

Prácticas Dossier PHP



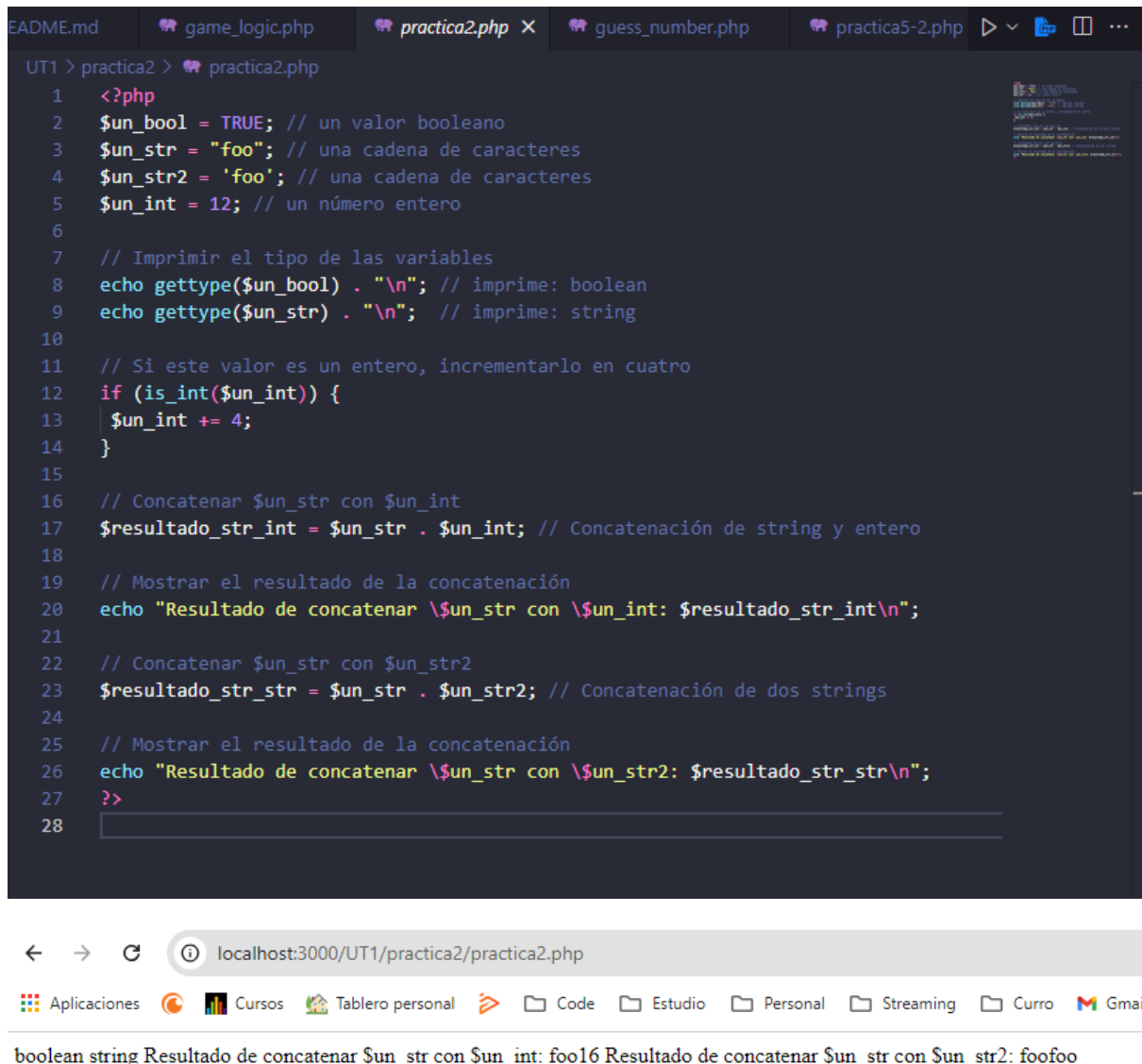
Práctica 1	4
Práctica 2	4
Explicación de los Resultados	5
Nota sobre comillas simples y dobles:	6
Práctica 3	6
Práctica 4	7
Práctica 5	8
Práctica 6	9
Práctica 7	9
Diferencias entre null y unset()	10
Conclusión	10
Práctica 8	10
Práctica 9	10
Práctica 10	11
Práctica 11	12
Práctica 12	14
Práctica 13	14
Practica 14	15
Practica 15	16
Practica 16	17
Conclusión	17
Practica 17	18
Resultados:	19
Diferencia con Java	19
Practica 18	19
Observaciones	20
Practica 19	21
¿Qué sucede en el código?	21
Resultado esperado	22
Conclusión	22
Practica 20	22
Resultado esperado	23

Salida del código	23
Practica 21	24
Explicación:.....	24
Practica 22	25
Explicación:.....	25
Practica 23	26
Practica 24	27
Practica 25	28
Explicación del código:.....	29
Practica 26	29
Explicación del código:.....	30
Practica 27	30
Practica 28	32
¿Se obtiene resultado o se detiene el programa?:.....	33
Practica 29	34
Explicación.....	34
Practica 30	35
Explicación del Código	35
Practica 31	36
Practica 32	37
Practica 33	38
Practica 34	39
Practica 35	40
Explicación del Código	41
Practica 36	42
Explicación del Código	43
Practica 37	44
Explicación del Código	46

Crear el script como se ha comentado sustituyendo “alumno” por nuestro nombre completo. Tomar captura de pantalla del resultado.



Crear el script anterior. Modificarlo para sumar a \$un_str el valor de \$un_int y mostrarlo en pantalla ¿qué ocurre?. Sumar \$un_str con \$un_str2 ¿qué ocurre? ¿se puede concatenar una cadena con comillas simples con una con comillas dobles?



The image shows a code editor with a dark theme. The top bar has tabs for 'EADME.md', 'game_logic.php', 'practica2.php' (active), 'guess_number.php', and 'practica5-2.php'. The code in 'practica2.php' is as follows:

```
1 <?php
2 $un_bool = TRUE; // un valor booleano
3 $un_str = "foo"; // una cadena de caracteres
4 $un_str2 = 'foo'; // una cadena de caracteres
5 $un_int = 12; // un número entero
6
7 // Imprimir el tipo de las variables
8 echo gettype($un_bool) . "\n"; // imprime: boolean
9 echo gettype($un_str) . "\n"; // imprime: string
10
11 // Si este valor es un entero, incrementarlo en cuatro
12 if (is_int($un_int)) {
13     $un_int += 4;
14 }
15
16 // Concatenar $un_str con $un_int
17 $resultado_str_int = $un_str . $un_int; // Concatenación de string y entero
18
19 // Mostrar el resultado de la concatenación
20 echo "Resultado de concatenar \$un_str con \$un_int: $resultado_str_int\n";
21
22 // Concatenar $un_str con $un_str2
23 $resultado_str_str = $un_str . $un_str2; // Concatenación de dos strings
24
25 // Mostrar el resultado de la concatenación
26 echo "Resultado de concatenar \$un_str con \$un_str2: $resultado_str_str\n";
27 ?>
28
```

Below the code editor is a web browser window showing the output of the script at 'localhost:3000/UT1/practica2/practica2.php'. The browser's address bar and tabs are visible. The output in the browser is:

```
boolean
string
Resultado de concatenar $un_str con $un_int: foo16
Resultado de concatenar $un_str con $un_str2: foofoo
```

Explicación de los Resultados

Concatenar ` \$un_str ` con ` \$un_int `:

- Cuando concatenas una cadena con un entero en PHP, el entero se convierte en una cadena y luego se concatenan.

Resultado: ` "foo12" `

Concatenar ` \$un_str ` con ` \$un_str2 `:

- Aquí simplemente concatenas dos cadenas.

Resultado: ` "foofoo" `

Nota sobre comillas simples y dobles:

- Las comillas simples (```) en PHP no interpretan variables dentro de la cadena. Por ejemplo, ``foo`` siempre será ``foo`` y no interpretará ninguna variable dentro.
- Las comillas dobles (`"`) permiten la interpretación de variables y secuencias de escape, así que `"foo"` es interpretado como `"foo"` y las variables dentro de la cadena serán reemplazadas por su valor.

Por lo tanto, la concatenación entre cadenas con comillas simples y dobles funcionará igual en el contexto de la concatenación, porque estás concatenando cadenas con cadenas, no variables.

Práctica 3

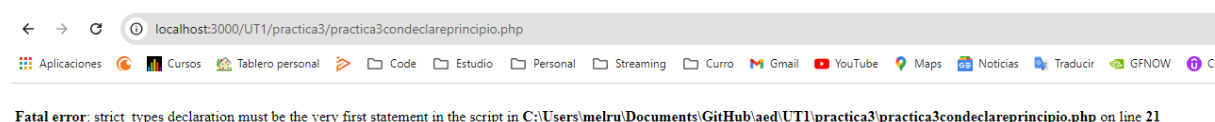
Realizar el código anterior y tomar captura de pantalla del resultado. ¿qué es lo que ha ocurrido? Poner código html antes de la declaración de `strict_types` y probar de nuevo ¿qué ocurre ahora?

Declare al principio:



La suma de uno más dos es: 3

Declare después:



La declaración `declare(strict_types=1)` debe estar al principio del archivo PHP. Si intentas colocar cualquier contenido antes de esta declaración, obtendrás un error o `strict_types` no tendrá efecto.

Práctica 4

Tomar captura de pantalla de: (y explicar lo ocurrido) - probar el código anterior ¿da error? Por qué? - quitar el comentario a: `return $a;` ¿da error ahora? por qué? - quitar comentario a: `print fun("e",3);` ¿da error?

El código:



The screenshot shows a code editor with a dark theme. At the top, there are tabs for 'README.md', 'practica4.php 1 M X', 'game_logic.php', and 'guess_number.php'. A red error message box is displayed over the code, stating: 'strict_types declaration must be the very first statement in the script'. Below the error message, there are two buttons: 'View Problem (Alt+F8)' and 'Quick Fix... (Ctrl+.)'. The code in the editor is as follows:

```
1 |  
2 |  
3 | declare( strict_types=1);  
4 | ?>  
5 | <!DOCTYPE html>  
6 | <html>  
7 |  
8 |     <head>  
9 |         <meta charset="UTF-8">  
10 |         <title></title>  
11 |     </head>  
12 |     <body>  
13 |         <?php  
14 |             function fun( int $a, int $b): int {  
15 |                 $a = "o";  
16 |                 //return $a;  
17 |                 return $b ;  
18 |             }  
19 |             print fun(1,2);  
20 |             //print fun("e",3);  
21 |             echo "</p>"  
22 |         ?>  
23 |     </body>  
24 | </html>
```

Nos indica el error de que la declaración debe estar primero en el script.

Al descomentar `return $a;`, la función intentará devolver una cadena ("o"), lo que causará un error debido a que el tipo de retorno debe ser `int`.



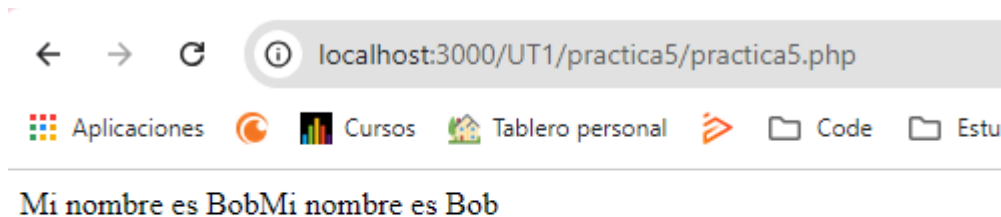
Al quitar el segundo comentario sigue dando error de `(print fun(1, 2);)`: La ejecución es correcta y muestra 2, ya que el valor de retorno es del tipo esperado (int).

Práctica 5

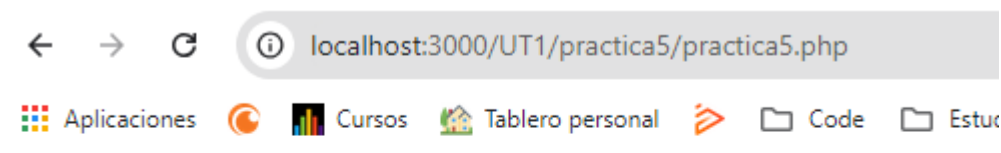
Probar el código anterior. Probar ahora con números ¿también funcionan las referencias?

```
<?php
$foo = 'Bob'; // Asigna el valor 'Bob' a $foo
$bar = &$foo; // Referencia $foo vía $bar.
$bar = "Mi nombre es $bar"; // Modifica $bar...
echo $foo; // $foo también se modifica.
echo $bar;
?>
```

Con el código proporcionado:



Con números:



Las referencias funcionan indistintamente con números o string.

