PHP PANDAS

EL LENGUAJE DE PROGRAMACIÓN PHP PARA PRINCIPIANTES

DAYLE REES
TRADUCCIÓN POR ANTONIO LAGUNA

PHP Pandas (ES)

El lenguaje de programación PHP para principiantes

Dayle Rees y Antonio Laguna

Este libro está a la venta en http://leanpub.com/php-pandas-es

Esta versión se publicó en 2016-01-13

¡Twitea sobre el libro!

Por favor ayuda a Dayle Rees y Antonio Laguna hablando sobre el libro en Twitter!

El tweet sugerido para este libro es:

I’m learning about PHP and Pandas AT THE SAME TIME. You can too! @ http://leanpub.com/php-pandas #PHPPandas @daylerees

El hashtag sugerido para este libro es #PHPPandas.

Descubre lo que otra gente está diciendo sobre el libro haciendo click en este enlace para buscar el hashtag en Twitter:

https://twitter.com/search?q=#PHPPandas

Índice general

Agradecimientos

Antes que nada, me gustaría agradecer a mi novia Emma, no solo por animarme con todas mis aventuras, ¡si no también por hacer esas increíbles fotos a los pandas rojos para ambos libros! ¡Te amo Emma!

Gracias a mis padres, ¡que han estado apoyando mis esfuerzos con estas cajas para hacer números durante 30 años! ¡También gracias por comprar un billón de copias del primer libro para la familia!

Gracias a todo el que haya comprado mis otros libros, Code Happy y Code Bright, y a toda la comunidad de Laravel. Sin vuestro soporte nunca habría tenido confianza en seguir escribiendo.

Errata

Este puede ser mi tercer libro y mi escritura puede haber mejorado desde la última vez, pero te aseguro que habrá muchos, muchos errores. Puedes ayudarme a apoyar el libro enviándome un correo con cualquier error que encuentres a sombragriselros@gmail.com[1] junto con el título de la sección.

Los errores serán corregidos conforme vayan siendo descubiertos. Las correcciones serán lanzadas con actualizaciones futuras del libro.

[1] mailto:sombragriselros@gmail.com

Feedback

De la misma forma, puedes enviarme cualquier feedback que tengas sobre el contenido del libro o lo que quieras, enviando un correo a sombragriselros@gmail.com[2] o un tweet a @belelros. Me esforzaré en responder a todo correo que reciba.

[2] mailto:sombragriselros@gmail.com

Traducciones

Si quieres traducir PHP Pandas a tu idioma, por favor envíame un correo a me@daylerees.com[3] con tus intenciones. Ofreceré un 50/50 de los beneficios de la copia traducida, que tendrán el mismo precio que la copia en Inglés.

El libro está escrito en formato markdown.

[3] mailto:me@daylerees.com

¡Hola! ¡Estoy seguro de que eres el/la lector más guap@ Y/O bell@ del planeta! Me alegra que te hayas hecho con PHP Pandas y comiences el camino hacia tu carrera como programador web de fama mundial.

¿Quién soy? Es una pregunta sencilla. Me llamo Dayle y seré el autor de esta aventura. He estado escribiendo libros para principiantes durante unos años a estas alturas y he tomado a otros encantadores lectores como tú en las aventuras de aprendizaje de nuevas habilidades. Haremos descubrimientos juntos y, durante el camino, estaré a tu lado.

> ¿Por qué escribes como una persona que está mal de la azotea?

¿Perdona? Ah eso... Bueno verás, es la única forma en la que sé escribir. Si buscas un libro técnico con palabras raras, lamento decirte que has venido al lugar equivocado. Escribo libros para personas. Quiero pensar que somos colegas, sentados en un bar y hablando sobre PHP con una pinta de cerveza.

Lo cierto es que a los principiantes para los que he escrito antes les gusta mi estilo de escritura. No intentan sacarse la carrera de matemáticas con el libro si no aprender un par de cosas sobre PHP, y eso, ¡te lo puedo prometer!

Oh, te darás cuenta de que estamos hablando ahora mismo. Eso no lo consigues con otros autores, ¿no? Tengo este poder mágico que te hace hablar conmigo y respondo a tus preguntas.

> Espera, ¿cómo haces es...

Eso sería contarte mi secreto. Aun no podemos compartir eso pero, ¿acaso no sienta bien saber que formas parte de la aventura y no eres un mero espectador?

> Supongo... Vamos a probar.

Excelente.

Este es el espacio en el que cualquier otro libro te hablaría sobre PHP, su historia, su aplicación, su autor y un millón de cosas más. Ya te he dicho que no soy el autor más tradicional y esos capítulos no me gustan y me parecen aburridos. Has comprado este libro para aprender PHP, así que ya tienes algo de curiosidad. Creo que es todo lo que necesitas.

PHP es un lenguaje de programación que usan la mayoría de los sitios en internet. Fue escrito originalmente por alguien llamado Rasmus Lerdorf, al cual puedes ver sonriendo en la mayoría de las fotos que encuentres de él en Google. Rasmus es un gran tío y le agradezco cada día por este lenguaje pero esto es todo lo que creo

que tienes que saber sobre él. Otros libros probablemente te hablarían sobre sus cereales preferidos para el desayuno, pero ¿comenzamos ya a aprender?

Este libro es para principiantes ***absolutos***. Esto significa que si nunca has programado antes en tu vida, ¡estás de suerte amigo mío! Si ya has intentado programar te irá bien también. Si eres un experto en PHP quizá sea hora de refrescar tus habilidades y quizá descubras algunas cosas nuevas en este camino.

He usado a mi novia, a mis amigos no-programadores y a gente aleatoria en la calle, obligándoles a leer el libro como ratas de laboratorio para ver cómo funciona con gente que no tiene idea alguna de PHP. Mis pequeñas ratas lo han hecho genial, ahora es tu turno.

La meta de este libro es que se convierta en el libro más divertido, práctico y fantástico sobre PHP en el mercado. Quiero que sea **el** libro que se recomienda cuando alguien comience a aprender PHP. He trabajado duro para hacer que sea accesible para todos por lo que si disfrutas de esta aventura, te agradecería que escribieras sobre el libro en Twitter, tu blog o le regales copias a tus amigos y familiares.

