PHP for Server-Side Programming

By: Prof P Sai Prasad

Server-Side web programming

- Server-side pages are programs written using one of many web programming languages/frameworks
  - examples: PHP, Java/JSP, Ruby on Rails, ASP.NET, Python, Perl
- The web server contains software that allows it to run those programs and send back their output as responses to web requests
- each language/framework has its pros and cons
  - we use PHP for server-side programming in this Course

What is PHP?

- PHP stands for "PHP Hypertext Preprocessor"
- a server-side scripting language
- Used to make web pages dynamic:
  - provide different content depending on context
  - interface with other services: database, e-mail, etc
  - authenticate users
  - process form information
- PHP code can be embedded in HTML code

Why PHP?

There are many other options for server-side languages: Ruby on Rails, JSP, ASP.NET, etc. Why choose PHP?

- free and open source: anyone can run a PHP-enabled server free of charge
- compatible: supported by most popular web servers
- simple: lots of built-in functionality; familiar syntax
- easy to use and learn

Hello, World!

The following contents could go into a file `hello.php`:

```php
<?php
print "Hello, world!";
?>
```

PHP

```
Hello, world!
```

output

- a block or file of PHP code begins with **<?php** and ends with **?>**
- PHP statements, function declarations, etc. appear between these endpoints

Required Setup for PHP

OS: Windows/Linux/Mac

Windows: WAMP (Windows/Apache/Mysql/PHP)

Linux: LAMP (Linux/Apache/Mysql/PHP)

XAMPP (Crossplatform(Windows/Linux/Mac), Apache/Mysql/PHP,Perl)

Note: XAMPP is Recommended.

Normal Editor(notepad/gedit/Vi/Vim) is required to write the Program

Apache is a Webserver.

Mysql is a Relational database.

Installing XAMPP in Linux

Installation

Open Terminal

- $ Su (Login as SuperUser)
- << Give Password >>

- $ chmod +x   xamppinstaller.run

- $ ./xamppinstaller.run

Launch (Start the Services)

- Open Terminal

- $ Su (Login as SuperUser)
- << Give Password >>

- $ /opt/lampp/lampp start

Starting XAMPP for Linux 5.6.24-1...

XAMPP: Starting Apache...ok.

XAMPP: Starting MySQL...ok.

Where to locate the Programs ?

Xampp :

Create a Folder in /opt/lampp/htdocs

Example:

sai@node2:~$ mkdir /opt/lampp/htdocs/sample

Lamp

Create a Folder in /var/www

Wamp

Create a Folder C:/www

Static Example

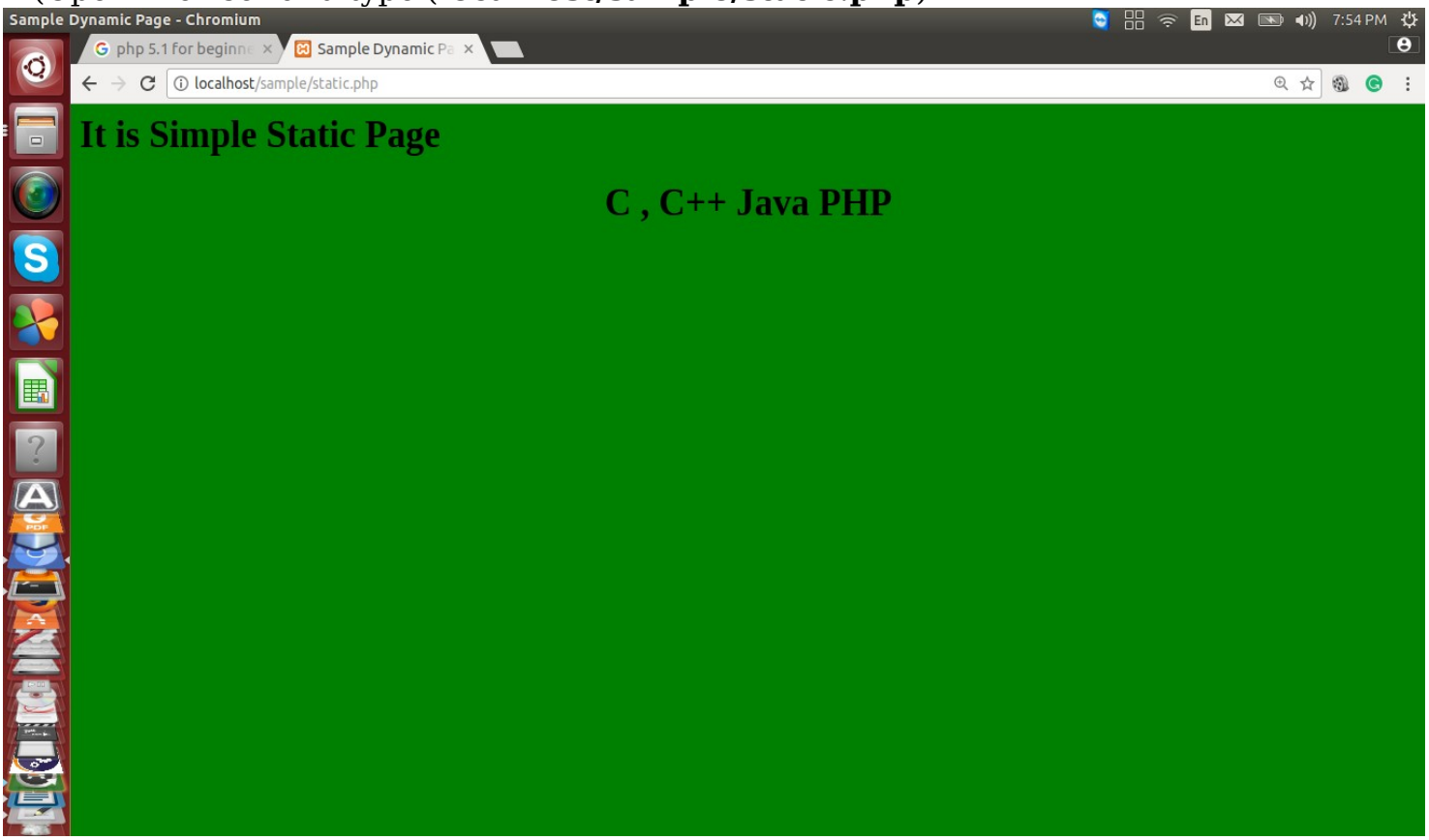
sai@node2:~$ cd /opt/lampp/htdocs/sample/

sai@node2:/opt/lampp/htdocs/sample$ **gedit static.php**

```
<html>
<title>Sample Static Page </title>
<body bgcolor="GREEN" >
<h1> It is Simple Static Page </h1>

<h1 Align="Center" > C , C++ Java PHP   </h1>

</body>
</html>
```

Viewing PHP output

(Open Browser and type (**localhost/sample/static.php**)



