



PHP: Hypertext Preprocessor

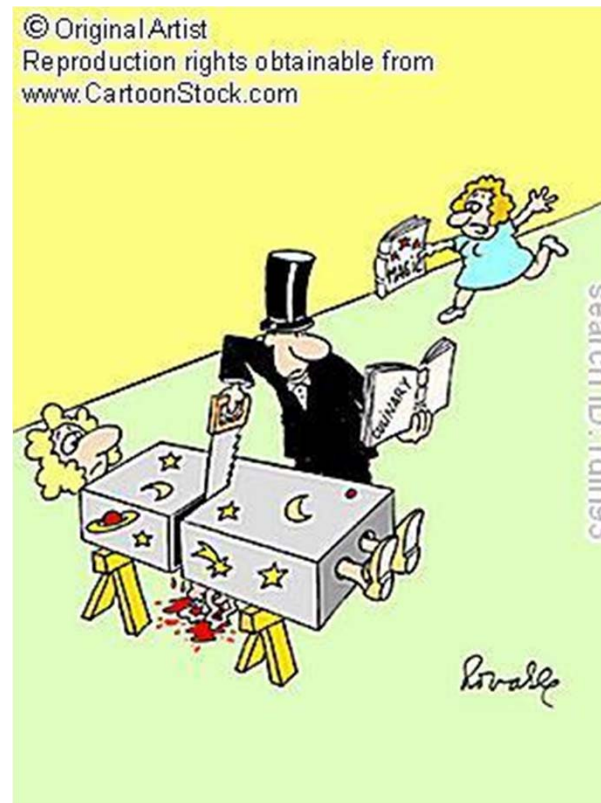


The Plan

- Get familiar with php, and how it works together with html
- Make a fun project
- Do some research

PHP is an interpreted language

- The interpreter reads and executes commands





`print` statements

- Basic construct for output
- Double quotes or single quotes
- Semi-colon finishes the statement

```
print "Double Quotes";
```

```
print 'Single Quotes';
```



Print (cont)

- Certain characters must be **escaped**
 - Double quotes - \"
 - Single quotes - \'
 - Dollar sign - \\$
 - Backslash - \

```
print "She said, \"The coat  
cost $79.95\"";
```

Try it!



- Application -> Accessories-> Terminal
- Open a terminal and type

```
php -r 'print "hello\n";'
```


and hit Enter
- type multiple print statements inside the single quotes.

How can we run it without typing it in over and over?



Write it in a file

- In a terminal, type `gedit hello.php &`
- Open with `<?php` and close with `?>`, put `print` in between.

