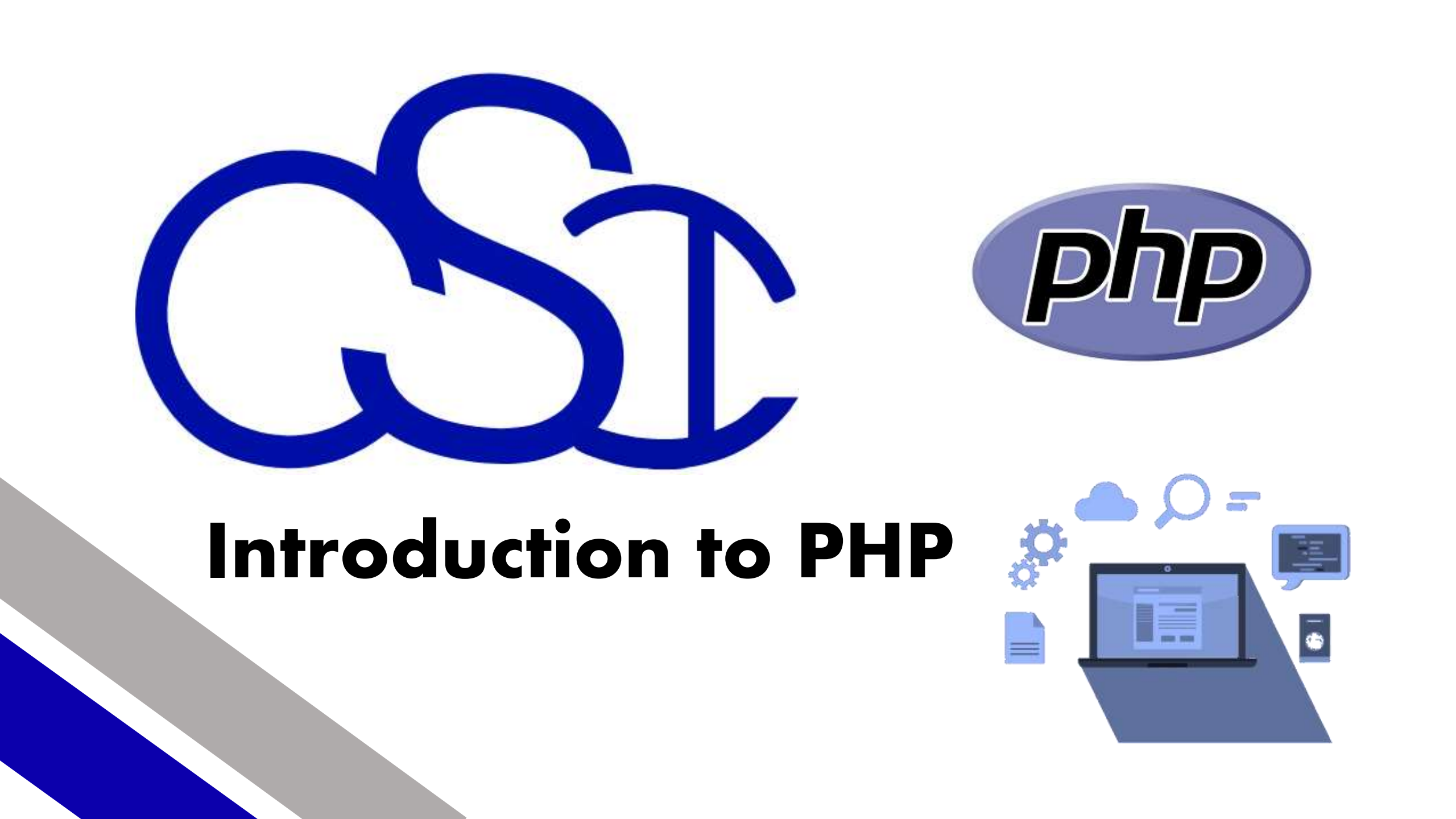
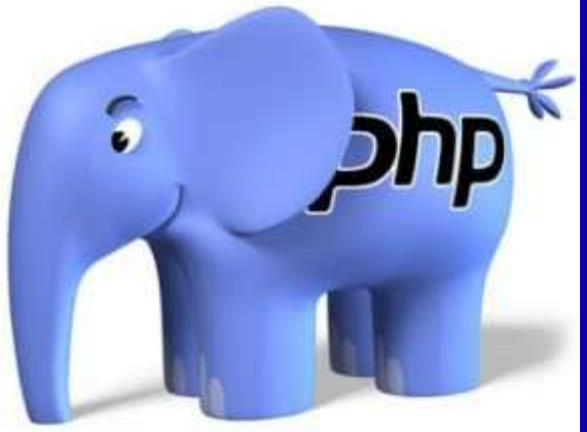


Introduction to PHP

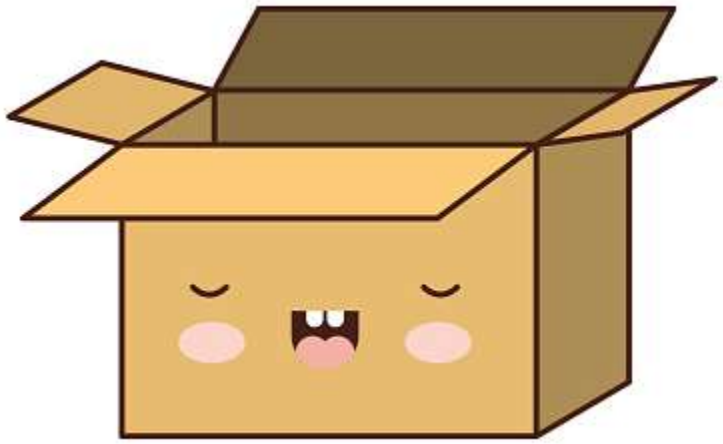
3. Variables y tipos de datos.

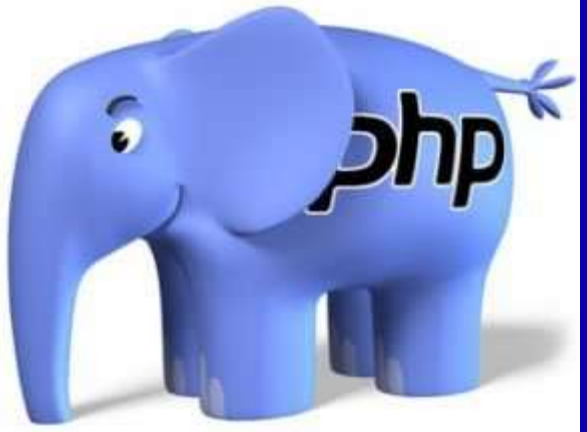
- ¿Qué es una variable?
- ¿Qué es una constante?
- Enteros
- Flotantes
- Constantes
- Booleanos
- Cadenas
- Variables variables
- Variables de formularios



¿Qué es una variable?

Las **variables** son espacios en la memoria temporal, dónde podemos guardar datos y accederlos a través de un nombre. También las podríamos pensar que son como cajas nombradas de la memoria.

