# Study Unit 7/8: SERVER-SIDE PROGRAMMING (PHP)

## Outline

- Server-Side Basics • Dynamic web page
- Introduction to PHP
- PHP Basic Syntax.

## Learning Outcomes of Study Unit 7/8

Students will learn the basic concepts of web Static and Dynamic pages and server side languages. PHP Lifecycle.

Variables Types, Arrays and loops and PHP syntax

1.1: Server-Side Basics • Dynamic web page

- Server-side programming

1.2: Introduction to PHP

- Lifecycle of PHP Web Request
- PHP code must be executed!

1:3 PHP Basic Syntax

- Comments, print/echo

- Variables, types, int/float, arithmetic operators

---

- bool, NULL
- String, string functions, interpreted strings
- Array, array functions
- for, if/else, while, foreach
- Math functions
- PHP syntax template

## Outline

- **1.1: Server-Side Basics**
- Introduction to PHP
- PHP Basic Syntax

# URLs and Web Servers

```
http://www.aw-bc.com:80/info/regesstepp/index.html
~~~  ~~~~~~~~~~~~~~    ~~~ ~~~~~~~~~~~~~~~~~~~
protocol     host              port         path
```

- Usually when you type a URL in your browser:
  - Your computer looks up the server's IP address using DNS
  - Your browser connects to that IP address and requests the given file
  - The web server software (e.g. Apache) grabs that file from the server's local file system, and sends back its content to you
- Some URLs actually specify **programs** that the web server should run, and then send their output back to you
- as the result: http://php.net/manual/en/function.sqrt.php
  - The above URL tells the server php.net to run the program manual/en/function.sqrt.php and send back its output

## Dynamic Vs. Static

- Static Page
  - Client/Consumer’s viewpoint: a URL referring to an identical HTML file
  - Server/Producer’s viewpoint: a file stored within or sub-within the root folder of a Web Server
  - It is an HTML…
  - Can be displayed directly in a browser
- Dynamic Page
  - Client/Consumer’s viewpoint: a URL referring to a dynamic HTML (may vary each time requested)
  - Server/Producer’s viewpoint: a program/script produces HTML
  - It is **NOT an HTML**, but a program producing HTML(s)
  - Can’t be displayed directly in a browser
- Dynamic Web Page vs. Dynamic HTML ( DHTML )

## Server-Side Web Programming

- Server-side pages are programs written by one of many web programming languages/frameworks
  - i.e. 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

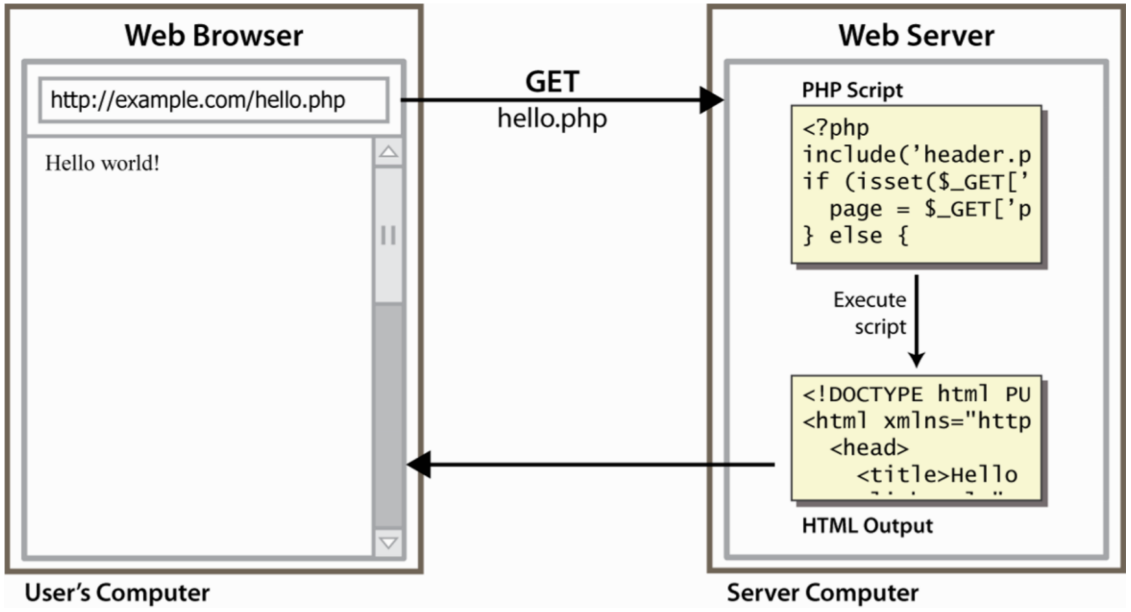
## Outline

- Server-Side Basics
- **1.2: Introduction to PHP**
- PHP Basic Syntax

## What is PHP?

- PHP stands for "PHP Hypertext Preprocessor" ● A server-side scripting language ● Used to make web pages dynamic:
  - Provide different contents depending on context ● Interface with other services: database, e-mail, etc.
  - Authenticate users
  - Process form information
- PHP code can be embedded in XHTML code

# Lifecyle of PHP Web Request



