Here are the PHP solutions to the coding challenges:

Basic Syntax and Data Types

- **Write a function to reverse a given string.**

```
<?php
function reverseString($str) {
  return strrev($str);
}

$testString = "hello";
echo "Original String: " . $testString . "<br>";
echo "Reversed String: " . reverseString($testString); // Output: olleh
?>
```

- **Create a function that determines if a given integer is even or odd.**

```
<?php
function isEvenOrOdd($num) {
  if ($num % 2 == 0) {
    return "Even";
  } else {
    return "Odd";
  }
}

$testNumber = 7;
echo "Number: " . $testNumber . "<br>";
echo isEvenOrOdd($testNumber); // Output: Odd
?>
```

- **Implement a function to calculate the factorial of a non-negative integer.**

```
<?php
function factorial($n) {
  if ($n == 0) {
    return 1;
  } else {
    return $n * factorial($n - 1);
  }
}

$testFactorial = 5;
echo "Factorial of " . $testFactorial . " is " . factorial($testFactorial); // Output: 120
?>
```

- **Write code to swap the values of two variables without using a temporary**

variable.

```
<?php
$a = 10;
$b = 20;

echo "Before swap: a = " . $a . ", b = " . $b . "<br>";

$a = $a + $b; // a now holds the sum of a and b
$b = $a - $b; // b now holds the original value of a
$a = $a - $b; // a now holds the original value of b

echo "After swap: a = " . $a . ", b = " . $b;
?>
```

Arrays

- **Write a function that finds the largest element in an array of numbers.**

```
<?php
function findLargest($arr) {
  return max($arr);
}

$numbers = [1, 5, 2, 9, 3];
echo "Largest element: " . findLargest($numbers); // Output: 9
?>
```

- **Create a function to remove duplicate elements from an array.**

```
<?php
function removeDuplicates($arr) {
  return array_unique($arr);
}

$numbersWithDuplicates = [1, 5, 2, 9, 5, 1, 3];
$uniqueNumbers = removeDuplicates($numbersWithDuplicates);
echo "Unique elements: ";
print_r($uniqueNumbers); // Output: Array ( [0] => 1 [1] => 5 [2] => 2 [3] => 9 [4] => 3 )
?>
```

- **Implement a function that merges two sorted arrays into a single sorted array.**

```
<?php
function mergeSortedArrays($arr1, $arr2) {
  $mergedArray = array_merge($arr1, $arr2);
  sort($mergedArray); // Sort the merged array
  return $mergedArray;
```

```
}

$sortedArray1 = [1, 3, 5, 7];
$sortedArray2 = [2, 4, 6, 8];
$mergedSortedArray = mergeSortedArrays($sortedArray1, $sortedArray2);
echo "Merged and sorted array: ";
print_r($mergedSortedArray); // Output: Array ( [0] => 1 [1] => 2 [2] => 3 [3] => 4 [4] => 5 [5] =>
6 [6] => 7 [7] => 8 )
?>
```

- **Write a function to check if a specific element exists in an array.**

```
<?php
function elementExists($arr, $element) {
  return in_array($element, $arr);
}

$myArray = [10, 20, 30, 40, 50];
$elementToCheck = 30;
$elementToCheck2 = 60;

if (elementExists($myArray, $elementToCheck)) {
  echo $elementToCheck . " exists in the array.<br>";
} else {
  echo $elementToCheck . " does not exist in the array.<br>";
}

if (elementExists($myArray, $elementToCheck2)) {
    echo $elementToCheck2 . " exists in the array.";
} else {
    echo $elementToCheck2 . " does not exist in the array.";
}
?>
```

Control Structures

- **Write a script that prints a multiplication table (e.g., up to 10x10) using nested for loops.**

```
<?php
echo "<table border='1'>";
for ($i = 1; $i <= 10; $i++) {
  echo "<tr>";
  for ($j = 1; $j <= 10; $j++) {
    echo "<td>" . $i * $j . "</td>";
  }
```

```
  echo "</tr>";
}
echo "</table>";
?>
```

