**Strings in PHP:** Definition, Creating, Declaring, formatting strings, String Functions.

**Strings in php:**

**What is string?**

In PHP, a string is a sequence of characters, where a character is a single unit of information. Strings can contain letters, numbers, symbols, and spaces. PHP provides various functions and features for working with strings, making it versatile for tasks like text processing, manipulation, and output.
In PHP, you can define strings using either **single quotes (') or double quotes (").**

Here are examples of string declarations:

```
$singleQuotedString = 'This is a string with single quotes.';
$doubleQuotedString = "This is a string with double quotes.";
```

Both single-quoted and double-quoted strings can contain variables, and variables will be expanded (interpreted) inside double-quoted strings but not within single-quoted strings. Here's an example:

```
$name = "John";
$greeting1 = 'Hello, $name!'; // Output: Hello, $name!
$greeting2 = "Hello, $name!"; // Output: Hello, John!
```

In the example above, variable interpolation (expanding variables within the string) only occurs in the double-quoted string.

PHP also provides numerous string functions for tasks such as concatenation, finding substrings, replacing text, converting case, and more. Here are a few examples:

```
$string1 = "Hello";
$string2 = "World";

// String length
$length = strlen($result); // Output: 11

// Uppercase and lowercase
$upper = strtoupper($result); // Output: HELLO WORLD
$lower = strtolower($result); // Output: hello world
```

Strings in PHP are versatile and widely used in web development for tasks such as handling user input, generating HTML content, working with databases, and more.

**Creating a string:**

In PHP, you can create strings using either single quotes (') or double quotes ("). Here are examples of both:

- Single-Quoted String:

```
$singleQuotedString = 'This is a string with single quotes.';
```

In single-quoted strings, variables are not expanded (interpreted). For example:

```
$name = "John";
$greeting = 'Hello, $name!'; // Output: Hello, $name!
```

- Double-Quoted String:

```
$doubleQuotedString = "This is a string with double quotes.";
```

Double-quoted strings allow variable interpolation, meaning variables are expanded inside the string. For example:

```
$name = "John";
$greeting = "Hello, $name!"; // Output: Hello, John!
```

- Heredoc and Nowdoc Syntax:

PHP also provides two additional syntaxes for creating strings:

- Heredoc Syntax:

Heredoc is useful for creating multiline strings:

```
$heredocString = <<<EOT
This is a heredoc string.
It can span multiple lines.
Variables, like $name, are expanded.
EOT;
```

- Nowdoc Syntax:

Nowdoc is similar to heredoc but treats the contents as a plain string without variable expansion:

```
$nowdocString = <<<'EOT'
This is a nowdoc string.
It can also span multiple lines.
Variables, like $name, are not expanded.
EOT;
```

Choose the string syntax that suits your needs. Single quotes are often used when there is no need for variable expansion, while double quotes are used when variables need to be interpolated. Heredoc and nowdoc are useful for creating multiline strings in a clean and readable way.

**String functions:**

PHP provides a variety of built-in string functions that allow you to manipulate and work with strings. Here are some commonly used string functions in PHP:

**strlen():** strlen() function returns the length of a string.

**Example**

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

**Output:** 12

**str_word_count():** str_word_count() function is used to count numbers of words in given string.

**Example**

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

**Output:** 2

**strrev():** strrev() function is used to revers any string.

**Example**

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

**Output:** !dlrow olleH

**strtolower():** strtolower() function is used to convert uppercase latter into lowercase latter.

**Example**

```
<?php
$str="Hello friends i am HITESH";
$str=strtolower($str);
echo $str;
?>
```

**Output:** hello friends i am hitesh

**strtoupper():** PHP strtoupper() function is used to convert lowercase latter into uppercase latter.

**Example**

```
<?php
$str="Hello friends i AM Hitesh";
$str=strtoupper($str);
echo $str;
?>
```

**Output:** HELLO FRIENDS I AM HITESH

**ucwords():** ucwords() function is used to convert first letter of every word into upper case.

**Example**

```
<?php
```

```
$str="hello friends i am hitesh";
$str=ucwords($str);
echo $str;
?>
```

**Output:** Hello Friends I Am Hitesh

**ucfirst():** ucfirst() function returns string converting first character into uppercase. It doesn't change the case of other characters.

**Example**

```
<?php
function firstUpper($string)
{
return(ucfirst($string));
}
$string="welcome to GeeksforGeeks";
echo (firstUpper($string));
?>
```

**Output:** Welcome to GeeksforGeeks

**lcfirst():** lcfirst() function is used to convert first character into lowercase. It doesn't change the case of other characters.

