



Programación Web

Tema 1: Introducción a PHP

Curso 2023/24

7. Tipos de datos

- **Escalares**

- Enteros (int)
- Coma flotante (float)
- Cadena de caracteres (string)
- Booleano (bool)

- **Tipos compuestos**

- Tablas (arrays)
- Objetos

- **Tipos especiales**

- NULL
- recurso (resource)
- Función de llamada
- Iteradores

7. Tipos de datos

- **Escalares**

- Enteros (int)
- Coma flotante (float)
- Cadena de caracteres (string)
- Booleano (bool)

- **Tipos compuestos**

- Tablas (arrays)
- Objetos

- **Tipos especiales**

- NULL
- recurso (resource)
- Función de llamada
- Iteradores

7.1. Escalares

- En la mayoría de los casos, funcionan exactamente igual que en C/C++ y Java.
- Los valores numéricos menor y mayor soportados por PHP están contenidos en constantes predefinidas. Por ejemplo:
 - PHP_INT_MIN y PHP_INT_MAX
 - PHP_FLOAT_MIN y PHP_FLOAT_MAX
- Cuando un número de coma flotante se convierte a entero, el número **se trunca**.
 - **Novedad PHP 8.1:** Queda obsoleta la conversión **implícita** de un número de coma flotante a un entero que provoque una pérdida de precisión y esto provoca una alerta.
 - Esto último se soluciona con una conversión **explícita**. *intval(1.2)*
- Los números pueden contener el carácter guion bajo (_) sin que afecte a su valor.
 - 1_234.56 = 1234.56
- **Novedad PHP 8:** Al convertir un número de coma flotante en una cadena de caracteres, no se tendrá en cuenta la localización.
 - **Antes:** 1234.56 se convertía a 1234,56 en español.
 - **Ahora:** Siempre será 1234.56.

7.2. Cadenas de caracteres

- Una expresión literal de tipo cadena de caracteres puede especificarse entre comillas simples ("esto es una cadena") o entre comillas dobles ('esto también es una cadena') con diferencias de comportamiento muy importantes.

Ejemplo

```
<?php
echo 'What\'s up?<br />';
echo "Digo \"hola\".<br />";
?>
```

Resultado

```
What's up?
Digo "hola".
```

7.2. Cadenas de caracteres

- Cuando usamos comillas dobles, cualquier secuencia que comience por el signo '\$' se interpretará como una variable. No hay mecanismo de sustitución equivalente para las constantes;
- Con comillas simples, esto no ocurre.
- Si es necesario que el valor de la variable sea inmediatamente seguido por un carácter, tendremos que usar llaves. Con esta sintaxis, para que se conserve la llave, hay que duplicarla ({{\$variable}}).

```
Me llamo Daniel.  
Me llamo $nombre.  
$nombre = DanielUna manzana no cuesta mucho dinero.
```

```
Warning: Undefined variable $frutas in C:\xampp\htdocs\codigos\ejemplos\string.php on line 9  
Dos cuestan dos veces más.  
Dos manzanas cuestan dos veces más.  
{ $fruta } = { manzana }.
```

```
string.php x +  
Archivo Editar Ver  
  
<?php  
$nombre = 'Daniel';  
echo "Me llamo $nombre.<br />";  
echo 'Me llamo $nombre.<br />'; // no funciona  
echo "\$nombre = $nombre"; // se escapa el $  
  
$fruta = 'manzana';  
echo "Una $fruta no cuesta mucho dinero.<br />";  
echo "Dos $frutas cuestan dos veces más.<br />"; // !  
echo "Dos {{$fruta}}s cuestan dos veces más.<br />";  
echo "{\$fruta} = {{$fruta}}.<br />";  
?>
```

| Secuencia | Valor |
|-------------------------|--|
| <code>\n</code> | Salto de línea (= LF = código ASCII 10) |
| <code>\r</code> | Retorno de carro (= CR = código ASCII 13) |
| <code>\t</code> | Tabulación (= HT = código ASCII 9) |
| <code>\v</code> | Tabulación vertical (= VT = código ASCII 11) |
| <code>\e</code> | Escape (=ESC = código ASCII 27) |
| <code>\f</code> | Página siguiente (= FF = código ASCII 12) |
| <code>\\</code> | \ (ya visto) |
| <code>\\$</code> | \$ (ya visto) |
| <code>\"</code> | " (ya visto) |
| <code>\nnn</code> | El carácter designado por el código ASCII <code>nnn</code> expresado en octal |
| <code>\xnn</code> | El carácter designado por el código ASCII <code>nn</code> expresado en hexadecimal |
| <code>\u{nnnnnn}</code> | El carácter designado por el código Unicode (UTF-8) <code>nnnnnn</code> expresado en hexadecimal (añadido en la versión 7) |