Este libro es un libro de sintaxis para PHP. No te va a enseñar a hacer webs (estoy trabajando en otro título para esto), es el primer paso que te ayudará a sentar una buena base de conocimiento del lenguaje para que a la hora de crear tu primera web, seas bueno.

Si lees el libro y descubres que falta algo, un capítulo en concreto es confuso o hay algo que te moleste, por favor envíame un correo a `me@daylerees.com` para hacérmelo saber. ¡Quiero que el libro sea perfecto para todos!

Si lees el libro y no encuentras nada malo, bueno... mándame un correo y me cuentas que lo has disfrutado!

No gastemos más tiempo. ¡Aprendamos algo! Pasa de página e imagina la música de Jurassic Park en el momento en que se abren las puertas, y prepárate para entrar en el mundo del desarrollo.

1. Instalación

Antes de empezar a trabajar con PHP, primero tenemos que instalarlo. Como verás, PHP es una aplicación como otra cualquiera. Necesita ser instalada en nuestro sistema antes de que pueda procesar código PHP.

El método de instalación varía ampliamente dependiendo del sistema operativo que estés usando. Por ese mismo motivo, os ofrezco tres guías diferentes para instalar PHP. La primera sección explicará cómo instalar PHP en una distribución de Linux, en Ubuntu concretamente debido a su popularidad. La segunda sección explicará cómo instalar PHP en un sistema Mac OSX de Apple. Finalmente, la tercera sección explicara cómo instalar PHP en Windows.

Únicamente vamos a instalar la versión de consola de PHP. No vamos a crear un servidor web aun. Ya llegaremos a eso. La versión de consola de PHP es todo lo que necesitamos para comenzar con nuestro proceso de aprendizaje.

Recuerda, solo tienes que leer la sección adecuada a tu equipo. Una vez que tengas PHP instalado, salta al siguiente capítulo del libro.

Linux

La mejor forma de instalar PHP en una distribución de Linux es usar un gestor de paquetes. El gestor de paquetes disponible depende ampliamente de la distribución de Linux que tengas. He decidido facilitar instrucciones para instalar PHP en Ubuntu, una de las distribuciones de Linux más populares hoy en día.

Ubuntu usa el gestor de paquetes conocido como `apt` para instalar sus paquetes. Para instalar la versión de consola de PHP, tenemos que instalar el paquete `php5-cli`. Vamos a hacerlo. Abre un nuevo terminal antes de nada. Escribe lo siguiente en la ventana del terminal.

```
1 $ sudo apt-get install php5-cli
```

No tienes que escribir el símbolo del dolar `$`, es simplemente lo que se suele ver en el terminal y así marcamos que estamos escribiendo en la consola. Una vez que pulses enter, `apt` obtendrá el paquete PHP y lo instalará por ti.

¡Ya está! Has terminado. Bueno, o deberías. Vamos a revisar, ¿vale? Escribe simplemente...

```
$ php -v
```

Este comando se usa para mostrar la versión de PHP instalada actualmente. Deberías ver algo similar a esto.

```
PHP 5.5.13 (cli) (built: Jun  5 2014 19:13:23)
Copyright (c) 1997-2014 The PHP Group
Zend Engine v2.5.0, Copyright (c) 1998-2014 Zend Technologies
```

La tuya no será exactamente igual, después de todo, somos todos diferentes, ¿no te parece? En el ejemplo de arriba la versión es `5.5.13`. Espero que tu versión de PHP sea `5.4.0` o superior.

Si la versión no es la correcta, tendrás que consultar la documentación de tu distribución de Linux para descubrir cómo instalar la versión adecuada.

Ve y salta al próximo capítulo, ¡has terminado!

Mac OSX

En las máquinas Mac, PHP viene instalado de fábrica. Ve y abre el terminal y escribe lo siguiente para descubrir la versión de PHP que estás usando.

```
$ php -v
```

No escribas el símbolo del dolar, ¡es la marca del terminal! Deberáis ver algo similar a esto, pero no exactamente lo mismo.

```
PHP 5.4.24 (cli) (built: Jan 19 2014 21:32:15)
Copyright (c) 1997-2013 The PHP Group
Zend Engine v2.4.0, Copyright (c) 1998-2013 Zend Technologies
```

La versión de PHP del ejemplo de arriba es `5.4.24`. Mientras tu versión de PHP sea `5.4.0` o superior, has terminado y puedes ir al siguiente capítulo.

Si no, vamos a usar un gestor de paquetes de terceros para OSX para instalar una nueva versión de PHP.

Vamos a usar un gestor de paquetes llamado 'Homebrew' o tan solo 'Brew'. Para instalarlo, sigue las instrucciones que puedes encontrar en el siguiente sitio:

brew.sh[1]

No quiero copiar las instrucciones aquí ya que suelen cambiar con cada nueva versión. Una vez que tengas Homebrew instalado, es hora de instalar una versión más nueva de PHP. Recomiendo instalar la versión `5.5`. Puedes hacerlo usando el siguiente comando.

[1] http://brew.sh/

```
$ brew install php55
```

Luego, tendrás que añadir la ubicación de esta versión de PHP al PATH de tu sistema. No te preocupes, escribe lo siguiente.

```
$ PATH=~/usr/local/Cellar/php55/5.5.13/bin:$PATH
```

Puede que tengas que actualizar el número de la versión para que coincida con lo que Homebrew te ha instalado. Ahora vamos a volver a mirar la versión de PHP.

```
$ php -v
```

Espero que, esta vez, tengas una versión de PHP más alta que `5.4.0`. Ve y salta al siguiente capítulo.

Windows

Instalar PHP en Windows es un poco más difícil, al menos lo es para mi. He probado las instrucciones que he escrito en mi máquina con Windows 10, pero si tienes alguna dificultad replicando estos pasos, házmelo saber y encontraré a alguien que tenga más experiencia con Windows y reescriba esta sección.

Primero ve a:

http://windows.php.net/download[2]

Allí descárgate la última versión de PHP, que sea al menos la 5.4 en un archivo zip. Una vez que se haya descargado el archivo, descomprímelo en un lugar adecuado. Yo lo puse aquí:

```
C:\Users\Dayle\PHP
```

Necesitas una línea de comandos para ejecutar los scripts que escribamos en este libro. He aquí una buena forma de ejecutar una línea de comandos en Windows.

