Practical 9 (Writeup)

Title:

Aim:

Theory:

- PHP String function

Function	Description	Example	Output
strlen()	Returns the length of a string	$str = "Hello, world!"; echo strlen($str);	13
strtoupper()	Converts a string to uppercase	$str = "Hello, world!"; echo strtoupper($str);	HELLO, WORLD!
strtolower()	Converts a string to lowercase	$str = "Hello, world!"; echo strtolower($str);	hello, world!
substr()	Returns a substring from a string	$str = "Hello, world!"; echo substr($str, 0, 5);	Hello
str_replace()	Replaces all occurrences of a search string with a replacement string within a string	$str = "Hello, world!"; echo str_replace("world", "PHP", $str);	Hello, PHP!
strpos()	Returns the position of the first occurrence of a substring within a string	$str = "Hello, world!"; echo strpos($str, "world");	7
explode()	Splits a string into an array using a specified delimiter	$str = "Apple, Banana, Cherry"; $arr = explode(", ", $str); print_r($arr);	Array ([0] => Apple [1] => Banana [2] => Cherry)
implode()	Joins array elements into a string using a specified delimiter	$arr = array("Apple", "Banana", "Cherry"); $str = implode(", ", $arr); echo $str;	Apple, Banana, Cherry
trim()	Removes whitespace or specified characters from the beginning and end of a string	$str = " Hello, world! "; echo trim($str);	Hello, world!
htmlspecialchars()	Converts special characters to HTML entities to prevent XSS attacks	$str = "<script>alert('XSS')</script>"; echo htmlspecialchars($str);	<script>alert('XSS')</script>
strlen()	Returns the length of a string	$str = "Hello, world!"; echo strlen($str);	13
strrev()	Reverses a string	$str = "Hello, world!"; echo strrev($str);	!dlrow ,olleH

- Condition Statement in PHP

In PHP, condition statements are used to make decisions based on certain conditions. The most commonly used condition statement is the "if" statement. It allows you to execute a block of code if a specified condition is true. Here's a short explanation of condition statements in PHP:

1. If Statement: The "if" statement evaluates a condition and executes a block of code if the condition is true. It has the following structure:

```
if (condition) {
    // Code to be executed if the condition is true
}
```

2. If-else Statement: The "if-else" statement allows you to execute different blocks of code depending on whether a condition is true or false. It has the following structure:

```
if (condition) {
    // Code to be executed if the condition is true
} else {
    // Code to be executed if the condition is false
}
```

3. If-elseif-else Statement: The "if-elseif-else" statement allows you to test multiple conditions and execute different blocks of code based on the first condition that evaluates to true. It has the following structure:

```
if (condition1) {
    // Code to be executed if condition1 is true
} elseif (condition2) {
    // Code to be executed if condition2 is true
} else {
    // Code to be executed if all conditions are false
}
```

- switch statement in PHP

The switch statement in PHP allows you to perform different actions based on the value of a variable or an expression. It provides an alternative way to write multiple if-elseif-else conditions when you have a single variable to compare. Here's a short explanation of the switch statement in PHP:

The basic structure of a switch statement is as follows:

```
switch (expression) {
    case value1:  // Code to be executed if expression matches value1
        break;
    case value2:  // Code to be executed if expression matches value2
        break;
    // More cases can be added as needed
    default:      // Code to be executed if none of the cases match
```

```
        break;
    }
```

Explanation:

- The expression is the variable or value that you want to compare against different cases.

- Each case represents a possible value that the expression can match. If a case matches the expression, the corresponding code block is executed.

- The "break" statement is used to terminate each case. It ensures that the execution flow doesn't continue to the next case. Without the "break" statement, all the code blocks below the matching case will be executed until a "break" or the end of the switch statement is reached.

- The "default" case is optional and is executed when none of the cases match the expression.

- Loops in PHP

1. for Loop: The for loop is used when you know the number of iterations in advance. It consists of an initialization step, a condition to check before each iteration, and an increment or decrement step.

```
for (initialization; condition; increment/decrement) {
    // Code to be executed in each iteration
}
```

2. while Loop: The while loop is used when you don't know the number of iterations in advance but have a condition that needs to be true for the loop to continue. It checks the condition before each iteration.

```
while (condition) {
    // Code to be executed in each iteration
}
```

3. do-while Loop: The do-while loop is similar to the while loop, but it checks the condition after executing the loop's code block. This guarantees that the code block executes at least once.

```
do {
    // Code to be executed in each iteration
} while (condition);
```

- PHP constants

In PHP, constants are named values that cannot be changed during the execution of a script. They provide a way to store fixed values that remain consistent throughout the code. Here's a short explanation of constants in PHP:

Defining a Constant: Constants are defined using the define() function or the const keyword. Once defined, their values cannot be modified.

Using define() function: `define('CONSTANT_NAME', value);`

Using const keyword: const CONSTANT_NAME = value;

- User define function in PHP

User-defined functions in PHP allow you to create custom functions that encapsulate a block of code and can be called multiple times within your script. Here's a short explanation of user-defined functions in PHP:

Function Definition: User-defined functions are defined using the function keyword followed by the function name and a pair of parentheses (). The code to be executed is enclosed within a pair of curly braces {}.

Syntax:

```
function functionName(parameters) {
  // Code to be executed
}
```

- File handling in PHP

File handling in PHP allows you to read from and write to files on the server. It provides functions and methods to perform various operations on files. Here's a short explanation of file handling in PHP:

1. Opening a File: To open a file, you can use the fopen() function, which takes the file path and the mode (e.g., read, write, append) as parameters.

2. Reading from a File: Once a file is opened, you can read its contents using functions like fgets(), fread(), or file(). These functions allow you to read the file line by line or retrieve its entire contents.

3. Writing to a File: To write to a file, you can use functions like fwrite() or file_put_contents(). They allow you to write data to a file, either by overwriting its existing contents or by appending new data.

4. Closing a File: After performing the required file operations, it's important to close the file using the fclose() function. This frees up system resources and ensures that changes are saved.

Example (Reading from a File):

```
$file = fopen('data.txt', 'r');
if ($file) {
  while (($line = fgets($file)) !== false) { echo $line;
  } fclose($file);
} else { echo "Unable to open the file.";
}
```

Conclusion: String functions manipulate strings, condition statements control code flow based on conditions, loops repeat code, constants store fixed values, and file handling manages file operations in PHP.

****Code and output not available in this file.***