



CEWP 459

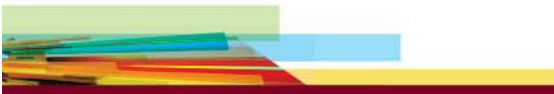
PHP Programming with MySQL – Level I
Intro Slides



Instructor

- Brendan Wood
 - Email: brendan.wood@concordia.ca
 - Phone: (514) 892-7883

Generally unavailable Monday and Tuesday.



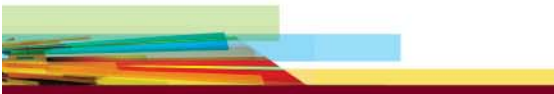
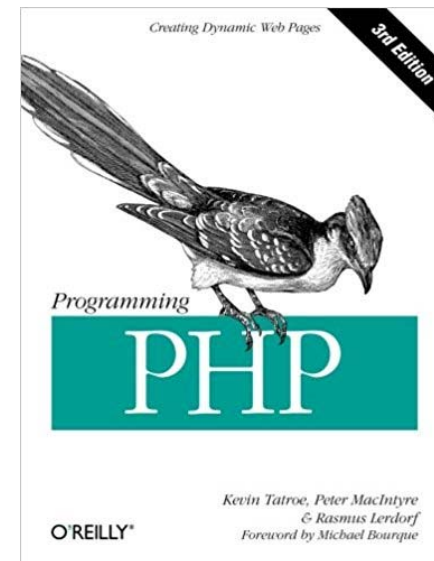
Classroom Policies

- Please, no **cellphone use** in class. Please exit the class to take any calls or text messages.
- **Personal laptops** are permitted. Please use the same software as we use in class to ensure compatibility.



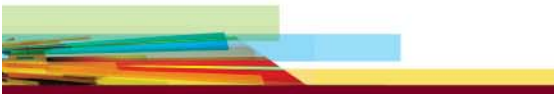
Resources

- PHP Cheat-Sheet on Moodle
- Suggested reference manual
Programming PHP, O'Reilly Media; 3rd edition ISBN
1449392776
- PHP on the web;
<http://www.php.net>



Today's Objectives

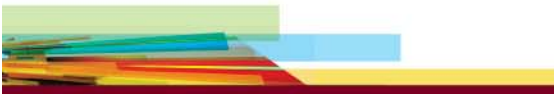
1. Course outline
2. PHP Introduction
3. Basics
4. Flow Control
5. String Functions
6. Date Functions
7. Functions





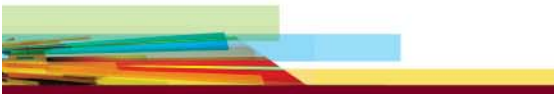
PHP (recursive acronym for PHP: Hypertext Preprocessor) is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML.

www.php.net



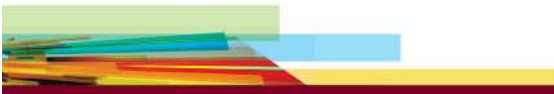
PHP Basics

- PHP is a **server-side** scripting language, so scripts are executed on the server
- PHP supports many **databases** (MySQL, Oracle, PostgreSQL, etc.)
- PHP is **open source** software, which is free to download and use



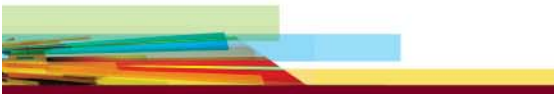
Some Benefits ...

- Easy to learn, easy to read, easy to use.
- Capable of handling database-driven e-commerce Web sites
- Improved with every release (supported)
- Converts static web pages to dynamic web pages.
- Open source means it's free to use @home.



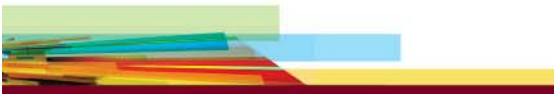
Server Side Scripting

- All processing takes place before a Web page is sent to the browser. As a result, the code remains hidden from users.
- Browser independent. That is, the browser used does not have any impact on the processing of code because all the processing takes place on the server.
- To use server-side scripting you might require a high-configuration server.
- Running complex tasks on the server side might require a lot of processing speed and also possibly slow down the Web site.
- Few server side scripting languages are PHP, Python, JSP, Perl



Client Side Scripting (JavaScript, VBScript, Jscript)

- Reduces the load on a server
- Reduces network traffic
- The scope of client-side scripting is limited. You cannot use client-side scripting languages to perform complex tasks such as interacting with a database server or accessing local files and directories.
- The client-side scripting code that you use is visible to users.
- We will use Javascript in this course to show PHP and AJAX.



For thought

Server vs. Client scripting

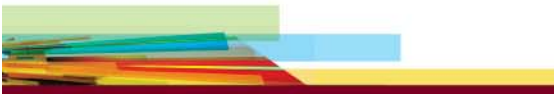
As a group of 2, list out the following four lists, as participation activity.

Server

Advantages	Disadvantages
1,2,3	1,2,3

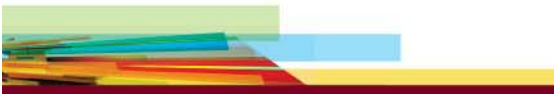
Client

Advantages	Disadvantages
1,2,3	1,2,3



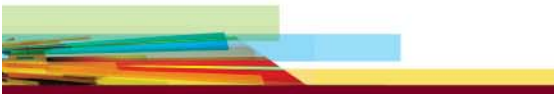
Open Source Software (OSS)

- Open-source software (OSS) is computer software with its source code made available with a license in which the copyright holder provides the rights to study, change, and distribute the software to anyone and for any purpose.
- Open-source software may be developed in a collaborative public manner.



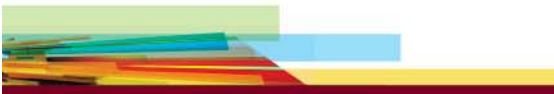
PHP and PHP2

- Created by Rasmus Lerdof in 1995 in order to augment his home page
- Original name “Personal Home Page”
- Newer name “PHP Hypertext Processor”
- PHP1 was used personally by the creator, never released.
- **PHP2 was the original public project**



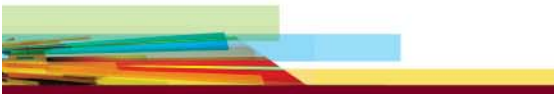
PHP 3

- Released in 1997
- Ran on Apache only
- Andy Gutmans and Zeev Suraski launched the PHP 3 project
- Modular approach
- Powered 10% of all sites on the internet at the time.



