****

**PHP**

**Apuntes y Documentación**

[**https://www.w3schools.com/php/php\_variables.asp**](https://www.w3schools.com/php/php_variables.asp)

**php -f nombreDelFichero.PHP**

**1 CARACTERÍSTICAS PRINCIPALES**

# ***1.1- Sintaxis y aclaraciones***

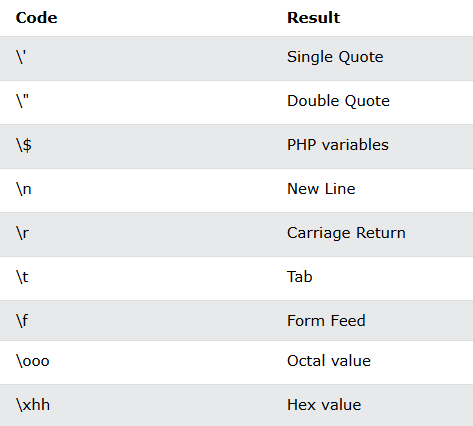
* La extensión de un documento PHP es **.php**
* Se puede combinar en un documento .php tanto HTML como PHP a la vez.
* Los comentarios en una sola línea se escriben con **//** o #.
* // Comentario de una línea. # Comentario de una línea.
* /\* Comentario multilínea. \*/
* Un script de PHP empieza con **<?php** y acaba con **?>**.

<?php

// PHP code goes here.

?>

**Caracteres especiales o caracteres de escape (Escape Characters):**

****

# ***1.2- Sensibilidad***

En PHP, palabras clave (por ejemplo: if, else, while, o echo), clases, funciones y las funciones definidas por el usuario **NO SON SENSIBLES**.

En el siguiente ejemplo, *ECHO* es lo mismo que *echo* y que *EcHo*:

<!DOCTYPE html>

<html>

<body>

<?php

// ECHO es lo mismo que echo y que EcHo.

ECHO "Hello World!<br>";

echo "Hello World!<br>";

EcHo "Hello World!<br>";

?>

</body>

</html>

En PHP, las variables se designan con un **$** al principio y sus nombres **SI SON SENSIBLES.**

En el siguiente ejemplo, podemos comprobar que *$color, $COLOR* y *$coLOR* son 3 variables diferentes:

<!DOCTYPE html>

<html>

<body>

<?php

$color = "red";

echo "My car is " . $color . "<br>";

echo "My house is " . $COLOR . "<br>";

echo "My boat is " . $coLOR . "<br>";

?>

</body>

</html>

**2 VARIABLES**

# ***1.1- Variables***

Se inicializan con **$** seguido de un nombre:

$txt = "cheese";

echo "I love $txt!";

I love cheese!

También se pueden imprimir de esta otra manera:

$txt = "cheese";

echo "I love" **.** $txt **.** "!";

// ¡También se pueden concatenar variables de esta forma!

$number = 1;

echo "I like " **.** $number **.** " type of " **.** $txt **.** "!";

I love cheese!

I love 1 cheese!

Los **“.”** de PHP equivalen a los **“+”** en Java.

**Tipos de variables:**

* String
* Integer
* Float (double)
* Boolean
* Array
* Object
* NULL
* Resource

# ***1.2- Asignar múltiples variables a un valor***

$x = $y = $z = "Fruta";

echo $x;

echo $y;

echo $z;

FrutaFrutaFruta

**3 VARIABLES SCOPE**

**4 ECHO / PRINT**

Las diferencias son mínimas:

* *echo* no tiene valor de retorno mientras que *print* tiene un valor de retorno de 1, por lo que se puede utilizar en expresiones.
* *echo* puede tomar múltiples parámetros (aunque es raro) mientras que *print* puede tomar un argumento.
* *echo* es ligeramente más rápido que *print*.

# ***4.1- Uso de comillas y variables de visualización.***

Las cadenas están rodeadas por comillas, pero existe una diferencia entre comillas simples y dobles en PHP.