Dynamic Example

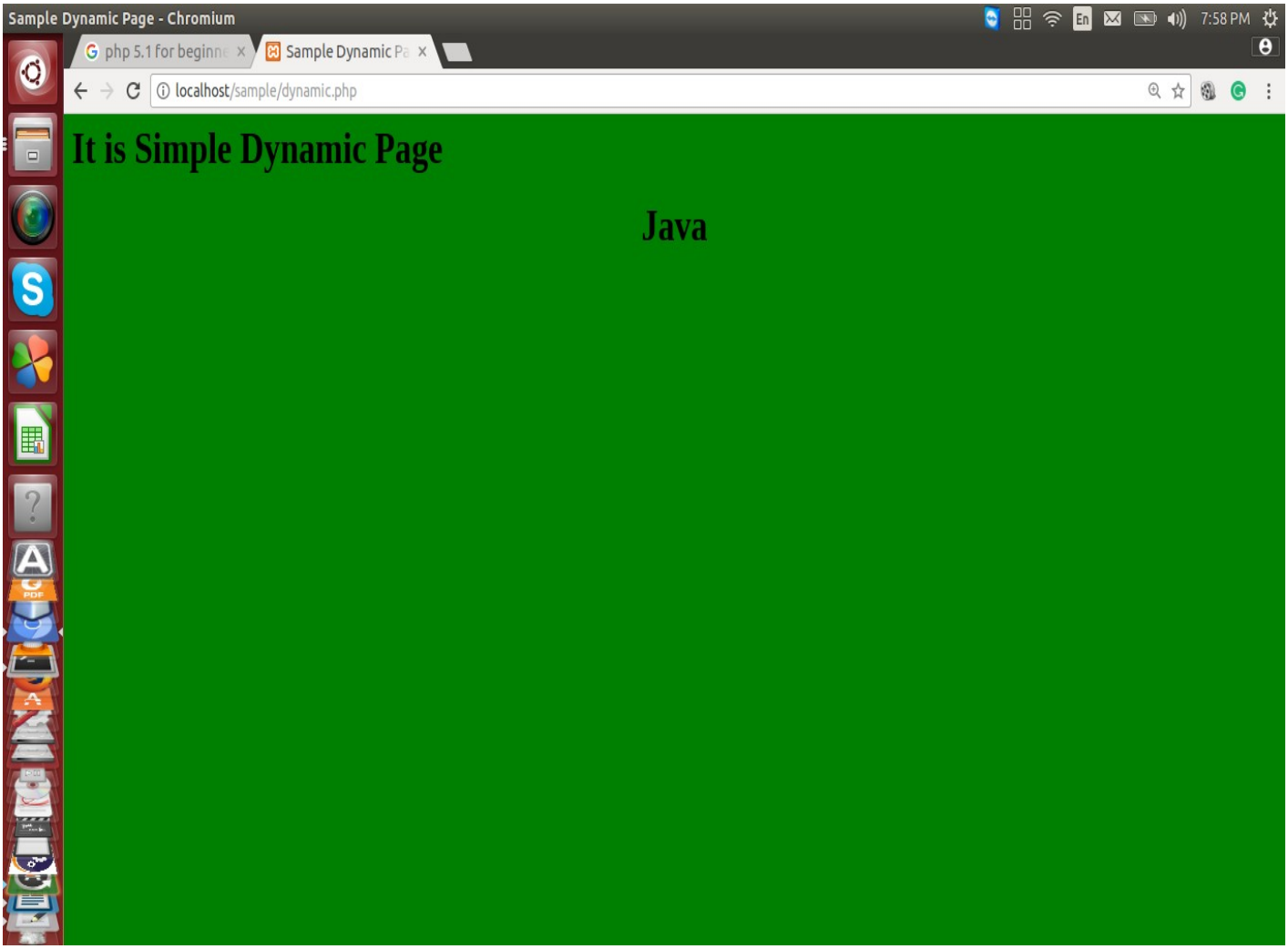
sai@node2:~$ cd /opt/lampp/htdocs/sample/

sai@node2:/opt/lampp/htdocs/sample$ **gedit dynamic.php**

```
<html>
<title>Sample Dynamic Page </title>
<body bgcolor="GREEN" >
<h1> It is Simple Dynamic Page </h1>
<?php
$books=array("C", "C++", "Java", "PHP");
$item=rand(0,sizeof($books)-1);
?>
<h1 Align="Center" >
<?php
echo $books[$item];
?>
</h1>
</body>
</html>
```

Viewing PHP output

(Open Browser and type (**localhost/sample/dynamic.php**)



Note: Keep refreshing the page to get the dynamic(runtime) output

Variables

```
$name = expression;                                                    PHP

$user_name = "PinkHeartLuvr78";
$age = 16;
$drinking_age = $age + 5;
$this_class_rocks = TRUE;                                              PHP
```

- Names are case sensitive; separate multiple words with _
- Names always begin with $, on both declaration and usage
- Always implicitly declared by assignment
- A loosely typed language (like JavaScript or Python:type is not written)

