nombre del trait incluye el espacio de nombres en el que fue declarado (p.ej. <code>Foo\Bar</code> ).
<code>__METHOD__</code>	Nombre del método de la clase.
<code>__NAMESPACE__</code>	Nombre del espacio de nombres actual.
<code>ClassName::class</code>	El nombre de clase completamente cualificado. Véase también <a href="#">::class</a> .

Arrays o arreglos en php a partir de la versión 7:

## Arrays en PHP

Un array en PHP es un **tipo especial de datos** que representa los llamados **mapas ordenados de datos**. Un mapa es un tipo de datos que asocia **valores con claves**, también llamado '**array asociativo**'.

Los valores que almacenamos en un array **no tienen por qué ser del mismo tipo** como ocurre en otros lenguajes de programación.

La **posibilidad de utilizar** como valores de un array otros arrays, árboles, les permite la definición de **arrays multidimensionales**.



PHP soporta tanto **arrays escalares** (índice numérico), **arrays asociativos** (índice por clave) y **arrays multidimensionales**.

Para acceder a los elementos de un array se utilizan los corchetes **[]**, dentro de los cuales se indicará un **índice o clave** de localización.

A diferencia de otros lenguajes en PHP no hace falta definir el array antes de utilizarlo. Cuando se definen elementos de un array, PHP reconoce automáticamente que se trata de un array sin necesidad de declaración previa.

Tipos de arrays

## Tipos de Arrays

### Escalares

Los arrays escalares **son** aquellos en los que para acceder a los elementos utilizamos un índice que representa la posición del valor dentro del array comenzando desde el índice 0.

### Asociativos

Para acceder a los elementos del array utilizamos la clave asociada con el elemento, donde este toma un cierto número de parejas utilizando la sintaxis clave => valor como argumentos.

### Multidimensionales

Un array multidimensional es aquel cuyos valores son otros arrays. Para acceder a sus elementos se tienen que indicar los índices de cada una de sus dimensiones, utilizando tantos pares de corchetes como dimensiones se definan en el array: **[][] -> 2 dimensiones, [][] -> 3 dimensiones, etc...**

## Definir un Array

En PHP podemos definir arrays utilizando el constructor del lenguaje **array()** o utilizando la notación de **corchetes []** (a partir de PHP 5.4) también llamada sintaxis corta.

Si a la hora de definir el array no se **indican los valores**, estaremos creando un **array vacío**.

### A partir de PHP 7

```
define('ANIMALES', array(  
    'perro',  
    'gato',  
    'pájaro'  
));
```

```
echo ANIMALES[1];
```

Ejemplo de arreglos escalares:

Arreglo almacenado en una **constante** llamada NOMBRES

The screenshot shows a code editor with several tabs at the top: 'variables.php', 'constantes.php', 'array.php' (which is the active tab), and 'info.php'. The 'array.php' tab contains the following code:

```
<?php  
# Creo y lleno mi arreglo  
define("NOMBRES",array("Benji", "Brian", "Marlot", "Vani"));  
# Muestro Los valores de mi array ubicados en el indice 0 y 2  
echo NOMBRES[0];  
echo NOMBRES[3];
```

```
BenjiVani
```

Arreglo en una variable llamada **estudiantes**, (aquí si podemos cambiar el valor de cualquier posición del array ya que está dentro de una **variable**).

```
<?php
# Creo mi arreglo en una constante Llamada NOMBRES;
define("NOMBRES",array("Benji", "Brian", "Marlot", "Vani"));
echo NOMBRES[0];

// Creo mi arreglo en una variable y puedo hacer cambios en Los valores del array:
$estudiantes = array("losCar", "Benja", "Matteo");
$estudiantes[0] = "Carlos";
echo $estudiantes[0];

# De forma mas nueva tambien se puede definir un array con corchetes
$valores = ["Hola a todos", true, -12, 55.9];
echo $valores[3];
?>
```

```
BenjiCarlos55.9
```

Ejemplo de arreglos asociativos:

```
<?php
$tutor = [
    "nombre" => "Benjamin",
    "apellido" => "Peña",
    "edad" => 50
];

# Modificamos el valor en la clave edad
$tutor["edad"] = 20;
echo $tutor["edad"];
?>
```

## Ejemplo de arreglos multidimensionales:

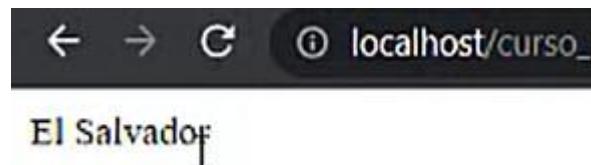
```
... variables.php constantes.php arraysEscalares.php arraysAsociativos.php arraysMultidimensionales.php ● info.php  
p arraysMultidimensionales.php  
1 <?php  
2 /*Son el tipo de arreglo en donde accedes al elemento por medio de una clave que le asignas  
3 a cada valor almacenado en el array por ejemplo nombre es la clave del valor Benjamin y  
4 tambien que tiene arreglos dentro del arreglo principal por eso es multidimensional*/  
5  
6 # Este array tiene 2 dimensiones, el arreglo principal de tipo asociativo y el de cursos que es escalar  
7 $tutor = [  
8     "nombre"=>"Vania",  
9     "apellido"=>"Velazquez",  
10    "edad"=>50,  
11    "cursos"=>["Piton","PHP","Java"]  
12];  
13 $tutor["cursos"][0] = "Python";  
14 echo $tutor["cursos"][0];  
15  
16 # Tambien se puede tener el segundo array asociativo:  
17 $datos = [  
18     "nombre"=>"Vania",  
19     "apellido"=>"Velazquez",  
20     "edad"=>50,  
21     "carreras"=>[["nombre"=>"licenciado",  
22                     "tiempo"=>4,  
23                     "Distancia"=>"No mucha"  
24                 ]  
25];  
26  
27 $datos["carreras"]["nombre"] = "programador";  
28 echo $datos["carreras"]["nombre"];  
29  
30 ?>
```

Pythonprogramador

**También podemos** crear un nuevo campo en un arreglo, aunque ya este definido, por ejemplo:

Aquí añadimos un nuevo campo (país) en el array principal (tutor\_2) y un valor a ese campo (El salvador) y después lo mostramos.

```
3 # Array de multiples dimensiones
4 $tutor_2=[
5     "nombre"=>"Vanessa",
6     "apellido"=>"Calles",
7     "edad"=>20,
8     "cursos"=>["PHP", "Python", "CSS"]
9 ];
10
11 $tutor_2["pais"]="El Salvador";
12
13 echo $tutor_2["pais"];
```



Contar elementos almacenados en un array:

Eso lo logramos usando la función **count()**

```
variables.php constantes.php arraysEscalares.php arraysAsociativos.php arraysMultidimensionales
contarArray.php
1 <?php
2     # Aquí tengo un array multidimensional con 2 arreglos de tipo Asociativo.
3     $datos = [
4         "nombre"=>"Vania",
5         "apellido"=>"Velazquez",
6         "edad"=>50,
7         "carreras"=>["nombre"=>"licenciado",
8                     "tiempo"=>4,
9                     "Distancia"=>"No mucha"
10                ];
11
12
13     # Muestra el numero de elementos que tiene el arreglo almacenado en la variable datos
14     echo count($datos);
15 ?>
```

```
< > C localhost:90/curso_php/contarArray.php  
4
```

También se puede **contar los de un segundo arreglo solamente o de todos los arreglos totales de un array multidimensional** con la función **COUNT\_RECURSIVE**: en este caso son 7

```
arraysEscalares.php arraysAsociativos.php arraysMultidimensionales.php • contarArray.php X  
contarArray.php  
1 <?php  
2     # Aquí tengo un array multidimensional con 2 arreglos de tipo Asociativo.  
3     $datos = [  
4         "nombre"=>"Vania",  
5         "apellido"=>"Velazquez",  
6         "edad"=>50,  
7         "carreras"=>[ "nombre"=>"licenciado",  
8                         "tiempo"=>4,  
9                         "Distancia"=>"No mucha"  
10                    ]  
11    ];  
12  
13     # Muestra el numero de elementos que tiene el arreglo almacenado en la variable datos  
14 echo count($datos);  
15 echo (" ");  
16 /* Muestra el numero de elementos que tiene el arreglo almacenado en la variable datos  
en la posicion de la clave carreras*/  
17 echo count($datos);  
18 echo (" ");  
19     # Muestra el numero de elementos totales de todos los arreglos  
20 echo count($datos,COUNT_RECURSIVE);  
21  
22 ?>
```

```
< > C localhost:90/curso_php/contarArray.php  
4 4 7
```

## Ver información de tu php:

Con esto puedes saber la versión con la que estás trabajando además de más datos:

The screenshot shows a code editor with several tabs: variables.php, constantes.php, array.php, and info.php. The info.php tab is active, displaying the following code:

```
<?php  
echo phpinfo();  
??
```

Below the code editor is a browser window with the URL `localhost:90/curso_php/info.php`. The page title is "PHP Version 8.1.10". The content area displays detailed PHP configuration information in a table:

System	Windows NT BENJAMIN 10.0 build 22631 (Windows 11) AMD64
Build Date	Aug 30 2022 18:02:43
Build System	Microsoft Windows Server 2019 Datacenter [10.0.17763]
Compiler	Visual C++ 2019
Architecture	x64
Configure Command	<code>./configure --enable-snapshot-build --enable-debug-pack --with-pdo-oci=../oci8/instantclient/sdk/shared --with-oci8-19=../oci8/instantclient/sdk/shared --enable-object-out-dir=../obj --enable-com-dotnet-shared --without-analyzer --with-pgo</code>
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	<code>no value</code>
Loaded Configuration File	<code>C:\laragon\bin\php\php-8.1.10-Win32-vs16-x64\php.ini</code>
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20210902
PHP Extension	20210902
Zend Extension	420210902

## Como concatenar string o cadenas de texto:

El símbolo que se utiliza para concatenar strings o variables en php es el punto ( . ) no es como en otros lenguajes que es con el signo de más, por ejemplo aquí concatenamos el valor de 2 variables.

The screenshot shows a code editor with several tabs: arraysEscalares.php, arraysAsociativos.php, arraysMultidimensionales.php, and concatenacion.php. The concatenacion.php tab is active, displaying the following code:

```
<?php  
$nombre="Benjamin";  
$pais="Mexico";  
echo $nombre.$pais;  
??
```

Below the code editor is a browser window with the URL `localhost:90/curso_php/concatenacion.php`. The page displays the concatenated string: **BenjaminMexico**.

También se puede almacenar todo dentro de una variable:

```
arraysEscalares.php arraysMultidimensionales.php concatenacion.php X
concatenacion.php
1 <?php
2     $nombre="Benjamin ";
3     $apellido="Peña";
4
5     $resultado = $nombre.$apellido;
6
7     echo $resultado;
8 ?>
```

< > C localhost:90/curso\_php/concatenacion.php

Benjamin Peña

También se puede hacer así:

```
arraysEscalares.php arraysMultidimensionales.php concatenacion.php X
concatenacion.php
1 <?php
2     $nombre="Benjamin ";
3     $apellidoPaterno="Peña ";
4     $apellidoMaterno="Marín";
5
6     $nombreCompleto = $nombre.$apellidoPaterno.$apellidoMaterno;
7
8     echo "Mi nombre es: ".$nombreCompleto;
9 ?>
10
```

Operadores aritméticos o matemáticos:

## Operadores aritméticos

Nombre	Símbolo en PHP
Suma	+
Resta	-
División	/
Resto de (modulo o residuo)	%
Multiplicación	*
Exponenciación	**

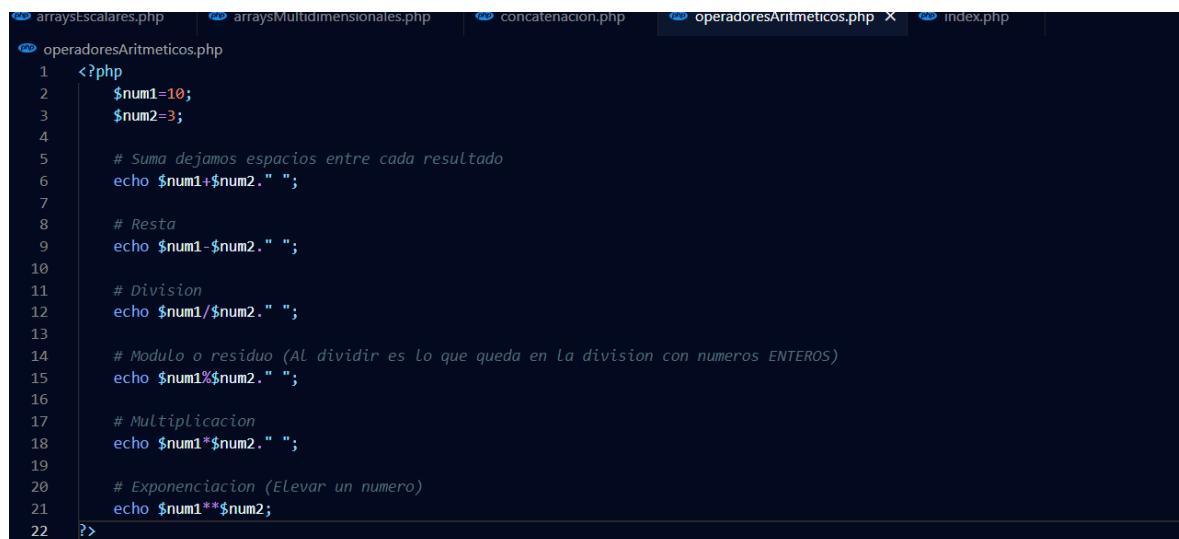


La jerarquía de operadores determina el orden en el que se resuelven las expresiones cuando se involucran operaciones aritméticas como la suma, resta, multiplicación, división, potencia, módulo de la división.

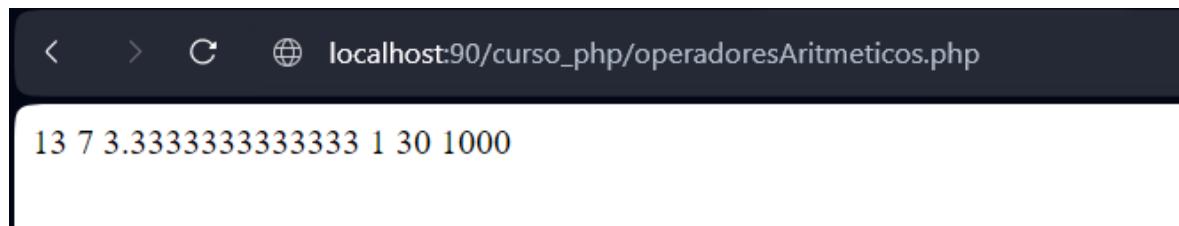
Jerarquía	Operador	Comentario
1	( )	Se ejecuta primero
2	**	Después de ( )
3	* , /	Después de los anteriores
4	+ , -	Después de los anteriores
5	%	Después de los anteriores

Cuando dos operadores tienen el mismo nivel de prioridad, dentro de una expresión se evalúan de izquierda a derecha.

Ejemplo en código del uso de cada uno (Siempre hay que tener en cuenta las jerarquías):



```
arraysEscalares.php arraysMultidimensionales.php concatenacion.php operadoresAritmeticos.php index.php
operadoresAritmeticos.php
1 <?php
2     $num1=10;
3     $num2=3;
4
5     # Suma dejamos espacios entre cada resultado
6     echo $num1+$num2." ";
7
8     # Resta
9     echo $num1-$num2." ";
10
11    # Division
12    echo $num1/$num2." ";
13
14    # Modulo o residuo (Al dividir es lo que queda en la division con numeros ENTEROS)
15    echo $num1%$num2." ";
16
17    # Multiplicacion
18    echo $num1*$num2." ";
19
20    # Exponentiacion (Elevar un numero)
21    echo $num1**$num2;
22 ?>
```



localhost:90/curso\_php/operadoresAritmeticos.php

13 7 3.3333333333333 1 30 1000

Shortcuts u operadores de asignación:

## Operadores de asignación

Nombre	Símbolo en PHP	Ejemplo
Asignar	=	\$a=10;
Sumar y asignar	+=	\$a+=5
Restar y asignar	-=	\$a-=5
Multiplicar y asignar	*=	\$a*=5
Dividir y asignar	/=	\$a/=5
Concatenar y asignar	.=	\$a.=" es el valor de A"

Los operadores de asignación nos permiten realizar operaciones de una forma mas corta por ejemplo:  
\$a+=5 es lo mismo que \$a=\$a+5

Funcionan de la siguiente manera: Es como poner **\$numero = \$numero.2;**

Pero de manera más corta, por eso se llaman shortcuts.

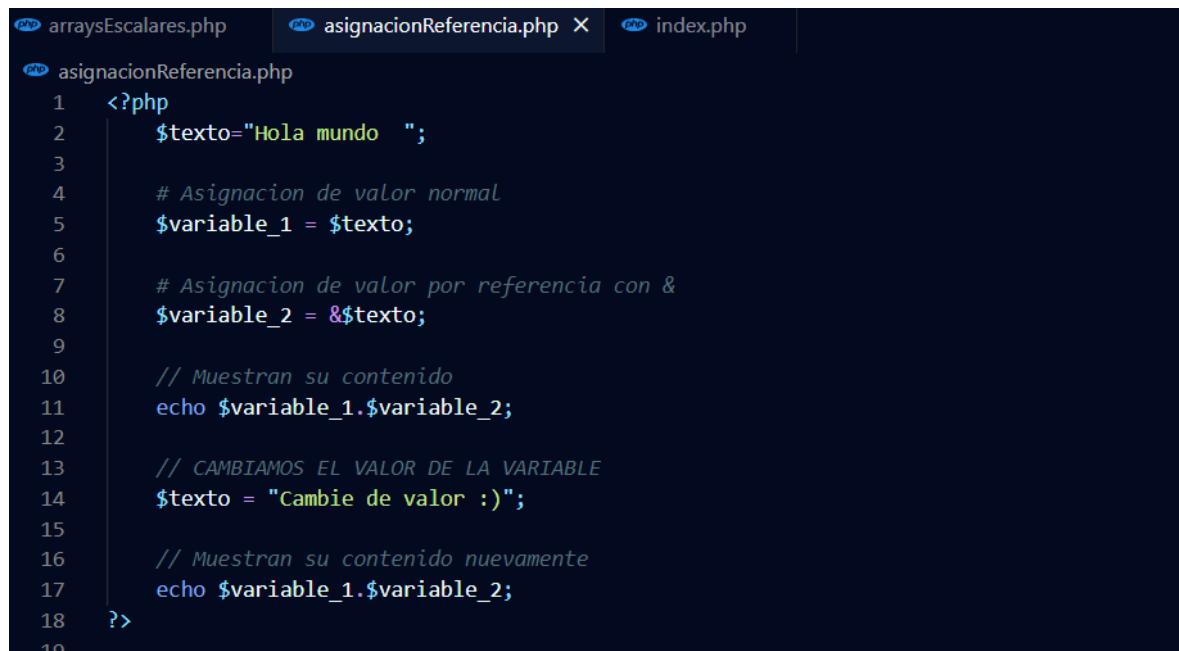
```
operadores_asignacion.php X
operadores_asignacion.php
1 <?php
2
3 $numero="Texto de prueba ";
4 $numero.=2;
5
6 echo $numero;
```

localhost/curso\_php/operadores\_asignacion.php

Texto de prueba 2

## Asignación de valor por referencia a una variable:

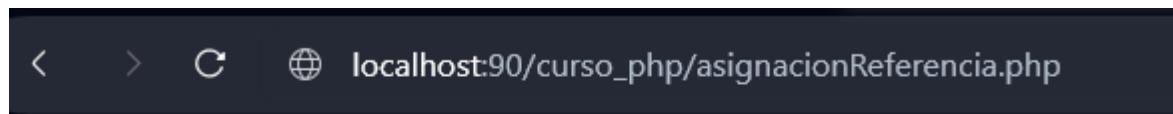
En este ejemplo se puede ver que pareciera lo mismo, pero NO ES LO MISMO, ya que en el primer caso estamos COPIANDO el valor que tiene mi variable y almacenarla, y al asignar un valor por referencia ocupando el símbolo & significa que, si el valor de la variable cambia, también cambiara el valor de la segunda variable con ese valor.



```
arraysEscalares.php | asignacionReferencia.php X | index.php

asignacionReferencia.php
1  <?php
2      $texto="Hola mundo ";
3
4      # Asignacion de valor normal
5      $variable_1 = $texto;
6
7      # Asignacion de valor por referencia con &
8      $variable_2 = &$texto;
9
10     // Muestran su contenido
11     echo $variable_1.$variable_2;
12
13     // CAMBIAMOS EL VALOR DE LA VARIABLE
14     $texto = "Cambio de valor :)";
15
16     // Muestran su contenido nuevamente
17     echo $variable_1.$variable_2;
18 ?>
19
```

Al principio observaremos que ambas variables variable\_1 y variable\_2 tienen el mismo contenido, pero luego al cambiar el valor que almacena la variable texto, también cambia el de variable\_2 ya que su valor este asignado por referencia usando el símbolo &.



