## UD2-Lenguage PHP

**Índice:**

[UD2-Lenguage PHP 1](#_Toc144219399)

[1 Introducción 2](#_Toc144219400)

[2 Características del Lenguaje 2](#_Toc144219401)

[3 Generación de código HTML 3](#_Toc144219402)

[3.1 Echo y print 4](#_Toc144219403)

[3.2 Printf 4](#_Toc144219404)

[3.3 Especificadores de tipo para las funciones printf y sprintf 5](#_Toc144219405)

[3.4 Sprintf 5](#_Toc144219406)

[act.1) Práctica especificadores 6](#_Toc144219407)

[3.5 Cadenas de texto 6](#_Toc144219408)

[act.2) Cadenas 7](#_Toc144219409)

[4 Tipos de Datos. Funciones y Variables 7](#_Toc144219410)

[4.1 Isset, unset, isnull 8](#_Toc144219411)

[4.2 Evitar Warning 9](#_Toc144219412)

[4.3 Constantes (define) 9](#_Toc144219413)

[4.4 Fechas y Horas 9](#_Toc144219414)

[act.3) Zona, date y getdate 11](#_Toc144219415)

[4.5 Variables Especiales de PHP (superglobals) 11](#_Toc144219416)

[act.4) Variables Especiales 13](#_Toc144219417)

[5 Estructuras de Control (RA2. CE1) 14](#_Toc144219418)

[5.1 Condicionales 14](#_Toc144219419)

[5.2 Bucles 15](#_Toc144219420)

[6 Funciones 16](#_Toc144219421)

[6.1 PECL 16](#_Toc144219422)

[6.2 Inclusión de ficheros externos (include/require). 17](#_Toc144219423)

[act.5) Include 18](#_Toc144219424)

[6.3 Creación y Ejecución de funciones. (function) 18](#_Toc144219425)

[6.4 Argumentos 19](#_Toc144219426)

[7 Arrays 20](#_Toc144219427)

[7.1 Arrays multidimensionales 21](#_Toc144219428)

[7.2 Recorrer Arrays (foreach) 21](#_Toc144219429)

[act.6) Foreach para $\_SERVER 22](#_Toc144219430)

[7.3 Funciones para recorrer Arrays (next, prev…) 23](#_Toc144219431)

[7.4 Funciones datos compuestos (array, unset…) 24](#_Toc144219432)

[8 Formularios 25](#_Toc144219433)

[act.7) Formulario1.php 25](#_Toc144219434)

[8.2 Curso HTML 26](#_Toc144219435)

[8.3 Procesamiento información de formularios 26](#_Toc144219436)

[8.4 Validación de datos. Misma página 27](#_Toc144219437)

[8.5 Formularios Web en PHP 28](#_Toc144219438)

[9 Mapa Conceptual 30](#_Toc144219439)

[10 Temporalización 32](#_Toc144219440)

[11 Resultados de Aprendizaje 32](#_Toc144219441)

[12 Bibliografía 33](#_Toc144219442)

[13 Pendiente 33](#_Toc144219443)

### Introducción

En la unidad anterior, aprendiste a preparar un entorno para programar en PHP. Además, también viste algunos de los elementos que se usan en el lenguaje, como las variables y tipos de datos, comentarios, operadores y expresiones.

También sabes ya cómo se integran las etiquetas HTML con el código del lenguaje, utilizando los delimitadores **<?php** **y ?>.**

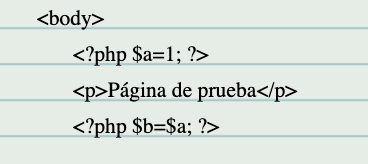
### Características del Lenguaje

**RA3**: Escribe bloques de sentencias embebidos en lenguajes de marcas, seleccionando y utilizando las estructuras de programación.

**C.Ev.2.** **A**: Se han reconocido los mecanismos de generación de páginas Web a partir de lenguajes de marcas con código embebido..

En esta unidad aprenderás a utilizar **otros elementos** del lenguaje que te permitan crear programas completos en PHP. Los programas escritos en PHP, además de encontrarse estructurados normalmente en **varias páginas** suelen incluir en una misma página **varios bloques de código**. Cada bloque de código debe ir entre **delimitadores**, y en caso de que genere alguna salida, ésta se introduce en el código HTML en el mismo punto en el que figuran las instrucciones en PHP.

Por ejemplo, en las siguientes líneas tenemos dos bloques de código en PHP:



Aunque no se utilice el valor de las variables, en el segundo bloque de código la variable **$a** mantiene el valor 1 que se le ha asignado anteriormente.

En esta unidad empezarás a crear tus propios programas en PHP. Deberías organizar tus programas en proyectos, y almacenarlos en una estructura en árbol, colgando todos en la carpeta **/home/usuario/DWES**.

### Generación de código HTML

**RA2**: Escribe sentencias ejecutables por un servidor Web reconociendo y aplicando procedimientos de integración del código en lenguajes de marcas.

**C.Ev.** **2.B**: Se han identificado las principales tecnologías asociadas.

**C.Ev.** **2.C** Se han utilizado etiquetas para la inclusión de código en el lenguaje de marcas.

Existen varias formas incluir contenido en la página web a partir del resultado de la ejecución de código PHP. La forma más sencilla es usando **echo**, que no devuelve nada (**void**), y genera como salida el texto de los parámetros que recibe.



#### Echo y print

Otra posibilidad es **print**, que funciona de forma similar. La diferencia más importante entre **print** y **echo**, es que print sólo puede recibir un parámetro y devuelve siempre 1.

int print (string $arg);

Tanto **print** como **echo** no son realmente funciones, por lo que no es obligatorio que pongas paréntesis cuando las utilices. Por ejemplo, el código del siguiente documento puede hacerse igualmente utilizando **echo**.

****

#### Printf

**printf** es otra opción para generar una salida desde PHP. Puede recibir varios parámetros, el primero de los cuales es siempre una cadena de texto que indica el **formato** que se ha de aplicar. Esa cadena debe contener un especificador de **conversión** por cada uno de los demás parámetros que se le pasen a la función, y en el mismo orden en que figuran en la lista de parámetros. Por ejemplo:



Cada especificador de conversión va precedido del caracter **%** y se compone de las siguientes partes:

**signo** (opcional). Indica si se pone signo a los número negativos (por defecto) o también a los positivos (se indica con un signo +).

**relleno** (opcional). Indica que carácter se usará para ajustar el tamaño de una cadena. Las opciones son el carácter 0 o el carácter espacio (por defecto se usa el espacio).

**alineación** (opcional). Indica que tipo de alineación se usará para generar la salida: justificación derecha (por defecto) o izquierda (se indica con el carácter -).

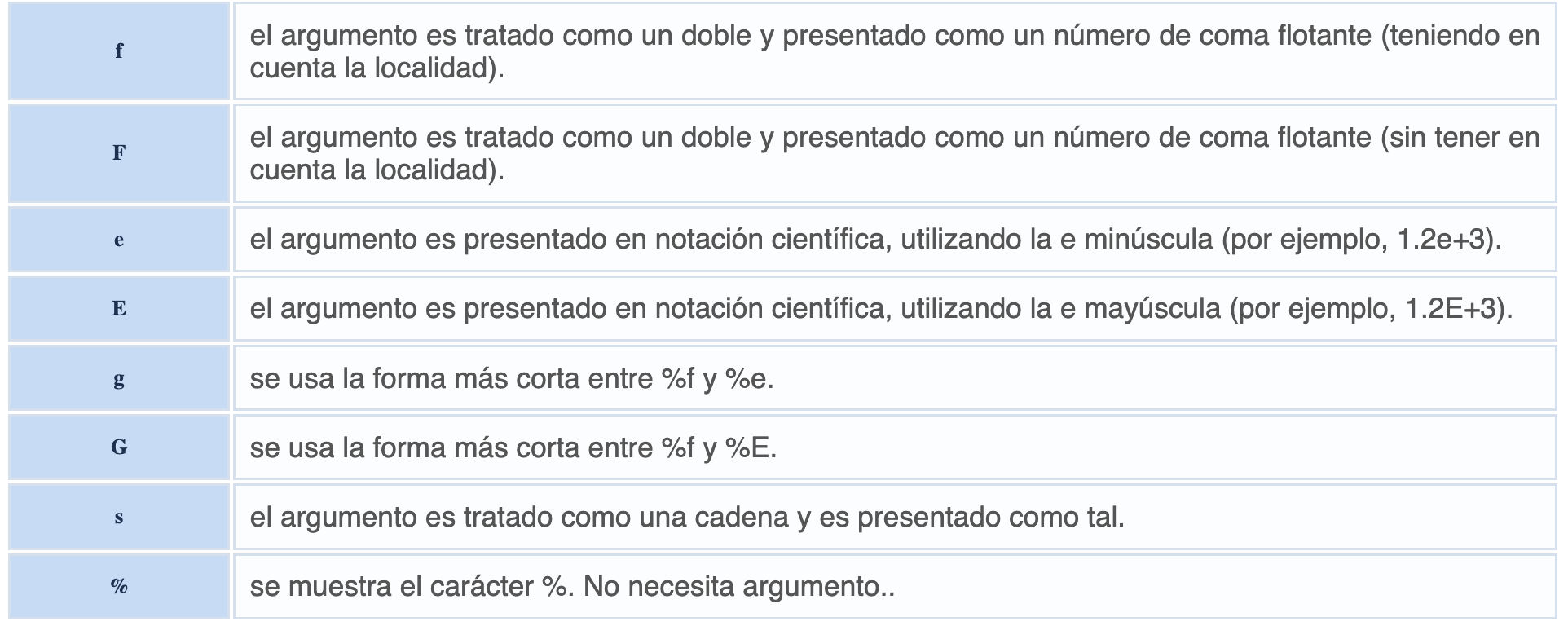
**ancho** (opcional). Indica el mínimo número de caracteres de salida para un parámetro dado.

**precisión** (opcional). Indica el número de dígitos decimales que se mostrarán para un número real. Se escribe como un dígito precedido por un punto.

**tipo** (obligatorio). Indica cómo se debe tratar el valor del parámetro correspondiente. En la siguiente tabla puedes ver una lista con todos los especificadores de tipo.

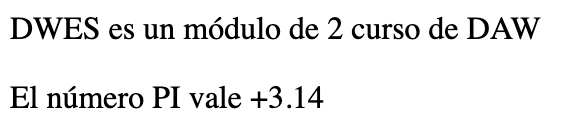
#### Especificadores de tipo para las funciones printf y sprintf





Por ejemplo, para obtener un número con signo y dos decimales, lo haríamos:





#### Sprintf

Existe una función similar a **printf** pero en vez de generar una salida con la cadena obtenida, permite guardarla en una variable: **sprintf**.

