



DIPLOMADO: PROGRAMACIÓN EN PHP

FUNCIONES

Una función es un bloque de declaraciones de PHP que se pueden usar repetidamente en un programa. Las funciones no se ejecutan inmediatamente cuando se carga una página, estas se ejecutan al momento de realizarse una llamada a la misma. Algunas de las características que ofrecen las funciones son:

- Todas las funciones tienen un ámbito global.
- Siempre comienza con la palabra reservada `function`.
- Permiten reutilizar bloques de código.
- Las funciones deben estar definidas antes de realizar la llamada a la función (como es lógico).
- Para llamar a una función se utiliza simplemente su nombre y los paréntesis de apertura y cierre. Además, si esta recibe argumentos, se deben adjuntar.
- El nombre de una función puede comenzar con una letra o un guión bajo (no un número).
- Se debe asignar a la función un nombre que refleje lo que hace la función.
- Los nombres de funciones NO distinguen entre mayúsculas y minúsculas.
- Las funciones pueden o no recibir argumentos.
- Pueden retornar valores.

La estructura de una función es la siguiente:

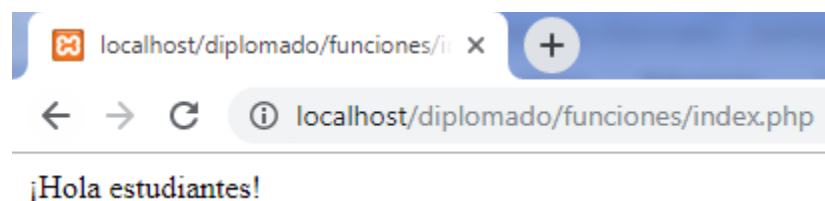
```
function functionName() {  
    code to be executed;  
}
```

En el siguiente ejemplo, creamos una función llamada "saludo()". La llave de apertura ({) indica el comienzo del código de función, y la llave de cierre (}) indica el final de la función. La función emite "¡Hola estudiantes!". Para llamar a la función, simplemente escriba su nombre seguido de paréntesis ():

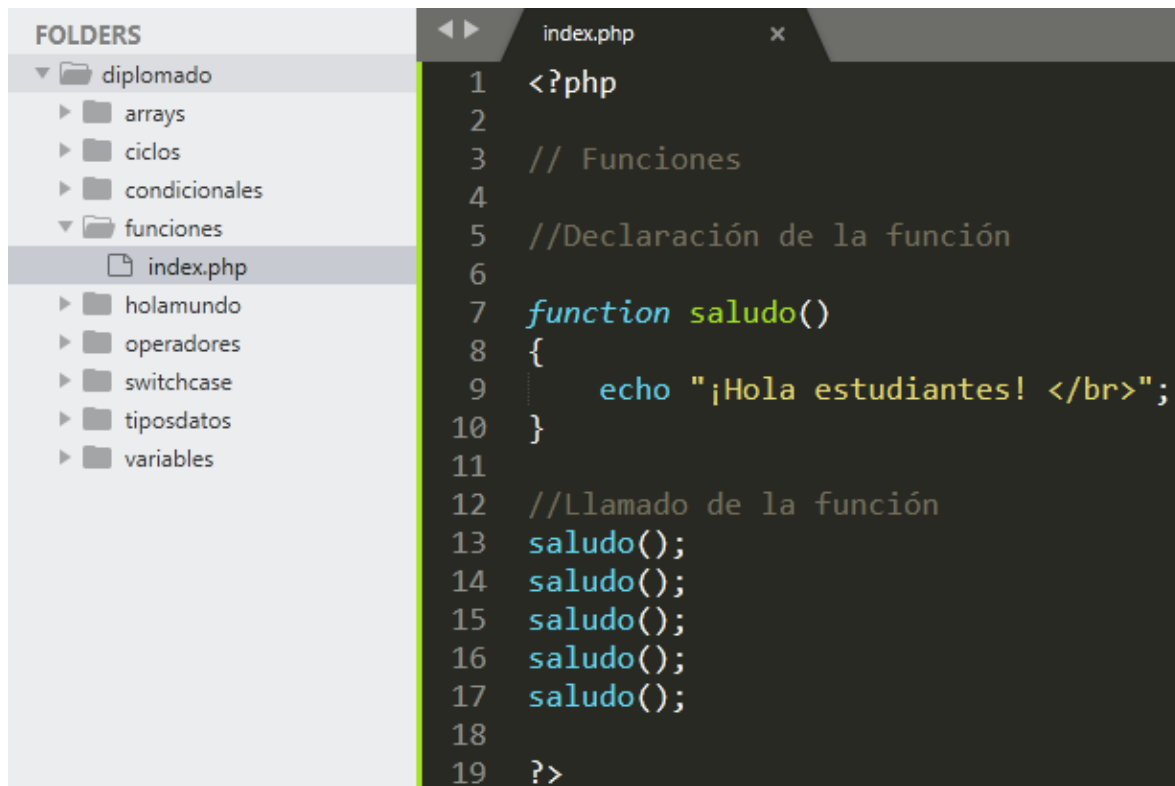


The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a folder structure with 'funciones' selected, containing 'index.php'. The code editor shows the following PHP code:

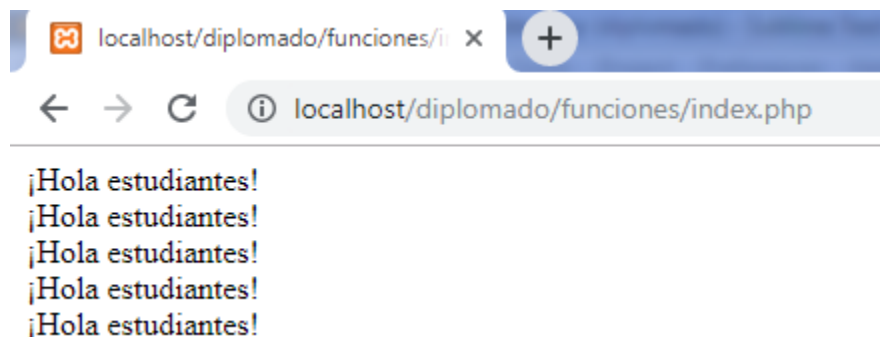
```
1 <?php  
2  
3 // Funciones  
4  
5 //Declaración de la función  
6  
7 function saludo()  
8 {  
9     echo "¡Hola estudiantes!";  
10 }  
11  
12 //Llamado de la función  
13 saludo();  
14  
15 ?>
```



