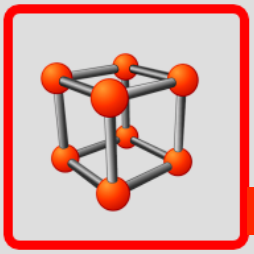
PHP 5 Essentials Course

by Paul Reinheimer

Contents

php | architect
The Magazine For PHP Professionals

Live Interactive
PHP TRAINING



PHP Essentials

Day 1 - PHP Basics

Course Author: **Paul Reinheimer**

The World Wide Web

- The web has been around for a long time
- The web isn't the internet
- Most web pages work by sending your web browser (a *user agent*) an HTML document
- Your web browser then interprets this HTML document and displays the results on the screen
- If the HTML document refers to any external resources (images, cascading style sheets, etc.) they will also be downloaded

Notes

Client Side Languages

- There are a number of client side languages (such as JavaScript), these are transmitted along with the HTML document
- The web browser then interprets the client side language and modifies the output accordingly
- It might even do neat things like request additional information from the server every once in a while and update the page based on this
- The problem with client side languages is that the client might ignore them

Notes

Server Side Languages

- Server side languages (like PHP) are executed on the server, they output the HTML (xml, JavaScript, etc.) that is sent to the browser
- Because they are executed in a controlled environment (your server) their output can be accurately assessed, you're also guaranteed execution
- These languages also have access to server side resources directly (like databases)

Notes

PHP

- PHP was designed to be integrated directly with your HTML code, making it easy to take the pages you already have, and add some dynamic information using PHP
- Three things are required in order for PHP to be used on your server:
  - PHP must be installed and set up with your web-server
  - The document must be saved with an extension the web-server associates with PHP
  - The document must contain the necessary tags

Notes

PHP and your web-server

- This is well beyond the scope of this course. Simply put PHP must be installed, and your web-server must be configured to use PHP
- The web-server will be instructed to associate an extension (generally .php) with PHP.
- When a request comes in for a document with the configured extension, the web server will hand the request off to PHP, which will execute your code and return the output

Notes

PHP Tags

Because one document can contain both HTML and PHP code, we must include instructions indicating where our PHP code starts and ends, this is where PHP tags come into play

```
<html>
<head><title>Test Page</title></head>
<p>
<?php
    echo "This is my PHP test page"
?>
</p>
</html>
```


Notes

PHP Tags

- The PHP engine sees the code on the previous slide, and sends the following to the web browser:

```
<html>
<head><title>Test Page</title></head>
<p>
This is my PHP test page</p>
</html>
```


Notes

A Variable

- Variables are containers for information, they can contain things like integers, strings, booleans, or floats:

```
$name = "Paul";
$greeting = "Good Afternoon";
echo "$greeting $name";
//Good Afternoon Paul
```

- Here we see the = being used as an assignment operator. The value on the right side is assigned to the variable on the left.

Notes

Variables Continued

- When you create a variable you precede its name with a dollar sign ($), this identifies the string as a variable for PHP
- Try to choose descriptive names for variables, names like “greeting” are helpful, they identify what the variable contains, while names like “x” or “foo” will be useless in the future.
- Variable names can only contain letters, numbers or underscores, and must start with a letter or an underscore.

Notes

More Variables

PHP is a *loosely typed* language, which means it tries to do what you tell it to, without making you worry about what kind of information a variable holds:

```
$x = 5;
$y = "3";
$z = $x + $y;
echo "$x + $y = $z";
//5 + 3 = 8
```


Notes

Variable Types

- Even though PHP is loosely typed language, internally it stores data in one of the following types:
  - Integer - A whole number, no decimal places
  - Float - Numbers with decimal places, also used for really large or really small numbers
  - String - Any combination of characters
  - Boolean - TRUE or FALSE
- PHP also has a few other variable types: array, object, resource and null

Notes

Conditions

- We can now create simple programs that print out variables, but they have to do the same thing every time, that's where conditions come into play
- With a conditional statement we can alter the flow of execution, executing some code sometimes, and other code other times

Notes

IF

An if statement will execute given code if the condition it is presented with evaluates to TRUE

```
$hour = 13;
$mGreeting = "Good Morning";
$aGreeting = "Good Afternoon";
if ($hour < 12)
{
    echo "$mgreeting $name";
}else
{
    echo "$aGreeting $name";
}
```


Notes

Exercise 1

- First create a program that stores your name in a variable, and prints it out to the screen
  - Remember your PHP tags
  - Remember to use a .php file extension
- Extend your program to store your age as well as your name, make it print out one message if the age is greater than 19, and another message if the age is less than 19
- Post your source code to the forums once you are done
- If you encapsulate your code with [php] [/php] the code will appear syntax highlighted

Notes

Exercise 1 - Sample Solution

- This (along with every solution presented) is only a sample, there is always more than one way to solve a problem

```
$name = "Paul Reinheimer";
$age = 50;
if ($age > 19)
{
    echo "Hello $name did you vote in the election?";
}else
{
    echo "Hello $name remember to vote when you're older";
}
```


Notes

IF - Complex Example

```
if ($age < 19 AND $age > 55)
{
    //Student & Senior Discount, Double Discount
    if ($day == "Tuesday")
    {
        $discount = 10;
    }elseif ($day == "Sunday")
    {
        $discount = 20;
    }else
    {
        $discount = 0;
    }
}else
{
    $discount = 0;
}
```


Notes

Changing Variables

- The simplest way to change a variable is to simply assign a new value to it:

```
$name = "Paul";
echo "Hello $name"; //Hello Paul
$name = "Bob";
echo "Hello $name"; //Hello Bob
```

- We could also use the string concatenation operator to glue strings together:

```
$greeting = "Hello";
$greeting = $greeting . " " . $name;
echo $greeting; //Hello Bob
```


Notes

Arithmetic Operations

- A selection of arithmetic operations:

```
$apples = 5;
$apples = $apples - 1; //Got Hungry
$apples -= 1; //Still Hungry
$apples--; //Tasty
echo $apples; //2

$oranges = 5;
$oranges = $oranges * 2; //They're on sale!
$oranges /= 2; //Half were rotten
echo ($oranges % 2); // 1
```


Notes

While Loop

- The IF statement executed certain code depending on a condition, the While loop executes given code repeatedly as long as the condition remains true:

```
$peaches = 10;
while ($peaches > 0)
{
    echo "I am Hungry, I will eat a peach";
    $peaches--;
}
```


Notes

Do Loops

- The difference between a do loop and a while loop is that a do loop guarantees at least one execution of the code, since the condition is at the end

```
$apples = 5;
do
{
    $apples--; //got hungry
}while ($apples > 0);

echo $apples;
```


