Funciones predefinidas: Math

¿Qué son las funciones matemáticas?

Las funciones matemáticas en PHP son un conjunto de funciones incorporadas en el lenguaje que permiten realizar operaciones y cálculos matemáticos de forma sencilla. Estas funciones proporcionan una amplia variedad de herramientas matemáticas que facilitan el desarrollo de aplicaciones y scripts que requieren realizar operaciones numéricas y trigonométricas.

Funciones matemáticas básicas

- abs(): Devuelve el valor absoluto de un número, es decir, el valor positivo sin tener en cuenta su signo. Si el número es positivo, el resultado es el mismo número. Si el número es negativo, el resultado es el número positivo equivalente.

```php
echo abs(-5); // Imprime: 5
echo abs(10); // Imprime: 10
```

- sqrt(): Devuelve la raíz cuadrada de un número. Si el número es negativo, la función retornará NaN (Not a Number).

```php
echo sqrt(9); // Imprime: 3
echo sqrt(25); // Imprime: 5
```

- round(): Redondea un número al entero más cercano o a una cantidad específica de decimales (indicada por el segundo argumento).

```php
echo round(3.49); // Imprime: 3
echo round(3.51); // Imprime: 4
```

```
echo round(3.14159, 2); // Imprime: 3.14
```

- ceil(): Redondea un número hacia arriba al entero más cercano.

```
echo ceil(3.1); // Imprime: 4
echo ceil(5.8); // Imprime: 6
```

- floor(): Redondea un número hacia abajo al entero más cercano.

```
echo floor(3.9); // Imprime: 3
echo floor(7.2); // Imprime: 7
```

- rand(): Genera un número aleatorio entre un valor mínimo `($min)` y un valor máximo `($max)`. El valor generado puede ser un número entero o decimal dependiendo de los valores proporcionados.

```
echo rand(1, 10); // Genera un número aleatorio entre 1 y 10 (por ejemplo, 5)
echo rand(100, 200); // Genera un número aleatorio entre 100 y 200 (por ejemplo,
143)
```

Funciones trigonométricas

- `sin()`: Calcular el seno de un ángulo. Recibe un argumento numérico que representa el ángulo en radianes y devuelve el valor del seno correspondiente.

```
$angulo = 0.5;
$seno = sin($angulo);
echo "El seno de $angulo es: $seno";
```

- `cos()`: Calcular el coseno de un ángulo. Al igual que sin(), recibe un argumento numérico en radianes y devuelve el valor del coseno correspondiente.

```
$angulo = 1.2;
$coseno = cos($angulo);
echo "El coseno de $angulo es: $coseno";
```

- `tan()`: Calcula la tangente de un ángulo dado. Al igual que las funciones anteriores, toma un argumento numérico en radianes y devuelve el valor de la tangente correspondiente.

```
$angulo = 0.8;
$tangente = tan($angulo);
echo "La tangente de $angulo es: $tangente";
```

- `asin()`: Calcula el arcoseno de un número y devuelve el ángulo correspondiente en radianes.

```
$numero = 0.5;
$arcoseno = asin($numero);
echo "El arcoseno de $numero es: $arcoseno";
```

- `acos()`: Calcula el arcocoseno de un número y devuelve el ángulo correspondiente en radianes.

```
$numero = 0.8;
$arcocoseno = acos($numero);
echo "El arcocoseno de $numero es: $arcocoseno";
```

- `atan()`: Calcula la arcotangente de un número y devuelve el ángulo correspondiente en radianes.

```
$numero = 1.5;
$arcotangente = atan($numero);
echo "La arcotangente de $numero es: $arcotangente";
```

- `atan2()`: Calcula la arcotangente de las dos variables dadas, y devuelve el ángulo correspondiente en radianes.

```
$y = 2;
$x = 1;
$arcotangente = atan2($y, $x);
echo "La arcotangente de ($y, $x) es: $arcotangente";
```

> 💯 **TIP** recuerda que estas funciones trabajan con ángulos en radianes. Si necesitas trabajar con ángulos en grados, puedes utilizar las funciones deg2rad() para convertir de grados a radianes, y rad2deg() para convertir de radianes a grados.