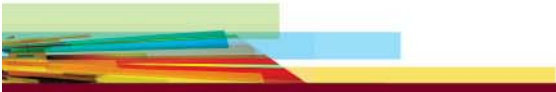
PHP 4

- Built on the Zend engine (new backbone)
- Zend added:
 - Support for more servers, like IIS.
 - Avoided memory leaks (better memory management).
 - Improved efficiency and performance for large scale applications.
- Added CLASSES. (True object orientation - OO)

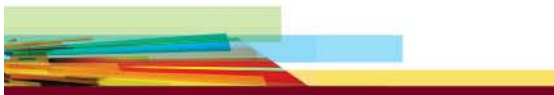
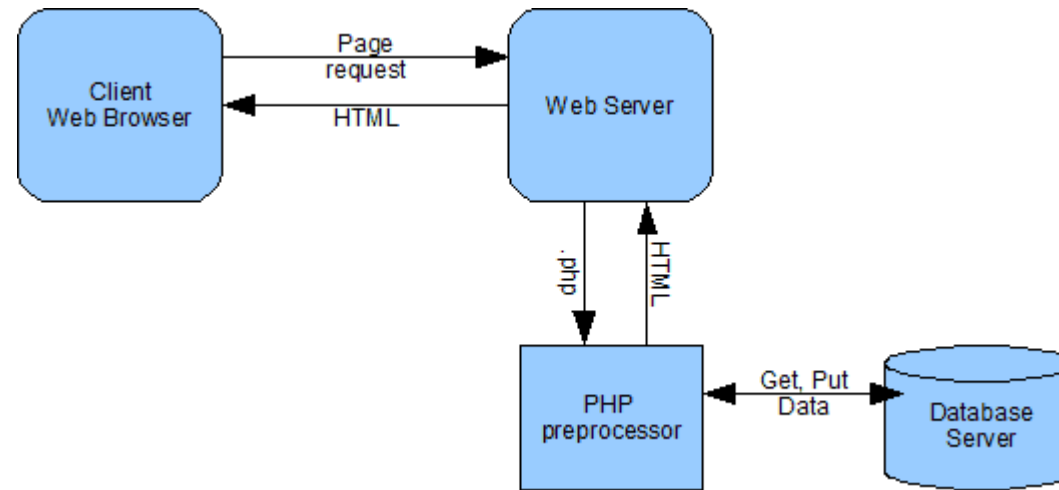


PHP 5

- Improved OO
- Error handling and exception management
- XML Document handling
- SQLite support (embedded db)



Simple Flow



Using XAMPP

- <https://www.apachefriends.org/download.html>

 XAMPP for **Windows** 5.5.30, 5.6.15 & 7.0.1

Version		Checksum			Size
5.5.30 / PHP 5.5.30	What's Included?	md5	sha1	Download (32 bit)	106 Mb
5.6.15 / PHP 5.6.15	What's Included?	md5	sha1	Download (32 bit)	108 Mb
7.0.1 / PHP 7.0.1	What's Included?	md5	sha1	Download (32 bit)	116 Mb

[Requirements](#) [Add-ons](#) [More Downloads »](#)

Windows XP or 2003 are not supported. You can download a compatible version of XAMPP for these platforms [here](#).

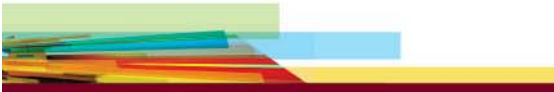
... for Mac



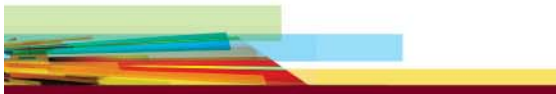
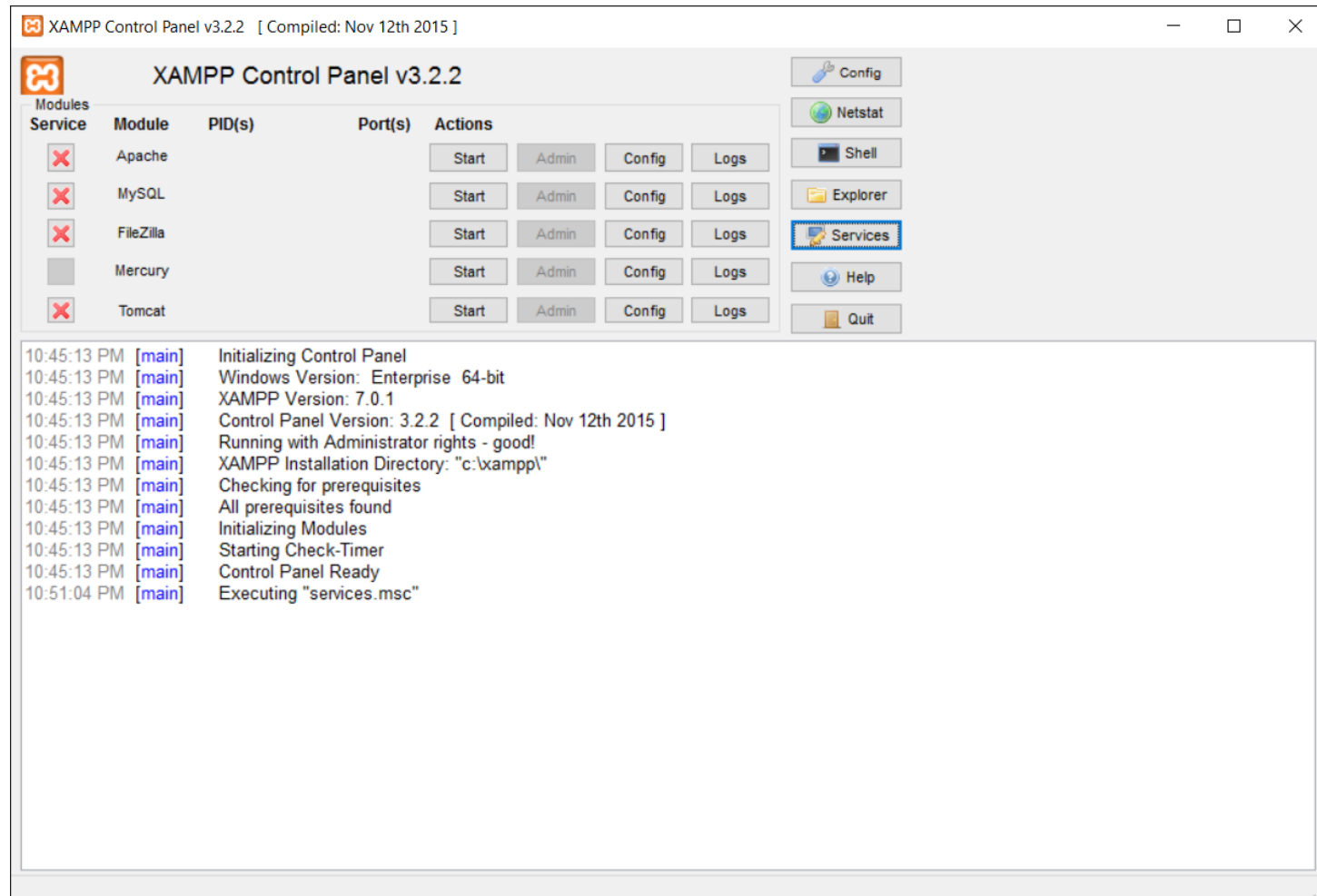
XAMPP for **OS X** 5.5.30, 5.6.15 & 7.0.1