Notes

Exercise 2

- Write a program that stores three variables, hunger, apples and peaches
- As long as hunger is non-zero, it should eat an apple or a peach, eating an apple reduces hunger by one, eating a peach reduces it by two
- It should eat whichever fruit it has more of, it doesn't matter what happens if the two values are the same
- Don't worry about running out of food

Notes

Exercise 2 - Sample Solution

```
$hunger = 10;
$apples = 4;
$peaches = 5;
while ($hunger > 0)
{
    if ($apples > $peaches)
    {
        $apples--;
        echo "Ate an apple\n";
    }else
    {
        $peaches--;
        $hunger--;
        echo "Ate a peach\n";
    }
    $hunger--;
    echo "I have $hunger left\n";
}
```


Notes

Superglobals

- PHP provides several special variables called *Superglobals* they are accessible throughout your program, and generally contain information sent by the user, or information about the request in general
- You can use these variables to get information about your user when they fill out a form, or just when accessing your site

Notes

Basic Superglobals

- Here is some basic code to print out some Superglobals:

```
echo "Hello User, You are using " .
      $_SERVER['HTTP_USER_AGENT'];
echo " as your web browser, and your IP is
      {$_SERVER['REMOTE_ADDR']}";
```

- If you would like to see all the data inside a Superglobal (or any other Array) you can use code like this:

```
<pre>
<?php
print_r($_SERVER);
?>
</pre>
```


Notes

GET and POST

- When a user fills out a form, that data is sent to your script using either GET or POST, the method is selected via the "`method`" attribute of the form tag
- GET is appropriate for retrieving information from a site, like selecting a given page, ordering results or doing a search
- POST is appropriate for sending information to a site, like logging in, submitting a message, purchasing a book, or deleting a bad link

Notes

GET and POST continued

- Once the form has been submitted, the data entered into the form will be available in the corresponding superglobal array ($_GET or $_POST)

greet.html:

```
<form action="greet.php" method="POST">
  Please Enter Your Name:
  <input type="text" name="firstName" maxlength="25">
  <input type="submit">
</form>
```

greet.php:

```
<?php
echo "Hello " . $_POST['firstName'] . " Welcome!";
?>
```


Notes

Exercise 3

- Create an HTML form that accepts three pieces of information, a name, an age, and a colour
- Print a welcome message to the user, depending on their age (children, teenagers, adults)
- Print out all the numbers between 1 and their age, that are not divisible by 3
- If their colour is the same as your favourite colour compliment their choice

Notes

Exercise 3 - Sample Solution

```
if ($age < 10){
    echo "hello youngster";
}elseif($age < 20)
{
    echo "you've grown";
}else {
    echo "good day sir/madam";
}
while ($age-- > 0){
    if ($age % 3){
        echo $age;
    }
}
if ($colour === "purple"){
    echo "Great Choice!";
}
```


Notes

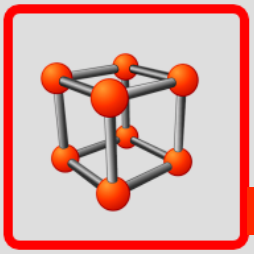
Homework Exercise

- Time to put your code to the test
- Build a form to ask a user for three numbers and a phrase
- Find the largest number of the three, subtract the smallest from it, call that value *result*
- Print the phrase to the screen *result* times
- Using a loop (brute force) calculate the Least Common Denominator (LCD) between the middle number and the largest one
- LCD: The smallest number evenly divisible by two other numbers, the LCD of 7 & 3 is 21

Notes

php | architect
The Magazine For PHP Professionals

Live Interactive
PHP TRAINING



PHP Essentials

Day 2 - Strings & Functions

Course Author: **Paul Reinheimer**

Strings

- Strings are the variable type used most often with PHP, all data that comes from a form will be passed to you as a string (even check boxes)
- Strings can be as long or as short as you like there's no built in limit
- PHP has a lot of built in functions to handle strings, we're going to start looking at those shortly

Notes

Strings & Quotes

- Strings can be quoted in one of three ways
  - ' ' – The single quote or string literal, characters within single quotes will be recorded as is, without variable or escape sequences interpreted and replaced
  - " " – The double quotes, variables and escape sequences will be interpreted and replaced
  - <<< – Heredoc (next slide)

Notes

Strings - Heredoc

- The heredoc syntax functions in a similar manner to double quotes, but is designed to span multiple lines, and allow for the use of quotes without escaping
  $greeting = <<<GREETING
  She said "That is $name's" dog!
  While running towards the thief
  GREETING;
- The closing heredoc tag must be the first thing on the line, the only other permissible character on the line is the semi-colon.

Notes

Comparing Strings

- Strings can be compared using the comparison operator (==) it will return true if the two strings are the same, or functionally equivalent (23 == “23”)
- Strings can also be compared using the identical operator (===) which will also check the type (23 !== “23”)

Notes

Logical Operators

- It may be necessary to combine multiple tests, this can be done with logical operators. Logical operators examine operand(s) and return a single value
  - AND (&&) - Returns true if the two operands are true
  - OR (||) - Returns true if either of the two operands are true
  - XOR - Returns true if exactly one of the two operands is true
  - ! - Returns the inverse if the operand

Notes

Logical Operators - Examples

```
if ($a AND $b)
{
    //1
}

if (($a AND $b) OR ($a AND $c))
{
    //2
}

if (($b OR $c) OR !$a)
{
    //3
}

if (($a AND !$b) OR (!$a AND $c))
{
    //4
}
```


Notes

Playing with strings

- Concatenating strings has already been discussed (using the concatenation operator ".") you can also very easily extract a portion of a string by using an array like syntax:

```
$name = "Paul Reinheimer";
echo $name[0];  //P
echo $name[5];  //R
```

- In earlier versions of PHP the curly braces were the recommended syntax, this decision has been reversed, the {} syntax is new deprecated

Notes

Functions

- Basic concepts like control structures and statements have been introduced but their utility is limited, functions fill in the gaps
- Functions can be used to provide accomplish any number of separate tasks
- Generally functions work by being passed a series of parameters and returning the result (whatever that is)

```
echo strtoupper("Paul Reinheimer");
    //PAUL REINHEIMER
```


Notes

Functions

- Functions take a list of parameters, they use those parameters to decide what to do. The `strtoupper()` function takes one parameter and returns that string in all upper case, turning the string to an upper case string.
- If we would like to change a variable rather than simply echo'ing it out, we can assign the output of the function back to the variable

```
$name = "Paul Reinheimer";
$name = strtoupper($name);
echo $name; //PAUL REINHEIMER
```


Notes

String Functions

- `strpos()` Will search a haystack for a given needle, and return the position, or false if it wasn't found (note the index could be 0)

```
$string = "the quick brown fox";
echo $position = strpos($string,"brown"); //10
```

