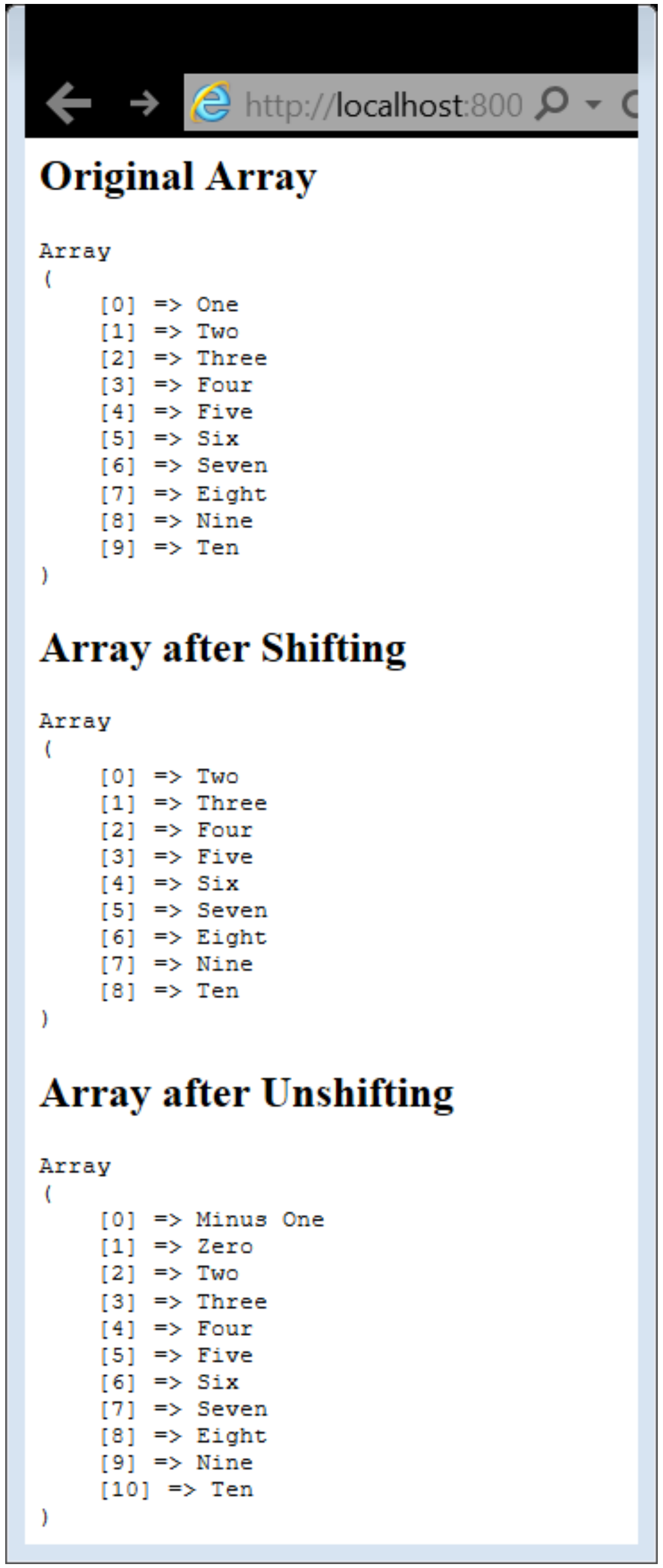
Array Manipulation

Adding and Removing from the beginning of an Array

- The array_shift() function removes the first element from the beginning of an array.
- The array_unshift() function adds one or more elements to the beginning of an array.

The array_shift_unshift.php

```
<?php
$Numbers = array( "One",
"Two", "Three", "Four", "Five", "Six",
"Seven", "Eight", "Nine", "Ten");
echo "<h2>Original Array</h2>\n";
echo "<pre>\n";
print_r($Numbers);
echo "</pre>\n"; array_shift($Numbers);
echo "<h2>Array after Shifting</h2>\n";
echo "<pre>\n";
print_r($Numbers);
echo "</pre>\n";
array_unshift($Numbers, "Minus One",
"Zero");
echo "<h2>Array after Unshifting</h2>
\n";
echo "<pre>\n";
print_r($Numbers);
echo "</pre>\n";
```



Original Array

```
Array
(
    [0] => One
    [1] => Two
    [2] => Three
    [3] => Four
    [4] => Five
    [5] => Six
    [6] => Seven
    [7] => Eight
    [8] => Nine
    [9] => Ten
)
```

Array after Shifting

```
Array
(
    [0] => Two
    [1] => Three
    [2] => Four
    [3] => Five
    [4] => Six
    [5] => Seven
    [6] => Eight
    [7] => Nine
    [8] => Ten
)
```

Array after Unshifting

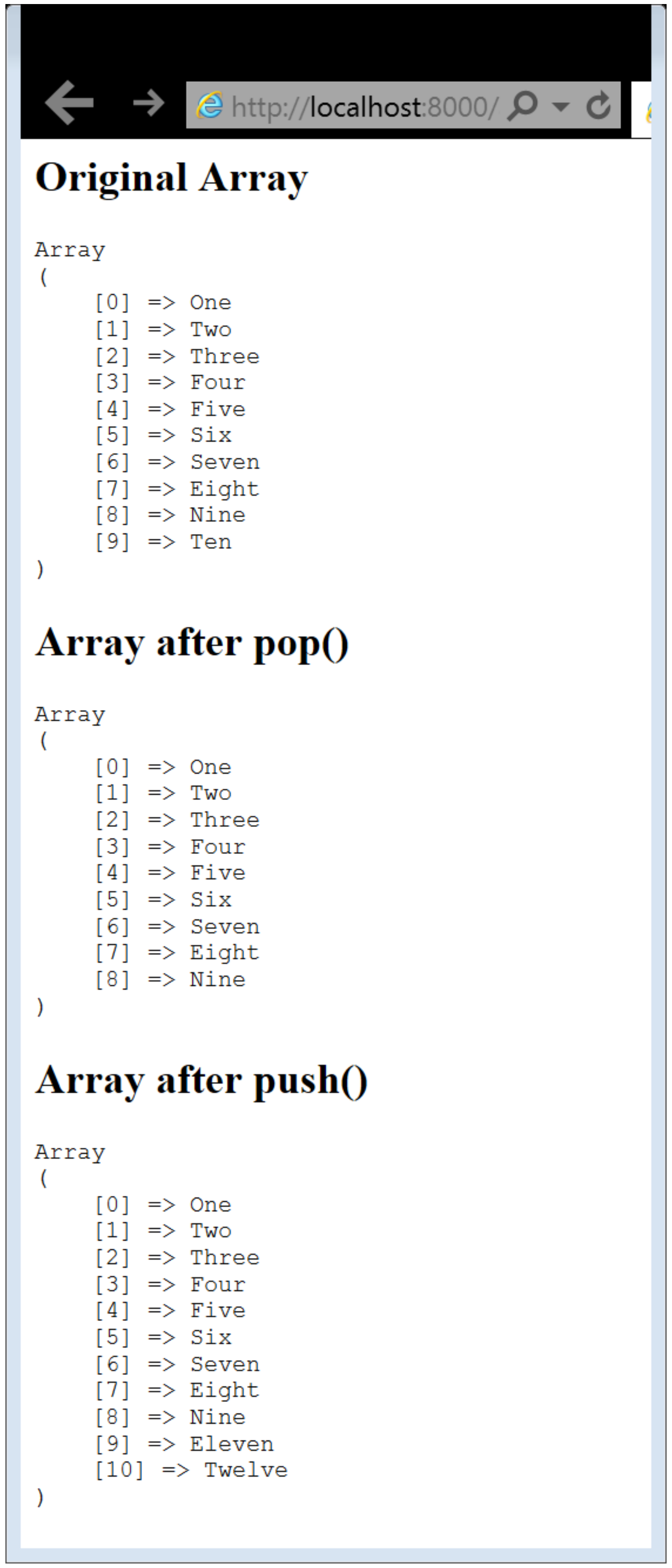
```
Array
(
    [0] => Minus One
    [1] => Zero
    [2] => Two
    [3] => Three
    [4] => Four
    [5] => Five
    [6] => Six
    [7] => Seven
    [8] => Eight
    [9] => Nine
    [10] => Ten
)
```