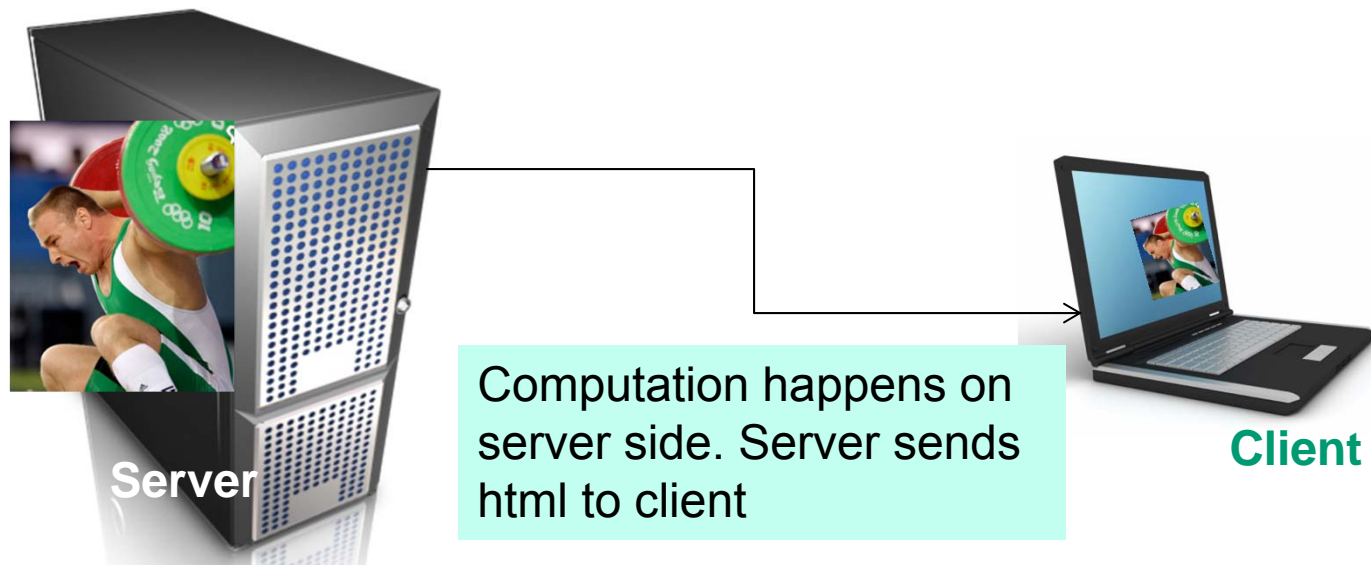
```
<?php print "Hello!\n"; ?>
```

- Save the file
- In a terminal, type `php hello.php`

How do we get it into a web page?

Motivation for PHP on the web

1. We want many people to be able to run our program
2. Our program uses lots of data and does lots of computation.





```
<html>
```

```
<head>
```

```
  <title>My First PHP  
  Script</title>
```

```
</head>
```

```
<body>
```

```
This web page contains html  
  and php.
```

```
<?php
```

```
print "<h1>Hello!</h1>";
```

```
?>
```

```
</body>
```

```
</html>
```

Comments??




- It's a good idea to add comments to your code
- A comment is text that humans can read and computers ignore
- Helps you remember what your code does
- Helps other people who want to use your code understand it.



Comments

- Single line
`// undetected errors are ignored`
- Multi-line
`/* Either the program is exiting
or the world is coming to an end.
Take your pick. */`

Exercise

1. In your terminal type
`cd /var/www/workshop`
2. Type `gedit hello.php` &
3. Open browser to 
<http://localhost/workshop>
4. Click `hello.php`

Variables



\$var

- Variables hold data
- All variables are represented with a dollar sign and a name: `$var`
- Variables **names** represent **memory locations**
- Variable names
 - contain letters, numbers and underscores **only**
 - Cannot start with a number
- Seven types of data: `bool`, `string`, `int`, `float`, `array`, `object`, `resource`

Variables Assignment

Assignment is done with the
assignment operator =

`$var` = `data;`

`$var` gets set to `data;`



`$var`



`$var`



Bad Variable Names

```
$123_go! = 4;
```

```
$data = 3;
```

```
$IsNotCanNotUnDisable = 1;
```

```
$1 = 1;
```

Good variable names describe the data they contain

```
average_rainfall = 32;
```

Variables – integer

Whole numbers.

Some Operations	
<code>\$c = \$a + \$b;</code>	
<code>\$c = \$a - \$b;</code>	
<code>\$c = \$a * \$b;</code>	Multiplication
<code>\$c = \$a / \$b;</code>	Assigns quotient
<code>\$c = \$a % \$b;</code>	Assigns remainder
<code>\$c++;</code>	<code>\$c = \$c + 1;</code> increment \$c



Variables - string

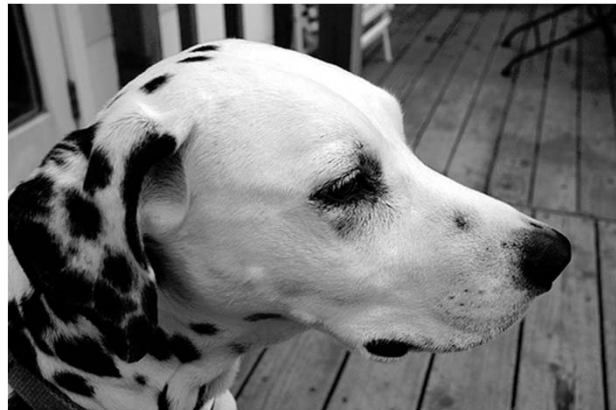
- Composed of letters, numbers, characters on the keyboard
- **Surrounded by quotes**
- Certain characters must be **escaped**
- The escape character is the forward slash
- Single quotes inside single quotes must be escaped
- Double quotes inside double quotes must be escaped