$txt\_pi = sprintf("El número PI vale %+.2f", 3.1416);

##### Práctica especificadores

–- Crea variables numéricas y de cadena relacionadas con productos de alguna empresa, dos de ellas con 2 decimales, otras dos con 4 decimales, un entero (sin decimales), una tipo cadena

Muestra una de ellas con dos decimales

Muestra una de ellas en binario

Preséntalo en notación científica

Muestra la cadena con printf

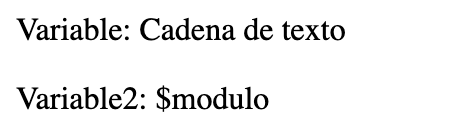
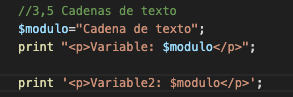
Almacena una variable real,con 3 decimales exactamente, en otra variable sin utilizando sprintf función printf y muestrala

Cuando la tengas:

* Súbela a tu **portfolio**

#### Cadenas de texto

En PHP las cadenas de texto pueden usar tanto comillas **simples** como comillas **dobles**. Sin embargo, hay una diferencia importante entre usar unas u otras. Cuando se pone una variable dentro de unas comillas dobles, se procesa y se sustituye por su valor. Así, el ejemplo anterior sobre el uso de **print** también podía haberse puesto de la siguiente forma:



La variable **$modulo** se reconoce dentro de las comillas dobles, y se sustituye. Con comillas simples, no se realizaría sustitución alguna.

Cuando se usan comillas simples, sólo se realizan dos sustituciones dentro de la cadena: cuando se encuentra la secuencia de caracteres **\'**, se muestra en la salida una comilla simple; y cuando se encuentra la secuencia **\\**, se muestra en la salida una barra invertida.

Estas secuencias se conocen como **secuencias de escape**. En las cadenas que usan comillas dobles, además de la secuencia **\\**, se pueden usar algunas más, pero no la secuencia **\'**. En esta tabla puedes ver las secuencias de escape que se pueden utilizar, y cuál es su resultado.



En PHP tienes dos operadores exclusivos para trabajar con cadenas de texto. Con el operador de **concatenación punto (.)** puedes unir las dos cadenas de texto que le pases como operandos. El operador de **asignación y concatenación (.=)** concatena al argumento del lado izquierdo la cadena del lado derecho.

En PHP tienes otra alternativa para crear cadenas: la sintaxis **heredoc**.

##### Cadenas

* Sube a tu **portfolio** la captura de tu editor de código con las pruebas que hayas hecho con las cadenas de texto

### Tipos de Datos. Funciones y Variables

**RA3**: Escribe bloques de sentencias embebidos en lenguajes de marcas, seleccionando y utilizando las estructuras de programación.

**C.Ev.** **D** Se han creado y utilizado funciones.

En PHP existen funciones específicas para comprobar y establecer el tipo de datos de una variable, **gettype** obtiene el tipo de la variable que se le pasa como parámetro y devuelve una cadena de texto, que puede ser

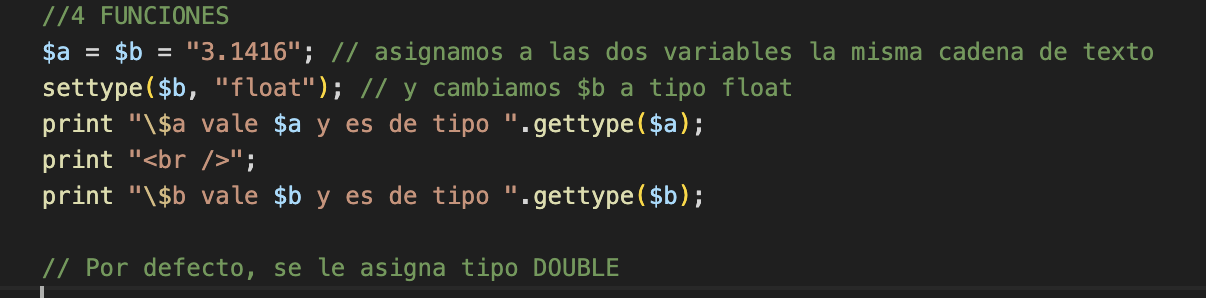
**array, boolean, double, integer, object, string, null, resource o unknowntype.**

También podemos comprobar si la variable es de un tipo concreto utilizando una de las siguientes funciones:

**is\_array(), is\_bool(), is\_float(), is\_integer(), is\_null(), is\_numeric(), is\_object(),  is\_resource(), is\_scalar() e is\_string().**

Devuelven true si la variable es del tipo indicado.

Análogamente, para establecer el tipo de una variable utilizamos la función **settype** pasándole como parámetros la variable a convertir, y una de las siguientes cadenas: **boolean**, **integer**, **float**, **string**, **array**, **object** o **null**. La función **settype** devuelve true si la conversión se realizó correctamente, o false en caso contrario.



El resultado del código anterior es:

**$a** vale 3.1416 y es de tipo string

**$b** vale 3.1416 y es de tipo double

Si lo único que te interesa es saber si una variable está definida y no es **null**, puedes usar la función **isset**. La función **unset** destruye la variable o variables que se le pasa como parámetro.

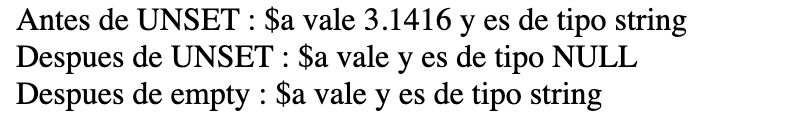
#### Isset, unset, isnull

Si lo único que te interesa es saber si una variable está definida y no es **null**, puedes usar la función **isset**.

La función **unset** destruye la variable o variables que se le pasa como parámetro.



Es importante no confundir el que una variable esté definida o valga null, con que se considere como vacía debido al valor que contenga. Esto último es lo que nos indica la función **empty**.



#### Evitar Warning

Aunque es bueno que los errores y las advertencias (warning) sean mostradas en nuestro navegador, a veces son un poco incómodas para nuestra creación de código. Para evitar el siguiente mensaje de error existen varias formas de hacerlo, tanto dentro de nuestro código como dentro del fichero de configuración **php.ini**

**Enlace a** [**“cómo ocultar warning en PHP”**](https://www.baulphp.com/como-ocultar-los-warning-notice-en-php/#Como_ocultar_los_warning,_notice_en_PHP)

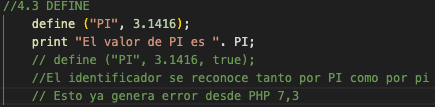


#### Constantes (define)

Existe también en PHP una función, **define**, con la que puedes definir constantes, esto es, identificadores a los que se les asigna un valor que no cambia durante la ejecución del programa.

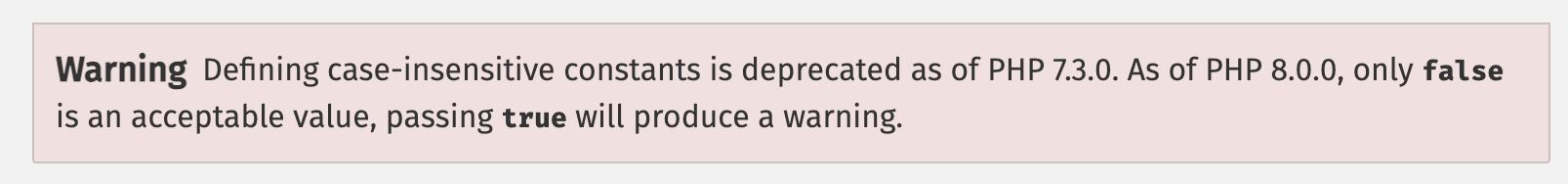
bool **define** ( string $identificador , mixed $valor [, bool $case\_insensitive = false ] );

Los identificadores **no van precedidos por el signo "$"** y suelen escribirse en MAYÚSCULAS, aunque existe un tercer parámetro opcional, que si vale true hace que se reconozca el identificador independientemente de si está escrito en mayúsculas o en minúsculas.



Sólo se permiten los siguientes tipos de valores para las constantes:  **integer,**  **float**, **string**, **boolean** y **null**.

El caso insensitivo genera error desde PHP 7.3 <https://www.php.net/manual/en/function.define.php>



#### Fechas y Horas

En PHP **no existe un tipo de datos** específico para trabajar con fechas y horas. La información de fecha y hora se almacena internamente como un **número entero**. Sin embargo, dentro de las funciones de PHP tienes a tu disposición unas cuantas para trabajar con ese tipo de datos.

Una de las más útiles es quizás la función **date**, que te permite obtener una cadena de texto a partir de una fecha y hora, con el formato que tú elijas. La función recibe dos parámetros, la descripción del formato y el número entero que identifica la fecha, y devuelve una cadena de texto formateada.

string date (string $formato [, int $fechahora]);

El formato lo debes componer utilizando como base una serie de caracteres de los que figuran en la siguiente tabla.

| **Carácter** | **Resultado** |
| --- | --- |
| **d** | día del mes con dos dígitos. |
| **j** | día del mes con uno o dos dígitos ( sin ceros iniciales ). |
| **z** | día del año, comenzando por el cero ( 0 = 1 de enero ). |
| **N** | día de la semana ( 1 = lunes, ..., 7 = domingo ). |
| **w** | día de la semana ( 0 = domingo, ..., 6 = sábado ). |
| **l** | texto del día de la semana, en inglés ( Monday, ..., Sunday ). |
| **D** | texto del día de la semana, solo tres letras, en inglés ( Mon, ..., Sun ). |
| **W** | número de la semana del año. |
| **m** | número del mes con dos dígitos. |
| **n** | número del mes con uno o dos dígitos ( sin ceros iniciales ). |
| **t** | número de días que tiene el mes. |
| **F** | texto del día del mes, en inglés ( January, ..., December ). |
| **M** | texto del día del mes, solo tres letras, en inglés ( Jan, ..., Dec ). |
| **Y** | número del año. |
| **y** | dos últimos dígitos del número del año. |
| **L** | 1 si el año es bisiesto, 0 si no lo es. |
| **h** | formato de 12 horas, siempre con dos dígitos. |
| **H** | formato de 24 horas, siempre con dos dígitos. |
| **g** | formato de 12 horas, con uno o dos dígitos ( sin ceros iniciales ). |
| **G** | formato de 24 horas, con uno o dos dígitos ( sin ceros iniciales ). |
| **a** | am o pm, en minúsculas. |
| **A** | AM o PM, en mayúsculas. |
| **r** | fecha entera con formato RFC 2822. |

Además, el segundo parámetro es opcional. Si no se indica, se utilizará la hora actual para crear la cadena de texto.

Para que el sistema pueda darte información sobre tu fecha y hora, debes indicarle tu zona horaria. Puedes hacerlo con la función **date\_default\_timezone\_set**. Para establecer la zona horaria en España peninsular debes indicar:

**date\_default\_timezone\_set('Europe/Madrid');**

Si utilizas alguna función de fecha y hora sin haber establecido previamente tu zona horaria, lo más probable es que recibas un error o mensaje de advertencia de PHP indicándolo.

Otras funciones como **getdate** devuelven un array con información sobre la fecha y hora actual.

En la documentación de PHP puedes consultar todas las funciones para gestionar fechas y horas: <https://www.php.net/manual/en/ref.datetime.php>

##### Zona, date y getdate

Establece la zona geográfica en tu código y prueba las funciones vistas en este punto, muestra con al menos 5 formatos diferentes.

