CS4116 Week 4 Lab Guide

Software Development Project

Teaching Assistant Guide

Prepared by: Zia Ur Rehman
Course: CS4116 - Software Development Project
Professor: Conor Ryan

University of Limerick
2026

Contents

Overview

What this lab reinforces

This lab builds directly on the Week 3 lectures about PHP and Bootstrap. You will practise using PHP to generate HTML, and use Bootstrap to make the output more readable.

In this lab, students will:

- Set up a simple PHP project for the lab.
- Understand how PHP and HTML work together in a single file.
- Use `for`, `while` and `do_while` loops to generate sequences.
- Work with associative arrays (`key` → `value`) in PHP.
- Use basic Bootstrap classes (`container`, `table`, `list-group`) to format output.

Prerequisite

It is assumed that you have already read the Week 3 slides on:

- PHP basics, conditionals and loops
- PHP arrays and `foreach`
- Bootstrap containers, rows and basic layout

1 Why PHP and Bootstrap?

1.1 Big picture: what are we doing in this lab?

Before jumping into the tasks, make sure you understand this high–level picture:

- We use **PHP** to write *logic*: variables, loops, arrays.
- That PHP code *outputs HTML*.
- The browser receives only the final HTML.
- **Bootstrap** is just a CSS library that makes that HTML look nice and responsive.

So for every task in this lab, you need to know:

1. What PHP logic do we need?
2. What HTML do we want to produce?
3. Which Bootstrap classes will make that HTML easier to read?

1.2 Concept 1: Minimal PHP syntax you must know

Students do not need all of PHP. For this lab, they only need a small subset.

- **PHP block**: starts with `<?php` and ends with `?>`.
- **Variables**: always start with `$`.
- **Assignment**: use `=` to give a value to a variable.
- **Output**: use `echo` to send text/HTML to the browser.
- **Loops**: `for`, `while`, `do {...} while(...);` to repeat actions.
- **Arrays**: use `[key => value]` syntax.
- **foreach**: the easiest way to go through each item in an array.

Listing 1: Tiny PHP example

```php
<?php
    // Define a variable
    $name = "Student";

    // Output some HTML that uses the variable
    echo "<h2>Hello, $name!</h2>";
?>
```

Key idea

PHP runs on the server. It generates HTML as plain text. The browser never sees the PHP code, only the final HTML.

1.3 Concept 2: Minimal HTML + Bootstrap you must know

For this lab, you only need a few HTML elements:

- Headings: `<h1>`, `<h2>`, ...
- Paragraphs: `<p>` for short pieces of text.
- Lists: `<ul>` (unordered list) and `<li>` (list item).
- Tables: `<table>`, `<tr>` (row), `<td>` (cell).

Bootstrap adds classes that you attach to these elements:

- `class="container"`: centre the content and give it margins.
- `class="bg-light"`: light grey background.
- `class="list-group"` and `list-group-item`: nicely styled lists.
- `class="table"`, `table-striped`, `table-bordered`: clean tables.
- Spacing classes like `my-4` (margin on Y-axis) and `mb-3` (margin bottom).

Listing 2: HTML with Bootstrap classes

```
<div class="container my-4">
  <h1 class="mb-4">CS4116 Week 4 Lab</h1>

  <p>This text sits inside a Bootstrap container.</p>

  <ul class="list-group">
    <li class="list-group-item">Item 1</li>
    <li class="list-group-item">Item 2</li>
  </ul>
</div>
```

1.4 Concept 3: Combining PHP and Bootstrap

A typical pattern used in this lab is:

1. Write the normal HTML structure.
2. Put a `<div class="container">` inside `<body>`.
3. Open a PHP block inside that `div`.
4. Use `echo` to output HTML (including Bootstrap classes).