```
<?php
include('header.p
if (isset($_GET['
  page = $_GET['p
} else {
```

```
<!DOCTYPE html PU
<html xmlns="http
  <head>
    <title>Hello
```

- Browser requests an HTML file (**static content**): server just sends that file
- Browser requests a PHP file (**dynamic content**): server reads it, runs any script code inside it, then sends result across the network
    - Script produces output as the response sent back

## 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
  - **Compatibility**: supported by most popular web servers
  - **Simplicity**: lots of built-in functionality; familiar syntax
  - **Availability**: already installed on most commercial web hosts

## An Introduction To PHP

- PHP set its roots in 1995, when an independent software development contractor named Rasmus Lerdorf developed a Perl/CGI script that enabled him to know how many visitors were reading his online resume.
- Lerdorf thus began giving away his toolset, called Personal Home Page (PHP), or Hypertext Preprocessor.
- PHP is an embedded server-side Web-scripting language that provides developers with the capability to quickly and efficiently build dynamic Web applications.
- PHP can also serve as a valuable tool for creating and managing dynamic content, embedded directly beside JavaScript, Stylesheets and many others.
- PHP talk to various database systems, giving you the ability to generate a web page based on a SQL query.
- PHP provides hundreds of predefined functions and capable of handling anything a developer can dream of.
- Graphic creation and manipulation
- Mathematical calculations
- Ecommerce
- Extensible Markup Language (XML)
- Open database connectivity (ODBC)

## PHP: Hypertext Preprocessor

- PHP is a scripting language that brings websites to life in the following ways:
  - Sending feedback from your website directly to your mailbox
  - Sending email with attachments
  - Uploading files to a web page
  - Watermarking images
  - Generating thumbnails from larger images
  - Displaying and updating information dynamically
  - Using a database to display and store information
  - Making websites searchable
  - And much more.

## Processing PHP

- PHP is a server-side language. This means that the web server(apache or IIS) processes your PHP code and sends only the results usually as XHTML to the browser.
- Because all the action is on the server, you need to tell it that your pages contain PHP code. This involves two simple steps, namely:
    - Give every page a PHP filename extension the default is .php.
    - Enclose all PHP code within PHP tags

## PHP

- HTML and PHP can be used interchangeably as needed, working alongside one another in harmony. With PHP, we can simply do the following:

```
<html>
<title><?php print "Hello world!"; ?></title>
</html>
```

- Hello world! will be displayed in the Web page title bar. Interestingly, the single line *print* is enclosed in PHP's escape characters (<?… ?>) is a complete program.

## Running PHP

**Working with php files**

- Create a folder called test in **htdocs**
- Create a file index.php
- Open a browser, type localhost/test

Note: xampp server must be started and running

## Index.php

```
<html>
<title><?php print "Hello world!"; ?></title>
<body>
<?php echo “Welcome to PHP programming"; ?>
</body>
</html>
```

## Display of HTML using PHP

```
<html>
<body>
<?
// Notice how HTML tags are included in the print statement.
echo "<h3>PHP/HTML integration is cool.</h3>";
?>
</body>
</html>
```

## Dynamic date insertion

```
<? echo date("F d, Y"); ?>
<?php echo date(‘H:i:s’);?>
```

## Comments

- // or # Single-line comment. Everything to the end of the current line is ignored.
- /* ... */ Single- or multiple-line comment. Every thing between /* and */ is ignored.

## THE BIG PICTURE

- A typical PHP page will use some or all of the following elements:
  - Variables to act as placeholders for unknown or changing values
  - Arrays to hold multiple values
  - Conditional statements to make decisions
  - Loops to perform repetitive tasks
  - Functions to perform preset tasks

## Variables

- A **variable** is simply a name that you give to something that may change or that you don't know in advance.
- A variable always begins with a dollar sign ($). The following are all valid variables:

```
$color
$operating_system
$_some_variable
$model
```

## Naming variables

- You can choose just about anything you like as the name for a variable, as long as you keep the following rules in mind:
  - Variables always begin with a dollar sign ($).
  - The first character after the dollar sign cannot be a number.
  - No spaces or punctuation are allowed, except for the underscore (_).
  - Variable names are case-sensitive: $startYear and $startyear are not the same.

## Variable Scope

Scope can be defined as the range of availability a variable has to the program in which it is declared. PHP variables can be one of four scope types:

- Local variables:

  A variable declared in a function is considered local; that is, it can be referenced solely in that function. Any assignment outside of that function will be considered to be an entirely different variable from the one contained in the function.

```
$x = 4;
function assignx () {
        $x = 0;
        print "\$x inside function is $x. <br>";
}
```

- Function parameters

  As is the case with many other programming languages, in PHP any function that accepts arguments must declare these arguments in the function header.

  // multiply a value by 10 and return it to the caller

```
$value = 10;
function multi($value) {
        $value = $value * 10;
        return $value;
}
```

- Global variables

  In contrast to local variables, a global variable can be accessed in any part of the program. However, in order to be modified, a global variable must be explicitly declared to be global in the function in which it is to be modified

```
$somevar = 15;
function addit() {
        GLOBAL $somevar;
        $somevar++;
        print "Somevar is $somevar";
}
addit();
```

- Static variables

  In contrast to the variables declared as function parameters, which are destroyed on the function's exit, a static variable will not lose its value when the function exits and will still hold that value should the function be called again.
  You can declare a variable to be static simply by placing the keyword STATIC in front of the variable name.