Una de las principales cualidades y características de las funciones es la reutilización de código, veamos en el ejemplo anterior, como realizar varios llamados a la función para mostrar en 5 ocasiones, "Hola estudiantes".



```
1 <?php
2
3 // Funciones
4
5 //Declaración de la función
6
7 function saludo()
8 {
9     echo "¡Hola estudiantes! <br>";
10 }
11
12 //Llamado de la función
13 saludo();
14 saludo();
15 saludo();
16 saludo();
17 saludo();
18
19 ?>
```



En este simple ejercicio, ahorramos el trabajo de escribir en 5 ocasiones **echo "¡Hola estudiantes!
"**; es un simple proceso en el que ahorra una línea de código, pero supongamos el caso de un proceso algo más complejo.

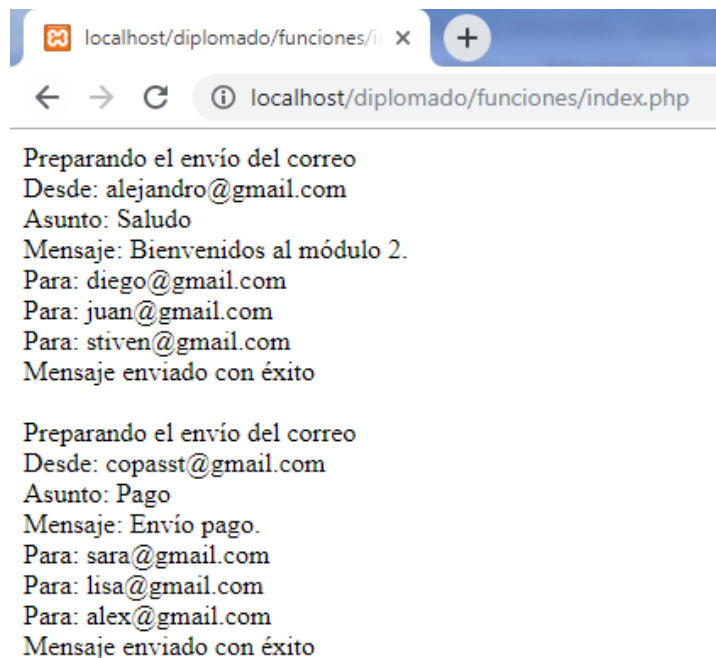
Suponga que debe enviar un correo a x cantidad de personas con una cantidad de información determinada como: correo desde el que se envía, los destinatarios, asunto y mensaje.

```

FOLDERS
└─ diplomado
  └─ arrays
  └─ ciclos
  └─ condicionales
  └─ funciones
    └─ index.php
  └─ holamundo
  └─ operadores
  └─ switchcase
  └─ tiposdatos
  └─ variables

index.php
1  <?php
2
3  // Funciones
4
5  //Declaración de la función
6
7  function enviarMail($desde, $destinatarios, $asunto, $mensaje)
8  {
9      echo "Preparando el envío del correo <br>";
10     echo "Desde: " . $desde . " <br>";
11     echo "Asunto: " . $asunto . " <br>";
12     echo "Mensaje: " . $mensaje . " <br>";
13
14     foreach ($destinatarios as $destinatario)
15     {
16         echo "Para: " . $destinatario . "<br>";
17     }
18
19     echo "Mensaje enviado con éxito <br>";
20     echo "<br>";
21 }
22
23
24 $destinatarios = array("diego@gmail.com", "juan@gmail.com", "stiven@gmail.com");
25
26 //Llamado 1 a la función
27
28 enviarMail("alejandro@gmail.com", $destinatarios, "Saludo", "Bienvenidos al módulo 2.");
29
30 $destinatarios = array("sara@gmail.com", "lisa@gmail.com", "alex@gmail.com");
31
32 //Llamado 2 a la función
33
34 enviarMail("copasst@gmail.com", $destinatarios, "Pago", "Envío pago.");
35
36
37 ?>

```



En este caso sí se evidencia de mejor forma el funcionamiento y la característica principal de las funciones.

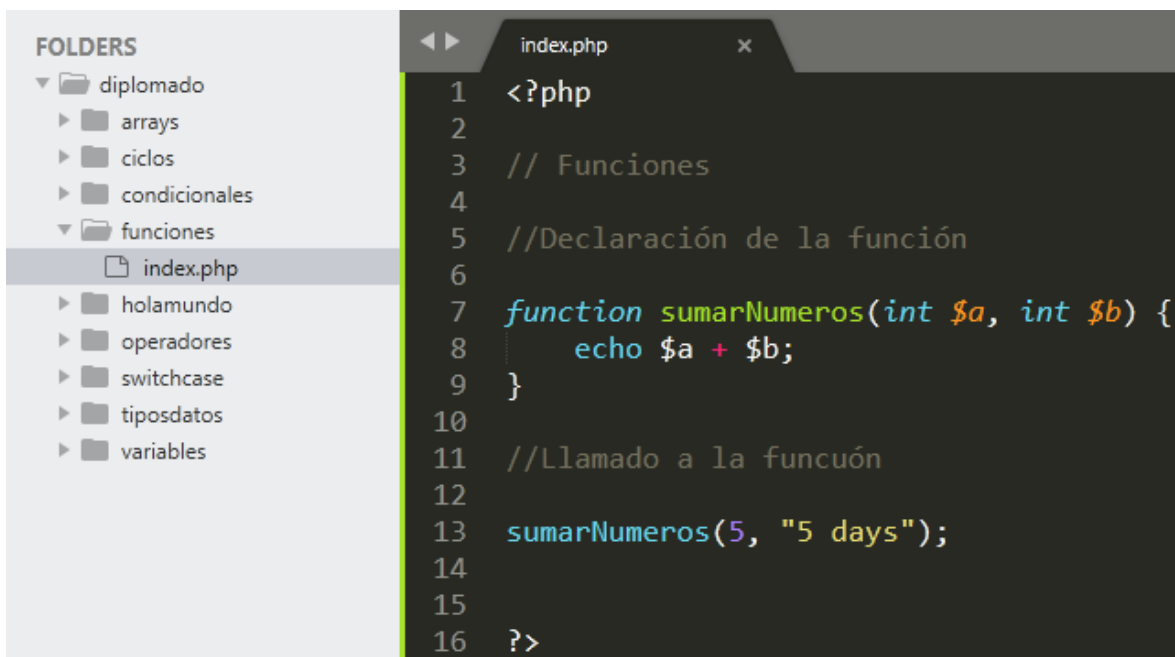
En el ejemplo anterior, observe que no teníamos que decirle a PHP qué tipo de datos es el argumento que se debe recibir en la función.

