**ScopeTreasure Courses**

PHP COURSES · LESSON PH.7 · CODE SCOPE SERIES

# Arrays

An array holds many values under one name — learn numbered arrays, key => value arrays, nested arrays, count, and reading them with foreach.

## Introduction

One variable holds one value. But real data comes in groups: a class has 30 students, a shop has 50 products, a menu has 6 items. Writing 30 separate variables is madness. An **array** holds many values under one name — one box with numbered compartments inside.

PHP has two kinds. An **indexed array** numbers its compartments from 0: `$fruits[0]` is the first fruit. An **associative array** names its compartments with keys: `$marks["Ali"]` is Ali's marks. And arrays can hold arrays — a list of students where each student is itself an array of name and marks.

*(Roman Urdu: Array matlab "aik naam mein kai khaane" — pehla khaana number 0 se shuru hota hai, ya naam wali key se.)*

Almost every PHP page you will ever build is a loop over an array: rows from a database, items in a form, students in a class. Learn arrays properly and the rest of PHP gets much easier.

## Key points

- **An indexed array is a numbered list.** `$fruits = ["apple", "banana", "mango"];` — indexes 0, 1, 2.
- **Indexes start at 0.** The first item is `[0]`, the last is `[count($arr) - 1]`.
- **An associative array uses named keys:** `$marks = ["Ali" => 88, "Sara" => 92];` — read with `$marks["Ali"]`.
- **`count($arr)` tells you how many items there are.** Use it in loops: `for ($i = 0; $i < count($arr); $i++)`.
- **`$arr[] = $value` appends one item** to the end of an array.
- **`foreach` is the natural way to read an array** — `foreach ($arr as $item)` or `foreach ($arr as $key => $value)`.
- **Arrays can nest.** `$students[0]["name"]` reads the name of the first student in a list of student arrays.
- **`array_push($arr, $x)`** also appends; **`array_pop($arr)`** removes and returns the last item.
- **`in_array($x, $arr)`** asks whether a value exists in the array; **`array_key_exists($k, $arr)`** asks about a key.
- **`sort($arr)`** sorts values; **`asort($arr)`** sorts an associative array while keeping its keys.

## Examples

### Example 1 — An indexed array

```
<?php
$fruits = ["apple", "banana", "mango"];
echo $fruits[0];
echo "<br>";
echo $fruits[2];
echo "<br>";
echo count($fruits);
?>
```

What you will get:

```
apple
mango
3
```

Three items, indexes 0 to 2. `count` says 3 — which is why the last index is always `count - 1`.

### Example 2 — An associative array

```
<?php
$marks = ["Ali" => 88, "Sara" => 92, "Bilal" => 76];
echo $marks["Sara"];
echo "<br>";
$marks["Hira"] = 95;
echo $marks["Hira"];
?>
```

What you will get:

```
92
95
```

Keys are labels, not numbers. Adding a new student is one line: give the key and the value.

### Example 3 — Appending and looping

```
<?php
$cart = ["chai"];
$cart[] = "coffee";
$cart[] = "juice";
foreach ($cart as $index => $item) {
    echo "$index: $item<br>";
}
?>
```

What you will get:

```
0: chai
1: coffee
2: juice
```

`$cart[] = ...` always adds at the end. `foreach` with `$index => $item` shows the number and the value together.

### Example 4 — A nested array: students with marks

```
<?php
$students = [
    ["name" => "Ali", "marks" => 88],
    ["name" => "Sara", "marks" => 92],
];
echo $students[0]["name"];
echo " scored ";
echo $students[0]["marks"];
echo "<br>";
echo $students[1]["name"];
echo " scored ";
echo $students[1]["marks"];
?>
```

What you will get:

```
Ali scored 88
Sara scored 92
```

Each student is an associative array inside the outer list. Read it in two steps: which student `[0]`, then which field `["name"]`. This is exactly the shape database rows arrive in (PH.11).

