# Chapter 13

# How to create and use functions


© 2010, Mike Murach & Associates, Inc.

# The syntax for a function definition

```
function function_name([$param_1,$param_2, ... ,
                        $param_n])
{
    // Code for function
    [return [value];]
}
```

# A function with no parameters that returns a value

```
function coin_toss() {
    $coin = (mt_rand(0, 1) == 0) ? 'heads' : 'tails';
    return $coin;
}
```


© 2010, Mike Murach & Associates, Inc.

## A function with one parameter

```
function display_error($error) {
    echo '<p class="error">' . $error . '</p>';
}
```

## A function with three parameters

```
function avg_of_3($x, $y, $z) {
    $avg = ($x + $y + $z) / 3;
    return $avg;
}
```

## Calling functions that return values

```
$average = avg_of_3(5, 2, 8);  // $average is 5
echo coin_toss();              // Displays heads or tails
```


© 2010, Mike Murach & Associates, Inc.

## A function call that doesn't return a value

```
display_error('Value out of range.');
```

## A function call that discards the return value

```
$list = array('Apples', 'Oranges', 'Grapes');
$last = array_pop($list);
array_pop($list);       // Removes Oranges
                        // Discards return value
```


© 2010, Mike Murach & Associates, Inc.

# In PHP arguments can be passed by value or by reference


© 2010, Mike Murach & Associates, Inc.

# An argument passed by value

```
function add_3_by_val($value) {
    $value += 3;
    echo '<p>Number: ' . $value . '</p>';
}

$number = 5;
add_3_by_val($number);                          // Displays 8
echo '<p>Number: ' . $number . '</p>';          // Displays 5
```

# An argument passed by reference

```
function add_3_by_ref(&$value) {
    $value += 3;
    echo '<p>Number: ' . $value . '</p>';
}

$number = 5;
add_3_by_ref($number);                          // Displays 8
echo '<p>Number: ' . $number . '</p>';          // Displays 8
```


© 2010, Mike Murach & Associates, Inc.

# How to modify a string that's passed by reference

```
function wrap_in_tag(&$text, $tag) {
    $before = '<'  . $tag . '>';
    $after  = '</' . $tag . '>';
    $text = $before . $text . $after;
}

$message = 'Value out of range.';
wrap_in_tag($message, 'p');
echo $message;              // <p>Value out of range.</p>
```


© 2010, Mike Murach & Associates, Inc.

# How to return multiple values

```
function array_analyze($array, &$sum, &$prod, &$avg) {
    $sum = array_sum($array);
    $prod = array_product($array);
    $avg = $sum / count($array);
}

$list = array(1, 4, 9, 16);
array_analyze($list, $s, $p, $a);
echo '<p>Sum: ' . $s . '<br />
         Product: ' . $p . '<br />
         Average ' . $a .
     '</p>';
```


© 2010, Mike Murach & Associates, Inc.

## A variable with global scope

```
$a = 10;                // $a has global scope
function show_a() {
    echo $a;            // Inside function, $a is NULL
}
show_a();               // Displays nothing
```

## How to access a global variable from a function

```
$b = 10;                    // $b has global scope
function show_b() {
    global $b;              // $b refers global variable $b
    echo $b;
}
show_b();                   // Displays 10
```


© 2010, Mike Murach & Associates, Inc.

# Another way to access a global variable

```
$c = 10;                        // $c has global scope
function show_c() {
    $c = $GLOBALS['c'];         // $c refers to global $c
    echo $c;
}
show_c();                       // Displays 10
```

# A variable with local scope

```
function show_d() {
    $d = 10;          // $d has local scope within function
    echo $d;
}
echo $d;              // Outside function, $d is NULL
```


© 2010, Mike Murach & Associates, Inc.

# A function with one default parameter

```
function get_rand_bool_text($type = 'coin') {
    $rand = mt_rand(0, 1);
    switch ($type) {
        case 'coin':
            $result = ($rand == 1) ? 'heads' : 'tails';
            break;
        case 'switch':
            $result = ($rand == 1) ? 'on' : 'off';
            break;
    }
    return $result;
}
```


© 2010, Mike Murach & Associates, Inc.

# A function with an optional parameter

```
function is_leap_year($date = NULL) {
    if (!isset($date)) {
        $date = new DateTime();
    }
    if ($date->format('L') == '1') return true;
    else return false;
}
```


© 2010, Mike Murach & Associates, Inc.

# A function with one required and two default parameters

```
function display_error($error,
                       $tag = 'p',
                       $class = 'error') {
    $opentag  = '<'  . $tag . ' class="' . $class . '">';
    $closetag = '</' . $tag . '>';
    echo $opentag . $error . $closetag;
}
```


© 2010, Mike Murach & Associates, Inc.

# Calling a function with a default parameter value

## Omitting optional parameters

```
echo get_rand_bool_text();
echo display_error('Out of range');
$is_leap_year = is_leap_year();
```

## Providing optional parameters

```
echo get_rand_bool_text('switch');
echo display_error('Out of range', 'li');
$is_leap_year =
    is_leap_year(new DateTime('March 15, 2015'));
```


© 2010, Mike Murach & Associates, Inc.

# Functions for working with variable-length parameter lists

```
func_get_args()
func_num_args()
func_get_arg($i)
```

# A function that adds a list of numbers

```
function add() {
    $numbers = func_get_args();
    $total = 0;
    foreach($numbers as $number) {
        $total += $number;
    }
    return $total;
}

$sum = add(5, 10, 15);          // $sum is 30
```


© 2010, Mike Murach & Associates, Inc.

# A function that averages one or more numbers

```
function average($x) {   // $x forces one argument
    $count = func_num_args();
    $total = 0;
    for($i = 0; $i < $count; $i++) {
        $total += func_get_arg($i);
    }
    return $total / $count;
}

$avg = average(75, 95, 100);   // $avg is 90
```


© 2010, Mike Murach & Associates, Inc.

# Using required parameters with a variable parameter list

```
function array_append(&$array, $x) {
    $values = func_get_args();  // Also contains $array
    array_shift($values);    // Removes $array from front
    foreach($values as $value) {
        $array[] = $value;
    }
}

$data = array('apples', 'oranges');
array_append($data, 'grapes', 'pears');
```


© 2010, Mike Murach & Associates, Inc.