```
STATIC $somevar;
```

## String Assignments

- Strings can be delimited in two ways, using either double quotations ("") or single quotations ('').
- The following two string declarations produce the same result:

```
$food = "meatloaf";
$food = 'meatloaf';
```

- However, the following two declarations result in two drastically different outcomes:

```
Echo $sentence = "My favourite food is $food";
Echo $sentence2 = 'My favourite food is $food';
```

- The following string is what exactly will be assigned to $sentence. Notice how the variable $food is automatically interpreted:

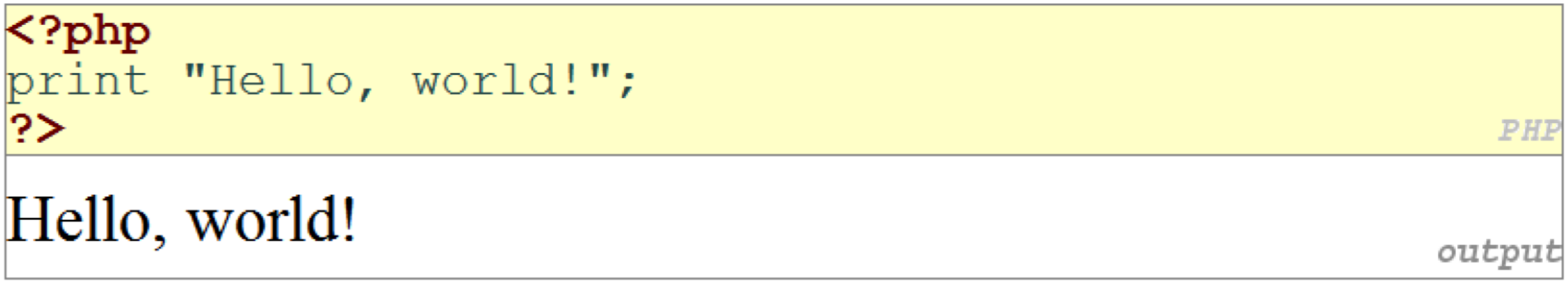
```
My favourite food is meatloaf.
```

- Whereas $sentence2 will be assigned the string as follows:

```
My favourite food is $food.
```

- These differing outcomes are due to the usage of double and single quotation marks in assigning the corresponding strings to $sentence and $sentence2.

## Hello, World!



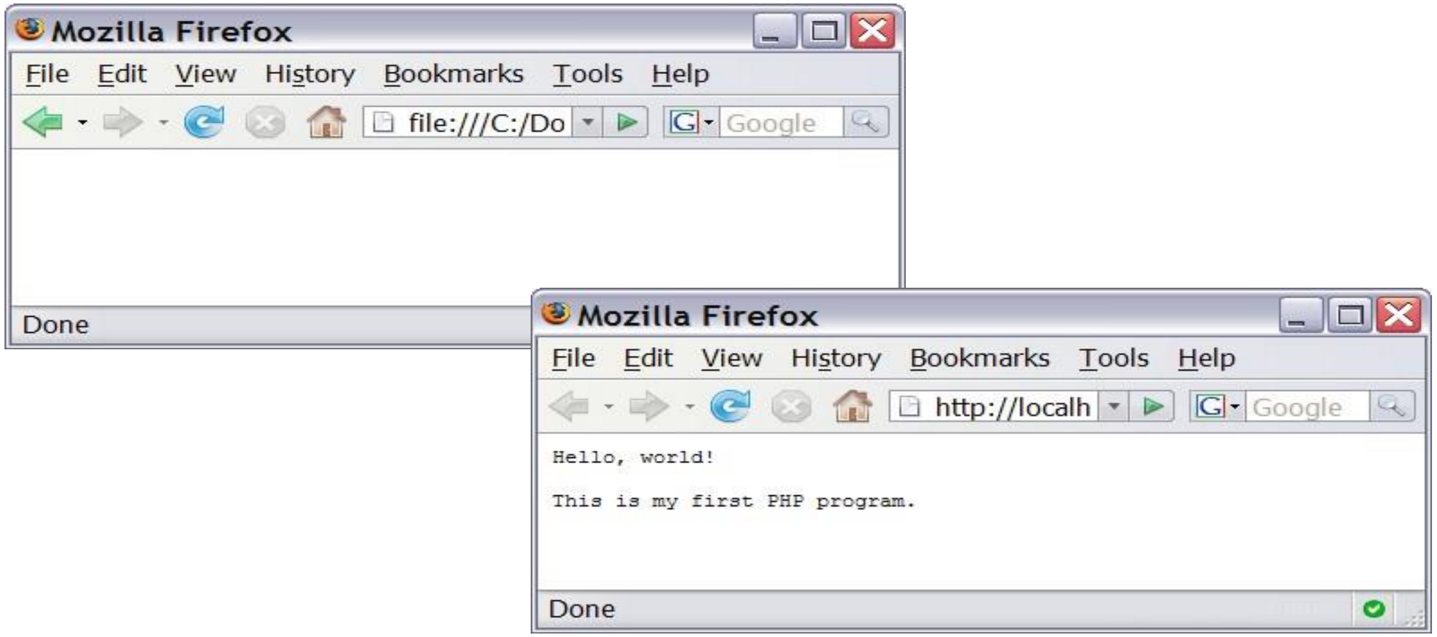
```php
<?php
print "Hello, world!";
?>
```