Version		Checksum		Size	
5.5.30 / PHP 5.5.30	What's Included?	md5	sha1	Download (64 bit)	129 Mb
5.6.15 / PHP 5.6.15	What's Included?	md5	sha1	Download (64 bit)	130 Mb
7.0.1 / PHP 7.0.1	What's Included?	md5	sha1	Download (64 bit)	127 Mb

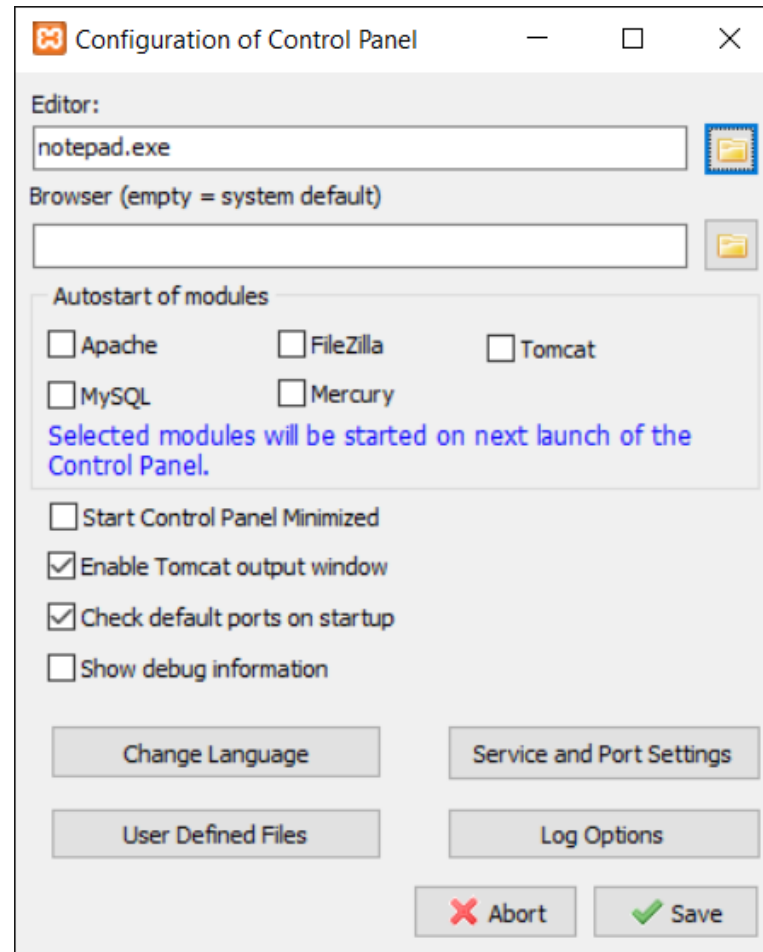
[Requirements](#) [Add-ons](#) [More Downloads »](#)



XAMPP Control Center



Control Center - Configuration



Configuration of Control Panel

Editor:
notepad.exe

Browser (empty = system default)

Autostart of modules

☐ Apache ☐ FileZilla ☐ Tomcat
☐ MySQL ☐ Mercury

Selected modules will be started on next launch of the Control Panel.

☐ Start Control Panel Minimized
☒ Enable Tomcat output window
☒ Check default ports on startup
☐ Show debug information

Change Language Service and Port Settings

User Defined Files Log Options

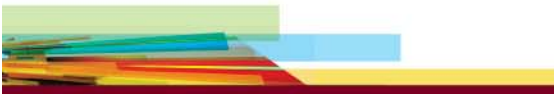
Abort Save

Netbeans

<https://netbeans.org/downloads/>

Supported technologies *	Java SE	Java EE	HTML5/JavaScript	PHP	C/C++	All
NetBeans Platform SDK	•	•				•
Java SE	•	•				•
Java FX	•	•				•
Java EE		•				•
Java ME						•
HTML5/JavaScript		•	•	•		•
PHP			•	•		•
C/C++					•	•
Groovy						•
Java Card™ 3 Connected						•
Bundled servers						
GlassFish Server Open Source Edition 4.1.1		•				•
Apache Tomcat 8.0.27		•				•

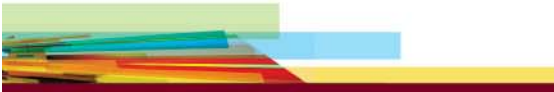
Download	Download	Download x86	Download x86	Download x86	Download
		Download x64	Download x64	Download x64	
Free, 95 MB	Free, 197 MB	Free, 108 - 112 MB	Free, 108 - 112 MB	Free, 107 - 110 MB	Free, 221 MB



Netbeans Basic Setup

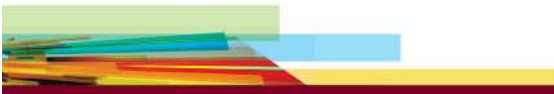


NetBeans



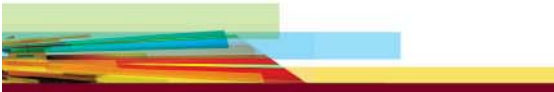
How to embed PHP

START TAGS	END TAGS
<?	?>
<?php	?>
<script language="php">	</script>

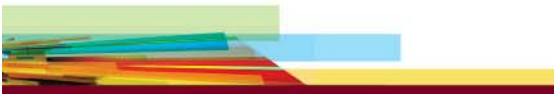


Commenting Code

- `//` Comment
- `/*` Block of comments `*/`

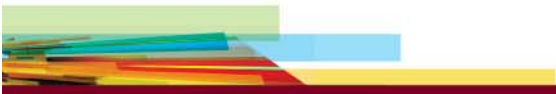


PHP vs. HTML



Basic Functions

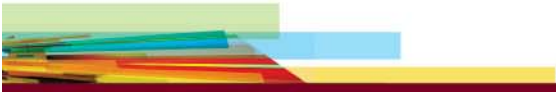
- `echo`
 - Output data to the current output method.
- `time()`
 - Returns the time.