7.2. Cadenas de caracteres

- Es posible acceder a una posición concreta de la cadena de caracteres con el operador '['].
 - **ATENCIÓN:** Antes de PHP 8, también podía usarse el carácter '{}'. Desde PHP 8, esto lanza una excepción, provocando un error fatal y terminando la ejecución del programa.
- PHP es capaz de convertir una cadena en número (entero o decimal) **mediante unas reglas que se han hecho más restrictivas a partir de la versión 8** en comparación con las versiones anteriores.
- Una cadena puede convertirse en número **si contiene un número que pueda interpretarse** como entero (int) o como número de coma flotante (float), autorizando los caracteres "blancos" (espacio, tabulación) al principio o al final (caracteres ignorados en la conversión).
- Si la cadena comienza por un valor numérico pero contiene caracteres no numéricos al final, aparece un error, aunque el valor numérico se tendrá en cuenta.

Tat



corchetes-string.php



Archivo

Editar

Ver

```
<?php
$nombre = 'Torcuato';
echo $nombre[0],$nombre[5],$nombre[-2];
?>
```

1 + "1" = 2

1 + "1,5" =

Warning: A non-numeric value encountered in **C:\xampp\htdocs\codigos\ejemplos\conversiones-string.php** on line 3

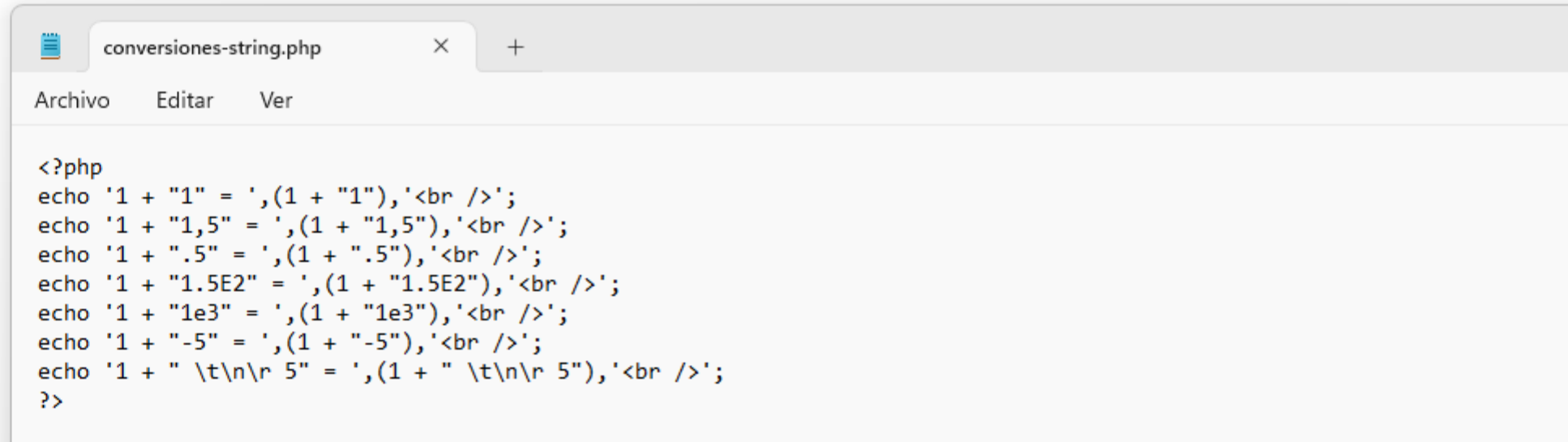
1 + ".5" = 1.5

1 + "1.5E2" = 151

1 + "1e3" = 1001

1 + "-5" = -4

1 + " \t\n\r 5" = 6



```
<?php
echo '1 + "1" = ',(1 + "1"),'<br />';
echo '1 + "1,5" = ',(1 + "1,5"),'<br />';
echo '1 + ".5" = ',(1 + ".5"),'<br />';
echo '1 + "1.5E2" = ',(1 + "1.5E2"),'<br />';
echo '1 + "1e3" = ',(1 + "1e3"),'<br />';
echo '1 + "-5" = ',(1 + "-5"),'<br />';
echo '1 + " \t\n\r 5" = ',(1 + " \t\n\r 5"),'<br />';
?>
```

