Tutorial5

Topic 1: Functions & Return Values

Consider that we need a book categorizer for a Library. In PHP, we can simply create a function for that specific purpose, which categorizes books based on their length.

In PHP, such a function looks like this:

```
<?php
// First we declare the function to determine book category based on lengt
h
function categorizeBook($pages) {
    if ($pages < 100) {
        return "Light Read";
    } elseif ($pages < 300) {
        return "Standard Novel";
    } else {
        return "Epic Saga";
    }
}

// Then we call that function to categorize a particular book
$harryPotterPages = 600;
$category = categorizeBook($harryPotterPages);
echo "Harry Potter is considered " . $category;
?>
```

Now, create a function named `recommendBook` that accepts one argument `$genre`

Based on `$genre` recommend:

- "A Game of Thrones" for Fantasy genre.
- “Frankenstein” for Sci-Fi
- “The Silent Patient” for Mystery
- and “Man's Search for Meaning” for any other genre

Topic 2: String Handling (Explode, Implode and Other Common String Functions)

Books often have tags that represent them. Suppose multiple tags are stored as a single string in the library database. However, we need them as a list while displaying. We can explode() to break the content of the string apart.

```
<?php
$hptags = "magic,wizards,school,friendship";

// Converts string to array. Each element is obtained by cutting the string at every comma.
$tagsArray = explode(",", $hptags);

// We can use foreach method to loop through the array.
echo "<ul>";
foreach ($tagsArray as $tag) {
   echo "<li>$tag</li>";
}
echo "</ul>";
?>
```

Similarly, we can convert array elements into a single string without having to loop through them. The method is called `implode()`

Create an array called `$favAuthors` containing three of your favourite authors eg: `[Stephen King, Jodi Picoult, Haruki Murakami]`.

Then use the pipe symbol (`|`) to display them together in a single string using `implode()` .

eg: `Stephen King | Jodi Picoult | Haruki Murakami`

Now, use `strlen(), str_word_count(), strrev(), strpos() and str_replace()` string functions to manipulate the given string.

```
<?php

$quote = "Happiness can be found even in the darkest of times, if one only remembers to turn on the light.";

// Question 1: Use strlen() to display the total number of characters

// Question 2: Use str_word_count() to count how many words are in the qu ote

// Question 3: Use strrev() to reverse the entire quotation

// Question 4: Use strpos() to check if the word "light" appears in the quote
// If it appears, display its position. If not, display a message.

// Question 5: Use str_replace() to replace the word "darkest" with "terribl e"
// Then display the updated quote

?>
```

Topic 3: File Handling (Writing Data)

Let’s learn how PHP handles files. Suppose the library records the book borrowing event into a log file. We can use PHP file manipulation methods to open, read, and write that file.

```php
<?php

// Open the file in READ mode ("r")
// This allows us to read the current contents of the file
if (file_exists("library_log.txt")) {
    $file = fopen("library_log.txt", "r") or die("Unable to open file!");

    // Read the entire file content
    // filesize() tells PHP how many bytes to read
    $fileSize = filesize("library_log.txt");
    $existingLogs = $fileSize > 0 ? fread($file, $fileSize) : "";

    // Close the file after reading
    fclose($file);
} else {
    $existingLogs = "";
}

// Display existing logs in the browser
echo "<h3>Existing Logs:</h3>";
echo nl2br($existingLogs); // nl2br converts \n to <br>

// New log message to be added
$logMessage = "Sarayu borrowed 'The Hidden Pictures'\n";

// Open the file again, this time in APPEND mode ("a")
// Append mode allows us to add new content at the end of the file
$file = fopen("library_log.txt", "a") or die("Unable to open file!");

// Write the new log message into the file
fwrite($file, $logMessage);

// Close the file to save the changes
fclose($file);

// Confirmation message
```

```
echo "<p><strong>Log updated successfully.</strong></p>";
?>
```

But we have already studied the simpler syntax in a previous tutorial ie, `file_get_contents` and `file_put_contents`

```
<?php
// Read the entire file into a variable
$existingLogs = file_get_contents("library_log.txt");

// Display existing logs
echo "<h3>Existing Logs:</h3>";
echo nl2br($existingLogs);

// New log message to add
$logMessage = "Sarayu borrowed '11/22/63'\n";

// Append the new log entry to the file
file_put_contents("library_log.txt", $logMessage, FILE_APPEND);

// Confirmation message
echo "<p><strong>Log updated successfully.</strong></p>";
?>
```

However, `file_get_contents` and `file_put_contents` are just helper methods that wrap the original ones. For larger files and for better performance, we still use `fopen, fclose, fread, and fwrite`.

Suppose you want to save the books you want to read into a `wishlist`.

Write a script that defines a variable `$tbr` with the title of a book you want to read.

Open a file named `wishlist.txt` in `write mode ('w')`. Write your book title into the file and close it. Remember that the file becomes empty before writing in write mode.

After testing the write mode, change it to `append mode ('a')` and keep adding the books to your wishlist.

Topic 4: Include vs Require

Rename the filename of Task 3 code to `wishlist.php`. Now create `index.php` with this code.

```
<?php
require "wishlist.php";
echo "My next read is " . $tbr;
?>
```

Replace `require` with `include`. Was there any changes to the index page?

Then, rename `wishlist.php` to `wish.php`. Check what happens at your index page. Test it with `require` as well.

Topic 5: Error handling with Die vs Try-Catch

Let's try to simulate **simple error handling** while opening a file in a book management system.

We can use the `die()` function to stop the script if an error occurs while opening the file `booklist.txt`.

```php
<?php

$file = fopen("booklist.txt", "r") or die("Error: Unable to open the file.");

?>
```

Now instead of using the die() function, use `try-catch-finally` block for structured error handling.

```php
<?php

// Inside the try block:
//     - Try to open a file called "booklist.txt" in read mode using fopen()
//     - If fopen() returns false, throw a new Exception with the message:
//       "Error: Unable to open the file."
//     - If successful, display: "File opened successfully!"

// In the catch block:
//     - Display the exception message using $e->getMessage()


// Add a finally block to:
//     - Close the file if it was opened
//     - Display: "File handling process complete."

?>
```