# UT2 Guiones de servidor PHP I

Desarrollo web en entorno servidor

# Algunos contenidos iniciales

- Estructura de un página PHP
- Sentencias y comentarios
- Salida de información en el navegador
- Variables y tipos de datos. Constantes.
- Operadores aritméticos

# Estructura de una página PHP

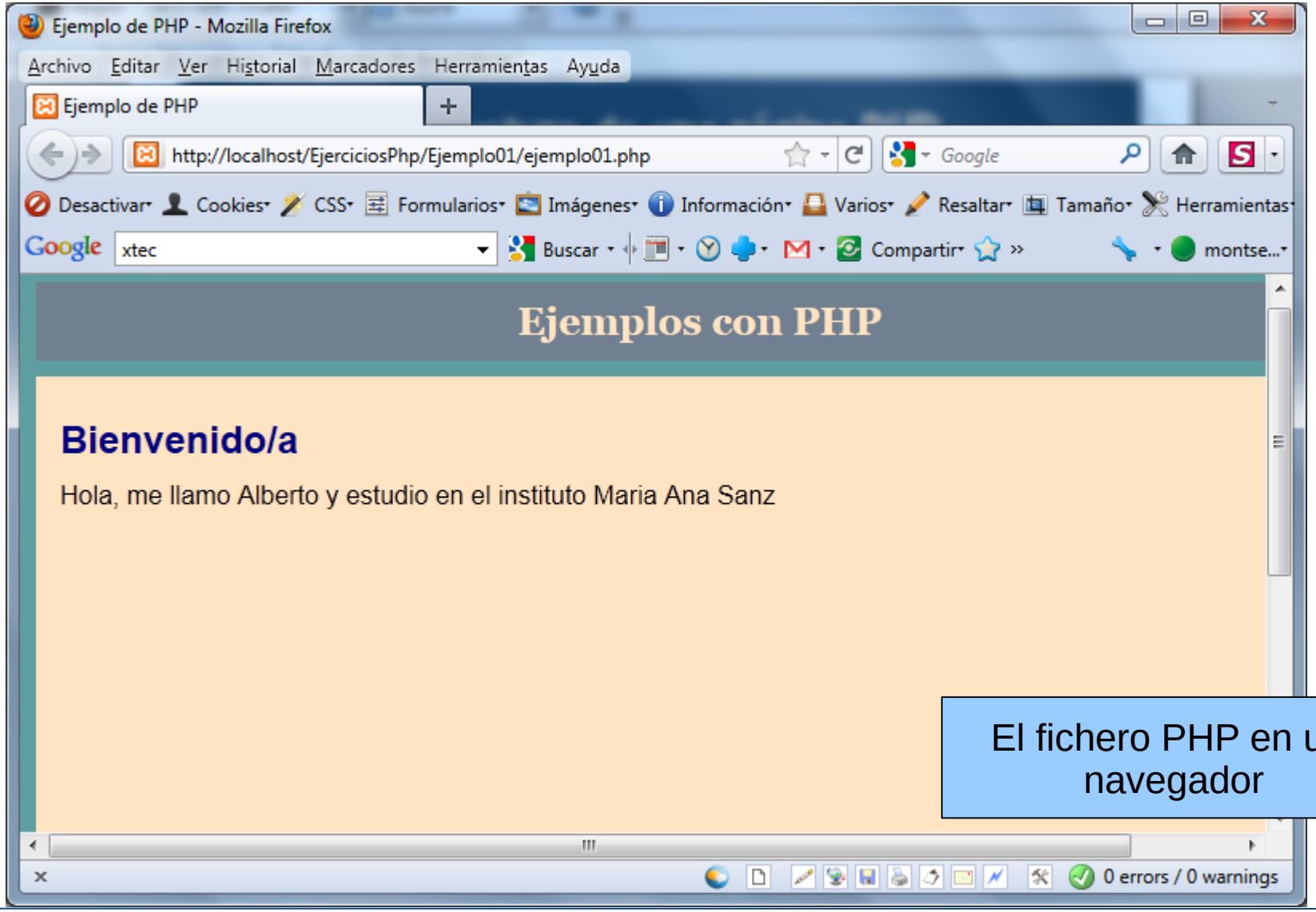
- Extensión de un fichero PHP .php
- Un fichero PHP incluye código HTML y PHP “embebido”

```
<<!DOCTYPE html>   o

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">          o…..(de otro tipo)

<html xmlns="http://www.w3.org/1999/xhtml">
   <head>
      <title>Un ejemplo en PHP</title>
      <link rel="stylesheet" type="text/css"  href="estilo.css"/>
   </head>
   <body>
      <h2>Bienvenido/a</h2>
     <?php
                  echo “Mi primer ejemplo en PHP”;
        ?>
   </body>
</html>
```

# Estructura de una página PHP

```
<!DOCTYPE html>
<html >
   <head>
      <title>Otro ejemplo en PHP</title>
      <link rel="stylesheet" type="text/css"  href="estilo.css"/>
   </head>
   <body>
         <div id="contenido">
      <div id="cabecera"><h2>Primeros ejemplos PHP</h2></div>
      <div id="principal">
         <?php
            $nombre = "Alberto";
            $instituto = "Maria Ana Sanz";
         ?>
         <h2>Bienvenido/a</h2>
         <?php
            echo "Hola, me llamo $nombre y estudio en el instituto $instituto";
         ?>
          </div>
       <div id="pie“>Desarrollo web en entorno servidor</div>
      </div>
   </body>
</html>
```

# Estructura de una página PHP




El fichero PHP en un navegador

# Estructura de una página PHP

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset= "UTF-8" />
      <link rel="stylesheet" type="text/css" href="estilo.css" />
      <title>Ejemplo de PHP</title>
   </head>
   <body>
      <div id="contenido">
      <div id="cabecera"><h2>Ejemplos con PHP</h2></div>
      <div id="principal">