Funciones exponenciales y logarítmicas

- `exp()`: Esta función se utiliza para calcular el valor de la función exponencial, es decir, la constante de Euler (e) elevada a una potencia dada.

```
$exponente = 2;
$resultado = exp($exponente); // Calcula e^2
```

```
echo "El resultado de e^$exponente es: $resultado"; // Imprime: El resultado de
e^2 es: 7.3890560989307
```

- `log()`: Calcula el logaritmo natural (base e) de un número dado.

```
$numero = 10;
$resultado = log($numero); // Calcula ln(10)
echo "El logaritmo natural de $numero es: $resultado"; // Imprime: El logaritmo
natural de 10 es: 2.302585092994
```

- `log10()`: Calcula el logaritmo en base 10 de un número dado.

```
$numero = 100;
$resultado = log10($numero); // Calcula log(100) en base 10
echo "El logaritmo en base 10 de $numero es: $resultado"; // Imprime: El logaritmo
en base 10 de 100 es: 2
```

- `pow()`: Calcula la potencia de un número elevado a un exponente dado.

```
$base = 2;
$exponente = 3;
$resultado = pow($base, $exponente); // Calcula 2^3
echo "$base elevado a la $exponente es: $resultado"; // Imprime: 2 elevado a la 3
es: 8
```

Funciones de redondeo y truncamiento

- `round()`: Esta función realiza el redondeo de un número a una precisión determinada. Puede redondear hacia arriba o hacia abajo, dependiendo del valor decimal siguiente. Si el decimal siguiente es igual o mayor que 0.5, el número se redondea hacia arriba; de lo contrario, se redondea hacia abajo.

```
$numero = 3.75;
$redondeo = round($numero, 1); // Redondea a una precisión de 1 decimal
echo "El redondeo de $numero es: $redondeo"; // Imprime: El redondeo de 3.75 es:
3.8
```

- `ceil()`: Esta función redondea un número hacia arriba al entero más cercano, independientemente del valor decimal.

```
$numero = 4.2;
$redondeoArriba = ceil($numero);
echo "El redondeo hacia arriba de $numero es: $redondeoArriba"; // Imprime: El
redondeo hacia arriba de 4.2 es: 5
```

- `floor()`: Realiza el redondeo de un número hacia abajo al entero más cercano, independientemente del valor decimal.

```
$numero = 7.9;
$redondeoAbajo = floor($numero);
echo "El redondeo hacia abajo de $numero es: $redondeoAbajo"; // Imprime: El
redondeo hacia abajo de 7.9 es: 7
```

- `trunc()`: Esta función trunca un número, eliminando su parte decimal y dejando solo la parte entera.

```
$numero = 9.99;
$truncamiento = trunc($numero);
echo "El truncamiento de $numero es: $truncamiento"; // Imprime: El truncamiento
de 9.99 es: 9
```

Funciones de valor mínimo y máximo

- `min()`: Esta función devuelve el valor mínimo entre varios valores proporcionados como argumentos.

```
$minimo = min(5, 10, 2, 8, 4); // Obtiene el valor mínimo entre los valores dados
echo "El valor mínimo es: $minimo"; // Imprime: El valor mínimo es: 2
```

- `max()`: Devuelve el valor máximo entre varios valores proporcionados como argumentos.

```
$maximo = max(5, 10, 2, 8, 4); // Obtiene el valor máximo entre los valores dados
echo "El valor máximo es: $maximo"; // Imprime: El valor máximo es: 10
```

- `fmin()`: Similar a min(), pero se utiliza para encontrar el valor mínimo entre valores de punto flotante (números con decimales).

```
$minimoFloat = fmin(3.14, 2.71, 1.618); // Obtiene el valor mínimo entre los
valores de punto flotante
```

```
echo "El valor mínimo de punto flotante es: $minimoFloat"; // Imprime: El valor
mínimo de punto flotante es: 1.618
```

- `fmax()`: Similar a max(), pero se utiliza para encontrar el valor máximo entre valores de punto flotante.

```
$maximoFloat = fmax(3.14, 2.71, 1.618); // Obtiene el valor máximo entre los
valores de punto flotante
echo "El valor máximo de punto flotante es: $maximoFloat"; // Imprime: El valor
máximo de punto flotante es: 3.14
```