Práctica 6

Hacer un script php que haga echo de \$_SERVER y de \$_SERVER [PHP_SELF] tomar captura de pantalla de los resultados



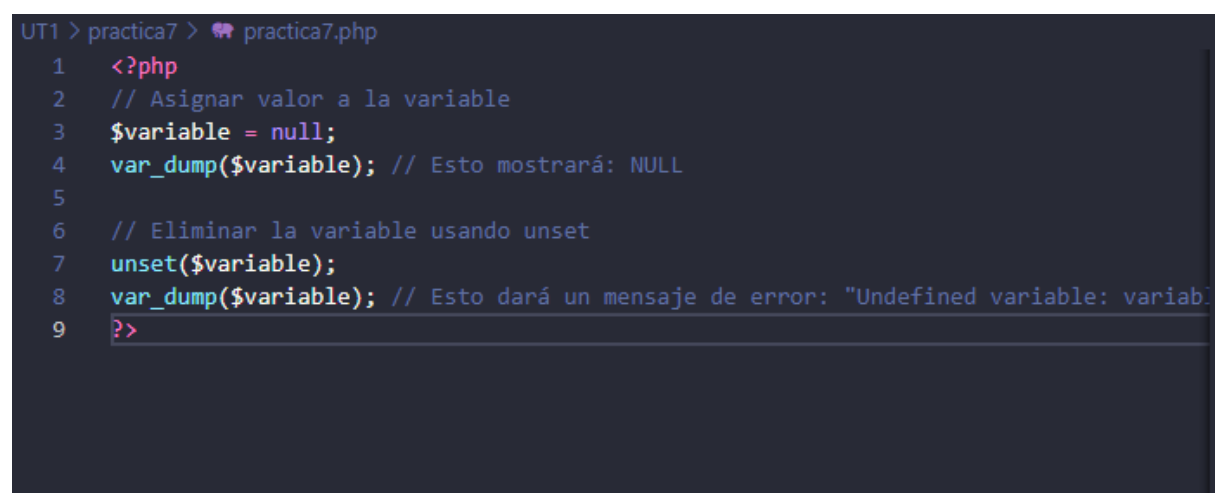
The image shows a terminal window with a dark background and a web browser window below it. The terminal shows the command 'UT1 > practica6 > practica6.php' and the output of a PHP script. The script uses var_dump to display the contents of \$_REQUEST, \$_SERVER['SERVER_NAME'], and \$_SERVER. The web browser shows the output of the script as a long string of PHP var_dump output, including details about the server, request, and user agent.

```
UT1 > practica6 > practica6.php
1 <?php
2     var_dump($_REQUEST);
3     var_dump($_SERVER["SERVER_NAME"]);
4     var_dump($_SERVER)
5
6 ?>
```

array(0) {} string(9) "localhost" array(29) (["DOCUMENT_ROOT"]=> string(35) "C:\Users\melru\Documents\GitHub\aed" ["REMOTE_ADDR"]=> string(3) "1" ["REMOTE_PORT"]=> string(5) "57136" ["SERVER_SOFTWARE"]=> string(29) "PHP 8.2.12 Development Server" ["SERVER_PROTOCOL"]=> string(5) "HTTP/1.1" ["SERVER_NAME"]=> string(9) "localhost" ["SERVER_PORT"]=> string(4) "3000" ["REQUEST_URI"]=> string(28) "/UT1/practica6/practica6.php" ["REQUEST_METHOD"]=> string(3) "GET" ["SCRIPT_NAME"]=> string(28) "/UT1/practica6/practica6.php" ["SCRIPT_FILENAME"]=> string(63) "C:\Users\melru\Documents\GitHub\aed\UT1/practica6/practica6.php" ["PHP_SELF"]=> string(28) "/UT1/practica6/practica6.php" ["HTTP_HOST"]=> string(14) "localhost:3000" ["HTTP_CONNECTION"]=> string(10) "keep-alive" ["HTTP_SEC_CH_UA"]=> string(64) "\"Google Chrome\";v=129\", \"Not-A-Brand\";v=8\", \"Chromium\";v=129\" ["HTTP_SEC_CH_UA_MOBILE"]=> string(2) "0\" ["HTTP_SEC_CH_UA_PLATFORM"]=> string(9) "\"Windows\" ["HTTP_UPGRADE_INSECURE_REQUESTS"]=> string(1) "1\" ["HTTP_USER_AGENT"]=> string(111) "Mozilla/5.0 (Windows NT 10.0; Win64; x64; AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Safari/537.36\" ["HTTP_ACCEPT"]=> string(135) "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7\" ["HTTP_SEC_FETCH_SITE"]=> string(4) "none\" ["HTTP_SEC_FETCH_MODE"]=> string(8) "navigate\" ["HTTP_SEC_FETCH_USER"]=> string(2) "1\" ["HTTP_SEC_FETCH_DEST"]=> string(8) "document\" ["HTTP_ACCEPT_ENCODING"]=> string(23) "gzip, deflate, br, zstd\" ["HTTP_ACCEPT_LANGUAGE"]=> string(47) "es-ES;q=0.9,en-GB;q=0.8,en-XA;q=0.7,en;q=0.6\" ["HTTP_COOKIE"]=> string(87) "Ideas-af5f5ec=231cca95-0a8f-4905-a18c-bd1a23786ad, PHPSESSID=jyoghs2e9pskf9cfredekthurt\" ["REQUEST_TIME_FLOAT"]=> float(1727260006.887941) ["REQUEST_TIME"]=> int(1727260006) }

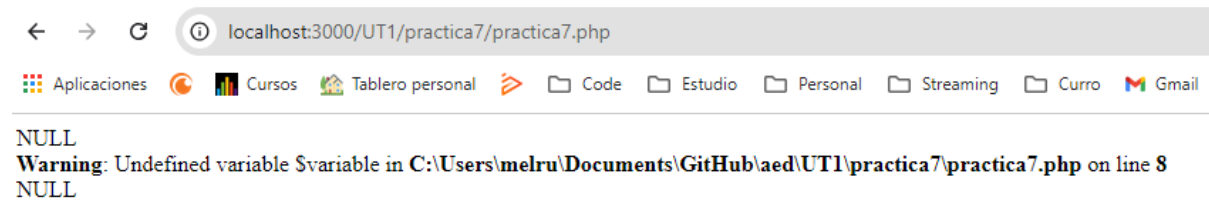
Práctica 7

Visualizar lo anterior ¿se encuentran diferencias entre null y unset() ? Tomar captura de pantalla



The image shows a terminal window with a dark background. The terminal shows the command 'UT1 > practica7 > practica7.php' and the output of a PHP script. The script assigns a null value to a variable, dumps it, and then unsets it, dumping it again to show an error message.

```
UT1 > practica7 > practica7.php
1 <?php
2 // Asignar valor a la variable
3 $variable = null;
4 var_dump($variable); // Esto mostrará: NULL
5
6 // Eliminar la variable usando unset
7 unset($variable);
8 var_dump($variable); // Esto dará un mensaje de error: "Undefined variable: variabl
9 ?>
```



```
← → ↻ localhost:3000/UT1/practica7/practica7.php
Aplicaciones Cursos Tablero personal Code Estudio Personal Streaming Curro Gmail
NULL
Warning: Undefined variable $variable in C:\Users\melru\Documents\GitHub\aed\UT1\practica7\practica7.php on line 8
NULL
```

Diferencias entre null y unset()

Asignar una variable a null significa que la variable todavía existe pero no tiene un valor.

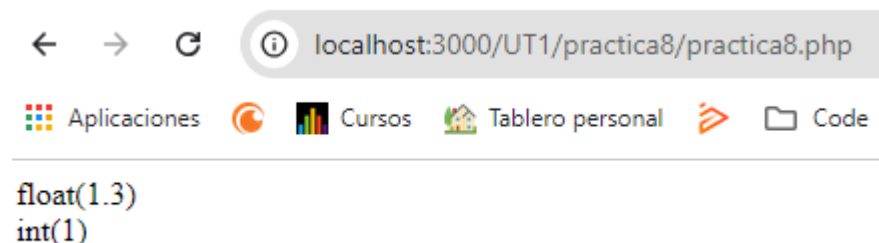
Usar unset(\$variable) elimina la variable por completo, como si nunca hubiera existido.

Conclusión

null y unset() son diferentes en cuanto a cómo manejan las variables. null mantiene la variable existente pero sin valor, mientras que unset() elimina completamente la variable de la memoria.

Práctica 8

Ejecutar el script anterior ¿hay alguna diferencia antes y después del cast? Tomar captura de pantalla



```
← → ↻ localhost:3000/UT1/practica8/practica8.php
Aplicaciones Cursos Tablero personal Code
float(1.3)
int(1)
```

El "cast" convierte el tipo de dato y suprime la parte decimal del número. La variable pasa de ser un float a un int con la conversión.

Práctica 9

Crear un script que muestre las potencias del número 2 desde 2^1 hasta 2^9 hacer uso del operador: ** Ir concatenando las salidas en pantalla de las potencias en una string mediante el operador de concatenación y asignación: .=

```
UT1 > practica9 > practica9.php
1  <?php
2  // Inicializar una cadena vacía para almacenar el resultado
3  $resultado = "";
4
5  // Calcular las potencias del 2 desde 2^1 hasta 2^9
6  for ($i = 1; $i <= 9; $i++) {
7      // Calcular 2 elevado a la potencia $i
8      $potencia = 2 ** $i;
9
10     // Concatenar el resultado a la cadena
11     $resultado .= "2^$i = $potencia<br>";
12 }
13
14 // Mostrar el resultado final
15 echo $resultado;
16 ?>
```

← → ↻ ⓘ localhost:3000/UT1/practica9/practica9.php

🧩 Aplicaciones 🔄 Cursos 🏠 Tablero personal 📁 Code

```
2^1 = 2
2^2 = 4
2^3 = 8
2^4 = 16
2^5 = 32
2^6 = 64
2^7 = 128
2^8 = 256
2^9 = 512
```

Práctica 10

Crear un programa en php que obtenga la descomposición de un número que esté almacenado en la variable: \$numero Por ejemplo: \$numero = 3102 Se pretende que se utilicen en el programa los operadores: .= , ** Para el ejemplo anterior se debe mostrar en pantalla: 2 * 1 + 0 * 10 + 1 * 100 + 3 * 1000

```
UT1 > practica10 > practica10.php
1  <?php
2  // Número a descomponer
3  $numero = 3102;
4
5  // Convertir el número a cadena para poder recorrer cada dígito
6  $numero_str = (string) $numero;
7
8  // Inicializar la variable para almacenar el resultado de la descomposición
9  $descomposicion = "";
10
11 // Obtener la longitud del número
12 $longitud = strlen($numero_str);
13
14 // Recorrer cada dígito del número
15 for ($i = 0; $i < $longitud; $i++) {
16     // Obtener el dígito actual
17     $digito = $numero_str[$i];
18
19     // Calcular la potencia de 10 correspondiente a la posición
20     $potencia = ($longitud - 1) - $i;
21     $valor = 10 ** $potencia;
22
23     // Concatenar el resultado en el formato requerido
24     $descomposicion .= $digito . " * " . $valor;
25
26     // Añadir el signo '+' si no es el último dígito
27     if ($i < $longitud - 1) {
28         $descomposicion .= " + ";
29     }
30 }
31
32 // Mostrar la descomposición
33 echo $descomposicion;
34 ?>
```

← → ↻ ⓘ localhost:3000/UT1/practica10/practica10.php

Aplicaciones Cursos Tablero personal Code Es

$3 * 1000 + 1 * 100 + 0 * 10 + 2 * 1$

Práctica 11

Ejecutar el script y tomar captura de pantalla

```
UT1 > practica11 > practica11.php
1  <?php
2  $var = "";
3  if(empty($var)){ // true because "" is considered empty
4      echo '<br>empty($var) para $var="" ';
5
6  }else{
7      echo '<br>!empty($var) para $var="" ';
8  }
9  if(isset($var)){ //true because var is set
10     echo '<br>isset($var) para $var="" ';
11 }else{
12     echo '<br> !isset($var) para $var="" ';
13 }
14
15 if(empty($otherVar)){ //true because $otherVar is null
16     echo '<br>empty($otherVar) para $otherVar que no se ha establecido ';
17 } else {
18     echo '<br> !empty($otherVar) para $otherVar que no se ha establecido ';
19 }
20 if(isset($otherVar)){ //false because $otherVar is not set
21     echo '<br>isset($otherVar) para $otherVar que no se ha establecido ';
22 } else {
23     echo '<br> !isset($otherVar) para $otherVar que no se ha establecido ';
24 }
25 ?>
```



Observamos que `isset()` es equivalente a `!empty()` salvo en el caso de una variable que es comillas vacías. En ese caso `isset()` devolverá `false` mientras que `!empty()` devolverá `true`

Práctica 12

Probar el script anterior y observar que ocurre. ¿qué mensaje de error se observa?

