Overview

Function

- Functions provide an abstraction layer.
  - Hides implementation details
- Naming a function
  - Name may begin with a letter or underscore , not a number
  - Function names are **not** case sensitive
  - Give the function a name that reflects its purpose

Function (continued)

- Use the function keyword to create a function.

```
function MyFunction ()
{
  ...
}
```

- A function is called by name

```
MyFunction()
```

Feeding a Function

- Data may be passed into a function
  - Referred to as arguments
    - An argument is simply a variable to the function

```
function MyFunction ($myArg)
{
...
}
```

    - Multiple arguments may be listed in the parentheses , separated by commas

```
function MyFunction ($arg1,$arg2,$arg3)
{
 // do something with that args
}
```

Feeding a Function(continued)

- Call a function that requires arguments by name passing values in the parentheses.

```
MyFunction("One",2,"Three");
```

- Parameter vs. Argument: technically a parameter is a variable listed in the function definition while an argument is the actual value passed when the function is called. However, the terms are now generally used interchangeably

Return Data from a Function

- A function's purpose is to abstract details of the code that performs a specific task
  - When the task is complete we often require some data that is the result of the function's execution
- To return data from a function we use the keyword return

Return Data from a Function (continued)

- return may also be used at anytime within the function to terminate its execution returning control to the caller.

```
function MyFunction($arg1,$arg2,$arg2) {
// do something with the args
$result = arg1+$arg2+$arg3;
return $result;
  }
```

- The caller of the function must "catch" the return value in variable

```
    $returnFromFunction=MyFunction("one",2,"Three");
```

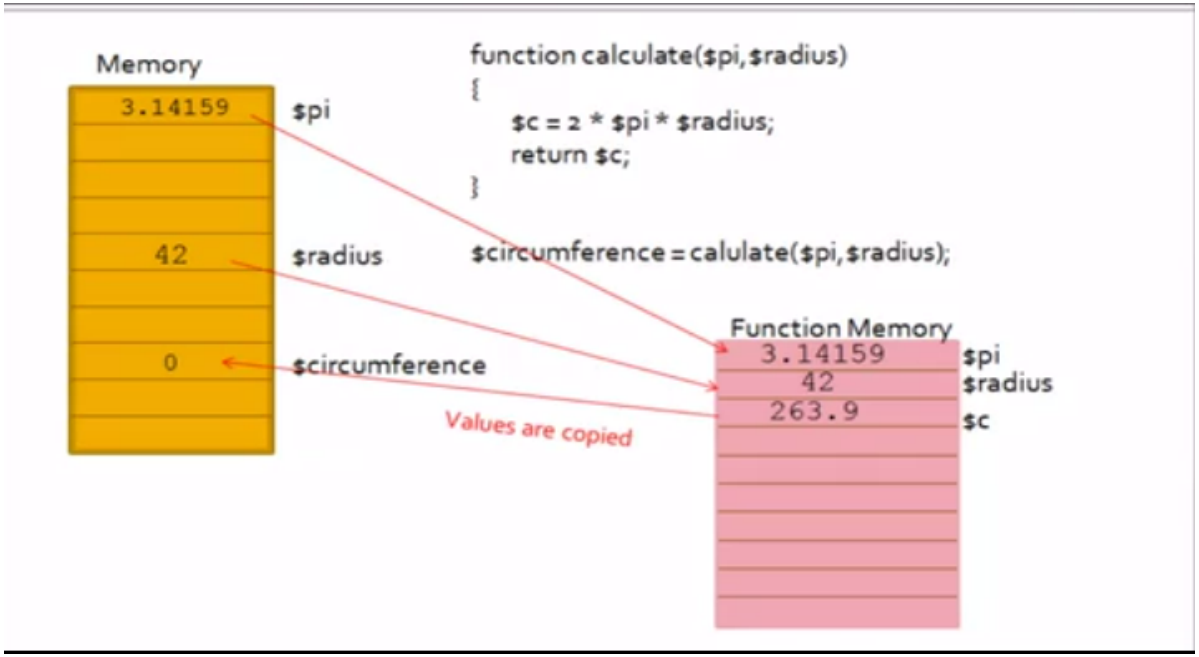
- Only one value may be returned from a function
- Can use an array to return multiple values.

Return Data from a Function (continued)

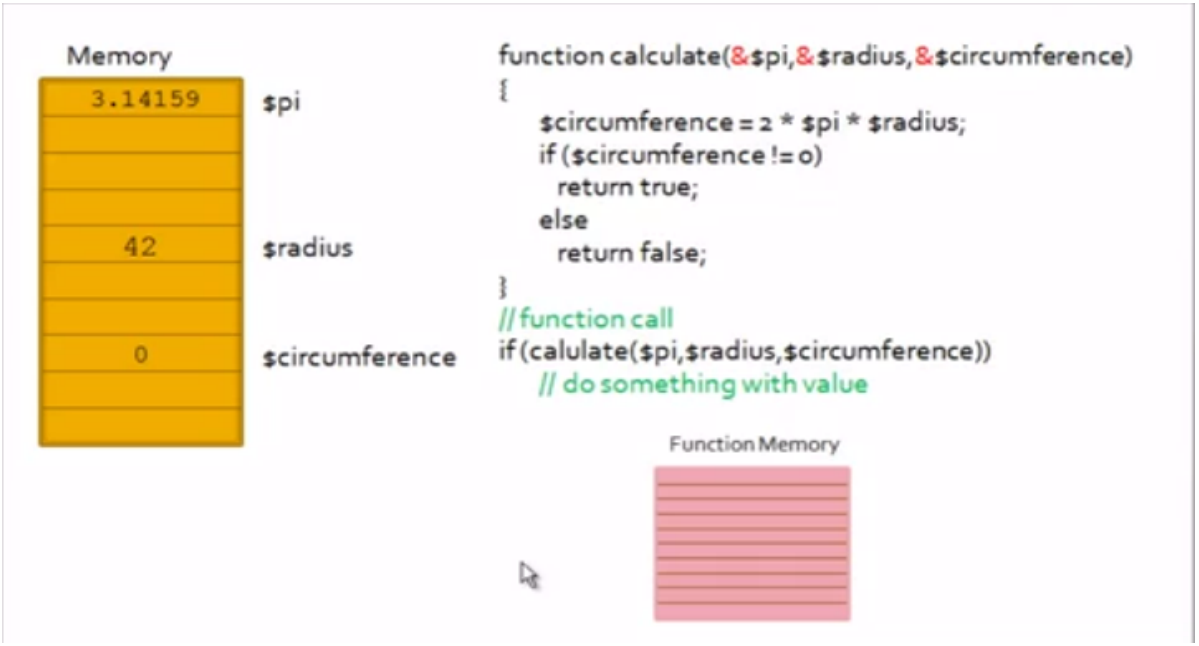
- Pass-by-reference allows a function argument to be changed within the function

```
function MyFunction (&$arg1)
```

