

```

<!DOCTYPE html> <!-- Declara que el documento es HTML5; ayuda al navegador a renderizar correctamente -->
<html lang="es"> <!-- El contenido principal está en español; mejora accesibilidad y SEO -->
<head> <!-- Cabecera del documento: metadatos y recursos no visibles directamente -->
    <meta charset="UTF-8"> <!-- Conjunto de caracteres: UTF-8 permite tildes, ñ, símbolos -->
    <title>Calculadora Básica en PHP</title> <!-- Título que se muestra en la pestaña del navegador -->
</head>
<body> <!-- Comienza el contenido visible de la página -->
    <h2>Calculadora Básica</h2> <!-- Encabezado para el usuario -->

    <!-- FORMULARIO: captura de datos del usuario -->
    <!-- action (omitido) => por defecto envía a esta misma URL; method="post" => manda datos en el cuerpo de la petición -->
    <form method="post"> <!-- Inicia formulario; POST evita mostrar datos en la URL y permite más volumen -->

        <!-- Etiqueta asociada al input con id="a"; al pinchar la etiqueta, enfoca el input -->
        <label for="a">Número 1:</label>
        <!-- type="number": pide números; los valores igualmente llegan como cadena a PHP -->
        <!-- step="any": PERMITE CUALQUIER DECIMAL; sin esto el navegador valida "pasos" de 1 (0,1,2,...) -->
        <!-- name="a": clave con la que PHP recibirá el valor en $_POST['a'] -->
        <!-- id="a": necesario para vincular la etiqueta <label for="a"> -->
        <!-- required: el navegador no permite enviar el formulario si está vacío -->
        <input type="number" step="any" name="a" id="a" required>
        <br><br> <!-- Dos saltos de línea para separación visual (presentacional) -->

        <label for="b">Número 2:</label> <!-- Mismo concepto que el anterior pero para "b" -->
        <!-- step="any" otra vez para admitir decimales libres en el segundo número -->
        <input type="number" step="any" name="b" id="b" required>
        <br><br>

        <label for="operacion">Operación:</label> <!-- Texto asociado al <select id="operacion"> -->
        <!-- <select>: control desplegable; name="operacion" será la clave en $_POST['operacion'] -->
        <!-- required: obliga a elegir una opción (aunque ya hay una por defecto) -->
        <select name="operacion" id="operacion" required>
            <!-- Cada <option> tiene un value (lo que viaja al servidor) y un texto visible (lo que ve el usuario) -->
            <option value="suma">Suma (+)</option> <!-- value="suma" => string que usará PHP en el switch -->
            <option value="resta">Resta (-)</option>
            <option value="multiplicacion">Multiplicación (×)</option>
            <option value="division">División (÷)</option>
        </select>
        <br><br>

        <!-- type="submit": botón que envía el formulario -->
        <!-- value="Calcular": texto que aparece en el botón -->
        <input type="submit" value="Calcular">
    </form>

    <hr> <!-- Regla horizontal; separación visual entre formulario y resultado -->

    <?php /* A partir de aquí corre PHP: lógica del lado servidor */ ?>

    <?php
    /**
    * calcular(): ejecuta la operación pedida sobre $a y $b.
    * PHP es de tipado dinámico: aquí trataremos $a y $b como números (float).
    * Ejemplos de tipado dinámico:
    * $variable = 10;    // Ahora es un entero (int)
    * $variable = "Hola"; // Ahora es una cadena (string)
    * $variable = 3.14;  // Ahora es un decimal (float)
    * $variable = true;  // Ahora es booleano