1 + "1abc" =

Warning: A non-numeric value encountered in C:\xampp\htdocs\codigos\ejemplos\conversiones-string-errores.php on line 2

1 + "1.5abc" =

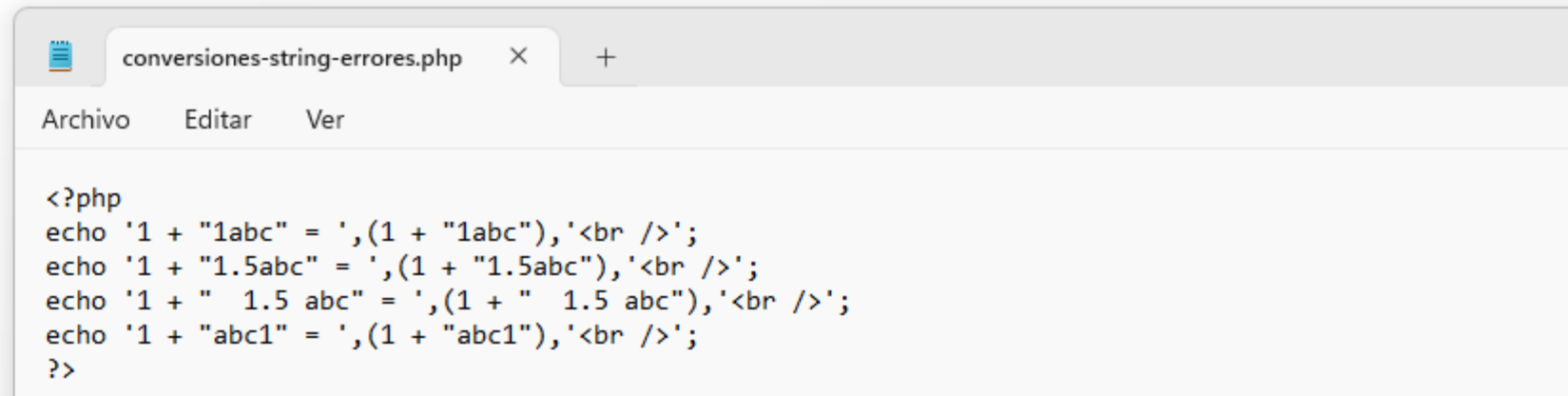
Warning: A non-numeric value encountered in C:\xampp\htdocs\codigos\ejemplos\conversiones-string-errores.php on line 3

1 + " 1.5 abc" =

Warning: A non-numeric value encountered in C:\xampp\htdocs\codigos\ejemplos\conversiones-string-errores.php on line 4

1 + "abc1" =

Fatal error: Uncaught TypeError: Unsupported operand types: int + string in C:\xampp\htdocs\codigos\ejemplos\conversiones-string-errores.php:5 Stack trace: #0 {main} thrown in C:\xampp\htdocs\codigos\ejemplos\conversiones-string-errores.php on line 5



The screenshot shows a web browser window with a single tab titled "conversiones-string-errores.php". The browser's address bar is empty. Below the address bar, there is a menu bar with three items: "Archivo", "Editar", and "Ver". The main content area of the browser displays the output of a PHP script, which is a series of echo statements showing the results of string conversions. The output is as follows:

```
<?php
echo '1 + "1abc" = ', (1 + "1abc"), '<br />';
echo '1 + "1.5abc" = ', (1 + "1.5abc"), '<br />';
echo '1 + " 1.5 abc" = ', (1 + " 1.5 abc"), '<br />';
echo '1 + "abc1" = ', (1 + "abc1"), '<br />';
?>
```

7.3. Booleano

- PHP es capaz de convertir cualquier tipo de datos en booleano

| Valor | Convertido en |
|-------------------------------------|---------------|
| número entero 0 | FALSE |
| número decimal 0.000... | |
| cadena vacía ("") | |
| cadena igual a 0 ("0") | |
| tabla vacía | |
| objeto vacío | |
| constante NULL (véase el tipo NULL) | |
| el resto | TRUE |

| | TRUE | FALSE |
|--------------------|------|------------------|
| Booleano -> Número | 1 | 0 |
| Booleano -> Cadena | "1" | ""(cadena vacía) |

7.4. Tipos especiales

- **Constante NULL (null):** lo poseen las variables sin inicializar.
- **Tipo recurso (resource):** referencia a recurso externo (fichero, base de datos, etc.)
- **Funciones callable:** se pueden pasar funciones o métodos de objetos como parámetros a otras funciones.
- **Iteradores (iterable):** utilizados en *foreach* o clases que implementen la interfaz *Traversable*.