Tipos de datos en funciones

PHP asocia automáticamente un tipo de dato a la variable, dependiendo de su valor. Dado que los tipos de datos no están establecidos en un sentido estricto, puede hacer cosas como agregar una cadena a un entero sin causar un error.

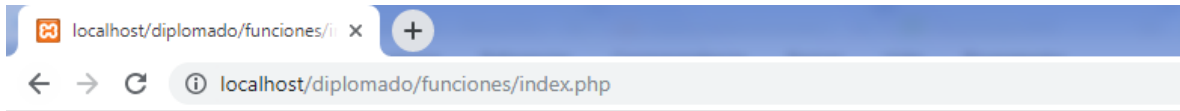
En PHP 7, se agregaron declaraciones de tipo. Esto nos da una opción para especificar el tipo de datos esperado al declarar una función, y al habilitar el requisito estricto, arrojará un "Error fatal" en una falta de coincidencia de tipos.

En el siguiente ejemplo intentamos agregar un número y una cadena sin el requisito estricto:



```
1 <?php
2
3 // Funciones
4
5 //Declaración de la función
6
7 function sumarNumeros(int $a, int $b) {
8     echo $a + $b;
9 }
10
11 //Llamado a la función
12
13 sumarNumeros(5, "5 days");
14
15
16 ?>
```

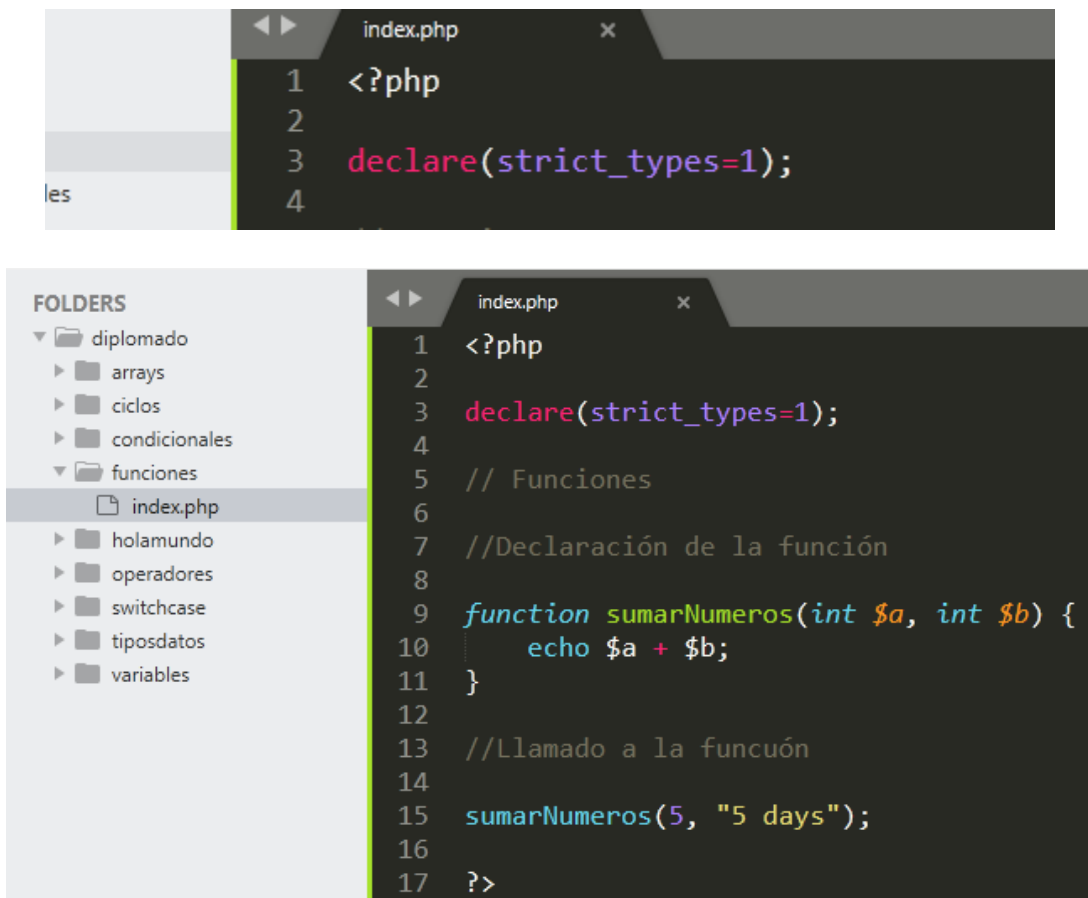
En este ejemplo, en la declaración de la función, definimos los tipos de datos que se debe recibir para la misma (2 enteros en este caso), al no tener el requisito estricto de datos, PHP toma el dato "5 days" y lo convierte automáticamente en 5 y realiza la suma arrojando el siguiente resultado:



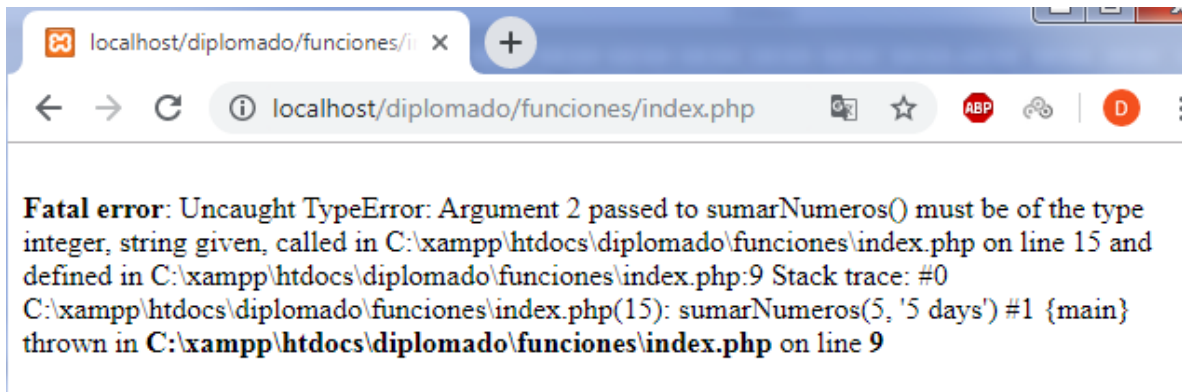
Primero como se espera, viene una notificación con el siguiente mensaje: “un valor numérico no está bien formateado”. Haciendo alusión a “5 days”. Pero la función realiza la suma y arroja el resultado.

En el siguiente ejemplo intentamos agregar un número y una cadena con el requisito de estricto:

Para habilitar el requisito de tipos estricto, simplemente se debe agregar la siguiente línea de código en el encabezado el archivo.



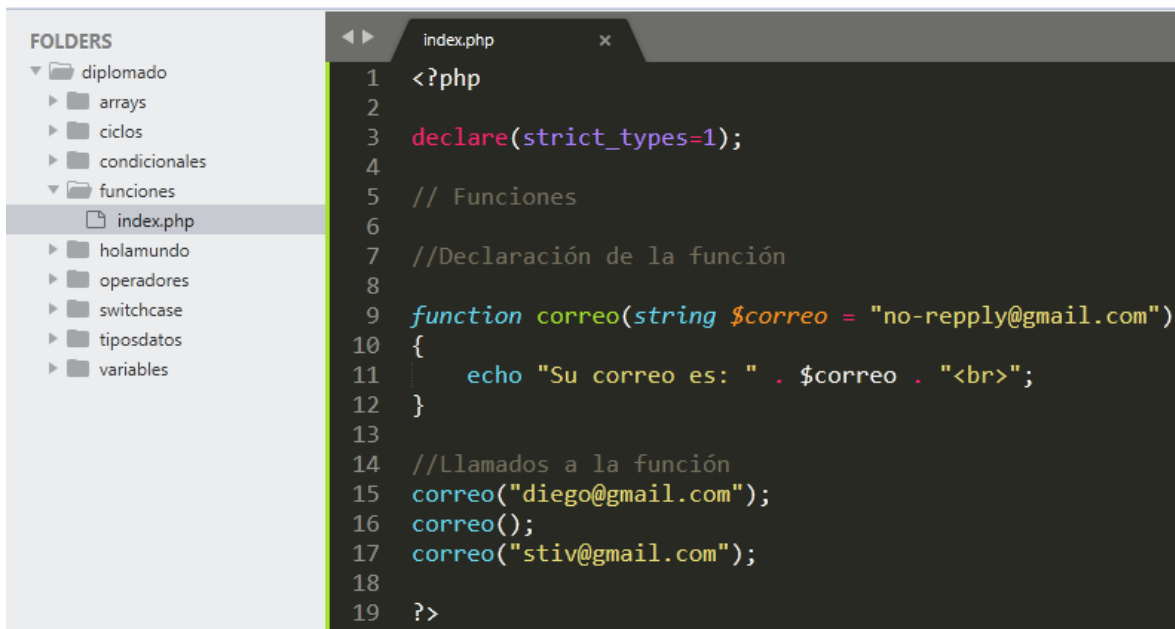
```
1 <?php
2
3 declare(strict_types=1);
4
5 // Funciones
6
7 //Declaración de la función
8
9 function sumarNumeros(int $a, int $b) {
10     echo $a + $b;
11 }
12
13 //Llamado a la función
14
15 sumarNumeros(5, "5 days");
16
17 ?>
```



Con el modo estricto activado, ya no se obtiene una notificación como en el ejemplo anterior, ahora el resultado es un error de TIPO, donde se tiene un choque entre un String y un Integer en la función `sumarNumeros` y el argumento "5 days".

Valores predeterminados en argumentos

PHP permite definir valores por defecto a los argumentos en caso de no recibir ningún tipo de dato, por ejemplo:

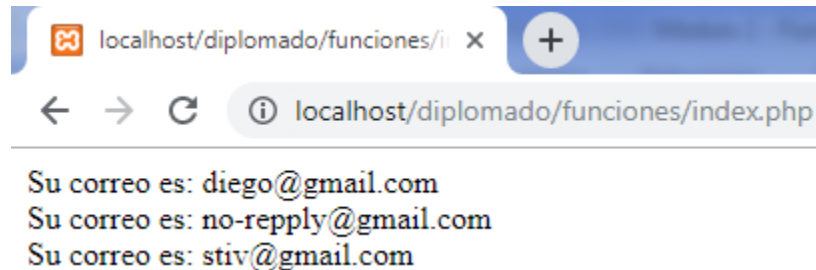


En este básico ejemplo, la función `correo` recibe un email y simplemente muestra un mensaje con el respectivo email definido por los parámetros, pero la particularidad de los valores predeterminados es que, en caso de que no se