**Example**

```
<?php
function firstLower($string)
{
return(lcfirst($string));
}
$string="WELCOME to GeeksforGeeks";
echo (firstLower($string));
?>
```

**Output:** wELCOME to GeeksforGeeks

**strstr()(**or) **strchr():** The strchr() function searches for the first occurrence of a string inside another string.

**Syntax:** strchr(string,search)

**Example:**

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

**output:** world!

**Example:** Search a string for the ASCII value of "o" and return the rest of the

string:

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

**Output:** o world!

**Example:**

```
<?php
$originalStr = "geeks for geeks";
$searchStr = "geeks" ;
echo strchr($originalStr, $searchStr);
?>
```

**Output:** geeks for geeks

**Program 2:** Program to demonstrate strchr() fucntion when word is not found.

```
<?php
$originalStr = "geeks for geeks";
$searchStr = "gfg" ;
echo strchr($originalStr, $searchStr);
?>
```

**Output:** -NIL**strcmp():**

**Compare two strings with case-sensitive manner**

**Syntax:** strcmp(string1,string2)

**This function returns:**

0 - if the two strings are equal

<0 - if string1 is less than string2

>0 - if string1 is greater than string2

**Example:**

```
<?php
echo strcmp("Hello","Hello");
echo "<br>";
echo strcmp("Hello","hELLo");
?>
```

**Output:** 0

-1

**Example:**

```
<?php
echo strcmp("Hello world!","Hello world!")."<br>";
echo strcmp("Hello world!","Hello")."<br>"; // string1 is greater than string2
echo strcmp("Hello world!","Hello world! Hello!")."<br>"; // string1 is less than
```

string2
?>
**Output:** 0
7
-7

**strcasecmp():**Compare two strings with case-insensitive manner
**Syntax:** strcasecmp(string1,string2)
**This function returns:**
0 - if the two strings are equal
<0 - if string1 is less than string2
>0 - if string1 is greater than string2
**Example:**
<?php
echo strcasecmp("Hello","HELLO");
echo strcasecmp("Hello","hELLo");
?>
**Output:** 0
0
**Example:**
<?php
echo strcasecmp("Hello world!","HELLO WORLD!"); // The two strings are equal
echo strcasecmp("Hello world!","HELLO"); // String1 is greater than string2
echo strcasecmp("Hello world!","HELLO WORLD! HELLO!"); // String1 is less
than string2
?>
**Output:** 0
7
-7

**strncmp()**: The strncmp() is used to compare first n character of two strings. This function is case-sensitive which points that capital and small cases will be treated differently, during comparison.
**Syntax:** strncmp( $str1, $str2, $len )
This function returns:
0 - if the two strings are equal
<0 - if string1 is less than string2
>0 - if string1 is greater than string2
**Example:**
<?php
$str1 = "Geeks for Geeks ";
$str2 = "Geeks for Geeks ";

```
// Both the strings are equal
$test=strncmp($str1, $str2, 16 );
echo "$test";
?>
```

**Output:** 0

**Example:**

```
?php
// Input strings
$str1 = "Geeks for Geeks ";
$str2 = "Geeks for ";
$test=strncmp($str1, $str2, 16 );
// In this case the second string is smaller
echo "$test";
?>
```

**Output:** 6

**Example:**

```
<?php
// Input Strings
$str1 = "Geeks for ";
$str2 = "Geeks for Geeks ";
$test=strncmp($str1, $str2, 16 );
// In this case the first string is smaller
echo "$test";
?>
```

**Output:** -6

**strncasecmp()**: The strncmp() is used to compare first n character of two strings. This function is case-sensitive which points that capital and small cases will be treated differently, during comparison.

**Syntax:** strncasecmp( $str1, $str2, $len )

This function returns:

0 - if the two strings are equal

<0 - if string1 is less than string2

>0 - if string1 is greater than string2

**Example:**

```
<?php
$str1 = "Geeks for Geeks ";
$str2 = "Geeks for Geeks ";
// Both the strings are equal
$test=strncasecmp($str1, $str2, 16 );
```

```
echo "$test";
?>
```
**Output :** 0

**Example:**
```
?php
// Input strings
$str1 = "Geeks for Geeks ";
$str2 = "Geeks for ";
$test=strncasecmp($str1, $str2, 16 );
// In this case the second string is smaller
echo "$test";
?>
```
**Output:** 6

**Example:**
```
<?php
// Input Strings
$str1 = "Geeks for ";
$str2 = "Geeks for Geeks ";
$test=strncasecmp($str1, $str2, 16 );
// In this case the first string is smaller
echo "$test";
?>
```
**Output:** -6

**strpos()** It enables searching particular text within a string. It works simply by matching the specific text in a string. If found, then it returns the specific position. If not found at all, then it will return "False".

