DESARROLLO WEB EN ENTORNO SERVIDOR

UT 2: INTRODUCCIÓN A PHP

UT 2: INTRODUCCIÓN A PHP

1. **¿Qué es PHP?**
2. ¿Cómo funciona PHP?
3. Sintaxis básica
4. Tipos de datos
5. Variables
6. Constantes
7. Salida y Depuración de Datos
8. Operadores
9. Combinar PHP y HTML
10. Estructuras de control
11. Bucles
12. Arrays y bucles
13. Codificación
14. Inclusión

1. ¿QUÉ ES PHP?

PHP es un lenguaje de programación de uso general, especialmente adecuado para el **desarrollo web**.

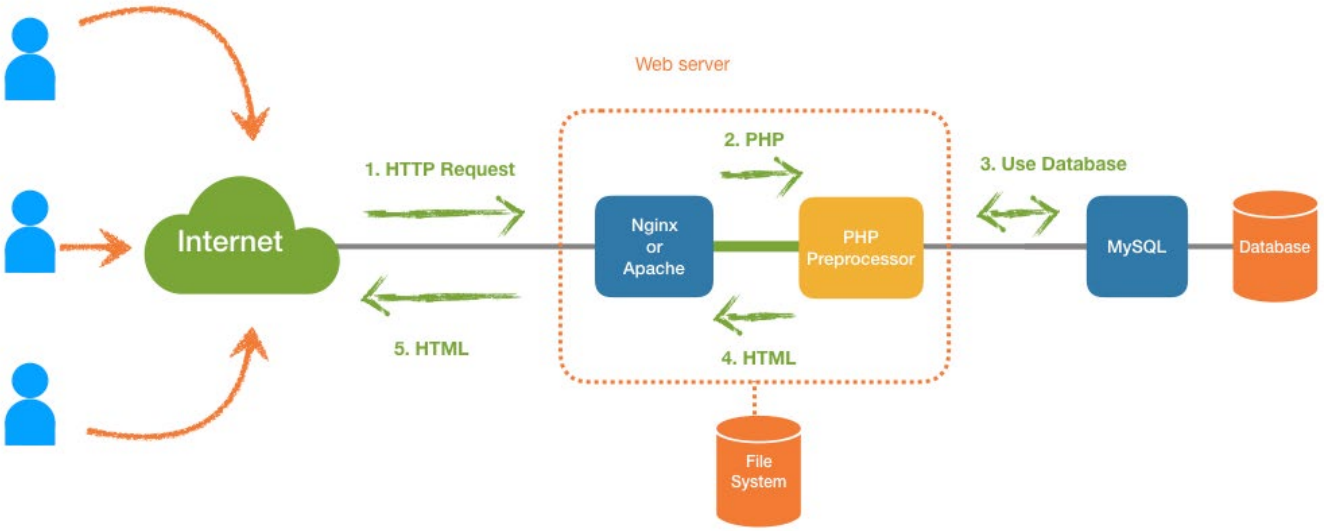
El código PHP puede ser interpretado y ejecutado desde la interfaz de línea de comandos (CLI) o desde un servidor web que tenga implementado un intérprete PHP.

Con PHP se puede utilizar para programación orientada a objetos (**POO**).

UT 2: INTRODUCCIÓN A PHP

1. ¿Qué es PHP?
2. **¿Cómo funciona PHP?**
3. Sintaxis básica
4. Tipos de datos
5. Variables
6. Constantes
7. Salida y Depuración de Datos
8. Operadores
9. Combinar PHP y HTML
10. Estructuras de control
11. Bucles
12. Arrays y bucles
13. Codificación
14. Inclusión

2. ¿CÓMO FUNCIONA PHP?



UT 2: INTRODUCCIÓN A PHP

1. ¿Qué es PHP?
2. ¿Cómo funciona PHP?
3. **Sintaxis básica**
4. Tipos de datos
5. Variables
6. Constantes
7. Salida y Depuración de Datos
8. Operadores
9. Combinar PHP y HTML
10. Estructuras de control
11. Bucles
12. Arrays y bucles
13. Codificación
14. Inclusión

3. SINTAXIS BÁSICA

ETIQUETAS DE APERTURA Y CIERRE DE PHP

El código PHP se encierra entre las etiquetas de apertura y cierre: **<?php** y **?>**. Todo el código que se encuentre estas dos etiquetas será interpretado como código PHP.

Ejemplo**:**

```
<?php
echo "¡Hola mundo!";
?>
```

3. SINTAXIS BÁSICA

CARÁCTER SEPARADOR DE INSTRUCCIONES

Todas las instrucciones en PHP terminan con el carácter punto y coma (**;**).

Ejemplo**:**

```
<?php
    echo "¡Hola ";
    echo "mundo!";
?>
```

3. SINTAXIS BÁSICA

CARÁCTER SEPARADOR DE INSTRUCCIONES

El único caso donde se puede omitir el carácter punto y coma (;) en la última instrucción de un bloque PHP, ya que la etiqueta de cierre de un bloque PHP (?>) implica un punto y coma.

