

# Curso Básico de PHP

Carlos Gómez  
@RetaxMaster



1  
0  
1  
0  
1

1  
0  
1  
0  
1



# ¡Te doy la bienvenida!

Soy Carlos (RetaxMaster) y hago parte del equipo de Associates de Platzi (el **mejor equipo** 😊). Tal vez me recuerdes de aportes en otros cursos como este:

RetaxMaster  
Platzi Team · Hace 9 meses

Ohh si PHPCito!!

138

En PHP se pueden hacer muuuuchas cosas con los arrays, pero en serio, **MUCHAS**, en PHP puedes declarar arrays usando corchetes `[]` o usando la función `array()` que es mucho más elegante.

En PHP puedes declarar los arreglos en múltiples líneas, y además puedes usar una cosa llamada "trailing commas", que es básicamente dejar una coma al final del último valor del array, por ejemplo:

```
$weather = array(  
    "Bogota" => "cold",  
    "Monteria" => "hot",  
    "Medellin" => "mild", // <- Este es el trailing  
    comma  
>);
```

Para ti ¿qué  
es programar?





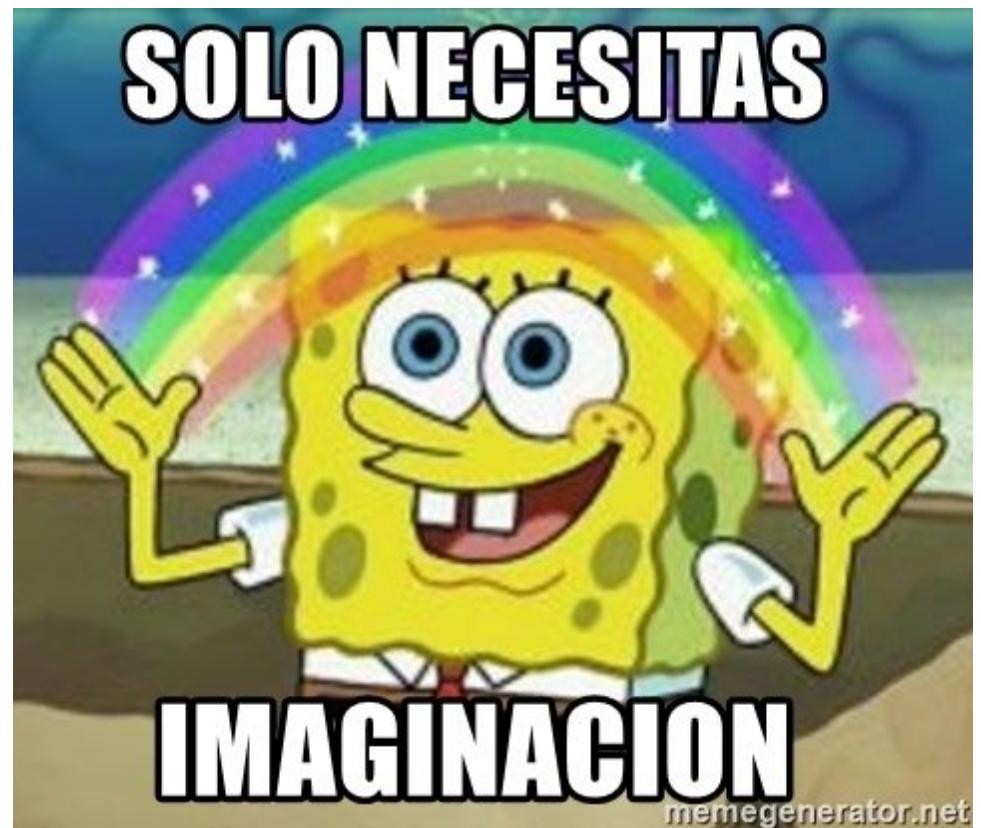
# ¿Qué es programar?

En palabras sencillas, es decirle a una máquina increíblemente obediente qué es lo que debe hacer.



# ¿Qué es programar?

En programación  
puedes hacer TODO  
lo que te puedas  
imaginar.





# Algoritmos

Podemos visualizarlos como una lista de pasos ordenados que cualquier persona que los lea puede seguir y deben generar el mismo resultado siempre.



# Algoritmos

Por ejemplo, ¿cómo haces un taco?

1. Pones la carne.
2. Pones la tortilla.
3. Pones el plato.



# Algoritmos

**X** ¡No es correcto!

1. Pones la carne.
2. Pones la tortilla.
3. Pones el plato.



# Algoritmos

 Debemos ordenar este algoritmo para producir el resultado que deseamos obtener.

1. Pones el plato.
2. Pones la tortilla.
3. Pones la carne.

# ¿Qué es un programa?





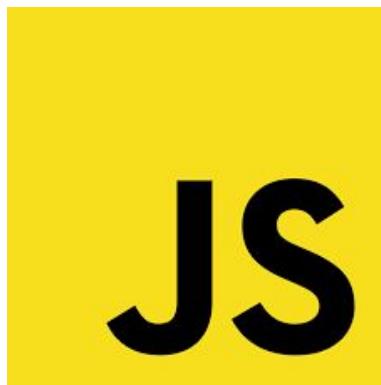
# ¿Qué es un programa?

Un algoritmo escrito con algún lenguaje de programación.

Al estar escritos con algún lenguaje las computadoras son capaces de seguir estos pasos.



# ¿Qué lenguaje de programación elijo?





# ¿Qué lenguaje de programación elijo?





# ¿Por qué PHP?

- Muy fácil de aprender.
- Muy flexible.
- Tiene una amplia documentación en español.
- Es muy demandado.
- Tiene a Laravel y WordPress.
- Una gran parte de los sitios web lo ocupan.



# ¿Quieres aprender PHP?



1  
0

1 0 1

1 0 1

¿PHP está  
muerto? :

1  
0





# ¿Qué es PHP?

PHP es un lenguaje de scripting de uso general que es especialmente adecuado para el desarrollo web.

- Del lado del servidor.
- Hypertext Preprocessor.



# ¿Qué es PHP?





# ¿Qué es PHP?

## De propósito general

- Código en el servidor.
- Comandos para la terminal.
- Aplicaciones de escritorio.

## De alto nivel.

Es un lenguaje interpretado.

Es de tipado débil.

*“PHP puede usarse para cualquier cosa, pero solo porque puedes escribir algo con PHP no significa que debas hacerlo”.*

*Rasmus Lerdorf*

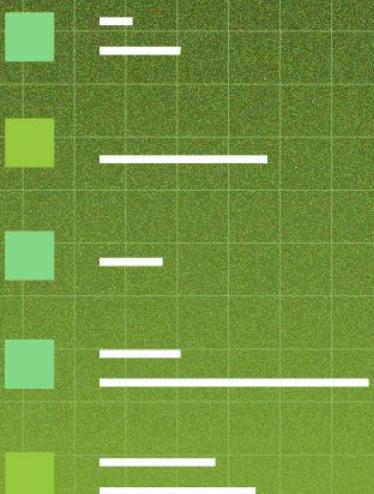
Pero... ¿PHP no  
estaba muerto?

- =
- \_
- -
- ==
- ==

La mayoría de las  
personas que dicen  
esto no han usado

**PHP**

(O lo han usado muy poco)



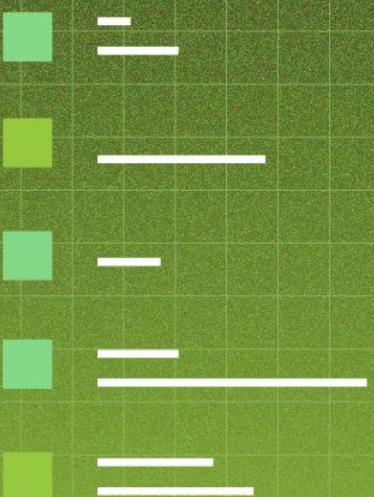


# PHP es la mayoría del Internet



- Según las estadísticas de W3Techs, PHP está en el 79% de todos los sitios web del mundo.
- Tan solo el 53.4% usan su versión 7.
- El 30% de los sitios de todo internet están construidos con WordPress (el CMS de PHP).