Single quotes	\'
Double quotes	\''
Dollar sign	\\$
Backslash	\\
Carriage Return (new line)	\n

Why \$ is escaped

When within quotes, if \$ is not escaped, php treats it as a variable, evaluates it and prints the value

```
//$name is evaluated inside " "  
$name = "Spot";  
print "My name is $name";
```



String operator Concatenation

Main Entry: ²**concatenate** 

Pronunciation: \-,nāt\

Function: *transitive verb*

Inflected Form(s): **concatenat·ed**; **concatenat·ing**

Date: 1598

: to link together in a series or chain



- Different types of data can be merged together into a single `string`

```
$var = "Hello";  
print $var . " World";
```

Try these in a terminal



```
php -r  
`print "The coat cost \$79.95";`  
  
php -r `print "\\0/\n \$\n/ \\";`  
  
php -r `$location = "Buffalo";  
print "I live in ". $location;`
```

Concatenation
or period?



Variables – float

- `float` represents rational (real) numbers

```
$price = 7.95;
```

- What operations can we do on floats?

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

```
$c = $a - $b;
```


```
$c = $a * $b;
```

```
$c = $a / $b;
```

```
$c = $a % $b;
```

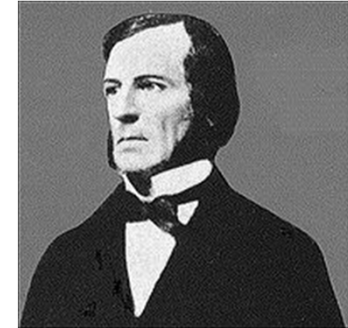
```
$c++;
```

Exercise

1. In your terminal type
`cd /var/www/workshop`
2. Type `gedit variables.php` &
3. Open browser to 
<http://localhost/workshop>
4. Click `variables.php`

Variables

boolean



George Boole

- Most basic type of data in any computer language
- The only values it can have are **true** and **false**

```
$I_chopped_down_the_cherry_tree =  
  true;
```

```
$I_am_over_five_feet = false;
```



Comparison Operators

- (Left-hand side *operator* right-hand side)
- Evaluate to **true** or **false** (booleans!!)

(50 > 5)

(5 < 1)

(10 <= 12)

(10.2 >= 10)



Comparison Operators

Equal	==	
Identical	===	Equal value, and same type
Not equal	!=	
Not Identical	!==	Not equal value, or not same type
Greater than	>	
Less than	<	
Greater than or equal	>=	
Less than or equal	<=	



What is it?

```
$price = 5.60;  
$five_greater_than_three =  
    ( 5 > 3 );  
$data = 10;  
$data = false;  
$print = "print";  
print $print.$price;
```

Variables - array

- Arrays are a **collection** of data with memory addresses
- All data types can be stored in an array

```
$names = array("Matt", "Ann",  
              "Jim");
```

Matt	Ann	Jim
0	1	2

```
$numbers = array(2, 4, 6, 8);
```

2	4	6	8
0	1	2	3

- Elements are accessed with a **key** using the array operators

```
print $names[0]; // Matt
```



Variables – associative array

- By default, keys are integers starting at 0
- Instead of integers, **keys** can be strings

```
$ages = array("Matt" => 20, "Ann" => 19, "John" => 21);
```

Alternate way to build an array, using single statements:

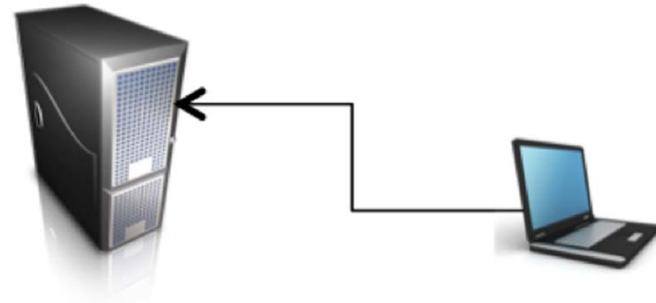
```
$ages["Alisa"] = 30; // I wish
```

- Elements are accessed the same way

```
print $ages["Ann"]; // 19
```

Associative Array Example: Form Data

- Sometimes a web application requires input from a user
- HTML form tags for input
 - textbox
 - checkbox
- Data sent to the server as associative array named GET or POST
 - POST: input data hidden from the user
 - GET: input data is part of the URL string





Form Data

```
<html>
<head>
<title>PHP and Form Data</title>
</head>
<body>
<form action="formdata.php" method="GET">
Number: <input type="text" name="number" /> <br />
Letter: <input type="text" name="letter" /> <br />
<input type="submit" name="submit" value="Submit Data" />
</form>
<?php      //If form is submitted echo Data
if ( isset($_GET['submit']) ){
echo "<b> Number: </b>" . $_GET['number'] . "<br />";
echo "<b> Letter: </b>" . $_GET['letter'];}
?>
</body>
</html>
```

Loops: motivation

- In a program, there's often a sequence of actions that need to be done over and over (a loop!)
- Arrays and loops go together like peanut butter and jelly

Next: need some volunteers..



Loops - for

- Three parts to a for loop
 - Variable Initialization
 - Sentinel Condition
 - Variable Modification (iteration)




```
$booknames = array("Calculus", "Algorithms",  
    "High Performance Computing", "Statistics");
```

```
for( $i = 0; $i < 4; $i++ )
```

```
{  
    say "add " . $booknames[$i] . " please";  
    add a book;  
    jump over pile;  
}
```

Code
Block




Loops - foreach

- Used to iterate over arrays

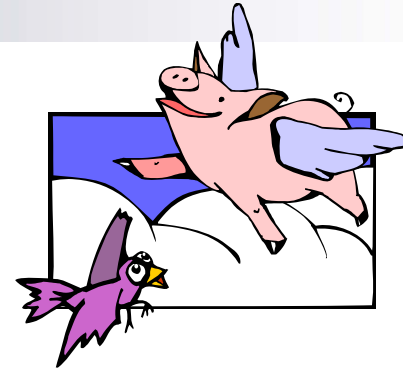
```
foreach($array as $value)
{
    print $value;
}
```

```
foreach($array as $key => $value)
{
    print $array[$key]; //value
    print $key;
    print $value;
}
```

Exercise


1. In your terminal type
`cd /var/www/workshop`
2. Type `gedit arrays.php` &
3. Open browser to 
<http://localhost/workshop>
4. Click `arrays.php`

If and Else



- Control structure – changes the flow of the script
- A way to choose between possible actions
- Tests whether an expression is true and *if* so, does the code immediately following
- Multiple if statements that are **mutually exclusive**, can be combined with **else if** and **else**

```
if ( condition is true )  
{  
    // do something  
}
```



```
$age = 21;
```

```
if ($age >= 17)
```

```
{
```

```
    print "You can drive!";
```

```
}
```

```
else if ($age >= 16)
```

```
{
```

```
    print "You are almost old enough to  
    drive!";
```

```
}
```