Al utilizar **COMILLAS DOBLES (“ “)**, se pueden insertar variables en la cadena de esta manera:

$txt1 = "Learn PHP";

$txt2 = "W3Schools.com";

echo"<h2>$txt1</h2>";

print "<p>Study PHP at $txt2</p>";

Al utilizar **COMILLAS SIMPLES (‘ ‘)**, las variables deben insertarse utilizando el operador **.** de la siguiente manera:

$txt1 = "Learn PHP";

$txt2 = "W3Schools.com";

print '<h2>' **.** $txt1 **.** '</h2>';

print '<p>Study PHP at ' **.** $txt2 **.** '</p>';

También se pueden imprimir de esta otra manera:

$txt = "cheese";

echo "I love" **.** $txt **.** "!";

// ¡También se pueden concatenar variables de esta forma!

$number = 1;

echo "I like " **.** $number **.** " type of " **.** $txt **.** "!";

I love cheese!

I love 1 type of cheese!

Los **“.”** de PHP equivalen a los **“+”** en Java.

**5 CASTING**

En PHP, el casteo se puede realizar con estas sentencias:

* **(string)** - Convierte al tipo de datos String.
* **(int)** - Convierte al tipo de dato entero.
* **(float)** - Convierte al tipo de dato Float.
* **(bool)** - Convierte al tipo de datos booleano.
* **(array)** - Convierte al tipo de datos Array.
* **(object)** - Convierte al tipo de datos Objeto.
* **(unset)** - Convierte al tipo de datos NULL.

# ***5.1- Castear a String***

Se castea todo tipo de datos sin problema.

$a = 5;

$a = (string) $a;

var\_dump($a);

string(1) "5"

# ***5.2- Castear a int***

Si al convertir un String que empieza con un número en un número entero, la función (int) utiliza ese número. Si no empieza con un número, la función (int) convierte las cadenas al número 0.

$a = "25 kilometers";

$b = "kilometers 25";

$c = "hello";

var\_dump($a);

var\_dump($b);

var\_dump($c);

entero(25)

entero(0)

entero(0)

# ***5.3- Castear a float***

Funciona igual que el casteo a int.

$a = 5;

$a = (float) $a;

var\_dump($a);

float(5)

# ***5.4- Castear a boolean***

Si un valor es 0, *NULL*, *False* o *vacío*, (bool) lo convierte en falso, de lo contrario, en verdadero.

$a = 5;

$b = 5.34;

$c = 0;

$a = (bool) $a;

$b = (bool) $b;

$c = (bool) $c;

var\_dump($a);

var\_dump($b);

var\_dump($c);

bool(true)

bool(true)

bool(false)

# ***5.5- Castear a Array***

Al castear a array, la mayoría de los tipos de datos se convierten en un array indexado con un elemento.

El valor NULL se convierte en un array vacío.

$a = 5;

$b = 5.34

$c = "hello";

$d = true;

$e = NULL;

$a = (array) $a;

$b = (array) $b;

$c = (array) $c;

$d = (array) $d;

$e = (array) $e;

var\_dump($a);

var\_dump($b);

var\_dump($c);

var\_dump($d);

var\_dump($e);

array(1) {

[0]=>

int(5)

}

array(1) {

[0]=>

float(5.34)

}

array(1) {

[0]=>

string(5) "hello"

}

array(1) {

[0]=>

bool(true)

}

array(0) {

}

# ***5.6- Castear a object***

Al castear a objeto, la mayoría de los tipos de datos se convierten en un objeto con una propiedad, denominada *"escalar"* (scalar), con el valor correspondiente.

El valor NULL se convierte en un objeto vacío.

Los arrays se convierten en objetos con el número de índice como nombre de propiedad y el valor como valor de propiedad.

Las arrays asociativos (maps) se convierten en objetos con las claves como nombres de propiedad y los valores como valores de propiedad.

$a = 5;

$b = 5.34

$c = "hello";

$d = true;

$e = NULL;

$f = array("Volvo", "BMW", "Toyota");

$g = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

$a = (object) $a;

$b = (object) $b;

$c = (object) $c;

$d = (object) $d;

$e = (object) $e;

$f = (object) $f;

$g = (object) $g;

var\_dump($a);

var\_dump($b);

var\_dump($c);

var\_dump($d);

var\_dump($e);

var\_dump($f);

var\_dump($g);

object(stdClass)#1 (1) {

["scalar"]=>

int(5)

}

object(stdClass)#2 (1) {

["scalar"]=>

float(5.34)

}

object(stdClass)#3 (1) {

["scalar"]=>

string(5) "hello"

}

object(stdClass)#4 (1) {

["scalar"]=>

bool(true)

}

object(stdClass)#5 (0) {

}

object(stdClass)#6 (3) {

[0]=>

string(5) "Volvo"

[1]=>

string(3) "BMW"

[2]=>

string(6) "Toyota"

}

object(stdClass)#7 (3) {

["Peter"]=>

string(2) "35"

["Ben"]=>

string(2) "37"

["Joe"]=>

string(2) "43"

}

# ***5.7- Castear a NULL***

Se castea todo tipo de datos sin problema.