Cuando la tengas:

* Súbela a tu **portfolio** y

#### Variables Especiales de PHP (superglobals)

En la unidad anterior ya aprendiste qué eran y cómo se utilizaban las variables globales. PHP incluye unas cuantas **variables internas** predefinidas que pueden usarse desde cualquier ámbito, por lo que reciben el nombre de **variables superglobales**. Ni siquiera es necesario que uses **global** para acceder a ellas.

Cada una de estas variables es un **array** que contiene un conjunto de valores (en esta unidad veremos más adelante cómo se pueden utilizar los arrays). Las variables superglobales disponibles en PHP son las siguientes:

**$\_SERVER**. Contiene información sobre el entorno del servidor web y de ejecución. Entre la información que nos ofrece esta variable, tenemos:



En la documentación de PHP puedes consultar toda la información que ofrece **$\_SERVER**: <https://www.php.net/manual/es/reserved.variables.server.php>

**$\_GET**, **$\_POST** y **$\_COOKIE** contienen las variables que se han pasado al script actual utilizando respectivamente los métodos GET (parámetros en la URL), HTTP POST y Cookies HTTP.

**$\_REQUEST** junta en uno solo el contenido de los tres arrays anteriores, **$\_GET**, **$\_POST** y **$\_COOKIE**.

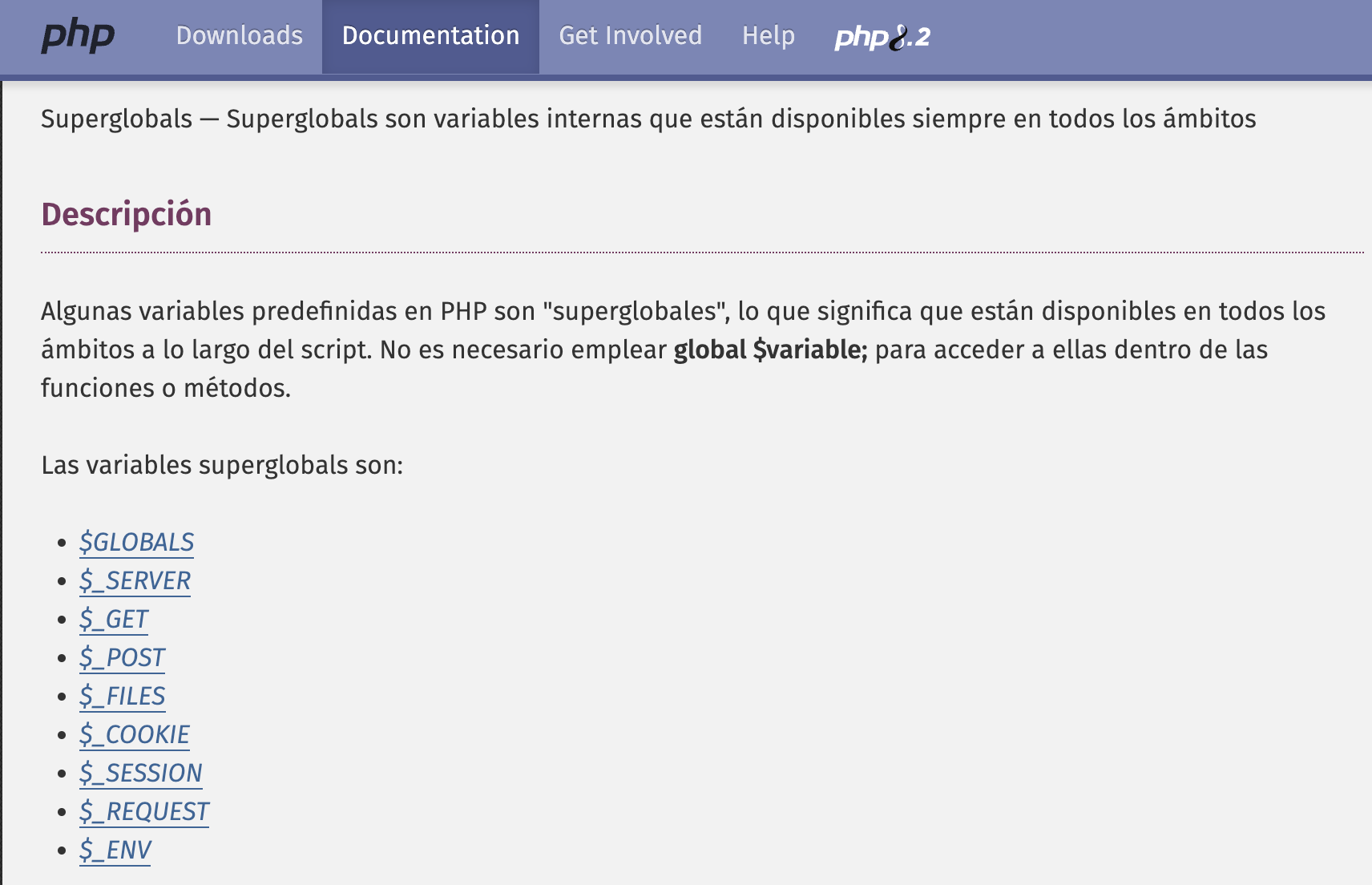
**$\_ENV** contiene las variables que se puedan haber pasado a PHP desde el entorno en que se ejecuta.

**$\_FILES** contiene los ficheros que se puedan haber subido al servidor utilizando el método POST.

**$\_SESSION** contiene las variables de sesión disponibles para el guión actual.

En posteriores unidades iremos trabajando con estas variables.

Conviene tener a mano la información sobre estas variables superglobales disponible en el manual de PHP.



<https://www.php.net/manual/es/language.variables.superglobals.php>



##### Variables Especiales

Busca info de algunas de estas variables especiales y pruébalas en tu código

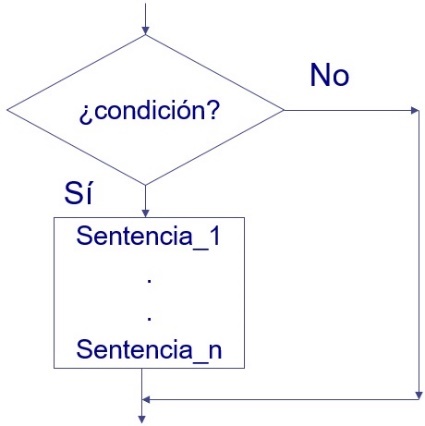
Cuando la tengas:

* Súbela a tu **portfolio** y a tu repositorio **GitHub**

### Estructuras de Control

**RA3**: Escribe bloques de sentencias embebidos en lenguajes de marcas, seleccionando y utilizando las estructuras de programación.

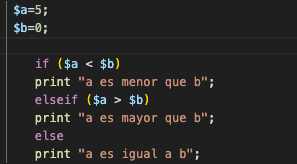
**C.Ev.** **B**: Se han utilizado bucles y se ha verificado su funcionamiento..

Para definir el flujo de un programa en PHP, al igual que en la mayoría de lenguajes de programación, hay sentencias para dos tipos de **estructuras de control**:

* **sentencias condicionales**, que permiten definir las condiciones bajo las que debe ejecutarse una sentencia o un bloque de sentencias; y
* **sentencias de bucle**, con las que puedes definir si una sentencia o conjunto de sentencias se repite o no, y bajo qué condiciones.

#### Condicionales

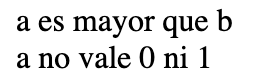
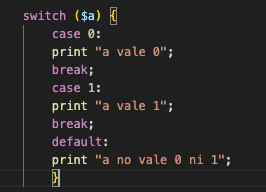
**if / elseif / else**. La sentencia **if** permite definir una expresión para ejecutar o no la sentencia o conjunto de sentencias siguiente. Si la expresión se evalúa a true (verdadero), la sentencia se ejecuta. Si se evalúa a false (falso), no se ejecutará.

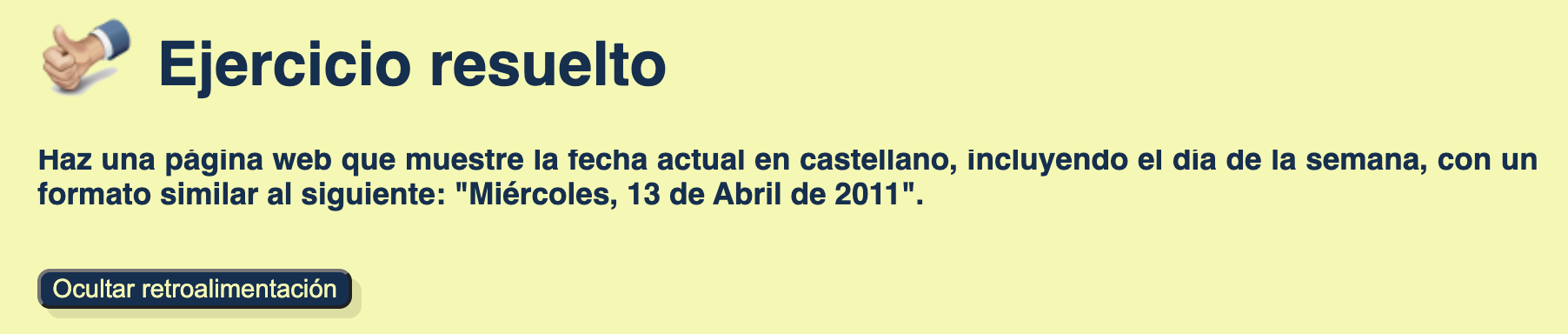
Cuando el resultado de la expresión sea false, puedes utilizar **else** para indicar una sentencia o grupo de sentencias a ejecutar en ese caso. Otra alternativa a **else** es utilizar **elseif** y escribir una nueva expresión que comenzará un nuevo condicional.

Cuando, como sucede en el ejemplo, la sentencia **if elseif** o **else** actúe sobre una única sentencia, no será necesario usar llaves. Tendrás que usar llaves para formar un conjunto de sentencias siempre que quieras que el condicional actúe sobre más de una sentencia.

**switch**. La sentencia **switch** es similar a enlazar varias sentencias **if** comparando una misma variable con diferentes valores.

* Cada valor va en una sentencia **case**.
* Cuando se encuentra una coincidencia, comienzan a ejecutarse las sentencias siguientes hasta que acaba el bloque **switch**, o hasta que se encuentra una sentencia **break**.
* Si no existe coincidencia con el valor de ningún **case**, se ejecutan las sentencias del bloque **default**, en caso de que exista.