Adding and Removing from the end of an Array

- The **array_pop**() function removes the last element from the end of an array
- the **array_push**() function adds one or more elements to the end of an array. You pass to the array_pop() function the name of the array whose last element you want to remove.

The array_pop_push.php

```
<?php
$Numbers = array( "One",
"Two", "Three", "Four", "Five", "Six", "Seven", "Eight",
"Nine", "Ten");
echo "<h2>Original Array</h2>\n";
echo "<pre>\n";
print_r($Numbers);
echo "</pre>\n";
array_pop($Numbers);
echo "<h2>Array after pop()</h2>\n";
echo "<pre>\n";
print_r($Numbers);
echo "</pre>\n";
array_push($Numbers, "Eleven", "Twelve");
echo "<h2>Array after push()</h2>\n";
echo "<pre>\n";
print_r($Numbers);
echo "</pre>\n";
```



Original Array

```
Array
(
    [0] => One
    [1] => Two
    [2] => Three
    [3] => Four
    [4] => Five
    [5] => Six
    [6] => Seven
    [7] => Eight
    [8] => Nine
    [9] => Ten
)
```

Array after pop()

```
Array
(
    [0] => One
    [1] => Two
    [2] => Three
    [3] => Four
    [4] => Five
    [5] => Six
    [6] => Seven
    [7] => Eight
    [8] => Nine
)
```

Array after push()

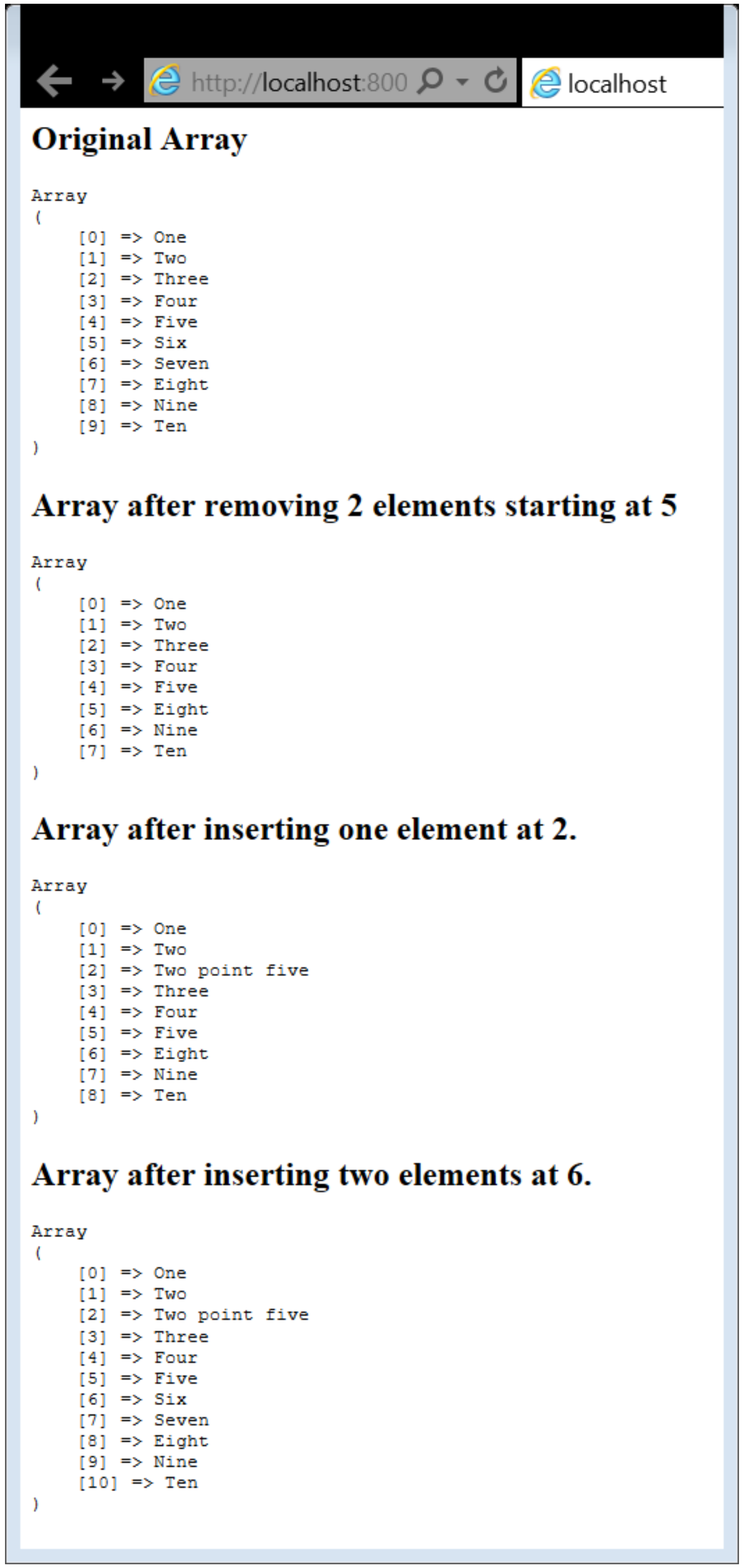
```
Array
(
    [0] => One
    [1] => Two
    [2] => Three
    [3] => Four
    [4] => Five
    [5] => Six
    [6] => Seven
    [7] => Eight
    [8] => Nine
    [9] => Eleven
    [10] => Twelve
)
```

Adding and Removing Elements within an Array

- The **array_splice**() function allows you to add or remove elements located anywhere else in an array. After adding or removing array elements, the array_splice() function also renumbers the indexes for an array.
- The syntax of the **array_splice**() is array_splice(array_name, start_index, number_to_delete, values_to_insert);.

