

## PHP Dossier

- [Práctica 01](#)
- [Práctica 02](#)
- [Práctica 03](#)
- [Práctica 04](#)
- [Práctica 05](#)
- [Práctica 06](#)
- [Práctica 07](#)
- [Práctica 08](#)
- [Práctica 09](#)
- [Práctica 10](#)
- [Práctica 11](#)
- [Práctica 12](#)
- [Práctica 13](#)
- [Práctica 14](#)
- [Práctica 15](#)
- [Práctica 16](#)
- [Práctica 17](#)
- [Práctica 18](#)
- [Práctica 19](#)
- [Práctica 20](#)
- [Práctica 21](#)
- [Práctica 22](#)
- [Práctica 23](#)
- [Práctica 24](#)
- [Práctica 25](#)
- [Práctica 26](#)
- [Práctica 27](#)
- [Práctica 28](#)
- [Práctica 29](#)
- [Práctica 30](#)
- [Práctica 31](#)
- [Práctica 32](#)
- [Práctica 33](#)
- [Práctica 34](#)
- [Práctica 35](#)
- [Práctica 36](#)
- [Práctica 37](#)

### Extras:

- [Formulario - Descomponer Número](#)
- [Formulario - Tablas](#)



### 📁 Sumar \$un\_str con \$un\_str2 ¿qué ocurre ?

No se nos permite concatenar cadenas de texto utilizando operadores como '+' al contrario que en Java.

- Captura:

```
26 /**
27  * Al contrario que en Java, no se nos permite concatenar dos strings utilizando el operador '+'
28  * */
29
30 $sumStr = $un_str+$un_str2;
31 echo "$sumStr";
32
33 ?>
```

booleanstring  
PHP Fatal error: Uncaught TypeError: Unsupported operand types: string + string in C:\Users\nabil\repositorios-git\aed-nabil\unit-01\php-dossier\practice02.php:29  
Stack trace:  
#0 {main}  
thrown in C:\Users\nabil\repositorios-git\aed-nabil\unit-01\php-dossier\practice02.php on line 29

### 📁 ¿ Se puede concatenar una cadena con comillas simples con una con comillas dobles ?

Gracias al operador '.' nos es posible concatenar cadenas de texto sin importar si estas son de comillas simples o dobles.

- Captura:

```
21
22 /**
23  * Utilizando '.' podemos concatenar cadenas independientemente de si son de comillas dobles o simples.
24  * */
25
26 echo ($un_str.$un_str2)
27
28 ?>
```

foofoo

## Práctica 03

### 📁 Realizar el código anterior y tomar captura de pantalla del resultado. ¿ qué es lo que ha ocurrido ?

Se nos muestra el parrafo del echo. La suma se visualiza gracias al uso del print en vez de a la declaración de la variable resultado.

- Captura:



localhost:3000/php-dossier/practice03.php

# la suma de uno más dos es: 3

📁 Poner código html antes de la declaración de strict\_types y probar de nuevo ¿ qué ocurre ahora ?

Tenemos un error, ya que las declaraciones de strict\_types deben ser lo primero en nuestro fichero php.

- Captura:

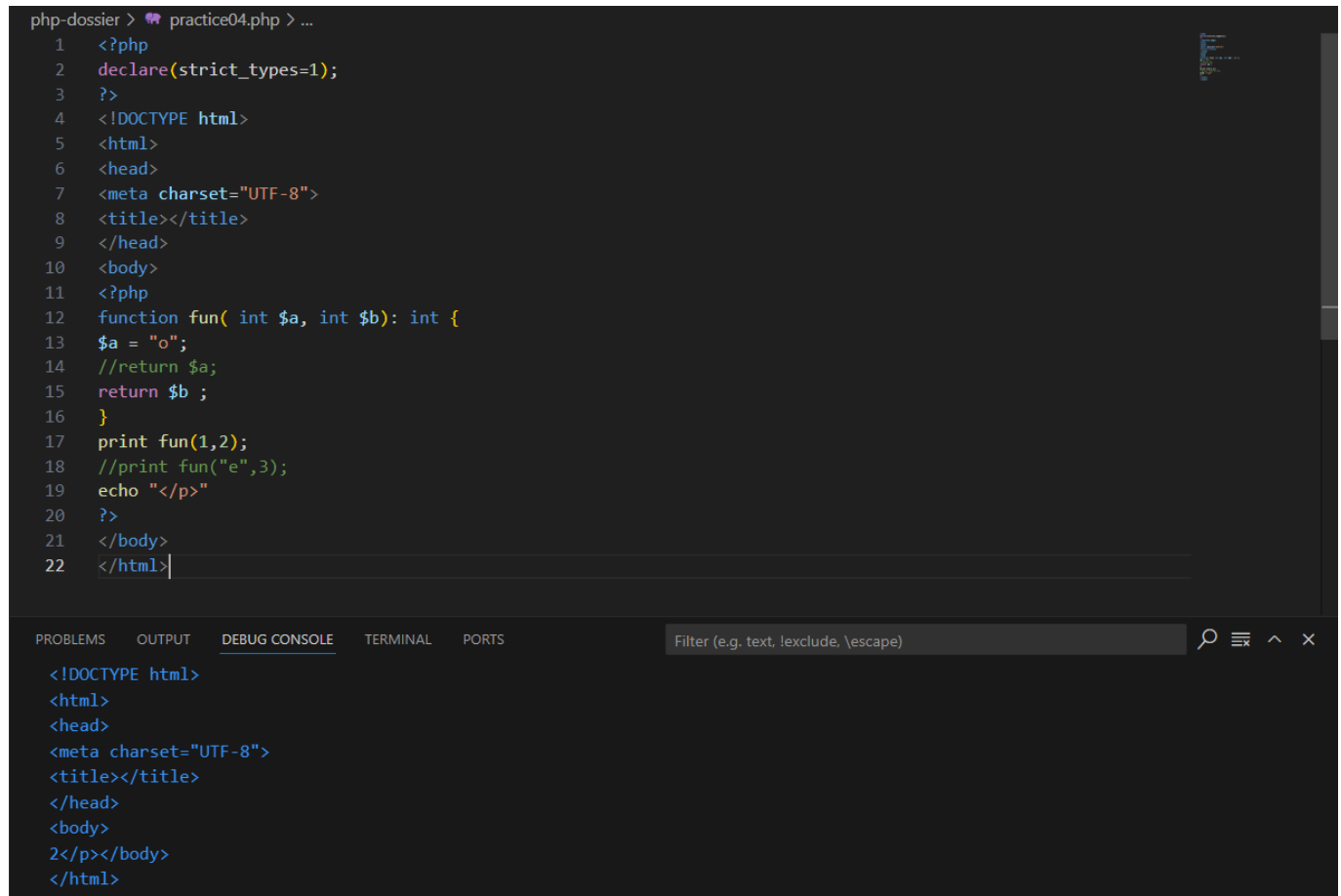
```
php-dossier > 🐘 practice03.php > ...
 1  <!DOCTYPE html>
 2  <html>
 3  <head>
 4  <meta charset="UTF-8">
 5  <title></title>
 6  </head>
 7  <body>
 8  <?php
 9  function sum( int $a, int $b): int {
10  return $a + $b;
11  }
12  echo "<p> la suma de uno más dos es: ";
13  $resultado = sum(1,2);
14  print sum(1,2);
15  echo "</p>"
16  ?>
17  </body>
18  </html>
19
20  <?php
21  declare( strict_types=1);
22  ?>
```

## Práctica 04

📁 ¿ Da error ? ¿ Por qué ?

Al probar el código proporcionado, observamos que no hay error y en efecto la función funciona correctamente. Esto es debido a que estamos devolviendo un resultado que tiene el tipo de variable que se espera.

- Captura:



```
php-dossier > practice04.php > ...
1  <?php
2  declare(strict_types=1);
3  ?>
4  <!DOCTYPE html>
5  <html>
6  <head>
7  <meta charset="UTF-8">
8  <title></title>
9  </head>
10 <body>
11 <?php
12 function fun( int $a, int $b): int {
13     $a = "o";
14     //return $a;
15     return $b ;
16 }
17 print fun(1,2);
18 //print fun("e",3);
19 echo "</p>"
20 ?>
21 </body>
22 </html>
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Filter (e.g. text, lexclude, \escape)

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title></title>
</head>
<body>
2</p></body>
</html>
```

📁 Quitar el comentario a: return \$a; ¿ Da error ahora ? ¿ Por qué ?

Al realizar el cambio, nos encontramos con un error puesto que en la función igualamos '\$a' a una cadena de texto y al ahora retornar este argumento en vez de '\$b' tenemos un error, ya que la propia función especifica que devuelve un valor entero.

- Captura:

```
php-dossier > 🐞 practice04.php > ...
1  <?php
2  declare( strict_types=1);
3  ?>
4  <!DOCTYPE html>
5  <html>
6  |   <head>
7  <meta charset="UTF-8">
8  <title></title>
9  </head>
10 <body>
11 <?php
12 function fun( int $a, int $b): int {
13     $a = "o";
14     return $a;
15     //return $b ;
16 }
17 print fun(1,2);
18 //print fun("e",3);
19 echo "</p>"
20 ?>
21 </body>
22 </html>
```

PROBLEMS 42 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
<!DOCTYPE html>
<html>
  <head>
<meta charset="UTF-8">
<title></title>
</head>
<body>
PHP Fatal error:  Uncaught TypeError: fun(): Return value must be of type int, string returned in C:\Users\
Stack trace:
#0 C:\Users\nabil\repositorios-git\aed-nabil\unit-01\php-dossier\practice04.php(17): fun('o', 2)
#1 {main}
  thrown in C:\Users\nabil\repositorios-git\aed-nabil\unit-01\php-dossier\practice04.php on line 14
```

📁 Quitar comentario a: `print fun("e",3);` ¿ Da error ?

Tras realizar este cambio, también tenemos un error. En este caso es por pasarle por parametros a la función un tipo de dato distinto al esperado.

- Captura:

```
php-dossier > practice04.php > ...
1  <?php
2  declare( strict_types=1);
3  ?>
4  <!DOCTYPE html>
5  <html>
6  |   <head>
7  |   <meta charset="UTF-8">
8  |   <title></title>
9  |   </head>
10 |   <body>
11 |   <?php
12 |   function fun( int $a, int $b): int {
13 |   $a = "o";
14 |   //return $a;
15 |   return $b ;
16 |   }
17 |   //print fun(1,2);|
18 |   print fun("e",3);
19 |   echo "</p>"
20 |   ?>
21 |   </body>
22 |   </html>
```

PROBLEMS 44 OUTPUT DEBUG CONSOLE TERMINAL PORTS

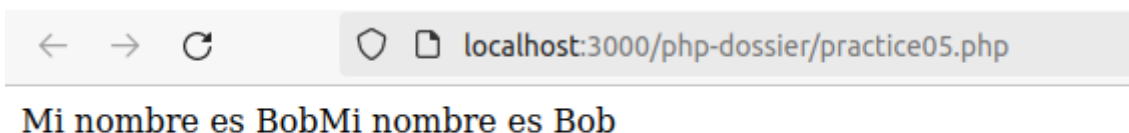
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title></title>
  </head>
  <body>
    PHP Fatal error:  Uncaught TypeError: fun(): Argument #1 ($a) must be of type int, string given, called in C:\Users\
ossier\practice04.php:12
    Stack trace:
    #0 C:\Users\nabil\repositorios-git\aed-nabil\unit-01\php-dossier\practice04.php(18): fun('e', 3)
    #1 {main}
    thrown in C:\Users\nabil\repositorios-git\aed-nabil\unit-01\php-dossier\practice04.php on line 12
```

