

# TEMA 2.1

Introducción al lenguaje PHP.

# Introducción al lenguaje PHP (Personal Home Page)

- ❑ PHP es un lenguaje de programación interpretado para desarrollo web en el lado del servidor.
- ❑ Es el más extendido para desarrollo en servidor, y es habitual que PHP sea un módulo del servidor web.
- ❑ Es flexible y permite programar pequeños scripts con rapidez.
- ❑ La sintaxis es parecida a la de lenguajes extendidos como Java y C.
- ❑ En desarrollo web, es muy habitual utilizar PHP incrustado en ficheros HTML, como un bloque de código entre las etiquetas:

`<?php código PHP ?>`

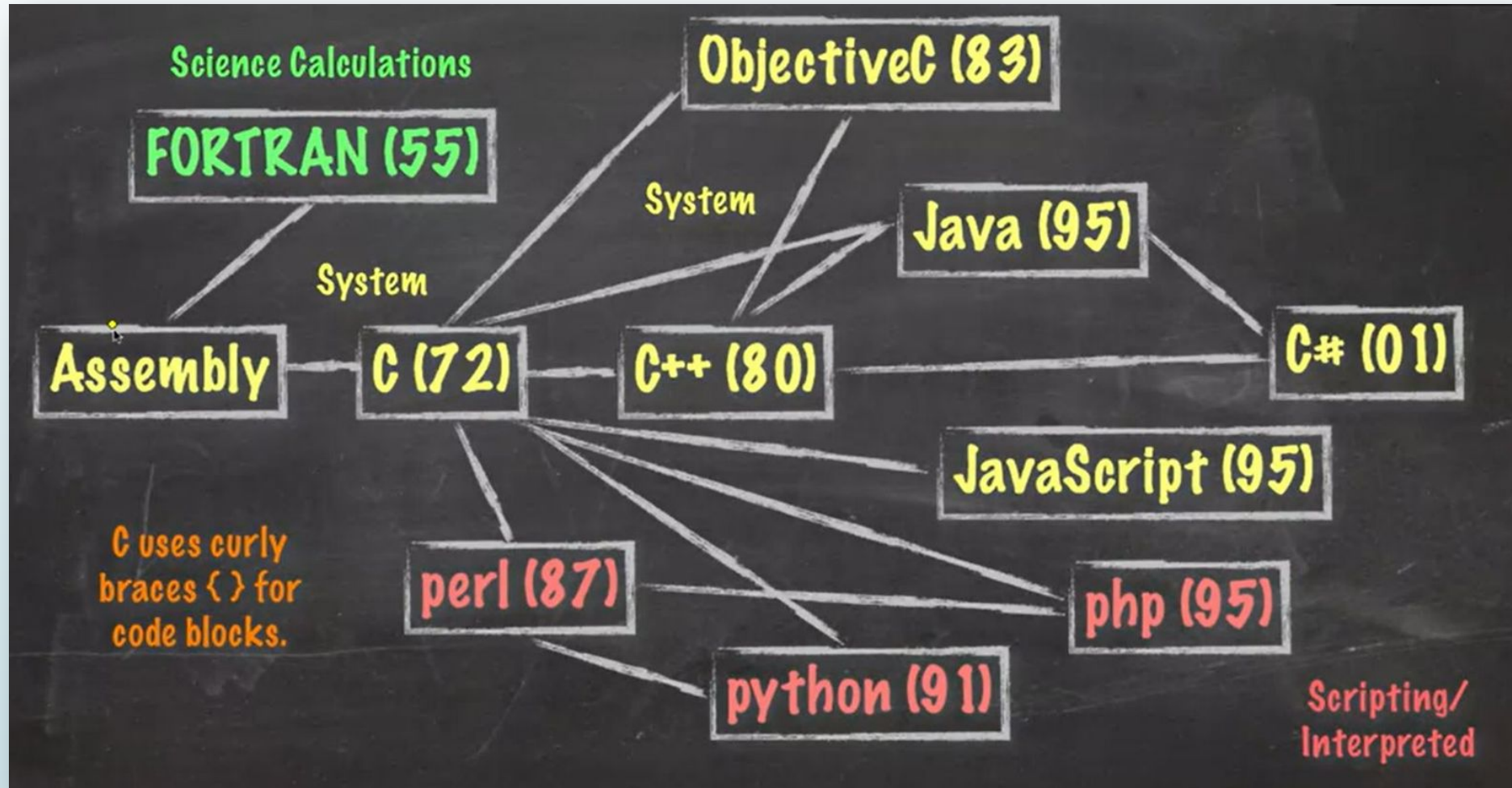
- ❑ Los ficheros se guardan con la extensión `.php`