Hola mundo Hola mundo Hola mundo Cambie de valor :)

# Operadores de comparación

Los operadores de comparación comparan dos expresiones y devuelven un valor Booleano que puede ser TRUE (1) o FALSE (0).

Ejemplo	Símbolo (Nombre)	Resultado
1 == "1"	== (Igual)	TRUE
10 === "10"	=== (Idéntico)	FALSE
7 != "7"	!= (Diferente)	FALSE
21 <> "21"	<> (Diferente)	FALSE
19 !=== "19"	!== (No idéntico)	TRUE
7 < 4	< (Menor que)	FALSE
7 > 4	> (Mayor que)	TRUE
2 <= 2	<= (Menor o igual que)	TRUE
3 >= 7	>= (Mayor o igual que)	FALSE

Estas devolverán un resultado booleano, aquí la diferencia entre igual == e idéntico ===

```
arraysEscalares.php | asignacionReferencia.php | operadoresComparacion.php | index.php
operadoresComparacion.php
1  <?php
2  # Es lo mismo aunque uno es int y otro String pero eso no interesa
3  var_dump(1=="1"); //TRUE
4
5  # Aquí si importa que sean hasta el mismo tipo de dato
6  var_dump(1==='1') //FALSE
7 ?>
8
```

localhost:90/curso\_php/operadoresComparacion

bool(true) bool(false)

Operadores lógicos (and or not etc)

## Operadores lógicos

Los operadores lógicos permiten combinar expresiones simples en expresiones más complejas, estas expresiones se evalúan y devuelven un valor booleano.

Símbolo	Nombre
and	And (y)
or	Or (o)
!	Not (no)
&&	And (y)
	Or (o)

### Tabla de operador AND

EXPRESIÓN 1	EXPRESIÓN 2	RESULTADO
FALSE	&&	FALSE
FALSE	&&	TRUE
TRUE	&&	FALSE
TRUE	&&	TRUE

**Conclusión:** Ambas expresiones tienen que ser verdaderas para que el resultado sea verdadero.

### Tabla de operador OR

EXPRESIÓN 1	EXPRESIÓN 2	RESULTADO
FALSE		FALSE
FALSE		TRUE
TRUE		FALSE
TRUE		TRUE

**Conclusión:** Al menos tiene que ser verdadera una expresión para que el resultado sea verdadero

## Tabla de operador NOT

EXPRESIÓN	RESULTADO
!FALSE	TRUE
!TRUE	FALSE

**Conclusión:** Devuelve el valor contrario de la expresión

La siguiente expresión da FALSE, pero como se está **negando** con **!** devuelve TRUE.

```
php operadores_logicos.php •
operadores_logicos.php
1 <?php
2 $valor_1=7;
3 $valor_2=2;
4
5 var_dump(!($valor_1==$valor_2));
```

Esta expresión devuelve TRUE usando and **&&**. De la misma forma se puede ocupar or **||**

```
php operadores_logicos.php ×
operadores_logicos.php
1 <?php
2 $valor_1=7;
3 $valor_2=2;
4
5 var_dump($valor_1==7 && 9>3);
```

## Operadores de incremento y decremento

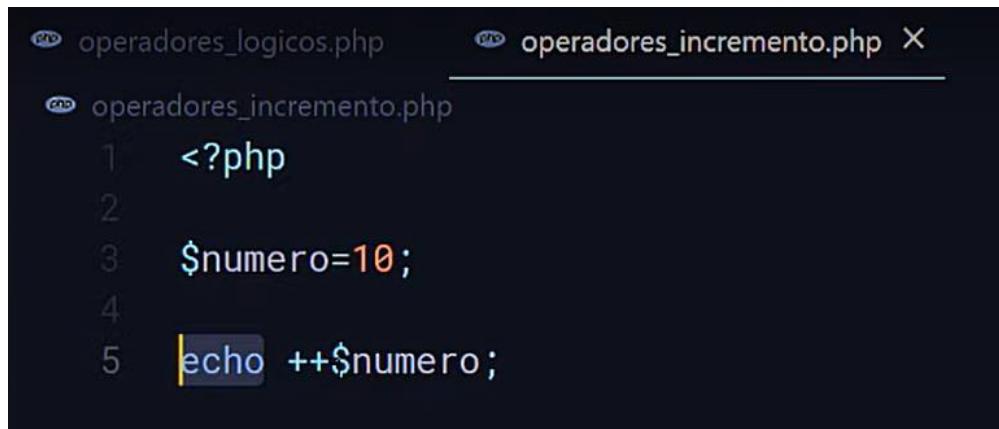
### Operadores de incremento y decremento

Símbolo	Nombre
++	Incremento
--	Decremento

### EJEMPLOS

++\$variable	Pre-Incremento
\$variable++	Post-Incremento
--\$variable	Pre-Decremento
\$variable--	Post-Decremento

Aquí primero incrementamos y luego mostramos con la función echo, **mostrara 11**



```
operadores_logicos.php  operadores_incremento.php X
operadores_incremento.php
1 <?php
2
3 $numero=10;
4
5 echo ++$numero;
```

Aquí primero se muestra con la función echo y se incrementa por lo que, **mostrara 10 y luego en el siguiente echo mostrara ya 11.**

```
<?php
$numero=10;
echo $numero++;
echo $numero;
```

## Estructura condicionar simple (IF)

### ESTRUCTURA CONDICIONAL SIMPLE (IF)

Permite la ejecución fragmentos de código, la expresión es evaluada a su valor booleano. Si la expresión se evalúa como **TRUE**, PHP ejecutará la sentencia y si se evalúa como **FALSE** la ignorará.

```
if(expresión){
    Código a ejecutar
}
```

```
if(expresión):
    Código a ejecutar
endif;
```

Se puede hacer un if con las 2 estructuras mostradas anteriormente:

Aquí dirá: **Expresión verdadera** debido a que 9 si es mayor a 8 **TRUE**.

```
<?php
if(9>=8):
| echo "Expresion verdadera";
| endif;|
```

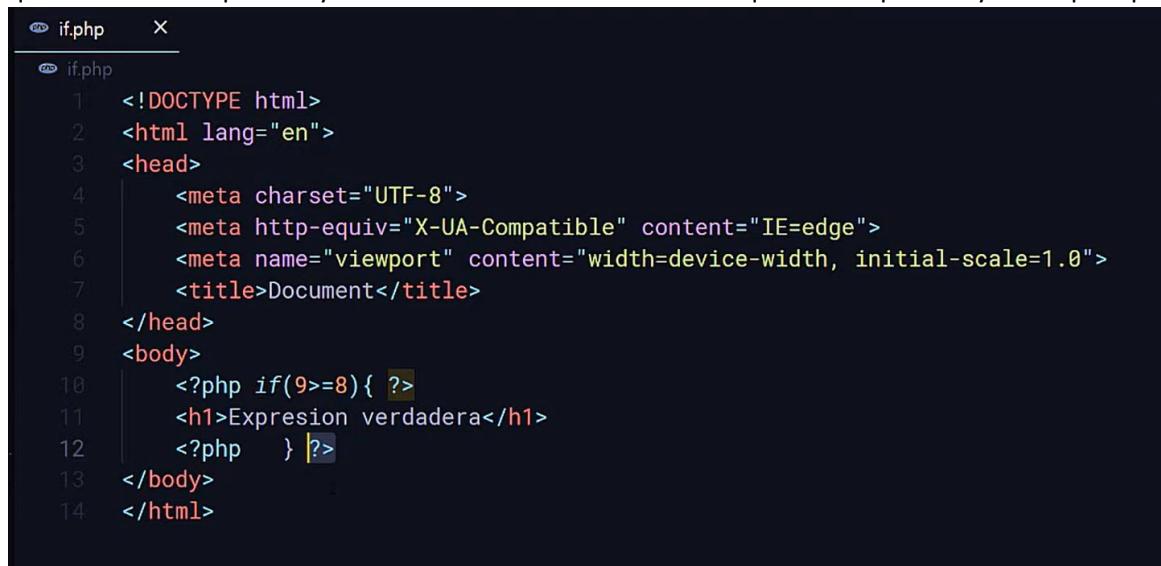
Es lo mismo que hacerlo así, ambas son correctas:



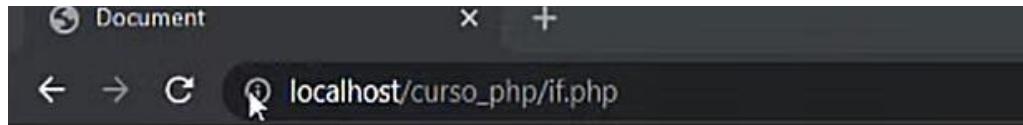
```
if.php
if.php
1 <?php
2
3     if(1>=0){
4         echo "Expresion verdadera";
5     }
```

Se ocupan mucho al combinar HTML con PHP por ejemplo:

Como 9 es mayor a 8 se ejecutara el código HTML con la etiqueta h1 que se observa. Observemos que las llaves de apertura y cierre del if están dentro de las etiquetas de apertura y cierre para php.



```
if.php
if.php
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7     <title>Document</title>
8 </head>
9 <body>
10    <?php if(9>=8){ ?>
11        <h1>Expresion verdadera</h1>
12    <?php } ?>
13 </body>
14 </html>
```



# Expresion verdadera

## EJERCICIOS (IF)

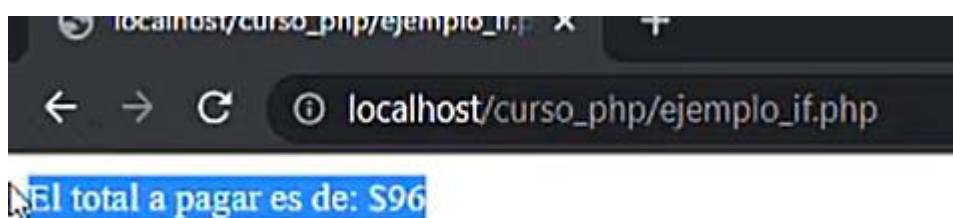
1 - Realizar un programa en donde se pide la edad del usuario; si es mayor de edad debe aparecer un mensaje indicándolo.

```
ejemplo_if.php
1 <?php
2
3 $edad=20;
4
5 if($edad>=18){
6     echo "Eres mayor de edad";
7 }
```

2 - En un almacén se hace un 20% de descuento a los clientes cuya compra supere los \$100 ¿Cuál será la cantidad que pagara una persona por su compra?

```
ejemplo_if.php X
ejemplo_if.php
1 <?php
2
3 $total=120;
4
5 if($total>100){
6     $total=$total-($total*0.20);
7 }
8
9 echo "El total a pagar es de: ".$total;
```

Como la compra es de 120 y supera los 100 se aplica el 20% de descuento y me da 96:



## IF doble (IF else)

### ESTRUCTURA CONDICIONAL DOBLE (IF - ELSE)

Con frecuencia se desea ejecutar una sentencia si una determinada condición se cumple y una sentencia diferente si la condición no se cumple. Esto es para lo que sirve **else**. El **else** extiende una sentencia **if** para ejecutar una sentencia en caso que la expresión en la sentencia if se evalúe como **FALSE**.

```
if(expresión){  
    Código a ejecutar si la expresión  
    es verdadera  
}else{  
    Código a ejecutar si la expresión  
    es falsa  
}
```

```
if(expresión):  
    Código a ejecutar si la expresión  
    ↴ es verdadera  
else:  
    Código a ejecutar si la expresión  
    es falsa  
endif;
```

Se ejecutará la parte del **else** debido a que la condición es FALSE. Dirá Condición falsa.

```
if_else.php  
if_else.php  
1 <?php  
2  
3     if(1>7){  
4         echo "Condicion verdadera";  
5     }else{  
6         echo "Condicion falsa";  
7     }
```

### EJERCICIOS (IF - ELSE)

- 1 - Calcular el total que una persona debe pagar en una llantera, el precio de cada llanta es de \$800 si se compran menos de 5 llantas y de \$700 si se compran 5 o mas.

```
ejemplo_else.php X
ejemplo_else.php
1 <?php
2     $cantidad=9;
3
4     if($cantidad<5){
5         $total=$cantidad*800;
6     }else{
7         $total=$cantidad*700;
8     }
9
10    echo "El total a pagar es de ".$total;
```

2- Determinar si un alumno aprueba o repreuba un curso, sabiendo que aprobara si su promedio de tres calificaciones es mayor o igual a 7.0; repreuba en caso contrario

```
ejemplo_else.php ^
ejemplo_else.php
1 <?php
2     $calificacion_1=6;
3     $calificacion_2=5;
4     $calificacion_3=4;
5
6     $promedio=($calificacion_1+$calificacion_2+$calificacion_3)/3;
7
8     if($promedio>=7){
9         echo "Estudiante aprobado con calificacion: ".$promedio;
10    }else{
11        echo "Estudiante reprobado con calificacion: ".$promedio;
12    }
```

Operador ternario (IF de una sola línea)

## OPERADOR TERNARIO

El operador ternario puede considerarse como una instrucción if de una sola línea. Se compone de tres partes. **El operator , y dos resultados**. La sintaxis es la siguiente:

<operator> ? <true value> : <false value>

Es un if else de una sola línea, la condición, lo que se ejecuta si es true, y lo que se ejecuta si es false. Ese es el orden en el que se coloca la sintaxis.

```
<?php  
(9>7) ? 10*7 : 10*5;
```

Aquí almacenamos las operaciones en una variable para mostrar con un echo, ya que no se puede poner echo dentro del operador ternario:

```
<?php  
(2>7) ? $total=10*7 : $total=10*5;  
echo $total;
```

### EJERCICIO (OPERADOR TERNARIO)

1 - Hacer un programa que calcule el total a pagar por la compra de camisas. Si se compran tres camisas o mas se aplica un descuento del 20% sobre el total de la compra y si son menos de tres camisas un descuento del 10%.

## ejemplo\_ternario.php

```
1 <?php  
2  
3 $camisas=7;  
4 $precio=10;  
5  
6 $total=$camisas*$precio;  
7  
8 $total= ($camisas>=3) ? $total-($total*0.20) : $total-($total*0.10);  
9  
10 echo "El total a pagar es $".$total;|
```

## IF multiple (IF-ELSEIF-ELSE)

### ESTRUCTURA CONDICIONAL MULTIPLE (IF – ELSEIF)

**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 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**.

```
if ($a > $b) {  
    echo "a es mayor que b";  
} elseif ($a == $b) {  
    echo "a es igual que b";  
} else {  
    echo "a es menor que b";  
}
```

```
if ($a > $b):  
    echo "a es mayor que b";  
elseif ($a == $b):  
    echo "a es igual que b";  
else:  
    echo "a es menor que b";  
endif;
```

```
1 <?php
2     /* Diseña un programa con elseif que escriba los nombres de sus días
3      dependiendo el numero que escribas */
4
5     $dia=0;
6
7     if($dia==1){
8         echo "Lunes";
9     }elseif ($dia==2) {
10        echo "Martes";
11    }elseif ($dia==3) {
12        echo "Miercoles";
13    }elseif ($dia==4) {
14        echo "Jueves";
15    }elseif ($dia==5) {
16        echo "Viernes";
17    }elseif ($dia==6) {
18        echo "Sabado";
19    }elseif ($dia==7) {
20        echo "Domingo";
21    }else{
22        echo "Error ha introducido un valor no valido";
23    }
24 ?>
```

If anidados (if dentro de otro if)

## ESTRUCTURAS CONDICIONALES ANIDADAS

```
if (condicion) {
    if(condicion){
        Bloque de código a ejecutar si la condición es TRUE
    }else{
        Bloque de código a ejecutar si la condición es FALSE
    }
} elseif (condicion) {
    Bloque de código a ejecutar si la condición es TRUE
} else {
    Bloque de código a ejecutar si la condición es FALSE
}
```

## Switch y Match (se parecen un poco)

### 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 que valor es igual. Para esto es exactamente la expresión switch.

La comparación es una comprobación de **igualdad débil** (`==`)

```
switch (variable) {
    case valor1:
        Código a ejecutar
        break;

    case valor2:
        Código a ejecutar
        break;

    default:
        Código a ejecutar
}
```

```
switch.php
switch.php
1  <?php
2
3      $fruta="Fresa";
4
5      switch($fruta){
6          case "Fresa":
7              echo "Eres una fresa";
8              break;
9          case "Pera":
10             echo "Eres una Pera";
11             break;
12         default:
13             echo "No eres ni Fresa ni Pera";
14     }
}
```

### MATCH

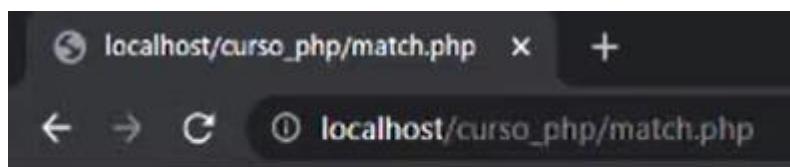
La expresión match ramifica la evaluación basada en una comprobación de identidad de un valor. De forma similar a una **sentencia switch**, una expresión match tiene una expresión de sujeto que se compara con múltiples alternativas.

A diferencia de switch, la comparación es una comprobación de **identidad** (`==`) en lugar de una comprobación de igualdad débil (`==`). Las expresiones match están disponibles a partir de **PHP 8.0.0.**

```
match (variable) {
    $variable => Código a
    ejecutar,
    default => Código a
    ejecutar
};
```

Es muy parecido al switch pero aquí va comparando que sea **idéntico** `==` por ejemplo en el siguiente código, compara si el valor que tiene la variable `a` es idéntico al que tiene la variable `x`. la variable `y`, la variable `z` que es la que coincide, entonces dirá:

```
match.php X
match.php
1 <?php
2
3 $a=7;
4
5 $x=10;
6 $y=9;
7 $z=7;
8
9 $resultado = match($a){
10   $x => "Valor igual a X",
11   $y => "Valor igual a Y",
12   $z => "Valor igual a Z",
13   default => "No coincide con ninguna variable"
14 };
15
16 echo $resultado;
```

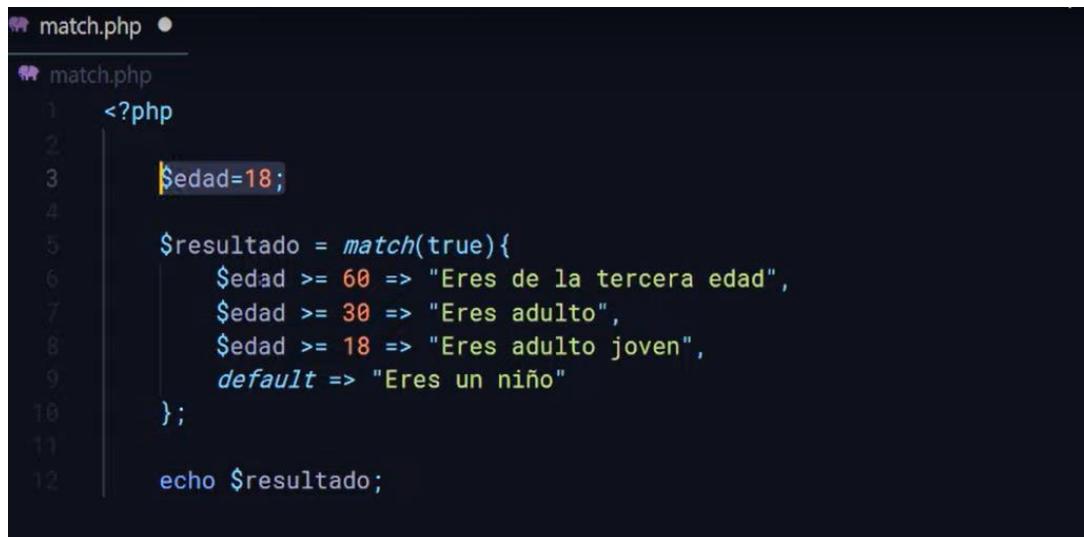


Valor igual a Z

También podemos comparar con mas de un numero en el match, como en este caso, a no es igual a x pero si es igual a y así que se va a ejecutar lo que tengan después de la flecha.

```
match.php X
match.php
1 <?php
2
3 $a=9;
4
5 $x=10;
6 $y=9;
7 $z=7;
8
9 $resultado = match($a){
10   $x,$y => "Valor igual a X o Y",
11   $z => "Valor igual a Z",
12   default => "No coincide con ninguna variable"
13 };
14
15 echo $resultado;
```

O también se puede hacer la comparación con booleanos:



```
match.php •
match.php
1 <?php
2
3 $edad=18;
4
5 $resultado = match(true){
6     $edad >= 60 => "Eres de la tercera edad",
7     $edad >= 30 => "Eres adulto",
8     $edad >= 18 => "Eres adulto joven",
9     default => "Eres un niño"
10 };
11
12 echo $resultado;
```

## Ciclos en php, bucles o en ingles loops

### Ciclo while

#### CICLO WHILE

La instrucción while (mientras) ejecuta una porción de programa mientras se cumpla una cierta condición. Mientras la condición sea **verdadera**, se **ejecutan las instrucciones contenidas** en el while. Cuando **deja de cumplirse la condición**, se **sale del ciclo** y se continúa ejecutando el resto del programa.

```
while(Condicion){
    Código a ejecutar
}
```

```
while(Condicion):
    Código a ejecutar
endwhile;
```

### EJEMPLOS

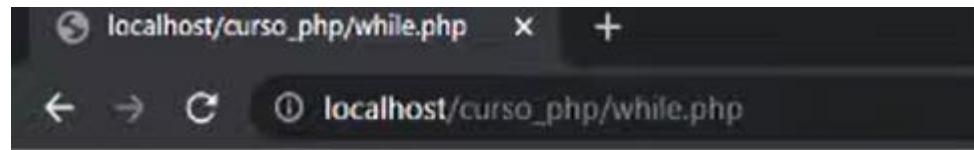
1. Diseñe un programa que imprima los números del 1 hasta el 20. (Incremento y decremento).