- `substr()` Will return a portion of a string, depending on the input parameters:

```
echo substr($string, $position); //brown fox
echo substr($string, $position, 5); //brown
echo substr($string, -3); //fox
```


Notes

Reading PHP Documentation

- PHP has *many* built in functions that can do almost anything
- Being able to read PHP documentation will be **critical** to your success as a PHP developer. Please load http://php.net/substr
- You can search for a PHP function by simply going to http://php.net/<search string> for example try loading http://php.net/strlen

Notes

Exercise 1

- Create a form to allow a user to enter two strings, a long phrase and a single word
- Print out the word all in uppercase, then print out the phrase in all lower case except for the word which should be upper case.
- You can accomplish this using the functions we have used thus far, or use the documentation to look for new ones. A little research can make this exercise much easier

Notes

Exercise 1 - Sample Solution

```php
$phrase="The rain in spain falls in the plains";
$word = "spain";

$word = strtoupper($word);
$length = strlen($word);

echo $word . "\n";
$phrase = strtoupper($phrase);
$location = strpos($phrase, $word);

$newPhrase = substr($phrase, 0, $location);
$newPhrase .= strtolower($word);
$newPhrase .= substr($phrase, $location + $length);

echo $newPhrase;
```


Notes

A few more string functions

- Here is a few more useful string functions:

```php
$phrase = "I love pad thai";
echo str_replace("pad thai", "burrito", $phrase);
//I love burrito

echo str_word_count($phrase); //4

$statement = "Hello
My name is Paul!";

echo nl2br($statement);
//Hello<br />
//My name is Paul!
```


Notes

Yet more string functions

- These functions are frequently used to validate user input

```
$data = "preinheimer";
var_dump(ctype_alpha($data));
//bool(true)

$data = "bad ' ' \ data";
var_dump(ctype_alpha($data));
//bool(false)

$data = "90210";
var_dump(ctype_digit($data));
//bool(true)
```


Notes

Exercise 2

- Time to combine a few things we have learned so far
- Use a form to accept a phrase
- Make every other letter upper case
  - You will need to check a character before attempting the replacement
- Print
  - The mixed phrase
  - The number of replacements
  - The number of non-letters that you looked at

Notes

Exercise 2 - Sample Solution

```php
$phrase = "The rain in spain falls mainly in the plains";
$length = strlen($phrase);
$replaceNext = true;
$nonLetters  = 0;
$replacements = 0;
for ($i = 0; $i < $length; $i++) {
    if (ctype_alpha($phrase[$i])) {
        if ($replaceNext === true) {
            $phrase[$i] = strtoupper($phrase[$i]);
            $replaceNext = false;
            $replacements++;
        }else {
            $replaceNext = true;
        }
    }else {
        $nonLetters++;
    }
}
echo $phrase;
echo $nonLetters;
echo $replacements;
```


Notes

User Input

- Tragically, user input is often bad, it can be bad for one of several reasons
  - It could be mangled during transmission
  - The user might not understand what you wanted (data, formatting, etc)
  - The user might be attempting to break your application
- For these reasons it is **critical** that you validate user input before accepting it

Notes

Validating

- The process of validation involves subjecting a piece of data to a series of rules, should the data pass these rules it is termed *valid*, otherwise it is termed *invalid*.
  - Whitelisting - A series of rules are generated describing valid data, each portion of the data is subjected to these rules, if there is a match it is valid
  - Blacklisting - A series of rules are generated describing invalid data, should any portion of the data match, it is considered invalid

Notes

Some sample validation rules

- Username - Only alpha numeric characters
  - ctype_alnum()
- Message Topic - Only alpha numeric, period, space, dash
  - Replace period, space & dash with a 0 then check for validity with ctype_alnum()
- Phone Number - ?
- Address - ?
- Zip Code - ?
- Name - ?

Notes

Exercise 3

- Create a form to accept a:
  - Name
  - Address
  - Username
  - Password
  - Tag line
- Validate the data, defend your validation technique in your comments

Notes

Exercise 3 - Sample Solution

- Rather than look at mine, look at the solutions posted by your classmates

Notes

Formatted Output

- Methods to print out strings have already been discussed but we lack a manner to format them
- printf() takes a string containing formating specifiers, followed by a comma separated list of variables to be replaced into the string echoing out the result
- sprintf() takes the same specifiers but returns the result
- vprintf() takes the same specifiers, but takes the variables as an array

Notes

Formatting

- Formatting examples

```
$fakeSalary = 104000 / 12; //8666.666666
echo "Fake Salary: $fakeSalary";
     //Fake Salary: 8666.6666666667
printf("Fake Salary: %.2f", $fakeSalary);
     //Fake Salary: 8666.67

$total = 15.50; $bill = 20.00; $change = $bill - $total;
echo "Total: $total, Bill = $bill, Change = $change";
//Total: 15.5, Bill = 20, Change = 4.5

printf("Total: %.2f, Bill = %.2f, Change = %
05.2f", $total, $bill, $change);
//Total: 15.50, Bill = 20.00, Change = 04.50
```


Notes

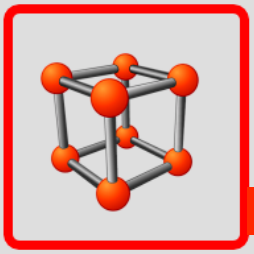
Homework Exercise

- Present a form to the user accepting: a name, phone number, the number of apples, pizzas and televisions the user would like to purchase, as well as a tax rate (either in the format of a percentage or decimal). Validate these inputs.
- Print out a formatted invoice, including the above information, subtotal & total. Don't display products if none were purchased.
- Expand the program to allow for more products, store name, and products that are not taxed, validate everything

Notes

php | architect
The Magazine For PHP Professionals

Live Interactive
PHP TRAINING



PHP Essentials

Day 3 - Array & User Functions

Course Author: **Paul Reinheimer**

Arrays

- Arrays are the appropriate way to store sets of related data, rather than using many different variables, you can use one variable to store many pieces of information
- Enumerated Arrays
  - Appropriate for storing pieces of related information, such as lists. Data will be index numerically
- Associative Arrays
  - Appropriate for storing information where the key describes the data .