Otras funciones matemáticas

- `dechex()`: Realiza la conversión de un número decimal a su representación en hexadecimal.

```
$decimal = 255;
$hexadecimal = dechex($decimal); // Convierte el número decimal a hexadecimal
echo "El número $decimal en hexadecimal es: $hexadecimal"; // Imprime: El número
255 en hexadecimal es: ff
```

- `hexdec()`: Convierte un número hexadecimal a su equivalente decimal.

```
$hexadecimal = '1A';
$decimal = hexdec($hexadecimal); // Convierte el número hexadecimal a decimal
echo "El número hexadecimal $hexadecimal en decimal es: $decimal"; // Imprime: El
número hexadecimal 1A en decimal es: 26
```

- `decbin()`: Convierte un número decimal a su representación en binario.

```
$decimal = 10;
$binario = decbin($decimal); // Convierte el número decimal a binario
echo "El número $decimal en binario es: $binario"; // Imprime: El número 10 en
binario es: 1010
```

- `bindec()`: Convierte un número binario a su equivalente decimal.

```
$binario = '101010';
$decimal = bindec($binario); // Convierte el número binario a decimal
echo "El número binario $binario en decimal es: $decimal"; // Imprime: El número
```

```
binario 101010 en decimal es: 42
```

- `deg2rad()`: Conversión de grados a radianes.

```
$grados = 45;
$radianes = deg2rad($grados); // Convierte los grados a radianes
echo "$grados grados es igual a $radianes radianes"; // Imprime: 45 grados es
igual a 0.78539816339745 radianes
```

- `rad2deg()`: Conversión de radianes a grados.

```
$radianes = 1.57;
$grados = rad2deg($radianes); // Convierte los radianes a grados
echo "$radianes radianes es igual a $grados grados"; // Imprime: 1.57 radianes es
igual a 90 grados
```

PI

En PHP, la constante pi está predefinida y se puede utilizar en cualquier parte del código donde se necesite su valor.

Para utilizar la constante pi en PHP, simplemente se debe escribir pi en cualquier parte del código donde se necesite su valor. No es necesario definir ni declarar la constante, ya que está predefinida en el lenguaje.

```
pi()
```

Consideraciones y precauciones

Al trabajar con funciones matemáticas en PHP, es esencial tener en cuenta algunas consideraciones y precauciones para garantizar la precisión y el correcto funcionamiento de tus cálculos.

Limitaciones y precisiones en el uso de funciones matemáticas

- Ten en cuenta que las operaciones matemáticas pueden llevar a **errores de redondeo o pérdida de precisión** en números con decimales. Para cálculos críticos, considera el uso de funciones matemáticas específicas o bibliotecas que ofrezcan mayor precisión, si es necesario.
- Al realizar cálculos con números muy grandes o muy pequeños, ten en cuenta las **limitaciones de representación de números en punto flotante**. Los números extremadamente grandes o pequeños pueden dar lugar a resultados inexactos debido a las limitaciones de precisión.

Importancia de validar y verificar los datos de entrada

- Antes de realizar operaciones matemáticas, asegúrate de **validar y verificar los datos de entrada**. Verifica que los valores proporcionados como argumentos a las funciones sean del tipo y rango adecuados para evitar errores en tiempo de ejecución.
- Si trabajas con datos proporcionados por el usuario o datos provenientes de fuentes externas, considera la posibilidad de **sanitizar y validar** esos datos antes de utilizarlos en cálculos matemáticos para evitar problemas de seguridad y errores inesperados.

Posibles errores y soluciones

- Algunas funciones matemáticas pueden generar errores o advertencias en determinadas situaciones, como **divisiones por cero, logaritmos de números no positivos, o desbordamientos de números**. Siempre verifica y maneja adecuadamente los errores generados por las funciones para evitar interrupciones en la ejecución del programa.
- Utiliza **estructuras de control como if y try-catch para capturar y manejar posibles errores o excepciones** que puedan surgir durante el procesamiento de operaciones matemáticas.
- Considera el **uso de funciones matemáticas específicas o bibliotecas que proporcionen funcionalidades más avanzadas y precisas cuando sea necesario**. PHP ofrece extensiones matemáticas y bibliotecas de terceros que pueden ampliar las capacidades de cálculo y manejo de números complejos.