PhpInfo

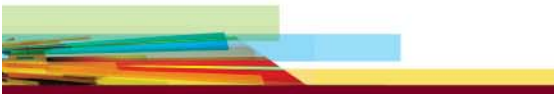
- phpinfo()
 - Returns the information about the server
 - Blocked on some servers

```
bool phpinfo ([ int $what = INFO_ALL ] )
```



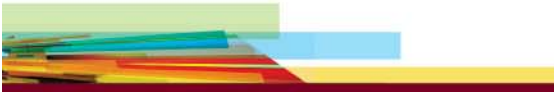
PHP_INFO constants

INFO_GENERAL	1	The configuration line, <i>php.ini</i> location, build date, Web Server, System and more.
INFO_CREDITS	2	PHP Credits. See also phpcredits() .
INFO_CONFIGURATION	4	Current Local and Master values for PHP directives. See also ini_get() .
INFO_MODULES	8	Loaded modules and their respective settings. See also get_loaded_extensions() .
INFO_ENVIRONMENT	16	Environment Variable information that's also available in \$ENV .
INFO_VARIABLES	32	Shows all predefined variables from EGPCS (Environment, GET, POST, Cookie, Server).
INFO_LICENSE	64	PHP License information.
INFO_ALL	-1	Shows all of the above.



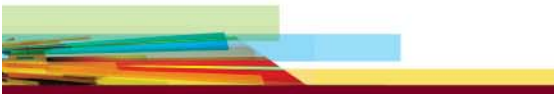
Hands On

- Create a page which displays the information about the current PHP server.



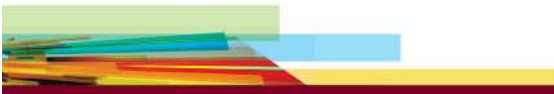
Variables

- Starts with the \$ sign, followed by the name of the variable
- Must start with a letter or the underscore character
- Cannot start with a number
- Can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Case sensitive (\$y and \$Y are two different variables)



Constants

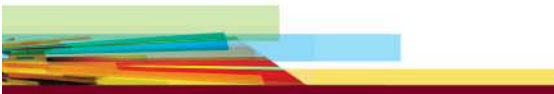
- A constant is an identifier (name) for a simple value. The value cannot be changed during the script.
- Name starts with a letter or underscore (no \$ sign before the constant name)
- As a recommended practice, constant identifiers are UPPERCASE
- Can be case sensitive or not depending on the way it's defined.
 - `define ("COMPNAME", "IBM"); // Case sensitive`
 - `define ("COMPNAME", "IBM", true); // Case insensitive`

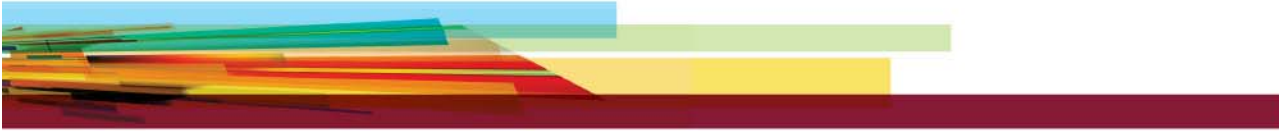


Display Raw Values (Debugging)

var_dump(), **print_r()**

- The **var_dump** function displays structured information about variables/expressions including its type and value. Arrays are explored recursively with values indented to show structure. It also shows which array values and object properties are references.
- **print_r()** displays information about a variable in a way that's readable by humans. array values will be presented in a format that shows keys and elements. Similar notation is used for objects.



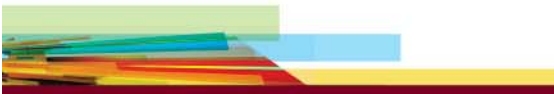


Strings and String Related Functions

ECHO

Format

```
echo "string";  
echo 'string';  
echo $a;
```



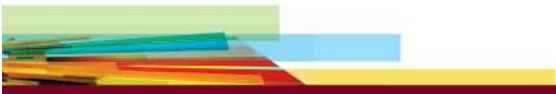
Concatination

To concatenate text, use the “.” period.

- Comma may be used.

```
echo "Abso" . "lutely";
```

```
Echo "Abso", "lutely";
```



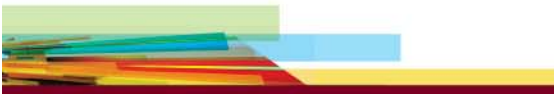
ECHO quote Format

- Single quotes: Literal value
- Double quotes: parsed value

```
$a = 2;
```

```
echo "2 + $a"; ?
```

```
echo '2 + $a'; ?
```



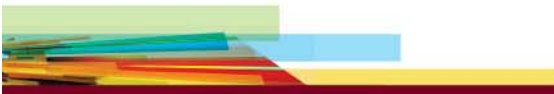
Concatination vs. Inline

Example

`$a = 2; $b = 3;`

GOAL: “The result of $2 + 3$ is 5.”

What are two methods of achieving this goal?

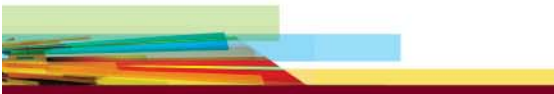


Be careful with echo + arithmetic

- It's tricky to display text and arithmetic on the same echo. Follow the order of operations.

```
echo "The result of 2 + 3 is : " . 2 + 3 ;
```

```
echo "the result of 2 + 3 is : " . (2 + 3) ;
```



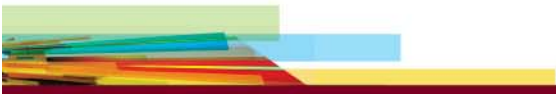
Echo output properties.

- Echo depends on the output method. PHP can be run via command line or via HTML browser.
- Echo outputs raw data.

```
echo "This is <b>bold</b>";
```

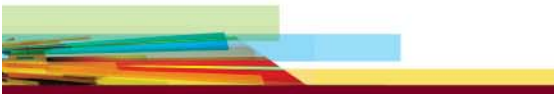
```
echo "Line 1<br />Line 2";
```

```
echo "Line 1 \n Line 2";
```



Special HTML characters

- `echo htmlspecialchars("abc & < >");`
- `echo htmlentities("<label> AAAAA");`
- `echo "<label> AAAAA";`



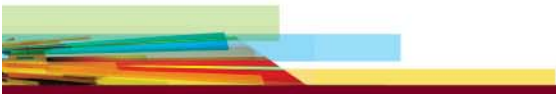
Escape Quotes

Using backslashes.

- `"From time to \"time\"";`

Creatively using single quotes.

- `'From time to "time";`



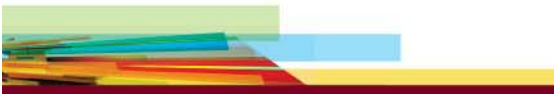
Hands on

- Create a “Hello world” program in PHP. Hello should be on top, and world should be on the bottom.
 - Hello
 - World
- Hint – In HTML the `
` element means newline.



