CSY2028
Web Programming

Topic 5 – Arrays and User Input

- Introduction to arrays
- User input and Super Globals
- Useful array functions

Arrays

- So far we've used variables that store individual values e.g.

```php
<?php
$str = 'a string variable';
$num = 1234;
?>
```

Arrays

- Arrays allow you to store more than one value inside a single variable
- This is useful for many tasks whenever you need a collection of values

Arrays

- An array lets you store a value under a key
- An array lets you store a value under a key
- To create an array, create a variable using two square brackets:

```
<?php

$myArray = [];

?>
```

Arrays

- Once you have declared a variable as an array, you can write values to it under keys
- For example:

```php
<?php

$myArray = [];

$myArray[1] = 'Value 1';
$myArray[2] = 'Value 2';
$myArray[3] = 'Value 3';

?>
```

Arrays

- Once you've written to an array key you can read the value back out of the array using the same square bracket notation
- You can use array values like any other variable

```php
<?php

$myArray = [];

$myArray[1] = 'Value 1';
$myArray[2] = 'Value 2';
$myArray[3] = 'Value 3';


echo '<p>' . $myArray[2] . '</p>';
?>
```

```
Output:
<p>value 2</p>
```

Arrays

- However, you can't use an array like a string/integer

```php
<?php

$myArray = [];

$myArray[1] = 'Value 1';
$myArray[2] = 'Value 2';
$myArray[3] = 'Value 3';

echo '<p>' . $myArray . '</p>';
?>
```

- Error: Array to string conversion

Numerical arrays - Shorthand

- Instead of:

```php
<?php
$myArray = [];

$myArray[1] = 'Value 1';
$myArray[2] = 'Value 2';
$myArray[3] = 'Value 3';
?>
```

- You can achieve the same result with the shorthand notation:

```php
<?php

$myArray = ['Value 1', 'Value 2', 'Value 3'];

echo '<p>' . $myArray[2] . '</p>';

?>
```

Arrays

```php
<?php

$myArray = ['Value 1', 'Value 2', 'Value 3'];

echo '<p>' . $myArray[2] . '</p>';
```

```
Output:
<p>Value 3</p>
```

- Note: Arrays start from zero!

```php
<?php

$myArray = ['Value 1', 'Value 2', 'Value 3'];

echo '<p>' . $myArray[0] . '</p>';
echo '<p>' . $myArray[1] . '</p>';
echo '<p>' . $myArray[2] . '</p>';
?>
```

```
Output:
<p>Value 1</p>
<p>Value 2</p>
<p>Value 3</p>
```

Array Sizes

- Every array has a size
- This is the number of elements in the array
- This can be read using the count() function on the array variable

```php
<?php

$myArray = ['Value 1', 'Value 2', 'Value 3'];

echo count($myArray);

?>
```

```
Output:
3
```

Array Indexes

- You can use a variable to read an array index:

```php
$myArray = ['Value 1', 'Value 2', 'Value 3'];

$num = 2;

echo $myArray[$num];
```

```
Output:
<p>Value 3</p>
```

Looping through arrays

- You can use a for loop along with count to loop through each element in an array

```
$myArray = ['Value 1', 'Value 2', 'Value 3'];

for ($i = 0; $i < count($myArray); $i++) {
        echo '<p>' . $myArrray[$i] . '</p>';
}
```

```
Output:
<p>Value 1</p>
<p>Value 2</p>
<p>Value 3</p>
```

Strings are arrays of characters!

