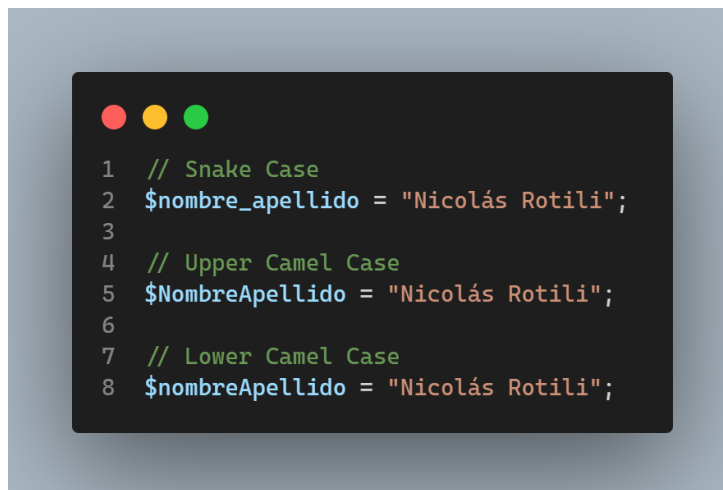


Variables

Las [variables](#)¹ en PHP se comportan de igual manera que en cualquier otro lenguaje, almacenan la información hasta que se reemplace o hasta que se detenga el funcionamiento del sistema.

Se representan con el signo pesos (\$) continuado por el nombre de la variable. En este lenguaje las variables son sensibles a mayúsculas y minúsculas y su nombre no puede iniciar con un número, debe ser con letras o _ (guion bajo).


Cada variable puede llamarse con el nombre que uno quisiese, pero se recomienda nombrarla con valores que permitan identificar lo que se almacena y utilizar convenciones en la escritura (*Fig. 1*).



```
1 // Snake Case
2 $nombre_apellido = "Nicolás Rotili";
3
4 // Upper Camel Case
5 $NombreApellido = "Nicolás Rotili";
6
7 // Lower Camel Case
8 $nombreApellido = "Nicolás Rotili";
```

Fig. 1

Una particularidad de PHP es que el lenguaje es débilmente tipado, es decir que no se necesita indicar el tipo de dato que va a almacenar la variable.



```
1 $nombre = "Nicolás"; //string
2 $apellido = "Rotili"; //string
3 $nombreCompleto = "$nombre $apellido"; //string
4 $nombreCompleto = '$nombre $apellido'; //string
5 $edad = 28; //int
6 $altura = 1.70; //float
7 $casado = true; //boolean
8 $hijos = null; //null
```

Fig. 2

¹ <https://www.php.net/manual/es/language.variables.basics.php>



Como se puede observar en la Fig. 2, los datos se almacenan sin problema y es PHP quien determinará de acuerdo a lo almacenado el tipo de dato para cada variable.

Existen varios tipos de datos en PHP. Si bien nosotros trabajaremos con los más comunes es importante reconocerlos en caso de visualizar algunos.

Más utilizados	Menos vistos
Integer	Resource
Float	Callable
Boolean	Iterable
String	
Array	
Object	
Null	

Para introducir contenido dentro de una variable, debemos utilizar luego de su nombre el **operador de asignación = (igual)** y a continuación lo que se desea contener. Si observamos en la Fig. 2, para insertar texto debemos encerrarlo entre comillas simples ('texto') o comillas dobles ("texto"). Podemos detectar la particularidad de que si utilizamos comillas dobles tenemos la posibilidad de interpolar texto con variables. Si deseamos almacenar números, valor null o booleanos, se escriben sin comillas.

Si deseamos mostrar los datos contenidos en cada una de las variables, debemos anteponer la palabra reservada [print](https://www.php.net/manual/es/function.print.php)² (solo 1 variable), [echo](https://www.php.net/manual/es/function.echo.php)³ (varias variables) o utilizar su forma abreviada al mezclarlo con HTML `<?= $variable ?>`. En ocasiones, para poder ver la estructura de una variable, deberemos usar la función [var_dump\(\\$variable\)](https://www.php.net/manual/es/function.var-dump.php)⁴ o [print_r\(\\$variable\)](https://www.php.net/manual/es/function.print-r.php)⁵.

² <https://www.php.net/manual/es/function.print.php>

³ <https://www.php.net/manual/es/function.echo.php>

⁴ <https://www.php.net/manual/es/function.var-dump.php>

⁵ <https://www.php.net/manual/es/function.print-r.php>

```
1 echo "Nombre completo: $nombreCompleto <br>";
2 echo "Edad: $edad <br>";
3 echo "Altura: $altura <br>";
4 echo "Casado: $casado <br>";
5 echo "Hijos: $hijos <br>";
```

Fig. 3

Al igual que en otros lenguajes, PHP también contiene arreglos unidimensionales, multidimensionales y asociativos.

La característica de los array es que podemos almacenar en una misma variable, varios tipos de datos (numéricos, booleanos, cadenas). Para poder generar una variable del tipo array, debemos usar luego de la asignación [] y dentro de ellos todos los datos a almacenar.

```
1 // Array unidimensional
2 $datos = array("Nicolás Rotili", 28, 1.70, true, null);
3 $datos = ["Nicolás Rotili", 28, 1.70, true, null];
```

Cada dato contenido en un array se encuentra en una posición denominada índice y como particularidad, la posición inicial siempre es el 0. Es decir que si nosotros queremos recuperar la altura contenida en el arreglo datos debemos tener en cuenta la posición. Ej: `$datos[2]`.

Recordar las posiciones de cada dato guardado resulta complicado, más aún cuando el desarrollo es colaborativo. En estos casos se utilizan arreglos asociativos en el que el índice en vez de ser un número, pasa a ser una palabra descriptiva de lo que contiene.

```
1 $datos = [
2     "nombre" => "Nicolás",
3     "apellido" => "Rotili",
4     "edad" => 28,
5     "altura" => 1.70,
6     "casado" => false,
7     "hijos" => null,
8 ];
```

En este caso, para poder asignar los valores al índice creado no utilizamos el operador de asignación (=), sino => , simulando la forma de una flecha.

De esta manera, obtener los datos contenidos en el array resulta mucho más sencillo ya que, para poder recuperar el nombre del registro contenido,

solo deberíamos llamarlo de la siguiente manera: `$datos['nombre']`.

En el caso que quisiéramos almacenar más de un registro, deberíamos acudir a un arreglo multidimensional.



```
1 $coches = [  
2     [  
3         "marca" => "Seat",  
4         "modelo" => "Ibiza",  
5         "año" => 2015,  
6     ],  
7     [  
8         "marca" => "Renault",  
9         "modelo" => "Megane",  
10        "año" => 2015,  
11    ],  
12 ];
```

Fig. 5

Este array, a diferencia de los otros, tiene dos índices, el primero numérico y el segundo con el nombre que lo definimos. Es decir, si quisiéramos recuperar todos los valores del Ibiza, debemos hacerlo de la siguiente manera:

- `$coches[0]['marca']`
- `$coches[0]['modelo']`
- `$coches[0]['año']`