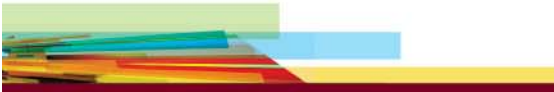
printf

- The printf() function writes a formatted string.
- Note: sprintf() writes to a variable (return value).
- The arg1, arg2, ++ parameters will be inserted at percent (%) signs in the main string. This function works "step-by-step". At the first % sign, arg1 is inserted, at the second % sign, arg2 is inserted, etc.
- Note: If there are more % signs than arguments, you must use placeholders. A placeholder is inserted after the % sign, and consists of the argument-number and "\\$". See example two.



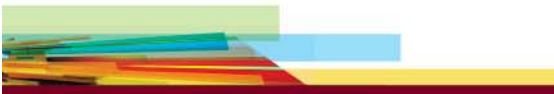
Printf Parameters

- %% - Returns a percent sign
- %b - Binary number
- %c - The character according to the ASCII value
- %d - Signed decimal number (negative, zero or positive)
- %e - Scientific notation using a lowercase (e.g. 1.2e+2)
- %E - Scientific notation using a uppercase (e.g. 1.2E+2)
- %u - Unsigned decimal number (equal to or greater than zero)
- %f - Floating-point number (local settings aware)
- %F - Floating-point number (not local settings aware)
- %g - shorter of %e and %f
- %G - shorter of %E and %f
- %o - Octal number
- %s - String
- %x - Hexadecimal number (lowercase letters)
- %X - Hexadecimal number (uppercase letters)



Printf Parameters Ctd...

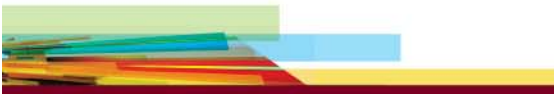
- Additional format values. These are placed between the % and the letter (example %.2f):
- + (Forces both + and - in front of numbers. By default, only negative numbers are marked)
- ' (Specifies what to use as padding. Default is space. Must be used together with the width specifier. Example: %'x20s (this uses "x" as padding)
- - (Left-justifies the variable value)
- [0-9] (Specifies the minimum width held of to the variable value)
- .[0-9] (Specifies the number of decimal digits or maximum string length)



```
$num1 = 123456789;  
$num2 = -123456789;  
$char = 50; // The ASCII Character 50 is 2
```

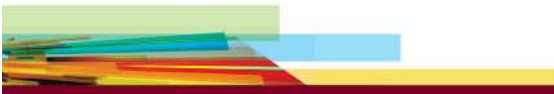
```
// Note: "%%" returns a percent sign
```

```
// Binary number  
printf("%%b = %b <br>", $num1);  
// The ASCII Character  
printf("%%c = %c <br>", $char);  
// Signed decimal number  
printf("%%d = %d <br>", $num2);
```



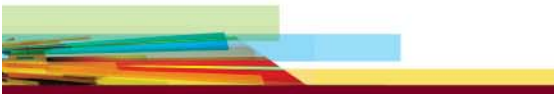
ltrim, rtrim, trim (*string*)

Trims whitespace.



count_chars(*string*,*mode*)

- 0 - an array with the byte-value as key and the frequency of every byte as value.
- 1 - same as 0 but only byte-values with a frequency greater than zero are listed.
- 2 - same as 0 but only byte-values with a frequency equal to zero are listed.
- 3 - a string containing all unique characters is returned.
- 4 - a string containing all not used characters is returned.



htmlentities()

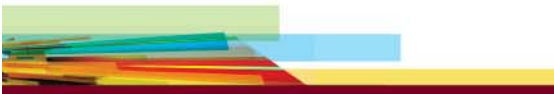
- Convert all applicable characters to HTML entities

number_format(*number,decimals,decimalpoint,separator*)

- Format a number with grouped thousands
 - `echo number_format("1000000",2). "
";`
`echo number_format("1000000",2," ",".");`

Print()

- Same as echo.



similar_text(*string1*,*string2*,*percent*)

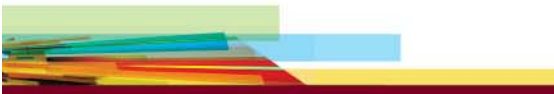
- Calculate the similarity between two strings
- Percent = percent similar, return value = number of matching values.

strcmp(*string1*,*string2*)

- Returns < 0 if str1 is less than str2; > 0 if str1 is greater than str2, and 0 if they are equal.

sscanf()

- ```
$str = "age:30 weight:60kg";
sscanf($str,"age:%d weight:%dkg",$age,$weight);
// show types and values
var_dump($age,$weight);
```



**str\_ireplace(*find,replace,string,count*)**

**str\_replace(*find,replace,string,count*)**

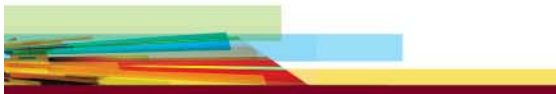
- Replace all occurrences of the search string with the replacement string

**strchr(*string,search,before\_search*) : strstr()**

- Returns part of haystack string starting from and including the first occurrence of needle to the end of haystack.

**strpos(*string,find,start*)**

- Returns the position of where the needle exists relative to the beginning of the haystack string (independent of offset). Also note that string positions start at 0, and not 1.



## **substr(string,start,length)**

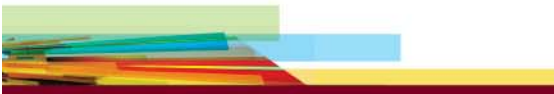
- Returns the portion of string specified by the start and length parameters.

## **substr\_replace(string,replacement,start,length)**

- substr\_replace() replaces a copy of string delimited by the start and (optionally) length parameters with the string given in replacement.

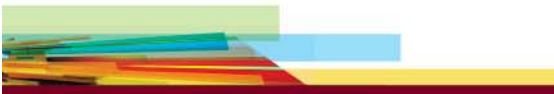
## **strip\_tags(string,allow)**

- strip\_tags(\$input, '<br>');
- Note: This allows <br> and also </br> do not use the closing tag here.

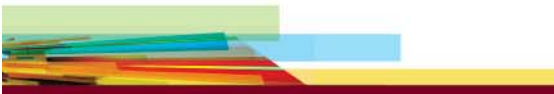


## **wordwrap(*string,width,break,cut*)**

- Width : The number of characters at which the string will be wrapped.
- Break : The line is broken using the optional break parameter.
- Cut : If the cut is set to TRUE, the string is always wrapped at or before the specified width. So if you have a word that is larger than the given width, it is broken apart. (See second example). When FALSE the function does not split the word even if the width is smaller than the word width.



