|  |  |  |
| --- | --- | --- |
| Functions | Description | Signature |
| Strtoupper | It is used to convert all alphabetical characters in a string to uppercase. | string strtoupper ( string $string ) |
| Strtolower | is used to convert all alphabetic characters in a string to lowercase. | string strtolower ( string $string ) |
| Ucfirst | is used to capitalize the first character of a string | ucfirst(string $string): string |
| Ucwords | is used to capitalize the first letter of each word in a string | string ucwords ( string $str ) |
| Substr | is used to extract a portion of a string | substr(string $string, int $start, int|null $length = null): string|false |
| Chunk\_split | is used to split a string into smaller chunks of a specified length and then concatenate them with a specified delimiter | string chunk\_split ( string $body [, int $chunklen = 76 [, string $end = "\r\n" ]] ) |
| Strpos | is used to find the position of the first occurrence of a substring within a string  $haystack: The main string to search in.  $needle: The substring to search for.  $offset (optional): The position in the haystack to start searching. If provided, the search starts from this position. | strpos(string $haystack, string $needle, int $offset = 0): int|false |
| Strstr | is used to find the first occurrence of a substring in a string | strstr(string $haystack, mixed $needle, bool $before\_needle = false): string|false |
| Sprintf | is a function in PHP used for formatting strings  **Format Specifiers:**  The **format** string can contain placeholders (format specifiers) that indicate where the corresponding values should be inserted. Some common format specifiers include:   * **%s**: String * **%d** or **%i**: Integer * **%f**: Floating-point number * **%c**: Character * **%b**: Binary * **%o**: Octal * **%x** or **%X**: Hexadecimal | sprintf(format, arg1, arg2, ...) |
| Str\_shuffle | takes a string as input and randomly shuffles (permutes) its characters. The resulting string will have the same characters as the input string but in a random order. | string str\_shuffle ( string $str ) |
| Preg\_replace | is a PHP function used for performing regular expression-based search and replace on strings. It stands for "Perl-compatible regular expression replacement."  Some important points to note:   1. **Pattern Delimiters**: The pattern is enclosed in delimiters (commonly **/**). For example, **/pattern/**. 2. **Modifiers**: You can include modifiers after the closing delimiter to modify the behavior of the regular expression. For example, **/pattern/i** makes the pattern case-insensitive. 3. **Limit**: You can specify a limit parameter to limit the number of replacements. | preg\_replace(pattern, replacement, subject);  preg\_replace($pattern, $replacement, $originalString, $limit); |
| Explode | allows you to split a string into an array of substrings based on a specified delimiter | array explode ( string $delimiter , string $string [, int $limit = PHP\_INT\_MAX ] ) |
| Basename | is used to extract the filename component from a given path | basename(string $path, string $suffix = ""); |
| ++ | is used for incrementing a variable's value by one. When applied to a character, the ASCII value of the character is incremented by one. |  |
| Substr\_replace | used to replace a portion of a string with another substring. It allows you to replace a specified number of characters in a string with another substring, starting at a specified position. | substr\_replace(string $original, mixed $replacement, int $start, ?int $length = null): string|false |
| Trim | is used to remove whitespace or other specified characters from both the beginning and the end of a string. | string trim ( string $str [, string $character\_mask = " \t\n\r\0\x0B" ] ) |
| Strrpos | used to find the position of the last occurrence of a substring in a given string. The name "strrpos" stands for "string reverse position." It is the counterpart of the strpos function, which finds the position of the first occurrence of a substring. | int strrpos ( string $haystack , string $needle [, int $offset = 0 ] ) |
| Str\_replace | used for replacing occurrences of a search string with another string in a given text. | str\_replace(search, replace, subject, count); |
| Str\_split | is used to split a string into an array of substrings. Each substring consists of a specified number of characters. | str\_split(string $string, int $split\_length = 1): array |
| Ord | is used to get the ASCII value of the first character of a string. ASCII (American Standard Code for Information Interchange) is a character encoding standard that assigns numeric values to different characters. | int ord ( string $string ); int ord ( string $string ); |

**Substr**

$str = "Hello, World!";

// Example 1: Extract a substring starting from position 7

$substring1 = substr($str, 7); // Outputs "World!"

// Example 2: Extract a substring starting from position 0 with a length of 5

$substring2 = substr($str, 0, 5); // Outputs "Hello"

// Example 3: Extract the last 6 characters

$substring3 = substr($str, -6); // Outputs "World!"

