

Funciones

Las funciones forman parte de las herramientas más utilizadas dentro de los lenguajes de programación para poder tener un código bien estructurado. **Son un conjunto de instrucciones que, a lo largo de la ejecución del programa, se van a poder invocar todas las veces que las necesitemos.**

Una de sus ventajas es que pueden ser invocadas desde cualquier parte del programa (siempre y cuando se haya incluido al archivo que las contiene).

Cuando se invoca una función, es probable que esta requiera de parámetros, ya que de ellos va a depender el resultado que nos retorne al finalizar su ejecución.

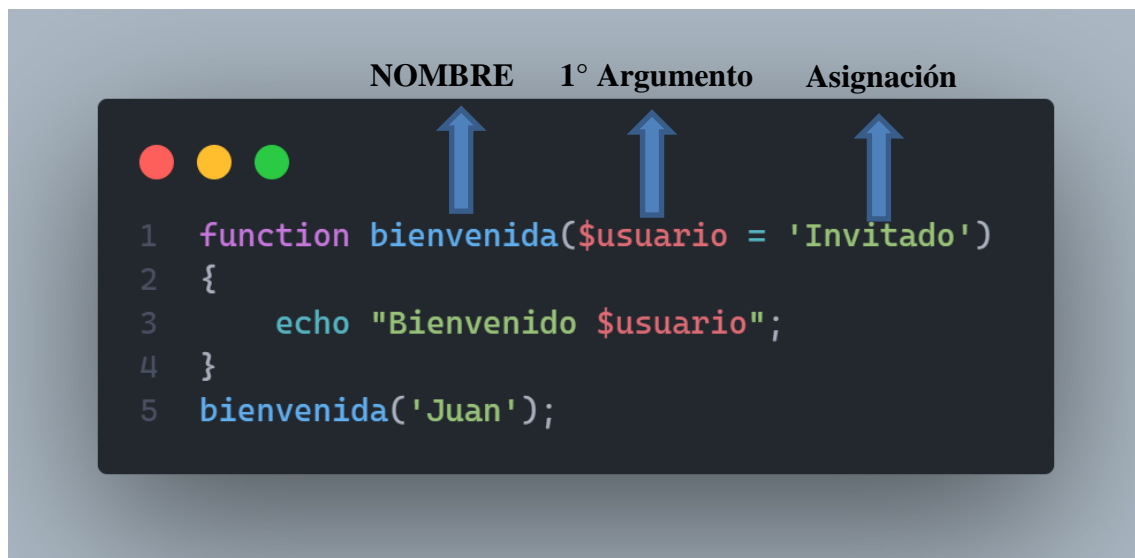


Fig. 1

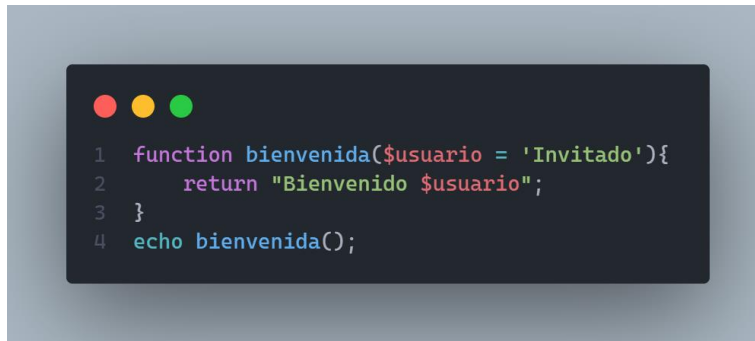
Como se observa en la fig. 1, la sintaxis para invocar una función es llamándola por su nombre (línea 5).

En este ejemplo, definimos una función cuyo nombre es bienvenida. Esta función espera un parámetro (no obligatorio, que lo hemos llamado \$usuario) cuando es invocada. Una vez que se invoque, la función se ejecuta sus instrucciones y devuelve el control al punto desde que se la fue invocada, como resultado nos escribirá en el navegador, “Bienvenido Juan”.

“Juan” (línea 5) es un parámetro que le estamos pasando a la función y que será guardado en la variable \$usuario.

La línea 1, contiene entre paréntesis los parámetros (tantos como necesitamos) que la función espera recibir. En este ejemplo, no sería obligatorio pasarle algún valor ya que estamos inicializando a la variable \$usuario con el texto ‘Invitado’. Es decir que, si la invocamos de esta forma bienvenida(); nos mostraría en pantalla “Bienvenido Invitado”.