## Práctica 05

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

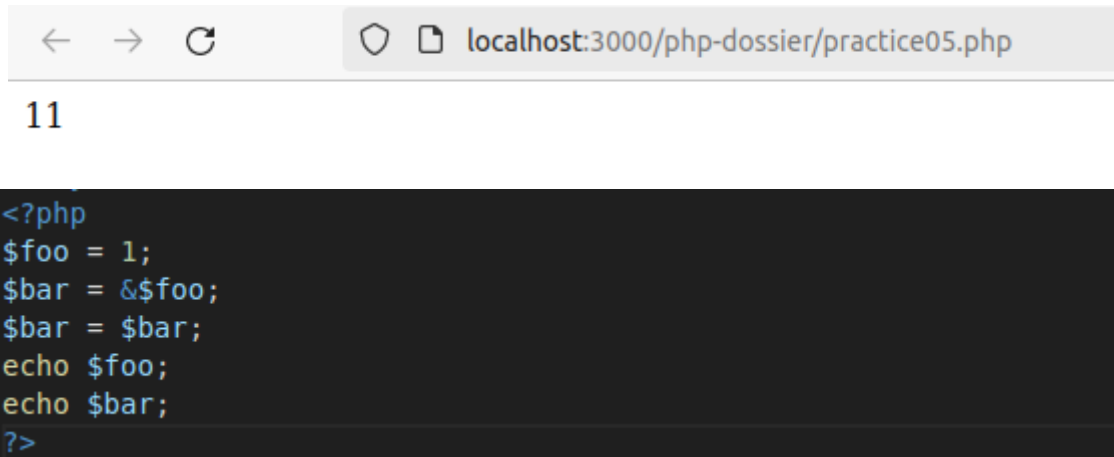
Originalmente funciona de esta manera.

- Captura:



Al realizar el cambio a valores numéricos se muestra de la siguiente forma, donde podemos apreciar que sigue funcionando;

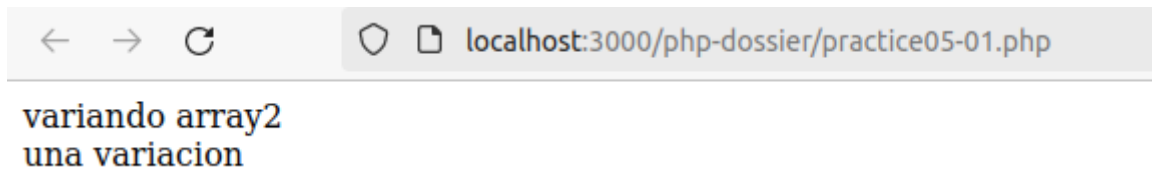
- Captura:



## Práctica 05.5

📁 Crear un array: `$mivar = [];` Introducir datos: `array_push($mivar,"uno");` y hacer una asignación a otras variables. Una por referencia y la otra por valor: `$arr1 = $mivar;` `$arr2 = &$mivar;` modificar la posición cero de esas variable : `$arr1[0] = "una variación";` `$arr2[0] = "variando array2 ";` y mostrar el contenido de `$mivar[0]` y `$arr1[0]` ¿ qué es lo que ha ocurrido ?

- Captura:



## Práctica 06

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

- Código:

```
<?php
    var_dump ($_REQUEST);
    var_dump ($_SERVER["SERVER_NAME"]);
    var_dump ($_SERVER[PHP_SELF]);
?>
```

- Captura:



```
← → ↺ localhost:3000/php-dossier/practice06.php?a=2&b=3 🌐 ☆ 📄 📁 ☰  
array(26) { ["DOCUMENT_ROOT"]=> string(35) "/home/dam/nalleon/aed-nabil/unit-01" ["REMOTE_ADDR"]=>  
string(9) "127.0.0.1" ["REMOTE_PORT"]=> string(5) "43774" ["SERVER_SOFTWARE"]=> string(40) "PHP  
8.1.2-1ubuntu2.14 Development Server" ["SERVER_PROTOCOL"]=> string(8) "HTTP/1.1" ["SERVER_NAME"]=>  
string(9) "localhost" ["SERVER_PORT"]=> string(4) "3000" ["REQUEST_URI"]=> string(35) "/php-dossier/  
practice06.php?a=2&b=3" ["REQUEST_METHOD"]=> string(3) "GET" ["SCRIPT_NAME"]=> string(27) "/php-  
dossier/practice06.php" ["SCRIPT_FILENAME"]=> string(62) "/home/dam/nalleon/aed-nabil/unit-01/php-dossier/  
practice06.php" ["PHP_SELF"]=> string(27) "/php-dossier/practice06.php" ["QUERY_STRING"]=> string(7)  
"a=2&b=3" ["HTTP_HOST"]=> string(14) "localhost:3000" ["HTTP_USER_AGENT"]=> string(70) "Mozilla/5.0  
(X11; Linux x86_64; rv:123.0) Gecko/20100101 Firefox/123.0" ["HTTP_ACCEPT"]=> string(85) "text/  
html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8"  
["HTTP_ACCEPT_LANGUAGE"]=> string(35) "es-ES;es;q=0.8,en-US;q=0.5,en;q=0.3"  
["HTTP_ACCEPT_ENCODING"]=> string(17) "gzip, deflate, br" ["HTTP_CONNECTION"]=> string(10) "keep-alive"  
["HTTP_UPGRADE_INSECURE_REQUESTS"]=> string(1) "1" ["HTTP_SEC_FETCH_DEST"]=> string(8)  
"document" ["HTTP_SEC_FETCH_MODE"]=> string(8) "navigate" ["HTTP_SEC_FETCH_SITE"]=> string(4) "none"  
["HTTP_SEC_FETCH_USER"]=> string(2) "?1" ["REQUEST_TIME_FLOAT"]=> float(1726498045.062622)  
["REQUEST_TIME"]=> int(1726498045) }
```

---

## Práctica 07

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

- Código:

```
<?php  
$variable = null;  
var_dump($variable);  
echo "<br>";  
unset($variable);  
var_dump($variable);  
?>
```

A simple vista, podemos apreciar como establecer un valor nulo en la variable simplemente lo define como tal mientras que si utilizamos unset se nos especifica donde hemos dejado la variable nula.

- Captura:

NULL

**Warning:** Undefined variable \$variable in C:  
\\Users\\nabil\\repositorios-git\\aed-nabil\\unit-01\\php-  
dossier\\practice07.php on line 14  
NULL

---

## Práctica 08

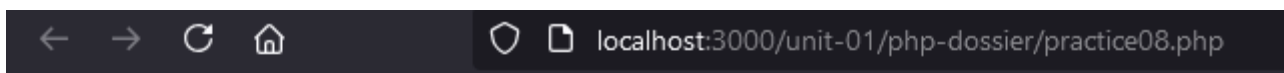
📁 Ejecutar el script anterior ¿ hay alguna diferencia antes y después del cast ? Tomar captura de pantalla

- Código:

```
<?php
    $unavar = 1.3;
    var_dump($unavar);
    echo "<br>";
    $unavar = (int) $unavar;
    var_dump($unavar);
?>
```

Podemos apreciar una diferencia clara puesto que el valor de '\$unavar' a cambiado de un float a un entero por lo que ha perdido su parte decimal al truncarse.

- Captura:



float(1.3)  
int(1)

---

## Práctica 08.5

📁 ¿ Qué ámbito tienen las constantes ? ¿ realmente no se puede poner varios valores en un constante ?

Las constantes tienen un ámbito global, es decir que se puede acceder a ellas una vez declaradas desde cualquier método del scripts. Una vez ya hayamos declarado el valor de una constante no podremos redefinirlo o modificarlo.

📁 Probar fuera de una función a crear constante: const PULGADA = 2.53; ahora tratar de establecerla de nuevo mediante: PULGADA = 7; const PULGADA = 8; \$PULGADA = 9; hacer echo en cada caso.

- Código:

```
<?php
    const PULGADA = 2.53;
    var_dump(PULGADA);
    echo "</br>";
```

```
const PULGADA = 8;
var_dump(PULGADA);
echo "</br>";
$PULGADA = 9;
var_dump(PULGADA);

?>
```

📁 Crear la constante en ámbito global ( fuera de función ) ¿ se puede acceder dentro de una función ? ¿ se puede establecer: const PULGADA = 10 dentro de una función ? Tomar capturas de pantalla en cada caso y explicar lo que ha ocurrido

- Código:

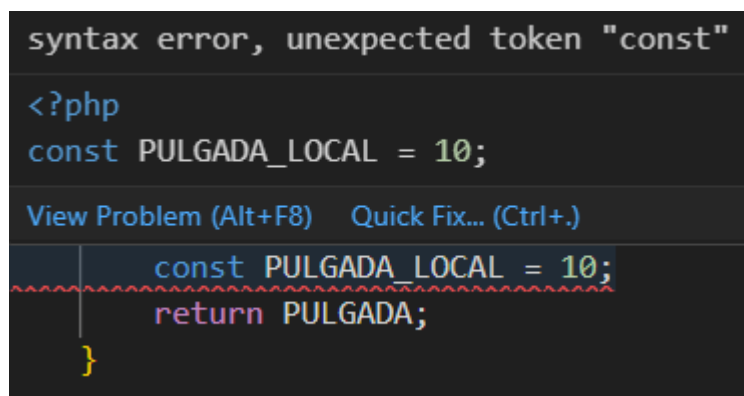
```
<?php
const PULGADA = 2.53;

function testConstants(){
    //const PULGADA_LOCAL = 10; NO se pueden establecer dentro de una función
    return PULGADA;
}

?>
```

Si hemos creado la variable globalmente no tendremos problema en llamarla dentro de una función para utilizarla, en cambio si intentamos declarar en este caso la constante 'PULGADA\_LOCAL' dentro de la función tendremos un error de sintaxis.