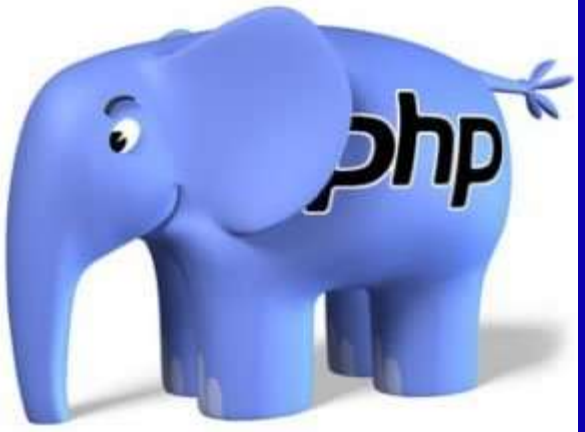




Ejercicio

```
a = 10
b = 20
c = 5
a = a + 3
b = b + 4 – a
c = a + b + c
a = a + c
b = 4
c = c + 3 - b + 2
```

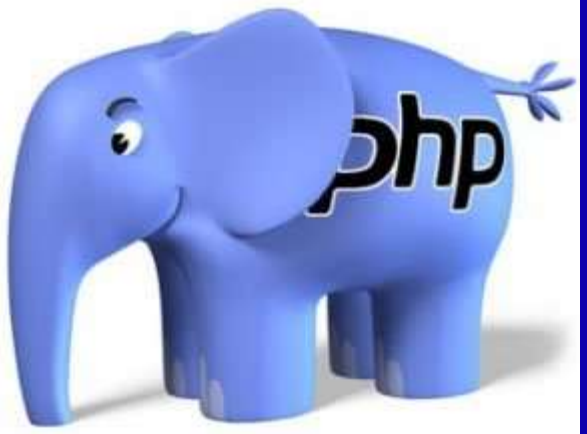
Qué valores quedan almacenados en las variables a, b y c ?



Ejercicio

```
a = 5
b = 18
c = 15
d = 25
a = a + 10
b = b + 5 – c
c = c + 4 + b
d = d + b + a
a = a + 1
b = b + c
c = b + c
d = b + b
```

Qué valores quedan almacenados en las variables a, b y c ?



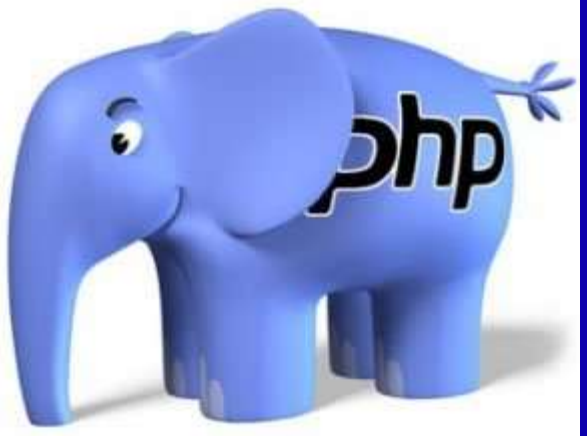
Variables en PHP

Es una espacio con nombre en memoria donde se almacena un valor de un cierto tipo de dato.

Todas la variables tienen un **identificador válido** que generalmente describe su propósito.

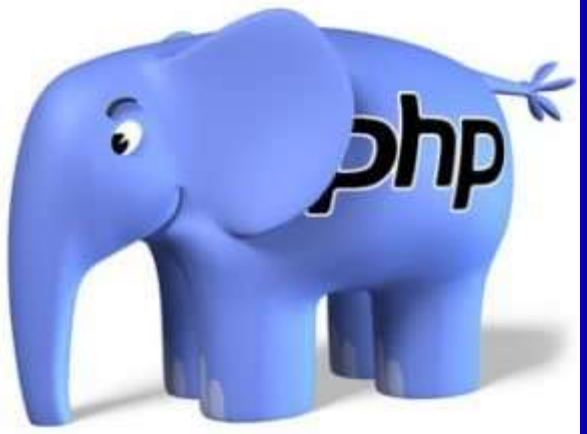
Ejemplos:

```
$x = 10;
$nombre = “Erick”;
$sinIva= 234.9;
```



Reglas de escritura de identificadores en PHP.

- Reglas básicas de escritura de los identificadores:
  - Iniciar con un carácter o guion bajo.
  - Constantes en mayúsculas
  - Funciones en letra mixta(mayúsculas/minúsculas)
  - Siempre van precedidos del signo $.
  - Los identificadores no pueden ser palabras reservadas



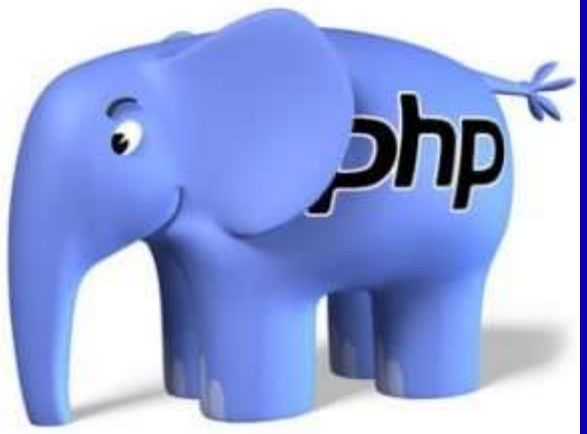
Declaración de una variable.

Es una sentencia que da información de la variable al interprete.

Formato: **$<variable> = <valor>**

<variable> es el identificador(nombre) valido en PHP siempre va precedido por un signo de dólar.

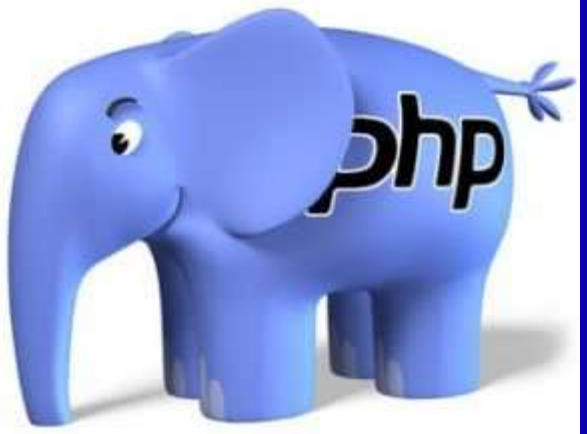
<valor> es un valor asociado a un tipo de dato que se le asignará a la variable, puede ser dado por el programador o podemos pedirlo al usuario.



Variables en PHP

Más ejemplos:

```
$numero = 345 ;
$letra= ‘A’;
$precio= 99.9;
$Var=‘Pepe’;
$var=‘Jorge’;
$dirNumero=&$numero;
$edades=[25,26,27,28,29];
```

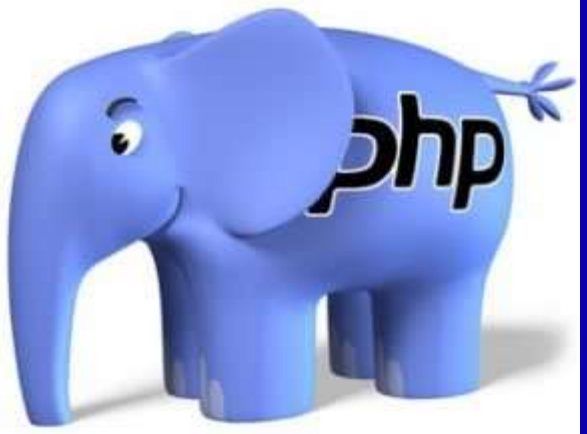


PHP es case sensitive

Quiere decir que detecta cuando algo es escrito con mayúsculas y cuando es escrito con minúsculas y lo toma como identidades distintas.