Listing 3: PHP generating Bootstrap-styled HTML

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Example</title>
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/
          bootstrap.min.css"
      rel="stylesheet">
</head>
<body class="bg-light">
<div class="container my-4">

    <h1>Example Page</h1>

    <?php
        // PHP block generates list items
        echo "<ul class='list-group'>";
        for ($i = 1; $i <= 3; $i++) {
            echo "<li class='list-group-item'>Item $i</li>";
        }
        echo "</ul>";
    ?>

</div>
</body>
</html>
```

During the lab you can repeatedly ask: *What HTML is this PHP producing?* Encourage students to right-click in the browser and use "View Page Source" to see the final HTML.

2 Environment Setup

2.1 Editor and extensions

Recommended editor

We recommend Visual Studio Code (VS Code):

- Install the **PHP IntelliSense** or **Intelephense** extension for syntax highlighting and completion.
- Optionally install **PHP Debug** for step–through debugging.
- The **Live Server** extension is useful for HTML preview, but PHP code must be run through a PHP interpreter or a local server.

2.2 PHP installation (Windows – easiest way (XAMPP))

What is XAMPP?

XAMPP is a software package that installs everything together.
XAMPP stands for:

Letter	Meaning
X	Cross-platform
A	Apache
M	MySQL
P	PHP
P	Perl

So XAMPP gives you:

- Apache (web server)
- PHP (language engine)
- MySQL (database)
- All configured automatically

Instead of installing everything separately.

What is Apache?

Apache Software Foundation develops many software projects, including:

- Apache HTTP Server

Apache is a web server.
A web server:

- Receives requests from browser (like `http://localhost`)
- Finds the file
- If it is PHP → sends it to PHP engine
- Sends the result back to browser

Without a web server:
Your browser cannot process PHP files.

Step 1 – Install XAMPP

1. Open a browser and go to: https://www.apachefriends.org
2. Click **XAMPP for Windows** (choose a version with PHP 8.x).
3. Run the installer:
   - When asked which components, keep defaults (Apache, MySQL, PHP, phpMyAdmin).
   - Finish installation.

Step 2 – Start PHP/Apache

1. Open the XAMPP Control Panel (search "XAMPP Control Panel" in Start menu).
2. Click **Start** next to Apache.
3. The "Apache" row should turn green.

Step 3 – Connect VS Code to your XAMPP folder

1. In File Explorer, go to:
   `C:\xampp\htdocs\`
2. Create a new folder there, e.g.:
   `C:\xampp\htdocs\week04`
3. Open VS Code → **File** → **Open Folder...** → choose `C:\xampp\htdocs\week04`.

Step 4 – Create and run a PHP file

1. In VS Code, inside `week04`, create a file `hello.php`.
2. Put this code inside:

```php
<?php
    $name = "Student";
    echo "<h2>Hello, $name!</h2>";
?>
```

3. Save the file.
4. Open a browser and go to:

   `http://localhost/week04/hello.php`

5. You should see "Hello, Student".

2.3 Folder structure

Students should create a folder called `week04`:

```
week04/
  week04.php
  css/
  images/
```

Organisation

All code for this lab can live in a single file `week04.php`.

3 Bootstrap + PHP Template

We will use a minimal HTML skeleton with Bootstrap and a PHP block. Students can copy this as a starting point.

Listing 4: Base template for week04.php

```php
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>CS4116 Week 4 Lab</title>
    <!-- Bootstrap CSS -->
    <link
        href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/
            bootstrap.min.css"
        rel="stylesheet"
    >
</head>
<body class="bg-light">
<div class="container my-4">
    <h1 class="mb-4">CS4116 Week 4 Lab</h1>

    <?php
```

```
        // Lab tasks will go here.
    ?>

</div>

<!-- Bootstrap JS bundle (optional for this lab) -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/
   bootstrap.bundle.min.js"></script>
</body>
</html>
```

Note!

Bootstrap is only used via the `link` tag in the head and by adding class names (like `container`, `table`) to our HTML tags. PHP is only used between `<?php` and `?>`.

4 Task 1 – Create the `week04` folder

Organisational task

1. In File Explorer, go to:

   `C:\xampp\htdocs\`

2. Create a new folder there, e.g.:

   `C:\xampp\htdocs\week04`

3. Open VS Code → **File** → **Open Folder...** → choose `C:\xampp\htdocs\week04`.

Create the file `week04.php` inside it.

This mimics how a small project is organised: one folder per feature or week, which keeps HTML, PHP, CSS and images together.

5 Task 2 – Define `$lower` and `$upper`

What knowledge is needed?

- PHP variables and assignment (`$lower = 10;`).
- String output with `echo`.
- Basic understanding that these variables will be used later as loop boundaries.

How the solution works

We declare two variables, guarantee that `$lower` has the smaller value, then print them in a short sentence so you can see the values they chose.

Solution

Listing 5: Task 2 – lower and upper