Me produce error

```
UT1 > practica12 > practica12.php
1  <?php
2
3  $array = array('uno' => 1, 'dos' => 2, 'tres' => 40, 'cuatro' => 55);
4  $cadena = "La posición 'tres' contiene el dato $array['tres']"
5
6  ?>
```

localhost:3000/UT1/practica12/practica12.php

Aplicaciones Cursos Tablero personal Code Estudio Personal Streaming Curro Gmail Todos I

Parse error: syntax error, unexpected token "=>", expecting ")" in C:\Users\melru\Documents\GitHub\aed\UT1\practica12\practica12.php on line 3

Al poner llave lo solucionamos:

```
UT1 > practica12 > practica12.php
1  <?php
2  $array = array('uno' => 1, 'dos' => 2, 'tres' => 40, 'cuatro' => 55);
3  $cadena = "La posición 'tres' contiene el dato {$array['tres']}";
4  echo $cadena;
5  ?>
```

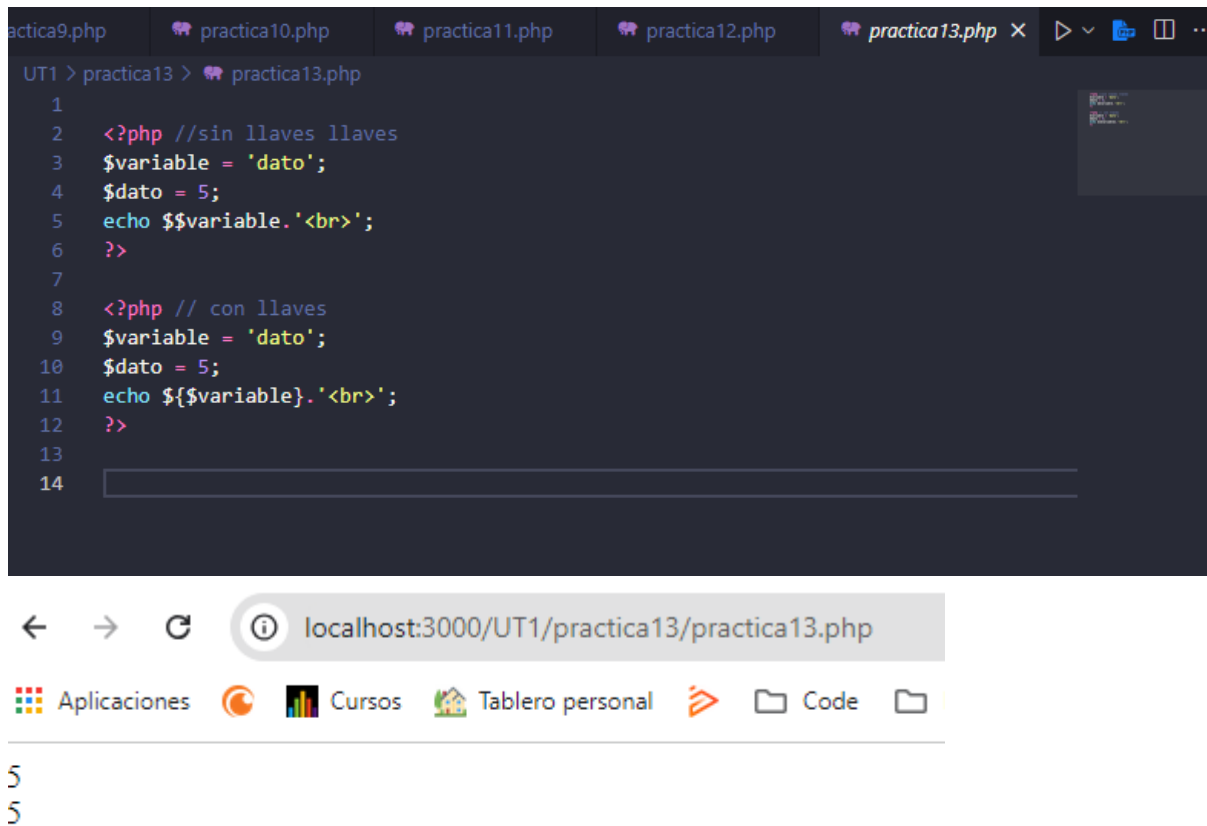
localhost:3000/UT1/practica12/practica12.php

Aplicaciones Cursos Tablero personal Code

La posición 'tres' contiene el dato 40

Práctica 13

Probar el script anterior y observar que ocurre. Probar ahora con llaves: \${\$variable}
¿hay diferencia?



The screenshot shows a web browser window with the address bar displaying `localhost:3000/UT1/practica13/practica13.php`. The browser's taskbar at the bottom includes icons for 'Aplicaciones', 'Cursos', 'Tablero personal', and 'Code'. The main content area of the browser shows the output of the PHP script, which consists of two lines, both displaying the number '5'.

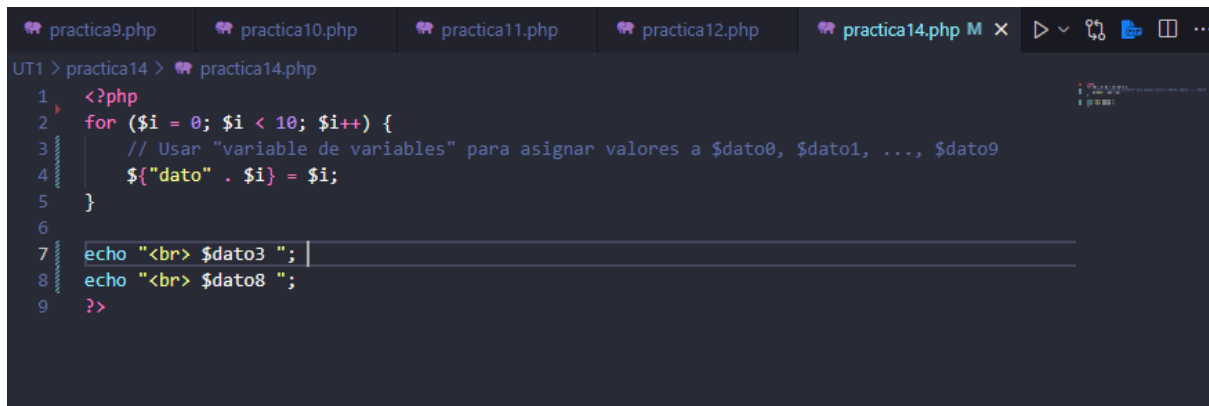
```
UT1 > practica13 > practica13.php
1
2 <?php //sin llaves llaves
3 $variable = 'dato';
4 $dato = 5;
5 echo $$variable.'<br>';
6 ?>
7
8 <?php // con llaves
9 $variable = 'dato';
10 $dato = 5;
11 echo ${$variable}.'<br>';
12 ?>
13
14
```

5
5

No hay diferencia funcional entre `$$variable` y `${$variable}` en este contexto; ambos se evalúan de la misma manera y producen el mismo resultado. La sintaxis con llaves (`${$variable}`) suele utilizarse para mayor claridad o en situaciones donde la expresión de la variable es más compleja, por ejemplo, cuando trabajas con arrays o necesitas desambiguar el nombre de la variable.

Practica 14

Toma el código anterior e introduce una expresión “variable de variables” que permita definir las variables: `$dato0`, `$dato1`, ..., `$dato9` Cada una de ellas con el valor correspondiente: 0, 1,...,9



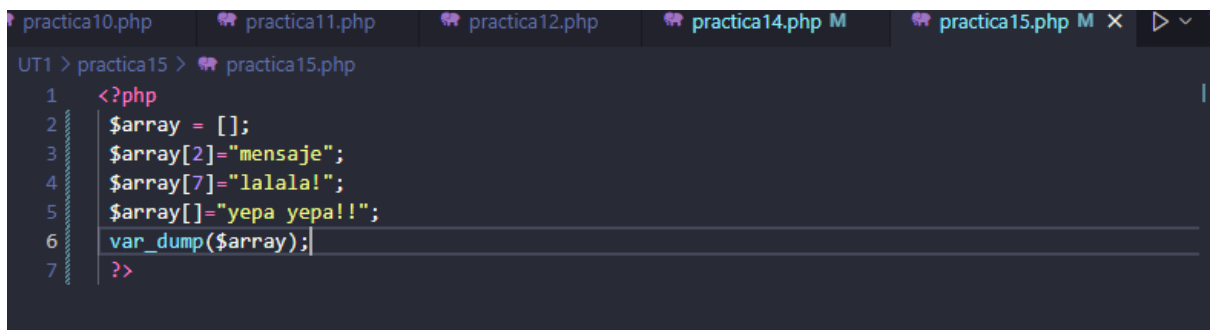
```
practica9.php practica10.php practica11.php practica12.php practica14.php M x
UT1 > practica14 > practica14.php
1  <?php
2  for ($i = 0; $i < 10; $i++) {
3      // Usar "variable de variables" para asignar valores a $dato0, $dato1, ..., $dato9
4      ${"dato" . $i} = $i;
5  }
6
7  echo "<br> $dato3 ";
8  echo "<br> $dato8 ";
9  ?>
```



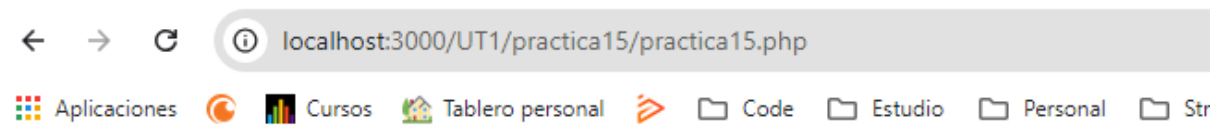
3
8

Recorro en un for para ir asignando valores y después hago echo para comprobar funcionamiento

Practica 15



```
practica10.php practica11.php practica12.php practica14.php M practica15.php M x
UT1 > practica15 > practica15.php
1  <?php
2  $array = [];
3  $array[2]="mensaje";
4  $array[7]="lalala!";
5  $array[]="yepa yepa!!";
6  var_dump($array);
7  ?>
```



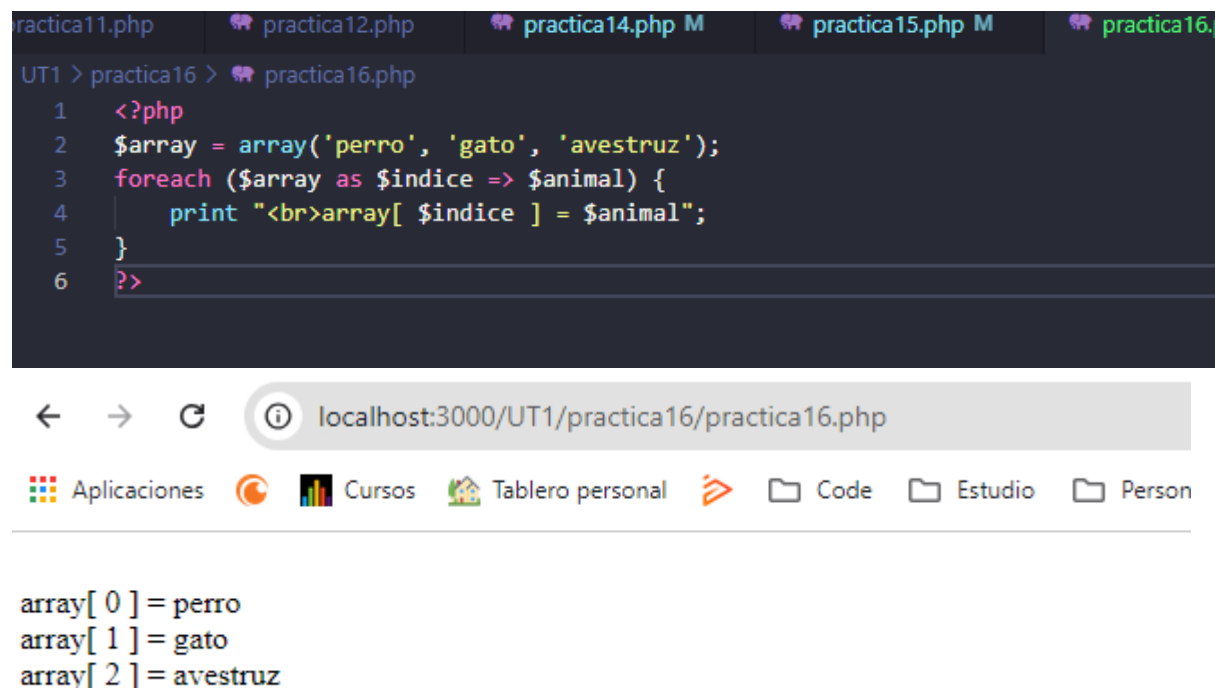
```
array(3) { [2]=> string(7) "mensaje" [7]=> string(7) "lalala!" [8]=> string(11) "yepa yepa!!" }
```

¿Hay diferencias en la salida en pantalla?

No hay diferencias entre usar `$array = []` y `$array = array()` en cuanto al resultado final. Ambos métodos de creación del array funcionan de la misma manera y generan la misma estructura de datos. La diferencia entre `[]` y `array()` es solo sintáctica; `[]` es la sintaxis corta introducida en PHP 5.4, mientras que `array()` es la forma tradicional.

- PHP muestra únicamente las posiciones que se han asignado explícitamente: 2, 7, y luego 8.
- No hay valores entre la posición 2 y 7 ni entre 0 y 1, ya que esas posiciones no fueron definidas.
- Cuando se usa `$array[]`, PHP asigna automáticamente el siguiente índice disponible (en este caso, 8).

Practica 16



```
practica11.php  practica12.php  practica14.php M  practica15.php M  practica16.php
UT1 > practica16 > practica16.php
1  <?php
2  $array = array('perro', 'gato', 'avestruz');
3  foreach ($array as $indice => $animal) {
4      print "<br>array[ $indice ] = $animal";
5  }
6  ?>
```

array[0] = perro
array[1] = gato
array[2] = avestruz

En este caso, hemos sustituido `$key` por `$indice` y `$val` por `$animal`.

El bucle `foreach` sigue funcionando de la misma manera, iterando a través del array y mostrando el par clave-valor.

Conclusión

Puedes usar cualquier nombre de variable que desees, siempre que las variables sean coherentes dentro del bloque `foreach`. PHP no tiene restricciones sobre los nombres de las variables en este contexto.

Practica 17

Ejecutar el script anterior. En Java eliminar elementos de un array en un foreach implica un error ¿también en php? Tomar captura de pantalla del resultado



```
UT1 > practica17 > practica17.php
1  <?php
2  $array = [];
3  for ($i = 0; $i < 5; $i++) {
4      $array[] = "a" . $i;
5  }
6
7  $j = count($array);
8  foreach ($array as $key => $val) {
9      $j--;
10     unset($array[$j]); // Elimina el elemento en la posición $j
11     echo "<br>";
12     var_dump($array); // Muestra el estado actual del array
13     echo "<br> $key => $val ";
14     echo "<br>";
15 }
16 ?>
17
```

localhost:3000/UT1/practica17/practica17.php

Aplicaciones Cursos Tablero personal Code Estudio Personal St

```
array(4) { [0]=> string(2) "a0" [1]=> string(2) "a1" [2]=> string(2) "a2" [3]=> string(2) "a3" }
0 => a0

array(3) { [0]=> string(2) "a0" [1]=> string(2) "a1" [2]=> string(2) "a2" }
1 => a1

array(2) { [0]=> string(2) "a0" [1]=> string(2) "a1" }
2 => a2

array(1) { [0]=> string(2) "a0" }
3 => a3

array(0) { }
4 => a4
```

```
array(4) { [0]=> string(2) "a0" [1]=> string(2) "a1" [2]=> string(2) "a2" [3]=> string(2) "a3" }
0 => a0
```

```
array(3) { [0]=> string(2) "a0" [1]=> string(2) "a1" [2]=> string(2) "a2" }
1 => a1
```

```
array(2) { [0]=> string(2) "a0" [1]=> string(2) "a1" }
2 => a2
```

```
array(1) { [0]=> string(2) "a0" }
3 => a3
```

```
array(0) { }
4 => a4
```

El bucle for crea un array: ["a0", "a1", "a2", "a3", "a4"].