- **Create a script that takes the current month as input and outputs the corresponding season using if, elseif, and else statements.**

```
<?php
$month = date("F"); // Get the full name of the current month

if ($month == "December" || $month == "January" || $month == "February") {
  $season = "Winter";
} elseif ($month == "March" || $month == "April" || $month == "May") {
  $season = "Spring";
} elseif ($month == "June" || $month == "July" || $month == "August") {
  $season = "Summer";
} else {
  $season = "Autumn";
}

echo "The current month is " . $month . ", and the season is " . $season . ".";
?>
```

- **Implement the classic FizzBuzz problem: print numbers from 1 to 100, but for multiples of 3 print "Fizz", for multiples of 5 print "Buzz", and for multiples of both 3 and 5 print "FizzBuzz".**

```
<?php
for ($i = 1; $i <= 100; $i++) {
  if ($i % 3 == 0 && $i % 5 == 0) {
    echo "FizzBuzz ";
  } elseif ($i % 3 == 0) {
    echo "Fizz ";
  } elseif ($i % 5 == 0) {
    echo "Buzz ";
  } else {
    echo $i . " ";
  }
}
?>
```

Functions

- **Write a function that takes two numeric arguments and returns their sum.**

```
<?php
```

```
function sum($num1, $num2) {
  return $num1 + $num2;
}

$number1 = 15;
$number2 = 25;
echo "The sum of " . $number1 . " and " . $number2 . " is " . sum($number1, $number2);
?>
```

- **Create a function that checks if a given string is a palindrome (reads the same forwards and backwards).**

```
<?php
function isPalindrome($str) {
  $str = strtolower(str_replace(' ', '', $str)); // Remove spaces and convert to lowercase
  $reversedStr = strrev($str);
  return $str == $reversedStr;
}

$testString1 = "racecar";
$testString2 = "hello";
$testString3 = "A man a plan a canal Panama";

if (isPalindrome($testString1)) {
  echo $testString1 . " is a palindrome.<br>";
} else {
  echo $testString1 . " is not a palindrome.<br>";
}

if (isPalindrome($testString2)) {
    echo $testString2 . " is a palindrome.<br>";
} else {
    echo $testString2 . " is not a palindrome.<br>";
}

if (isPalindrome($testString3)) {
    echo $testString3 . " is a palindrome.";
} else {
    echo $testString3 . " is not a palindrome.";
}
?>
```

Object-Oriented Programming

- **Define a simple PHP class Car with properties like color and model, and a method**

startEngine() that prints a message.

```
<?php
class Car {
  public $color;
  public $model;

  public function __construct($color, $model) {
    $this->color = $color;
    $this->model = $model;
  }

  public function startEngine() {
    echo "The " . $this->color . " " . $this->model . "'s engine is starting.<br>";
  }
}

$myCar = new Car("red", "Toyota");
$myCar->startEngine(); // Output: The red Toyota's engine is starting.
?>
```

- **Implement a basic inheritance scenario where you have a parent class Vehicle with a property speed, and a child class Bicycle that extends Vehicle and has its own property hasBasket.**

```
<?php
class Vehicle {
  public $speed;

  public function __construct($speed) {
    $this->speed = $speed;
  }

  public function move() {
    echo "The vehicle is moving at " . $this->speed . " km/h.<br>";
  }
}

class Bicycle extends Vehicle {
  public $hasBasket;

  public function __construct($speed, $hasBasket) {
    parent::__construct($speed); // Call the parent constructor
    $this->hasBasket = $hasBasket;
  }
```

```
  public function ringBell() {
    echo "Ring ring!<br>";
  }
}

$myBicycle = new Bicycle(20, true);
$myBicycle->move(); // Output: The vehicle is moving at 20 km/h.
$myBicycle->ringBell(); // Output: Ring ring!
echo "Has basket: " . ($myBicycle->hasBasket ? "Yes" : "No") . "."; //Output: Has basket: Yes.
?>
```