- Captura:



```
syntax error, unexpected token "const"

<?php
const PULGADA_LOCAL = 10;
View Problem (Alt+F8) Quick Fix... (Ctrl+.)
const PULGADA_LOCAL = 10;
return PULGADA;
}
```

## Práctica 09

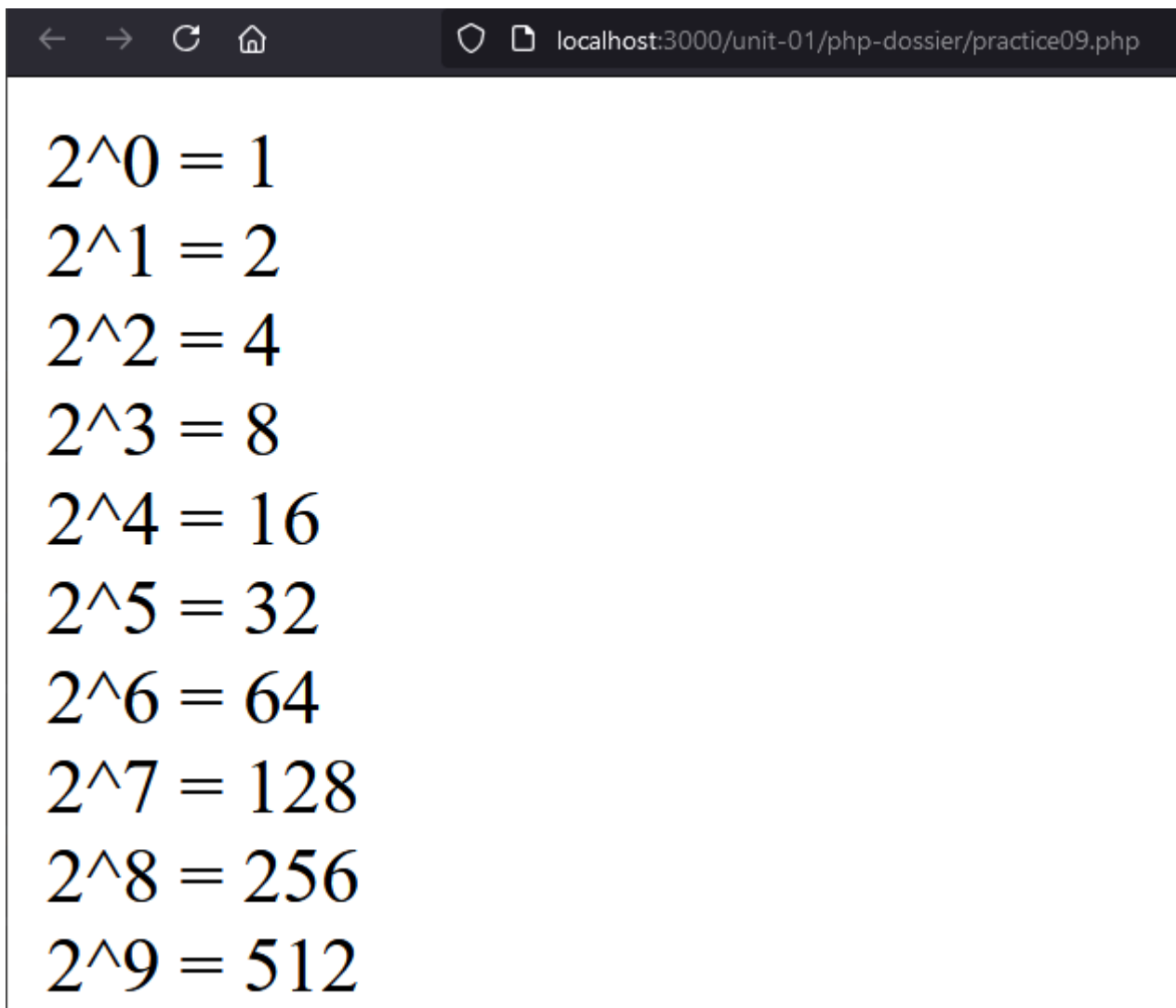
📁 Crear un script que muestre las potencias del número 2 desde  $2^1$  hasta  $2^9$  hacer uso del operador: \*\* Ir concatenando las salidas en pantalla de las potencias en una string mediante el operador de concatenación y asignación: .=

- Código:

```
<?php
function exponentation(){
    $num = 2;
    for($i = 0; $i < 10; $i++){
        $str = $num;
        $str .= "^".$i;
        echo $str. " = ". ($num**$i). "<br>";
    }
}

exponentation();
echo "<br>";
?>
```

- Captura:



---

## 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

- Código:

```
<?php

function decompositionNum($num) {
    $numAux = $num;

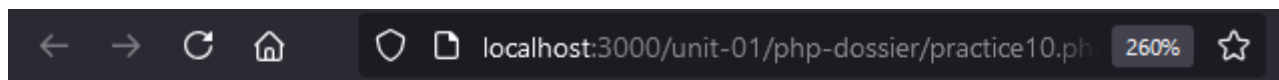
    $units = (int) $numAux % 10;
    $numAux = (int) ($numAux / 10);
    $tens = (int) $numAux % 10;
    $numAux = (int) ($numAux / 10);
    $hundreds = (int) $numAux % 10;
    $numAux = (int) ($numAux / 10);
    $thousand = (int) $numAux % 10;

    return $units . " * 1 + " . $tens . " * 10 + " . $hundreds . " * 100 + " .
    $thousand . " * 1000";
}

echo decompositionNum(3102);

?>
```

- Captura:



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

---

## Práctica 11

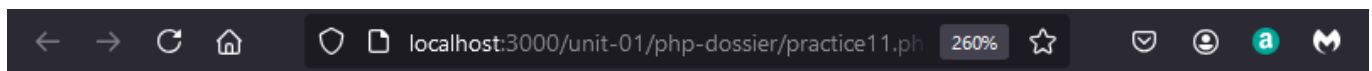
📁 Ejecutar el script y tomar captura de pantalla

- Código:

```
<?php
$var = "";
if(empty($var)){ // true because "" is considered empty
    echo '<br>empty($var) para $var="" ';
}else{
    echo '<br>!empty($var) para $var="" ';
}
```

```
if(isset($var)){ //true because var is set
    echo '<br>isset($var) para $var="" ';
}else{
    echo '<br> !isset($var) para $var="" ';
} if(empty($otherVar)){ //true because $otherVar is null
    echo '<br>empty($otherVar) para $otherVar que no se ha establecido ';
} else {
    echo '<br> !empty($otherVar) para $otherVar que no se ha establecido ';
}
if(isset($otherVar)){ //false because $otherVar is not set
    echo '<br>isset($otherVar) para $otherVar que no se ha establecido ';
} else {
    echo '<br> !isset($otherVar) para $otherVar que no se ha establecido ';
}
?>
```

- Captura:



isset(\$var) para \$var=""  
empty(\$otherVar) para \$otherVar que no se ha establecido  
!isset(\$otherVar) para \$otherVar que no se ha establecido

---

## Práctica 12

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

- Código:

```
<?php
$array = array('uno' =>1, 'dos'=>2, 'tres'=>40, 'cuatro'=>55);
$str = "La posición 'tres' contiene el dato $array['tres']";
?>
```

Se nos muestra un error de sintax al estar llamando a la variable '\$array' de esta forma.

- Captura:

```
syntax error, unexpected string content "", expecting "-" or identifier or variable or number
'Expression' expected. php
']' expected. php
View Problem (Alt+F8) Quick Fix... (Ctrl+.)
$str = "La posición 'tres' contiene el dato $array['tres']";
?>
```

Para poder solucionarlo debemos de utilizar la siguiente sintaxis:

```
<?php
$array = array('uno' =>1, 'dos'=>2, 'tres'=>40, 'cuatro'=>55);
$str = "La posición 'tres' contiene el dato {$array['tres']}";
?>
```

## Práctica 13

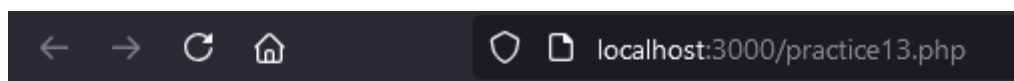
📁 Probar el script anterior y observar que ocurre. Probar ahora con llaves: `{{$variable}}` ¿hay diferencia?

- Código:

```
<?php
$variable = 'dato';
$dato = 5;
echo {{$variable}}.<br>;
?>
```

No hay ninguna diferencia, se nos sigue mostrando el valor de '\$dato' como 5;

- Captura:



5

## Práctica 14

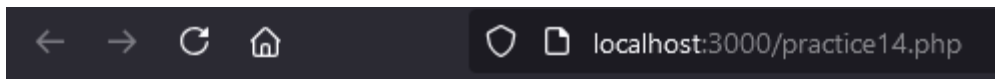
📁 Toma el código anterior e introduce una expresión “variable de variables” que permita definir las variables: \$dato0, \$dato1, ..., \$dato9 Cada una de ellas con el valor correspondiente: 0, 1,...,9

- Código:

```
<?php
    $var = 'dato';
    for($i=0; $i<10; $i++){
        ${$var.$i} = $i;
    }
    echo "<br> ${'dato0'} ";
    echo "<br> ${'dato1'} ";
    echo "<br> ${'dato2'} ";
    echo "<br> ${'dato3'} ";
    echo "<br> ${'dato4'} ";
    echo "<br> ${'dato5'} ";
    echo "<br> ${'dato6'} ";
    echo "<br> ${'dato7'} ";
    echo "<br> ${'dato8'} ";
    echo "<br> ${'dato9'} ";
?>
```

- Captura:





0  
1  
2  
3  
4  
5  
6  
7  
8  
9

---

## Práctica 15

📁 Ejecutar el script anterior ¿ se muestran las posiciones anteriores a la 2 ? ¿ y entre la 2 y la 7 ?.  
Realizar el mismo script pero en lugar de crear el array mediante los corchetes: `$array = []` hacerlo con la función `array()` ¿ hay diferencias en la salida en pantalla ? Ejecutar `var_dump($array)` después de cada asignación de un valor al array. Tomar captura de pantalla de los resultados

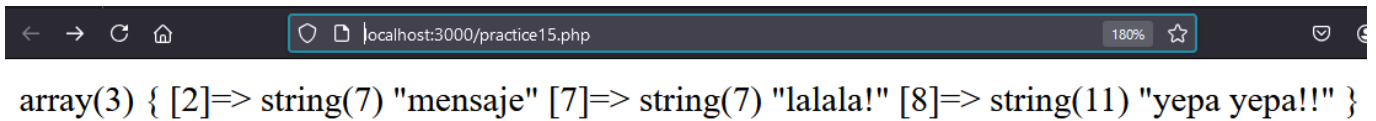
- Código:

```
<?php
    $array = [];
    $array[2]="mensaje";
    $array[7]="lalala!";
    $array[]="yepa yepa!!";
    var_dump($array);
?>
```

- 

NO se muestran las posiciones anteriores a la 2 ni entre esta y la 7, puesto que están vacías.