# ¿Quiénes usan PHP?



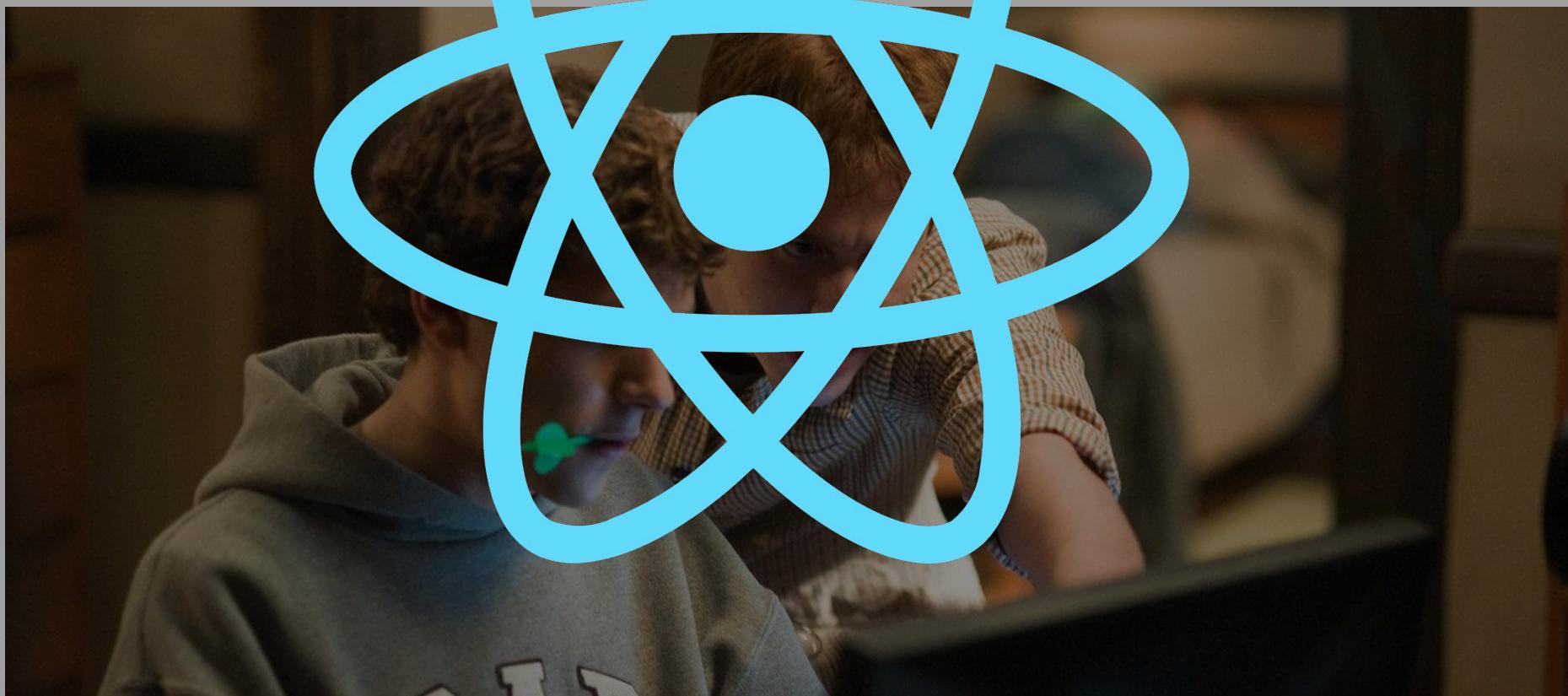


# Facebook





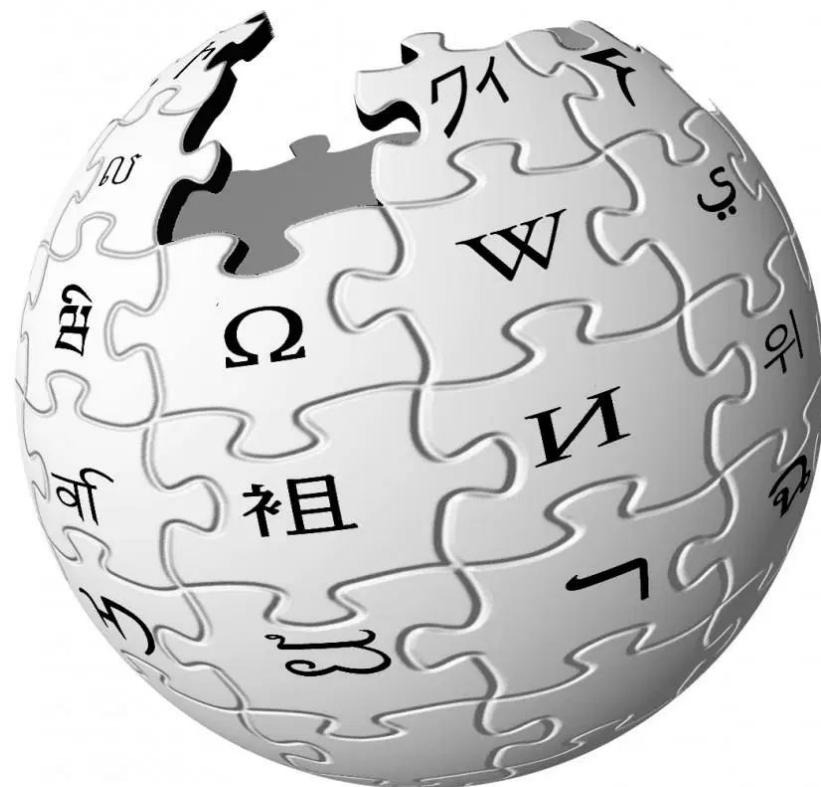
# Facebook





# Wikipedia

WIKIPEDIA  
*The Free Encyclopedia*





**Platzi**



**Platzi**



# Slack





# PHP está más vivo que nunca

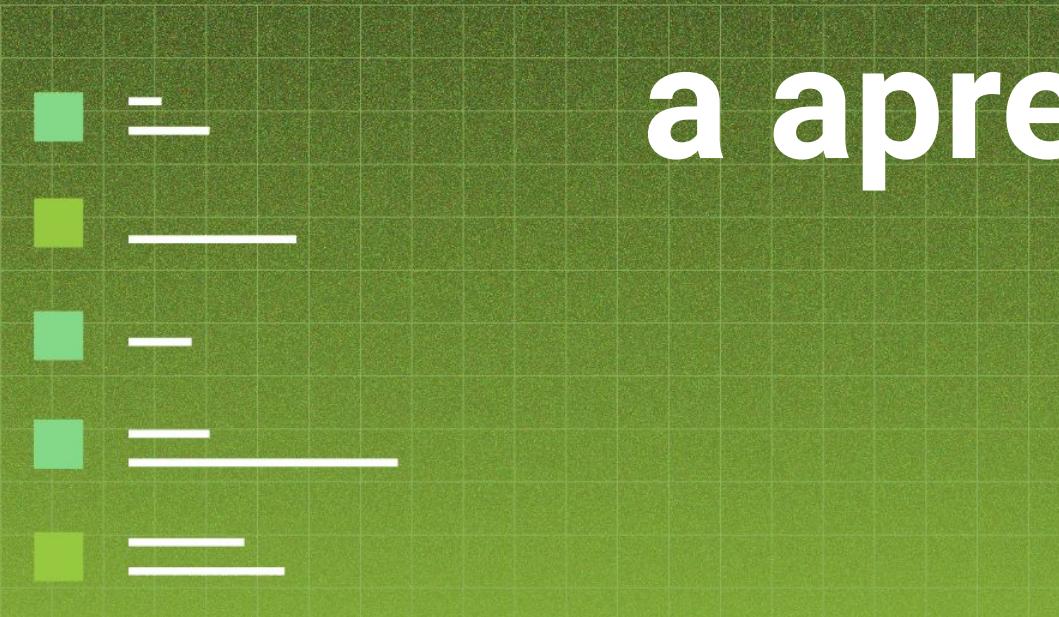
Con cada versión nueva el lenguaje ha ido mejorando. Se han agregado nuevas funcionalidades que permiten programar más cómodamente y que hacen a PHP un lenguaje más amigable.

*“PHP es una herramienta,  
no una religión”.*

*Rasmus Lerdorf*

*“People use PHP to  
solve their problems”.*

*Rasmus Lerdorf*



¿Me acompañas  
a aprender PHP?



# ¿Cómo interactúa una página web con tu backend?

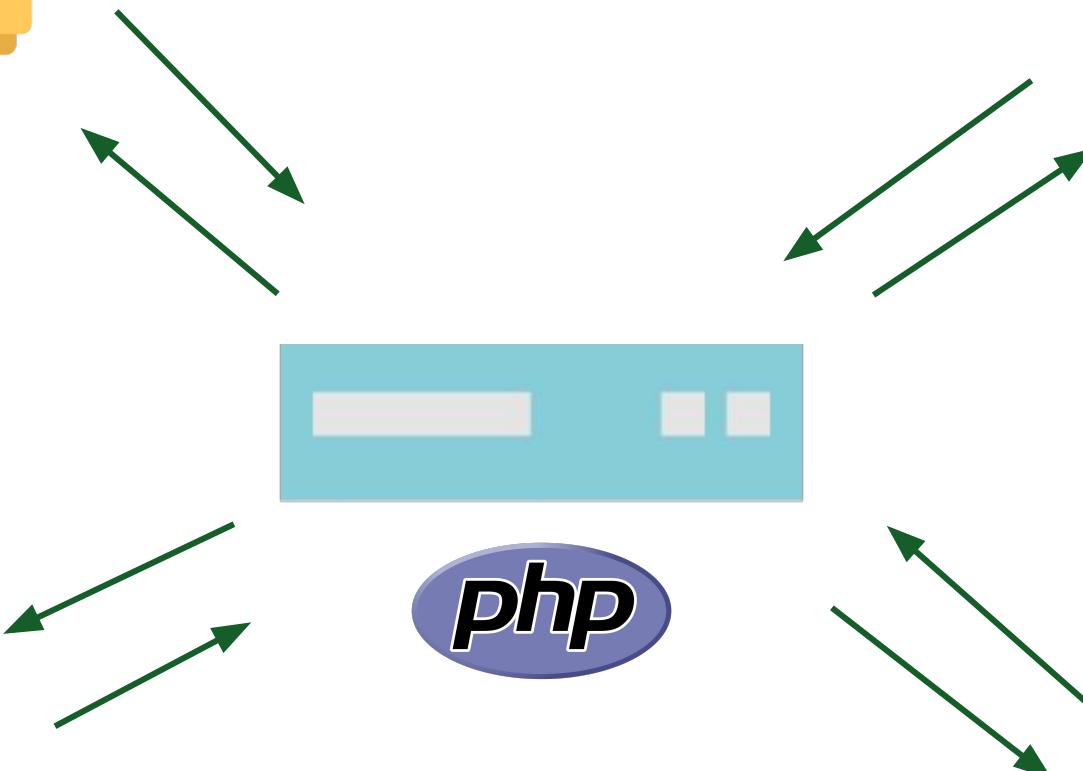
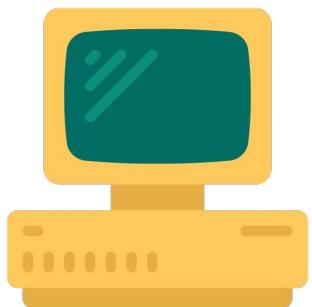
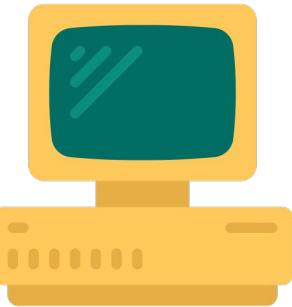




# Cliente y servidor

Toda nuestra aplicación está guardada en un servidor, el cual entrega una copia de la misma a cada cliente que la solicite.