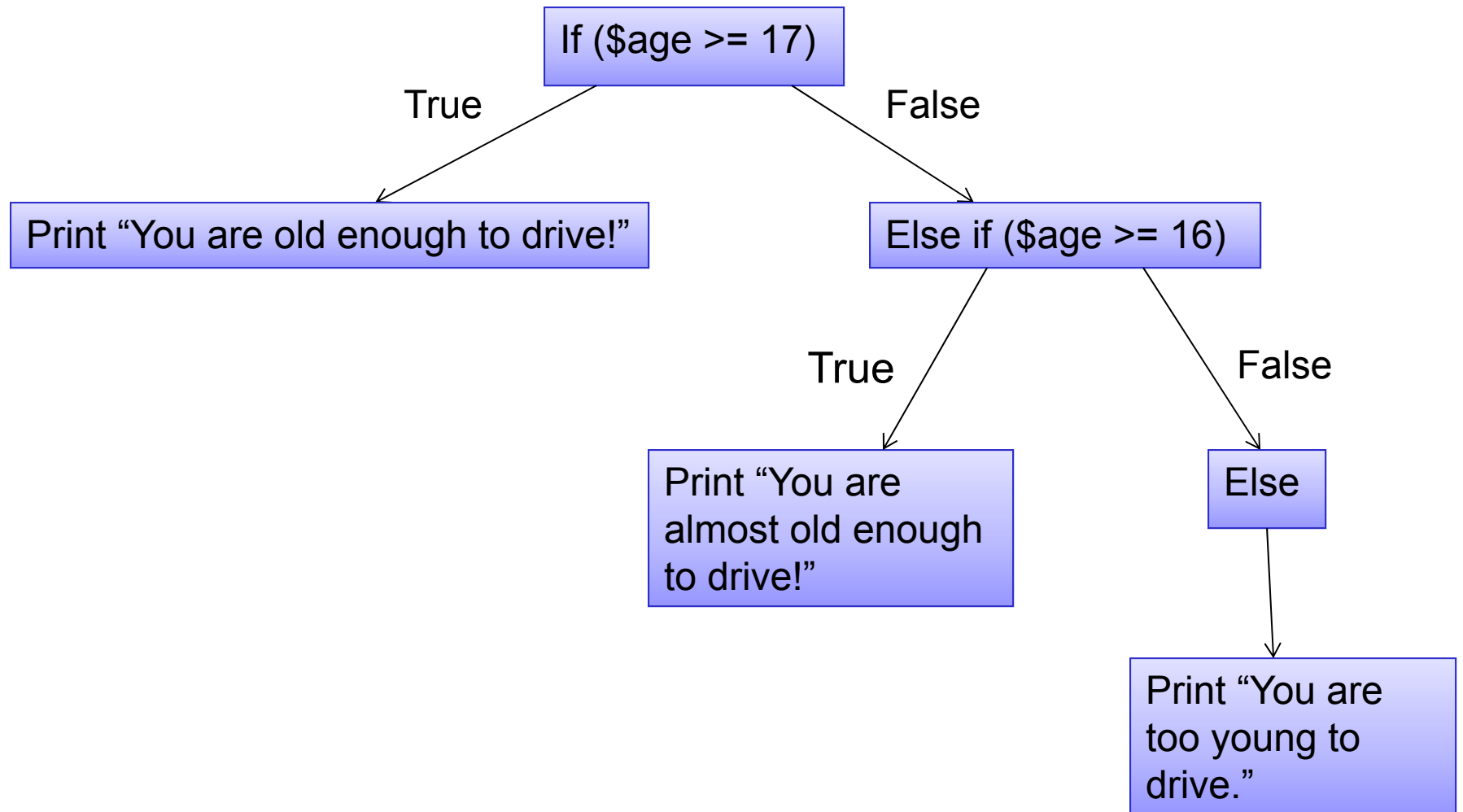
```
else
```

```
{
```

```
    print "You are not old enough to  
    drive."
```

```
}
```

If and Else Flowchart





Logical Operators

- Multiple conditionals can be combined using logical operators **AND** and **OR**
- Conditionals can be negated using the not logical operator - **!**


```
$age = 20;  
if ($age > 18 AND $age < 21) {  
    // go straight to college OR  
    // get a job  
}
```



