

# CMPS 3680 Homework 3

Spring 2026

## Array Types

In most scripting languages there are 3 different types of arrays: indexed, associative, and multi-dimensional. For example, dictionaries in Python are multidimensional arrays.

<https://www.tutorialrepublic.com/php-tutorial/php-arrays.php>

## Iterating Through Arrays

- Traditional for loop: [https://www.w3schools.com/php/php\\_looping\\_for.asp](https://www.w3schools.com/php/php_looping_for.asp)
- The foreach loop: [https://www.w3schools.com/php/php\\_looping\\_foreach.asp](https://www.w3schools.com/php/php_looping_foreach.asp)

## Useful Array functions

- [array\\_merge](#)
- [array\\_reverse](#)
- [array\\_map](#)
- [usort](#) and [uasort](#)
- [list\(\)](#)
- Full list of array functions: <https://www.php.net/manual/en/ref.array.php>

## Callbacks

A [callback](#) is basically a function that is passed as a parameter to another function to be called at a later time. There are three ways to define callbacks:

- Traditional named functions (with quotes)
- Anonymous functions aka Closures (inline, or saved to a variable)
- Arrow functions (like anonymous functions, but can access parent scoped data by default)

## Variables In Strings (interpolation)

### Using variables inline:

You can place variables inside of a string if you use double quotes like this:

```
echo "This is a variable: $data";
```

However, if the variable is an array you must put curly braces around it like this:

```
echo "This is an array variable: {$data['details']}";
```

### Using [printf](#):

PHP has a printf function similar to other languages:

```
printf("This is a variable: %s", $data);
```

### Using [heredoc](#):

When generating large sections of html or mixed string/variable data use heredoc:

```
echo <<<STUFF
  <body>
  <div>{$data['details']}</div>
</body>
STUFF;
```

## Useful String functions

- [nl2br](#)
- [str\\_replace](#)
- [parse\\_str](#)
- [explode](#)
- [implode](#)
- [count\\_chars](#)
- Full list of array functions: <https://www.php.net/manual/en/ref.strings.php>

## Functions In PHP

Since PHP is loosely typed, there is no return type and parameter types are not enforced by default:

```
function foo($x, $y){  
    return $x + $y;  
}
```

To enforce parameter and return types you must set strict typing using `declare(strict_types=1);`

Notice how the return type goes at the end of the function header:

```
declare(strict_types=1);  
  
function foo(int $x, int $y) : int {  
    return $x + $y;  
}
```

More about PHP type declarations: <https://www.php.net/manual/en/language.types.declarations.php>

## Additional Reading:

- Context Shifting/Switching:  
<https://www.youtube.com/watch?v=0b02-Wkpwo>  
<https://www.software.com/devops-guides/context-switching>