Haz click derecho sobre tu escritorio o cualquier carpeta y elige `Crear Acceso Directo`. En la caja que aparece escribe:

[2] http://windows.php.net/download

```
cmd.exe
```

Haz click en siguiente y nombra el acceso directo como `PHP`.

Finalmente, dale click derecho sobre el acceso directo y haz click en “Propiedades”. En la pestaña de “Acceso Directo”, cambia el campo “Comenzar en”, para que coincida con el directorio donde ubicamos el archivo de PHP. Haz click en “Ok” cuando termines.

Ahora haz doble click en tu fichero y deberías ver la línea de comandos. Escribe…

```
php -v
```

…y deberías ver la información sobre la versión de PHP. Confirma que la versión es igual o superior a PHP 5.4 y pasa al siguiente capítulo.

Lamento las pocas explicaciones en este subcapítulo. No he usado Windows como máquina para programar en unos cuantos años. Si alguien conoce una mejor manera de ejecutar PHP en Windows, que me envíe un correo con instrucciones y me encargaré de actualizarlo.

2. Encontrando respuestas

Lo sé. Es un título un poco raro, ¿no crees? Vas a tener que confiar en mi cuando digo que esto es importante. Este capítulo es sobre tu confianza como programador en desarrollo. Aprender es duro, pero no te preocupes; voy a echarte una mano.

Los programadores son robots

¿Por qué decidiste convertirte en programador? ¡No, espera! Déjame adivinar. Viste bajarse a un programador PHP de una Limusina en uno de los mejores hoteles de Nueva York, pidió 5 botellas de Champán y pasó la noche relajándose con Jay-Z y el fantasma de Tupac.

Es cierto, la vida del programador es glamurosa. Tengo que escribir estos capítulos en las 5 horas de sobriedad que tengo al día. Probablemente has visto a un programador escribiendo código y pensaste…

> Oh dios, ese programador debe ser un robot. Sabe todas esas palabras del código y funciones y cómo funcionan.

Cuando la gente sin experiencia en programación se acercan a los programadores, asumen que son genios con matrícula de honor en matemáticas. Quizá esto sea cierto para algunos programadores, pero ciertamente no lo es para mi. Quiero pensar que otros programadores están en mi misma situación.

La verdad es, que no somos perfectos. Ni siquiera estamos cercas de serlo. Si piensas que los programadores saben todas las funciones de PHP y los fragmentos de memoria, te estás engañando a ti mismo si piensas que nunca podrás hacerlo.

Simplemente, no es cierto. No memorizamos todo. De hecho, la mayoría del código que usamos día a día es de una referencia. Somos guerreros de Google. Hay funciones en PHP que hacen cosas sencillas con cadenas de texto y busco en la documentación de PHP casi cada semana para saber el orden de los parámetros que tengo que pasar.

Cuando estoy completamente perdido, intento usar Google para ver si otro programador descubrió un problema similar. A menudo encontraré una solución buena que otro haya descubierto, o suficiente información para llevarme a una solución. Por supuesto esto funciona en el otro sentido también, intentaré dar mis soluciones

a la comunidad. Publicaré respuestas en Stack Overflow y contribuiré en los foros o discusiones. Es importante colaborar con la comunidad.

Como ves, no somos robots. No lo sabemos todo sobre el lenguaje, y no tenemos una solución a cada problema. Sin embargo, somos buenos investigadores. Somos oportunistas. Somos resolutores de problemas con muchos recursos. Somos programadores.

El arte de usar Google

Cuando la gente te dice que busques algo en Google, es fácil tomárselo como un insulto. ¿O quizá sarcasmo? No lo es. Google es nuestra página de inicio por un buen motivo. Vamos a ver cómo podemos encontrar respuestas a problemas de programación comunes.

Estamos escribiendo un programa, y en algún sitio tenemos que darle la vuelta a una sentencia por lo que 'Los pandas molan' se convierte en 'nalom sadnap soL'. No tenemos ni idea de cómo hacerlo. Acabamos de empezar con PHP.

Sabemos que en PHP una secuencia de texto se llama 'cadena'. Lo sabemos porque no dejamos de leer este libro con ejemplos de Pandas, y descubrimos esto en otro capítulo, ¿verdad?

Ahora sabemos lo que queremos hacer. Queremos invertir una cadena. Vamos a hacer una consulta a Google.

```
1 invertir cadena
```

¡No, espera! El problema es que hay miles de lenguajes de programación. Los ordenadores llevan mucho entre nosotros.

Si buscamos 'invertir cadena', vamos a obtener resultados para C++, ASP.NET, Erlang, lo que sea. Nuestro foco es PHP. No nos importan el resto de lenguajes. Ya tendremos tiempo de jugar con ellos más tarde cuando nos convirtamos en cerebritos de PHP. Vamos a arreglar el problema añadiendo el lenguaje a la consulta.

```
1 invertir cadena php
```

Perfecto. Echemos un vistazo a los resultados que obtenemos de Google. Puede ser un buen momento para mencionar que no trabajo para Google, y no estoy trabajando a comisión. Puedes usar Bing si quieres aunque quizá acabes comprando un trailer para caballos usados antes de encontrar la función que buscas. Así que, ¿dónde están los resultados?

Invierte una string - PHP

http://php.net/manual/es/function.strrev.php[1]

Invertir una cadena (string) en PHP | Esteban Novo

http://www.notasdelprogramador.com/2010/09/10/invertir-una-cadena-en-php/[2]

Haciendo la pregunta adecuada, obtenemos recursos útiles en respuesta. El manual de PHP (a veces conocidos como la documentación de PHP) y Stack Overflow son dos de los mejores recursos para solucionar problemas de PHP en internet. No digo que siempre tengan la respuesta adecuada. Hay otros sitios buenos también, pero estoy seguro de que acabarás visitando esos sitios muchas veces.

> N. del T. : Como nota personal, he de añadir que hacer las búsquedas en inglés, suele llevar a mejores resultados.

Ahora mismo estamos buscando alguna forma de invertir una cadena. No estamos intentando resolver un problema abstracto, sabemos exactamente lo que queremos.