Ejemplo:

```
<?php
    echo "¡Hola ";
    echo "mundo!"
?>
```

3. SINTAXIS BÁSICA

COMENTARIOS

Podemos escribir comentarios de una sóla línea con: // y #, y comentarios multilínea con /* ... */.

Ejemplo:

```
<?php
    // Esto es un comentario de una línea
    echo "Frase 1";
    # Esto es otro comentario de una línea
    echo "Frase 2";
    /* Este comentario
    es un comentario de múltiples líneas */
    echo "Frase 3";
?>
```

3. SINTAXIS BÁSICA

CÓDIGO PHP EMBEBIDO EN DOCUMENTOS HTML

El uso de estas etiquetas de apertura y cierre, nos permite embeber código PHP en documentos HTML. De modo que sólo el código que aparezca entre las etiquetas <?php y ?> será interpretado por el intérprete de PHP y el resto de etiquetas serán ignoradas.

3. SINTAXIS BÁSICA

CÓDIGO PHP EMBEBIDO EN DOCUMENTOS HTML

Ejemplo de código PHP embebido en un documento HTML:

```
<!DOCTYPE html>
<html>
<head>
    <title>Ejemplo</title>
</head>
<body>
    <p>Esto es contenido estático en HTML</p>
    <p>Hoy es <?php echo date('Y/m/d'); ?></p>
</body>
</html>
```

UT 2: INTRODUCCIÓN A PHP

1. ¿Qué es PHP?
2. ¿Cómo funciona PHP?
3. Sintaxis básica
4. **Tipos de datos**
5. Variables
6. Constantes
7. Salida y Depuración de Datos
8. Operadores
9. Combinar PHP y HTML
10. Estructuras de control
11. Bucles
12. Arrays y bucles
13. Codificación
14. Inclusión

4. TIPOS DE DATOS

Datos escalares: boolean, integer float y string.

Existen diferentes tipos de datos, pero **no se declaran**.

```
$nombre = "Juan";
$edad = 30;
```

Sin embargo, es importante mencionar que PHP es un lenguaje de **tipado débil**, lo que significa que las variables pueden cambiar de tipo de dato durante la ejecución del programa.

```
$variable = "Hola";
$variable = 42;
```

https://www.php.net/manual/es/language.types.php

4. TIPOS DE DATOS

FUNCIONES

La función **gettype**() devuelve el tipo de una variable

Las funciones **is_ type** comprueban si una variable es de un tipo dado: is_array(), is_bool(), is_float(), is_integer(), is_null(), is_numeric(), is_object(), is_resource(), is_scalar(), is_string()

4. TIPOS DE DATOS

- Tipo **integer** (números enteros): 27, - 5, 0

- Tipo **double** (números reales): 1.234, -5.33

- Tipo **boolean** (lógico)
  - Valores: true , false (no distingue entre mayúsculas y minúsculas)
  - El 0 y la cadena vacía tienen valor false.
  - El 1, true.

4. TIPOS DE DATOS

STRING

Las cadenas se encierran entre comillas simples o dobles:

- **'simples'**: admite los caracteres de escape \' (comilla simple) y \\(barra). Las variables NO se expanden, es decir, las variables dentro de comillas simples no se sustituyen por su valor real.
- **“dobles”**: Los nombres de variables SÍ se expanden.

```
$a = 9;
print 'a vale $a'; // muestra a vale $a
print “a vale $a”; // muestra a vale 9
```

Acceso a un carácter de la cadena: `$inicial = $nombre{0};`

4. TIPOS DE DATOS

STRING

Con las comillas dobles, se admite más caracteres de escape, como \n, \r, \t, \\, \$, \" para usar en archivos de texto, pero no cuando la salida es en una página web.

Acceso a un carácter de la cadena: `$inicial = $nombre{0};`

4. TIPOS DE DATOS

STRING

- Formas de asignar cadenas
  - Clásica $a="valor de cadena"
  - Documento Incustrado

```
$a=<<< PALABRA
        texto
        puede ir en varias
        líneas,
        pero el inicio y fin en
        otra línea diferente
        PALABRA;
```

- Concatenación de cadenas: operador.

UT 2: INTRODUCCIÓN A PHP

1. ¿Qué es PHP?
2. ¿Cómo funciona PHP?
3. Sintaxis básica
4. Tipos de datos
5. **Variables**
6. Constantes
7. Salida y Depuración de Datos
8. Operadores
9. Combinar PHP y HTML
10. Estructuras de control
11. Bucles
12. Arrays y bucles
13. Codificación
14. Inclusión

5. VARIABLES

- El nombre de la variable es **sensible** a mayúsculas y minúsculas.
- Los nombres de variables comienzan con **$**.
- Las variables **no se declaran**, no hay que indicar el tipo. Se asigna su valor en el momento de utilizarla.
- Comienzan por **letra o subrayado**, seguido de letras, números o subrayado
- Para hacer referencia al valor de una variable, necesario utilizar su nombre incluido el $.
- El valor puede ser un literal (texto o número), otra variable o una expresión.

5. VARIABLES

Ámbito: globales al fichero (excepto funciones) o locales en una función.