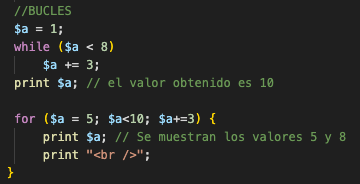
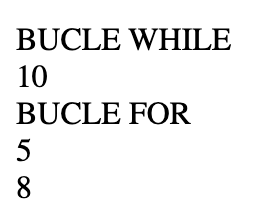
#### Bucles

Existen diferentes tipos de bucles en PHP:

* **while:** Usando **while** puedes definir un bucle que se ejecuta mientras se cumpla una expresión. La expresión se evalúa antes de comenzar cada ejecución del bucle.
* **do / while:** Es un bucle similar al anterior, pero la expresión se evalúa al final, con lo cual se asegura que la sentencia o conjunto de sentencias del bucle se ejecutan al menos una vez.
* **for:** Son los bucles más complejos de PHP. Al igual que los del lenguaje C, se componen de tres expresiones:

for (expr1; expr2; expr3)

* + La primera expresión, **expr1**, se ejecuta solo una vez al comienzo del bucle.
  + La segunda expresión, **expr2**, se evalúa para saber si se debe ejecutar o no la sentencia o conjunto de sentencias. Si el resultado el false, el bucle termina.
  + Si el resultado es true, se ejecutan las sentencias y al finalizar se ejecuta la tercera expresión, **expr3**, y se vuelve a evaluar **expr2** para decidir si se vuelve a ejecutar o no el bucle.
* Puedes anidar cualquiera de los bucles anteriores en varios niveles. También puedes usar las sentencias **break**, para salir del bucle, y **continue**, para omitir la ejecución de las sentencias restantes y volver a la comprobación de la expresión respectivamente.

### Funciones

**RA3**: Escribe bloques de sentencias embebidos en lenguajes de marcas, seleccionando y utilizando las estructuras de programación.

**C.Ev.** **D** Se han creado y utilizado funciones.

Cuando quieres repetir la ejecución de un bloque de código, puedes utilizar un bucle. Las **funciones** tienen una utilidad similar: **nos permiten asociar una etiqueta** (el nombre de la función) **con un bloque de código** a ejecutar. Además, al usar funciones estamos ayudando a **estructurar** mejor el código. Como ya sabes, las funciones permiten crear variables locales que no serán visibles fuera del cuerpo de las mismas.

Como programador puedes aprovecharte de la gran cantidad de funciones **disponibles en PHP**. De éstas, muchas están incluidas en el núcleo de PHP y se pueden usar directamente. Otras muchas se encuentran disponibles en forma de **extensiones**, y se pueden incorporar al lenguaje cuando se necesitan.

Con la distribución de PHP se incluyen varias extensiones. Para poder usar las funciones de una extensión, tienes que asegurarte de activarla mediante el uso de una directiva **extensión** en el fichero **php.ini**. Muchas otras extensiones no se incluyen con PHP y antes de poder utilizarlas tienes que descargarlas.

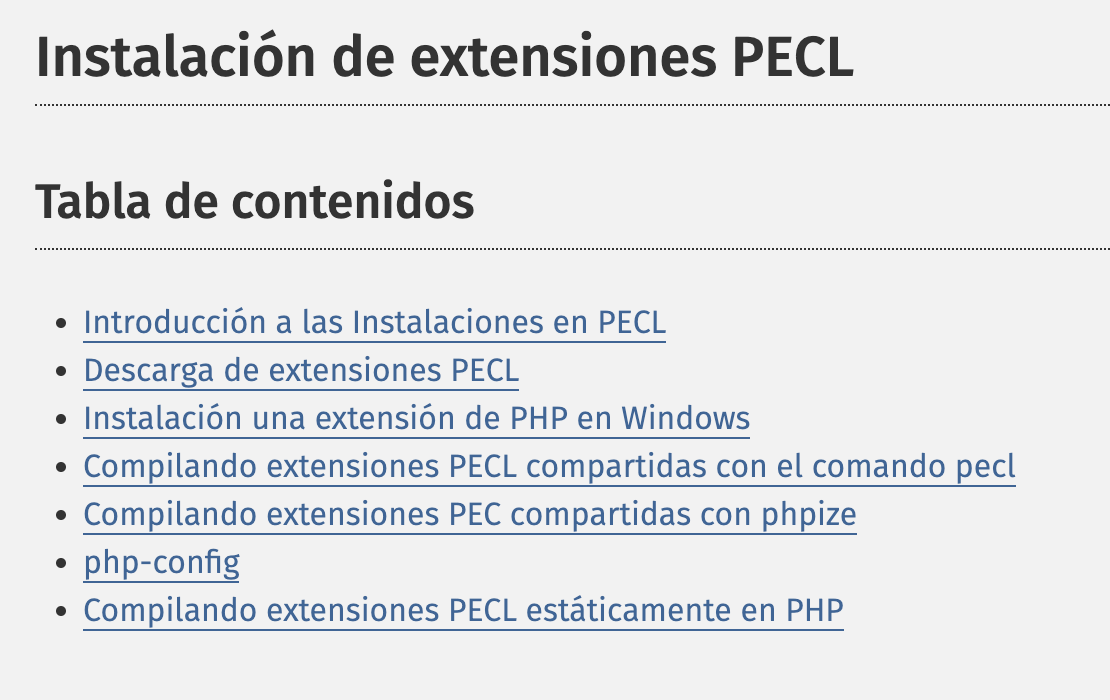
Para obtener extensiones para el lenguaje PHP puedes utilizar PECL. PECL es un **repositorio de extensiones** para PHP. Junto con PHP se incluye un **comando pecl** que puedes utilizar para instalar extensiones de forma sencilla:

**pecl** install nombre\_extensión

#### PECL

En el manual de PHP tienes más información sobre PECL.

<https://www.php.net/manual/es/install.pecl.php>



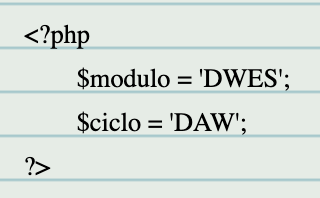
#### Inclusión de ficheros externos (include/require).

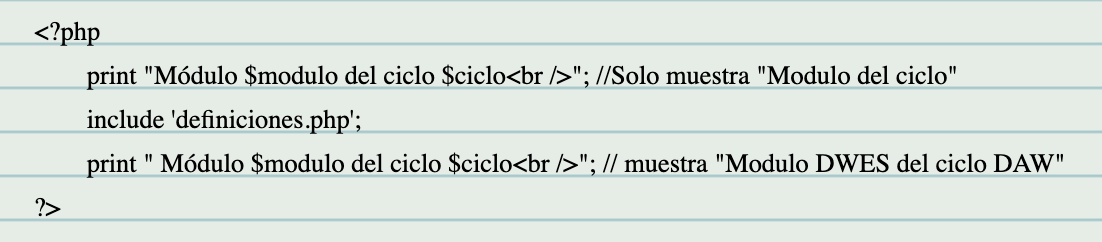
Conforme vayan creciendo los programas que hagas, verás que resulta trabajoso encontrar la información que buscas dentro del código. En ocasiones resulta útil agrupar ciertos grupos de funciones o bloques de código, y ponerlos en un fichero aparte. Posteriormente, puedes hacer referencia a esos ficheros para que PHP incluya su contenido como parte del programa actual.

Para incorporar a tu programa contenido de un archivo externo, tienes varias posibilidades:

* **include:** Evalúa el contenido del fichero que se indica y lo incluye como parte del fichero actual, en el mismo punto en que se realiza la llamada. La ubicación del fichero puede especificarse utilizando una ruta absoluta, pero lo más usual es con una ruta relativa. En este caso, se toma como base la ruta que se especifica en la directiva **include\_path** del fichero **php.ini**. Si no se encuentra en esa ubicación, se buscará también en el directorio del guión actual, y en el directorio de ejecución.

**Definiciones.php**



**Programa.php**

Cuando se comienza a evaluar el contenido del **fichero externo**, se abandona de forma automática el **modo PHP** y su contenido se trata en principio como etiquetas HTML. Por este motivo, es necesario delimitar el código PHP que contenga nuestro archivo externo utilizando dentro del mismo los **delimitadores** **<?php y ?>**.

* **include\_once:** Si por equivocación incluyes más de una vez un mismo fichero, lo normal es que obtengas algún tipo de error (por ejemplo, al repetir una definición de una función). **include\_once** funciona exactamente igual que **include**, pero solo incluye aquellos ficheros que aún no se hayan incluido.
* **require:** Si el fichero que queremos incluir no se encuentra, **include** da un aviso y continua la ejecución del guión. La diferencia más importante al usar **require** es que en ese caso, cuando no se puede incluir el fichero, se detiene la ejecución del guión.
* **require\_once**. Es la combinación de las dos anteriores. Asegura la inclusión del fichero indicado solo una vez, y genera un error si no se puede llevar a cabo.

Muchos programadores utilizan la doble extensión **.inc.php** para aquellos ficheros en lenguaje PHP cuyo destino es ser incluidos dentro de otros, y nunca han de ejecutarse por sí mismos.

##### Include

Crea una estructura con dos ficheros donde uno sea incluido parecida a la del ejemplo y ejecútala

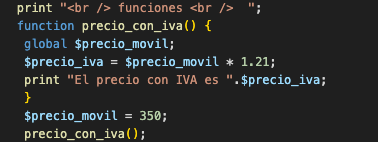
Cuando la tengas:

* Súbela a tu **portfolio** y
* **Sube a Moodle** una captura de pantalla del Portfolio actualizado en este punto

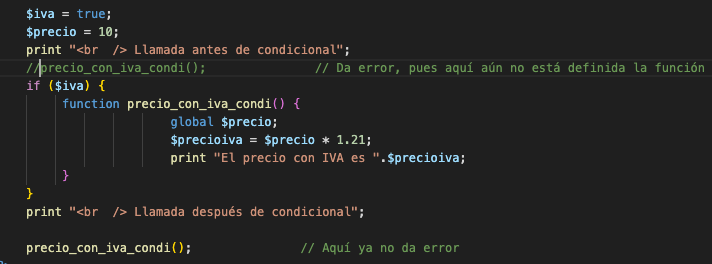
#### Creación y Ejecución de funciones. (function)

Ya sabes que, para hacer una llamada a una función, basta con poner su nombre y unos paréntesis: **phpinfo();**

Para crear tus propias funciones, deberás usar la palabra **function**.

En PHP no es necesario que definas una función antes de utilizarla, excepto cuando está condicionalmente definida como se muestra en el siguiente ejemplo:



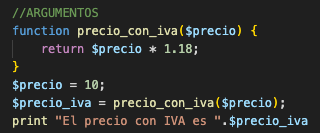
Cuando una función está definida de una **forma condicional** sus definiciones deben ser procesadas antes de ser llamadas. Por tanto, la definición de la función debe estar **antes** de cualquier llamada.



#### Argumentos

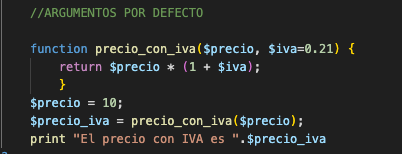
En el ejemplo anterior en la función usabas una variable **global**, lo cual no es una buena práctica. Siempre es mejor utilizar **argumentos o parámetros** al hacer la llamada. Además, en lugar de mostrar el resultado en pantalla o guardar el resultado en una variable global, las funciones pueden devolver un valor usando la sentencia **return**. Cuando en una función se encuentra una sentencia **return**, termina su procesamiento y devuelve el valor que se indica.

Puedes reescribir la función anterior de la siguiente forma:



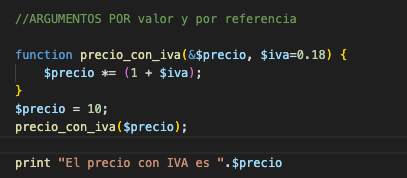
Los argumentos se indican en la **definición** de la función como una lista de variables separada por comas. No se indica el tipo de cada argumento, al igual que no se indica si la función va a devolver o no un valor (si una función no tiene una sentencia **return**, devuelve **null** al finalizar su procesamiento).