Dentro del foreach, se elimina un elemento en la posición j (de 4 a 0).

var_dump(\$array) muestra la estructura del array después de cada eliminación.

La iteración de foreach sigue recorriendo el array original, aunque se hayan eliminado algunos elementos.

Resultados:

Al eliminar elementos con unset(\$array[\$j]), PHP ajusta el array pero no genera un error.

El bucle foreach sigue iterando sobre las claves y valores que existían al inicio, a pesar de las eliminaciones.

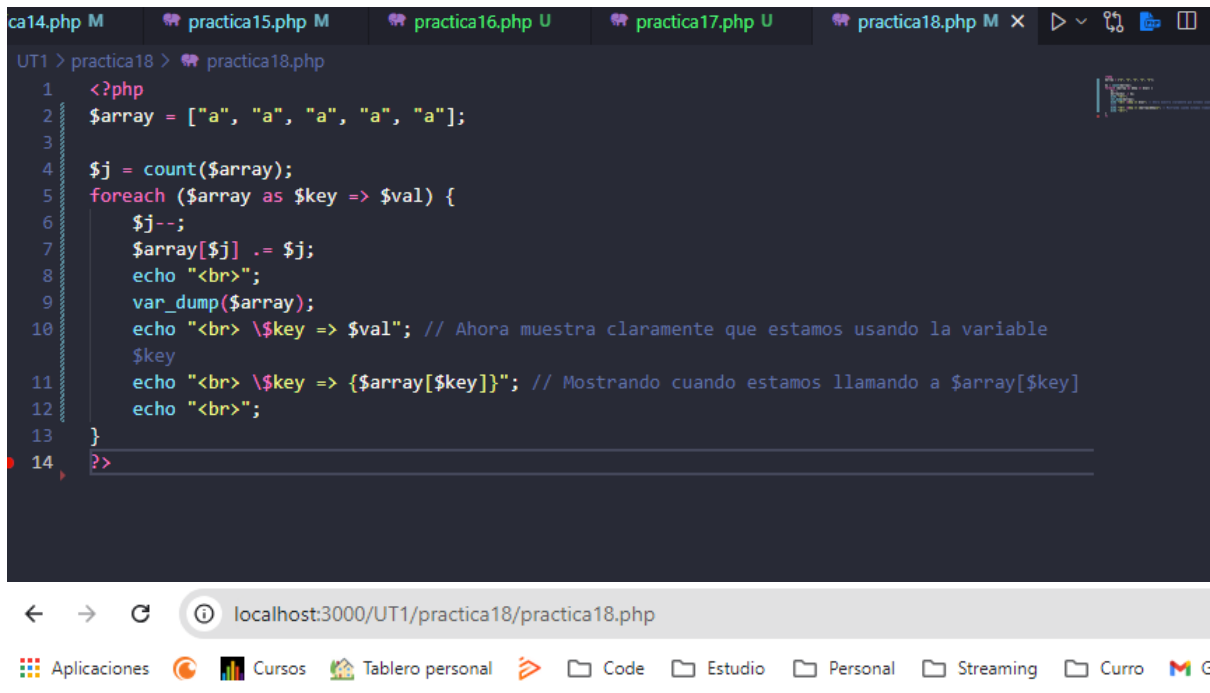
Diferencia con Java

En Java, eliminar elementos de un array o lista mientras se itera provoca un error (ConcurrentModificationException). PHP, en cambio, permite la modificación sin problemas durante la iteración, aunque los cambios no afecten a la estructura que el foreach está usando internamente.

En conclusión, no se genera un error en PHP al eliminar elementos de un array dentro de un foreach, lo que refleja la flexibilidad de cómo PHP maneja la iteración y la manipulación de arrays.

Practica 18

Ejecutar el script anterior. Modificar los echo para que se sepa cuando llamamos a \$array (recordar que con comillas simples no interpreta) Tomar captura de pantalla



```
UT1 > practica18 > practica18.php
1  <?php
2  $array = ["a", "a", "a", "a", "a"];
3
4  $j = count($array);
5  foreach ($array as $key => $val) {
6      $j--;
7      $array[$j] .= $j;
8      echo "<br>";
9      var_dump($array);
10     echo "<br> \$key => $val"; // Ahora muestra claramente que estamos usando la variable
11     $key
12     echo "<br> \$key => {$array[$key]}"; // Mostrando cuando estamos llamando a $array[$key]
13     echo "<br>";
14 }
15 ?>
```

localhost:3000/UT1/practica18/practica18.php

Aplicaciones Cursos Tablero personal Code Estudio Personal Streaming Curro

```
array(5) { [0]=> string(1) "a" [1]=> string(1) "a" [2]=> string(1) "a" [3]=> string(1) "a" [4]=> string(2) "a4" }
$key => a
$key => a
```

```
array(5) { [0]=> string(1) "a" [1]=> string(1) "a" [2]=> string(1) "a" [3]=> string(2) "a3" [4]=> string(2) "a4" }
$key => a
$key => a
```

```
array(5) { [0]=> string(1) "a" [1]=> string(1) "a" [2]=> string(2) "a2" [3]=> string(2) "a3" [4]=> string(2) "a4" }
$key => a
$key => a2
```

```
array(5) { [0]=> string(1) "a" [1]=> string(2) "a1" [2]=> string(2) "a2" [3]=> string(2) "a3" [4]=> string(2) "a4" }
$key => a
$key => a3
```

```
array(5) { [0]=> string(2) "a0" [1]=> string(2) "a1" [2]=> string(2) "a2" [3]=> string(2) "a3" [4]=> string(2) "a4" }
$key => a
$key => a4
```

Usar `\$key` indica que `$key` es una variable en la salida.

La estructura `{ $array[$key] }` asegura que la variable se interprete correctamente dentro del `echo` y destaca que es parte de un array.

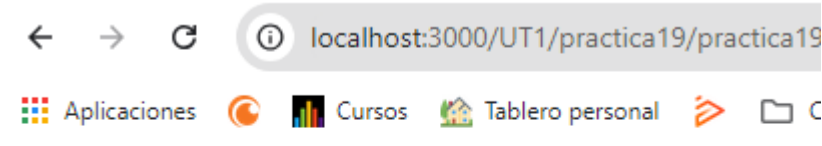
Los `var_dump($array)` y otros `echo` mostrarán el estado del array en cada iteración, y ahora se sabe claramente cuándo se hace referencia a las variables.

Observaciones

El código ahora es más legible y muestra claramente cuándo se está accediendo a las variables y elementos del array durante la iteración.

Practica 19

```
UT1 > practica19 > practica19.php
1  <?php
2  $arr = array(1, 2, 3, 4);
3  foreach ($arr as &$amp;val) {
4  $val = $val * 2;
5  }
6  foreach ($arr as $key => $val) {
7  echo "{$key} => {$val} <br>";
8  print_r($arr);
9  echo "<br><br>";
10 }
11 ?>
12
```



```
0 => 2
Array ( [0] => 2 [1] => 4 [2] => 6 [3] => 2 )

1 => 4
Array ( [0] => 2 [1] => 4 [2] => 6 [3] => 4 )

2 => 6
Array ( [0] => 2 [1] => 4 [2] => 6 [3] => 6 )

3 => 6
Array ( [0] => 2 [1] => 4 [2] => 6 [3] => 6 )
```

¿Qué sucede en el código?

Primer foreach con referencia (&\$val):

\$val es una referencia a cada elemento del array \$arr. Esto significa que cualquier cambio en \$val modifica directamente el elemento del array.

El resultado es que cada valor en \$arr se multiplica por 2, transformando \$arr en [2, 4, 6, 8].

Segundo foreach sin referencia:

Esta vez, \$val no es una referencia, sino una copia del valor actual.

Sin embargo, debido a que el primer bucle dejó \$val como una referencia, la última posición del array sigue referenciada a \$val.

Esto puede generar un comportamiento inesperado si \$val cambia durante la iteración.

Resultado esperado

Durante el segundo bucle:

En la primera iteración, `print_r($arr)` mostrará `[2, 4, 6, 8]`.

En las siguientes iteraciones, \$val se cambia y puede afectar el último elemento del array debido a la referencia residual, causando resultados inesperados.

Conclusión

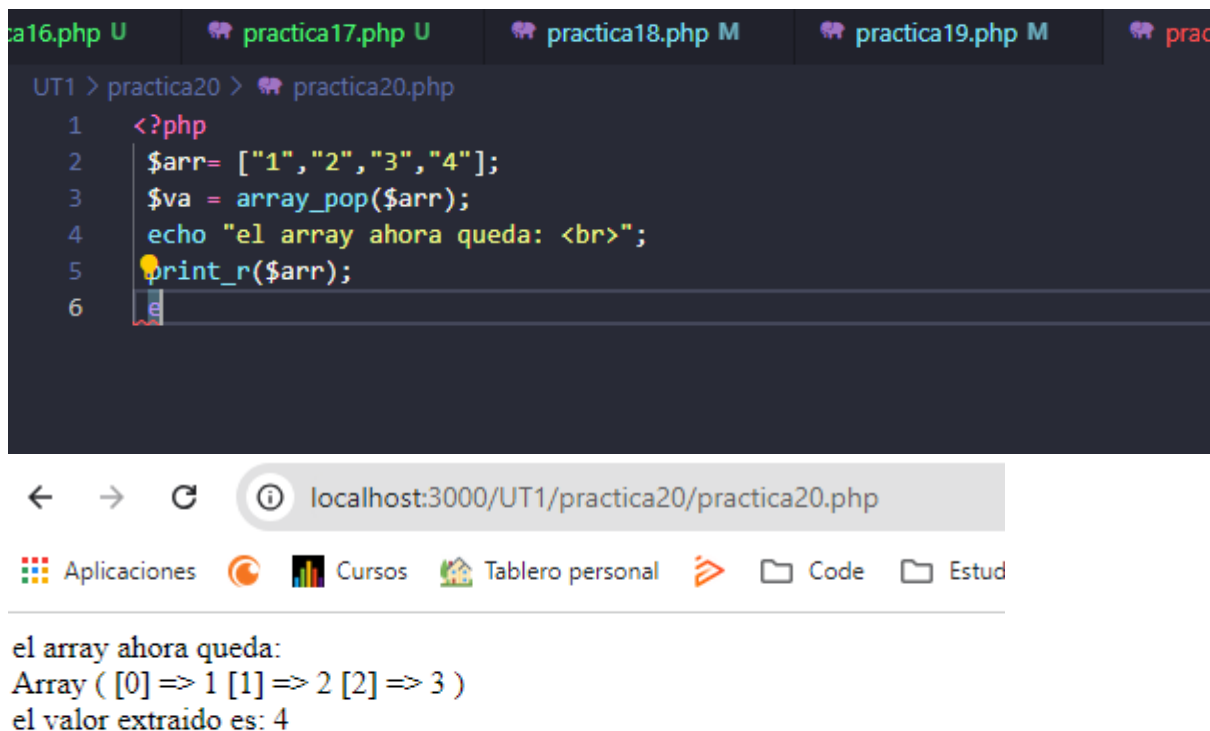
Este comportamiento es un efecto secundario de no liberar la referencia (&\$val) después del primer foreach. En PHP, para evitar este tipo de problemas, es una buena práctica usar `unset($val)`; después de un bucle foreach que usa referencias:

```
unset($val);
```

Al añadir `unset($val)` después del primer bucle foreach, evitarías que la referencia persista y el segundo bucle foreach se comportaría de manera esperada.

Practica 20

Ejecutar el script anterior. ¿ qué valor devuelve ?Tomar captura de pantalla



```
UT1 > practica20 > practica20.php
1  <?php
2  $arr= ["1","2","3","4"];
3  $va = array_pop($arr);
4  echo "el array ahora queda: <br>";
5  print_r($arr);
6
```

localhost:3000/UT1/practica20/practica20.php

Aplicaciones Cursos Tablero personal Code Estud

el array ahora queda:
Array ([0] => 1 [1] => 2 [2] => 3)
el valor extraido es: 4

La función `array_pop` elimina y devuelve el último elemento de un array.

Resultado esperado

`$arr` inicialmente es `["1", "2", "3", "4"]`.

`array_pop($arr)` elimina el último elemento "4" y lo asigna a `$va`.

Después de la ejecución, `$arr` es `["1", "2", "3"]`, y `$va` tiene el valor "4".

Salida del código

el array ahora queda:

Array ([0] => 1 [1] => 2 [2] => 3)

el valor extraido es: 4

En resumen, `array_pop` devuelve el último valor "4" y lo elimina del array, dejando `["1", "2", "3"]` en `$arr`.