$a = 5;

$a = (unset) $a;

var\_dump($a);

NULL

**6 CONSTANTES**

Una constante es un identificador (nombre) de un valor simple cuyo valor no se puede modificar durante la ejecución del script.

Las constantes son automáticamente globales y pueden usarse en todo el script.

Para crear una se utiliza la función **define():**

**define(name, value);**

* **name:** especifica el nombre de la constante.
* **value:** especifica el valor de la constante.

define("GREETING", "¡Gracias por venir!");

echo GREETING;

¡Gracias por venir!

También se puede crear una constante con la palabra clave **const**.

const MYCAR = "Volvo";

echo MYCAR;

Volvo

La diferencia entre *define()* y *const* es que la primera se puede crear dentro de otro ámbito de bloque, mientras que la segunda no.

# ***6.1- Arrays constantes***

También se pueden crear arrays constantes con la función *define()*:

define("cars", [

"Alfa Romeo",

"BMW",

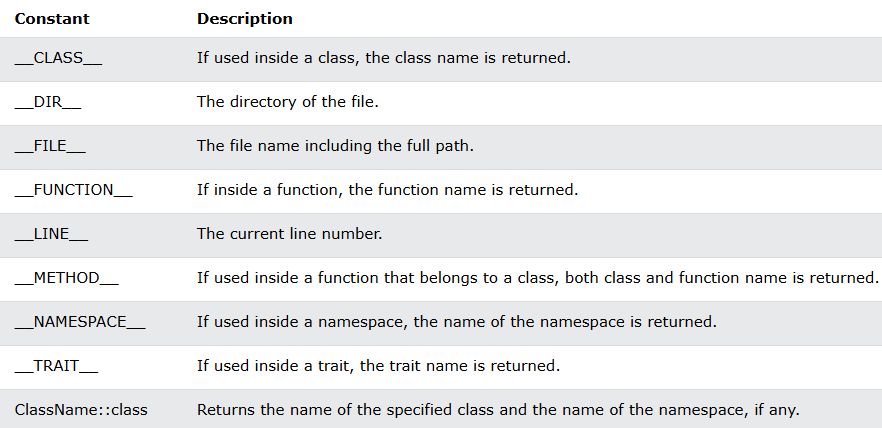
"Toyota"

]);

echo cars[0];