**Syntax:** Strpos(string,text);

**Example**
```
<?php
echo strpos("Welcome to Cloudways","Cloudways");
?>
```
**Output:** 11

**Str_replace()**: It used for replacing specific text within a string.

**Syntax:** Str_replace(string to be replaced,text,string)

**Example**
```
<?php
echo str_replace("cloudways", "the programming world", "Welcome to
```

cloudways”);
?>
**Output:** Welcome to the programming world

**str_repeat():** This function is used for repeating a string a specific number of times.
**Syntax:** Str_repeat(string,repeat)
**Example**
<?php
echo str_repeat(“=”,13);
?>
**Output:** ==========

**substr():** This function you can display or extract a string from a particular position.
**Syntax: substr(string,start,length)**
**Example**
<?php
echo substr(“Welcome to Cloudways”,6).”<br>”;
echo substr(“Welcome to Cloudways”,0,10).”<br>”;
?>
**Output:** e to Cloudways
Welcome to

**chunk_split():**The chunk_split() function splits a string into a series of smaller parts.
**Syntax:** chunk_split(string,length,end)
**Example:** Split the string after each sixth character and add a "..." after each split:
<?php
$str = "Hello world!";
echo chunk_split($str,6,"...");
?>
**Output:** Hello ...world!...

**trim()** :Removes whitespace or other characters from both sides of a string 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:**

```
<?php
echo trim(" testing ").”<br>”;
echo trim(" testing ", " teng");
?>
```

**Output:** testing

**stiltrim()**: Removes whitespace or other characters from the left side of a string

**Syntax:** ltrim(string,charlist)

```
<?php
$str = "Hello World!";
echo $str . "<br>";
echo ltrim($str,"Hello");
?>
```

**Output:** Hello World!
World!

**rtrim()**: Removes whitespace or other characters from the right side of a string

**Syntax:** rtrim(string,charlist)

```
<?php
$str = "Hello World!";
echo $str . "<br>";
echo rtrim($str,"World!");
?>
```

**Output:** Hello World!
Hello

**Write a PHP program to count number of characters in a given string.**

```
<?php
$text="w3resource";
$search_char="r";
$count="0";
for($x="0"; $x< strlen($text); $x++)
{
$count=$count+1;
}
echo $count."\n";
?>
```

**List Of Functions in PHP**

| Function | Description |
|---|---|
| chop() | Removes whitespace or other characters from the right end of a string |
| chr() | Returns a character from a specified ASCII value |
| count_chars() | Returns information about characters used in a string |
| echo() | Outputs one or more strings |
| explode() | Breaks a string into an array |
| fprintf() | Writes a formatted string to a specified output stream |
| parse_str() | Parses a query string into variables |
| print() | Outputs one or more strings |
| printf() | Outputs a formatted string |
| sprintf() | Writes a formatted string to a variable |
| sscanf() | Parses input from a string according to a format |
| str_getcsv() | Parses a CSV string into an array |
| str_ireplace() | Replaces some characters in a string (case-insensitive) |
| str_pad() | Pads a string to a new length |
| str_replace() | Replaces some characters in a string (case-sensitive) |
| str_shuffle() | Randomly shuffles all characters in a string |
| str_split() | Splits a string into an array |
| strcspn() | Returns the number of characters found in a string before any part of some specified characters are found |
| stripos() | Returns the position of the first occurrence of a string inside another string (case-insensitive) |
| stristr() | Finds the first occurrence of a string inside another string (case-insensitive) |
| strrchr() | Finds the last occurrence of a string inside another string |
| strripos() | Finds the position of the last occurrence of a string inside another string (case-insensitive) |
| strrpos() | Finds the position of the last occurrence of a string inside another string (case-sensitive) |
| strspn() | Returns the number of characters found in a string that contains only characters from a specified charlist |
| substr() | Returns a part of a string |
| substr_compare() | Compares two strings from a specified start position (binary safe and optionally case-sensitive) |
| substr_count() | Counts the number of times a substring occurs in a string |
| substr_replace() | Replaces a part of a string with another string |