

Taller de Programación Web

MSc. Yuliana Jáuregui Rosas

























Universidad
Tecnológica
del Perú

Conocimientos previo

- ¿Para qué se usa JavaScript?
- ¿Entre que etiquetas colocamos un código JavaScript?

Lenguaje PHP



Language Rank	Types	Spectrum Ranking
1. Python	  	100.0
2. C++	  	99.7
3. Java	  	97.5
4. C	  	96.7
5. C#	  	89.4
6. PHP		84.9
7. R		82.9
8. JavaScript	 	82.6
9. Go	 	76.4
10. Assembly		74.1

Logro



Al finalizar esta sesión el
alumno desarrolla
aplicaciones web empleando
el lenguaje PHP.

Desaprende lo que te limita

Temario



- PHP
- Funcionamiento
- Etiquetas de PHP
- Comentarios
- Definiciones de sentencias
- Variables
- Tipos de datos
- Operadores
- Métodos Post, Get, Request
- Arreglos
- Estructuras de control
- Ejemplos

PHP

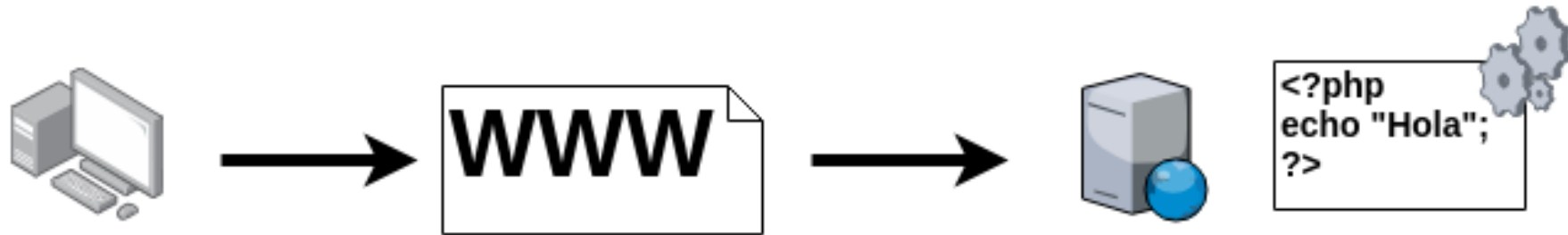
- PHP (Hypertext Preprocessor) es un lenguaje de código abierto adecuado para el desarrollo web que puede ser incrustado en HTML. Creado por Rasmus Lerdorf en 1994.
- Permite embeber pequeños fragmentos de código dentro de una página HTML.
- PHP a diferencia de JavaScript ejecuta el código en el servidor (Backend), por lo que puede generar una página según el contenido de un formulario, según el contenido de una base de datos, según la hora del servidor, etc.

Funcionamiento

- Un usuario hace una petición de página web mediante un navegador (Google Chrome, Firefox, Mozilla, Safari, etc.)
- El servidor web recibe la petición de una pagina HTML que contiene código PHP, el cual no puede ser interpretado por los navegadores, entonces el servidor lo procesa antes de responder la petición.
- El navegador recibe el código HTML procesado y lo muestra al usuario.

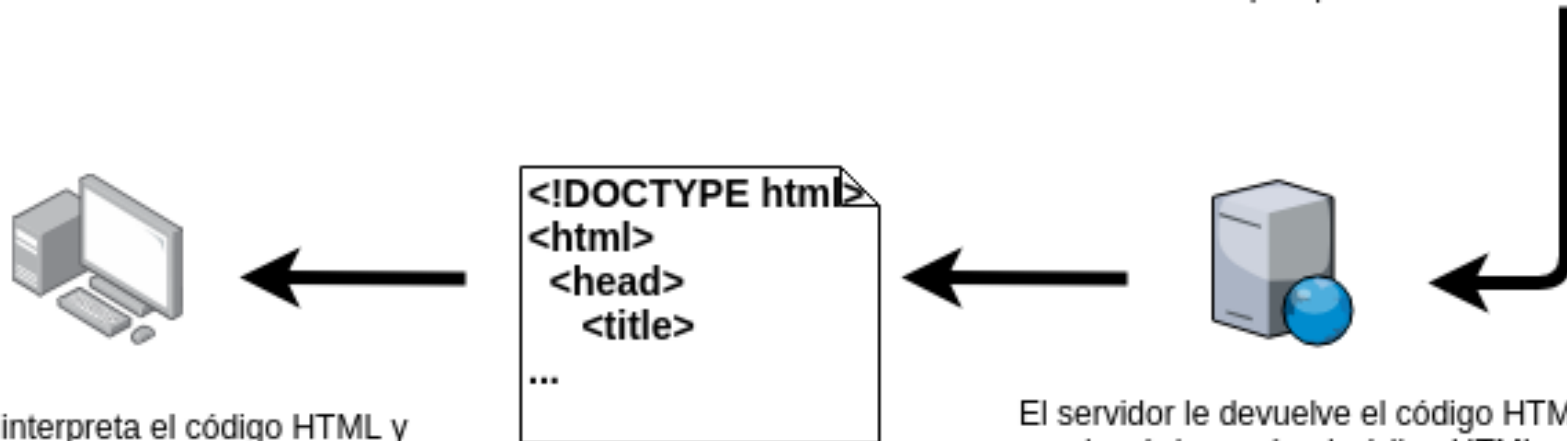
El PHP al ser interpretado en el servidor, el usuario no puede modificar en su ordenador haciendo así, aplicaciones más seguras.