The array_splice. php

```
$Numbers = array( "One",
"Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine",
"Ten");
echo "<h2>Original Array</h2>\n";
echo "<pre>\n";
print_r($Numbers);
echo "</pre>\n";
array_splice($Numbers, 5, 2);
echo "<h2>Array after removing 2 elements starting at 5</
h2>\n";
echo "<pre>\n";
print_r($Numbers);
echo "</pre>\n";
array_splice($Numbers, 2, 0, "Two point five");
echo "<h2>Array after inserting one element at 2.</h2>\n";
echo "<pre>\n";
print_r($Numbers);
echo "</pre>\n";
array_splice($Numbers, 6, 0, array("Six", "Seven"));
echo "<h2>Array after inserting two elements at 6.</h2>
\n";
echo "<pre>\n";
print_r($Numbers);
echo "</pre>\n";
```



http://localhost:800

localhost

Original Array

```
Array
(
    [0] => One
    [1] => Two
    [2] => Three
    [3] => Four
    [4] => Five
    [5] => Six
    [6] => Seven
    [7] => Eight
    [8] => Nine
    [9] => Ten
)
```

Array after removing 2 elements starting at 5

```
Array
(
    [0] => One
    [1] => Two
    [2] => Three
    [3] => Four
    [4] => Five
    [5] => Eight
    [6] => Nine
    [7] => Ten
)
```

Array after inserting one element at 2.

```
Array
(
    [0] => One
    [1] => Two
    [2] => Two point five
    [3] => Three
    [4] => Four
    [5] => Five
    [6] => Eight
    [7] => Nine
    [8] => Ten
)
```

Array after inserting two elements at 6.

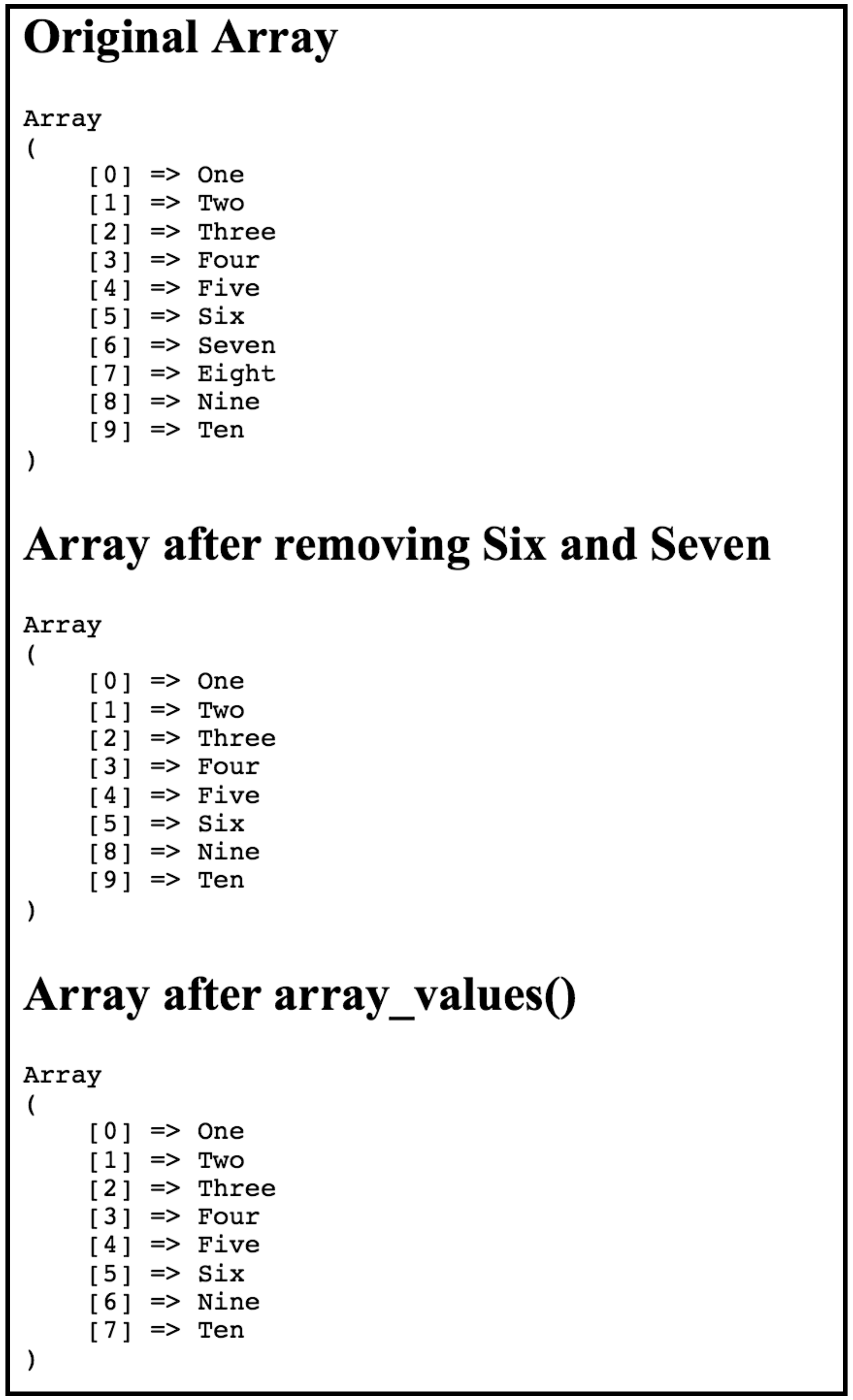
```
Array
(
    [0] => One
    [1] => Two
    [2] => Two point five
    [3] => Three
    [4] => Four
    [5] => Five
    [6] => Six
    [7] => Seven
    [8] => Eight
    [9] => Nine
    [10] => Ten
)
```

The unset() and array_values()

- The unset() function to remove array elements and other variables. To remove multiple elements, separate each element name with a comma.
- However, the unset() function is that it does not renumber the remaining elements in the array.
- To renumber an indexed array’s elements, you need to use the array_values() function.

The array_unset.php

```php
<?php

$Numbers = array(
    "One",
    "Two",
    "Three",
    "Four",
    "Five",
    "Six",
    "Seven",
    "Eight",
    "Nine",
    "Ten");
echo "<h2>Original Array</h2>\n";
echo "<pre>\n";
print_r($Numbers);
echo "</pre>\n";
unset($Numbers[6], $Numbers[7]);
echo "<h2>Array after removing Six and Seven</h2>\n";
echo "<pre>\n";
print_r($Numbers);
echo "</pre>\n";
$Numbers = array_values($Numbers);
echo "<h2>Array after array_values()</h2>\n";
echo "<pre>\n";
print_r($Numbers);
echo "</pre>\n";
```



Original Array

```
Array
(
    [0] => One
    [1] => Two
    [2] => Three
    [3] => Four
    [4] => Five
    [5] => Six
    [6] => Seven
    [7] => Eight
    [8] => Nine
    [9] => Ten
)
```

Array after removing Six and Seven

```
Array
(
    [0] => One
    [1] => Two
    [2] => Three
    [3] => Four
    [4] => Five
    [5] => Six
    [8] => Nine
    [9] => Ten
)
```

Array after array_values()