Notes

Enumerated Arrays

- Can be created in two ways:

```
$fruit =
     array("apple", "banana", "cherry");

$vegetable = array();
$vegetable[] = "carrot";
$vegetable[] = "pepper";
$vegetable[] = "onion";

echo $fruit[0]; //apple
echo $vegetable[1]; //pepper
```


Notes

Associative Arrays

- Similarly can be created in two ways

```
$people = array
("name"=>"Paul", "location"=>"Montreal");

$friend = array();
$friend['name'] = 'Tom';
$friend['location'] = 'Windsor';

echo $people['name']; //Paul
echo $friend['location']; //Windsor
```


Notes

Examining an Array

- The easiest way to examine an Array during development is with `print_r()` which will print out a plain text formated representation of the array

```
print_r($friend);
/*
Array
(
    [name] => Tom
    [location] => Windsor
)
*/
```


Notes

Exercise 1

- Create an array that contains a list of several people in this session
- Create another array holding a list of your favourite movies and their rating *(eg. 4 Stars)*
  - Consider which element should be the key, and which should be the value. Which is more likely to be unique, the title, or the rating?
- Practice accessing parts of the array using either `print_r()` or directly with `echo`

Notes

Exercise 1 - Sample Solution

- Easy Possibilities

```
$class = array();
$class[] = "Paul Reinheimer";
$class[] = "Sean Coates";
$class[] = "Marco Tabini";

$movie = array();
$movie['Serenity'] = 5;
$movie['Bad Santa'] = 5;
$movie['Matrix Revolutions'] = 2;
```


Notes

Accessing the Array

- The `foreach()` loop is the easiest way to iterate through the items in an array

```
$fruits = array("apple", "banana", "cherry");
foreach($fruits AS $fruit)
{
    echo "I like $fruit\n";
}
/*
    I like apple
    I like banana
    I like cherry
*/
```


Notes

Accessing the Array

- Associative arrays have a slightly different syntax

```
$squares = array(2 => 4, 3 => 9, 4 => 12);
foreach ($squares AS $value => $square)
{
    echo "$value squared is $square\n";
}
/*
    2 squared is 4
    3 squared is 9
    4 squared is 12
*/
```


Notes

Internal Array Pointer

- You can also use the internal pointer to iterate an array

```
$countries = array("Canada", "Sweden", "Spain", "Norway");
while (list($index, $country) = each($countries))
{
    echo "I liked visiting $country\n";
}
$countries[] = "USA";
while (list($index, $country) = each($countries))
{
    echo "I liked visiting $country\n";
}
/*
    I liked visiting Canada
    I liked visiting Sweden
    I liked visiting Spain
    I liked visiting Norway
    I liked visiting USA
*/
```


Notes

Exercise 2

- Create an array to store a series of favourite foods
- Iterate over that array prepending the phrase “I like eating “ to each element using the internal array pointer
- Iterate over the array again using a foreach loop to print out each element

Notes

Exercise 2 - Sample Solution

```
$food = array
     ("burrito", "pad thai", "pizza");
while(list($key, $value) = each($food))
{
    $food[$key]="I like eating " . $value;
}

foreach($food AS $type)
{
    echo $type . "\n";
}
```


Notes

Sorting Arrays

- PHP has a series of built in functions to handle sorting of an array, this example array will be sorted using various functions over the next several slides

```
$students = array();
$students['Paul'] = 'Canada';
$students['Bob'] = 'USA';
$students['Winston'] = 'United Kingdom';
$students['David'] = 'South Africa';
```


Notes

sort()

- The sort function sorts the values in an array, and re-keys the array to the new order

```
Array
(
    [0] => Canada
    [1] => South Africa
    [2] => USA
    [3] => United Kingdom
)
```


Notes

ksort()

- The ksort function will sort an array by keys, maintaining the key value associations

```
Array
(
    [Bob] => USA
    [David] => South Africa
    [Paul] => Canada
    [Winston] => United Kingdom
)
```


Notes

asort()

- The asort() function can be used to sort an array by the values, while maintaining key value associations

```
Array
(
    [Paul] => Canada
    [David] => South Africa
    [Bob] => USA
    [Winston] => United Kingdom
)
```


Notes

shuffle()

- Rather than sorting an array, shuffle randomizes it
- Does shuffle maintain key-value associations?

Notes

Exercise 3

- Create an associative array that contains a shopping list, product name as a key, price as a value. Add several items, at different price points.
- Sort the array by products, then by price, print both lists to the screen
- Research the sorting functions to see what sort of optional parameters you can pass
- Try sorting the array numerically, or alphabetically, what happens?

Notes

Exercise 3 - Sample Solution

```
$shoppingList = array();
$shoppingList['banana'] = 0.69;
$shoppingList['bread'] = 1.29;
$shoppingList['milk'] = 3.99;
$shoppingList['pizza'] = 14.99;
$shoppingList['wine'] = 34.99;
$sortedList = $shoppingList;
echo "asort()\n";
asort($sortedList);
print_r($sortedList);
/*  asort()
Array
(
    [banana] => 0.69
    [bread] => 1.29
    [milk] => 3.99
    [pizza] => 14.99
    [wine] => 34.99
)  */
```


Notes

Exercise 3 - Sample Solution

- The SORT_STRING flag sorts by character, as such numbers are sorted incorrectly

```
$sortedList = $shoppingList;
echo "asort(, SORT_STRING)\n";
asort($sortedList,SORT_STRING);
print_r($sortedList);
/*
asort(, SORT_STRING)
Array
(
    [banana] => 0.69
    [bread] => 1.29
    [pizza] => 14.99
    [milk] => 3.99
    [wine] => 34.99
)
*/
```


Notes

Exercise 3 - Sample Solution

- The SORT_NUMERIC flag will sort the values numerically

```
$sortedList = $shoppingList;
echo "asort(, SORT_NUMERIC)\n";
asort($sortedList,SORT_NUMERIC);
print_r($sortedList);
/*
asort(, SORT_NUMERIC)
Array
(
    [banana] => 0.69
    [bread] => 1.29
    [milk] => 3.99
    [pizza] => 14.99
    [wine] => 34.99
)
*/
```


Notes

Creating your own functions

- We've been using functions for a while, functions built into PHP
- We can actually create our own, this will make writing bigger programs easier, and realistically, possible
- Functions can work in a series of different ways, they really make creating larger applications easier, since different tasks can be allocated to different functions

Notes

First Function

- This function accepts a single parameter, and returns the passed parameter multiplied by five

```
function x5($number)
{
    return ($number * 5);
}
echo x5(10); //50
```


Notes

Function Scope

```
function scopeDemo($apples)
{
    $apples = 10;
    echo $apples;
}

$apples = 5;
scopeDemo($apples);
echo $apples;
```


Notes

Exercise 4

- Create two functions, one that creates and returns an array with the names of various places
- The other function should accept an array as a parameter, and prints it out to the screen

Notes

Exercise 4 - Sample Solution

```php
function create()
{
    $names = array();
    $names[] = "Windsor";
    $names[] = "New York";
    $names[] = "Montreal";
    $names[] = "Bergen";
    $names[] = "Waterloo";
    return $names;
}

function display($array)
{
    foreach ($array AS $element)
    {
        echo $element;
    }
}

$names = create();
display($names);
```


