Strings

String is a series of character or set of character. When a number is placed into the single or double quote then it's also a string so it can't be used in arithmetic. PHP provides a lot of pre-define function for string manipulation like strlen, strrev, htmlspecialchars etc.

Some Valid String Are:

"hello"
"1245"
"436^%$"
"hello and welcome to php"

String Example

```
<?php
$str =" And welcome to php string " ;
echo " $str "."<br>";
echo " Hello, $str! " . "<br>" ;
echo " Hello, $str! " . "<br>";
echo " hi friends" . "<br>";
echo " length:-".strlen($str);
?>
```

Creating and Accessing Strings

creating a string variable is as simple as assigning a literal string value to a new variable name. Both single and double quote are used to create string.

Syntax

```
$var="swati";
$my_variable = " hello " ;
```

Illustrate With Example

```
<?php
$myvariable = " Welcome to php string " ;
echo "$myvariable "."<br>";
echo 'Example of php';
?>
```

Creating More Complex String Expression

```
<?php
$car = "Ferrari 458";
echo "My favorite car is $car"."<br>";
```

```
echo "My favourite car is ${car}"."<br>";
echo "My favourite car is ${car}"."<br>";
?>
```

Searching Strings With Strstr() Function

strstr() function is used to find out whether some text occurs within a string or not ,if string match print all the string from where string matched. If word is not found strstr function return false.

Syntax Of Strstr() Function

Strstr($expression ,search text);

Strsrt() Example

```
<?php
$string1= "Hello world!";
echo strstr( $string1, "H" )."<br>";
echo ( strstr( $string1, "xyz" ) ? "Yes" : "No" ) ."<br>";
$string2= "Welcome to php string";
echo strstr( $string1, "java" ) ."<br>"; //nothing print
echo ( strstr( $string2, "php" ) ? "Yes" : "No" ) ."<br>";
?>
```

Accessing Characters within a String

To access a character at a particular position.

Syntax

$char = $str [position] ;

String Example

```
<?php
$myStr = "Welcome to the php string";
echo $myStr[0] ."<br>"; // print "W"
echo $myStr[6] ."<br>"; // print "e"
$myStr[25] = '?';//Welcome to the php string?
echo $myStr ."<br>";
?>
```

Searching & Replacing String

str_replace() replaces one parts of a string with new parts you specify. Str_replace() takes a minimum of three parameters: what to look for, what to replace it with, and the string to work with. It also has an optional fourth parameter, which, if passed, will be filled with the number of replacements made.

Str_replace() is a very easy way to find and replace text in a string.

Example

```
<?php
$string = "An infinite number of stars";
$newstring = str_replace("stars", "plants", $string);
print $newstring;
?>
```

Formatting String

When you are using PHP to put a string on the page most of the time you will use the syntax echo, which will take the following string and display that string in HTML.
You can even concatenate multiple strings or variables together to be outputted by the echo syntax.
Example of using echo to output multiple options in a select box.

```
<?php
$options = array('option1' => 'title1',
'option2' => 'title2',
'option3' => 'title3',
'option4' => 'title4',
'option5' => 'title5'
);
echo '<select>';
foreach($options as $key => $val)
{
echo '<option value="'.$key.'">'.$val.'</option>';
}
echo '</select>';
?>
```

String Related Library functions

Function(Parameters)	Description
echo(string)	Outputs Strings
print(string)	Outputs a String
printf(string)	Outputs a Formatted String
ltrim(string)	Strips Whitespace From the Left Side of a String
rtrim(string)	Strips Whitespace From the Right Side of a String
trim(string)	Strips Whitespace From Both Sides of a String
lcfirst(string)	Makes a String's First Character Lowercase
ucfirst(string)	Makes a String's First Character Uppercase
strtolower(string)	Converts a String to Lowercase Letters
strtoupper(string)	Converts a String to Uppercase Letters
str_word_count(string)	Count the Number of Words In a String
ucwords(string)	Makes the First Character of Each Word In a String Uppercase
wordwrap(string, width, break)	Wraps a String to a Given Number of Characters (Default Width: 75) (Default Break: \n)
count_chars(string)	Returns How Many Times an ASCII Character Occurs Within a String & Returns the Information
substr_count(string, substring)	Counts the Number of Times a Substring Occurs In a String
str_pad(string, length, pad_string)	Pads a String to a New Length
strlen(string)	Returns the Length of a String
substr(string, start)	Returns a Part of a String (Start Value of "0" to Begin at First Character)
strrev(string)	Reverses a String
str_shuffle(string)	Randomly Shuffles All Characters In a String
str_repeat(string, repeat)	Repeats a String a Specified Number of Times ("Repeat" Is Number of Times to Repeat)
strpbrk(string, characters)	Searches a String For Any of a Set of Characters
str_replace(find, replace, string)	Replaces Some Characters In a String (Case-Sensitive)
substr_replace(string, replacement, start)	Replaces a Part of a String With Another String
stristr(string, search)	Finds the First Occurrence of a String Inside Another String (Case-Insensitive)
strstr(string, search)	Finds the First Occurrence of a String Inside Another String (Case-Sensitive)
strrchr(string, char)	Finds the Last Occurrence of a String Inside Another String

