String Functions

- ## chr()

The chr() function returns a character from the specified ASCII value.

```
chr(ascii)
```

Note: The x parameter can be specified in decimal, octal, or hex values. Octal values are defined by a leading 0, while hex values are defined by a leading 0x.

Example

```
<?php
echo chr(52)."<br />";
echo chr(052)."<br />";
echo chr(0x52)."<br />";
?>
```

The output of the code above will be:

```
4
*
R
```

- ## ord()

The ord() function returns the ASCII value of the first character of a string.

```
ord(string)
```

Example

```
<?php
echo ord("h")."<br />";
echo ord("hello")."<br />";
?>
```

The output of the code above will be:

```
104
104
```

- ## strtolower()

The strtolower() function converts a string to lowercase.

```
strtolower(string)
```

Example

```
<?php
echo strtolower("Hello WORLD.");
?>
```

The output of the code above will be:

```
hello world.
```

- ## strtoupper()

The strtoupper() function converts a string to uppercase.

```
strtoupper(string)
```

Example

```
<?php
echo strtoupper("Hello WORLD!");
?>
```

The output of the code above will be:

```
HELLO WORLD!
```

- ## strlen()

The strlen() function returns the length of a string.

```
strlen(string)
```

Example

```
<?php
echo strlen("Hello world!");
?>
```

The output of the code above will be:

```
12
```

- ## strcmp()

The strcmp() function compares two strings.

This function returns:

- 0 - if the two strings are equal
- <0 - if string1 is less than string2
- >0 - if string1 is greater than string2

```
strcmp(string1,string2)
```

Parameter	Description
string1	Required. Specifies the first string to compare
string2	Required. Specifies the second string to compare

Note: The strcmp() function is binary safe and case-sensitive.

Example

```
<?php
echo strcmp("Hello world!","Hello world!");
?>
```

The output of the code above will be:

```
0
```

- ## strcasecmp()

The strcasecmp() function compares two strings.

This function returns:

- 0 - if the two strings are equal
- <0 - if string1 is less than string2
- >0 - if string1 is greater than string2

```
strcasecmp(string1,string2)
```

Parameter	Description
string1	Required. Specifies the first string to compare
string2	Required. Specifies the second string to compare

Tip: The strcasecmp() function is binary safe and case-insensitive.

Example

```
<?php
echo strcasecmp("Hello world!","HELLO WORLD!");
?>
```

The output of the code above will be:

```
0
```

- ## ltrim()

The ltrim() function will remove whitespaces or other predefined character from the left side of a string.

```
ltrim(string,charlist)
```

Parameter	Description
string	Required. Specifies the string to check
charlist	Optional. Specifies which characters to remove from the string. If omitted, all of the following characters are removed: • "\0" - NULL • "\t" - tab • "\n" - new line • "\x0B" - vertical tab

	• "\r" - carriage return • " " - ordinary white space

Example

```
<html>
<body>
<?php
$str = "    Hello World!";
echo "Without ltrim: " . $str;
echo "<br />";
echo "With ltrim: " . ltrim($str);
?>
<body>
<html>
```

The browser output of the code above will be:

```
Without ltrim:     Hello World!
With ltrim: Hello World!
```

- rtrim()

The rtrim() function will remove whitespaces or other predefined character from the right side of a string.

- strpos()

The strpos() function returns the position of the first occurrence of a string inside another string.

If the string is not found, this function returns FALSE.

```
strpos(string,find,start)
```

Parameter	Description
string	Required. Specifies the string to search
find	Required. Specifies the string to find
start	Optional. Specifies where to begin the search

Note: The strpos() function is case-sensitive.

Example

```
<?php
echo strpos("Hello world!","wo");
?>
```

The output of the code above will be:

6

- ## stripos()

The stripos() function returns the position of the first occurrence of a string inside another string. If the string is not found, this function returns FALSE.

```
stripos(string,find,start)
```

Parameter	Description
string	Required. Specifies the string to search
find	Required. Specifies the string to find
start	Optional. Specifies where to begin the search

Note: The stripos() function is case-insensitive.