Ejemplos especiales:

$Var=‘Pepe’;
$var=‘Jorge’;

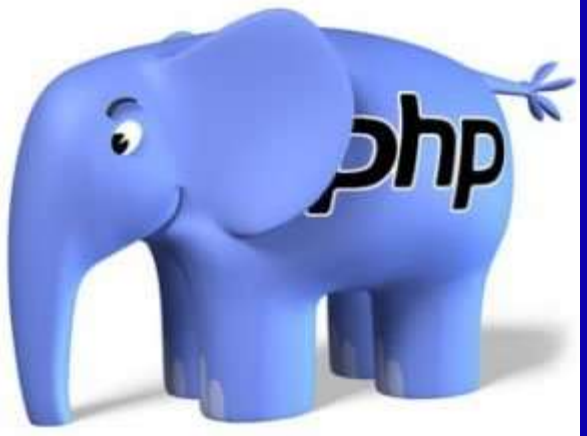


Inicialización de una variable.

Decimos inicialización de una variable al **valor inicial** de una variable, que puede ser expresión valida cuyo valor es del mismo tipo de tipo o en algunos casos diferente.

Ejemplo:

```
$a=3;
$a=$a+4;
$a=$a*2;
$a=$a*4;
```



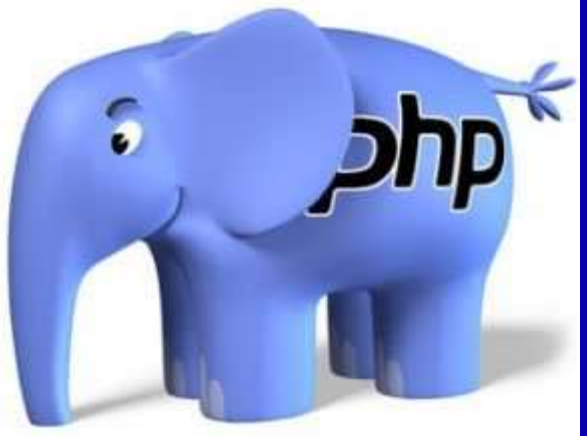
Duración de una variables.

Dependiendo del lugar donde se declare una variable, ésta puede ser temporal, ya sea dentro de un bloque o de una función.

En programación la zona en la que está activa una variable se llama ***ámbito*** *o* ***alcance(scope)***.

Clasificación de las variables según su alcance.

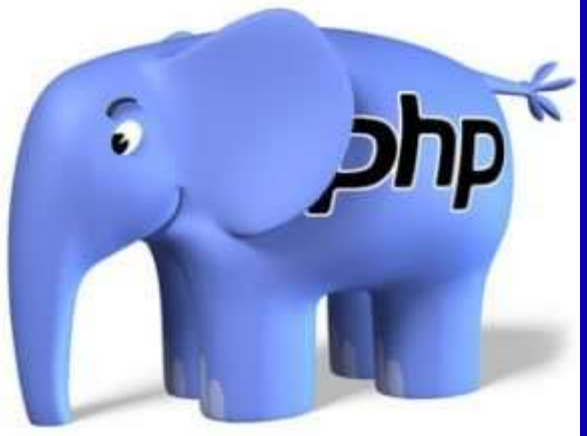
- Variables locales
- Variables globales



Variables locales

También llamadas automáticas. Son aquellas que se declaran dentro de una función y solo están activas dentro de tal función.

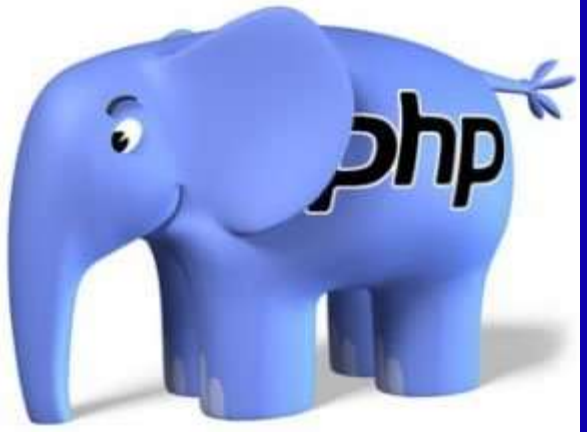
```
function test()
{
  $b=$a*3;
   echo $b; /* Variable del ámbito local */
}
```



Variables globales

Son aquellas que se declaran dentro de un archivo **php** pero fuera de una función y por lo tanto son visibles para cualquier función.

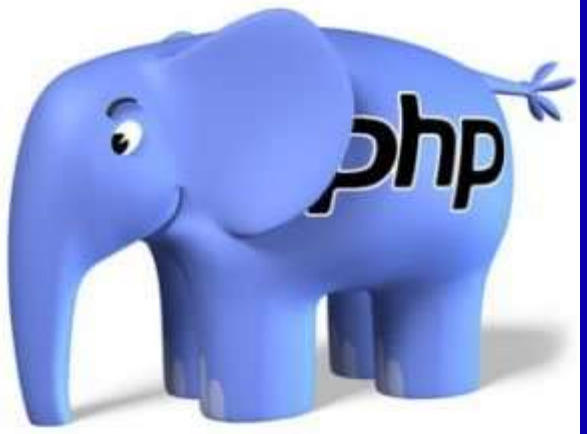
```
$a = 100; /* Variable de ámbito global  */

function test()
{
  $b=$a*3;
   echo $b; /* Variable del ámbito local */
}
```



Diferencias entre locales y globales.

Diferencias entre variables locales y globales.

- Locales, desaparecen cuando su bloque termina.
- Globales, es visible desde la sentencia donde se declara hasta el final del programa (archivo fuente).



Tipo de datos en PHP.

PHP admite diez tipos de datos.

Cuatro tipos escalares:

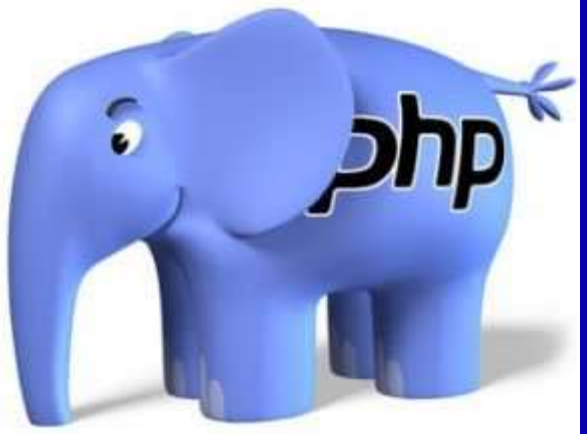
- boolean
- integer
- float
- string

Cuatro tipos compuestos:

- array
- object
- callable
- iterable

Dos tipos especiales:

- resource
- NULL

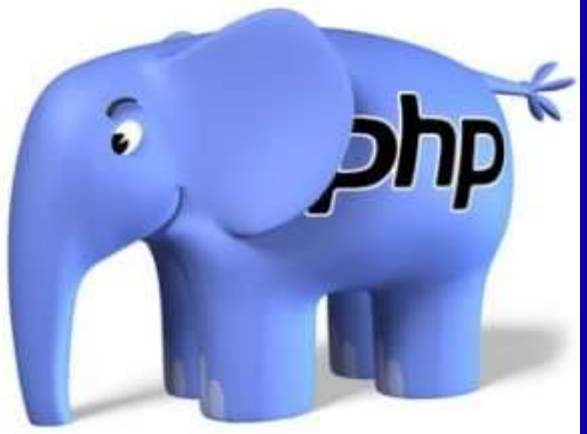


Números enteros.