Funcionamiento



El visitante abre el navegador Web, Google Chrome o Firefox, y accede a esta página.

El servidor Web atiende la petición del usuario Web cuya petición tiene una cabecera dónde se le dice que información quiere recuperar de la Web a la que quiere ir.

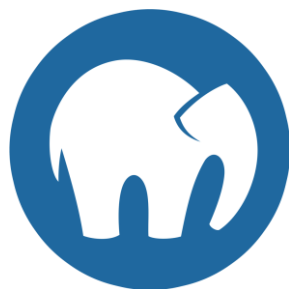


El navegador interpreta el código HTML y lo muestra en su pantalla para que lo pueda visualizar.

El servidor le devuelve el código HTML estático de la propia página, más el código HTML generado de forma dinámica por el PHP.

Servidores locales

- Para que PHP funcione es necesario simular un servidor remoto como: XAMPP, WAMPServer, MAMP, BitNami, EasyPHP, NMP Server, UwAmp, etc.



Desaprende lo que te limita

Lado del cliente



Lado del servidor



Desaprende lo que te limita

Etiquetas de PHP

- Las etiquetas de apertura y cierre indican a PHP dónde empieza y finaliza la interpretación del código. Estas pueden ser:

```
<?php [expresión] ?>
```

```
<?= [expresión] ?>
```

- Ejemplo:**

```
<?php echo ("Conociendo PHP"); ?>
```

```
<?= echo ("Conociendo PHP"); ?>
```

- Si un fichero contiene solamente código de PHP, se puede omitir la etiqueta de cierre de PHP al final del mismo.

Delimitación de sentencias

- Las instrucciones en PHP finalizan con un punto y coma ";".
- La última instrucción, la que se encuentra antes del cierre de etiqueta no es necesario que finalice con el punto y coma.

Salida de datos: echo y print

echo y **print** realizan la misma función, con pequeñas diferencias:

- **echo** tiene un retorno void , mientras que **print** devuelve un int con un valor de 1.
- **echo** puede tomar múltiples argumentos (sin paréntesis), mientras que **print** solo toma un argumento.
- **echo** es un poco más rápido que la **print**.
- Tanto el **echo** como el **print** son construcciones de lenguaje, no funciones. Eso significa que no requieren paréntesis alrededor de sus argumentos. Para consistencia estética con funciones, se pueden incluir paréntesis.

Ejemplos

```
echo "Hola mundo";  
print "Hola mundo";
```

```
echo "Para identificar caracteres se hace \"así\".";  
print "Para identificar caracteres se \"hace así\".";
```

Comentarios

- A la hora de programar es conveniente añadir comentarios para poder saber que es lo que hace cada parte del código.

Comentario de una sola línea

```
//Esto es un comentario de una línea
```

Comentario de varias líneas

```
/* Esto es un comentario que tiene mas de una  
línea es decir, varias líneas */
```

Comentario de una sola línea

```
# Esto también es un comentario de una sola linea
```


Variables

- Las variables en PHP son definidas comenzando siempre con el símbolo dólar (\$).

Por ejemplo:

```
<?php  
    $total = 300  
?>
```

Variables

- El tipo de una variable usualmente no lo declara el programador; al contrario, es decidido en tiempo de ejecución por PHP dependiendo del contexto en el que se emplea dicha variable.
- Así se tiene los siguientes tipos de datos:

Tipos de datos en PHP

Tipo	Descripción	Ejemplo
booleano	Expresa un valor que indica verdad o falsedad.	<code>\$valor = true;</code>
entero	Expresa los números enteros tanto positivos como negativos.	<code>\$a = 1234;</code>
real	Expresan los números de punto flotante.	<code>\$a = 1.234;</code>
cadena	Representa a una serie de caracteres donde cada carácter es lo mismo que un byte. Se puede especificar entre comillas dobles (" ") o comillas simples (' ').	<code>\$cadena1='Sean bienvenidos';</code> <code>\$cadena2="Es un lindo día";</code>
array	Permite almacenar un conjunto de datos.	<code>\$planeta[0]="Tierra";</code> <code>\$planeta[1]="Venus";</code> <code>\$planeta[2]="Júpiter";</code> <code>\$planeta[3]="Marte";</code>

Etiquetas HTML en una variable PHP

- Una variable también puede contener etiquetas HTML.
- Ejemplo:

```
$titulo='<h1>Mi pagina</h1>';
```

Variable PHP en una etiqueta HTML

- Se puede utilizar una variable PHP como valor del atributo de una etiqueta.
- Ejemplo:

```
<input type="text" value="<?= $dato ?>">
```

Constantes

- Las constantes en PHP no llevan el signo dólar (\$).
- Antes de la versión PHP 5.3 solo se definía constantes mediante la palabra reserva **define**.
- Después de la versión PHP 5.3 para definir constantes también se puede usar la palabra reservada **const**.