### Example 5 — Checking, sorting, removing

```
<?php
$nums = [5, 2, 9, 1];
sort($nums);
echo $nums[0];
echo "<br>";
$fruits = ["apple", "banana"];
if (in_array("mango", $fruits)) {
    echo "Mango is here";
} else {
    echo "No mango";
}
echo "<br>";
$last = array_pop($fruits);
echo "Removed: $last";
?>
```

What you will get:

```
1
No mango
Removed: banana
```

`sort` puts the smallest first. `in_array` answers yes or no. `array_pop` takes the last item off and hands it back.

### Example 6 — Building an HTML table from an array

```
<?php
$students = ["Ali" => 88, "Sara" => 92, "Bilal" => 76];
echo "<table border='1'>";
foreach ($students as $name => $marks) {
    echo "<tr><td>$name</td><td>$marks</td></tr>";
}
echo "</table>";
?>
```

What you will get:

```
A table with three rows: Ali 88, Sara 92, Bilal 76.
```

One loop, three rows. Add a fourth student to the array and the table grows by itself — no extra HTML needed.

## Quick recap

- Indexed arrays use numbers from 0; associative arrays use named keys.
- `count()` gives the size; the last index is `count - 1`.
- `$arr[] = $x` appends; `foreach` reads; `$arr[0]["name"]` reads a nested array.
- `in_array` checks a value; `sort` orders values; `array_pop` removes the last.
- Arrays + foreach + HTML is the pattern behind nearly every list page.

## Common mistakes

1. **Reading index 3 of a 3-item array** — the indexes are 0, 1, 2, so `[3]` is undefined. Fix: use `count($arr) - 1` for the last item.
2. **Forgetting the quotes on a string key** — `$marks[Ali]` makes PHP look for a constant called Ali. Fix: always write `$marks["Ali"]`.
3. **Using `count()` inside a for condition with `<=`** — `for ($i = 0; $i <= count($arr); $i++)` runs one round too many. Fix: use `<`, not `<=`.
4. **Assuming `sort()` returns the sorted array** — it sorts in place and returns true/false. Fix: call `sort($arr);` then use `$arr`.
5. **Overwriting an array by assigning to it twice** — `$arr = ["a"]; $arr = ["b"];` leaves only `["b"]`. Fix: append with `$arr[] = "b";`.
6. **Echoing a whole array** — `echo $arr;` prints the word "Array" and a warning. Fix: loop over it, or use `print_r($arr)` while debugging.
7. **Mixing up key and value in foreach** — `foreach ($marks as $name => $score)` gives the key first. Fix: remember the order is key, then value.

## Practice

1. Make an indexed array of 5 fruits, print the first and the last, then print the count.
2. Make an associative array of 4 students and their marks, and print each one with foreach.
3. Add a 5th student to that array with `$marks["New"] = 90;` and print the whole list again.
4. Use `in_array` to check whether "Hira" is one of the keys — then learn why you need `array_key_exists` for keys, and test it.
5. Make an array of numbers, `sort` it, and print the smallest and the largest.
6. Build a nested array of 3 students (name + marks each) and print "Name scored Marks" for each with a foreach loop.
7. Build an HTML `<ul>` from an array of 4 tasks with a foreach loop — then add a 5th task and reload to see the list grow.

## Practice quiz

**1. What is the index of the first item in a PHP array?**

- 1
- 0
- -1
- It has no index

**Answer:** 0

**2. How do you read Ali's marks from $marks = ["Ali" => 88, "Sara" => 92];?**

- $marks[0]
- $marks["Ali"]
- $marks.Ali
- $marks("Ali")

**Answer:** $marks["Ali"]

**3. What does $cart[] = "juice"; do?**

- It empties the array
- It appends "juice" to the end of the array
- It removes the last item
- It sorts the array

**Answer:** It appends "juice" to the end of the array

---

**Watch the Code Scope channel:** https://www.youtube.com/@CodeScopePK — this lesson's own video joins the channel when it is recorded.

**Read online:** https://scopetreasure.com/notes/ph-arrays · More free tools and notes at https://scopetreasure.com