En este código me mostrara del 1 al 20, y además use una etiqueta HTML <br> para dar un salto de línea entre cada numero para que sea mas legible

```
while.php X
while.php
1 <?php
2
3     $c=1;
4
5     while($c<=20){
6         echo $c."<br>";
7         $c++;
8     }
```

2. Diseñe un programa que imprima la tabla de multiplicar de un numero dado, desde el factor 1 hasta el 12. (Incremento y decremento).

```
while.php X
while.php
1 <?php
2
3     $c=1;
4     $num=7;
5
6     while($c<=12){
7         echo $num." X ".$c." = ".$num*$c."<br>";
8         $c++;
9     }
```



## Ciclo DO-WHILE

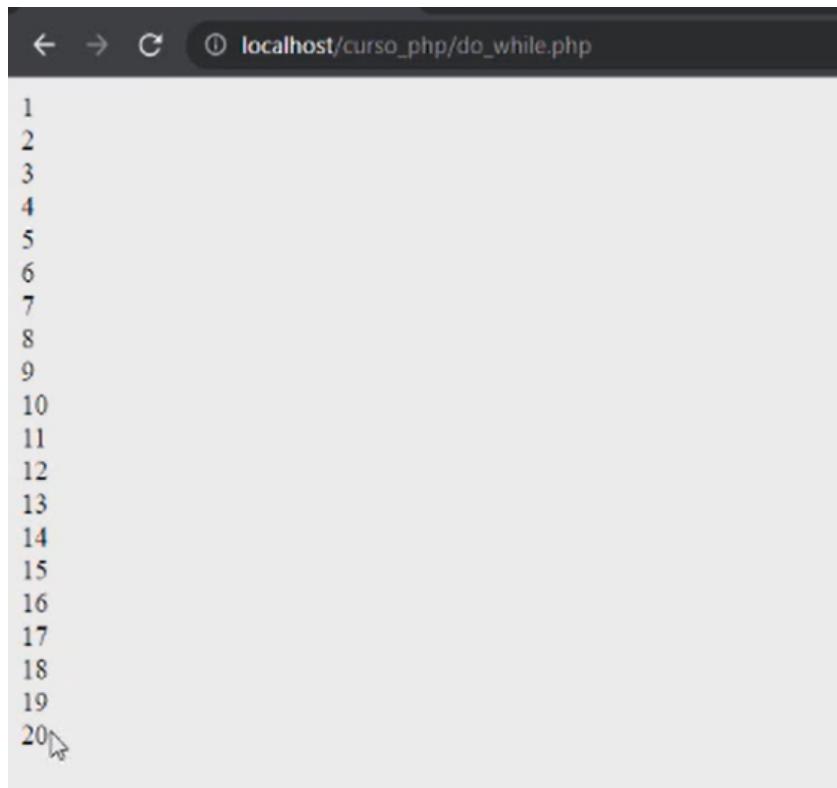
### CICLO DO-WHILE

La estructura repetitiva do-while (hacer-mientras) es muy similar a la estructura while, excepto que **la expresión verdadera es verificada al final** de cada iteración en lugar de al principio. **Hay una sola sintaxis para bucles do-while.**

```
do{  
    Código a ejecutar  
}while(Condicion);
```

Ejemplo en código:

```
<?php  
$c=1;  
do{  
    echo $c."<br>";  
    $c++;  
}while($c<=20);
```



```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

## Ciclo FOR

### CICLO FOR

La estructura repetitiva for (para) se utiliza generalmente cuando tenemos bien determinada la cantidad de repeticiones a realizar. Se **diferencia** de las anteriores en que **se debe incluir** en la propia instrucción una **variable de control**, la cual se **incrementa** o **decrementa** de forma automática

<pre>for (var; Condicion; incre   decre){     Código a ejecutar }</pre>	<pre>for(var; Condicion; incre   decre):     Código a ejecutar endfor;</pre>
---	--

2. Diseñe un programa que imprima la tabla de multiplicar de un número dado, desde el factor 1 hasta el 12. (Incremento y decremento).

```
for.php
for.php
1 <?php
2
3     $numero=7;
4     for($i=1; $i<=12; $i++){
5         echo $numero." X ".$i." = ".$i*$numero."<br>";
6     }

```

localhost/curso\_php/for.php

7 X 1 = 7  
7 X 2 = 14  
7 X 3 = 21  
7 X 4 = 28  
7 X 5 = 35  
7 X 6 = 42  
7 X 7 = 49  
7 X 8 = 56  
7 X 9 = 63  
7 X 10 = 70  
7 X 11 = 77  
7 X 12 = 84

## Ciclo Foreach

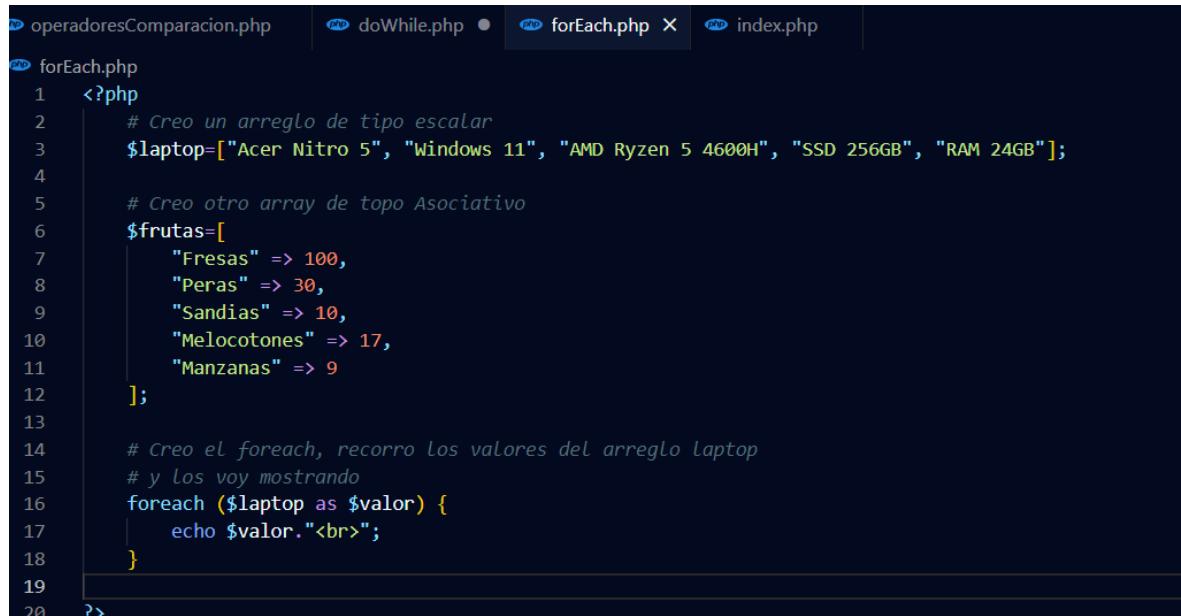
### CICLO FOREACH

El ciclo foreach proporciona un modo sencillo de **iterar sobre arrays**. foreach funciona **sólo sobre arrays** y emitirá un error al intentar usarlo con una variable de un tipo diferente de datos o una variable no inicializada. Existen dos sintaxis:

```
foreach($array as $valor){  
    $valor tendrá en cada iteración  
    un valor del array  
}
```

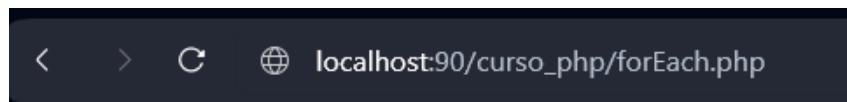
```
foreach($array as $clave => $valor){  
    $clave tendrá en cada iteración  
    una clave del array  
    $valor tendrá en cada iteración  
    un valor del array  
}
```

Aquí el nombre que puse en la variable valor en el foreach, puede ir el nombre que queramos nosotros, esa variable en este caso valor, va a ir tomando el valor de cada valor que se vaya encontrando en el arreglo.



```
operadoresComparacion.php doWhile.php forEach.php index.php

forEach.php
1 <?php
2     # Creo un arreglo de tipo escalar
3     $laptop=["Acer Nitro 5", "Windows 11", "AMD Ryzen 5 4600H", "SSD 256GB", "RAM 24GB"];
4
5     # Creo otro array de tipo Asociativo
6     $frutas:[
7         "Fresas" => 100,
8         "Peras" => 30,
9         "Sandias" => 10,
10        "Melocotones" => 17,
11        "Manzanas" => 9
12    ];
13
14    # Creo el foreach, recorro Los valores del arreglo laptop
15    # y Los voy mostrando
16    foreach ($laptop as $valor) {
17        echo $valor."<br>";
18    }
19
20 ?>
```



```
< > C localhost:90/curso_php/foreach.php
```

Acer Nitro 5  
Windows 11  
AMD Ryzen 5 4600H  
SSD 256GB  
RAM 24GB

Si queremos que nos muestre tanto la clave como el valor añadimos otra variable para la clave igual con el nombre que queramos dentro del foreach:

```
operadoresComparacion.php | doWhile.php | forEach.php X | index.php |  
forEach.php  
1 <?php  
2     # Creo un arreglo de tipo escalar  
3     $laptop=["Acer Nitro 5", "Windows 11", "AMD Ryzen 5 4600H", "SSD 256GB", "RAM 24GB"];  
4  
5     # Creo otro array de tipo Asociativo  
6     $frutas=[  
7         "Fresas" => 100,  
8         "Peras" => 30,  
9         "Sandias" => 10,  
10        "Melocotones" => 17,  
11        "Manzanas" => 9  
12    ];  
13  
14     # Creo el foreach, recorro Los valores del arreglo Laptop  
15     # y los voy mostrando  
16     foreach ($laptop as $clave => $valor) {  
17         echo $clave." - ".$valor."<br>";  
18     }  
19  
20 ?>
```

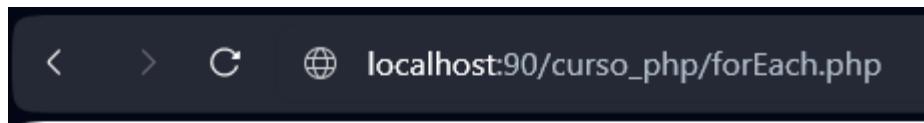
En este caso la clave son las posiciones de cada valor en el array de tipo escalar

```
< > C localhost:90/curso_php/forEach.php
```

```
0 - Acer Nitro 5  
1 - Windows 11  
2 - AMD Ryzen 5 4600H  
3 - SSD 256GB  
4 - RAM 24GB
```

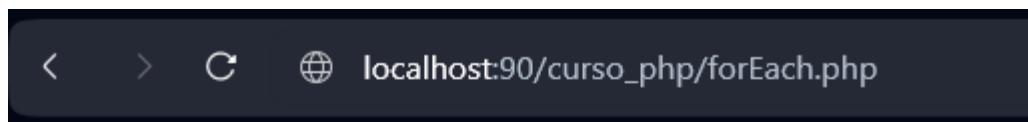
Pero esto pasa si lo hago en uno de tipo Asociativo, en este caso en mi arreglo frutas:

```
operadoresComparacion.php | doWhile.php | forEach.php X | index.php |  
forEach.php  
1 <?php  
2     # Creo un arreglo de tipo escalar  
3     $laptop=["Acer Nitro 5", "Windows 11", "AMD Ryzen 5 4600H", "SSD 256GB", "RAM 24GB"];  
4  
5     # Creo otro array de tipo Asociativo  
6     $frutas=[  
7         "Fresas" => 100,  
8         "Peras" => 30,  
9         "Sandias" => 10,  
10        "Melocotones" => 17,  
11        "Manzanas" => 9  
12    ];  
13  
14     # Creo el foreach, recorro Los valores del arreglo frutas  
15     # y los voy mostrando  
16     foreach ($frutas as $clave => $valor) {  
17         echo $clave." - ".$valor."<br>";  
18     }  
19  
20 ?>
```



```
operadoresComparacion.php doWhile.php forEach.php X index.php

forEach.php
1 <?php
2     # Creo un arreglo de tipo escalar
3     $laptop=["Acer Nitro 5", "Windows 11", "AMD Ryzen 5 4600H", "SSD 256GB", "RAM 24GB"];
4
5     # Creo otro array de tipo Asociativo
6     $frutas=[
7         "Fresas" => 100,
8         "Peras" => 30,
9         "Sandias" => 10,
10        "Melocotones" => 17,
11        "Manzanas" => 9
12    ];
13
14    # Creo el foreach, recorro los valores del arreglo frutas
15    # y los voy mostrando
16    foreach ($frutas as $valor) {
17        echo $valor."<br>";
18    }
19
20 ?>
```



100  
30  
10  
17  
9

Para un array multidimensional se puede de la siguiente manera:

```
foreach.php X
foreach.php
1 <?php
2
3 $productos = [
4     | ["codigo" => "A0001", "descripcion" => "Mouse"],
5     | ["codigo" => "A0002", "descripcion" => "Teclado"],
6     | ["codigo" => "A0003", "descripcion" => "Monitor"],
7     | ["codigo" => "A0004", "descripcion" => "Impresor"]
8 ];
9
10 foreach($productos as $prod){
11     echo $prod["codigo"]." - ".$prod["descripcion"]."<br>";
12 }
```

The screenshot shows a browser window with the URL `localhost/curso_php/foreach.php`. The page displays four lines of text, each consisting of a product code followed by a dash and its description, separated by a new line (`<br>`). The output is:

A0001 - Mouse  
A0002 - Teclado  
A0003 - Monitor  
A0004 - Impresor

Como detener un ciclo:

Esto se puede lograr con un **break** y una **condición** dentro de mi ciclo:

```
break_continue.php ●
break_continue.php
1 <?php
2
3 $c=1;
4 while($c<=20){
5     echo $c."<br>";
6     if($c==10){
7         break;
8     }
9     $c++;
10 }
```

Aquí mi foreach se va a detener cuando encuentre el valor GPU:

```
break_continue.php
<?php
$pc=[ "SO", "SSD", "GPU", "RAM", "CPU"];
foreach($pc as $componente){
    echo $componente."<br>";
    if($componente=="GPU"){
        break;
    }
}
```

Ahora si usamos continue lo que va a pasar es que ya no va a ejecutar el resto de código que hay después, sino que va a pasar al siguiente valor del array.

```
break_continue.php
<?php
$pc=[ "SO", "SSD", "GPU", "RAM", "CPU"];
foreach($pc as $componente){
    if($componente=="GPU"){
        continue;
    }
    echo $componente."<br>";
}
```

Por esa razón se muestran todos los valores menos GPU porque se omitió el echo que tenía.

The browser window shows the output of the PHP script. The address bar indicates the URL is `localhost/curso_php/break_continue.php`. The page content displays the following text:  
SO  
SSD  
RAM  
CPU

Lo mismo puede pasar en otro ciclo:

```
break_continue.php
```

```
<?php
for($i=1; $i<=10; $i++){
    if($i==5){
        continue;
    }
    echo $i."<br>";
}
```

```
break_continue.php
```

```
<?php
$i=1;
while($i<=10){
    if($i==5){
        $i++;
        continue;
    }
    echo $i."<br>";
    $i++;
}
```

## INCLUDE Y REQUIRE para incluir archivos con código a otro archivo

### INCLUDE & REQUIRE

Ambas funciones sirven para añadir otros ficheros a nuestros scripts en PHP.

**include:** inserta en nuestro script un código procedente de otro archivo, si no existe dicho archivo o si contiene algún tipo de error nos mostrará un '**warning**' por pantalla y **el script seguirá ejecutándose**.

**require:** hace la misma operación que include, pero en caso de no existir el archivo o error en el mismo mostrará un '**fatal error**' y el script **no se sigue ejecutando**.

### INCLUDE, INCLUDE\_ONCE, REQUIRE & REQUIRE\_ONCE

```
include("ruta_archivo.php");
include "ruta_archivo.php";
```

```
require("ruta_archivo.php");
require "ruta_archivo.php";
```

```
include_once("ruta_archivo.php");
include_once "ruta_archivo.php";
```

```
require_once("ruta_archivo.php");
require_once "ruta_archivo.php";
```

En este caso estoy incluyendo a mi archivo llamado include.php, otro archivo llamado for.php usando **include**, si el archivo esta dañado o marca errores el script de **include** continuara ejecutándose de todos modos.

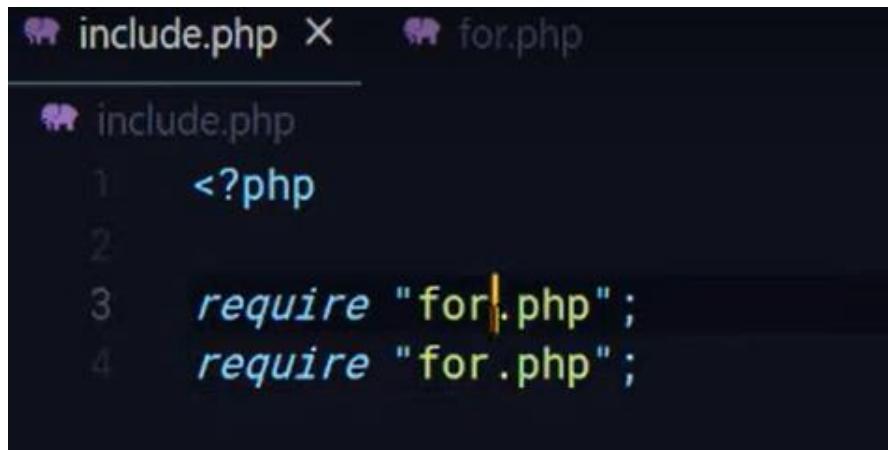
```
<?php
include "for.php";?>
```

Puedo incluir el mismo archivo varias veces, y se ejecutaran en mi archivo principal la cantidad de veces que yo lo haya incluido.



```
include.php X for.php
include.php
1 <?php
2
3 include "for.php";
4 include "for.php";
```

Aquí un ejemplo de lo mismo pero con require. La diferencia es que si mi archivo tiene algún error o coloque mal su nombre, o no se encuentra el archivo debido que no existe, todo mi script dejara de correr y me marcara un error, lo cual es lo contrario al usar include.



```
include.php X for.php
include.php
1 <?php
2
3 require "for|.php";
4 require "for.php";
```

Tambien puedo ocupar sus variantes, si ocupo \_once entonces aunque yo coloque 2 veces el archivo, solo se va a incluir 1 sola vez y si ya esta incluido, YA NO se va a volver a incluir.

The screenshot shows a code editor with two tabs: "include.php" and "for.php". The "include.php" tab is active, displaying the following code:

```
1 <?php
2
3 require_once "for.php";
4 require_once "for.php";
```

En html puede funcionar de la siguiente manera:

The screenshot shows a code editor with several tabs: "index.php", "index2.php" (which is the active tab), "footer.php", and "nav.php". The "index2.php" tab displays the following HTML code with PHP includes:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Document</title>
8 </head>
9 <body>
10  <?php include_once "./inc/nav.php"; ?>
11
12  <h1>Pagina secundaria</h1>
13
14  <?php include_once "inc/footer.php"; ?>
15 </body>
16 </html>
```

Lo que hago es incluir por ejemplo el archivo que tengo llamado footer.php una sola vez:

The screenshot shows a code editor with four tabs: "index.php", "index2.php", "footer.php" (the active tab), and "nav.php". The "footer.php" tab contains the following simple code:

```
1 <footer>
2   pie de pagina modificado
3 </footer>
```

## Funciones propias

### FUNCIONES (PROPIAS)

Una función es un **conjunto de instrucciones** a la que podemos **recurrir siempre que queramos**. Éstas **pueden recibir parámetros** y realizar todo tipo de tareas, ya sean complejas o sencillas.