- Captura:



📁 Realizar el mismo script pero en lugar de crear el array mediante los corchetes: `$array = []` hacerlo con la función `array()` ¿hay diferencias en la salida en pantalla? Ejecutar `var_dump($array)` después de cada asignación de un valor al array. Tomar captura de pantalla de los resultados

- Código:

```
<?php
    $array = [];

    $array[2]="mensaje";
    var_dump($array);
    echo "</br>";

    $array[7]="lalala!";
    var_dump($array);
    echo "</br>";

    $array[]="yepa yepa!!";
    var_dump($array);
    echo "</br>";

    $array2 = array();
    $array2[2]="mensaje";
    var_dump($array2);
    echo "</br>";

    $array2[7]="lalala!";
    var_dump($array2);
    echo "</br>";

    $array2[]="yepa yepa!!";
    var_dump($array2);

?>
```

- Captura:



```
array(1) { [2]=> string(7) "mensaje" }
array(2) { [2]=> string(7) "mensaje" [7]=> string(7) "lalala!" }
array(3) { [2]=> string(7) "mensaje" [7]=> string(7) "lalala!" [8]=> string(11) "yepa yepa!!" }
array(1) { [2]=> string(7) "mensaje" }
array(2) { [2]=> string(7) "mensaje" [7]=> string(7) "lalala!" }
array(3) { [2]=> string(7) "mensaje" [7]=> string(7) "lalala!" [8]=> string(11) "yepa yepa!!" }
```

---

## Práctica 15.5

📁 Crear un array asociativo dejando sin poner en algunas ocasiones la parte de la clave dejando únicamente el valor ( al estilo de si fuera un array no asociativo ) hacer un `var_dump()` y recorrerlo con un `for` ( no con un `foreach` ) ¿ muestra algún valor ? ¿ genera error ?

- Código:

```
<?php
$array = [];
$array[2]="mensaje";
$array[7]="lalala!";
$array[]="yepa yepa!!";

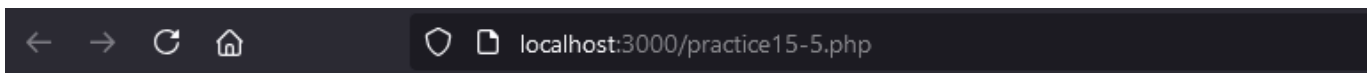
echo "<br>";

for($i = 0; $i < 9; $i++){
    if(isset($array[$i])){
        var_dump($array[$i]);
    }
}

?>
```

Muestra correctamente valor con su clave asociada sin generar errores.

- Captura:



```
string(7) "mensaje" string(7) "lalala!" string(11) "yepa yepa!!"
```

---

## Práctica 16

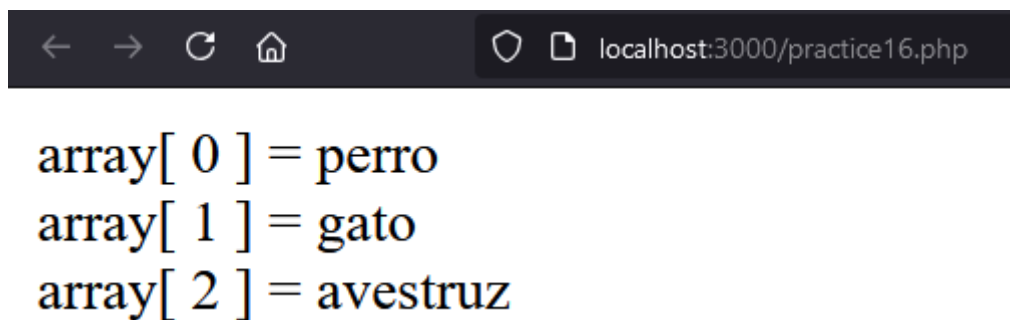
📁 Ejecutar el script anterior. ¿Tenemos que usar los nombres de variables \$key y \$val ? Sustituir por otros nombres de variables y ver si hay algún problema

- Código:

```
<?php
$array = array('perro', 'gato', 'avestruz');
foreach ($array as $index => $animals) {
    print "array[ $index ] = $animals </br>";
}
?>
```

Funciona exactamente igual tanto si dejamos las variables como '\$key' y '\$value' que intercambiándolos por otros nombres mientras se cumpla bien la sintaxis.

- Captura:



---

## Práctica 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

- Código:

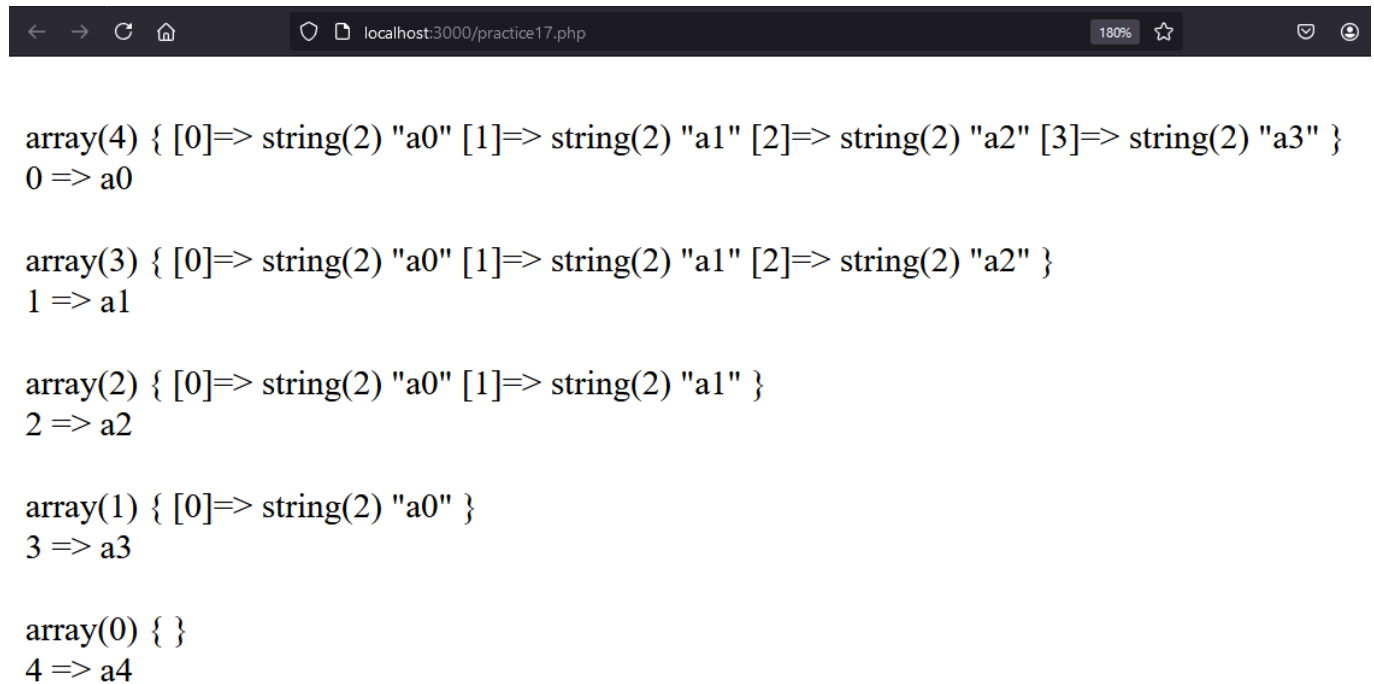
```
<?php
$array = [];
for($i=0;$i<5;$i++){
    $array[] = "a" . $i;
}

$j=count($array);
foreach( $array as $key => $val){
    $j--;
    unset($array[$j]);
    echo "<br>";
    var_dump($array);
    echo "<br> $key => $val ";
    echo "<br>";
}
```

```
}
?>
```

En PHP he podido apreciar que hay ningún tipo de error al eliminar los valores de un array usando un for each, en cambio como apreciamos como los va eliminando correctamente.

- Captura:



```
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
```

## Práctica 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

- Código:

```
<?php
$array = ["a","a","a","a","a"];
$j=count($array);
foreach( $array as $key => &$val){
    $j--;
    $array[$j] .= $j;
    echo "<br>";
    var_dump($array);
    echo "<br> $key => $val";
    echo "<br> $key => $array[$key]";
    echo "<br>";
}

$arr = array(1, 2, 3, 4);
foreach ($arr as &$val) {
```

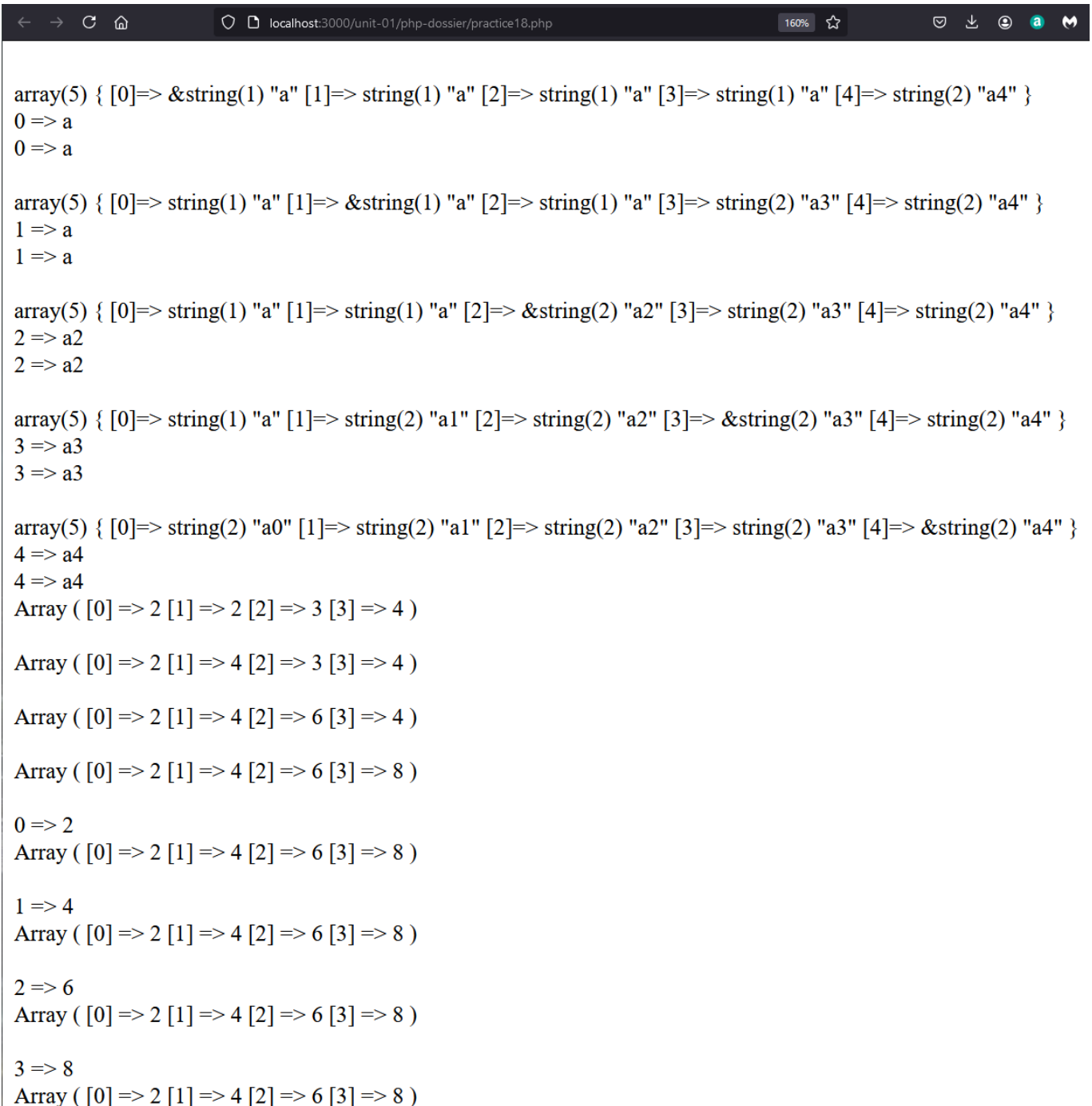
```

        $val = $val * 2;
        print_r($arr);
        echo "<br><br>";
    }

    foreach ($arr as $key => $val2) {
        echo "{$key} => {$val2} <br>";
        print_r($arr);
        echo "<br><br>";
    }
?>