```php
<?php
    // Task 2: define lower and upper bounds

    $lower = 10;  // smaller value
    $upper = 20;  // larger value

    echo "<h2>Task 2: Lower and Upper</h2>";
    echo "<p>Lower is $lower, upper is $upper.</p>";
?>
```

6 Task 3 – Iterate from $lower to $upper using loops

6.1 Understanding the Requirements

From the lab sheet:

- We must use each of the following loops:
  - `for`
  - `while`
  - `do_while`
- For each loop, we should:
  - start at `$lower`,
  - end at `$upper`,
  - print out *every* value in between (including the endpoints).

We will also present the results in a neat, readable way using Bootstrap list classes.

Important

This task is not about fancy output. The goal is to practise using three different loop types to do *exactly the same job*.

6.2 PHP and Bootstrap Concepts You'll Need

Concept 1: `for` loops

A `for` loop has three important parts:

```php
for (initialise; condition; change) {
    // code to repeat
}
```

Example:

```php
for ($i = 1; $i <= 3; $i++) {
    echo $i;
}
```

This prints `123` because:

- Start: `$i = 1`
- While `$i <= 3`, print `$i`
- Increase `$i` by 1 each time (`$i++`)

Concept 2: `while` loops

A `while` loop checks the condition **before** running the body:

```
$i = 1;                    // start value
while ($i <= 3) {          // condition checked first
    echo $i;
    $i = $i + 1;           // remember to update!
}
```

Common bug

If you forget to update `$i` inside the `while` loop, you get an infinite loop.

Concept 3: `do {...} while(...);` loops

A `do-while` loop always runs the body at least once:

```
$i = 1;
do {
    echo $i;
    $i++;
} while ($i <= 3);
```

Even if the condition is false at the end, the body will have run once.

Concept 4: Bootstrap list groups

We will show the loop output as a Bootstrap list:

```
<ul class="list-group">
  <li class="list-group-item">10</li>
  <li class="list-group-item">11</li>
  <li class="list-group-item">12</li>
</ul>
```

The class names:

- `list-group` goes on the `<ul>` element.
- `list-group-item` goes on each `<li>` element.

6.3 Step-by-Step Plan

1. Make sure `$lower` and `$upper` are already defined (Task 2).
2. Add a heading `Task 3` so you can see where this task starts.
3. For the `for` loop:
   (a) Open a `<ul>` with class `list-group`.
   (b) Run a `for` loop from `$lower` to `$upper`.
   (c) In each iteration, output one `<li>` with the current value.
4. Repeat the same idea with a `while` loop.
5. Repeat again with a `do-while` loop.
6. Check in the browser that all three lists show the same sequence.

6.4 Complete Solution with Comments

Listing 6: Task 3 – all values with three loop types

```php
<?php
    echo "<h2>Task 3: All values from lower to upper</h2>";

    // ========================
    // For loop version
    // ========================
    echo "<h5>For loop</h5>";
    echo "<ul class='list-group mb-3'>";

    // Start at $lower; stop when $i is greater than $upper; add 1 each time
    for ($i = $lower; $i <= $upper; $i++) {
        // Each iteration outputs one Bootstrap list item
        echo "<li class='list-group-item'>$i</li>";
    }

    echo "</ul>";

    // ========================
    // While loop version
    // ========================
    echo "<h5>While loop</h5>";
    echo "<ul class='list-group mb-3'>";

    $i = $lower;                    // 1) set starting value
    while ($i <= $upper) {          // 2) check condition BEFORE running body
        echo "<li class='list-group-item'>$i</li>";  // 3) output current value
        $i++;                       // 4) update counter to avoid infinite loop
    }

    echo "</ul>";

    // ========================
    // Do-while loop version
    // ========================
    echo "<h5>Do-while loop</h5>";
    echo "<ul class='list-group mb-3'>";