Ve y prueba el primer enlace, serás recibido por la página del manual del PHP para la función `strrev()`. No necesitas saber lo que es una función ahora mismo. No te preocupes si esto te resulta demasiado.

Una vez que te hayas familiarizado con las funciones, verás que esta página ofrece todo lo que necesitamos saber para usar la función `strrev()`, y ejemplos sobre cómo usarla.

Como ves, haciendo las preguntas adecuadas, hemos recibido toda la ayuda que necesitábamos para continuar con nuestro trabajo. No teníamos conocimiento alguno sobre `strrev()` pero sabíamos el problema que teníamos que resolver. Eso fue suficiente para llevarnos a una solución. No importa si tenemos que volver a la página más tarde.

Quizá no usemos la función con la frecuencia necesaria para tener que recordar cómo usarla. Aunque, te darás cuenta que si comienzas a usar la función más y más, y frecuentas la página del manual, antes de que te des cuenta, dejarás de tener que usar el manual. Pensarás enseguida, 'Ey, debería usar `strrev()` que uso siempre y sé cómo funciona'. Será parte de tu memoria y de tu conjunto de conocimientos.

[1] http://php.net/manual/es/function.strrev.php

[2] http://www.notasdelprogramador.com/2010/09/10/invertir-una-cadena-en-php/

La lección que espero que hayas aprendido es que no deberías entrar en pánico. No tienes que recordarlo todo, es perfectamente natural buscar ayuda. De hecho, es humano, y es humano aprender de tu experiencia.

¡Enhorabuena! Eres un humano, no un robot.

3. Archivos

He aquí una noticia impactante. El código PHP se almacena en archivos. Lo siento, ¡pero es verdad! Vas a trabajar con muchos archivos. Bueno, de hecho, a veces uno solo pero más tarde trabajarás con muchos, muchos de ellos.

Ahora que te he mostrado la cruda realidad, es hora de que aprendas a crear un archivo PHP.

> Dayle„ entiendo lo fundamental sobre el sistema de ficheros de un ordenador

¡Bien hecho! Bien por ti, pero no vamos por ahí. Como verás, la mayoría de los archivos PHP tienen algo en común. Estoy hablando sobre la etiqueta de PHP.

Échale un vistazo:

Ejemplo 01: Etiqueta PHP.

```
1 <?php
```

Preciosa, ¿no te parece? Una etiqueta maravillosa. Un especimen absolutamente fantástico.

> Yo… esto…

¿Qué? ¿No sientes lo mismo sobre ella? Confía en mi, tras muchos años de desarrollo con PHP sentirás lo mismo. La verás cuando cierres los ojos para ir a dormir por las noches. Es tu mejor amiga. Te permite usar PHP.

Soy de los que prefieren ejemplos prácticos así que intentemos algo juntos. Crea un nuevo archivo, llámalo `prueba.php`. Los archivos PHP normalmente tienen la extensión `.php`. Para ser sincero, podemos ejecutar PHP sin la extensión, pero deberías usarla si no quieres que los programadores mayores se rían de ti, te quiten la merienda y te hagan llorar. Solo bromeaba… los programadores son gente afable, pero deberías usar la extensión.

Lo primero, vamos a escribir las palabras…

Ejemplo 02: Algo de texto.

```
¡Los pandas molan!
```

...en el archivo, y guárdalo.

Genial, ahora ejecuta el archivo. Podemos hacerlo llamando a la aplicación `php` de la línea de comando y pasando el nombre del fichero como parámetro. Por ejemplo, en mi Mac, escribiré lo siguiente:

Ejemplo 03: Ejecutando un fichero PHP.

```
php prueba.php
```

Verás las palabras `¡Los pandas molan!` en la pantalla. Esto es porque todo lo que esté fuera de las etiquetas PHP, es impreso cuando la aplicación se ejecuta. Intentemos otra cosa. Vamos a usar nuestra primera etiqueta PHP.

Editemos el fichero para que ponga lo siguiente:

Ejemplo 04: Segmento PHP.

```
<?php

// ¡Los pandas son increíbles!

?>
¡Los pandas molan!
```

Ejecutemos el archivo de nuevo. ¿Qué es lo que vemos?

Ejemplo 05: Salida.

```
¡Los pandas molan!
```

> ¡Ey espera! ¿Dónde está el resto?

¡Bien visto, futuro programador! Hay una sección de nuestro fichero que falta. Esto es porque todo lo que está dentro de nuestras etiquetas PHP, es tratado como código PHP y se procesa de manera adecuada.

Así que, ¿qué son las etiquetas pHP? Bueno, ya has visto la etiqueta de apertura PHP. ¿Recuerdas nuestra amiga `<?php`?. La etiqueta `<?php` marca el inicio de nuestro código PHP. Así que, ¿cuándo termina? Es fácil ver el código PHP en este fichero. Es la siguiente línea.

Ejemplo 06: Comentario.

```
1 // ¡Los pandas son increíbles!
```

Entonces, ¿qué es lo que hace esta línea? Absolutamente nada. Es conocida como comentario. Ayuda a los programadores a documentar su propio código. No te preocupes. Aprenderemos más sobre los comentarios más adelante.

Bueno, este ha sido un bonito y corto capítulo, ¿no te parece? Ahora ha llegado la hora de las buenas noticias. En el siguiente capítulo vas a escribir las primeras líneas de código PHP ***reales***.

¿Nervioso? ¿¡Entonces por qué esperar?! Pasa al siguiente capítulo.

4. Aritmética básica

Estoy seguro de que has escuchado que la programación es todo matemáticas. ¿No es cierto? Ha llegado la hora de las matemáticas. Comencemos.

$$\left|\sum_{i=1}^{n} a_i b_i\right| \le \left(\sum_{i=1}^{n} a_i^2\right)^{1/2} \left(\sum_{i=1}^{n} b_i^2\right)^{1/2}$$

Ahora, resuelve la X.

Bromeaba de nuevo. De hecho, no hay X en esa ecuación. De hecho, ni siquiera es una ecuación por lo que ha sido una broma terrible. Lo cierto es, que ni siquiera sé lo que hace esa ecuación. No somos gurús de las matemáticas (no todos).