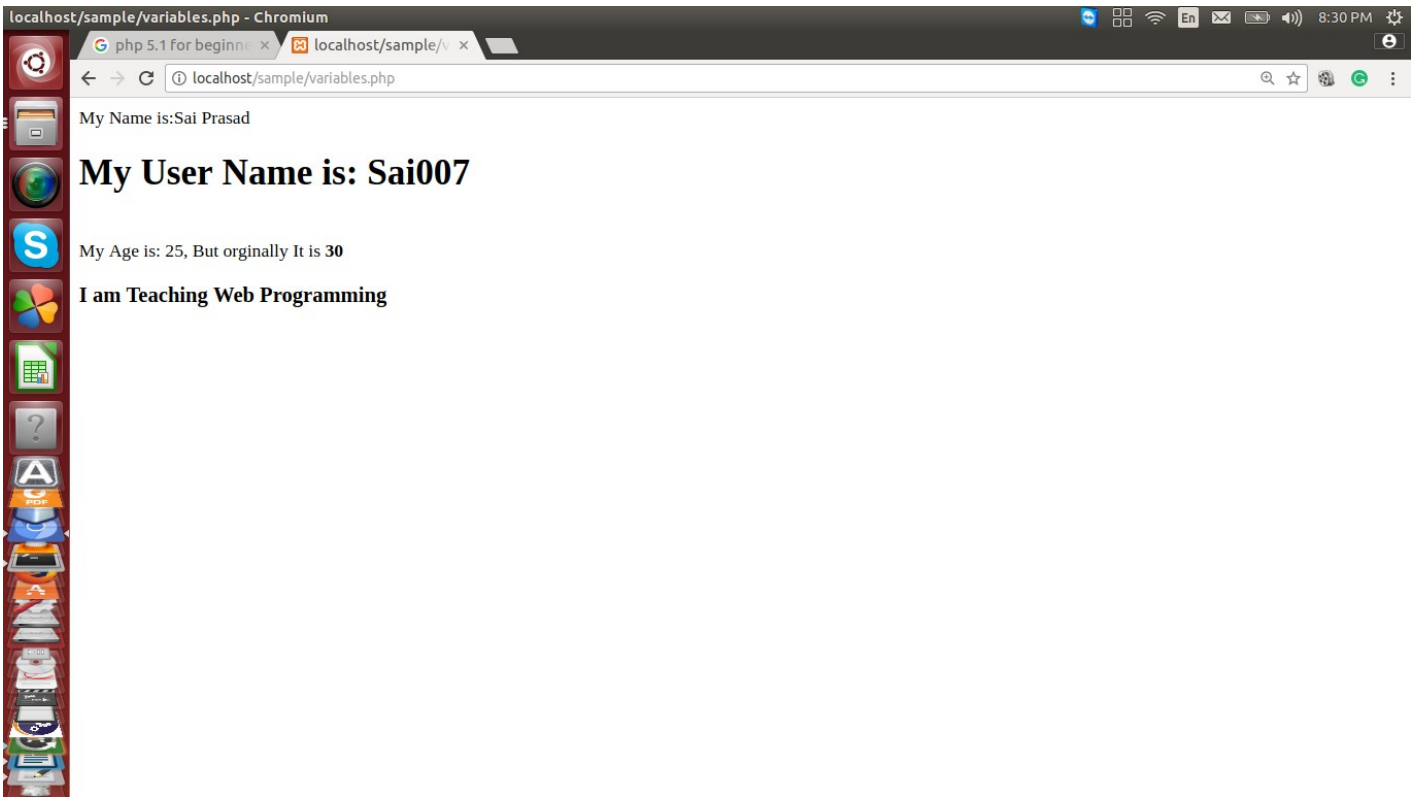
Example

sai@node2:/opt/lampp/htdocs/sample$ gedit variables.php

```
<?php
$name = "Sai Prasad";
$user_name = "Sai007";
$age = 25;
$correct_age = $age + 5;
$are_u_correct = TRUE;
$teach="Web Programming";
echo "My Name is:$name";
echo "<br>";
echo "<h1> My User Name is: $user_name  </h1> <br>";
echo  "My Age is: $age, But orginally It is <b> $correct_age <br>";
echo "<h3> I am Teaching $teach </h3>";

?>
```

Viewing PHP output

(Open Browser and type (**localhost/sample/variables.php**)



Types

- Basic types: `int`, `float`, `boolean`, `string`, `array`, `object`, `NULL`
  - test what type a variable is with `is_type` functions, e.g. `is_string`

◦ `gettype` function returns a variable's type as a string (not often needed)

- PHP converts between types automatically in many cases:
  - `string → int` auto-conversion on `+`
  - `int → float` auto-conversion on `/`
- type-cast with `(type)`:
  - `$age = (int) "21";`

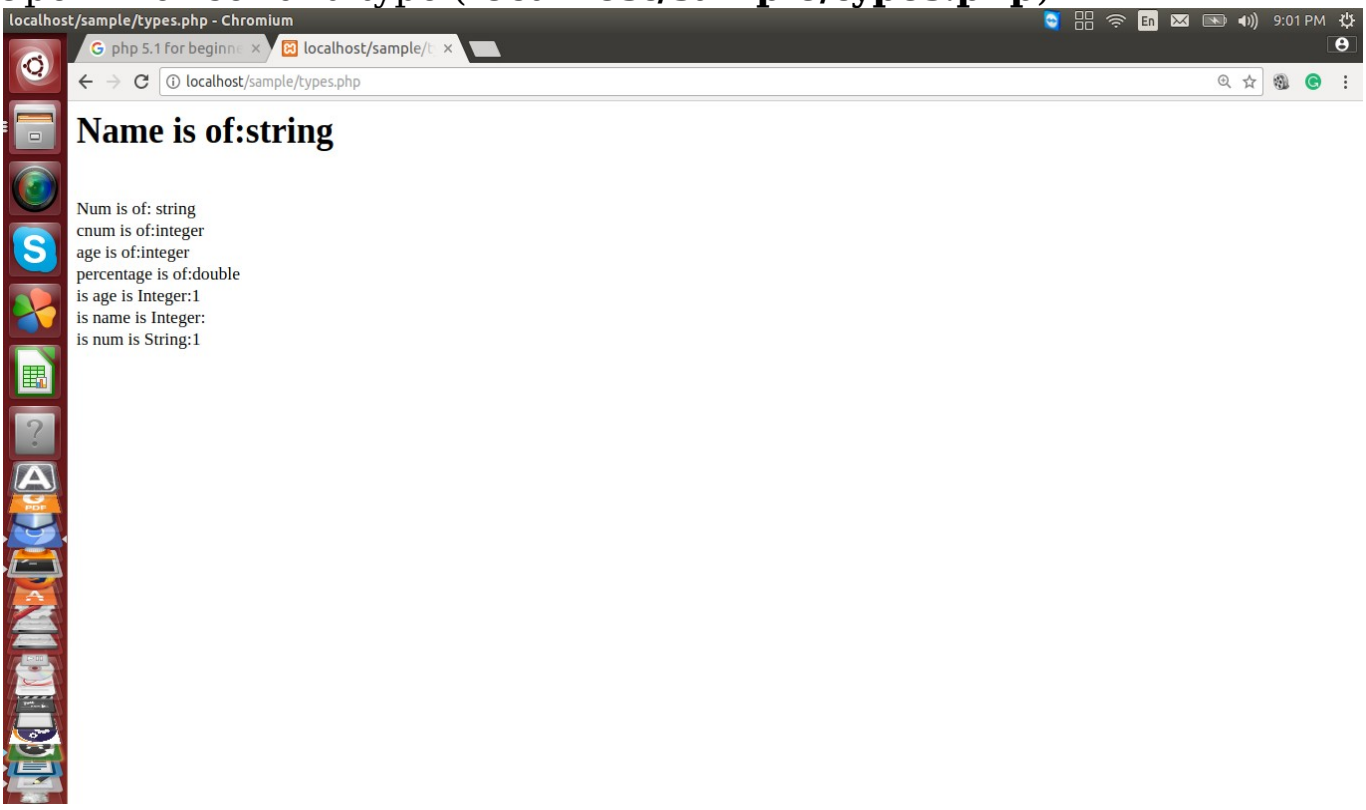
Example

sai@node2:/opt/lampp/htdocs/sample$ gedit types.php

```
<?php
$name = "Sai Prasad";
$num= "7";
$cnum =(int) "$num";
$age = 25;
$percentage=75.3;
echo "<h1> Name is of:" . gettype($name) ."</h1><br>" ;

//Note: .(dot) is for concatenation

$tnum=gettype($num);
echo "Num is of: $tnum <br>" ;
echo "cnum is of:" . gettype($cnum)."<br>" ;
echo "age is of:" . gettype($age)."<br>" ;
echo "percentage is of:" . gettype($percentage)."<br>" ;
echo "is age is Integer:" . is_int($age)."<br>" ;
echo "is name is Integer:" . is_int($name)."<br>" ;
echo "is num is String:" . is_string($num)."<br>" ;

?>
```

Viewing PHP output

(Open Browser and type (**localhost/sample/types.php**)



Operators (5.2.4)

