

# Conceptos básicos de PHP (I)

Ricardo Pérez López

IES Doñana, curso 2019/2020

## Índice general

<b>1. Introducción a PHP</b>	<b>2</b>
1.1. Prueba . . . . .	2
1.2. Página web de PHP . . . . .	2
1.3. Instalación de PHP . . . . .	2
1.4. Documentación y búsqueda de información . . . . .	4
<b>2. Sintaxis básica</b>	<b>4</b>
2.1. Datos e instrucciones . . . . .	4
2.2. Sentencias y comandos . . . . .	4
2.3. Expresiones, operadores y funciones . . . . .	4
<b>3. Funcionamiento del intérprete</b>	<b>4</b>
3.1. Ejecución . . . . .	4
3.2. Etiquetas <code>&lt;?php</code> y <code>?&gt;</code> . . . . .	4
3.3. Modo dual de operación . . . . .	4
<b>4. Variables</b>	<b>5</b>
4.1. Conceptos básicos . . . . .	5
4.2. Destrucción de variables . . . . .	5
4.3. Operadores de asignación por valor y por referencia . . . . .	5
4.4. Variables predefinidas . . . . .	5
<b>5. Tipos básicos de datos</b>	<b>5</b>
5.1. Lógicos ( <code>bool</code> ) . . . . .	5
5.2. Numéricos . . . . .	6
5.3. Cadenas ( <code>string</code> ) . . . . .	6
5.4. Nulo ( <code>null</code> ) . . . . .	7
<b>6. Manipulación de datos</b>	<b>7</b>
6.1. Precedencia de operadores . . . . .	7
6.2. Operadores de asignación compuesta . . . . .	7
6.3. Comprobaciones . . . . .	7
6.4. Conversiones de tipos . . . . .	7
6.5. Comparaciones . . . . .	9

<b>7. Constantes</b>	<b>9</b>
7.1. <code>define()</code> y <code>const</code> . . . . .	10
7.2. Constantes predefinidas . . . . .	10
7.3. <code>defined()</code> . . . . .	10
<b>8. Ejercicios</b>	<b>10</b>
8.1. Actividades . . . . .	10
8.2. Problemas . . . . .	10
<b>9. Respuestas a las preguntas</b>	<b>11</b>

## 1. Introducción a PHP

### 1.1. Prueba

```

espec NAT3
generos natural
operaciones
  0:  $\rightarrow$  natural
  suc: natural  $\rightarrow$  natural
  _ + _: natural  $\times$  natural  $\rightarrow$  natural
var
  x, y: natural
ecuaciones
   $x + 0 = 0$ 
   $x + \text{suc}(y) = \text{suc}(x + y)$ 
fspec

```

### 1.2. Página web de PHP

### 1.3. Instalación de PHP

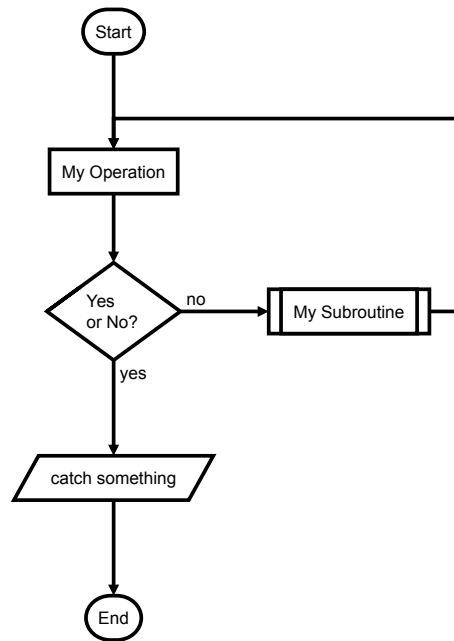
```

<?php

$j = json_decode($argv[1], true);
// print_r($j);

echo '```{=latex}' . PHP_EOL;
foreach ($j['questions'] as $question) {
    echo '\begin{Exercise}[label={\the\value{Exercise}}]' . PHP_EOL;
    echo $question['q'] . PHP_EOL;
    echo PHP_EOL;
    echo '(Para ver la respuesta pulsa aquí:~\ref{\ExerciseLabel-Answer})' . PHP_EOL;
    echo '\end{Exercise}' . PHP_EOL;
}

```



Prueba de diagrama de flujo

```

echo '\begin{Answer}[ref=\ExerciseLabel]' . PHP_EOL;
echo '\begin{itemize}' . PHP_EOL;
foreach ($question['a'] as $ans) {
    if ($ans['correct']) {
        echo '\item ' . $ans['option'] . PHP_EOL;
    }
}
echo '\end{itemize}';
echo PHP_EOL;
echo $question['correct'] . PHP_EOL;
echo '\end{Answer}' . PHP_EOL;
}
echo '````' . PHP_EOL;
<?php

// $f = file_get_contents('juan');

$j = json_decode($argv[1], true);
// print_r($j);

echo '````{=html}' . PHP_EOL;
echo '<script data-quiz>' . PHP_EOL;
foreach ($j['questions'] as $question) {
    $question['correct'] = '<p><span>Eso es!</span>' . $question['correct'] . '</p>';
    $question['incorrect'] = '<p><span>Mmmm... no.</span>' . $question['incorrect'] . '</p>';
}
echo 'quiz = ' . json_encode($j, JSON_UNESCAPED_SLASHES);
echo '</script>' . PHP_EOL;
echo '````' . PHP_EOL;

```

## 1.4. Documentación y búsqueda de información

# 2. Sintaxis básica

## 2.1. Datos e instrucciones

### Pregunta 1

What number is the letter A in the English alphabet?