Sentencias

Vamos a intentar algo más cercano a mi nivel de matemáticas. Sabes cómo hacer ficheros PHP y sabes cómo abrir y cerrar etiquetas PHP. Así que saltemos directamente a un fichero PHP. Vamos a llamarlo `mates.php`. He aquí el contenido.

Ejemplo 01: Suma.

```
1  <?php
2          3 + 3;
3  ?>
```

Un momento, espera un segundo. No vamos a imprimir nada tras nuestro código PHP. ¿Por qué preocuparnos en poner una etiqueta PHP de cierre? Lo cierto es, que la mayoría de los programadores PHP, omiten esta etiqueta si no hay contenido que siga al código. Vamos a hacerlo.

Ejemplo 02: No necesitamos etiqueta de cierre.

```
1  <?php
2          3 + 3;
```

¡Mucho mejor!

Bueno, en el caso de que tus matemáticas no sean tan buenas como las mías, déjame ayudarte un poco. Cuando sumas tres y tres, te da seis. Vale, ahora estás listo.

La línea `3 + 3;` contiene una sentencia. Es una línea de PHP que será evaluada por PHP. Normalmente terminan con punto y coma. Así es como se ven `;`. Al principio se te olvidará siempre, pero no te preocupes, pronto estarás incluso terminando tus frases con ellos;

Dado que ahora entiendes la suma básica, ¿qué piensas que pasará cuando ejecutemos el fichero?

> Siete punto cinco.

Bien, veamos si estás en lo cierto. Ve y ejecuta `php mates.php` para ver qué ocurre.

Ejemplo 03: Salida.

```
1 [Nada por aquí]
```

¡Vaya! Absolutamente nada. Este lenguaje es estúpido. Dejémoslo. Vale, sigo bromeando. Tengo un sentido del humor peculiar, no te preocupes, te acostumbrarás.

¿Por qué no vimos nada en la pantalla? Bueno, eso es porque no le dijimos a PHP que nos mostrara nada. PHP es obediente. Vamos a decirle que nos de la respuesta. Usemos `echo`. Es una construcción del lenguaje que nos permite ver el resultado de una sentencia.

Vamos a alterar la sentencia para incluir `echo`.

<?php echo 3 + 3;

Allá vamos. Colocamos `echo` delante de la sentencia de la que queremos ver el resultado. Vamos a intentar ejecutar nuestra aplicación de nuevo. Allá vamos.

Ejemplo 05: Salida.

```
1 6
```

¡Woohoo! ¡Seis! **¡NO SIETE PUNTO CINCO!** Hemos conseguido ver el resultado de nuestra primera sentencia evaluada con PHP. Maravilloso, ¿no te parece?

> Podría haber hecho esto en una calculadora

Lo sé, lo sé. No es exactamente ciencia para cohetes. La ciencia para cohetes se cubre en otro capít… Espera, ya he dicho esa broma en otro libro. Necesito conseguir nuevo material.

Operadores aritméticos

Sé que nuestro ejemplo de `3 + 3` es código sencillo pero pronto nos haremos con cosas más grandes y mejores. ¿Sabías que hay más operadores aritméticos? Estoy seguro de que algunos de ellos te suenan de algo.

```
+ Suma
- Resta
* Multiplicación
/ División
% Módulo
```

Estoy seguro de que habrás visto algunos de esos operadores anteriormente. Sé que la multiplicación y la división son un poco diferentes a los signos que aprendiste en el colegio. Esto es común a la mayoría de los lenguajes de programación y descubrirás que el signo de división es definitivamente más sencillo de escribir en el teclado. No dejes que te preocupen, antes de que te des cuenta te habrás acostumbrado a ellos.

Si no has usado el operador 'Módulo' anteriormente, es sencillo de explicar. Puede ser usado para calcular el resto de una división. Por ejemplo, la operación '3 % 2' resultaría en la figura de '1'. Se usa normalmente para determinar si un número es par o impar, dividiéndolo entre dos.

Ahora vamos a darle a PHP algo difícil:

Ejemplo 06: Matemáticas difíciles.

```php
<?php
echo 4 + 3 * 2 / 1;
```

Así que, ¿cuál es el resultado? Bueno, puede ser difícil calcularla en nuestra mente ya que no sabemos en qué orden calcular. ¿Deberíamos sumar primero? ¿O dividir primero? Hmm. ¡Difícil!

Por supuesto, en matemáticas aprendemos a usar paréntesis para separar las ecuaciones. Podemos hacer lo mismo con PHP. Vamos a intentarlo.

Ejemplo 07: Paréntesis para separar operaciones.

```php
<?php
echo (4 + 3) * (2 / 1);
```

Ahora podemos estar seguros de que `4 * 3` y `2 / 1` son evaluados primero y los resultados serán multiplicados. Genial, ejecutemos y veamos el resultado...

Ejemplo 08: Salida.

```
14
```

Genial, ¿pero eso no es hacer trampas? ¿Qué conseguiríamos sin los paréntesis? Vamos a quitarlos.

Ejemplo 09: Sin paréntesis.

```
<?php
echo 4 + 3 * 2 / 1;
```

¿Cuál es el resultado? Ejecutémoslo.

Ejemplo 10: Salida.

```
10
```

Vaya, esto es algo totalmente distinto. ¿Por qué? Bueno, eso es porque PHP no está gestionando los operadores en el mismo orden. Vamos a ver el orden en que se gestionan los operadores.

Este es el orden:

```
* Multiplicación
/ División
% Módulo
+ Suma
- Resta
```

El operador con la prioridad más alta está en la parte de arriba de la tabla. Esto significa que cuando PHP examina `4 + 3 * 2 / 1`, primero calcula `3 * 2 = 6`, luego `6 / 1 = 6` y finalmente `4 + 6` para obtener `10`.

Cuando escribo cosas matemáticas, me gusta añadir paréntesis para evitar confusiones. También me parece que ayuda a clarificar lo que se pretende hacer con la línea, haciendo que sea más legible.

Procedimiento

El código PHP se procesa de manera secuencial. Esto significa que se ejecuta sentencia a sentencia. Aunque es posible poner más de una sentencia en una línea, es poco común y el resto de programadores no suelen hacerlo así que ejecutemos el código línea a línea. Podemos ver esto en acción añadiendo más sentencias a nuestro fichero. Vamos a probar lo siguiente:

Ejemplo 11: Múltiples sentencias.

```
1 <?php
2 echo 2 + 2;
3 echo 3 + 3;
4 echo 4 + 4;
5 echo 5 + 5;
```

Ahora ejecutemos el fichero...