- `+ - * / % . ++ --`
  `= += -= *= /= %= .=`
  `= != === !== > < >= <= && || !`
- `==` just checks value (`"5.0" == 5` is `TRUE`)
- `===` also checks type (`"5" === 5` is `FALSE`)
- many operators auto-convert types: `5 < "7"` is `TRUE`

int and float types

```php
$a = 7 / 2;              # float: 3.5
$b = (int) $a;           # int: 3
$c = round($a);          # float: 4.0
$d = "123";              # string: "123"
$e = (int) $d;           # int: 123
```

- `int` for integers and `float` for reals
- division between two `int` values can produce a `float`

`String` type

```php
$favorite_food = "Ethiopian";
print $favorite_food[2];              # h
```

- zero-based indexing using bracket notation
- string concatenation operator is `.` (period), not `+`
  - `5 + "2 turtle doves" === 7`
    - `5 . "2 turtle doves" === "52 turtle doves"`

`String` functions

```php
$name = "Kenneth Kuan";
$length = strlen($name);              # 12
$cmp = strcmp($name, "Jeff Prouty");  # > 0
$index = strpos($name, "e");          # 1
$first = substr($name, 8, 4);         # "Kuan"
$name = strtoupper($name);            # "KENNETH KUAN"
```

Name	Java Equivalent
explode, implode	split, join
strlen	length
strcmp	compareTo
strpos	indexOf
substr	substring
strtolower, strtoupper	toLowerCase, toUpperCase
trim	trim

Example:

Example:
sai@node2:/opt/lampp/htdocs/sample$ gedit stringdemo.php

```
<!DOCTYPE html>
<html>
	<head>
		<title>StringDemo</title>

	</head>
	<body>

		<div class="container" style="margin-top: 50px">

			<?php

				// If the submit button has been pressed
				if(isset($_POST['submit']))
				{
						$s = $_POST['string1'] ;
					$length = strlen($s);
					$low = strtolower($s);
					$index = strpos($s, "e");
					$first = substr($s, 8, 4);
					$uname = strtoupper($s);

					// Print total to the browser
					echo "<h1> String  Given is $s <br> Length is: $length
<br> Lower case : $low <br> Index of e :$index <br> substring : $first <br> Uppercase : $uname
```

```
<br> </h1>";
                                }

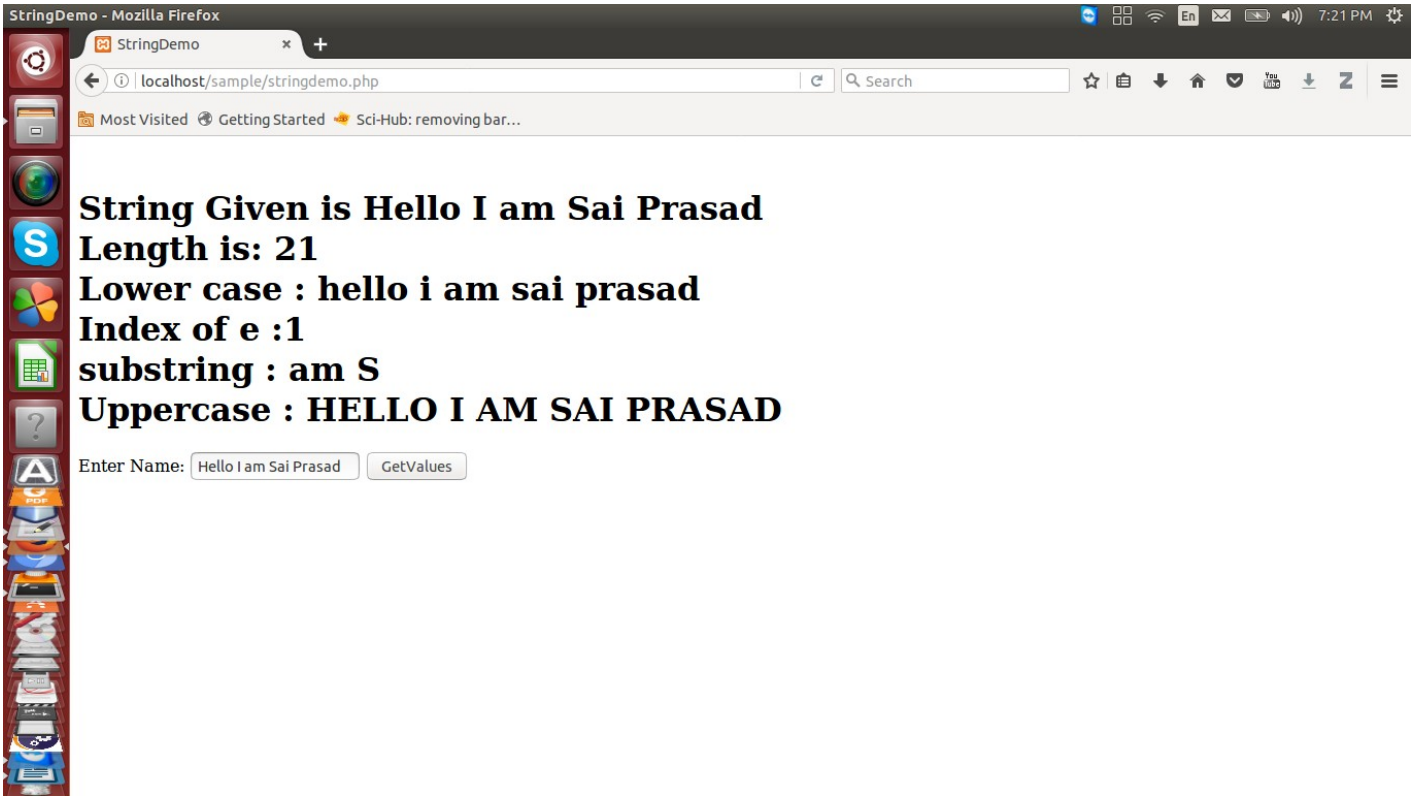

                            ?>


                    <!-- String form -->
                    <form method="post" action="stringdemo.php">
                        Enter Name: <input name="string1" type="text" class="form-control"
style="width: 150px; display: inline" />
                        <input name="submit" type="submit" value="GetValues" class="btn btn-
primary" />
                    </form>


                </div>


        </body>
</html>
```

OUTPUT:



Interpreted strings

- ```
  $age = 16;
  print "You are " . $age . " years old.\n";
  print "You are $age years old.\n";      # You are 16 years old.
  ```
-

```
print  'You are $age years old.\n'; # You are $age years old.\n
Note: Single code not interpreted
```