Conviene visitar <https://www.php.net/manual/es/language.types.php> para estar al tanto.

- **Declaración de tipos**

- Parámetros de funciones.
- Valor de retorno de funciones.
- Atributos de clases.

`int`

El valor debe ser un entero.

`float`

El valor debe ser un número de coma flotante.

`string`

El valor debe ser una cadena de caracteres.

`bool`

El valor debe ser un booleano.

`array`

El valor debe ser una tabla.

`callable`

El valor debe ser el nombre de una función de llamada.

`iterable`

El valor debe ser de tipo iterable (tabla o clase que implemente la interfaz `Traversable`).

`object`

El valor debe ser un objeto.

`self`

El valor debe ser un objeto (una instancia) de la misma clase que donde se encuentra la declaración de tipo. Solo puede utilizarse en una clase.

`parent`

El valor debe ser un objeto (una instancia) del padre de la clase donde se encuentra la declaración de tipo. Solo puede utilizarse en una clase.

`mixed`

El valor puede ser de cualquier tipo (incluido `NULL`). Es una novedad de la versión 8.

- Valores de retorno de una función

| | |
|---------------------|--|
| <code>void</code> | La función o el método no devuelve ningún valor. |
| <code>static</code> | El valor debe ser un objeto (una instancia) de la misma clase que donde se llama ese método. Es una novedad de la versión 8. |
| <code>never</code> | La función o el método interrumpe la ejecución del programa, genera una excepción o no se termina nunca. Es una novedad de la versión 8.1. |

Desde la versión 8, se puede especificar un tipo en forma de una unión de tipos. Para ello, basta con hacer una lista de los tipos separados por una barra vertical, por ejemplo: `int | float`.

7.5. Tablas

- En PHP, una tabla (también conocida como **array**), es una lista de elementos ordenadas en pares clave/valor.
 - `std::map` en C++ es un buen ejemplo conocido y similar.
- La clave será siempre de tipo **entero** o **string**.
 - Cuando es de tipo **entero**, la tabla se considera numérica y, su clave, el **índice**.
 - Cuando es de tipo **string**, la tabla se considera **asociativa**.
- Las claves no serán necesariamente consecutivas ni ordenadas, y una tabla puede mezclar claves **enteras** y claves de tipo **string**.

| Clave/Índice | Valor |
|--------------|-------|
| 0 | cero |
| 1 | uno |
| 2 | dos |
| 3 | tres |

| Clave/Índice | Valor |
|--------------|---------|
| 20 | veinte |
| 30 | treinta |
| 10 | diez |

| Clave/Índice | Valor |
|--------------|-------|
| 0 | cero |
| cero | 0 |
| uno | 1 |
| 1 | uno |
| dos | 2 |
| 2 | dos |
| tres | 3 |
| 3 | tres |

| Clave/Índice | Valor | |
|--------------|--------------|-----------|
| ESPAÑA | Clave/Índice | Valor |
| | 0 | Madrid |
| | 1 | Barcelona |
| | 2 | Zaragoza |
| FRANCIA | Clave/Índice | Valor |
| | 0 | París |
| | 1 | Nantes |

7.5. Tablas

- Las tablas pueden ser creadas de varias formas.
 - Como una variable, definiéndose explícitamente mediante la función **array()**.
 - Como una variable, definiéndose implícitamente mediante el operador **[]**.
- Una variable utilizada por primera vez como **\$variable[...]** es automáticamente creada como tabla.
 - **OJO:** Hacerlo sobre una variable ya definida con un tipo escalar provoca un ERROR.
- Con una asignación del tipo **\$variable[] = valor**, PHP busca el índice más grande entero utilizado y asocia el valor del índice inmediatamente superior. Si la tabla está vacía (o solo utiliza claves alfanuméricas no convertibles en entero), el elemento se establece con el índice 0.
- Con una asignación del tipo **\$tabla[clave] = valor**, PHP asocia el valor con la clave indicada.
- Su uso es **igualmente válido en constantes**, solo que después no será posible actualizar sus valores.