**str\_repeat(*string*,*repeat*)**  
**str\_shuffle(*string*)**  
**strrev(*string*)**  
**strtolower(*string*)**  
**strtoupper(*string*)**  
**ucfirst(*string*)**

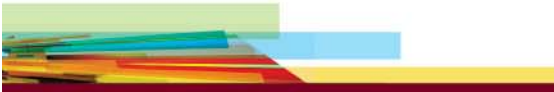


# Money Format

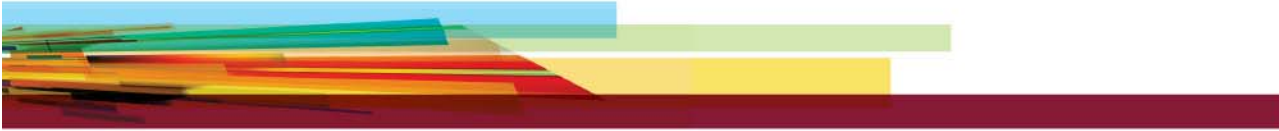
## money\_format()

- Warning, this doesn't exist in Windows implementations of PHP.
- Replace with:  

```
$price = number_format($price, 2);
echo "$" . $price;
```







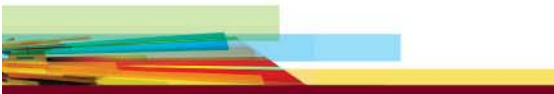
# Date Functions

# Setting the Time Zone

```
date_default_timezone_set('America/New_York');
```

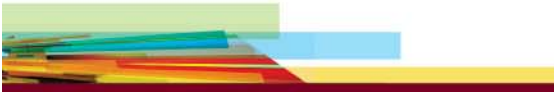
OR

You can modify the php.ini file to permanently change your time zone.



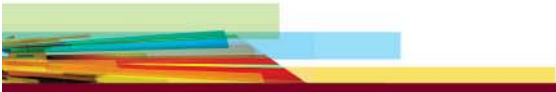
# “date” function

- `$today = date("F j, Y, g:i a");` // March 10, 2001, 5:16 pm
- `$today = date("m.d.y");` // 03.10.01
- `$today = date("j, n, Y");` // 10, 3, 2001
- `$today = date("Ymd");` // 20010310
- `$today = date('h-i-s, j-m-y, it is w Day');` // 05-16-18, 10-03-01, 1631 1618 6 Satpm01
- `$today = date('\i\t \i\s \t\h\e jS \d\a\y.');` // it is the 10th day.
- `$today = date("D M j G:i:s T Y");` // Sat Mar 10 17:16:18 MST 2001
- `$today = date('H:m:s \m \i\s\ \m\o\n\t\h');` // 17:03:18 m is month
- `$today = date("H:i:s");` // 17:16:18
- `$today = date("Y-m-d H:i:s");` // 2001-03-10 17:16:18  
(the MySQL DATETIME format)



# Other date functions of importance

- `date_create(time,timezone);`
- `date_add(object,interval);`
- `date_sub(object,interval);`
- `date_diff(datetime1,datetime2,absolute);`
- `date_format(object,format);` (See guide)



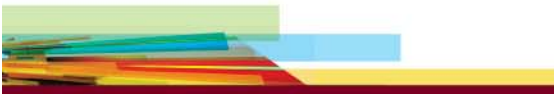
# Exercise 3

- Create a page that displays the time, under the time display the date.

Eg:

“The time is : 5:45pm”

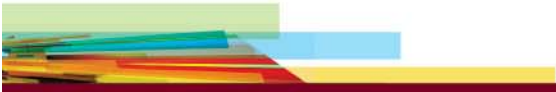
“The date is : January 14, 2015”



# Basic Arithmetic

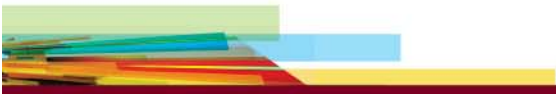
`$a = 10;`

- `echo $a + 1;`
- `echo $a - 1;`
- `echo $a * 1;`
- `echo $a / 0; *`
- Modulus : `%`



# Operators

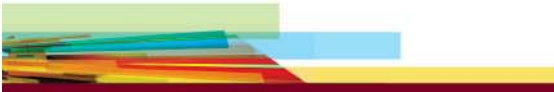
- Incremental/Decremental
  - ++
  - --
  - Position of increment operator?
- String (already covered)
  - .
  - " " Quotes
  - ' ' Quotes



# If / then / else

- The if statement is used to execute some code only if a specified condition is true.

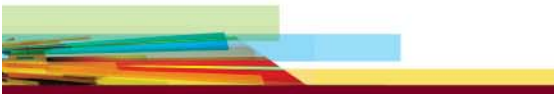
```
if (condition)
{
 condition is true;
}
else
{
 condition is false;
}
```





■ ■ ■

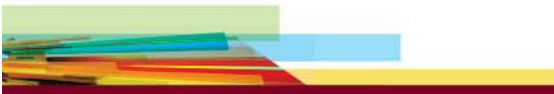
```
if (condition)
{
 code to be exec if condition is true;
}
elseif (condition)
{
 code to be exec if condition is true;
}
else
{
 code to be exec if condition is false;
}
```



# Ternary IF

```
$x = 2;
echo $x == 2 ? "2" : "Something else";
```

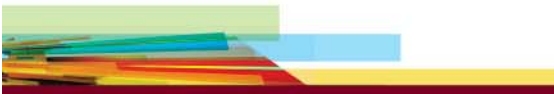
TODO: Try to write a ternary IF to determine if a value is even or odd.



# for

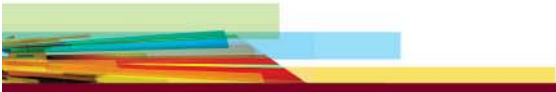
- Execute block of code repetitively based on a condition.

```
for (init; test; increment counter)
{
 code to be executed;
}
```



# FOR Loop

```
for ($x = 0; $x < 10; $x = $x + 1)
{
 echo 'Hello ' . $x . '</br>';
}
```



# Hands-on

Display this programmatically

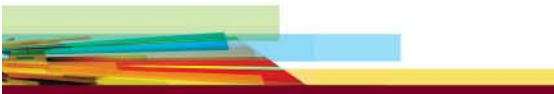
\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*



# Hands On

Display the reverse as well.

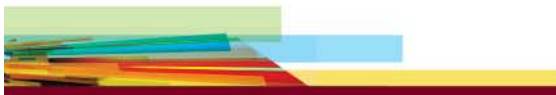
\*\*\*\*\*

\*\*\*\*

\*\*\*

\*\*

\*

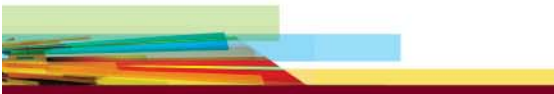


# Hands-On

This hands-on also tests for redundant programming and style. It's a popular interview question. 🤖

Write a program that counts from 1 to 100.