Math operations

```
$a = 3;
$b = 4;
$c = sqrt(pow($a, 2) + pow($b, 2));                                    PHP
```

abs()	Returns the absolute (positive) value of a number
acos()	Returns the arc cosine of a number
acosh()	Returns the inverse hyperbolic cosine of a number
asin()	Returns the arc sine of a number
```

asinh()	Returns the inverse hyperbolic sine of a number
atan()	Returns the arc tangent of a number in radians
atan2()	Returns the arc tangent of two variables x and y
atanh()	Returns the inverse hyperbolic tangent of a number
base_convert()	Converts a number from one number base to another
bindec()	Converts a binary number to a decimal number
ceil()	Rounds a number up to the nearest integer
cos()	Returns the cosine of a number
cosh()	Returns the hyperbolic cosine of a number
decbin()	Converts a decimal number to a binary number
dechex()	Converts a decimal number to a hexadecimal number
decoct()	Converts a decimal number to an octal number
deg2rad()	Converts a degree value to a radian value
exp()	Calculates the exponent of e
expm1()	Returns exp(x) - 1
floor()	Rounds a number down to the nearest integer
fmod()	Returns the remainder of x/y
getrandmax()	Returns the largest possible value returned by rand()
hexdec()	Converts a hexadecimal number to a decimal number
hypot()	Calculates the hypotenuse of a right-angle triangle
is_finite()	Checks whether a value is finite or not
is_infinite()	Checks whether a value is infinite or not
is_nan()	Checks whether a value is 'not-a-number'
lcg_value()	Returns a pseudo random number in a range between 0 and 1
log()	Returns the natural logarithm of a number
log10()	Returns the base-10 logarithm of a number
log1p()	Returns log(1+number)
max()	Returns the highest value in an array, or the highest value of several specified values
min()	Returns the lowest value in an array, or the lowest value of several specified values
mt_getrandmax()	Returns the largest possible value returned by mt_rand()
mt_rand()	Generates a random integer using Mersenne Twister algorithm
mt_srand()	Seeds the Mersenne Twister random number generator
octdec()	Converts an octal number to a decimal number
pi()	Returns the value of PI
pow()	Returns x raised to the power of y
rad2deg()	Converts a radian value to a degree value
rand()	Generates a random integer
round()	Rounds a floating-point number
sin()	Returns the sine of a number
sinh()	Returns the hyperbolic sine of a number
sqrt()	Returns the square root of a number
srand()	Seeds the random number generator
tan()	Returns the tangent of a number
tanh()	Returns the hyperbolic tangent of a number

Example:

sai@node2:/opt/lampp/htdocs/sample$ gedit maths.php

```
<?php

$a = 3;
```

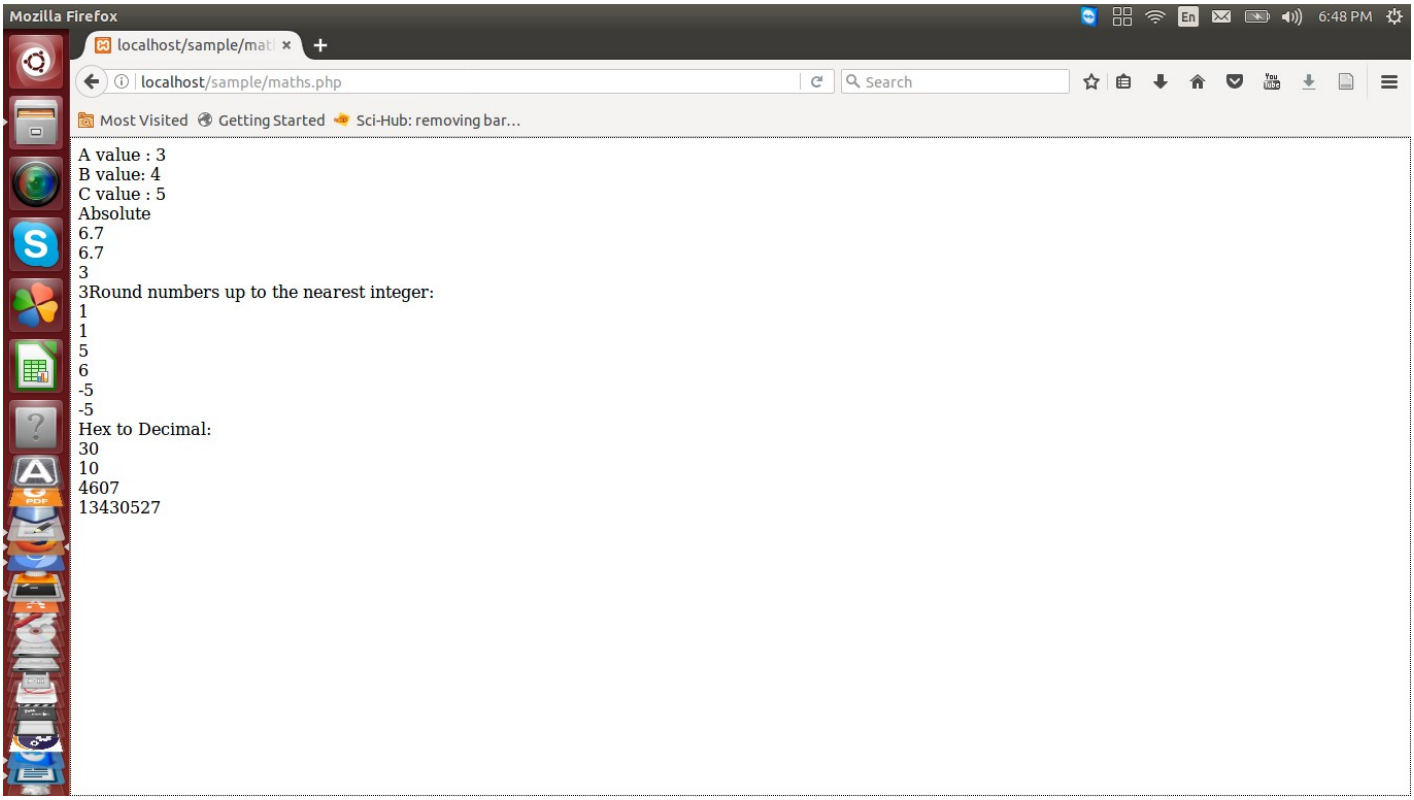
```
$b = 4;
$c = sqrt(pow($a, 2) + pow($b, 2));

echo "A value : $a <br> B value: $b <br> C value : $c <br> ";
echo "Absolute <br>";
echo(abs(6.7) . "<br>");
echo(abs(-6.7) . "<br>");
echo(abs(-3) . "<br>");
echo(abs(3));

echo "Round numbers up to the nearest integer: <br>";
echo(ceil(0.60) . "<br>");
echo(ceil(0.40) . "<br>");
echo(ceil(5) . "<br>");
echo(ceil(5.1) . "<br>");
echo(ceil(-5.1) . "<br>");
echo(ceil(-5.9)."<br>");

echo "Hex to Decimal: <br>";
echo hexdec("1e") . "<br>";
echo hexdec("a") . "<br>";
echo hexdec("11ff") . "<br>";
echo hexdec("cceeff");
?>
```

View Output:



Assignment

/ * Implement Simple Calculator */