7.5. Tablas

```
<?php
$numeros[] = 'cero'; // => índice 0
$numeros[] = 'uno'; // => índice máx (0) + 1 = 1
$numeros[] = 'dos'; // => índice máx (1) + 1 = 2
$numeros[] = 'tres'; // => índice máx (2) + 1 = 3
$numeros[5] = 'cinco'; // => índice 5
$numeros[] = 'seis'; // => índice máx (5) + 1 = 6
$numeros['uno'] = 1; // índice 'uno'
$numeros[] = 'siete'; // => índice máx (6) + 1 = 7
$numeros[-1] = 'menos uno'; // => -1
?>
```

| Clave/Índice | Valor |
|--------------|-----------|
| 0 | cero |
| 1 | uno |
| 2 | dos |
| 3 | tres |
| 5 | cinco |
| 6 | seis |
| uno | 1 |
| 7 | siete |
| -1 | menos uno |

7.5. Tablas

- Como hemos visto anteriormente, es posible crear tablas multidimensionales.
 - `$tabla[...] = $tabla_interior`
 - `$tabla[...][...] = valor`

```
<?php
// Creación de una tabla que contiene las ciudades de España
$ciudades_espania[] = 'Madrid';
$ciudades_espania[] = 'Barcelona';
$ciudades_espania[] = 'Zaragoza';
// Almacenamiento de la tabla de las ciudades de España
// en la tabla de las ciudades.
$ciudades['ESPAÑA'] = $ciudades_espania[];
// Idem con las ciudades de Francia
$ciudades_francia[] = 'París';
$ciudades_francia[] = 'Nantes';
$ciudades['FRANCIA'] = $ciudades_francia[];
?>
```

```
<?php
// Almacenamiento directo de las ciudades en la tabla
// - para España
$ciudades['ESPAÑA'][] = 'Madrid';
$ciudades['ESPAÑA'][] = 'Barcelona';
$ciudades['ESPAÑA'][] = 'Zaragoza';
// - para Francia
$ciudades['FRANCIA'][] = 'París';
$ciudades['FRANCIA'][] = 'Nantes';
?>
```

7.5. Tablas

| Clave/Índice | Valor | |
|--------------|--------------|-----------|
| ESPAÑA | Clave/Índice | Valor |
| | 0 | Madrid |
| | 1 | Barcelona |
| | 2 | Zaragoza |
| FRANCIA | Clave/Índice | Valor |
| | 0 | París |
| | 1 | Nantes |

7.5. Tablas

- El uso de la función **array()** es sencillo.

```
tabla array([mixto valor[, ...]])  
o  
tabla array([{cadena | entero} clave => mixto valor[, ...]])
```

- En la primera forma, no se especifican los índices, que serán **enteros** y **correlativos**.
- En la segunda forma, el índice o clave se define y recibe un valor a través del operador =>
- La función array acepta como argumento los datos de tipo tabla (bien una variable, bien una invocación anidada en array), lo que permite construir una tabla multidimensional, más compleja.

7.5. Tablas

```
<?php
$numeros = array('cero', 'uno', 'dos', 'tres',
    5 => 'cinco', 'seis', 'uno' => 1, 'siete', -1 => 'menos uno');
?>
```

| Clave/Índice | Valor |
|--------------|-----------|
| 0 | cero |
| 1 | uno |
| 2 | dos |
| 3 | tres |
| 5 | cinco |
| 6 | seis |
| uno | 1 |
| 7 | siete |
| -1 | menos uno |

7.5. Tablas

```
<?php
// Creación de una tabla que contiene las ciudades de España
// y de Francia
// La tabla de las ciudades de España es creada previamente.
$ciudades_espania = array('Madrid', 'Barcelona', 'Zaragoza');
// La de las ciudades de Francia se crea por una invocación anidada
// en array().
$ciudades = array('ESPAÑA' => $ciudades_espania,
                  'FRANCIA' => array('París', 'Nantes'));
?>
```

| Clave/Índice | Valor | |
|--------------|--------------|-----------|
| ESPAÑA | Clave/Índice | Valor |
| | 0 | Madrid |
| | 1 | Barcelona |
| | 2 | Zaragoza |
| FRANCIA | Clave/Índice | Valor |
| | 0 | París |
| | 1 | Nantes |