Practica 21

Crear un script que por medio de un bucle for que vaya de 1 a 10 agregue esos números en un array. En cada iteración mostrar el contenido del array. Después en un bucle PHP básico de 1 a 5 ir ejecutando sentencias `array_pop()` y mostrar como queda el array en cada iteración

```

1  <?php
2  // Primer bucle: Agregar números del 1 al 10 al array
3  $array = [];
4
5  for ($i = 1; $i <= 10; $i++) {
6      $array[] = $i; // Agregar número al array
7      echo "Contenido del array en la iteración $i: ";
8      print_r($array); // Mostrar contenido del array
9  }
10
11 echo "\n";
12
13 // Segundo bucle: Ejecutar array_pop() de 1 a 5
14 for ($j = 1; $j <= 5; $j++) {
15     array_pop($array); // Eliminar el último elemento del array
16     echo "Contenido del array después de la iteración $j: ";
17     print_r($array); // Mostrar contenido del array
18 }
19 ?>
20

```

Contenido del array en la iteración 1: Array ([0] => 1) Contenido del array en la iteración 2: Array ([0] => 1, [1] => 2) Contenido del array en la iteración 3: Array ([0] => 1, [1] => 2, [2] => 3) Contenido del array en la iteración 4: Array ([0] => 1, [1] => 2, [2] => 3, [3] => 4) Contenido del array en la iteración 5: Array ([0] => 1, [1] => 2, [2] => 3, [3] => 4, [4] => 5) Contenido del array en la iteración 6: Array ([0] => 1, [1] => 2, [2] => 3, [3] => 4, [4] => 5, [5] => 6) Contenido del array en la iteración 7: Array ([0] => 1, [1] => 2, [2] => 3, [3] => 4, [4] => 5, [5] => 6, [6] => 7) Contenido del array en la iteración 8: Array ([0] => 1, [1] => 2, [2] => 3, [3] => 4, [4] => 5, [5] => 6, [6] => 7, [7] => 8) Contenido del array en la iteración 9: Array ([0] => 1, [1] => 2, [2] => 3, [3] => 4, [4] => 5, [5] => 6, [6] => 7, [7] => 8, [8] => 9) Contenido del array en la iteración 10: Array ([0] => 1, [1] => 2, [2] => 3, [3] => 4, [4] => 5, [5] => 6, [6] => 7, [7] => 8, [8] => 9, [9] => 10) Contenido del array después de la iteración 1: Array ([0] => 1, [1] => 2, [2] => 3, [3] => 4, [4] => 5, [5] => 6, [6] => 7, [7] => 8, [8] => 9) Contenido del array después de la iteración 2: Array ([0] => 1, [1] => 2, [2] => 3, [3] => 4, [4] => 5, [5] => 6, [6] => 7, [7] => 8) Contenido del array después de la iteración 3: Array ([0] => 1, [1] => 2, [2] => 3, [3] => 4, [4] => 5, [5] => 6, [6] => 7) Contenido del array después de la iteración 4: Array ([0] => 1, [1] => 2, [2] => 3, [3] => 4, [4] => 5) Contenido del array después de la iteración 5: Array ([0] => 1, [1] => 2, [2] => 3, [3] => 4)

Explicación:

Primer bucle:

Se inicializa un array vacío `$array`.

Se usa un bucle for que va de 1 a 10.

En cada iteración, se agrega el número actual al array y se muestra su contenido.

Segundo bucle:

Se usa otro bucle for que va de 1 a 5.

En cada iteración, se elimina el último elemento del array usando `array_pop()` y se muestra el contenido actualizado del array.

Practica 22

Crear un script que por medio de un bucle for que vaya de 1 a 10 agregue esos números en un array mediante `array_unshift()`. En cada iteración mostrar el contenido del array. Después en un bucle for de 1 a 5 ir ejecutando sentencias `array_shift()` y mostrar como queda el array en cada iteración

```

1 > practica22 > practica22.php
2 <?php
3 // Primer bucle: Agregar números del 1 al 10 al array usando array_unshift
4 $array = [];
5
6 for ($i = 1; $i <= 10; $i++) {
7     array_unshift($array, $i); // Agregar número al principio del array
8     echo "Contenido del array en la iteración $i: ";
9     print_r($array); // Mostrar contenido del array
10 }
11
12 echo "\n";
13
14 // Segundo bucle: Ejecutar array_shift() de 1 a 5
15 for ($j = 1; $j <= 5; $j++) {
16     array_shift($array); // Eliminar el primer elemento del array
17     echo "Contenido del array después de la iteración $j: ";
18     print_r($array); // Mostrar contenido del array
19 }
20 ?>

```

Contenido del array en la iteración 1: Array ([0] => 1) Contenido del array en la iteración 2: Array ([0] => 2 [1] => 1) Contenido del array en la iteración 3: Array ([0] => 3 [1] => 2 [2] => 1) Contenido del array en la iteración 4: Array ([0] => 4 [1] => 3 [2] => 2 [3] => 1) Contenido del array en la iteración 5: Array ([0] => 5 [1] => 4 [2] => 3 [3] => 2 [4] => 1) Contenido del array en la iteración 6: Array ([0] => 6 [1] => 5 [2] => 4 [3] => 3 [4] => 2 [5] => 1) Contenido del array en la iteración 7: Array ([0] => 7 [1] => 6 [2] => 5 [3] => 4 [4] => 3 [5] => 2 [6] => 1) Contenido del array en la iteración 8: Array ([0] => 8 [1] => 7 [2] => 6 [3] => 5 [4] => 4 [5] => 3 [6] => 2 [7] => 1) Contenido del array en la iteración 9: Array ([0] => 9 [1] => 8 [2] => 7 [3] => 6 [4] => 5 [5] => 4 [6] => 3 [7] => 2 [8] => 1) Contenido del array en la iteración 10: Array ([0] => 10 [1] => 9 [2] => 8 [3] => 7 [4] => 6 [5] => 5 [6] => 4 [7] => 3 [8] => 2 [9] => 1) Contenido del array después de la iteración 1: Array ([0] => 9 [1] => 8 [2] => 7 [3] => 6 [4] => 5 [5] => 4 [6] => 3 [7] => 2 [8] => 1) Contenido del array después de la iteración 2: Array ([0] => 8 [1] => 7 [2] => 6 [3] => 5 [4] => 4 [5] => 3 [6] => 2 [7] => 1) Contenido del array después de la iteración 3: Array ([0] => 7 [1] => 6 [2] => 5 [3] => 4 [4] => 3 [5] => 2 [6] => 1) Contenido del array después de la iteración 4: Array ([0] => 6 [1] => 5 [2] => 4 [3] => 3 [4] => 2 [5] => 1) Contenido del array después de la iteración 5: Array ([0] => 5 [1] => 4 [2] => 3 [3] => 2 [4] => 1)

Explicación:

Primer bucle:

Se inicializa un array vacío `$array`.

Se usa un bucle for que va de 1 a 10.

En cada iteración, se agrega el número actual al principio del array usando `array_unshift()`, y se muestra su contenido.

Segundo bucle:

Se usa otro bucle for que va de 1 a 5.

En cada iteración, se elimina el primer elemento del array usando `array_shift()` y se muestra el contenido actualizado del array.

Practica 23

Haz una página PHP que utilice `foreach` para mostrar todos los valores del array `$_SERVER` en una tabla con dos columnas. La primera columna debe contener el nombre de la variable, y la segunda su valor

```
practica19.php M  practica20.php U  practica21.php U  practica22.php U  practica23.php U x
UT1 > practica23 > practica23.php
1  <!DOCTYPE html>
2  <html Lang="es">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Valores de $_SERVER</title>
7  </head>
8  <body>
9      <h1>Valores de $_SERVER</h1>
10     <table border="1">
11         <thead>
12             <tr>
13                 <th>Nombre de la Variable</th>
14                 <th>Valor</th>
15             </tr>
16         </thead>
17         <tbody>
18             <?php
19                 // Iterar sobre el array $_SERVER
20                 foreach ($_SERVER as $variable => $valor) {
21                     echo "<tr>";
22                     echo "<td>" . htmlspecialchars($variable) . "</td>";
23                     echo "<td>" . htmlspecialchars($valor) . "</td>";
24                     echo "</tr>";
25                 }
26             ?>
27         </tbody>
28     </table>
29 </body>
30 </html>
31
```

localhost:3000/UT1/practica23/practica23.php

Aplicaciones Cursos Tablero personal Code Estudio Personal Streaming Curro Gmail YouTube Maps Noticias Traducir GFNOW Cursos en

Valores de \$_SERVER

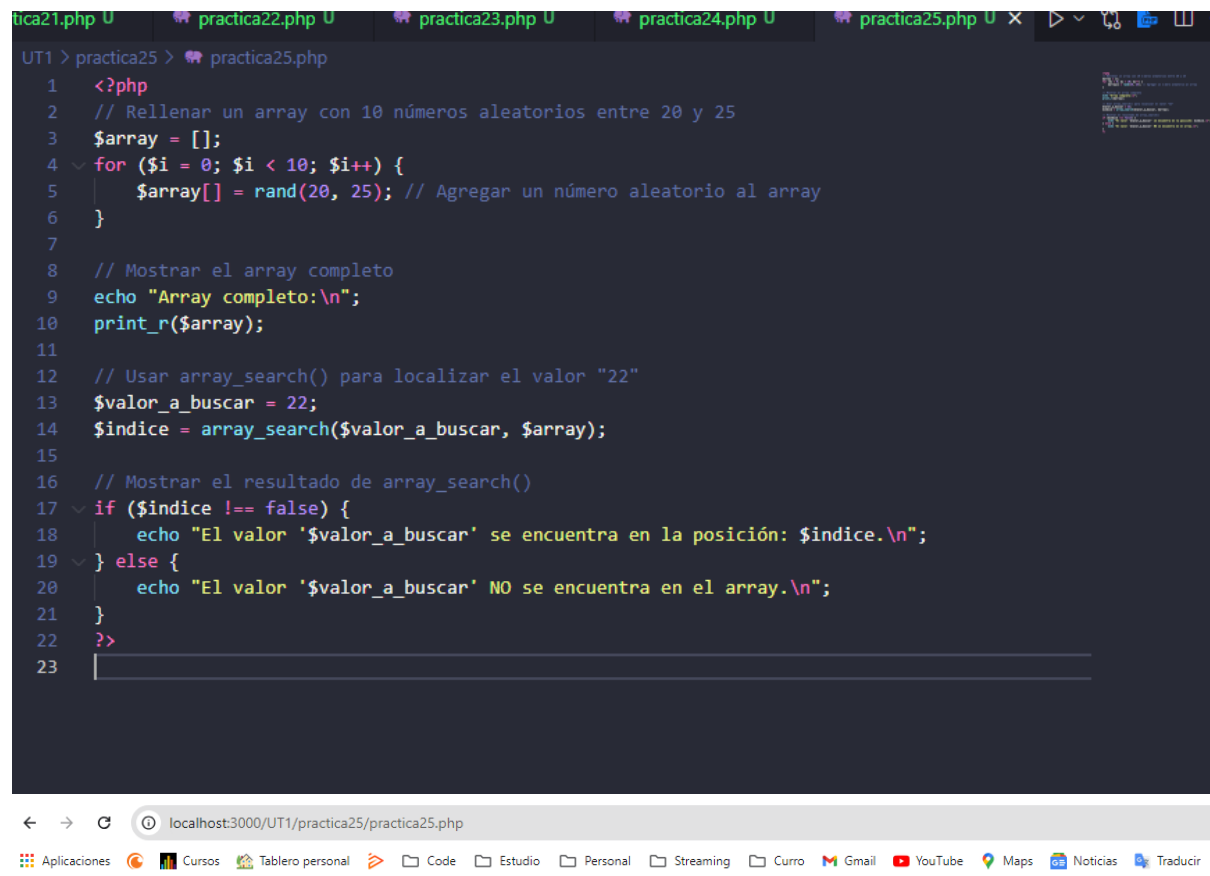
Nombre de la Variable	Valor
DOCUMENT_ROOT	C:\Users\melru\Documents\GitHub\aed
REMOTE_ADDR	::1
REMOTE_PORT	60118
SERVER_SOFTWARE	PHP 8.2.12 Development Server
SERVER_PROTOCOL	HTTP/1.1
SERVER_NAME	localhost
SERVER_PORT	3000
REQUEST_URI	/UT1/practica23/practica23.php
REQUEST_METHOD	GET
SCRIPT_NAME	/UT1/practica23/practica23.php
SCRIPT_FILENAME	C:\Users\melru\Documents\GitHub\aed\UT1\practica23\practica23.php
PHP_SELF	/UT1/practica23/practica23.php
HTTP_HOST	localhost:3000
HTTP_CONNECTION	keep-alive
HTTP_SEC_CH-UA	"Google Chrome";v="129", "Not=A?Brand";v="8", "Chromium";v="129"
HTTP_SEC_CH-UA-MOBILE	?0
HTTP_SEC_CH-UA-PLATFORM	"Windows"
HTTP_UPGRADE_INSECURE_REQUESTS	1
HTTP_USER_AGENT	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Safari/537.36
HTTP_ACCEPT	text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
HTTP_SEC_FETCH_SITE	none
HTTP_SEC_FETCH_MODE	navigate
HTTP_SEC_FETCH_USER	?1
HTTP_SEC_FETCH_DEST	document
HTTP_ACCEPT_ENCODING	gzip, deflate, br, zstd
HTTP_ACCEPT_LANGUAGE	es-ES,es;q=0.9,en-GB;q=0.8,en-XA;q=0.7,en;q=0.6
HTTP_COOKIE	Idea-a1ff3ec=f51cca95-0a8f-4505-a18c-bd2dc237b8ad; PHPSESSID=fgqe5jph0en2o6tmck9jbh8911
REQUEST_TIME_FLOAT	1727298472.6865
REQUEST_TIME	1727298472

Practica 24

1. Definición del Array: Se crea un array con colores.
2. Mostrar el Array Original: Se imprime el array inicial.
3. Uso de unset(): Se eliminan los elementos en las posiciones 2 y 3 ('verde' y 'amarillo').
4. Mostrar el Array Después de unset(): Se imprime el array modificado.
5. Uso de array_values(): Se limpia el array para que sus índices sean consecutivos (0, 1, 2, ...).
6. Buscar Elementos:
 - in_array(): Se verifica si un color está en el array.
 - array_search(): Se busca la posición de un color específico.