Ejemplo 12: Salida.

```
1 46810
```

¿!CUARENTA Y SEIS MIL GIGAQUEEEE!?

¡Cálmate lector! Le dijimos a PHP que mostrara los resultados, nada de espacios ni nuevas líneas. Esto significa que PHP ha calculado los valores correctamente. Si añadimos espacios, el resultado que PHP nos ha dado es...

4 6 8 10

...ahora vemos que los cálculos son, de hecho, correctos. Únicamente ocurre que PHP es muy obediente y ha mostrado los valores uno detrás del otro.

He mencionado con anterioridad que PHP es un lenguaje flexible. Vamos a ver eso, ¿no te parece? Hasta ahora, nuestras sentencias solo tienen un espacio entre cada 'palabra' (o número). Vamos a añadir algunos espacios adicionales en un formato inconsistente para ver qué ocurre. He aquí nuestro código modificado.

Ejemplo 14: Espacios en blanco.

```
1 <?php
2 echo    2           +   2;
3 echo                   3   +3;
4 echo 4+4;
5 echo    5+     5        ;
```

Aunque esto no es que sea muy bonito, si ejecutaras el código, descubrirías que el código funciona a la perfección. A PHP no le importan la cantidad de espacios en blanco entre las palabras y el código. Símplemente lidia con ellos.

Observarás que algunas de las operaciones aritméticas, por ejemplo `4+4` no necesitan un espacio para nada. Aunque es cierto, no es consistente en todas las variaciones. Por ejemplo, considera lo siguiente:

Ejemplo 15: Sin espacios tras echo.

```
1 <?php
2 echo5 + 5;
```

Si intentas ejecutar esto, descubrirás que PHP lanzará un error 'Use of undefined constant echo5 - assumed 'echo5'. Esto es porque PHP no sabe qué es la palabra `echo5` ni lo que debe hacer. Por este motivo, lo mejor es colocar al menos un espacio entre cada 'palabra'.

Por otro lado, las sentencias. Si fuéramos masoquistas, podríamos elegir poner cada sentencia en la misma línea. He aquí un ejemplo.

Ejemplo 16: Múltiples sentencias, una línea.

```
1 <?php echo 2 + 2; echo 3 + 3; echo 4 + 4; echo 5 + 5;
```

Este código es perfectamente válido, pero no verás a muchos programadores haciéndolo. Tener una sentencia en cada línea, hace el código mucho más sencillo de leer y entender. También causa problemas con los sistemas de control de versiones.

Hemos visto que a PHP no le importa si usas muchos espacios en el código, pero también considera una nueva línea como un espacio en blanco. Esto implica que el siguiente fragmento es completamente legal.

Ejemplo 17: Una sentencia, múltiples líneas.

```
1 <?php
2 echo
3 2
4 +
5 2
6 ;
```

¿No me crees? ¡Ve e inténtalo! Aunque el código funciona como se pretende, no es lo más legible del mundo. Si te pillo escribiendo código así, ¡te daré una azotaina!

No obstante, hay un uso práctico para los saltos de línea. Si la línea que estás escribiendo es demasiado larga, también se convertirá en un problema de legibilidad. Podemos resolver este problema aplicando saltos para que resulte legible. Muchos programadores también aplican cuatro espacios (o tu elección de tabulación) en la siguiente línea, para indicar que es una continuación.

He aquí un ejemplo de saltos de línea apropiados.

Ejemplo 18: Saltos de línea limpios.

```
<?php
echo (3 * 5) / (7 / 12) * (7 * 6) + (7 % 3)
        + (6 + 7) * (12 / 3);
```

Aquí hay matemáticas serias, pero espero que lo veas más sencillo de leer.

Merece la pena destacar que puedes también dejar líneas en blanco para añadir claridad. He aquí un ejemplo.

Ejemplo 19: Líneas adicionales por claridad.

```
<?php
echo 3 + 2;
echo 7 * 7;
echo 5;
```

Así que, como ves, PHP puede ser extremadamente flexible. Pero no te olvides de añadir un punto y coma al final de la línea, ya que nunca te perdonará.

JAMÁS.

5. Variables y asignaciones

Ahora estamos llegando al quid de la cuestión. Las variables son una parte extremadamente útil del conjunto de herramientas del programador. Comencemos, ¿no te parece?

Pequeñas cajas

Quiero que pienses en las variables como en pequeñas cajas en las que metemos cosas. Las variables son palabras que comienzan con el símbolo del dolar: `$`. Veamos un ejemplo.

Ejemplo 01: Asignación básica.

```
1  <?php
2
3  $tres = 3;
```

Si piensas en la variable `$tres` como en una caja, acabamos de meter el valor `3` dentro. Eso es lo que hace el símbolo de igual. En las matemáticas, usamos el símbolo de igual para indicar el resultado de una ecuación. No obstante, en PHP es una historia completamente diferente.

En PHP, el signo igual `=` es conocido como el operator de asignación. Se usa para ***asignar*** algo. Le estamos indicando a PHP que ***asigne*** la variable `$tres` al número `3`.

Si ejecutas el script que hemos creado arriba, verás que PHP no nos muestra nada. Esto es ya que la asignación es eso, una asignación y nada más. No le estamos pidiendo a PHP que nos muestre nada. No obstante, ahora que hemos asignado la variable `$tres` al valor `3`, podemos usar `echo` sobre la variable.

Ejemplo 02: Mostrando un valor

```php
<?php

// Asignamos el valor
$tres = 3;

// Mostramos el valor
echo $tres;
```

Primero asignamos nuestra variable y luego usamos la sentencia `echo` para mostrar el valor que contiene. Si ejecutamos nuestro código ahora, veremos un `3` como salida.

Esto es genial porque significa que podemos ponerles motes a las cosas: ya sabes, igual que a esos molestos gamberros del colegio. Por ejemplo, el número '3.14159265359' es un número precioso para los amantes de los círculos, pero es difícil de recordar, ¿no te parece? Vamos a ponerle un mote. Vamos a llamarlo... Pedrín. No espera, tengo una idea mejor.