Para poder acceder desde una función a variables externas es necesario definirlas con la palabra reservada global.

```php
<?php
$variable_global = "¡Hola, mundo!";

function mostrarMensaje() {
    global $variable_global;
    echo $variable_global;
}

mostrarMensaje();  // Imprime: ¡Hola, mundo!
?>
```

5. VARIABLES

Ejemplo:

```php
<?php
    $nombre = "Pepe";
    $edad = 30;
      echo "El nombre es: " . $nombre . "\n";
      echo "El nombre es: $nombre\n";
 ?>
```

5. VARIABLES

VARIABLES PREDEFINIDAS

Variables que contienen información para el programador. Realmente son arrays asociativos.

- $GLOBALS: contiene todas las variables globales disponibles en el ámbito global del script PHP.

```
$mensaje = "Hola, mundo!"; // Variable global

function mostrarMensajeConGlobals() {
    echo $GLOBALS['mensaje'];
}
```

5. VARIABLES

VARIABLES PREDEFINIDAS

- **$_ SERVER**: proporciona información sobre el servidor web y el entorno en el que se ejecuta el script PHP
- **$_GET**: se utiliza para acceder a datos pasados a través de una solicitud HTTP GET.
- **$_POST**: se utiliza para acceder a datos pasados a través de una solicitud HTTP POST.
- **$_COOKIES**: se utiliza para acceder a las cookies que se han enviado al navegador del cliente y que posteriormente son enviadas de vuelta al servidor con cada solicitud HTTP.

5. VARIABLES

VARIABLES PREDEFINIDAS

- **$_FILES**: se utiliza para manejar archivos que se han enviado al servidor a través de un formulario HTML.
- **$_ ENV**: se utiliza para acceder a las variables de entorno del sistema operativo en el que se ejecuta el script PHP.
- **$_ REQUEST**: se utiliza para recopilar datos enviados a través de una solicitud HTTP. A diferencia de $_GET y $_POST, que se utilizan para acceder a datos específicamente enviados a través de una solicitud GET o POST, $_REQUEST puede utilizarse para acceder a datos de ambos métodos de solicitud, así como de cookies
- **$_SESSION**: se utiliza para gestionar variables de sesión en una aplicación web. Para utilizar $_SESSION, es necesario iniciar la sesión en PHP utilizando session_start() al principio de tu script.

5. VARIABLES

VARIABLES ESTATICAS

Posibilidad de definirdentro de las funciones variables estáticas No

pierden su valor cuando se finaliza la función

```
static $var= valor;
```

5. VARIABLES

VARIABLES DE VARIABLES

Posibilidad de definir variables con el nombre del contenido de otras variables

```
$color=“azul” ;//Variable con nombre color
$$color=“es bonito”;//Variable con nombre azul

$mensaje_es="Hola";
$mensaje_en="Hello";
$idioma = "es"
$mensaje = "mensaje_" . $idioma;
print $$mensaje;
```

UT 2: INTRODUCCIÓN A PHP

1. ¿Qué es PHP?
2. ¿Cómo funciona PHP?
3. Sintaxis básica
4. Tipos de datos
5. Variables
6. **Constantes**
7. Salida y Depuración de Datos
8. Operadores
9. Combinar PHP y HTML
10. Estructuras de control
11. Bucles
12. Arrays y bucles
13. Codificación
14. Inclusión

6. CONSTANTES

Las constantes no pueden modificar su valor durante la ejecución del script.

- El nombre de la constante **no tiene** que ir precedido por el símbolo del dólar
- Es sensible a mayúsculas y minúsculas.
- Por convención, las constantes siempre se declaran en **mayúsculas**.
- Si el valor es numérico no es necesario utilizar las comillas dobles.
- Son accesibles desde todo el documento, incluidas las funciones.
- Para acceder al valor de una constante, hay que utilizar su nombre
- Sólo se pueden definir constantes de los tipos escalares (booleano, entero, real, string)

6. CONSTANTES

Ejemplo:

```php
<?php
    // Ejemplo de una constante numérica de tipo real
    define("PI", 3.141592);

    // Ejemplo de una constante de tipo string
    define("CONSTANTE", "Hola mundo");
?>
```

6. CONSTANTES

CONSTANTES PREDEFINIDAS

PHP define una serie de constantes: _FILE_,_LINE_ , PHP_OS, PHP_VERSION

UT 2: INTRODUCCIÓN A PHP

1. ¿Qué es PHP?
2. ¿Cómo funciona PHP?
3. Sintaxis básica
4. Tipos de datos
5. Variables
6. Constantes
7. **Salida y Depuración de Datos**
8. Operadores
9. Combinar PHP y HTML
10. Estructuras de control
11. Bucles
12. Arrays y bucles
13. Codificación
14. Inclusión

7. SALIDA Y DEPURACIÓN DE DATOS

ECHO

Es una construcción del lenguaje (no es una función) que nos permite mostrar cadenas de texto y el contenido de las variables.

Ejemplo:

```php
<?php
echo "Esto es una cadena de texto.";
// Con echo también podemos mostrar el contenido de una variable
$numero = 10;
echo "El contenido de la variable es: $numero";
?>
```

7. SALIDA Y DEPURACIÓN DE DATOS

VAR_DUMP

Es una función nos permite mostrar el contenido de una variable. Esta función muestra el tipo de dato y el valor de la variable.