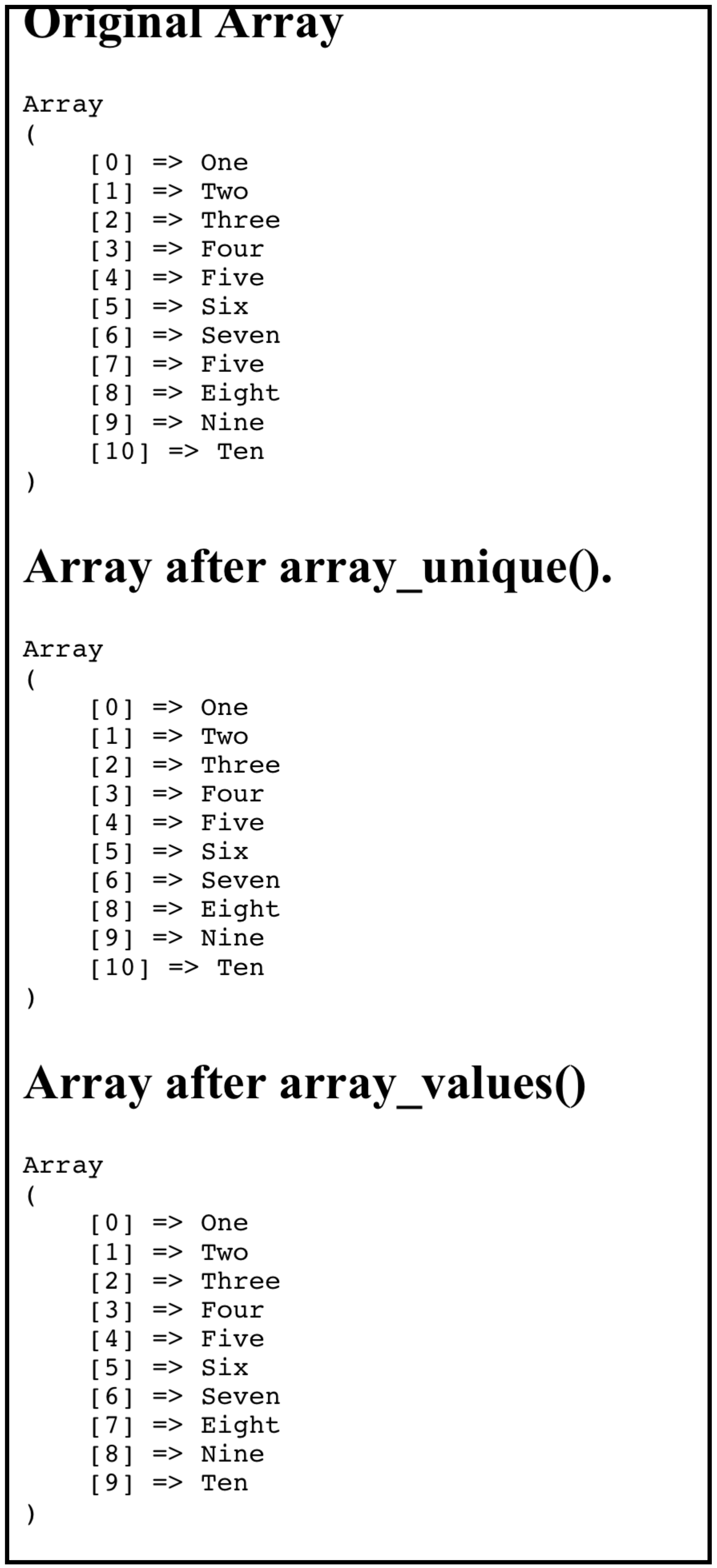
```
Array
(
    [0] => One
    [1] => Two
    [2] => Three
    [3] => Four
    [4] => Five
    [5] => Six
    [6] => Nine
    [7] => Ten
)
```

Removing Duplicates

- The array_unique() function to remove duplicate elements from an array. You pass to this function the name of the array from which you want to remove duplicate elements.
- The array_unique() function does not operate directly on an array. Instead, it returns a new array with the renumbered indexes.

The array_unique

```
<?php

$Numbers = array(
    "One",
    "Two",
    "Three",
    "Four",
    "Five",
    "Six",
    "Seven",
    "Five",
    "Eight",
    "Nine",
    "Ten");
echo "<h2>Original Array</h2>\n";
echo "<pre>\n";
print_r($Numbers);
echo "</pre>\n";
$Numbers = array_unique($Numbers);
echo "<h2>Array after array_unique().</h2>\n";
echo "<pre>\n";
print_r($Numbers);
echo "</pre>\n";
$Numbers = array_values($Numbers);
echo "<h2>Array after array_values()</h2>\n";
echo "<pre>\n";
print_r($Numbers);
echo "</pre>\n";
```



Original Array

```
Array
(
    [0] => One
    [1] => Two
    [2] => Three
    [3] => Four
    [4] => Five
    [5] => Six
    [6] => Seven
    [7] => Five
    [8] => Eight
    [9] => Nine
    [10] => Ten
)
```

Array after array_unique().

```
Array
(
    [0] => One
    [1] => Two
    [2] => Three
    [3] => Four
    [4] => Five
    [5] => Six
    [6] => Seven
    [8] => Eight
    [9] => Nine
    [10] => Ten
)
```

Array after array_values()

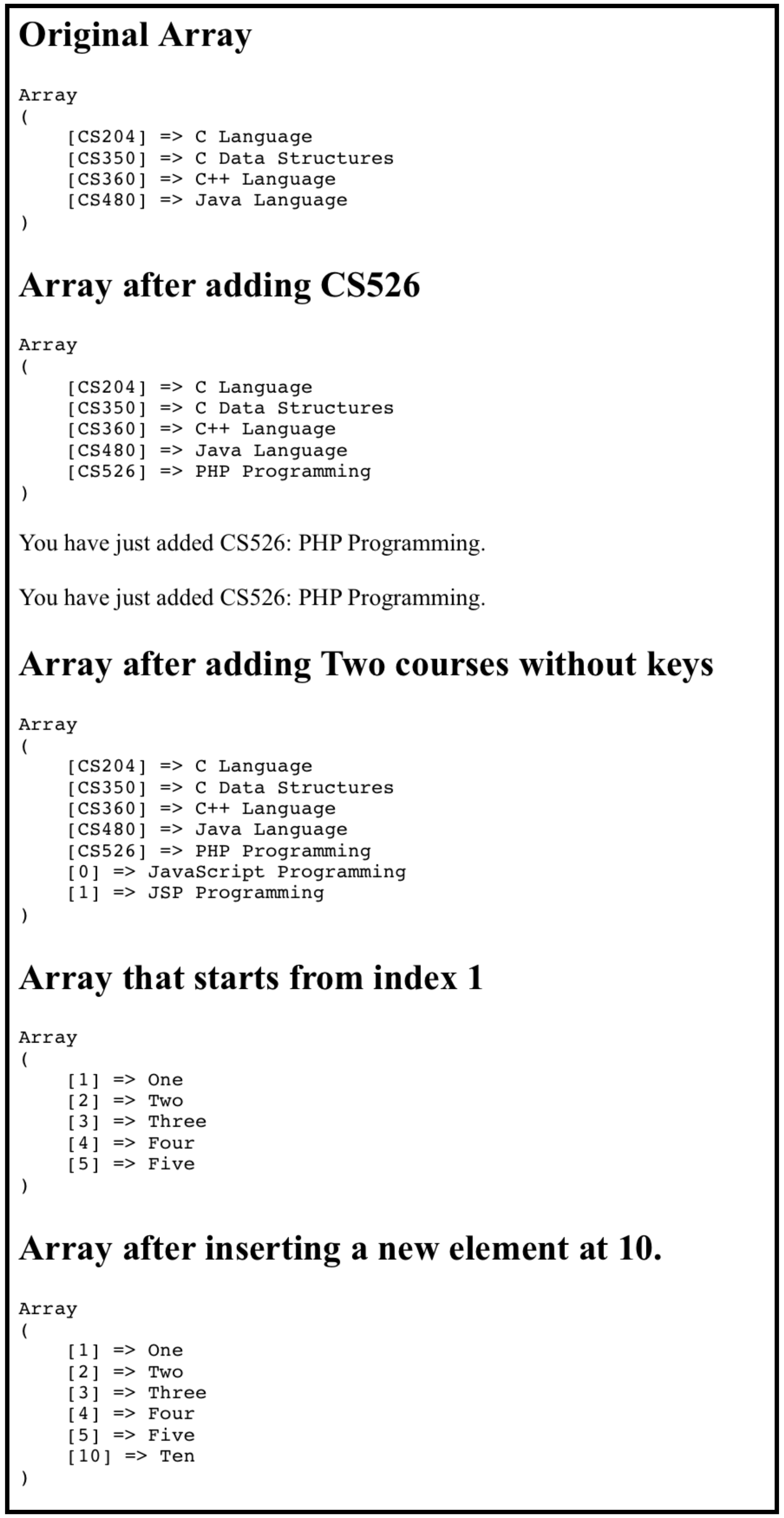
```
Array
(
    [0] => One
    [1] => Two
    [2] => Three
    [3] => Four
    [4] => Five
    [5] => Six
    [6] => Seven
    [7] => Eight
    [8] => Nine
    [9] => Ten
)
```