Ejemplos

```
<?php
define('saludo',"Hola a todos");
echo saludo."<br>";
define ('colores', array('rojo', 'verde', 'azul'));
echo colores[1]."<br>";

const val=24;
echo val."<br>";
const paises=array('Perú', 'Argentina', 'México');
echo paises[0];
?>
```

Hola a todos
verde
24
Perú

Operadores aritméticos

Símbolo	Definición	Expresión	Resultado \$a = 5 y \$b = 2
+	Adición	$r = a + b;$	7
-	Sustracción	$r = a - b;$	3
*	Multiplicación	$r = a * b;$	10
/	División	$r = a / b;$	2
%	Resto	$r = a \% b;$	1
**	Exponenciación	$r = a^{**}2$	25
++	Incremento	$a++$	6
--	Decremento	$b--$	1

Operadores relacionales

Símbolo	Definición	Expresión	Resultado \$a = 10 y \$b = 7
>	Mayor que	$a > b$	Verdadero
>=	Mayor o igual que	$a \geq b$	Verdadero
<	Menor que	$a < b$	Falso
<=	Menor o igual que	$a \leq b$	Falso
==	Igual a	$a == b$	Falso
===	Exactamente igual	$a === b$	Falso
!=	Diferente a	$a != b$	Verdadero
!==	Diferente en valor a tipo	$a !== b$	Verdadero
<>	Diferente a (igual a !=)	$a <> b$	Verdadero
<=>	Comparador de orden	$a <=> b$	1

Operadores lógicos

Símbolo	Definición	Expresión	Resultado \$a = true y \$b = false
and	Operación lógica “y”	\$a and \$b	Falso
or	Operación lógica “o”	\$a or \$b	Verdadero
xor	Operación lógica “o exclusivo”	\$a xor \$b	Verdadero
!	Operación lógica “negación”	!\$a	Falso
&&	Operación lógica “y”	\$a && \$b	Falso
	Operación lógica “o”	\$a \$b	Verdadero

Operadores de comparación

Símbolo	Nombre	Descripción	Ejemplo	
?	Elvis	Evalúa si un resultado se ejecuta o no.	<pre><?php \$var1=10; \$var2=15; (\$var1<\$var2)?(\$resp="\$var1 es menor a \$var2"):(\$resp="\$var2 es menor a \$var1"); echo \$resp; ?></pre>	
			10 es menor a 15	
<=>	Nave espacial	Devuelve un valor entero luego de comparar dos variables	Integer	
			echo 1 <=> 1; // 0	echo 1.5 <=> 1.5; // 0
			echo 1 <=> 2; // -1	echo 1.5 <=> 2.5; // -1
			echo 2 <=> 1; // 1	echo 2.5 <=> 1.5; // 1
			String	
			echo "a" <=> "a"; // 0	echo [] <=> []; // 0
<=>	Nave espacial	Devuelve un valor entero luego de comparar dos variables	echo "a" <=> "b"; // -1	echo [1, 2, 3] <=> [1, 2, 3]; // 0
			echo "b" <=> "a"; // 1	echo [1, 2, 3] <=> []; // 1
			echo "a" <=> "aa"; // -1	echo [1, 2, 3] <=> [1, 2, 1]; // 1
			echo "zz" <=> "aa"; // 1	echo [1, 2, 3] <=> [1, 2, 4]; // -1
			Arrays	
			echo [] <=> []; // 0	echo [1, 2, 3] <=> [1, 2, 3]; // 0
			echo "a" <=> "b"; // -1	echo [1, 2, 3] <=> []; // 1
			echo "b" <=> "a"; // 1	echo [1, 2, 3] <=> [1, 2, 1]; // 1
			echo "a" <=> "aa"; // -1	echo [1, 2, 3] <=> [1, 2, 4]; // -1
			echo "zz" <=> "aa"; // 1	

Operadores de cadena

Símbolo	Definición	Expresión	Resultado
.	Cuando trabajamos con cadenas de caracteres tenemos un operador especial que es de concatenación. Sirve para unir una cadena con otra.	$\$a = \text{"Feliz"} . \text{"día"}$	$\$a = \text{"Feliz día"}$

Ejemplo

- Si se tiene las variables:

```
$nombre="Claudia";  
$apellidos="Torres Meza";  
$edad=16;
```