stripos(string, find)	Returns the Position of the First Occurrence of a String Inside Another String (Case-Insensitive)
strpos(string, find)	Returns the Position of the First Occurrence of a String Inside Another String (Case-Sensitive)
strripos(string, find)	Returns the Position of the Last Occurrence of a String Inside Another String (Case-Insensitive)
strrpos(string, find)	Returns the Position of the Last Occurrence of a String Inside Another String (Case-Sensitive)
strcasecmp(string1, string2)	Compares Two Strings (Case-Insensitive)
strcmp(string1, string2)	Compares Two Strings (Case-Sensitive)
strtok(string, split)	Splits a String Into Smaller Strings
chunk_split(string, length)	Splits a String Into a Series of Smaller Parts (Default Length Is 76)
str_split(string, length)	Splits a String Into an Array
explode(separator, string)	Breaks a String Into an Array
implode(separator, array)	Returns a String From the Elements of an Array
str_getcsv(string, delimiter, enclosure)	Parses a CSV String Into an Array
addcslashes(string)	Returns a String With Backslashes In Front of Single Quotes, Double Quotes & Backslashes
stripcslashes(string)	Unquotes a String Quoted With addcslashes()
addslashes(string,characters)	Returns a String With Backslashes in Front of Predefined Characters
stripslashes(string)	Unquotes a String Quoted With addslashes()
nl2br(string)	Inserts HTML Line Breaks In Front of Each Newline In a String
strip_tags(string)	Strips HTML & PHP Tags From a String
html_entity_decode(string)	Converts HTML Entities to Characters
htmlentities(string)	Converts Characters to HTML Entities
htmlspecialchars_decode(string)	Converts Some Predefined HTML Entities to Characters
htmlspecialchars(string)	Converts Some Predefined Characters to HTML Entities
get_html_translation_table()	Returns the Translation Table Used by htmlspecialchars() & htmlentities()

Regular Expression

Regular expressions are nothing more than a sequence or pattern of characters. They provide the foundation for pattern-matching functionality.

Using regular expression you can search a particular string inside a string, you can replace one string by another string and you can split a string into many chunks.

PHP offers functions specific to two sets of regular expression functions, each corresponding to a certain type of regular expression. You can use any of them based on your comfort.

- POSIX Regular Expressions
- PERL Style Regular Expressions

POSIX Regular Expressions

The structure of a POSIX regular expression is not dissimilar to that of a typical arithmetic expression: various elements (operators) are combined to form more complex expressions.

The simplest regular expression is one that matches a single character, such as g, inside strings such as g, haggle, or bag.

POSIX regular expression:-

Brackets

Brackets ([]) have a special meaning when used in the context of regular expressions. They are used to find a range of characters.

Sr.No	Expression & Description
1	**[0-9]** It matches any decimal digit from 0 through 9.
2	**[a-z]**

	It matches any character from lower-case a through lowercase z.
3	**[A-Z]** It matches any character from uppercase A through uppercase Z.
4	**[a-Z]** It matches any character from lowercase a through uppercase Z.

The ranges shown above are general; you could also use the range [0-3] to match any decimal digit ranging from 0 through 3, or the range [b-v] to match any lowercase character ranging from b through v.

Quantifiers

The frequency or position of bracketed character sequences and single characters can be denoted by a special character. Each special character having a specific connotation. The +, *, ?, {int. range}, and $ flags all follow a character sequence.

Sr.No	Expression & Description
1	**p+** It matches any string containing at least one p.
2	**p*** It matches any string containing zero or more p's.
3	**p?** It matches any string containing zero or more p's. This is just an alternative way to use p*.

4	**p{N}** It matches any string containing a sequence of **N** p's
5	**p{2,3}** It matches any string containing a sequence of two or three p's.
6	**p{2, }** It matches any string containing a sequence of at least two p's.
7	**p$** It matches any string with p at the end of it.
8	**^p** It matches any string with p at the beginning of it.

Examples

Following examples will clear your concepts about matching characters.

Sr.No	Expression & Description
1	**[^a-zA-Z]** It matches any string not containing any of the characters ranging from a through z and A through Z.
2	**p.p** It matches any string containing p, followed by any character, in turn followed by another p.

3	**^.{2}$** It matches any string containing exactly two characters.
4	**(.*)** It matches any string enclosed within and .
5	**p(hp)*** It matches any string containing a p followed by zero or more instances of the sequence php.

Predefined Character Ranges

For your programming convenience several predefined character ranges, also known as character classes, are available. Character classes specify an entire range of characters, for example, the alphabet or an integer set −

Sr.No	Expression & Description
1	**[[:alpha:]]** It matches any string containing alphabetic characters aA through zZ.
2	**[[:digit:]]** It matches any string containing numerical digits 0 through 9.
3	**[[:alnum:]]** It matches any string containing alphanumeric characters aA through zZ and 0 through 9.

