PHP - Returning Values

A PHP function can have an optional return statement as the last statement in its function body. Most of the built-in functions in PHP return a certain value. For example the strlen() function returns the length of a string. Similarly, a user defined function can also return a certain value.

A function is an independent, complete and reusable block of statements. When called, it performs a certain task and sends the program control back to position from where it was called even if return statement is not used. The return statement allows it to take a value, along with the control, back to the calling environment.

```
function foo($arg_1, $arg_2) {
   statements;
   return $retval;
}
```

A function may return any type of data, including scalar variables, arrays and objects. A return keyword without any expression in front of it returns null, and is equivalent to a function not having a return value at all.

The value returned by a function can be stored in a variable, can be put in an expression, or, if appearing inside print or echo, is displayed in the output.

```
$res = foo($x, $y);
```

It allows the returned value of a function to be used further in the program.

Example

Let us modify the addition() function in the previous chapter to include a **return** statement to return the result of addition.

</> Open Compiler

```
<?php
   function addition($first, $second) {
      $result = $first+$second;
      return $result;
   }
```

```
   $x=10;
   $y=20;
   $z = addition($x, $y);
   echo "First number: $x Second number: $y Addition: $z". PHP_EOL;
?>
```

It will produce the following **output** –

```
First number: 10 Second number: 20 Addition: 30
```

A function in PHP may have any number of arguments, but can return only one value. The function goes back to the calling environment as soon as it comes across a **return** statement for the first time, abandoning the rest of statements in the function body.

Example

If you try to include more than one values in the **return** statement, a PHP parse error is encountered as below –

</> Open Compiler

```
<?php
   function raiseto($x) {
      $sqr =  $x**2;
      $cub =  $x**3;
      return $sqr, $cub;
   }
   $a = 5;
   $val = raiseto($a);
?>
```

It will produce the following **output** –

```
PHP Parse error: syntax error, unexpected token ",", expecting ";"
```


Conditional Return

You can have multiple **return** statements executed under different conditional statements.

Example

In the following program, the **raiseto()** function returns either the square or the cube of a number of the index argument, that is either 2 or 3, respectively.

</> Open Compiler

```
<?php
   function raiseto($x, $i) {
      if ($i == 2) {
         return $x**2;
      } elseif ($i==3) {
         return $x**3;
      }
   }
   $a = 5;
   $b = 2;
   $val = raiseto($a, $b);
   echo "$a raised to $b = $val" . PHP_EOL;

   $x = 7;
   $y = 3;
   echo "$x raised to $y = " . raiseto($x, $y) . PHP_EOL;
?>
```

It will produce the following **output** –

```
5 raised to 2 = 25
7 raised to 3 = 343
```

Return Multiple Values as Array

The function in PHP is capable of returning only a single value. However, that single value can be an array of more than one values. We can take the advantage of this feature to return the square as well as the cube of a number at once.

Example

Take a look at the following example –

</> Open Compiler

```php
<?php
   function raiseto($x){
      $sqr =  $x**2;
      $cub =  $x**3;
      $ret = ["sqr" => $sqr, "cub" => $cub];
      return $ret;
   }
   $a = 5;
   $val = raiseto($a);
   echo "Square of $a: " . $val["sqr"] . PHP_EOL;
   echo "Cube of $a: " . $val["cub"] . PHP_EOL;
?>
```

It will produce the following **output** –

```
Square of 5: 25
Cube of 5: 125
```