- Usamos el operador punto (.) para concatenar las variables.

```
echo $nombre." ".$apellidos." de ".$edad." años de edad";
```

- Otra opción para poder concatenar variables es colocando entre llaves a las variables y todas estas a su vez entre comillas doble.

```
echo "{$nombre} {$apellidos} de {$edad} años de edad";
```

- O simplemente:

```
echo "$nombre $apellidos de $edad años de edad";
```

Operadores de asignación

Símbolo	Expresión	Expresión abreviada	Resultado a = 10
+=	\$a=\$a+6	\$a+=6	16
-=	\$a=\$a-5	\$a-=5	5
*=	\$a=\$a*4	\$a*=4	40
/=	\$a=\$a/2	\$a/=2	5
%=	\$a=\$a%3	\$a%=3	1
.=	\$cad=\$cad."hola " \$cad=\$cad."mundo"	\$cad="hola " \$cad.="mundo"	"hola mundo"

Operadores para array

Símbolo	Definición	Expresión	Resultado
+	Unión	$\$a + \b	Unión de $\$a$ y $\$b$.
==	Igual a	$\$a == \b	TRUE si $\$a$ i $\$b$ tienen las mismas parejas clave/valor.
===	Exactamente igual	$\$a === \b	TRUE si $\$a$ y $\$b$ tienen las mismas parejas clave/valor en el mismo orden y de los mismos tipos.
!=	Diferente a	$\$a != \b	TRUE si $\$a$ no es igual a $\$b$.
<>	Diferente a (igual a !=)	$\$a <> \b	TRUE si $\$a$ no es igual a $\$b$.
!==	No identidad	$\$a !== \b	TRUE si $\$a$ no es idéntica a $\$b$.

Función unset()

- La función “unset()” permite quitar el valor de una variable.

- Ejemplo:

```
$texto = "hola";  
unset($texto);
```

- La variable texto luego de ejecutar la función unset() tiene valor Null.

Función isset()

- La función “isset()” indica que una variable se encuentra inicializada.
- Ejemplo:

```
if( isset($texto) ){  
    echo ("La variable texto tiene un valor");  
}
```

Función gettype() y var_dump()

- Devuelve el tipo de dato de una variable.
- Ejemplo:

```
$dia = 23;  
echo gettype($dia);
```

```
$dia = 23;  
echo var_dump($dia);
```

Métodos Post, Get y Request

- **Post**, **Get** y **Request** son métodos predeterminados del servidor, los cuales devuelven una variable en formato arreglo asociativo con llave y valor, se refiere a variables globales de PHP.
- Una variable que tenga **\$_** significa que es una **variable del servidor**.
- A través de esta variable se pueden enviar parámetros a través de formularios o enlaces.

Métodos Post, Get y Request

- Por el método **post** los valores son enviados de manera oculta para el navegador, y llegan al arreglo global `$_POST`.
- Por método **get** los valores son enviados a la vista del navegador, por la URL del script, y llegan al arreglo global `$_GET`.
- El arreglo global `$_REQUEST` del método **request** recibe tanto lo que llega por POST como lo que llega por GET.

Arreglos

- Los arreglos permiten almacenar un conjunto de elementos de diferentes tipos.
- Los arreglos de una dimensión también se conoce como vectores.
- Los vectores utilizan un par de corchetes [], dentro de los cuales se especifica un índice el cual indica la posición de un elemento del vector.
- Los índices inicializan en 0.
- Un arreglo puede tener mas dimensiones al agregarle mas subíndices.

Declaración de arreglos

- Declaración básica de un arreglo:

```
$datos;
```

- Asignación de elementos a un arreglo:

```
$datos[0] = 5;  
$datos[1] = "hola";  
$datos[2] = 17.5;
```

- Declaración y asignación de un arreglo mediante objetos

```
$datos = array(5, "hola", 17.5);
```

Declaración de arreglos asociativos

- Declaración asociativo

```
$datos = array( "ciclo" => 5, "saludo"=>"Hola", "nota" => 17.5);
```

- En este caso para acceder a los datos se utiliza el indicador y no los índices numéricos, es decir el índice puede ser una cadena.

```
$var1 = $datos['ciclo'];  
$var2 = $datos['saludo'];
```

Declaración de arreglos asociativos

```
<?php
    $persona = array(
        'nombre' => 'Carmen',
        'apellido' => 'Rios Rojas',
        'profesion' => 'Ingeniero de Sistemas',
        'edad' => 35
    );
?>
<pre>
    <?php print_r($persona); ?>
</pre>

<?php echo $persona['profesion'];?>

<pre>
    <?php print_r(array_values( $persona ) ); ?>
</pre>

<pre>
    <?php print_r(array_keys( $persona ) ); ?>
</pre>
```

```
Array
(
    [nombre] => Carmen
    [apellido] => Rios Rojas
    [profesion] => Ingeniero de Sistemas
    [edad] => 35
)
```

```
Ingeniero de Sistemas
Array
(
    [0] => Carmen
    [1] => Rios Rojas
    [2] => Ingeniero de Sistemas
    [3] => 35
)
```

```
Array
(
    [0] => nombre
    [1] => apellido
    [2] => profesion
    [3] => edad
)
```