- A string is an array of characters and you can use it like an array
- You can loop through a string in the same way as an array

```php
$myString = 'hello world';

for ($i = 0; $i < strlen($myString); $i++) {
        echo '<p>' . $myString[$i] . '</p>';
}
```

```
Output:
<p>h</p>
<p>e</p>
<p>l</p>
<p>l</p>
<p>o</p>
<p> </p>
<p>w</p>
<p>o</p>
<p>r</p>
<p>l</p>
<p>d</p>
```

Strings

- You can read a character at a specific position in a string like an array:

```php
$myString = 'hello world';
echo $myString[2];
```

```
Output:
l
```

```php
$myString = 'hello world';
echo $myString[8];
```

```
Output:
r
```

Strings

- You can’t do this with numbers:

```
$myNum = 123;
echo $myNum[2];
```

```
Output:
```

- But you can convert the number to a string using strval() and then read the character

```
$myNum = 123;
$myNumAsString = strval($myNum);
echo $myNumAsString[2];
```

```
Output:
3
```

Remember this exercise?

```php
<?php
echo '<ul>';
for ($i = 0; $i < 10; $i++) {
        echo '<li>';

        if ($i === 0) {
                echo 'Zero';
        }
        else if ($i === 1) {
                echo 'One';
        }
        else if ($i === 2) {
                echo 'Two';
        }
        else if ($i === 3) {
                echo 'Three';
        }
        else if ($i === 4) {
                echo 'Four';
        }
        else if ($i === 5) {
                echo 'Five';
        }
        else if ($i === 6) {
                echo 'Six';
        }
        else if ($i === 7) {
                echo 'Seven';
        }

        echo '</li>';
}
echo '</ul>';
?>
```

Exercise 1

1. Print a list of the textual representation of each number (“One”, “Two”, etc) without using an if-else statement
   - Hint: Use an array to store each word
   - Do not use an if statement!
2. Optional - Difficulty: HARD Change the 9 times table exercise from last week to print the nine times table using the textual representation up to 12 times nine
   - One times Nine equals Nine
   - Two times Nine equals Eighteen
   - Three times Nine equals Twenty Seven
   - Four times Nine equals Thirty Six
3. Hint: Move the code to print out a number into its own function
4. Hint: A function can call itself

Exercise 1 Solution

- This exercise can be more easily solved with an array and a loop

```php
<?php
$myArray = [];

$myArray[0] = 'Zero';
$myArray[1] = 'One';
$myArray[2] = 'Two';
$myArray[3] = 'Three';
$myArray[4] = 'Four';
$myArray[5] = 'Five';
$myArray[6] = 'Six';
$myArray[7] = 'Seven';
$myArray[8] = 'Eight';
$myArray[9] = 'Nine';

echo '<ul>';
for ($i = 0; $i < count($myArray); $i++) {
        echo '<li>' . $myArray[$i] . '</li>';
}
echo '</ul>';
?>
```

```
Output:
<ul>
    <li>Zero</li>
    <li>One</li>
    <li>Two</li>
    <li>Three</li>
    <li>Four</li>
    <li>Five</li>
    <li>Six</li>
    <li>Seven</li>
    <li>Eight</li>
    <li>Nine</li>
</ul>
```

```php
$basicNumbers = [];
$basicNumbers[0] = 'Zero';
$basicNumbers[1] = 'One';
$basicNumbers[2] = 'Two';
$basicNumbers[3] = 'Three';
$basicNumbers[4] = 'Four';
$basicNumbers[5] = 'Five';
$basicNumbers[6] = 'Six';
$basicNumbers[7] = 'Seven';
$basicNumbers[8] = 'Eight';
$basicNumbers[9] = 'Nine';
$basicNumbers[10] = 'Ten';
$basicNumbers[11] = 'Eleven';
$basicNumbers[12] = 'Twelve';
$basicNumbers[13] = 'Thirteen';
$basicNumbers[14] = 'Fourteen';
$basicNumbers[15] = 'Fifteen';
$basicNumbers[16] = 'Sixteen';
$basicNumbers[17] = 'Seventeen';
$basicNumbers[18] = 'Eighteen';
$basicNumbers[19] = 'Ninteteen';

$tens = [];
$tens[2] = 'Twenty';
$tens[3] = 'Thirty';
$tens[4] = 'Forty';
$tens[5] = 'Fifty';
$tens[6] = 'Sixty';
$tens[7] = 'Seventy';
$tens[8] = 'Eighty';
$tens[9] = 'Ninety';
```

```php
function showNumber($num) {
	$result = '';
	$numAsString = strval($num);
	$firstDigit = $numAsString[0];

	if ($num > 100) {
		return $basicNumbers[$nextDigit] .
		' hundred and ' . showNumber($numAsString[1] . $numAsString[2]);
	}
	else if ($num > 20) {
		return $tens[$nextDigit] . ' ' . showNumber($numAsString[1]);
	}
	else {
		return $basicNumbers[$num];
	}
}

echo '<ul>';
for ($i = 0; $i < 13; $i++) {
	echo '<li>';
	echo  showNumber($i);
	echo ' times Nine equals ';
	echo  showNumber($i*9);
	echo '</li>';
}
echo '</ul>';
?>
```