Un **nombre de función válido** comienza con una **letra o guión bajo**, seguido de cualquier **número de letras, números o guiones bajos**.

```
function nombre_de_función($parametros){  
    Código de la función  
}
```

Aquí creo una función muy sencilla y la mando a llamar para que se ejecute lo que hay dentro de la función.



```
funciones.php ●  
funciones.php  
<?php  
  
function saludo(){  
    echo "Hola, mi nombre es: Carlos";  
}  
  
7   saludo();|
```

Y aquí un ejemplo de una función con un parámetro:



```
funciones.php ●  
funciones.php  
<?php  
  
function saludo($nombre){  
    return "Hola, mi nombre es: $nombre";  
}  
  
echo saludo("Nicole");  
  
$usuario="Ashley";  
10  echo saludo($usuario);
```

Aquí otro ejemplo con 3 parámetros para sacar un promedio con una función:

```
funciones.php X
funciones.php
<?php

function promedio_alumno($nota_1,$nota_2,$nota_3){
    $promedio=($nota_1+$nota_2+$nota_3)/3;
    return $promedio;
}

$promedio=promedio_alumno(7,9,6);
echo "El promedio es: ".$promedio;
```

localhost/curso\_php/funciones.php

El promedio es: 7.3333333333333

### Como incluir y llamar una función desde otro archivo php:

Regresando con el ejemplo anterior, aquí tenemos un archivo llamado **funciones.php**, en donde tenemos una función que calcula promedios para alumnos con 3 calificaciones.

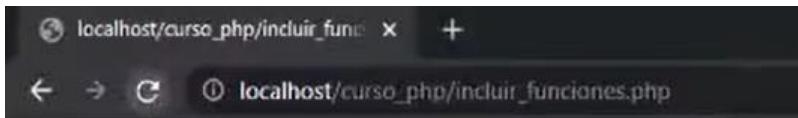
```
funciones.php ● incluir_funciones.php
funciones.php
<?php

3 function promedio_alumno($nota_1,$nota_2,$nota_3){
    $promedio=($nota_1+$nota_2+$nota_3)/3;
    return $promedio;
}
```

El archivo **funciones.php** donde tengo mi función lo voy a incluir a mi archivo llamado **incluir\_funciones.php** con **include**, y ya puedo usar la función de otro archivo:

```
funciones.php incluir_funciones.php
incluir_funciones.php
<?php
include "funciones.php";
5 echo "El promedio es: ".promedio_alumno(7,3,9);
```

Y al ejecutar el archivo de incluir\_funciones.php observaremos que ya puedo utilizar la función correctamente, solo tengo que pasarle los parámetros:



El promedio es: 6.3333333333333

De esta forma ponemos la ruta en caso de que el archivo que queremos incluir se encuentre dentro de una carpeta: En este caso accedemos a la carpeta funciones y después al archivo llamado funciones.php

```
incluir_funciones.php
incluir_funciones.php
<?php
3 require "./funciones/funciones.php";
echo "El promedio es: ".promedio_alumno(7,3,9);
```

## Funciones predefinidas de PHP para Convertir una cadena de texto String a MAYUSCULAS o minúsculas:

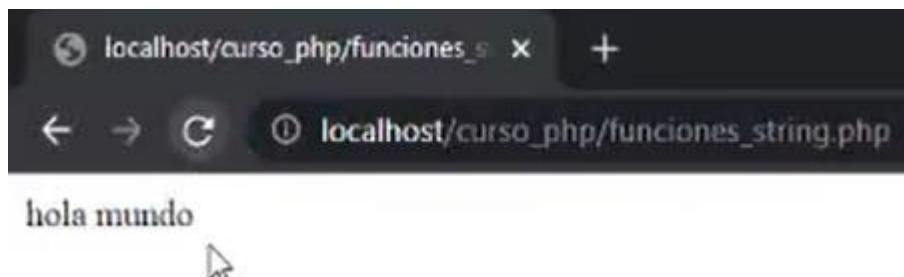
En este caso tenemos un **Hola Mundo** almacenada en la variable llamada `cadena_texto` y la mostramos con un echo usando la función ya predefinida por php llamada `strtolower()` que es **para convertir TODO lo que contiene una cadena a minúsculas**. En este caso la H y la M.

```
funciones_string.php ●  
funciones_string.php  
1 <?php  
  
3 $cadena_texto="Hola Mundo";  
  
5 echo strtolower($cadena_texto);
```

O también de esta manera (es lo mismo) pero diferente forma que la anterior.

```
funciones_string.php ●  
funciones_string.php  
1 <?php  
  
3 $cadena_texto="Hola Mundo";  
  
5 $cadena_texto=strtolower($cadena_texto);  
  
7 echo $cadena_texto;
```

Como resultado me muestra así la cadena en pura minúscula:



localhost/curso\_php/funciones\_string.php

hola mundo

Y también tenemos la siguiente función `strtoupper()` para convertir todo el contenido de una cadena a letras mayúsculas:

```
funciones_string.php ●
funciones_string.php
1 <?php
2
3 $cadena_texto="Hola Mundo";
4
5 $cadena_texto=strtoupper($cadena_texto);
6
7 echo $cadena_texto;
```

```
localhost/curso_php/funciones_s X +
← → C ① localhost/curso_php/funciones_string.php
```

HOLA MUNDO

Con la función **ucfirst()** convertimos solo la primera letra del String a mayúscula:

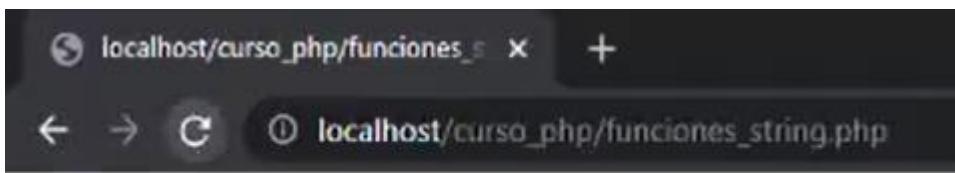
```
funciones_string.php ●
funciones_string.php
1 <?php
2
3 $cadena_texto="hola mundo";
4
5 $cadena_texto=ucfirst($cadena_texto);
6
7 echo $cadena_texto;
```

```
localhost/curso_php/funciones_s X +
← → C ① localhost/curso_php/funciones_string.php
```

Hola mundo

Y con la función **ucwords()** convertimos la primera letra de cada palabra en mayúscula:

```
funciones_string.php ●
funciones_string.php
1 <?php
2
3 $cadena_texto="hola mundo";
4
5 $cadena_texto=ucwords($cadena_texto);
6
7 echo $cadena_texto;
```



Hola Mundo



Funciones predefinidas de PHP para contar caracteres o palabras de un String:

La función **strlen()** nos permite contar cuantos caracteres tiene una cadena de texto.

```
funciones_string.php X
funciones_string.php
<?php

$cadena_texto="hola mundo";

$longitud=strlen($cadena_texto);

7 echo $cadena_texto." tiene ".$longitud." caracteres";
```

localhost/curso\_php/funciones\_string.php

hola mundo tiene 10 caracteres

También tenemos la función **str\_word\_count()** que nos dice cuantas PALABRAS hay en un String.

```
funciones_string.php
<?php
$cadena_texto="hola mundo";
$longitud=strlen($cadena_texto);
echo $cadena_texto." tiene ".$longitud." caracteres <br>";
$palabras=str_word_count($cadena_texto);
echo $cadena_texto." tiene ".$palabras." palabras <br>";
```

localhost/curso\_php/funciones\_string.php

hola mundo tiene 10 caracteres

hola mundo tiene 2 palabras

Como convertir un String en un Array en php:

```
string_array.php
<?php
$fecha_1="2021/11/29";
$fecha_2="2021-11-30";
$numeros="Uno Dos Tres Cuatro Cinco Seis Siete";
explode(delimitador,string,limitador);
```

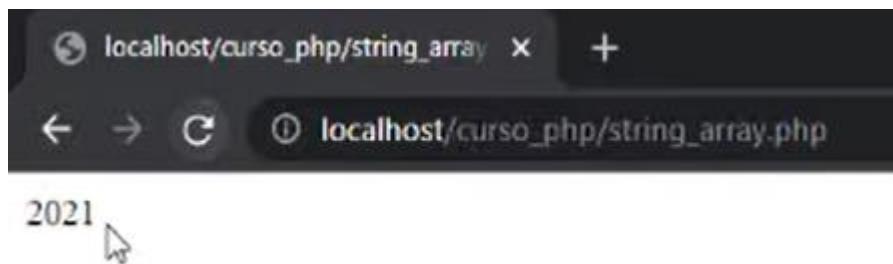
En este ejemplo tenemos 3 variables almacenando diferentes cadenas de texto que quiero convertir a un array, esto se hace con la función **explode()** en donde tengo que poner el (delimitador) que es el carácter por el que vamos a separar cada elemento del array y el (string o cadena de texto) a fuerza, y si quiero también puedo poner un (limitador) pero es opcional.

En este ejemplo convertimos la cadena 2021/11/29 en un array, ya que pusimos en el explode que el delimitador es el símbolo de slash “ / ”. Ahora 2021 está en la posición 0 del índice del array, 11 en la posición 1 y 29 en la posición 2 de mi array.

```
string_array.php
<?php
$fecha_1="2021/11/29";
$fecha_2="2021-11-30";
$numeros="Uno Dos Tres Cuatro Cinco Seis Siete";

$array_fecha=explode("/", $fecha_1);

echo $array_fecha[0];
```

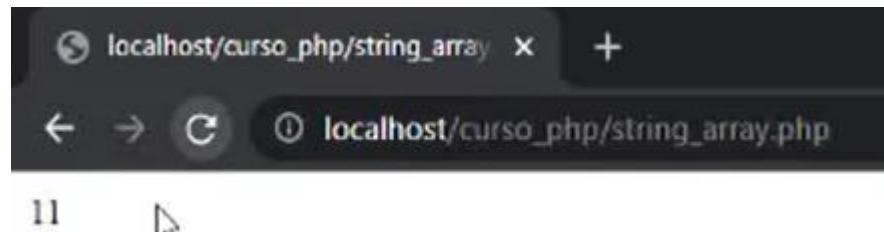


Aquí en este ejemplo hago lo mismo pero con la siguiente cadena y ahora el delimitador es el símbolo de guion “ – ”.

```
string_array.php
<?php
$fecha_1="2021/11/29";
$fecha_2="2021-11-30";
$numeros="Uno Dos Tres Cuatro Cinco Seis Siete";

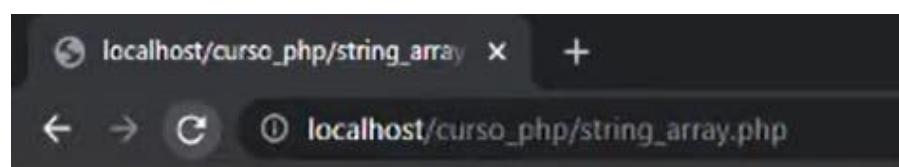
$array_fecha=explode("-", $fecha_2);

echo $array_fecha[1];
```



Aquí en este ejemplo hago lo mismo pero con la siguiente cadena y ahora el delimitador es el símbolo de un espacio “ ” por cada palabra o caracteres que quiero separar para mi array.

```
string_array.php
string_array.php
<?php
$fecha_1="2021/11/29";
$fecha_2="2021-11-30";
$ numeros="Uno Dos Tres Cuatro Cinco Seis Siete";
$array_numeros=explode(" ",$ numeros);
echo $array_numeros[6];
```



Si quiero también puedo poner OPCIONALMENTE un limitador, es decir a cantidad en la que quiero que se divida mi cadena de texto para el string. En este ejemplo divido mi cadena de texto de la variable números en 2 partes, Uno esta en el índice 0, y el resto de la cadena en el índice 1.

```
string_array.php
string_array.php
<?php
$fecha_1="2021/11/29";
$fecha_2="2021-11-30";
$ numeros="Uno| Dos Tres Cuatro Cinco Seis Siete";
$array_numeros=explode(" ",$ numeros,2);
echo $array_numeros[0];
```

A screenshot of a web browser window. The address bar shows the URL `localhost/curso_php/string_array`. The main content area displays a code editor with the file `string_array.php`. The code is as follows:

```
<?php  
$fecha_1="2021/11/29";  
$fecha_2="2021-11-30";  
$numeros="Uno Dos Tres Cuatro Cinco Seis Siete";  
$array_numeros=explode(" ",$numeros,2);  
echo $array_numeros[1];
```

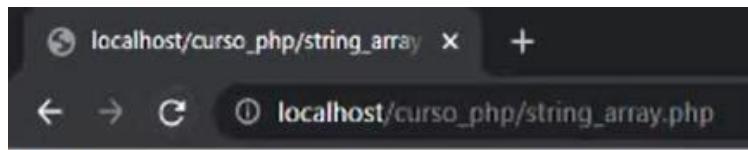
A screenshot of a web browser window. The address bar shows the URL `localhost/curso_php/string_array`. The main content area displays the output of the PHP script, which is the string "Dos Tres".

Ahora si ponemos números negativos en el limitador, por ejemplo -1, vamos a decir que haga todos los valores array excepto el ultimo, ese no se va a incluir en el array. En este caso Uno esta en el índice 0, Dos en el índice 1, etc...

Y el ultimo elemento de la cadena de texto (Siete) no está en el array, es decir que solo se creo un array con índice 5, dentro de ese índice esta (Seis).

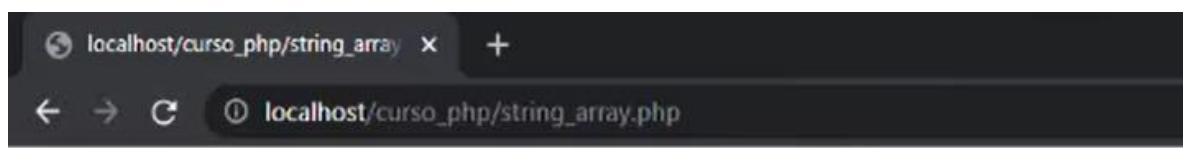
A screenshot of a web browser window. The address bar shows the URL `localhost/curso_php/string_array`. The main content area displays a code editor with the file `string_array.php`. The code is as follows:

```
<?php  
$fecha_1="2021/11/29";  
$fecha_2="2021-11-30";  
$numeros="Uno Dos Tres Cuatro Cinco Seis Siete";  
$array_numeros=explode(" ",$numeros,-1);  
echo $array_numeros[5];
```



Si colocamos el 6 pues nos mandara un error ya que en mi arreglo NO hay índice 6.

```
string_array.php X
string_array.php
1 <?php
2
3 $fecha_1="2021/11/29";
4 $fecha_2="2021-11-30";
5 $numeros="Uno Dos Tres Cuatro Cinco Seis Siete";
6
7 $array_numeros=explode(" ",$numeros,-1);
8
9 echo $array_numeros[6];
```



Warning: Undefined array key 6 in C:\laragon\www\curso\_php\string\_array.php on line 9

## Funciones matemáticas en php:

Solo veremos algunas, por lo que para verlas todas las funciones matemáticas predefinidas de php podemos entrar a la documentación con este link:

<https://www.php.net/manual/es/book.math.php>

Funcion pow() para **elevar un número**, por ejemplo aquí elevamos 5 a la 3 y lo mostramos con echo. Esto es igual a 125.

```
funciones_matematicas.php ●
funciones_matematicas.php
1 <?php
2
3 echo pow(5,3);
```

Esta función es para sacar la **raíz cuadrada** en este caso de 9 que es igual a 3.

```
funciones_matematicas.php
```

---

```
funciones_matematicas.php
1 <?php
2
3 echo sqrt(9);
```

Función para **escoger un numero aleatorio**, en este caso entre el 1 y el 100: Puede agarrar un numero aleatorio entre números una y otra vez, cada vez que recargues escogerá uno diferente.

```
funciones_matematicas.php
```

---

```
funciones_matematicas.php
1 <?php
2
3 echo rand(1, 100);
```

Funcion para darte el valor de pi

```
funciones_matematicas.php
```

---

```
funciones_matematicas.php
1 <?php
2
3 echo pi();
```

La función floor redondea un numero con decimal a un entero hacia ABAJO

La función ceil redondea un numero con decimal a un entero hacia ARRIBA por ejemplo:

```
funciones_matematicas.php
```

---

```
funciones_matematicas.php
1 <?php
2
3 echo floor(4.3);
4 echo "<br>";
5 echo ceil(4.3);
```

```
localhost/curso_php/funciones_m.php
← → C ⓘ localhost/curso_php/funciones_m.php
4
5
```

La función round sirve para redondear un numero con decimales a entero, si tiene en sus decimales apartir del 50% hacia arriba lo redondea hacia arriba, sino lo redondea hacia abajo.

En este caso como es .5 es el 50%, por lo que el resultado será 4.

```
funciones_matematicas.php
funciones_matematicas.php
1 <?php
2
3 echo round(3.5);
```

Si a esta función yo le paso 2 parámetros separado por comas, me redondea los decimales al numero de decimales que le dije en este caso a 2, por ejemplo aquí me va a redondear 1.955 a 1.96

```
funciones_matematicas.php
funciones_matematicas.php
1 <?php
2
3 echo round(1.955,2);
```

Por ejemplo este 5.055 me reduce los decimales a 2, ósea 5.06

```
funciones_matematicas.php
funciones_matematicas.php
1 <?php
2
3 echo round(5.055,2);
```

Formatear números con un formato específico en PHP (cantidad de decimales, separadores de decimales, separadores de miles):

Esta función recibe 4 parámetros, la cantidad a formatear, los decimales que va a contener la cantidad, separador de decimales y el separador de miles es decir: 134,234,234 las comas.

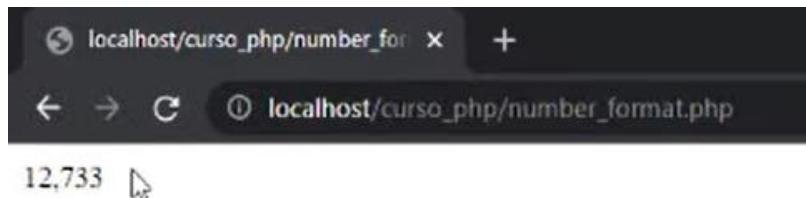
Esta función se puede utilizar únicamente con 1 parámetro (cantidad), con 2 (cantidad y decimales), o con los 4, NO con 3 parámetros tienen que ser los 4.

```
number_format(cantidad, decimales, sep_decimal, sep_millar);
```

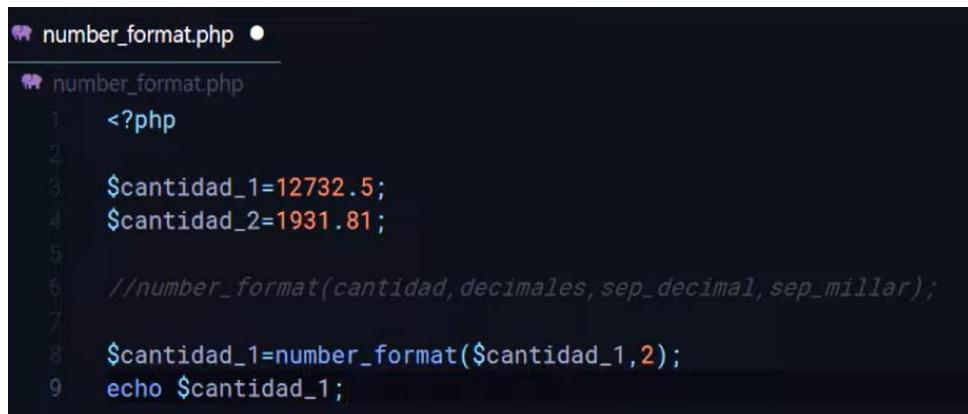
  


```
number_format.php X
number_format.php
1 <?php
2
3 $cantidad_1=12732.5;
4 $cantidad_2=1931.81;
5
6 //number_format(cantidad, decimales, sep_decimal, sep
7
8 $cantidad_1=number_format($cantidad_1);
9 echo $cantidad_1;
```

Como podemos observar le puso separador de miles y lo redondeo según corresponda:

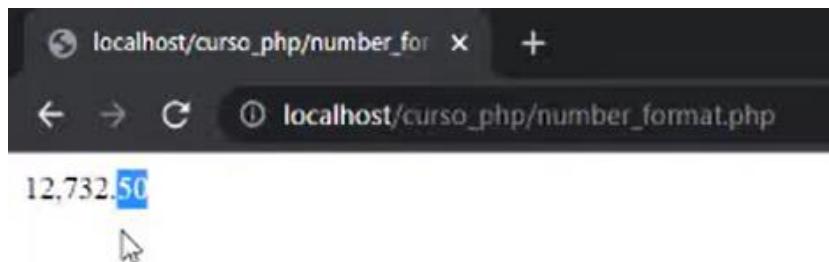


Ahora si agrego otro parámetro a la función (el numero de decimales que quiero)

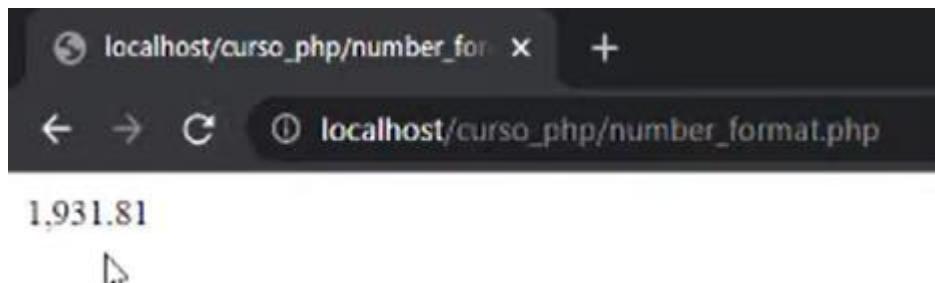
```
number_format.php ●
number_format.php
1 <?php
2
3 $cantidad_1=12732.5;
4 $cantidad_2=1931.81;
5
6 //number_format(cantidad, decimales, sep_decimal, sep_millar);
7
8 $cantidad_1=number_format($cantidad_1,2);
9 echo $cantidad_1;
```

Usa el punto como separador de decimales y la coma como separador de miles, además de mostrar la cantidad de decimales que le indique con el segundo parámetro que mande a la función.