Desaprende lo que te limita

Impresión de un arreglo

- Uso de la función `print_r()`

```
<?php
$colores = ['rojo', 'verde', 'azul', 'celeste'];
?>

<pre>
<?php print_r($colores); ?>
</pre>
```

Array
(
[0] => rojo
[1] => verde
[2] => azul
[3] => celeste
)

- Uso de la función `var_dump()`

```
<?php
$lenguajes = array('Java', 'C++', 'PHP', 'JavaScript');
?>

<pre>
<?php var_dump($lenguajes); ?>
</pre>
```

array(4) {
[0]=>
string(4) "Java"
[1]=>
string(3) "C++"
[2]=>
string(3) "PHP"
[3]=>
string(10) "JavaScript"
}

te limita

Funciones dentro de un arreglo

Función	Descripción	Ejemplo
array_pop()	borrar ultimo elemento y traerlo en variable	<pre><?php \$ultimo = array_pop(\$colores); ?> <h1><?php echo \$ultimo; ?></h1> <pre> <?php print_r(\$colores); ?> </pre></pre>
unset()	remover un elemento del array	<pre><?php unset(\$colores[2]); ?> <pre> <?php print_r(\$colores); ?> </pre></pre>

Funciones dentro de un arreglo

Función	Descripción	Ejemplo
array_shift()	remover primer elemento y agregarlo a variable	<pre><?php \$primero = array_shift(\$colores); ?> <h1><?php echo \$primero; ?></h1> <pre> <?php print_r(\$colores); ?> </pre></pre>
array_splice()	remover ciertos elementos y agregar otros	<pre><?php \$array = array_splice(\$colores, 1,1, array('morado', 'turqueza')); ?> <pre> <?php print_r(\$array); ?> </pre> <pre> <?php print_r(\$colores); ?> </pre></pre>

Función count()

- Cuenta el número de elementos en un array u objeto.

```
count ($arreglo; $modo)
```

- **\$arreglo**: Es la variable donde tenemos el objeto o array.
- **\$modo**: Admite los valores 0 y 1, con 0 (por defecto), no cuenta el número de elementos en un **array multidimensional**. Y con valor 1 cuenta los arrays de forma recursiva para sacar el número total de elementos.

Arreglos multidimensionales

- Permite almacenar datos anidando un arreglo dentro de otro.

Por ejemplo:

- Si queremos almacenar dentro de un arreglo el nombre, capital e idioma de un conjunto de países haremos:

```
<?php
    $pais=array
    (
        "pais1" =>array
        (
            "nombre"=>"Perú",
            "capital"=>"Lima",
            "idioma"=>"Español"
        ),
        "pais2" =>array
        (
            "nombre"=>"Brasil",
            "capital"=>"Brasilia",
            "idioma"=>"Portugués"
        )
    );
?>
```

Arreglos multidimensionales

```
<?php
    $persona = array(
        'datos' => array(
            'dni' => '15263487',
            'nombre' => 'Carlos',
            'apellido' => 'Paredes Fuentes'
        ),
        'hobby' => array(
            'idiomas' => array('ingles', 'postugues', 'frances'),
            'deporte' => array('natacion', 'futbol', 'basquet')
        )
    );
?>
<pre>

    <?php print_r($persona); ?>

    <br>
    <?php print_r($persona['hobby']['idiomas'][1]); ?>
</pre>
```

```
Array
(
    [datos] => Array
        (
            [dni] => 15263487
            [nombre] => Carlos
            [apellido] => Paredes Fuentes
        )
    [hobby] => Array
        (
            [idiomas] => Array
                (
                    [0] => ingles
                    [1] => postugues
                    [2] => frances
                )
            [deporte] => Array
                (
                    [0] => natacion
                    [1] => futbol
                    [2] => basquet
                )
        )
)
```

Función in_array()

- Permite buscar un elemento dentro de un arreglo. Devuelve un valor lógico (true, false).

```
<?php
    $colores = array('rojo', 'azul', 'celeste', 'verde');
    $existe = in_array('azulino', $colores );
?>
<pre>
    <?php var_dump($existe);    ?>
</pre>

<?php
    $persona = array(
        'nombre' => 'Luis',
        'pais'   => 'Peru',
        'edad'   => '28'
    );
?>

<?php $existe2 = in_array('Luis', array_values($persona)); ?>

<pre>
    <?php var_dump($existe2);    ?>
</pre>
```

bool(false)

bool(true)