Notes

Multiple Parameters

- Functions can also accept more than one parameter

```php
function repeatGreeting($phrase, $count)
{
    for ($i = 0; $i < $count; $i++)
    {
        echo $phrase . "\n";
    }
}
repeatGreeting("Hello, how are you today?", 3);
/*
Hello, how are you today?
Hello, how are you today?
Hello, how are you today?
*/
```


Notes

Optional Parameters

- Functions can also accept optional parameters, they must come after required ones

```
function optionalGreeting($phrase, $count = 1)
{
    for ($i = 0; $i < $count; $i++)
    {
        echo $phrase . "\n";
    }
}
optionalGreeting("Good Day");
optionalGreeting("Good Eve", 2);
/*
Good Day
Good Eve
Good Eve
*/
```


Notes

Big function example

```
function lcd($a, $b)
{
    for($i = 1; $i <= $b; $i++)
    {
        if (evenlyDivided($i * $a, $b))
        {
            return $i;
        }
    }
}

function evenlyDivided($x, $y)
{
    if ($x % $y == 0)
    {
        return true;
    }else
    {
        return false;
    }
}
echo "The LCD of 7 and 3 is " . lcd(7, 3);
```


Notes

By Reference vs By Value

- When the variable is preceded by an ampersand (&) the variable is passed by reference, rather than by value. Any changes the function makes to the variable will apply externally

```
function x7(&$value)
{
    $value *= 7;
}

$apples = 5;
x7($apples);
echo $apples; //35
```


Notes

Homework

- Create a function that will validate a username, and another that will validate a tag line
- Create a form that will accept a series of username & tag line combinations, validate each in turn using the functions, then store them all in a single array
- Iterate over the array printing them all out in an attractive (HTML) manner
- Bonus: Repopulate the form with the data if an element fails the test

Notes

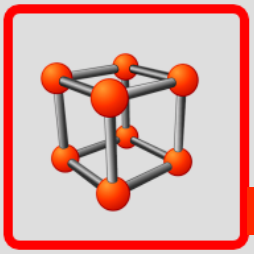
Challenge

- Research the `usort()` function
- Create an array of associative arrays arrays, each internal array should hold two pieces of information a name and an age
- Create a function that will accept two parameters to be used in combination with the `usort()` function to sort your array by age

Notes

php | architect
The Magazine For PHP Professionals

Live Interactive
PHP TRAINING



PHP Essentials

Day 4 - Cookies & OOP

Course Author: **Paul Reinheimer**

HTTP

- We use PHP to develop web applications, generally users will enter the address of our application into their web browser, which will then load and display the application
- The protocol that the web browser will use to communicate with the web server is called the Hyper Text Transfer Protocol (HTTP)
- This protocol allows the web browser to identify the document it wants, and the web server to indicate wether the request was successful

Notes

Sample HTTP Headers

```
GET / HTTP/1.1
Host: example.preinheimer.com
User-Agent: Mozilla/5.0 (Macintosh; U; Intel Mac OS X; en-US; rv:1.8.0.8)
Gecko/20061025 Firefox/1.5.0.8
Accept: text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache

HTTP/1.x 200 OK
Date: Thu, 14 Dec 2006 22:56:09 GMT
Server: Apache/2.2.3 (Debian) mod_ssl/2.2.3 OpenSSL/0.9.8c
Last-Modified: Mon, 28 Aug 2006 01:56:37 GMT
Etag: "5f0348-59-3c7e3f40"
Accept-Ranges: bytes
Content-Length: 89
Keep-Alive: timeout=15, max=99
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```


Notes

Forms GET vs POST

- GET is appropriate for information retrieval, like running a search at Google, sorting the list of things on a page, or viewing a forum post
- POST is appropriate for submitting information, like registering at a site, purchasing a book, or selling stocks
- It's easy to look at the headers associated with a request using a packet sniffer like ethereal or a browser extension (Live Headers in Firefox is popular)

Notes

Forms

- When you submit a form the information included in the form is encapsulated within the HTTP request

```
<form action="submit.php" method="GET">
    <input name="test" type="text">
    <input type="submit">
</form>
```


Notes

Form Submission - GET

- With a GET form post the form data is shown in the address bar, and in the first line of the header

```
GET /submit.php?test=test HTTP/1.1
Host: example.preinheimer.com
User-Agent: Mozilla/5.0 (Macintosh; U; Intel Mac OS X; en-US; rv:
1.8.0.8) Gecko/20061025 Firefox/1.5.0.8
Accept: text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://example.preinheimer.com/form.html
```


Notes

Form Submission - POST

- With a POST submission the data is not visible in the address bar, and it is sent within body of the header

```
POST /submit.php HTTP/1.1
Host: example.preinheimer.com
User-Agent: Mozilla/5.0 (Macintosh; U; Intel Mac OS X; en-US; rv:1.8.0.8) Gecko/
20061025 Firefox/1.5.0.8
Accept:text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://example.preinheimer.com/form.html
Content-Type: application/x-www-form-urlencoded
Content-Length: 9
test=test
```


Notes

Exercise 1

- Create a simple form using the GET method, submit it to a script of your creation that prints out the $_GET super global array
- Book mark the resulting page (after the form submission) close your browser, open it again then visit the bookmarked page, your data should appear
- Change the form to use POST, repeat
- Investigate ways to view HTTP headers

Notes

Exercise 1 - Sample Solution

- The code was trivial, the purpose was the investigation
  - Bookmarking the GET result worked fine, bookmarking the POST did nothings
- Resources
  - Ethereal (packet sniffer)
    - http://www.ethereal.com/
  - HTTP Developer's Handbook
    - Amazon
  - Live Headers
    - http://livehttpheaders.mozdev.org/

Notes

HTTP State

- State (or statelessness rather) is a rather difficult concept
- Essentially, each and every request you send to the webserver is independent of every other request, so when you view an index document, then a follow up page the webserver doesn't know (or care) that you're the same user
- This can cause problems for applications that want to remember you from one request to the next

Notes

A Solution

- Cookies!
- Realizing that applications would work better if they could remember who their users wore, Netscape (*remember them?*) added two headers `Set-Cookie:` and `Cookie:`
- The `Set-Cookie:` header is sent by the server to the client
- On subsequent requests the client will send the `Cookie:` header (along with any data) to the server

Notes

Setting a Cookie

- You can set a cookie in one of two ways, using the header() function, or with setcookie()
- header() - When using header you can specify the cookie parameters as per the cookie specifications

```
header("Set-Cookie: name=paul");
```

- setcookie() - A simple wrapper for sending those cookies, takes the work out of setting a cookie

```
setcookie("name", "paul");
```


Notes

Accessing those cookies

- Like GET or POST PHP provides a super global for accessing cookies, it's called $_COOKIE