# Cronología de PHP

- ❑ **1995:** La versión 1.0 surgió, cuando Rasmus Lerdorf creó unos scripts CGI para su página personal. Su intención no era crear un nuevo lenguaje de programación.
- ❑ 1997: Se reescribió el parser y se creó la versión php 3
- ❑ 2004: Salió la versión php 5, con soporte de orientación a objetos e interfaz con base de datos
- ❑ 2005: Versión 6, inicio de integración con Unicode
- ❑ 2020: Versión 8, con importantes mejoras tales como tipos static, un compilador, y mejoras en la sintaxis de clases, herencia y métodos abstractos, así como interesantes librerías adicionales.
- ❑ 2021: Versión 8.1, se incluyen varias mejoras, entre ellas nuevos tipos de datos.
- ❑ **Noviembre 2022:** Versión 8.2
- ❑ Actualmente en pruebas, la versión 8.3. No se recomienda para producción.

**Conclusión:** Es un lenguaje de larga trayectoria y mejoras continuas

# Cronología de PHP



# Sintaxis PHP: Cómo se ejecuta

- ❑ **¿ Dónde se coloca ?**: En cualquier parte intercalado dentro de código HTML, delimitado siempre por:

`<?php . . . ?>`

- ❑ **¿ Cómo se ejecuta ?**:

1. En un ciclo web request-response, siendo invocado por el cliente en una request. Por ejemplo, voy a un navegador y escribo:

<http://miservidor/miprograma.php>

2. Desde línea de comando. En este caso la salida sale por consola, no se recomienda. Ejemplo:

`$ php miprograma.php`

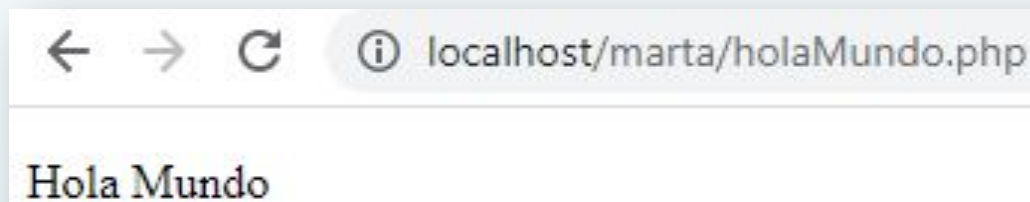


# Ejecución PHP

Dónde escribirlo: Intercalado en HTML

```
<body>
  <p>
    <?php echo "Hola Mundo" ?>
  </p>
</body>
```

1. Ejecutar desde Navegador



A screenshot of a web browser window. The address bar shows 'localhost/marta/holaMundo.php'. The page content displays 'Hola Mundo'.

2. Ejecutar desde Línea de comandos Windows

```
c:\xampp\php>php ..\htdocs\marta\holaMundo.php
<!DOCTYPE html>
<html>
  <head>
    <title>Saludos</title>
  </head>
  <body>
    <p>
      Hola Mundo
    </p>
  </body>
</html>

c:\xampp\php>
```