Associative Arrays

- PHP creates indexed arrays by default with a starting index of 0.
- Associative arrays that allows you to use any alphanumeric keys that you want for the array elements. You specify an element's key by using the array operator (=>) in the array() construct. The syntax for declaring and initializing an associative array is as follows:
  - $array_name = array(key => value, ...);
- If you create an associative array and then add a new element without specifying a key, PHP automatically assumes that the array is indexed and assigns the new element an index of 0 or the next available integer.
- Associative arrays also allows you to start the numbering of indexed arrays at any integer you want.
- In other programming languages, if you declare an array and use a starting index other than 0, empty elements are created for each index between 0 and the index value you specify. In PHP, only the elements specified are created, regardless of the index. No empty elements are created.

The associative_arrays.php

```
<?php
$courses = array(
   "CS204" => "C Language",
   "CS350" => "C Data Structures",
   "CS360" => "C++ Language",
   "CS480" => "Java Language",
);

echo "<h2>Original Array</h2>\n";
echo "<pre>\n";
print_r($courses);
echo "</pre>\n";

$courses["CS526"] = "PHP Programming";
echo "<h2>Array after adding CS526</h2>\n";
echo "<pre>\n";
print_r($courses);
echo "</pre>\n";

echo "<p>You have just added CS526: " .$courses["CS526"] . ".</p>\n";
echo "<p>You have just added CS526: {$courses["CS526"]}.</p>\n";

$courses[] = "JavaScript Programming";
$courses[] = "JSP Programming";
echo "<h2>Array after adding Two courses without keys</h2>\n";
echo "<pre>\n";
print_r($courses);
echo "</pre>\n";

$numbers = array(
   1 => "One",
   2 => "Two",
   3 => "Three",
   4 => "Four",
   5 => "Five"
);

echo "<h2>Array that starts from index 1</h2>\n";
echo "<pre>\n";
print_r($numbers);
echo "</pre>\n";

$numbers[10] = "Ten";
echo "<h2>Array after inserting a new element at 10.</h2>\n";
echo "<pre>\n";
print_r($numbers);
echo "</pre>\n";
```



Original Array

```
Array
(
    [CS204] => C Language
    [CS350] => C Data Structures
    [CS360] => C++ Language
    [CS480] => Java Language
)
```

Array after adding CS526

```
Array
(
    [CS204] => C Language
    [CS350] => C Data Structures
    [CS360] => C++ Language
    [CS480] => Java Language
    [CS526] => PHP Programming
)
```

You have just added CS526: PHP Programming.

You have just added CS526: PHP Programming.

Array after adding Two courses without keys

```
Array
(
    [CS204] => C Language
    [CS350] => C Data Structures
    [CS360] => C++ Language
    [CS480] => Java Language
    [CS526] => PHP Programming
    [0] => JavaScript Programming
    [1] => JSP Programming
)
```

Array that starts from index 1

```
Array
(
    [1] => One
    [2] => Two
    [3] => Three
    [4] => Four
    [5] => Five
)
```

Array after inserting a new element at 10.

```
Array
(
    [1] => One
    [2] => Two
    [3] => Three
    [4] => Four
    [5] => Five
    [10] => Ten
)
```

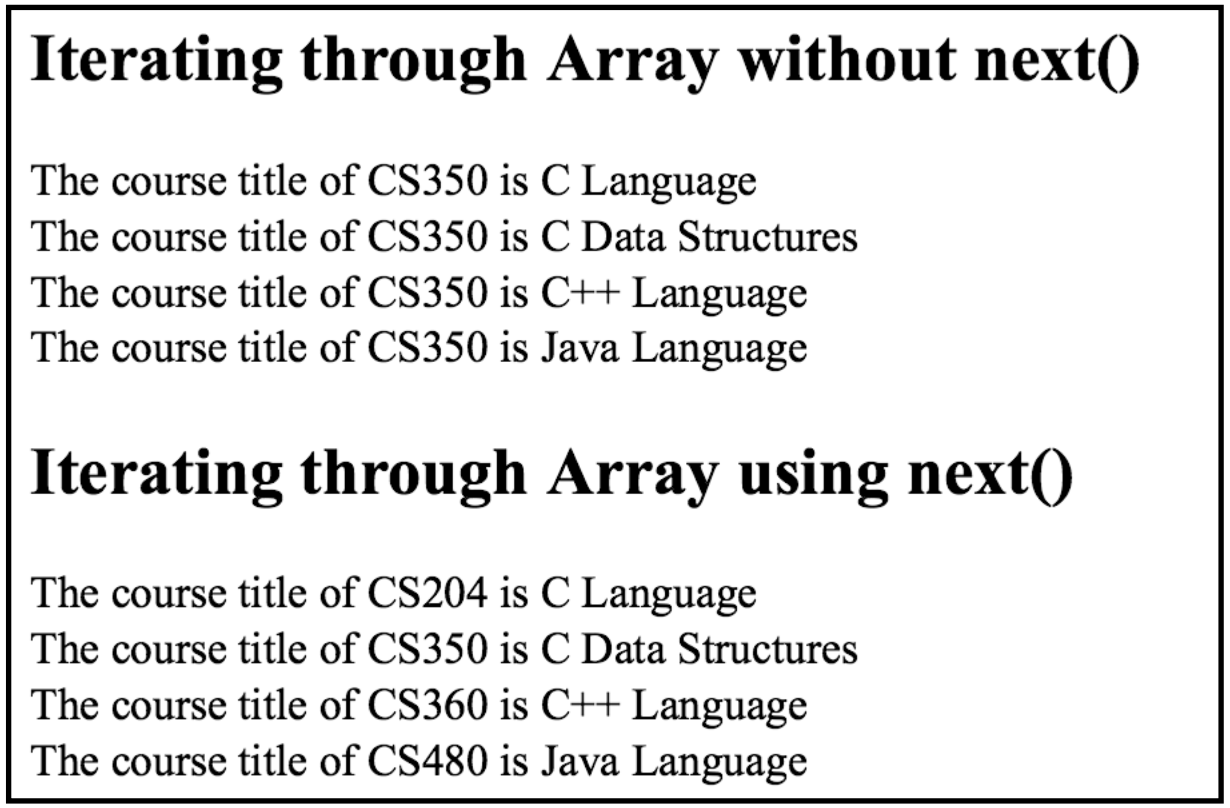
Iterating an Array

- A foreach statement allows you to loop through the elements of an array, but it does not change the position of the internal array pointer.
- The following functions that your code to iterate through an array with the internal array pointer.