```
echo "Hello " . $_COOKIE['name'];
```


Notes

Exercise 2

- Part 1
  - Create two PHP scripts, page 1 should set a cookie, and present a link to page 2. Page 2 should simply print out the value in the cookie from page 1
- Part 2 (aye there's the rub)
  - Research the additional options in setcookie() see how you can write a cookie that can expire.
  - Modify your Page 2 to print out one message if the cookie is present, and another message if the cookie is not

Notes

OOP

- Arrays have been examined as a way of grouping related data
- Functions are a method of grouping pieces of related code together
- Objects are a way of grouping pieces of related code *and* data together, all in one thing

Notes

Terminology and use

- instantiate - When you create an “instance” of an object and assign it to a variable
- method - The name given to a function within an object
- property - The name given to a variable that exists within the scope of an object
- constructor - A special method that is called when the object is instantiated
- destructor - A special method that is called when the object is destroyed

Notes

Object Example

```
class pet
{
    public $name;
    private $sound;
    function __construct($name, $sound)
    {
        $this->name = $name;
        $this->sound = $sound;
    }
    function makeSound()
    {
        echo $this->sound . "\n";
    }
}
$tinker = new pet("Tinker", "meow"); $tinker->makeSound();
$buddy = new pet("Buddy", "woof"); $buddy->makeSound();
/*
    meow
    woof
*/
```


Notes

Object Example

```
class person
{
    private $name;
    private $age;
    function __construct($givenName, $givenAge)
    {
        $this->name = $givenName;
        $this->age = $givenAge;
    }
    function sayGreeting()
    {
      echo $this->name . " is " . $this->age . " years old\n";
    }
}
$me = new person("Paul", 100);
$me->sayGreeting();
//Paul is 100 years old
```


Notes

Exercise 3

- Create an object called pizza, it should have a property called toppings, it should be an array
- During construction, the pizza should be instantiated with a size and a name “meat lovers”
- Add a method called add topping, which pushes a new topping onto the toppings property
- Add a final method called bake which prints out all the information on the pizza

Notes

Exercise 3 - Sample Solution

```
class pizza
{
    public $name;
    public $size;
    public $toppings;
    function __construct($name, $size)
    {
        $this->name = $name;
        $this->size = $size;
        $this->toppings = array();
    }
    function addTopping($topping)
    {
        $this->toppings[] = $topping;
    }
    function printInformation()
    {
        echo "You ordered a {$this->size} {$this->name}\n";
        foreach($this->toppings as $topping)
        {
            echo "It will have $topping on it \n";
        }
    }
}
```


Notes

Visibility

- Properties and methods can be assigned a visibility, there are three possibilities: Public, Private, Protected
  - Public - This property is externally accessible, it can be read from or written to
  - Private - This property can not be accessed externally
  - Protected - This property can only be accessed through children

Notes

Visibility Example

```
class person
{
    public $name;
    private $age;
    function __construct($givenName, $givenAge)
    {
        $this->name = $givenName;
        $this->age = $givenAge;
    }
    function sayGreeting()
    {
   echo $this->name . " is " . $this->age . " years old\n";
    }
}
$me = new person("Paul", 100);
echo "Hello " . $me->name;
echo "You are " . $me->age . " years old"; //wont work
```


Notes

Exercise 4

- Create an object called pet, it should store name, type (cat, dog), and age
- The three properties should be stored during construction
- Create a method called birthday, age cat's 6 years during a birthday, and dogs 7
- Create another method to print out all the information about the pet
- Instantiate two different copies of the pet, age them both, then print out the information

Notes

Exercise 4 - Sample Solution

```
class pet
{
    public $name;
    private $age;
    private $type;
    public function __construct($name, $age, $type)
    {
        $this->name = $name;
        $this->age = $age;
        $this->type = $type;
    }

    public function birthday()
    {
        if ($type == "cat")
        {
            $this->age += 6;
        }else
        {
            $this->age += 7;
        }
    }
```


Notes

Exercise 4 - Solution Continued

```
 public function info()
    {
        echo "Cute {$this->name} is a {$this->type} which is {$this-
>age} years old \n";
    }

}

$tinker = new pet("tinker", 16, "cat");
$rover = new pet("rover", 5, "dog");
$tinker->birthday();
$rover->birthday();
$tinker->info();
$rover->info();
```


Notes

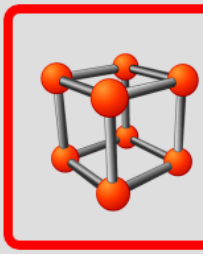
Homework

- Create a form to accept several different pieces of information about a car (year, kilometers/ miles, colour, model)
- Create a few functions to validate those pieces of information
- Once validated, store it in a users cookie, print out information on the vehicle
- Store that information in an object, include methods to add a year or mileage

Notes

php | architect
The Magazine For PHP Professionals

Live Interactive
PHP TRAINING



PHP Essentials

Day 5 - Sessions & OOP

Course Author: **Paul Reinheimer**

Includes

- Functions & OO are very useful tools, but using them all can result in very long files, and reusing code between applications is difficult if not impossible
- Luckily PHP provides mechanisms to write code in one file, the bring it into another script:

```
include('myObject.inc.php');
require('myObject.inc.php');
```


Notes

Include vs Require

- They both do the same thing, they read the specified file into the current one (essentially replacing the include or require line with the contents of the destination file)
- The difference between include and require only comes into play when the file isn't found: `include()` returns a notice while `require()` returns a fatal error.

Notes

Example

```
//index.php
echo "Hello";
include('greeting.inc.php');

//greeting.inc.php
echo "Good Morning, and Good Afternoon";

/*
    Output:
        Hello
        Good Morning, and Good Afternoon

*/
```


Notes

Exercise

- Create two files, include one from the other, have them both print something to see it working
- Further steps:
  - Declare a variable in one file, use it in another how does that work?
  - What if you declare a function in one file, use it in another? Does order matter?

Notes

Exercise 1 - Sample Solution

- Remarkably similar to the example:

```
//index.php
echo "Hello";
include('greeting.inc.php');

//greeting.inc.php
echo "Good Morning, and Good Afternoon";
```


Notes

Sessions

- Cookies are great, but something of a pain to deal with
- Cookies are a client side data store, which means the data stored inside can be inspected and modified by the user, while occasionally desirable this is frequently a draw back
- Sessions are PHP's way of making storing state easier, PHP does all the hard work, giving the user a cookie containing a unique identifier, then matching that identifier with a local*ish* datastore

Notes

Sessions - Continued

- User gets a cookie
- Server gets a file (maybe)
- The programmer gets a super global array with full read write access (information is available from the array in the same request it is placed there):