Existen dos tipos de funciones, las que retornan valores, variables u objetos **y las que no**. El primer ejemplo que vimos, sería del segundo tipo ya que solo al llamarla, lo único que hace es recibir un valor y escribirlo en pantalla.



```

1 function bienvenida($usuario = 'Invitado'){
2     return "Bienvenido $usuario";
3 }
4 echo bienvenida();

```

Fig. 2

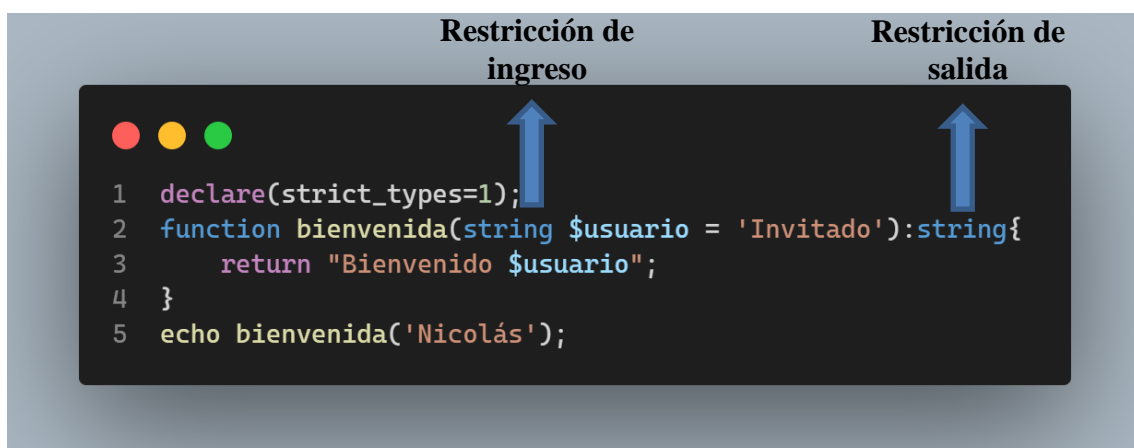
Realizando algunas modificaciones la transformamos a las del primer tipo utilizando la estructura de control **return** y son las más comunes de encontrar.

Este tipo de funciones, que son creadas por necesidad del desarrollador se las

llama “*funciones definidas por el usuario*”, porque como ya hemos visto anteriormente, también existen las que vienen establecidas por PHP como `date()`, `rand()`, `array_push()`...

Sería ilógico que en pantalla nos muestre “Bienvenido 1” con un valor numérico, pero puede ocurrir en este caso porque la función no está limitada. Para este caso, existen las “*funciones con tipado definido*” a las cuales les podemos restringir los parámetros de ingreso como así también los de salida.

Trabajando con el mismo caso de la fig. 2, la vamos a restringir para que solo reciba como parámetro un string y nos retorne un valor del mismo tipo.



Restricción de ingreso **Restricción de salida**

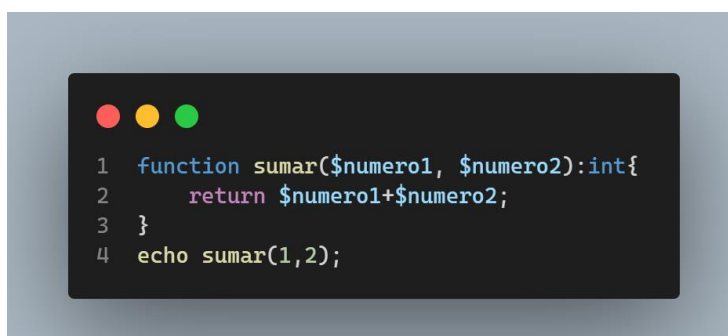
```

1 declare(strict_types=1);
2 function bienvenida(string $usuario = 'Invitado'):string{
3     return "Bienvenido $usuario";
4 }
5 echo bienvenida('Nicolás');

```

Fig. 3

Así como recibimos textos en variables, podríamos hacerlo con números y retornar un valor exacto, en el próximo ejemplo, generamos una función suma, que nos devuelve el resultado (en enteros) de la suma de dos valores pasados por parámetros.



```

1 function sumar($numero1, $numero2):int{
2     return $numero1+$numero2;
3 }
4 echo sumar(1,2);

```

Fig. 4

La inquietud surge cuando nos planteamos sumar más de dos valores, tranquilamente podríamos plantear en agrega otro parámetro más, ¿pero si yo quisiera que sirva para cualquier cantidad?

PHP nos permite recibir en un array todos los parámetros que quisiéramos y lo hacemos anteponiendo ... (tres puntos) al nombre de la expresión.

