中山大學
SUN YAT-SEN UNIVERSITY

Lecture 6
Basic PHP for Server Side Programming

SE-805 Web 2.0 Programming (supported by Google)

http://my.ss.sysu.edu.cn/courses/web2.0/

School of Software, Sun Yat-sen University

Outline

- **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

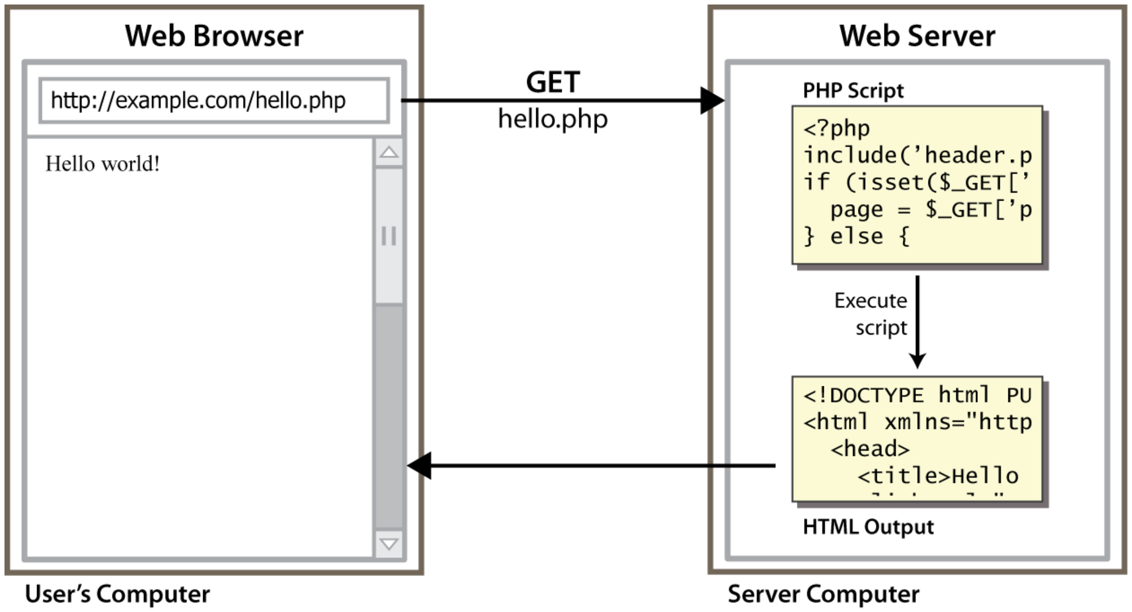
Outline

- Server-Side Basics
- **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

Hello, World!