Looping Through Arrays

- This results in less code and no need to write a very long if-else statement

Array Indexes

- It's possible that an array may not have a specific index set:

```php
<?php

$myArray = [];

$myArray[1] = 'Value 1';
$myArray[3] = 'Value 3';


echo '<p>' . $myArray[2] . '</p>';
?>
```

```
Output:

Error: Undefined index “2”
```

Unset Array Indexes

- This can cause problems with for loops

```php
<?php
$myArray = [];

$myArray[0] = 'Zero';
$myArray[1] = 'One';
$myArray[2] = 'Two';
$myArray[3] = 'Three';
$myArray[4] = 'Four';


$myArray[7] = 'Seven';
$myArray[8] = 'Eight';
$myArray[9] = 'Nine';

echo '<ul>';
for ($i = 0; $i < count($myArray); $i++) {
        echo '<li>' . $myArray[$i] . '</li>';
}
echo '</ul>';
?>
```

```
Output:

<ul>

      <li>Zero</li>
      <li>One</li>
      <li>Two</li>
      <li>Three</li>
      <li>Four</li>
      <li>Error: Undefined index "5"</li>
      <li>Error: Undefined index "6"</li>
      <li>Seven</li>
</ul>
```

Unset Array Indexes

- This causes two problems:
  - Error: undefined index” because an index that doesn't exist is being read
  - count() shows the number of elements in the array, not the maximum value.

Checking whether an index is set

- You can use the inbuilt php function isset() to test whether an array index has been set or not

```php
<?php
$myArray = [];
$myArray[0] = 'Zero';
$myArray[1] = 'One';
$myArray[2] = 'Two';
$myArray[3] = 'Three';
$myArray[4] = 'Four';

$myArray[7] = 'Seven';
$myArray[8] = 'Eight';
$myArray[9] = 'Nine';

if (isset($myArray[3])) {
        echo 'Index 3 is set';
}
else {
        echo 'Index 3 is not set';
}

if (isset($myArray[6])) {
        echo 'Index 6 is set';
}
else {
        echo 'Index 6 is not set';
}
?>
```

```
Output:

Index 3 is set
Index 6 is not set
```

isset function

- The isset function is unique, it allows you to test for existence without generating an error
- Normally if you try to access an index that doesn't exit an error will be displayed
- However, if this is done inside an isset() call it will not

```php
<?php
$myArray = [];

echo $myArray[5]; //ERROR: undefined index 5

if ($myArray[5] === 'five') {}  //ERROR: undefined index 5

echo isset($myArray[5]); //NO ERROR; prints "false"
?>
```

Using isset() inside a loop

```php
<?php

$myArray = [];

$myArray[0] = 'Zero';
$myArray[1] = 'One';
$myArray[2] = 'Two';
$myArray[3] = 'Three';
$myArray[4] = 'Four';


$myArray[7] = 'Seven';
$myArray[8] = 'Eight';
$myArray[9] = 'Nine';

echo '<ul>';
for ($i = 0; $i < count($myArray); $i++) {
        if (isset($myArray[$i])) {
                echo '<li>' . $myArray[$i] . '</li>';
        }
}
echo '</ul>';
?>
```

```
Output:

<ul>
      <li>Zero</li>
      <li>One</li>
      <li>Two</li>
      <li>Three</li>
      <li>Four</li>
      <li>Seven</li>
</ul>
```

Using isset() inside a loop

```php
<?php
$myArray = [];

$myArray[0] = 'Zero';
$myArray[1] = 'One';
$myArray[2] = 'Two';
$myArray[3] = 'Three';
$myArray[4] = 'Four';


$myArray[7] = 'Seven';
$myArray[8] = 'Eight';
$myArray[9] = 'Nine';

echo count($myArray);
?>
```

```
Output: 8
```

```php
<?php

echo '<ul>';
for ($i = 0; $i < count($myArray); $i++) {
        echo '<li>' . $myArray[$i] . '</li>';
}
echo '</ul>';
?
```

This loop will never reach index 8 or 9!