Practica 25

Rellenar un array con 10 números aleatorios entre 20 y 25 (hacer uso de: rand (int \$min , int \$max) : int) y luego hacer uso de array_search() para localizar el valor: “22” Se debe mostrar en pantalla el array completo y el valor devuelto por array_search()



```
UT1 > practica25 > practica25.php
1  <?php
2  // Rellenar un array con 10 números aleatorios entre 20 y 25
3  $array = [];
4  for ($i = 0; $i < 10; $i++) {
5      $array[] = rand(20, 25); // Agregar un número aleatorio al array
6  }
7
8  // Mostrar el array completo
9  echo "Array completo:\n";
10 print_r($array);
11
12 // Usar array_search() para localizar el valor "22"
13 $valor_a_buscar = 22;
14 $indice = array_search($valor_a_buscar, $array);
15
16 // Mostrar el resultado de array_search()
17 if ($indice !== false) {
18     echo "El valor '$valor_a_buscar' se encuentra en la posición: $indice.\n";
19 } else {
20     echo "El valor '$valor_a_buscar' NO se encuentra en el array.\n";
21 }
22 ?>
23
```

Array completo: Array ([0] => 24 [1] => 23 [2] => 24 [3] => 25 [4] => 22 [5] => 25 [6] => 25 [7] => 23 [8] => 23 [9] => 20) El valor '22' se encuentra en la posición: 4.

Explicación del código:

1. Inicialización del Array: Se crea un array vacío \$array.
2. Relleno del Array:
 - Un bucle for se utiliza para llenar el array con 10 números aleatorios entre 20 y 25, utilizando rand(20, 25).
3. Mostrar el Array: Se imprime el array completo con print_r().
4. Buscar el Valor "22":
 - Se utiliza array_search() para buscar el índice del valor "22".
5. Mostrar Resultado:
 - Se verifica si array_search() devuelve un índice válido y se muestra el resultado correspondiente.

Practica 26

Variar el ejemplo anterior para que se haga uso del operador nave espacial: <=>

```
1 > practica26 > practica27.php > ...
1  <?php
2  // Función de comparación utilizando el operador nave espacial
   Tabnine | Edit | Test | Explain | Document | Ask
3  function cmp($a, $b) {
4      return $a <=> $b; // Retorna -1, 0 o 1
5  }
6
7  // Array de números
8  $a = array(3, 2, 5, 6, 1);
9
10 // Ordenar el array utilizando la función de comparación
11 usort($a, "cmp");
12
13 // Mostrar el array ordenado
14 foreach ($a as $valor) {
15     echo "$valor, ";
16 }
17 ?>
18
```

← → ↻ ⓘ localhost:3000/UT1/practica26/practica26.php

📱 Aplicaciones 🌀 Cursos 🏠 Tablero personal 📄 Code 📁 E

1, 2, 3, 5, 6,

Explicación del código:

Función de Comparación:

La función cmp toma dos parámetros, \$a y \$b, y utiliza el operador nave espacial (<=>) para comparar los dos valores.

El operador devuelve:

-1 si \$a es menor que \$b.

0 si son iguales.

1 si \$a es mayor que \$b.

Definición del Array:

Se crea un array \$a con algunos números desordenados.

Ordenar el Array:

Se utiliza usort() para ordenar el array \$a según la función de comparación cmp.

Mostrar el Array Ordenado:

Se recorre el array ordenado con un bucle foreach y se imprimen los valores.

Practica 27

Crear un array con los valores: [7,2,8,1,9,4] Hacer búsqueda con array_search() de: 4
Ordenar el array con usort y repetir la búsqueda de: 4 Mostrar los array antes y después de ordenación así como lo que devuelve array_search()

```
UT1 > practica27 > practica27.php > ...
1  <?php
2  // Crear el array con los valores iniciales
3  $array = [7, 2, 8, 1, 9, 4];
4
5  // Mostrar el array original
6  echo "Array original:\n";
7  print_r($array);
8
9  // Búsqueda del valor 4 antes de ordenar
10 $valor_a_buscar = 4;
11 $indice = array_search($valor_a_buscar, $array);
12 echo "Resultado de array_search() para el valor $valor_a_buscar antes de ordenar: ";
13 if ($indice !== false) {
14     echo "El valor '$valor_a_buscar' se encuentra en la posición: $indice.\n";
15 } else {
16     echo "El valor '$valor_a_buscar' NO se encuentra en el array.\n";
17 }
18
19 // Función de comparación para usort
20 function cmp($a, $b) {
21     return $a <=> $b; // Usar el operador nave espacial
22 }
23
24 // Ordenar el array utilizando usort
25 usort($array, "cmp");
26
27 // Mostrar el array después de ordenar
28 echo "\nArray después de ordenar:\n";
29 print_r($array);
30
31 // Repetir búsqueda del valor 4 después de ordenar
32 $indice = array_search($valor_a_buscar, $array);
33 echo "Resultado de array_search() para el valor $valor_a_buscar después de ordenar: ";
34 if ($indice !== false) {
35     echo "El valor '$valor_a_buscar' se encuentra en la posición: $indice.\n";
36 } else {
37     echo "El valor '$valor_a_buscar' NO se encuentra en el array.\n";
38 }
39 ?>
40
```

Array original: Array ([0] => 7 [1] => 2 [2] => 8 [3] => 1 [4] => 9 [5] => 4) Resultado de array_search() para el valor 4 antes de ordenar: El valor '4' se encuentra en la posición: 5. Array después de ordenar: Array ([0] => 1 [1] => 2 [2] => 4 [3] => 7 [4] => 8 [5] => 9) Resultado de array_search() para el valor 4 después de ordenar: El valor '4' se encuentra en la posición: 2.

Explicación del código:

1. Crear el Array:
 - Se inicializa el array `$array` con los valores `[7, 2, 8, 1, 9, 4]`.
2. Mostrar el Array Original:
 - Se imprime el array original.
3. Buscar el Valor 4:
 - Se utiliza `array_search()` para buscar el índice del valor 4 en el array y se muestra el resultado.
4. Ordenar el Array:

- Se define una función de comparación `cmp` que utiliza el operador nave espacial (`<=>`) para comparar los elementos.
 - Se usa `usort()` para ordenar el array.
5. Mostrar el Array Ordenado:
- Se imprime el array después de la ordenación.
6. Repetir la Búsqueda del Valor 4:
- Se vuelve a usar `array_search()` para buscar el índice del valor 4 en el array ordenado y se muestra el resultado.

Practica 28

Modifica el código anterior y quita el valor por defecto del parámetro `$print`. Ejecuta el programa y toma captura de pantalla de los mensajes del IDE y responde: ¿ se obtiene resultado o se detiene el programa ?


```
UT1 > practica28 > practica28.php > ...
1  <?php
   Tabnine | Edit | Test | Explain | Document | Ask
2  function sumar($a, $b, $print): float
3  {
4      $suma = $a + $b;
5      if ($print) {
6          echo "resultado suma: $suma <br>";
7      }
8      return $suma;
9  }
10
11 // Llamadas a la función
12 $sum1 = sumar(1, 2, false); // Sin imprimir
13 $sum2 = sumar(4, 5, true);  // Con imprimir
14
15 echo "las operaciones para sum1 y sum2 dan: $sum1 , $sum2";
16 ?>
17
```

← → ↻ ⓘ localhost:3000/UT1/practica28/practica28.php

📱 Aplicaciones 🔄 Cursos 🏠 Tablero personal 📁 Code 📁 Estudio

resultado suma: 9
las operaciones para sum1 y sum2 dan: 3 , 9

Se elimina el valor por defecto del parámetro `$print` en la función `sumar`. Esto obligará a que se pase un argumento cada vez que se llame a la función.

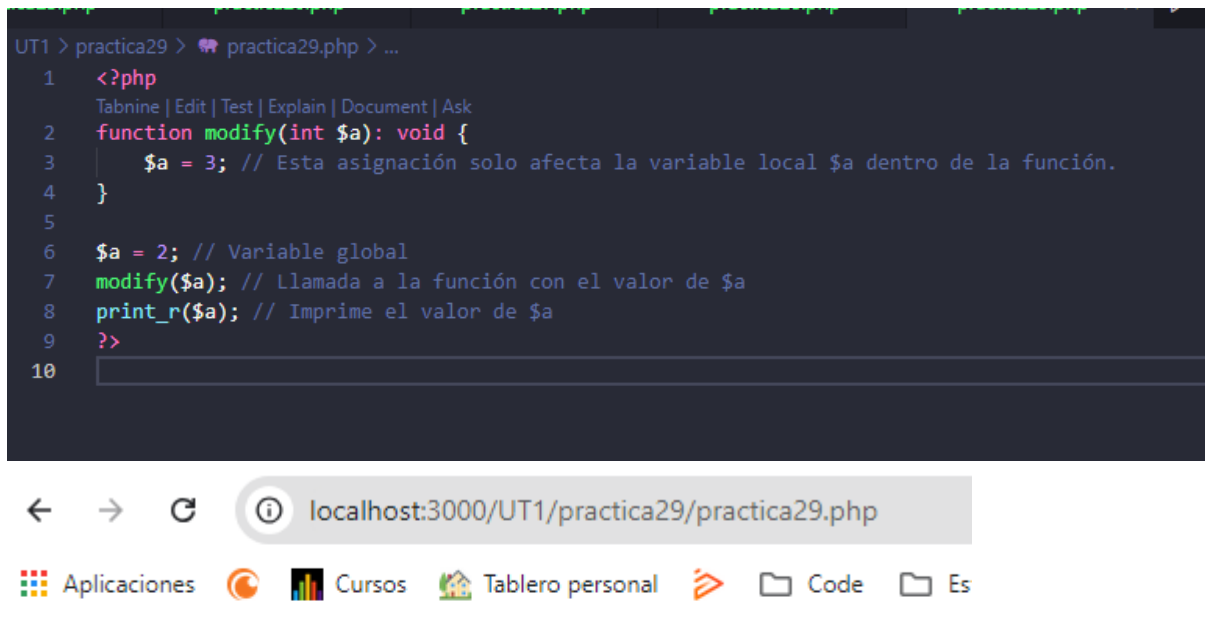
¿Se obtiene resultado o se detiene el programa?:

Si llamas a la función sin proporcionar un valor para `$print`, el programa se detendrá y mostrará un error, indicando que se han pasado menos argumentos de los que se esperaban.

Si todas las llamadas son correctas y se proporciona el argumento, entonces obtendrás el resultado esperado sin errores.

Practica 29

Probar el código anterior. Observamos que no se ha modificado el valor de la variable después de la ejecución de la función Así que ¿ estamos en un caso de paso por valor o por referencia ? Tomar captura de pantalla



The screenshot shows a code editor with the following PHP code:

```
1 <?php
2 function modify(int $a): void {
3     $a = 3; // Esta asignación solo afecta la variable local $a dentro de la función.
4 }
5
6 $a = 2; // Variable global
7 modify($a); // Llamada a la función con el valor de $a
8 print_r($a); // Imprime el valor de $a
9 ?>
10
```

Below the code editor, a browser window shows the output of the script. The address bar indicates the file is located at `localhost:3000/UT1/practica29/practica29.php`. The output of `print_r($a)` is displayed as the number `2`.

Explicación

1. Paso de Parámetros por Valor:

- En PHP, los parámetros se pasan por valor de manera predeterminada. Esto significa que se pasa una copia del valor de la variable a la función, no la variable en sí.
- Cuando llamas a `modify($a)`, el valor de `$a` (que es 2) se copia en el parámetro `$a` de la función `modify`.

2. Modificación Local:

- Dentro de la función `modify`, la variable `$a` se establece en 3, pero esto solo modifica la copia local de `$a`, no la variable original que se encuentra fuera de la función.
- Por lo tanto, después de la ejecución de la función, la variable `$a` que se encuentra fuera de la función sigue siendo 2.

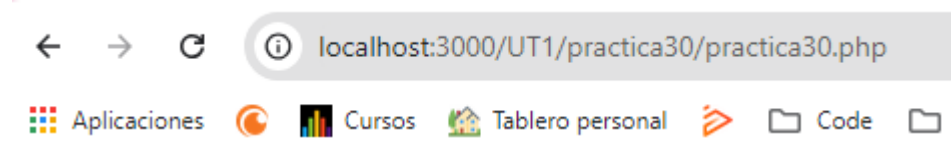
3. Salida:

- Cuando ejecutas `print_r($a);`, se imprime el valor de la variable global `$a`, que sigue siendo 2.

Practica 30

Ejecutar el ejemplo y observar que ahora la variable sí se ve modificada. Tomar captura de pantalla

```
UT1 > practica30 > practica30.php > ...
1  <?php
   Tabnine | Edit | Test | Explain | Document | Ask
2  function modify(int &$a): void { // Parámetro pasado por referencia
3      $a = 3; // Modifica la variable original
4  }
5
6  $a = 2; // Inicializa la variable
7  modify($a); // Llama a la función que modifica $a
8  print_r($a); // Imprime el valor de $a
9  ?>
10
```



Explicación del Código

1. Parámetro por Referencia:

- La función `modify` recibe un parámetro `$a` como referencia (usando `&`). Esto significa que cualquier cambio realizado en `$a` dentro de la función afectará la variable original fuera de la función.

