

## Bloque 1: Sintaxis y fundamentos

*Objetivo: Dominar la lógica básica y el control de flujo.*

- **Nivel 1 (Calculadora Simple):** Crea un script que defina dos variables `$numero1` y `$numero2` y una variable `$operacion` (puede ser "+", "-", "\*", "/"). Usa una estructura `switch` para realizar la operación e imprimir el resultado.
- **Nivel 2 (El Evaluador de Notas):** Define una variable con una nota numérica (0-10). Usa `if/elseif/else` para asignar una calificación textual (Insuficiente, Bien, Notable, Sobresaliente). Si la nota no es válida, muestra un error.
- **Nivel 3 (Tabla de Multiplicar Dinámica):** Usa un bucle `for` anidado dentro de HTML para generar una tabla visual que muestre las tablas de multiplicar del 1 al 10.
- **Nivel 4 (Analizador de Primos):** Escribe un script con un bucle `while` o `for` que recorra los números del 1 al 100. Para cada número, determina si es primo o no. Imprime solo los primos separados por comas.

## Bloque 2: Arrays y manejo de colecciones

*Objetivo: Manipular estructuras de datos.*

- **Nivel 1 (Lista de la Compra):** Crea un array indexado con 5 productos. Añade dos más usando la sintaxis de corchetes `[]` y elimina el tercero con `unset()`. Imprime la lista final con un `foreach`.
- **Nivel 2 (Agenda de Contactos):** Crea un array asociativo donde la clave sea el nombre de una persona y el valor su teléfono. Ordena el array por nombre (orden alfabético) usando `ksort()` y muéstralo.
- **Nivel 3 (Inventario Multidimensional):** Crea un array multidimensional que contenga "Productos". Cada producto debe tener: Nombre, Precio y Stock. Recorre el array y muestra solo los productos con `Stock > 0` y cuyo precio sea mayor a 10€.
- **Nivel 4 (Estadísticas con Funciones de Array):** Dado un array de 20 números aleatorios (puedes usar `rand()`), utiliza funciones nativas para: eliminar duplicados (`array_unique`), obtener los valores mayores a 50 (`array_filter`) y calcular la media aritmética de los restantes (`array_sum / count`).

## Bloque 3: Funciones y modularidad

*Objetivo: Reutilización de código y alcance de variables.*

- **Nivel 1 (Saludo Personalizable):** Crea una función saludar(\$nombre, \$horario = "dia") que devuelva "Buenos días, Juan" o "Buenas noches, Juan" según el parámetro opcional.
- **Nivel 2 (Calculadora de Referencia):** Crea una función que reciba una variable numérica **por referencia** (&\$numero). La función debe elevar ese número al cuadrado. Comprueba cómo cambia la variable original fuera de la función.
- **Nivel 3 (Formateador de Texto con Callback):** Crea una función procesarTexto(\$texto, \$formato) donde \$formato sea una función (callable/closure) pasada como argumento. Usa esto para pasar una función anónima que convierta el texto a mayúsculas o lo invierta.
- **Nivel 4 (Manejo de Errores Personalizado):** Crea una función de división. Si el divisor es 0, dispara un error con trigger\_error() o lanza una Exception. Captúralo fuera de la función.

## Bloque 4: Programación Orientada a Objetos (POO)

*Objetivo: Abstracción, encapsulamiento y polimorfismo.*

- **Nivel 1 (Clase Coche):** Define una clase Coche con propiedades marca y color (públicas). Crea un constructor para inicializarlas y un método mostrarInfo() que imprima "Coche [marca] de color [color]". Instancia dos objetos.
- **Nivel 2 (Encapsulamiento y Mágicos):** Crea una clase Usuario con propiedades private. Implementa los métodos mágicos \_\_get y \_\_set para acceder a ellas validando que el 'email' contenga un '@'. Implementa \_\_toString para imprimir el usuario como un JSON.
- **Nivel 3 (Herencia y Polimorfismo):** Crea una clase abstracta FiguraGeometrica con un método abstracto calcularArea(). Crea las clases hijas Cuadrado y Circulo que implementen ese método. Crea un array de figuras y calcula el área de todas en un bucle (polimorfismo).
- **Nivel 4 (Interfaces, Traits y Namespaces):**
  - Crea una **Interface** Logeable (método log()).
  - Crea un **Trait** Utilidades que tenga una función generarIdUnico().
  - Crea una clase en un **Namespace** App\Modelos que use el Trait e implemente la Interface. Implementa una **Clase Anónima** en el script principal para probar la interfaz al vuelo.