7.5. Tablas

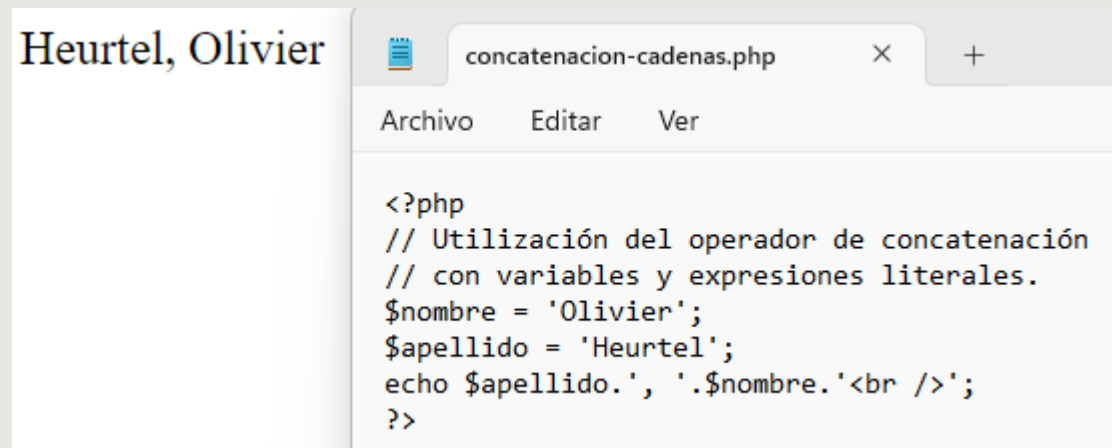
- En versiones recientes de PHP, su uso también se convalida con el operador [].

```
<?php
$numeros = ['cero', 'uno', 'dos', 'tres',
    5 => 'cinco', 'seis', 'uno' => 1, 'siete', -1 => 'menos uno'];
?>
```

```
<?php
$ciudades = [ 'ESPAÑA' => ['Madrid', 'Barcelona', 'Zaragoza'],
    'FRANCIA' => ['París', 'Nantes']];
?>
```

8. Expresiones y operadores

- En PHP, una expresión es **cualquier cosa que pueda contener un valor**.
 - Las expresiones más simples son las variables y las constantes y otras más complicadas serán las funciones, puesto que cada función devuelve un valor al ser invocada, es decir, contiene un valor, por lo tanto, es una expresión.
- Las expresiones en PHP, por norma general, son **exactamente las mismas que en C**. Con algunas excepciones notables.
 - **El operador “.” (operador de cadena o concatenación)**: devuelve una cadena igual a la primera cadena inmediatamente seguida de la segunda. *cadena1.cadena2*



The screenshot shows a web browser window. On the left, the output of a PHP script is displayed as "Heurtel, Olivier". On the right, the source code of the script, "concatenacion-cadenas.php", is visible. The code uses the concatenation operator "." to join the variables \$nombre and \$apellido.

```
<?php
// Utilización del operador de concatenación
// con variables y expresiones literales.
$nombre = 'Olivier';
$apellido = 'Heurtel';
echo $apellido.'', '.$nombre.'<br />';
?>
```

8. Expresiones y operadores

- Las expresiones en PHP, por norma general, son **exactamente las mismas que en C**. Con algunas excepciones notables.
 - El operador “===” (igualdad y tipos idénticos). Similar al operador “==”, pero comprobando además el tipo.
 - El operador “!==” (diferente o tipos diferentes). La negación del anterior.
 - Los operadores lógicos tienen sus versiones sobrecargadas (and y &&), (or y ||), etc.

9. Estructuras de control

- Las estructuras de control, de nuevo, **son idénticas a lo que encontramos en C.**
- Aunque esta vez, PHP ofrece una sintaxis alternativa a las mismas.
 - Destacan **if**, **while**, **for** y **switch**.
 - En cada caso, la forma básica de la sintaxis alternativa es cambiar abrir-llave por **dos puntos (:)** y cerrar-llave por **endif;; endwhile;; endfor;; or endswitch;;**, respectivamente.

```
<?php
// Estructura if / elseif / else
$nombre = 'Olivier';
$edad = NULL;
if ($nombre == NULL) {
    echo "¡Hola desconocido!<br />";
} elseif ($edad == NULL) {
    echo "¡Hola $nombre! No conozco su edad.<br />";
} else {
    echo "¡Hola $nombre! Tiene $edad años.<br />";
}
?>
```

```
<?php if (condición): ?>
    codigo_HTML
[ <?php elseif (condición): ?>
    codigo_HTML ]
[ ... ]
[ <?php else: ?>
    codigo_HTML ]
<?php endif; ?>
```