Ejemplo 03: Un nombre de variable apropiado

```php
<?php

$pi = 3.14159265359;
```

Ahora hemos creado una nueva variable llamada `$pi`, que contiene el valor `3.14159265359`. Esto significa que podemos usar la variable en cualquier lugar de nuestro código para hacer cálculos. He aquí algunos ejemplos.

Ejemplo 04: Usando variables en sentencias

```php
<?php

// Asignando pi a una variable.
$pi = 3.14159265359;

// Realizar cálculos de circunferencias.
echo $pi * 5;
echo $pi * 3;
```

Tras asignar `$pi` podemos usarla en otras sentencias para realizar cálculos.

Podemos declarar y asignar tantas variables como quieras, pero hay varias reglas que tenemos que seguir al escoger nombres. Los nombres de las variables pueden contener números, letras y guiones bajos. No obstante, ***tienen que*** comenzar por una letra o un guión bajo, ¡nunca un número!. Son sensibles a mayúsculas y minúsculas lo cuál significa que `$panda` es diferente a `$pAnda`. He aquí un par de ejemplos.

Ejemplo 05: Nombrando variables

```
1  <?php
2
3  $panda = 1;      // Legal
4  $Panda = 1;      // Legal
5  $_panda = 1;     // Legal
6  $pan_da = 1;     // Legal
7  $pan_d4 = 1;     // Legal
8  $pan-da = 1;     // Ilegal
9  $4panda = 1;     // Ilegal
```

Aunque los nombres de las variables pueden contener guiones bajos y comenzar con letras mayúsculas, es una práctica común usar el formato llamado `camelCasing`. No te preocupes, no hacen falta camellos.

Los nombres CamelCase, comienzan con un carácter en minúscula. Las variables que tienen varias palabras, tendrán las primeras letras de estas en mayúsculas. Por si no queda claro, he aquí algunos ejemplos.

Ejemplo 06: Variables en CamelCase

```
1  <?php
2
3  $earthWormJim
4  $powerRangers
5  $bobEsponja
```

¿Recuerdas cómo nuestras sentencias devuelven un valor? Una noticia, nuestras asignaciones son also sentencias. ¿Puedes adivinar lo que significa? Eso es, también devuelven un valor. Podemos probar esto usando nuestro viejo amigo `echo`.

Ejemplo 07: Las sentencias devuelven un valor

```
1  <?php
2
3  echo $panda = 1337;
```

Recibimos el número `1337` como la salida. Esto es porque la asignación de la variable `$panda` se lleva a cabo antes de que sea mostrada. Este proceso nos permite usar un astuto truco. No es algo que vayas a usar muy a menudo, pero creo que es un buen truco a saber. Ve y echa un vistazo a este ejemplo.

Ejemplo 08: Asignación múltiple

```
1  <?php
2
3  $primerPanda = $segundoPanda = $tercerPanda = 1337;
```

El fragmento de arriba puede parecer un poco alocado, pero tiene más sentido si lo lees de derecha a izquierda. El `$tercerPanda` es asignado al valor `1337`, luego el `$segundoPanda` es asignado al valor del `$tercerPanda`, y finalmente el `$primerPanda` es asignado al valor del `$segundoPanda`. Esto significa que todas las variables están asignadas al valor final. Bonito, ¿no te parece?

De mi tipo

Hasta ahora hemos estado trabajando con números. Sería aburrido si solo pudiéramos trabajar con valores de ese tipo. Creo que es el momento de examinar las otras posibilidades. He aquí algunos de los valores comunes usados en aplicaciones PHP.

- entero (*integer*)
- coma flotante (*float*)
- booleano (*boolean*)
- cadena (*string*)
- null
- matriz (*array*)

Hay unos pocos más pero no vamos a complicarnos por ahora. Tenemos que aprender poco a poco. ¡No quieres una sobrecarga de conocimiento!

Echemos un vistazo a estos tipos uno a uno. Primero tenemos los enteros. Son números entero, los hemos estado usando en nuestros ejemplos anteriores.

Ejemplo 09: Enteros

```
<?php

$panda = 2;
$pandaRojo = -23;
```

Los números de coma flotante son los que tienen decimales, y por ende contienen fracciones. Pueden ser usados de manera similar a los enteros. De hecho, ya hemos usado uno. ¿Recuerdas a nuestro amigo `$pi`? Era uno de ellos. Sigamos, ¿no te parece?

Ejemplo 10: Coma flotante

```
<?php

$panda = 2.34;
$pandaRojo = -23.43;
```

Los booleanos, son tipos de datos binarios. ¡No te preocupes! No vamos a hacer nada de aritmética binaria. Es solo una forma de expresar que pueden contener dos valores. Un booleano puede ser o `true` (*verdadero*) o `false` (*falso*). Más tarde, echaremos un ojo a cómo estos valores pueden ser usados para alterar el flujo de nuestra aplicación.

Ejemplo 11: Booleanos

```
<?php

$panda = false;
$pandaRojo = true;
```

Lo siguiente son los valores de 'cadena'. Las cadenas son usadas para almacenar una palabra, un caracter, o una secuencia de texto. Las cadenas son especiales por lo que he decidido dedicarles un capítulo corto. ¡Volveremos a ellas!

Ejemplo 12: Cadenas

```php
<?php

$panda = 'Panda normal';
$pandaRojo = "Panda rojo";
```

Null es un valor especial. Es nada. Cero. Vacío. Bueno, no es cero. El cero es numérico y podemos usar un entero para eso. Los valores nulls son exactamente nada. Null es el valor que tiene una variable antes de que se haya asignado. Es un valor muy útil y lo verás mucho en el futuro.

Ejemplo 13: Valores null

```php
<?php

$noPanda = null;
```

Las matrices o vectores son otro tipo de valor especial. De hecho, es mi favorito. Tanto es así que tienen un capítulo completo dedicados. Por ahora, todo lo que necesitas saber es que es un valor que contiene una colección de otros valores. ¡Vaya! ¡Inception!

Ejemplo 14: Matrices

```php
<?php

$cuentaLosPandas = [1, 2, 3];
$masPandas = array(5, 6, 7, 8);
```

Asignación avanzada

En un capítulo anterior descubrimos los operadores que podemos usar en otras variables, y aprendimos sobre el operador de asignación. Así que, ¿qué pasa si los ponemos juntos? Bueno, crearemos un agujero negro y consumiremos el universo al completo. Me siento aventurero, ¿lo descubrimos?

