Lecture 17: More PHP Fundamentals, Output, Control Structures, Functions
PHP Constants

To define a constant use `define`.  

Example:
```
define("VALUE", 5);
$a = VALUE + 3;
define("SCHOOL", "UCLA");
print(SCHOOL);
```

Note: constants do not start with `$`.  

PHP Output

**echo**
- Is a native PHP construct
- does not return a value
- can have multiple string arguments
- parentheses () are optional
- only a single argument if using ()

**print**
- Is also a native PHP construct and not a true function.
- returns integer value 1 always
- can only have 1 string argument
- parentheses () are optional
PHP output examples

```php
echo "<p>Cool</p>", "<h1>Hi</h1>";
echo("PIC40A");
echo "The sum is: $sum";

$s = "hello";
echo $s;
echo "$s";
echo ($s);
echo ("$s");
```
PHP output examples

$s = "hello";
$t = "world";
print("Internet Programming Rocks");
print(5); #PHP coerces 5 to string "5"

print $s;
print ($s);
print ("$s");

print ($s,$t) // Does not work!

print $s."".$t;
Some PHP Predefined Functions

• \texttt{isset($x)}
  - TRUE if variable $x$ is not NULL, FALSE otherwise
  - Often used when processing form input

• \texttt{gettype($x)}
  - returns the type of $x$
PHP Date/Time Functions

time()

Returns current UNIX timestamp (Number of seconds elapsed since Jan 1, 1970)

date($formatString, $intTimestamp)

Returns a string formatted according to the given format string using the given integer timestamp or the value of time() if no timestamp is given.

mktime($hr,$min,$sec,$m,$d,$y)

Gets the UNIX timestamp for the given date
Examples

```php
<?php
$timestamp = time(); // Get current timestamp

$timestamp += 3600 * 24; // Move timestamp forward by 24 hrs

$year = date("Y", $timestamp);
$month = date("m", $timestamp);
$day = date("d", $timestamp);

$hour = date("h", $timestamp);
$minute = date("i", $timestamp);
$second = date("s", $timestamp);

echo "Tomorrow is $month/$day/$year<br/>";
echo "Time 24 hours from now is $hour:$minute:$second<br/>";
?>```
Good PHP reference


If you google for PHP date to learn more about the date function the first thing that comes up is the page for this manual.
PHP control structures

• Selection
  
  if, if-else, if-elseif-else

• Loop
  
  while, for, do-while, foreach
Example

```php
if($x < $y) {
    print("$x < $y");
}
elseif($x == $y) {
    print("$x == $y");
}
else {
    print("$x > $y");
}
```
Remark

```php
elseif($x == $y) {
    print("$x == $y");
}
```

When using the conventional form of control expressions the above is the same as

```php
else if($x == $y) {
    print("$x == $y");
}
```
Example:

```php
while($x < 10) {
    $x++;
    print "\$x<br/>";
}
```
Example

```php
<?php
$year = 2011;
$ts = mktime(0,0,0,1,1,$year);

for($day = 1; $day < 32; $day++)
{
    echo date('n/j/Y', $ts), "was a ", date('l', $ts), "<br/>";
    $ts += 24*3600;
}
?>
```
Alternative syntax for control structures

PHP has an alternative syntax for some of its control structures.

The alternate syntax is to change the opening brace to a colon and
the closing brace to endif;, endwhile;, endfor; or
endforeach;

Example:

while($x< 10):
    $x++; 
    print($x); 
endwhile;
PHP functions

Syntax:

```php
function name($param1,$param2,..){
    // write function body here
}
```
Notes on PHP functions

Function overloading is not allowed

Functions cannot be redefined

Function names are not case sensitive

Use of `return` statement is optional (can return no value)
function display($formal_parameter){
    print($formal_parameter);
}

$formal_parameter = "message";
display($actual_parameter);

**Formal parameters** are in the function prototype and must be variable names

**Actual parameters** are in the function call and can be any expression

The number of formal parameters need not match the number of actual parameters

Excess formal parameters are NULL

Excess actual parameters are ignored
Parameter Passing

By default, PHP passes parameters by value.

Example:

```php
function increment($num) {
    // a local copy of the $num defined below
    $num++;
    print('$num in function: ' . "$num" );
}

$num = 4;
increment($num);
print('$num after function: ' . "$num" );

# Output
# $num in function: 5
# $num after function: 4
```
Passing a Parameter by Reference

1. place an & before a formal parameter in the function prototype

or

2. place an & before an actual parameter in a function call.

(Both ways are identical.)
Pass by Reference Example

function increment(&$num) {
    // this $num is the actual parameter below
    $num++;
    print('$num in function: ' . "$num");
}

$num = 4;
increment($num);
print('$num after function: ' . "$num");

# Output
# $num in function: 5
# $num after function: 5
function increment($num) {
    // this $num is the actual parameter below
    $num++;  
    print('$num in function: ' . "$num"DEFINED);
}

$num = 4;
increment(&$num);
print('$num after function: ' . "$num"DEFINED);

# Output
# $num in function: 5
# $num after function: 5
PHP Variable Scope

A variable defined in a function call:
• Has local scope
• Is visible only inside the function body
• Masks out any variable defined outside the function with the same name
To get access to a variable defined outside the function body, use a `global` declaration.

Example:
```php
$product = 0;
function multiply($a, $b)
{
    // this is the $product defined above
    global $product;
    $product = $a*$b;
    return $product;
}
```
(static variables)

To make a variable "history sensitive" declare it with the reserved word static.

Example

```php
function call_counter()
{
    // assigned 0 only the first time counter is called
    static $num_times_called = 0;
    $num_times_called++;
    print("This function has been called
           $num_times_called times");
}
```

$num_times_called lives even after the function call_counter terminates and keeps track of how many times it has been called, even if it was called from several different places
There is no "real" debugger.

What you can do is turn on error reporting and the PHP interpreter will tell you what line it thinks it has encountered an error on.

Replace the top line in your code with:

```
#!/usr/local/bin/php -d display_errors=STDOUT
```