Un número entero (o integer) es un número del conjunto $\mathbb{Z} = \{..., -2, -1, 0, 1, 2, ...\}$.

Los integer pueden especificarse mediante notación decimal (base 10), hexadecimal (base 16), octal (base 8) o binaria (base 2), opcionalmente precedidos por un signo (- o +).

```php
<?php
$a = 1234; // número decimal
$a = -123; // un número negativo
$a = 0123; // número octal (equivale a 83 decimal)
$a = 0x1A; // número hexadecimal (equivale a 26 decimal)
$a = 0b11111111; // número binario (equivale al 255 decimal)
?>
```



¿Cómo funciona el sistema binario?

78 → **1001110**

78/2=39 resto 0

39/2=19 resto 1

19/2=9 resto 1

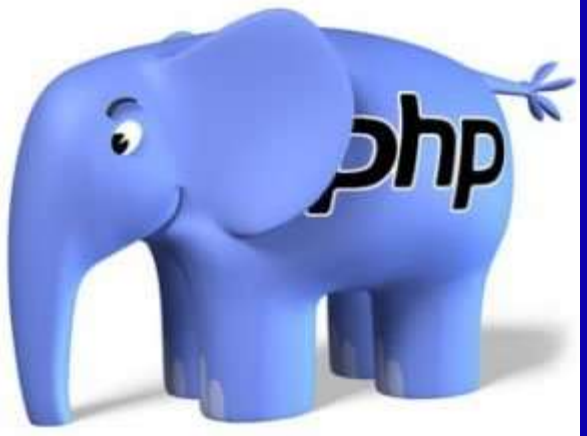
9/2=4 resto 1

4/2=2 resto 0

2/2=1 resto 0

1/2=0 resto 1

N

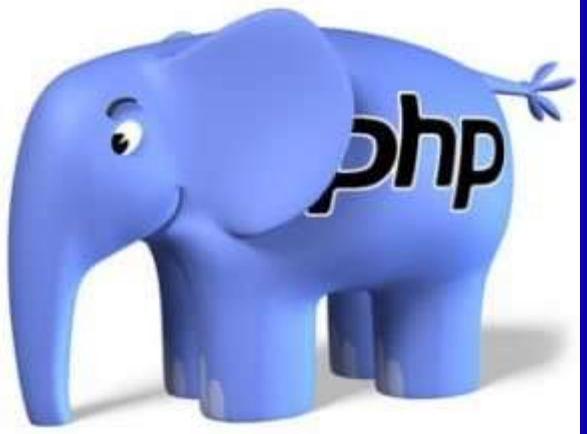


Entradas y salidas.

El usuario interactúa con un programa a través de datos de entrada y salida. Dentro de la terminal tenemos algunas opciones que nos permiten hacer pruebas.

Tipos de instrucciones

- Instrucciones de ***entrada/salida***
- Instrucciones de ***cálculo***
- Instrucciones de ***control***

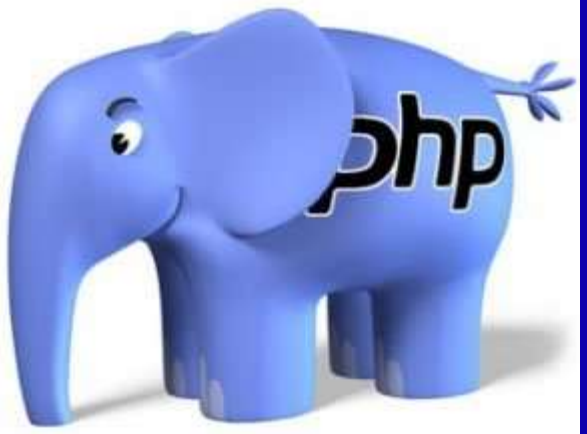


Entradas.

readline() es una función que nos permite leer una línea del usuario por medio del teclado. Esta línea es guardada como un tipo de dato string así que debemos transformarla a nuestra conveniencia.

Como parámetro podemos lanzar un mensaje que le permita al usuario seguir instrucciones para que introduzca un dato.

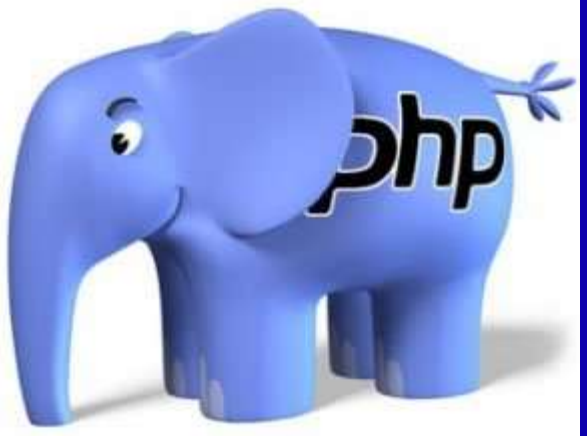
```
<?php //Vamos a la práctica
    $x = (int)readline(“Escribe un numero entero: ”);
?>
```



Manipulación de tipos.

PHP no requiere (ni soporta) la definición explicita de tipos en la declaración de variables; el tipo de la variable se determina por el contexto en el cual se emplea la variable.

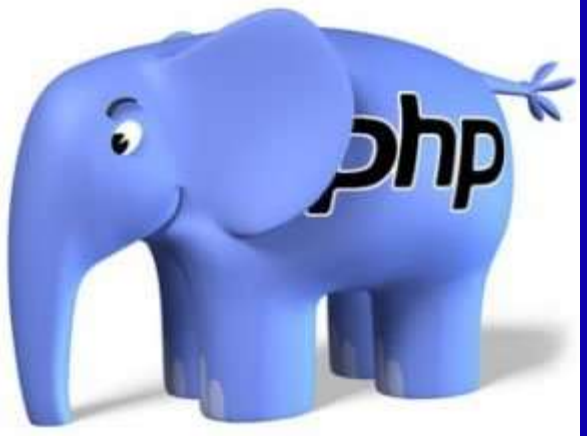
```
<?php
    $x = 11; /*PHP sabe que es un entero*/
?>
```



Forzado de tipos.

El forzado de tipos en PHP funciona de la misma manera que en C:, donde el nombre del tipo deseado se escribe entre paréntesis antes de la variable que se quiera forzar.

Los siguientes forzados de tipos están permitidos:
(int), (integer) - forzado a entero
(bool), (boolean) - forzado a booleano
(float), (double), (real) - forzado a floatante
(string) - forzado a string (texto, cadena)
(array) - forzado a arreglo
(object) - forzado a objecto
(unset) - forzado a NULL (PHP 5)

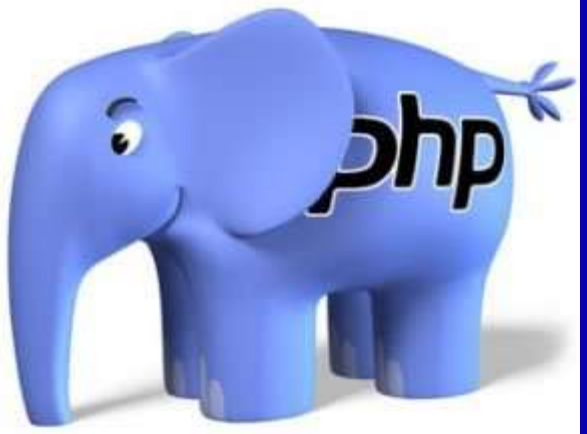


Forzado de tipos.

