Web Design and Web Development

Assignment 2 - Unit 2

31. Remove the first element from an array in PHP using array functions.

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

array_shift($fruits); // Removes "apple"

print_r($fruits);
?>
```

OUTPUT:

```
Array
(
  [0] => banana
  [1] => cherry
  [2] => date
)
```

32.a) Infer the result of the following PHP code?

```
<?php
$names = array("alex", "jean", "emily", "jane");
$name = preg_grep("/^e/", $names);
print_r($name);
?>
```

OUTPUT:

```
Array
(
  [2] => emily
)
```

b)

```
?php
$names = array("alex", "jean", "emily", "jane");
$name = preg_grep("/^e/", $names);
print_r($name);
?>
```

OUTPUT:

```
Array
(
  [2] => emily
)
```

33. Construct a PHP code to create a multidimensional array representing a matrix and display the value in the second row and third column.

```
<?php
// Create a 3x3 matrix (multidimensional array)
$matrix = array(
  array(1, 2, 3),   // Row 1
  array(4, 5, 6),   // Row 2
  array(7, 8, 9)    // Row 3
);

// Access the value in the second row and third column
// Note: Arrays are zero-indexed, so row 2 → index 1, column 3 → index 2
$value = $matrix[1][2];
```

```
echo "The value in the second row, third column is: $value";

?>
```

OUTPUT:

```
The value in the second row, third column is: 6
```

34. Replace all occurrences of a specific word with another word in a string using regular expressions in PHP.

```
<?php

// Original string

$text = "The sky is blue. Blue is my favorite color. I like blue skies.";


// Replace all occurrences of the word 'blue' (case-insensitive) with 'green'

$result = preg_replace("/\bblue\b/i", "green", $text);


echo $result;

?>
```

OUTPUT:

```
The value in the second row, third column is: 6
```

35. Write a PHP script using an array that checks if a string contains another string and displays the result.

Code

```
<?php

// Array of strings to check

$strings = array(

   "I love PHP programming",

   "JavaScript is fun",

   "PHP is great for web development"

);
```

```
// The substring we want to search for
$search = "PHP";

// Loop through each string and check
foreach ($strings as $text) {
    if (strpos($text, $search) !== false) {
        // strpos() returns position if found, false if not
        echo "The string '$text' contains '$search'.<br>";
    } else {
        echo "The string '$text' does NOT contain '$search'.<br>";
    }
}
?>
```

OUTPUT:

```
The string 'I love PHP programming' contains 'PHP'.<br>The string 'JavaScript is fun'
does NOT contain 'PHP'.<br>The string 'PHP is great for web development' contains
'PHP'.<br>
```

36. Create an array of fruits in PHP and display the third element.

Code:

```
<?php
// Create an array of fruits
$fruits = array("Apple", "Banana", "Cherry", "Date", "Mango");

// Display the third element (index 2, since arrays are zero-indexed)
echo "The third fruit is: " . $fruits[2];
?>
```

OUTPUT:

```
The third fruit is: Cherry
```

37. Explain Push and Pop in array functions.

code:

```
<?php
// Create an array
$fruits = array("Apple", "Banana");

// Push: Add elements to the end
array_push($fruits, "Cherry", "Mango");
echo "After Push: ";
print_r($fruits);

// Pop: Remove the last element
$removed = array_pop($fruits);
echo "Removed: $removed\n";

// Final array after pop
echo "After Pop: ";
print_r($fruits);
?>
```

OUTPUT:

```
After Push: Array
(
    [0] => Apple
    [1] => Banana
    [2] => Cherry
    [3] => Mango
)
Removed: Mango
After Pop: Array
(
  [0] => Apple
  [1] => Banana
  [2] => Cherry
)
```

38. Interpret the steps to iterate over a PHP array using a while loop with an example

Code

```
<?php
// Step 1: Create the array
$fruits = array("Apple", "Banana", "Cherry", "Mango");

// Step 2: Initialize counter
$i = 0;

// Step 3: Get array length
$length = count($fruits);

// Step 4: Loop through array
while ($i < $length) {
  echo "Fruit at index $i: " . $fruits[$i] . "<br>";

  // Step 6: Increment counter
  $i++;
}
```

```
?>
```

OUTPUT:

```
Fruit at index 0: Apple<br>Fruit at index 1: Banana<br>Fruit at index 2: Cherry<br>Fruit
at index 3: Mango<br>
```

39. A school wants to automate the calculation of student grades. Design a system that allows teachers to input

student scores, calculates their grades, and generates a summary report. How would you utilize arrays and

array functions to store and process the student data effectively?

Code:

```
<?php
// Step 1: Input student data (could be from form, here hardcoded)
$students = array(
   array("name" => "Alice", "score" => 92),
   array("name" => "Bob", "score" => 76),
   array("name" => "Charlie", "score" => 64),
   array("name" => "David", "score" => 58)
);

// Step 2: Function to calculate grade based on score
function calculateGrade($score) {
   if ($score >= 90) return "A";
   elseif ($score >= 80) return "B";
   elseif ($score >= 70) return "C";
   elseif ($score >= 60) return "D";
   else return "F";
}
```

```
// Step 3: Add grades to each student using array_map
$students = array_map(function($student) {
   $student["grade"] = calculateGrade($student["score"]);
   return $student;
}, $students);

// Step 4: Generate summary report
$totalScore = array_sum(array_column($students, "score"));
$averageScore = $totalScore / count($students);

// Step 5: Count students per grade using array_count_values
$gradeCounts = array_count_values(array_column($students, "grade"));

// Step 6: Display report
echo "<h2>Student Grades Report</h2>";
foreach ($students as $student) {
   echo "{$student['name']} - Score: {$student['score']} - Grade: {$student['grade']}<br>";
}

echo "<br><strong>Class Average:</strong> " . round($averageScore, 2) . "<br>";
echo "<strong>Grade Distribution:</strong><br>";
foreach ($gradeCounts as $grade => $count) {
   echo "Grade $grade: $count student(s)<br>";
}
?>
```

OUTPUT:

```
<h2>Student Grades Report</h2>Alice - Score: 92 - Grade: A<br>Bob - Score: 76 - Grade: C<br>Charlie - Score: 64 - Grade: D<br>David - Score: 58 - Grade: F<br><br><strong>Class Average:</strong> 72.5<br><strong>Grade Distribution:</strong><br>Grade A: 1 student(s)<br>Grade C: 1 student(s)<br>Grade D: 1 student(s)<br>Grade F: 1 student(s)<br>
```

40. Write a PHP script to remove all characters from a string except a-z A-Z 0-9 or " " using an array.

Code:

```php
<?php

// Original string

$text = "Hello@# World! 123_Ready?";


// Convert string to array of characters

$chars = str_split($text);


// Allowed characters array (letters, numbers, space)

$allowed = array_merge(

   range('a', 'z'),

   range('A', 'Z'),

   range('0', '9'),

   array(" ")

);


// Filter out unwanted characters

$resultArray = array_filter($chars, function($char) use ($allowed) {

   return in_array($char, $allowed);

});
```

```
// Join the array back into a string
$cleaned = implode("", $resultArray);

// Output result
echo "Original: $text\n";
echo "Cleaned:  $cleaned\n";
?>
```

output:

```
Original: Hello@# World! 123_Ready?
Cleaned:  Hello World 123Ready
```

41. How can you use regular expressions to extract all email addresses from a given string using an array in PHP?

Code:

```
<?php
// Given string containing emails
$text = "Contact us at admin@example.com or support@mydomain.org.
You can also email john.doe123@company.co.uk for more info.";

// Regular expression for matching email addresses
$pattern = '/[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-z]{2,}/i';

// Extract all matches into an array
preg_match_all($pattern, $text, $matches);

// $matches[0] contains the actual email addresses
$emails = $matches[0];
```

```
// Display the array of emails
print_r($emails);
?>
```

OUTPUT:

```
(
  [0] => admin@example.com
  [1] => support@mydomain.org
  [2] => john.doe123@company.co.uk
)
```

42. Write a PHP script to find the maximum and minimum marks from the following set of arrays

```
<?php
$marks1 = array(360,310,310,330,313,375,456,111,256);
$marks2 = array(350,340,356,330,321);
$marks3 = array(630,340,570,635,434,255,298);

// Merge all arrays into one
$allMarks = array_merge($marks1, $marks2, $marks3);

// Find maximum and minimum
$maxMark = max($allMarks);
$minMark = min($allMarks);

// Display results
echo "Maximum Mark: $maxMark<br>";
echo "Minimum Mark: $minMark";
?>
```

OUTPUT:

```
Maximum Mark: 635<br>Minimum Mark: 111
```

43. Develop a regular expression pattern that validates a password based on the following criteria: at least 8 characters long, contains at least one uppercase letter, one lowercase letter, one digit, and one special character.

Code:

```
<?php
$password = "Hello@123"; // Change to test different passwords

$pattern = '/^(?=.*[A-Z])(?=.*[a-z])(?=.*\d)(?=.*[\W_]).{8,}$/';

if (preg_match($pattern, $password)) {
    echo "✅ Valid password";
} else {
    echo "❌ Invalid password";
}
?>
```

OUTPUT:

```
✅ Valid password
```

44. Develop a music playlist management system for a streaming service. The system should allow users to create, modify, and organize playlists. How would you use arrays and array functions to store and manipulate the song data and playlist information efficiently?

Code:

```
<?php
// Playlist management using arrays and array functions
```

```php
// Step 1: Create some songs
$songs = [
  ["title" => "Shape of You", "artist" => "Ed Sheeran"],
  ["title" => "Blinding Lights", "artist" => "The Weeknd"],
  ["title" => "Someone Like You", "artist" => "Adele"],
  ["title" => "Believer", "artist" => "Imagine Dragons"]
];

// Step 2: Create an empty playlist
$playlist = [];

// Step 3: Add songs to the playlist
array_push($playlist, $songs[0], $songs[2]); // Add Shape of You and Someone Like You

// Step 4: Display the playlist
echo "🎵 Playlist:\n";
foreach ($playlist as $song) {
  echo "- {$song['title']} by {$song['artist']}\n";
}

// Step 5: Add another song
$playlist[] = $songs[1]; // Add Blinding Lights

// Step 6: Remove a song (remove "Shape of You")
foreach ($playlist as $index => $song) {
  if ($song['title'] === "Shape of You") {
    unset($playlist[$index]);
    }
}
$playlist = array_values($playlist); // Reindex array

// Step 7: Sort playlist alphabetically by title
usort($playlist, function($a, $b) {
    return strcmp($a['title'], $b['title']);
});

// Step 8: Display final playlist
echo "\n🎵 Final Playlist (Sorted):\n";
foreach ($playlist as $song) {
    echo "- {$song['title']} by {$song['artist']}\n";
}
?>
```

OUTPUT:

```
🎵 Playlist:
- Shape of You by Ed Sheeran
- Someone Like You by Adele

🎵 Final Playlist (Sorted):
- Blinding Lights by The Weeknd
- Someone Like You by Adele
```

45. Write a PHP function to compare two multidimensional arrays and return the difference.

Code:

```php
<?php
// Sample array
$fruits = array("apple", "banana", "orange", "mango", "grape");

// Value to search
$searchValue = "mango";

// Find index
$index = array_search($searchValue, $fruits);

if ($index !== false) {
  echo "The index of '$searchValue' is: $index";
} else {
  echo "'$searchValue' not found in the array.";
}
?>
```

OUTPUT:

```
The index of 'mango' is: 3
```

46. Write a PHP program to find the index of a specific value in an array.

Code:

```php
<?php
// Sample array
$fruits = array("apple", "banana", "orange", "mango", "grape");

// Value to search
$searchValue = "mango";
```

```
// Find index
$index = array_search($searchValue, $fruits);

if ($index !== false) {
   echo "The index of '$searchValue' is: $index";
} else {
   echo "'$searchValue' not found in the array.";
}
?>
```

OUTPUT:

```
The index of 'mango' is: 3
```

47. Delete an element from the below array. And print the array elements in PHP. $x = array (1, 2, 3, 4, 5);

Code:

```
<?php
// Original array
$x = array(1, 2, 3, 4, 5);

// Delete element at index 2 (value = 3)
unset($x[2]);

// Re-index array (optional)
$x = array_values($x);
```

```
// Print array elements
foreach ($x as $value) {
  echo $value . " ";
}
?>
```

OUTPUT:

```
1 2 4 5
```

48. Record number handling in PHP with suitable examples.

Code:

```
<?php
// Example dataset (records)
$records = array("Apple", "Banana", "Cherry", "Date", "Elderberry");

// Access record number 3 (indexes start at 0)
$recordNumber = 3; // 4th record
echo "Record $recordNumber: " . $records[$recordNumber - 1]; // -1 because human count starts from 1
?>
```

OUTPUT:

```
Record 3: Cherry
```

49. A sports team wants to evaluate player performance based on various statistical metrics. Design a system

that utilizes numerical types and mathematical operators to calculate performance indices, averages, and

rankings. How would you handle large datasets and perform complex calculations efficiently?

Code:

```
<?php

// Step 1: Sample dataset (could come from DB in real-world)

$players = [

  ["name" => "Alice", "goals" => 10, "assists" => 8, "fouls" => 2],

  ["name" => "Bob", "goals" => 7,  "assists" => 12, "fouls" => 5],

  ["name" => "Charlie", "goals" => 15, "assists" => 5, "fouls" => 1],

  ["name" => "David", "goals" => 5, "assists" => 6, "fouls" => 0]

];


// Step 2: Calculate Performance Index

foreach ($players as $key => $player) {

  $performanceIndex = ($player['goals'] * 5) + ($player['assists'] * 3) - ($player['fouls'] *
2);

  $players[$key]['performanceIndex'] = $performanceIndex;

}


// Step 3: Calculate Average Performance Index

$totalIndex = array_sum(array_column($players, 'performanceIndex'));

$averageIndex = $totalIndex / count($players);


// Step 4: Sort players by Performance Index (descending)

usort($players, function ($a, $b) {

  return $b['performanceIndex'] <=> $a['performanceIndex'];

});


// Step 5: Display results

echo "Average Performance Index: " . round($averageIndex, 2) . "\n\n";
echo "Player Rankings:\n";
foreach ($players as $rank => $player) {
   echo ($rank+1) . ". " . $player['name'] . " - Index: " . $player['performanceIndex'] . "\n";
}
?>
```

OUTPUT:

```
Average Performance Index: 65.5

Player Rankings:
1. Charlie - Index: 88
2. Alice - Index: 70
3. Bob - Index: 61
4. David - Index: 43
```

50. Construct a PHP script to lower-case and upper-case, all elements in an array.

Code:

```
<?php
// Original array
$players = array("Virat", "Rohit", "Dhoni", "Bumrah");

// Convert all elements to lowercase
$lowercaseArray = array_map('strtolower', $players);

// Convert all elements to uppercase
$uppercaseArray = array_map('strtoupper', $players);

// Print results
```

```
echo "Original Array:\n";
print_r($players);

echo "\nLowercase Array:\n";
print_r($lowercaseArray);

echo "\nUppercase Array:\n";
print_r($uppercaseArray);
?>
```

OUTPUT:

```
Original Array:
Array
(
   [0] => Virat
   [1] => Rohit
   [2] => Dhoni
   [3] => Bumrah
)

Lowercase Array:
Array
(
   [0] => virat
   [1] => rohit
   [2] => dhoni
   [3] => bumrah
)

Uppercase Array:
Array
(
    [0] => VIRAT
    [1] => ROHIT
    [2] => DHONI
    [3] => BUMRAH
)
```

51. Differentiate between array_shift() and array_unshift() in PHP.

```
<?php
// Initial array
$fruits = array("Apple", "Banana", "Cherry", "Mango");

echo "Original Array:\n";
print_r($fruits);

// array_shift() → Removes first element
$removed = array_shift($fruits);
echo "\nAfter array_shift() (Removed: $removed):\n";
print_r($fruits);

// array_unshift() → Adds element(s) to the beginning
array_unshift($fruits, "Strawberry", "Pineapple");
echo "\nAfter array_unshift():\n";
print_r($fruits);
?>
```

OUTPUT:

```
Original Array:
Array
(
   [0] => Apple
   [1] => Banana
   [2] => Cherry
   [3] => Mango
)

After array_shift() (Removed: Apple):
Array
(
   [0] => Banana
   [1] => Cherry
   [2] => Mango
)

After array_unshift():
Array
(
   [0] => Strawberry
   [1] => Pineapple
   [2] => Banana
   [3] => Cherry
   [4] => Mango
```

)

52. Compare stack and queue operations using PHP with appropriate examples.

Code:

```
<?php
// Stack Example (LIFO)
echo "=== STACK (LIFO) Example ===\n";
$stack = array();

// Push elements onto stack
array_push($stack, "A");
array_push($stack, "B");
array_push($stack, "C");

echo "Stack after pushes: ";
print_r($stack);

// Pop element from stack (last element removed)
$popped = array_pop($stack);
echo "Popped from stack: $popped\n";

echo "Stack after pop: ";
print_r($stack);

echo "\n";

// Queue Example (FIFO)
echo "=== QUEUE (FIFO) Example ===\n";
$queue = array();
```