         <h2>Bienvenido/a</h2>
         Hola, me llamo Alberto y estudio en el instituto Maria Ana Sanz
          </div>
     <div id="pie“>Desarrollo web en entorno servidor</div>
     </div>
   </body>
</html>
```

Código HTML que recibe y visualiza el navegador
*Ver / Código fuente*

# Instrucciones y comentarios

- Las instrucciones en PHP terminan en ;
  - En la última instrucción puede omitirse el ;
- PHP ignora los espacios en blanco en las sentencias
- Los comentarios aumentan la legibilidad del programa
  - El intérprete PHP los ignora
  - // comentario de una línea
  - # comentario de una línea
  - /* */ Comentario de varias líneas

# Instrucciones y comentarios

```php
<?php
        /*
         * Este bloque de código PHP calcula y
         * muestra la nota media de una alumna
         */
        $nombre = "Elena"; // asignar el nombre de la alumna
        $nota1 = 6; // asignar cada una de sus notas
        $nota2 = 7.5;
        $media = ($nota1 + $nota2) / 2; //calcular la media
        echo "Alumno/a: ".$nombre."<br />Nota media: ".number_format($media, 2);

  ?>
```

ejemplo02.php

# Instrucciones y comentarios

## Ejemplos con PHP

Alumno/a: Elena
Nota media: 6.75

# Sentencias echo y print

- Muestran información en el navegador
- **echo**
  - Permite visualizar una o más cadenas (en general expresiones)
  - echo “ Primer ejemplo ”;
  - echo “ Primer ejemplo ”, “ Segundo ejemplo ”;
  - echo '<h3>Probando sentencia echo</h3>';
- Escapar caracteres dentro de una cadena (importante)
  - echo '<h3>Probando sentencia \'echo\'</h3>';
  - echo “ Leyendo mejor el código fuente\n en varias líneas gracias a \\n ”;

# Sentencias echo y print

- **print**
  - Similar a echo, pero solo permite mostrar una expresión
  - print '<h4>Probando ahora con print</h4>';
  - print " Ahora un <span class=\"tex\">texto  con estilo ";

# Tipos de datos

- **Tipo de datos** – conjunto de valores que una variable puede tomar
- PHP débilmente tipado. El tipo de una variable se determina por el valor que se le asigna
- Tipos de datos en PHP

| | | |
|---|---|---|
| **integer** | 15 -21 | |
| **double** | 32.98 5.0 | |
| **boolean** | true false | Case insensitive |
| **string** | “Probando cadenas”<br>'Ejemplo con comillas simples'<br>'' | Cadena vacía |
| **array** | | |
| **object** | | |

# Variables

- **Variable** – contenedor en memoria que guarda un valor de un determinado tipo. Se identifican con un nombre
- Su valor puede cambiar a lo largo del script PHP y tomar valores de diferentes tipos
- Las variables en PHP
  - No se declaran
  - Empiezan con $
    - El siguiente carácter es letra o _
    - El resto de caracteres en el nombre son letras, dígitos y/o _
    - No palabras reservadas de PHP
    - **$nombre  $numero1  $cantidad_total**
  - Son case sensitive - **$total** es diferente de **$Total**

# Variables. Convenciones de nombres.

- Nombres descriptivos (¡IMPORTANTE¡)
- Notación camelCase
  - $totalArticulos $haAprobado $articulosEnCarrito
- Otras notaciones
  - $total_articulos $articulos_en_carrito

# Variables. Operador de asignación

- Operador de asignación **=**
  - Permite asignar un valor a una variable
    - Cuando se asigna un valor a una variable se considera en ese momento ya declarada (existe la variable)
  - $suma = 0;  // literal integer
  - $notaMedia = 7.8;  // literal double
  - $cadena = “Mensaje de texto”;  // string con comillas dobles
  - $cadena2 = 'Ahora con comillas simples';
  - $esValido = false; // literal de valor booleano
  - $mensaje = $cadena;

# Constantes

- Se definen utilizando la función **define o const**
- Asocian un valor a un nombre simbólico
  - define ('MAX', 100);  // constante integer
  - define ('PI', 3.14159265);  // constante double
  - define ('SEXO', 'm');
  - define (“IVA”, 18);
- Por convención nombres en mayúsculas
- Case sensitive

# Operadores aritméticos

- Para construir expresiones aritméticas (se evalúan a un valor numérico)

| Operador | Ejemplo | Resultado |
|---|---|---|
| + | 5 + 9 | 14 |
| - | 5 - 9 | -4 |
| * | 5 * 9 | 45 |
| / | 13 / 4 | 3.25 |
| % | 13 % 4 | 1 |
| ++ | $contador++ | incrementar en 1 el contador |
| -- | $contador-- | decrementar en 1 el contador |
| += -= *= /= %= | $suma += 10;<br><br>$producto *= 10; | añade 10 al contenido de $suma<br>multiplica 10 al valor de producto |

# Operadores aritméticos

- Ejemplos

```
$x = 14;
$y = 8;
$resul = $x + $y;          // 22
$resul = $x - $y;          // 6
$resul = $x * $y;          // 112
$resul = $x / $y;          // 1.75
$resul = $x % $y;          // 6
$x++;                      // 15
$y--;                      // 7
```

# Operadores aritméticos

- Precedencia de los operadores

| Orden | Operador | Resultado |
|---|---|---|
| **1** | ++ | izda a dcha |
| **2** | -- | izda a dcha |
| **3** | * / % | izda a dcha |
| **4** | + - | izda a dcha |

```
3 + 4 * 5          // 23
(3 + 4) * 5        // 35
```

# Strings

- Secuencia de caracteres entre comillas dobles o comillas simples
  - “Me llamo Juan y tengo 23 años”
  - 'Me llamo Luis y vivo en Pamplona'
- Las cadenas se pueden concatenar (unir) con el operador **.**