Showing the contents of an array

- You can display the contents of an array (both keys and values) using the inbuilt function var_dump

```php
<?php
$myArray = [];

$myArray[0] = 'Zero';
$myArray[1] = 'One';
$myArray[2] = 'Two';
$myArray[3] = 'Three';
$myArray[4] = 'Four';


$myArray[7] = 'Seven';
$myArray[8] = 'Eight';
$myArray[9] = 'Nine';

var_dump($myArray);
?>
```

```
array (size=8)
  0 => string 'Zero' (length=4)
  1 => string 'One' (length=3)
  2 => string 'Two' (length=3)
  3 => string 'Three' (length=5)
  4 => string 'Four' (length=4)
  7 => string 'Seven' (length=5)
  8 => string 'Eight' (length=5)
  9 => string 'Nine' (length=4)
```

Showing the contents of an array

- var_dump will display all the set keys and their values
- This is very useful for debugging
- But it shouldn't be used for anything else

Writing to arrays

- You can write a value to an array by specifying an index:

```
$myArray[2] = 'Value 2';
```

- If there is already something stored under that index, it will be overwritten:

```
$myArray = [];

$myArray[2] = 'value 2';
echo $myArray[2];

$myArray[2] = 'value updated';
echo $myArray[2];
```

```
Output:
value 2
value updated
```

Array Size

- In PHP an array does not have a fixed size. Elements can be added at any point during the program
- You can append an element to the next available index using the code:

```
$myArray[] = 'next value';
```

Adding to the end of an array

```php
<?php

$myArray = [];
var_dump($myArray);

$myArray[] = 'value 1';
var_dump($myArray);

$myArray[] = 'value 2';
var_dump($myArray);

$myArray[] = 'value 3';
var_dump($myArray);
?>
```

```
Output:
array (size=0)
  Empty

array (size=1)
  0 => string 'value 1' (length=7)

array (size=2)
  0 => string 'value 1' (length=7)
  1 => string 'value 2' (length=7)

array (size=3)
  0 => string 'value 1' (length=7)
  1 => string 'value 2' (length=7)
  2 => string 'value 3' (length=7)
```

foreach

- There is a second type of for loop which will loop through all elements in an array
- This is foreach and it does not use a counter
- The foreach loop takes an array and loops over it

foreach

```php
<?php

$myArray = [];

$myArray[0] = 'Zero';
$myArray[1] = 'One';
$myArray[2] = 'Two';
$myArray[3] = 'Three';
$myArray[4] = 'Four';

$myArray[7] = 'Seven';
$myArray[8] = 'Eight';
$myArray[9] = 'Nine';


foreach ($myArray as $key => $value) {
        echo '<p>Key : ' . $key . ', Value: ' . $value  '</p>';
}

?>
```

```
Output:
<p>Key: 0, Value: Zero</p>
<p>Key: 1, Value: One</p>
<p>Key: 2, Value: Two</p>
<p>Key: 3, Value: Three</p>
<p>Key: 4, Value: Four</p>
<p>Key: 7, Value: Five</p>
<p>Key: 8, Value: Eight</p>
<p>Key: 9, Value: Nine</p>
```

foreach

- Foreach is usually simpler than for to loop through an array
- You should use foreach instead of for when looping over an array
- Foreach will only loop over keys that are set