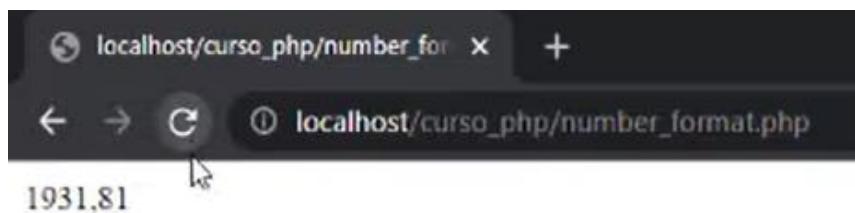
También podemos enviar los 4 parámetros, dependiendo del país la forma de separar los decimales y los miles puede ser diferente, entonces tu ya eliges con que símbolo separar cada uno.

```
number_format.php
number_format.php
<?php
$cantidad_1=12732.5;
$cantidad_2=1931.81;
//number_format(cantidad, decimales, sep_decimal, sep_millar);
$cantidad_2=number_format($cantidad_2,2,".",",");
echo $cantidad_2;
```



O si solo quiero por ejemplo separador de decimales, pero no quiero de miles, lo dejo vacío ese parámetro:

```
number_format.php
number_format.php
<?php
$cantidad_1=12732.5;
$cantidad_2=1931.81;
//number_format(cantidad, decimales, sep_decimal, sep_millar);
$cantidad_2=number_format($cantidad_2,2,".",",");
echo $cantidad_2;
```



Obtener fecha y hora actual de cualquier lugar con la función date:

La fecha que obtienes con esta función está en inglés, puedes consultar su documentación aquí:

<https://www.php.net/manual/es/function.date.php>

ANTES de usar la función date, debemos de definir la zona horaria con la función date\_default\_timezone\_set

Podemos ver el listado de zonas horarias admitidas en este link de la documentación, de la cual tenemos que buscar ahí la zona horaria que queremos:

<https://www.php.net/manual/es/timezones.php>

Ahora si ya puedo ocupar la función date: En este caso quiero el dia en el que se encuentran en El Salvador, por eso coloco la d

```

date.php
<?php
    date_default_timezone_set("America/El_Salvador");
5 echo date("d");

```

Como lo dice en la documentación, que colocando d, me da el día:

Los siguientes caracteres están reconocidos en el parámetro de cadena format		
Carácter de format	Descripción	Ejemplo de valores devueltos
Día	---	---
d	Día del mes, 2 dígitos con ceros iniciales	01 a 31

Como resultado me da el día en el que se encuentran los salvadoreños en este preciso momento:



También se puede ocupar mas de una letra, por ejemplo aquí coloco **Y** para que me de él año en 4 dígitos, un guion para separar solamente (**también puedo separar con espacios u otro símbolo que yo quiera**), y **m** para que me de el mes en 2 dígitos numéricos, y **d** para que me de él día, estas letras clave se sacan de la documentación. <https://www.php.net/manual/es/function.date.php>

```
date.php
date.php
<?php

date_default_timezone_set("America/El_Salvador");

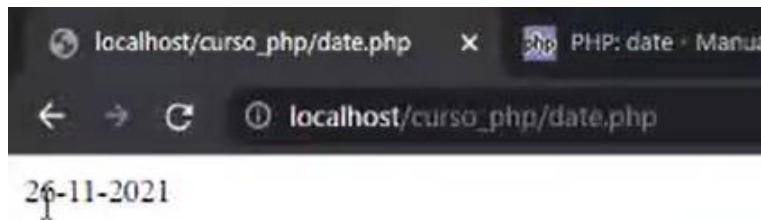
$fecha_us=date("Y/m/d");
$fecha_es=date("d-m-Y");
echo $fecha_es;
```

A screenshot of a code editor window titled "date.php". The code inside the editor is:

```
<?php

date_default_timezone_set("America/El_Salvador");

$fecha_us=date("Y/m/d");
$fecha_es=date("d-m-Y");
echo $fecha_es;
```



## Obtener fecha y hora actual (EN ESPAÑOL) de cualquier lugar con la función date:

Para esto necesito crear una función que servirá para poner la fecha completa en español, dentro ocupo 1 array de tipo asociativo, es decir clave-valor para los días, y otro igual asociativo para los meses del año.

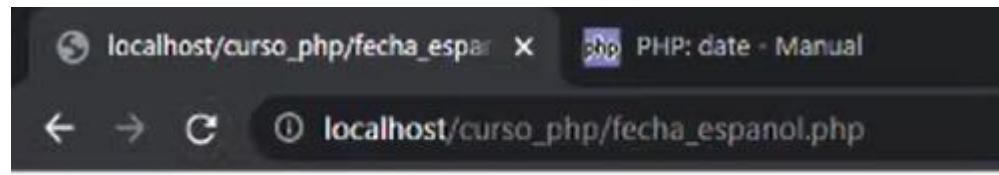


```
fecha_espanol.php
1 <?php
2 date_default_timezone_set("America/El_Salvador");
3
4 function fecha_espanol_larga(){
5
6     $fecha_dia=date("d");
7     $fecha_mes=date("m");
8     $fecha_year=date("Y");
9
10    $dia_semana=[
11        "Monday"=>"Lunes",
12        "Tuesday"=>"Martes",
13        "Wednesday"=>"Miércoles",
14        "Thursday"=>"Jueves",
15        "Friday"=>"Viernes",
16        "Saturday"=>"Sábado",
17        "Sunday"=>"Domingo"
18    ];
19
20    $meses_year=[
21        "01"=>"Enero",
22        "02"=>"Febrero",
23        "03"=>"Marzo",
24        "04"=>"Abril",
25        "05"=>"Mayo",
26        "06"=>"Junio",
27        "07"=>"Julio",
28        "08"=>"Agosto",
29        "09"=>"Septiembre",
30        "10"=>"Octubre",
31        "11"=>"Noviembre",
32        "12"=>"Diciembre"
33    ];
34
35    $fecha_final=$dia_semana[date("l")]. " ".$fecha_dia." de ".$meses_year[$fecha_mes]." de ".
36    $fecha_year;
37
38    return $fecha_final;
39 }
40 echo fecha_espanol_larga();
```

En la línea 35 dentro de la función date, coloco l porque sirve para lo siguiente según nos lo dice la documentación.

l ('L' minúscula)	Una representación textual completa del día de la semana	Sunday hasta Saturday
----------------------	--	-----------------------

Entonces al traernos por ejemplo Friday, es la clave del valor Viernes almacenado en el array asociativo de días de la semana, y en ese momento el array lo traduce por nosotros trayendo el valor que pusimos en español.



Viernes 26 de Noviembre de 2021



Como encriptar una contraseña o clave en php con hash (modos antiguos y no tan recomendables):

#### ¿Qué Es Un Hash Y Cómo Funciona?

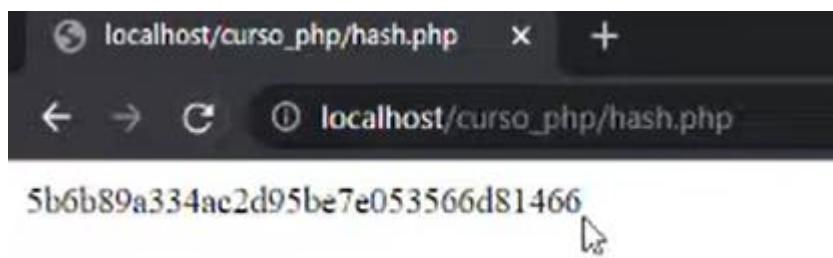
Una función criptográfica hash- usualmente conocida como “hash”- es un algoritmo matemático que transforma cualquier bloque arbitrario de datos en una **nueva serie de caracteres con una longitud fija.**

Independientemente de la **longitud de los datos de entrada**, el valor hash de salida tendrá **siempre la misma longitud**.

Hash tiene la función [md5\(\)](#) que es antigua, pero se usaba mucho ANTES, **convierte un string a otro string pero encriptado**. Por ejemplo:

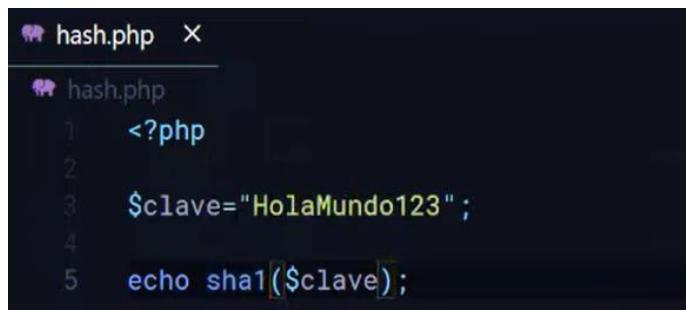
```
hash.php ●  
hash.php  
1 <?php  
2  
3 $clave="HolaMundo123";  
4  
5 echo md5($clave);
```

La encriptación **solo está conformada por números y letras**.



A screenshot of a web browser window. The address bar shows "localhost/curso\_php/hash.php". The main content area displays the output of a SHA-1 hash function: "5b6b89a334ac2d95be7e053566d81466". A cursor arrow is visible at the bottom right of the text.

También tenemos esta otra función demasiado parecida a la anterior, que encripta de la misma manera con solo números y letras:



```
hash.php
hash.php
1 <?php
2
3 $clave="HolaMundo123";
4
5 echo sha1($clave);
```

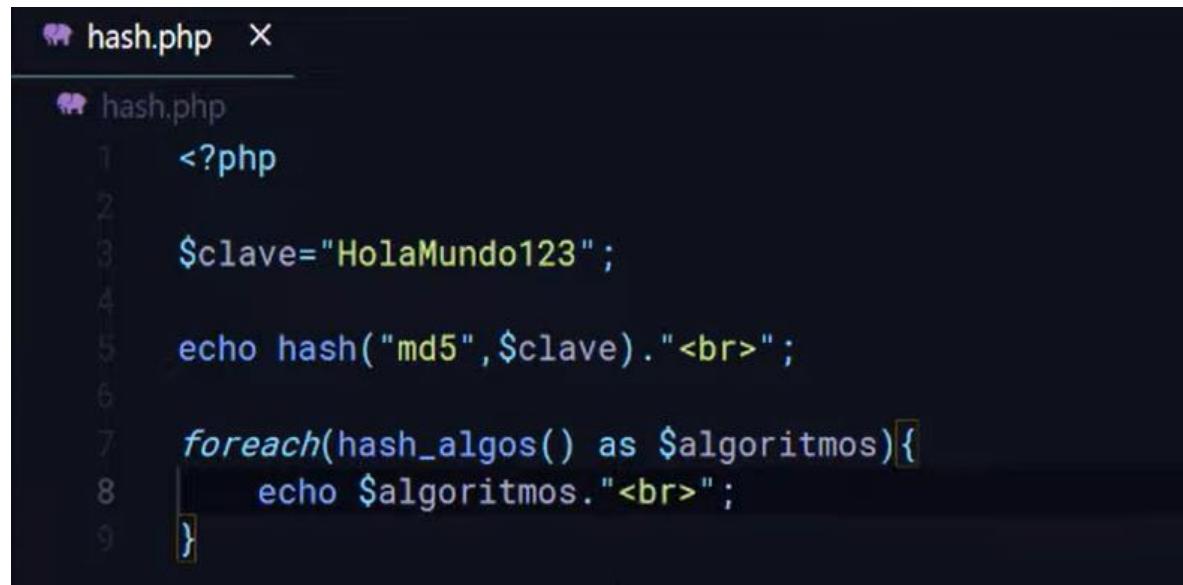
**NOTA: Sin embargo, hoy en día ya no se utilizan las funciones `md5()` ni `sha1()` debido a que hoy en día es muy fácil descifrar la encriptación de estos algoritmos.**

Estos vienen siendo algoritmos de la función hash, y existen múltiples algoritmos, todos te encriptan un string pero de manera FIJA.

Lo hecho anteriormente se puede hacer con hash, recibiendo 2 parámetros, el algoritmo que queremos usar para encriptar, y la cadena de texto que queremos encriptar. En este caso le digo que quiero ocupar la función hash con el algoritmo de encriptación md5, y que encripte lo que tiene la variable clave.

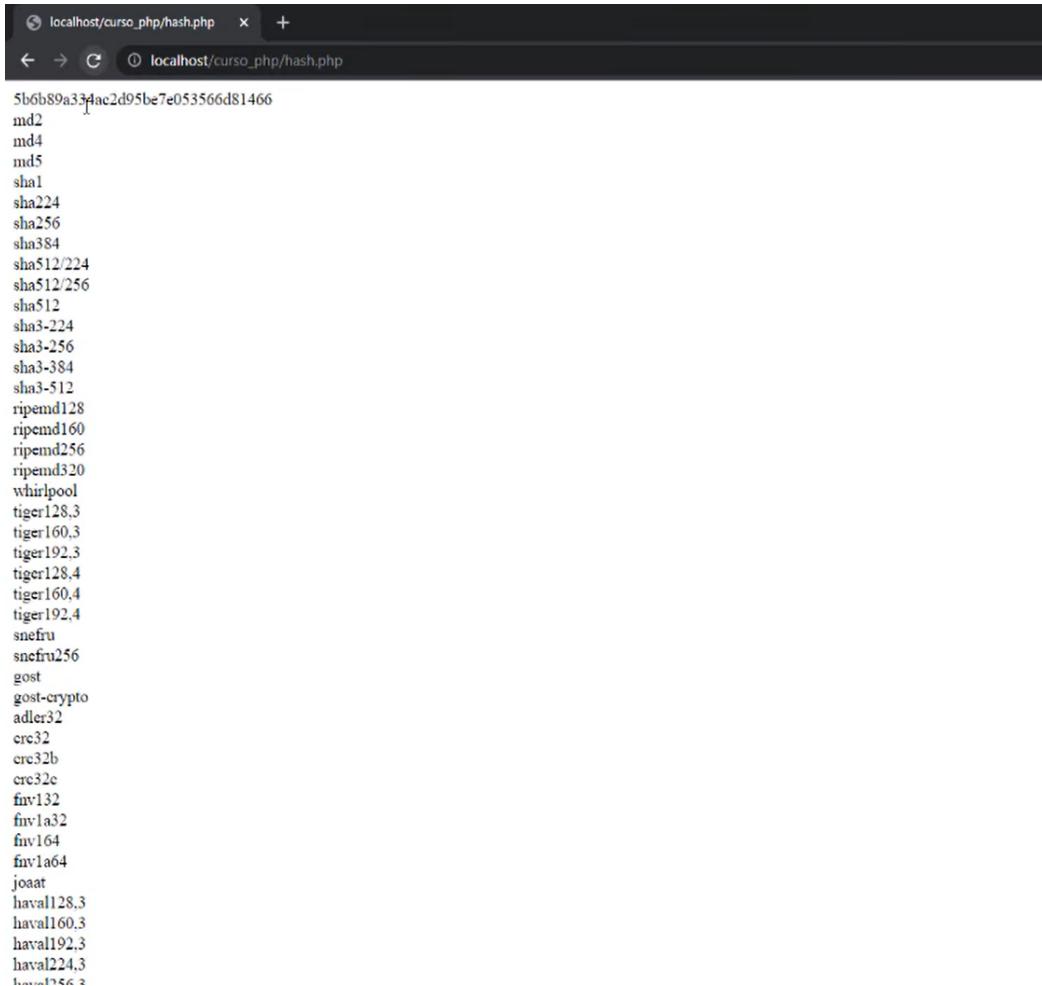
```
$clave="HolaMundo123";
echo hash("md5", $clave). "<br>";
```

Ahora para ver todos los algoritmos que hay, podemos verlos con un foreach, ya que estos algoritmos están almacenados en un array llamado `hash_algos()` de la siguiente manera:



```
hash.php  X
hash.php
<?php
$clave="HolaMundo123";
echo hash("md5",$clave)."<br>";
foreach(hash_algos() as $algoritmos){
    echo $algoritmos."<br>";
}
```

El foreach recorre todo el arreglo, mostrándonos todos los algoritmos para encriptación que almacena ese array. (hay mas)



A screenshot of a web browser window titled "localhost/curso\_php/hash.php". The page displays a long list of hash algorithms and their parameters. The list includes:

- 5b6b89a334ac2d95be7e053566d81466
- md2
- md4
- md5
- sha1
- sha224
- sha256
- sha384
- sha512/224
- sha512/256
- sha512
- sha3-224
- sha3-256
- sha3-384
- sha3-512
- ripemd128
- ripemd160
- ripemd256
- ripemd320
- whirlpool
- tiger128,3
- tiger160,3
- tiger192,3
- tiger128,4
- tiger160,4
- tiger192,4
- snefru
- snefru256
- gost
- gost-crypto
- adler32
- crc32
- crc32b
- crc32c
- fnv132
- fnv1a32
- fnv164
- fnv1a64
- joaat
- haval128,3
- haval160,3
- haval192,3
- haval224,3

Y si queremos ver como quedan las encriptaciones, podemos hacerlo igual con el foreach



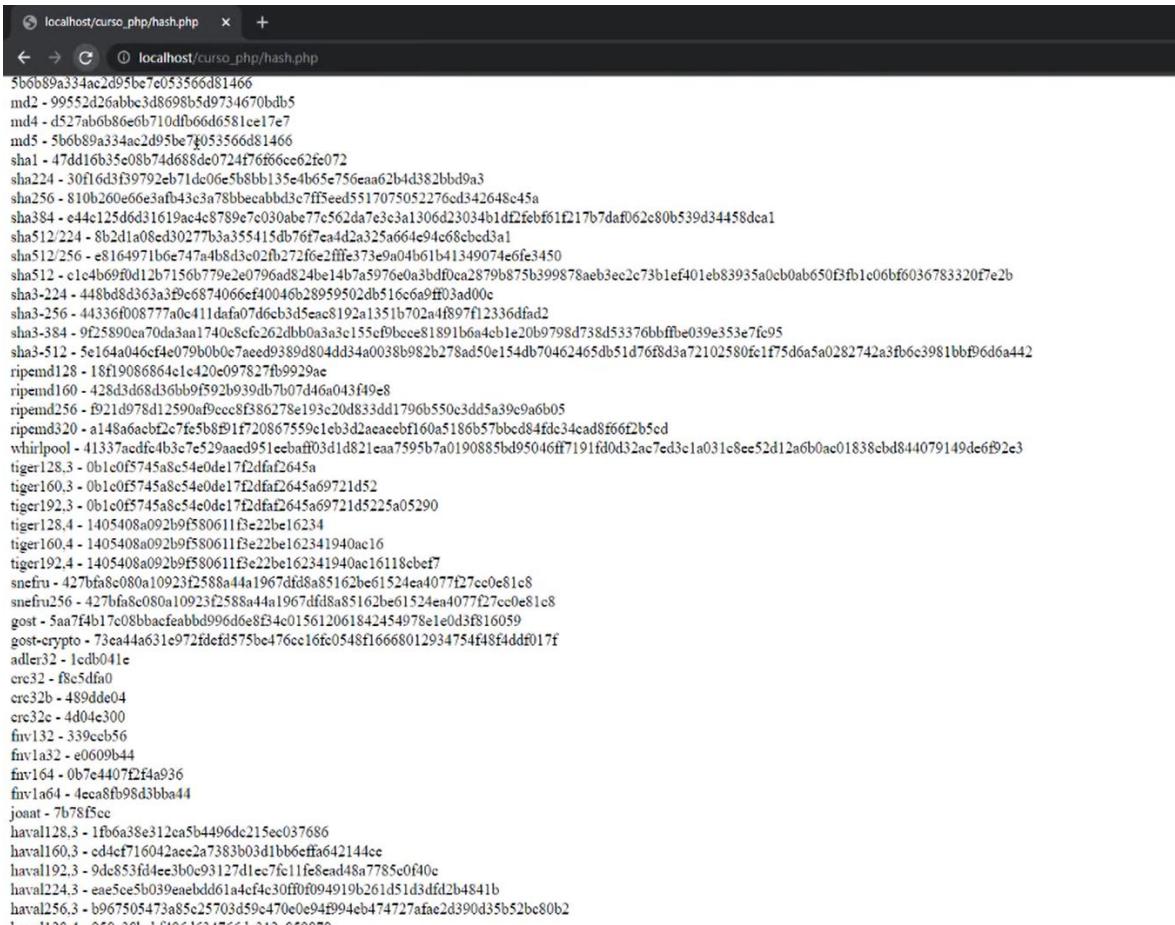
```
hash.php X
hash.php
<?php

$clave="HolaMundo123";

echo hash("md5",$clave)."<br>";

foreach(hash_algos() as $algoritmos){
    echo $algoritmos." - ".hash($algoritmos,$clave)."<br>";
}
```

Ahora aquí vemos el algoritmo y como encriptan el string cada uno. SIN EMBARGO las encriptaciones son fijas, no cambian si las hacemos varias veces con EL MISMO string, ya que es un patron ya definido el que viene en cada algoritmo.