Las tabulaciones y espacios dentro de los paréntesis

```
<?php
   $numero = 56.34;
   $parteEnt = (int)$numero;
?>
```

```
<?php
   $numero = 56.34;
   $parteEnt = (    int    )$numero;
?>
```

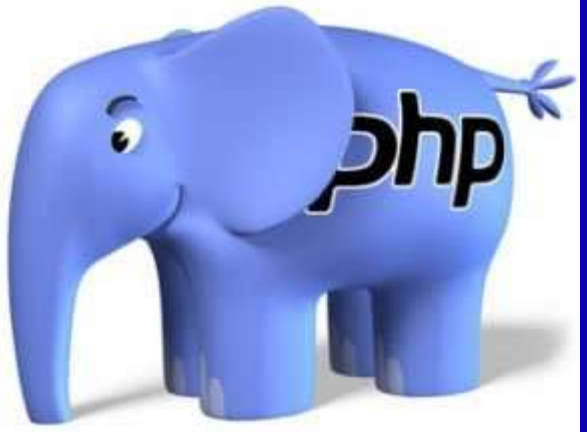


Salidas.

echo muestra una o más cadenas, en realidad echo no es una función como **readline()**, sino más bien una construcción del lenguaje.

No se requiere el uso del paréntesis.

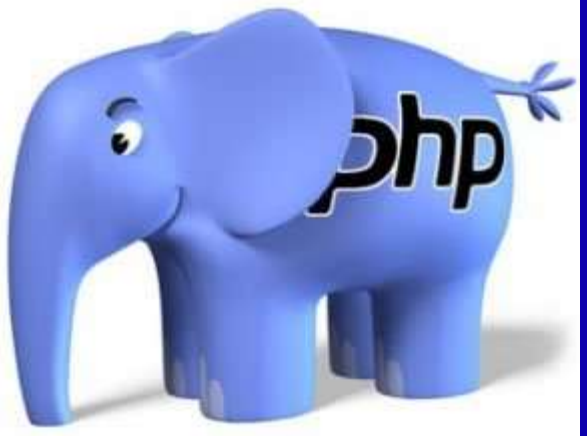
```
<?php //Vamos a la práctica
    echo “Bienvenidos a PHP en el 2022”;
?>
```



Ejercicio:

1. **Hacer un programa que compruebe los valores de a, b y c del primer ejercicio de variables.**

```
a = 10
b = 20
c = 5
a = a + 3
b = b + 4 – a
c = a + b + c
a = a + c
b = 4
c = c + 3 - b + 2
```

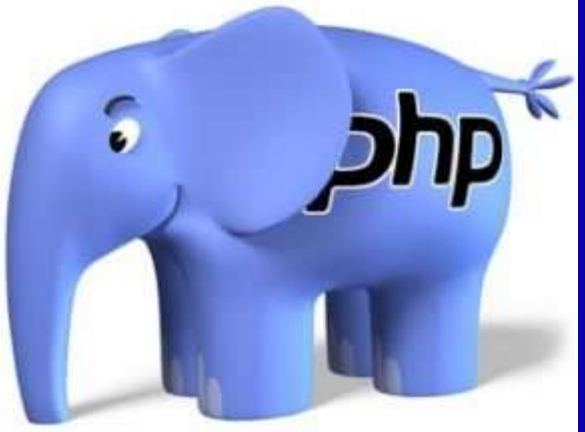


Secuencias de escape.

PHP también utiliza ***secuencias de escape*** para visualizar caracteres que no están representados por símbolos tradicionales

```
<?php //Vamos a la práctica
    echo “\n\t Bienvenidos a PHP en el 2022”;
?>
```

Código de escape	Significado	ASCII decimal	ASCII hexadecimal
‘\n’	Nueva línea	10 o 13	0D o 0A
‘\t’	Tabulación	9	09
‘\v’	Tabulación vertical	11	0B

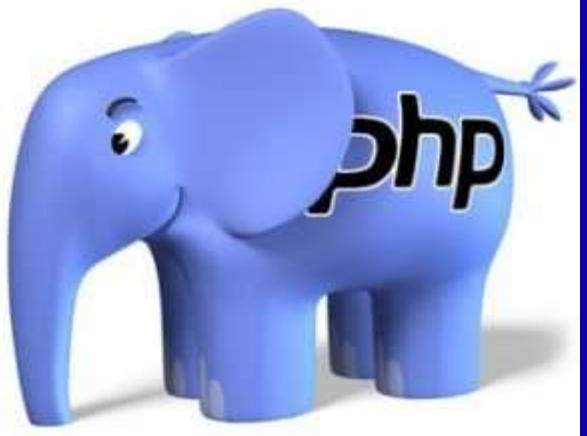


Ejercicios:

1. **Hacer un programa que pida dos números y nos devuelva el doble de la suma de ellos.**

2. **Hacer un programa que haga los siguientes cálculos y al final muestre en pantalla los valores de a, b y c.**

a = 5
b = 2
c = -4
a = a + 3
b = b + 4 – a
c = a + b + c
a = a + c
b = 4
c = c + 3 - b + 2

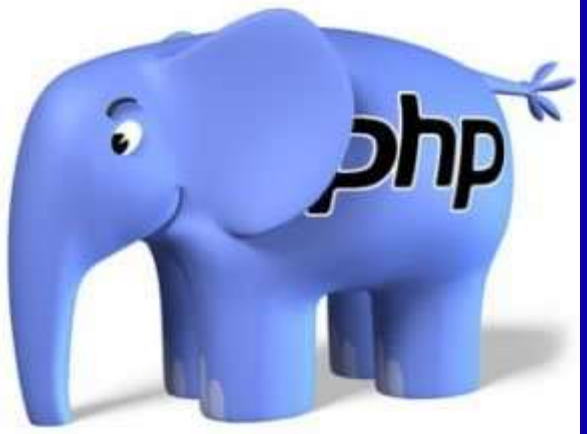


Números flotantes.

Los números de punto flotante (también conocidos como "de coma flotante") pueden ser especificados usando cualquiera de las siguientes sintaxis:

```
<?php
$a = 1.234;
$b = -0.7625;
$c = 10.524;
?>
```

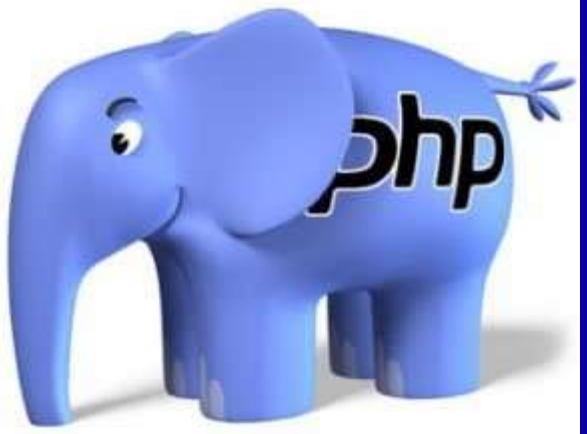
El tamaño de un 'float' depende de la plataforma, aunque un valor común consiste en un máximo de aproximadamente 1.8e308 con una precisión cercana a los 14 dígitos decimales (el formato de 64 bit del IEEE).