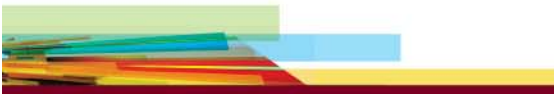
- When the value is divisible by 3
  - Output “Fizz”
- When the value is divisible by 5,
  - Output “Buzz”
- When the value is divisible by BOTH,
  - Output “FizzBuzz” (eg 15)



# While

Executes a block of code as long as the specified condition is true.

```
while (condition is true)
{
 code to be executed;
}
```





# Group Exercise 1

Please write a program that flips a coin (potentially forever), and detects when 3 heads or tails appears in a row.

- Show coin tosses (H,T, H, etc...)
- Display how many flips it took to get 3.

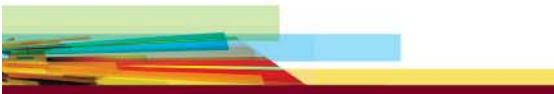
Sample run:

H, T, H, H, H

It took **5** roles to achieve 3 heads in a row.

Additionally, put a safeguard to limit tries to 20 turns.

**Keep an eye out for style (no magic numbers!)**

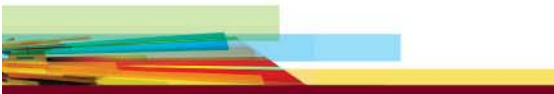


# Do / While

Always execute the block of code once

Then check the condition, and repeat the loop while the specified condition is true.

```
do
{
 Block of code to be executed;
}
while (condition is true);
```



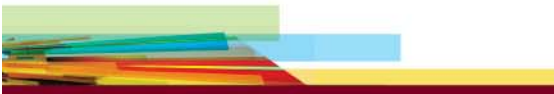
# Exercise

Given the following while loop:

```
$test = 0;
while ($test < 10)
{
 echo "$test is less than 10.
";
 $test++;
}
```

Write an equivalent set of code using do-while.

*For the new code, what can be a problem?*



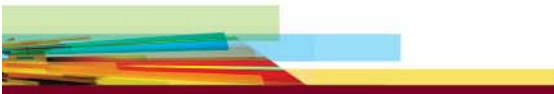
# break / continue

## **BREAK**

- Breaks out of a loop prematurely. End the loop.

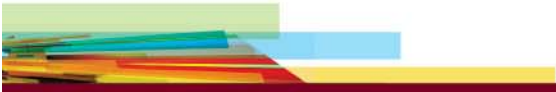
## **CONTINUE**

- Break out of the current loop, but continue with next loops.



# Continue

```
for($i=0; $i<10; $i++)
{
 if($i == 5)
 {
 echo "Reached five
";
 continue;
 }
 echo $i . "
";
}
```



# Hands-On

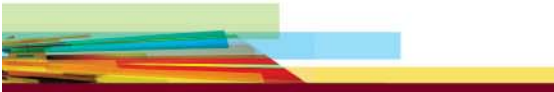
Display this using a continue statement;

\*

\*\*

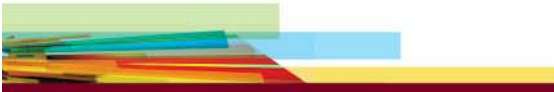
\*\*\*\*

\*\*\*\*\*



# Break

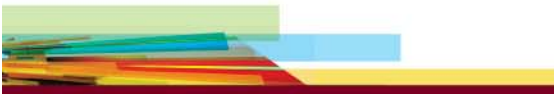
```
for($i=0; $i<10; $i++)
{
 if($i == 5)
 {
 echo "Reached the end
";
 break;
 }
 echo $i . "
";
}
```



# Switch

- Switch executes the appropriate block of code, based on the input variable.

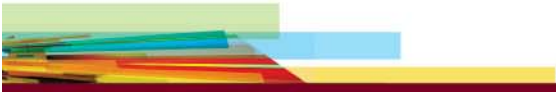
```
switch (n)
{
case label1:
 code to be executed if n=label1;
 break;
case label2:
 code to be executed if n=label2;
 break;
default:
 code to be executed if n is different from all
 labels;
}
```





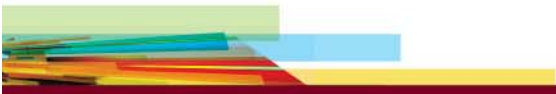
# Switch with enumerations.

```
switch ($i) {
 case 20:
 logic;
 break;
 case 30:
 logic;
 break;
 case 40:
 logic;
 break;
}
```



# Exercise

```
if ($i == 0)
{
 echo "i equals 0";
}
elseif ($i == 1)
{
 echo "i equals 1";
}
elseif ($i == 2)
{
 echo "i equals 2";
}
```



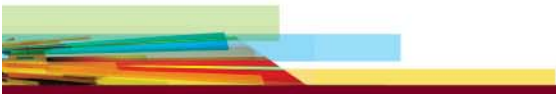
# Exercise (Switch)

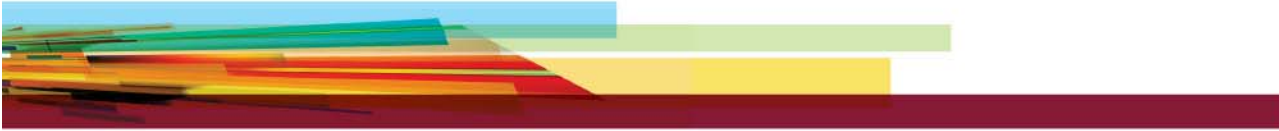
- Create a function that takes in a string. The string will be a province code such as “QC”, “ON”, etc.
- In the function create a switch statement that returns the full province name using the abbreviation (QC = ‘Quebec’).
- In the switch if something other than the ten provinces is given, return the string “Unknown”.
- In your main program, call the function 3 times, one with QC, one with ON, and one with XY.



# Switch with Ranges

```
$t = "2000";
switch (true) {
 case ($t < "1000"):
 alert("t is less than 1000");
 break
 case ($t < "1801"):
 alert("t is less than 1801");
 break
 default:
 alert("t is greater than 1800")
}
```





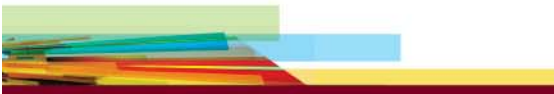
# Functions

# Functions

- A block of statements that can be used repeatedly in a program.
- **Will not execute immediately when a page loads.**
- Executed by a call to the function.

```
// Define function
function functionName()
{
 code to be executed;
}
```

```
// Run fcn
functionName();
```



# Functions

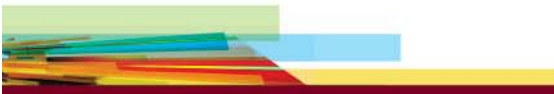
- Arguments

```
<?php
function familyName($fname,$year)
{
echo "$fname was born in $year
";
}

familyName("Smith","1975");
familyName("Wood","1972");
?>
```