```
<!DOCTYPE html>

<html>

        <head>

                <title>Calculator</title>


        </head>

        <body>


                <div class="container" style="margin-top: 50px">


                        <?php
```

```
                        // If the submit button has been pressed
                        if(isset($_POST['submit'])) //Reads the Action
                        {
                                // Check number values
                                if(is_numeric($_POST['number1']) &&
is_numeric($_POST['number2']))
                                {
                                        // Calculate total
                                        if($_POST['operation'] == 'plus')
                                        {
                                                $total = $_POST['number1'] + $_POST['number2'];
                                        }
                                        if($_POST['operation'] == 'minus')
                                        {
                                                $total = $_POST['number1'] - $_POST['number2'];
                                        }
                                        if($_POST['operation'] == 'times')
                                        {
                                                $total = $_POST['number1'] * $_POST['number2'];
                                        }
                                        if($_POST['operation'] == 'divided by')
                                        {
                                                $total = $_POST['number1'] / $_POST['number2'];
                                        }

                                        // Print total to the browser
                                        echo "<h1>{$_POST['number1']} {$_POST['operation']}
{$_POST['number2']} equals {$total}</h1>";
```

```
                                    } else {

                                            // Print error message to the browser
                                            echo 'Numeric values are required';

                                    }
                                }

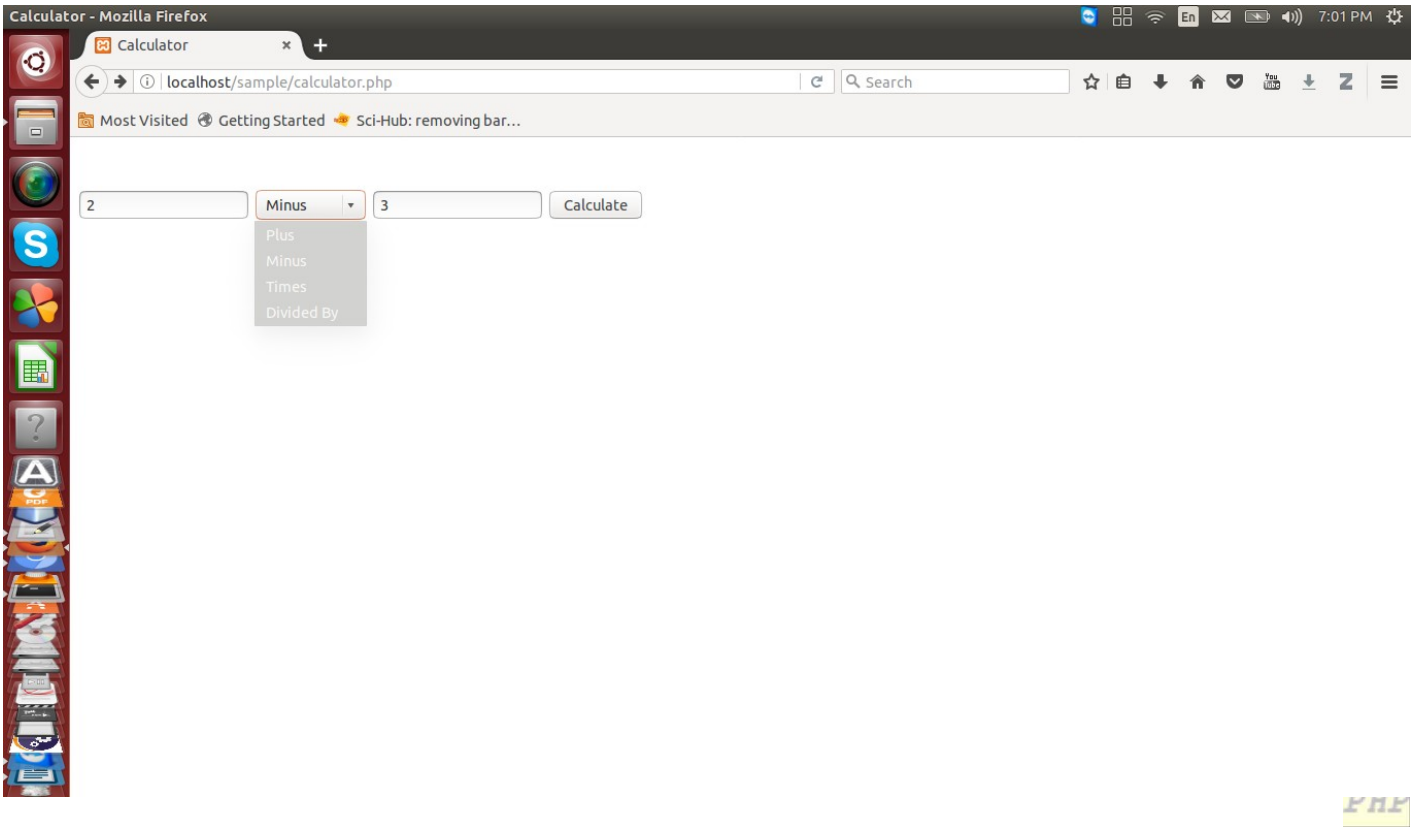
                            ?>

                    <!-- Calculator form -->
                    <form method="post" action="calculator.php">
                        <input name="number1" type="text" class="form-control" style="width: 150px;
display: inline" />
                        <select name="operation">
                            <option value="plus">Plus</option>
                            <option value="minus">Minus</option>
                            <option value="times">Times</option>
                            <option value="divided by">Divided By</option>
                        </select>
                        <input name="number2" type="text" class="form-control" style="width: 150px;
display: inline" />
                        <input name="submit" type="submit" value="Calculate" class="btn btn-
primary" />
                    </form>

                </div>

        </body>
</html>
```

OUTPUT:


Calculator - Mozilla Firefox
7:01 PM
Calculator
localhost/sample/calculator.php
Search
Most Visited
Getting Started
Sci-Hub: removing bar...
2
Minus
3
Calculate
Plus
Minus
Times
Divided By

for loop (same as Java) (5.2.9)

```
for (initialization; condition; update) {
  statements;
}
```

```
for ($i = 0; $i < 10; $i++) {
  print "$i squared is " . $i * $i . ".\n";
}
```

bool (Boolean) type

```
$feels_like_summer = FALSE;

$php_is_rad = TRUE;

$student_count = 217;
                                        # TRUE
$nonzero = (bool) $student_count;
```

- the following values are considered to be FALSE (all others are TRUE):
  - 0 and 0.0 (but NOT 0.00 or 0.000)
  - "", "0", and NULL (includes unset variables)
  - arrays with 0 elements
- can cast to boolean using (bool)
- FALSE prints as an empty string (no output); TRUE prints as a 1
- TRUE and FALSE keywords are case insensitive

`if/else` statement

```
if (condition) {
  statements;
} elseif (condition)
  { statements;
} else {
  statements;
}
```

- NOTE: although `elseif` keyword is much more common, `else if` is also supported

`while` loop (same as Java)

