3...

Functions and Strings

Learning Objectives...

- To learn the Functions in PHP.
- To study the concept of Strings in PHP.

3.1 FUNCTIONS IN PHP

- A function in PHP is a block of organized, reusable code that is used to perform a single, related action. Functions provide better modularity for your application and a high degree of code reuse.
- There are two types of functions in PHP:
  1. **Built-in functions:** PHP's standard library contains a large number of built-in functions for string processing, file IO, mathematical computations and more.
  2. **User-defined functions:** You can create user-defined functions too, specific to the requirements of the programming logic.
- A function may be invoked from any other function by passing required data (called parameters or arguments). The called function returns its result back to the calling environment.
- There are two parts which should be clear to you:
  1. Defining a PHP Function
  2. Calling a PHP Function

3.1.1 Defining a PHP Function

- The first step is to write a function and then you can call it as many times as required.
- To create a new function, use the function keyword, followed by the name of the function you may want to use. In front of the name, put a parenthesis, which may or may not contain arguments.

```
function sayHello()
{
   echo "Hello!";
}
```

3.1.2 Calling a Function

- To call (or invoke) a function, use its name followed by parentheses:

```
display();
```

- You can call a function any number of times after it's defined.

For Example,

```
$globalVar = 10;
function testScope() {
    global $globalVar;
    echo $globalVar;
}
```

3.1.3 Variable Scope

- Scope can be defined as the range of availability a variable has to the program in which it is declared. PHP variables can be one of four scope types:
  1. Local Scope
  2. Global Scope
  3. Static Variables
  4. Functional Parameter

3.1.3.1 Local Variables

- A variable declared within a PHP function is local and can only be accessed within that function. (the variable has local scope)

Program 3.1:

```
<?php
    $a = 5; // global scope
    function myTest()
    {
        echo $a; // local scope
    }
    myTest();
?>
```

Output:

```
PHP Notice: Undefined variable: a
```

- The script above will not produce any output because the echo statement refers to the local scope variable $a, which has not been assigned a value within this scope.

- You can have local variables with the same name in different functions, because local variables are only recognized by the function in which they are declared.
- Local variables are deleted as soon as the function is completed.

3.1.3.2 Global Variables

- Global scope refers to any variable that is defined outside of any function.
- Global variables can be accessed from any part of the script that is not inside a function. To access a global variable from within a function, use the global keyword.

Program 3.2:

```
<?php
    $a = 5;
    $b = 10;
    function myTest()
    {
        global $a, $b;
        $b = $a + $b;
    }
    myTest();
    //Variable Scope in PHP
    echo $b;
?>
```

Output:

```
15
```

- PHP also stores all global variables in an array called $GLOBALS[index]. Its index is the name of the variable. This array is also accessible from within functions and can be used to update global variables directly.
- The program 3.2 can be rewritten as this:

```
<?php
    $a = 5;
    $b = 10;
    function myTest()
    {
        $GLOBALS['b'] = $GLOBALS['a'] + $GLOBALS['b'];
    }
    myTest(); echo $b;
?>
```

3.1.3.3 Static Variables

- When a function is completed, all of its variables are normally deleted. However, sometimes you want a local variable to not be deleted.
- To do this, use the static keyword when you first declare the variable:

```
static $rememberMe;
```

- Then, each time the function is called, that variable will still have the information it contained from the last time the function was called.

3.1.4 Function Parameters

- A parameter is a local variable whose value is passed to the function by the calling code. Parameters are declared in a parameter list as part of the function declaration:

```
function myTest($para1,$para2,...)
{
// function code
}
```

- Parameters are also called arguments.

```
function add($a, $b)
{
    return $a + $b;
}
echo add(5, 10); // Output: 15
```

Default parameters:

```
function greet($name = "Ali")
{
    echo "Hello, $name!";
}
```

3.1.5 Return Values

- Functions can return data using return:

```
function square($n)
{
    return $n * $n;
}
$result = square(4); // $result = 16
```

3.1.6 Variable Functions

- In PHP, "variable functions" refers to the ability to call a function whose name is determined by the value of a variable. This means that if a variable contains a string that corresponds to the name of an existing function, PHP will treat that variable followed by parentheses as a call to that function.