9. Estructuras de control

- En este curso, será útil destacar **foreach**. Esta operación se encuentra también disponible en las últimas versiones de C++.

```
foreach(tabla as variable_valor)
{ instrucciones }
```

o

```
foreach(tabla as variable_clave => variable_valor)
{ instrucciones }
```

10. Uso y definición de funciones

- La palabra clave **function** permite introducir la definición de una función.

```
function nombre_función([parámetros]) [: tipo] {  
    instrucciones;  
}
```

- **nombre_función:** nombre de la función.
- **parámetros:** parámetros posibles de la función.
- **tipo (OPCIONAL):** declaración del tipo de datos que devuelve una función.
 - **int, float, string, bool, array, callable, iterable, object, mixed, void, never...**
- El nombre de la función no debe ser una palabra reservada de PHP (nombre de función, de instrucción) ni ser igual al nombre de otra función previamente definida.
- Una función de usuario se invoca como una función nativa de PHP: en una asignación, en una comparación, etc.
- Si la función devuelve un valor, es posible utilizar la instrucción **return** para definir el valor de retorno de la función.

¡Hola!
2 x 4 = 8
10 x 12 es superior a 100.

```
funciones.php
Archivo  Editar  Ver

<?php
// Función sin parámetro que muestra "¡Hola!"
// Sin valor de retorno
function mostrar_hola() {
    echo '¡Hola!<br />';
}
// Función con 2 parámetros que devuelve el producto
// de dos parámetros.
function producto($valor1,$valor2) {
    return $valor1 * $valor2;
}
// Llamada a la función mostrar_hola
mostrar_hola();
// Utilización de la función producto:
// - en una asignación
$resultado = producto(2,4);
echo "2 x 4 = $resultado<br />";
// - en una comparación
if (producto(10,12) > 100) {
    echo '10 x 12 es superior a 100.<br />';
}
?>
```

```
<?php
// Función que toma un parámetro por valor
function porValor($valor) {
    $valor = $valor * 2;
    echo "Valor dentro de la función porValor(): $valor<br>\n";
}

// Función que toma un parámetro por referencia
function porReferencia(&$valor) {
    $valor = $valor * 2;
    echo "Valor dentro de la función porReferencia(): $valor<br>\n";
}

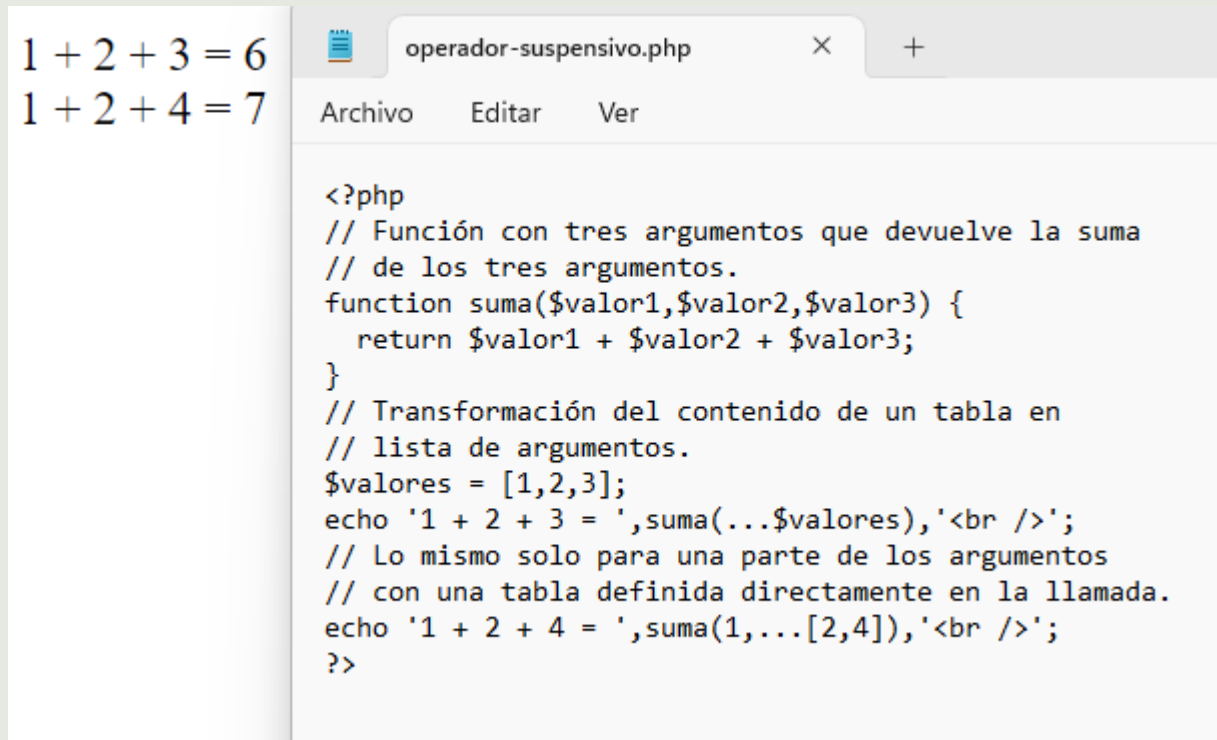
$num = 5;

// Llamada a la función que toma un parámetro por valor
echo "Valor antes de la llamada porValor(): $num<br>\n";
porValor($num);
echo "Valor después de la llamada porValor(): $num<br>\n";

// Llamada a la función que toma un parámetro por referencia
echo "Valor antes de la llamada porReferencia(): $num<br>\n";
porReferencia($num);
echo "Valor después de la llamada porReferencia(): $num<br>\n";
?>
```