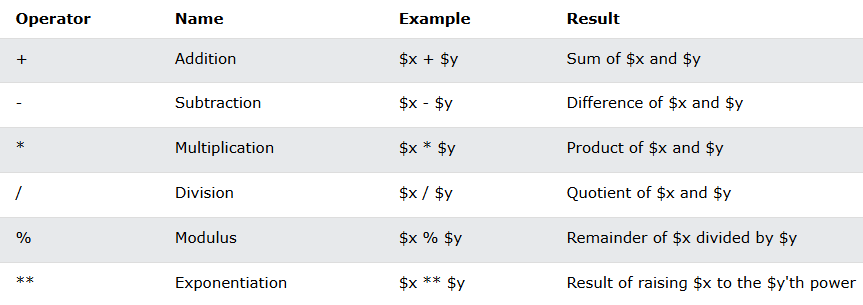
Alfa Romeo

# ***6.2- Constantes mágicas***

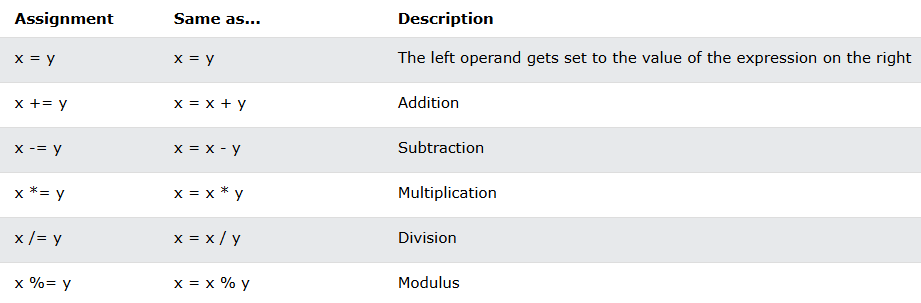


**7 OPERADORES**

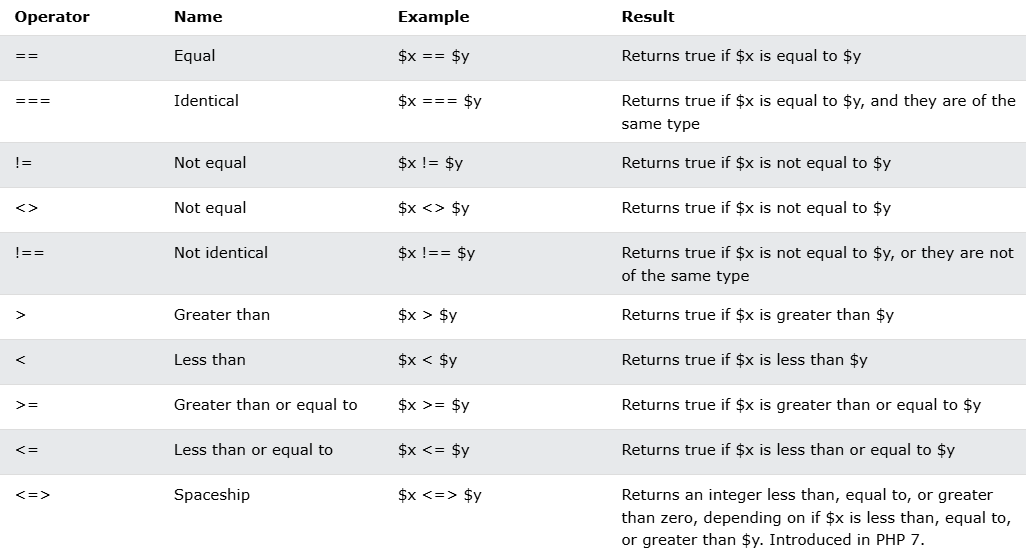
# ***7.1- Operadores aritméticos***



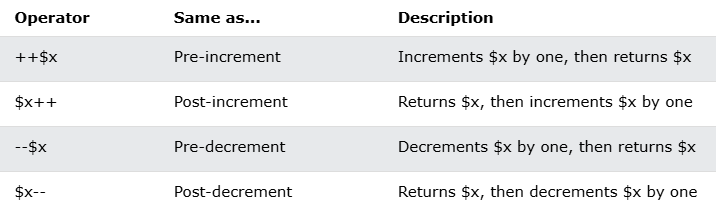
# ***7.2- Operadores de asignación***



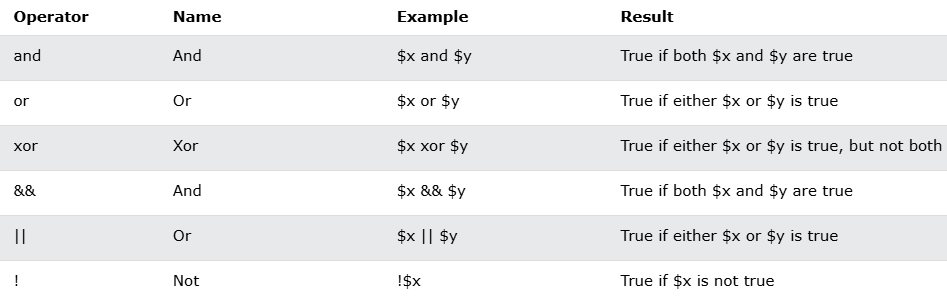
# ***7.3- Operadores de comparación***



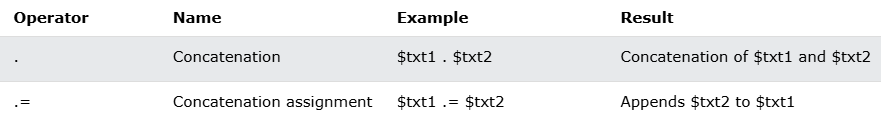
# ***7.4- Operadores de incrementación y decrementación***