Pass-By-Value



Pass-By-Reference



Functions Default Value

- Function parameters may be give "default" values.
  - The default values will be used if the caller does not provide an argument when calling the function

```
function MyFunction($arg1,$arg2,$arg3="A default value") {
 // do something with the args
 $result = arg1+$arg2+$arg3;
     return $result;
 }
```

Functions Default Value (continued)

- Function variables have local, independent scope within the function.
  - Values of the variables are NOT shared between function calls
- Use the kewword static to allow a function variable to be shared by all callers.

```
function MyFunction($arg1,$arg2,$arg3="A default value") {
    static $callCount;
// do something with the args
}
```

External PHP

- PHP allows us to manage the reuse of our functions or any other code using a set of keywords.
  - *include*" $<$ *filename* $>$ "
    - An automatic copy-paste command that makes the contents of any file available in the including file at runtime
  - *include_once*" $<$ *filename* $>$ "
    - Same as include except that it checks to ensure that the file has not already been included
  - *require*" $<$ *filename* $>$ "
    - Same as include except that if the file cannot be included a fatal error is generated
  - *require_once*" $<$ *filename* $>$ "
    - Same as *include_once* except that if the file cannot be included a fatal error is generated

Function

Demonstration

Date and Tme Manipulation

- Date and time manipulation is a foundational PHP skill
- Date and time values are represented as the number of seconds that have passed since January $1st197000 : 00 : 00GMT$.
  - This date format is known as "epoch time" or a "Unix timestamp"
- PHP date and time functions act upon the integer value representing the epoch time.
  - The advantage is that there is no specified format

Date and Time Manipulation

- The time() function returns an integer value.
  - *echotime*();
  - The value represents the number of seconds since epoch
- Use *strtotime*($<$ *datavalue* $>$) to convert date values of the form $'01/12/2014'$ to time values.
  - The function assumes that the first value is the month thus, 01/12/2014 is January 12,2014 NOT Decembr 1, 2014!
  - 01/12/2014 = 12th January 2014 = 2014/01/12 = 12 Jan 2014 = 1389495600

```
print strtotime("Next Friday")."is movie night!";// 1387508400 is mov
print strtotime("7 days ago",time())."is last week!";// 1386628285 i

print strtotime("1 year ago",time());// 1355696979
```

Date and Time Manipulation (continued)

- The date() function returns a string value representing the date.
  - *echodate*(< *dateformatcodes* >, < *timestamp* >);
  - E.g *echodate*("*DMj*, *Y*", *time*()); //*MonJan*14, 2014

Date and Time Manipulation (continued)

Demonstration

PHP Built-in Constants

- Built-in or "magic" constants are available to any PHP script
  - Controlled extensions
    - Some magic constants will only be available if the extension is loaded
    - Magic constants are case-insensitive

PHP Built-in Constants (continued)

- Useful magic constants
  - *__LINE__*
    - Contains current line number
  - *__FILE__*
    - Contains path and filename
  - *__FUNCTION__*
    - Contains function name currently executing
  - *__CLASS__*
    - Contains the class name
  - *__METHOD__*
    - Contains the name of the class function currently executing

Debugging PHP

- Debugging is the process of finding errors in your code during development.
- During the development phase your PHP preprocessor configuration should be set to be report errors.
  - The *display_errors* must be included in your servers *php.ini* file.
    - During development I recommend reporting all errors by setting $error_reporting = E\_ALL$
  - Check this setting now by running *phpinfo*() within php tags on a webpage on your server

Debugging PHP (continued)

- Common errors.
  - Missing semicolon ";"
  - Missing closing brace "}"
  - Misspelling a variable name
    - Remember case-sensitivity
  - Using "=" instead of "=="
    - $if(\$a = \$b) is ALWAYS TRUE)$

Debugging PHP (continued)

- Scaffold your code using echo to print the values of variables and other pertinent program information during execution
- Useful information necessary for debugging includes the call stack and variable values.

Application Logging

- Application logging is a fundamental part of applications.
  - Particularly useful for support and administration
- Application Logging should be used to capture information related to the application
  - Providing information about problems and unusual conditions
  - Contributing additional application-specific data for incident investigation which is lacking in other sources
  - Business process monitoring e.g. sales process abandonment, transactions,connections
  - Audit trails e.g. data addition, modification and deletion, data exports
  - Performance monitoring e.g data load time, page timeouts
  - Compliance monitoring

Application Logging(continued)

- A log entry should include the **Who**, **What**, **Where**, **When**, Result of a logged event
  - Use standard log formats when available

Debugging and Logging PHP

Demonstration