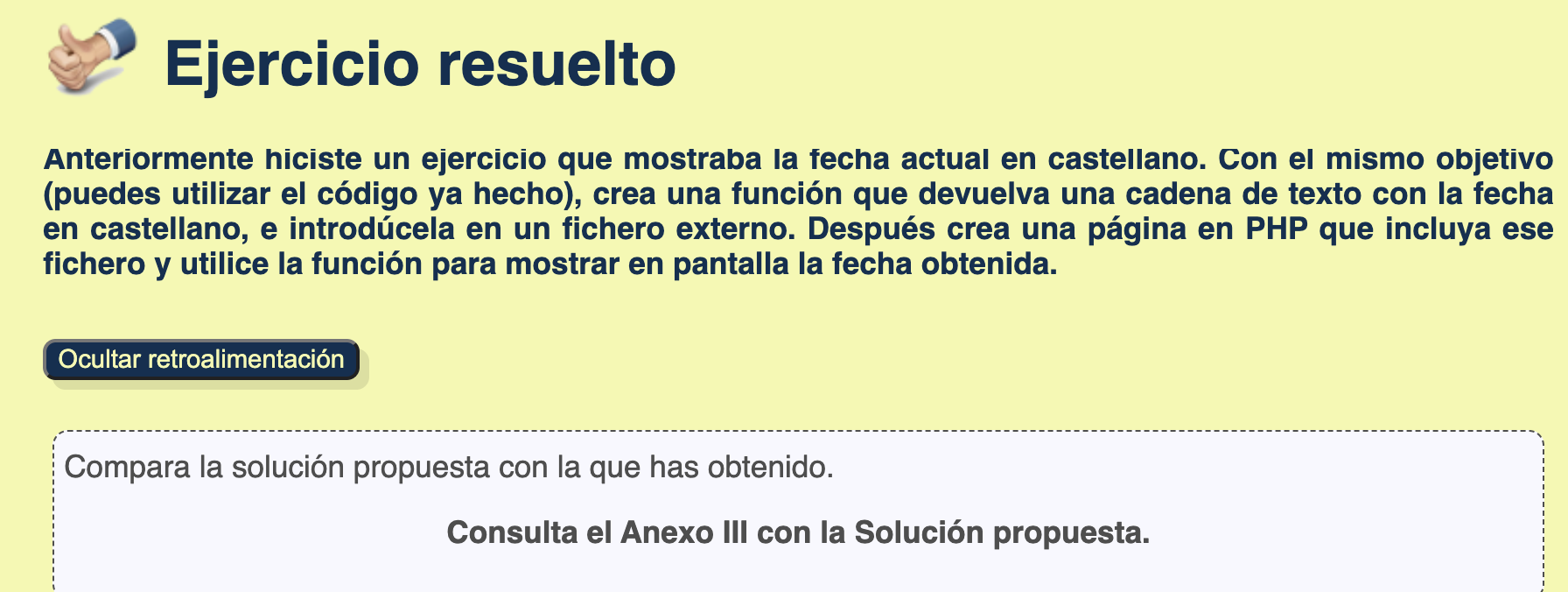
Al definir la función, puedes indicar **valores por defecto** para los argumentos, de forma que cuando hagas una llamada a la función puedes no indicar el valor de un argumento; en este caso se toma el valor por defecto indicado.



Puede haber valores por defecto definidos para varios argumentos, pero en la lista de argumentos de la función todos ellos deben estar a la **derecha** de cualquier otro argumento sin valor por defecto.

En los ejemplos anteriores los argumentos se pasaban **por valor**. Esto es, cualquier cambio que se haga dentro de la función a los valores de los argumentos no se reflejará fuera de la función. Si quieres que esto ocurra debes definir el parámetro para que su valor se pase **por referencia**, añadiendo el símbolo **&** antes de su nombre.



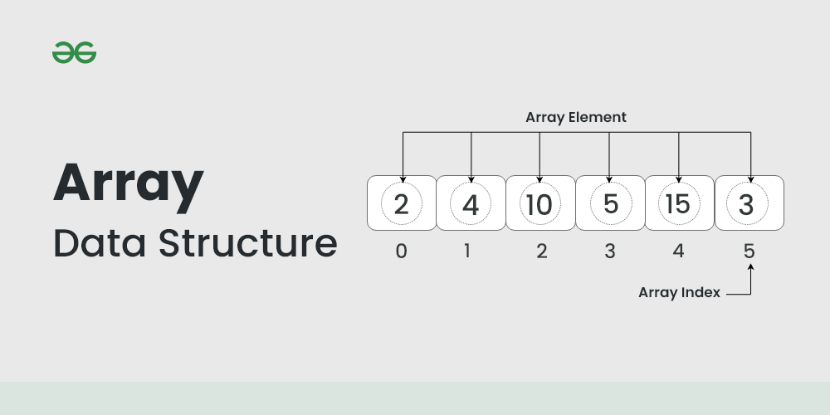
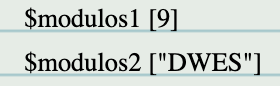


### Arrays

**RA3**: Escribe bloques de sentencias embebidos en lenguajes de marcas, seleccionando y utilizando las estructuras de programación.

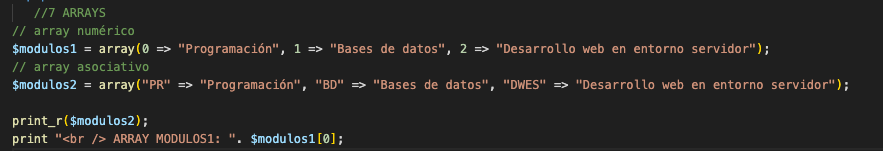
**C.Ev. C** Se han utilizado «arrays» para almacenar y recuperar conjuntos de datos..

Un tipo de datos compuesto es aquel que te permite almacenar más de un valor. En PHP puedes utilizar dos tipos de datos compuestos: el **array** y el **objeto**. Los objetos los veremos más adelante; vamos a empezar con los arrays.

Un **array** es un tipo de datos que nos permite almacenar varios valores. Cada miembro del **array** se almacena en una posición a la que se hace referencia utilizando un valor clave. Las **claves** pueden ser numéricas o asociativas.

Para hacer referencia a los elementos almacenados en un array, tienes que utilizar el valor clave entre **corchetes**:

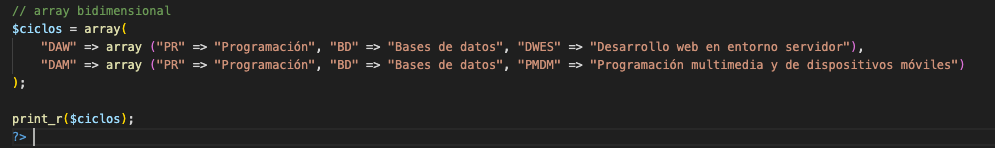
En PHP existe la función **print\_r**, que nos muestra todo el contenido del array que le pasamos. Es muy útil para tareas de depuración.

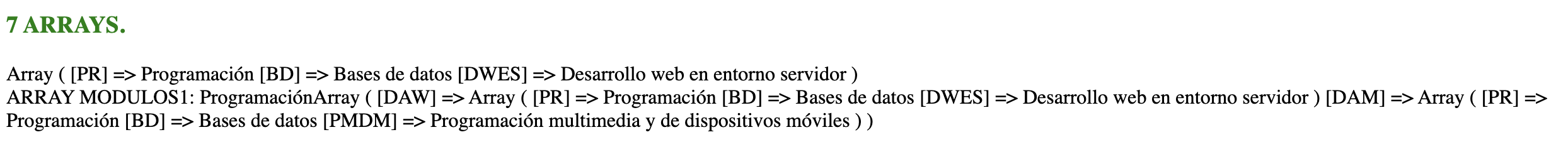


#### Arrays multidimensionales

Los arrays anteriores son vectores, esto es, arrays unidimensionales. En PHP puedes crear también arrays de **varias dimensiones** almacenando otro array en cada uno de los elementos de un array.

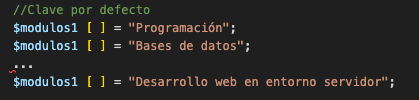
Para hacer referencia a los elementos almacenados en un **array multidimensional**, debes indicar las claves para cada una de las dimensiones:





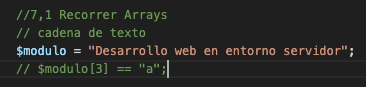
En PHP no es necesario que indiques el **tamaño** del array antes de crearlo. Ni siquiera es necesario indicar que una variable concreta es de tipo array. Simplemente puedes comenzar a asignarle valores:

Ni siquiera es necesario que especifiques el **valor** de la clave. Si la omites, el array se irá llenando a partir de la última clave numérica existente, o de la **posición 0** si no existe ninguna:



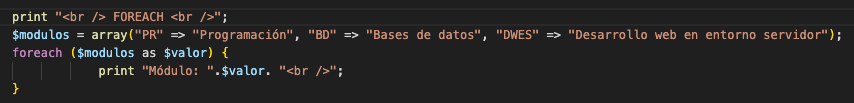
#### Recorrer Arrays (foreach)

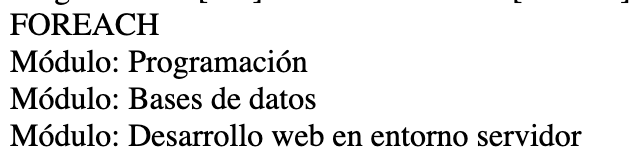
Las **cadenas de texto** o **strings** se pueden tratar como arrays en los que se almacena una letra en cada posición, siendo 0 el índice correspondiente a la primera letra, 1 el de la segunda, etc.



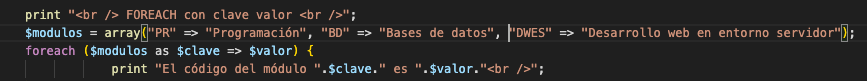
Para recorrer los elementos de un array, en PHP puedes usar un bucle específico: **foreach**. Utiliza una variable temporal para asignarle en cada iteración el valor de cada uno de los elementos del array. Puedes usarlo de dos formas.