Boolean Algebra

Input	Output
true AND true	true
true AND false	false
false AND false	false
true OR true	true
false OR true	true
false OR false	false
(true AND true) OR false	true

Exercise

1. In your terminal type
`cd /var/www/workshop`
2. Type `gedit boolean.php` &
3. Open browser to 
<http://localhost/workshop>
4. Click `boolean.php`

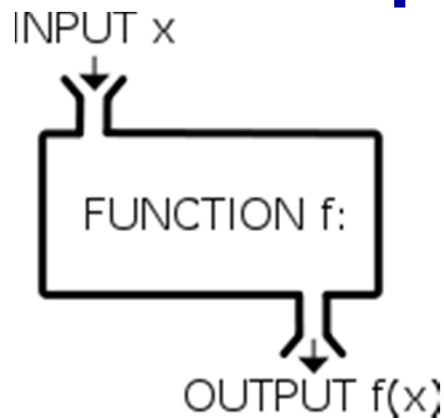


Review

What's wrong with it?

```
$radius = 5
if($radius >= 5); {print $radius;}
$myarray == array(6,7,8,9);
foreach ($myarray as $value {
    print $myarray;}
if ($age = 18) { print $age }
print "She said "hi";
for(i=0; i<5; i++){print "*" ;}
```

Functions



http://www.youtube.com/watch?v=_ve4M4UsJQo&feature=related

- A function f takes an input, x , and returns an output $f(x)$.
- Like a "machine" or "black box" that converts the input into the output.



Functions

- Piece of code that performs a task you want to do multiple times, or share with others
- A function is called by specifying
 - **function name**
 - Any **inputs** it may require
- It may **return a value (output)**

```
exit(); // simple function call  
print "This won't show up";
```



Functions – User Defined

```
//function declaration
function name( parameters )
{
    // do something
    // return a variable or value
}
```


- Function names follow the same rules as variables
 - Letters, numbers, underscore
 - Must start with a letter or underscore

Input Arguments



- In the function **code**, you use “formal” parameters.
 - Variable-like names, start with \$
- In function **call** you use “actual” parameters
 - Can be a variable or a value

Let's take a look.



```
// arg: $number a number to square
// return: the square of the $number
function square($input)
{
    $number_squared = $input*$input;
    return $number_squared;
}
```

```
$number = square(10);
print $number;
$number = square($number);
//what happens?
print $number;
```

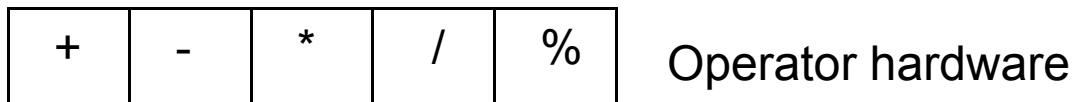
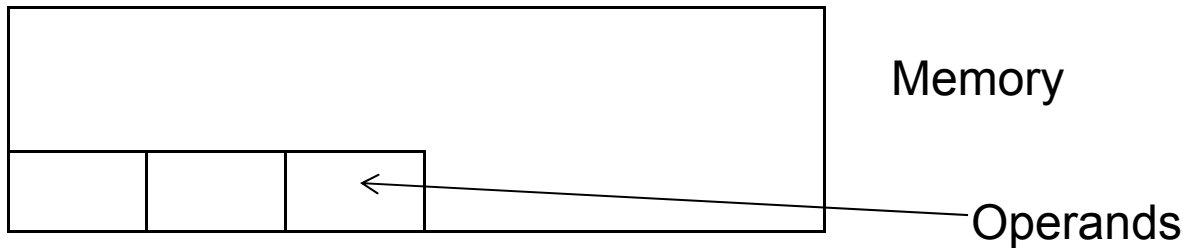
Computer Architecture

- Why do formal parameters and actual parameters have different names?

```
$y = array(2, 4, 6);
```