```

```

*/
function calcular($a, $b, $operacion) { // Definición de función con 3 parámetros
    // switch: selecciona un bloque según el valor exacto de $operacion
    switch ($operacion) {
        case "suma": // Coincide con value="suma" del <option>
            return $a + $b; // Operador + (suma). return: sale de la función y entrega el resultado
            // (No hace falta "break" porque return ya corta la ejecución)
        case "resta":
            return $a - $b; // Operador - (resta)
        case "multiplicacion":
            return $a * $b; // Operador * (multiplicación)
        case "division":
            // Operador ternario condicional: condición ? valor_si_verdadero : valor_si_falso
            // Validación para evitar división entre 0 (error matemático/Infinity)
            return ($b != 0) ? $a / $b : "Error: división por cero";
        default: // Si $operacion tiene un valor no previsto
            return "Operación no válida"; // Mensaje de error por robustez
    }
}

/* PROCESAMIENTO DEL FORMULARIO: solo se ejecuta si el usuario acaba de enviar (POST) */
if ($_SERVER['REQUEST_METHOD'] === "POST") { // $_SERVER['REQUEST_METHOD'] => método HTTP usado; uso ===
(comparación estricta)
    // $_POST es un array superglobal con los datos del formulario (clave = name del input)
    // Casteamos a float para asegurar tipo numérico en operaciones (los inputs llegan como cadena)
    $a = isset($_POST["a"]) ? (float) $_POST["a"] : 0.0; // (float): conversión explícita a número decimal
    $b = isset($_POST["b"]) ? (float) $_POST["b"] : 0.0;
    $operacion = isset($_POST["operacion"]) ? $_POST["operacion"] : ""; // String con la operación elegida

    // Llamada a la función con los datos capturados
    $resultado = calcular($a, $b, $operacion); // $resultado puede ser número o texto de error

    // echo imprime HTML. htmlspecialchars evita que, si $resultado fuese texto con caracteres especiales, rompa el HTML
    // (Defensa básica XSS, aunque aquí solo imprimimos número o mensajes controlados)
    echo "<h3>Resultado: " . htmlspecialchars((string)$resultado, ENT_QUOTES, "UTF-8") . "</h3>";
}
?>
</body> <!-- Cierra el contenido visible -->
</html> <!-- Fin del documento HTML -->

```

¡De una, Héctor! Aquí va **el mismo programa** pero **comentado línea por línea**, explicando **cada comando y atributo** (incluido step="any"), con un par de micro-mejoras de práctica (añadí id a los inputs para que coincidan con los label for, uso de === y casteo a float).

```

<!DOCTYPE html> <!-- Declara que el documento es HTML5; ayuda al navegador a renderizar correctamente -->
<html lang="es"> <!-- El contenido principal está en español; mejora accesibilidad y SEO -->
<head> <!-- Cabecera del documento: metadatos y recursos no visibles directamente -->
    <meta charset="UTF-8"> <!-- Conjunto de caracteres: UTF-8 permite tildes, ñ, símbolos -->
    <title>Calculadora Básica en PHP</title> <!-- Título que se muestra en la pestaña del navegador -->
</head>
<body> <!-- Comienza el contenido visible de la página -->
    <h2>Calculadora Básica</h2> <!-- Encabezado para el usuario -->

    <!-- FORMULARIO: captura de datos del usuario -->
    <!-- action (omitido) => por defecto envía a esta misma URL; method="post" => manda datos en el cuerpo de la petición -->
    <form method="post"> <!-- Inicia formulario; POST evita mostrar datos en la URL y permite más volumen -->

        <!-- Etiqueta asociada al input con id="a"; al pinchar la etiqueta, enfoca el input -->
        <label for="a">Número 1:</label>