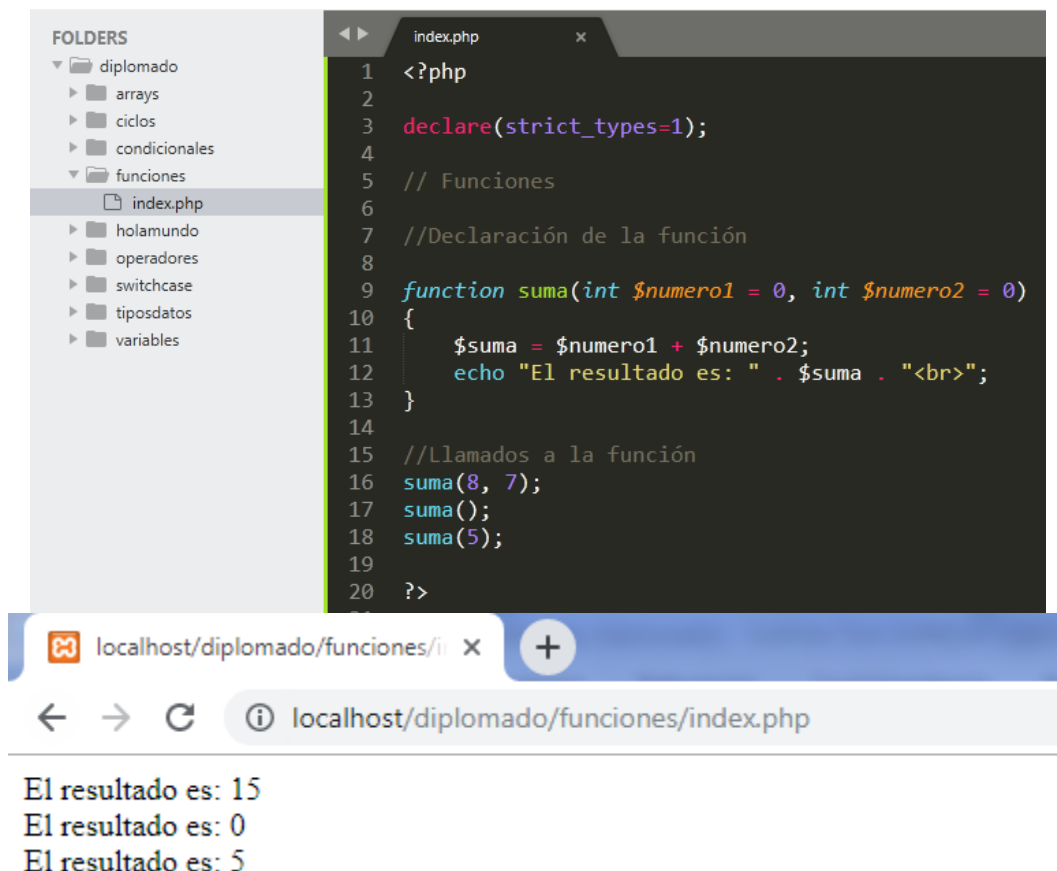
envíe un argumento, en este caso un email, en la declaración de la función se define un valor predeterminado para esos casos.

```
correo(string $correo = "no-reply@gmail.com")
```

Y el resultado de la función será el siguiente para los tres llamados respectivamente:

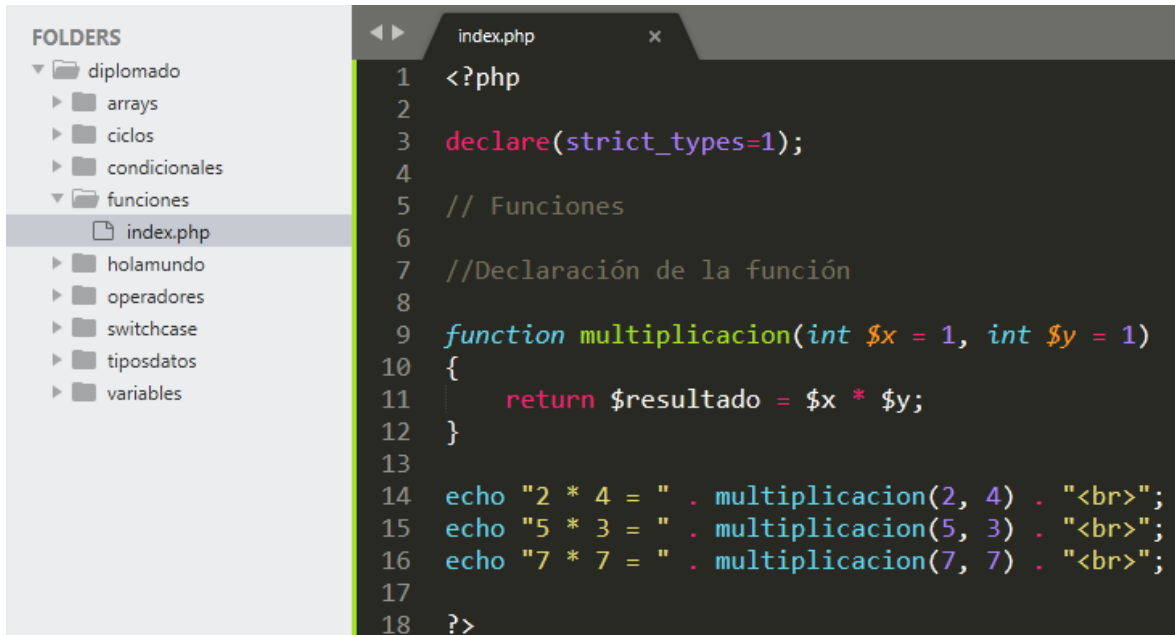


Otro ejemplo de la implementación de los valores predeterminados es el siguiente donde la función sumar recibe dos números y muestra el resultado de la suma de estos:

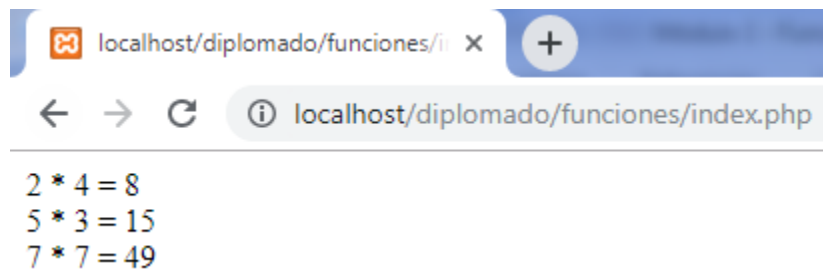


Retorno de valores en funciones

Las funciones en PHP no solo permiten recibir argumentos, definir tipos de datos o predeterminedar valores para estos, además ofrece la característica de retornar valores en caso de que se ocupe, para retornar valores, simplemente hay que hacer uso de la palabra reservada del lenguaje **return**.