```
while (condition) {
  statements;
}

do {
  statements;
} while (condition);
```

- `break` and `continue` keywords also behave as in Java

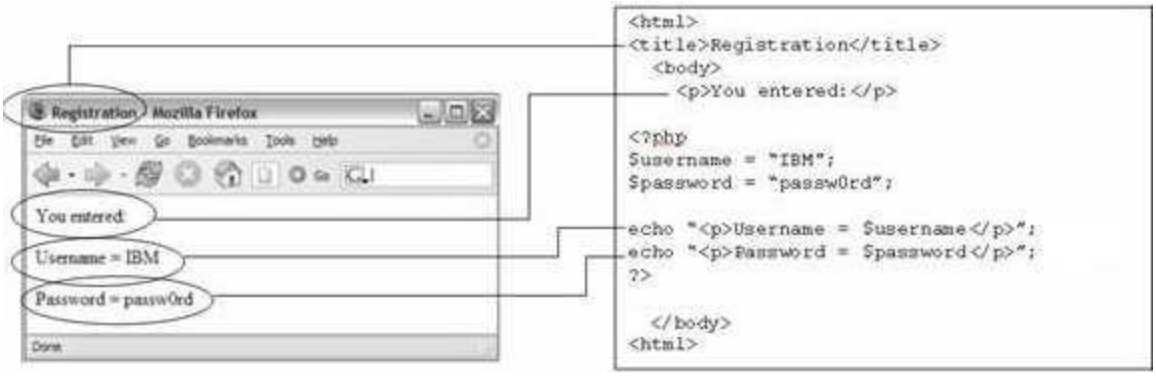
NULL

```
$name = "Victoria";
$name = NULL;
if (isset($name)) {
  print "This line isn't going to be reached.\n";
}
```

- a variable is `NULL` if
  - it has not been set to any value (undefined variables)
  - it has been assigned the constant `NULL`
  - it has been deleted using the `unset` function
- can test if a variable is `NULL` using the `isset` function • `NULL` prints as an empty string (no output)

Embedding code in web pages

- most PHP programs actually produce HTML as their output
  - dynamic pages; responses to HTML form submissions; etc.
- an embedded PHP program is a file that contains a mixture of HTML and PHP code



A bad way to produce HTML in PHP

```
<?php
print "<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.1//EN\"\n";
print " \"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd\">\n";
print "<html xmlns=\"http://www.w3.org/1999/xhtml\">\n"; print
" <head>\n";
print "    <title>My web page</title>\n";
...
?>
```

- printing HTML code with `print` statements is ugly and error-prone:
  - must quote the HTML and escape special characters, e.g. `\"`
  - must insert manual `\n` line breaks

after each line • don't `print` HTML; it's bad style!

Syntax for embedded PHP (5.3.1)

```
HTML content

<?php
PHP code
?>

HTML content
```
PHP

- any contents of a `.php` file that are not between `<?php` and `?>` are output as pure HTML • can switch back and forth between HTML and PHP "modes"

Embedded PHP example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head><title>CSE 190 M: Embedded
  PHP</title></head> <body>
    <h1>Geneva's Counting Page</h1>
    <p>Watch how high I can count:
      <?php
      for ($i = 1; $i <= 10; $i++) {
        print "$i\n";
      }
      ?>
    </p>
  </body>
</html>
```
PHP

- the above code would be saved into a file such as `count.php` • How many lines of numbers will appear? (View Source!)

Embedded PHP + `print` = bad

```
...
    <h1>Geneva's Counting Page</h1>
    <p>Watch how high I can count:
      <?php
      for ($i = 1; $i <= 10; $i++) {
        print "$i\n";
      }
      ?>
    </p>
```
PHP

- best PHP style is to use as few `print`/`echo` statements as possible in embedded PHP code
- but without `print`, how do we insert dynamic content into the page?

PHP expression blocks (5.3.2)

```
<?= expression ?>
```
PHP

```
<h2>The answer is <?= 6 * 7 ?></h2>
```
PHP

The answer is 42

output

- PHP expression block: a small piece of PHP that evaluates and embeds an expression's value into HTML
  - `<?= expression ?>` is equivalent to:

    ```
    <?php print expression; ?>
    ```
    PHP

  - useful for embedding a small amount of PHP (a variable's or expression's value) in a large block of HTML without having to switch to "PHP-mode"

Expression block example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head><title>CSE 190 M: Embedded
  PHP</title></head> <body>
    <?php
    for ($i = 99; $i >= 1; $i--) {
      ?>
      <p><?= $i ?> bottles of beer on the wall,
         <br /> <?= $i ?> bottles of beer. <br />
         Take one down, pass it around, <br />
         <?= $i - 1 ?> bottles of beer on the
      wall.</p> <?php
    }
    ?>
  </body>
</html>
```

PHP

- this code could go into a file named `beer.php`

Common error: unclosed braces

```
...
  <body>
    <p>Watch how high I can count:
      <?php
      for ($i = 1; $i <= 10; $i++) {
        ?>
        <?= $i ?>
    </p>
  </body>
</html>
```

PHP

- if you open a { brace, you must have a matching } brace later
  - `</body>` and `</html>` above are inside the `for` loop, which is never closed • if you forget to close your braces, you'll see an error about 'unexpected $end'

Common error fixed

```
...
  <body>
    <p>Watch how high I can count:
      <?php
      for ($i = 1; $i <= 10; $i++) {     # PHP mode
        ?>
        <?= $i ?>                      <!-- HTML mode -->
        <?php
      }                                    # PHP mode
      ?>
    </p>
  </body>
</html>                                                    PHP
```

Common error: Missing = sign

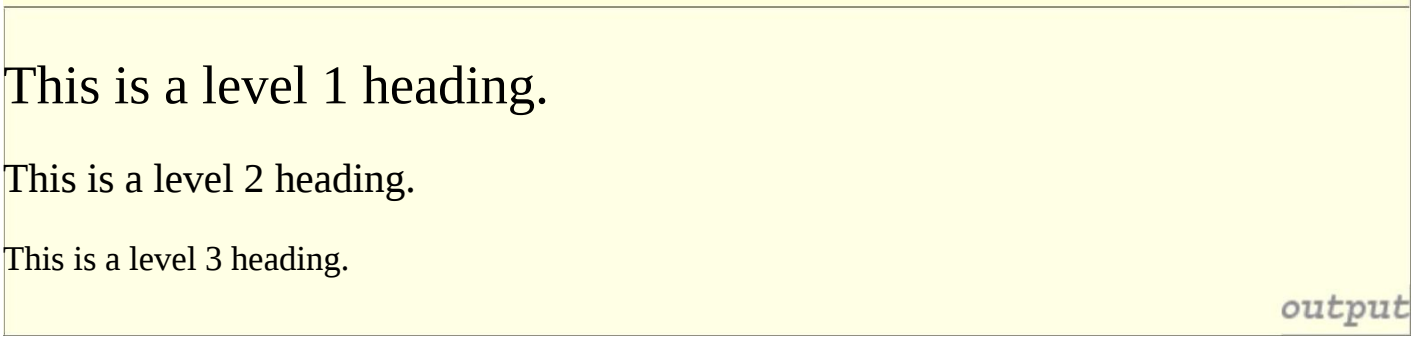
```
...
  <body>
    <p>Watch how high I can count:
      <?php
      for ($i = 1; $i <= 10; $i++) {
        ?>
        <? $i ?>
        <?php
      }
      ?>
    </p>
  </body>
</html>                                                    PHP
```

- a block between <? ... ?> is often interpreted the same as one between <?php ... ?> • PHP evaluates the code, but $i does not produce any output

Complex expression blocks

