**CLASE 01 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**PHP:** Es un lenguaje de código abierto muy popular, adecuado para desarrollo web y que puede ser incrustado en HTML.

**Popular** -> Muchas páginas están creadas con PHP / **Código** **abierto**: Uso libre y gratuito / **Incrustado**: Combinación con HTML.

Multiplataforma / Invisible al navegador / Capaz de conexión a base de datos / Capaz de expandir su potencial con plugins.

Servidores -> Ordenadores potentes con un software y hardware especial, que resuelven peticiones de otros ordenadores.

Clientes -> Cualquier dispositivo (PC, Notebook, Tablet) que generen peticiones hacia los servidores.

**¿Cómo funciona?** Cuando un cliente hace una petición, el servidor ejecuta el interprete de PHP (Para nuestro caso). Este compila el código fuente que genera el sitio web. El resultado es enviado al navegador del cliente.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**ESTRUCTURA BASICA:**

**<?php** El codigo va en el medio. Con echo se muestra mensaje en el navegador. Todas las

echo “HOLA MUNDO”; instrucciones terminan en (;). La extension del archivo Fuente debe ser .php

**?>**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Variables: PHP soporta 8 tipos primitivos**

**Escalares**: Boolean – Integer – Float – String // **Compuestos**: Array – Object // **Especiales**: Resource – NULL.

Las variables comienzan con símbolo **$**. Son case sensitivity. Los tipos se definen cuando se les asigna valor.

Ejemplos: **$**nombre = “JUAN”; **$**edad = 25; **$**sueldo = 8500 // **echo** “edad:”, $edad;

Las conversiones las realiza PHP dependiendo del contenido de las variables. También se puede realizar una conversión explicita.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Funciones de cadenas:**

**strlen():** Retorna la cantidad de caracteres de una cadena.

**strcmp():** Compara dos cadenas (Teniendo en cuenta el case sensitive).

**strtolower():** Convierte la cadena a minúsculas.

**strtoupper():** Convierte la cadena a mayúsculas.

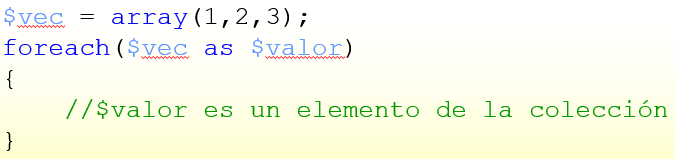
**substr():** Retorna una porción de la cadena.

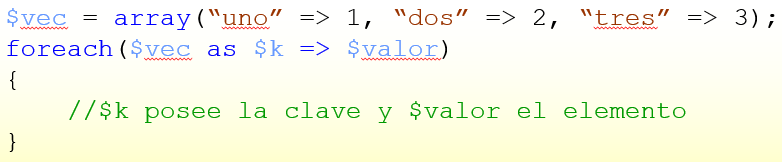
**ucfirst():** Convierte el primer carácter de la cadena a mayúsculas.

**ucwords():** Convierte el primer carácter de cada palabra de la cadena en mayúsculas.

**Operadores**: Aritméticos / Asignación / Comparación / Decremento / Lógicos / De Cadena / De Array.

**Sentencia Foreach:**





**ARRAYS:** Los Arrays en PHP son realmente mapas ordenados. Un mapa es un tipo de dato que asocia valores con claves.

**Arrays indexados** -> Índices numéricos **Arrays asociativos** -> Índices nombrados

**Arrays multidimensionales** -> Arrays que contienen otros Arrays dentro.

**Los Arrays pueden ser creados con el constructor del lenguaje array().**

$vec = array (1,2,3);

var\_dump($vec); **// array(3){[0]=>int(1) [1]=>int(2) [2]=>int(3)}**

**O sino simplemente:**

$vec[0] = 1;

$vec[1] = 2;

$vec[2] = 3;

var\_dump($vec); **// array(3){[0]=>int(1) [1]=>int(2) [2]=>int(3)}**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Arrays Asociativos** con el constructor **array().**

$vec = array("Juan"=>22, "Romina"=>12, "Uriel"=>8);

var\_dump($vec); **// array(3){["Juan"]=>int(22)["Romina"]=>int(12)[" Uriel"]=>int(8)}**