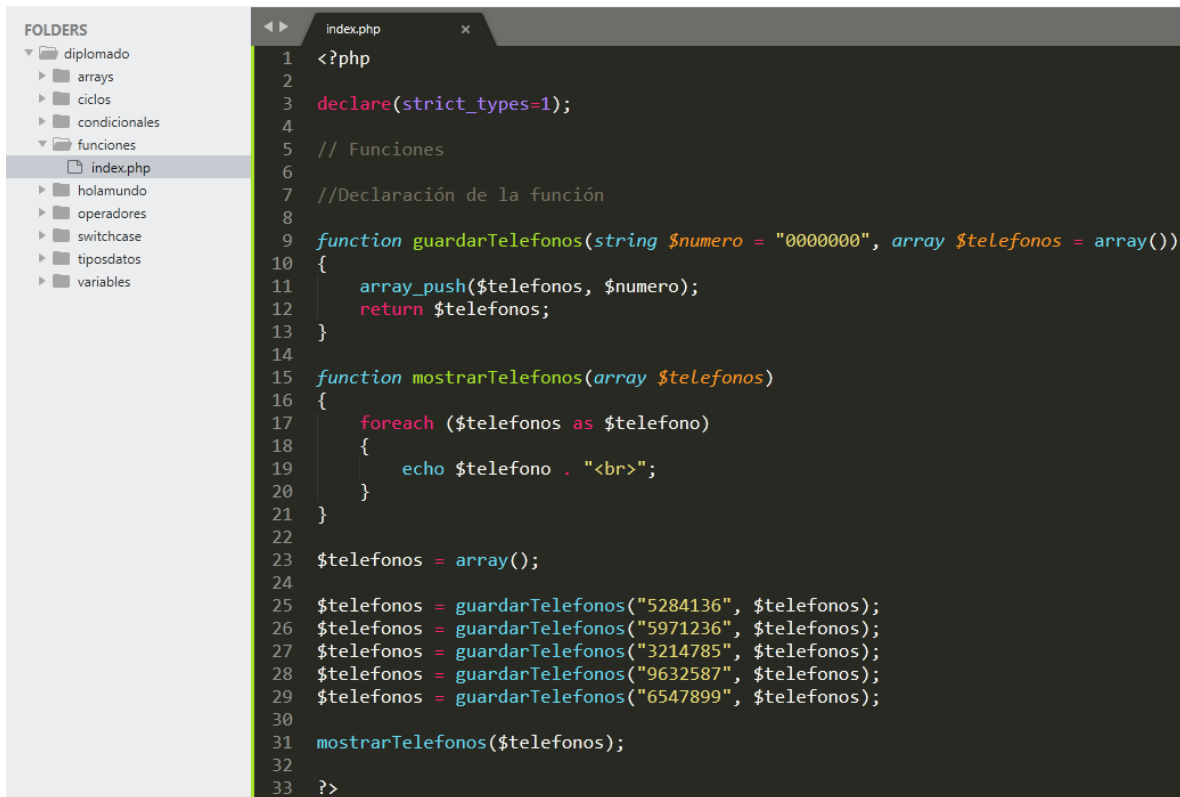


```
1 <?php
2
3 declare(strict_types=1);
4
5 // Funciones
6
7 //Declaración de la función
8
9 function multiplicacion(int $x = 1, int $y = 1)
10 {
11     return $resultado = $x * $y;
12 }
13
14 echo "2 * 4 = " . multiplicacion(2, 4) . "<br>";
15 echo "5 * 3 = " . multiplicacion(5, 3) . "<br>";
16 echo "7 * 7 = " . multiplicacion(7, 7) . "<br>";
17
18 ?>
```

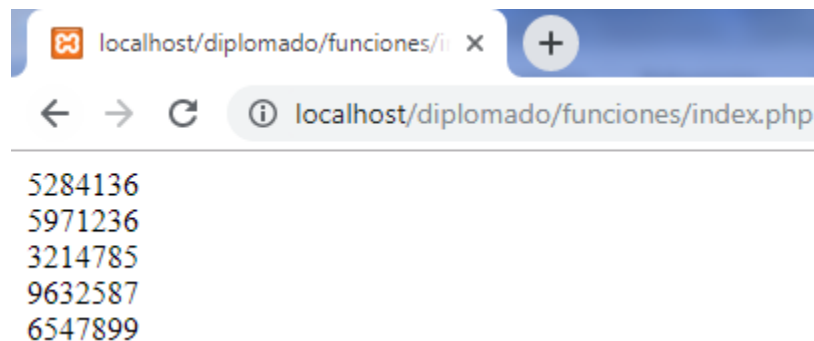


Otros ejemplos de retornos en PHP con otras operaciones y retornos serían los siguientes.

