1.INCLUIR FUNCIONES

INCLUDE-REQUIRE	
funciones.php	**script.php**

// **funciones.php**
<?php
function sumar($a, $b) {
return $a + $b;
}
function restar($a, $b) {
return $a - $b;
}
?>
	// script.php
	<?php
	include("funciones.php");
	$resultado_suma = sumar(5, 3);
	$resultado_resta = restar(10, 4);
	echo "Resultado de la suma: $resultado_suma
	";
	echo "Resultado de la resta: $resultado_resta";
	?>
	
	// script.php
	<?php
	require("funciones.php");
	$resultado_suma = sumar(5, 3);
	$resultado_resta = restar(10, 4);
	echo "Resultado de la suma: $resultado_suma
	";
	echo "Resultado de la resta: $resultado_resta";
	?>

/* Usamos include("funciones.php") para incluir el archivo "funciones.php" en nuestro script "script.php".Después de incluir el archivo, podemos llamar a las funciones sumar() y restar() definidas en "funciones.php" como si estuvieran definidas en "script.php". */	

2.PROCESAR IMAGEN

PROCESAR IMAGEN EN DOS ARCHIVOS INDEPENDIENTES	
Formulario.php	**Procesar-imagen.php**

<!DOCTYPE html>
<html>
<head>
<title>Subir Imagen</title>
</head>
<body>
 Subir Imagen
<form method="POST" action="procesar_imagen.php"
enctype="multipart/form-data">
<input type="file" name="imagen" id="imagen"
accept="image/*" required>
<input type="submit" value="Subir">
</form>
</body>
</html>
	<?php
	// Verificar si se ha enviado un archivo
	if(isset($_FILES["imagen"])) {
	// Obtener detalles del archivo
	$nombre_archivo = $_FILES["imagen"]["name"];
	$tipo_archivo = $_FILES["imagen"]["type"];
	$tamano_archivo = $_FILES["imagen"]["size"];
	$ubicacion_temporal = $_FILES["imagen"]["tmp_name"];
	// Definir la ubicación donde se guardará el archivo subido
	$directorio_destino = "archivos_subidos/";
	// Mover el archivo temporal al directorio de destino
	if(move_uploaded_file($ubicacion_temporal, $directorio_destino .
	$nombre_archivo)) {
	echo "El archivo se ha subido con éxito.";
	} else {
	echo "Error al subir el archivo.";
	}
	} else {
	echo "No se ha enviado ningún archivo.";
	}
	?>

Primero, verificamos si se ha enviado un archivo usando isset($_FILES["imagen"]).

• **Luego, accedemos a los detalles del archivo subido,** como su nombre, tipo, tamaño y ubicación temporal, a través de $_FILES["imagen"].

• **Definimos un directorio de destino donde se guardará el archivo subido.** En este caso, el archivo se guarda en un directorio llamado "archivos_subidos/".

• **Usamos la función move_uploaded_file()** para mover el archivo temporal desde la ubicación temporal a la ubicación de destino.

• **Finalmente,** mostramos un mensaje de éxito o error según el resultado de la operación de carga.

Este ejemplo muestra cómo utilizar $_FILES para procesar archivos subidos a través de un formulario HTML y cómo moverlos a un directorio de destino en el servidor. Debes establecer permisos adecuados en el directorio de destino para permitir la escritura.

PROCESAR IMAGEN EN 1 ARCHIVO: "procesar-imagen.php"

```
<!DOCTYPE html>
<html lang="en">
	<head>
		<meta charset="UTF-8">
		<meta name="viewport" content="width=device-width, initial-scale=1.0">
		<title>Formulario de Contacto</title>
	</head>
	<body>
		<h1>Procesar Imágen</h1>
			<?php
			// Verificar si se ha enviado un archivo
			if(isset($_FILES["imagen"])) {
			// Obtener detalles del archivo
			$nombre_archivo = $_FILES["imagen"]["name"];
			$tipo_archivo = $_FILES["imagen"]["type"];
			$tamano_archivo = $_FILES["imagen"]["size"];
			$ubicacion_temporal = $_FILES["imagen"]["tmp_name"];
			// Definir la ubicación donde se guardará el archivo subido
			$directorio_destino = "archivos_subidos/";
			// Mover el archivo temporal al directorio de destino
			if(move_uploaded_file($ubicacion_temporal, $directorio_destino . $nombre_archivo)) {
			echo "El archivo se ha subido con éxito.";
			} else {
			echo "Error al subir el archivo.";
			}
			} else {
			echo "No se ha enviado ningún archivo.";
			}
			?>
		<h2>Subir Imagen</h2>
		<form method="POST" action="" enctype="multipart/form-data">
			<input type="file" name="imagen" id="imagen" accept="image/*" required>
			<input type="submit" value="Subir">
		</form>
	</body>
</html>
```