array_keys

- In PHP array keys can be either integers or strings

```php
<?php

$myArray = [];

//String Key
$myArray['stringKey'] = 'value 1';

//Numeric key
$myArray[5] = 'value 2';

var_dump($myArray);

?>
```

```
Output:

array (size=2)
  'stringKey' => string 'value 1' (length=7)
  5 => string 'value 2' (length=7)
```

array_keys

```php
<?php

$myArray = [];


$myArray['one'] = 'value 1';
$myArray['two'] = 'value 2';

echo $myArray['one'];

?>
```

```
Output:

value 1
```

array_keys

- You can also use foreach with arrays that have string keys

```php
<?php

$myArray = [];

$myArray['one'] = 'value 1';
$myArray['two'] = 'value 2';


foreach ($myArray as $key => $value) {
        echo '<p>Key : ' . $key . ', Value: ' . $value . '</p>';
}

?>
```

```
Output:

<p>Key: one, Value: value 1</p>
<p>Key: two, Value: value 2</p>
```

User input - Basics

- PHP has several “superglobals” which are variables that are available at any point in the program's code
- These are generally used for input from users
- There are two ways of capturing input from users over the web:
  - GET
  - POST

$_GET

- GET is a URI Variable and part of the HTTP protocol
- You can change the value of the PHP $_GET variable by changing the URI
- Until now you have accessed PHP scripts using the file name
- e.g. http://example.com/file.php

$_GET

- $_GET is an array that uses string keys
- Instead of setting the array contents in the PHP code, PHP automatically sets $_GET based on the URL the page has been accessed on
- You can access the page on
- file.php?name=John and it will set the $_GET variable's name key to John

$_GET

- file.php contents

```php
<?php
echo 'Hello ' . $_GET['name'];
?>
```

- Going to http://v.je/file.php will generate the output:

```
Hello ERROR: Undefined Index “name”
```

$_GET

- It will cause an error because the name key has not been set
- To set a key you can amend the URL with ?name=John
- http://v.je/file.php?name=John

```php
<?php

echo 'Hello ' . $_GET['name'];

?>
```

```
Hello John
```

$_GET

- You can set any key you want
- http://v.je/file.php?num=123

```php
<?php

var_dump($_GET);

?>
```

```
array (size=1)
  'num' => string '123' (length=3)
```

$_GET

- You can use $_GET variables like any other variable:
- http://v.je/file.php?num=6

```php
<?php

echo '<ul>';
for ($i = 0; $i < $_GET['num']; $++) {
        echo '<li>' . $i . '</li>';
}
echo '</ul>';

?>
```

```
Output:
<ul>
<li>0</li>
<li>1</li>
<li>2</li>
<li>3</li>
<li>4</li>
<li>5</li>
</ul>
```

Multiple $_GET variables

- You can specify more than one variable by separating them with am ampersand (&)
- file.php?key1=value1&key2=value2

```php
<?php

var_dump($_GET);

?>
```

```
array (size=1)
  'key1' => string 'value1' (length=6)
  'key2' => string 'value2' (length=6)
```

```php
<?php

echo '<p>' . $_GET['key1'] . '</p>';

echo '<p>' . $_GET['key2'] . '</p>';

?>
```

```
<p>value1</p>
<p>value2</p>
```

Exercise 2

1. Create a file double.php which takes a GET variable of a number and doubles it:
   - If you go to double.php?num=11 it should print 22. It must work for any number and not just 11!
2. Create a file called multiply.php that takes two numbers and multiplies one by the other
   - If you go to multiply.php?num1=4&num2=6 it should print 24, it must work for any numbers!
3. Create a file called loop.php that takes two numbers and prints out everything between `start` and `end`
   - If you go to loop.php?start=84&end=88 it should print 84 85 86 87 88 in a list, it must work for any numbers!
4. Optional – Difficulty: Medium – Create a function called isEven() that takes one argument and returns true or false depending on whether the argument is even. isEven(4) should return true, isEven(5) should return false
   - Use the function with a GET variable so you can visit isEven.php?num=4 and it will print “4 is an even number” or isEven.php?num=5 and it will print “5 is not an even number”.