```
Hello, world!
```

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

## Viewing PHP Output

- **Your PHP code must be run/executed first, before it reaches a browser!**


Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///C:/Do
Google
Done
Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localh
Google
Done

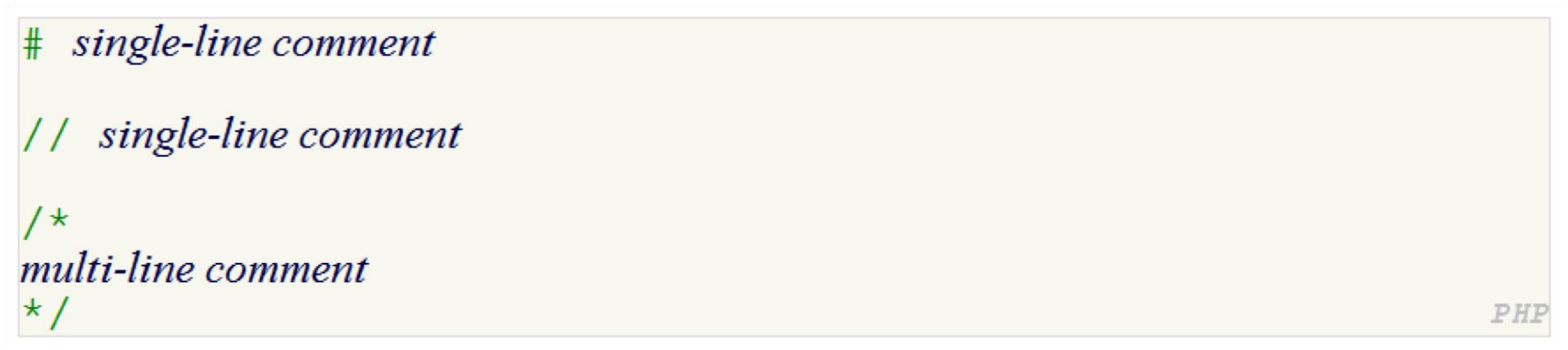
```
Hello, world!

This is my first PHP program.
```

## Outline

- Server-Side Basics
- Introduction to PHP
- **1.3: PHP Basic Syntax**

## Comments



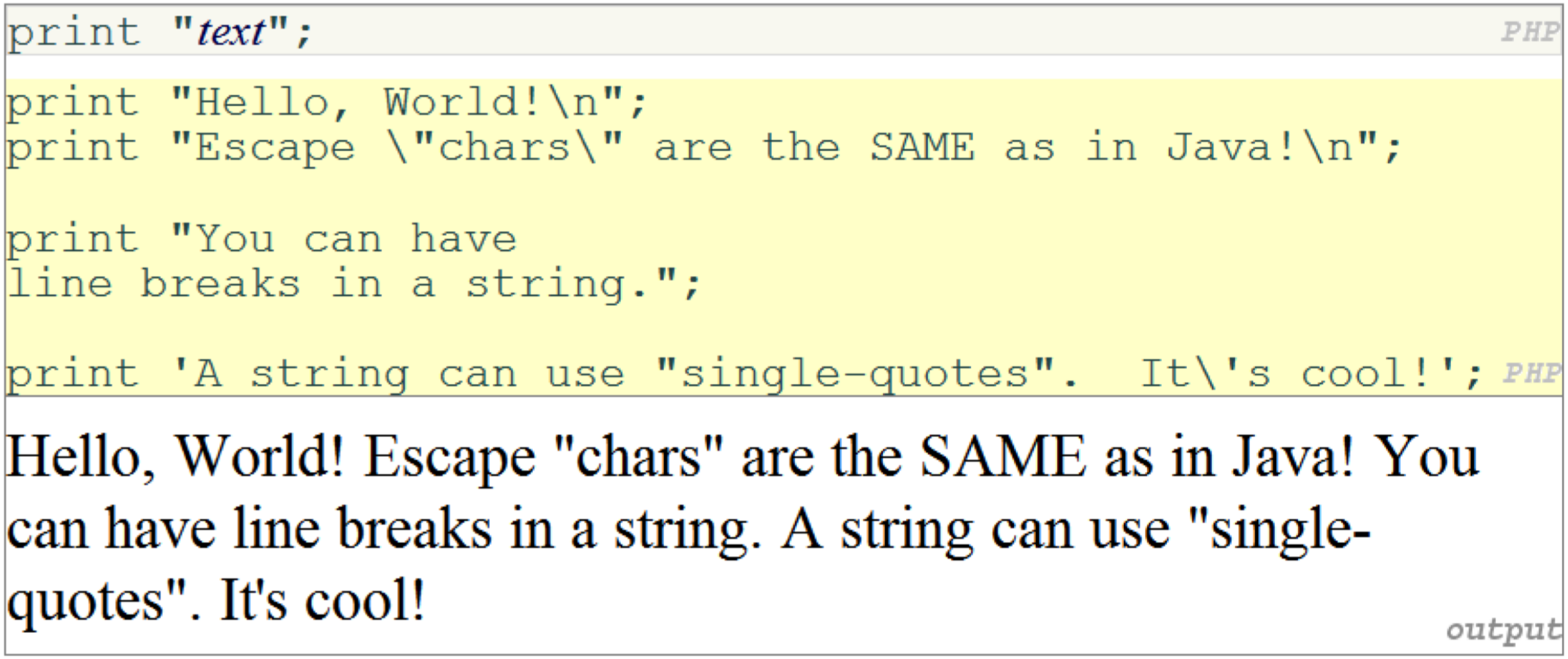
```
#   single-line comment