(Para ver la respuesta pulsa aquí: 1)

## 2.2. Sentencias y comandos

### 2.2.1. Comando `echo`

## 2.3. Expresiones, operadores y funciones

ricpelo's note: *Ejemplos:* aritmética, `cos()`, `max()`

ricpelo's note: `print()` no es una función. Cuidado.

# 3. Funcionamiento del intérprete

## 3.1. Ejecución

### 3.1.1. Por lotes

### 3.1.2. Interactiva

#### 3.1.2.1. `php -a`

#### 3.1.2.2. PsySH

## 3.2. Etiquetas `<?php` y `?>`

## 3.3. Modo dual de operación

ricpelo's note: Se llaman *modo HTML* y *modo PHP*.

## 4. Variables

### 4.1. Conceptos básicos

### 4.2. Destrucción de variables

### 4.3. Operadores de asignación por valor y por referencia

ricpelo's note: En `$b =& $a;`, `$b` **NO** está apuntando a `$a` o viceversa. Ambos apuntan al mismo lugar.

### 4.4. Variables predefinidas

ricpelo's note: `$_ENV` no funciona en la instalación actual (ver `variables_order` en `php.ini`. Habría que usar `get_env()`.

## 5. Tipos básicos de datos

### 5.1. Lógicos (`bool`)

ricpelo's note: Se escriben en minúscula: `false` y `true`.

ricpelo's note: `boolean` es sinónimo de `bool`, pero debería usarse `bool`.

#### 5.1.1. Operadores lógicos

ricpelo's note: *Cuidado:*

- `false and (true && print('hola'))` no imprime nada y devuelve `false`, por lo que **el código va en cortocircuito y se evalúa de izquierda a derecha** incluso aunque el `&&` y los paréntesis tengan más prioridad que el `and`.

- Otra forma de verlo es comprobar que `print('uno') and (1 + print('dos'))` escribe `uno-dos` (y devuelve `true`), por lo que la evaluación de los operandos del `and` se hace de izquierda a derecha aunque el `+` tenga más prioridad (y encima vaya entre paréntesis).

- En el manual de PHP se dice que: *"La precedencia y asociatividad de los operadores solamente determinan cómo se agrupan las expresiones, no especifican un orden de evaluación. PHP no especifica (en general) el orden en que se evalúa una expresión y se debería evitar el código que se asume un orden específico de evaluación, ya que el comportamiento puede cambiar entre versiones de PHP o dependiendo de código circundante."*

- Pregunta que hice al respecto en StackOverflow.

## 5.2. Numéricos

### 5.2.1. Enteros (**int**)

ricpelo's note: `integer` es sinónimo de `int`, pero debería usarse `int`.

### 5.2.2. Números en coma flotante (**float**)

ricpelo's note: `double` es sinónimo de `float`, pero debería usarse `float`.

### 5.2.3. Operadores

#### 5.2.3.1. Operadores aritméticos

#### 5.2.3.2. Operadores de incremento/decremento

## 5.3. Cadenas (**string**)

ricpelo's note: Se usa `{$var}` y no `${var}`

### 5.3.1. Operadores de cadenas

#### 5.3.1.1. Concatenación

**5.3.1.2. Acceso y modificación por caracteres** ricpelo's note: - `echo $a[3]`  
- `$a[3] = 'x';`

#### 5.3.1.3. Operador de incremento #opcional

### 5.3.2. Funciones de manejo de cadenas

#### 5.3.3. Extensión *mbstring*

ricpelo's note: - `$a[3]` equivale a `mb_substr($a, 3, 1)`  
- `$a[3] = 'x';` no tiene equivalencia directa. Se podría hacer:  
`$a = mb_substr($a, 2, 1) . 'x' . mb_substr($a, 4);`

## 5.4. Nulo (`null`)

ricpelo's note: `is_null()` vs. `=== null`

ricpelo's note: El tipo `null` y el valor `null` se escriben en minúscula.

## 6. Manipulación de datos

### 6.1. Precedencia de operadores

### 6.2. Operadores de asignación compuesta

ricpelo's note: `$x <op>= $y`

### 6.3. Comprobaciones

#### 6.3.1. De tipos

##### 6.3.1.1. `gettype()`

6.3.1.2. `is_*()` ricpelo's note: Poco útiles en formularios, ya que sólo se reciben `strings`.

