Unit 5.3: Working and formattingwith strings.

1. Unit Structure

- Learning Objectives
- Introduction
- Difference between Single and Double Quoted Strings
- String Handling Functions

2. Learning Objectives

After completion of this unit students will be able to:

- Understand the difference between single and double quoted strings.
- Handle the string data in PHP.
- Format the string data as per requirements.

3. Introduction

In web applications, there are certain occasions where we need to store text data for further processing such as to store name of person, city, address, name of product, description etc. To store all these types of information, strings are used.

PHP string is a sequence of characters which is used to store and manipulate text. Normally PHP string is represented using single and double quoted string.

Example:

$name= “James Dean”;

Or

$name=’James Dean’;

Both are valid representations.

4. Difference between Single and Double Quoted Strings

However, there is a difference between single and double quoted strings in PHP where we are combining string with variable to print any message. Consider the following example:

```php
<?php
$name="James";
Echo "My name is $name";
//output is "My name is James"
$name1="James";
Echo 'My name is $name';
//output is "My name is $name"
?>
```

In double quoted string the variable is replaced with its value, but it is not replaced in single quoted string.

5. String Handling Functions

Sometimes we need to perform certain operations on the string values for better results such as converting the name of person into lower or upper case, concatenating two strings, replacing certain parts of string, counting number of characters and words of the string etc. All these tasks require certain string handling functions. Let's us discuss some of the important string handling functions in brief.

5.1. **strlen()** – returns the length (number of characters) of the string.

```php
<?php
$name="Hello";
$len=strlen($name);
echo "Length of the string is $len";
?>
```

Output: Length of the string is 5

5.2. str_word_count()

It counts the number of words in the given string.

```php
<?php
$name="Hello How Are You";
$count=str_word_count($name);
echo "Total words of the string is $count";
?>
```

Output: Total words of the string is 4

5.3. strtolower()

It converts the string in to lowercase. It Returns the lowercased string. Original string will remain unchanged.

```php
<?php
$name="HELLO";
echo strtolower($name);
?>
```

Output: hello

5.4. strtoupper()

Converts the string into upper case. This function is useful where you want to store all the names of persons in upper case. Consider the scenario where you have designed customer registration page where user is supposed to enter his or her name in upper case but some of the users are still entering their names in lower case. Here you can convert the names in to upper case with the use of strtoupper() before storing them in to your data base.

```php
<?php
$name="hello";
echo strtoupper($name);
?>
```

Output: HELLO

5.5. ucfirst()

Converts the first character of the string in to uppercase.

```php
<?php
$name="hello";
echo ucfirst($name);
?>
```

Output: Hello

5.6. lcfirst()

It converts the first character of the string into lower case.

```php
<?php
$name="Hello";
echo lcfirst($name);
?>
```

Output: hello

5.7. ucwords()

Converts the first character of each word of string in to uppercase.

```php
<?php
$name="my name is Parwez";
echo ucwords($name);
?>
```

Output: My Name Is Parwez

5.8. strrev()

Reverse the string.

```php
<?php
$name="Parwez";
echo strrev($name)."<p>";
echo ucwords($name);
?>
```

Output:

zewraP

Parwez

strcmp()

Compares two strings. Returns 0 if both the strings are same. It is case sensitive (Parwez and parwez are not same)

```php
<?php
$name1="Parwez";
$name2="parwez";
if(strcmp($name1,$name2)==0)
echo "<br> Both are same ";
else
echo "<br> both strings are different ";
?>
```

Output: both strings are different

strcasecmp() – compares two strings. Returns 0 if both the strings are same. It is case insensitive (Parwez and parwez are same)

```php
<?php
$name1="Parwez";
$name2="parwez";
if(strcasecmp($name1,$name2)==0)
    echo "<br> Both are same";
else
    echo "<br> Both the strings are different ";
?>
```

Output: Both are same

5.9. strstr()

Finds the first occurrence of the string inside the string and returns rest of the string. It is case sensitive.

```
<?php
$name="My name is Parwez";
$find="is";
echo strstr($name,$find);
?>
```

Output: is Parwez

In the above example, we need to find ‘is’ from the string ‘My name is Parwez’. It searches ‘is’ from the string and when it finds it returns the rest of the string starting from the search string ‘is’.

5.10. stristr()

Finds the first occurrence of the string inside the string and returns the rest of the string. It is case insensitive.

```
<?php
$name="My name is Parwez";
$find="Is";
echo stristr($name,$find);
?>
```

Output: is Parwez

5.11. ltrim()

Remove space or characters from the left side of the string and returns new string.

```
<?php
$name="     My name is Parwez";
echo ltrim($name);
echo "<p>";
$str = "Hello World!";
echo ltrim($str,"Hello");
?>
```

Output:

My name is Parwez

World!

Here in the first example, ltrim has removed white spaces from the left side of the string and in second example, it has removed the word hello from the string.

5.12. rtrim()

Remove space or characters from the right side of the string.

```
<?php
$name="My name is Parwez          ";
echo rtrim($name);
echo "<p>";
$str = "Hello World!";
echo rtrim($str,"World!");
?>
```

Output:

My name is Parwez

Hello

5.13. trim()

Remove space or characters from both the side of the string

```
<?php
$name="      My name is Parwez      ";
echo trim($name);
?>
```

Output: My name is Parwez

5.14. str_replace()

Replace the characters from the string. Parameters:

- Find – the value to find. [Required Argument]
- Replace – the value to replace with the find value. [Required Argument]
- String – the string to be searched. [Required Argument]
- Count – variable that counts the number of replacements. [Optional Argument]

```
<?php
$url="http://contact_us.open.ac.mu";
echo str_replace("contact_us","support",$url,$count);
echo "<p>The word contact_us has been replaced by the word support ".$count."
time/s";
?>
```

Output:

http://support.open.ac.mu

The word contact_us has been replaced by the word support 1 time/s

Here, the word “contact_us” has been replaced by the word “support”. Only one replacement has been done, so count equals 1.

5.15. substr_replace()

It replaces a part of a string with another string.

Parameters:

- string – the string to check. [Required Argument]
- Replacement – specifies the string to insert. [Required Argument]
- Start – Specifies where to start replacing in the string. [Required Argument]
- length – Specifies how many characters should be replaced. [Optional Argument]

```php
<?php
$string="Port-Louis";
echo "<br>".substr_replace($string,"Mathurin",5);
?>
```

Output: Port-Mathurin

Here it places the pointer to the 5th position and replaces all the characters from 5th position to the end because we have not specified the length.

```php
<?php
$string="Petite-Retraite";
echo "<br>".substr_replace($string,"Grande",0,6);
?>
```

Output: Grande-Retraite

Here we have specified the length, so it places the pointer to the 0th position and replaces the first six (6) characters only.

5.16. substr()

It returns the part of the string. If we want to return any part of the string with its position then we can use this function,

Parameters:

- string – the string to check. [Required Argument]
- Start – Specifies where to start in the string. [Required Argument]
- length – Specifies the length of the returned string. [Optional Argument]

```php
<?php
$name="My name is Parwez";
echo "<br> substr: ".substr($name,11,3);
?>
```

Output : Par

So, these are some of the important string handling functions used to perform actions on the string data in PHP.