```
// Enqueue elements into queue
array_push($queue, "A");
array_push($queue, "B");
array_push($queue, "C");

echo "Queue after enqueue: ";
print_r($queue);

// Dequeue element from queue (first element removed)
$dequeued = array_shift($queue);
echo "Dequeued from queue: $dequeued\n";

echo "Queue after dequeue: ";
print_r($queue);
?>
```

OUTPUT:

```
=== STACK (LIFO) Example ===
Stack after pushes: Array
(
    [0] => A
    [1] => B
    [2] => C
)
Popped from stack: C
Stack after pop: Array
(
    [0] => A
    [1] => B
)

=== QUEUE (FIFO) Example ===
Queue after enqueue: Array
(
    [0] => A
    [1] => B
    [2] => C
)
Dequeued from queue: A
Queue after dequeue: Array
(
    [0] => B
    [1] => C
)
```

53. Demonstrate the difference in behaviour of array_pop() and array_shift() using a numeric array.

Code:

```php
<?php
// Initial numeric array
$numbers = [10, 20, 30, 40, 50];

echo "Original Array:\n";
print_r($numbers);

// array_pop() removes the last element
```

```
$lastElement = array_pop($numbers);
echo "\nAfter array_pop():\n";
echo "Removed Element: $lastElement\n";
print_r($numbers);

// array_shift() removes the first element
$firstElement = array_shift($numbers);
echo "\nAfter array_shift():\n";
echo "Removed Element: $firstElement\n";
print_r($numbers);

/*
Difference:
- array_pop() removes from the END of the array.
- array_shift() removes from the BEGINNING of the array.
*/
?>
```

OUTPUT:

```
Original Array:
Array
(
   [0] => 10
   [1] => 20
   [2] => 30
   [3] => 40
   [4] => 50
)

After array_pop():
Removed Element: 50
Array
(
    [0] => 10
    [1] => 20
    [2] => 30
    [3] => 40
)

After array_shift():
Removed Element: 10
Array
(
    [0] => 20
    [1] => 30
    [2] => 40
)
```

54. Design a PHP program that simulates a ticket booking queue using built-in array functions.

Code:

```php
<?php
// Ticket booking queue simulation

// Initial empty queue
$ticketQueue = [];
```

```
// Function to add a person to the queue
function bookTicket(&$queue, $personName) {
  array_push($queue, $personName); // Adds to the end of queue
  echo "$personName has booked a ticket and joined the queue.\n";
}

// Function to serve a person from the queue
function serveTicket(&$queue) {
  if (!empty($queue)) {
    $person = array_shift($queue); // Removes from the start of queue
    echo "$person has been served and removed from the queue.\n";
  } else {
    echo "No one in the queue to serve.\n";
  }
}

// Function to show current queue
function showQueue($queue) {
  if (empty($queue)) {
    echo "The queue is empty.\n";
  } else {
    echo "Current Queue: " . implode(", ", $queue) . "\n";
  }
}

// Booking tickets
bookTicket($ticketQueue, "Alice");
bookTicket($ticketQueue, "Bob");
bookTicket($ticketQueue, "Charlie");

showQueue($ticketQueue);

// Serving customers
serveTicket($ticketQueue);
showQueue($ticketQueue);

serveTicket($ticketQueue);
showQueue($ticketQueue);

// Another booking
bookTicket($ticketQueue, "David");
showQueue($ticketQueue);

serveTicket($ticketQueue);
serveTicket($ticketQueue);
serveTicket($ticketQueue); // Trying to serve when queue is empty
?>
```

OUTPUT:

```
Alice has booked a ticket and joined the queue.
Bob has booked a ticket and joined the queue.
Charlie has booked a ticket and joined the queue.
```

```
Current Queue: Alice, Bob, Charlie

Alice has been served and removed from the queue.

Current Queue: Bob, Charlie

Bob has been served and removed from the queue.

Current Queue: Charlie

David has booked a ticket and joined the queue.

Current Queue: Charlie, David

Charlie has been served and removed from the queue.

David has been served and removed from the queue.