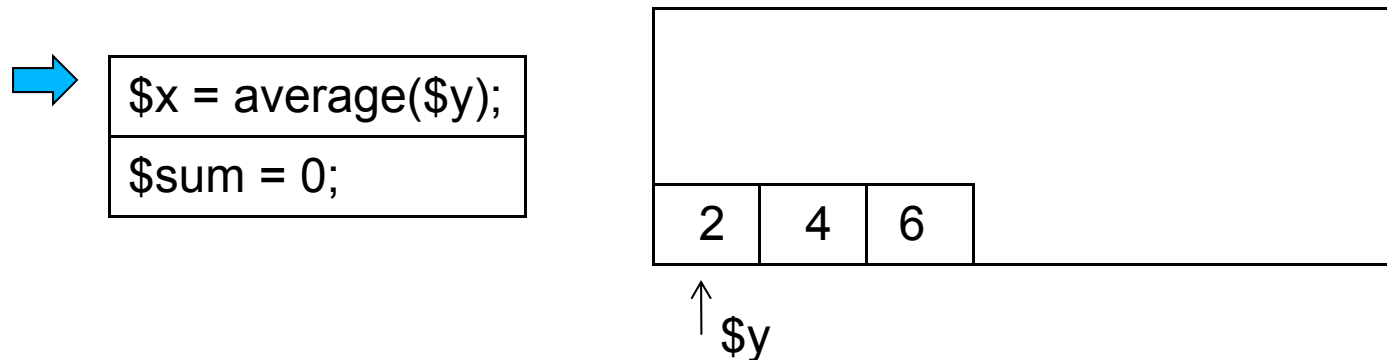
Command Stack

```
$x = average($y);
```



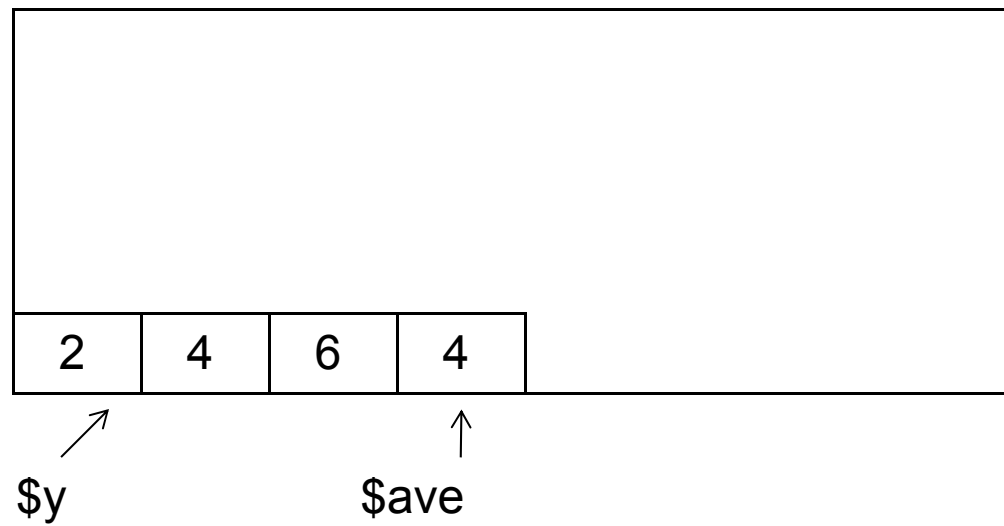
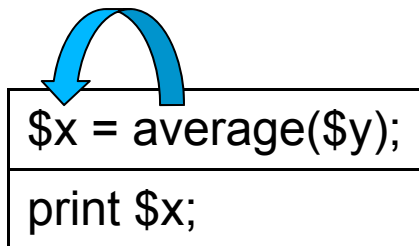
Function Calls

- 1) Save your place in the program.
- 2) Fetch the parameter values.
- 3) Fetch the function commands.
- 4) Run the function using **local** memory addresses



Function Call

- 5) Put the return value in memory
- 6) Do the assignment statement
- 7) Continue in program