- Defaults

```
function familyName($fname,$year = 2000)
```

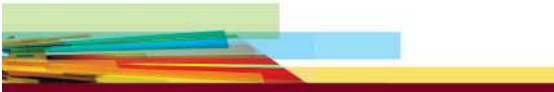


# Functions

- Returning values

```
<?php
function sum($x,$y)
{
 $z=$x+$y;
 return $z;
}
```

```
echo "5 + 10 = " . sum(5,10) . "
";
echo "7 + 13 = " . sum(7,13) . "
";
echo "2 + 4 = " . sum(2,4);
?>
```



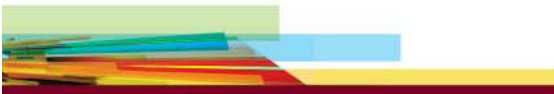


# Passing by Reference

```
function foo(&$var)
```

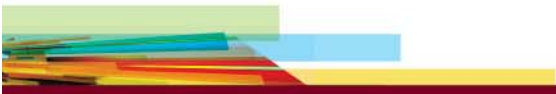
```
function &foo($var)
```

- Definition: Does not create a copy of the function before it enters the function.
- Note: These are not like C++ pointers, you cannot perform arithmetic on them.



# Hands-On

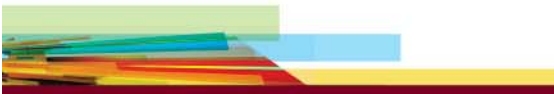
- Write a proof of pass-by reference and pass-by value in PHP.



# Exercise (functions – 1)

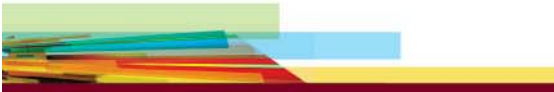
Create a function called “evenOrOdd” and have it accept 1 parameter (a number), and have the function return the word “even” or “odd” depending on the number given.

*Hint: The modulus sign (%) is the remainder of a number when divided. Eg:  $7 \% 2 = 1$  (the remainder).*



# Exercise (functions – 2)

- Create a function called “multiplyNum”, which takes 2 parameters, and returns the result of the multiplication of them.



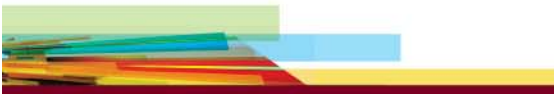
# Anonymous functions

Anonymous functions are created by defining a function as you would any other, but without a name.

In order to be able to later call that function, the unnamed definition needs to be assigned to a variable

```
$hello = function($who)
{
 echo "<p>Hello, $who</p>";
};
```

```
echo $hello("Brendan");
```

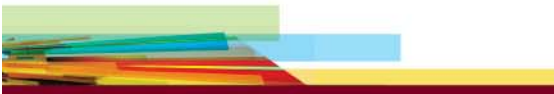


# Major Benefit of Anon (or inline) function(s).

- We can pass a function as a parameter to another function.

```
function test($num, $func)
{
 while ($num)
 {$func($num); $num--;}
}
```

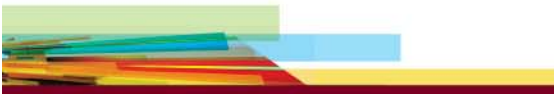
```
$anon = function ()
{ echo "hello world<br\>"; };
```



# Write the following program

We need a function that accepts another function as a parameter.

- Function 1 : `filterString(&$string, $filter)`
- Inline 1 (filter): `noCaps`
- Inline 2 (filter): `allCaps`



# Variable Scope

## **local**

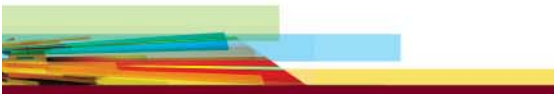
- Variables defined within a function body { } will be local to that scope.

## **global**

- Defined outside all functions. Available to the whole program.

## **static**

- Keeps a local variable “alive” even when it goes out of scope.



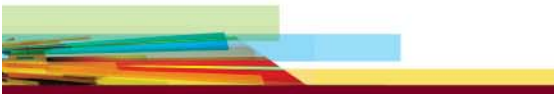


# Scope - Example 1

```
<?php
$a = 1; /* global scope */

function test()
{
 echo $a; /* reference to local scope variable */
}

test();
?>
```



# Scope – Example 2

```
$a = 1;
```

```
$b = 2;
```

```
function Sum()
```

```
{
```

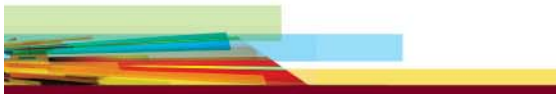
```
 global $a, $b;
```

```
 $b = $a + $b;
```

```
}
```

```
Sum();
```

```
echo $b;
```

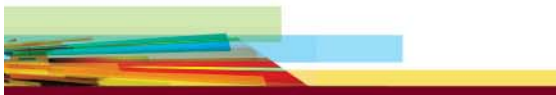


# \$GLOBALS superGlobal

```
$a = 1;
$b = 2;

function Sum()
{
 $GLOBALS['b'] = $GLOBALS['a'] + $GLOBALS['b'];
}

Sum();
echo $b;
```

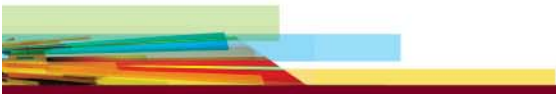


# Include/Require

- Includes another file into the current html page.
- Reusability oriented
- For security, make sure you're in control of the file you include.
- vars.php

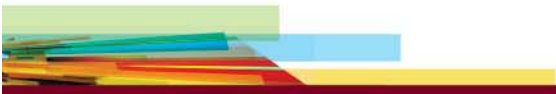
```
<?php $color = 'green'; $fruit = 'apple';
?>
```
- test.php

```
<?php include 'vars.php';
echo "A $color $fruit";
?>
```



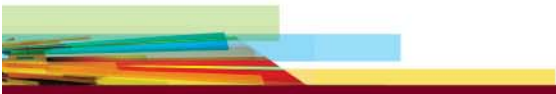
# Include/Require

- **require** will produce a fatal error (E\_COMPILE\_ERROR) and stop the script
- **include** will only produce a warning (E\_WARNING) and the script will continue



# Include\_Once

- This will make sure that the file is not included more than once in the file.
- It's return type is 0 or 1 depending if the file was included or not.



# Exercise (Include)

- Write an include file containing 2 variables. One is the company name, the other is an array of employee names.

Eg: “Microsoft Corp”.

array(“Bill”, “Steve”, “Joe”)

- Include this file into your main PHP program using the INCLUDE keyword.
- Write some logic to display the company name, and all the employee names.