3. Ejecutar desde Línea de comandos Ubuntu

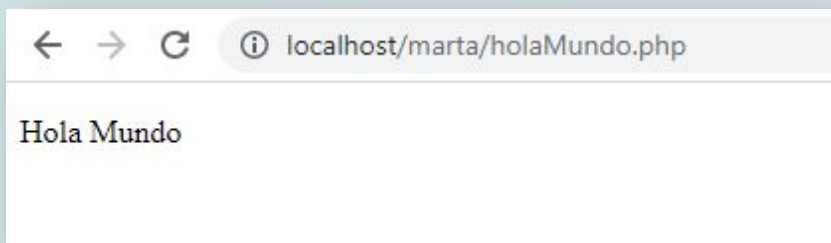
```
alumno@alumno-VirtualBox:/opt/lampp$ ./bin/php ./htdocs/marta/index.php
<h1>Hola mundo soy Marta</h1>
<p>
  Hola amigos
  He calculado 6 * 7 y me da: 42
</p>
<p> Otro párrafo, why not?</p>alumno@alumno-VirtualBox:/opt/lampp$
```

# Ejercicio 1:

## Hola Mundo



```
holaMundo.php x
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Saludos</title>
5   </head>
6   <body>
7     <p>
8       Hola Mundo
9     </p>
10  </body>
11 </html>
12
13
```



### Primer programa con PHP: Hola Mundo

1. Crea un fichero **holaMundo.php**, con el código **HTML** necesario para visualizar la frase "Hola Mundo" en un navegador.
2. Crea una carpeta con *tu nombre* en el servidor web y pon en ella tu fichero holaMundo.php
3. Inicia el servidor web
4. En un navegador, escribe la URL  
`localhost/tunombre/holaMundo.php`
5. **Modifica** holaMundo.php para sustituir el saludo "Hola Mundo" por código PHP. Este código mostrará el mismo saludo.
6. Refresca el navegador y ve a "Ver código fuente de la página". ¿Por qué no aparece el código php?

**Pista:** Puedes utilizar la función **echo** de PHP

# Generación de contenido

- Los programas PHP generan contenido dinámico en el documento mediante las siguientes sentencias:

- Con **echo** "expresión". Admite varios parámetros separados por comas: **echo** "hola","adios"

`<?php echo 3+2 ?>`

Salida: 5

`<?php echo "Hola Mundo" ?>`

Salida: Hola Mundo

- Como **abreviatura** de un `<?php echo`, podemos utilizar `<?=>`

`<?= "Hola Mundo" ?>`

Salida: Hola Mundo

`<?= 3+2 ?>`

Salida: 5

- Con **print**(expresión)

`<?php print(3+2); ?>`

Salida: 5

`<?php print("Hola"); ?>`

Salida: Hola

- En un programa PHP, cada sentencia termina en ;

```
<?php
    echo "Hola";
    echo "Adios";
?>
```

Salida: HolaAdios



## Ejercicio 2:



### Ecoss

1. Crea un documento .php llamado ecos.php
2. El documento generará salidas de las tres formas estudiadas en la transparencia anterior.
3. Súbelo a tu servidor web, o ponlo en la carpeta de tu servidor web local.
4. Haz un request al fichero desde un navegador.
5. Abre un terminal
6. Ejecuta el fichero desde línea de comandos.
7. Las salidas deben aparecer en líneas diferentes

# Comentarios

- En PHP, como en otros lenguajes, es posible incluir comentarios que no serán ejecutados.
- Hay dos formas de poner comentarios en PHP:

1. Comentarios de una sola línea (Figura 1)

`// Este es un comentario de una sola línea, tipo Java`

2. Comentarios multilínea

`/* Puedo escribir comentarios que ocupen  
varias líneas, siempre  
que vayan entre  
los símbolos en rojo */`

3. Comentarios tipo shell:

`# Esto es un comentario de una sola línea, tipo shell`

```
<?="// Pretendo ser un comentario, pero en realidad soy un echo"?>  
<?php "//Pretendo ser un comentario, pero en realidad soy un string"?>  
<?php // Yo SI soy un comentario en PHP ?>
```