Function	Description
current(array)	Returns the current array element
each(array)	Returns the key and value of the current array element and moves the internal array pointer to the next element
end(array)	Moves the internal array pointer to the last element
key(array)	Returns the key of the current array element
next(array)	Moves the internal array pointer to the next element
prev(array)	Moves the internal array pointer to the previous element
reset(array)	Resets the internal array pointer to the first element

The iterate_arrays.php

```
<?php
$courses = array(
   "CS204" => "C Language",
   "CS350" => "C Data Structures",
   "CS360" => "C++ Language",
   "CS480" => "Java Language"
);

echo "<h2>Iterating through Array without next()</h2>\n";
foreach ($courses as $course) {
    echo "The course title of " . key($courses) . " is $course<br />\n";
}
echo "<h2>Iterating through Array using next()</h2>\n";
reset($courses);
$key = key($courses);
foreach ($courses as $course) {
    echo "The course title of " . $key . " is $course<br />\n";
    $key = key($courses);
    next($courses);
}
```



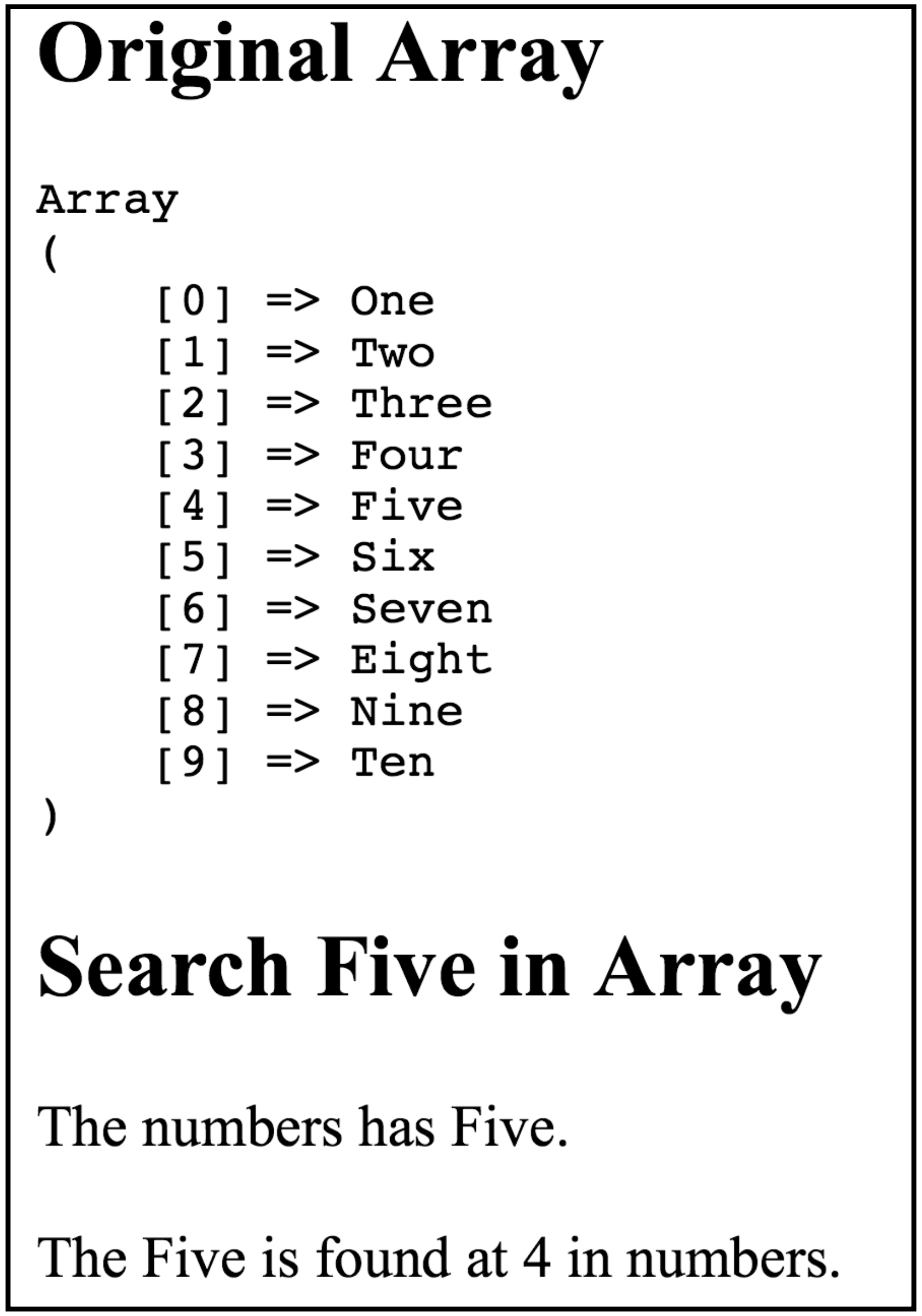
```
Iterating through Array without next()

The course title of CS350 is C Language
The course title of CS350 is C Data Structures
The course title of CS350 is C++ Language
The course title of CS350 is Java Language

Iterating through Array using next()

The course title of CS204 is C Language
The course title of CS350 is C Data Structures
The course title of CS360 is C++ Language
The course title of CS480 is Java Language
```

Finding Array Elements

- The in_array() function returns a Boolean value of TRUE if a given value exists in an array.
- The array_search() function determines whether a given value exists in an array, then returns the index or key of the first matching element if it exists or FALSE otherwise.
  - The strict not equal operator (!==) is necessary to compare the search result because PHP equates a Boolean value of FALSE with 0, which is also the value that identifies the first element in an indexed array.

The array_search.php

```
<?php

$numbers = array(
    "One",
    "Two",
    "Three",
    "Four",
    "Five",
    "Six",
    "Seven",
    "Eight",
    "Nine",
    "Ten");
echo "<h2>Original Array</h2>\n";
echo "<pre>\n";
print_r($numbers);
echo "</pre>\n";
echo "<h2>Search Five in Array</h2>\n";
if (in_array("Five", $numbers))
    echo "<p>The numbers has Five.</p>";

$found = array_search("Five", $numbers);
if ($found !== FALSE)
     echo "<p>The Five is found at " . $found . " in numbers.</p>\n";
else
    echo "<p>The Five is not found at " . $found . " in numbers.</p>\n";
```



Original Array

```
Array
(
    [0] => One
    [1] => Two
    [2] => Three
    [3] => Four
    [4] => Five
    [5] => Six
    [6] => Seven
    [7] => Eight
    [8] => Nine
    [9] => Ten
)
```

Search Five in Array

The numbers has Five.

