Web-Programming-Series-2-Notes-PYQs

For more notes visit

https://rtpnotes.vercel.app

1. How to Use Functions in a PHP Program

Functions in PHP are reusable blocks of code that perform specific tasks. You can define a function and then call it whenever you need it. Here's a simple example:

```php
<?php
// Define a function
function greet($name) {
    return "Hello, " . $name . "!";
}

// Call the function
$message = greet("Alice");
echo $message; // Output: Hello, Alice!
?>
```

Explanation:

- `function greet($name)` defines a function called `greet` that takes one parameter, `$name`.
- Inside the function, it returns a greeting string.
- You can call the function with a name, and it will output a personalized greeting.

2. Distinguish Between implode and explode Functions in PHP

- **`explode()`**: This function splits a string into an array based on a specified delimiter.

Example:

```php
<?php
$string = "apple,banana,orange";
$array = explode(",", $string);
print_r($array);
/*
Output:
Array
(
    [0] => apple
    [1] => banana
    [2] => orange
)
*/
?>
```

Explanation:

- The `explode(",", $string)` function takes a string and splits it at each comma, creating an array.
- **`implode()`**: This function joins array elements into a single string, using a specified delimiter.

Example:

```php
<?php
$array = array("apple", "banana", "orange");
$string = implode(", ", $array);
echo $string; // Output: apple, banana, orange
?>
```

Explanation:

- The `implode(", ", $array)` function takes an array and joins its elements into a single string, separated by a comma and space.

3. What is a Cookie? How Does PHP Support the Concept of Cookies?

Cookies are small pieces of data that websites store on a user's computer to remember information about the user. They are often used to keep track of user sessions, preferences, or other information.

PHP Support for Cookies:
PHP has built-in functions to create, retrieve, and delete cookies.

Setting a Cookie:

```php
<?php
// Set a cookie
setcookie("username", "JohnDoe", time() + (86400 * 30), "/"); // 86400
seconds = 1 day
?>
```

Explanation:

- The `setcookie()` function creates a cookie named "username" with the value "JohnDoe". It expires in 30 days and is available across the entire site ("/").

Retrieving a Cookie:

```php
<?php
if(isset($_COOKIE["username"])) {
    echo "Welcome back, " . $_COOKIE["username"];
} else {
    echo "Hello, guest!";
}
?>
```

Explanation:

- The code checks if the cookie "username" is set. If it is, it greets the user; otherwise, it welcomes a guest.

Deleting a Cookie:

```php
<?php
// Delete a cookie
setcookie("username", "", time() - 3600, "/"); // Set expiration time in the
```

```
past
?>
```

Explanation:

- To delete a cookie, you can set its expiration time to a time in the past.

4. Simple Connection Script to Make a Connection to MySQL with PHP

You can use the `mysqli` extension or `PDO` to connect to a MySQL database. Here's a simple example using `mysqli`:

```php
<?php
$servername = "localhost"; // MySQL server address
$username = "your_username"; // MySQL username
$password = "your_password"; // MySQL password
$dbname = "your_database"; // Database name

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";

// Close connection
$conn->close();
?>
```

- The script creates a new connection to the MySQL server and checks if the connection is successful. If there's an error, it displays an error message.

5. PHP Program to Compute the Sum of Positive Integers Up to 100 Using a Do-While Loop

```php
<?php
$sum = 0; // Initialize sum
$i = 1; // Start from 1

do {
    $sum += $i; // Add the current number to sum
    $i++; // Increment the counter
} while ($i <= 100); // Continue until $i is 100

echo "The sum of positive integers up to 100 is: " . $sum;
?>
```

- The `do-while` loop ensures that the code block is executed at least once. It sums up integers from 1 to 100 and then prints the total.

6. Different Ways to Create an Array in PHP

1. **Indexed Array:**

```php
$fruits = array("apple", "banana", "cherry");
```

2. **Associative Array:**

```php
$ages = array("Alice" => 25, "Bob" => 30, "Charlie" => 35);
```

3. **Multidimensional Array:**

```php
$contacts = array(
    "John" => array("email" => "john@example.com", "phone" => "123-456-
7890"),
    "Jane" => array("email" => "jane@example.com", "phone" => "987-654-
3210")
);
```

4. **Short Array Syntax:**

```php
$colors = ["red", "green", "blue"];
```

7. Functions that Deal with PHP Arrays

1. `count()` : Counts the number of elements in an array.

   ```php
   $fruits = array("apple", "banana", "cherry");
   echo count($fruits); // Output: 3
   ```

2. `array_push()` : Adds one or more elements to the end of an array.

