1. Introducción a PHP

- **PHP (Hypertext Preprocessor)** es un lenguaje de programación del lado del servidor diseñado principalmente para el desarrollo web.
- Los scripts de PHP se ejecutan en el servidor y el resultado es enviado al navegador como HTML.
- Los archivos PHP tienen la extensión `.php`.

2. Sintaxis Básica

- Un script PHP comienza con `<?php` y termina con `?>`.
- Un simple ejemplo de script PHP:
- `<?php`
- `echo "Hola, mundo!";`
- `?>`

3. Variables en PHP

- En PHP, las variables comienzan con el signo `$`.
- Las variables no necesitan declaración previa ni definir el tipo.
- Ejemplo:
- `<?php`
- `$nombre = "Juan";`
- `$edad = 25;`
- `echo "Mi nombre es $nombre y tengo $edad años.";`
- `?>`

4. Operadores

- **Aritméticos**: `+`, `-`, `*`, `/`, `%`.
- **De comparación**: `==`, `!=`, `>`, `<`, `>=`, `<=`.
- **Lógicos**: `&&`, `||`, `!`.

Ejemplo de operadores aritméticos:

```
<?php
$a = 5;
$b = 10;
echo $a + $b; // Salida: 15
?>
```

5. Estructuras de Control

- **Condicionales (if, else, elseif):**

```php
<?php
$edad = 20;
if ($edad >= 18) {
    echo "Eres mayor de edad.";
} else {
    echo "Eres menor de edad.";
}
?>
```

- **Bucles (while, for):**

```php
// Bucle for
for ($i = 0; $i < 10; $i++) {
    echo $i;
}

// Bucle while
$contador = 0;
while ($contador < 10) {
    echo $contador;
    $contador++;
}
```

6. Funciones

- Las funciones en PHP se definen con la palabra clave `function`.
- Ejemplo de función:

```php
<?php
function saludar($nombre) {
    echo "Hola, $nombre!";
}
saludar("María"); // Salida: Hola, María!
?>
```

7. Arrays

- Un array puede contener múltiples valores bajo una sola variable.
- Existen dos tipos de arrays:
    - **Indexados:** Los índices son números.
    - **Asociativos:** Los índices son cadenas.

Ejemplos:

```php
// Array indexado
$frutas = array("manzana", "banana", "naranja");
echo $frutas[0]; // Salida: manzana

// Array asociativo
$edad = array("Juan" => 25, "María" => 30);
echo $edad["Juan"]; // Salida: 25
```

8. Formularios en PHP

- Los datos de los formularios HTML se pueden enviar y procesar con PHP usando `$_GET` o `$_POST`.
- Ejemplo:
- `<form method="POST" action="procesar.php">`
- `    Nombre: <input type="text" name="nombre">`
- `    <input type="submit" value="Enviar">`
- `</form>`
  En el archivo `procesar.php`:

  ```php
  <?php
  $nombre = $_POST['nombre'];
  echo "Hola, $nombre!";
  ?>
  ```

9. Manejo de Archivos

- PHP permite abrir, leer y escribir archivos con las funciones `fopen()`, `fread()`, `fwrite()`, etc.
- Ejemplo para escribir en un archivo:
- `<?php`
- `$archivo = fopen("archivo.txt", "w");`
- `fwrite($archivo, "Este es un texto.");`
- `fclose($archivo);`
- `?>`

10. Conexión a Base de Datos (MySQL)

- PHP se puede usar para interactuar con bases de datos MySQL utilizando `mysqli` o `PDO`.
- Ejemplo con `mysqli`:
- ```
  <?php
  $conexion = mysqli_connect("localhost", "usuario", "contraseña", "base_datos");

  if (!$conexion) {
      die("Error en la conexión: " . mysqli_connect_error());
  }

  $sql = "SELECT * FROM usuarios";
  $resultado = mysqli_query($conexion, $sql);

  while ($fila = mysqli_fetch_assoc($resultado)) {
      echo $fila["nombre"];
  }

  mysqli_close($conexion);
  ?>
  ```

11. Sesiones y Cookies

- **Sesiones**: Se usan para almacenar información del usuario durante su navegación.
- ```
  <?php
  session_start();
  $_SESSION['usuario'] = "Juan";
  echo $_SESSION['usuario'];
  ?>
  ```
- **Cookies**: Son pequeños archivos que se almacenan en el navegador del usuario.
- ```
  <?php
  setcookie("usuario", "Juan", time() + 3600); // Expira en una hora
  echo $_COOKIE['usuario'];
  ?>
  ```