```
$nombre = "Luis";
$ciudad = "Pamplona";
$resul =  "<p>Me llamo ".$nombre." y vivo en ".$ciudad."</p>";
```

# Strings

- Escapar caracteres dentro de las comillas con \
  - “Cursillo de \”PHP \” ”
  - 'Esto es un comilla simple \' '
- Incluir algunas secuencias de control  \n  \\
- Si la cadena va entre **comillas dobles** podemos:
  - incluir variables y su valor se sustituye, se expande (interpolación de variables)
  - las secuencias de control tienen efecto
- Con comillas simples no hay sustitución de variables ni tienen efecto las secuencias de control

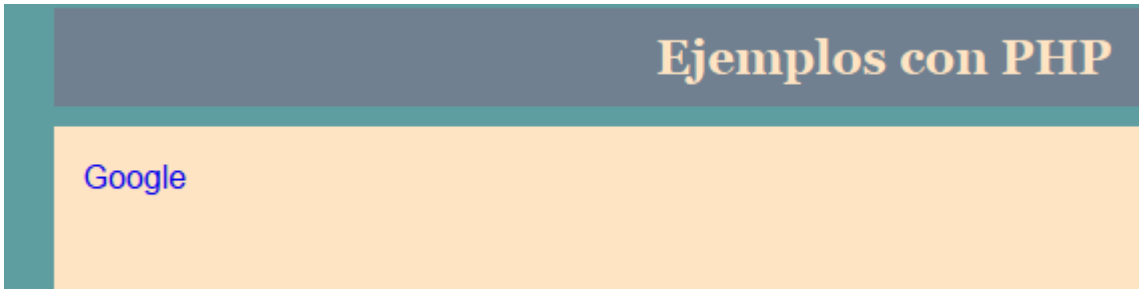
# Strings

- Ejemplos de sustitución de variables con string
  - $curso = “Desarrollo web en entorno servidor”;
  - echo “Estamos en clase de $curso”
- Concatenación de strings
  - $apellido = 'Arbeloa';
  - echo 'Pedro '.$apellido;
- Concatenando string y valores numéricos
  - $totalArticulos = 10;
  - $mensaje = “Su carrito tiene “.$totalArticulos;
  - echo $mensaje;
  - $mensaje .= “ artículos”;

# Strings

- Otros ejemplos con strings (ejemplo03.php)
  - generar un enlace

```
echo "<a href='http://www.google.es'>Google</a>";
```



# Algunas funciones de PHP

- rand(min,  max)
  - genera un valor aleatorio, en el ejemplo, entre min y max
  - $numero = rand(1, 50);
- min(valor1, valor2)  max(valor1, valor2)
  - calculan el valor mínimo (máximo) de dos valores
  - $maximo = max(100, 89);  // asigna el valor 100
- number_format($numero , $decimales)
  - $nf = number_format(12345, 2);  // 12,345.00

# Algunas funciones de PHP

- date($formato)
  - obtener la fecha actual

| Carácter | Descripción |
|---|---|
| **Y** | año de cuatro dígitos - 2011 |
| **y** | año de dos dígitos - 11 |
| **m** | Representación numérica del mes |
| **d** | Representación numérica del día del mes |

```
$fecha = date('d-m-Y');   // 12-06-2010
$fecha = date('d/m/y');   // 06/12/10
$fecha = date('m.d.Y');   // 06.12.2010
$fecha = date('Y');       // 2010
```

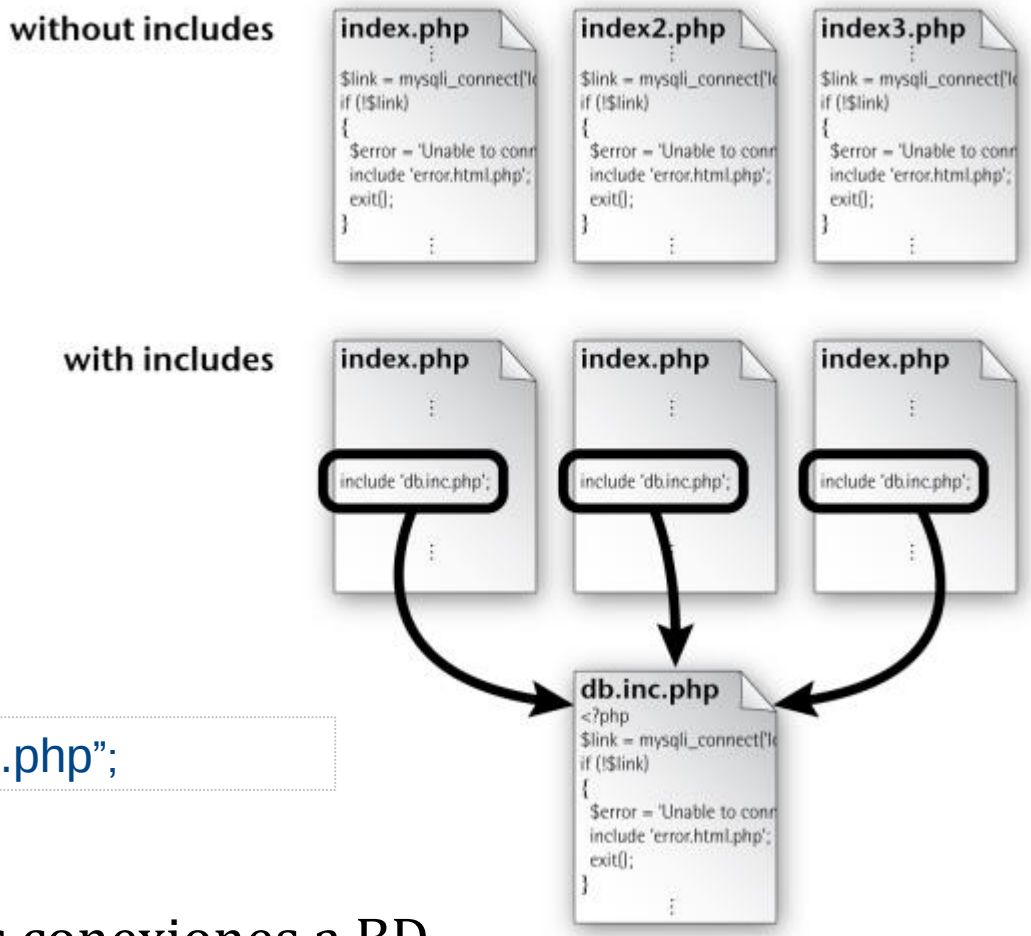
# Ejercicios

Ejercicios

# include / require

- Ficheros **includes** – contienen trozos de código que pueden incluirse en otro script php
- Habitualmente contienen: solo PHP, solo HTML o mezcla de PHP y HTML
- Evitan repetir código
- Permiten organizar adecuadamente el código
- El fichero include se busca a partir el directorio actual (a no ser que se especifique una ruta absoluta para el fichero)

# include / require




include “db.inc.php”;

Habitual en las conexiones a BD

# include / require

- 4 tipos de includes

| | |
|---|---|
| **include** | Incluye el fichero especificado. Si no se encuentra se genera un mensaje de advertencia pero el script continúa ejecutándose |
| **require** | Incluye el fichero especificado. Si no se encuentra se genera una mensaje de error fatal y el script se para |
| **include_once** | Funciona como include pero si el fichero ya ha sido incluido al menos una vez no volverá a incluirse |
| **require_once** | Funciona como require pero si el fichero ya ha sido incluido al menos una vez no volverá a incluirse |

# include / require

- Ejemplos
  - include 'index.php';
  - require_once 'funciones.php';
  - include 'vistas/resultado.php';
  - include 'includes/cabecera.php';
  - include '../error.php'; //subir un directorio para buscar el fichero
  - include_once 'conexion.php';
  - include_once $_SERVER['DOCUMENT_ROOT'].
    '/ejercicios/ejercicio12/includes/conexion.php';

# include / require