```

- Captura:



```

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

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

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

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

array(5) { [0]=> string(2) "a0" [1]=> string(2) "a1" [2]=> string(2) "a2" [3]=> string(2) "a3" [4]=> &string(2) "a4" }
4 => a4
4 => a4
Array ( [0] => 2 [1] => 2 [2] => 3 [3] => 4 )

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

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

Array ( [0] => 2 [1] => 4 [2] => 6 [3] => 8 )

0 => 2
Array ( [0] => 2 [1] => 4 [2] => 6 [3] => 8 )


1 => 4
Array ( [0] => 2 [1] => 4 [2] => 6 [3] => 8 )

2 => 6
Array ( [0] => 2 [1] => 4 [2] => 6 [3] => 8 )

3 => 8
Array ( [0] => 2 [1] => 4 [2] => 6 [3] => 8 )

```

## Práctica 19

 Ejecutar el script anterior. Tomar captura de pantalla del resultado

- Código:

```
<?php
$array = ["a","a","a","a","a"];
$j=count($array);
foreach( $array as $key => &$val){
    $j--;
    $array[$j] .= $j;
    echo "<br>";
    var_dump($array);
    echo "<br> $key => $val";
    echo "<br> $key => $array[$key]";
    echo "<br>";
}
?>
```

- Captura:

```
← → ↺ 🏠 localhost:3000/unit-01/php-dossier/practice19.php

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 )

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 )
```

## Práctica 20

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

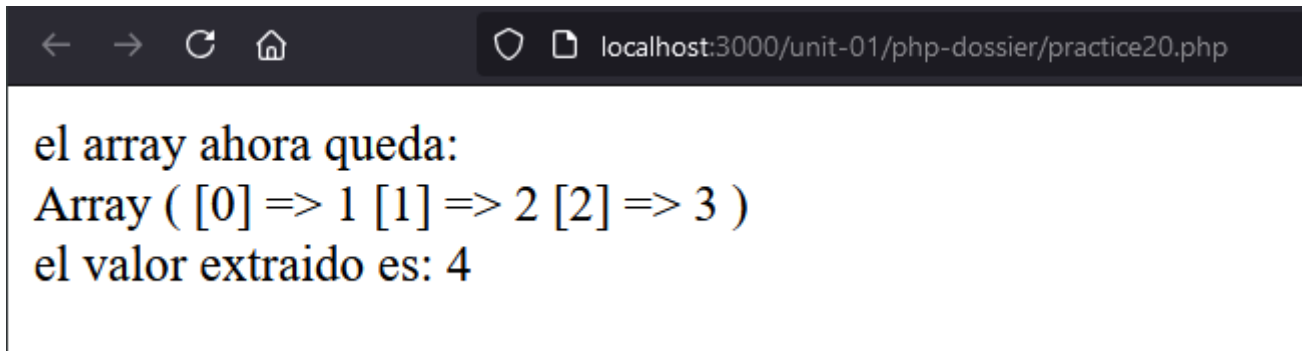
- Código:

```
<?php
$arr= ["1","2","3","4"];
```



```
$va = array_pop($arr);  
echo "el array ahora queda: <br>";  
print_r($arr);  
echo "<br>el valor extraido es: " . $va;  
?>
```

- Captura:



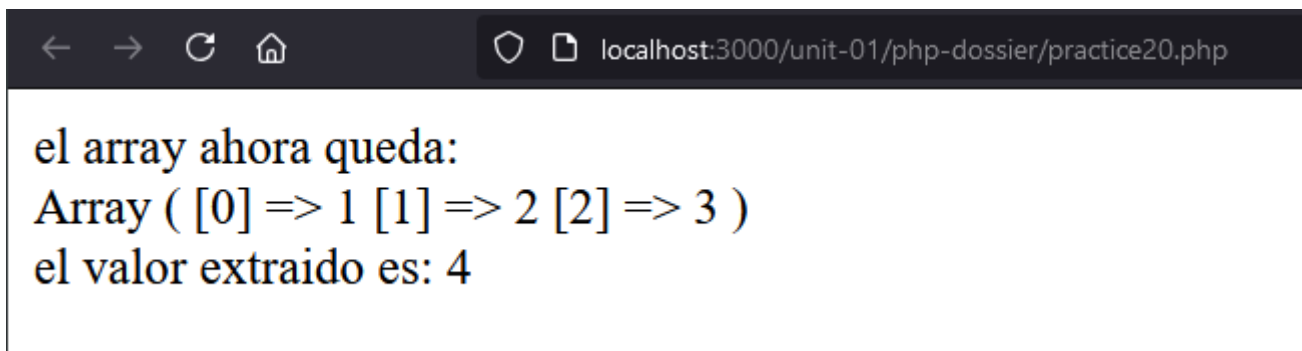
## Práctica 20

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

- Código:

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

- Captura:



## Práctica 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 for de 1 a 5 ir ejecutando sentencias array\_pop() y mostrar como queda el array en cada iteración

- Código:

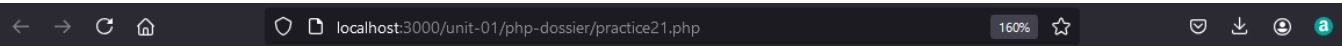
```
<?php
$arr = [];

for($i=0; $i<10; $i++){
    $arr[] = $i;
    echo "Array value $i: ";
    print_r($arr);
    echo "<br>";
}

for($j=0; $j<5; $j++){
    $arrPop = array_pop($arr);
    echo "Popped array value $j: ";
    print_r($arr);
    echo "<br>";
    echo "Value popped: " . $arrPop;
    echo "<br>";
}

?>
```

- Captura:



```
Array value 0: Array ( [0] => 0 )
Array value 1: Array ( [0] => 0 [1] => 1 )
Array value 2: Array ( [0] => 0 [1] => 1 [2] => 2 )
Array value 3: Array ( [0] => 0 [1] => 1 [2] => 2 [3] => 3 )
Array value 4: Array ( [0] => 0 [1] => 1 [2] => 2 [3] => 3 [4] => 4 )
Array value 5: Array ( [0] => 0 [1] => 1 [2] => 2 [3] => 3 [4] => 4 [5] => 5 )
Array value 6: Array ( [0] => 0 [1] => 1 [2] => 2 [3] => 3 [4] => 4 [5] => 5 [6] => 6 )
Array value 7: Array ( [0] => 0 [1] => 1 [2] => 2 [3] => 3 [4] => 4 [5] => 5 [6] => 6 [7] => 7 )
Array value 8: Array ( [0] => 0 [1] => 1 [2] => 2 [3] => 3 [4] => 4 [5] => 5 [6] => 6 [7] => 7 [8] => 8 )
Array value 9: Array ( [0] => 0 [1] => 1 [2] => 2 [3] => 3 [4] => 4 [5] => 5 [6] => 6 [7] => 7 [8] => 8 [9] => 9 )
Popped array value 0: Array ( [0] => 0 [1] => 1 [2] => 2 [3] => 3 [4] => 4 [5] => 5 [6] => 6 [7] => 7 [8] => 8 )
Value popped: 9
Popped array value 1: Array ( [0] => 0 [1] => 1 [2] => 2 [3] => 3 [4] => 4 [5] => 5 [6] => 6 [7] => 7 )
Value popped: 8
Popped array value 2: Array ( [0] => 0 [1] => 1 [2] => 2 [3] => 3 [4] => 4 [5] => 5 [6] => 6 )
Value popped: 7
Popped array value 3: Array ( [0] => 0 [1] => 1 [2] => 2 [3] => 3 [4] => 4 [5] => 5 )
Value popped: 6
Popped array value 4: Array ( [0] => 0 [1] => 1 [2] => 2 [3] => 3 [4] => 4 )
Value popped: 5
```

## Práctica 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

- Código:

```
<?php
    $arr = [];

    for($i=0; $i<10; $i++){
        $arr[] = $i;
        echo "Array unshift values $i: ";
        print_r($arr);
        echo "<br>";
    }

    echo "<br>";

    for($j=0; $j<5; $j++){
        $arrShift = array_shift($arr);
        echo "Shifted array value $j: ";
        print_r($arr);
        echo "<br>";
        echo "Value shifted: " . $arrShift;
        echo "<br>";
    }
?>
```

- Captura:

```

Array unshift values 0: Array ( [0] => 0 )
Array unshift values 1: Array ( [0] => 0 [1] => 1 )
Array unshift values 2: Array ( [0] => 0 [1] => 1 [2] => 2 )
Array unshift values 3: Array ( [0] => 0 [1] => 1 [2] => 2 [3] => 3 )
Array unshift values 4: Array ( [0] => 0 [1] => 1 [2] => 2 [3] => 3 [4] => 4 )
Array unshift values 5: Array ( [0] => 0 [1] => 1 [2] => 2 [3] => 3 [4] => 4 [5] => 5 )
Array unshift values 6: Array ( [0] => 0 [1] => 1 [2] => 2 [3] => 3 [4] => 4 [5] => 5 [6] => 6 )
Array unshift values 7: Array ( [0] => 0 [1] => 1 [2] => 2 [3] => 3 [4] => 4 [5] => 5 [6] => 6 [7] => 7 )
Array unshift values 8: Array ( [0] => 0 [1] => 1 [2] => 2 [3] => 3 [4] => 4 [5] => 5 [6] => 6 [7] => 7 [8] => 8 )
Array unshift values 9: Array ( [0] => 0 [1] => 1 [2] => 2 [3] => 3 [4] => 4 [5] => 5 [6] => 6 [7] => 7 [8] => 8 [9] => 9 )

Shifted array value 0: Array ( [0] => 1 [1] => 2 [2] => 3 [3] => 4 [4] => 5 [5] => 6 [6] => 7 [7] => 8 [8] => 9 )
Value shifted: 0
Shifted array value 1: Array ( [0] => 2 [1] => 3 [2] => 4 [3] => 5 [4] => 6 [5] => 7 [6] => 8 [7] => 9 )
Value shifted: 1
Shifted array value 2: Array ( [0] => 3 [1] => 4 [2] => 5 [3] => 6 [4] => 7 [5] => 8 [6] => 9 )
Value shifted: 2
Shifted array value 3: Array ( [0] => 4 [1] => 5 [2] => 6 [3] => 7 [4] => 8 [5] => 9 )
Value shifted: 3
Shifted array value 4: Array ( [0] => 5 [1] => 6 [2] => 7 [3] => 8 [4] => 9 )
Value shifted: 4