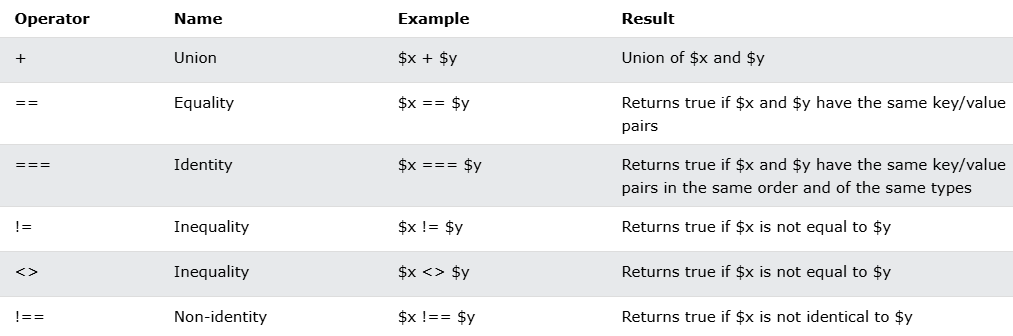
# ***7.5- Operadores lógicos***



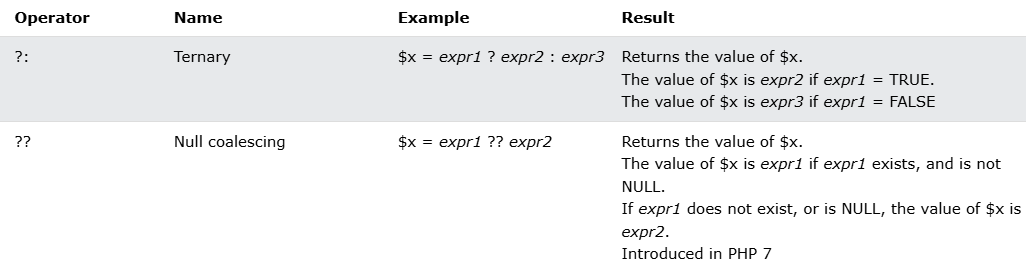
# ***7.6- Operadores para las cadenas de caracteres***



# ***7.7- Operadores de arrays***



# ***7.8- Operadores de asignación condicional***



**8 CONDICIONALES**

# ***8.1- Estructura If***

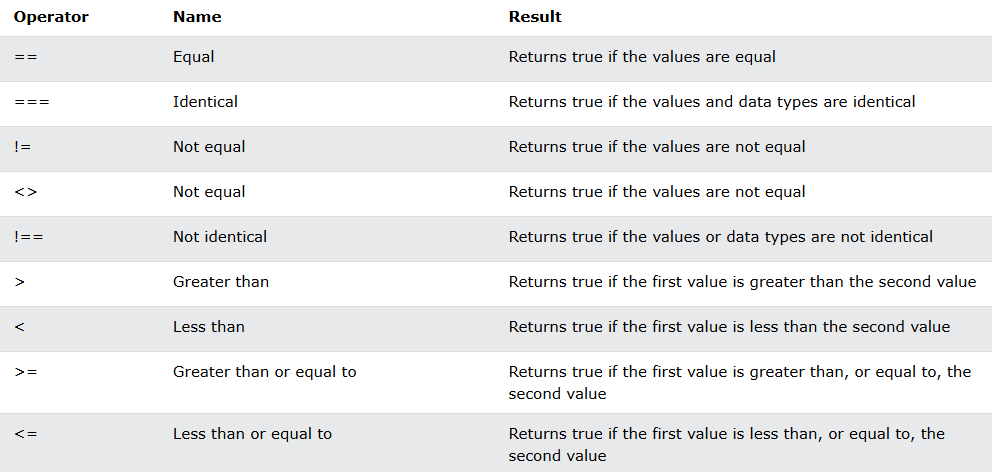
$t = 14;

if ($t < 20) {

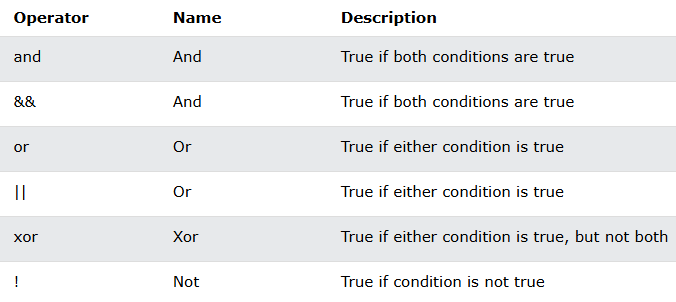
echo "Have a good day!";