## Bloque 5: Comunicación con el cliente (Formularios)

*Objetivo: Interacción HTTP y seguridad.*

- **Nivel 1 (Formulario GET simple):** Crea un formulario HTML con un campo "Nombre". Al enviar, el mismo archivo PHP debe detectar si llega información por `$_GET` y mostrar "Hola, [Nombre]".
- **Nivel 2 (Calculadora POST):** Haz un formulario con dos inputs numéricos y un select para la operación. Recíbelo por `$_POST`, valida que sean números con `is_numeric()` y muestra el resultado.
- **Nivel 3 (Validación con Filtros):** Crea un formulario de registro (Email, Edad, URL web). Usa `filter_var()` con `FILTER_VALIDATE_EMAIL`, `FILTER_VALIDATE_INT` y `FILTER_VALIDATE_URL` para asegurar los datos. Muestra mensajes de error específicos si fallan.
- **Nivel 4 (Subida de Ficheros):** Crea un formulario multipart/form-data para subir una imagen de perfil. En PHP, usa `$_FILES` para validar que sea una imagen (jpg/png), que pese menos de 2MB y muévela con `move_uploaded_file()` a una carpeta "uploads".

## Bloque 6 y 7: Cookies y Sesiones

*Objetivo: Persistencia de datos entre peticiones.*

- **Nivel 1 (Contador de Visitas - Cookie):** Crea una cookie que dure 1 año. Cada vez que el usuario recargue la página, lee el valor de la cookie, súmalo 1 y vuélvela a guardar. Muestra "Has visitado esta página X veces".
- **Nivel 2 (Preferencia de Tema - Cookie):** Un formulario con dos botones: "Modo Claro" y "Modo Oscuro". Al hacer clic, guarda la preferencia en una cookie y recarga. Usa CSS condicional según el valor de la cookie.

- **Nivel 3 (Login Básico - Sesión):** Crea `login.php` (formulario) y `admin.php` (zona privada). En el login, si usuario/pass son "admin/1234", inicia `session_start()` y guarda `$_SESSION['usuario']`. En `admin.php`, si la sesión no existe, redirige al login con `header()`.
- **Nivel 4 (Carrito de Compras - Sesión):** Crea una lista de productos con botones "Añadir". Al pulsar, guarda el ID del producto en un array dentro de `$_SESSION['carrito']`. Muestra en el lateral cuántos productos hay en el carrito sin perderlos al navegar. Incluye un botón para `session_destroy()` (vaciar carrito).

## Bloque 8: Validación Avanzada (Regex)

*Objetivo: Patrones complejos.*

- **Nivel 1:** Valida si una cadena comienza por "PHP" (insensible a mayúsculas/minúsculas).
- **Nivel 2:** Valida un formato de fecha DD/MM/AAAA.
- **Nivel 3:** Extrae todas las direcciones de correo de un texto largo usando `preg_match_all`.
- **Nivel 4:** Valida una contraseña fuerte: Mínimo 8 caracteres, una mayúscula, una minúscula, un número y un símbolo especial.

## Bloque 9: Bases de Datos (PDO)

*Objetivo: Persistencia real de datos.*

- **Nivel 1 (Conexión y Lectura):** Conéctate a una BBDD MySQL (creada en PhpMyAdmin) llamada "tienda" usando PDO. Haz un `SELECT * FROM productos` y muéstralos en una lista `<ul>`.
- **Nivel 2 (Inserción Segura):** Crea un formulario para añadir productos. Usa **Sentencias Preparadas** (`prepare`, `bindValue`, `execute`) para insertar los datos evitando inyección SQL.
- **Nivel 3 (Edición y Borrado):** Lista los productos en una tabla HTML con enlaces "Editar" y "Borrar" que pasen el ID por GET. Crea la lógica para borrar (`DELETE`) y un formulario pre-llenado para actualizar (`UPDATE`).
- **Nivel 4 (Transacciones):** Simula una transferencia bancaria. Debes restar saldo al usuario A y sumar al usuario B. Envuelve ambas consultas en una transacción.

(beginTransaction, commit, rollback) para asegurar que si falla la segunda, no se ejecute la primera.

## Bloque 10 y 11: Librerías y Errores

*Objetivo: Código profesional y robusto.*

- **Nivel 1 (Try/Catch):** Intenta conectar a la BBDD con credenciales incorrectas dentro de un bloque try. Captura la `PDOException` en el catch y muestra un mensaje amigable en lugar del error fatal.
- **Nivel 2 (Composer básico):** Instala Composer e inicializa un proyecto. Instala la librería `monolog/monolog`.
- **Nivel 3 (Uso de Librería):** Usa Monolog para crear un archivo de log `app.log` y registra un aviso cada vez que un usuario inicie sesión (del ejercicio de sesiones).
- **Nivel 4 (PHPMailer y Excepciones):** Configura PHPMailer (instalado vía Composer) para enviar un correo de "Bienvenida". Envuelve el envío en un try/catch para manejar errores SMTP.