```

## Práctica 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

- Código:

```

<?php
    $arr = [];

    for($i=0; $i<10; $i++){
        $arr[] = $i;
        echo "Array unshift values $i: ";
        print_r($arr);
        echo "<br>";
    }

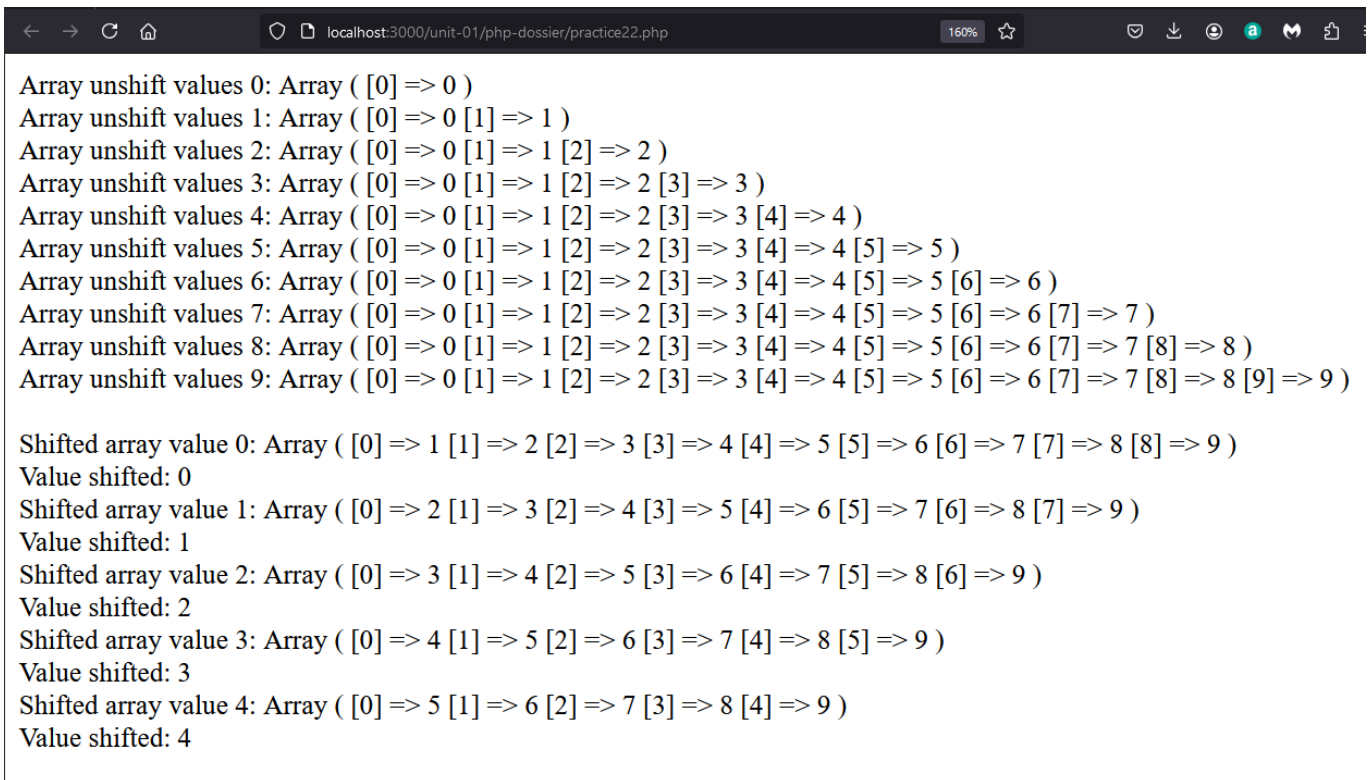
    echo "<br>";

    for($j=0; $j<5; $j++){
        $arrShift = array_shift($arr);
        echo "Shifted array value $j: ";
        print_r($arr);
        echo "<br>";
        echo "Value shifted: " . $arrShift;
        echo "<br>";
    }

```

```
}
?>
```

- Captura:



The screenshot shows a web browser window with the address bar displaying 'localhost:3000/unit-01/php-dossier/practice22.php'. The page content lists 10 unshifted arrays and 5 shifted arrays, each with its corresponding 'Value shifted'.

```
Array unshift values 0: Array ( [0] => 0 )
Array unshift values 1: Array ( [0] => 0 [1] => 1 )
Array unshift values 2: Array ( [0] => 0 [1] => 1 [2] => 2 )
Array unshift values 3: Array ( [0] => 0 [1] => 1 [2] => 2 [3] => 3 )
Array unshift values 4: Array ( [0] => 0 [1] => 1 [2] => 2 [3] => 3 [4] => 4 )
Array unshift values 5: Array ( [0] => 0 [1] => 1 [2] => 2 [3] => 3 [4] => 4 [5] => 5 )
Array unshift values 6: Array ( [0] => 0 [1] => 1 [2] => 2 [3] => 3 [4] => 4 [5] => 5 [6] => 6 )
Array unshift values 7: Array ( [0] => 0 [1] => 1 [2] => 2 [3] => 3 [4] => 4 [5] => 5 [6] => 6 [7] => 7 )
Array unshift values 8: Array ( [0] => 0 [1] => 1 [2] => 2 [3] => 3 [4] => 4 [5] => 5 [6] => 6 [7] => 7 [8] => 8 )
Array unshift values 9: Array ( [0] => 0 [1] => 1 [2] => 2 [3] => 3 [4] => 4 [5] => 5 [6] => 6 [7] => 7 [8] => 8 [9] => 9 )

Shifted array value 0: Array ( [0] => 1 [1] => 2 [2] => 3 [3] => 4 [4] => 5 [5] => 6 [6] => 7 [7] => 8 [8] => 9 )
Value shifted: 0
Shifted array value 1: Array ( [0] => 2 [1] => 3 [2] => 4 [3] => 5 [4] => 6 [5] => 7 [6] => 8 [7] => 9 )
Value shifted: 1
Shifted array value 2: Array ( [0] => 3 [1] => 4 [2] => 5 [3] => 6 [4] => 7 [5] => 8 [6] => 9 )
Value shifted: 2
Shifted array value 3: Array ( [0] => 4 [1] => 5 [2] => 6 [3] => 7 [4] => 8 [5] => 9 )
Value shifted: 3
Shifted array value 4: Array ( [0] => 5 [1] => 6 [2] => 7 [3] => 8 [4] => 9 )
Value shifted: 4
```

## Práctica 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

- Código:

```
<?php
    echo "<table>";
    echo "<tr>";
    echo "<th>Var</th>";
    echo "<th>Value</th>";
    echo "</tr>";

    foreach ($_SERVER as $key => $value) {
        echo "<tr>";
        echo "<td>$key</td>";
        echo "<td>$value</td>";
        echo "</tr>";
    }
```

```
        echo "</table>";
    }>
```

- Captura:

Var	Value
DOCUMENT_ROOT	C:\Users\nabil\repositorios-git\aed-nabil
REMOTE_ADDR	::1
REMOTE_PORT	50440
SERVER_SOFTWARE	PHP 8.2.12 Development Server
SERVER_PROTOCOL	HTTP/1.1
SERVER_NAME	localhost
SERVER_PORT	3000
REQUEST_URI	/unit-01/php-dossier/practice23.php
REQUEST_METHOD	GET
SCRIPT_NAME	/unit-01/php-dossier/practice23.php
SCRIPT_FILENAME	C:\Users\nabil\repositorios-git\aed-nabil\unit-01\php-dossier\practice23.php
PHP_SELF	/unit-01/php-dossier/practice23.php
HTTP_HOST	localhost:3000
HTTP_USER_AGENT	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:130.0) Gecko/20100101 Firefox/130.0
HTTP_ACCEPT	text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
HTTP_ACCEPT_LANGUAGE	en-US,en;q=0.5
HTTP_ACCEPT_ENCODING	gzip, deflate, br, zstd
HTTP_CONNECTION	keep-alive
HTTP_COOKIE	Idea-ec639071=50b52c77-fc29-4c6d-be14-060c9b94513e
HTTP_UPGRADE_INSECURE_REQUESTS	1
HTTP_SEC_FETCH_DEST	document
HTTP_SEC_FETCH_MODE	navigate
HTTP_SEC_FETCH_SITE	none
HTTP_SEC_FETCH_USER	?1
HTTP_PRIORITY	u=0, i
REQUEST_TIME_FLOAT	1726826631.1
REQUEST_TIME	1726826631

## Práctica 24

📁 Ejecutar los códigos de in\_array(), array\_search(), array\_values() tomar captura del resultado de la ejecución

- Código:

```
<?php
    echo "in_array code:";
    echo "</br>";

    $os = array("Mac", "NT", "Irix", "Linux");
    if (in_array("Irix", $os)) {
        echo "Existe Irix";
    }
    if (in_array("mac", $os)) {
        echo "Existe mac";
    }

    echo "</br>";

    $a = array('1.10', 12.4, 1.13);
    if (in_array('12.4', $a, true)) {
        echo "Se encontró '12.4' con comprobación estricta\n";
    }
    if (in_array(1.13, $a, true)) {
        echo "Se encontró 1.13 con comprobación estricta\n";
    }

    echo "</br>";
    echo "array_search code:";
    echo "</br>";

    $array = array(0 => 'azul', 1 => 'rojo', 2 => 'verde', 3 => 'rojo');
    $clave = array_search('verde', $array);
    echo $clave . "<br>";
    $clave = array_search('marrón', $array);
    if( $clave === FALSE) {
        echo "no se ha localizado el valor";
    } else {
        echo $clave;
    }

    echo "</br>";
    echo "array_values code:";
    echo "</br>";

    $array = array('azul', 'rojo', 'verde', 'amarillo', "blanco");
    unset($array[2]);
    unset($array[3]);
    print_r($array);
    $array = array_values($array);
    echo "</br>";
    print_r($array);
?>
```

- Captura:

in\_array code:

Existe Irix

Se encontró 1.13 con comprobación estricta

array\_search code:

2

no se ha localizado el valor

array\_values code:

Array ( [0] => azul [1] => rojo [4] => blanco )