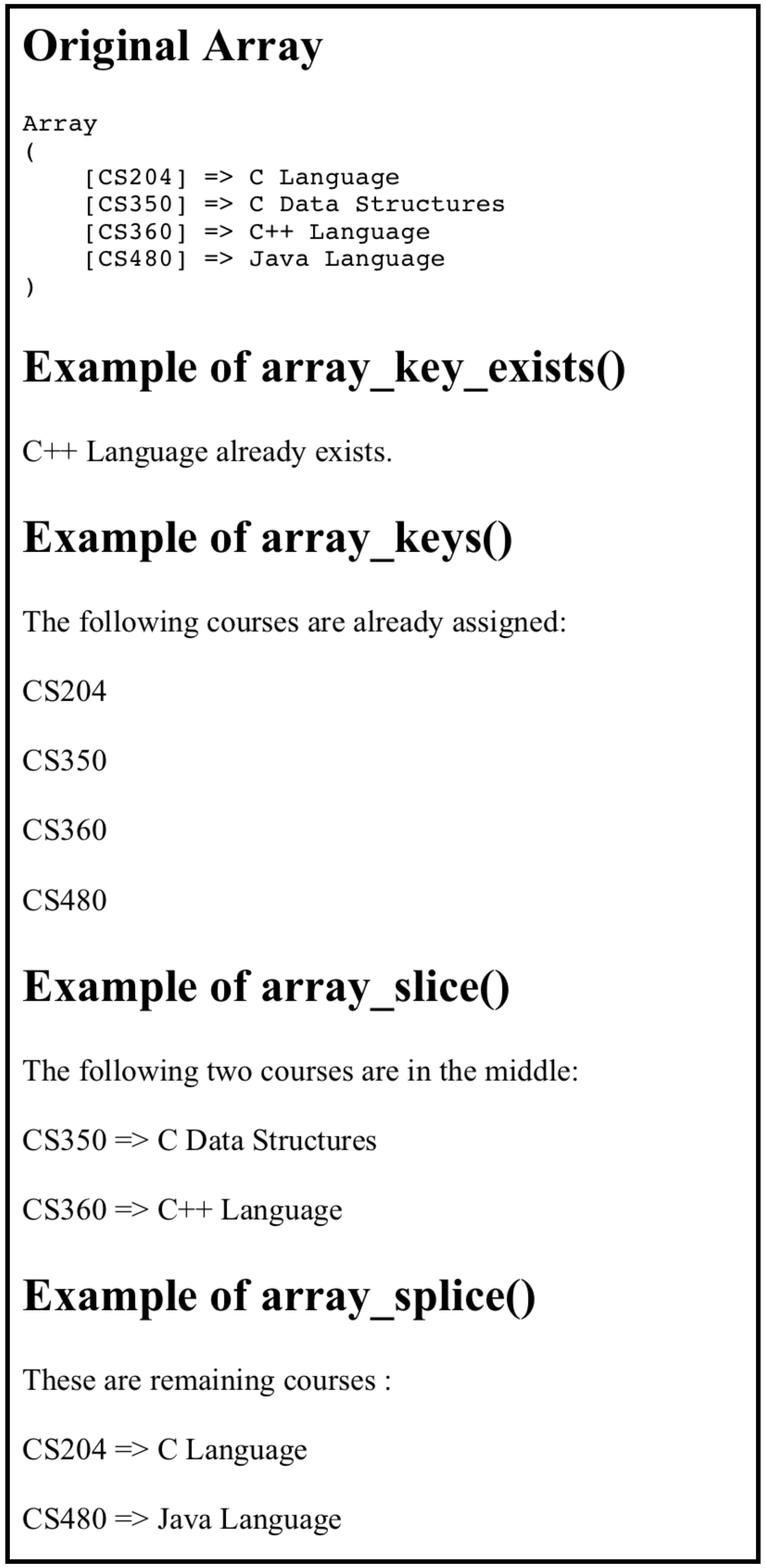
The Five is found at 4 in numbers.

Other Array Functions for Manipulating Array Elements

- The **array_key_exists**() function that determines whether a given index or key exists.
- The **array_keys**() function that returns an indexed array that contains all the keys in an associative array.
- The **array_slice**() function that returns (copy) a portion of an array and assign it to another array.
  - array_slice(array_name,start_index, number_to_return)
- The **array_splice**() function that for deleting a portion of an array.

The other_array_functions.php

```php
<?php
$courses = array(
   "CS204" => "C Language",
   "CS350" => "C Data Structures",
   "CS360" => "C++ Language",
   "CS480" => "Java Language",
);

echo "<h2>Original Array</h2>\n";
echo "<pre>\n";
print_r($courses);
echo "</pre>\n";

echo "<h2>Example of array_key_exists()</h2>\n";
if (array_key_exists("CS360", $courses))
    echo "<p>{$courses['CS360']} already exists.</p>\n";
else {
    $courses['CS360'] = "C++ Language";
    echo "<p>{$courses['CS360']} is now available.</p>";
}

echo "<h2>Example of array_keys()</h2>\n";
$courseNos = array_keys($courses);
echo "<p>The following courses are already assigned:</p>\n";
for ($i = 0; $i < count($courseNos); $i++) {
    echo "<p>{$courseNos[$i]}</p>";
}

echo "<h2>Example of array_slice()</h2>\n";
$twoCourses = array_slice($courses, 1, 2);
echo "<p>The following two courses are in the middle:</p>\n";
foreach ($twoCourses as $no => $course) {
    echo "<p>$no => $course</p>";
}

echo "<h2>Example of array_splice()</h2>\n";
array_splice($courses, 1, 2);
echo "<p>These are remaining courses :</p>\n";
foreach ($courses as $no => $course) {
    echo "<p>$no => $course</p>";
}
```



Original Array

```
Array
(
    [CS204] => C Language
    [CS350] => C Data Structures
    [CS360] => C++ Language
    [CS480] => Java Language
)
```

Example of array_key_exists()

C++ Language already exists.

Example of array_keys()

The following courses are already assigned:

CS204

CS350

CS360

CS480

Example of array_slice()

The following two courses are in the middle:

CS350 => C Data Structures

CS360 => C++ Language

Example of array_splice()