```php
<?php
        include "cabecera.php";
        echo "Estamos probando la sentencia include";
    include "pie.php";
?>
```

**ejemplo04.php**

# include / require

```
<!DOCTYPE html>
<html >
   <head>
      <meta charset= "UTF-8" />
      <link rel="stylesheet" type="text/css" href="estilo.css" />
      <title>Primeros ejemplos PHP</title>
   </head>
   <body>
     <div id="contenido">
      <div id="cabecera"><h2>Ejemplos con PHP</h2></div>
      <div id="principal">
```

**cabecera.php**

```
    </div>
       <div id="pie">Desarrollo en entorno servidor</div>
       </div>
    </body>
</html>
```

**pie.php**

# include / require

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset= "UTF-8" />
        <link rel="stylesheet" type="text/css" href="estilo.css" />
        <title>Primeros ejercicios PHP</title>
    </head>
    <body>
       <div id="contenido">
        <div id="cabecera"><h2>Ejemplos con PHP</h2></div>
        <div id="principal">
                Estamos probando la sentencia include </div>
         </div>
       <div id="pie">Desarrollo de aplicaciones  web</div>
       </div>
    </body>
</html>
```

**Código fuente que recibe el navegador**

# Más contenidos

- Operadores relacionales y lógicos
- Estructuras de control
  - Condicional (if / switch)
  - Iterativas (for / while)
- Testeando variables
- Organizando código con plantillas
- Funciones
- Arrays
- Más sobre strings

# Operadores relacionales y lógicos

- Permiten construir expresiones boolenas (condiciones)
- Su aplicación devuelve un valor true/false

| Operadores relacionales | Ejemplo |
|---|---|
| == | $nombre == 'Ana' |
| != | $apellido != 'Ruiz' |
| < | $edad < 18 |
| <= | $edad <= 18 |
| > | $cantidad > 0 |
| >= | $numeroArticulos >= 3 |
| === (identidad) | $valor1 === $valor2 ( *$valor1=0 y $valor2="0")*<br>Devuelve true si coinciden en valor y tipo |
| !== | $valor1 !== $valor2<br>Devuelve true si no coinciden en valor y tipo |

# Operadores relacionales y lógicos

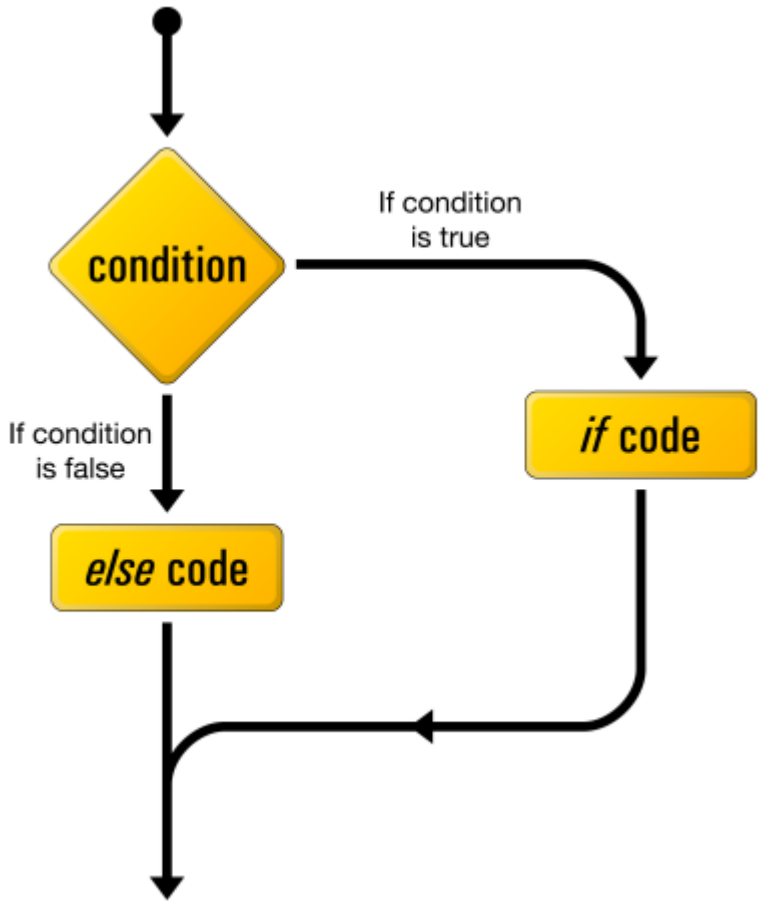
| Operadores lógicos (de menor a mayor prioridad) | Ejemplo |
|---|---|
| **!** | !is_numeric($edad) |
| **&&** | $edad >= 18 && $altura <= 1.78 (condición compuesta) |
| **\|\|** | !is_numeric($edad) \|\| !is_numeric($nota) (condición compuesta) |