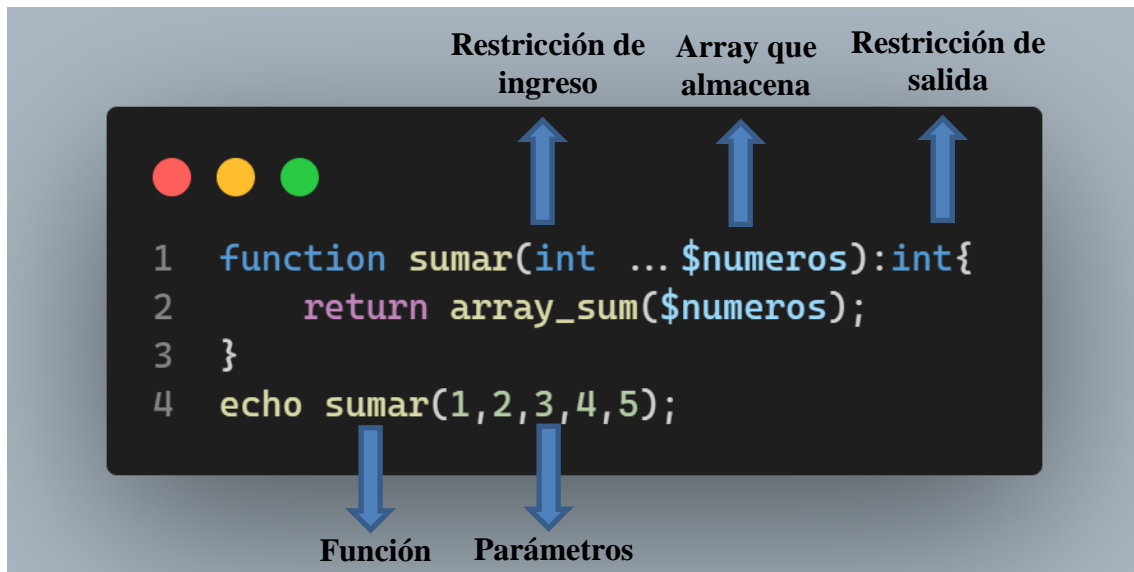


Fig. 5

En el ejemplo de la fig. 5, declaramos una función llamada sumar que solo va a recibir y retornar números enteros, con la particularidad de que puede recibir todos los valores que quisiéramos ya que los almacena en un array (a diferencia de los otros casos que lo guarda en una variable). Una vez invocada la función, se ejecutan las sentencias y nos devuelve la suma de todos los valores contenidos dentro del arreglo.

Generalmente todas las funciones definidas por el desarrollador se incluyen en un mismo archivo para tener acceso a todas ellas en todo momento, por convención podremos encontrarlas en un archivo con el nombre functions o helpers. No debemos olvidarnos de incluirlas dentro de los archivos donde las queremos utilizar con las estructuras include o require.

Funciones de PHP

Nombre	Expresiones	Descripción
<code>time();</code>	Sin parámetro	Devuelve fecha unix actual.
<code>date();</code> ¹	Un parámetro (string)	Devuelve fecha según formato indicado.
<code>sqrt();</code>	Un parámetro (número)	Devuelve la raíz cuadrada.
<code>rand(..., ...);</code>	Dos parámetros (número)	Devuelve un número aleatorio entre los valores indicados.
<code>pi();</code>	Sin parámetro	Devuelve el valor de π .
<code>strlen();</code>	Un parámetro (string)	Devuelve longitud de string.
<code>strcmp(..., ...);</code>	Dos parámetros (string)	Comparación de string, devuelve booleano.
<code>strtolower();</code>	Un parámetro (string)	Texto a minúsculas.
<code>strtoupper();</code>	Un parámetro (string)	Texto a mayúsculas.
<code>ucfirst();</code>	Un parámetro (string)	Primer carácter del string a mayúsculas.
<code>ucwords();</code>	Un parámetro (string)	Primer carácter de cada palabra a mayúsculas.
<code>str_contains(..., ...);</code>	Dos parámetros (variable y palabra)	Busca si la palabra está contenida en la variable.
<code>count();</code>	Un parámetro (array)	Cuenta valores del arreglo.
<code>array_push();</code>	Un parámetro (array)	Añadir valores al arreglo.
<code>sort();</code>	Un parámetro (array)	Ordena el arreglo en orden asc.
<code>rsort();</code>	Un parámetro (array)	Ordena el arreglo en orden desc.
<code>unset();</code>	Un parámetro, elemento de un array; o una variable.	Elimina el elemento indicado.
<code>isset();</code>	Un parámetro (variable)	Determina si una variable está definida y no es nula.
<code>var_dump();</code>	Un parámetro (variable)	Muestra el tipo y contenido de la variable.
<code>empty();</code>	Un parámetro (variable)	Determina si una variable está vacía.

¹ <https://www.php.net/manual/es/function.date.php>