2. Modificación de la Variable:

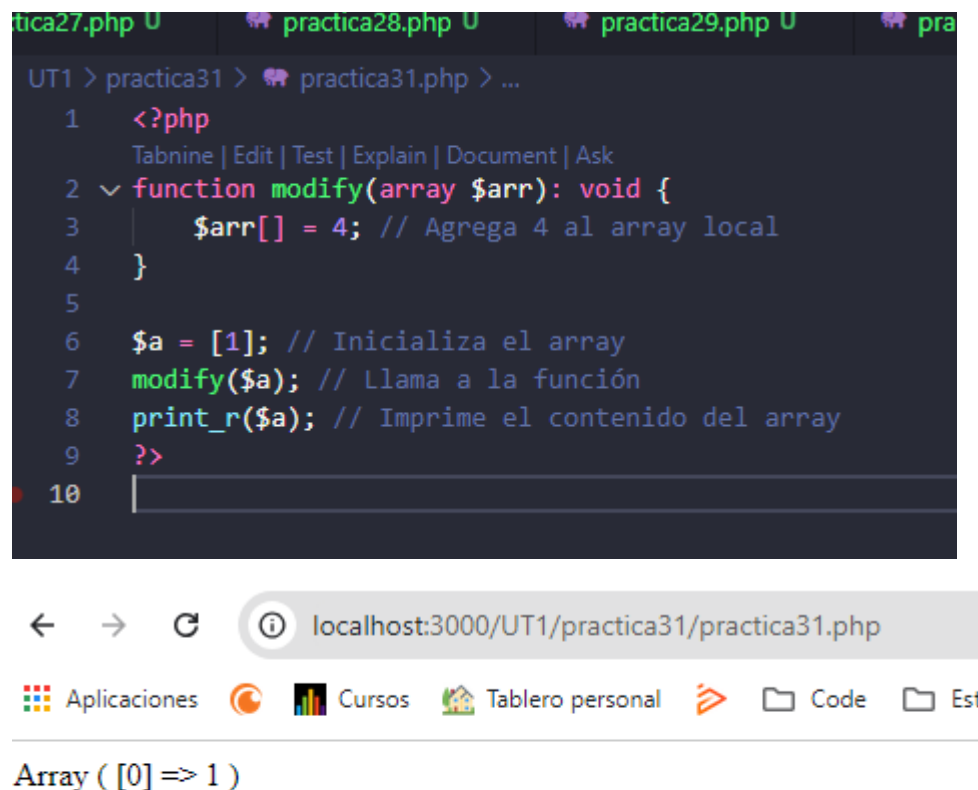
- Dentro de la función, se asigna el valor 3 a `$a`, lo que también modifica la variable `$a` que se definió fuera de la función.

3. Salida Esperada:

- Al ejecutar `print_r($a);`, verás que el valor de la variable es 3, ya que ha sido modificada por la función.

Practica 31

Hacer lo anterior, comprobar el resultado. Ahora debiera mostrar todos los datos del array. Tomar captura de pantalla.



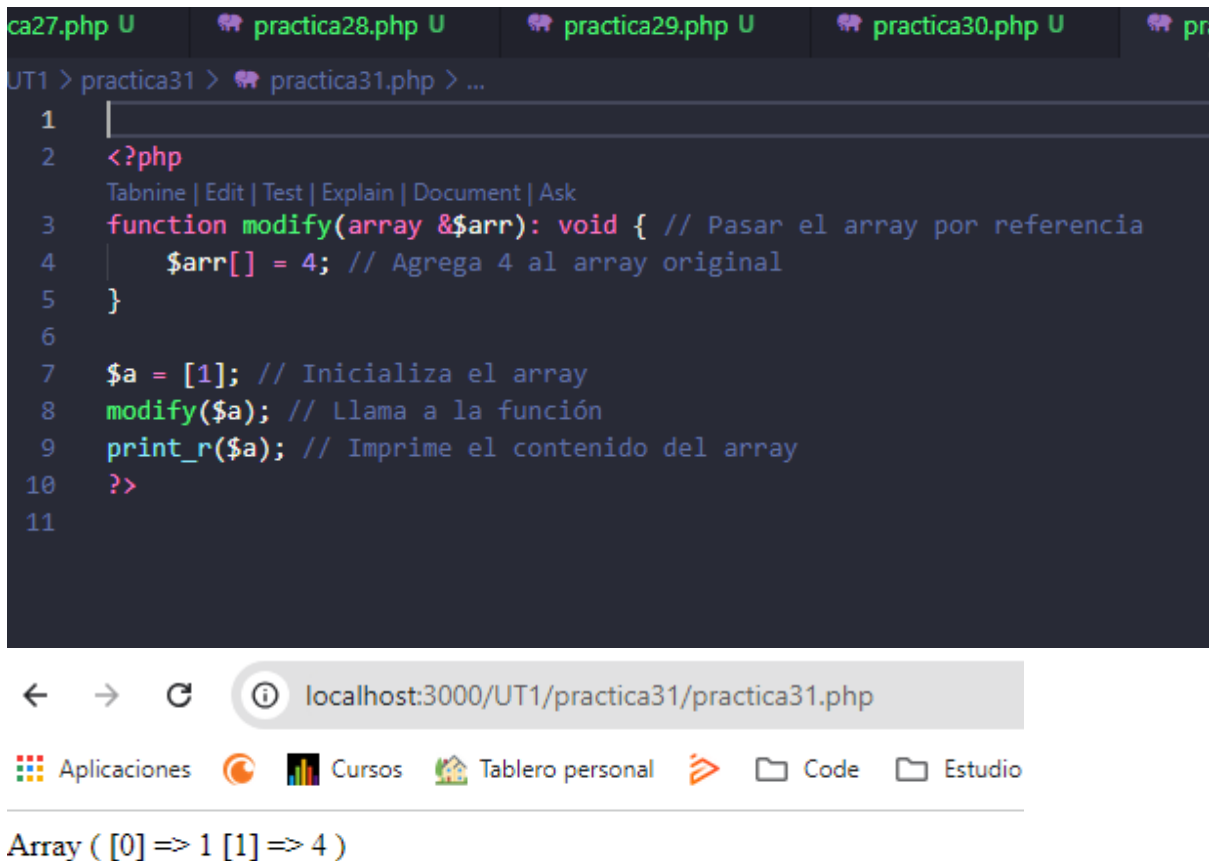
The screenshot shows a code editor with a dark theme. The editor has several tabs at the top: 'practica27.php U', 'practica28.php U', 'practica29.php U', and 'pra'. The active tab is 'practica31.php U'. The code in the editor is as follows:

```
UT1 > practica31 > practica31.php > ...
1  <?php
   Tabnine | Edit | Test | Explain | Document | Ask
2  function modify(array $arr): void {
3      $arr[] = 4; // Agrega 4 al array local
4  }
5
6  $a = [1]; // Inicializa el array
7  modify($a); // Llama a la función
8  print_r($a); // Imprime el contenido del array
9  ?>
10
```

Below the code editor, a browser window is visible. The address bar shows 'localhost:3000/UT1/practica31/practica31.php'. The browser's taskbar at the bottom includes icons for 'Aplicaciones', 'Cursos', 'Tablero personal', 'Code', and 'Est'. The browser's output area displays the result of the `print_r($a);` statement:

```
Array ( [0] => 1 )
```

Esto ocurre porque el array `$arr` se pasa por valor, por lo que la modificación dentro de `modify` no afecta al array `$a` original.



```
ca27.php U | practica28.php U | practica29.php U | practica30.php U | pr
UT1 > practica31 > practica31.php > ...
1
2 <?php
   Tabnine | Edit | Test | Explain | Document | Ask
3 function modify(array &$arr): void { // Pasar el array por referencia
4     $arr[] = 4; // Agrega 4 al array original
5 }
6
7 $a = [1]; // Inicializa el array
8 modify($a); // Llama a la función
9 print_r($a); // Imprime el contenido del array
10 ?>
11

← → ↻ ⓘ localhost:3000/UT1/practica31/practica31.php
Aplicaciones Cursos Tablero personal Code Estudio

Array ( [0] => 1 [1] => 4 )
```

En este caso, se mostrará 1 y 4 porque la función modify ahora modifica el array original

Practica 32

Primera Línea de Echo:

- Cuando ejecutas test.php, la primera línea echo "Una \$fruta \$color"; intentará imprimir una cadena que utiliza las variables \$fruta y \$color, pero estas variables no están definidas todavía, así que no se mostrará nada relevante.

Error de Require:

- Al intentar ejecutar require 'vars1.php';, se generará un error porque vars1.php no existe. Esto detendrá la ejecución del script y mostrará un mensaje de error similar a:

Fatal error: Uncaught Error: Failed opening required 'vars1.php'...

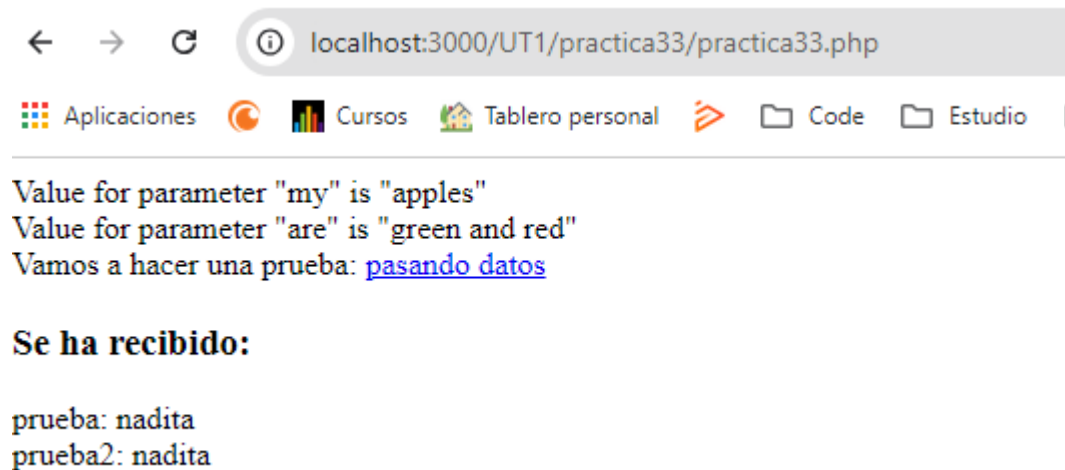
No se Ejecuta la Segunda Línea:

- La segunda línea echo "Una \$fruta \$color"; no se ejecutará debido al error anterior.

Practica 33

Hacer lo anterior, pero se debe comprobar la diferencia de pasar el texto con urlencode y sin urlencode. Así que se propone poner dos parámetros: prueba y prueba2 uno de ellos con urlencode y el otro sin él pasando en ambos casos el mismo texto en el value. Tomar captura de pantalla de lo obtenido

```
UT1 > practica33 > 🐞 practica33.php
1  <?php
2  $query = "my=apples&are=green+and+red";
3  foreach (explode('&', $query) as $chunk) {
4      $param = explode("=", $chunk);
5      if ($param) {
6          printf("Value for parameter \"%s\" is \"%s\"<br/>\n", urldecode($param[0]), urldecod
7      }
8  }
9  ?>
10
11 Vamos a hacer una prueba:
12 <?php
13 // Definimos el texto a pasar
14 $text = "Pasando datos diría.. que hay que usar urlencode";
15
16 // Creación de enlaces con parámetros
17 echo "<a href='index.php?prueba=$text&prueba2=" . urlencode($text) . "'>pasando datos</a>";
18
19 // Recibir y mostrar los parámetros
20 $recibido1 = $_GET["prueba"] ?? "nadita"; // Parámetro sin urlencode
21 $recibido2 = $_GET["prueba2"] ?? "nadita"; // Parámetro con urlencode
22
23 echo "<h3>Se ha recibido:</h3>";
24 echo "prueba: " . $recibido1 . "<br>";
25 echo "prueba2: " . $recibido2 . "<br>";
26 ?>
27
```



← → ↻ ⓘ localhost:3000/UT1/practica33/practica33.php

Aplicaciones Cursos Tablero personal Code Estudio

Value for parameter "my" is "apples"
Value for parameter "are" is "green and red"
Vamos a hacer una prueba: [pasando datos](#)

Se ha recibido:

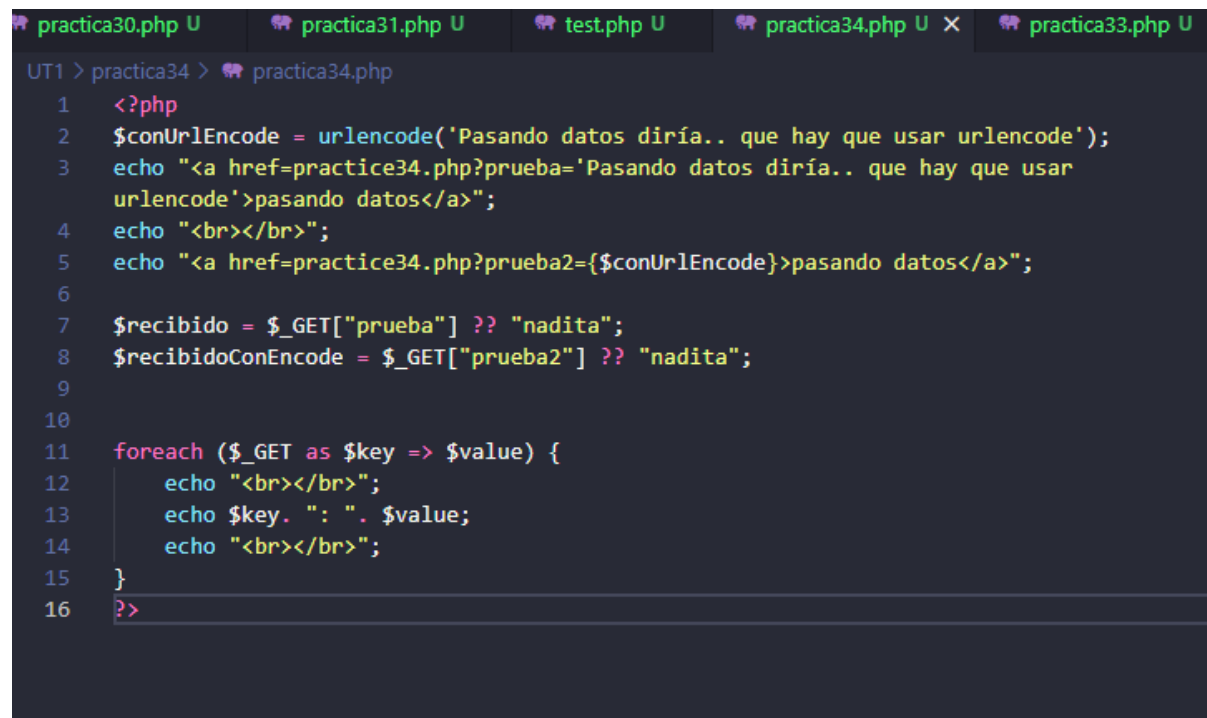
prueba: nadita
prueba2: nadita

NOTA: Si no se usa urlencode, caracteres especiales como espacios se reemplazan por + y pueden causar problemas al recibir los datos.