//   single-line comment

/*
multi-line comment
*/
```

- Like Java, but # is also allowed
  - A lot of PHP code use # comments instead of //

## Console Output: print



```php
print "text";
```

PHP

```php
print "Hello, World!\n";
print "Escape \"chars\" are the SAME as in Java!\n";

print "You can have
line breaks in a string.";

print 'A string can use "single-quotes".  It\'s cool!';
```

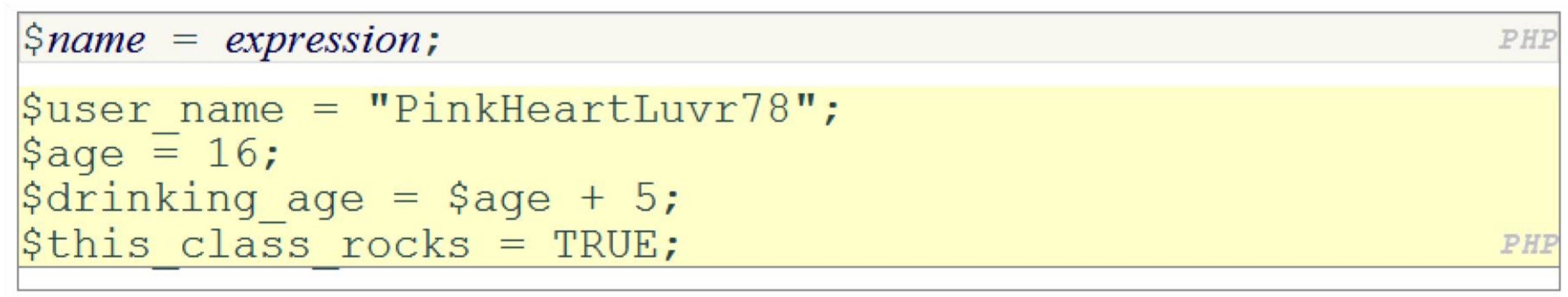
PHP

```
Hello, World! Escape "chars" are the SAME as in Java! You
can have line breaks in a string. A string can use "single-
quotes". It's cool!
```

output

- Some PHP programmers use the equivalent **echo** instead of **print**
  - Arguments of echo vs. print

## Variables



```
$name = expression;
```

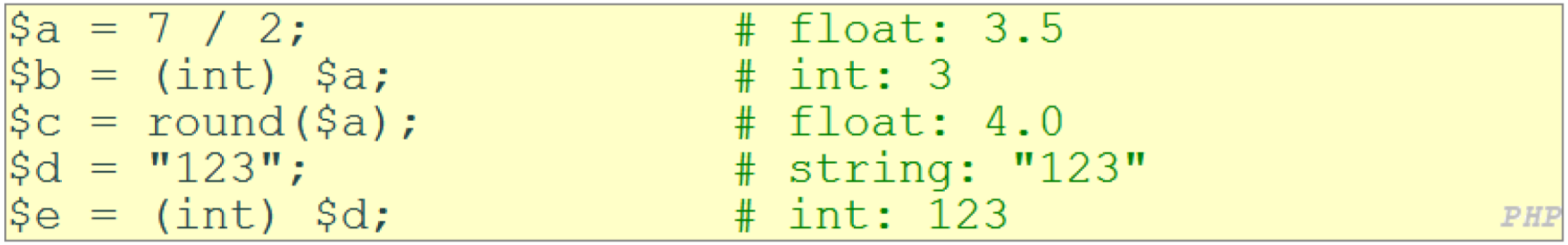
```
$user_name = "PinkHeartLuvr78";
$age = 16;
$drinking_age = $age + 5;
$this_class_rocks = TRUE;
```

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

## Types *Web 2.0 Programming*

- Basic types: int, float, boolean, string, array, object, NULL
  - Test what type a variable is with `is_type` functions, i.e. `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 /
- Explicit type-casting with (*type*):
  - $age = (*int*) "21";

## int and float Types