Desaprende lo que te limita

Estructuras Condicional Simple

- Sintaxis:

```
if (condición){  
    instrucción;  
}
```

- Ejemplo:

```
if ($a > $b) {  
    echo "a es mayor que b";  
}
```


Estructuras Condicional doble



Universidad
Tecnológica
del Perú

- Sintaxis:

```
if (condición) {  
    instrucción 1;  
}  
else {  
    instrucción 2;  
}
```

- Ejemplo:

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

Desaprende lo que te limita

Estructuras Condicional anidada

- En PHP escribir “else if” (en dos palabras) o “elseif” (en una sola palabra) equivale lo mismo.
- Sintaxis:

```
if (condición1) {  
    instruccion1;  
}  
elseif (condición2) {  
    instruccion2;  
}  
elseif {  
    instruccion3;  
}
```

- Ejemplo:

```
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";  
}
```

Estructuras Condicional anidada

- Sintaxis:

```
switch (variable) {  
    case c1: instruccion1;  
        break;  
    case c2: instruccion2;  
        break;  
    ...  
    case cn: instruccionN;  
        break;  
    default: instruccionX;  
}
```

- Ejemplo:

```
switch ($nro) {  
    case 0: echo "nro es igual a 0";  
        break;  
    case 1: echo "nro es igual a 1";  
        break;  
    case 2: echo "nro es igual a 2";  
        break;  
    default: echo "nro no es igual a 0, 1 ni 2";  
}
```

Estructura repetitiva while

- La estructura repetitiva **while**, es aquella en que el cuerpo del bucle se repite mientras se cumple una determinada condición.
- Sintaxis:
- Ejemplo:

```
while (condición){  
    instrucciones;  
}
```

```
<?php  
$colores = array('rojo', 'verde', 'azul', 'amarillo','morado');  
$i = 0;  
while($i < count($colores)) {  
    echo $colores[$i]. '<br>';  
    if($i+1 === count($colores)) {  
        echo "Fin";  
    }  
    $i++;  
}  
?>
```

```
rojo  
verde  
azul  
amarillo  
morado  
Fin
```

Estructura repetitiva do - while

- Ejecuta una secuencia de instrucciones, repitiéndolas siempre que la condición (expresión lógica) sea verdadera.
- La ejecución finaliza cuando la condición (expresión lógica) es falsa.
- Sintaxis:
- Ejemplo:

```
do {  
    instrucciones;  
}  
while (condición);
```

```
<?php  
$nro=0;  
do{  
    $nro=$nro+2;  
    echo $nro."<br>";  
}  
while($nro<10);  
?>
```

```
2  
4  
6  
8  
10
```

Estructura repetitiva for

- La instrucción **for** ejecuta una secuencia de instrucciones, un número determinado de veces.

```
for (inicialización; condición; incremento){  
    instrucciones;  
}
```

```
for (inicialización; condición; incremento):  
    instrucciones;  
endfor;
```

...que te limita

Ejemplo

```
<?php
for($i = 0; $i <=10;$i++) {
    echo $i . '<br/>';
}

$colores = array('rojo', 'verde', 'azul', 'amarillo','morado');

for($i = 0; $i<count($colores); $i++ ) {
    echo $colores[$i] . '<br/>';
}
?>
```

0
1
2
3
4
5
6
7
8
9
10

rojo
verde
azul
amarillo
morado

Estructura repetitiva foreach

- La instrucción **foreach** ejecuta una secuencia de instrucciones, un número determinado de veces.
- Sintaxis:

```
foreach (arreglo as valor){  
    instrucciones;  
}
```

```
foreach (arreglo as valor):  
    instrucciones;  
endforeach;
```


Ejemplo: acceso a un arreglo

- El siguiente ejemplo muestra los elementos de un arreglo usando el control **foreach**.

```
<?php
$valores=array(2, 8, 15, 1, 10);
foreach ($valores as $num) {
    echo $num . "<br>";
}
echo"<br>";

foreach ($valores as $indice=>$valor){
    echo $indice.">".$valor."<br>";
}
echo"<br>";
```

```
2
8
15
1
10

0->2
1->8
2->15
3->1
4->10
```

Ejemplo: Acceso a un arreglo asociativo

- Cuando se hace referencia a los elementos de un arreglo por medio de un nombre.

```
$persona = array(  
    'nombre' => 'Carla',  
    'pais' => 'Peru',  
    'edad' => '23'  
);  
  
foreach($persona as $key => $valor ):  
    echo $key . ' : ' . $valor . "<br>";  
endforeach;  
?>
```

nombre: Carla
pais: Peru
edad: 23

Ejemplo: Acceso a un arreglo multidimensional

- El siguiente ejemplo muestra la impresión de un arreglo multidimensional usando el control **foreach**.