The screenshot shows a browser window with the URL `localhost/curso.php/hash.php`. The page displays a very long string of encrypted data, starting with `5b6b89a334ac2d95be7e05356d81466` and ending with `0f4eb4747af7b7a1f10356d81466`. The string contains numerous SHA-1 hash values separated by newlines, such as `sha224 - 47dd16b35e08b74d688de07247f76f6cc62fe072`, `sha224 - 5b6b89a334ac2d95be7e05356d81466`, and `sha224 - 810b260e66e3ab43c378bbecabbd3c7ffeed5517075052276cd342648e45a`.

Para que no devuelva un valor fijo siempre, sino que la encriptación sea diferente siempre con cualquier string tenemos la siguiente más moderna función:

Como encriptar una contraseña o clave en php con hash (la manera más recomendable):

Para esto tenemos una función llamada `password_hash()` que recibe 2 parámetros, el string que se va a encriptar y el algoritmo que se va a usar, para ver los algoritmos que admite esta función hash mas moderna podemos ir a la documentación:

<https://www.php.net/manual/es/function.password-hash.php>

Por ejemplo nos da los siguientes algoritmos:

- **PASSWORD\_DEFAULT** - Usar el algoritmo bcrypt (predeterminado a partir de PHP 5.5.0). Observe que esta constante está diseñada para cambiar siempre que se añade un algoritmo nuevo y más fuerte a PHP. Por esta razón, la longitud del resultado de usar este identificador puede cambiar con el tiempo. Por lo tanto, se recomienda almacenar el resultado en una columna de una base de datos que pueda apliarse a más de 60 caracteres (255 caracteres sería una buena elección).
  - **PASSWORD\_BCRYPT** - Usar el algoritmo **CRYPT\_BLOWFISH** para crear el hash. Producirá un hash estándar compatible con [crypt\(\)](#) utilizando el identificador "\$2y\$". El resultado siempre será un string de 60 caracteres, o **false** en caso de error.
- Opciones admitidas:

En este caso utilizo el algoritmo **PASSWORD\_DEFAULT** pero la diferencia es que siempre va a cambiar la manera en que encripta mi String, si yo refresco la pagina varias veces, observare que la encriptación siempre cambia, aunque sea el mismo String que estoy encriptando, cosa que no era así con los anteriores algoritmos antiguos, por eso es más recomendable la función **password\_hash()**.

```
<?php
$clave='HolaMundo123';
echo password_hash($clave,PASSWORD_DEFAULT);
```

También puedo mandarle un tercer parámetro si quiero para hacer mas fuerte la encriptación, el default es 10 pero aquí lo cambio a 11, si yo no mando este tercer parámetro a la función **password\_hash** mantiene el 10.

```
<?php
$clave='HolaMundo123';
echo password_hash($clave,PASSWORD_BCRYPT,[ "cost"=>11]);
```

Ahora con la función **password\_verify()** verifica, comparando la clave original con la encriptada, y si es igual nos da un valor booleano, false o true en caso de que si coincidan. En este caso muestra un 1 que significa true ya que si coinciden.

```
<?php
$clave='HolaMundo123';
$clave_procesada=password_hash($clave,PASSWORD_BCRYPT,[ "cost"=>11]);
echo password_verify($clave,$clave_procesada);
```

Esto puede servirnos ya que podemos ocuparlo en una condición if



```
<?php  
$clave="HolaMundo123";  
$clave_procesada=password_hash($clave,PASSWORD_BCRYPT,[ "cost"=>11]);  
if(password_verify($clave,$clave_procesada)){  
    echo "Las claves coinciden";  
}
```

## Como enviar formularios con el método get y post en php:

Para dar una explicación de esto, tenemos el siguiente formulario sencillo con html:



```
<!DOCTYPE html>  
<html lang="es">  
  <head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
  </head>  
  <body>  
    <form>  
      <div>  
        <label for="nombre">Nombre</label>  
        <input type="text" id="nombre" name="nombre">  
      </div>  
  
      <br>  
  
      <label for="asignatura">Asignatura</label>  
      <select id="asignatura" name="asignatura" >  
        <option value="Ingles">Ingles</option>  
        <option value="Matematicas">Matemáticas</option>  
        <option value="Ciencia">Ciencia</option>  
        <option value="Lenguaje">Lenguaje</option>  
      </select>  
  
      <br><br>
```

Como observamos necesitamos tener en el formulario un **botón de tipo submit** para poder enviar los datos y **TODOS los input de un formulario deben de tener el atributo name** como se muestra en este formulario de ejemplo, es importante porque mediante este nombre nosotros podemos recuperar estos datos que enviamos mediante el método get o el método post

```
24         <option value="Lenguaje">Lenguaje</option>
25     </select>
26
27     <br><br>
28
29     <label for="opcion-1">
30         <input type="checkbox" value="Manzana" id="opcion-1" name="frutas">
31         Manzana
32     </label>
33
34     <br><br><br>
35
36     <button type="submit" >Enviar</button>
37
38 </form>
39
40 </body>
41 </html>
```

Anteriormente no la tenía, pero es importante que nuestra etiqueta form de apertura tenga el atributo action, en donde vamos a colocar la dirección del archivo a donde queremos que se envíen los datos de nuestro formulario, en este caso queremos enviar los datos a otro archivo llamado index.php:

```
formulario.php • index.php
formulario.php
 7     <title>Document</title>
 8 </head>
 9 <body>
10
11     <form action="index.php" >
12         <div>
13             <label for="nombre">Nombre</label>
```

Dentro del form también va un segundo atributo que es method y dentro podemos poner GET o POST

```
<form action="index.php" method="POST" >
    <div>
        <label for="nombre">Nombre</label>
        <input type="text" id="nombre" name="nombre">
```

ó

```
<form action="index.php" method="GET" >
    <div>
        <label for="nombre">Nombre</label>
        <input type="text" id="nombre" name="nombre">
```

Lo siguiente es que en el archivo a donde queremos enviar los datos coloquemos una variable para almacenar los datos y coloquemos `$_POST['']` y dentro de corchetes el nombre (name) que le pusimos al input de donde queremos extraer los datos en el html.

```
<form action="index.php" method="POST" >
  <div>
    <label for="nombre">Nombre</label>
    <input type="text" id="nombre" name="nombre">
  </div>
```

The screenshot shows a code editor with two tabs: 'formulario.php' and 'index.php'. The 'index.php' tab is active, displaying the following PHP code:

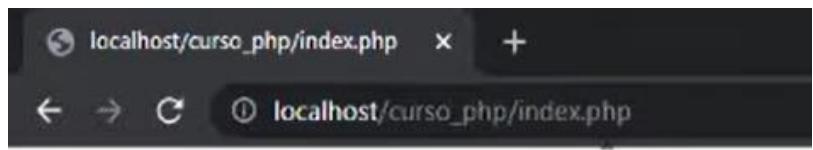
```
1 <?php
2
3 $nombre=$_POST['nombre'];
4 echo $nombre;
```

Ahora si yo lleno mi formulario en ese input text con el **name** de nombre, y le doy clic a mi botón de enviar

The screenshot shows a web browser window with the URL 'localhost/curso\_php/formulario.php'. The page displays a form with the following fields:

- Nombre:
- Asignatura:
- Manzana
- 

Me redirecciona al archivo a donde quiero enviar los datos



Ahora si yo hago lo mismo con todos mis name del formulario, quedaría de la siguiente forma:

```
formulario.php index.php  
index.php  
1 <?php  
2  
3 $nombre=$_POST['nombre'];  
4 $asignatura=$_POST['asignatura'];  
5 $frutas=$_POST['frutas'];  
6  
7 echo $nombre." - ".$asignatura." - ".$frutas;|
```

Document × +

localhost/curso\_php/formulario.php

Nombre

Asignatura

Manzana

localhost/curso\_php/index.php × +

localhost/curso\_php/index.php

Carlos - Lenguaje - Manzana

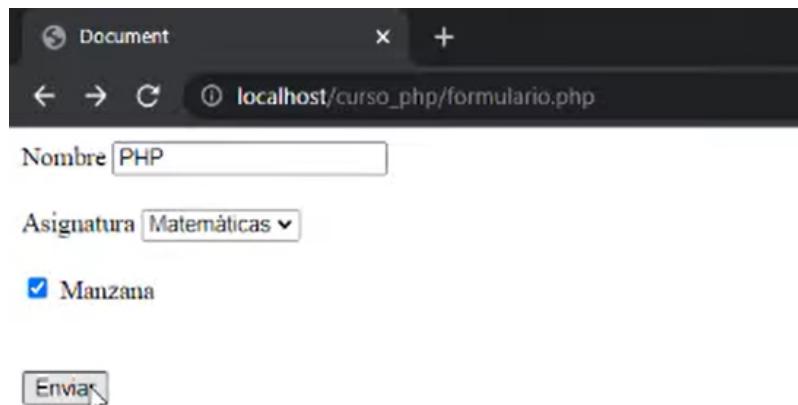
Ahora si yo lo hago por medio del método GET tengo que especificarlo en la etiqueta form de mi formulario en html.

```
<body>  
  
<form action="index.php" method="GET" >  
  <div>  
    <label for="nombre">Nombre</label>  
    <input type="text" id="nombre" name="nombre">
```

Y para eso se ocupa ahora `$_GET[' ']` poniendo dentro los name de los input



```
formulario.php index.php X
index.php
1 <?php
2
3 $nombre=$_GET['nombre'];
4 $asignatura=$_GET['asignatura'];
5 $frutas=$_GET['frutas'];
6 [?] $_GET
7 echo $nombre." - ".$asignatura." - ".$frutas;
```

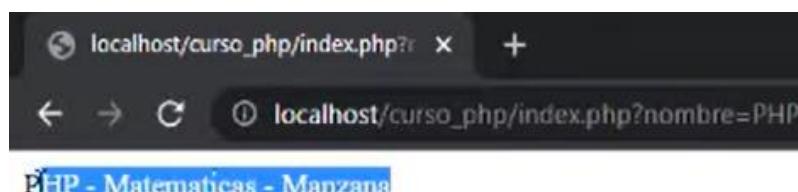


Document x +  
localhost/curso\_php/formulario.php

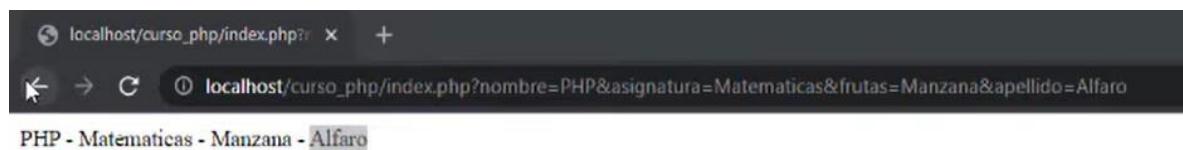
Nombre

Asignatura

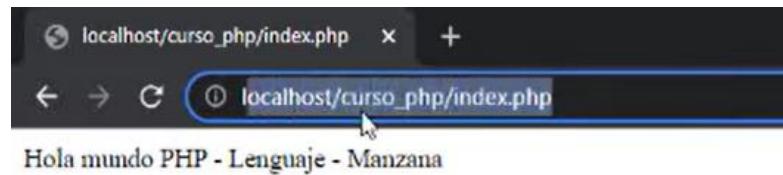
Manzana



DIFERENCIA entre el método GET y POST, el método get se suele utilizar mucho con proyectos con MVC o con APPS, se utiliza mas POST por la seguridad de los datos, con GET se muestran los datos en la URL de la siguiente forma, y manipulando la url se pueden modificar los datos:



Y con el método POST no pasa eso:



## Select y Checkbox múltiples en formularios php para mandar multiples datos:

Como sabemos solo podemos seleccionar una opción en el select y checkbox de los ejemplos anteriores, aquí veremos como poder seleccionar múltiples opciones.

Para hacer eso con un select, necesitamos modificar el select del código anterior agregando el atributo múltiple a la etiqueta select y en el name agregaremos corchetes para indicar que será un array para posteriormente recuperarlos en el otro archivo php.

```
<body>

    <form action="index.php" method="POST" >

        <label for="asignatura">Asignatura</label>
        <select id="asignatura" name="asignatura[]" multiple >
```

Si veo el tipo de dato que me devuelve esto

```
formulario.php          index.php ●
index.php
<?php
3 var_dump($_POST['asignatura']);
```

Si selecciono varios valores en el select, presionando (Ctrl+clic en las opciones) que quiero seleccionar y doy clic al botón enviar me dirá lo siguiente:

A screenshot of a web browser window showing a form. The form has a dropdown menu labeled "Asignatura" with options "Ingles", "Matemáticas", "Ciencia", and "Lenguaje". Below the dropdown is a checkbox labeled "Manzana" which is unchecked. At the bottom of the form is a button labeled "Enviar".

Es un array que contiene los 3 valores que seleccione en el select

```
localhost/curso_php/index.php x +  
localhost/curso_php/index.php  
array(3) { [0]=> string(6) "Ingles" [1]=> string(11) "Matematicas" [2]=> string(8) "Lenguaje" }
```

Puedo mostrar los datos con un foreach ya que se encuentran dentro de un arreglo:

```
formulario.php index.php x  
index.php  
<?php  
  
foreach($_POST['asignatura'] as $asignatura){  
    echo $asignatura."<br>";  
}
```

En este caso seleccione las 4 opciones de mi select y di clic al botón para enviar:

```
localhost/curso_php/index.php x +  
localhost/curso_php/index.php  
  
Ingles  
Matematicas  
Ciencia  
Lenguaje
```

Ahora para hacer lo mismo con un **checkbox para que sea múltiple** vamos a agregar los corchetes en los name de mis checkbox, en mi ejemplo agregue y ahora tengo 3 diferentes

```

<label for="opcion-1">
    <input type="checkbox" value="Manzana" id="opcion-1" name="frutas[]">
    Manzana
</label>

<label for="opcion-2">
    <input type="checkbox" value="Fresa" id="opcion-2" name="frutas[]">
    Fresa
</label>

<label for="opcion-3">
    <input type="checkbox" value="Uva" id="opcion-3" name="frutas[]">
    Uva
</label>

<br><br><br>

<button type="submit" >Enviar</button>

```

Y el en el archivo a donde quiero que se manden hago lo mismo con un foreach para mostrar todos los datos que selecciono el usuario:



The screenshot shows a code editor with two tabs: "index.php" (active) and "formulario.php". The "index.php" tab contains the following PHP code:

```

<?php
$materias=$_POST['asignatura'];
foreach($materias as $asignatura){
    echo $asignatura."<br>";
}

echo "<br>";
echo "<br>";
echo "<br>";

$fresa=$_POST['frutas'];
foreach($fresa as $fruta){
    echo $fruta."<br>";
}

```

The "formulario.php" tab is visible in the background.

The top screenshot shows a web page with a dropdown menu labeled "Asignatura" containing "Ingles", "Matemáticas", "Ciencia", and "Lenguaje". Below the dropdown is a list of checked checkboxes: "Manzana", "Fresa", and "Uva". At the bottom is a button labeled "Enviar".

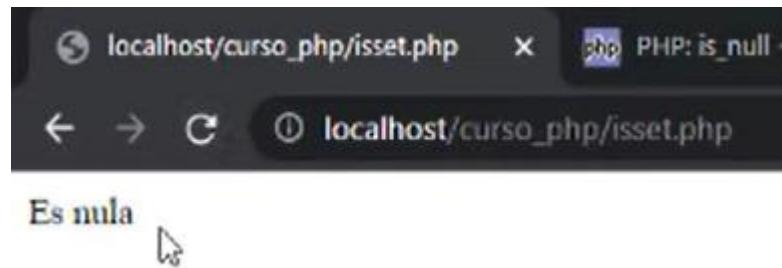
The bottom screenshot shows the results of the form submission. It displays the selected value from the dropdown ("Matemáticas") and the values of the checked checkboxes ("Manzana", "Fresa", "Uva").

Como saber si una variable esta vacía o definida):

Para esto veremos 3 funciones diferentes, comenzando con **is\_null** sirve para comprobar si la variable es nula devolviendo valores booleanos, y esta es su documentación:

<https://www.php.net/manual/es/function.is-null.php> Una variable se considera nula si le agregamos valor nulo, o si simplemente solo creamos la variable pero no le definimos un valor.

```
<?php
$numero=NULL;
if(is_null($numero)){
    echo "Es nula";
} else{
    echo "No es nula";
}
```



Lo mismo pasa al [eliminar una variable con la función unset\(\)](#)

```
isset.php X
isset.php
1 <?php
2
3 $numero="9";
4
5 unset($numero);
6
7 if(is_null($numero)){
8     echo "Es nula";
9 }else{
10    echo "No es nula";
11 }
```

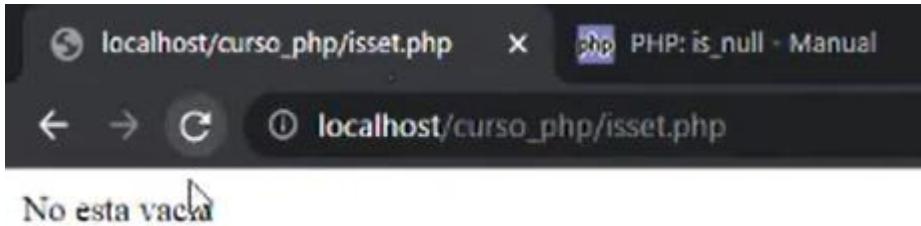
También podemos saber si una variable esta vacía o no con la función [empty\(\)](#) la cual

Devuelve **false** si **var** existe y tiene un valor no vacío, distinto de cero. De otro modo devuelve **true**.

Los siguientes valores son considerados como vacíos:

- "" (una cadena vacía)
- 0 (0 como un **integer**)
- 0.0 (0 como un **float**)
- "0" (0 como un **string**)
- **null**
- **false**
- **array()** (un array vacío)

```
isset.php X
isset.php
1 <?php
2
3 $numero="9";
4
5 if(empty($numero)){
6     echo "Esta vacia";
7 }else{
8     echo "No esta vacia";
9 }
```



Ahora para determinar si una variable esta definida y no es null utilizamos la función `isset()` a diferencia de `empty` que determina si esta vacía o indefinida, aquí aunque tenga 0 pero si está definida con algún valor dará TRUE excepto si la definimos con NULL.