12. Errores y Depuración

- PHP tiene varias funciones para manejar errores y depurar código, como `error_reporting()` o `var_dump()`.
  ```

- Ejemplo:
- <?php
- error_reporting(E_ALL);
- $x = "Hola";
- var_dump($x); // Salida: string(4) "Hola"
- ?>

13. Buenas Prácticas

- Usar **nombres descriptivos** para las variables.
- Comentar el código donde sea necesario.
- Validar y sanitizar los datos del usuario.
- Mantener el código modular usando funciones.

1. Instalar un servidor web con PHP (entorno local)

Si estás trabajando en tu computadora localmente, necesitarás un servidor que soporte PHP. Algunas opciones comunes son:

- **XAMPP** (Windows, macOS, Linux): Viene con Apache, PHP y MySQL.
- **MAMP** (macOS, Windows): Incluye Apache, PHP y MySQL.
- **LAMP** (Linux): Apache, MySQL y PHP para Linux.
- **WAMP** (Windows): Apache, MySQL y PHP para Windows.

Pasos para XAMPP o MAMP (ejemplo local):

1. **Instalar XAMPP o MAMP**: Descárgalo desde su sitio web oficial y sigue el asistente de instalación.
2. **Ubicar el directorio de tu servidor web**:
    - Para **XAMPP**, el directorio de trabajo se encuentra en `C:/xampp/htdocs/`.
    - Para **MAMP**, es `Applications/MAMP/htdocs/`.
3. **Crear un archivo PHP**:
    - Crea un archivo con extensión `.php`, por ejemplo, `index.php` en el directorio `htdocs`.
    - Puedes usar cualquier editor de texto como **VS Code**, **Sublime Text**, o **Notepad++** para escribir tu código PHP.

    Ejemplo de un script PHP simple (dentro del archivo `index.php`):

    ```
    <?php
    echo "¡Hola, mundo! Este es mi primer script PHP";
    ?>
    ```

4. **Iniciar el servidor**:
    - Abre el panel de control de XAMPP o MAMP y enciende Apache.
5. **Acceder a tu script PHP**:
    - Abre un navegador web y escribe `http://localhost/index.php` para ejecutar el script.

2. Vincular PHP en un servidor remoto (producción)

Si tienes un servidor web remoto (por ejemplo, un hosting compartido o un VPS), sigue estos pasos:

1. **Asegúrate de tener acceso FTP/SFTP o SSH**:
    - Si tu proveedor de hosting ofrece cPanel o Plesk, también puedes usar estas herramientas para subir archivos.
    - Para **FTP/SFTP**, necesitarás un cliente FTP como **FileZilla** o **Cyberduck**.
2. **Conéctate al servidor**:
    - Abre FileZilla o tu cliente FTP y conéctate con las credenciales proporcionadas por tu proveedor de hosting.
    - En muchos casos, los archivos públicos del sitio web estarán en una carpeta como `public_html` o `www`.
3. **Subir el script PHP**:
    - Crea un archivo PHP (por ejemplo, `index.php`) en tu computadora local y súbelo al servidor en la carpeta correcta (normalmente `public_html`).

    Ejemplo de un archivo `index.php`:

    ```php
    <?php
    echo "¡Hola desde mi servidor remoto!";
    ?>
    ```
4. **Configurar los permisos de archivo** (si es necesario):
    - Asegúrate de que los permisos del archivo sean correctos. Generalmente, deben estar en `644` para archivos PHP.
5. **Acceder al archivo desde el navegador**:
    - Una vez que el archivo esté subido al servidor, puedes acceder a él desde tu navegador ingresando la URL de tu sitio web, por ejemplo: `http://tu-dominio.com/index.php`.

3. Conectar PHP a una base de datos (opcional)

Si quieres vincular tu script PHP a una base de datos como MySQL, puedes hacerlo con los siguientes pasos básicos:

1. **Crear una base de datos MySQL**:

    - En tu panel de control (como cPanel), crea una nueva base de datos y un usuario.
2. **Escribir un script PHP para conectarse a MySQL**:
    - Puedes usar la extensión `mysqli` o `PDO` en PHP para conectarte a una base de datos.

    Ejemplo de una conexión simple usando `mysqli`:

```php
<?php
$servername = "localhost";
$username = "usuario_db";
$password = "contraseña_db";
$dbname = "nombre_db";

// Crear la conexión
$conn = new mysqli($servername, $username, $password, $dbname);

// Verificar la conexión
if ($conn->connect_error) {
    die("Conexión fallida: " . $conn->connect_error);
}
echo "Conexión exitosa";
?>
```

3. **Verificar la conexión**:
    - Subes este archivo al servidor o lo pruebas localmente para verificar que se conecta correctamente a la base de datos.

Recomendaciones adicionales:

- **Depuración**: Usa la función `error_log()` o habilita `display_errors` en PHP para depurar tus scripts.
- **Seguridad**: Asegúrate de proteger tu código contra inyecciones SQL y otros ataques. Utiliza consultas preparadas con PDO o `mysqli`.
- **Versiones de PHP**: Verifica la versión de PHP en tu servidor para asegurarte de que es compatible con el código que estás escribiendo.

Con estos pasos deberías poder crear y vincular tus scripts PHP tanto en un entorno local como en un servidor remoto.