Sharing Your Functions

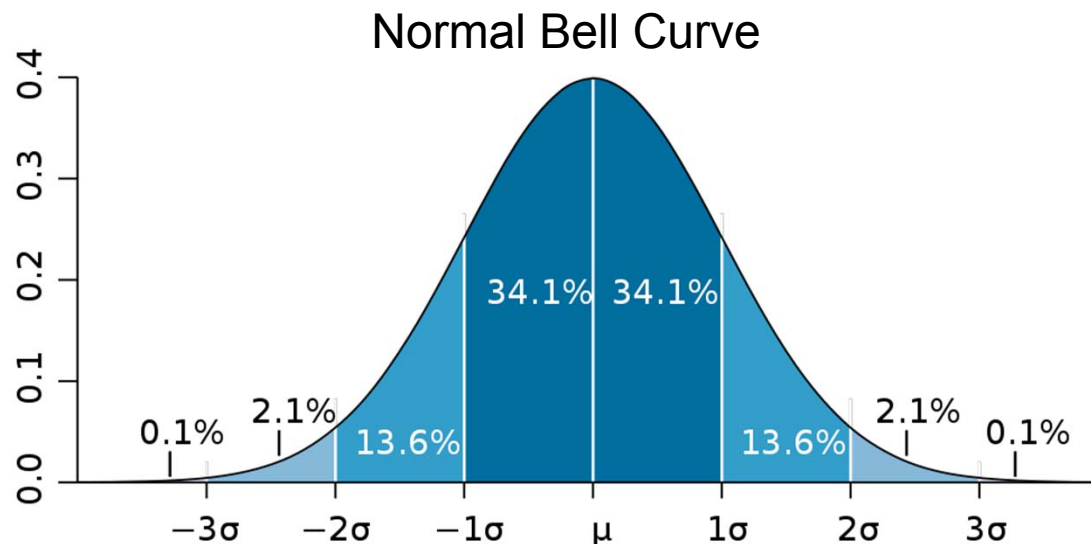
1. Copy your function into a separate php file, inside `<?php ?>`, e.g. `myfunction.php`
2. In an HTML file, just within `<?php`

```
require_once  
( 'myfunction.php' );
```

Exercise:

Summary Statistics

You'll be writing functions to calculate average, and, if you wish standard deviation.



What is this useful for?



Formulas

$$\text{Average} = (1/\text{number_of_things}) * (\text{sum_of_all_things})$$

$$\text{Stdev} = \sqrt{[(1/\text{number_of_things}) * \text{sum_of_all} ((\text{thing} - \text{average})^2)]}$$

http://en.wikipedia.org/wiki/Standard_deviation



What is it?

```
if ($color == "blue")
    { print $color; }
$blue = "001";
$number = square(10);
foreach($line as $key => $val)
    {print $key;}
print_r($array);
for($i=0; $i<4; $i++ )
    {$i = $i- 1;}
function silly($thing)
    { return "silly $thing";}
$silliness = silly("wabbit");
```



Parsing

intransitive verb

1 : to give a grammatical description of a word or a group of words

- Scientific research produces data
 - Numbers
 - Strings describing RNA sequences
- The data gets saved in files
- If we parse the files we can do analysis or visualization



Some helpful functions for parsing in PHP

```
// open and read a file into an array of
lines

$lines =
    file('playlist.tsv', FILE_SKIP_EMPTY_LINES);

//turn tab-separated string into array
$arr = explode("\t", $mystring);

// turn array into a comma-separated string
$line = implode(",", $myarray);
```



Helpful Array functions

```
print_r($my_array);  
    // no return value
```

```
$array_length = count($my_array);  
    // returns length of array
```


Jukebox!!

1. In your terminal type
cd /var/www/workshop
2. Type gedit parsing.php &
3. Open browser to
<http://localhost/workshop/>
4. Click parsing.php



Doubly Nested Loops and Tables

```
foreach($row as $rkey => $rvalue){  
    $cols = $rvalue;  
    foreach($cols as $ckey => $cvalue){  
        print $cvalue;  
    }  
}
```

1	2	3
4	5	6