```
$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`

## Arithmetic Operators

Web 2.0 P rogramming

- `+ - * / % . ++ --`
  `= += -= *= /= %= .=`

- Many operators auto-convert types: `5 + "7"` is `12`

## bool (Boolean) Type

```php
$feels_like_summer = FALSE;
$php_is_rad = TRUE;

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

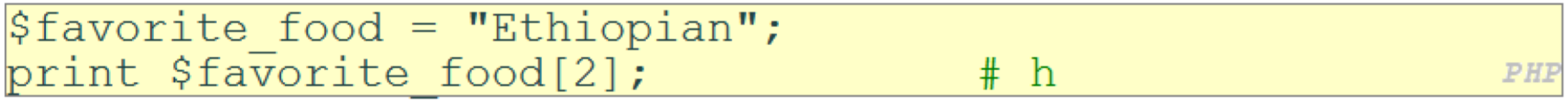
- The following values are considered to be **FALSE** (all others are **TRUE**):
  - 0 and 0.0 (but NOT 0.00 or 0.000)
  - "", "0", and **NULL** ( includes unset variables )
  - arrays with 0 elements
- Can cast to boolean using (**bool**)
- **FALSE** is printed as an empty string (no output); **TRUE** is printed as a “1”

## NULL

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

- 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** is printed as an empty string (no output)

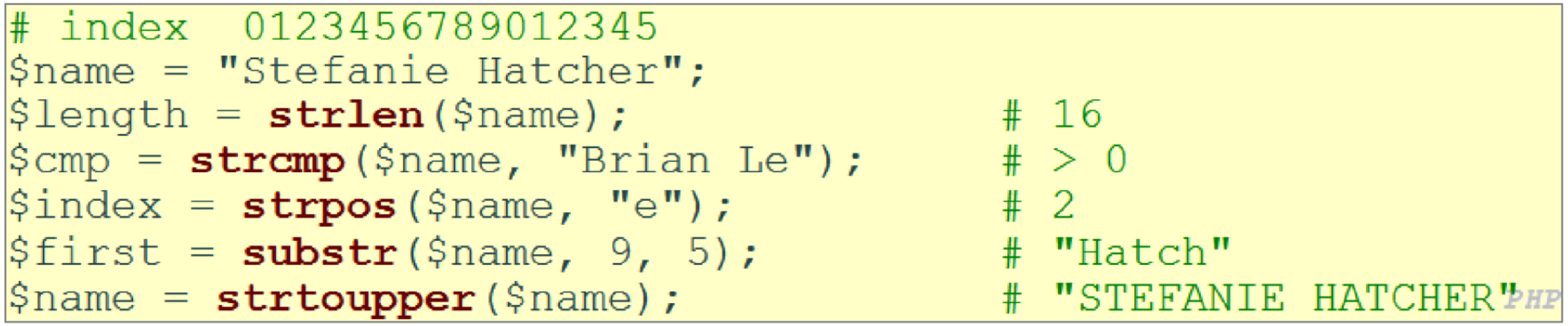
## String Type



```
$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"
- Can be specified with " " or ' '

## String Operations



```
# index  0123456789012345
$name = "Stefanie Hatcher";
$length = strlen($name);                  # 16
$cmp = strcmp($name, "Brian Le");         # > 0
$index = strpos($name, "e");              # 2
$first = substr($name, 9, 5);             # "Hatch"
$name = strtoupper($name);                # "STEFANIE HATCHER"
```

| Name | Java Equivalent |
|---|---|
| strlen | length |
| strpos | indexOf |
| substr | substring |
| strtolower, strtoupper | toLowerCase, toUpperCase |

| | |
|---|---|
| trim | trim |
| explode, implode | split, join |
| strcmp | compareTo |

## Interpreted Strings

- Strings inside **" "** are interpreted, and variables inside a **" "** string will have their values inserted into the string

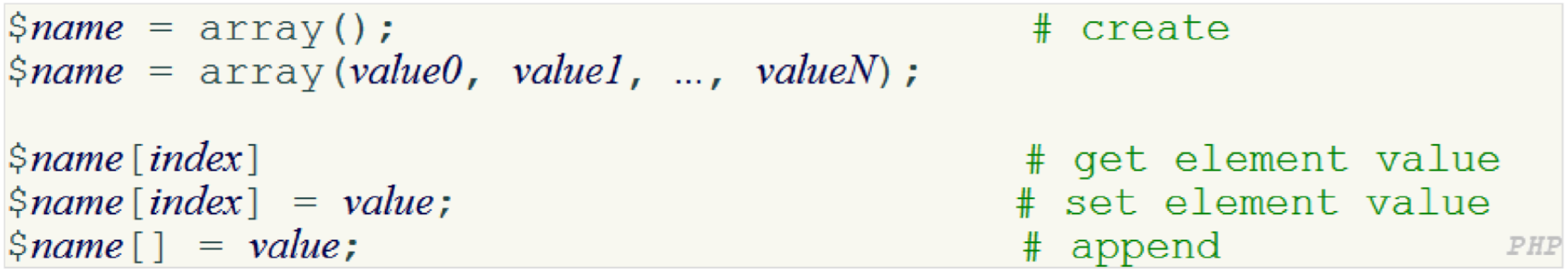
```
$age = 16;
~~print "You are " . $age . " years old.\n";~~
print "You are $age years old.\n";     # You are 16 years old. PHP
```

- Strings inside **' '** are *not* interpreted:

```
print 'You are $age years old.\n';     # You are $age years old.\n  PHP
```