```

```

<!-- type="number": pide números; los valores igualmente llegan como cadena a PHP -->
<!-- step="any": PERMITE CUALQUIER DECIMAL; sin esto el navegador valida "pasos" de 1 (0,1,2,...) -->
<!-- name="a": clave con la que PHP recibirá el valor en $_POST['a'] -->
<!-- id="a": necesario para vincular la etiqueta <label for="a"> -->
<!-- required: el navegador no permite enviar el formulario si está vacío -->
<input type="number" step="any" name="a" id="a" required>
<br><br> <!-- Dos saltos de línea para separación visual (presentacional) -->

<label for="b">Número 2:</label> <!-- Mismo concepto que el anterior pero para "b" -->
<!-- step="any" otra vez para admitir decimales libres en el segundo número -->
<input type="number" step="any" name="b" id="b" required>
<br><br>

<label for="operacion">Operación:</label> <!-- Texto asociado al <select id="operacion"> -->
<!-- <select>: control desplegable; name="operacion" será la clave en $_POST['operacion'] -->
<!-- required: obliga a elegir una opción (aunque ya hay una por defecto) -->
<select name="operacion" id="operacion" required>
  <!-- Cada <option> tiene un value (lo que viaja al servidor) y un texto visible (lo que ve el usuario) -->
  <option value="suma">Suma (+)</option> <!-- value="suma" => string que usará PHP en el switch -->
  <option value="resta">Resta (-)</option>
  <option value="multiplicacion">Multiplicación (×)</option>
  <option value="division">División (÷)</option>
</select>
<br><br>

<!-- type="submit": botón que envía el formulario -->
<!-- value="Calcular": texto que aparece en el botón -->
<input type="submit" value="Calcular">
</form>

<hr> <!-- Regla horizontal; separación visual entre formulario y resultado -->

<?php /* A partir de aquí corre PHP: lógica del lado servidor */ ?>

<?php
/**
 * calcular(): ejecuta la operación pedida sobre $a y $b.
 * PHP es de tipado dinámico: aquí trataremos $a y $b como números (float).
 */
function calcular($a, $b, $operacion) { // Definición de función con 3 parámetros
  // switch: selecciona un bloque según el valor exacto de $operacion
  switch ($operacion) {
    case "suma": // Coincide con value="suma" del <option>
      return $a + $b; // Operador + (suma). return: sale de la función y entrega el resultado
      // (No hace falta "break" porque return ya corta la ejecución)
    case "resta":
      return $a - $b; // Operador - (resta)
    case "multiplicacion":
      return $a * $b; // Operador * (multiplicación)
    case "division":
      // Operador ternario (tercero) condicional: condición ? valor_si_verdadero : valor_si_falso
      // Validación para evitar división entre 0 (error matemático/Infinito)
      return ($b != 0) ? $a / $b : "Error: división por cero";
    default: // Si $operacion tiene un valor no previsto
      return "Operación no válida"; // Mensaje de error por buena práctica.
  }
}

```

**/\* PROCESAMIENTO DEL FORMULARIO: solo se ejecuta si el usuario acaba de enviar (POST) \*/**

if (\$\_SERVER["REQUEST\_METHOD"] === "POST") { // \$\_SERVER['REQUEST\_METHOD'] => método HTTP usado; uso ===  
(comparación estricta)

// \$\_POST es un array superglobal con los datos del formulario (clave = name del input)

// Convertir a float para asegurar tipo numérico en operaciones (los inputs llegan como cadena)

\$a = isset(\$\_POST["a"]) ? (float) \$\_POST["a"] : 0.0; // (float): conversión explícita a número decimal

\$b = isset(\$\_POST["b"]) ? (float) \$\_POST["b"] : 0.0;

\$operacion = isset(\$\_POST["operacion"]) ? \$\_POST["operacion"] : ""; // String con la operación elegida

// Llamada a la función con los datos capturados

\$resultado = calcular(\$a, \$b, \$operacion); // \$resultado puede ser número o texto de error

// echo imprime HTML. htmlspecialchars evita que, si \$resultado fuese texto con caracteres especiales, rompa el HTML

// (Defensa básica XSS, aunque aquí solo imprimimos número o mensajes controlados)

echo "<h3>Resultado: " . htmlspecialchars((string)\$resultado, ENT\_QUOTES, "UTF-8") . "</h3>";

}

?>

</body> <!-- Cierra el contenido visible -->

</html> <!-- Fin del documento HTML -->

#### NOTAS CLAVE:

**step="any"**: en <input type="number">, define el **incremento permitido**.

Sin step, el navegador asume step="1" (solo enteros).

Con step="0.01", aceptas múltiplos de 0.01 (útil en precios).

Con **step="any"**, **aceptas cualquier decimal** (2.5, 0.333, 1.2345...), y el navegador **no marca "step mismatch"**.

**required**: validación **del lado del navegador**; impide enviar si el campo está vacío.

**name**: es la **clave** con la que recibes el dato en PHP (\$\_POST['name']).

**id + label for**: accesibilidad y usabilidad (clic en la etiqueta enfoca el campo). Enlaza for="a" con id="a".

**method="post"**: envía en el **cuerpo** de la petición (no en la URL, a diferencia de GET).

**\$\_SERVER["REQUEST\_METHOD"]**: te dice si la página se cargó "normal" (GET) o por envío de formulario (POST).

**=== vs ==**: === compara **tipo y valor** (más estricto y recomendado).

**switch + return**: usamos return dentro de cada case, por eso **no** se necesita break.

**Ternario ?::** condición ? si\_true : si\_false; lo usamos para checar **división por cero**.

**Conversión (float)**: garantiza que lo que llega como texto numérico se convierta a número real antes de operar.