Array ( [0] => azul [1] => rojo [2] => blanco )

---

## Práctica 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()

- Código:

```
<?php
$arr = [];
for ($i = 0; $i < 10; $i++) {
    $arr[$i] = rand (20, 25);
}

print_r($arr);

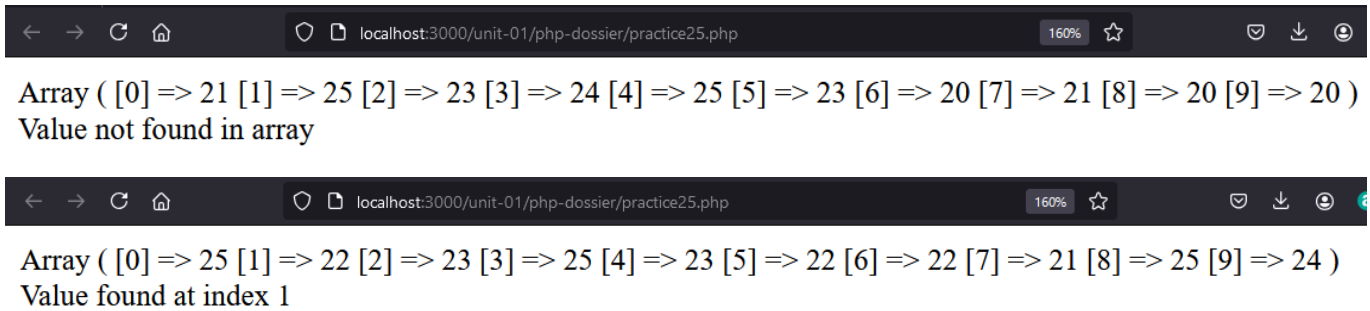
echo "<br>";

$key = array_search(22, $arr);
if( $key === FALSE) {
    echo "Value not found in array";
} else {
    echo "Value found at index " . $key;
}

?>
```

- Captura:





## Práctica 26

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

- Código:

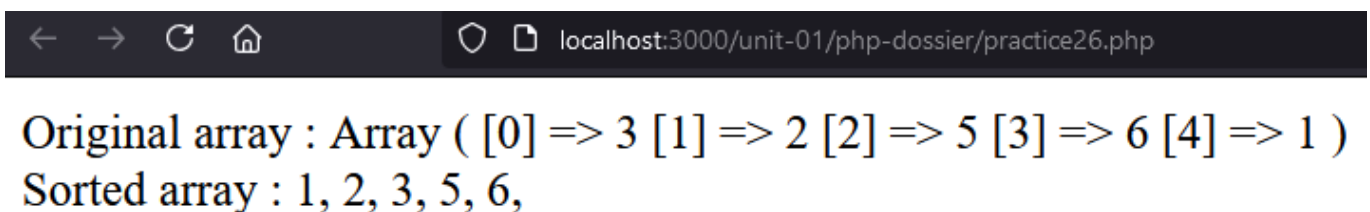
```
<?php
function cmp($a, $b) {
    return $a <=> $b;
}

$a = array(3, 2, 5, 6, 1);
echo "Original array : ";
print_r($a);
echo "</br>";

echo "Sorted array : ";
usort($a, "cmp");
foreach ($a as $valor) {
    echo " $valor, ";
}

?>
```

- Captura:



## Práctica 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()

- Código:

```
<?php
    $arr = [7,2,8,1,9,4];

    print_r($arr);
    echo "</br>";

    $key = array_search(4, $arr);
    found($key);

    function found($key) {
        if( $key === FALSE) {
            echo "Value not found in array";
        } else {
            echo "Value found at index " . $key;
        }
    }

    echo "</br>";

    function compare($val1, $val2) {
        return $val1 <=> $val2;
    }

    echo "</br>";
    usort($arr, "compare");
    $sortArr = [];

    foreach ($arr as $value) {
        $sortArr[] = $value;
    }

    print_r($sortArr);
    echo "</br>";

    $key = array_search(4, $sortArr);
    found($key);
?>
```

- Captura:




localhost:3000/unit-01/php-dossier/practice27.php

Array ( [0] => 7 [1] => 2 [2] => 8 [3] => 1 [4] => 9 [5] => 4 )  
Value found at index 5

Array ( [0] => 1 [1] => 2 [2] => 4 [3] => 7 [4] => 8 [5] => 9 )  
Value found at index 2

---

## Práctica 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 ?

Al quitar que el valor de "\$print" sea false tendremos un error al ejecutarlo, deteniendose el programa

- Código:

```
<?php
function sumar($a, $b, $print): float
{
    $suma = $a + $b;
    if ($print) {
        echo "resultado suma: $suma <br>";
    }
    return $suma;
}
$sum1=sumar(1,2);
$sum2=sumar(4,5,true);
echo "las operaciones para sum1 y sum2 dan: $sum1 , $sum2";
?>
```

- Captura:

```
[Mon Sep 23 11:15:05 2024] PHP Fatal error:  Uncaught ArgumentCountError: Too few arguments to function
sumar(), 2 passed in C:\Users\nabil\repositorios-git\aed-nabil\unit-01\php-dossier\practice28.php on
line 18 and exactly 3 expected in
C:\Users\nabil\repositorios-git\aed-nabil\unit-01\php-dossier\practice28.php:10
Stack trace:
#0 C:\Users\nabil\repositorios-git\aed-nabil\unit-01\php-dossier\practice28.php(18): sumar(1, 2)
#1 {main}
  thrown in C:\Users\nabil\repositorios-git\aed-nabil\unit-01\php-dossier\practice28.php on line 10
[Mon Sep 23 11:15:05 2024] [::1]:50089 [200]: GET /unit-01/php-dossier/practice28.php - Uncaught
ArgumentCountError: Too few arguments to function sumar(), 2 passed in
C:\Users\nabil\repositorios-git\aed-nabil\unit-01\php-dossier\practice28.php on line 18 and exactly 3
expected in C:\Users\nabil\repositorios-git\aed-nabil\unit-01\php-dossier\practice28.php:10
```

## Práctica 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

Podemos observar como los cambios que hacemos dentro de la función no influyen en el valor de la variable, por lo tanto estamos ante un caso de paso por valor.

- Código:

```
<?php
function modify(int $a): void {
    $a = 3;
}
$a = 2;
modify($a);
print_r($a);
?>
```

- Captura:

```
[Mon Sep 23 11:15:05 2024] PHP Fatal error:  Uncaught ArgumentCountError: Too few arguments to function
sumar(), 2 passed in C:\Users\nabil\repositorios-git\aed-nabil\unit-01\php-dossier\practice28.php on
line 18 and exactly 3 expected in
C:\Users\nabil\repositorios-git\aed-nabil\unit-01\php-dossier\practice28.php:10
Stack trace:
#0 C:\Users\nabil\repositorios-git\aed-nabil\unit-01\php-dossier\practice28.php(18): sumar(1, 2)
#1 {main}
  thrown in C:\Users\nabil\repositorios-git\aed-nabil\unit-01\php-dossier\practice28.php on line 10
[Mon Sep 23 11:15:05 2024] [::1]:50089 [200]: GET /unit-01/php-dossier/practice28.php - Uncaught
ArgumentCountError: Too few arguments to function sumar(), 2 passed in
C:\Users\nabil\repositorios-git\aed-nabil\unit-01\php-dossier\practice28.php on line 18 and exactly 3
expected in C:\Users\nabil\repositorios-git\aed-nabil\unit-01\php-dossier\practice28.php:10
```

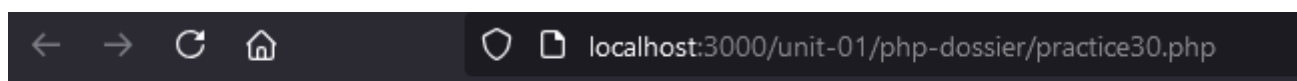
## Práctica 30

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

- Código:

```
<?php
function modify(int &$a): void {
    $a = 3;
}
$a = 2;
modify($a);
print_r($a);
?>
```

- Captura:



3

---

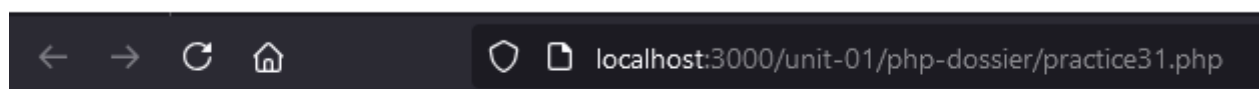
## Práctica 31

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

- Código:

```
<?php
function modify(array &$arr): void {
    $arr[] = 4;
}
$a = [1];
modify($a);
print_r($a);
?>
```

- Captura:



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

---

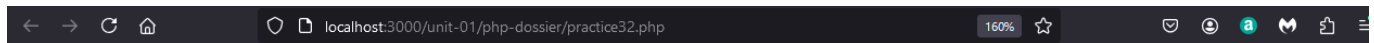
## Práctica 32

📁 Hacer lo anterior, pero usando require en lugar de include. Para que se note la diferencia la llamada de require debiera ser a un nombre de fichero incorrecto. Por ejemplo haremos que llame a: vars1.php cuando como sabemos el fichero es vars.php Tomar captura de pantalla

- Código:

```
<?php
    echo "Una $fruta $color";
    require 'vars1.php';
    echo "Una $fruta $color";
?>
```

- Captura:



**Warning:** Undefined variable \$fruta in C:\Users\nabil\repositorios-git\aed-nabil\unit-01\php-dossier\practice32.php on line 10

**Warning:** Undefined variable \$color in C:\Users\nabil\repositorios-git\aed-nabil\unit-01\php-dossier\practice32.php on line 10

Una

**Warning:** require(vars1.php): Failed to open stream: No such file or directory in C:\Users\nabil\repositorios-git\aed-nabil\unit-01\php-dossier\practice32.php on line 11

**Fatal error:** Uncaught Error: Failed opening required 'vars1.php' (include\_path='C:\xampp\php\PEAR') in C:\Users\nabil\repositorios-git\aed-nabil\unit-01\php-dossier\practice32.php:11 Stack trace: #0 {main} thrown in C:\Users\nabil\repositorios-git\aed-nabil\unit-01\php-dossier\practice32.php on line 11

## Práctica 33

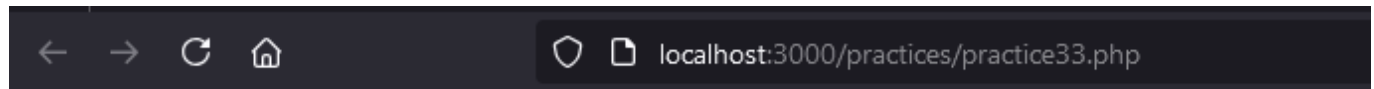
📁 Hacer lo anterior, pero se debe comprobar la diferencia de pasar el texto con urlencode y sin urlencode. Así que se propone poner dos parámetros: prueba y prueba2 unode ellos con urlencode y el otro sin él pasando en ambos casos el mismo texto en el value. Tomar captura de pantalla de lo obtenido