No one in the queue to serve.
```

55. Develop a PHP script that uses stack functions to reverse a string.

Code:

```php
<?php
// Original string
$string = "Hello World";

// Convert string into array of characters
$charArray = str_split($string);

// Initialize an empty stack
$stack = [];

// Push each character onto the stack
foreach ($charArray as $char) {
  array_push($stack, $char);
}
```

```
// Pop characters from the stack to reverse the string
$reversedString = "";
while (!empty($stack)) {
   $reversedString .= array_pop($stack);
}

// Output
echo "Original String: " . $string . "<br>";
echo "Reversed String: " . $reversedString;
?>
```

OUTPUT:

```
Original String: Hello World<br>Reversed String: dlroW olleH
```

56. What are all the Functions available to sort a PHP array?

Code:

```
<?php
$fruits = ["banana", "apple", "cherry", "mango"];

// Simple sort
sort($fruits);
print_r($fruits);

// Reverse sort
rsort($fruits);
print_r($fruits);

// Associative array sort by values
```

```
$prices = ["apple" => 50, "banana" => 20, "mango" => 40];

asort($prices);

print_r($prices);


// Sort by keys

ksort($prices);

print_r($prices);

?>
```

OUTPUT:

```
Array

(

   [0] => apple

   [1] => banana

   [2] => cherry

   [3] => mango

)

Array

(

   [0] => mango

   [1] => cherry

   [2] => banana

   [3] => apple

)

Array

(

   [banana] => 20
   [mango] => 40
   [apple] => 50
)
Array
(
   [apple] => 50
   [banana] => 20
   [mango] => 40
)
```

57. Outline the Regular Expression with appropriate examples.

Code:

```php
<?php
// Sample text
$text = "Contact us at test@example.com or admin123@mail.com.
Order 123 costs $45 and delivery takes 2 days. Hello    World   PHP";

// 1. Validate emails
preg_match_all("/[\w\.-]+@[\w\.-]+\.\w+/", $text, $emails);
echo "Extracted Emails:\n";
print_r($emails[0]);

// 2. Extract all numbers
preg_match_all("/\d+/", $text, $numbers);
echo "\nExtracted Numbers:\n";
print_r($numbers[0]);

// 3. Replace multiple spaces with single space
$cleanText = preg_replace("/\s+/", " ", $text);
```

```
echo "\nText after removing extra spaces:\n";
echo $cleanText . "\n";

// 4. Split string by comma or space
$fruitsText = "apple, banana orange,grape";
$parts = preg_split("/[\s,]+/", $fruitsText);
echo "\nSplit Text into Parts:\n";
print_r($parts);
?>
```

OUTPUT:

```

Extracted Emails:
Array
(
    [0] => test@example.com
    [1] => admin123@mail.com
)

Extracted Numbers:
Array
(
    [0] => 123
    [1] => 123
    [2] => 45
    [3] => 2
)

Text after removing extra spaces:
```

```
Contact us at test@example.com or admin123@mail.com. Order 123 costs $45 and
delivery takes 2 days. Hello World PHP


Split Text into Parts:

Array

(

   [0] => apple

   [1] => banana

   [2] => orange

   [3] => grape

)
```

58. Construct a PHP program to extract the mail addresses in the given

string using regular expression.

```
<?php

// Sample string containing emails

$text = "For support contact support@example.com, for sales reach
sales123@shop.com or admin@mail.org";


// Regular expression pattern to match email addresses

$pattern = "/[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}/";


// Extract all email addresses

preg_match_all($pattern, $text, $matches);


// Display extracted emails

echo "Extracted Email Addresses:\n";

print_r($matches[0]);

?>
```

OUTPUT:

```
Extracted Email Addresses:
Array
(
   [0] => support@example.com
   [1] => sales123@shop.com
   [2] => admin@mail.org
)
```

59. Create a function that takes an array of numbers as input and returns the average value.

```php
<?php
// Function to calculate average
function calculateAverage($numbers) {
   if (empty($numbers)) {
      return 0; // Avoid division by zero
   }
   $sum = array_sum($numbers);    // Sum all numbers
   $count = count($numbers);      // Count total numbers
   $average = $sum / $count;      // Calculate average
   return $average;
}

// Example usage
$nums = [10, 20, 30, 40, 50];
$avg = calculateAverage($nums);
echo "The average value is: " . $avg;
?>
```

OUTPUT:

```
The average value is: 30
```

60. Write a PHP function to search a specified value within the values of an associative array.

Code:

```php
<?php
// Function to search a value in an associative array
function searchValueInAssocArray($array, $valueToSearch) {
   $foundKeys = array_keys($array, $valueToSearch); // Get all keys with the value
   if (!empty($foundKeys)) {
      return $foundKeys;
   } else {
      return false; // Value not found
   }
}