    $i = $lower;                    // reset starting value
    do {
        echo "<li class='list-group-item'>$i</li>";  // body always runs at least once
        $i++;                       // move to the next value
    } while ($i <= $upper);         // condition checked AFTER the body

    echo "</ul>";
?>
```

Common Mistakes to Avoid

- **Off-by-one errors**: using `$i < $upper` instead of `$i <= $upper`, which skips the upper bound.
- **Forgetting the increment**: missing `$i++` in the `while` or `do-while` loops, causing an infinite loop.
- **Printing without HTML**: using `echo $i;` without wrapping it in `<li>` tags, which makes the page hard to read.
- **Missing Bootstrap classes**: forgetting `list-group` or `list-group-item`, so the list looks unstyled.

7 Task 4: Print Every Second Value

7.1 Understanding the Requirements

Building on Task 3, we still want to move from `$lower` up to `$upper`, but this time:

- We do *not* print every number.
- Instead, we print every **second** value in the range.
- For example, if `$lower = 10` and `$upper = 20`, we would print:

10, 12, 14, 16, 18, 20

We only need to implement this once (for example, using a `for` loop), but you can ask students to repeat it with `while` or `do-while` if you want extra practice.

7.2 PHP and Bootstrap Concepts You'll Need

Concept 1: Changing the "step size" in a loop

In Task 3 our `for` loop increased the counter by 1 each time:

```
for ($i = $lower; $i <= $upper; $i++) {
    // ...
}
```

To print every second value, we simply change the increment:

```
for ($i = $lower; $i <= $upper; $i += 2) {
    // ...
}
```

Here:

- `$i += 2` means "take the current value of `$i` and add 2".
- So the sequence of values is: `$lower`, `$lower + 2`, `$lower + 4`, ...

Concept 2: Reusing Bootstrap list groups

We will again use a Bootstrap list to show the result:

```
<ul class="list-group">
  <li class="list-group-item">10</li>
  <li class="list-group-item">12</li>
  <!-- etc. -->
</ul>
```

This keeps the visual style consistent with Task 3.

7.3 Step-by-Step Plan

1. Make sure `$lower` and `$upper` are already defined above (Task 2).
2. Add a heading `Task 4` so you can see the section clearly.
3. Start an unordered list with classes `list-group mb-3`.
4. Write a `for` loop:
   (a) Initialise `$i` to `$lower`.
   (b) Use the condition `$i <= $upper`.
   (c) Increment `$i` by 2 each time.
5. Inside the loop, output one `<li>` element for each value.
6. Close the list after the loop.
7. Reload the page and check that only every second number appears.

7.4 Complete Solution with Comments

Listing 7: Task 4 – every second value using step size 2

```
<?php
    echo "<h2>Task 4: Every second value from lower to upper</h2>";

    // We will use a Bootstrap list group again for a clean layout.
    echo "<h5>For loop (step of 2)</h5>";
    echo "<ul class='list-group mb-3'>";

    // Start at $lower; continue while $i is less than or equal to
       $upper;
    // increase $i by 2 on each iteration (so we visit every second
       value).
    for ($i = $lower; $i <= $upper; $i += 2) {

        // Output the current value as a list item.
        echo "<li class='list-group-item'>$i</li>";
    }

    // Close the list.
    echo "</ul>";
?>
```

Common Mistakes to Avoid

- **Using `$i++` instead of `$i += 2`** This will print every value (10, 11, 12, ...) instead of every second value.
- **Using the wrong condition** Writing `$i < $upper` instead of `$i <= $upper` will skip the last value when it is exactly two steps away.
- **Starting at the wrong place** If you mistakenly start from `$lower + 1`, your sequence becomes "odd-even-odd" instead of "even-even-even" (or vice versa).
- **Forgetting the list tags** Printing plain numbers without `<ul>` and `<li>` makes the output hard to read and inconsistent with Task 3.

8 Task 5: From $upper Down to $lower, Every Third Value

8.1 Understanding the Requirements

This task is similar to Task 4, but with two important differences:

- We start at `$upper` and move *downwards* to `$lower`.
- We print every **third** value instead of every second value.

For example, if `$lower = 10` and `$upper = 25`, the sequence would be:

25, 22, 19, 16, 13, 10

We will again use a `for` loop and a Bootstrap list to display the results.

8.2 PHP and Bootstrap Concepts You'll Need

Concept 1: Descending `for` loops

In a descending loop:

- The *initial value* is the largest number.
- The *condition* checks that we have not gone below the smallest number.
- The *update* step *subtracts* instead of adds.

General pattern:

```
for ($i = $start; $i >= $end; $i--) {
    // ...
}
```

For this task we subtract 3 each time:

```
for ($i = $upper; $i >= $lower; $i -= 3) {
    // ...
}
```

Concept 2: Matching condition and direction

The direction of the step and the condition must match:

- If we use `$i -= 3` (going down), the condition should use `>=`:

```
$i >= $lower
```

- If we mistakenly write `$i <= $lower` with a descending loop, the body may never run.

Concept 3: Reusing Bootstrap list styling

We will keep the same visual style as Tasks 3 and 4:

```
<ul class="list-group mb-3">
  <li class="list-group-item">25</li>
  <li class="list-group-item">22</li>
  <!-- ... -->
</ul>
```

8.3 Step-by-Step Plan

1. Ensure `$lower` and `$upper` are set (Task 2).
2. Add a heading `Task 5: From upper down to lower, every third value`.
3. Start an unordered list with classes `list-group mb-3`.
4. Write a `for` loop that:
   (a) Starts at `$upper`.
   (b) Continues while `$i >= $lower`.
   (c) Decreases `$i` by 3 each time with `$i -= 3`.
5. Inside the loop, output one list item per value.
6. Close the list after the loop.
7. Test with different values of `$lower` and `$upper` (for example 0 and 20) to see the pattern.

8.4 Complete Solution with Comments

Listing 8: Task 5 – descending loop, step of 3

```
<?php
    echo "<h2>Task 5: From upper down to lower, every third value</h2>"
        ;

    // Use a Bootstrap list group to display the values.
    echo "<ul class='list-group mb-3'>";

    // Start at $upper (largest value).
    // Keep looping while $i is still greater than or equal to $lower.
    // Subtract 3 on each iteration to get every third value.
    for ($i = $upper; $i >= $lower; $i -= 3) {

        // Show the current value as a list item.
```

```
13          echo "<li class='list-group-item'>$i</li>";
14      }
15
16      // Close the list.
17      echo "</ul>";
18  ?>
```

Common Mistakes to Avoid

- **Using the wrong comparison operator** Writing `$i <= $lower` instead of `$i >= $lower` means the loop body might never run.
- **Forgetting to subtract** If you accidentally write `$i += 3` here, `$i` will increase forever and the loop may hang.
- **Starting at the wrong end** If you start from `$lower` and subtract, you immediately go below the range.
- **Mismatched bounds** If `$lower` is greater than `$upper` from Task 2, this loop will not execute at all.

9 Task 6: Create an Associative Array

9.1 Understanding the Requirements

The lab sheet gives you a small table of key/value pairs:

Key	Value
23	John
16	Edward
81	Conor
14	Gauri
99	Allan

For this task you must:

- Create a PHP array that stores these pairs.
- Use the numbers as the *keys*.
- Use the names as the *values*.
- Optionally display the array so you can see its structure.

This prepares the data you will use in Tasks 7–10.

9.2 PHP Concepts You'll Need

Concept 1: Associative arrays (key → value)

In PHP, arrays can have custom keys, not just `0,1,2,...`. We call this an *associative array*.
Syntax:

```php
<?php
    $ages = [
        "Alice" => 21,
        "Bob"   => 19
    ];
?>
```

Here:

- `"Alice"` and `"Bob"` are the keys.
- `21` and `19` are the values.

You can also use numbers as keys:

```php
<?php
    $numbers = [
        10 => "ten",
        20 => "twenty"
    ];
?>
```

Concept 2: Using `print_r` for quick inspection

Sometimes you just want to "see" what is inside an array.

```php
<?php
    echo "<pre>";       // keep formatting
    print_r($ages);     // dump the array structure
    echo "</pre>";
?>
```

`<pre>` tells the browser to preserve spacing and line breaks, which makes the output easier to read.

9.3 Step-by-Step Plan

1. Add a heading for Task 6 so the output is clearly labelled.
2. Create a new PHP variable, for example `$people`.
3. Assign it an associative array with the keys and values from the table.
4. Optionally use `print_r` inside a `<pre>` block to check the array.
5. Keep this variable `$people` available for the following tasks.

9.4 Complete Solution with Comments

Listing 9: Task 6 – associative array of key/value pairs

```php
<?php
    echo "<h2>Task 6: Create the key/value array</h2>";

    // Create an associative array where:
    //  - the keys are the numbers from the lab sheet
    //  - the values are the corresponding names
    $people = [
```

```php
        23 => "John",
        16 => "Edward",
        81 => "Conor",
        14 => "Gauri",
        99 => "Allan"
    ];