```
$_SESSION['userName'] = "preinheimer";
```


Notes

Sessions - The Good

- User only has an ID, they can change the ID but this will only invalidate their session, the likely hood of them randomly choosing the ID of another user is remarkably low
- Since the datastore is on the server, the user can't change anything
- The data stored in the session does not need to be transferred to and from the client with each and ever request

Notes

Sessions - The Bad

- Since the session id is the only thing identifying a client, if that id is stolen an attacker may impersonate the victim the vulnerable surfaces are:
  - The local datastore, on shared servers this is frequently /tmp which may be world readable
  - On servers supporting GET based sessions a user may post a link including their session id, or may be a victim of session fixation
- Sessions may not scale well

Notes

Using Sessions

- Using sessions is almost as easy as using any other array, simply write data to an array to store information, and read data from an array to retrieve it, the big difference is that the information will be available on subsequent requests.
- The one caveat with sessions is that it is imperative to call `session_start();` at the very **top** of any page that will make use os session data

Notes

A session example:

```
session_start();
$_SESSION['userName'] = "preinheimer";

//Next Page
echo $_SESSION['userName'];
```

- Easier than you thought?

Notes

Exercise 2

- Phase 1
  - Create one page that to store several pieces of information about a user inside a session, on the next page print out that information
- Phase 2
  - Create a form that accepts a username, the destination page should store that value in a session, and present another form accepting a password, the final page should display both pieces of information

Notes

Exercise 2 - Sample Solution

```
//Page 1
session_start();
$_SESSION['userName'] = $_POST
     ['username'];

//Page 2
session_start();
echo "Username " . $_SESSION
     ['userName'] . "<br>\n";
echo "Password " . $_GET
     ['password'] . "<br>\n";
```


Notes

Session Security

- Since session IDs can be vulnerable to attack, it is wise to take some steps to protect them
- A common practice is one of browser fingerprinting, where several browser attributes are stored in the session on the first request, then compared with the received values on subsequent requests
- Another common technique is to regenerate session identifiers when a user changes access levels (logging in for example)

Notes

Browser Fingerprinting

```
$_SERVER =
Array (
    [HTTP_ACCEPT] => */*
    [HTTP_ACCEPT_LANGUAGE] => en
    [HTTP_ACCEPT_ENCODING] => gzip, deflate
    [HTTP_USER_AGENT] => Mozilla/5.0 (Macintosh; U; Intel Mac OS X; en)
AppleWebKit/418.9.1 (KHTML, like Gecko) Safari/419.3
    [HTTP_CONNECTION] => keep-alive
    [HTTP_HOST] => example.preinheimer.com
    [PATH] => /usr/local/bin:/usr/bin:/bin
    [SERVER_SIGNATURE] =>
    [SERVER_SOFTWARE] => Apache/2.2.3 (Debian) mod_ssl/2.2.3 OpenSSL/0.9.8c
    [SERVER_NAME] => example.preinheimer.com
    [SERVER_ADDR] => 65.111.169.79
    [SERVER_PORT] => 80
    [REMOTE_ADDR] => 24.201.61.253
    [DOCUMENT_ROOT] => /www/example.preinheimer.com
    [SERVER_ADMIN] => paul@preinheimer.com
    [SCRIPT_FILENAME] => /www/example.preinheimer.com/trainingExamples.php
    [REMOTE_PORT] => 47170
    [GATEWAY_INTERFACE] => CGI/1.1
    [SERVER_PROTOCOL] => HTTP/1.1
    [REQUEST_METHOD] => GET
    [REQUEST_TIME] => 1166401662
```


Notes

Session Regeneration

- Session regeneration is a common defense to session hijacking, it involves replacing the current session id with a new one
- As of PHP 5.1.0 you can also destroy this session by passing session_regenerate_id() true, take this option if you can get it

Notes

Exercise 3

- Create a log in form, accept a username and password
- Validate this information (direct comparison if you wish) and generate a browser fingerprint
- Provide a link to a third page which confirms the browser fingerprint

Notes

Exercise 3 - Sample Solution

```
//On login
$_SESSION['fingerprint'] =
  md5($_SERVER['HTTP_USER_AGENT'] .
      $_SERVER['HTTP_ACCEPT_LANGUAGE']);
//Subsequent Pages
$fingerprint = md5(
     $_SERVER['HTTP_USER_AGENT'] .
     $_SERVER['HTTP_ACCEPT_LANGUAGE']);
if ($fingerprint != $_SESSION['fingerprint'])
{
    //Bad fingerprint, kick the user
      to the login page
}
```


Notes

Advanced OOP techniques

- Discussed Today:
  - Extending
    - Using one object as a base, then building upon it with other objects
- Recommended Self Study
  - Sleep
    - Preparing an object for serialization
  - Wakeup
    - Waking up from serialization

Notes

Extending

- Situations in which several different objects will have the same base are common, this is facilitated with the extends keyword
- One object provides the base for one or many others
  - For example, an automobile base class, that can be extended by sports car, SUV, or minivan

Notes

Extending

```
class automobile
{
    protected $km;
    protected $year;
    protected $lights;

    public function __construct()
    {
        $this->lights = false;
    }
    public function addGas()
    {
        //
    }
    public function toggleLights()
    {
        $this->lights = !$this->lights;
    }
}
```


Notes

Extending Continued

```
class sportsCar extends automobile
{
    protected $superCharger;
    public function __construct($hasSuperCharger)
    {
        parent::__construct();
        $this->superCharger = $hasSuperCharger;
    }

    public function driveReallyFast()
    {

    }
}
```


Notes

Exercise 4

- Build a base class called person with several basic properties, and at least one method: birthday
- Extend the person class with a class called friend, add methods like phoneCall,
- Instantiate two friends, age them, and make a phone call

Notes

Exercise 4 - Anti-Solution

- The solutions provided in class only have so much to offer, look at the other solutions in the discussion forums, note differences and similarities between their solutions and yours

Notes

Extra Functions

- A couple other places to look with interest:
  - date()
  - strtotime()
  - htmlentities()
  - str_word_count()
  - nl2br()
  - explode()
  - implode()

Notes

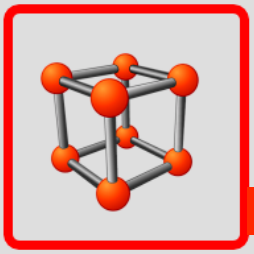
Homework

- Build upon several pieces of your earlier code:
  - Use your previous exercise build a log in form accepting a username and password, continue browser fingerprinting throughout
  - Once the user is authenticated, allow them to design a pizza, on the first page have them list a type and size, extend the pizza with order to include cost information, cost increases with toppings
  - Store the object in the users session (serialize())
  - On a repeating page allow the user to add items to the pizza, storing them inside the pizza object

Notes

php | architect
The Magazine For PHP Professionals

Live Interactive
PHP TRAINING



PHP Essentials

Day 5 - Databases and Includes

Course Author: **Paul Reinheimer**

Databases

- Database Management Systems (DBMS) are pieces of software that manage your data for you
- Databases are a collection of tables that actually store information
- Fields or Columns label types of information, like username, password, email
- Rows or records contain correlated information under each of the columns.

Notes

Sample Table

- The easiest way to think of a table is to visualize it

username	firstName	lastName	email	password
preinheimer	paul	reinheimer	paul@phparch.com	password
scoates	sean	coates	sean@phparch.com	secret
marco	marco	tabini	marcot@phparch.com	random

Notes

Creating a table

- To create a table we simply need to describe the table we want to the DBMS:
- ```
  mysql> CREATE TABLE users (
    username varchar(32) NOT NULL,
    password char(32),
    email varchar(64));
  ```
- ```
  Query OK, 0 rows affected (0.01
  sec)
  ```


Notes

Checking your work