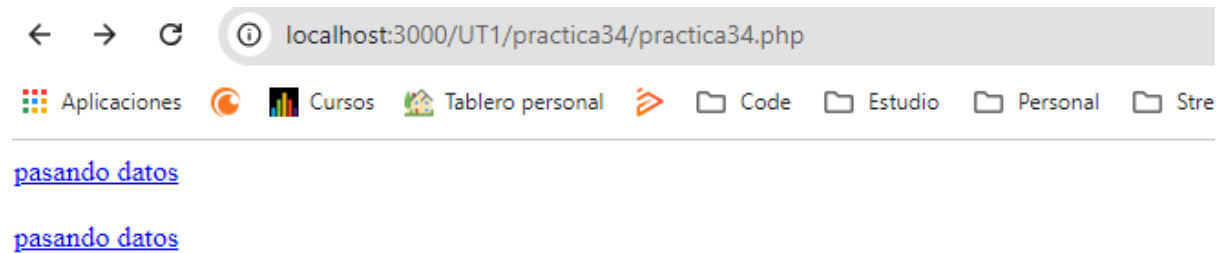
urlencode asegura que todos los caracteres sean seguros para una URL, evitando problemas de interpretación.

Practica 34

recorrer el array \$_GET con un foreach y mostrar el conjunto de clave valor para la actividad anterior



```
practica30.php U practica31.php U test.php U practica34.php U X practica33.php U
UT1 > practica34 > practica34.php
1  <?php
2  $conUrlEncode = urlencode('Pasando datos diría.. que hay que usar urlencode');
3  echo "<a href=practice34.php?prueba='Pasando datos diría.. que hay que usar
4  urlencode'>pasando datos</a>";
5  echo "<br></br>";
6  echo "<a href=practice34.php?prueba2={$conUrlEncode}>pasando datos</a>";
7
8  $recibido = $_GET["prueba"] ?? "nadita";
9  $recibidoConEncode = $_GET["prueba2"] ?? "nadita";
10
11 foreach ($_GET as $key => $value) {
12     echo "<br></br>";
13     echo $key. ": ". $value;
14     echo "<br></br>";
15 }
16 ?>
```



Practica 35

Realiza una página con un formulario que se llame a si misma para mostrar la tabla de un número introducido por el usuario. Se deberá controlar que el usuario haya introducido un número entero positivo. Hacer uso para ello de la función: `is_int()` buscando su funcionamiento en el manual oficial: [php.net](https://www.php.net)

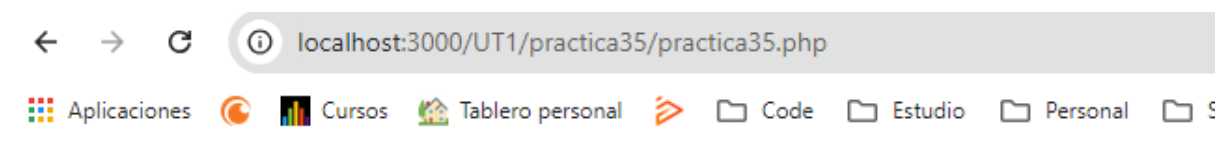


Tabla de Multiplicar

Introduce un número entero positivo:

Tabla de multiplicar del 6

Multiplicación	Resultado
6 x 1	6
6 x 2	12
6 x 3	18
6 x 4	24
6 x 5	30
6 x 6	36
6 x 7	42
6 x 8	48
6 x 9	54
6 x 10	60


```
practica29.php U practica30.php U practica31.php U test.php U practica34.php U practica35.php U x
UT1 > practica35 > practica35.php
1  <?php
2  // Inicializar variables
3  $numero = null;
4  $error = "";
5
6  // Verificar si se ha enviado el formulario
7  if ($_SERVER["REQUEST_METHOD"] == "POST") {
8      // Obtener el número ingresado
9      $numero = $_POST['numero'];
10
11     // Validar que el número sea un entero positivo
12     if (filter_var($numero, FILTER_VALIDATE_INT) !== false && $numero > 0) {
13         // Es un entero positivo
14         $numero = intval($numero);
15     } else {
16         // Si no es válido, establecer un mensaje de error
17         $error = "Por favor, introduce un número entero positivo.";
18     }
19 }
20 ?>
21
22 <!DOCTYPE html>
23 <html lang="es">
24 <head>
25     <meta charset="UTF-8">
26     <meta name="viewport" content="width=device-width, initial-scale=1.0">
27     <title>Tabla de Multiplicar</title>
28 </head>
29 <body>
30     <h1>Tabla de Multiplicar</h1>
31     <form method="post" action="">
32         <label for="numero">Introduce un número entero positivo:</label>
33         <input type="number" id="numero" name="numero" required>
34         <button type="submit">Mostrar Tabla</button>
35     </form>
36
37     <?php if ($error): ?>
38         <p style="color: red;"><?php echo $error; ?></p>
39     <?php elseif ($numero !== null): ?>
40         <h2>Tabla de multiplicar del <?php echo $numero; ?></h2>
41         <table border="1">
42             <tr>
43                 <th>Multiplicación</th>
44                 <th>Resultado</th>
45             </tr>
46             <?php for ($i = 1; $i <= 10; $i++): ?>
47                 <tr>
48                     <td><?php echo "$numero x $i"; ?></td>
49                     <td><?php echo $numero * $i; ?></td>
50                 </tr>
51             <?php endfor; ?>
52         </table>
53     <?php endif; ?>
54 </body>
55 </html>
```

Explicación del Código

1. Validación del Número:

- Se utiliza `filter_var($numero, FILTER_VALIDATE_INT)` para verificar si el número ingresado es un entero.
- También se comprueba que el número sea mayor que 0 para asegurarse de que es positivo.

2. Formulario:

- Se crea un formulario que utiliza el método POST para enviar los datos. El formulario se llama a sí mismo a través de `action=""`.

3. Tabla de Multiplicar:

- Si el número ingresado es válido, se genera una tabla que muestra la multiplicación del número por los números del 1 al 10.

4. Manejo de Errores:

- Si el usuario ingresa un número inválido, se muestra un mensaje de error en rojo.

Practica 36

```
JT1 > practica36 > practica36.php
1  <?php
2  // Inicializar variables
3  $numeros = [];
4  $impares = [];
5  $pares = [];
6
7  // Verificar si se ha enviado el formulario
8  if ($_SERVER["REQUEST_METHOD"] == "POST") {
9      // Obtener la cadena de números
10     $cadena = trim($_POST['numeros']);
11
12     // Dividir la cadena en un array usando explode
13     $numeros = explode(" ", $cadena);
14
15     // Separar los números en impares y pares
16     foreach ($numeros as $numero) {
17         if (is_numeric($numero)) {
18             if ($numero % 2 == 0) {
19                 $pares[] = (int)$numero; // Convertir a entero
20             } else {
21                 $impares[] = (int)$numero; // Convertir a entero
22             }
23         }
24     }
25
26     // Ordenar los números impares y pares
27     usort($impares, fn($a, $b) => $a - $b); // Ordenar impares
28     usort($pares, fn($a, $b) => $a - $b); // Ordenar pares
29 }
30 ?>
31
32 <!DOCTYPE html>
33 <html lang="es">
```



Introducir Números

Introduce una cadena de números separados por espacios:

Números Impares:

- 3
- 5

Números Pares:

- 2
- 6
- 6

Explicación del Código

1. Formulario:

- Se crea un formulario que utiliza el método POST para enviar los datos. El formulario se llama a sí mismo a través de `action=""`.

2. Procesamiento de la Cadena:

- La cadena de números ingresada se procesa con `explode(" ", $cadena)` para convertirla en un array de números.
- Se utiliza un bucle `foreach` para separar los números en dos arrays: `$impares` y `$pares`.

3. Validación:

- Se verifica que cada entrada sea un número utilizando `is_numeric()` antes de agregarlo a los arrays correspondientes.

4. Ordenación:

- Se utiliza `usort()` con una función anónima (arrow function) para ordenar los números impares y pares de forma ascendente.

5. Mostrar Resultados:

- Se muestran los números impares primero y luego los pares, cada uno en una lista.

Practica 37

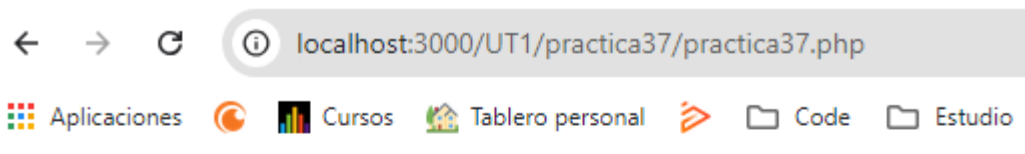
Realizar una página como la anterior que se valide a si misma. Obligando que el correo sea válido, que el nombre no sea vacío al igual que el género. Si los datos están correctamente introducidos se mostrarán por debajo de “Datos ingresados” si no superan la validación se dirá los campos que no la superan con texto en rojo

UT1 > practica37 > practica37.php

```
1  <?php
2  // Inicializar variables
3  $errores = [];
4
5  // Verificar si se ha enviado el formulario
6  if ($_SERVER["REQUEST_METHOD"] == "POST") {
7      // Obtener los datos del formulario
8      $nombre = trim($_POST['nombre']);
9      $correo = trim($_POST['correo']);
10     $genero = $_POST['genero'];
11
12     // Validar nombre
13     if (empty($nombre)) {
14         $errores[] = "El nombre no puede estar vacío.";
15     }
16
17     // Validar correo
18     if (!filter_var($correo, FILTER_VALIDATE_EMAIL)) {
19         $errores[] = "El correo no es válido.";
20     }
21
22     // Validar género
23     if (empty($genero)) {
24         $errores[] = "Debes seleccionar un género.";
25     }
26 }
27 ?>
```

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Formulario de Registro</title>
</head>
<body>
    <h1>Formulario de Registro</h1>
    <form method="post" action="">
        <label for="nombre">Nombre:</label><br>
        <input type="text" id="nombre" name="nombre" required><br><br>
        <label for="correo">Correo:</label><br>
        <input type="email" id="correo" name="correo" required><br><br>
        <label>Género:</label><br>
        <input type="radio" id="masculino" name="genero" value="masculino" required>
        <label for="masculino">Masculino</label><br>
        <input type="radio" id="femenino" name="genero" value="femenino" required>
        <label for="femenino">Femenino</label><br><br>
        <button type="submit">Enviar</button>
    </form>
```

```
<?php if ($_SERVER["REQUEST_METHOD"] == "POST"): ?>
    <?php if (empty($errores)): ?>
        <h2>Datos ingresados:</h2>
        <p><strong>Nombre:</strong> <?php echo htmlspecialchars($nombre); ?></p>
        <p><strong>Correo:</strong> <?php echo htmlspecialchars($correo); ?></p>
        <p><strong>Género:</strong> <?php echo htmlspecialchars($genero); ?></p>
    <?php else: ?>
        <h2 style="color: red;">Errores en los campos:</h2>
        <ul style="color: red;">
            <?php foreach ($errores as $error): ?>
                <li><?php echo $error; ?></li>
            <?php endforeach; ?>
        </ul>
    <?php endif; ?>
<?php endif; ?>
</body>
</html>
```



Formulario de Registro

Nombre:

Correo:

Género:

☐ Masculino
☒ Femenino

Datos ingresados:

Nombre: hola

Correo: msdklfsd@sjkf.com

Género: femenino

Explicación del Código

1. Formulario:

- Se crea un formulario con campos para el nombre, el correo electrónico y opciones de género (masculino y femenino).
- Se utiliza el método POST para enviar los datos y se llama a sí mismo a través de `action=""`.

2. Validación:

- Se verifica si el formulario se ha enviado y se recogen los datos.
- Se valida que el nombre no esté vacío, que el correo sea un correo electrónico válido (usando `filter_var()`), y que se haya seleccionado un género.
- Los errores se almacenan en un array `$errores`.

3. Mostrar Resultados:

- Si no hay errores, se muestran los datos ingresados.
- Si hay errores, se muestran los mensajes de error en rojo.