// Example usage
$students = [
   "John" => 85,
   "Alice" => 90,
   "Bob" => 75,
   "Charlie" => 90
];

$searchValue = 90;
$result = searchValueInAssocArray($students, $searchValue);

if ($result !== false) {
   echo "Value $searchValue found at key(s): " . implode(", ", $result);
} else {
   echo "Value $searchValue not found in the array.";
}
?>
```

OUTPUT:

```
Value 90 found at key(s): Alice, Charlie
```

61. Recall the steps to delete an element from an array?

Code:

```
<?php
$fruits = ["apple", "banana", "cherry"];

// Step 1 & 2: Delete "banana" (index 1)
unset($fruits[1]);

// Step 3: Re-index
$fruits = array_values($fruits);

// Step 4: Verify
print_r($fruits);
?>
```

OUTPUT:

```
Array
(
   [0] => apple
   [1] => cherry
)
```

62. Demonstrate a PHP script which rounds the following values with 1 decimal digit precision.

```php
<?php
// Sample values
$values = [1.65, 1.65, -1.54];

// Loop through each value and round to 1 decimal place
foreach ($values as $value) {
    echo "Original: $value => Rounded: " . round($value, 1) . "<br>";
}
?>
```

output:

```
Original: 1.65 => Rounded: 1.7<br>Original: 1.65 => Rounded: 1.7<br>Original: -1.54 =>
Rounded: -1.5<br>
```

63. Discover a function that takes an array of numbers as input and returns the sum of all the even numbers in

the array.

Code:

```php
<?php
function sumOfEvenNumbers($numbers) {
    $sum = 0;

    foreach ($numbers as $num) {
        if ($num % 2 == 0) { // Check if even
            $sum += $num;
        }
    }

    return $sum;
}

// Example usage
$array = [1, 2, 3, 4, 5, 6];
echo "Sum of even numbers: " . sumOfEvenNumbers($array);
?>
```

OUTPUT:

```
Sum of even numbers: 12
```

64. A retail company wants to forecast future sales based on historical data. Develop a system that utilizes numerical types, mathematical operators to analyze sales trends, calculate growth rates, and generate sales forecasts using arrays in php.

Code:

```
<?php
// Step 1: Historical sales data (Year => Sales in thousands)
$salesData = [
    2020 => 150,
    2021 => 165,
    2022 => 180,
    2023 => 210
];
```

```
// Step 2: Calculate total and average sales
$totalSales = array_sum($salesData);
$averageSales = $totalSales / count($salesData);

// Step 3: Calculate yearly growth rates
$growthRates = [];
$previousYearSales = null;

foreach ($salesData as $year => $sales) {
  if ($previousYearSales !== null) {
    $growthRate = (($sales - $previousYearSales) / $previousYearSales) * 100; // %
    $growthRates[$year] = round($growthRate, 2);
  }
  $previousYearSales = $sales;
}

// Step 4: Average growth rate
$averageGrowthRate = array_sum($growthRates) / count($growthRates);

// Step 5: Forecast sales for the next 3 years
$forecastYears = 3;
$lastYear = max(array_keys($salesData));
$lastSales = end($salesData);
$forecast = [];

for ($i = 1; $i <= $forecastYears; $i++) {
  $lastSales = $lastSales * (1 + ($averageGrowthRate / 100)); // Apply growth rate
  $forecast[$lastYear + $i] = round($lastSales, 2);
}

// Step 6: Display results
echo "<h3>Sales Analysis Report</h3>";
echo "Total Sales: $totalSales K<br>";
echo "Average Sales: " . round($averageSales, 2) . " K<br>";
echo "Average Growth Rate: " . round($averageGrowthRate, 2) . "%<br><br>";

echo "<strong>Yearly Growth Rates:</strong><br>";
foreach ($growthRates as $year => $rate) {
   echo "$year: $rate%<br>";
}

echo "<br><strong>Sales Forecast:</strong><br>";
foreach ($forecast as $year => $value) {
   echo "$year: $value K<br>";
}
?>
```

OUTPUT:

```
<h3>Sales Analysis Report</h3>Total Sales: 705 K<br>Average Sales: 176.25
K<br>Average Growth Rate: 11.92%<br><br><strong>Yearly Growth
Rates:</strong><br>2021: 10%<br>2022: 9.09%<br>2023:
16.67%<br><br><strong>Sales Forecast:</strong><br>2024: 235.03 K<br>2025: 263.05
K<br>2026: 294.4 K<br>
```

65. Demonstrate PHP script that checks if a string contains another string and displays the result.

Code:

```
<?php
// Main string
$mainString = "Welcome to the world of PHP programming!";

// Search string
$searchString = "PHP";

// Check if $searchString exists in $mainString
if (strpos($mainString, $searchString) !== false) {
   echo "The string '$searchString' was found in the main string.";
} else {
   echo "The string '$searchString' was NOT found in the main string.";
}
?>
```

OUTPUT:

```
The string 'PHP' was found in the main string.
```

66. Difference between count() and sizeof() function in PHP.

