Introducción rápida a la Programación con PHP

Material de estudio preparatorio para el curso de **«Desarrollo de aplicaciones Web en PHP orientado a objetos con MVC y PDO/MySQL»** a cargo de **Eugenia Bahit**

Índice de contenidos

Elementos que componen los lenguajes de programación algebraicos de alto nivel

Variable

- **ES** una forma de almacenar datos de forma temporal
- **SIRVE** para guardar datos temporalmente y utilizarlos más de una vez
- **SE IDENTIFICA** anteponiendo un signo dólar $

Se ve así:

```
$variable = 1
```

Instrucción

- **ES** una orden que se le da al ordenador en forma de código fuente
- **SIRVE** para que el ordenador haga lo que necesitamos
- **SE IDENTIFICA** colocando un punto y coma ; al final de cada orden

Se ve así:

```
$variable = 1;
```

Tipo de dato

- **ES** el tipo de información que puede contener una variable
- **SIRVE** para Para saber qué puedo hacer con la información de esa variable

  (por ejemplo, si se que la información es un número sabré que puedo hacer operaciones matemáticas)

Se ve así:

Número entero:

```
$edad = 45;
```

Número real (la coma se indica con un punto .):

```
$precio = 43.75;
```

Verdadero o Falso:

Hay un tipo de datos llamado "boolean" (o mal dicho "buleano" en español) que sirve para decir si algo es verdadero o no:

```
$estoy_viva = true;
$soy_italiana = false;
```

Texto y otros datos:

Todos los demás datos, son considerados "cadenas de texto" y se usan entre comillas dobles:

```
$ciudad = "Buenos Aires";
$fecha_de_nacimiento = "25/08/1978";
```

Operador matemático

- **ES** un símbolo que indica una operación a realizar entre dos operandos
- **SIRVE** para efectuar operaciones matemáticas con números y variables numéricas

Se ve así:

```
$suma = 10 + 15;
$resta = $a - $b;
$multiplicacion = 15 * $b;
$division = $a / $b;
$operacion_combinada = ($a + $b) * $c – (12 / 4 * $n);
```

Operador de comparación

- **ES** un símbolo que permite comparar dos valores
- **SIRVE** para comparar datos y variables obteniendo sí o no (true o false) como respuesta a la comparación
- **SE IDENTIFICA** agrupando los valores a comparar entre paréntesis

Se ve así:

```
$es_igual_a = ($a == $b);
$es_distinto_que = ($a != $b);
$es_mayor_que = ($a > $b);
$es_mayor_o_igual_que = ($a >= $b);
$es_menor_que = ($a < $b);
$es_menor_o_igual_que = ($a <= $b);
```

Operador lógico

- **ES** un símbolo o palabra que evalúa dos o más valores como verdaderos o falsos
- **SIRVE** para evaluar la información y controlar el flujo de la misma
- **SE IDENTIFICA** agrupando los valores a evaluar entre paréntesis

Se ve así:

```
($a AND $b)      $a y $b deben ser verdaderos
($a OR $b)       $a o $b deben ser verdaderos
($a XOR $b)      $a o $b deben ser verdaderos (pero no ambos)
(!$a)            $a NO debe ser verdadero
($a)             $a DEBE ser verdadero
```

Token

- **ES** una instrucción directa del lenguaje
- **SIRVE** para ejecutar instrucciones predefinidas en el lenguaje

Se ve así:

```
print "Hola Mundo";
print $variable;
print 15;
```

Estructura de control de flujo

- **ES** un token que permite agrupar instrucciones
- **SIRVE** para controlar el flujo de la información
- **SE IDENTIFICA** agrupando las instrucciones entre llaves { }

Se ve así:

```
if ($a == $b AND !$c) {
     print "$a es igual a $b";
}

while ($a < $b) {
     print "$a es menor que $b";
     $a = $a * 2;
}
```

Condicional

- **ES** una estructura de control para evaluar condiciones
- **SIRVE** para ejecutar determinadas instrucciones dependiendo de si una condición se cumple o no

Se ve así:

Actúo solo si UNA condición se cumple:

```
if ($edad > 18) {
     print "Solo imprimo esto si la variable edad
            es mayor que 18";
}
```

Hago una cosa si la condición se cumple y sino hago otra:

```
if ($edad > 18) {
     print "Imprimo esto si la variable edad
            es mayor que 18";
} else {
     print "Pero si la condición anterior no se cumple
            imprimo esto otro";
}
```

Evalúo más de una condición:

```
if ($edad <= 11) {
     print "Eres un niño";
} elseif ($edad > 11 AND $edad < 14) {
     print "Eres preadolescente";
} elseif ($edad >= 14 AND $edad < 18) {
     print "Eres adolescente";
} else {
     print "Ya eres adulto";
}
```

Colección de datos (array)

- **ES** una forma de almacenar varios valores de diferentes tipos en una misma variable
- **SIRVE** para agrupar valores
- **SE IDENTIFICA** con la palabra (token) array

Se ve así:

Uso simple (cada valor se asocia implícitamente a su número de posición, comenzando en cero, dentro de la colección):

```
$coleccion = array("Pérez", "Juan", 75, 1.83, True);
print $coleccion[1];  // Imprime Juan
```

Asociado a nombres de claves explícitos:

```
$coleccion = array(
    "Apellido"=>"Pérez",
    "Nombre"=>"Juan",
    "Edad"=>75,
    "Estatura"=>1.83,
    "Casado"=>True
);

print $coleccion["Nombre"];  // Imprime Juan
```