# Variables en PHP (1)

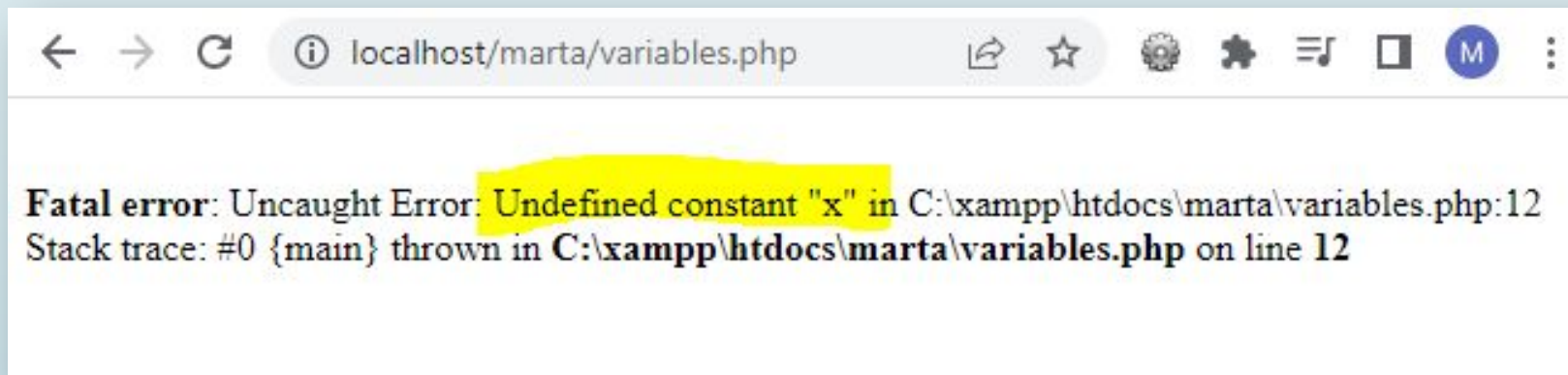
- ❑ Las variables almacenan información.
- ❑ En algunos lenguajes hay que indicar qué tipo de información van a guardar antes de poder utilizarlas (si número, o caracteres, etc). Son lenguajes tipados, que requieren una "declaración" de la variable, indicando el tipo de dato que almacenará.
- ❑ En PHP no es necesario indicar el tipo de las variables, ni tampoco declararlas.
- ❑ Conviene asignarles un valor inicial antes de utilizarlas, sino da error.
- ❑ En PHP los nombres de las variables:
  - ❑ Empiezan por el símbolo **\$**, seguido por letra, número o el símbolo "\_". Ejemplo **\$vidas**
  - ❑ En la práctica:
    - ❑ No utilizar el símbolo "\_"
    - ❑ Empezar *siempre* por letra minúscula
    - ❑ Las mayúsculas son distintas a las minúsculas, así **\$vidas** es una variable distinta a **\$Vidas**
- ❑ Si no empiezan por \$, entonces no son variables, se ignoran y se produce un error

## Variables en PHP (2)

- El siguiente fragmento de código php da error, pues x no es una variable.

```
<?php
// x parece una variable, pero como no hay $ delante no lo es
$x = 2;
$y = x+5;
print $y;
?>
```

- Si abrimos ese documento nos da error



# Constantes en PHP

- Una constante almacena información. A diferencia de las variables, no puede ser modificada.
- Su nombre no empieza por \$. Puede llevar mayúsculas, minúsculas, y "\_"
- Se recomienda ponerlas en mayúsculas.
- Para definir una constante, es necesario asignar un valor inicial de una de las dos formas siguientes:

<?php

```
define('IVA',0.21); // Definimos una constante llamada IVA, asignando un valor de 0,21
echo IVA; // Podemos mostrar la constante IVA con echo o utilizarla en operaciones, pero no modificarla
const PI = 3.1416; // Otra forma de definir una constante, en este caso se llama PI y es 3,1416
echo PI; // Toda constante puede utilizarse en expresiones, pero no modificarse
```

?>

- Si miramos el manual de php, indica que no se recomienda utilizar el define para definir constantes dentro de una clase. En cambio, recomienda utilizar "const".