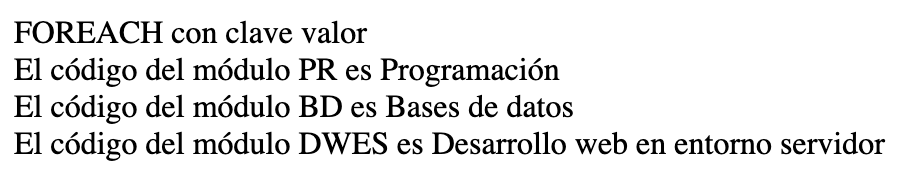
* Recorriendo sólo los elementos:





* O recorriendo sus **valores clave** y sus **elementos** de forma simultánea:



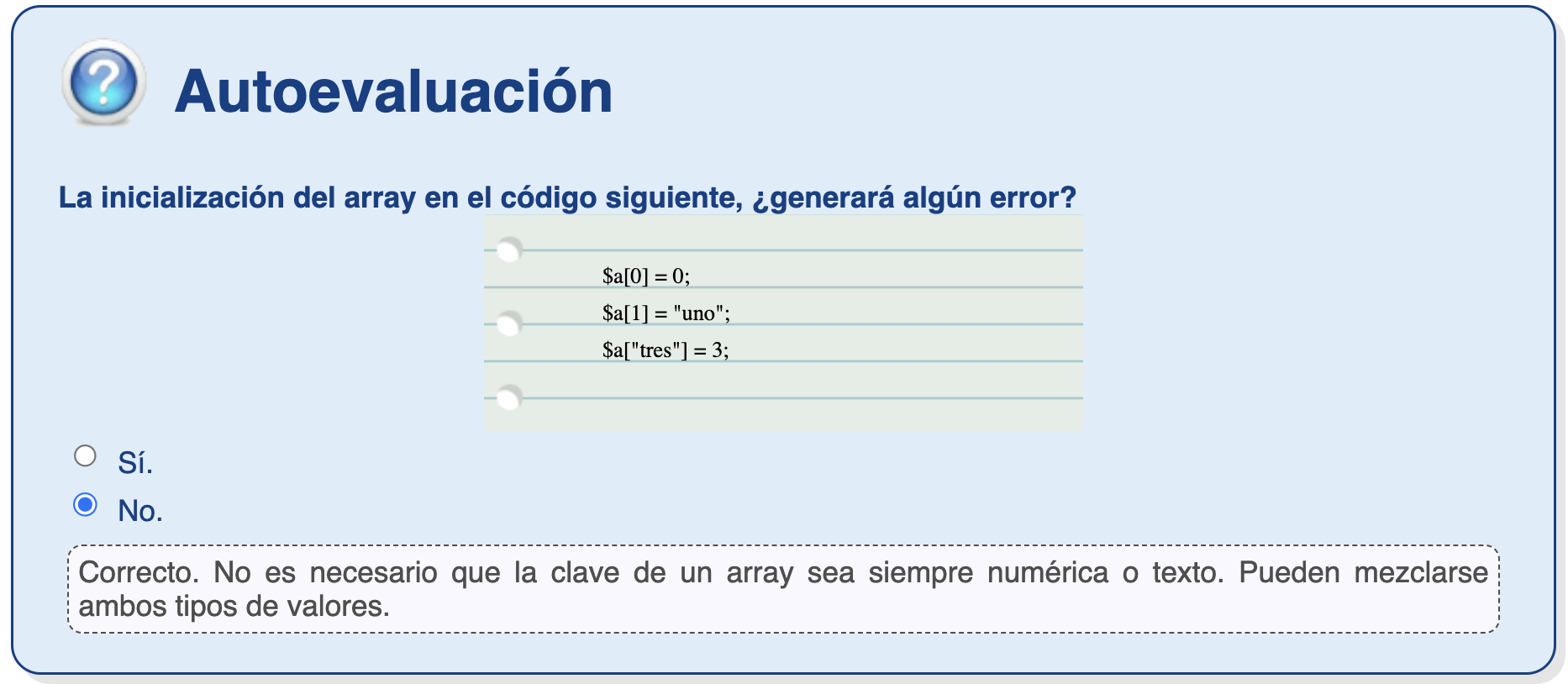


##### Foreach para $\_SERVER

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.

Cuando la tengas:

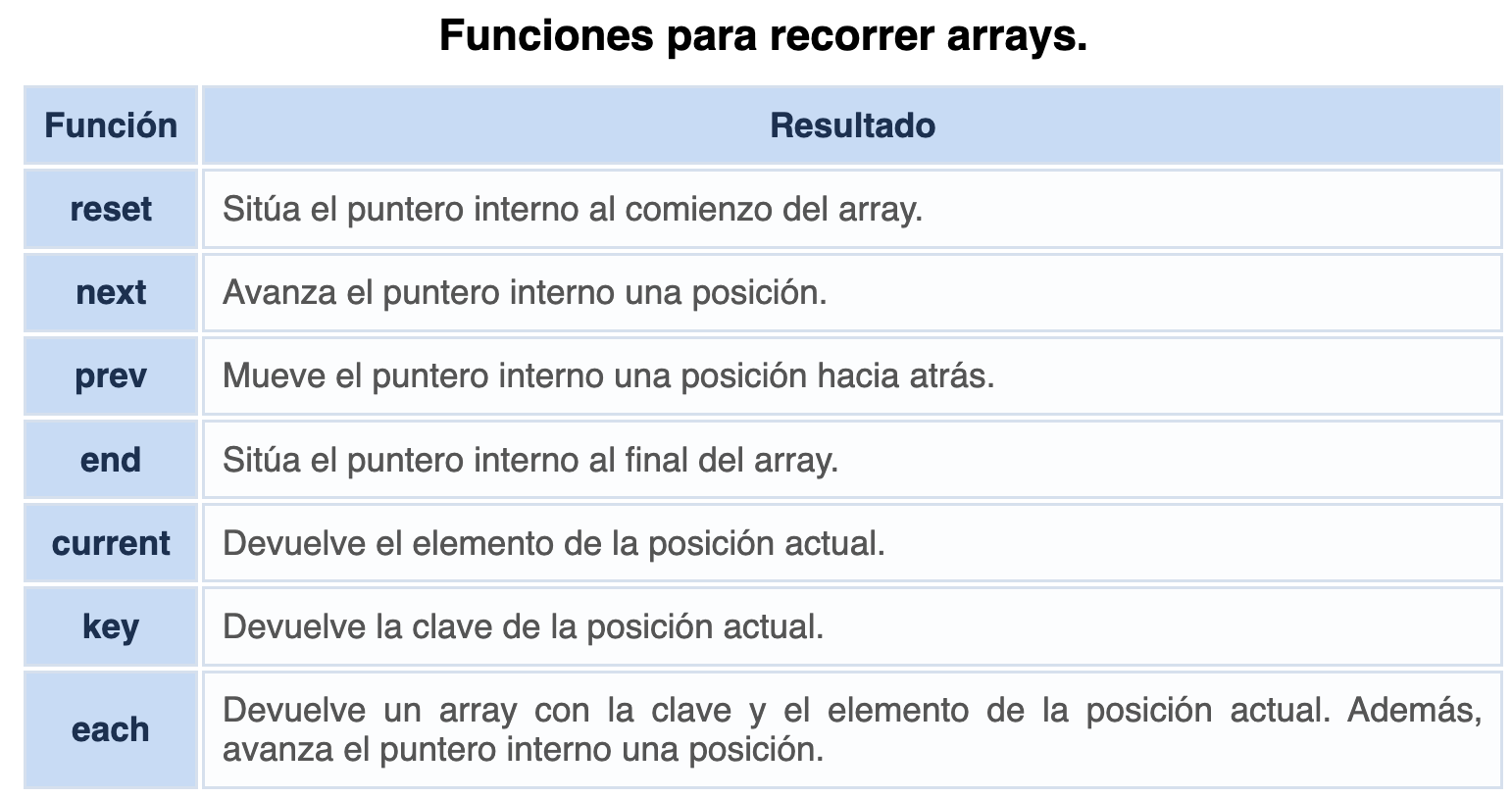
* Súbela a tu **portfolio** y
* **Sube a Moodle** una captura de pantalla del Portfolio actualizado en este punto
* **Compara la solución** propuesta con la que has obtenido



#### Funciones para recorrer Arrays (next, prev…)

Pero en PHP también hay otra **forma** de recorrer los valores de un array. Cada array mantiene un **puntero** interno, que se puede utilizar con este fin. Utilizando funciones específicas, podemos **avanzar, retroceder o inicializar el puntero**, así como **recuperar** los valores del elemento (o de la pareja clave / elemento) al que apunta el puntero en cada momento.

Algunas de estas funciones son:



**each** **ha sido eliminado** desde la versión 8 de PHP;

Las funciones **reset**, **next**, **prev** y **end**, además de mover el puntero interno devuelven, al igual que **current**, el valor del nuevo elemento en que se posiciona. Si al mover el puntero te sales de los límites del array (por ejemplo, si ya estás en el último elemento y haces un **next**), cualquiera de ellas devuelve **false**. Sin embargo, al comprobar este valor devuelto no serás capaz de distinguir si te has salido de los límites del array, o si estás en una posición válida del array que contiene el valor "false".

La función **key** devuelve **null** si el puntero interno está **fuera** del array.



#### Funciones datos compuestos (array, unset…)

Además de asignando valores directamente, **la función array permite crear un array con una sola línea de código**, tal y como vimos anteriormente. Esta función recibe un conjunto de parámetros, y crea un array a partir de los valores que se le pasan. Si en los parámetros no se indica el valor de la clave, crea un array **numérico** (con base 0). Si no se le pasa ningún parámetro, crea un array vacío.



Una vez definido un array puedes añadir **nuevos elementos** y **modificar** los ya existentes (utilizando el **índice** del elemento a modificar). También se pueden **eliminar** elementos de un array utilizando la función **unset**.

En el caso de los arrays numéricos, eliminar un elemento significa que las claves del mismo ya no estarán consecutivas.

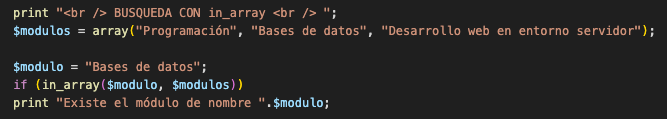




La función **array\_values** recibe un array como parámetro, y devuelve un nuevo array con los mismos elementos y con índices numéricos consecutivos con base 0.

Para comprobar si una variable es de tipo array, utiliza la función **is\_array**. Para obtener el número de elementos que contiene un array, tienes la función **count**.

Si quieres **buscar** un elemento concreto dentro de un array, puedes utilizar la función **in\_array**. Recibe como parámetros el elemento a buscar y la variable de tipo array en la que buscar, y devuelve true si encontró el elemento o false en caso contrario.



Otra posibilidad es la función **array\_search**, que recibe los mismos parámetros pero **devuelve la clave** correspondiente al elemento, o false si no lo encuentra.

Y si lo que quieres buscar es una clave en un array, tienes la función **array\_key\_exists**, que devuelve true o false.



<https://www.php.net/manual/es/ref.array.php>

### Formularios

**RA3**: Escribe bloques de sentencias embebidos en lenguajes de marcas, seleccionando y utilizando las estructuras de programación.

**C.Ev.** **E**: Se han utilizado formularios Web para interactuar con el usuario del navegador Web.

La forma natural para hacer llegar a la aplicación web los datos del usuario desde un navegador, es utilizar **formularios HTML**.

Los formularios HTML van encerrados siempre entre las etiquetas **<FORM> </FORM>**. Dentro de un formulario se incluyen los elementos sobre los que puede actuar el usuario, principalmente usando las etiquetas **<INPUT>**, **<SELECT>**, **<TEXTAREA>** y **<BUTTON>**.

El atributo **action** del elemento **FORM** indica la página a la que se le enviarán los datos del formulario. En nuestro caso se tratará de un guión PHP.