Información de una variable.

var_dump() Esta función muestra información estructurada sobre una o más expresiones incluyendo su tipo y valor. En el caso de los arreglos y objetos muestra su estructura interna con sangrado.

```
<?php
    $a = 1.234;
    var_dump($a);
?>
```

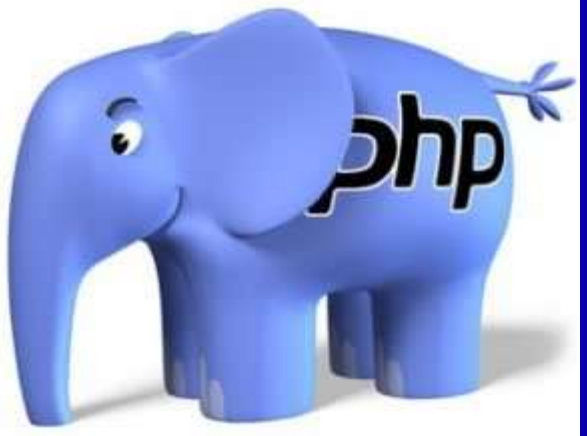


Ejercicios:

1. Hacer un programa que pida una cantidad sin I.V.A. y nos imprima en consola la cantidad con el I.V.A. agregado.

2. Hacer un programa que pida un monto a cobrar y un porcentaje de descuento, entonces se calcula el monto con el descuento aplicado y se imprime en consola.

3. Hacer un programa que pida un número flotante y se imprima en consola la parte entera del número y la parte decimal por separado.

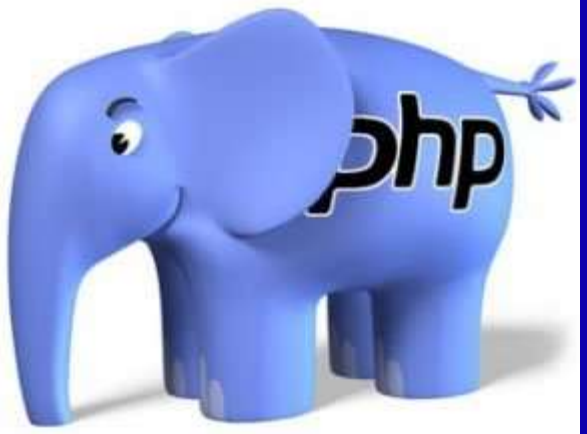


Constantes.

A diferencia de las variables son valores que guardamos en memoria pero una vez declarados e inicializados ya no es posible modificar su valor desde otra sentencia.

```
<?php
    define(“PI”, 3.1415929,true);
?>
```

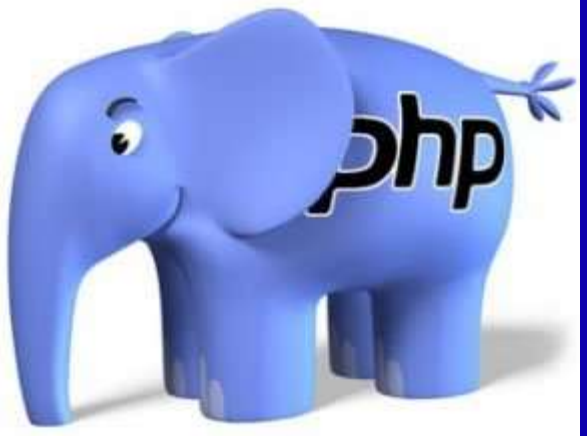
El tercer parámetro es llamado **case_insensitive** que quiere decir que la constante será insensible a mayusculas o minusculas, el valor por default es **false**.



Ejercicios:

1. Hacer un programa que pida el radio de un circulo y calcule su área y su perímetro. Usar constantes.

2. Hacer un programa que fijada una tarifa de 0.987 pesos por kWh pida al usuario su consumo de kWh y entregue al usuario el monto a pagar por su consumo de electricidad.

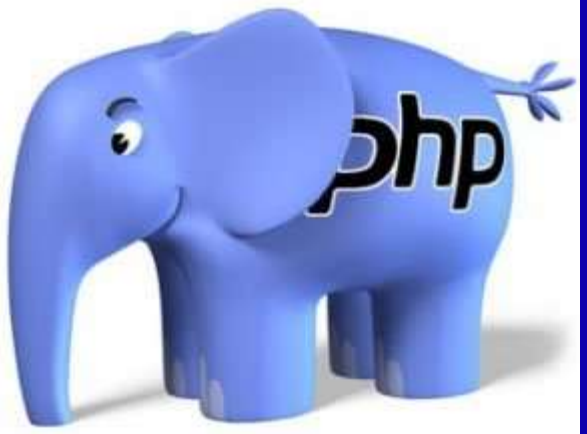


Booleanos.

Este es el tipo más simple. Un boolean expresa un valor que indica verdad. Puede ser true (verdadero) o false (falso).

```
<?php
    $centinela =True;
?>
```

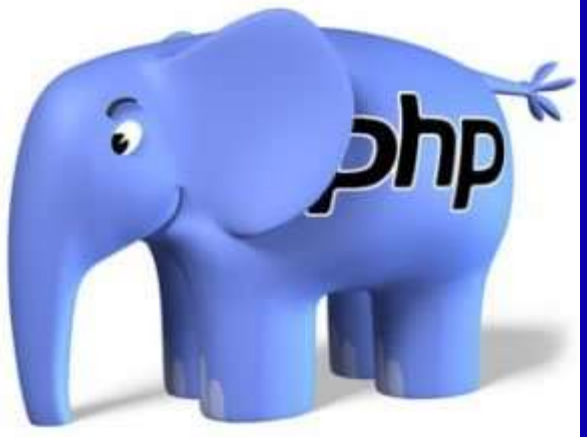
Para especificar un literal de tipo **boolean** se emplean las constantes true o false.



Ejercicios:

1. Hacer un programa que haga un forzado de tipo booleano a tipo entero. Probar en consola con los valores true y false.

2. Forzar un tipo de dato flotante a booleano. Probar varios valores.



Cadenas de caracteres.

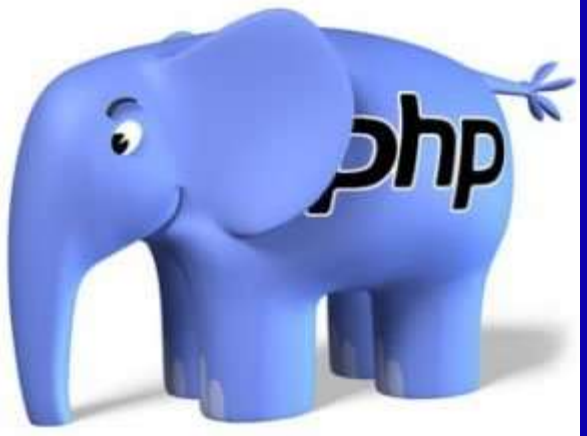
Un string, o cadena, es una serie de caracteres donde cada carácter es lo mismo que un byte.

Los caracteres que forman un string pertenecen al **ASCII**, esto significa que PHP solo admite un conjunto de 256 caracteres.

Sintaxis

- Entrecomillado simple.
- Entrecomillado doble.
- Sintaxis heredoc.
- Sintaxis nowdoc.

Hagamos un archivo para ejemplos usando Apache…

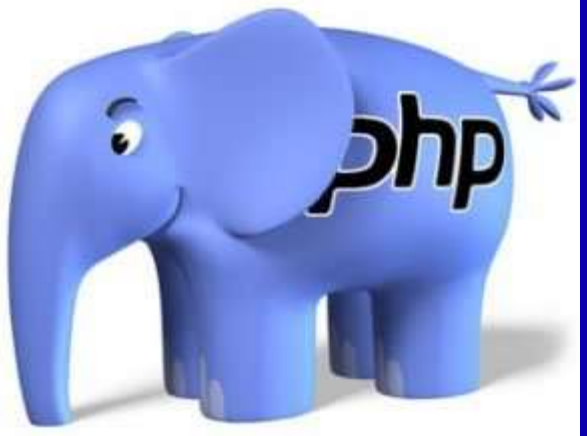


Entrecomillado simple.