    // Optional: show the internal structure of the array.
    // The <pre> tags keep the formatting readable.
    echo "<pre>";
    print_r($people);
    echo "</pre>";
?>
```

Common Mistakes to Avoid

- **Swapping keys and values** Writing `"John" => 23` instead of `23 => "John"` will break later tasks that expect numeric keys.
- **Forgetting commas between items** Each `key => value` pair must be separated by a comma, except the last one.
- **Missing quotes around strings** Names like `John` must be written as `"John"` (or `'John'`).
- **Overwriting the variable** Reusing the name `$people` for something else later will lose this data; keep this array for Tasks 7–10.

10 Task 7: Iterate Through the Array

10.1 Understanding the Requirements

Now that we have the `$people` array from Task 6, we want to:

- Loop through every key/value pair in the array.
- Display each pair in a row of an HTML table.
- Keep the order in which we defined the array (23, 16, 81, 14, 99).

The final output should look like a simple two-column table: one column for the key, one for the name.

10.2 PHP and Bootstrap Concepts You'll Need

Concept 1: `foreach` with key and value

`foreach` is the easiest way to loop over arrays in PHP.

General form:

```php
foreach ($array as $key => $value) {
    // use $key and $value
}
```

Example:

```php
$ages = ["Alice" => 21, "Bob" => 19];

foreach ($ages as $name => $age) {
    echo "$name is $age years old";
}
```

Concept 2: Basic Bootstrap table classes

Bootstrap gives you simple table styling with just a few classes:

```html
<table class="table table-striped">
  <thead>
    <tr><th>Key</th><th>Value</th></tr>
  </thead>
  <tbody>
    <tr><td>23</td><td>John</td></tr>
  </tbody>
</table>
```

We will generate the `<tr>` rows using PHP.

10.3 Step-by-Step Plan

1. Ensure the `$people` array from Task 6 is defined above this code.
2. Output a heading for Task 7.
3. Start a Bootstrap table with headers `Key` and `Value`.
4. Use a `foreach` loop: `foreach ($people as $key => $name){`.
5. Inside the loop, output one table row with two cells:
   - First cell: the numeric key.
   - Second cell: the name.
6. Close the table after the loop.

10.4 Complete Solution with Comments

Listing 10: Task 7 – table output with foreach

```php
<?php
    echo "<h2>Task 7: Iterate and display array in table order</h2>";

    // Start a Bootstrap-styled table.
    echo "<table class='table table-striped'>";
    echo "<thead><tr><th>Key</th><th>Value</th></tr></thead>";
    echo "<tbody>";

    // Go through each key/value pair in the $people array.
    foreach ($people as $key => $name) {

        // For each pair, create one table row with two cells.
        echo "<tr>";
        echo "<td>$key</td>";   // numeric key from the array
        echo "<td>$name</td>";  // corresponding name
```

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

    echo "</tbody></table>";
?>
```

Common Mistakes to Avoid

- **Using `foreach ($people as $value)`** This ignores the keys, so you cannot print both key and value.
- **Forgetting to close tags** Missing `</tr>` or `</table>` can break the layout for later tasks.
- **Echoing HTML outside the loop** Make sure each `<tr>` is inside the `foreach`, otherwise only one row appears.

11 Task 8: Find the Highest Key

11.1 Understanding the Requirements

Using the same `$people` array, we now want to:

- Determine which numeric key in the array is the largest.
- Display that key on the page.

For the given data, this should be `99`, because it is greater than 23, 16, 81 and 14.

We will first show a manual approach using a loop, then optionally show the built-in helper functions.

11.2 PHP Concepts You'll Need

Concept 1: Tracking a running maximum

The basic idea for finding a maximum with a loop is:

1. Start with a variable that holds the "current best" value.
2. For each new value:
    - If it is larger than the current best, update the variable.
3. At the end, the variable holds the maximum.

Example:

```
$numbers = [5, 10, 2];
$max = null;

foreach ($numbers as $n) {
    if ($max === null || $n > $max) {
        $max = $n;
    }
}
```

Concept 2: Using `array_keys` and `max`

PHP also provides shortcuts:

```php
$keys = array_keys($people); // get all keys
$maxKey = max($keys);        // largest key
```

It is useful to show both approaches so students see the underlying logic and the convenient built-ins.