4	**[[:space:]]** It matches any string containing a space.

PHP's Regexp POSIX Functions

PHP currently offers seven functions for searching strings using POSIX-style regular expressions −

Sr.No	Function & Description
1	**ereg()** The ereg() function searches a string specified by string for a string specified by pattern, returning true if the pattern is found, and false otherwise.
2	**ereg_replace()** The ereg_replace() function searches for string specified by pattern and replaces pattern with replacement if found.
3	**eregi()** The eregi() function searches throughout a string specified by pattern for a string specified by string. The search is not case sensitive.
4	**eregi_replace()** The eregi_replace() function operates exactly like ereg_replace(), except that the search for pattern in string is not case sensitive.
5	**split()** The split() function will divide a string into various elements, the boundaries of each element based on the occurrence of pattern in string.

6	**spliti()** The spliti() function operates exactly in the same manner as its sibling split(), except that it is not case sensitive.
7	**sql_regcase()** The sql_regcase() function can be thought of as a utility function, converting each character in the input parameter string into a bracketed expression containing two characters.

PHP's Regexp PERL Compatible Functions

PHP offers following functions for searching strings using Perl-compatible regular expressions –

Sr.No	Function & Description
1	**preg_match()** The preg_match() function searches string for pattern, returning true if pattern exists, and false otherwise.
2	**preg_match_all()** The preg_match_all() function matches all occurrences of pattern in string.
3	**preg_replace()** The preg_replace() function operates just like ereg_replace(), except that regular expressions can be used in the pattern and replacement input parameters.
4	**preg_split()** The preg_split() function operates exactly like split(), except that regular

	expressions are accepted as input parameters for pattern.
5	**preg_grep()** The preg_grep() function searches all elements of input_array, returning all elements matching the regexp pattern.
6	**preg_ quote()** Quote regular expression characters

Functions

A PHP function is a piece of code which takes one or more input in the form of parameter and does some processing and returns a value.

There are two parts of a function −

- Creating a PHP Function
- Calling a PHP Function

Creating PHP Function

It's very easy to create your own PHP function. Suppose we want to create a PHP function which will simply write a simple message on your browser when you will call it. Following example creates a function called writeMessage() and then calls it just after creating it.

Note that while creating a function its name should start with keyword **function** and all the PHP code should be put inside { and } braces as shown in the following example below −

```
<html>

  <head>
    <title>Writing PHP Function</title>
  </head>

  <body>

    <?php
      /* Defining a PHP Function */
      function writeMessage() {
        echo "You are really a nice person, Have a nice time!";
      }

      /* Calling a PHP Function */
      writeMessage();
```

```
   ?>

  </body>
</html>
```

This will display following result −

```
You are really a nice person, Have a nice time!
```

PHP Functions with Parameters

PHP gives us option to pass your parameters inside a function. You can pass as many as parameters you like. These parameters work like variables inside your function. Following example takes two integer parameters and adds them together and then print them.

```
<html>

  <head>
    <title>Writing PHP Function with Parameters</title>
  </head>

  <body>

    <?php
      function addFunction($num1, $num2) {
        $sum = $num1 + $num2;
        echo "Sum of the two numbers is : $sum";
      }

      addFunction(10, 20);
    ?>

  </body>
```

```
</html>
```

This will display following result −

```
Sum of the two numbers is : 30
```

Passing Arguments by Reference

It is possible to pass arguments to functions by reference. This means that a reference to the variable is manipulated by the function rather than a copy of the variable's value.

Any changes made to an argument in these cases will change the value of the original variable. You can pass an argument by reference by adding an ampersand to the variable name in either the function call or the function definition.

Following example depicts both the cases.

```
<html>

  <head>
    <title>Passing Argument by Reference</title>
  </head>

  <body>

    <?php
      function addFive($num) {
        $num += 5;
      }

      function addSix(&$num) {
        $num += 6;
      }
```

```
      $orignum = 10;
      addFive( $orignum );

      echo "Original Value is $orignum<br />";

      addSix( $orignum );
      echo "Original Value is $orignum<br />";
   ?>

  </body>
</html>
```

This will display following result −

```
Original Value is 10
Original Value is 16
```

PHP Functions returning value

A function can return a value using the **return** statement in conjunction with a value or object. Return stops the execution of the function and sends the value back to the calling code.

You can return more than one value from a function using **return array(1,2,3,4)**.

Following example takes two integer parameters and add them together and then returns their sum to the calling program. Note that **return** keyword is used to return a value from a function.

```
<html>

  <head>
    <title>Writing PHP Function which returns value</title>
  </head>

  <body>
```

```
    <?php
      function addFunction($num1, $num2) {
        $sum = $num1 + $num2;
        return $sum;
      }
      $return_value = addFunction(10, 20);

      echo "Returned value from the function : $return_value";
    ?>

  </body>
</html>
```