Example

```
<?php
echo stripos("Hello world!","WO");
?>
```

The output of the code above will be:

6

- ## strrpos()

The strrpos() function finds the position of the last occurrence of a string inside another string. This function returns the position on success, otherwise it returns FALSE.

```
strrpos(string,find,start)
```

Parameter	Description
string	Required. Specifies the string to search
find	Required. Specifies the string to find
start	Optional. Specifies where to begin the search

Note: The strrpos() function is case-sensitive.

Example

```
<?php
echo strrpos("Hello world!","wo");
?>
```

The output of the code above will be:

6

- ## strripos()

The strripos() function finds the position of the last occurrence of a string inside another string. This function returns the position on success, otherwise it returns FALSE.

```
strripos(string,find,start)
```

Parameter	Description
string	Required. Specifies the string to search
find	Required. Specifies the string to find
start	Optional. Specifies where to begin the search

Note: The strripos() function is case-insensitive.

Example

```
<?php
echo strripos("Hello world!","WO");
?>
```

The output of the code above will be:

```
6
```

- ## strstr()

The strstr() function searches for the first occurrence of a string inside another string. This function returns the rest of the string (from the matching point), or FALSE, if the string to search for is not found.

```
strstr(string,search)
```

Parameter	Description
string	Required. Specifies the string to search
search	Required. Specifies the string to search for. If this parameter is a number, it will search for the character matching the ASCII value of the number

Note: This function is binary safe. This function is case-sensitive.

Example

```
<?php
echo strstr("Hello world!","world");
?>
```

The output of the code above will be:

```
world!
```

Example 2

In this example we will search a string for the ASCII value of "o":

```
<?php
echo strstr("Hello world!",111);
?>
```

The output of the code above will be:

```
o world!
```

- stristr()

The stristr() function searches for the first occurrence of a string inside another string. This function returns the rest of the string (from the matching point), or FALSE, if the string to search for is not found.

```
stristr(string,search)
```

Parameter	Description
String	Required. Specifies the string to search
Search	Required. Specifies the string to search for. If this parameter is a number, it will search for the character matching the ASCII value of the number

Note: This function is binary safe. This function is case-insensitive.

Example

```
<?php
echo stristr("Hello world!","WORLD");
?>
```

The output of the code above will be:

```
world!
```

Example 2

In this example we will search a string for the ASCII value of "o":

```
<?php
echo stristr("Hello world!",111);
?>
```

The output of the code above will be:

```
o world!
```

- str_replace()

The str_replace() function replaces some characters with some other characters in a string.

This function works by the following rules:

- If the string to be searched is an array, it returns an array
- If the string to be searched is an array, find and replace is performed with every array element
- If both find and replace are arrays, and replace has fewer elements than find, an empty string will be used as replace
- If find is an array and replace is a string, the replace string will be used for every find value

```
str_replace(find,replace,string,count)
```

Parameter	Description
Find	Required. Specifies the value to find
Replace	Required. Specifies the value to replace the value in find
String	Required. Specifies the string to be searched
Count	Optional. A variable that counts the number of replacements

Note: This function is case-sensitive. This function is binary-safe.

Example 1

```
<?php
echo str_replace("world","Peter","Hello world!");
?>
```

The output of the code above will be:

```
Hello Peter!
```

Example 2

In this example we will demonstrate str_replace() with an array and a count variable:

```
<?php
$arr = array("blue","red","green","yellow");
print_r(str_replace("red","pink",$arr,$i));
echo "Replacements: $i";
?>
```

The output of the code above will be:

```
Array
(
[0] => blue
[1] => pink
[2] => green
[3] => yellow
)
Replacements: 1
```

Example 3

In this example we will demonstrate str_replace() with less elements in replace than find:

```
<?php
$find = array("Hello","world");
$replace = array("B");
$arr = array("Hello","world","!");
print_r(str_replace($find,$replace,$arr));
?>
```

The output of the code above will be:

```
Array
(
[0] => B
[1] =>
[2] => !
)
```

- ## strrev()

The strrev() function reverses a string.

```
strrev(string)
```

Example

```
<?php
echo strrev("Hello World!");
?>
```

The output of the code above will be:

```
!dlroW olleH
```

- ## echo()

The echo() function outputs one or more strings.

```
echo(strings)
```

Parameter	Description
Strings	Required. One or more strings to be sent to the output

Note: The echo() function is not actually a function, so you are not required to use parentheses with it. However, if you want to pass more than one parameter to echo(), using parentheses will generate a parse error.