$vec["Hugo"] =15;

$vec["Juana"] = 36;

var\_dump($vec); **// array(2){["Hugo"]=>int(15)["Juana"]=>int(36)}**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Funciones para ordenar Arrays:**

**sort():** Ordena un array de forma ascendente.

**rsort():** Ordena un array de forma descendente.

**asort():** Ordena un array asociativo de forma ascendente por su valor.

**ksort():** Ordena un array asociativo de forma ascendente por su clave.

**arsort():** Ordena un array asociativo de forma descendente por su valor.

**krsort():** Ordena un array asociativo de forma descendente por su clave.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**CLASE 02 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Programación del lado del servidor – Funciones propias:**

La declaración de la función comienza con la palabra **function**. El nombre de la función puede empezar con letra o guion bajo, pero no con números. Los nombres de las funciones NO son case sensitive.

**function NombreFuncion($par\_1, $par\_2,..., $par\_n){ //código }**

**Puede retornar valores. Puede tener parametros con valores por default.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**INCLUIR ARCHIVOS:**

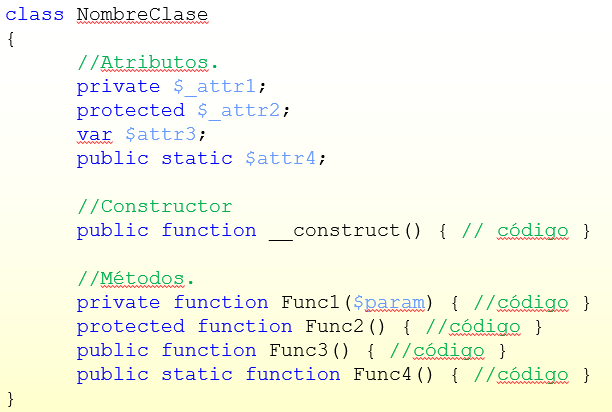
**Include o require:** Copia todo el código existente del archivo especificado dentro del archivo que posee dicha declaración.

**require**: Produce un error fatal (E\_COMPILE\_ERROR) y frenará el script

**include**: Produce una advertencia (E\_WARNING) y el script va a continuar.

**CLASES Y OBJETOS:**

**¿Cómo se declara una clase en PHP?**



**¿Cómo se declara un objeto?**

$**nombreObj** = new **NombreClase**();

Se utiliza el operador “**->”** para acceder a los miembros de instancia de la clase:

$nombreObj->Func3();

$nombreObj->attr3;

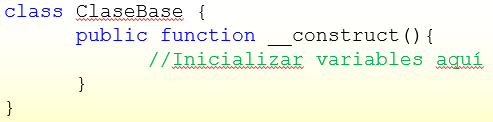
El operador “**::”** es utilizado para acceder a los miembros estaticos de la clase:

NombreClase::Func4();

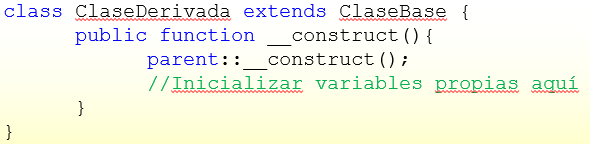
NombnreClase::$attr4;

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

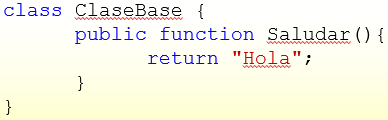
**HERENCIA: Se indica herencia a partir de extends.**

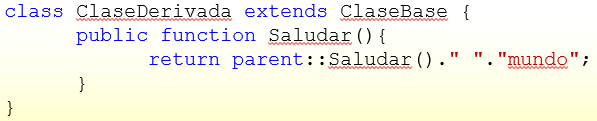


**CLASE DERIVADA:**

****

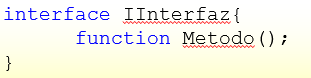
**POLIMORFISMO: Cualquier método puede ser modificado en sus clases derivadas:**