Solutions

```php
//double.php
<?php
echo  $_GET['num']*2;
?>
```

```php
//multiply.php
<?php
echo  $_GET['num1']*$_GET['num2'];
?>
```

```php
//loop.php
<?php

echo '<ul>';
for ($i = $_GET['start']; $i < $_GET['end']; $++) {
        echo '<li>' . $i . '</li>';
}
echo '</ul>';

?>
```

```php
//isEven.php
<?php
function isEven($num) {
      if ($num % 2 == 0) {
            return true;
      }
      else {
            return false;
      }
}

if (isEven($_GET['num'])) {
      echo  $_GET['num'] . ' is an even number';
}
else {
      echo  $_GET['num'] . ' is not an even number';
}
?>
```

Removing elements from an array

- Once an array has been created, you can remove elements using the unset function

```php
<?php

$myArray = [];

$myArray['key1'] = 'value 1';
$myArray['key2'] = 'value 2';
$myArray['key3'] = 'value 3';


var_dump($myArray);

unset($myArray['key2']);

var_dump($myArray);
?>
```

```
array (size=3)
  'key1' => string 'value 1' (length=7)
  'key2' => string 'value 2' (length=7)
  'key3' => string 'value 3' (length=7)


array (size=2)
  'key1' => string 'value 1' (length=7)
  'key3' => string 'value 3' (length=7)
```

Arrays and Functions

- Last week finished with this function:

```php
<?php

function add($num1, $num2) {
        $result = $num1 + $num2;
        return $result;

}

$result1 = add(5, 10);
$result1 = add(99, 100);

echo '<p>Result 1 is  ' . $result1 . '</p>';
echo '<p>Result 2 is  ' . $result2 . '</p>';

?>
```

Arrays and Functions

- The function takes two numbers as arguments and adds them together

```php
<?php

function add($num1, $num2) {
        $result = $num1 + $num2;
        return $result;

}

$result1 = add(5, 10);
$result1 = add(99, 100);

echo '<p>Result 1 is  ' . $result1 . '</p>';
echo '<p>Result 2 is  ' . $result2 . '</p>';
```

```
Output:
<p>Result 1 is 15</p>
<p>Result 2 is 199</p>
```

Functions

- To add 3 numbers together you'd need to add an third argument:

```php
<?php

function add($num1, $num2, $num3) {
        $result = $num1 + $num2 + $num3;
        return $result;

}

$result1 = add(5, 10, 1);
$result1 = add(99, 100, 10);

echo '<p>Result 1 is  ' . $result1 . '</p>';
echo '<p>Result 2 is  ' . $result2 . '</p>';
```

```
Output:
<p>Result 1 is 16</p>
<p>Result 2 is 209</p>
```

Functions

- And to add 4 you would need to add a fourth:

```php
<?php

function add($num1, $num2, $num3, $num4) {
        $result = $num1 + $num2 + $num3 + $num3;
        return $result;

}

$result1 = add(5, 10, 1, 3);
$result1 = add(99, 100, 10, 7);

echo '<p>Result 1 is  ' . $result1 . '</p>';
echo '<p>Result 2 is  ' . $result2 . '</p>';

?>
```

```
Output:
<p>Result 1 is 19</p>
<p>Result 2 is 216</p>
```

Functions

- This is inflexible and requires writing a new function for each number of argument
- Instead, a function can take an array as an argument

```php
<?php

function myFunction($argument) {
        var_dump($argument);
}


$myArray = [];

$myArray['key1'] = 'value 1';
$myArray['key2'] = 'value 2';
$myArray['key3'] = 'value 3';


myFunction($myArray);
?>
```

```
array (size=3)
  'key1' => string 'value 1' (length=7)
  'key2' => string 'value 2' (length=7)
  'key3' => string 'value 3' (length=7)
```

Functions

- The whole array with all its contents is passed into the method.
- The add method could be rewritten like this:

```php
<?php

function add($array) {
     $result = 0;

     foreach ($array as $key => $value) {
        $result = $result +  $value;
     }

     return $result;
}

?>
```