Por su parte, el atributo **method** especifica el método usado para enviar la información. Este atributo puede tener dos valores:

* **get**: con este método los datos del formulario se agregan al URI utilizando un signo de interrogación "?" como separador.
* **post**: con este método los datos se incluyen en el cuerpo del formulario y se envían utilizando el protocolo HTTP.

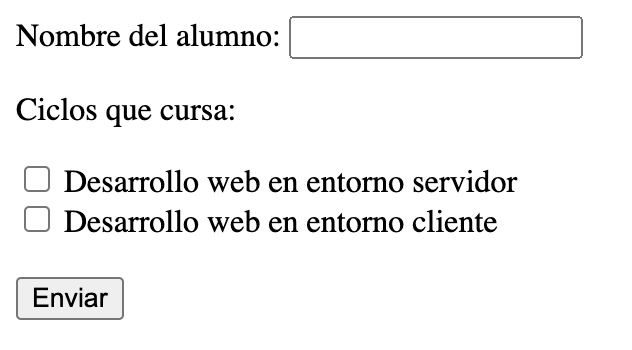
Como vamos a ver, los datos se recogerán de distinta forma dependiendo de cómo se envíen.

##### Formulario1.php

Crea un **formulario HTML** llamado formulario1.php para introducir el nombre del alumno y el módulo que cursa, a escoger entre “Desarrollo Web en Entorno Servidor” y “Desarrollo Web en Entorno Cliente”.

Envía el resultado a la página **“procesa.php”**, que será la encargada de procesar los datos.





Fíjate que si en un formulario web tienes que enviar alguna variable en la que sea posible **almacenar más de un valor**, como es el caso de las **casillas** de verificación en el ejemplo anterior (se pueden marcar varias a la vez), tendrás que ponerle corchetes al nombre de la variable para indicar que se trata de un **array**.

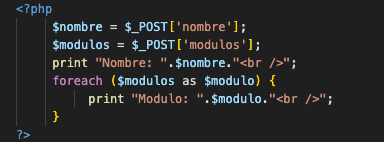
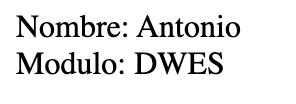
#### Curso HTML

Para no tener problemas al programar en PHP, debes conocer el lenguaje **HTML**, concretamente los detalles relativos a la creación de **formularios** web. Puedes consultar esta información por ejemplo en el curso sobre HTML de aulaClic:

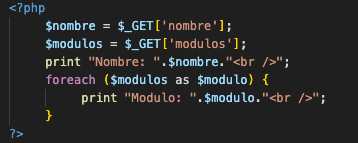
<https://www.aulaclic.es/html/t_8_1.htm>

#### Procesamiento información de formularios

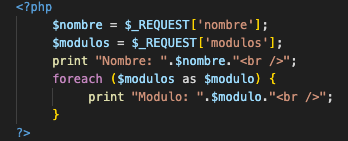
En el ejemplo anterior creaste un **formulario en una página HTML** que recogía datos del usuario y los enviaba a una página PHP para que los procesara. Como usaste el método **POST**, los datos se pueden recoger utilizando la variable **$\_POST**. Si simplemente los quisieras mostrar por pantalla, éste podría ser el código de **"procesa.php":**

Si por el contrario hubieras usado el método **GET**, el código necesario para procesar los datos sería similar; simplemente haría falta cambiar la variable **$\_POST** por **$\_GET**.

****

En **cualquiera** de los dos casos podrías haber usado **$\_REQUEST** sustituyendo respectivamente a **$\_POST** y a **$\_GET**.

****

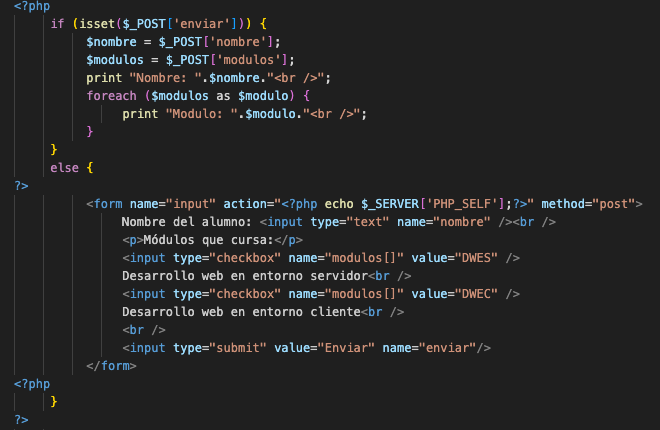
#### Validación de datos. Misma página

Siempre que sea posible, es preferible ***validar* los datos que se introducen en el navegador antes de enviarlos**. Para ello deberás usar código en **lenguaje Javascript.**

Si por algún motivo hay datos que se tengan que validar en el servidor, por ejemplo, porque necesites comprobar que los datos de un usuario no existan ya en la base de datos antes de introducirlos, será necesario hacerlo con código PHP en la página que figura en el atributo **action** del formulario.

En este caso, una posibilidad que deberás tener en cuenta es **usar la misma página** que muestra el formulario como destino de los datos. Si tras comprobar los datos éstos son correctos, se reenvía a otra página. Si son incorrectos, se rellenan los datos correctos en el formulario y se indican cuáles son incorrectos y por qué.

Para hacerlo de este modo, tienes que comprobar si la página recibe datos (hay que mostrarlos y no generar el formulario), o si no recibe datos (hay que mostrar el formulario). Esto se puede hacer utilizando la función **isset** con una variable de las que se deben recibir (por ejemplo, poniéndole un nombre al botón de enviar y comprobando sobre él). En el siguiente código de ejemplo se muestra cómo hacerlo.

****

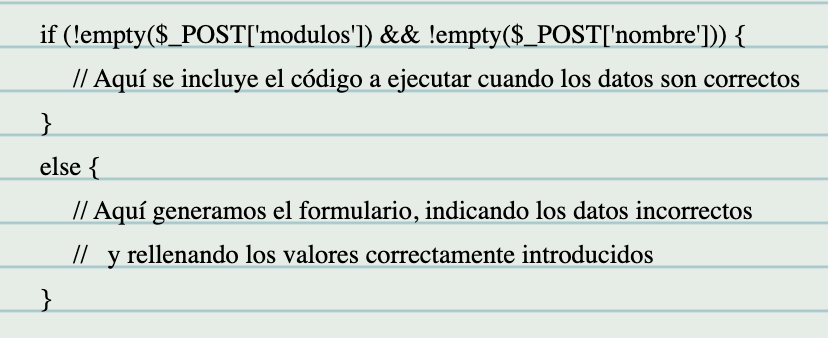
Fíjate en la forma de englobar el formulario dentro de una sentencia **else** para que sólo se genere si no se reciben datos en la página. Además, para enviar los datos a la misma página que contiene el formulario puedes usar **$\_SERVER['PHP\_SELF']** para obtener su nombre; esto hace que no se produzca un error aunque la página se cambie de nombre.

#### Formularios Web en PHP

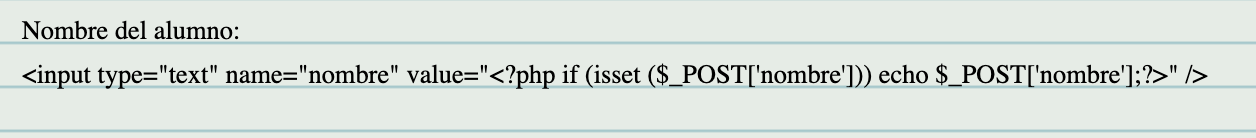
Vamos a volver sobre el ejemplo anterior, revisando los datos que se obtienen antes de mostrarlos. Concretamente, tienes que comprobar que el **nombre** no esté vacío, y que se haya seleccionado como **mínimo uno** de los módulos.

Además, en el caso de que falte algún dato, deberás generar el formulario **rellenando** aquellos datos que el usuario haya introducido correctamente.

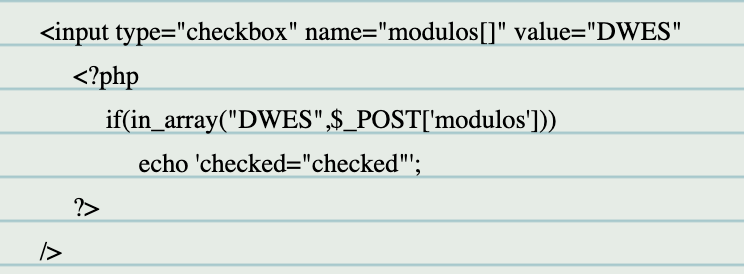
Lo primero que tienes que hacer es la validación de los datos. En el ejemplo propuesto será algo así como:



Para que el usuario no pierda, después de enviar el formulario, los datos correctamente introducidos, utiliza el atributo **value** en las entradas de texto (utilizamos la función isset para comprobar que la variable que queremos mostrar existe, para que no nos muestre un mensaje de error si es la primera vez que se carga la página y todavía no ha enviado nada el formulario):

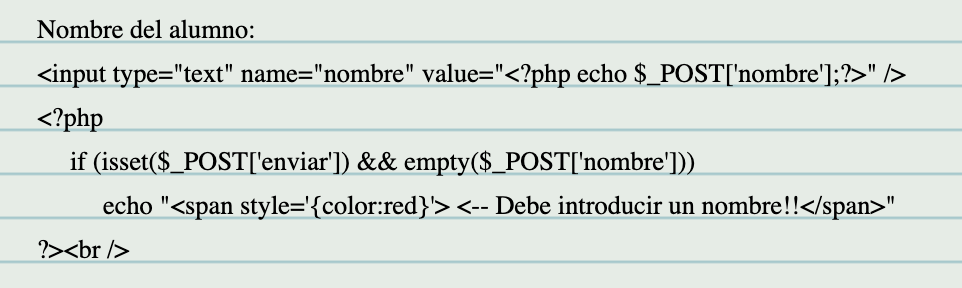


Y el atributo **checked** en las casillas de verificación:



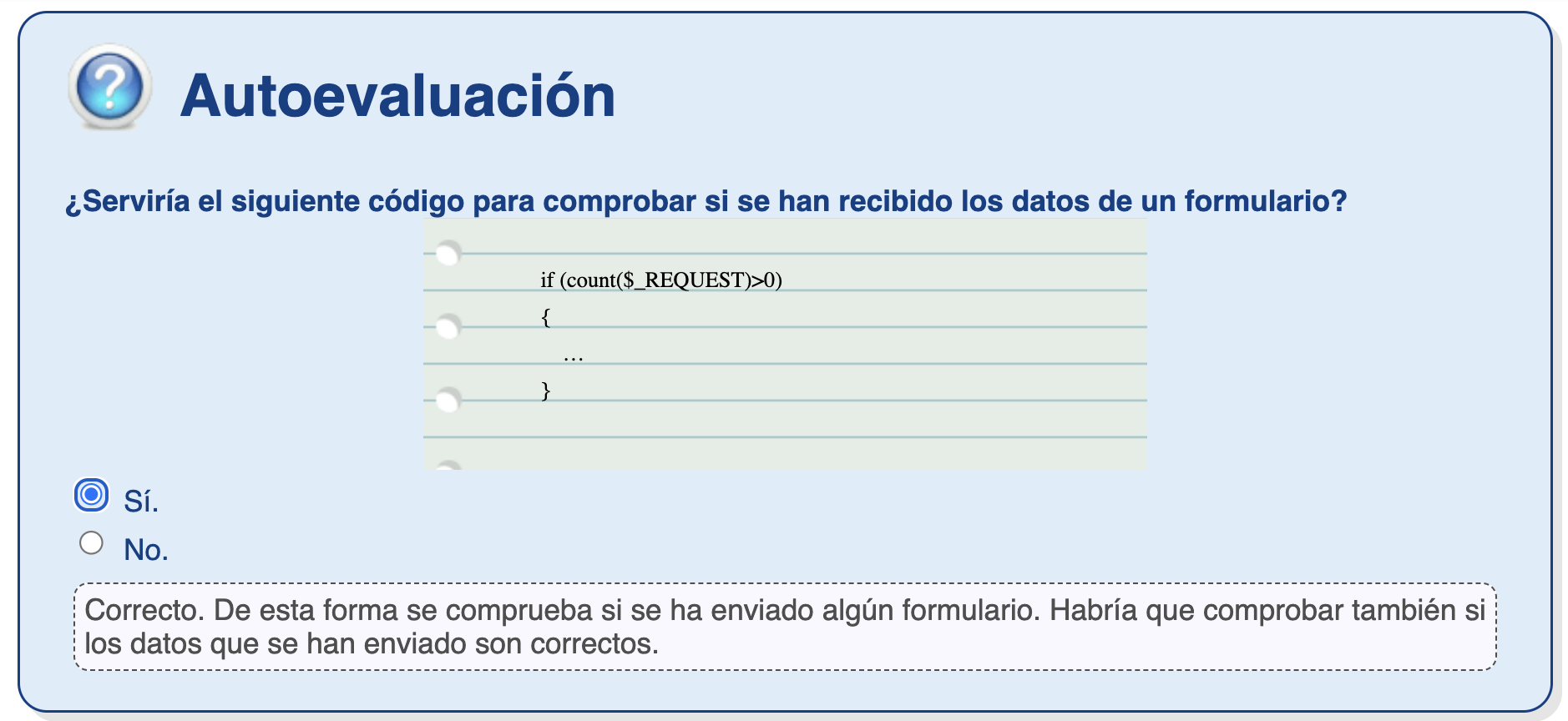
Fíjate en el uso de la función **in\_array** para buscar un elemento en un array.

Para indicar al usuario los datos que **no ha rellenado** (o que ha rellenado de forma incorrecta), deberás comprobar si es la primera vez que se visualiza el formulario, o si ya se ha enviado. Se puede hacer por ejemplo de la siguiente forma:



Revisa el documento con el ejemplo completo, fijándote en las **partes** que hemos comentado anteriormente.

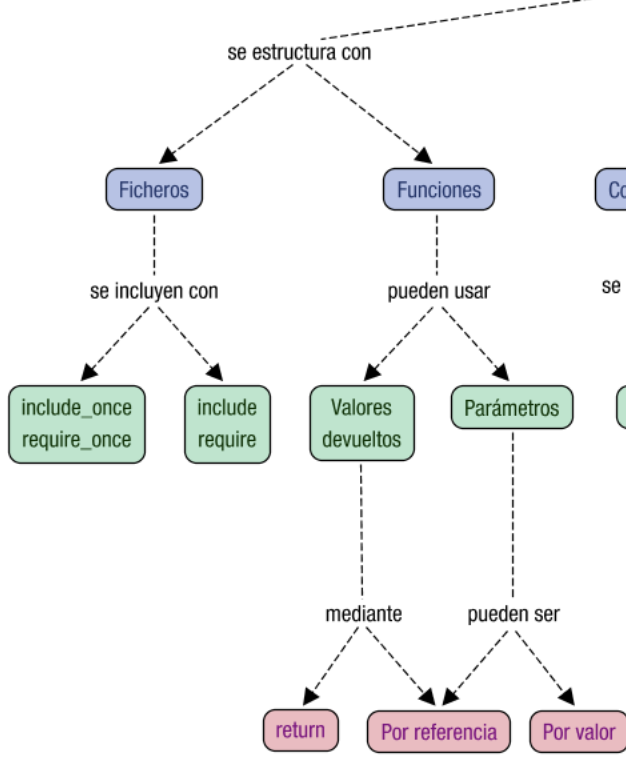
Una forma de enviar información de una página PHP a otra, es incluyéndola en **campos ocultos** dentro de un formulario.

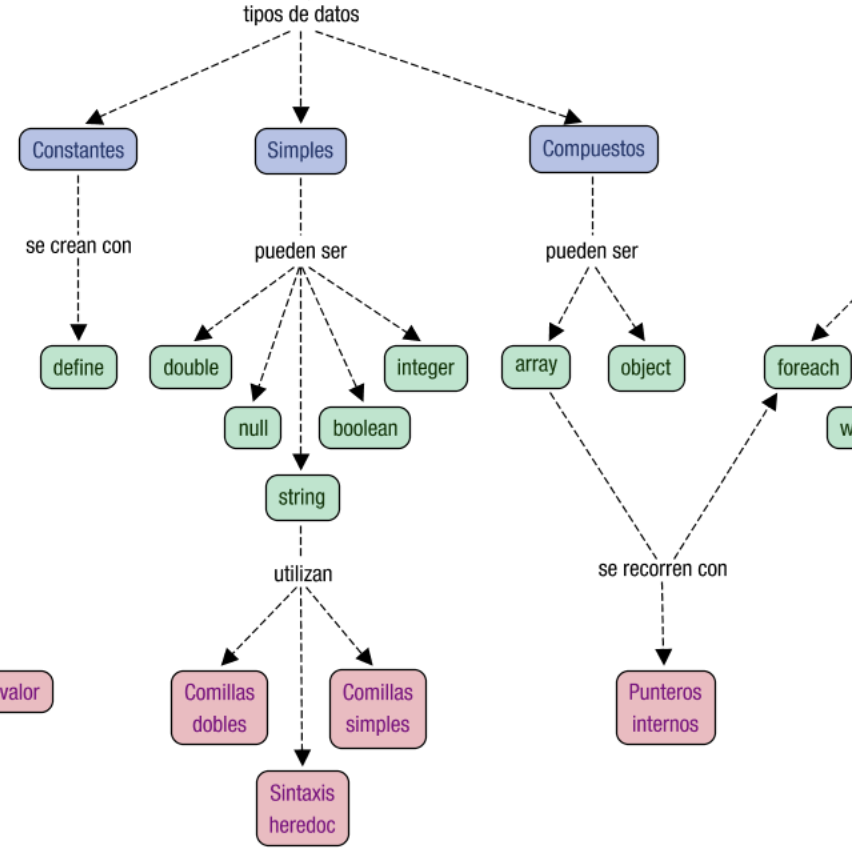


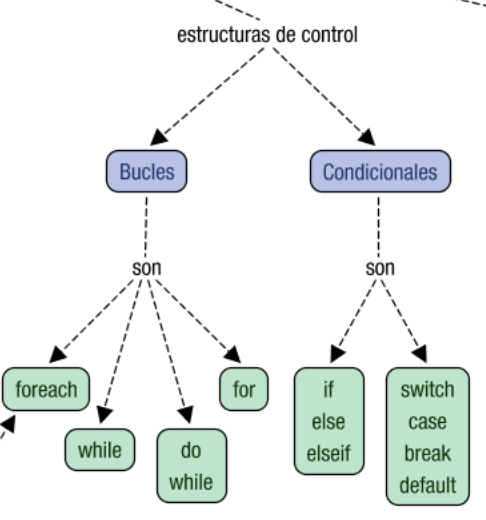
### Mapa Conceptual

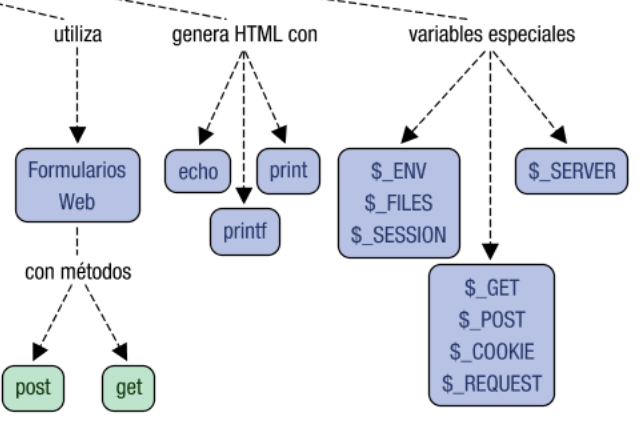
Se presenta ahora el mapa conceptual de la unidad, que se disecciona en las diferentes partes para que pueda ser visualizado mejor:











### Temporalización

La presente unidad está diseñada su desarrollo en **26 horas lectivas**.

### Resultados de Aprendizaje

En esta unidad se han tratado los siguientes Resultados de Aprendizaje y Criterios de Evaluación

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **RA** | **RESULTADOS APRENDIZAJE** | **CE** | **CRITERIOS DE EVALUACIÓN** | **CB** | **CONTENIDOS BÁSICOS** |
| **RA3** | Escribe bloques de sentencias embebidos en lenguajes de marcas, seleccionando y utilizando las estructuras de programación. Criterios de evaluación: | **A** | Se han utilizado mecanismos de decisión en la creación de bloques de sentencias. | **3** | **Programación basada en lenguajes de marcas con código embebido:** |
| **B** | Se han utilizado bucles y se ha verificado su funcionamiento. | **3.1** | Sentencias. Tipos. Bloques. |
| **C** | Se han utilizado «arrays» para almacenar y recuperar conjuntos de datos. | **3.2** | Comentarios. |
| **D** | Se han creado y utilizado funciones. | **3.3** | Tomas de decisión. |
| **E** | Se han utilizado formularios Web para interactuar con el usuario del navegador Web. | **3.4** | Bucles. |
| **F** | Se han empleado métodos para recuperar la información introducida en el formulario. | **3.5** | Tipos de datos compuestos: Arrays. |
| **G** | Se han añadido comentarios al código. | **3.6** | Funciones. Parámetros. |
|  |  |  |  | **3.7** | Recuperación y utilización de información proveniente del cliente Web. |
|  |  |  |  | **3.8** | Procesamiento de la información introducida en un formulario. Métodos POST y GET. |

### Bibliografía

* Unidad 2: Características del Lenguaje PHP; materiales a distancia de la **Junta de Andalucía - Consejería de Desarrollo Educativo y Formación Profesional.**
  + Materiales desarrollados inicialmente por el **Ministerio de Educación, Cultura y Deporte** y actualizados por el profesorado de la **Junta de Andalucía**
* Curso sobre HTML de aulaClic
* Libro….
* PildorasInformáticas

Libro

### Pendiente

1. Comparar otros recursos: libros, webs..
2. Actividades GitHub
3. Resultados y criterios en puntos
4. Cortar unidad para subir poco a poco
5. Anexos
6. Diapositivas
7. Porfolio