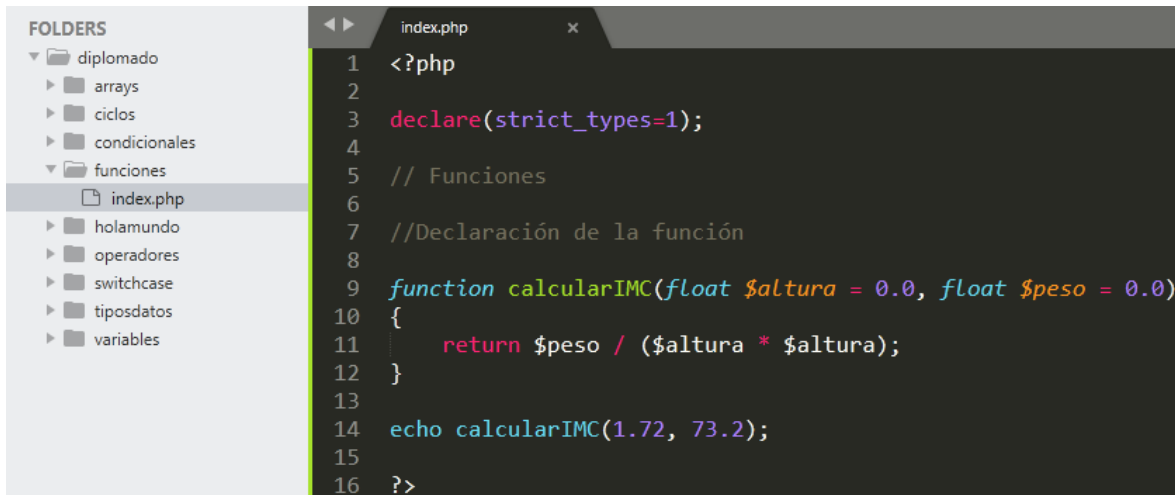
En el siguiente ejemplo vamos a crear una especie de agenda de teléfonos, con dos funciones, `guardarTelefonos` y `mostrarTelefonos`, la forma de almacenar los teléfonos será por medio de un array pasando como argumentos el número del teléfono y el array en cada llamado; la forma de retorno de la función `guardar`, será retornar el array para volver a pasarlo en la siguiente llamada y finalmente se mostrarán los datos por medio del `foreach` en la función `mostrarTelefonos` que recibe como argumento simplemente el array.



```
1 <?php
2
3 declare(strict_types=1);
4
5 // Funciones
6
7 //Declaración de la función
8
9 function guardarTelefonos(string $numero = "0000000", array $telefonos = array())
10 {
11     array_push($telefonos, $numero);
12     return $telefonos;
13 }
14
15 function mostrarTelefonos(array $telefonos)
16 {
17     foreach ($telefonos as $telefono)
18     {
19         echo $telefono . "<br>";
20     }
21 }
22
23 $telefonos = array();
24
25 $telefonos = guardarTelefonos("5284136", $telefonos);
26 $telefonos = guardarTelefonos("5971236", $telefonos);
27 $telefonos = guardarTelefonos("3214785", $telefonos);
28 $telefonos = guardarTelefonos("9632587", $telefonos);
29 $telefonos = guardarTelefonos("6547899", $telefonos);
30
31 mostrarTelefonos($telefonos);
32
33 ?>
```

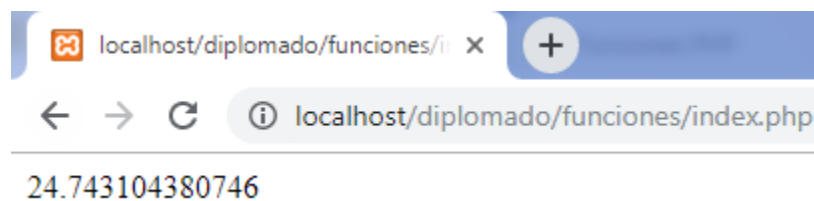


El índice de masa corporal de una persona se calcula a partir del peso (kg) y la altura (cm) diviniendo este primero por el cuadrado de la altura. Simplemente utilizaremos una función que reciba dos argumentos float's (altura y peso) y realizaremos la operación en el retorno.



```
FOLDERS
└─ diplomado
  └─ arrays
  └─ ciclos
  └─ condicionales
  └─ funciones
    └─ index.php
  └─ holamundo
  └─ operadores
  └─ switchcase
  └─ tiposdatos
  └─ variables

index.php
1 <?php
2
3 declare(strict_types=1);
4
5 // Funciones
6
7 //Declaración de la función
8
9 function calcularIMC(float $altura = 0.0, float $peso = 0.0)
10 {
11     return $peso / ($altura * $altura);
12 }
13
14 echo calcularIMC(1.72, 73.2);
15
16 ?>
```

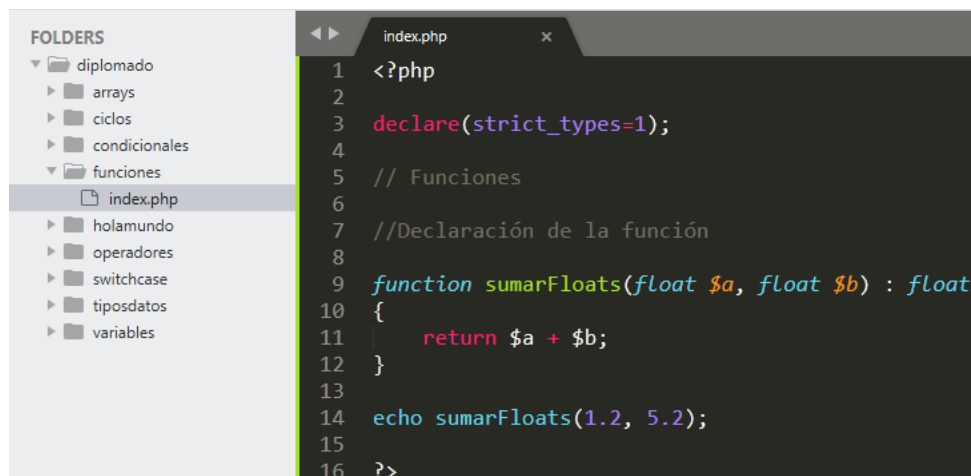


Retorno estricto por tipos

PHP 7 también admite declaraciones de tipo para el retorno de una función. Al igual que con la declaración de tipo para argumentos de función, al habilitar el requisito estricto, arrojará un "Error fatal" en una discrepancia de tipo.

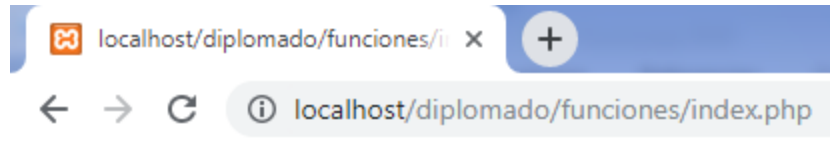
Para declarar un tipo para el retorno de la función, agregue dos puntos : y el tipo justo antes del {corchete de apertura () al declarar la función.