Functions

- And called with an array of numbers

```php
<?php

function add($array) {
     $result = 0;

        foreach ($array as $key => $value) {
            $result = $result +  $value;
        }

        return $result;
}

$myArray = [2, 5, 6, 9];

$total = add($myArray);

echo $total;
```

```
Output:
22
```

Functions

- Which can be called with an array of any size:

```php
<?php
function add($array) {
     $result = 0;

        foreach ($array as $key => $value) {
            $result = $result +  $value;
        }

        return $result;
}

$myArray = [2, 5, 6, 9];
$total = add($myArray);

echo $total;

$myArray2 = [1, 2];
$total = add($myArray2);

echo $total;

$myArray3 = [5, 5, 5, 2];
$total = add($myArray3);

echo $total;
```

```
Output:
22
3
17
```

Passing arrays into functions

- When you pass an array into a function, it passes a copy
- Changes to the array in the function, will not be reflected elsewhere in the program

Arrays and functions

```php
<?php

function processArray($array) {
        $array[] = 4;
        $array[] = 5;

        echo 'Array in processArray:';
        var_dump($array);
}

$myArray = [1, 2, 3];

echo 'Original array:';
var_dump($myArray);


processArray($myArray);

echo 'Array after processArray:';
var_dump($myArray);

?>
```

```
Original array:
array (size=3)
  0 => int 1
  1 => int 2
  2 => int 3

Array in processArray:
array (size=5)
  0 => int 1
  1 => int 2
  2 => int 3
  3 => int 4
  4 => int 5


Array after processArray:
array (size=3)
  0 => int 1
  1 => int 2
  2 => int 3
```

Arrays

- To change the array, you have to return it back to where the function was called

```php
<?php
function processArray($array) {
        $array[] = 4;
        $array[] = 5;

        echo 'Array in processArray:';
        var_dump($array);
         return $array;
}

$myArray = [1, 2, 3];

echo 'Original array:';
var_dump($myArray);


$myArray =  processArray($myArray);

echo 'Array after processArray:';
var_dump($myArray);

?>
```

```
Original array:
array (size=3)
  0 => int 1
  1 => int 2
  2 => int 3

Array in processArray:
array (size=5)
  0 => int 1
  1 => int 2
  2 => int 3
  3 => int 4
  4 => int 5


Array after processArray:
array (size=5)
  0 => int 1
  1 => int 2
  2 => int 3
  3 => int 4
```

Exercise 3

- Write a program which simulates a game of rock-paper-scissors.
- When the page loads there should be a choice of 3 links, one for rock, one for paper and one for scissors
- When a link is clicked it should display something like:
  - “You chose rock. The computer chose scisssors, you win!”
- Hint: You will need to use the random number generator to make the computer pick a move.
- If both players chose the same print “It’s a tie”
- Grade B: Can you do this with a single .php file?
- Grade A: Extend the final screen to include the first page with the words “Play again?” and links to select your move.

Exercise 4

- Use an array to store the following people and their extension numbers
  - John – 389
  - Kate – 012
  - Sue – 586
  - Dave – 675
  - Jo – 434
- Using a loop print out each of the names as links in a list
- When you click on a name, display (for example) “John is on extension 389”

Exercise 5

1. Write a function that takes an array of numbers and prints the smallest
   - e.g. $array = [7, 9, 4, 6, 7, 13, 7, 9]; findSmallest($array) would print “13”
   - The function should be able to work with any array!
2. Write a function that takes an array of numbers and prints the largest
3. Allow the array to be created using $_GET. You can use file.php?num1=7&num2=6&num3=7&num4=8 and show both the largest and the smallest of the entered numbers
4. Difficulty: medium After allowing entering numbers via the URL, display all the numbers on the page in a list and add two links to the page: “Show smallest” and “Show largest” which, once clicked, shows either the smallest or largest of the entered numbers
5. Difficulty: medium Adjust (4) so that the links are “Sort low to high” and “Sort high to low”