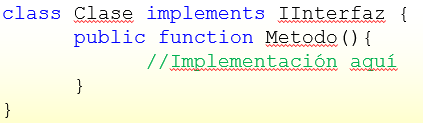




**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

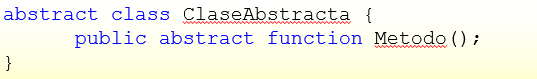
**INTERFACES: Las interfaces en PHP solo pueden contener declaraciones de métodos:**

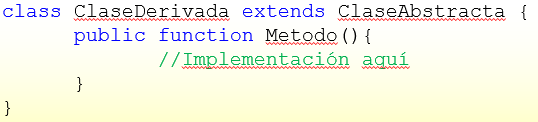




**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**CLASES ABSTRACTAS Pueden contener atributos y métodos. Pero solo ellas pueden contener métodos con el modificador abstract.**





**CLASE 03 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Manipulación de archivos:**

**Abrir archivos:**  **int** fopen(archivo, **modo**);

**MODOS:**

r Abre un archivo para solo lectura. El cursor comienza al principio del archivo.

w Abre un archivo para solo escritura. Si no existe, crea uno nuevo. Si existe, borra el contenido.

a Abre un archivo para solo escritura. Si no existe, crea uno nuevo. Si existe, mantiene el contenido. Cursor al final.

x Crea un archivo para solo lectura. Retorna false y un error si el archivo existe.

r+ Abre un archivo para lectura / escritura. Idem r.

w+ Abre un archivo para lectura / escritura. Idem w.

a+ Abre un archivo para lectura / escritura. Idem a.

x+ Abre un archivo para lectura / escritura. Idem x.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**CERRAR ARCHIVOS:**

**$ar** = fopen(archivo, modo);

fclose(**$ar**);

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**LEER ARCHIVOS:**

**echo** fread(indicador\_archivo, filesize(archivo)); // Retorna un string con todo el contenido del archivo.

**string** fgets(indicador\_archivo); // Nos permite leer una linea de un archivo abierto.

**¿Cómo uso fgets para leer todo un archivo?**

**$ar** = fopen(archivo, modo);

**while**(!feof($ar)) {

**echo** fgets($ar); }

**fclose**($ar);

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ESCRIBIR ARCHIVOS:**

**fwrite**: Nos permite escribir un archivo abierto. **fputs**: Nos permite escribir un archivo abierto.

**fwrite**(indicador\_archivo, texto [,longitud]); **fputs**(indicador\_archivo, texto [,longitud]);

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**COPIAR ARCHIVOS:**

**copy**(archivo\_origen, archivo\_destino);

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**BORRAR ARCHIVOS:**

**unlink**(archivo);

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ENVIO DE DATOS:**

**HTTP – Hypertext Transfer Protocol.**

Esta diseñado para permitir comunicación entre clientes y servidores. Funciona como un protocolo de pedido-respuesta. Un navegador web puede ser un cliente y una aplicación sobre un computador que aloja un sitio web puede ser el servidor.

**GET.**

El par de nombres / valores es enviado en la dirección URL (Texto claro). Las peticiones GET se pueden almacenar en caché. Permanecen en el historial del navegador. Pueden ser marcadas (book marked). Nunca debe ser utilizado cuando se trata de datos confidenciales. Tienen una longitud máxima de 2048 caracteres en la URL.

**POST.**

El par de nombres / valores es enviado en el cuerpo del mensaje HTTP. Las peticiones POST no se almacenan en el caché. No permanecen en el historial del navegador. No pueden ser marcadas. No poseen restricciones de longitud de datos.

**MANEJO DE FORMULARIOS:**

* Tanto GET como POST crean un *array asociativo*.
* Dicho array contiene pares de clave-valor, dónde las claves son los nombres (atributo ***name***) de los controles del formulario y los valores son la entrada de datos del usuario.
* PHP utiliza las ***super globales*** $\_GET, $\_POST y $\_REQUEST para recolectar datos provenientes de un Form.
* $\_GET es un array pasado por GET.
* $\_POST es un array pasado por POST.
* $\_REQUEST es un array asociativo que contiene $\_GET, $\_POST y $\_COOKIE.

**CLASE 04 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***