Además, el servidor también se encarga de responder cada solicitud del usuario.





# Dominio

El dominio es nuestra dirección en internet. Gracias a él cualquier computadora es capaz de encontrar nuestra página web.

<http://www.platzi.com> 



# Servidor físico o VPS

Es la computadora que se encarga de guardar tu página web y mantenerla accesible 24/7. Se le conoce como servidor y siempre está conectado a internet.

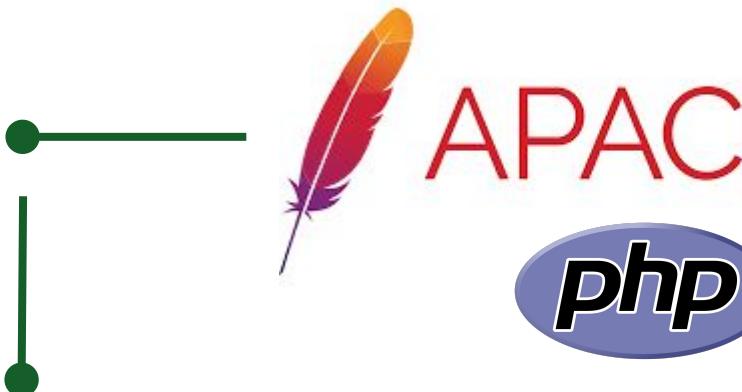
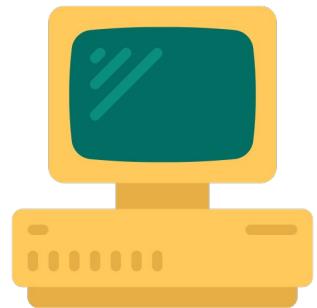
A través de él podemos definir ciertas reglas de seguridad para nuestra página.



# Servidor web

Es un programa que corre dentro de nuestro servidor físico y se encarga de gestionar cualquier petición que llegue al mismo.

Esta petición es procesada por algún lenguaje de programación y al final devuelve una respuesta.



The screenshot shows the Platzi platform dashboard. At the top, it displays the user's name, Carlos Eduardo, and the date range from October 18 to 24. Below this, there are three main statistics: 32 minutes of study, 0 approved courses (out of 130 total), and 117 Platzi Rank points (out of 112,666 total). A section titled "Continúa aprendiendo" (Keep learning) shows course recommendations, including "Desafío de creación" by Carlos Eduardo and "Curso de Fundamentos de la Co...".



# Métodos HTTP

Los métodos HTTP son una forma de comunicación entre el cliente y el navegador. A través de una solicitud HTTP el cliente es capaz de pedirle al servidor que realice una acción.

Hay varios, pero tenemos 5 métodos principales.



Este método permite solicitar información al servidor. Por ejemplo, podemos pedirle una lista de productos en el caso de que estemos haciendo un e-commerce o una lista de cursos si tenemos una página como Platzi.



- Este método permite guardar información. Por ejemplo, podemos recabar datos del usuario desde un formulario y mandarlos a nuestro servidor para procesarlos.
- Podríamos guardarlos para armar una base de datos de usuarios o incluso un sistema de login.



# PUT/PATCH

- Estos métodos permiten actualizar información ya guardada. Por ejemplo, podemos darle la oportunidad a un usuario de actualizar su correo electrónico o incluso cambiar su contraseña.
- La diferencia es que PUT reemplaza toda la información existente y PATCH solo reemplaza lo necesario, es decir, “parcha” la información.



# DELETE

- Este método lo usamos para eliminar un recurso del servidor. Por ejemplo, podemos usarlo si deseamos eliminar un blogpost o un comentario.
- Esto no significa que debamos eliminarlo necesariamente dentro de nuestra base de datos, podemos hacer un “soft delete”.