```
...
  <body>
    <?php
    for ($i = 1; $i <= 3; $i++) {
      ?>
      <h<?= $i ?>>This is a level <?= $i ?> heading.</h<?= $i ?
      >> <?php
    }
    ?>
  </body>
```

PHP



output

- expression blocks can even go inside HTML tags and attributes

Functions (5.4.1)

```
function name(parameterName, ..., parameterName)
  { statements;
}
```
PHP

```
function quadratic($a, $b, $c) {
  return -$b + sqrt($b * $b - 4 * $a * $c) / (2 * $a);
}
```
PHP

- parameter types and return types are not written

Calling functions

```
name(parameterValue, ..., parameterValue);
```
PHP

```
$x = -2;
$a = 3;
$root = quadratic(1, $x, $a - 2);
```
PHP

- if the wrong number of parameters are passed, it's an error

Default parameter values

```
function name(parameterName, ..., parameterName)
  { statements;
}
```
PHP

```
function print_separated($str, $separator = ", ")
  { if (strlen($str) > 0) {
    print $str[0];
    for ($i = 1; $i < strlen($str); $i++)
      { print $sep . $str[$i];
    }
  }
}
```
PHP

```
print_separated("hello");         # h, e, l, l, o
print_separated("hello", "-");    # h-e-l-l-o
```
PHP

- if no value is passed, the default will be used (defaults must come last)

Variable scope: global and local vars

```
$school = "UW";                   # global
...
function downgrade() {

  global $school;
  $suffix = "Tacoma";             # local
  $school = "$school $suffix";

  print "$school\n";
}
```
PHP

- variables declared in a function are local to that function
- variables not declared in a function are global
- if a function wants to use a global variable, it must have a `global` statement

Including files: `include()` (5.4.2)

```php
include("filename");
```

```php
include("header.php");
```

- inserts the entire contents of the given file into the PHP script's output page
- encourages modularity
- useful for defining reused functions like form-checking

Arrays (5.4.3)

```php
$name = array();                              # create
$name = array(value0, value1, ..., valueN);
$name[index]                                  # get element value

$name[index] = value;                         # set element value
$name[] = value;                              # append
```

```php
$a = array();      #empty array (length 0)
$a[0] = 23;        #stores 23 at index 0 (length 1)
$a2 = array("some", "strings", "in", "an", "array");
$a2[] = "Ooh!";    #add string to end (at index 5)
```

- to append, use bracket notation without specifying an index • element type is not specified; can mix types

Array functions

function name(s)	description
count	number of elements in the array
print_r	print array's contents
array_pop, array_push, array_shift, array_unshift	using array as a stack/queue
in_array, array_search, array_reverse, sort, rsort, shuffle	searching and reordering
array_fill, array_merge, array_intersect, array_diff, array_slice, range	creating, filling, filtering
array_sum, array_product, array_unique, array_filter, array_reduce	processing elements

Array function example

```php
$tas = array("MD", "BH", "KK", "HM", "JP");
for ($i = 0; $i < count($tas); $i++) {
  $tas[$i] = strtolower($tas[$i]);
}                                  # ("md", "bh", "kk", "hm", "jp")
$morgan = array_shift($tas);       # ("bh", "kk", "hm", "jp")
array_pop($tas);                   # ("bh", "kk", "hm")
array_push($tas, "ms");            # ("bh", "kk", "hm", "ms")
array_reverse($tas);               # ("ms", "hm", "kk", "bh")
sort($tas);                        # ("bh", "hm", "kk", "ms")
$best = array_slice($tas, 1, 2);   # ("hm", "kk")
```

PHP

- the array in PHP replaces many other collections in Java
  - list, stack, queue, set, map, ...

The `foreach` loop (5.4.4)

```
foreach ($array as $variableName) {
  ...
}
```
PHP

```
$stooges = array("Larry", "Moe", "Curly", "Shemp");
for ($i = 0; $i < count($stooges); $i++) {
  print "Moe slaps {$stooges[$i]}\n";
}
foreach ($stooges as $stooge) {
  print "Moe slaps $stooge\n"; # even himself!
}
```
PHP

- a convenient way to loop over each element of an array without indexes

Splitting/joining strings

```
$array = explode(delimiter, string);
$string = implode(delimiter, array);
```
PHP

```
$s  = "CSE 190 M";
$a  = explode(" ", $s);     # ("CSE", "190", "M")
$s2 = implode("...", $a);   # "CSE...190...M"
```
PHP

- `explode` and `implode` convert between strings and arrays
- for more complex string splitting, we'll use regular expressions (later)

Unpacking an array: `list`

```php
list($var1, ..., $varN) = array;
```

```php
$line = "stepp:17:m:94";
list($username, $age, $gender, $iq) = explode(":", $line);
```

- the `list` function accepts a comma-separated list of variable names as parameters
- assign an array (or the result of a function that returns an array) to store that array's contents
  into the variables

Non-consecutive arrays

```php
$autobots = array("Optimus", "Bumblebee",
"Grimlock"); $autobots[100] = "Hotrod";
```

- the indexes in an array do not need to be consecutive
- the above array has a `count` of 4, with 97 blank elements between `"Grimlock"` and `"Hotrod"`

PHP file I/O functions (5.4.5)

- reading/writing entire files: `file_get_contents`, `file_put_contents`
- asking for information: `file_exists`, `filesize`, `fileperms`, `filemtime`, `is_dir`, `is_readable`, `is_writable`, `disk_free_space`
- manipulating files and directories: `copy`, `rename`, `unlink`, `chmod`, `chgrp`, `chown`, `mkdir`, `rmdir`
- reading directories: `scandir`, `glob`

Reading/writing files

```php
$text = file_get_contents("schedule.txt");
$lines = explode("\n", $text); $lines =
array_reverse($lines);
$text = implode("\n", $lines);
file_put_contents("schedule.txt", $text);
```

- `file_get_contents` returns entire contents of a file as a string
  - if the file doesn't exist, you'll get a warning
- `file_put_contents` writes a string into a file, replacing any prior contents

Reading files example

```
# Returns how many lines in this file are empty or just
spaces. function count_blank_lines($file_name) {
  $text = file_get_contents($file_name);
  $lines = explode("\n", $text); $count
  = 0;
  foreach ($lines as $line) {
    if (strlen(trim($line)) == 0)
      { $count++;
    }
  }
  return $count;
}
…
print count_blank_lines("ch05-php.html");
```
PHP

Reading directories

```
$folder = "images";
$files = scandir($folder);
foreach ($files as $file) {
  if ($file != "." && $file != "..") {
    print "I found an image: $folder/$file\n";
  }
}
```
PHP

- `scandir` returns an array of all files in a given directory
- annoyingly, the current directory (`"."`) and parent directory (`".."`) are included in the array; you probably want to skip them