- Qué es false en PHP
  - el valor booleano false
  - el valor entero 0
  - el valor float 0.0
  - la cadena vacía “”

- Qué es false en PHP
  - un array sin elementos
  - una variable unset

# Sentencias condicionales: if

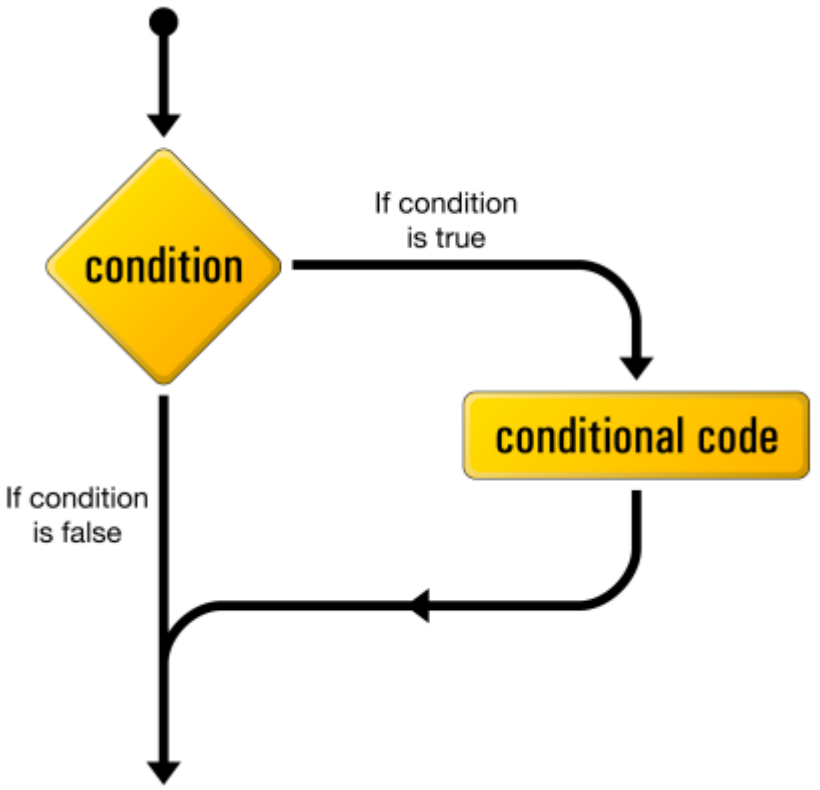
```
if ( expresión_booleana )
{
    sentencias a ejecutar
}
elseif ( expresión_booleana )
{
        sentencias a ejecutar
}
else
{
     sentencias a ejecutar
}
```




elseif puede ir separado expresión_booleana entre paréntesis

# Sentencias condicionales: if

```
if ( expresión_booleana )
{
    sentencias a ejecutar
}

sentencias que siguen a if
```



# Ejemplos if

```
if ( $precio <= 0 )
{
    $mensaje = 'El precio debe ser
                                mayor que 0.';
}
```

```
if ( empty($nombre) )
{
    $mensaje = 'Debe introducir un nombre.';
}
else
{
    $mensaje = 'Hola ' . $nombre.'!';
}
```

```
if ( empty($edad) )
 {
    $error = 'Introduzca edad';
}
else if ( !is_numeric($edad) )
{
   $error = 'Edad debe ser numérica';
}
else if ( $edad<= 0 )
{
            $error = 'Edad debe ser positiva';
}
```

# Sentencias condicionales: sintaxis alternativa de if

```
<?php
        if( condicion ) :
?>
        <!-- Codigo HTML -->
<?php endif;  ?>
```

```
<?php
        $ventas = 50;
        if ($ventas >=50):
    ?>
        <p>Ventas mayor que 50</p>
    <?php endif; ?>
```

# Testeando variables

- función **isset()** - está definida una variable? IMPORTANTE
  - if (isset($nombre)) // true si la variable $nombre está definida,
    // existe
- función **empty()** - está vacía una variable?
  - if (empty($nombre)) // true si la variable $
    //está vacía, “”, 0
- función **unset()** - destruye una variable
  - unset($nombre) // $nombre ya no existe

# Testeando variables

```
$nombre = "Juan";
if (isset($nombre))
{
    echo "La variable \$nombre existe<br />";
}
unset($nombre);
if (isset($nombre))
{
    echo "La variable \$nombre existe";
}
else
{
    echo "La variable \$nombre no existe ya";
                                        }
```

ejemplo05.php

# Sentencias condicionales: switch

```
switch( expresion )
{
        case valor1:
            // Instrucciones
               break;
        case valor2:
            // Instrucciones
               break;
          ……..……..

       default:
                // Instrucciones
}
```

- Se evalúa la expresión (a un tipo integer, float, string o boolean)
- Se ejecuta el case cuyo valor coincida con el de la expresión
- break sale del case
- sin break la ejecución continúa en el siguiente case
- default (en cualquier otro caso)
  - no break