}

**Operadores de comparación**



**Operadores lógicos**



# ***8.2- Estructura If… Elseif… Else***

$t = date("H");

if ($t < "10") {

echo "Have a good morning!";

} elseif ($t < "20") {

echo "Have a good day!";

} else {

echo "Have a good night!";

}

# ***8.3- If de una línea y Ternario***

Para escribir un código más corto, escribimos la declaración *“if”* en una sóla línea de la siguiente manera:

$a = 5;

if ($a < 10) $b = "Hello";

Y para escribir un *If… Else”* en una sóla línea utilizamos la estructura **ternaria**:

**condición ? CódigoSeCumpleLaCondición : CódigoNOSeCumpleLaCondición**

$a = 13;

$b = $a < 10 **?** "Hello" **:** "Good Bye";

**+ MÉTODOS ÚTILES**

**var\_dump(var):**devuelve el tipo de una variable.

**strlen(String):** devuelve la longitud de una cadena de caracteres.

**str\_word\_count(String):** cuenta el número de palabras que tiene el String.

**strpos(String):** busca un texto específico dentro de una cadena. Si se encuentra una coincidencia, la función devuelve la posición del primer carácter coincidente. Si no se encuentra ninguna, devolverá *False*.

**is\_int(int):** devuelve si el tipo de la variable es un entero.

(Los métodos **is\_integer(int)** y **is\_long(int)** hacen exactamente lo mismo).

# ***1- Modificar Cadenas***

**strtoupper(String):** devuelve el String en *UPPERCASE*.

**strtolower(String):** devuelve el String en *lowercase*.

**str\_replace("World", "Dolly", “Hello World!”):** reemplaza los caracteres de una cadena por otros. En este caso devolvería: *“Hello Dolly!”*.

**strrev(String):** devuelve la cadena de caracteres invertida.

**trim(“ Hello World ”):** elimina cualquier espacio en blanco del principio o del final de la cadena. En este caso devolvería: *“Hello World”* en lugar de *“ Hello World ”*.

**explode(“ “, “Hello World!”):** divide la cadena en un array según el separador impuesto. En este caso devolvería: *Array( [“Hello”], [“World!”] )*.

# ***2- Segmentar Cadenas (Slicing Strings)***

**substr(“Hello World!”, 6, 5):** devuelve un rango de caracteres de una cadena. Se especifica el índice de inicio (que empieza en 0) y el número de caracteres que se desean devolver a partir de este (este incluido). En este caso devolvería: *“World”*.

* **Segmentar hasta el final (Slice to the end)**

Si se escribe el método sin el último parámetro (el de longitud): **substr(“Hello World!”, 6)**,el rango recorrerá la cadena hasta el final.

* **Segmentar desde el final (Slice from the end)**

Se puede escribir el método utilizando un índice negativo para segmentar la cadena desde su final. Por ejemplo, **substr(“Hello World!”, -5, 3)** devolvería: *“orl”*. Por cierto, EL ÚLTIMO CARACTER TIENE ÍNDICE -1.

* **Segmentar utilizando longitud negativa**

Si se escribe el método asignándole al parámetro de longitud un número negativo, esta especificará cuántos caracteres se omitirán comenzando desde el final de la cadena. Por ejemplo: (**substr(“Hola ¿cómo andas?”, 6, -3)**),en este caso, el rango de la cadena empezará a partir del índice 6 y acabará en el índice -3 (este no se incluirá en el rango) por lo que el resultado sería este: *“¿cómo and”*.

# ***3- Math***

**pi():** devuelve el valor del número pi.

**min(0, 150, 30):** devuelve el menor número de la lista de argumentos. En este caso: 0.

**max(0, 150, 30):** devuelve el mayor número de la lista de argumentos. En este caso: 150.

**abs(int):** devuelve el valor absoluto del número.

**sqrt(int):** devuelve la raíz cuadrada del número.

**round(float):** redondea el número a su entero más cercano.

**rand():** genera un número aleatorio.

* **rand(10, 30):** genera un número aleatorio entre 2 números (estos incluidos).