Ejemplo:

```php
<?php
$nombre = "Pepe";
$edad = 30;
$nota = 7.5;
var_dump($nombre);
// string(4) "Pepe"
var_dump($edad);
// int(30)
var_dump($nota);
// float(7.5)
?>
```

7. SALIDA Y DEPURACIÓN DE DATOS

PRINT_R

Es una función que nos permite mostrar el contenido de una variable de una forma legible.

Ejemplo:

```php
<?php
$lista = array("Pepe", "María", "Juan");
print_r($lista);
//Array
//(
//    [0] => Pepe
//    [1] => María
//    [2] => Juan
//)
?>
```

UT 2: INTRODUCCIÓN A PHP

1. ¿Qué es PHP?
2. ¿Cómo funciona PHP?
3. Sintaxis básica
4. Tipos de datos
5. Variables
6. Constantes
7. Salida y Depuración de Datos
8. **Operadores**
9. Combinar PHP y HTML
10. Estructuras de control
11. Bucles
12. Arrays y bucles
13. Codificación
14. Inclusión

8. OPERADORES

OPERADORES ARITMÉTICOS

Arithmetic Operators		
Example	**Name**	**Result**
-$a	Negation	Opposite of *$a*.
$a + $b	Addition	Sum of *$a* and *$b*.
$a - $b	Subtraction	Difference of *$a* and *$b*.
$a * $b	Multiplication	Product of *$a* and *$b*.
$a / $b	Division	Quotient of *$a* and *$b*.
$a % $b	Modulus	Remainder of *$a* divided by *$b*.
$a ** $b	Exponentiation	Result of raising *$a* to the *$b*'th power. Introduced in PHP 5.6.

8. OPERADORES

OPERADORES ARITMÉTICOS

- Funciones: sqrt, pow, ceil, round, floor, etc
- Si se pretende ejecutar una instrucción aritmética dentro de echo o print es necesario incluirla entre paréntesis
- Existen funciones matemáticas
  - Logarítmicas log, Log 10, Exp
  - Trigonométricas Cos Sin, Tan, Asin, Atan, Acos

8. OPERADORES

OPERADORES DE INCREMENTO/DECREMENTO

Increment/decrement Operators		
Example	**Name**	**Effect**
++$a	Pre-increment	Increments *$a* by one, then returns *$a*.
$a++	Post-increment	Returns *$a*, then increments *$a* by one.
--$a	Pre-decrement	Decrements *$a* by one, then returns *$a*.
$a--	Post-decrement	Returns *$a*, then decrements *$a* by one.

8. OPERADORES

OPERADORES RELACIONALES

Devuelven True o False

Operand	Example	Meaning
==	$variable1 == $variable2	Has the same value as
!=	$variable1 != $variable2	Is NOT the same value as
<	$variable1 < $variable2	Less Than
>	$variable1 > $variable2	Greater Than
<=	$variable1 <= $variable2	Less than or equals to
>=	$variable1 >= $variable2	Greater than or equals to

8. OPERADORES

OPERADORES LÓGICOS

Logical Operators		
Example	**Name**	**Result**
$a and $b	And	**TRUE** if both *$a* and *$b* are **TRUE**.
$a or $b	Or	**TRUE** if either *$a* or *$b* is **TRUE**.
$a xor $b	Xor	**TRUE** if either *$a* or *$b* is **TRUE**, but not both.
! $a	Not	**TRUE** if *$a* is not **TRUE**.
$a && $b	And	**TRUE** if both *$a* and *$b* are **TRUE**.
$a || $b	Or	**TRUE** if either *$a* or *$b* is **TRUE**.

UT 2: INTRODUCCIÓN A PHP

1. ¿Qué es PHP?
2. ¿Cómo funciona PHP?
3. Sintaxis básica
4. Tipos de datos
5. Variables
6. Constantes
7. Salida y Depuración de Datos
8. Operadores
9. **Combinar PHP y HTML**
10. Estructuras de control
11. Bucles
12. Arrays y bucles
13. Codificación
14. Inclusión

9. COMBINAR PHP Y HTML

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>HolayAdios</title>
</head>
<body>
    <?php
        $a = 1
        if ($a == 1) { ?>
            <p>Hola></p>
        <?php
        } else { ?>
            <p>Adios></p>
        <?php
        }
    ?>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>HolayAdios</title>
</head>
<body>
    <?php
        $a = 1
        if ($a == 1) {
            echo'<p>Hola></p>';
        } else {
            echo'<p>Adios></p>';
        }
    ?>
</body>
</html>
```

9. COMBINAR PHP Y HTML

Una mala práctica: colocar código HTML

embebido dentro de variables de PHP

```
<?php
$alicuota_iva = 21;
$codigo_de_producto = 1284;
$nombre_producto = "Agua Mineral Manantial x 500 ml";
$precio_bruto = 3.75;
$iva = 3.75 * 21 / 100;
$precio_neto = $precio_bruto + $iva;