<https://www.php.net/manual/en/language.constants.php>

# Tipos de datos en PHP

Aunque no se declaran tipos de datos para las variables, PHP trabaja internamente con los siguientes tipos de datos

## ▣ **Escalares:**

- ▣ Boolean: Se ha asignado un valor lógico (true o false): `$quedaSaldo = true;`
- ▣ Integer: Se ha asignado un número entero: `$edad = 25;`
- ▣ Float: Se ha asignado un número decimal: `$valorPI = 3.1416;`
- ▣ String: Se ha asignado una cadena de caracteres: `$saludo = "Hola Mundo";`

## ▣ **Compuestos:** Los veremos más adelante:

- ▣ Array: Almacena colecciones de elementos, la gestión es muy potente
  - ▣ Object: Soporta la orientación objetos de forma completa
  - ▣ Callable: Para pasar funciones como parámetro de otras funciones.
  - ▣ Resource: Para recursos externos, como por ejemplo una Base de Datos
- ▣ **null:** Si una variable no ha sido asignada, o bien tiene asignado un null, es de este tipo. Su valor es siempre null: `$vacio = null;`



## Ejercicio 3:

### Variables

1. Crea un documento php llamado variables.php
2. Asigna valores a diversas variables, cubriendo los tipos escalares de la página anterior
3. Haz un echo de las variables indicadas
4. Añade comentarios indicando el tipo de datos que crees ha asignado PHP internamente a tus variables
5. Pruébalo en un navegador

# Strings en PHP

- Un String está formado por uno o varios caracteres
- Un valor tipo string puede definirse con comillas simples ('hola') o dobles ("hola"). Es mejor utilizar las comillas dobles, porque admite más posibilidades.
- El carácter de escape es el \

```
<php
```

```
    echo "El símbolo \"@\" se utiliza en las direcciones de email"
```

```
?>
```

- Para concatenar dos strings, se utiliza el símbolo "." No puede utilizarse el símbolo "+" para concatenar dos strings en php.
- Si queremos incluir un salto de línea en un string, "\n" no funcionará como esperamos, porque nuestra salida es HTML para visualización en un navegador. Lo suyo es utilizar <br>:

```
<?php
```

```
    echo "Esta es la primera línea <br> Esta es la segunda línea";
```

```
?>
```

## Ejemplos:

cadenas.php



Resultado

```
cadenas.php > html > head > title
1  <!DOCTYPE html>
2  <html lang="es">
3    <head>
4      <meta charset="UTF-8"/>
5      <title>Cadenas</title>
6    </head>
7    <body>
8      <?php
9        print("Esto es un string \"simple\" con salto de línea al final <br>");
10       echo "Puedes escribir el string de esta forma
11         en tu programa, saltando de línea,
12         se permite y no da error en PHP<br>";
13       echo "Línea 1 <br> Línea 2", "<br>gracioso";
14       echo "<br>El símbolo \"@\" se utiliza en las direcciones de email";
15     ?>
16   </body>
17 </html>
```

Cadenas x Ambient Study Mu x CorreoWeb Educa x TEMA 2.1 - Introdu x

localhost/marta/cadenas.php

Esto es un string "simple" con salto de línea al final  
Puedes escribir el string de esta forma en tu programa, saltando de línea, se permite y no da error en PHP  
Línea 1  
Línea 2  
gracioso  
El símbolo "@" se utiliza en las direcciones de email

# Expresiones y operadores (1). Operadores numéricos.

- Como en el resto de los lenguajes de programación, en PHP se dispone de operadores y expresiones que permiten realizar cálculos y comparar valores.
- Su sintaxis es igual a la de muchos otros lenguajes.
- Asignación: Para asignar un valor a una variable se utiliza el signo "="
- Los principales **operadores numéricos** son:

```
$total = $precio + $iva; // Suma
```

```
$descuento = $precio * $porcentaje / 100; // Multiplicación
```

```
$importeFinal = $total - $descuento; // Resta
```

```
$elevadoATres = $numero ** 3; // Potencia
```

```
$entreTres = $numero / 3; // División
```

```
$restoDeDivisión = $numero % 3; // Módulo (resto) de una división
```

## Expresiones y operadores (2). Comparaciones.

- Las **expresiones de comparación** devuelven **true** si se cumplen, o **false** si no se cumplen. Son las siguientes

- Valores **iguales**. Antes de comparar PHP convierte tipos para poder hacerlo.