```
mysql> DESCRIBE users;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| username | varchar(32) | NO   |     |         |       |
| password | char(32)    | YES  |     | NULL    |       |
| email    | varchar(64) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```


Notes

Exercise 2

- Create a table to contain the following information
  - Product ID
  - Product Name
  - Product Price
  - Description
  - Number on hand
- Choose the best available data type (float, char, int, varchar)

Notes

Exercise 2 - Solution

- ```
  mysql> CREATE TABLE products   (
    productID int,
    productName varchar(50),
    price float,
    description varchar(100),
    numberOnHand int);
  ```
- ```
  Query OK, 0 rows affected (0.04
  sec)
  ```


Notes

Adding data to your table

- `mysql> INSERT INTO users VALUES ('preinheimer',md5('password'), 'paul@phparch.com');`
- `mysql> INSERT INTO users (username, password, email) VALUES ('scoates', md5('password'), 'sean@phparch.com');`
- `mysql> INSERT INTO users (username, email) VALUES ('marcot', 'marco@phparch.com');`

Notes

Checking your work

```
mysql> SELECT * FROM users;
+-------------+------------------------------------+------------------+
| username    |      password                      | email            |
+-------------+----------+-------+-----------------------+-------------+
| preinheimer | 5f4dcc3b5aa765d61d8327deb882cf99   |  paul@phparch.com |
| scoates     | 5f4dcc3b5aa765d61d8327deb882cf99   |  sean@phparch.com |
| marcot      |                                    |marcot@phparch.com |
+-----------+-------------+-------+-----------------------+-------------+
3 rows in set (0.00 sec)
```


Notes

Exercise 3

- Add at least three records to your products table, check to make sure they are there
- Try all three ways of adding records

Notes

Exercise 3 - Sample Solution

- `mysql> INSERT INTO products VALUES(1,'banana', 0.69,'Beautiful ripe bananas', 1000);`
  `Query OK, 1 row affected (0.00 sec)`
- `INSERT INTO products (productID, productName, price, description, numberOnHand) VALUES (2, 'apple', 0.49, 'ripe apple', 123);`
  `Query OK, 1 row affected (0.00 sec)`
- `mysql> INSERT INTO products (productID, productName, numberOnHand) VALUES (3, 'pear', 0);`
  `Query OK, 1 row affected (0.00 sec)`

Notes

Retrieving records

- A method to retrieve records has already been briefly introduced: the select statement:
  `SELECT <fields> FROM <table> [WHERE <clause>]`
- `SELECT * FROM users;`
- `SELECT username, email FROM users;`
- `SELECT * FROM users WHERE username = 'preinheimer';`

Notes

Updating records

- You can update records using the UPDATE statement, it's most useful when combined with a WHERE clause:
  `UPDATE <table> SET <column> = <value> [WHERE <clause>]`
- `UPDATE users SET password = md5('newpass');`
- `UPDATE users SET email = 'paul@preinheimer.com' WHERE username = 'preinheimer';`

Notes

Deleting Records

- Deleting records is also easy:
  `DELETE FROM <table> [WHERE <clause>]`
- `DELETE FROM users;`
- `DELETE FROM users WHERE username = 'preinheimer';`

Notes

Exercise 4

- Select a few of the products in your database
  - Select records with at least 1 item on hand
  - Select the record with a given price
- Delete a few records in your database
  - Delete the items with no items on hand
- Update records in your database
  - Reduce the number of items on hand for all the items with a price under a set threshold

Notes

Bringing PHP into the picture

- PHP will connect to a database and use a link id (variable type is resource) to identify the connection

```
$linkID =
   mysql_connect('localhost', 'username', 'password');
mysql_select_db('example');
if(!$linkID)
{
  echo "no link";
}
$result = mysql_query('SELECT * FROM users');
while ($query_data = mysql_fetch_assoc($result))
{
    print_r($query_data);
}
```


Notes

Exercise 5

- Bring PHP into the picture, print out all the information from your table, experiment with various select queries
- Once that's done use the foreach() construct to iterate over the results and print them out (notice that they're just like an associative array)

Notes

Other MySQL Functions

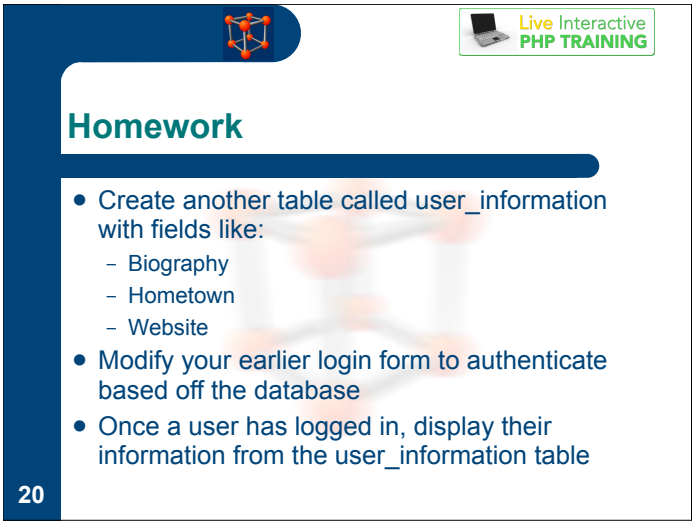
- mysql_affected_rows()
  - Accepts a link id, returns the number of rows affected by the previous insert/update/replace/delete query
- mysql_num_rows()
  - Accepts a resource for a result, returns the number of rows in the result set
- mysql_unbuffered_query()
  - Sends a query, the results can only be iterated through forwards, uses less memory

Notes

Exercise 5

- Build an object called myDatabase it should:
  - Have a constructor that accepts a host, username, password, and database, the connection should be made and the link id should be stored in a private property
  - Have a method that executes a query and returns a result resource
  - Have a method that accepts a result resource and gives you the next result

Notes



Notes