1  
0

1 0 1

1 0 1

# Instalación en Windows

1  
0



1

1 0 1

**sudo add-apt-repository ppa:ondrej/php**

**sudo apt update**

**1 sudo apt upgrade**

0 1

**sudo apt install php8.0 apache2**

**sudo a2dismod php8.0**

**sudo a2enmod php7.4**

**sudo nano /etc/php/7.3/apache2/php.ini**

**display\_errors = Off**

**por**

**display\_errors = On**

**dpkg --get-selections | grep php**

**sudo apt-get install libapache2-mod-php7.3**

**sudo service apache2 restart**

1

0



1  
0

1 0 1

1 0 1

# Instalación en Linux

1  
0



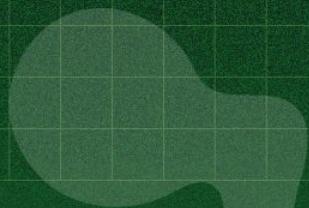
1  
0

1 0 1

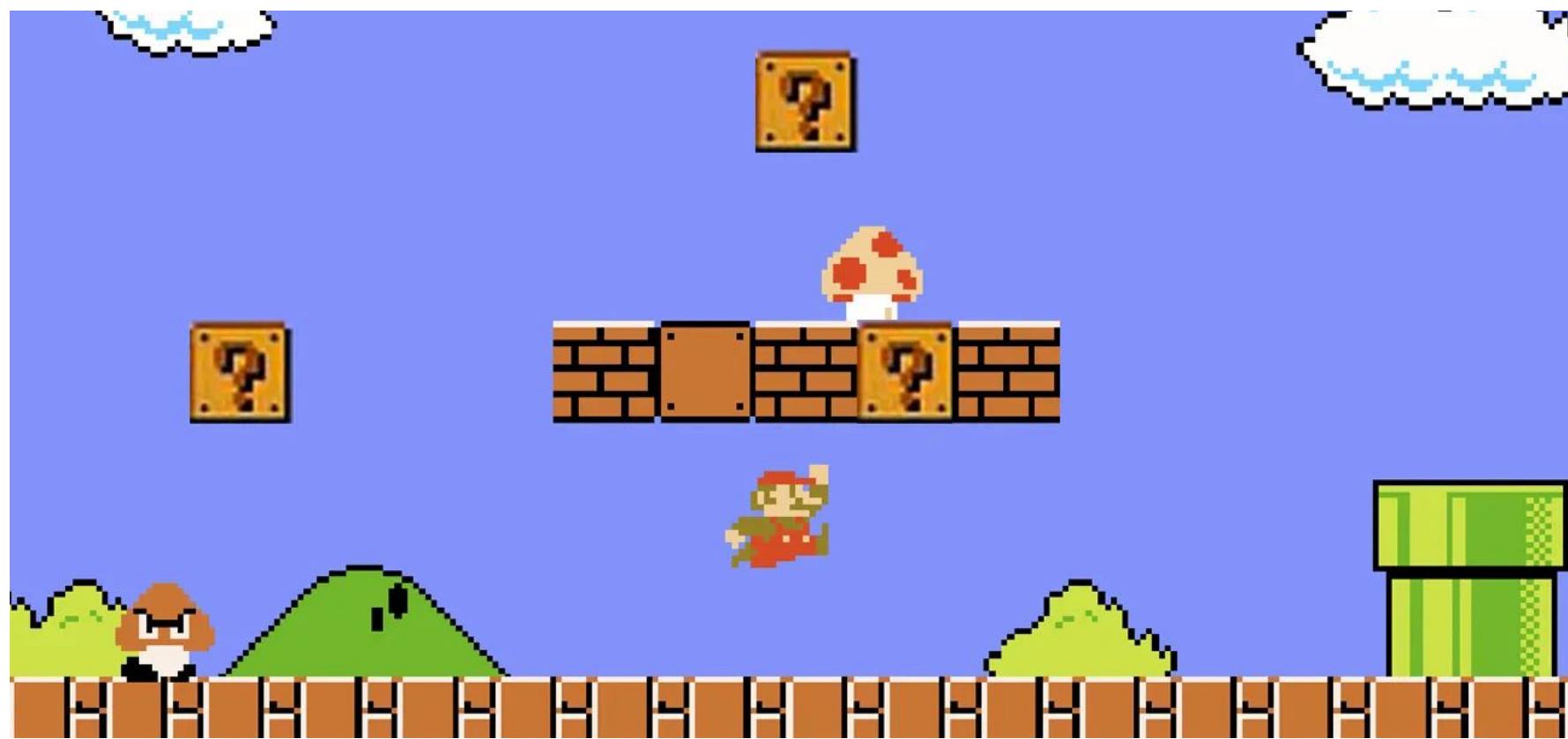
1 0 1

# Hackea tu miedo a la terminal

1  
0





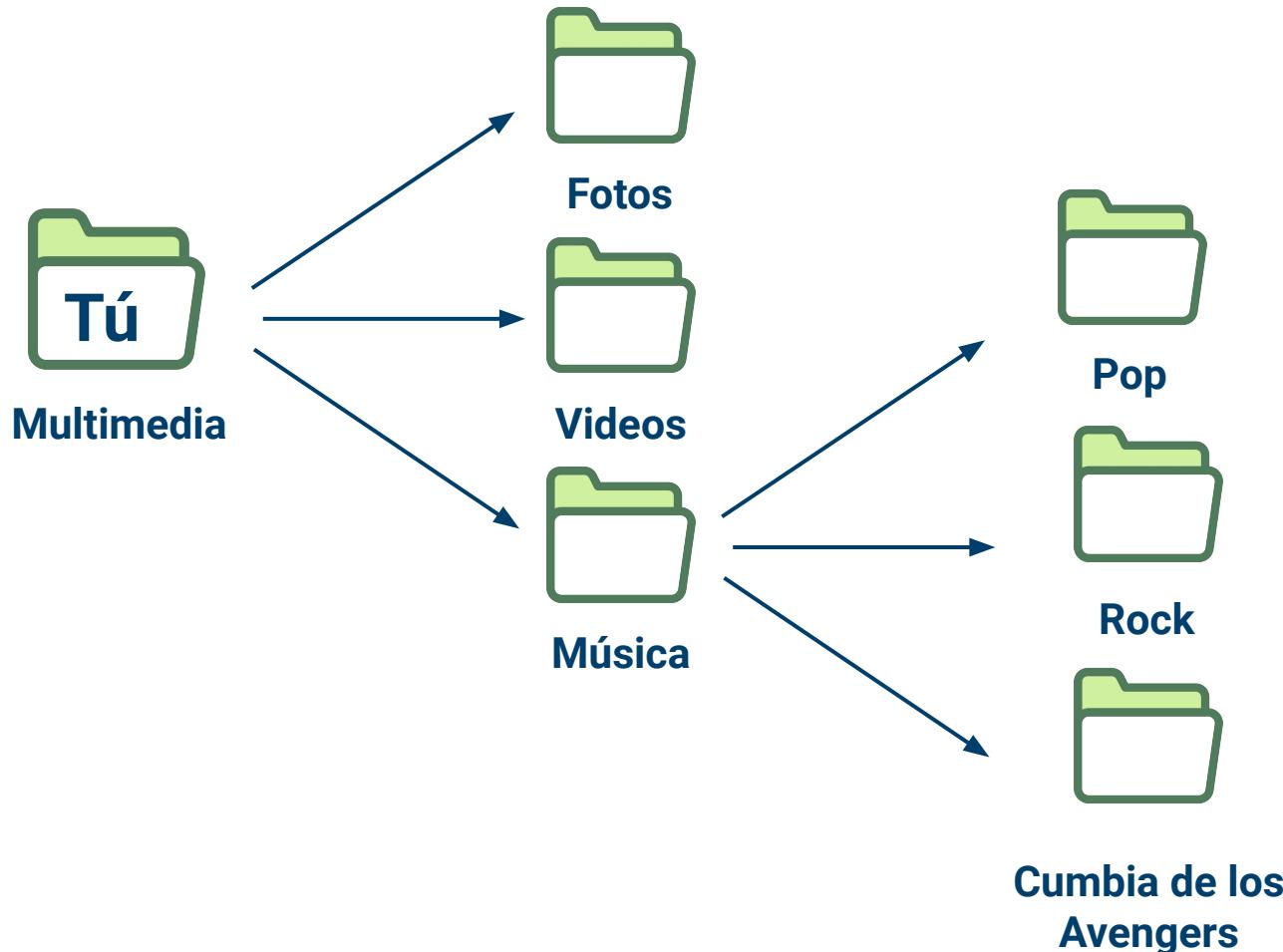




# Comando CD

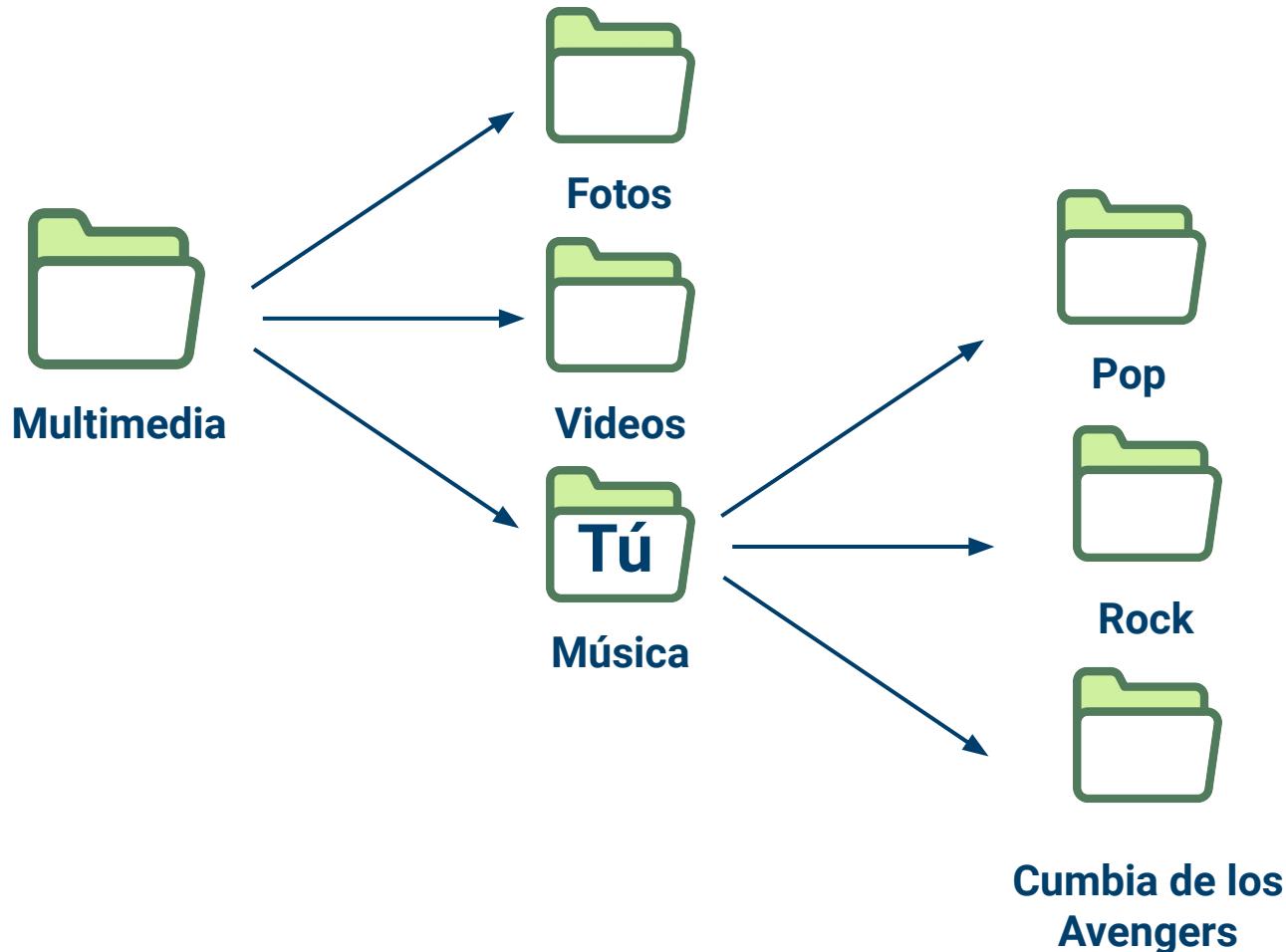
Tus opciones:

- cd Fotos
- cd Videos
- cd Música





# Comando CD

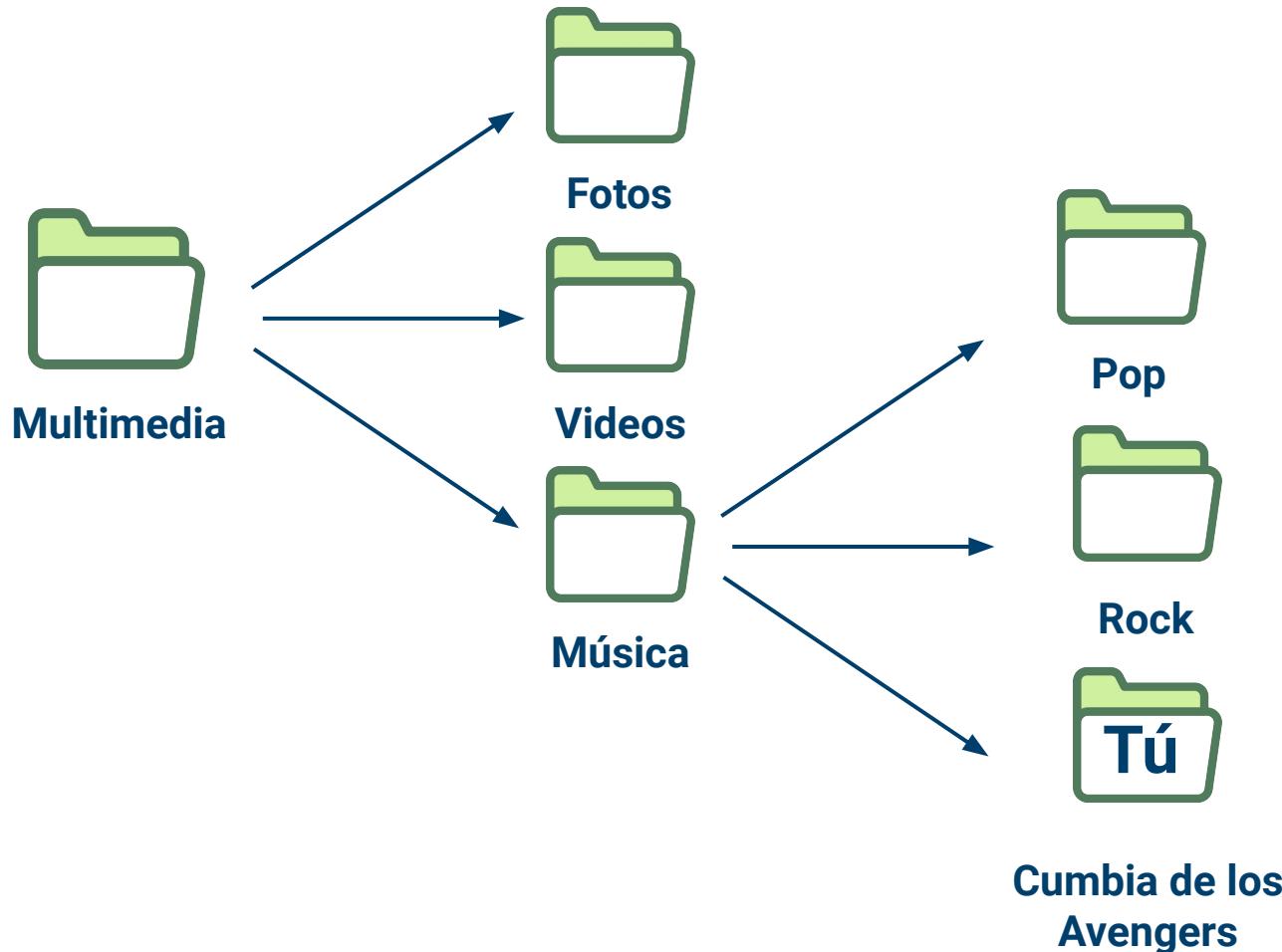


Tus opciones:

- cd Pop
- cd Rock
- cd "Cumbia de los Avengers"



# Comando CD





# Comandos en la terminal

Y así como este hay muchos más comandos útiles:

- **ls**: Para listar nuestras opciones.
- **mkdir**: Para crear una carpeta.
- **touch**: Para crear un archivo.
- **cat**: Para ver el contenido de un archivo.

*“La terminal está ahí para facilitarnos la vida, no para complicárnosla”.*

*RetaxMaster*

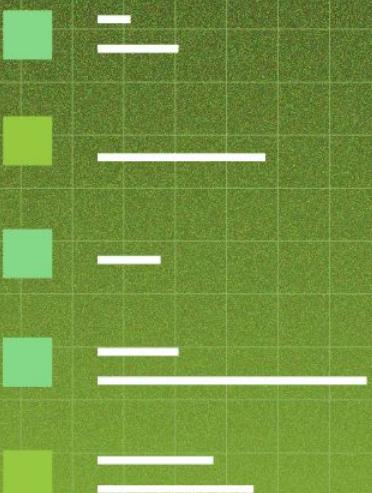


# ¡Vamos a la terminal!



# Cómo usar la terminal con PHP

(Y un pequeño spoiler de la siguiente clase 😊)



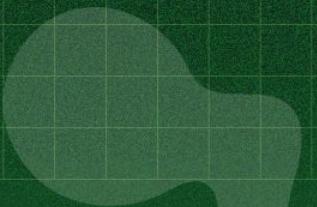
1  
0

1 0 1

1 0 1

# Cómo ejecutar tus archivos PHP

1  
0



1  
0

1 0 1

1 0 1

# Sintaxis básica de PHP

1  
0





# ¿Qué es una sintaxis?

Según la definición de Wikipedia:

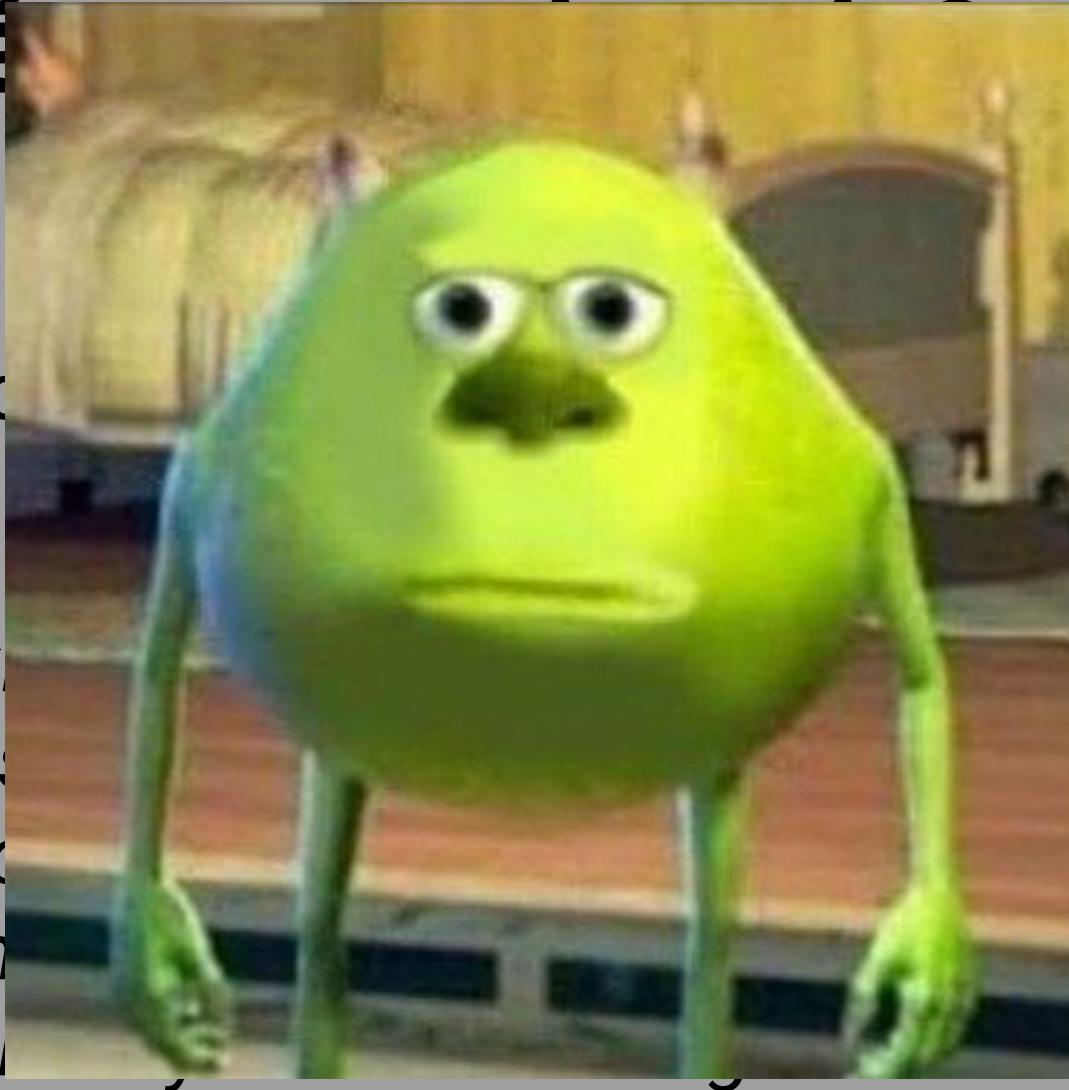
*“La sintaxis es la parte de la gramática que estudia las reglas y principios que gobiernan la combinatoria de constituyentes sintácticos y la formación de unidades superiores a estos, como los sintagmas y las oraciones gramaticales.”*



# ¿Qué es la sintaxis?

Según la definición de

*“La sintaxis es la rama de la lingüística que estudia las combinaciones posibles de palabras y la formación de los sintagmas y de las frases.”*



*“Las reglas de la sintaxis regulan la combinación de los y las palabras, como se ha visto.”*



# En palabras más sencillas...

Son las reglas que dictan cómo se debe estructurar un lenguaje para que tenga sentido lo que decimos.

X **Yo querer taco carne mucha.**

✓ **Yo quiero un taco con mucha carne.**



# Y lo mismo aplica para los lenguajes de programación

X “Mi primer programa echo >

✓ echo “Mi primer programa”;