// Example 4: Extract a substring starting from position 7 with a length of 3

$substring4 = substr($str, 7, 3); // Outputs "Wor"

// Example 5: Extract a substring starting from position -5 with a length of 2

$substring5 = substr($str, -5, 2); // Outputs "ld"

**chunk\_split**

$inputString = 'ThisIsALongStringThatNeedsToBeSplit';

$chunkedString = chunk\_split($inputString, 4, '-');

echo $chunkedString;

This-IsAL-ongS-trin-gTha-tNee-dsTo-BeSp-lit

**Strstr**

$string = "Hello, world!";

$substring = "world";

$result = strstr($string, $substring);

if ($result) {

echo "Substring found: " . $result;

} else {

echo "Substring not found";

}

**Sprint**

$name = "John";

$age = 30;

$string = sprintf("Hello, my name is %s and I am %d years old.", $name, $age);

echo $string;

// Output: Hello, my name is John and I am 30 years old.

Formatting Numbers

$amount = 12345.6789;

$string = sprintf("The amount is $%.2f", $amount);

echo $string;

// Output: The amount is $12345.68

Padding Numbers:

$number = 42;

$string = sprintf("Padded number: %05d", $number);

echo $string;

// Output: Padded number: 00042

Hexadecimal Representation:

$decimalNumber = 255;

$string = sprintf("Hexadecimal: %X", $decimalNumber);

echo $string;

// Output: Hexadecimal: FF

**Str\_shuffle**

$inputString = "Hello, World!";

$shuffledString = str\_shuffle($inputString);

echo $shuffledString;

**Preg\_replace**

$originalString = "Hello, World!";

$pattern = '/Hello/';

$replacement = 'Hi';

$newString = preg\_replace($pattern, $replacement, $originalString);

echo $newString; // Output: Hi, World!

**Explode**

$string = "apple,orange,banana,grape";

$fruits = explode(",", $string);

print\_r($fruits);

**basename**

// Example 1

$path = "/path/to/file.txt";

$filename = basename($path);

echo $filename; // Output: file.txt

// Example 2

$path = "/path/to/file.txt";

$filename = basename($path, ".txt");

echo $filename; // Output: file

// Example 3

$path = "folder/subfolder/file.txt";

$filename = basename($path);

echo $filename; // Output: file.txt

**++**

$cha = 'A'; // Initialize $cha with the character 'A'

++$cha; // Increment $cha by one

echo $cha; // Output: 'B'

**substr\_replace**

$originalString = "Hello, world!";

$replacement = "Universe";

$startPosition = 7;

$newString = substr\_replace($originalString, $replacement, $startPosition);

echo $newString;

**Trim**

$str = " Hello, World! ";

$trimmed = trim($str);

echo $trimmed; // Output: "Hello, World!"

$str = "###Hello, World!###";

$trimmed = trim($str, "#");

echo $trimmed; // Output: "Hello, World!"

**Strrpos**

$string = "Hello, world! This is a simple example.";

$needle = "is";

$position = strrpos($string, $needle);

if ($position !== false) {

echo "Last occurrence of '$needle' found at position $position.";

} else {

echo "Substring '$needle' not found in the string.";

}

**str\_replace**

$string = "Hello, World!";

$newString = str\_replace("World", "PHP", $string);

echo $newString; // Output: Hello, PHP!

Case-insensitive replacement:

$string = "Hello, world!";

$newString = str\_ireplace("WORLD", "PHP", $string);

echo $newString; // Output: Hello, PHP!

Replacing multiple occurrences:

$string = "apple, banana, cherry, apple";

$newString = str\_replace("apple", "orange", $string);

echo $newString; // Output: orange, banana, cherry, orange

Using arrays:

$search = array("apple", "banana", "cherry");

$replace = array("orange", "grape", "strawberry");

$string = "apple, banana, cherry";

$newString = str\_replace($search, $replace, $string);

echo $newString; // Output: orange, grape, strawberry

Counting replacements:

$string = "apple, banana, cherry, apple";

$newString = str\_replace("apple", "orange", $string, $count);

echo $newString; // Output: orange, banana, cherry, orange

echo "Replacements made: " . $count; // Output: Replacements made: 2

**str\_replace**

$string = "Hello, World!";

str\_replace("World", "PHP", $string, $count);

echo $string; // Output: Hello, World!

**Ord**

$character = 'A';

$asciiValue = ord($character);

echo "The ASCII value of '$character' is $asciiValue.";