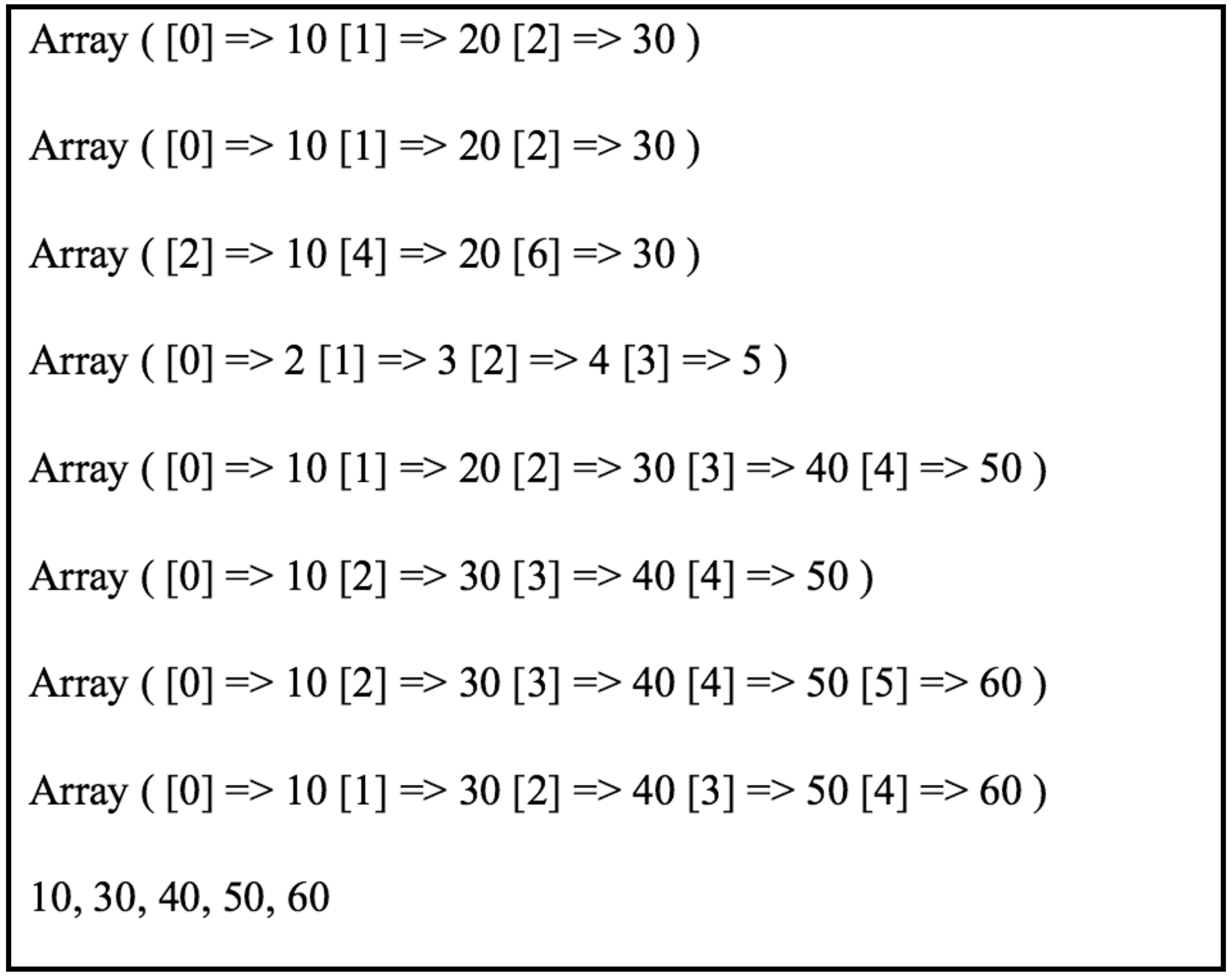
These are remaining courses :

CS204 => C Language

CS480 => Java Language

Other Array Examples

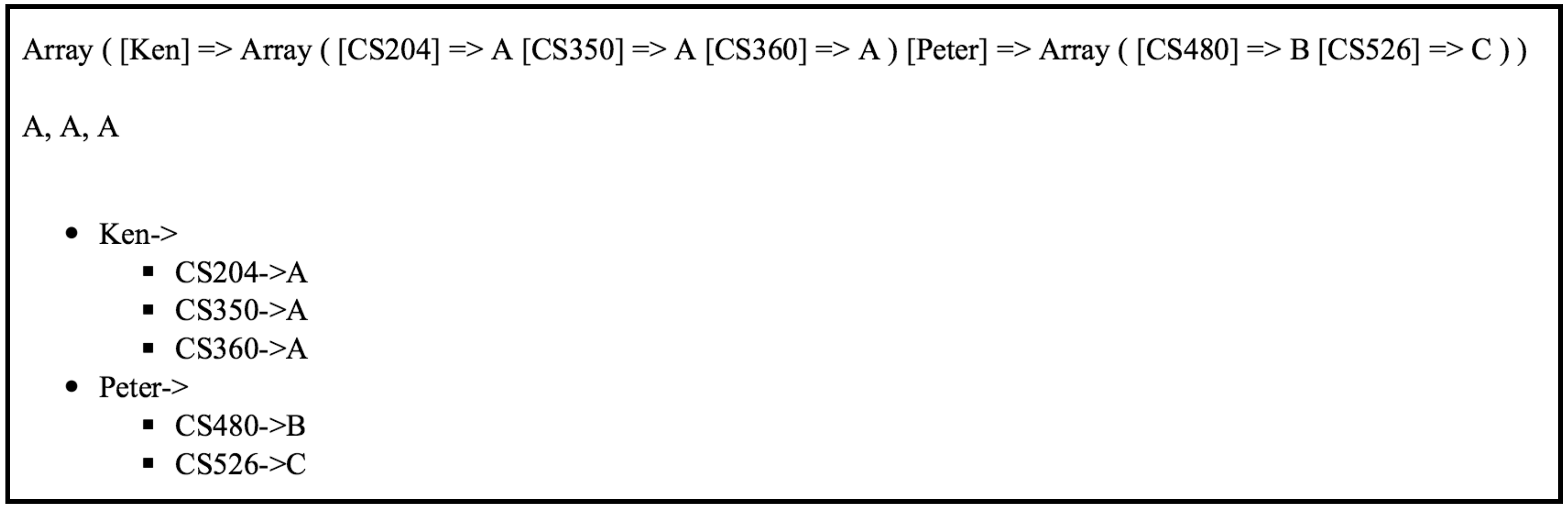
```php
<?php
 echo "<h1></h1>";
 $college = array(0 => 10, 1 => 20, 2 => 30);
 print_r ($college);
 echo '<br><br>';
 $college = array( 10, 20, 30);
 print_r ($college);
 echo '<br><br>';
 $college = array( 2=>10, 4=>20, 6=>30);
 print_r ($college);
 echo '<br><br>';
 $college = range(2, 5);
 print_r ($college);


 $college = array(10, 20, 30);
 $college[] = 40;
 $college[] = 50;
 echo '<br><br>';
 print_r ($college);
 unset($college[1]);
 echo '<br><br>';
 print_r ($college);
 $college[] = 60;
 echo '<br><br>';
 print_r ($college);

 $college2 = array_values($college);
 echo '<br><br>';
 print_r ($college2);
 echo '<br><br>';
 echo implode(', ', $college2);
```



```
Array ( [0] => 10 [1] => 20 [2] => 30 )

Array ( [0] => 10 [1] => 20 [2] => 30 )

Array ( [2] => 10 [4] => 20 [6] => 30 )

Array ( [0] => 2 [1] => 3 [2] => 4 [3] => 5 )

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

Array ( [0] => 10 [2] => 30 [3] => 40 [4] => 50 )

Array ( [0] => 10 [2] => 30 [3] => 40 [4] => 50 [5] => 60 )

Array ( [0] => 10 [1] => 30 [2] => 40 [3] => 50 [4] => 60 )

10, 30, 40, 50, 60
```

Other Array Examples

```php
<?php
  echo '<br><br>';
  $college = array(
       'Ken' => array('CS204' => 'A', 'CS350' => 'A', 'CS360' => 'A'),
       'Peter' => array('CS480'=>'B', 'CS526'=>'C')
    );
  print_r ($college);

  echo '<br><br>';
  echo implode(', ', $college['Ken']);

  echo '<br><br>';
  echo '<ul>';
     foreach ($college as $key => $value )
     {
       echo '<li>'.$key.'->';
         echo '<ul>';
         foreach ($value as $course => $grade )
         {
           echo '<li type="square">'.$course.'->'.$grade.'</li>';
         }
         echo '</li>';
         echo '</ul>';
     }
  echo '</ul>';
```



```
Array ( [Ken] => Array ( [CS204] => A [CS350] => A [CS360] => A ) [Peter] => Array ( [CS480] => B [CS526] => C ) )

A, A, A


  • Ken->
        ▪ CS204->A
        ▪ CS350->A
        ▪ CS360->A
  • Peter->
        ▪ CS480->B
        ▪ CS526->C
```