- Código:

```
<?php
    $conUrlEncode = urlencode('Pasando datos diría.. que hay que usar urlencode');
    echo "<a href=practice33.php?prueba='Pasando datos diría.. que hay que usar urlencode'>pasando datos</a>";
    echo "<br></br>";
    echo "<a href=practice33.php?prueba2={$conUrlEncode}>pasando datos</a>";
```

```
$recibido = $_GET["prueba"] ?? "nadita";  
$recibidoConEncode = $_GET["prueba2"] ?? "nadita";  
echo "<h3>se ha recibido:</h3>";  
echo "prueba: ". $recibido . "<br>";  
echo "prueba: ". $recibidoConEncode . "<br>";  
?>
```

- Captura:



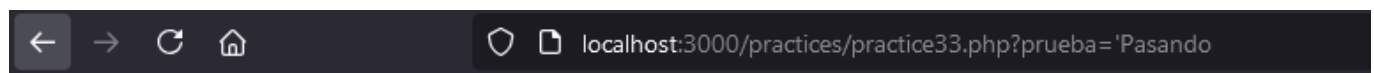
pasando datos

pasando datos

**se ha recibido:**

prueba: nadita

prueba: nadita



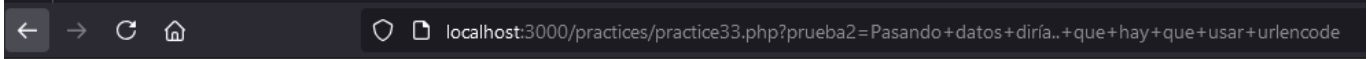
pasando datos

pasando datos

**se ha recibido:**

prueba: 'Pasando

prueba: nadita



[pasando datos](#)

[pasando datos](#)

**se ha recibido:**

prueba: nadita

prueba: Pasando datos diría.. que hay que usar urlencode

---

## Práctica 34

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

- Código:

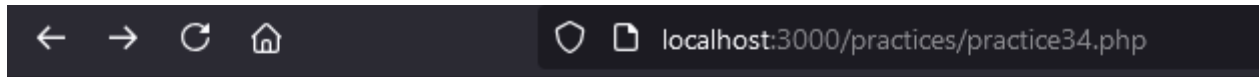
```
<?php
    $conUrlEncode = urlencode('Pasando datos diría.. que hay que usar urlencode');
    echo "<a href=practice34.php?prueba='Pasando datos diría.. que hay que usar
urlencode'>pasando datos</a>";
    echo "<br></br>";
    echo "<a href=practice34.php?prueba2={$conUrlEncode}>pasando datos</a>";

    $recibido = $_GET["prueba"] ?? "nadita";
    $recibidoConEncode = $_GET["prueba2"] ?? "nadita";

    foreach ($_GET as $key => $value) {
        echo "<br></br>";
        echo $key. ": ". $value;
        echo "<br></br>";
    }
?>
```

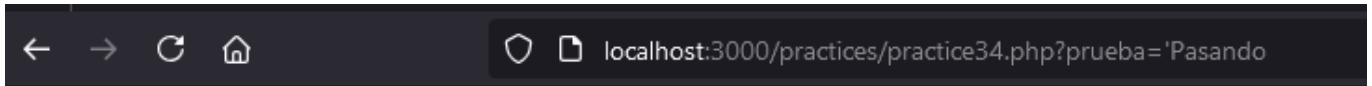
- Captura:





pasando datos

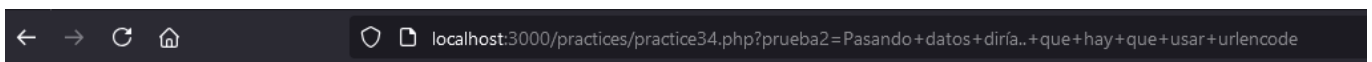
pasando datos



pasando datos

pasando datos

prueba: 'Pasando'



pasando datos

pasando datos

prueba2: Pasando datos diría.. que hay que usar urlencode

---

## Práctica 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)

Tras haber consultado en [php.net](https://www.php.net), he concluido que debemos de utilizar `is_numeric()` en vez de `is_int()` ya que esta última no identifica los string con contenido numérico como como son los inputs de los formularios y nunca se cumpliría la condición.

- Código:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
```

```
</head>
<body>
    <form action="practice35.php" method="post">
        <input type="text" id="num" name="num">
        <input type="submit" id="submitNum" name="submitNum" value="Send"> <!-- id
client, js || name server-->
    </form>

    <?php

        if (empty($_POST["num"]) || !is_numeric($_POST["num"])){
            echo "Please enter a valid integer num.";
            exit();
        }

        $numMulti = $_POST["num"];

        if ($numMulti < 1){
            echo "Num must be positive.";
            exit();
        }

        echo "Recived last petition </br>";
        echo "Num send: ". $numMulti. "</br>";

        function multiplicationTables($num) {
            for ($i = 1; $i <= 10; $i++) {
                $result = $num * $i;
                echo "$num * $i = ". $result. "</br>";
            }
        }

        echo "</br>";
        multiplicationTables($numMulti);
    ?>
</body>
</html>
```

- Captura:

←

→

↻

🏠

localhost:3000/unit-01/php-dossier/practices/practice35.php

Send

Please enter a valid integer num.

←

→

↻

🏠

localhost:3000/unit-01/php-dossier/practices/practice35.php

Send

Num must be positive.

←

→

↻

🏠

localhost:3000/unit-01/php-dossier/practices/practice35.php

Send

Recived last petition

Num send: 5

5 \* 1 = 5

5 \* 2 = 10

5 \* 3 = 15

5 \* 4 = 20

5 \* 5 = 25

5 \* 6 = 30

5 \* 7 = 35

5 \* 8 = 40

5 \* 9 = 45

5 \* 10 = 50

---

## Práctica 36

📁 Realizar una página con un formulario que se llame a si misma donde el usuario introduzca en un input una cadena de números separada por espacios ( ej: 2 5 8 7 3 4 ) y muestre un número por línea, mostrando primero los números impares y luego los pares. ( hacer uso de la función usort() y de la función explode() )

- Código:

```
<form action=<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?> method="post">
  <input type="text" id="num" name="num"/>
```

```
<input type="submit" id="submit" name="submit" value="Send"/>
</form>

<?php
if(!isset($_POST["num"]) || empty($_POST["num"])) {
    exit();
}

$numUser = $_POST["num"];
$numArray = explode(" ", $numUser); // same as java split

foreach ($numArray as $num) {
    if (!is_numeric($num)) {
        echo "Error: All values must be numbers.";
        exit();
    }
}

usort($numArray, function($a, $b) {
    if (!($a % 2 == 0) && ($b % 2 == 0)) {
        return -1;
    } elseif (($a % 2 == 0) && !($b % 2 == 0)) {
        return 1;
    } else {
        return 0;
    }
});

echo "<p> Sorted array: </p>";
foreach($numArray as $num) {
    echo $num. "<br></br> ";
}

?>
```

Si enviamos valores como: "1 a 2" tendremos un error ya que todos los elementos deben de ser numéricos.

- Captura:

← → ↻ 🏠

🛡️ 📄 localhost:3000/practices/practice36.php

Send

Sorted array:

1

43

2

Send

Error: All values must be numbers.

---

## Práctica 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

- Código:

```
<?php
    if (!empty($_POST)) {
        $name = $_POST["name"];
        $email = $_POST["email"];
        $gender = $_POST["gender"];

        if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
            echo "<p>Correo: Dato ingresado correctamente</p><br>";
        } else {
            echo "<p style='color: red'>Correo: El correo es invalido </p><br>";
        }

        if (!empty($name)) {
            echo "<p>Nombre: Dato ingresado correctamente</p><br>";
        } else {
            echo "<p style='color: red'>Nombre: El nombre no ha sido
```

```
introducido</p><br>";
    }

    if (!empty($gender)) {
        echo "<p>Género: Dato ingresado correctamente</p><br>";
    } else {
        echo "<p style=\"color: red\">Género: El género no ha sido
seleccionado</p><br>";
    }
}
?>
```

- Captura:



Nombre:  \*

Correo:  \*

Página web:

Comentario:

Género: ☐ Masculino ☐ Femenino ☐ Otro \*

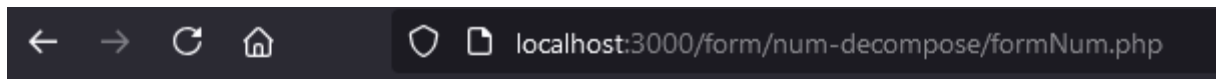
## Formulario - Descomponer Número

- Código:

```
<form action="formNum.php" method="post">
    <input type="text" id="num" name="num">
    <input type="submit" id="submitNum" name="submitNum" value="Send"> <!-- id
client, js || name server-->
</form>
<?php
    if (!isset($_POST["num"])){
        exit();
```

```
}  
echo "recived last petition";  
echo "</br>";  
echo "Num send: ". $_POST["num"]. "</br>";  
  
function decompositionNum($num) {  
    $actual = 0;  
    $operator = 1;  
    $result = "";  
  
    for ($i = 1; $i <= $num; $num/=10) {  
        $actual = $num % 10;  
        $result .= $actual . " * " . $operator;  
        if ($num > 10) {  
            $result .= " + ";  
        }  
        $operator*=10;  
    }  
    echo $result;  
}  
echo "</br>";  
decompositionNum($_POST["num"]);  
?>
```

- Captura:



recived last petition  
Num send: 1234

$4 * 1 + 3 * 10 + 2 * 100 + 1 * 1000$

---

## Formulario - Tablas

- Código:

```
<form action="./tables.php" method="post">  
    <input type="text" id="num" name="num">  
    <input type="submit" id="submitNum" name="submitNum" value="Send">
```

```
</form>
<?php
    if (empty($_POST["num"]) || !is_numeric($_POST["num"])){
        echo "Please enter a valid integer num.";
        exit();
    }

    $numMulti = $_POST["num"];

    if ($numMulti < 1){
        echo "Num must be positive.";
        exit();
    }

    echo "Recived last petition </br>";
    echo "Num send: ". $numMulti. "</br>";

    function multiplicationTables($num) {
        for ($i = 1; $i <= 10; $i++) {
            $result = $num * $i;
            echo "$num * $i = ". $result. "</br>";
        }
    }

    echo "</br>";
    multiplicationTables($numMulti);
?>
```

- Captura:





localhost:3000/form/tables/tables.php

Recived last petition

Num send: 1

$$1 * 1 = 1$$

$$1 * 2 = 2$$

$$1 * 3 = 3$$

$$1 * 4 = 4$$

$$1 * 5 = 5$$

$$1 * 6 = 6$$

$$1 * 7 = 7$$

$$1 * 8 = 8$$

$$1 * 9 = 9$$

$$1 * 10 = 10$$