   ```php
   array_push($fruits, "orange");
   print_r($fruits); // Output: Array ( [0] => apple [1] => banana [2] =>
   cherry [3] => orange )
   ```

3. `array_pop()` : Removes the last element from an array.

   ```php
   array_pop($fruits);
   print_r($fruits); // Output: Array ( [0] => apple [1] => banana [2] =>
   cherry )
   ```

4. `array_merge()` : Merges one or more arrays into one.

   ```php
   $moreFruits = array("mango", "pineapple");
   $combinedFruits = array_merge($fruits, $moreFruits);
   print_r($combinedFruits); // Output: Array ( [0] => apple [1] => banana
   [2] => cherry [3] => mango [4] => pineapple )
   ```

8. Loops Used in PHP

- `for` **Loop**: Executes a block of code a specific number of times.

  Example:

```
for ($i = 1; $i <= 5; $i++) {
    echo "Iteration $i<br>";
}
```

- `while` **Loop**: Executes a block of code as long as the condition is true.

  Example:

  ```
  $i = 1;
  while ($i <= 5) {
      echo "Iteration $i<br>";
      $i++;
  }
  ```

- `do-while` **Loop**: Executes a block of code at least once, and then continues as long as the condition is true.

  Example:

  ```
  $i = 1;
  do {
      echo "Iteration $i<br>";
      $i++;
  } while ($i <= 5);
  ```

- `foreach` **Loop**: Used specifically for looping through arrays.

  Example:

  ```
  $fruits = array("apple", "banana", "cherry");
  foreach ($fruits as $fruit) {
      echo $fruit . "<br>";
  }
  ```

---

9. Equivalent PHP Statements

i. Declare an Associative Array Named "ages"

```php
<?php
$ages = array(
    "Harry" => 21,
    "Alice" => 20,
    "Megha" => 22,
    "Bob" => 19
);
?>
```

ii. Modify the Value Associated with the Key "Megha" to 28

```php
<?php
$ages["Megha"] = 28;
?>
```

iii. Sort the Array According to Values Maintaining Key-Value Relationships and Print

```php
<?php
asort($ages); // Sort by values while maintaining keys

foreach ($ages as $name => $age) {
    echo "$name: $age<br>";
}
?>
```

iv. Access the Entry Identified by the Key "Alice"

```php
<?php
echo "Alice's age is: " . $ages["Alice"];
?>
```

10. HTML Page to Enter Two Numbers and Embed PHP Code to Display Calculations

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Calculator</title>
</head>
<body>
    <h1>Simple Calculator</h1>
    <form method="post">
        <label for="num1">Enter first number:</label>
        <input type="number" name="num1" id="num1" required><br>
        <label for="num2">Enter second number:</label>
        <input type="number" name="num2" id="num2" required><br>
        <input type="submit" name="calculate" value="CALCULATE">
    </form>

    <?php
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        $num1 = $_POST['num1'];
        $num2 = $_POST['num2'];

        $sum = $num1 + $num2;
        $difference = $num1 - $num2;
        $product = $num1 * $num2;
        $quotient = $num2 != 0 ? $num1 / $num2 : 'undefined';

        echo "<h2>Results:</h2>";
        echo "Sum: $sum<br>";
        echo "Difference: $difference<br>";
        echo "Product: $product<br>";
        echo "Quotient: $quotient<br>";
    }
    ?>
</body>
</html>
```

- This HTML form allows the user to enter two numbers and submits them via the POST method. The PHP code processes the input when the form is submitted, calculates the sum, difference, product, and quotient, and displays the results.

11. Uses of Cookies in Web Pages

Uses of Cookies:

- **Session Management**: Cookies can track user sessions, allowing users to stay logged in as they navigate a site.
- **Personalization**: Cookies help store user preferences, such as language or theme settings.
- **Tracking and Analytics**: Websites can use cookies to track user behavior and gather analytics data.

Syntax for Setting Cookies in PHP:

```
setcookie("cookie_name", "cookie_value", time() + (86400 * 30), "/"); //
86400 = 1 day
```

- **Parameters:**
  - `"cookie_name"`: The name of the cookie.
  - `"cookie_value"`: The value to be stored in the cookie.
  - `time() + (86400 * 30)`: Expiration time (in this case, 30 days from the current time).
  - `"/"`: Path on the server where the cookie is available. `/` means the cookie is available throughout the site.

Accessing and Deleting a Cookie Using `setcookie()`:

- **Accessing a Cookie:**

  ```
  if (isset($_COOKIE["cookie_name"])) {
      echo "Value of cookie_name: " . $_COOKIE["cookie_name"];
  } else {
      echo "Cookie is not set.";
  }
  ```

- **Deleting a Cookie:**