Estructura de control iterativa foreach

- **ES** una estructura de control de flujo
- **SIRVE** para repetir una misma acción de forma iterativa sobre cada elemento de una misma colección

Se ve así:

```
$coleccion = array("manzana", "pera", "naranja");

foreach($coleccion as $posicion=>$fruta) {
    print "La $fruta está en la posición $posicion del array.";
}



$datos = array(
    "Apellido"=>"Pérez",
    "Nombre"=>"Juan",
    "Edad"=>75,
    "Estatura"=>1.83,
    "Casado"=>True
);

foreach($datos as $clave=>$valor) {
    print "$clave: $valor \n";
}
```

Función

- **ES** una forma de agrupar instrucciones para usar luego varias veces (como las variables que almacenan datos, las funciones almacenan instrucciones)
- **SIRVE** para reutilizar código
- **SE IDENTIFICA** anteponiendo la palabra function al nombre (para definirla) y encerrando las isntrucciones entre llaves

Se ve así:

Al definirla para luego usarla:

```
function calcular_iva($importe_bruto=0) {
     $iva = $importe_bruto * 0.21;
     $importe_neto = $importe_bruto + $iva;
     print "El IVA de $importe_bruto es $importe_neto \n";
}
```

Al usarla:

```
calcular_iva(1500);
calcular_iva(303.45);
calcular_iva(100);
calcular_iva(97);
```

Clase

- **ES** una forma de agrupar variables y funciones
- **SIRVE** para crear una colección de datos personalizada
- **SE IDENTIFICA** anteponiendo la palabra class al nombre (para definirla) y encerrando las variables y funciones entre llaves

Se ve así:

Al definirla para luego usarla:

```
class Producto {

     public $denominacion = '';
     public $precio = 0.0;

     function guardar() {
           instrucciones de la función guardar
     }

     function eliminar($producto=0) {
           instrucciones de la función eliminar
     }
}
```

Al usarla, primero debe crearse una variable con el nuevo tipo de dato:

```
$producto = new Producto();
```

Para acceder a una variable dentro de una clase:

```
$producto->precio = 100.75;
print $producto->precio;
```

Para acceder a una función dentro de una clase:

```
$producto->guardar();
```

Funciones propias de PHP

Algunas de las funciones que trae PHP:

require_once("ruta/al/archivo.php")

Permite importar todo el código fuente del archivo archivo.php dentro del archivo en el que se llama a la función.

```
<?php

require_once("mis_funciones.php");
require_once("mis_variables.php");

...código propio de este archivo
?>
```

print_r($variable);

Muestra en pantalla como se compone internamente una colección de tipo array o personalizada (objeto):

```
<?php

require_once("clase_producto.php");

$array = array(1, 2, 3, 4);
print_r($array);

$producto = new Producto();
$producto->denominacion = "Zapatilla Adidad";
$producto->precio = 1500.75;

print_r($producto);

?>
```

file_get_contents("ruta/al/archivo.html")

Lee el contenido de un archivo alamacenandolo en una variable:

```
<?php

$html = file_get_contents("plantilla.html");
print $html;

?>
```

str_replace($buscar, $reemplazar_por, $donde);

Reemplaza los elementos del array $buscar por los elementos del array $reemplazar dentro de la cadena de texto $donde

```
<?php

$plantilla = "
	<header>Welcome!</header>
	<h1>{TITULO}</h1>
	<div>{CONTENIDO</div>
	<footer>&copy; 2015</footer>
";

$titulo = "Este es el titulo de mi página";
$contenido = file_get_contents("home.html");

$buscar = array("{TITULO}", "{CONTENIDO}");
$reemplazar = array($titulo, $contenido);

print str_replace($buscar, $reemplazar, $plantilla);

?>
```

Sobre el curso

Nombre del curso: Desarrollo de aplicaciones Web en PHP orientado a objetos con MVC y PDO/MySQL

Objetivo: Aprender a programar en PHP (desde cero) desarrollando un ABM orientado a objetos con arquitectura MVC y bases de datos MySQL

Duración: 3-4 meses

Docente: Eugenia Bahit

Nivel: Inicial

Complejida: Media

Modalidad:

- a distancia (online)
- 1 clase x semana en vivo vía Hangout (con participación directa de los alumnos)
- De 2 a 4 estudiantes por curso
- Consultas x e-mail de lunes a viernes

Costo*:

- Individual (1 solo alumno x curso):

  USD 180 / mes | **ARS** | MXN | EUR

- Colaborativo en pareja (2 estudiantes x curso):

  USD 132 / mes / alumno | **ARS** | MXN | EUR

- Colaborativo en equipo (3 a 4 estudiantes x curso):

  USD 112 / mes / alumno | **ARS** | MXN | EUR

(*) el pago es mensual. Precios válidos desde 02/03/2015 hasta el 02/05/2015. Los precios en MXN y EUR son solo a título informativo.

Días de cursada (sujeto a disponibilidad):

Miércoles, 15:00 a 16:30 HS (GMT -3:00, hora de Argentina)

Miércoles, 17:00 a 18:30 HS (GMT -3:00, hora de Argentina)

Reservas e Inscripción:

http://www.cursosdeprogramacionadistancia.com/contacto

Próximo inicio:

Miércoles 1º de Abril de 2015