10. Uso y definición de funciones

- **Novedad PHP 8:** es posible transformar el contenido de una tabla en una lista de argumentos en una llamada de función, con el operador ... (puntos suspensivos...)



```
1 + 2 + 3 = 6
1 + 2 + 4 = 7
```

```
operador-suspensivo.php
Archivo  Editar  Ver

<?php
// Función con tres argumentos que devuelve la suma
// de los tres argumentos.
function suma($valor1,$valor2,$valor3) {
    return $valor1 + $valor2 + $valor3;
}
// Transformación del contenido de un tabla en
// lista de argumentos.
$valores = [1,2,3];
echo '1 + 2 + 3 = ',suma(...$valores),'<br />';
// Lo mismo solo para una parte de los argumentos
// con una tabla definida directamente en la llamada.
echo '1 + 2 + 4 = ',suma(1,...[2,4]),'<br />';
?>
```

10. Uso y definición de funciones

- PHP permite el mecanismo de argumentos por defecto. Un ejemplo de esta característica es:

```
function hacerCafe($tipo="capuchino")  
{  
    return "he hecho un café $tipo\n";  
}
```

- En la llamada a esta función se obtendrá una frase u otra según se llame:

```
echo hacerCafe();  
echo hacerCafe("expreso");
```

- En el caso de tratarse de una función con argumentos por defecto y argumentos normales, los argumentos por defecto deberán estar agrupados al final de la lista de argumentos

ACTIVIDAD

- Revisar los ejemplos de estas diapositivas, ejecutarlos en el servidor y comprobar que el resultado es el esperado.
- Ejercicios:
 1. Implementa un programa en PHP que muestre la información que devuelve la función `phpinfo()`.
 2. Implementa un programa en PHP que muestre por pantalla **una tabla de dimensión 10x10 con los números del 1 al 100.**
 3. Copia la tabla del ejercicio 3 y muéstrala coloreando las filas alternas, y haciendo que el tamaño sea una constante: `define(TAM, 10)`.
 4. Almacena en una tabla los 10 primeros números pares e imprímelos, uno por cada línea.
 5. Imprime los valores del vector asociativo siguiente usando la estructura de control **foreach**:

```
$v[1]=90;  
$v[30]=7;  
$v['e']=99;  
$v['hola']=43;
```

ACTIVIDAD

- Implementa un programa en PHP que almacene la siguiente información anidada:
 - Comunidad autónoma (ANDALUCÍA).
 - Provincias (De Andalucía).
 - De cada provincia, tres ciudades cualesquiera.
 - De cada ciudad cualquiera, su población actual.
- Se deberá mostrar dicha tabla de forma similar a los ejemplos que hemos visto en clase, ordenando la información de la siguiente manera:
 - Provincias: por orden alfabético.
 - Ciudades por provincia: por número de habitantes.
- **Cualquier variación de este ejercicio sirve para dominar lo visto hasta ahora y preparar las actividades de la próxima semana.**