Database Interaction (Conceptual)

- **Given a database table named users with columns id, name, and email, write the basic SQL query to retrieve the names and emails of all users.**

```
SELECT name, email FROM users;
```

- **Explain in simple terms how you would prevent SQL injection vulnerabilities when interacting with a database in PHP.**
    - **Prepared Statements:** Use prepared statements with parameterized queries. Instead of directly embedding variables into the SQL query, you use placeholders (e.g., '?'). Then, you bind the actual values to these placeholders. The database driver handles the proper escaping and quoting of the values, preventing them from being interpreted as part of the SQL query itself.
    - **Escaping User Input:** If you cannot use prepared statements (which is highly discouraged), you *must* escape any user-provided input that is included in the SQL query. PHP provides functions like mysqli_real_escape_string() (for MySQL) or pg_escape_string() (for PostgreSQL) to do this. However, this is less secure than prepared statements.
    - **Principle of Least Privilege:** Ensure that the database user your PHP application uses has only the necessary permissions. For example, don't give it full administrative rights if it only needs to read and write specific tables.

Example from Snippets

- **Given an array of users, each with an array of purchases and purchase amounts, describe (in pseudo-code) how you would identify and display only the users who have made total purchases exceeding $200.**

```
// data structure:
// users = [
//   { "user_id": 1, "name": "Alice", "purchases": [ { "amount": 150 }, { "amount": 50 } ] },
//   { "user_id": 2, "name": "Bob",   "purchases": [ { "amount": 100 }, { "amount": 80 } ] },
//   { "user_id": 3, "name": "Charlie", "purchases": [ { "amount": 250 }, { "amount": 100 } ] }
// ]
```

```
function displayUsersExceeding200(users):
  for each user in users:
    total_purchase_amount = 0
    for each purchase in user["purchases"]:
      total_purchase_amount = total_purchase_amount + purchase["amount"]
    if total_purchase_amount > 200:
      display user["name"]
```

- **Implement a PHP function that takes an array of temperatures and returns the temperature closest to zero. If there are two temperatures equally close to zero, the function should return the positive one.**

```
<?php
function closestToZero($temperatures) {
    if (empty($temperatures)) {
        return 0;
    }

    $closest = $temperatures[0];
    foreach ($temperatures as $temp) {
        if (abs($temp) < abs($closest)) {
            $closest = $temp;
        } elseif (abs($temp) == abs($closest)) {
            $closest = max($closest, $temp); // Choose the positive
        }
    }
    return $closest;
}

$temps1 = [10, -5, 20, -12, 2];
$temps2 = [10, 5, -5, 20, -12, 2];
$temps3 = [-2, -1, 1, 2];
$temps4 = [];

echo "Closest to zero in temps1: " . closestToZero($temps1) . "<br>"; // Output: 2
echo "Closest to zero in temps2: " . closestToZero($temps2) . "<br>"; // Output: 5
echo "Closest to zero in temps3: " . closestToZero($temps3) . "<br>"; // Output: 1
echo "Closest to zero in temps4: " . closestToZero($temps4); // Output: 0
?>
```

- **Consider a parking lot that can hold motorcycles, cars, and vans. Motorcycles can park in any spot. Cars can park in a single compact or regular spot. Vans take up 3 regular spots. Given the number of each type of vehicle, write a function that calculates how many regular spots the vans are occupying.**

```php
<?php
function calculateVanSpots($motorcycles, $cars, $vans) {
    // Motorcycles don't take up regular spots.
    // Cars take up 1 regular spot each, but we only care about vans.
    $vanSpots = $vans * 3;
    return $vanSpots;
}

$motorcycles = 5;
$cars = 10;
$vans = 3;
$spotsNeeded = calculateVanSpots($motorcycles, $cars, $vans);
echo "Vans are occupying " . $spotsNeeded . " regular spots."; // Output: Vans are occupying
9 regular spots.
?>
```