# ¡Vamos al código!



1  
0

1 0 1

1 0 1

# Comentarios y debugging

1  
0



1  
0

1 0 1

1 0 1

# Variables y constantes

1  
0



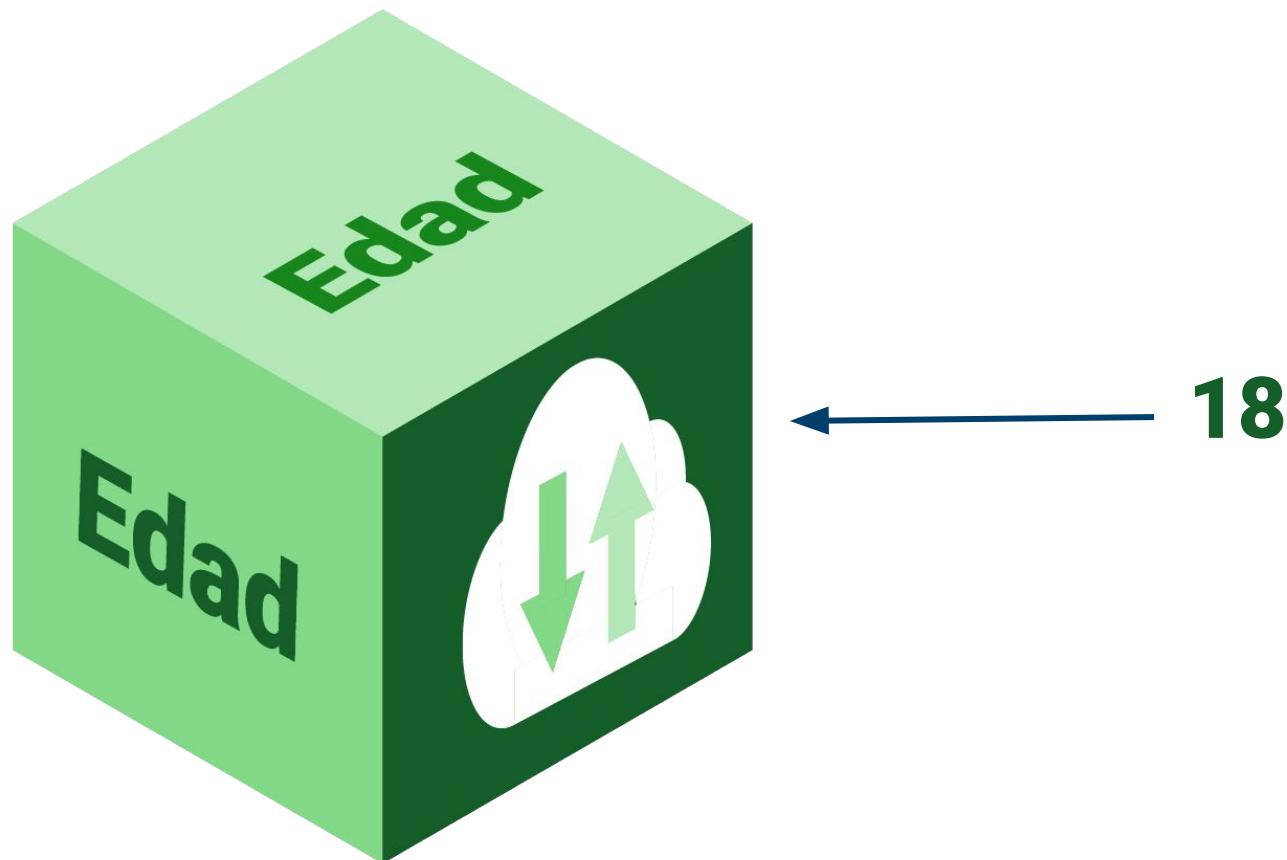


# ¿Qué es una variable?

Es algo **que varía**, es decir, es capaz de guardar cualquier cosa en su interior y puede modificar, sustituir o eliminar dicho elemento que esté guardando en cualquier momento.



# Podemos verlas como cajas





# ¿Qué es una constante?

Es algo que **nunca cambia**, es decir, una vez que se le asigne un valor ya no es posible modificarlo ni sustituirlo.

Una constante siempre mantendrá su valor intacto desde el momento en que se declara.



# Estructura de una variable

```
$nombre = "Carlitos";
```



**Signo  
de dólar**



# Estructura de una variable

```
$nombre = "Carlitos";
```

---

**Nombre de la  
variable**



# Estructura de una variable

```
$nombre = "Carlitos";
```

---

Asignación



# ¡Vamos al código!



1  
0

1 0 1

1 0 1

# Tipos de datos

1  
0





# Tipos de datos

Un tipo de dato simplemente es una forma de diferenciar los datos que tenemos por medio de su tipo.

**Por ejemplo, si es un número, una letra, una palabra, etc.**



# Tipos de datos

En programación podemos tener varios tipos de datos, pero los más conocidos son...

- **Numéricos**
  - Integer
  - Float
  - Double
- **Cadenas de caracteres**
  - Char
  - String
- **Booleanos**
  - Bool
- **Sin valor**
  - Null
  - Undefined



# Tipos de datos en PHP

PHP tiene tipado débil, pero eso no significa que no use tipos de datos.

Este lenguaje no necesita que definas de forma explícita un tipo de dato, ya que por sí mismo puede deducir qué tipo de dato estás usando.



# Conversión de tipos automática

PHP es capaz de convertir un tipo de dato a otro de forma automática según el contexto donde se use.

Si sumas un número con un string, PHP evaluará ambos como números.



# ¡Vamos al código!



1  
0

1  
0  
1

1  
0  
1

# Casting

1  
0





# ¿Qué es el *casting*?

Casting es cuando forzamos que un tipo de dato se convierta en otro tipo de dato.



# ¿Qué es el *casting*?

Casting es cuando forzamos que un tipo de dato se convierta en otro tipo de dato.



```
$numerito_string = "5"; // Esto es un string  
$numerito_int = (int) $numerito_string; // Esto es un integer
```



# ¡Vamos al código!





# Reto

¿Qué tipo de dato son las siguientes variables?

**¡Déjalo en los aportes de la clase!**

```
// Fácil
$nombre = "Carlos";
$apellido = "Gómez";
$edad = 18;
$aprobado = true;

// Medio
$promedio = (8 + 9.5 + 9 + 10 + 8) / 5;
$nombre_completo = $nombre . " " . $apellido;
$presento_examen = (bool) 1;

// Avanzado
$numero_preguntas = 5 + "5";
$numero_respuestas = "5" + 5;
$promedio_maximo = $numero_respuestas / 1.0;
$ michis = 3 + "5 michis";
```



# Pero antes... ¿Cómo te fue? 🕒

```
// Fácil
$nombre = "Carlos"; // String
$apellido = "Gómez"; // String
$edad = 18; // Integer
$aprobado = true; // Boolean

// Medio
$promedio = (8 + 9.5 + 9 + 10 + 8) / 5; // Decimal
$nombre_completo = $nombre . " " . $apellido; // String
$presento_examen = (bool) 1; // Boolean

// Avanzado
$numero_preguntas = 5 + "5"; // Integer
$numero_respuestas = "5" + 5; // Integer
$promedio_maximo = $numero_respuestas / 1.0; // Decimal
$ michis = 3 + "5 michis"; // Integer (con warning)
```

1  
0

1 0 1

1 0 1

# Operadores lógicos

1  
0





# Operadores lógicos

Son los operadores que nos ayudan a combinar dos o más afirmaciones para definir si una oración es cierta o falsa.

Su uso está basado en las tablas de verdad.



# Operadores lógicos

En la mayoría de sus usos se necesitan al menos dos afirmaciones y un operador:

Operador

“Los michis son felinos **y** tienen 4 patas”

---

Afirmación 1

---

Afirmación 2



# Y (And)

Se usa para verificar si dos afirmaciones son ciertas. Si ambas afirmaciones son ciertas, entonces la oración completa es cierta. Si una de ellas es falsa, entonces la oración completa es falsa.



# Y (And)

En PHP podemos usarlo de esta forma:

- **\$valor\_1 and  
\$valor\_2**
- **\$valor\_1 &&  
\$valor\_2**

Valor 1	Operador	Valor 2	Resultado
Verdadero	AND	Verdadero	Verdadero
Verdadero	AND	Falso	Falso
Falso	AND	Verdadero	Falso
Falso	AND	Falso	Falso



# Ejemplos con “And”

- ¿Los miembros de la familia tienen patas?  
Verdadero
- ¿Los miembros de la familia tienen alas?  
Falso
- ¿Los miembros de la familia programan en PHP?  
Falso





# O (Or)

Se usa para verificar si al menos una afirmación es cierta. Si al menos una de ellas es cierta, entonces la oración es cierta, de lo contrario, es falsa.



# O (Or)

En PHP podemos usarlo de esta forma:

- **\$valor\_1 or  
\$valor\_2**
- **\$valor\_1 || \$valor\_2**

Valor 1	Operador	Valor 2	Resultado
Verdadero	OR	Verdadero	Verdadero
Verdadero	OR	Falso	Verdadero
Falso	OR	Verdadero	Verdadero
Falso	OR	Falso	Falso



# Ejemplos con “Or”

- ¿Los michis son felinos **OR** tienen 4 patas?  
**Verdadero**
- ¿Los michis son aves **OR** tienen 4 patas?  
**Verdadero**
- ¿Los michis vuelan **OR** programan con PHP?  
**Falso**



# No (Not)

Se usa para invertir el valor de una afirmación.

**En PHP podemos usarlo de esta forma:**

- `!$valor`

Operador	Valor	Resultado
NOT	Verdadero	Falso
NOT	Falso	Verdadero



# Ejemplos con “Not”

- ¿NOT los michis son aves?

**Verdadero**

¿Los michis **NO** son aves?

- ¿NOT los michis tienen 4 patas?

**Falso**

¿Los michis **NO** tienen 4 patas?



# ¡Vamos al código!





# Reto

¿Cuál es el resultado de estas operaciones?

**¡Déjalo en los aportes de la clase!**



```
// Suponiendo estas variables
$es_un_michi_grande = true;
$le_gusta_comer = true;
$sabe_volar = false;
$tiene_2_patas = false;

// ¿Cuál es el resultado?
$es_un_michi_grande && $le_gusta_comer
$es_un_michi_grande || $sabe_volar
$sabe_volar || $tiene_2_patas
!$le_gusta_comer
!$le_gusta_comer || $es_un_michi_grande
```



# Veamos el reto

¿Cómo te fue? 00



```
// Suponiendo estas variables
$es_un_michi_grande = true;
$le_gusta_comer = true;
$sabe_volar = false;
$tiene_2_patas = false;

// ¿Cuál es el resultado?
$es_un_michi_grande && $le_gusta_comer // true

$es_un_michi_grande || $sabe_volar // true

$sabe_volar || $tiene_2_patas // false

!$le_gusta_comer // false

!$le_gusta_comer || $es_un_michi_grande // true
```

1  
0

1 0 1

1 0 1

# Operadores aritméticos

1  
0





# Operadores aritméticos

Son los operadores que nos ayudan a hacer operaciones matemáticas.