11.3 Step-by-Step Plan

1. Make sure `$people` from Task 6 is still available.
2. Add a heading for Task 8.
3. Initialise a variable `$maxKey` to `null`.
4. Loop over `$people` with `foreach ($people as $key => $name)`.
5. Inside the loop:
   - If `$maxKey` is `null` (first iteration) or `$key > $maxKey`, update `$maxKey`.
6. After the loop, echo the final value of `$maxKey`.
7. Optionally, show the built-in method using `array_keys` and `max`.

11.4 Complete Solution with Comments

Listing 11: Task 8 – maximum key (manual and built-in)

```php
<?php
    echo "<h2>Task 8: Find the highest key</h2>";

    // ---- Manual method using a loop ----

    $maxKey = null;  // start with "no key" yet

    // Look at every key in the $people array
    foreach ($people as $key => $name) {

        // If this is the first key, or if the current key is larger
           than what we have seen so far,
        // update $maxKey.
        if ($maxKey === null || $key > $maxKey) {
            $maxKey = $key;
        }
    }

    // Show the result of the manual search.
    echo "<p>The highest key (manual) is: <strong>$maxKey</strong></p>"
       ;

    // ---- Built-in method for comparison ----

    // Get an array containing just the keys from $people: [23, 16, 81,
        14, 99]
    $allKeys = array_keys($people);
```

```
25
26      // Use max() to find the largest value in that list.
27      $maxKeyBuiltIn = max($allKeys);
28
29      echo "<p>The highest key (built-in) is: <strong>$maxKeyBuiltIn</
            strong></p>";
30  ?>
```

Common Mistakes to Avoid

- **Initialising `$maxKey` to 0 without thinking** If all your keys are negative (not in this lab, but in general), starting at 0 would be wrong. Using `null` works for any numeric keys.
- **Using `$name` instead of `$key`** The maximum we care about is the numeric key, not the name string.
- **Comparing in the wrong direction** Writing `$key < $maxKey` would find the *smallest* key instead.
- **Forgetting to call `array_keys`** Trying `max($people)` directly will compare the *values* (names), not the numeric keys.

12 Task 9: Output Items in Key Order

12.1 Understanding the Requirements

So far, the `$people` array has kept the order in which we defined it:

$$23,\ 16,\ 81,\ 14,\ 99$$

For this task we want to:

- Sort the array by its numeric keys in **ascending** order.
- Then print the items in that new key order.

With the current data, the keys in ascending order are:

$$14,\ 16,\ 23,\ 81,\ 99$$

So the printed order of names should be: Gauri, Edward, John, Conor, Allan.

12.2 PHP and Bootstrap Concepts You'll Need

Concept 1: Sorting by key with `ksort`

PHP gives a function `ksort` that:

- Takes an associative array.
- Sorts it by its keys (ascending).
- Modifies the original array.

Example:

```php
$ages = [
    30 => "Alice",
    20 => "Bob"
];

ksort($ages);  // keys now in order: 20, 30
```

After `ksort`, a `foreach` loop will visit items in the new sorted order.

Concept 2: Reusing the table from Task 7

We can reuse the same Bootstrap table structure:

```html
<table class="table table-bordered">
  <thead><tr><th>Key</th><th>Name</th></tr></thead>
  <tbody>
    <!-- rows generated by PHP -->
  </tbody>
</table>
```

12.3 Step-by-Step Plan

1. Make sure `$people` is still defined as in Task 6.
2. Call `ksort($people);` to sort it by key.
3. Output a heading for Task 9.
4. Start a Bootstrap table (you can change the style slightly, e.g. use `table-bordered`).
5. Use `foreach ($people as $key => $name)` to print each row.
6. Close the table.

12.4 Complete Solution with Comments

Listing 12: Task 9 – sort by key using ksort

```php
<?php
    echo "<h2>Task 9: Output items in key order</h2>";

    // Sort the $people array by its keys in ascending order.
    // After this call, the order of keys will be: 14, 16, 23, 81, 99.
    ksort($people);

    // Start a Bootstrap table.
    echo "<table class='table table-bordered'>";
    echo "<thead><tr><th>Key</th><th>Name</th></tr></thead><tbody>";

    // Iterate through the now-sorted array.
    foreach ($people as $key => $name) {

        // For each key/value pair, output one row.
        echo "<tr><td>$key</td><td>$name</td></tr>";
    }

    echo "</tbody></table>";
?>
```

Common Mistakes to Avoid

- **Using `sort` instead of `ksort`** `sort` reindexes the array and loses the original keys; we need `ksort` to sort *by key*.
- **Calling `ksort` after the `foreach`** If you call `ksort` at the wrong place, the table will still show the unsorted order.
- **Expecting the original order to remain** Remember that `ksort` changes `$people`. If you need the original order later, copy the array first.