# Ejemplo switch

```
$opcion = 1;
switch ($opcion)
{
    case 1:
        $mensaje = 'Producto insertado
                                        correctamente';
        break;
    case 2:
        $mensaje = 'Producto borrado correctamente';
        break;
    case 3:
        $mensaje = 'Producto modificado
                                    correctamente';
        break;

}
echo $mensaje;
```

# Trabajando sentencias condicionales if / switch

Ejercicios

# Qué es xdebug

- **Xdebug** - extensión para PHP que proporciona un soporte muy completo para la depuración de los scripts
- Podemos:
  - ejecutar las instrucciones paso a paso
  - inspeccionar las variables

# Instalando debugger - xdebug

http://www.xdebug.org/find-binary.php

http://xdebug.org/wizard.php

**XDEBUG EXTENSION FOR PHP | DOCUMENTATION | INSTALLATION**

home | updates | download/GIT | documentation | license | support | donate | issue tracker

## TAILORED INSTALLATION INSTRUCTIONS

This page helps you finding which file to download, and how to configure PHP to get Xdebug running. Please paste the **full** output of phpinfo() (either a copy & paste of the HTML version, the HTML source or `php -i` output) and submit the form to receive tailored download and installation instructions. Do **not** paste the raw HTML (from view-source) into the form.

```
_SERVER["REMOTE_ADDR"]  127.0.0.1
_SERVER["DOCUMENT_ROOT"]        C:/xampp/htdocs
_SERVER["SERVER_ADMIN"] postmaster@localhost
_SERVER["SCRIPT_FILENAME"]      C:/xampp/htdocs/xampp/phpinfo.php
_SERVER["REMOTE_PORT"]  49341
_SERVER["GATEWAY_INTERFACE"]    CGI/1.1
_SERVER["SERVER_PROTOCOL"]      HTTP/1.1
_SERVER["REQUEST_METHOD"]       GET
_SERVER["QUERY_STRING"] no value
_SERVER["REQUEST_URI"]  /xampp/phpinfo.php
_SERVER["SCRIPT_NAME"]  /xampp/phpinfo.php
_SERVER["PHP_SELF"]     /xampp/phpinfo.php
_SERVER["REQUEST_TIME"] 1309711284

PHP License
```

# Instalando debugger - xdebug

## INSTRUCTIONS

1. Download php_xdebug-2.1.1-5.3-vc6.dll
2. Move the downloaded file to C:\xampp\php\ext
3. Edit `C:\xampp\php\php.ini` and add the line

   ```
   zend_extension = C:\xampp\php\ext\php_xdebug-2.1.1-5.3-vc6.dll
   ```

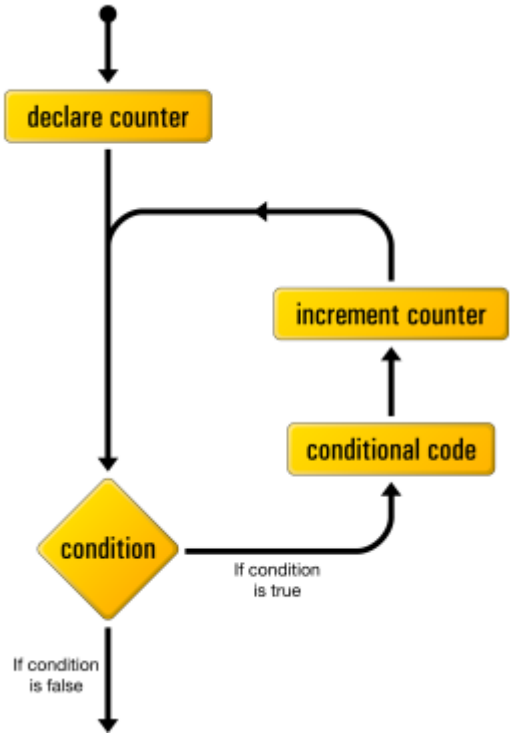
4. Restart the webserver

En C:\xammp\php\php.ini

```
zend_extension = C:\xampp\php\ext\php_xdebug-2.1.1-5.3-vc6.dll
xdebug.remote_enable=on
xdebug.remote_handler=dbgp
xdebug.remote_host=localhost
xdebug.remote_port=9000
```

# Sentencias repetitivas: for

```
for (declarar contador; condición; incrementar contador)
{
   // sentencias a ejecutar mientras condición true
}
```




*declarar contador* – se ejecuta una vez al principio del bucle
*condición* – se evalúa antes de ejecutar las instrucciones del bucle
*incrementar contador* -se ejecuta después de cada iteración

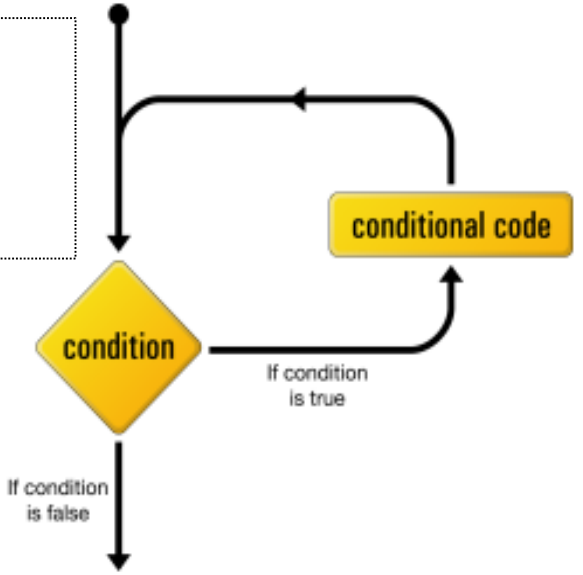
Nº repeticiones determinadas de antemano (10 veces, n veces, ...)