```
setcookie("cookie_name", "", time() - 3600, "/"); // Set expiration time
to the past
```

- To delete a cookie, you set its value to an empty string and its expiration time to a time in the past.

12. What is a PHP Session?

A **PHP session** is a way to store information (in variables) to be used across multiple pages. Unlike cookies, session data is stored on the server and is more secure.

How to Start, Modify, and Destroy Session Variables

Starting a Session:

```
<?php
session_start(); // Start the session
$_SESSION['username'] = 'JohnDoe'; // Set session variable
?>
```

Modifying Session Variables:

```
<?php
session_start(); // Start the session
$_SESSION['username'] = 'JaneDoe'; // Modify session variable
?>
```

Destroying a Session:

```
<?php
session_start(); // Start the session
session_unset(); // Unset all session variables
session_destroy(); // Destroy the session
?>
```

- `session_start()` : Initializes a new session or resumes an existing one.
- `$_SESSION` : An associative array to store session variables.

- `session_unset()` : Removes all session variables.
- `session_destroy()` : Deletes the session.

13. Explain CREATE, INSERT, and SELECT Operations on MySQL Table

CREATE

The **CREATE** operation is used to create a new table in the database.

Example:

```
CREATE TABLE users (
    id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(30) NOT NULL,
    email VARCHAR(50) NOT NULL
);
```

INSERT

The **INSERT** operation adds new records to a table.
Example:

```
INSERT INTO users (username, email) VALUES ('JohnDoe', 'john@example.com');
```

SELECT

The **SELECT** operation retrieves data from a table.
Example:

```
SELECT * FROM users;
```

- The `CREATE` statement defines a new table with columns.
- The `INSERT` statement adds a new row to the table.
- The `SELECT` statement retrieves data from the table.

14. PHP Program to Check Whether the Given Number is Prime

```php
<?php
function isPrime($number) {
    if ($number <= 1) {
        return false; // 0 and 1 are not prime numbers
    }
    for ($i = 2; $i <= sqrt($number); $i++) {
        if ($number % $i == 0) {
            return false; // Not a prime number
        }
    }
    return true; // Prime number
}

// Example usage
$number = 29; // Change this number to check
if (isPrime($number)) {
    echo "$number is a prime number.";
} else {
    echo "$number is not a prime number.";
}
?>
```

- The function checks if the number is prime by dividing it by all integers up to its square root.

15. Sorting Functions for Arrays in PHP

1. `sort()` : Sorts an indexed array in ascending order.

```php
$numbers = [4, 2, 8, 6];
sort($numbers);
print_r($numbers); // Output: Array ( [0] => 2 [1] => 4 [2] => 6 [3] =>
8 )
```

2. `asort()` : Sorts an associative array in ascending order while maintaining key-value relationships.

```
$ages = array("Alice" => 25, "Bob" => 30, "Charlie" => 20);
asort($ages);
print_r($ages); // Output: Array ( [Charlie] => 20 [Alice] => 25 [Bob]
=> 30 )
```

3. `ksort()` : Sorts an associative array by key in ascending order.

```
$ages = array("Alice" => 25, "Bob" => 30, "Charlie" => 20);
ksort($ages);
print_r($ages); // Output: Array ( [Alice] => 25 [Bob] => 30 [Charlie]
=> 20 )
```

4. `usort()` : Sorts an array with a user-defined comparison function.

```
$numbers = [4, 2, 8, 6];
usort($numbers, function($a, $b) {
    return $b - $a; // Descending order
});
print_r($numbers); // Output: Array ( [0] => 8 [1] => 6 [2] => 4 [3] =>
2 )
```

16. Steps for Establishing PHP-MySQL Connection

1. **Install MySQL**: Make sure you have a MySQL server running.
2. **Install PHP MySQL Extension**: Ensure the `mysqli` or `PDO` extension is enabled in your PHP installation.
3. **Create a Database**: Create a database in MySQL to store your data.
4. **Write PHP Code to Connect**:

```
<?php
$servername = "localhost";
$username = "your_username";
$password = "your_password";
$dbname = "your_database";
```

```
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

- Replace `your_username`, `your_password`, and `your_database` with actual credentials. The connection is established using `mysqli`.

17. Sample PHP Script for CREATE Operations on MySQL Table

```
<?php
$servername = "localhost";
$username = "your_username";
$password = "your_password";
$dbname = "your_database";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Create table
$sql = "CREATE TABLE users (
    id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(30) NOT NULL,
    email VARCHAR(50) NOT NULL
)";

if ($conn->query($sql) === TRUE) {
    echo "Table users created successfully";
```

```php
} else {
    echo "Error creating table: " . $conn->error;
}

// Close connection
$conn->close();
?>
```

- This script connects to the MySQL database and creates a table named `users` with specified columns.

