PHP - Array Destructuring

In PHP, the term Array destructuring refers to the mechanism of extracting the array elements into individual variables. It can also be called unpacking of array. PHP's list() construct is used to destrucrure the given array assign its items to a list of variables in one statement.

```
list($var1, $var2, $var3, . . . ) = array(val1, val2, val3, . . .);
```

As a result, **val1** is assigned to **$var1, val2** to **$var2** and so on. Even though because of the parentheses, you may think list() is a function, but it's not as it doesn't have return value. PHP treats a string as an array, however it cannot be unpacked with list(). Moreover, the parenthesis in list() cannot be empty.

Instead of list(), you can also use the square brackets [] as a shortcut for destructuring the array.

```
[$var1, $var2, $var3, . . . ] = array(val1, val2, val3, . . .);
```

Example

Take a look at the following example –

</> Open Compiler

```
<?php
   $marks = array(50, 56, 70);
   list($p, $c, $m) = $marks;
   echo "Physics: $p  Chemistry: $c  Maths: $m" . PHP_EOL;

   # shortcut notation
   [$p, $c, $m] = $marks;
   echo "Physics: $p  Chemistry: $c  Maths: $m" . PHP_EOL;
?>
```

It will produce the following **output** –

```
Physics: 50  Chemistry: 56  Maths: 70
```

```
Physics: 50  Chemistry: 56  Maths: 70
```

Destructuring an Associative Array

Before PHP 7.1.0, list() only worked on numerical arrays with numerical indices start at 0. PHP 7.1, array destructuring works with associative arrays as well.

Let us try to destructure (or unpack) the following associative array, an array with non-numeric indices.

```
$marks = array('p'=>50, 'c'=>56, 'm'=>70);
```

To destructure this array the list() statement associates each array key with a independent variable.

```
list('p'=>$p, 'c'=>$c, 'm'=>$m) = $marks;
```

Instead, you can also use the [] alternative destructuring notation.

```
['p'=>$p, 'c'=>$c, 'm'=>$m] = $marks;
```

Try and execute the following PHP script –

</> Open Compiler

```
<?php
   $marks = array('p'=>50, 'c'=>56, 'm'=>70);
   list('p'=>$p, 'c'=>$c, 'm'=>$m) = $marks;
   echo "Physics: $p  Chemistry: $c  Maths: $m" . PHP_EOL;

   # shortcut notation
   ['p'=>$p, 'c'=>$c, 'm'=>$m] = $marks;
   echo "Physics: $p  Chemistry: $c  Maths: $m" . PHP_EOL;
?>
```


Skipping Array Elements

In case of an indexed array, you can skip some of its elements in assign only others to the required variables

```php
<?php
   $marks = array(50, 56, 70);
   list($p, , $m) = $marks;
   echo "Physics: $p  Maths: $m" . PHP_EOL;

   # shortcut notation
   [$p, , $m] = $marks;
   echo "Physics: $p  Maths: $m" . PHP_EOL;
?>
```

In case of an associative array, since the indices are not incremental starting from 0, it is not necessary to follow the order of elements while assigning.

```php
<?php
   $marks = array('p'=>50, 'c'=>56, 'm'=>70);
   list('c'=>$c, 'p'=>$p, 'm'=>$m) = $marks;
   echo "Physics: $p  Chemistry: $c  Maths: $m" . PHP_EOL;

   ['c'=>$c, 'm'=>$m, 'p'=>$p] = $marks;        # shortcut notation
   echo "Physics: $p  Chemistry: $c  Maths: $m" . PHP_EOL;
?>
```

Destructuring a Nested Array

You can extend the concept of array destructuring to nested arrays as well. In the following example, the subarray nested inside is an indexed array.

```php
<?php
   $marks = ['marks' => [50, 60, 70]];
   ['marks' => [$p, $c, $m]] = $marks;
```

```
   echo "Physics: $p  Chemistry: $c  Maths: $m" . PHP_EOL;
?>
```

Destructuring works well even if the nested array is also an associative array.

</> Open Compiler

```
<?php
   $marks = ['marks' => ['p'=>50, 'c'=>60, 'm'=>70]];
   ['marks' => ['p'=>$p, 'c'=>$c, 'm'=>$m]] = $marks;
   echo "Physics: $p  Chemistry: $c  Maths: $m" . PHP_EOL;
?>
```