En el siguiente ejemplo, especificamos el tipo de retorno para la función:



```
FOLDERS
└─ diplomado
  └─ arrays
  └─ ciclos
  └─ condicionales
  └─ funciones
    └─ index.php
  └─ holamundo
  └─ operadores
  └─ switchcase
  └─ tiposdatos
  └─ variables

index.php
1 <?php
2
3 declare(strict_types=1);
4
5 // Funciones
6
7 //Declaración de la función
8
9 function sumarFloats(float $a, float $b) : float
10 {
11     return $a + $b;
12 }
13
14 echo sumarFloats(1.2, 5.2);
15
16 ?>
```



6.4

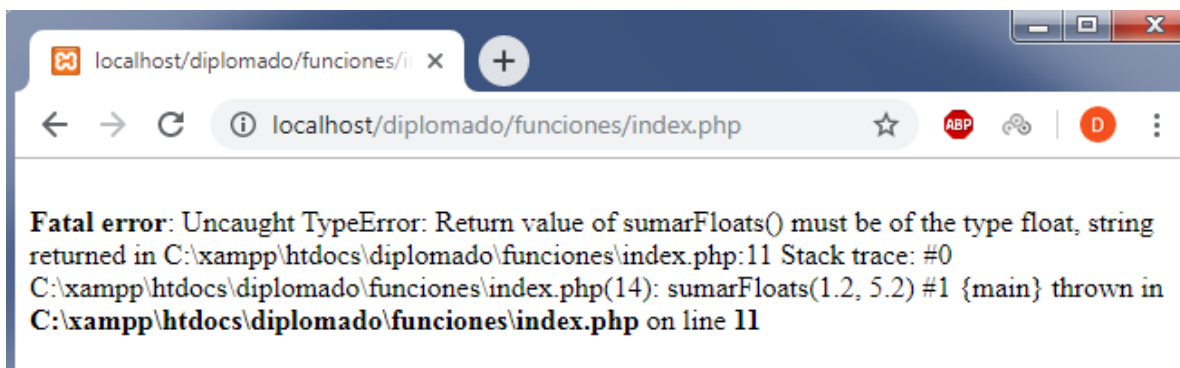
De esta forma, por medio de la declaración estricta float:

```
sumarFloats(float $a, float $b) : float
```

obligamos a PHP y a la función sumar floats, que el resultado a retornar debe ser ÚNICAMENTE un valor float y no otro. Veamos que ocurre en caso de retonar otro valor.

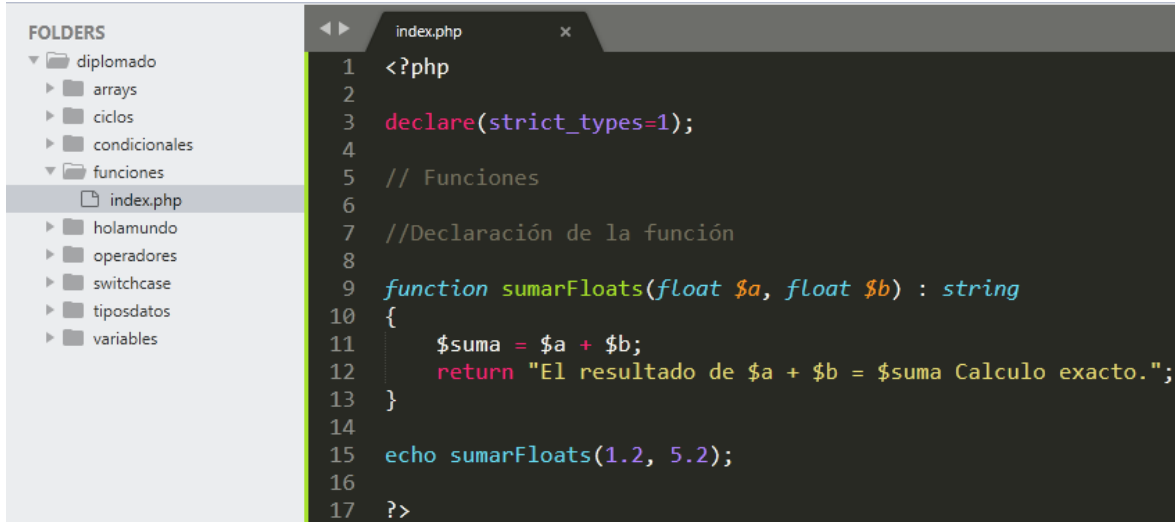
```
function sumarFloats(float $a, float $b) : float
{
    return "Brute Force";
}

echo sumarFloats(1.2, 5.2);
```

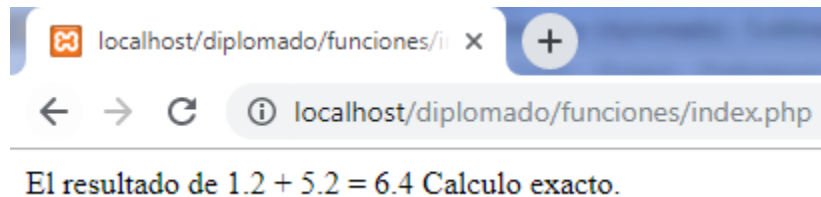


El error es evidente, el definir el tipo float estricto para el retorno de la función, al retornar "Brute force" el error de Tipo y Retonar se disparará.

Otra característica fundamental a tener en cuenta es, los retornos no deben ser necesariamente del tipo de los argumentos, el retorno lo establece el tipo estricto de retorno, para el ejemplo anterior, los argumentos son floats y el retorno float, pero fácil mente podría ser una cadena de texto. Veamos.



```
1 <?php
2
3 declare(strict_types=1);
4
5 // Funciones
6
7 //Declaración de la función
8
9 function sumarFloats(float $a, float $b) : string
10 {
11     $suma = $a + $b;
12     return "El resultado de $a + $b = $suma Calculo exacto.";
13 }
14
15 echo sumarFloats(1.2, 5.2);
16
17 ?>
```



Recursos disponibles para el aprendizaje



Las funciones juegan un papel clave como concepto en la programación con PHP, en la documentación del lenguaje hay información muy importante y que te puede ser útil, visítala, Disponible en:
<https://www.php.net/manual/es/language.functions.php>

Ejercicio

¿Deseas profundizar en la temática de las funciones? Entonces te sugiero realizar los siguientes ejercicios que pondrán a prueba los conocimientos adquiridos.
(MÓDULO 2 – EJERCICIOS DE FUNCIONES)
¡Inténtalo! 👍



¡Felicidades! 🍀 Recuerda que si tienes una duda o dificultad puedes escribirme: diegovalencia@politecnicodecolombia.edu.co.