Ejemplo 15: Asignación con suma

```php
<?php

// Establece un valor
$panda = 3;

// Intenta crear un agujero negro
$panda += 1;

// Mostremos el valor del universo
var_dump($panda);
```

Primero establecemos una variable al valor entero tres. Luego, hemos añadido el operador de suma frente al operador de asignación y le hemos facilitado el valor entero de uno.

Podemos usar la función `var_dump()` (¡pronto veremos más sobre las funciones!) para saber no solo el valor de una variable, si no también su tipo.

¿Qué obtenemos de salida?

Ejemplo 16: Salida

```
int(4)
```

¡Genial! El universo ha sido salvado. ¿Parece que tenemos un cuatro? Supongo que tiene sentido. Sabemos que `$a + $b` devuelve un valor sin asignarlo, y sabemos que el operador de asignación es usado para establecer el valor de las variables. Así que esto hace ambas cosas. Le estamos diciendo a PHP que establezca el valor de `$panda` a su valor actual, más uno.

Puedes usar esta sintaxis con cualquiera de los operadores que ya hemos descubierto. Solo hay una pega. No coloques el operador al otro lado del signo igual. Confía en mi, lo he intentado. Se abrió un portal a un mundo oscuro, criaturas mitad dinosaurio, mitad humana aparecieron y empezaron a sembrar el caos en Cardiff. Solo con la ayuda de un lanzallamas casero (que construí usando PHP) pude derrotar a las desagradables criaturas. Odiaría ver que te ocurre a ti. ¡Ten cuidado por favor!

Luego tenemos el operador incremental. De hecho, no nos olvidemos del operador decreciente. Suele tener menos atención. Ya que estamos, vamos a mostrar sus habilidades.

Me gusta poner un ejemplo. Pongamos el siguiente:

Ejemplo 17: – Después

```php
<?php

// Establece un valor
$panda = 3;

// Reducimos el valor
$panda--;

// Mostramos el valor
var_dump($panda);
```

Ahí, en el medio, ¿lo ves? El precioso operador decreciente. Colocamos dos signos de menos tras la variable. Así que, ¿qué hace esto? He aquí el resultado.

Ejemplo 18: Salida.

```
int(2)
```

Como podemos ver, el valor de `$panda` ha sido reducido en uno. Es un atajo rápido para reducir un valor. De manera análoga, podemos usar `++` para aumentar un valor. No obstante, estos son los dos únicos operadores que funcionan. No intentes usar el de multiplicación. ¡No funcionará como tú esperas!

Me pregunto qué ocurrirá si colocamos el operador antes del valor. Vamos a intentarlo, ¿vale?

Ejemplo 19: – Antes.

```php
<?php

// Establece un valor
$panda = 3;

// Reducimos el valor
--$panda;

// Mostramos el valor
var_dump($panda);
```

¿Cuál es la respuesta? ¿¡No estás nervioso!?

Ejemplo 20: Salida.

```
int(2)
```

Oh, es la misma. Bueno, eso ha sido aburrido, ¿no te parece? Lo cierto es que conozco un pequeño secreto. No es lo mismo. El valor parece idéntico pero el ejemplo que he puesto no le hace justicia.

Vamos a elaborar un ejemplo diferente. Mostraremos el estado del valor **antes** de usar el operador. Examinaremos el resultado **cuando** la operación se haya usado, y finalmente, examinaremos el valor **después** de que el operador se haya usado. No esperamos que el valor de después sea diferente.

Ejemplo 21: Las fases de–.

```
<?php

// Establece un valor
$panda = 3;

// Mostramos ANTES
var_dump($panda);

// Mostramos DURANTE.
var_dump(--$panda);

// Mostramos DESPUES
var_dump($panda);
```

Ejecutemos el código. ¿Cuales son los tres valores que recibimos?

Ejemplo 22: Salida.

```
int(3)
int(2)
int(2)
```

Así pues, el primer valor es tres. Deberíamos habernos esperado eso, después de todo, lo único que hemos hecho ha sido establecerlo, ¿verdad? El resultado de la sentencia usando el operador es igual a dos. El valor resultante también es dos. Eso significa que el valor se reduce en la segunda línea.

Vamos a mover el operador al otro lado del valor. Así:

Ejemplo 23: Las fases de – parte dos

```php
<?php

// Establece un valor.
$panda = 3;

// Mostramos ANTES
var_dump($panda);

// Mostramos DURANTE
var_dump($panda--);

// Mostramos DESPUES
var_dump($panda);
```

Ahora echemos un vistazo al resultado. Mira con cuidado para ver la diferencia.

Ejemplo 24: Salida.

```
int(3)
int(3)
int(2)
```

¿¡Ey!? ¡El valor de en medio es diferente! ¿Por qué no ha decrecido? Bueno, al cambiar el operador, le hemos dicho a PHP que reduzca el valor **DESPUÉS** de la línea actual. El resultado de la línea de la operación es el mismo que era inicialmente.

Déjame resumir.

$valor– - Cambia el valor *después* de la línea. –**$valor** - Cambia el valor en la línea actual.

Así pues, ¿para qué nos sirve esto? He aquí un ejemplo de para qué nos sirve. Estoy seguro de que si eres creativo descubrirás más. Usando el operador que cambia **después** de la línea, podemos establecer otra variable a su valor, y reducir el valor original en la misma línea. Por ejemplo:

Ejemplo 25: Asignar e incrementar.

```php
<?php

// Establece un valor
$panda = 3;

// Asigna, y luego incrementa
$pandaFriend = $panda++;
```

Lo que hemos hecho aquí es ahorrarnos una línea. Es una especie de atajo. He aquí cómo se vería si no usáramos el operador.

Ejemplo 26: Incrementar explicado.

```php
<?php

// Establece.
$panda = 3;

// Asigna.
$pandaFriend = $panda;

// Incrementa.
$panda = $panda + 1;
```

Más adelante, hablaremos de los bucles y descubrirás otro uso para este operador. En el próximo capítulo hablaremos de las cadenas.