# Ejemplos for

```
for ($contador = 1; $contador <= 10; $contador++)
 {
    echo "<p>Saludo ".$contador."</p>";
 }

$mensaje = '';
 for ($contador = 1; $contador <= 5; $contador++)
 {
    $mensaje = $mensaje . $contador . '|';
 }
 echo "<p>$mensaje</p>";

$suma = 0;
 for ($numero = 1; $numero <= 5; $numero++)
 {
    $suma += $numero;
 }
```

# Sentencias repetitivas: while

```
while (condición)
{
        sentencias a ejecutar
}
```




*condición* – se evalúa antes de ejecutar las instrucciones del bucle.
*sentencias a ejecutar* – tiene que haber alguna que modifique el valor de la condición

Nº repeticiones determinadas o no (hasta que la condición deje de cumplirse)

# Ejemplos while

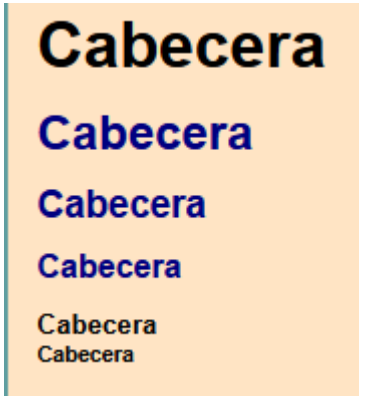
```
echo "<ul>";
$contador = 1;
while ($contador <= 10)
{
  echo "<li>".$contador."</li>";
  $contador ++;
}
 echo "</ul>";

echo "<table>";
$f = 1;
while ($f <= 7)
{
    echo "<tr><td>Fila $f</td></tr>";
    $f++;
}
echo "</table>";
```

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10

| |
|---|
| Fila 1 |
| Fila 2 |
| Fila 3 |
| Fila 4 |
| Fila 5 |
| Fila 6 |
| Fila 7 |

# Ejemplos while

```
$contador = 1;
while ($contador <= 6)
{
  echo "<h".$contador.">Cabecera</h".$contador.">";
  $contador ++;
}
```



# Trabajando sentencias repetitivas

Ejercicios

# Utilizando plantillas

- **Objetivo** - separar lo máximo posible el código HTML y el código PHP
- Con include hemos separado la cabecera y el pie de las páginas
- Podemos hacer algo similar con los resultados que se generan

```php
<?php
   $resultado = ""; // inicializamos la variable que contendrá el resultado a vacío
   $contador = 1;
   while ($contador <= 6)
   {
     $resultado .= $contador." | "; // añadimos cada número generado al resultado
     $contador ++;
   }

   include "vista_resultado.php";
 ?>
```

ejemplo07.php

# Utilizando plantillas

```
<?php
    include "cabecera.php";
 ?>
<div class='resultado'>
<?php
    echo $resultado;
   ?>
</div>
<?php
    include "pie.php";

 ?>
```

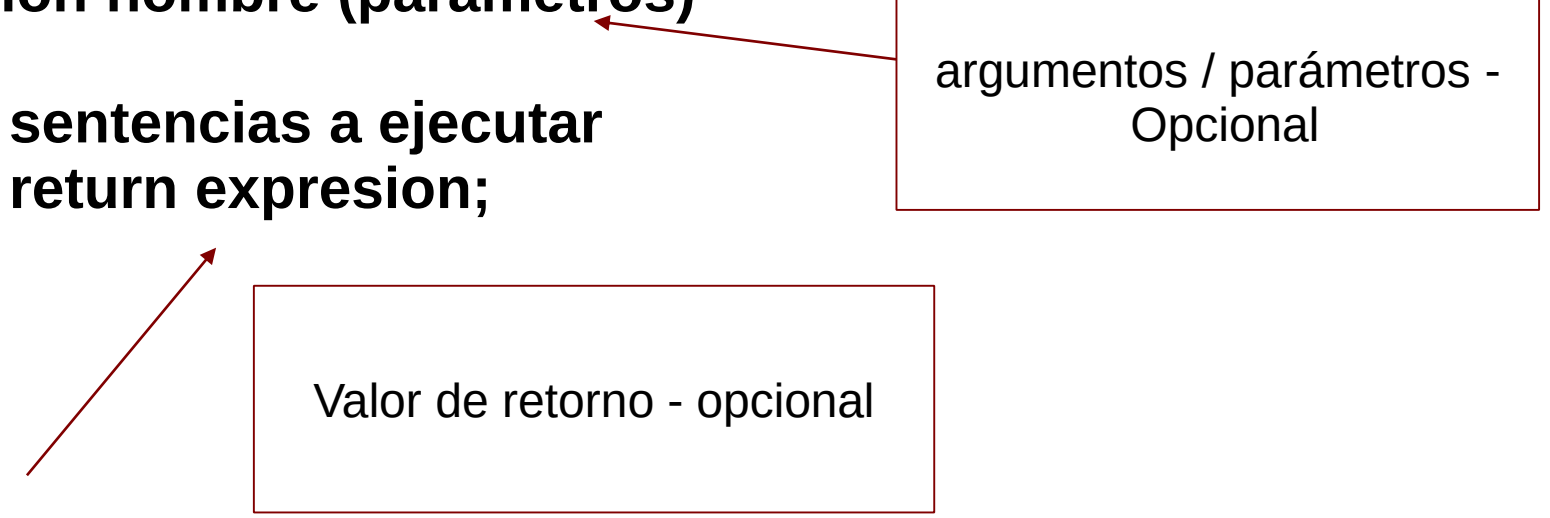
vista_resultado.php

# Funciones definidas por el usuario

- **Función** - conjunto de sentencias agrupadas bajo un nombre común y que realizan una tarea determinada
- Pueden ser invocadas tantas veces como queramos
- Opcionalmente incluyen argumentos
- Opcionalmente devuelven un resultado
- **Por qué utilizarlas?**
  - permiten dividir el programa en trozos más pequeños y manejables facilitando su legibilidad y mantenimiento
  - evitan repetir código