# Operadores aritméticos

En PHP tenemos los siguientes operadores:

Tipo	Descripción	Operador	Ejemplo
Adición	Suma dos o más números.	+	$\$a + \$b + \$c$
Sustracción	Resta dos o más números.	-	$\$a - \$b - \$c$
Multiplicación	Multiplica dos o más números.	*	$\$a * \$b * \$c$
División	Divide dos o más números.	/	$\$a / \$b / \$c$
Módulo	Devuelve el residuo de una división.	%	$\$a \% \$b$
Potenciación	Eleva un número al exponente dado.	**	$\$a ** \$b$
Identidad	Convierte un string a int o float, según sea el caso.	+	$+\$a$
Negación	Convierte un número positivo a negativo.	-	$-\$a$



# ¡Vamos al código!



1  
0

1 0 1

1 0 1

# Operadores relacionales

1  
0





# Operadores relacionales

Son los operadores que nos ayudan a comparar dos valores.



# ¡Vamos al código!



1  
0

1 0 1

1 0 1

# ¡Más operadores!

1  
0





# ¡Vamos al código!



1  
0

1 0 1

1 0 1

# Precedencia de operadores

1  
0





# ¿Qué es la precedencia?

En pocas palabras, nos dicen qué operación pasará primero y qué operación pasará después. Por ejemplo:

¿Primero suma y luego asigna? ¿O primero asigna y luego suma?

```
$contador = 0;  
$nuevo_valor = $contador++;
```



# ¿Qué es la precedencia?

La respuesta es: primero asigna y luego suma, por lo que nuestra variable \$nuevo\_valor valdrá “0”.

Pero... ¿por qué pasa esto?



```
$contador = 0;
```

```
$nuevo_valor = $contador++;
```



```
$contador = 0;
```

```
$nuevo_valor = $contador++;
```



```
$michis_4_patas = true;  
$michis_programan_con_PHP = false;  
  
$resultado = $michis_4_patas and $michis_programan_con_PHP;  
  
var_dump( $resultado ); // true
```



# ¡Vamos al código!



# Tu primer programa

¿Qué hora es?

1 0 1

1 0

1 0 1

1 0



# Reto

Ahora te toca hacer un programa que convierta horas, minutos y segundos a únicamente segundos. 



=

¡Suerte!



—



-



—



=

# Curso Básico de PHP

Carlos Gómez  
@RetaxMaster



1  
0  
1  
0  
1

1  
0  
1  
0  
1



# En este curso aprendiste...

- El proceso de interacción con el backend de una página web.
- La sintaxis básica de PHP.
- Uso de variables y constantes.
- Tipos de datos y casting.
- Diferentes tipos de operadores.
- Precedencia de operadores.

1  
0

1 0 1

1 0 1

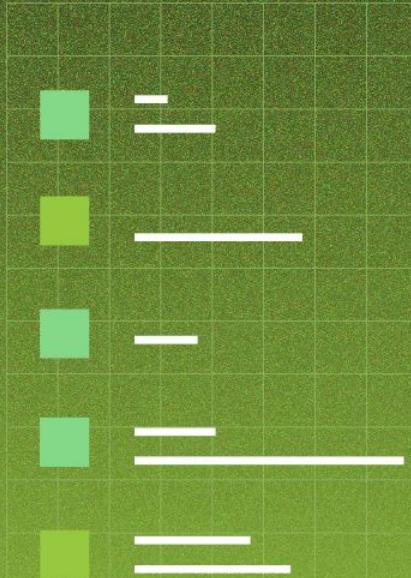
¿Quieres más  
cursos de PHP? ☺

1  
0





Nos vemos en el  
siguiente curso  
**UwU**



1  
0

1 0 1

1 0 1

# ¿Qué son los arreglos?

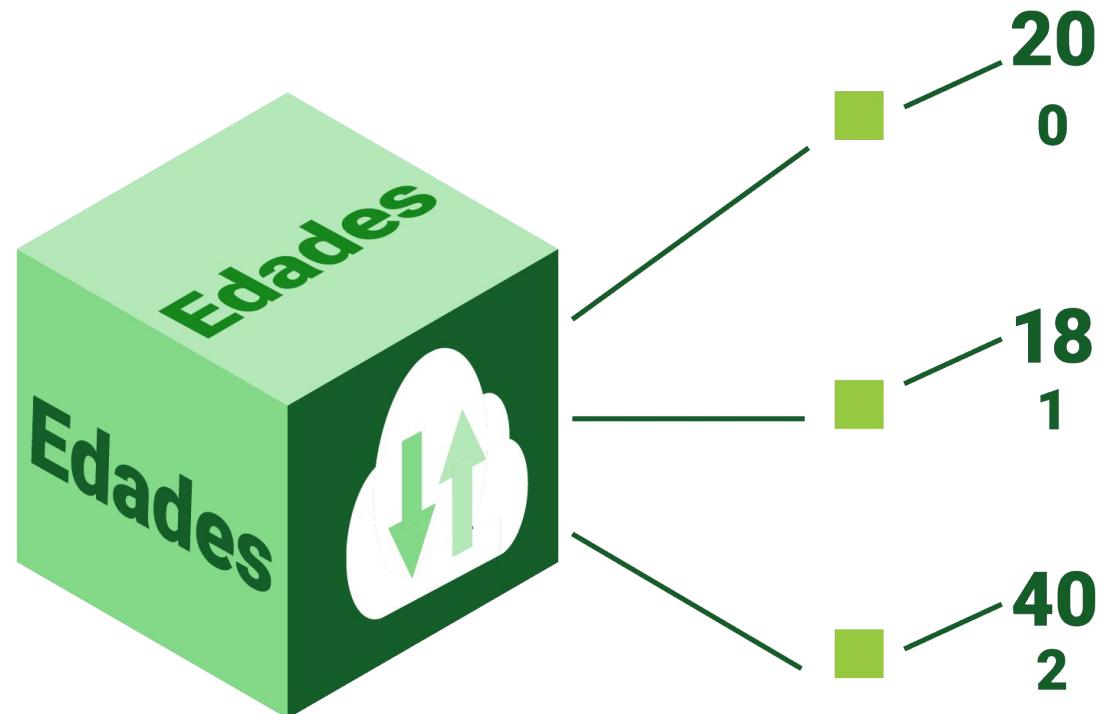
1  
0





# ¿Qué son los arreglos?

Una variable que  
puede guardar  
múltiples valores.





# ¡Vamos al código!



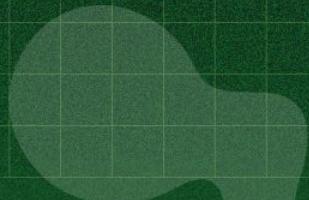
# Arreglos asociativos

1 0 1

1 0

1 0 1

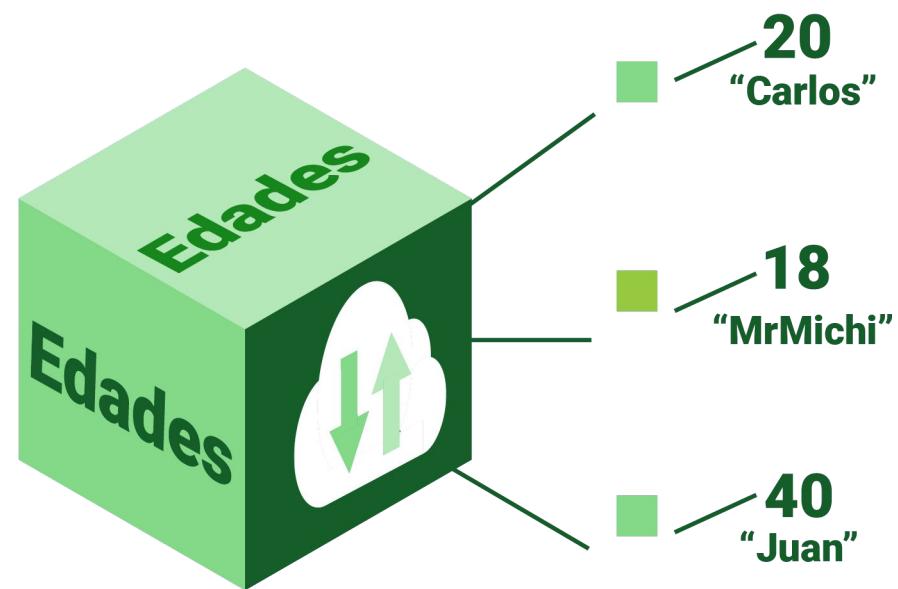
1 0





# Arreglos asociativos

Una variable que puede guardar múltiples valores y puedes acceder a ellos mediante una palabra.





# ¡Vamos al código!



1  
0

1 0 1

1 0 1

# Manipulando arreglos

1  
0





# ¡Vamos al código!



1  
0

1 0 1

1 0 1

Aprende a tomar  
decisiones con  
**if y else**

1  
0





# ¿Qué son las condicionales?

Muchas veces nos vemos en la necesidad de tomar una decisión u otra dependiendo de lo que el usuario elija. Las condicionales nos ayudan a elegir cuál bloque de código se debe ejecutar.



# if... else

La sentencia if nos ayuda a declarar una condición. Si la condición se cumple entonces se ejecuta el código del if; si no se cumple, se ejecuta el código del else.