18. Six String Handling Functions in PHP

Here are six common string handling functions used in PHP along with examples:

1. strlen()

Returns the length of a string.

```php
<?php
$string = "Hello, World!";
echo strlen($string); // Output: 13
?>
```

2. strpos()

Finds the position of the first occurrence of a substring in a string.

```php
<?php
$string = "Hello, World!";
$position = strpos($string, "World");
echo $position; // Output: 7
?>
```

3. substr()

Returns a part of a string.

```php
<?php
$string = "Hello, World!";
$sub = substr($string, 7, 5); // Start at position 7, get 5 characters
echo $sub; // Output: World
?>
```

4. strtoupper()

Converts a string to uppercase.

```php
<?php
$string = "Hello, World!";
$upper = strtoupper($string);
echo $upper; // Output: HELLO, WORLD!
?>
```

5. strtolower()

Converts a string to lowercase.

```php
<?php
$string = "Hello, World!";
$lower = strtolower($string);
echo $lower; // Output: hello, world!
?>
```

6. trim()

Removes whitespace (or other characters) from the beginning and end of a string.

```php
<?php
$string = "   Hello, World!   ";
$trimmed = trim($string);
echo $trimmed; // Output: Hello, World!
?>
```

19. Purpose of the Implicit Arrays $_POST and $_GET in PHP

Purpose:

- `$_POST`: This array is used to collect data from forms using the POST method. It does not append data to the URL, making it suitable for sensitive information.
- `$_GET`: This array collects data from forms using the GET method. It appends the data to the URL, which is visible to the user.

Example PHP Script

Here’s an example PHP script that collects numeric data from a form, calculates the sum, difference, and product, and displays the results. Assume this script is embedded in the web page specified by the form's action attribute.

HTML Form:

```
<form action="your_script.php" method="GET">
    <input type="text" name="num1" placeholder="Enter first number" 
required>
    <input type="text" name="num2" placeholder="Enter second number" 
required>
    <input type="submit" value="Calculate">
</form>
```

PHP Script (`your_script.php`):

```
<?php
if (isset($_GET['num1']) && isset($_GET['num2'])) {
    $num1 = $_GET['num1'];
    $num2 = $_GET['num2'];

    $sum = $num1 + $num2;
    $difference = $num1 - $num2;
    $product = $num1 * $num2;

    echo "Sum: $sum<br>";
    echo "Difference: $difference<br>";
    echo "Product: $product<br>";
}
?>
```

Explanation:

- The form collects two numbers and submits them using the GET method.
- The PHP script checks if `num1` and `num2` are set in the `$_GET` array, calculates the sum, difference, and product, and displays the results.

20. Sample PHP Script Demonstrating Function Implementation

Here's a simple PHP script illustrating how functions are implemented:

```
<?php
// Function to calculate the area of a rectangle
function calculateArea($length, $width) {
    return $length * $width;
}

// Function to display the area
function displayArea($length, $width) {
    $area = calculateArea($length, $width);
    echo "The area of a rectangle with length $length and width $width is:
$area";
}

// Example usage
$length = 5;
$width = 10;
displayArea($length, $width);
?>
```

Explanation:

- `calculateArea()` : A function that takes two parameters (length and width) and returns the area.
- `displayArea()` : A function that calls `calculateArea()` and displays the result.
- The script calls `displayArea()` with example values for length and width.

21. Why is PHP Considered to be Dynamically Typed?

Dynamically Typed:
PHP is considered a dynamically typed language because variable types are determined at runtime, meaning you do not need to explicitly declare a variable type before using it. You can assign a value of any type to a variable, and you can change that value to another type at any time. This flexibility allows for more concise code but can also lead to type-related errors if not managed carefully.

Example:

```php
<?php
$variable = "Hello, World!"; // Initially a string
echo $variable; // Output: Hello, World!

$variable = 42; // Now it's an integer
echo $variable; // Output: 42

$variable = 3.14; // Now it's a float
echo $variable; // Output: 3.14
?>
```

22. Distinguish Between `implode` and `explode` Functions in PHP

- `explode()`: This function splits a string into an array based on a specified delimiter.

Example of `explode()`:

```php
<?php
$string = "apple,banana,orange";
$array = explode(",", $string);
print_r($array); // Output: Array ( [0] => apple [1] => banana [2] => orange
)
?>
```

- `implode()`: This function joins elements of an array into a single string using a specified delimiter.