Program 3.3:

```
<?php
function display() {
  echo "Hello!";

}
$functionName = "display";
$functionName(); // This will call the display() function and output "Hello!"
?>
```

Output:

Hello

3.1.7 Anonymous Functions

- PHP allows defining anonymous functions. Normally, when we define a function in PHP, we usually provide it a name which is used to call the function whenever required. In contrast, an anonymous function is a function that doesn't have any name specified at the time of definition. Such a function is also called closure or lambda function. Often used for short tasks or as arguments.
- The syntax for defining an anonymous function:

```
$var=function ($arg1, $arg2) { return $val; };
```

- Note that there is no function name between the function keyword and the opening parenthesis, and the fact that there is a semicolon after the function definition. This implies that anonymous function definitions are expressions. When assigned to a variable, the anonymous function can be called later using the variable's name.

Program 3.4:

```
<?php
   $square = function($n)
   {
      return $n * $n;
   };
   echo $square(5);
?>
```

Output:

25

- They can also be passed to other functions like array filters, etc.

3.2 STRINGS IN PHP

- A string is a sequence of characters, like 'PHP supports string operations'. A string in PHP as an array of bytes and an integer indicating the length of the buffer. In PHP, a character is the same as a byte. This means that PHP only supports a 256-character set, and hence does not offer native Unicode support.
- PHP also has Heredoc and Newdoc representations of string data type.

3.2.1 Quoting String

- In PHP, strings are sequences of characters used to represent text. They can be defined using either single quotes (') or double quotes (").
  - **Single quotes (' ')**: No variable interpolation.
  - **Double quotes (" ")**: Variables are parsed.

For Example,

```
$name = "Suhel";
echo 'Hello $name';  // Outputs: Hello
$name echo "Hello $name";  // Outputs: Hello Suhel
```

3.2.2 Constants

- A constant in PHP is a name or an identifier for a simple value. A constant value cannot change during the execution of the PHP script. It basically improves code readability and maintainability. Constants make it easier to manage the configuration settings.
  - By default, a PHP constant is case-sensitive.
  - By convention, constant identifiers are always uppercase.
  - A constant name starts with a letter or underscore, followed by any number of letters, numbers, or underscore.
  - There is no need to write a dollar sign ($) before a constant, however one has to use a dollar sign before a variable.
  - Use `define()` to declare unchangeable values.

```
define("SITE_NAME", "MyWebsite");
echo SITE_NAME;
```

3.2.3 Printing Strings

- Several methods are available for printing strings, each with distinct characteristics:

1. **echo:**

- echo is a language construct, not a function, and can be used with or without parentheses.
- It can output one or more strings, separated by commas.

For Example,

```
echo "Welcome to PHP";
```

- You can also concatenate using the dot . operator:

```
$name = "Summaiya";
echo "Hello " . $name;
```

2. **print():**

- print() is also a language construct but behaves more like a function.
- It can only output a single string and always returns 1.

```
print "This is a string.";
```

3. **printf():**

- printf() is a function used for formatted output, allowing you to insert values into a template string.
- It uses format specifiers (e.g., %s for string, %d for integer, %.2f for float with two decimal places) to define how values should be formatted.

```
$name = "Safwan";
$rollno = 230;
printf("My name is %s and I am %d years old.", $name, $rollno);
```

4. **print_r():**

- print_r() is a function primarily used for debugging, displaying human-readable information about variables, especially arrays and objects.
- While it can display string values, its main purpose is for inspecting complex data structures.

```
$data = array("name" => "Sameer", "city" => "Pune");
print_r($data);
```

3.2.4 Accessing Individual Characters

- Individual characters within a string in PHP can be accessed using array-like syntax, treating the string as an array of characters.

```
$str = "Maseera";
echo $str[2];
```

Output: s

- Array-like Indexing: Characters can be accessed by their zero-based index using square brackets [].

Program 3.5:

```
<?php
   $string = "SYBsc(ca)";
    echo $string[0] . "<br>";
    echo $string[1] . "<br>";
?>
```

Output:

S

Y

3.2.5 Cleaning Strings

- Cleaning strings in PHP involves removing or modifying unwanted characters, whitespace, or formatting to prepare the string for specific uses, such as data storage, display, or security.
- Common String Cleaning Operations and PHP Functions:

1. **Removing Whitespace:**

- **trim($string, $character_mask):** Removes whitespace (and other specified characters) from both the beginning and end of a string.
- **ltrim($string, $character_mask):** Removes whitespace (and other specified characters) from the beginning (left side) of a string.
- **rtrim($string, $character_mask):** Removes whitespace (and other specified characters) from the end (right side) of a string.