13 Task 10: Names as Keys and Increment Values

13.1 Understanding the Requirements

In this final task we must:

- Transform the existing `$people` array so that:
  - The **names** become the keys.
  - The **numbers** become the values.
- Then go through this new array and add **10** to each numeric value.

Starting from:

$$23 \Rightarrow \textit{John},\ 16 \Rightarrow \textit{Edward},\ \ldots$$

we want to build:

$$\textit{John} \Rightarrow 23,\ \textit{Edward} \Rightarrow 16,\ \ldots$$

and then:

$$\textit{John} \Rightarrow 33,\ \textit{Edward} \Rightarrow 26,\ \ldots$$

13.2 PHP Concepts You'll Need

Concept 1: Building a new associative array

We can build a new array by looping over the old one:

```
1 $new = [];
2
3 foreach ($people as $key => $name) {
4     $new[$name] = $key;  // flip: name becomes key, number becomes
          value
5 }
```

Now `$new["John"]` is 23, etc.

Concept 2: Updating values by reference

If we want to modify the values *inside* an array during a `foreach` loop, we can use a reference:

```php
foreach ($new as $name => &$number) {
    $number = $number + 10;  // changes the array directly
}
unset($number); // good practice after using a reference
```

The `&` before `$number` tells PHP that `$number` is not a copy; it is a reference to the actual array element.

13.3 Step-by-Step Plan

1. Make sure `$people` from Task 6 (possibly sorted in Task 9) is available.
2. Create a new empty array, for example `$nameKeys = []`.
3. Use `foreach ($people as $key => $name)` to populate `$nameKeys[$name] = $key`.
4. Optionally print `$nameKeys` with `print_r` so students can see the flipped structure.
5. Use a second `foreach` loop with a reference (`&$number`) to add 10 to each value.
6. Print the updated array to confirm the values increased by 10.

13.4 Complete Solution with Comments

Listing 13: Task 10 – flip keys/values and increment

```php
<?php
    echo "<h2>Task 10: Use names as keys and increment values by 10</h2>";

    // ------------------------------
    // Step 1: Flip keys and values
    // ------------------------------

    // New array: names will be keys, numbers will be values.
    $nameKeys = [];

    // Loop through the original $people array.
    foreach ($people as $key => $name) {

        // In the new array, use the name as the key and the number as the value.
        $nameKeys[$name] = $key;
    }

    echo "<p>Array with names as keys (before increment):</p>";
    echo "<pre>";
    print_r($nameKeys);
    echo "</pre>";

    // ----------------------------------------
    // Step 2: Increment each value by 10
    // ----------------------------------------

```

```php
    // Use a foreach loop with a reference so we can modify the values in place.
    foreach ($nameKeys as $name => &$number) {

        // Add 10 to the current number.
        $number = $number + 10;
    }
    // Break the reference after the loop (good practice).
    unset($number);

    echo "<p>Array with names as keys (after incrementing each value by 10):</p>";
    echo "<pre>";
    print_r($nameKeys);
    echo "</pre>";
?>
```

Common Mistakes to Avoid

- **Assigning the wrong way around** Writing `$nameKeys[$key] = $name;` keeps the original structure and does not flip keys and values.
- **Forgetting the reference `&`** If you write `foreach ($nameKeys as $name => $number)` without `&`, you only change a copy; the array values stay the same.
- **Using `$number++` without understanding** `$number++` adds 1, not 10. Make sure you explicitly add 10.
- **Not resetting the reference** Leaving out `unset($number);` can cause confusing bugs later if you reuse `$number`.

Summary

What you should now be able to do

By completing this lab, students should be able to:

- Write simple PHP scripts embedded in HTML and styled with Bootstrap.
- Use different loop constructs to iterate over ranges of values.
- Create and manipulate associative arrays, including sorting and transforming them.
- Explain, in words, how PHP logic produces HTML that Bootstrap then styles.
- Connect lecture concepts on loops and arrays to practical web code.