$producto = "<p><b>Producto:</b> ($codigo_de_producto) $nombre_producto<br/>
<b>Precio:</b> USD $precio_neto.- (IVA incluido)</p>";
?>
<!doctype html>
<html>
<head>
    <title>Detalles del producto <?php echo $nombre_producto; ?></title>
</head>

<body>
    <?php echo $producto; ?>
</body>
</html>
```

Una buena práctica para evitar lo anterior

```
<?php
$alicuota_iva = 21;
$codigo_de_producto = 1284;
$nombre_producto = "Agua Mineral Manantial x 500 ml";
$precio_bruto = 3.75;
$iva = 3.75 * 21 / 100;
$precio_neto = $precio_bruto + $iva;
?>
<!doctype html>
<html>
<head>
    <title>Detalles del producto <?php echo $nombre_producto; ?></title>
</head>

<body>
    <p><b>Producto:</b> (<?php echo $codigo_de_producto; ?>)
    <?php echo $nombre_producto; ?><br/>
    <b>Precio:</b> USD <?php echo $precio_neto; ?>.- (IVA incluido)</p>
</body>
</html>
```

UT 2: INTRODUCCIÓN A PHP

1. ¿Qué es PHP?
2. ¿Cómo funciona PHP?
3. Sintaxis básica
4. Tipos de datos
5. Variables
6. Constantes
7. Salida y Depuración de Datos
8. Operadores
9. Combinar PHP y HTML
10. **Estructuras de control**
11. Bucles
12. Arrays y bucles
13. Codificación
14. Inclusión

10. ESTRUCTURAS DE CONTROL

1. **if**
2. if-else
3. elseif / else if
4. switch

10.1. IF

La instrucción se ejecuta siempre que la condición sea verdadera.

```
<?php
if (condicion_es_cierta) {
    bloque_de_sentencias
}
?>
```

Ejemplo:

```
<?php
$a = 10;
$b = 5;
if ($a > $b) {
    echo "a es mayor que b";
}
?>
```

EJERCICIO1: Escribir un script que muestre si un número aleatorio entre 1 y 100 es par.
Usar la función rand de php.

10. ESTRUCTURAS DE CONTROL

1. if
2. **if-else**
3. elseif / else if
4. switch

10.2. IF-ELSE

Ejemplo:

```php
<?php

$a = 1;
$b = 5;

if ($a > $b) {
    echo "a es mayor que b";
} else {
    echo "a NO es mayor que b";
}
?>
```

10.2. IF-ELSE

Si la condición es verdadera se ejecutan las acciones del primer bloque y si es falsa se ejecutan las acciones del segundo bloque.

```php
<?php
if (condicion_es_cierta) {
    bloque_de_sentencias_1
} else {
    bloque_de_sentencias_2
}
?>
```

EJERCICIO2: Escribir un script para reproducir que hemos lanzado una moneda al aire y muestre una imagen con cara o cruz.

10. ESTRUCTURAS DE CONTROL

1. if
2. if-else
3. **elseif / else if**
4. switch

10.3. ELSE IF/ ELSEIF

Dos formas equivalentes de encadenar if.

Se pueden añadir tantos bloques elseif como se necesiten.

```php
<?php


if (condición_1) {
    bloque_de_sentencias_1
} elseif (condición_2) {
    bloque_de_sentencias_2
} else {
    bloque_de_sentencias_3
}
?>
```

```php
<?php

    if (condición_1) {
        bloque_de_sentencias_1
    } else if (condición_2) {
        bloque_de_sentencias_2
    } else {
        bloque_de_sentencias_3
    }
?>
```

10.3. ELSE IF/ ELSEIF

Ejemplo:

```php
<?php
$a = 1;
$b = 1;

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";
}
?>
```

EJERCICIO3: Escribir un script en PHP que produzca un número al azar entre 1 y 10, imitando una nota numérica y muestre si es supenso, aprobado, notable o sobresaliente.

Usar la función rand de php.

EJERCICIO4: Realizar un script que genere dos números aleatorios y muestre si son iguales y si no lo son, cuál es el mayor de los dos.

10. ESTRUCTURAS DE CONTROL

1. if
2. if-else
3. elseif / else if
4. **switch**

10.4. SWITCH

Con frecuencia se presentan más de dos elecciones posibles de una cierta condición. Se evaluará una expresión que podrá tomar n valores distintos y se realizará una de las n acciones.

```php
<?php

    switch ($variable) {
        case valor1:
            bloque_de_sentencias_1
            break;

        case valor2:
            bloque_de_sentencias_2
            break;

        default:
            bloque_de_sentencias_3
    }

?>
```

10.4. SWITCH

Esta estructura de control es equivalente a:

```php
<?php

if ($variable == valor1) {
    bloque_de_sentencias_1
} elseif ($variable == valor2) {
    bloque_de_sentencias_2
} else {
    bloque_de_sentencias_3
}
?>
```

10.4. SWITCH

Ejemplo:

```php
<?php
$numero = 2;

switch ($numero) {
    case 1:
        echo "La variable es igual a 1";
        break;

    case 2:
        echo "La variable es igual a 2";
        break;

    default:
        echo "La variable es un número distinto a 1 y 2";
}

?>
```

EJERCICIO5: Escribir un script en PHP que produzca un número al azar entre 1 y 10, imitando una nota numérica y muestre si es supenso, aprobado, notable o sobresaliente.

Usar la estructura switch.

Usar la función rand de php.

EJERCICIO6: Escribir un script PHP que genere un número aleatorio entre 1 y 7, y mostrar un mensaje que indique que día de la semana es.

Por ejemplo, 1 es lunes, 2 martes, etc.

UT 3: INTRODUCCIÓN A PHP

1. ¿Qué es PHP?
2. ¿Cómo funciona PHP?
3. Sintaxis básica
4. Tipos de datos
5. Variables
6. Constantes
7. Salida y Depuración de Datos
8. Operadores
9. Combinar PHP y HTML
10. Estructuras de control
11. **Bucles**
12. Arrays y bucles
13. Codificación
14. Inclusión

11. BUCLES

1. **while**
2. do-while
3. for

11.1. WHILE

```
while (condicion_es_verdadera) {
    sentencias;
}
```

11.1. WHILE

Ejemplo: Mostrar los números del 1 al 10.

```php
<?php

$i = 1;
while ($i <= 10) {
    echo $i;
    echo "<br>";
    $i++;
}

?>
```

EJERCICIO7: Escribir un script PHP que muestre los números del 1 al 10 en una tabla de una fila y 10 columnas.

Utilizar un bucle while.

11. BUCLES

1. while
2. **do-while**
3. for

11.2. DO-WHILE

```
do {
    sentencias;
} while (condicion_es_verdadera)
```

11.2. DO-WHILE

Ejemplo: Mostrar los números del 1 al 10.

```php
<?php

$i = 1;
do {
    echo $i;
    echo "<br>";
    $i++;
} while ($i <= 10);

?>
```

11. BUCLES

1. while
2. do-while
3. **for**

11.3. FOR

Ejecuta un bloque de instrucciones un número determinado de veces. Para ello se utiliza un contador.

Utiliza una variable que va aumentado o decrementando en cada iteración.

```
for (expr1; expr2; expr3) {
    sentencias;
}
```

En expr1 se inicializa la variable denominada contador.

En expr2 se comprueba si tenemos que seguir en el bucle.

En expr3 se itera sobre la variable contador.

11.3. FOR

Esta estructura es muy útil cuando sabemos el número de veces exactas que queremos ejecutar un bloque de instrucciones.

Ejemplo: Mostrar los números del 1 al 10.

```php
<?php

for ($i = 1; $i <= 10; $i++) {
    echo $i;
    echo "<br>";
}

?>
```

11.3. FOR

Ejemplo: Mostrar los números del 10 al 1.

```php
<?php

for ($i = 10; $i >= 1; $i--) {
    echo $i;
    echo "<br>";
}

?>
```

EJERCICIO8: Realizar un script PHP que muestre los números del 1 al 10 en una tabla de una fila y 10 columnas.
Utilizar un bucle for.

EJERCICIO9: Realizar un script PHP que muestre la tabla de multiplicar de un número aleatorio.

EJERCICIO10: Realizar un script PHP que muestre en una tabla los números pares que existen entre 1 y 100.

UT 3: INTRODUCCIÓN A PHP

1. ¿Qué es PHP?
2. ¿Cómo funciona PHP?
3. Sintaxis básica
4. Tipos de datos
5. Variables
6. Constantes
7. Salida y Depuración de Datos
8. Operadores
9. Combinar PHP y HTML
10. Estructuras de control
11. Bucles
12. **Arrays y bucles**
13. Codificación
14. Inclusión
15. Interacción
16. Configuración PHP

12. ARRAYS Y BUCLES

Cuando, por ejemplo, tenemos que almacenar 40 notas de alumnos, no tiene sentido crear 40 variables.

Es más útil usar una sola variable, pero que almacene conjuntos de datos.

Un array es una estructura de datos que nos permite almacenar varios valores en una única variable.

12. ARRAYS Y BUCLES

1. **Array con índices**
2. Arrays asociativos

12.1. ARRAYS CON ÍNDICES

Para crear un array es suficiente con hacer:

```php
$productos = array();
```

Existen dos formas de inicializar los valores de un array indexado:

- Cuando se crea

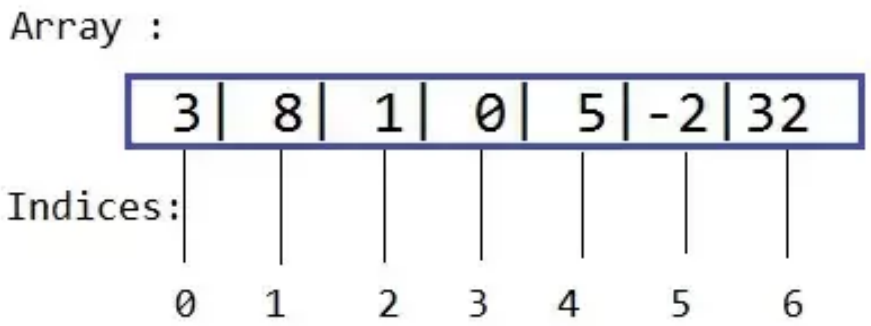
```php
$productos = array("Disco SSD", "Memoria RAM", "Monitor");
```

- asignando los valores posición a posición:

```php
$productos[0] = "Disco SSD";
$productos[1] = "Memoria RAM";
$productos[2] = "Monitor";
```

12.1. ARRAYS CON ÍNDICES

Recordad que los arrays siempre empiezan por la **posición 0**.



12.1. ARRAYS CON ÍNDICES

Para consultar su contenido:

```php
print_r($productos);
```

Para conocer el tamaño de un array con **count:**

```php
echo count($productos);
```

12.1. ARRAYS CON ÍNDICES

Para recorrer un array con for:

```php
$productos = array("Disco SSD", "Memoria RAM", "Monitor");
$numero_de_elementos = count($productos);
for ($i = 0; $i < $numero_de_elementos; $i++ ) {
    echo $productos[$i];
    echo "<br>";
}
```

12. ARRAYS Y BUCLES

1. Array con índices
2. **Arrays asociativos**

12.1. ARRAYS ASOCIATIVOS

Los arrays asociativos nos permiten usar claves en lugar de índices, para acceder a los valores del array.

Existen dos formas de inicializar los valores de un array asociativo:

- Cuando se crea

```php
$edades = array("Juan" => "25", "María" => "28", "Paco" => "27");
```

- asignando los valores a cada clave:

```php
$edades["Juan"] = "35";
$edades["María"] = "35";
$edades["Paco"] = "35";
```

12.1. ARRAYS ASOCIATIVOS

Cómo recorrer un array asociativo con foreach

```php
$edades = array("Juan" => "25", "María" => "28", "Paco" => "27");
foreach ($edades as $clave => $valor) {
    echo "Clave: " . $clave . " - Valor: " . $valor;
    echo "<br>";
}
```

EJERCICIO11: Elabora un código en PHP que ejecute las siguientes tareas:

- Inicializar un array de 10 elementos, con valores aleatorios entre 1 y 30.
- Una vez que ha inicializado el array, imprimir todos los valores que almacena.

EJERCICIO12: Elabora un código en PHP que ejecute las siguientes tareas:

- Inicializar un array de 10 elementos, con valores aleatorios entre 1 y 30.
- Una vez que ha inicializado el array, imprima todos los valores que almacena.
- Calcular el valor medio de los valores del array.
- Mostrar el valor medio que ha calculado.

EJERCICIO13: Elabora un código en PHP que ejecute las siguientes tareas:
Inicializar un array de 10 elementos, con valores aleatorios entre 1 y 30.
Una vez que ha inicializado el array, imprima todos los valores que almacena.
Buscar el valor máximo y el valor mínimo de los valores del array.
Mostrar el valor máximo y el valor mínimo que ha encontrado.

Resolverlo con un for.

EJERCICIO14: Elabora un código en PHP que ejecute las siguientes tareas:

- Inicializar un array de 10 elementos, con valores aleatorios entre 1 y 30.
- Una vez que ha inicializado el array, imprima todos los valores que almacena.
- Mostrar el listado ordenado de menor a mayor. Función sort.
- Mostrar el listado ordenado de mayor a menor. Función rsort.

EJERCICIO15: Elabora un código en PHP que ordene el siguiente array asociativo y lo muestre en una lista:

```
$Frutas = array(
    "Manzana" => "roja",
    "Plátano" => "amarillo",
    "Naranja" => "naranja"
);
```

- De forma ascendente ordenado por valor.
- De forma ascendente ordenado por clave.
- De forma descendente ordenado por valor.
- De forma descendente ordenado por clave.

Usando asort, arsort, ksort, krsort.

12.2. ARRAYS ASOCIATIVOS

JSON

JSON (JavaScript Object Notation) es un formato de texto ligero para el intercambio de datos. Es fácil para los humanos leer y escribir, y fácil para las máquinas analizar y generar. Se utiliza principalmente para transmitir datos entre un servidor y una cliente web, como un medio alternativo a XML.

```
{
  "nombre": "Juan",
  "edad": 30,
  "ciudad": "Valladolid"
}
```

La parte de la izquierda se llama clave y la otra parte valor.

La funcion `json_encode` convierte un array asociativo en un objeto JSON.

EJERCICIO16: Escribe un script PHP que convierta el array