Arreglo Multidimensional

Carla Peru 23

Idiomas:

ingles
frances
postugues
italiano

Deportes:

futbol
natacion
basquet

```
<?php
$persona = array(
    'datos' => array(
        'nombre' => 'Carla',
        'pais' => 'Peru',
        'edad' => '23'
    ),
    'hobbies' => array(
        'idiomas' => array('ingles', 'frances', 'postugues', 'italiano'),
        'deportes' => array('futbol', 'natacion', 'basquet')
    )
);
?>

<h3>Arreglo Multidimensional</h3>
<?php
foreach($persona['datos'] as $per):
    echo $per. " ";
endforeach;
echo "<br>";
foreach($persona as $hob):
    if(array_key_exists('idiomas', $hob)){
        echo "<h3>Idiomas:</h3>";
        foreach($hob['idiomas'] as $idiom):
            echo $idiom."<br>";
        endforeach;
    }

    if(array_key_exists('deportes', $hob)){
        echo "<h3>Deportes:</h3>";
        foreach($hob['deportes'] as $dep):
            echo $dep."<br>";
        endforeach;
    }
endforeach;
?>
```

Ejemplo: Acceso a un arreglo multidimensional

- Se refiere a los arreglos de 2 o mas dimensiones.

```
<?php
    $persona=array(
        array("Carmen", "Perez", 22),
        array("Luis", "Romero", 37),
        array("Sandra", "Vargas", 20),
    );

    for($i=0; $i<count($persona); $i++){
        for($j=0; $j<count($persona[$i]); $j++){
            echo $persona[$i][$j]. " ";
        }
        echo "<br>";
    }
?>
```

Carmen Perez 22 Luis Romero 37 Sandra Vargas 20

Aplicaciones

*Implementar en el laboratorio
lo explicado en clase*

Ejemplo 1

Ingresa el nombre, apellidos, correo, genero, estado civil y un comentario, luego mostrar los datos ingresados.

Registro de datos

Nombre:

Apellidos:

Correo:

Genero: ☐ Varon ☐ Mujer

Estado civil:

Comentario:

```
<?php
echo "Nombre: "; echo $_POST['nombre']; echo "<br/>";
echo "apellidos: "; echo $_POST['apellidos']; echo "<br/>";
echo "E-mail: "; echo $_POST['correo']; echo "<br/>";
echo "Genero: "; echo $_POST['sexo']; echo "<br/>";
echo "Estado civil: "; echo $_POST['estado']; echo "<br/>";
echo "Comentario: "; echo $_POST['comentario']; echo "<br/>";
?>
```

```
<body>
<h1>Registro de datos</h1>
<form action="datos.php" method="post">
  <table>
    <tr>
      <td>Nombre:</td>
      <td><input type="text" name="nombre"></td>
    </tr>
    <tr>
      <td>Apellidos:</td>
      <td><input type="text" name="apellidos"></td>
    </tr>
    <tr>
      <td>Correo:</td>
      <td><input type="email" name="correo"></td>
    </tr>
    <tr>
      <td>Genero:</td>
      <td>
        <input type="radio" name="sexo" value="V"/> Varon<br/>
        <input type="radio" name="sexo" value="M"/> Mujer
      </td>
    </tr>
    <tr>
      <td>Estado civil:</td>
      <td><select name="estado">
        <option>Estado civil</option>
        <option value="">soltero</option>
        <option value="Casado">casado</option>
        <option value="Viudo">viudo</option>
        <option value="Divorciado">divorciado</option>
      </select></td>
    </tr>
  </table>
  <p>Comentario:</p>
  <textarea name="comentario" rows="5" cols="50"></textarea>
  <p><input type="submit" value="Comprobar el formulario">
    <input type="reset" value="borrar todo"></p>
</form>
</body>
```

Ejemplo 2

- Sumar 2 números y visualizar el resultado de la suma en la misma pagina.

```
<body>
  <?php
    $suma="";
    if(isset($_POST['submit'])){
      $n1=$_POST['n1'];
      $n2=$_POST['n2'];
      $suma=$n1+$n2;
    }
  ?>
  <form method="post">
    <h1>Sumas</h1>
    <label for=""> Numero 1: </label>
    <input type="text" name="n1" value="<?php if(isset($n1)) echo $n1; ?>"><br>
    <label for="">Numero 2: </label>
    <input type="text" name="n2" value="<?php if(isset($n2)) echo $n2; ?>"><br>
    <label for="">Suma: </label>
    <input name="suma" value="<?=$suma?>"><br>
    <input type="submit" value="Aceptar" name="submit">
  </form>
</body>
```

Ejemplo 3

- Calcular el pago final que se realiza por la compra se cierta cantidad de un producto sabiendo que si se compra mas de 12 unidades se le aplica un descuento del 15%.

```
<body>
  <form method="post" action="Declase.php" target="resultado">
    <table >
      <tr>
        <td>Nombre de producto</td>
        <td><input type="text" name="producto"></td>
      </tr>
      <tr>
        <td>Cantidad</td>
        <td><input type="text" name="cantidad"></td>
      </tr>
      <tr>
        <td>Precio</td>
        <td><input type="text" name="precio"></td>
      </tr>
    </table>
    <input type="submit" value="Calcular">
  </form>
  <br>
  <iframe name="resultado" width="400" height="200"></iframe>
</body>
```