La manera más sencilla de especificar un string es delimitarlo con comillas simples (el carácter ').

```
<?php
echo 'Esto es una cadena sencilla';

echo 'También se pueden incluir nuevas líneas en
un string de esta forma, ya que es
correcto hacerlo así';

// Resultado: Arnold una vez dijo: "I'll be back"
echo 'Arnold una vez dijo: "I\'ll be back"';
?>
```

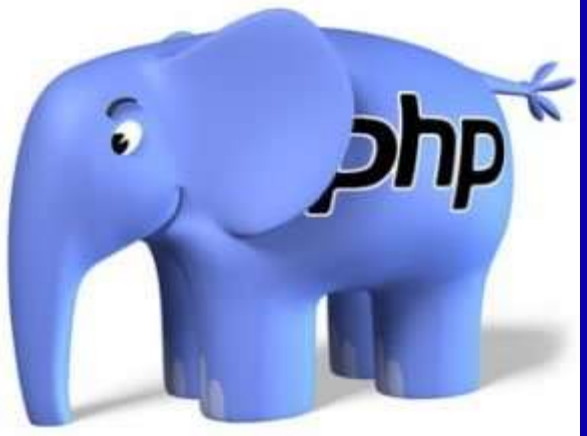
Bajo esta sintaxis los caracteres escapados como \n o \t se verán tal cual y no tendrán efecto.



Entrecomillado doble.

Si un string está delimitado con comillas dobles ("), PHP interpretará las siguientes secuencias de escape como caracteres especiales:

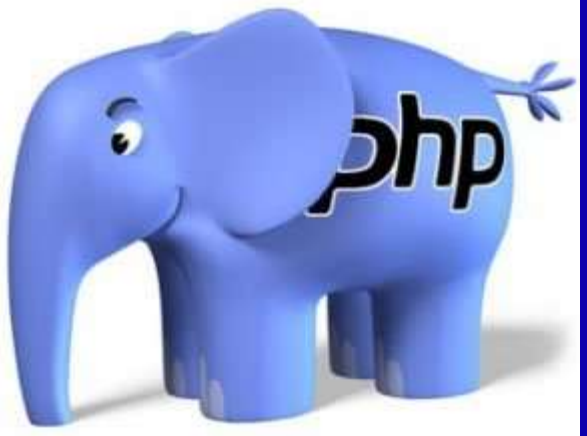
Caracteres escapados	
Secuencia	**Significado**
\n	avance de línea (LF o 0x0A (10) en ASCII)
\r	retorno de carro (CR o 0x0D (13) en ASCII)
\t	tabulador horizontal (HT o 0x09 (9) en ASCII)
\v	tabulador vertical (VT o 0x0B (11) en ASCII) (desde PHP 5.2.5)
\e	escape (ESC o 0x1B (27) en ASCII) (desde PHP 5.4.4)
\f	avance de página (FF o 0x0C (12) en ASCII) (desde PHP 5.2.5)
\\	barra invertida
\$	signo de dólar
\"	comillas dobles



Sintaxis heredoc.

Una tercera forma de delimitar un string es mediante la sintaxis heredoc: <<<. Después de este operador, se deberá proporcionar un identificador y justo después una nueva línea.

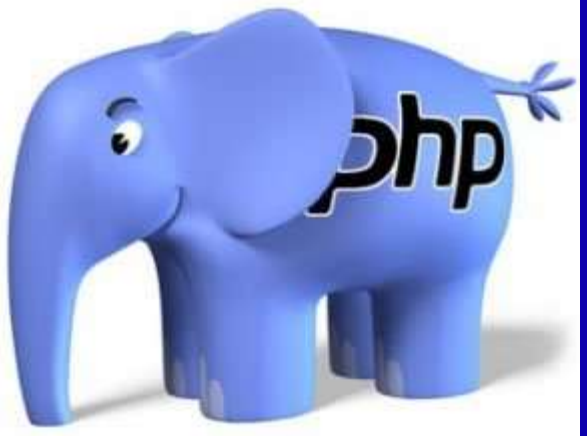
```
<?php
    $texto = <<<MARC
            Ejemplo de una cadena
            expandida en varias líneas
            empleando la sintaxis heredoc.
            MARC;
?>
```



Sintaxis nowdoc.

Nowdoc es a los string con comillas simples lo mismo que Heredoc lo es a los string con comillas dobles.

```
<?php
    $texto = <<<'EOD'
            Ejemplo de una cadena
            expandida en varias líneas
            empleando la sintaxis nowdoc.
            EOD;
?>
```



Funciones de cadenas.

Existen un conjunto de funciones que se pueden aplicar a las cadenas de caracteres, veamos algunas. Usemos el preformateado HTML **<pre>**.

strtolower() convierte un string a minúsculas.

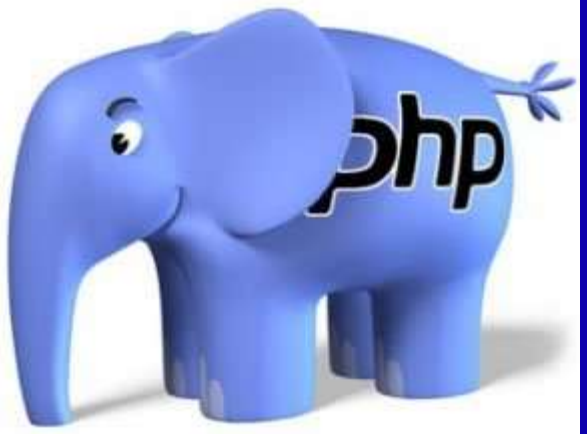
strtoupper() convierte un string a mayúsculas.

trim() , ltrim(), rtrim() Eliminan espacios en blanco.

substr() substrae un conjunto de caracteres de un string.

str_replace() reemplaza un carácter especificado por otro en una cadena.

strip_tags() quita etiquetas HTML y PHP de un string.

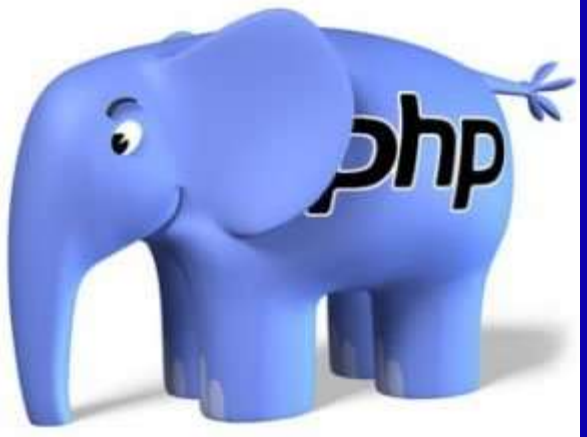


Ejercicios:

1. Hacer un programa que pida un número de tarjeta y rellene de asteriscos los espacios en blanco.

2. Hacer un programa que pida una frase en minúsculas y mandarla a imprimir en mayúsculas . (Usar acentos y ver que pasa en el navegador).