For Example,

```
$String = "  Junaid & Mohid   ";
$cleanString = trim($dirtyString);
// Result: "Junaid & Mohid";
```

2. **Changing Case:**

- **strtolower($string):** Converts a string to lowercase.
- **strtoupper($string):** Converts a string to uppercase.
- **ucfirst($string):** Converts the first character of a string to uppercase.
- **ucwords($string):** Converts the first character of each word in a string to uppercase.

3.2.6 Encoding and Escaping

- Encoding in PHP refers to the process of converting special characters into a safe format so that the data can be safely displayed in HTML, sent over URLs, or stored/transmitted in other formats like JSON or XML.
- It helps prevent:
  - Cross-Site Scripting (XSS) attacks
  - Broken HTML pages
  - Invalid URLs or JSON strings
- Useful for security and output formatting.
  - `htmlspecialchars()`: Converts special characters to HTML entities
  - `urlencode()`: Converts spaces and symbols for safe URL use

1. **htmlspecialchars function:**

Program 3.6

```
<?php
    $input = "<a href='test'>link</a>";
    $safe = htmlspecialchars($input);
    echo $safe;
?>
```

Output:

```
&lt;a href='test'&gt;link&lt;/a&gt;
```

Program 3.7:

```
<?php
$search_term = "Web technology using PHP";
$encoded_term = urlencode($search_term);
$url = "https://example.com/search?q=" . $encoded_term;
echo $url;
?>
```

Output:

https://example.com/search?q=Web+technology+using+PHP

2. urlencode() function:

- The urlencode() function in PHP is used to encode a string for safe use within a URL. URLs have specific rules regarding allowed characters, and characters like spaces, ampersands (&), and other special symbols are not permitted directly.

Functionality:

- urlencode() converts special characters into a URL-safe format.
- It replaces all non-alphanumeric characters (except for hyphen -, underscore _, and period .) with a percent sign (%) followed by two hexadecimal digits representing the character's ASCII value.
- Spaces are specifically encoded as plus signs (+). This differs from RFC 3986 (which rawurlencode() adheres to), where spaces are encoded as %20. The + encoding for spaces is a historical convention, particularly for application/x-www-form-urlencoded data submitted via HTML forms.

Usage:

- The urlencode() function takes a single string argument and returns the URL-encoded version of that string.

Program 3.8:

```
<?php
    $original_string = "This is a string with spaces & special characters!";
    $encoded_string = urlencode($original_string);
    echo "Original String: " . $original_string . "\n";
    echo "Encoded String: " . $encoded_string . "\n";
?>
```

Output:

Original String: This is a string with spaces & special characters!

Encoded String: This+is+a+string+with+spaces+%26+special+characters%21

Escape characters in PHP:

- Escape characters in PHP are special characters that allow you to include non-printable characters, represent special characters, or format strings in a way that might be challenging with regular characters alone. They are preceded by a backslash \. In this article, we'll explore how to use escape characters in PHP, covering various scenarios and providing detailed descriptions with examples.

Method 1: Escaping Special Characters

- Escape characters are often used to include special characters within strings without disrupting the code. For example, include a double quote within a double-quoted string.

Program 3.9:

```
<?php
    $str = "This is a \"quote\" inside a string.";
    echo $str;
?>
```

Output

This is a "quote" inside a string.

- Here, the escape character \" allows the double quote to be part of the string without ending it.

Method 2: Newline and Tab Characters

- Escape characters are handy for introducing newlines (\n) and tabs (\t) within strings.

Program 3.10:

```
<?php
    $str = "Line 1\nLine 2\nLine 3\n";
    echo $str;
    $str2 = "Name\tAge\tCity";
    echo $str2;
?>
```

Output

Line 1
Line 2
Line 3
Name Age City

Method 3: Backslash Itself

- If you need to include a backslash itself within a string, you can use \\.