3.USO de ASSERT

ASSERT

```
<? php
	class CuentaBancaria {
		private $saldo = 0;
		public function depositar($monto) {
		assert($monto > 0, "El monto del depósito debe ser positivo.");
		$this->saldo += $monto;
	}
	public function retirar($monto) {
		assert($monto > 0, "El monto del retiro debe ser positivo.");
		assert($monto <= $this->saldo, "No hay suficiente saldo
		retiro.");
		$this->saldo -= $monto;
		}
	}
?>
```

/*En el contexto de la POO, puedes usar assert para **verificar condiciones invariantes en los métodos de una clase**. Aquí hay un ejemplo de cómo podrías usar assert para verificar la condición invariante de saldo no negativo en una clase de cuenta bancaria en PHP:*/

/*En este **ejemplo**, hemos usado assert para verificar que el monto del depósito y el retiro son positivos, así como para verificar que el saldo es suficiente para un retiro. Si alguna de estas condiciones invariantes se viola durante la ejecución del programa, se generará una excepción o se detendrá el programa, lo que permite identificar y corregir errores de manera temprana y garantizar que los objetos se mantengan en un estado válido.*/

4.Métdos GETTER / SETTER

Getter-setter.php

```php
<?php
	class Usuario {
		private $nombre;
		private $correo;
		private $contrasena;
		// Setter para establecer el nombre del usuario
		public function setNombre($nombre) {
		   $this->nombre = $nombre;
		}
		// Getter para obtener el nombre del usuario
		public function getNombre() {
		   return $this->nombre;
		}
		// Setter para establecer el correo del usuario
		public function setCorreo($correo) {
		   $this->correo = $correo;
		}
		// Getter para obtener el correo del usuario
		public function getCorreo() {
		   return $this->correo;
		}
		// Setter para establecer la contraseña del usuario
		public function setContrasena($contrasena) {
		   $this->contrasena = $contrasena;
		}
		// Getter para obtener la contraseña del usuario
		public function getContrasena() {
		   return $this->contrasena;
		}
	}
	// Crear una instancia de Usuario
	$usuario = new Usuario();
	// Establecer los valores usando los métodos setter
	$usuario->setNombre("EjemploUsuario");
	$usuario->setCorreo("ejemplo@correo.com");
	$usuario->setContrasena("contrasenaSegura");
	// Obtener los valores usando los métodos getter
	$nombre = $usuario->getNombre();
	$correo = $usuario->getCorreo();
	$contrasena = $usuario->getContrasena();
?>
<!DOCTYPE html>
<html>
	<head>
	   <title>Usuario</title>
	</head>
	<body>
		<h1>Información del Usuario</h1>
		<p><strong>Nombre:</strong> <?php echo $nombre; ?></p>
		<p><strong>Correo Electrónico:</strong> <?php echo $correo; ?></p>
		<p><strong>Contraseña:</strong> <?php echo $contrasena; ?></p>
	</body>
</html>
```

/*En el contexto de la POO, puedes usar assert para **verificar condiciones invariantes en los métodos de una clase**. Aquí hay un ejemplo de cómo podrías usar assert para verificar la condición invariante de saldo no negativo en una clase de cuenta bancaria en PHP:*/

/*En este **ejemplo**, hemos usado assert para verificar que el monto del depósito y el retiro son positivos, así como para verificar que el saldo es suficiente para un retiro. Si alguna de estas condiciones invariantes se viola durante la ejecución del programa, se generará una excepción o se detendrá el programa, lo que permite identificar y corregir errores de manera temprana y garantizar que los objetos se mantengan en un estado válido.*/

5.ARRAY – ALMACENAR / ELIMINAR

Almacenar-borrar.php

```php
<?php

// Inicializar un array para almacenar los libros
$biblioteca = array();

// Función para agregar un libro a la biblioteca
function agregarLibro($titulo, $autor, $genero,
$anio_publicacion) {
    global $biblioteca;
    $libro = array(
        'titulo' => $titulo,
        'autor' => $autor,
        'genero' => $genero,
        'anio_publicacion' => $anio_publicacion
    );
    $biblioteca[] = $libro;
}

// Función para eliminar un libro de la biblioteca por índice
function eliminarLibro($indice) {
    global $biblioteca;
    if (isset($biblioteca[$indice])) {
        unset($biblioteca[$indice]);
        $biblioteca = array_values($biblioteca); // Reindexar el
array
    }
}

// Verificar si se ha enviado el formulario de agregar libro
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    if (isset($_POST['titulo']) && isset($_POST['autor'])) {
        $titulo = $_POST['titulo'];
        $autor = $_POST['autor'];
        $genero = isset($_POST['genero']) ? $_POST['genero'] : '';
        $anio_publicacion = isset($_POST['anio_publicacion']) ?
$_POST['anio_publicacion'] : '';

        agregarLibro($titulo, $autor, $genero, $anio_publicacion);
    }
}

// Verificar si se ha enviado el formulario de eliminar libro
if (isset($_GET['eliminar'])) {
    $indice = $_GET['eliminar'];
    eliminarLibro($indice);
}
?>
```

```php
<!DOCTYPE html>
<html>
<head>
    <title>Biblioteca de Libros</title>
</head>
<body>
    <h1>Biblioteca de Libros</h1>
    <h2>Agregar un nuevo libro:</h2>
    <form action="" method="post">
        Título: <input type="text" name="titulo" required><br>
        Autor: <input type="text" name="autor" required><br>
        Género: <input type="text" name="genero"><br>
        Año de Publicación: <input type="number"
name="anio_publicacion"><br>
        <input type="submit" value="Agregar Libro">
    </form>

    <h2>Libros en la biblioteca:</h2>
    <table>
        <tr>
            <th>Título</th>
            <th>Autor</th>
            <th>Género</th>
            <th>Año de Publicación</th>
            <th>Eliminar</th>
        </tr>
        <?php foreach ($biblioteca as $indice => $libro) { ?>
            <tr>
                <td><?php echo $libro['titulo']; ?></td>
                <td><?php echo $libro['autor']; ?></td>
                <td><?php echo $libro['genero']; ?></td>
                <td><?php echo $libro['anio_publicacion']; ?></td>
                <td><a href="?eliminar=<?php echo $indice;
?>">Eliminar</a></td>
            </tr>
        <?php } ?>
    </table>
</body>
</html>
```