# Definición de una función

```
function nombre (parámetros)
 {
        sentencias a ejecutar
        return expresion;
}
```




- Definiremos las funciones al principio de los scripts, antes de que sean invocadas

# Definición de una función

```
/*
 * función que calcula el área de un cuadrado
 */
function calcularArea($lado)
{
    $area = $lado * $lado;

    return $area;
}

echo "<h3>Cálculo del área de un cuadrado</h3>";
$lado = 3;
$area = calcularArea($lado);
echo "<p>El &aacute;rea de un cuadrado de lado $lado es $area</p>";
```

argumento formal

valor de retorno

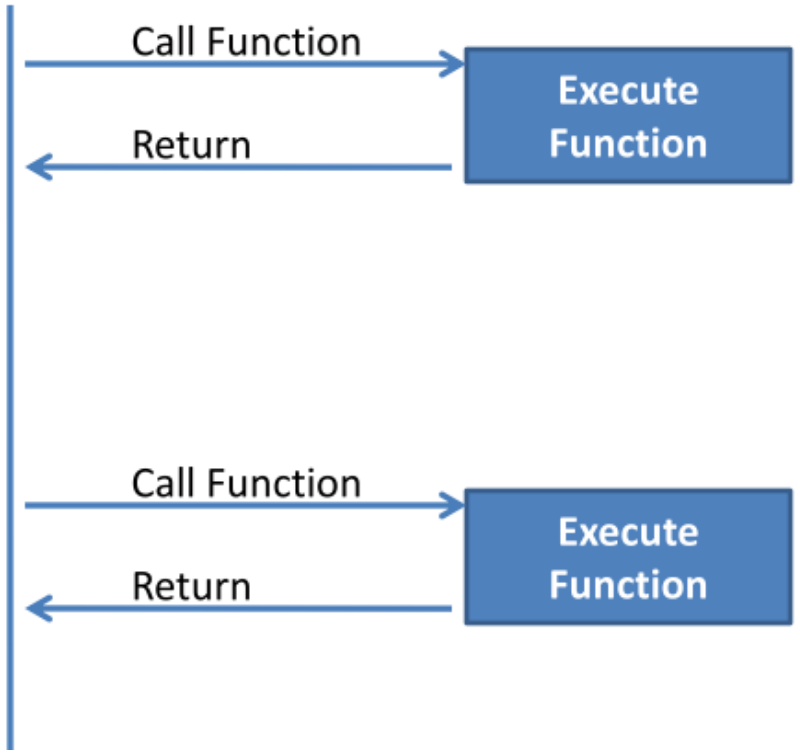
argumento actual

llamada a la función

**Cálculo del área de un cuadrado**

El área de un cuadrado de lado 3 es 9

# Flujo de ejecución de una función




- Se llama a la función a través de su nombre
- Se le pasan los argumentos actuales (si tiene)
- Se ejecuta el código de la función
- Se retorna al punto en que la función fue invocada
- Si hay sentencia return dentro de la función ésta devuelve el valor indicado

# Análisis de la función calcularArea()

- function calcularArea($lado)
  - Es la cabecera de definición de la función
  - nombre de la función: calcularArea
  - entre paréntesis los argumentos locales (si hay varios se separan por comas)
  - después va el código que realiza los cálculos
- return $area
  - cuando se ejecuta la función termina y se devuelve el control al script principal con el valor indicado en $area

# Parámetros en las funciones

- Una función puede incluir parámetros
  - En la llamada a la función hay que pasar tantos parámetros como indique la definición
  - se pasan en el orden indicado
  - cada parámetro actual se corresponde con un parámetro formal

```
function calcularMedia($num1, $num2, $num3)
{
    return ($num1 + $num2 + $num3) / 3;
}

    echo "<h3>La media de 4, 5, 6 es ".calcularMedia(4, 5, 6)."</h3>";
```

# Funciones sin parámetros y/o sin valor de retorno

- Una función puede tener parámetros y no devolver un valor

```
function mostrarMensaje($mensaje)
{
   echo “<p>$mensaje</p>”;
}

mostrarMensaje(“Ejemplo de función sin valor de retorno”);
```

- Una función puede no tener parámetros y no devolver un valor

```
function mostrarLineaHorizontal()
{
   echo "<hr />";
}

mostrarLineaHorizontal();
```

# Ámbito de las variables y funciones

- Ámbito de una variable
  - espacio dentro del script en que una variable es conocida y puede ser utilizada
- Variables locales y globales
  - Cualquier variable definida dentro de una función (incluyendo los argumentos formales) solo es conocida dentro la función
    - Su ámbito es **local**
  - Las variables definidas fuera de la función son conocidas en el script principal (incluyendo los ficheros include) aunque no dentro de las funciones. Para que éstas las utilicen hay que pasarlas como argumentos.
    - Su ámbito es **global**
- Una función puede no tener parámetros y no devolver un valor

# Más ejemplos de funciones

```
function ingresar($importe, $cantidad)
{
    // $importe y $cantidad son locales a la función
    // diferentes de las definidas en el script principal
    // aunque se llamen igual
     $importe += $cantidad;
     echo "<p>Nueva cantidad $importe</p>";
}

function ingresarV2($importe, $cantidad)
{
     $importe += $cantidad;
     return $importe;
}

$importe = 2000;
$cantidad = 300;
ingresar($importe, $cantidad);
echo "<p>Nueva cantidad = ".ingresarV2($importe, $cantidad + 200)."</p>";
```

# Funciones predefinidas de PHP

- Ya hemos visto algunas: min(), max(), date(), rand(), number_format()
- Otras funciones – sqrt() , pow(), strip_tags(), trim(), …

http://www.php.net/manual/es/book.math.php

# Trabajando con funciones

Ejercicios