# Sintaxis



```
if /* Condición */ {  
    // Si la condición se cumple, se ejecuta esto.  
}  
else {  
    // Si no se cumple, entonces se ejecuta esto.  
}
```



# ¿Recuerdas las tablas de verdad?

Un `if` **siempre** requerirá de un valor booleano. Por eso es que podemos poner **true** o **false** directamente.

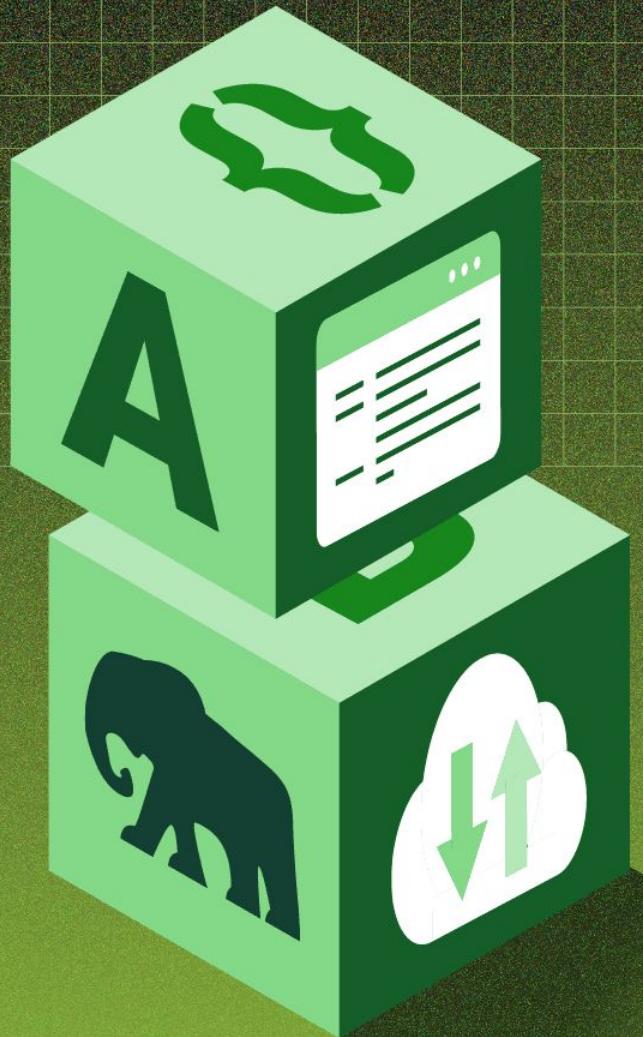
```
● ● ●  
if (true) {  
    // Si la condición se cumple, se ejecuta esto.  
}  
else {  
    // Si no se cumple, entonces se ejecuta esto.  
}
```



# ¡Vamos al código!



# Cómo organizar tu código con switch





# ¿Qué es switch?

Switch es una estructura de control que nos ayuda a elegir qué bloque de código ejecutar según el valor de alguna variable. Es una forma más fácil de tomar decisiones.



# ¿Cuándo debo usar switch?

## Úsalo cuando...

- Necesites decidir entre un caso u otro.
- Quieras una forma fácil de plantear diversas situaciones.
- Usarlo no signifique sacrificar la legibilidad de tu código.

## No lo uses cuando...

- Necesites hacer comparaciones complejas.
- Quieras verificar un rango de números.
- Necesites poner grandes bloques de código y/o tu código se vuelva ilegible.



# Sintaxis

```
switch /* Variable a evaluar */ {  
  
    case 1:  
        // Código para el caso 1  
        break;  
  
    case 2:  
        // Código para el caso 2  
        break;  
  
    case 3:  
        // Código para el caso 3  
        break;  
  
    default:  
        // Código por si ninguno de los casos se ejecuta.  
}
```



# ¡Vamos al código!



1  
0

1 0 1

1 0 1

# El ciclo while

1  
0





# ¿Qué es un bucle?

Un bucle es una estructura de control que nos ayuda a repetir un bloque de código las veces que lo necesitemos.

- Bucle
- Contador
- Iteración



# ¿Cómo funciona el bucle while?

- El bucle while, al igual que el if, recibe una condición booleana. Mientras que esa condición sea true, el ciclo se va a repetir. Dentro del while nosotros podemos cambiar el valor de nuestra variable para detener el ciclo.
- Por eso se dice que este es un ciclo indefinido.



# ¿Cómo funciona el bucle while?





# Sintaxis



```
while /* Condición */ {  
    // Aquí va todo el código que se va a repetir.  
}
```



# ¡Vamos al código!



# ¿Do... While?





# ¿Qué hace el ciclo Do... While?

- Este ciclo es exactamente igual que el ciclo while, también recibe una condición y puedes cambiar el valor de tu variable desde dentro del ciclo.
- La diferencia es que este ciclo se ejecuta al menos una vez, sin importar si la condición es verdadera o falsa.
- También es un ciclo indefinido.



# Sintaxis



```
do  {  
    // Aquí va todo el código que se va a repetir.  
}  
while /* Condición */;
```



# ¡Vamos al código!



1  
0

1 0 1

1 0 1

# El ciclo for

1  
0





# El ciclo for

Este ciclo nos permite definir una serie de condiciones, desde dónde queremos comenzar hasta dónde queremos terminar.

Nosotros le decimos explícitamente cuantas iteraciones hará, es por eso que decimos que este es un ciclo definido.



# Sintaxis



```
for /* Parámetros iniciales */; /* Condición */; /* Acciones */) {  
    // Este es el código que se repetirá  
}
```



# ¡Vamos al código!





# El ciclo foreach

La ventaja de este ciclo es que nos permite recorrer cualquier elemento que sea iterable sin tener que contar los elementos del mismo. Él por sí solo detecta cuántos elementos tiene dicho elemento.



# Sintaxis básica



```
foreach ($iterable as $valor) {  
    // Código a repetir por cada valor  
}
```



# Sintaxis obteniendo las llaves del elemento iterable



```
foreach ($iterable as $llave => $valor) {  
    // Código a repetir por cada valor  
}
```



# ¡Vamos al código!





# break y continue

También somos capaces de detener un ciclo o de avanzar a la siguiente iteración. Con estas palabras reservadas podemos tener más control sobre la ejecución de un ciclo.



# ¡Vamos al código!



1  
0

1 0 1

1 0 1

# Aprende a reutilizar tu código

Funciones

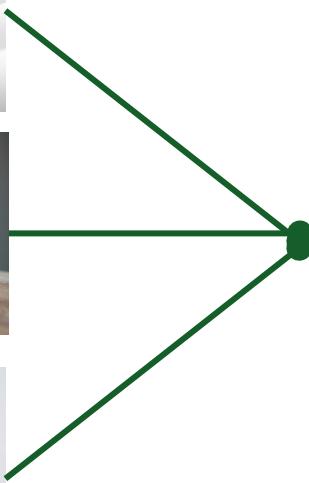
1  
0





# ¿Qué es una función?

Podemos verlas como un chef mágico al cual tú le das los ingredientes y te devuelve un pastel ya hecho.



**cocinar()**



# ¿Qué es una función?

También podemos verlas como “ponerle nombre a un bloque de código”.



```
echo "Poniendo los huevos";
echo "Poniendo la harina";
echo "Poniendo la leche";
echo "Mezclando los ingredientes";
echo "Metiéndolos al horno";

echo "¡Pastel listo!";
```



# ¿Qué es una función?

En resumen, una función nos sirve para separar todo un bloque de código, ponerle un nombre, y **utilizarla cuantas veces queramos** sin preocuparnos por cómo trabaja por dentro.



# ¿Qué es una función?

Nos ayuda a tener código más ordenado, reutilizable y fácil de entender.

Separar tu código en funciones para después reutilizarlo **es buena práctica**. Copiar y pegar código **no lo es**.



# ¡Vamos al código!





# Retornar valores de una función

¿Recuerdas que nuestro chef debía devolvernos un pastel ya cocinado? Retornar un valor significa decirle **explícitamente** a la función qué es lo que debe devolver.

Esto lo hacemos con la palabra reservada “return” y usualmente guardamos ese valor en una variable.



# Retornar valores de una función

Es como si el llamado a nuestra función se convirtiera en el resultado que deseamos por arte de magia.





# ¡Vamos al código!



¡Expande tu  
arsenal de  
funciones!



# Cómo interactúa PHP con HTML

1 0 1

1 0

1 0 1

1 0





# PHP es HTML con esteroides



PHP fue diseñado con el propósito de preprocesar HTML, es decir, de hacer operaciones como si fuera un lenguaje de programación.



# PHP es HTML con esteroides



Esto le permite imprimir páginas HTML dinámicas. Sí, PHP puede imprimir directamente HTML y podemos usar todas sus ventajas para generar documentos dinámicos.



# ¡Vamos al código!





# Ten en cuenta las buenas prácticas

Usualmente no es bueno mezclar tanta lógica de PHP con HTML. Hacer esto vuelve nuestro código ilegible, por lo que la mejor solución es separar nuestra lógica de nuestra vista.

Pero esto lo veremos en otro curso. 😊

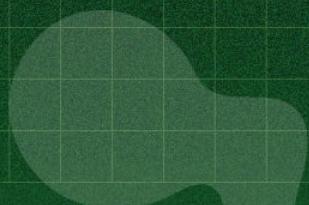
1  
0

1 0 1

1 0 1

# ¡Hagamos una calculadora!

1  
0





# En este curso aprendiste...

- Arreglos y cómo manipularlos.
- Toma de decisiones con condicionales.
- Repetir código con bucles.
- Reutilizar código con funciones.
- Combinar PHP con HTML.
- ¡Hicimos un proyecto!