- Este código utiliza un array llamado $biblioteca para almacenar los libros.
- La función agregarLibro agrega un nuevo libro al array, y la función eliminarLibro elimina un libro por su índice en el array.
- La información de los libros se muestra en la tabla HTML, y puedes eliminar libros haciendo clic en el enlace "Eliminar".
-

6.EJEMPLO $_SERVER

ejemplo-$_server.php

```php
<?php
    // Obtener la dirección IP del cliente
    $ip_cliente = $_SERVER['REMOTE_ADDR'];
    // Obtener el nombre del servidor
    $nombre_servidor = $_SERVER['SERVER_NAME'];
    // Obtener el método de solicitud HTTP (GET, POST, etc.)
    $metodo_http = $_SERVER['REQUEST_METHOD'];
    // Obtener la URL completa de la solicitud
    $url_completa = $_SERVER['REQUEST_URI'];
    // Obtener el agente de usuario del navegador del cliente
    $agente_usuario = $_SERVER['HTTP_USER_AGENT'];
    // Obtener el tipo de contenido preferido por el cliente
    $tipo_contenido = $_SERVER['HTTP_ACCEPT'];
    // Imprimir la información obtenida
    echo "Dirección IP del cliente: ".$ip_cliente."<br>";
    echo "Nombre del servidor: ".$nombre_servidor."<br>";
    echo "Método de solicitud HTTP: ".$metodo_http."<br>";
    echo "URL completa de la solicitud: ".$url_completa."<br>";
    echo "Agente de usuario del navegador: ".$agente_usuario."<br>";
    echo "Tipo de contenido preferido por el cliente: ".$tipo_contenido. <br>";
?>
```

7.APLICACIÓN PRÁCTICA: Calculadora

Archivos de la Aplicación	
index.php	**Calculadora.php**

```
<!DOCTYPE html>
<html>
<head>
   <title>Calculadora PHP</title>
</head>
<body>
   <h1>Calculadora PHP</h1>
<form method="post" action="calculadora.php">

<label for="numero1">Número 1:</label>
<input type="number" id="numero1" name="numero1"
required><br><br>
<label for="operacion">Operación:</label>
      <select id="operacion" name="operacion">
         <option value="suma">Suma</option>
         <option value="resta">Resta</option>
         <option value="multiplicacion">Multiplicación</option>
         <option value="division">División</option>
      </select><br><br>

      <label for="numero2">Número 2:</label>
      <input type="number" id="numero2" name="numero2"
required><br><br>

      <input type="submit" value="Calcular">
   </form>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
   <title>Resultado de la Calculadora</title>
</head>
<body>
   <h1>Resultado de la Calculadora</h1>
   <?php
   if ($_SERVER["REQUEST_METHOD"] === "POST") {
      // Obtener los valores del formulario
      $numero1 = $_POST["numero1"];
      $numero2 = $_POST["numero2"];
      $operacion = $_POST["operacion"];
      // Realizar la operación seleccionada
      switch ($operacion) {
         case "suma":
            $resultado = $numero1 + $numero2;
            break;
         case "resta":
            $resultado = $numero1 - $numero2;
            break;
         case "multiplicacion":
            $resultado = $numero1 * $numero2;
            break;
         case "division":
            if ($numero2 != 0) {
               $resultado = $numero1 / $numero2;
            } else {
               $resultado = "División por cero no permitida";
            }
            break;
         default:
            $resultado = "Operación no válida";
            break;
      }
      // Mostrar el resultado
      echo "El resultado de la $operacion es: $resultado";
   } else {
      echo "Este script debe ser invocado mediante una solicitud POST
desde el formulario.";
   }
   ?>
   <br><br>
   <a href="index.php">Volver al formulario</a>
</body>
</html>
```