Un ejemplo ya más práctico utilizando `isset()` en un formulario sería el siguiente, aquí dejamos el atributo `action` vacío de la etiqueta `form` que es lo mismo a colocar el nombre de ese mismo archivo en esa parte es decir que se enviarán los datos del formulario con POST a ese mismo archivo, ahora si lo que escriba el usuario en mi input de tipo text viene definida y es distinta a una cadena vacía entonces se va a ejecutar el código del archivo `for.php` por el include que sabemos que sirve para insertar código procedente de otro archivo.

The screenshot shows a code editor with two tabs open: 'iset.php' and 'for.php'. The 'iset.php' file contains the following PHP code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Document</title>
8 </head>
9 <body>
10
11   <form action="" method="POST">
12     <input type="text" name="numero">
13     <button type="submit">Enviar</button>
14   </form>
15
16   <?php
17     if(isset($_POST['numero']) && $_POST['numero']!=''){
18       include "for.php";
19     }
20   ?>
21 </body>
22 </html>
```

The 'for.php' file contains the following PHP code:

```
1 <?php
2
3   $numero=$_POST['numero'];
4   for($i=12; $i>=1; $i--){
5     echo $numero." X ".$i." = ".$i*$numero."<br>";
6 }
```

En este caso ingrese 7 y di clic al botón de enviar:

The browser window shows the output of the PHP code execution. The user has entered '7' into the input field and clicked the 'Enviar' button. The page displays the multiplication table for 7 from 1 to 12.

```
7 X 12 = 84
7 X 11 = 77
7 X 10 = 70
7 X 9 = 63
7 X 8 = 56
7 X 7 = 49
7 X 6 = 42
7 X 5 = 35
7 X 4 = 28
7 X 3 = 21
7 X 2 = 14
7 X 1 = 7
```

## Como subir o enviar archivos a servidor php con formularios:

Para esto tenemos el siguiente formulario y en la etiqueta form colocamos **action** para indicar a que archivo queremos pasar los datos o en este caso el archivo, el método con **method** a ocupar (solo se puede con POST) y un nuevo atributo que estamos viendo que también es necesario

**enctype="multipart/form-data"** para que se puedan enviar correctamente los archivos, con ese valor indicamos que queremos enviar archivos desde ese formulario que tenemos.

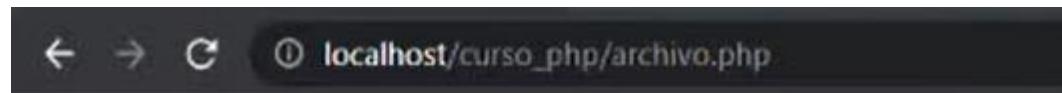
NOTA: Únicamente se puede con el método POST

```
archivo.php • carga.php
archivo.php
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4     <meta charset="UTF-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7     <title>Carga de archivo</title>
8 </head>
9 <body>
10    <h3>Subir archivo con PHP</h3>
11    <form action="carga.php" method="POST" enctype="multipart/form-data" >
12        <input type="file" name="fichero">
13        <br><br>
14        <button type="submit" >Enviar</button>
15    </form>
16 </body>
17 </html>
```

De esta manera accedemos a información del archivo que se selecciono y se envio en mi formulario, con name accedemos al nombre del archivo, con tmp\_name accedemos a la ruta en la que se encuentra el archivo, el type para el tipo de archivo que estamos cargando, error para detectar si el archivo se cargo bien o no, y size para el tamaño del archivo en bytes

```
archivo.php • carga.php
carga.php
1 <?php
2
3 echo $_FILES['fichero'][ 'name' ]. "<br>" ;
4 echo $_FILES['fichero'][ 'tmp_name' ]. "<br>" ;
5 echo $_FILES['fichero'][ 'type' ]. "<br>" ;
6 echo $_FILES['fichero'][ 'error' ]. "<br>" ;
7 echo $_FILES['fichero'][ 'size' ]. "<br>" ;
```

Por ejemplo:



## Subir archivo con PHP

No se eligió archivo

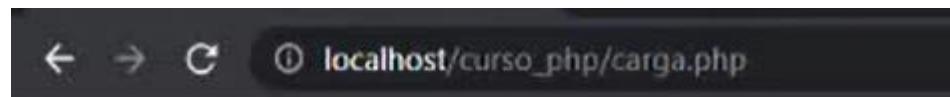
Si selecciono un archivo y doy clic al botón para enviar:



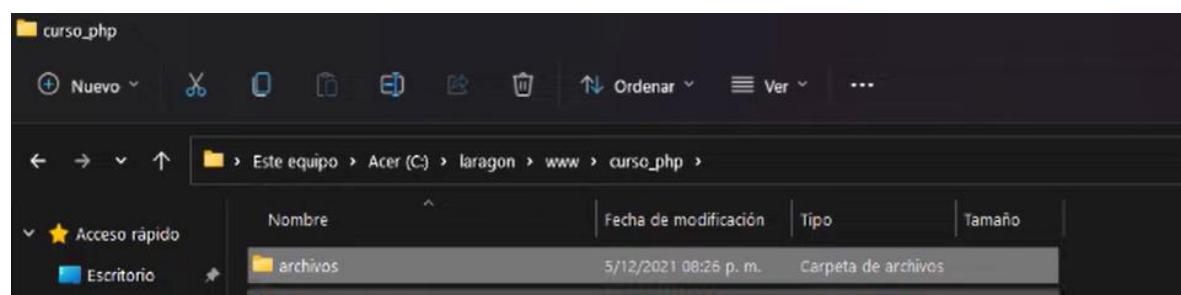
## Subir archivo con PHP

cover\_php.jpg

Y me muestra los datos del archivo:



Bueno para continuar vamos a crear una carpeta en el proyecto, que es en donde se van a almacenar todos los archivos que mandemos mediante el formulario en este caso se llama archivos.



Aquí ocupamos la función `file_exists("")` para saber si la carpeta o el archivo existe, entre las comillas va la ruta de la carpeta, esta función nos devuelve valores booleanos, en este caso decimos que si el formulario existe nos de FALSE ya que lo estamos negando con !



The screenshot shows a code editor interface with two tabs: "archivo.php" and "carga.php". In the left sidebar, there is a tree view under "CURSO\_PHP" with nodes "archivos", "include", "archivo.php", and "arrays.php". The "carga.php" tab is active, displaying the following code:

```
<?php  
if(!file_exists("archivos")){  
}  
}
```

Aquí decimos que si el directorio o la carpeta no esta creado, entonces se va a crear con el nombre de archivos usando la función `mkdir("")`, dentro de las comillas va el **nombre de la carpeta que queremos crear**, para eso sirve esa función, y también recibe otro parámetro que son los permisos, con 0777 damos todos los permisos de escritura y lectura para la carpeta para no tener problemas de permisos en el servidor. Y para **detener la ejecución de un script** lo hacemos con la función `exit()`.

Entonces aquí estamos diciendo que si no existe una carpeta llamada archivos, la cree, y si no se puede crear me mande un error y se deje de ejecutar el script.



The screenshot shows the "carga.php" tab with the following code:

```
<?php  
if(!file_exists("archivos")){  
    if(!mkdir("archivos", 0777)){  
        echo "Error al crear el directorio";  
        exit();  
    }  
}
```

Ahora utilizaremos la función `chmod("")` que sirve para dar permisos de lectura y escritura a una carpeta a esta función se le envían 2 parámetros, en este caso queremos hacer eso con la carpeta que YA tenemos creada llamada archivos.

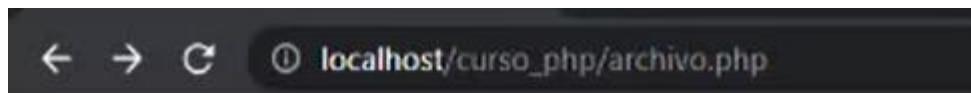
Y también usamos la función `move_uploaded_file()` en donde ocupa la ruta donde esta almacenado el archivo esto con `tmp_name` como ya habíamos visto un poco mas atrás, luego ponemos en donde queremos colocar el archivo que se encuentra en la carpeta temporal asignándole el nombre que tenia nuevamente.

Entonces aquí le estamos diciendo que mueva el archivo almacenado en la carpeta temporal, que vaya a mi carpeta archivos y que entre ( / ), después que mueva el archivo a esa carpeta asignándole su nombre.



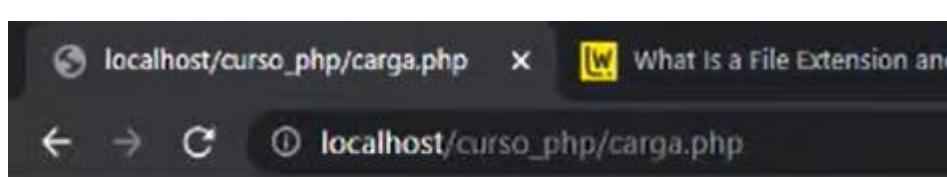
```
archivo.php carga.php
carga.php
1 <?php
2
3 if(!file_exists("archivos")){
4     if(!mkdir("archivos", 0777)){
5         echo "Error al crear el directorio";
6         exit();
7     }
8 }
9
10 chmod("archivos", 0777);
11
12 if(move_uploaded_file($_FILES['fichero']['tmp_name'], "archivos/".$_FILES['fichero']['name'])){
13     echo "Archivo subido con exito";
14 }else{
15     echo "Error al subir el archivo";
16 }
17 }
```

Este código ya es capaz de mover un archivo a una carpeta, si yo selecciono un archivo y lo muevo ocurrirá lo siguiente:



## Subir archivo con PHP

cover\_php.jpg



Archivo subido con exito

Si vemos la carpeta que creamos, observaremos que ya tiene el archivo, en caso de no existir la carpeta la va a crear y se va a mover el archivo a esa carpeta:



Como limitar el tipo de archivo a subir o enviar al servidor php con formularios:

Es decir, solo aceptar cierto tipo de archivos, para eso ocupamos el atributo accept en la etiqueta input para seleccionar archivos (de tipo file) y dentro colocamos los tipos de archivo que queremos aceptar únicamente,

```
<body>
    <h3>Subir archivo con PHP</h3>
    <form action="carga.php" method="POST" enctype="multipart/form-data" >
        <input type="file" name="fichero" accept=".jpg, .png, .jpeg">
        <br><br>
        <button type="submit" >Enviar</button>
    </form>
</body>
```

Podemos ver las extensiones de cualquier tipo de archivo en esta página, ocupamos el MIME Type para esta segunda forma de limitar el tipo de archivo. <https://www.lifewire.com/file-extensions-and-mime-types-3469109> en esta tabla: (no está completa).

Application	MIME Type	File Extension
Corel Envoy	application/envoy	evy
fractal image file	application/fractals	fif
Windows print spool file	application/futuresplash	spl
HTML application	application/hta	hta
Atari ST Program	application/internet-property-stream	acx
BinHex encoded file	application/mac-binhex40	hqx
Word document	application/msword	doc

En este código añadimos el primer if que nos dice que si el archivo que queremos subir ubicado en la carpeta temporal es de un tipo diferente a un pdf (application/pdf) entonces nos diga que no esta permitido y que detenga la ejecución del script.

```

archivo.php carga.php ●
carga.php
1 <?php
2
3 if(mime_content_type($_FILES['fichero']['tmp_name'])!="application/pdf"){
4     echo "Tipo de fichero no admitido";
5     exit();
6 }
7
8 if(!file_exists("archivos")){
9     if(!mkdir("archivos",0777)){
10         echo "Error al crear el directorio";
11         exit();
12     }
13 }
14
15 chmod("archivos",0777);
16
17 if(move_uploaded_file($_FILES['fichero']['tmp_name'],"archivos/".$_FILES['fichero'][name])){
18     echo "Archivo subido con exito";
19 }else{
20     echo "Error al subir el archivo";
21 }
22 }
```

Del link sacamos el MIME Type de los pdf.

Acrobat file	application/pdf	pdf
●	●	●

## Como limitar el peso del archivo que queremos subir:

En ese caso estamos limitando el peso del archivo a 3megabytes, recordemos que 1megabyte es equivalente a 1024 kilobytes

Primero calculamos con la división cuento pesa el archivo en kilobytes por eso dividimos el peso del archivo entre 1024, y luego ponemos en la condición el pero igual el kilobytes que no queremos que sobrepase el archivo.

El código final queda de la siguiente forma, limitamos el peso en el segundo if.



```
archivo.php  carga.php 1

carga.php
1 <?php
2
3 if(mime_content_type($_FILES['fichero']['tmp_name'])!="image/jpeg" && mime_content_type($_FILES
4 ['fichero']['tmp_name'])!="image/png"){
5     echo "Tipo de fichero no admitido";
6     exit();
7 }
8 if($_FILES['fichero']['size']/1024>3072){
9     echo "El archivo supera el peso permitido";
10    exit();
11 }
12 if(!file_exists("archivos")){
13     if(!mkdir("archivos",0777)){
14         echo "Error al crear el directorio";
15         exit();
16     }
17 }
18 chmod("archivos",0777);
19
20 if(move_uploaded_file($_FILES['fichero']['tmp_name'],"archivos/".$_FILES['fichero']['name'])){
21     echo "Archivo subido con exito";
22 }else{
23     echo "Error al subir el archivo";
24 }
```

Ejemplo de manejo de cookies en php (crear y eliminar):

### Cookies en PHP

Las cookies son un mecanismo por el que se **almacenan datos** en el navegador para monitorizar o identificar a los usuarios que vuelvan al sitio web. En otras palabras podemos decir que **las cookies son pequeños archivos** donde almacenamos un datos, estos archivos se **almacenan en el navegador del cliente**.

Las cookies se deben de crear **antes del Doctype**, ya que han de ser generadas antes de que el **navegador procese el código HTML**.

**Ejemplo de uso de cookies:** preferencias de idioma, seguimiento de anuncios etc.

Este tema me lo brinqué pero esta aquí: [https://youtu.be/4amiUJHLiy4?si=K-SND\\_TxYJIpQFLk](https://youtu.be/4amiUJHLiy4?si=K-SND_TxYJIpQFLk)

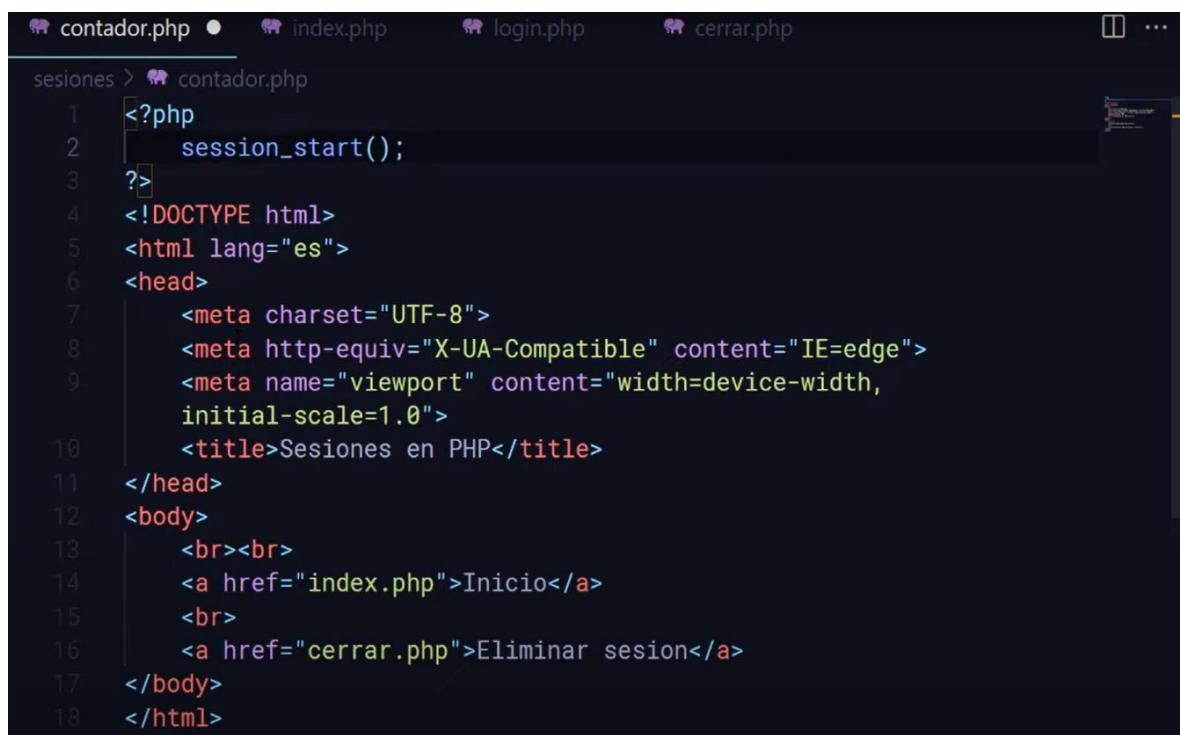
## Manejo de sesiones en php con variables de \$\_SESSION:

### Sesiones en PHP

Las **sesiones** en aplicaciones web desarrolladas con **PHP** nos sirven para **almacenar información** durante toda la visita de un usuario a un **sitio web**. Dicha información se almacena en el **servidor**.

**Ejemplo de uso de sesiones:** inicio de sesión (login), buscadores (que recuerde termino de búsqueda) etc.

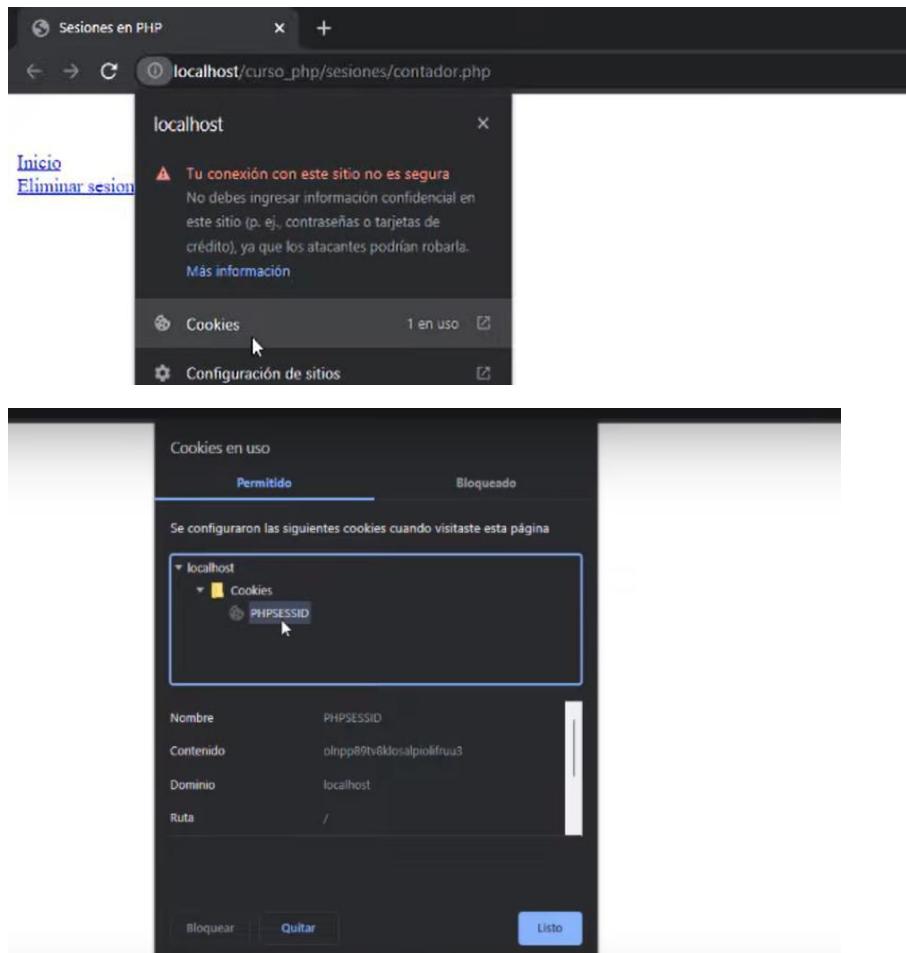
Las sesiones normalmente se crean al inicio de cualquier otro código, por eso es lo primero que tenemos en este ejemplo, creamos una sesión con la función `sesión_start()`



The screenshot shows a code editor interface with several tabs at the top: 'contador.php' (which is the active tab), 'index.php', 'login.php', and 'cerrar.php'. Below the tabs, there's a breadcrumb navigation bar: 'sesiones > contador.php'. The main area contains the following PHP code:

```
1 <?php
2     session_start();
3 ?>
4 <!DOCTYPE html>
5 <html lang="es">
6 <head>
7     <meta charset="UTF-8">
8     <meta http-equiv="X-UA-Compatible" content="IE=edge">
9     <meta name="viewport" content="width=device-width,
10        initial-scale=1.0">
11    <title>Sesiones en PHP</title>
12 </head>
13 <body>
14     <br><br>
15     <a href="index.php">Inicio</a>
16     <br>
17     <a href="cerrar.php">Eliminar sesión</a>
18 </body>
19 </html>
```

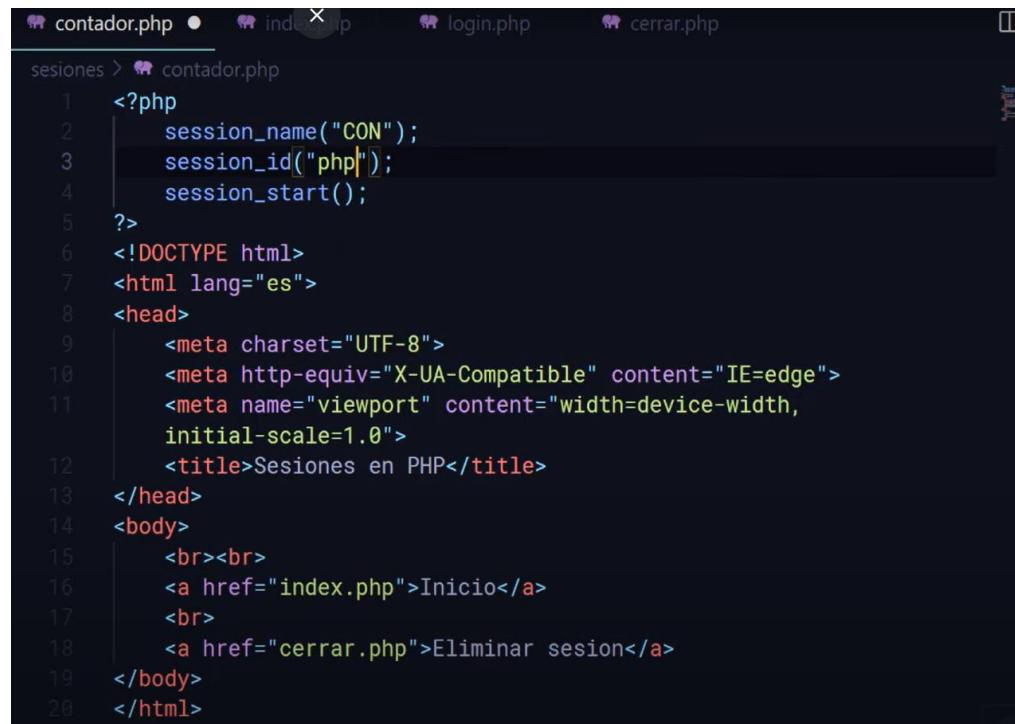
Al crear una sesión se crea una cookie para tener una conexión entre cliente servidor, para ver las cookies que tenemos es así:



También puedo verlo en aplicaciones al inspeccionar, aquí me dice el nombre y clave que tiene:

A screenshot of developer tools in a browser, specifically the "Aplicación" (Application) panel. On the left, there's a sidebar with sections like "Aplicación", "Almacenamiento", and "Almacenamiento". Under "Almacenamiento", "Cookies" is expanded, showing a list for "https://localhost". One cookie is selected: "PHPSESSID" with value "olnpp89tv8klosalpiolifruu3". To the right, a detailed view of the cookie is shown in a table with columns: Nombre, Valor, Domain, Path, Expires / Max-Age, and Tamaño. The "Valor" column shows the value "olnpp89tv8klosalpiolifruu3".

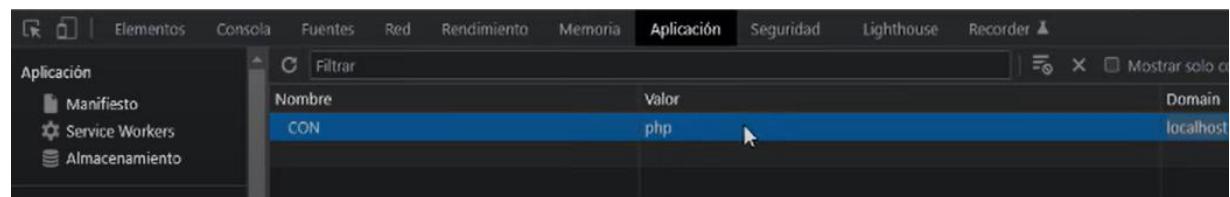
Si yo quiero que el nombre y la clave sean diferentes es de la siguiente forma, utilizando la función `sesión_name(" ")` para cambiarle el nombre y la función `session_id(" ")` para **cambiar el id o el valor**.



```

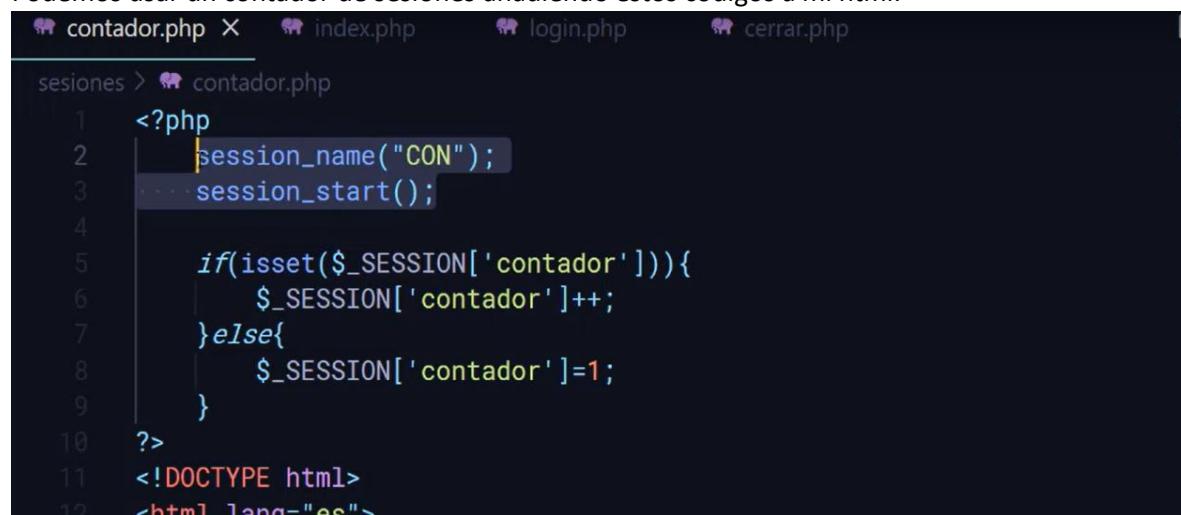
sesiones > contador.php
1  <?php
2  |     session_name("CON");
3  |     session_id("php");
4  |     session_start();
5  ?>
6  <!DOCTYPE html>
7  <html lang="es">
8  <head>
9      <meta charset="UTF-8">
10     <meta http-equiv="X-UA-Compatible" content="IE=edge">
11     <meta name="viewport" content="width=device-width,
12         initial-scale=1.0">
13     <title>Sesiones en PHP</title>
14 </head>
15 <body>
16     <br><br>
17     <a href="index.php">Inicio</a>
18     <br>
19     <a href="cerrar.php">Eliminar sesion</a>
20 </body>
21 </html>

```



Nombre	Valor	Domain
CON	php	localhost

Podemos usar un contador de sesiones añadiendo estos códigos a mi html.



```

sesiones > contador.php
1  <?php
2  |     session_name("CON");
3  |     session_start();
4
5  |     if(isset($_SESSION['contador'])){
6  |         $_SESSION['contador']++;
7  |     }else{
8  |         $_SESSION['contador']=1;
9  |     }
10 ?>
11 <!DOCTYPE html>
12 <html lang="es">

```

```

<body>
    <?php echo "Has recargado esta pagina ".$_SESSION['contador']." veces"; ?>
        <form action="login.php" method="POST">
            <label>Usuario</label>

```

Ahora para hacer una simulación de un login dejé mi html de la siguiente manera:

```

index.php X login.php cerrar.php

sesiones > index.php
  7   <head>
  8     <meta charset="UTF-8">
  9     <meta http-equiv="X-UA-Compatible" content="IE=edge">
 10     <meta name="viewport" content="width=device-width,
 11       initial-scale=1.0">
 12     <title>Sesiones en PHP</title>
 13   </head>
 14   <body>
 15     <form action="login.php" method="POST">
 16       <label>Usuario</label>
 17       <input type="text" name="usuario">
 18       <br>
 19       <label>Clave</label>
 20       <input type="password" name="clave">
 21       <br><br>
 22       <button type="submit">Login</button>
 23     </form>
 24   </body>
 25 </html>

```

Y en el otro archivo con código php hago lo siguiente, en donde digo que si los datos ingresados en el input con el name usuario son iguales a Carlos, y en el input con el name clave sea igual a 1234, entonces se cree o inicie la sesión con nombre Login, cree 3 variables de sesión y me diga sesión iniciada, sino datos incorrectos.

```

index.php login.php X contador.php cerrar.php

sesiones > login.php
  1   <?php
  2
  3     if($_POST['usuario']=="Carlos" && $_POST['clave']=="1234"){
  4       session_name("LOGIN");
  5       session_start();
  6
  7       $_SESSION["Nombre"]="Carlos";
  8       $_SESSION["Apellido"]="Alfaro";
  9       $_SESSION["Pais"]="El Salvador";
 10
 11       echo "Sesion iniciada";
 12     }else{
 13       echo "Datos incorrectos";
 14     }

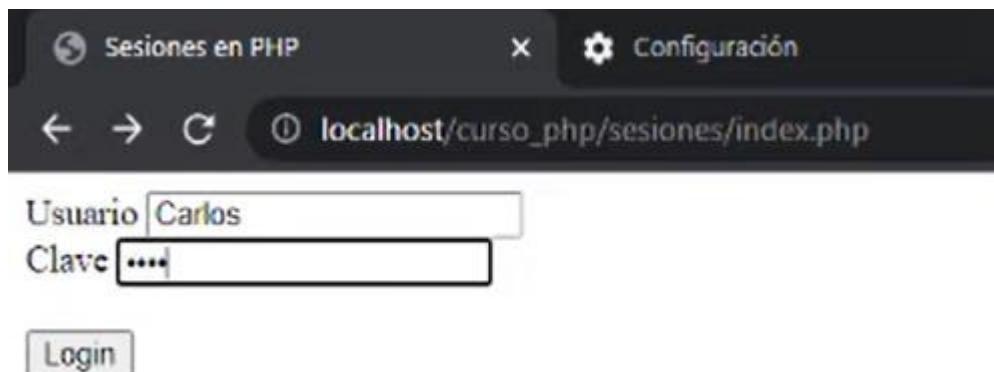
```

Tenemos que tener el nombre de la sesión y el `session_start()` para que funcione también en el archivo donde se van a mostrar los datos almacenados en las variables de sesión que se crearon en el archivo login.php, en este caso mostramos la variable Nombre que sabemos que contiene el valor Carlos.

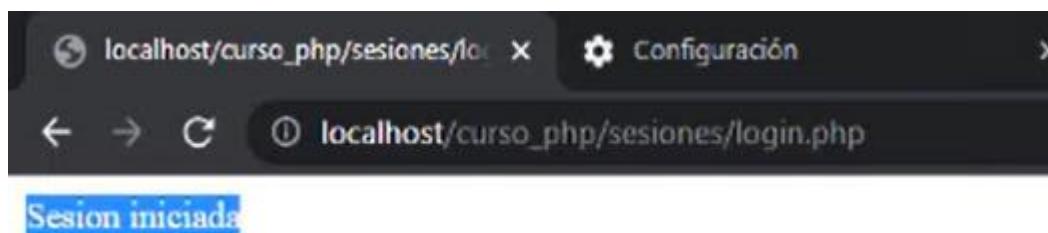
The screenshot shows a code editor with several files listed in the top bar: index.php, login.php, contador.php (which is currently selected), and cerrar.php. The code in contador.php is as follows:

```
<?php
    session_name("LOGIN");
    session_start();
?>
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
    initial-scale=1.0">
    <title>Sesiones en PHP</title>
</head>
<body>
    <?php echo "Hola ".$_SESSION['Nombre']."'"; ?>
    <a href="cerrar.php">Eliminar sesion</a>
</body>
</html>
```

Ahora si coloco Carlos, coloco en el password 1234 y doy clic al botón de Login me dirá sesión iniciada.



Significa que se han creado correctamente los datos o variables de sesión



Si ahora me voy al archivo contador.php observamos que me muestra el valor de sesión correctamente que almacenamos en nuestra variable se sesión llamada Nombre

Sesiones en PHP Configuración

localhost/curso\_php/sesiones/contador.php

Hola Carlos [Eliminar sesión](#)

Si ahora coloco Apellido observaremos que también se muestra, o si muestro País también:

```
index.php login.php contador.php cerrar.php
```

sesiones > contador.php

```
7 <head>
8     <meta charset="UTF-8">
9     <meta http-equiv="X-UA-Compatible" content="IE=edge">
10    <meta name="viewport" content="width=device-width,
11        initial-scale=1.0">
12    <title>Sesiones en PHP</title>
13 </head>
14 <body>
15     <?php echo "Hola ".$_SESSION['Apellido'].""; ?>
16     <a href="cerrar.php">Eliminar sesión</a>
17 </body>
</html>
```

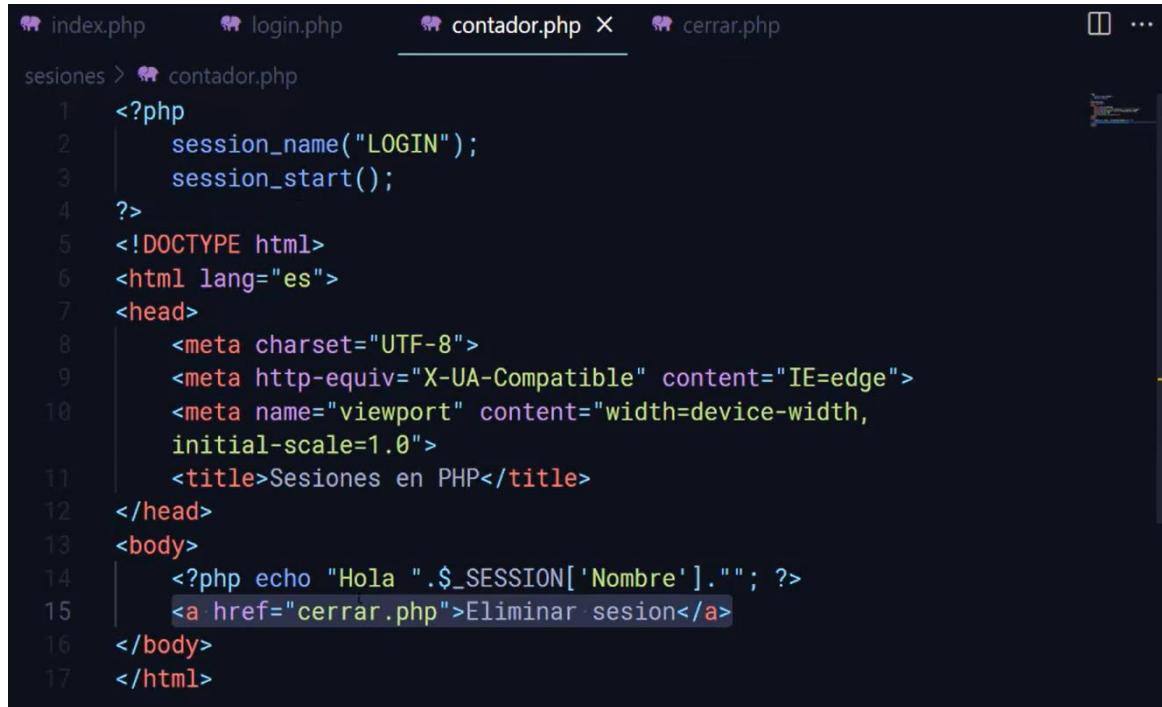
Sesiones en PHP Configuración

localhost/curso\_php/sesiones/contador.php

Hola Alfaro [Eliminar sesión](#)

Como eliminar una sesión:

Para eso tenemos la etiqueta <a/> que es un enlace que al presionarlo nos va a llevar al archivo que tenemos llamado cerrar.php



```
index.php login.php cuenta.php X cerrar.php ...
sesiones > cuenta.php
1 <?php
2     session_name("LOGIN");
3     session_start();
4 ?>
5 <!DOCTYPE html>
6 <html lang="es">
7 <head>
8     <meta charset="UTF-8">
9     <meta http-equiv="X-UA-Compatible" content="IE=edge">
10    <meta name="viewport" content="width=device-width,
11        initial-scale=1.0">
12    <title>Sesiones en PHP</title>
13 </head>
14 <body>
15     <?php echo "Hola ".$_SESSION['Nombre']."'"; ?>
16     <a href="cerrar.php">Eliminar sesion</a>
17 </body>
</html>
```

Recordemos para usar las variables de sesión o trabajar en la misma sesión dentro de otro archivo es necesario colocar el nombre y el `session_start()` esto solo en caso de que hayamos cambiado el nombre como lo hicimos en este ejemplo que le colocamos LOGIN, sino lo cambiamos solo es necesario el `session_start()`

Ahora para eliminar una sesión, ocupamos la función `session_destroy()` la cual elimina toda la información registrada, incluyendo variables de sesión y el id de la sesión (el valor que aparece al inspeccionar la cookie de muchos números y letras)



```
index.php login.php cuenta.php X cerrar.php ...
sesiones > cerrar.php
1 <?php
2     session_name("LOGIN");
3     session_start();
4
5     session_destroy();
```

O también podemos utilizar la función `session_unset()` que lo que hace **es únicamente eliminar las variables de sesión**, pero en los login reales lo que se ocupa es `session_destroy()` para eliminar completamente la sesión.



A screenshot of a code editor showing a file named 'cerrar.php'. The code is as follows:

```
<?php  
session_name("LOGIN");  
session_start();  
  
session_unset();
```

O si queremos eliminar únicamente una variable de sesión podemos hacerlo con `unset()` colocando dentro la variable de sesión que queremos eliminar de la siguiente manera :



A screenshot of a code editor showing a file named 'cerrar.php'. The code is as follows:

```
<?php  
session_name("LOGIN");  
session_start();  
  
unset($_SESSION["Nombre"]);
```

Al trabajar con php y mysql en un login real los datos se extraen de una BD y se comparan con los que escribimos en el formulario con los que queremos iniciar sesión.

## Cookies vs Sesiones:

### Diferencia entre cookies y sesiones en PHP

#### COOKIES

Se almacena la **información en local**, es decir en el navegador.

Se pueden **editar o borrar**

#### SESIONES

Se almacena la **información en el servidor**.

Es mas seguro que las cookies.

## Como redireccionar al usuario a otra página usando php:

Para esto ocuparemos el mismo código de los ejemplos anteriores que usábamos para el manejo de sesiones, lo que haremos es que en el archivo en donde se crean mis variables de sesión llamado login.php añadiremos la función `header(" ")` para redirigir al usuario, en donde escribiremos Location: seguido del nombre del archivo a donde queremos dirigir al usuario cuando se ejecute esa línea de código.



```
index.php      login.php      contador.php      cerrar.php      ...
sesiones > login.php
1  <?php
2
3      if($_POST['usuario']=="Carlos" && $_POST['clave']=="1234"){
4          session_name("LOGIN");
5          session_start();
6
7          $_SESSION["Nombre"]="Carlos";
8          $_SESSION["Apellido"]="Alfaro";
9          $_SESSION["Pais"]="El Salvador";
10
11         header("Location: contador.php");
12     }else{
13         echo "Datos incorrectos";
14     }

```

Ahora si coloco Carlos, coloco en el password 1234 y doy clic al botón de Login iniciara sesión

Sesiones en PHP Configuración

localhost/curso\_php/sesiones/index.php

Usuario: Carlos  
Clave: \*\*\*\*

Login

y me redirigirá al archivo contador.php

Sesiones en PHP

localhost/curso\_php/sesiones/contador.php

Hola Carlos [Eliminar sesión](#)

También podemos redireccionar al usuario usando JS de la siguiente manera: Entre las comillas simples colocamos el nombre del archivo al que queremos que vaya el usuario, en este caso cuando de clic al botón de cerrar sesión, se va a eliminar la sesión con la función `session_destroy()` y después va a dirigir al usuario al archivo index.php.

index.php login.php contador.php cerrar.php

sesiones > cerrar.php

```
1 <?php
2 session_name("LOGIN");
3 session_start();
4
5 session_destroy();
6
7 echo "<script> window.location.href='index.php'; </script>";
```

Si le damos clic a Eliminar sesión:

Sesiones en PHP

localhost/curso\_php/sesiones/contador.php

Hola Carlos [Eliminar sesión](#)

Me dirige al index.php

Sesiones en PHP

localhost/curso\_php/sesiones/index.php

Usuario

Clave