`$a == $b`

- Variables **idénticas** en tipo y valor. No se convierte, si el tipo es distinto no mira el valor, es false.

`$a === $b`

- Valores **distintos**. Antes de comparar convierte tipos.

`$a != $b`

`$a <> $b`

- No idénticas**, se cumple si bien los tipos, o bien los valores no son iguales.

`$a !== $b`

- Menor**: Un valor es menor que el otro

`$a < $b`

- Menor o igual**: Un valor es menor o igual que el otro

`$a <= $b`

- Mayor**: Un valor es mayor que el otro

`$a > $b`

- Mayor o igual**: Un valor es mayor o igual que el otro

`$a >= $b`

## Expresiones y operadores (3). Operadores lógicos.

- Las **expresiones lógicas** utilizan operadores que se aplican a variables o expresiones de tipo lógico (booleano).
- Cuando se aplica el operador, el resultado será otro booleano con valor **true** o **false**.
- Las más habituales son:

□ "Y", implica que deben cumplirse ambas expresiones o ser true ambas variables

```
$attractiva = ($rica and $guapa) // Primera forma, utilizando el operador "and". ¡Usar paréntesis!
```

```
$attractiva = $rica && $guapa // Otra forma, && es equivalente al operador "and"
```

□ "O", Implica que debe cumplirse al menos una expresión o ser true al menos una variable

```
$attractiva = ($rica or $guapa) // Primera forma, utilizando el operador "or". ¡Usar paréntesis!
```

```
$attractiva = $rica || $guapa // Otra forma, || es equivalente al operador "or"
```

□ Xor, implica que se cumple sólo una de las dos expresiones o variables

```
$pasa = $socio xor $paga_entrada
```

□ Not: Implica que no se cumple la expresión o variable

```
$socio = !$pobre
```



## Expresiones y operadores (4). Abreviaturas y Prioridad.

- Algunas **expresiones habituales** pueden simplificarse utilizando abreviaturas. Las más comunes son las siguientes:

`$x++` // Se suma 1 a la variable `$x`

`$x--` // Se resta 1 a la variable `$x`

`$x += $y` // Equivalente a `$x = $x + $y`, suma ambas y el resultado lo pone en `$x`

`$x *= $y` // Multiplica ambas variables y el resultado lo pone en `$x`

`$x /= $y` // Divide ambas variables y el resultado lo pone en `$x`

`$x .= $y` // Concatena ambas variables y el resultado lo pone en `$x`

- **Prioridad:** Cuando una expresión consta de varios operadores, el orden de cálculo no da igual a efectos del resultado. Si escribo `2 + 4 / 3`, no da igual hacer antes la suma que la división.
- Los lenguajes de programación aplican primero los operadores más prioritarios, no siguen el orden en que aparecen, sino que ejecutan primero los de mayor prioridad.
- En PHP, si escribo `2 + 4 / 3`, ejecutará primero la división y luego la suma. La prioridad es:
  1. Paréntesis: Ante la duda, se ponen paréntesis. ¡ El lenguaje los respeta siempre !
  2. Not o negación (!)
  3. División y multiplicación (`/`, `*`)
  4. Suma y resta (`+`, `-`)

## Conversiones o casting en PHP (1)

- Al no ser un lenguaje tipado, en ocasiones PHP necesita realizar conversiones de datos para realizar operaciones. Normalmente cuando las variables implicadas en la expresión son de tipos diferentes.
- PHP realiza conversiones implícitas (automáticas) agresivas. A esto se le llama **casting**. Si no puede realizar la operación, cambia la variable para poder hacerlo. Por ejemplo:
  - En cualquier **división** PHP convierte a **float** las variables, aunque sean enteras.
  - En **sumas**, las variables booleanas con true son convertidas al entero 1.
  - Si en una suma una variable es un string, si el contenido no es numérico da error.
  - En una concatenación de caracteres, si una variable es un entero, lo convierte a string antes de concatenarlo.
- A veces el programador quiere controlar la conversión, porque la que hay por defecto no le interesa. Para ello, PHP proporciona operadores de **casting explícito**, poniendo entre paréntesis el tipo de datos al que queremos que convierta:

```
echo (int) 9.9 - 1 // Convierte el 9.9 en entero, lo que trunca a 9. Resultado: 8
```