#### 6.3.2. De valores

##### 6.3.2.1. `is_numeric()`

##### 6.3.2.2. `ctype_*()`

### 6.4. Conversiones de tipos

#### 6.4.1. Conversión explícita (forzado o *casting*) vs. automática

ricpelo's note: Conversión de cadena a número

#### 6.4.2. Conversión a **bool**

#### 6.4.3. Conversión a **int**

#### 6.4.4. Conversión a **float**

#### 6.4.5. Conversión de **string** a número

ricpelo's note: ¡Cuidado!:

La documentación dice que `$x = 1 + "pepe"` o `$x = 1 + "10 pepe"` funciona, pero dependiendo del valor de `error_reporting` en `php.ini`, puede dar un **PHP Warning: A non-numeric value encountered** o un **PHP Warning: A non well formed numeric value encountered**, respectivamente.

- Si `error_reporting = E_ALL`, dará el mensaje de advertencia.

Además, en PsySH no funcionará, es decir, que `$x` no se asignará al valor. En `php -a` sí funcionará (aunque da el mismo mensaje de advertencia).

- Si `error_reporting = E_ALL & ~E_NOTICE`, no lo dará.

Además, funcionará tanto en PsySH como en `php -a`.

#### 6.4.6. Conversión a **string**

#### 6.4.7. Funciones de obtención de valores

ricpelo's note: Hacen más o menos lo mismo que los *casting* pero con funciones en lugar de con operadores. Puede ser interesante porque las funciones se pueden guardar, usar con *map*, *reduce*, etc.

##### 6.4.7.1. `intval()`

##### 6.4.7.2. `floatval()`

##### 6.4.7.3. `strval()`

##### 6.4.7.4. `boolval()`

#### 6.4.8. Funciones de formateado numérico

##### 6.4.8.1. `number_format()`

##### 6.4.8.2. `money_format()` `setlocale()`

ricpelo's note: `setlocale(LC_ALL, 'es_ES.UTF-8');` // Hay que poner el `*locale*` completo, con la codificación y todo (`.UTF-8`)



## 6.5. Comparaciones

### 6.5.1. Operadores de comparación

ricpelo's note: "250" < "27" devuelve `false`

ricpelo's note: Si se compara un número con un string o la comparación implica strings numéricos, entonces cada string es convertido en un número y la comparación realizada numéricamente.

### 6.5.2. `==` vs. `===`

### 6.5.3. Ternario (`?:`)

### 6.5.4. Fusión de `null` (`??`)

ricpelo's note: Equivalente al `COALESCE()` de SQL.

### 6.5.5. Reglas de comparación de tipos

## 7. Constantes

ricpelo's note: Diferencias entre constantes y variables:

- Las constantes no llevan el signo dólar (\$) como prefijo.
- Antes de PHP 5.3, las constantes solo podían ser definidas usando la función `define()` y no por simple asignación.
- Las constantes pueden ser definidas y accedidas desde cualquier sitio sin importar las reglas de acceso de variables.
- Las constantes no pueden ser redefinidas o eliminadas una vez se han definido.
- Las constantes podrían evaluarse como valores escalares. A partir de PHP 5.6 es posible definir una constante de array con la palabra reservada `const`, y, a partir de PHP 7, las constantes de array también se pueden definir con `define()`. Se pueden utilizar arrays en expresiones escalares constantes (por ejemplo, `const FOO = array(1,2,3)[0];`), aunque el resultado final debe ser un valor de un tipo permitido.

### 7.1. `define()` y `const`

### 7.2. Constantes predefinidas

### 7.3. `defined()`

## 8. Ejercicios

### 8.1. Actividades

1. Busca información sobre la función `time()` usando, al menos, tres formas distintas.
2. Explica, con tus propias palabras, la diferencia entre:
  - 2.1. Un dato y una instrucción.
  - 2.2. Una expresión y una sentencia.
  - 2.3. Una sentencia y un comando.
  - 2.4. Una función y un operador.
3. ¿Es `echo` una función? ¿A dónde acudes para saberlo?
4. ¿Es lo mismo *modo de ejecución* que *modo de operación*? Explica cuáles son y en qué consisten los diferentes modos de ejecución y de operación en PHP.
5. ¿Qué ventajas e inconvenientes tiene usar PsySH frente al intérprete integrado?
6. ¿Qué tipos de asignación de variables existen en PHP? Explica sus diferencias y pon ejemplos de uso.
7. ¿Qué son las variables predefinidas? Enumera las más importantes.
8. Calcula el valor de las siguientes expresiones y razona por qué tienen ese valor:
  - 8.1. `false and true or 1`
  - 8.2. `1 == 1.0`
  - 8.3. `1 == 0.9999999999999999`
  - 8.4. `floor((0.1 + 0.7) * 10)`
  - 8.5. `'1' == 1`
  - 8.6. `empty('')`
9. ¿`$a[3]` equivale a `mb_substr($a, 3, 1)`? Razona la respuesta.
10. Define con tus propias palabras el significado de *asociatividad* y de *prioridad*. ¿Por qué la expresión `1 == 1 == 1` es incorrecta pero `1 <= 1 == 1` es correcta (y cuál es su valor, por cierto)?

### 8.2. Problemas

1. Escribir un programa en PHP que...

## 9. Respuestas a las preguntas

### Respuesta a la Pregunta 1

- 1  
The letter A is the first letter in the alphabet!