Ejemplo 3

```
<?php
    $prod=$_POST["producto"];
    $cant=$_POST["cantidad"];
    $prec=$_POST["precio"];
    $desc=0;
    if($cant>12)
        $desc=0.15*$prec;
    $stot=$cant*$prec;
    $tot=$stot-($desc*$cant);
?>
<h3>
    Producto: <?=$prod?>
    <br>Cantidad :<?=$cant?>
    <br>Precio :<?=$prec?>
    <br>Descuento :<?=$desc?>
    <br>Subtotal :<?=$stot?>
    <br>Total :<?=$tot?>
</h3>
```

Nombre de producto	<input type="text" value="Cuaderno"/>
Cantidad	<input type="text" value="3"/>
Precio	<input type="text" value="7.5"/>
<input type="button" value="Calcular"/>	

Producto: Cuaderno
Cantidad :3
Precio :7.5
Descuento :0
Subtotal :22.5
Total :22.5

Ejemplo 4

- Guardar en un arreglo el nombre del curso y sus tres notas, calcular el promedio y mostrar su condición de aprobado o desaprobado junto con una imagen que represente tal resultado.

```
<body>
  <h1>Calculo de notas</h1>
  <form method="post" action="notas.php">
    <table>
      <tr>
        <td>Curso:</td>
        <td>
          <select name="curso">
            <option>Principios de Algoritmos
            <option>Taller de programacion
            <option>Herramientas de informacion
          </select>
        </td>
      <tr>
        <td>Ingrese Nota 1:</td>
        <td><input type="text" name="nota1"></td>
      </tr>
      <tr>
        <td>Ingrese Nota 2:</td>
        <td><input type="text" name="nota2"></td>
      <tr>
        <td>Ingrese Nota 3:</td>
        <td><input type="text" name="nota3"></td>
      </tr>
    </table>
    <tr><td><input type="submit" value="Promediar">
  </form>
</body>
```

Ejemplo 4

```
<?php
$dato[0]=$_REQUEST["curso"];
$dato[1]=$_REQUEST["nota1"];
$dato[2]=$_REQUEST["nota2"];
$dato[3]=$_REQUEST["nota3"];
$dato[4]=($dato[1]+$dato[2]+$dato[3])/3;
    if($dato[4]>=12){
        $dato[5]="Aprobado";
        $imagen="imagenes/feliz.jpg";
    }else{
        $dato[5]="Desaprobado";
        $imagen="imagenes/triste.jpg";
    }

?>
Curso: <?=$dato[0]?><br>
Promedio:<?= number_format($dato[4],1)?><br>
Condicion:<?=$dato[5]?><br>
<br>
<a href="ejem2.html" >Regresar</a>
```

Calculo de notas

Curso:	<input type="text" value="Taller de programacion"/>
Ingrese Nota 1:	<input type="text" value="14"/>
Ingrese Nota 2:	<input type="text" value="11"/>
Ingrese Nota 3:	<input type="text" value="18"/>
<input type="button" value="Promediar"/>	

Curso: Taller de programacion
Promedio:14.3
Condicion:Aprobado



[Regresar](#)

Ejercicio 1

- A un obrero se le paga cada día por sus horas trabajadas. Si la cantidad de horas trabajadas es mayor o igual a 8, la tarifa será de S/. 25 por hora, si las horas trabajadas es menor a 8 será de S/. 17 por hora, además a los trabajadores que laboraron menos de 8 horas se le sanciona con un descuento del 12% sobre su pago. Calcular el salario total pagado al obrero diariamente.

Ejercicio 2

- Un banco realiza el pago de intereses a sus clientes por un deposito a plaza fijo de acuerdo a la siguiente información: Tipo de moneda, Tiempo de depósito y monto depositado. Los intereses serán aplicados según el siguiente cuadro:

Meses	Soles (%)	Dólares (%)
0 – 5	3	2
6 – 12	6	4
13 – mas	9	7

- Mostrar el interés y el monto total a recibir.

Ejercicio 3

- Generar 15 números aleatorios entre 20 y 200 que corresponde a la velocidad con la maneja un conductor. Con estos datos calcular el monto de la multa que paga cada conductor por exceso de velocidad. El conductor será multado solo si la velocidad del auto es mayor a 60 Km/h. Las multas a aplicarse son:

Velocidad (Km/h)	Multa (S/.)
60 – 100	250
101 – 130	380
131 – más	590

Resumiendo...

- ¿Entre que signos se escribe el código PHP?
_____.
- ¿Cómo se definen las variables en PHP?
_____.
- ¿Para qué se usa la función isset()?
_____.
- ¿Cuál es la diferencia entre el método Post y el método Get?:
_____.



**Universidad
Tecnológica
del Perú**