Program 3.11:

```
<?php
    $str = "This is a backslash: \\";
    echo $str;
?>
```

Output:

This is a backslash: \

3.2.7 Comparing Strings

- The string comparison using the equal (==) operator & strcmp() Function in PHP, along with understanding their implementation through the example.
- == operator: The comparison operator called Equal Operator is the double equal sign "==". This operator accepts two inputs to compare and returns a true value if both of the values are the same (It compares the only value of the variable, not data types) and returns a false value if both of the values are not the same.
- You can compare strings using:
  - == : Checks if values are equal (case-insensitive)
  - === : Checks if values and types are equal
  - **strcmp():** Returns 0 if equal, positive/negative otherwise

Program 3.12: Program that describes the string comparison using the == operator.

```
<?php
  // Declaration of strings
  $name1 = "Ahad";
  $name2 = "Ahad";
  // Use == operator
  if ($name1 == $name2) {
      echo 'Both strings are equal';
  }
  else {
      echo 'Both strings are not equal';
  }
?>
```

Output:

Both the strings are equal

- **Php strcmp() function:** The strcmp() is an inbuilt function in PHP that is used to compare two strings. This function is case-sensitive which points that capital and small cases will be treated differently, during comparison. This function compares two strings and tells whether the first string is greater or smaller or equals the second string. This function is binary-safe string comparison.

Syntax:

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

- This function accepts two parameters as mentioned above and described below:
  - **$string1:** This parameter refers to the first string to be used in the comparison. It is a mandatory parameter.
  - **$string2:** This parameter refers to the second string to be used in the comparison. It is a mandatory parameter.

- **Return Values:** The function returns a random integer value depending on the condition of the match, which is given by:
  - Returns 0 if the strings are equal.
  - Returns a negative value (< 0), if $string2 is greater than $string1.
  - Returns a positive value (> 0) if $string1 is greater than $string2.

Program 3.13:

```
<?php
    // Declaration of strings
    $name1 = "Ammar";
    $name2 = "ammar";
    // Use strcmp() function
    if (strcmp($name1, $name2) !== 0) {
        echo 'Both strings are not equal';
    }  else {
        echo 'Both strings are equal';
    }
?>
```

Output:

Both strings are not equal

Sample programs

Program 3.14: Write a PHP program to count the number of words and characters in the string "PHP is fun".

```
<?php
    $text = "PHP is fun";
    $wordCount = str_word_count($text);
    $charCount = strlen($text);
    echo "Text: $text <br>";
    echo "Word Count: $wordCount <br>";
    echo "Character Count: $charCount";
?>
```

Output:

Text: PHP is fun
Word Count: 3
Character Count: 10

Program 3.15: Write a PHP program to take a string "hello world" and display it reversed and in uppercase.

```
<?php
    $str = "nishom";
    $reversed = strrev($str);
    $upper = strtoupper($str);
    echo "Original: $str <br>";
    echo "Reversed: $reversed <br>";
    echo "Uppercase: $upper";
?>
```

Output:

Original: nishom

Reversed: mohsin

Uppercase: NISHOM

Check Your Understanding

1. Which keyword is used to return a value from a function?
   (a) exit (b) return
   (c) break (d) continue
2. Which of the following defines a constant in PHP?
   (a) const VALUE = 10; (b) define("VALUE", 10);
   (c) constant VALUE = 10; (d) value = constant(10);
3. What is the output of echo "Hello"[1];?
   (a) H (b) e (c) l (d) o
4. Which of the following functions is used to remove whitespace from the beginning and end of a string?
   (a) trim() (b) strip()
   (c) clean() (d) escape()
5. Which operator is used for string concatenation in PHP?
   (a) + (b) . (c) & (d) ,

Answers

1. (b)	2. (b)	3. (b)	4. (a)	5. (b)

Practice Questions

Q.I Answer the following questions in short:

1. How do you call a user-defined function in PHP?
2. What is the difference between global and local scope?
3. What is the purpose of the returnstatement in a PHP function?
4. What is the output of this code: `$str = "PHP"; echo $str[1];`?
5. What is an anonymous function?
6. Define a constant in PHP with syntax.
7. How can you print a string using printand echoin PHP?
8. What does the function `htmlspecialchars()` do?
9. How do you compare two strings ignoring case in PHP?

Q. II Answer the following questions in detail:

1. Explain how to define and call a function in PHP. Provide suitable examples.
2. Describe different types of variable scopes in PHP. Illustrate with code.
3. What are function parameters in PHP? Explain with examples of default and optional parameters.
4. Explain how return values work in PHP functions. Write a function that returns the factorial of a number.
5. What are variable functions in PHP? How are they used in dynamic function calling?
6. Define anonymous functions in PHP. Write an example using an anonymous function as a callback.
7. Discuss the different ways to quote strings in PHP. Include the differences between single and double quotes.
8. What are PHP constants? How are they defined and used in string manipulation?
9. Explain various methods to clean, encode, and escape strings in PHP.
10. Write a program to compare two strings using different PHP string comparison functions and explain the output.

***