Tip: The echo() function is slightly faster than print(). The echo() function has the following shortcut syntax. See example 5.

Example 1

```
<?php
$str = "Who's Kai Jim?";
echo $str . "<br />";
echo $str."<br />I don't know!";
?>
```

The output of the code above will be:

```
Who's Kai Jim?
Who's Kai Jim?
I don't know!
```

Example 2

```
<?php
echo "This text
spans multiple
lines.";
?>
```

The output of the code above will be:

```
This text spans multiple lines.
```

Example 3

```
<?php
echo 'This ','string ','was ','made ','with multiple parameters';
?>
```

The output of the code above will be:

```
This string was made with multiple parameters
```

Example 4

Difference of single and double quotes. Single quotes will print the variable name, not the value:

```
<?php
$color = "red";
echo "Roses are $color";
echo "<br />";
echo 'Roses are $color';
?>
```

The output of the code above will be:

```
Roses are red
Roses are $color
```

Example 5

```
<html>
<body>
<?php
$color = "red";
?>
<p>Roses are <?=$color?></p>
</body>
</html>
```

- print()

The print() function outputs one or more strings.

```
print(strings)
```

Parameter	Description
Strings	Required. One or more strings to be sent to the output

Note: The print() function is not actually a function, so you are not required to use parentheses with it.

Tip: The print() function is slightly slower than echo().

Example 1

```
<?php
$str = "Who's Kai Jim?";
print $str;
print "<br />";
print $str."<br />I don't know!";
?>
```

The output of the code above will be:

```
Who's Kai Jim?
Who's Kai Jim?
I don't know!
```

Example 2

```
<?php
print "This text
spans multiple
lines.";
?>
```

The output of the code above will be:

```
This text spans multiple lines.
```

Example 3

Difference of single and double quotes. Single quotes will print the variable name, not the value:

```
<?php
$color = "red";
print "Roses are $color";
print "<br />";
print 'Roses are $color';
?>
```

The output of the code above will be:

```
Roses are red
Roses are $color
```

- printf()

The printf() function outputs a formatted string.

The arg1, arg2, ++ parameters will be inserted at percent (%) signs in the main string. This function works "step-by-step". At the first % sign, arg1 is inserted, at the second % sign, arg2 is inserted, etc.

```
printf(format,arg1,arg2,arg++)
```

Parameter	Description
Format	Required. Specifies the string and how to format the variables in it. Possible format values: • %% - Returns a percent sign • %b - Binary number • %c - The character according to the ASCII value • %d - Signed decimal number • %e - Scientific notation (e.g. 1.2e+2) • %u - Unsigned decimal number • %f - Floating-point number (local settings aware) • %F - Floating-point number (not local settings aware) • %o - Octal number • %s - String • %x - Hexadecimal number (lowercase letters) • %X - Hexadecimal number (uppercase letters) Additional format values. These are placed between the % and the letter (example %.2f): • + (Forces both + and - in front of numbers. By default, only negative numbers are marked) • ' (Specifies what to use as padding. Default is space. Must be used together with the width specifier. Example: %'x20s (this uses "x" as padding) • - (Left-justifies the variable value) • [0-9] (Specifies the minimum width held of to the variable value) • .[0-9] (Specifies the number of decimal digits or maximum string length) Note: If multiple additional format values are used, they must be in the same order as above.
Arg1	Required. The argument to be inserted at the first %-sign in the format string
Arg2	Optional. The argument to be inserted at the second %-sign in the format string
Arg++	Optional. The argument to be inserted at the third, fourth, etc. %-sign in the format string

Note: If there are more % signs than arguments, you must use placeholders. A placeholder is inserted after the % sign, and consists of the argument- number and "\$". See example three.

Example 1

```
<?php
$str = "Hello";
$number = 123;
printf("%s world. Day number %u",$str,$number);
?>
```

The output of the code above will be:

```
Hello world. Day number 123
```

Example 2

```
<?php
$number = 123;
printf("%f",$number);
?>
```

The output of the code above will be:

```
123.000000
```

Example 3

Use of placeholders:

```
<?php
$number = 123;
printf("With 2 decimals: %1\$.2f
<br />With no decimals: %1\$u",$number);
?>
```

The output of the code above will be:

```
With 2 decimals: 123.00
With no decimals: 123
```