## Arrays

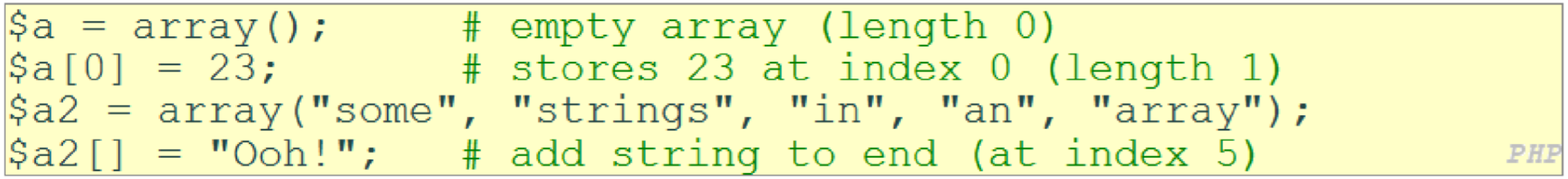
- T



```
$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)
```

PHP

d, use bracket notation without specifying an index

- Element type is not specified; mixed types are allowed

## Array Functions

| Function Name(s) | Description |
|---|---|
| count | number of elements in the array |
| print_r | print array's content |
| 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

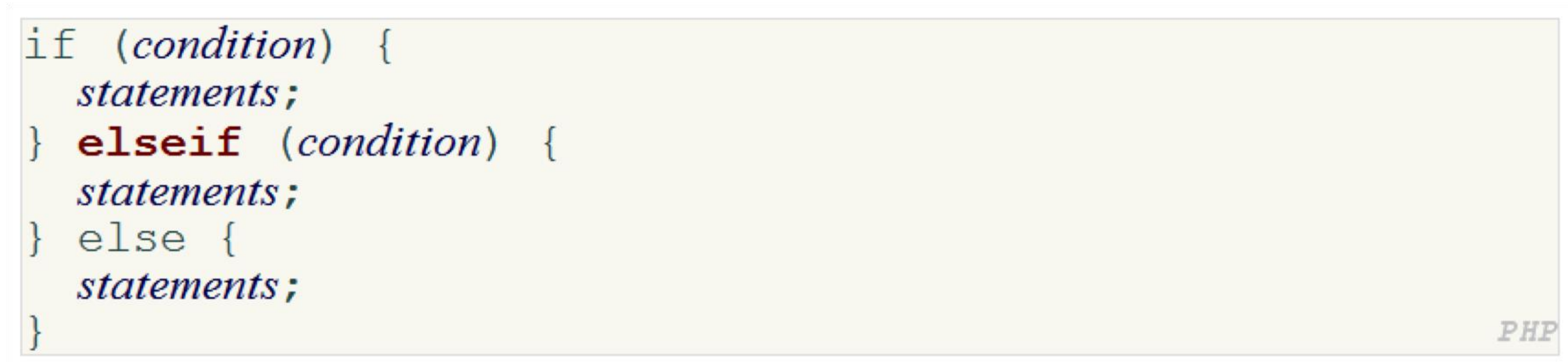
- The array in PHP acts as many other collections in Java
  - List, stack, queue, set, map, ...

## for Loop (same as C)

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

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

## if/else Statement



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

PHP

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

## while Loop (same as C)

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

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

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

## The foreach Loop

- A convenient way to traverse each element of an array without indexes

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

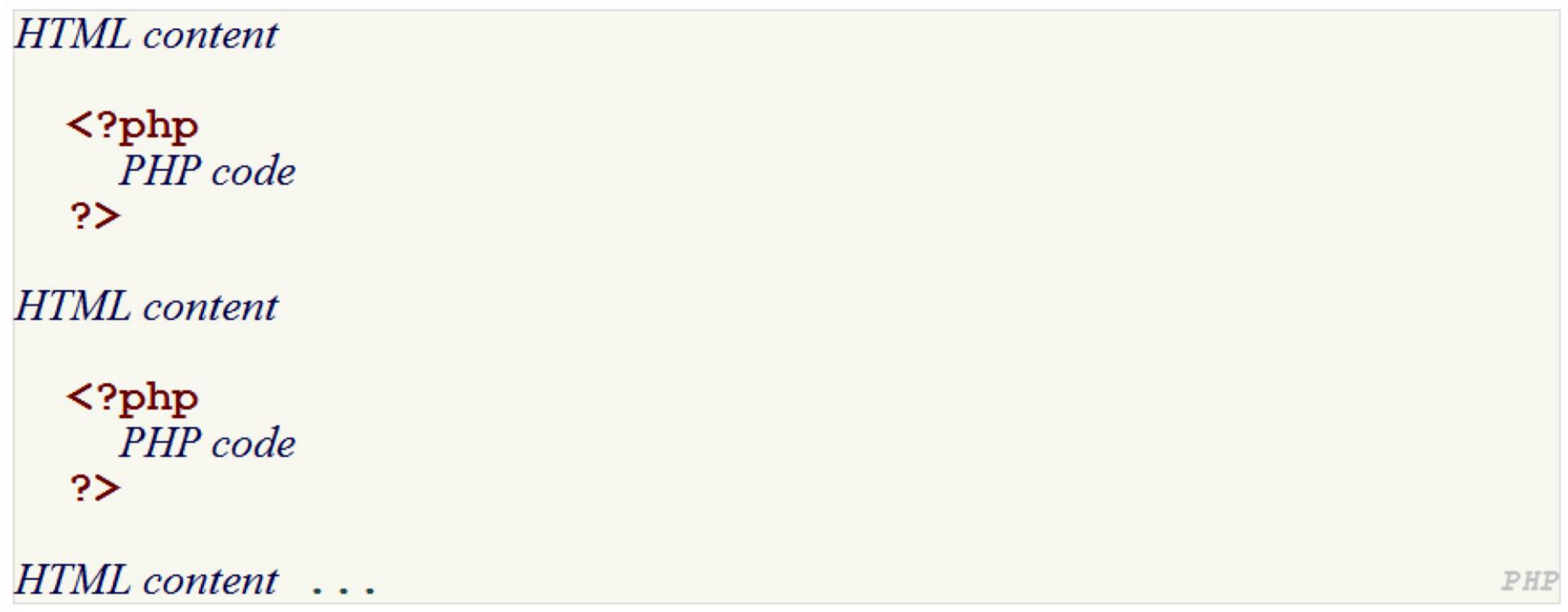
```
$stooges = array("Larry", "Moe", "Curly", "Shemp");
~~for ($i = 0; $i < count($stooges); $i++) {~~
~~  print "Moe slaps {$stooges[$i]}\n";~~
~~}~~
foreach ($stooges as $stooge) {
  print "Moe slaps $stooge\n";  # even himself!
}
```
PHP