```
$Frutas = array(
        "Manzana" => "roja",
        "Plátano" => "amarillo",
        "Naranja" => "naranja"
    );
```

En un objeto JSON.

Usa json_encode.

UT 3: INTRODUCCIÓN A PHP

1. ¿Qué es PHP?
2. ¿Cómo funciona PHP?
3. Sintaxis básica
4. Tipos de datos
5. Variables
6. Constantes
7. Salida y Depuración de Datos
8. Operadores
9. Combinar PHP y HTML
10. Estructuras de control
11. Bucles
12. Arrays y bucles
13. **Codificación**
14. Inclusión

13. CODIFICACIÓN

UTF-8 es un alfabeto con el que se puede representar casi todos los caracteres de todos los idiomas del mundo. Cada carácter, ya sea una letra del alfabeto inglés o un emoji, tiene un código único en UTF-8.

UTF-8 fue diseñado de tal manera que los códigos para los caracteres más comunes (como las letras y números en inglés) son más cortos, lo que ayuda a ahorrar espacio.

Además, UTF-8 es compatible con ASCII, una codificación de caracteres más antigua que sólo incluye caracteres básicos del inglés. Esto significa que cualquier texto que se pueda representar en ASCII también se puede representar en UTF-8 usando exactamente los mismos códigos.

13. CODIFICACIÓN

El problema es que json_encode escapa caracteres no ASCII.

Por ejemplo, si tienes una cadena con el carácter “á”, json_encode lo convertirá a su representación Unicode, como “\u00e1”.

Si quieres que json_encode mantenga los caracteres especiales tal como están, puedes pasar la constante JSON_UNESCAPED_UNICODE.

```
json_encode($string, JSON_UNESCAPED_UNICODE);
```

13. CODIFICACIÓN

Buenas prácticas para trabajar con UTF8:

- **Codificación del archivo**: El editor de texto debe estar configurado para guardar los archivos en UTF-8.
- **Metaetiqueta de codificación de caracteres**: En tus archivos HTML, se debe incluir una metaetiqueta en la sección <head> para especificar que estás utilizando UTF-8.

```
<meta charset="UTF-8">
```

- **Cabeceras HTTP**: Si se genera contenido dinámico con un lenguaje de servidor como PHP,se debe enviar una cabecera HTTP para especificar que se usa UTF-8.

```
header('Content-Type: text/html; charset=UTF-8');
```

13. CODIFICACIÓN

- **Formularios HTML**: Cuando se envían datos a través de un formulario HTML, esos datos se envían utilizando la codificación especificada en el formulario. Se deben de configurar para usar UTF-8.

```
<form accept-charset="UTF-8">
```

- **Base de datos**: Cuando se crea una base de datos o una tabla, se especifica que quieres usar UTF-8.

```
CREATE DATABASE mydatabase CHARACTER SET utf8mb4 COLLATE
utf8mb4_unicode_ci;
```

También debes asegurarte de que tu conexión a la base de datos está configurada para usar UTF-8.

UT 3: INTRODUCCIÓN A PHP

1. ¿Qué es PHP?
2. ¿Cómo funciona PHP?
3. Sintaxis básica
4. Tipos de datos
5. Variables
6. Constantes
7. Salida y Depuración de Datos
8. Operadores
9. Combinar PHP y HTML
10. Estructuras de control
11. Bucles
12. Arrays y bucles
13. Codificación
14. **Inclusión**

14. INCLUSIÓN

Inclusión de ficheros externos:

- include()
- require()

Ambos incluyen y evalúan el fichero especificado.

Diferencia: en caso de error include() produce un warning y require() un error fatal.

Se usará require() si al producirse un error debe interrumpirse la carga de la página.

EJEMPLO

```
<HTML>
    <HEAD>
      <TITLE>Título</TITLE>
      <?PHP
          // Incluir bibliotecas
            de funciones
          require ("conecta.php");
          require ("fecha.php");
          require ("cadena.php");
          require ("globals.php");
      ?>
    </HEAD>
    <BODY>
      <?PHP
          include ("cabecera.html");
      ?>
      // Código HTML + PHP
      . . .
      <?PHP
          include ("pie.html");
      ?>
    </BODY>
</HTML>
```

UT 3: INTRODUCCIÓN A PHP

1. ¿Qué es PHP?
2. ¿Cómo funciona PHP?
3. Sintaxis básica
4. Tipos de datos
5. Variables
6. Constantes
7. Salida y Depuración de Datos
8. Operadores
9. Combinar PHP y HTML
10. Estructuras de control
11. Bucles
12. Arrays y bucles
13. Codificación
14. Inclusión
15. **Interacción**
16. Configuración PHP

15. INTERACCIÓN

Existen varias formas de recibir información del cliente.

La más habitual es utilizar formularios o pasarlos por la URL.

Los datos de un formulario se almacenan en variables:

- Método get: $HTTP_GET_VARS['xxx'] o $_GET['xxx']
- Método post: $HTTP_POST_VARS['xxx'] o $_POST['xxx']
- $_REQUEST['xxx']: Almacena los valores de los métodos Post y Get.
  Xxx es el valor de la etiqueta name del componente del formulario.

Los datos pasados por URL se cogen con el método GET.

15. INTERACCIÓN

Si register_globals es ON, se puede acceder directamente con el valor del name, al contenido del componente Se crea una variable automática: $<valor del name>

$HTTP_SERVER_VARS[REQUEST_METHOD: Muestra el modo de envío.

EJERCICIO: Crear una página web que permita sumar dos números pasados por url.

UT 3: INTRODUCCIÓN A PHP

1. ¿Qué es PHP?
2. ¿Cómo funciona PHP?
3. Sintaxis básica
4. Tipos de datos
5. Variables
6. Constantes
7. Salida y Depuración de Datos
8. Operadores
9. Combinar PHP y HTML
10. Estructuras de control
11. Bucles
12. Arrays y bucles
13. Codificación
14. Inclusión