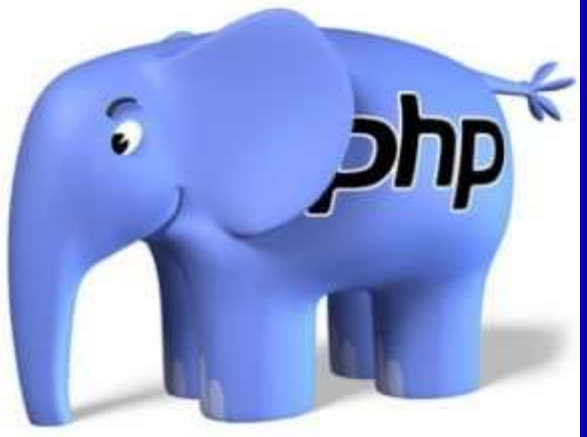
3. Crear un slug con el siguiente texto: “Curso de fundamentos de PHP CST”.



Elementos monobyte y multibyte.

Existen bastantes idiomas que requieren tantos caracteres para la comunicación escrita que no pueden ser representados dentro del rango que un mero byte como los caracteres ASCII.

```
<?php
    $mensaje=“La programación es increíble.”;
    echo “<br>”.strtoupper($mensaje);  //Monobyte
    echo “<br>”.mb_strtoupper($mensaje);  //Multibyte
?>
```



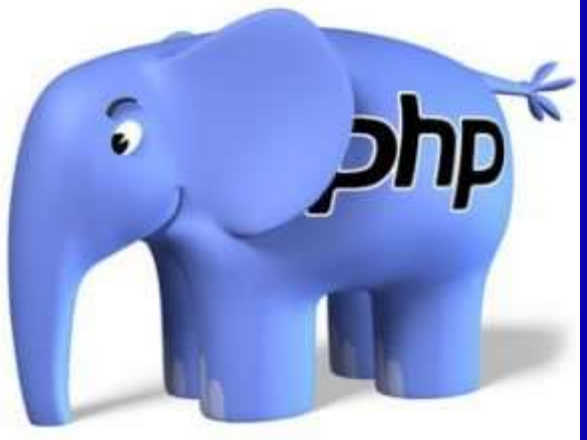
Expresiones regulares.

Las expresiones regulares son patrones que se utilizan para hacer coincidir combinaciones de caracteres en cadenas.

preg_match(<patrón>,<cadena>)

<patrón> La expresión regular a buscar.
<cadena> La cadena a analizar.

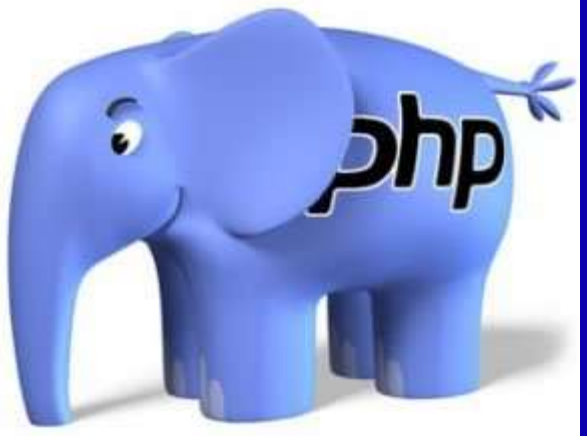
Hagamos un ejemplo con una contraseña con var_dump…



Expresiones regulares.

Algunas expresiones regulares comunes. Las expresiones regulares son parte de la teoría computacional para implementar mayor seguridad en un sitio al pedir datos y supervisar que no entren datos maliciosos.

Nombre de un usuario en un blog:	**/^[a-z0-9_-]{3,16}$/**
Contraseña de un usuario:	**/^[a-z0-9_-]{6,18}$/**
Un valor en hexadecimal:	**/^#?([a-f0-9]{6}|[a-f0-9]{3})$/**
Un slug:	**/^[a-z0-9-]+$/**

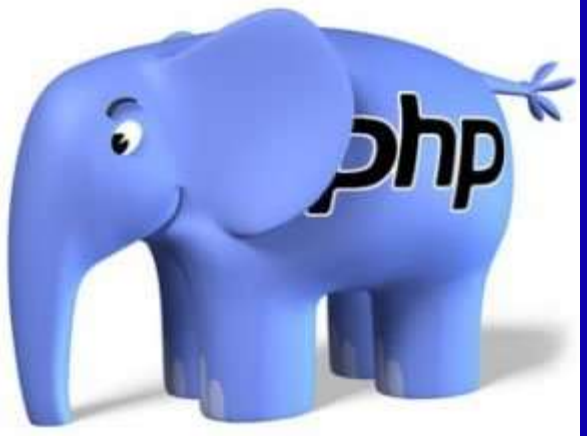


Variables variables.

A veces es conveniente tener nombres de variables variables. Dicho de otro modo, son nombres de variables que se pueden definir y usar dinámicamente.

```php
<?php
    $a = 'hola';  // Variable normal
    $$a = 'mundo';  //Variable variable
    echo "$a ${$a}";  //Veamos que pasa
?>
```

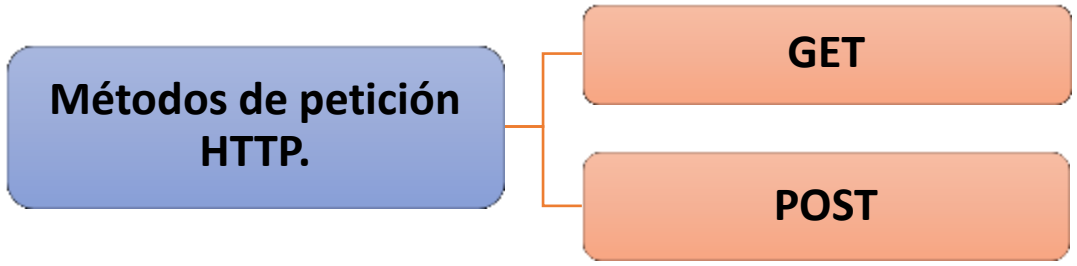
Hagamos un ejemplo…

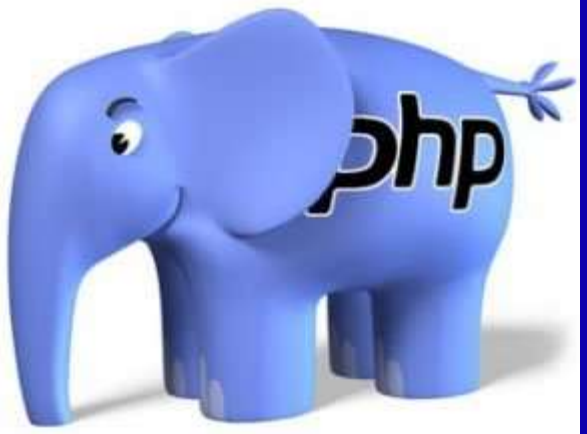


Variables desde fuentes externas.

Existen una variedad de fuentes externas a PHP de las cuales podemos captar datos y guardarlos, sin embargo, la fuente externa más común son los **formularios**.

Cuando se envía un formulario a un script de PHP, la información de dicho formulario pasa a estar automáticamente disponible en el script.





Ejercicios:

1. Hacer un formulario que pida el nombre y el correo electrónico de un usuario y luego mostrarlos en la pantalla.

2. Hacer un formulario que fijada una tarifa de 0.987 pesos por kWh pida al usuario su consumo de kWh y entregue al usuario el monto a pagar por su consumo de electricidad.

3. Hacer un formulario que pida base y altura de un rectángulo y muestre su área y perímetro.