## Math Operations

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

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | | |
| | | | | | | |

## PHP Syntax Template



```
HTML content

   <?php
      PHP code
   ?>

HTML content

   <?php
      PHP code
   ?>

HTML content  . . .
```

- Any contents of a .php file between **<?php** and **?>** are executed as PHP code
- All other contents are printed as pure HTML
- Can switch back and forth between HTML and PHP "modes"

## Questions For Study Session 7/8

Now that you have completed this study unit, you can assess how well you have achieved its Learning Outcomes by answering these questions. Write your answers in your Study Diary and discuss them with your Tutor at the next Study Support Meeting or Online interactive sessions. You can also check your answers at the Self-Review Answers section which is located at the end of this Module.

| |
|---|
| 1: Draw a UML sequence diagram of interactions between a Web browser and a PHP Web server when the browser requests a PHP page on the server. |
| 2: Write a PHP code snippet to figure out the day of today. |
| 3: Why PHP is a widely used programming language in the world. |
| 4: Create a PHP file that display numbers from 1-50 on a new line using arrays and loops. (while / do while/ for / foreach loops. |

## Self-Review Answers (SRA) for Study Unit 7/8

1: Draw a UML sequence diagram of interactions between a Web browser and a PHP Web server when the browser requests a PHP page on the server.
See above slides
2: Write a PHP code snippet to figure out the day of today.

The PHP `date()` function formats a timestamp to a more readable date and time.

### Syntax

date(*format*,*timestamp*)

| Parameter | Description |
|---|---|
| format | Required. Specifies the format of the timestamp |
| timestamp | Optional. Specifies a timestamp. Default is the current date and time |

```
<!DOCTYPE html>
<html>
<body>
<p>Result is :</p>
<?php
echo "Today is " . date("Y/m/d") . "<br>";
echo "Today is " . date("Y.m.d") . "<br>";
echo "Today is " . date("Y-m-d") . "<br>";
echo "Today is " . date("l");
?>

</body>
</html>
```

```
Result is:
Today is 2020/09/26
Today is 2020.09.26
Today is 2020-09-26
Today is Saturday
```

3: Why PHP is a widely used programming language in the world.

For answer See above slides

4: Create a PHP file that display numbers from 1-50 on a new line using arrays and loops. (while / do while/ for / foreach loops).

```
<?php
// while loop The while loop executes a block of code as long as the specified condition is true.
// while (condition is true) {
//     code to be executed;
//   }
$i = 1;
while($i <= 5){
    echo 'it is while loop'.$i.'<br>';
    $i++;
}
```

```
$x = 1;
/*The do...while loop will always execute the block of code once, it will then check the condition,
 and repeat the loop while the specified condition is true. */
do {
  echo "<b style='color:red'>The do while number is: $x </b><br>";
  $x++;
} while ($x <= 5);
?>
<br>
<?php
#The for loop is used when you know in advance how many times the script should run.
// for (init counter; test counter; increment counter) {
//     code to be executed for each iteration;
//   }
for($y= 1; $y<= 5; $y++){
    echo "it is for loop $y\n<br>";
}
//The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.
// foreach ($array as $value) {
//     code to be executed;
//   }
$colors = array(1,2,3,4,5);

foreach ($colors as $value) {
  echo "it is a foreach loop index $value <br>";
}
?>
```

## References and Additional Reading Materials

- PHP home page:
- W3Schools PHP tutorial: https://www.w3schools.com/php/default.asp
- Practical PHP Programming: http://hudzilla.org/phpwiki/
- https://www.php.net/