Example of `implode()`:

```php
<?php
$array = array("apple", "banana", "orange");
$string = implode(", ", $array);
echo $string; // Output: apple, banana, orange
?>
```

23 Significance of Cookies in Web

Significance:
Cookies are small pieces of data stored on the user's computer by the web browser while browsing a website. They are used for various purposes, such as:

- **Session Management**: Keeping track of user sessions (e.g., logged-in users).
- **Personalization**: Storing user preferences (e.g., language selection).
- **Tracking**: Gathering data about user behavior for analytics.

Creating and Destroying Cookies in PHP

- **Creating a Cookie**: Use the `setcookie()` function.

Example of Creating a Cookie:

```php
<?php
setcookie("username", "JohnDoe", time() + (86400 * 30), "/"); // 86400 = 1
day
?>
```

- **Destroying a Cookie**: To delete a cookie, set its expiration date to a time in the past.

Example of Destroying a Cookie:

```php
<?php
setcookie("username", "", time() - 3600, "/"); // Set expiration to one hour
ago
?>
```

24. Equivalent PHP Statements for the Given Tasks

i) Declare an Associative Array Named "marks"

```php
<?php
$marks = array(
    "Ram" => 40,
    "Alice" => 20,
    "Raj" => 45,
    "Mary" => 35
);
?>
```

ii) Modify the Value Associated with the Key "Ram" to 50

```php
<?php
$marks["Ram"] = 50; // Update Ram's marks
?>
```

iii) Sort the Array and Print the Sorted Key-Value Pairs

```php
<?php
asort($marks); // Sort the array by values, maintaining key association

foreach ($marks as $name => $score) {
    echo "$name: $score<br>";
}
?>
```

iv) The Entry Identified by the Key "Raj"

```php
<?php
$raj_marks = $marks["Raj"]; // Retrieve marks for Raj
echo "Raj's marks: $raj_marks"; // Output: Raj's marks: 45
?>
```

25. HTML Form for Entering a Number and PHP Code to Check if It's Positive or Negative

HTML Form:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Check Number</title>
</head>
<body>
    <form action="check_number.php" method="POST">
        <label for="number">Enter a number:</label>
        <input type="number" name="number" id="number" required>
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

PHP Code (`check_number.php`):

```php
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $number = $_POST['number']; // Get the number from the form

    if ($number > 0) {
        echo "The number $number is positive.";
    } elseif ($number < 0) {
        echo "The number $number is negative.";
    } else {
        echo "The number is zero.";
    }
}
?>
```

- The HTML form takes a number as input and submits it using the POST method to `check_number.php`.

- The PHP script checks whether the number is positive, negative, or zero and displays the appropriate message.

26. PHP Form Handling Program for User Registration

HTML Registration Form:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>User Registration</title>
</head>
<body>
    <form action="register.php" method="POST">
        <label for="username">Username:</label>
        <input type="text" name="username" id="username" required><br><br>

        <label for="email">Email:</label>
        <input type="email" name="email" id="email" required><br><br>

        <label for="password">Password:</label>
        <input type="password" name="password" id="password" required><br>
<br>

        <label for="fullname">Full Name:</label>
        <input type="text" name="fullname" id="fullname" required><br><br>

        <label for="age">Age:</label>
        <input type="number" name="age" id="age" required><br><br>

        <input type="submit" value="Register">
    </form>
</body>
</html>
```

PHP Code to Handle Registration (`register.php`):

```php
<?php
// Database connection settings
$servername = "localhost";
$username = "your_db_username";
$password = "your_db_password";
$dbname = "your_database_name";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Check if the form is submitted
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $user = $_POST['username'];
    $email = $_POST['email'];
    $pass = password_hash($_POST['password'], PASSWORD_DEFAULT); // Hashing
the password
    $fullname = $_POST['fullname'];
    $age = $_POST['age'];

    // Prepare and bind
    $stmt = $conn->prepare("INSERT INTO users (username, email, password,
fullname, age) VALUES (?, ?, ?, ?, ?)");
    $stmt->bind_param("ssssi", $user, $email, $pass, $fullname, $age);

    // Execute the statement
    if ($stmt->execute()) {
        echo "Registration successful!";
    } else {
        echo "Error: " . $stmt->error;
    }

    // Close the statement and connection
    $stmt->close();
}
```

```
$conn->close();
?>
```

- The HTML form collects user registration data: username, email, password, full name, and age. It submits the data to `register.php`.
- The PHP script connects to the MySQL database and checks if the form is submitted.
- It hashes the password for security, prepares an SQL statement to insert the user data into a `users` table, and executes the statement.