PHP - Default Arguments

Like most of the languages that support imperative programming, a function in PHP may have one or more arguments that have a default value. As a result, such a function may be called without passing any value to it. If there is no value meant to be passed, the function will take its default value for processing. If the function call does provide a value, the default value will be overridden.

```
function fun($arg1 = val1, $arg2 = val2) {
   Statements;
}
```

Such a function can be called in different ways –

```
fun();          # Function will use defaults for both arguments
fun($x);        # Function passes $x to arg1 and uses default for arg2
fun($x, $y);    # Both arguments use the values passed
```

Example 1

Here we define a function called **greeting()** with two arguments, both having **string** as their default values. We call it by passing one string, two strings and without any argument.

</> Open Compiler

```
<?php
   function  greeting($arg1="Hello", $arg2="world") {
      echo $arg1 . " ". $arg2 . PHP_EOL;
   }

   greeting();
   greeting("Thank you");
   greeting("Welcome", "back");
   greeting("PHP");
?>
```

It will produce the following **output** –

```
Hello world
Thank you world
Welcome back
PHP world
```

Example 2

You can define a function with only some of the arguments with default value, and the others to which the value must be passed.

```php
<?php
   function  greeting($arg1, $arg2="World") {
      echo $arg1 . " ". $arg2 . PHP_EOL;
   }

   # greeting(); ## This will raise ArgumentCountError
   greeting("Thank you");
   greeting("Welcome", "back");
?>
```

It will produce the following **output** –

```
Thank you World
Welcome back
```

The first call (without argument) raises **ArgumentCountError** because you must pass value for the first argument. If only one value is passed, it will be used by the first argument in the list.

However, if you declare arguments with **default** before arguments without defaults, such function can be only called if values for both are passed. You cannot have a situation where the first argument uses the default, and the second using the passed value.

The **greeting()** function now has **$arg1** with default and **$arg2** without any default value.

```php
function  greeting($arg1="Hello", $arg2) {
   echo $arg1 . " ". $arg2 . PHP_EOL;
}
```

If you pass a string "PHP" −

```
greeting("PHP");
```

with the intension to print the result as "Hello PHP", the following error message will be displayed.

```
PHP Fatal error: Uncaught ArgumentCountError: Too few arguments to function
greeting(), 1 passed in hello.php on line 10 and exactly 2 expected
```


Example 3

Let's define a function **percent()** that calculates the percentage of marks in three subjects.

Assuming that the marks in each subject are out of 100, the **$total** argument in the function definition is given a default value as 300.

</> Open Compiler

```
<?php
   function  percent($p, $c, $m, $ttl=300) {
      $per = ($p+$c+$m)*100/$ttl;
      echo "Marks obtained: \n";
      echo "Physics = $p Chemistry = $c Maths = $m \n";
      echo "Percentage = $per \n";
   }
   percent(50, 60, 70);
?>
```

It will produce the following **output** −

```
Marks obtained:
Physics = 50 Chemistry = 60 Maths = 70
Percentage = 60
```

However, if the maximum marks in each subject is 50, then you must pass the fourth value to the function, otherwise the percentage will be calculated out of 300 instead of 150.

```php
<?php
   function  percent($p, $c, $m, $ttl=300) {
      $per = ($p+$c+$m)*100/$ttl;
      echo "Marks obtained: \n";
      echo "Physics = $p Chemistry = $c Maths = $m \n";
      echo "Percentage = $per \n";
   }
   percent(30, 35, 40, 150);
?>
```

Open Compiler

It will produce the following **output** –

```
Marks obtained:
Physics = 30 Chemistry = 35 Maths = 40
Percentage = 70
```