Functions

H S Rana

CIT,UPES

February 17, 2014

Advantages

1. It reduces duplication within a program.
2. A function is created once but used many times, often from more than one program.
3. Debugging and testing a program becomes easier when the program is subdivided into functions, as it becomes easier to trace the source of an error and correct it with minimal impact on the rest of the program.
4. Functions encourage abstract thinking.

Creating and Invoking Functions

1. Arguments, which serve as inputs to the function
2. Return values, which are the outputs returned by the function
3. The function body, which contains the processing code to turn inputs into outputs

Creating and Invoking Functions

1. Arguments, which serve as inputs to the function
2. Return values, which are the outputs returned by the function
3. The function body, which contains the processing code to turn inputs into outputs

```
<?php
// function definition
// print today's weekday name
function whatIsToday() {
  echo "Today is Monday" ;
}

// function invocation
whatIsToday();
?>
```

Using Arguments and Return Values

defines a function accepting two arguments and uses these arguments to perform a geometric calculation

Using Arguments and Return Values

defines a function accepting two arguments and uses these arguments to perform a geometric calculation

```
<?php
// function definition
// calculate perimeter of rectangle
// p = 2 * (l+w)
function getPerimeter($length, $width) {
  $perimeter = 2 * ($length + $width);
  echo "The perimeter of a rectangle of length $length
  units and width $width units is: $perimeter units";
}
// function invocation
// with arguments
getPerimeter(4,2);
?>
```

Returning Values

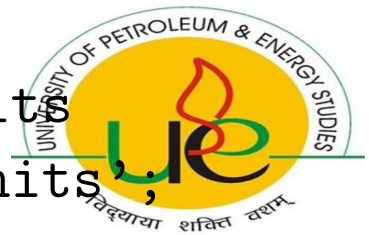
in PHP, you can have a function explicitly return a value

Returning Values

in PHP, you can have a function explicitly return a value

```
<?php
// function definition
// calculate perimeter of rectangle
// p = 2 * (l+w)
function getPerimeter($length, $width) {
  $perimeter = 2 * ($length + $width);
  return $perimeter;
}

// function invocation
// with arguments
echo 'The perimeter of a rectangle of length 4 units
and width 2 units is: ' . getPerimeter(4,2) . ' units';
?>
```



Ruturning Multiple Values

You can return multiple values from a function, by placing them all in an array and returning the array.

```
<?php
// break a string into words
// reverse and return as an array
function reverseMe($sentence) {
    $words = explode(' ', $sentence);
    foreach ($words as $k => $v) {
      $words[$k] = strrev($v);
    }
    return $words;
}
// output: 'evaH a doog yad'
echo implode(' ', reverseMe('Have a good day'));
// output: 'lliW uoy yrram em'
echo implode(' ', reverseMe('Will you marry me'));
?>
```

Setting Default Argument Values

the arguments that a function expects to receive from the main program are specified in the function's argument list. For convenience, you can assign default values to any or all of these arguments; these default values are used in the event the function invocation is missing some arguments.

Setting Default Argument Values

the arguments that a function expects to receive from the main program are specified in the function's argument list. For convenience, you can assign default values to any or all of these arguments; these default values are used in the event the function invocation is missing some arguments.

```
<?php
// generate e-mail address from supplied values
function buildAddress($username, $domain = 'mydomain.info') {
    return $username . '@' . $domain;
}
// without optional argument
// output: 'My e-mail address is john@mydomain.info'
echo 'My e-mail address is ' . buildAddress('john');
// with optional argument
// output: 'My e-mail address is jane@cooldomain.net'
echo 'My e-mail address is ' . buildAddress('jane', 'cooldomain
?>
```

Note

If only some of your function's arguments have default values assigned to them, place these arguments at the end of the function's argument list. This lets PHP correctly .

Using Dynamic Argument Lists

PHP also lets you define functions with so-called variable-length argument lists, where the number of arguments can change with each invocation of the function

Using Dynamic Argument Lists

PHP also lets you define functions with so-called variable-length argument lists, where the number of arguments can change with each invocation of the function

```
<?php
function calcAverage() {
  $args = func_get_args();
  $count = func_num_args();
  $sum = array_sum($args);
  $avg = $sum / $count;
  return $avg;
}
// output: 6
echo calcAverage(3,6,9);
// output: 150
echo calcAverage(100,200,100,300,50,150,250,50);
?>
```