Exercise 6

- Difficulty: Very hard! Given an array of any reasonable size (up to at least 10 items) print every possible combination of the entries e.g. given the array [‘red’, ‘green’, ‘blue’] print out:
  - red
  - blue
  - red blue
  - green
  - red green
  - blue green
  - red blue green

No solution available!

```php
<?php

if (!isset($_GET['choice'])) {
        echo '<ul>';
        echo '<li><a href="rps.php?choice=rock">Rock</a></li>';
        echo '<li><a href="rps.php?choice=paper">Paper</a></li>';
        echo '<li><a href="rps.php?choice=scissors">Scissors</a></li>';
        echo '</ul>';
}
else {
        $computerChoices = ['rock', 'paper', 'scissors'];
        $randChoice = rand(0, 2);

        //random computer choice
        $computer = $computerChoices[$randChoice];
        $player = $_GET['choice'];

        if ($computer == $player) {
                echo 'Draw! Both players chose ' .  $player;
        }
        else {
                echo 'You chose ' . $player . ' and the computer chose ' . $computer . '. ';

                if ($computer == 'rock' && $player == 'scissors') {
                        echo 'Computer wins!';
                }
                else if ($computer == 'rock' && $player == 'paper') {
                        echo 'You win!';
                }
                else if ($computer == 'paper' && $player == 'rock') {
                        echo 'Computer wins!';
                }
                else if ($computer == 'paper' && $player == 'scissors') {
                        echo 'You win!';
                }
                else if ($computer == 'scissors' && $player == 'paper') {
                        echo 'Computer wins!';
                }
                else if ($computer == 'scissors' && $player == 'rock') {
                        echo 'You win!';
                }
        }
}
```

```php
<?php
$phoneNumbers = [];
$phoneNumbers['John'] = '389';
$phoneNumbers['Kate'] = '012';
$phoneNumbers['Sue'] = '586';
$phoneNumbers['Dave'] = '675';
$phoneNumbers['Jo'] = '464';

if (isset($_GET['name'])) {
        echo $_GET['name'] . ' is on extension number ' . $phoneNumbers[$_GET['name']];
}
else {
        echo '<ul>';
        foreach ($phoneNumbers as $name => $number) {
                echo '<li><a href="phone.php?name=' . $name . '">' . $name . '</a></li>';
        }
        echo '</ul>';
}
```

```php
<?php

function findSmallest($array) {
        $currentSmallest = 9999999;

        foreach ($array as $value) {
                if ($value < $currentSmallest) {
                        $currentSmallest = $value;
                }
        }

        return $currentSmallest;
}

$array = [7, 9, 4, 6, 7, 13, 7, 9];
echo findSmallest($array);
```

```php
<?php

function findLargest($array) {
        $currentLargest = 0;

        foreach ($array as $value) {
                if ($value > $currentLargest) {
                        $currentLargest = $value;
                }
        }

        return $currentLargest;
}

$array = [7, 9, 4, 6, 7, 13, 7, 9];
echo findSmallest($array);
```

```php
echo '<p>Largest number: ' . findLargest($_GET) . '</p>';
echo '<p>Smallest number: ' . findSmallest($_GET) . '</p>';
```

```php
function sortLowHigh($array) {
        $result = [];


        while (count($array) > 0) {
                $smallest = findSmallest($array);
                $result[] = $smallest;

                foreach ($array as $key => $value) {
                        if ($value == $smallest) {
                                unset($array[$key]);
                                break;
                        }
                }
        }

        return $result;
}



$sorted = sortLowHigh($_GET);

echo '<ul>';
foreach ($sorted as $num) {
        echo '<li>' . $num . '</li>';
}


echo '</ul>';
```

```php
function sortHighLow($array) {
        $result = [];


        while (count($array) > 0) {
                $smallest = findLargest($array);
                $result[] = $smallest;

                foreach ($array as $key => $value) {
                        if ($value == $smallest) {
                                unset($array[$key]);
                                break;
                        }
                }
        }

        return $result;
}



$sorted = sortLowHigh($_GET);

echo '<ul>';
foreach ($sorted as $num) {
        echo '<li>' . $num . '</li>';
}


echo '</ul>';
```