```
<?php
// Sample array
$fruits = ["Apple", "Banana", "Cherry"];

// Using count()
echo "Using count(): " . count($fruits) . "<br>";

// Using sizeof()
```

```
echo "Using sizeof(): " . sizeof($fruits) . "<br>";

// Multidimensional array example
$multiArray = ["Apple", "Banana", ["Cherry", "Mango"]];

// Normal count
echo "count() normal mode: " . count($multiArray) . "<br>";

// Recursive count
echo "count() recursive mode: " . count($multiArray, COUNT_RECURSIVE) . "<br>";

// sizeof() works exactly the same
echo "sizeof() recursive mode: " . sizeof($multiArray, COUNT_RECURSIVE) . "<br>";
?>
```

OUTPUT:

```
Using count(): 3<br>
Using sizeof(): 3<br>count()
normal mode: 3<br>count()
recursive mode: 5<br>sizeof()
recursive mode: 5<br>
```

67. Construct a program that tokenizes a sentence into words using regular expressions. Then, count the number of occurrences of each word and display the results.

Code:

```
<?php
// Sample sentence
```

```
$sentence = "PHP is great. PHP is powerful, and PHP is easy to learn.";

// Step 1: Tokenize using regular expressions (extract only words)
preg_match_all('/\b\w+\b/', strtolower($sentence), $matches);

// Step 2: Count occurrences of each word
$wordCounts = array_count_values($matches[0]);

// Step 3: Display results
echo "<h3>Word Frequency:</h3>";
foreach ($wordCounts as $word => $count) {
   echo "$word : $count<br>";
}
?>
```

OUTPUT:

```
<h3>Word Frequency:</h3>php : 3<br>is : 3<br>great : 1<br>powerful : 1<br>and :
1<br>easy : 1<br>to : 1<br>learn : 1<br>
```

68. Construct a PHP script that catches a division by zero error using try-catch.

Code:

```
<?php
try {
   $numerator = 10;
   $denominator = 0;

   if ($denominator == 0) {
        throw new Exception("Division by zero is not allowed.");
    }

    $result = $numerator / $denominator;
    echo "Result: $result";

} catch (Exception $e) {
    echo "Error: " . $e->getMessage();
}
?>
```

OUTPUT:

```
ERROR!
Error: Division by zero is not allowed.
```

69. Build a PHP function to change the following array's all values to upper or lower case.

```
<?php
function changeArrayCase($array) {
    // Lowercase values
    $lowerCaseArray = array_map('strtolower', $array);
    echo "Values are in lower case.<br>";
    print_r($lowerCaseArray);
    echo "<br><br>";

    // Uppercase values
    $upperCaseArray = array_map('strtoupper', $array);
    echo "Values are in upper case.<br>";
    print_r($upperCaseArray);
}

// Sample array
$Color = array('A' => 'Blue', 'B' => 'Green', 'c' => 'Red');

// Call the function
changeArrayCase($Color);
?>
```

OUTPUT:

```
Values are in lower case.<br>Array
(
    [A] => blue
    [B] => green
    [c] => red
)
<br><br>Values are in upper case.<br>Array
(
    [A] => BLUE
    [B] => GREEN
    [c] => RED
)
```

70. Create a PHP program to take input, a sequence of numbers from the user and store it in a list or array.

Code:

```
<?php
// Simulate user input (replace with your own numbers)
$input = "10, 20, 30, 40, 50";

// Convert comma-separated string to array
$numbersArray = array_map('trim', explode(",", $input));

// Display the array
echo "Stored Numbers:\n";
print_r($numbersArray);
?>
```

OUTPUT:

```
Stored Numbers:
Array
(
   [0] => 10
   [1] => 20
   [2] => 30
   [3] => 40
   [4] => 50
)
```