Arrays (array)

What is an array?

In the previous class, you learned what a variable is.
The problem is that a variable can only contain one value.
Only one integer (or single string) can be stored in a variable.

An array, in PHP, is a special variable that can contain multiple integers or multiple strings at a time.
You can compare a painting to a box of chocolates. The array is the box of chocolates and the locations containing the chocolates represent the values stored in the array.

Numeric arrays

The syntax for declaring a numeric array is as follows:

```
<? Php
   $ var_name [n] = value;
?>
```

or

```
<? php
   $ var_name = array (n => value, ...);
?>
```

"$ Var_name" is the name of the array (variable).
"[N]" is the position (index) of the element.
"Value" is the value assigned to the array element.

You can also declare an empty array: **$ var_name = array ();**

Let's take our list of films. By storing the movie list with conventional variables, we would have something similar:

```
$ movie1 = 'Pirates of the Caribbean';
$ movie2 = 'Star Wars';
$ movie3 = 'Fight Club';
$ movie4 = 'Untouchables';
```

We have to create a different variable for each movie.

To declare this movie list in a array, we do like this:

```
<? Php
   $ movies [0] = 'Pirates of the Caribbean';
   $ movies [1] = 'Star Wars';
   $ movies [2] = 'Fight Club';
   $ movies [3] = 'Untouchables';
?>
```

or (other syntax):

```
<? php
   $ movies = array (0 => "Caribbean Pirates",
            1 => "Star Wars",
            2 => "Fight Club",
            3 => "Untouchables");
?>
```

You now know how to store values in an array.
But how can you access these values?

To get inside a array, you just have to tell which index / position you want to access: **$ movies [n]**

```
        echo $ movies [1]; // Result in position 1:
                        Star Wars

echo $ movies [3]; // Result in position 3:
                    Untouchable
```

Associative Arrays

The array differs from the numerical array in that the indexes of the arrays are not numbers but text.

This can help you remember what's in a key or what it's supposed to do.

The syntax for declaring an associative array is:

```
<? Php
   $ var_name ['key_name'] = value;
?>

<? php
   $ var_name = array ('keyname' => value);
?>
```

To declare this movie list in an associative array:

```
<? Php
   $ movies = array (
         "Caribbean Pirates" => "Action",
         "Star Wars" => "Science-Fiction",
         "Fight Club" => "Drama",
         "Untouchables" => "Comedy Dramatic"
  );
?>
```

To access the values in an associative array, call the desired key name.

For example:

```
echo "Star Wars is type". $ movies ["Star Wars"];
// Result: Star Wars is a Science-Fiction type
```

Multi-dimensional

arrays Multi-dimensional arrays are arrays containing other arrays (nested arrays).

The advantage of multidimensional arrays is that they allow us to group associated data.

Let's take our example of movies:

```
<? Php
   $ movies = array (
     "Comedy Drama" => array ("Untouchables"),
     "Action" => array ("Caribbean Pirates",
                        "Expendables"),
     "Drama" => array ("Fight Club", "Night Call")
   );
?>
```

This example multi-dimensional array is actually an associative array that contains multiple numeric arrays.

To access a value:

```
echo $ movies ["Action"] [1];
        // Result: Expendables
```

Sort (sorting)

In PHP, it is possible to sort the values of an array.
For that we use: **sort ($ movies);**

Example with our movie list:

```
<? Php
   $ movies [0] = 'Caribbean Pirates';
   $ movies [1] = 'Star Wars';
   $ movies [2] = 'Fight Club';
   $ movies [3] = 'Untouchables';

   echo $ movies [1]; // Back: Star Wars;

   sort ($ movies); // We sort our array

   echo $ movies [1]; // Back: Untouchable;
?>
```

For associative arrays, use the functions:
asort ($ movies); and **ksort ($ movies).**

The "a" indicates to PHP that you want to sort by the value and not by the key.
The "k" indicates the opposite. That is, sort by the key ('k' for 'key').

Count

The count () function is useful when you want to know the number of elements in your array.

With the same example as before:

```
<? Php
   $ movies [0] = 'Pirates of the Caribbean';
   $ movies [1] = 'Star Wars';
   $ movies [2] = 'Fight Club';
   $ movies [3] = 'Untouchables';

   echo count ($ movies); // return: 4;
?>
```

Why use an array ?

The content of the arrays can be stretched.
arrays help to easily group related information.
arrays help to write clearer and better code.