## Conversiones o casting en PHP (2). Ejemplos.

```
// CASTING
$a = 56;
$b = 12;
$c = $a / $b;
echo "<br> C = $c";

$d = "100" + 36.5 + true;
echo "<br> D= " . $d;

$e = (int) 9.9 - 1;
echo "<br> E= $e";

$f = "hola" . 25;
echo "<br>F= $f";

$h = true + false;
echo "<br>H= $h";
echo "<br>false como string=" . (string)false;
echo "<br>false como int= " . (int) false;

$g = "hola" + 25;
echo "<br>G= $g";
```



localhost/marta/expresiones.php

C = 4.6666666666667  
D= 137.5  
E= 8  
F= hola25  
H= 1  
false como string=  
false como int= 0  
**Fatal error:** Uncaught TypeError: Unsupported operand types: string + int in /opt/lampp/htdocs/marta/expresiones.php:66 Stack trace: #0 {main} thrown in /opt/lampp/htdocs/marta/expresiones.php on line 66

## Conversiones implícitas a booleano

- A veces, se utiliza variables en expresiones que requieren conversión a booleano. En estos casos, PHP hace una conversión al valor true o false según la tabla siguiente:

Tipo de variable	Valor convertido a booleano
integer	Si es <b>0</b> , se convierte a <b>false</b> , sino a true
float	Si es <b>0.0</b> se convierte a <b>false</b> , sino a true
string	Si es una cadena <b>vacía</b> o es <b>"0"</b> , se convierte a <b>false</b> , en otros casos a true
Variables no inicializadas	Siempre <b>false</b>
null	Siempre <b>false</b>
array	Si <b>no tiene elementos</b> es <b>false</b> , en caso contrario true

# Ejercicio 4:

## Casting

```
echo "A".false."B<br>";  
echo "X".true."Y<br>";  
$a = (123=="123");  
$b = (123 == "100"+23);  
$c = (false == "0" );  
$d = ( (5<6) == "2"-"1" );  
$e = ( (5<6) === true );  
echo "Igualdad a es $a" . "<br>";  
echo "Igualdad b es $b" . "<br>";  
echo "Igualdad c es $c" . "<br>";  
echo "Igualdad d es $d" . "<br>";  
echo "Igualdad e es $e" . "<br>";
```

1. Crea un documento php llamado casting.php
2. Incluye el código de la izquierda y pruébalo
3. Modifica los == por === y observa qué sucede

## Variables predefinidas

- En PHP hay muchas variables predefinidas, que contienen información sobre el servidor, el entorno de trabajo, o bien datos enviados por el cliente en un request.
- **Superglobales:** Son variables PHP con información del sistema, están siempre disponibles. Algunas de ellas son las siguientes:

Variable	Descripción de su contenido
\$_SERVER	Información sobre el servidor
\$_GET	Parámetros enviados con el método GET en la URL
\$_POST	Parámetros enviados con el método POST (formularios)
\$_FILES	Ficheros subidos al servidor
\$_COOKIE	Cookies enviadas por el cliente
\$_SESSION	Información de sesión
\$_REQUEST	Información de \$_GET, \$_POST y \$_COOKIE en una variable única
\$_ENV	Variables de entorno



# Funciones predefinidas

- En PHP hay más de 1000 funciones predefinidas que pueden utilizarse directamente. Son estas funciones las que aportan potencia a este lenguaje.
- Estas funciones son de varios tipos:
  - Criptográficas
  - Acceso a diversas bases de datos
  - Compresión de archivos
  - Gestión del sistema de archivos
  - Procesamiento y generación de imágenes
  - Relacionadas con Email
  - ... y muchas más
- A lo largo del curso iremos practicando con las más habituales
- El manual oficial de PHP dispone de una traducción al español realizada por personas. La parte correspondiente a las funciones predefinidas está en:  
<https://www.php.net/manual/es/funcref.php>
- PHP también permite que definamos nuestras propias funciones. Lo veremos más adelante.



<https://www.php.net/docs.php>