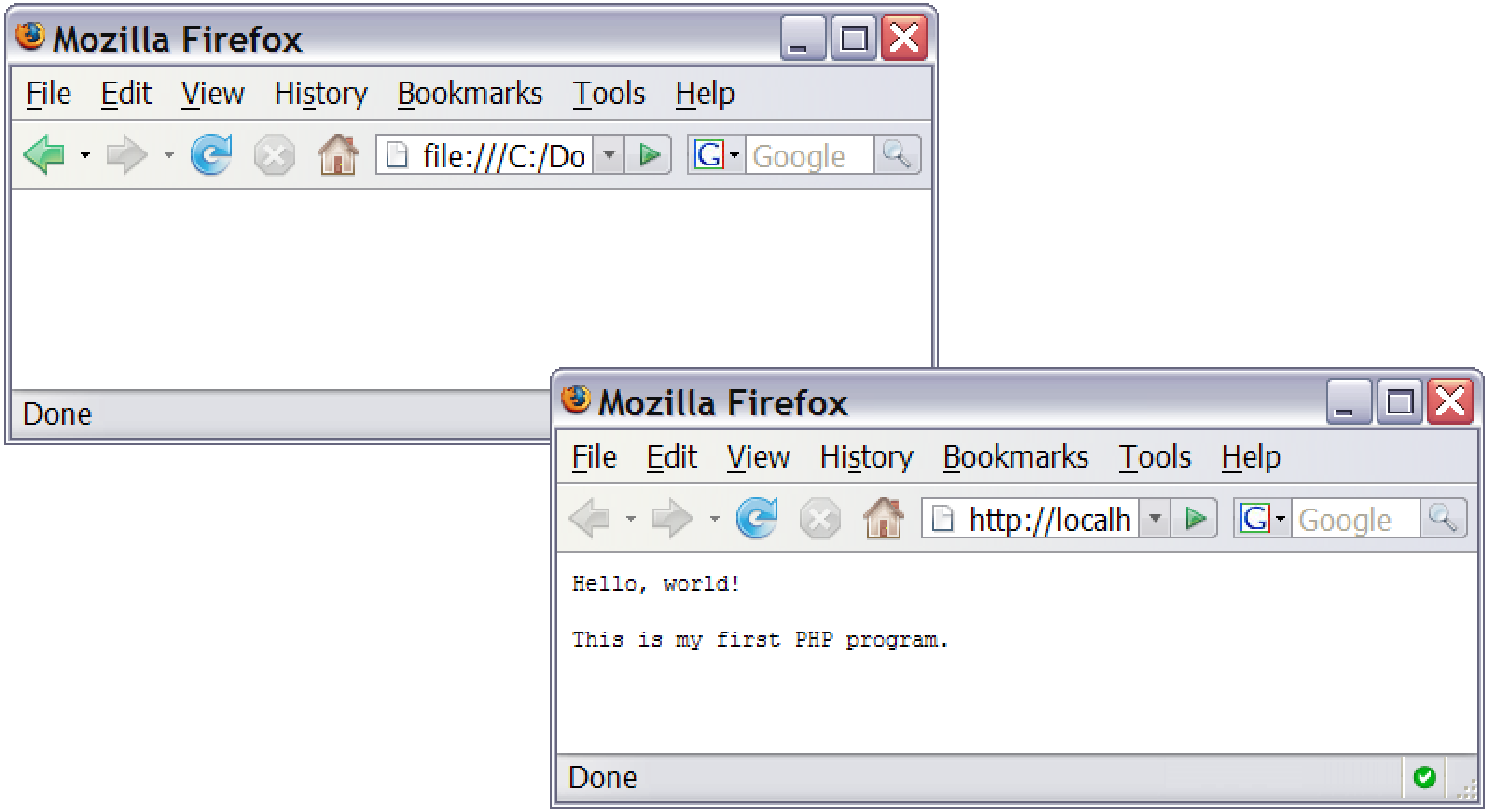
```php
<?php
print "Hello, world!";
?>
```
PHP

```
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

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

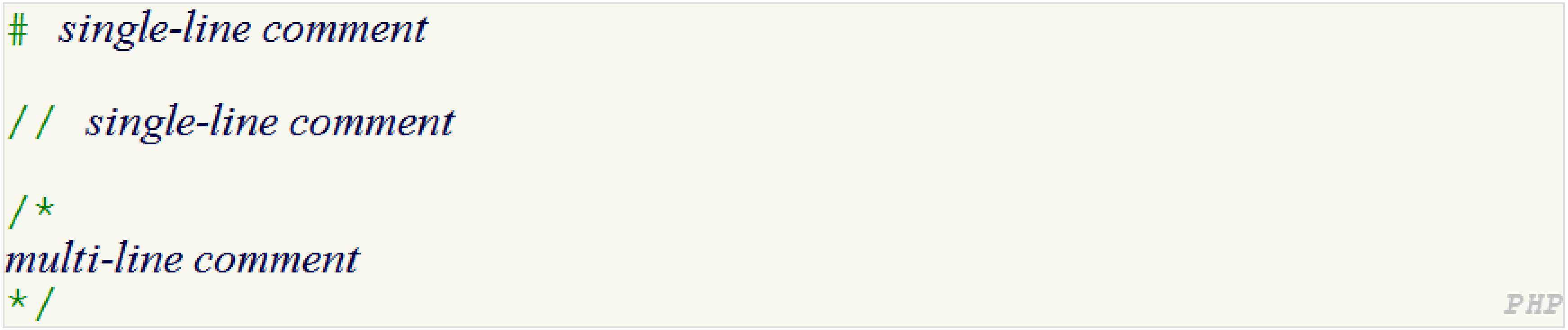
```
Hello, world!

This is my first PHP program.
```

Outline

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

Comments



```
#  single-line comment

//  single-line comment

/*
multi-line comment
*/
```

PHP

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

Console Output: print

```
print "text";                                                        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;                                      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 (**type is not written**)
- A weak-typing language (like JavaScript or Python)

Types

- 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

```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
```

PHP

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

Arithmetic Operators

- + - * / % . ++ --
  = += -= *= /= %= .=
- Many operators auto-convert types: 5 + "7" is 12

bool (Boolean) Type

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

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

- 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

```php
$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           PHP
```

- 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"
```
PHP

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

```
$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

- To append, 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)*

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

```
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 traverse each element of an array without indexes

Math Operations

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

abs	**ceil**	**cos**	**floor**	**log**	**log10**	**max**
min	pow	rand	round	sin	sqrt	tan

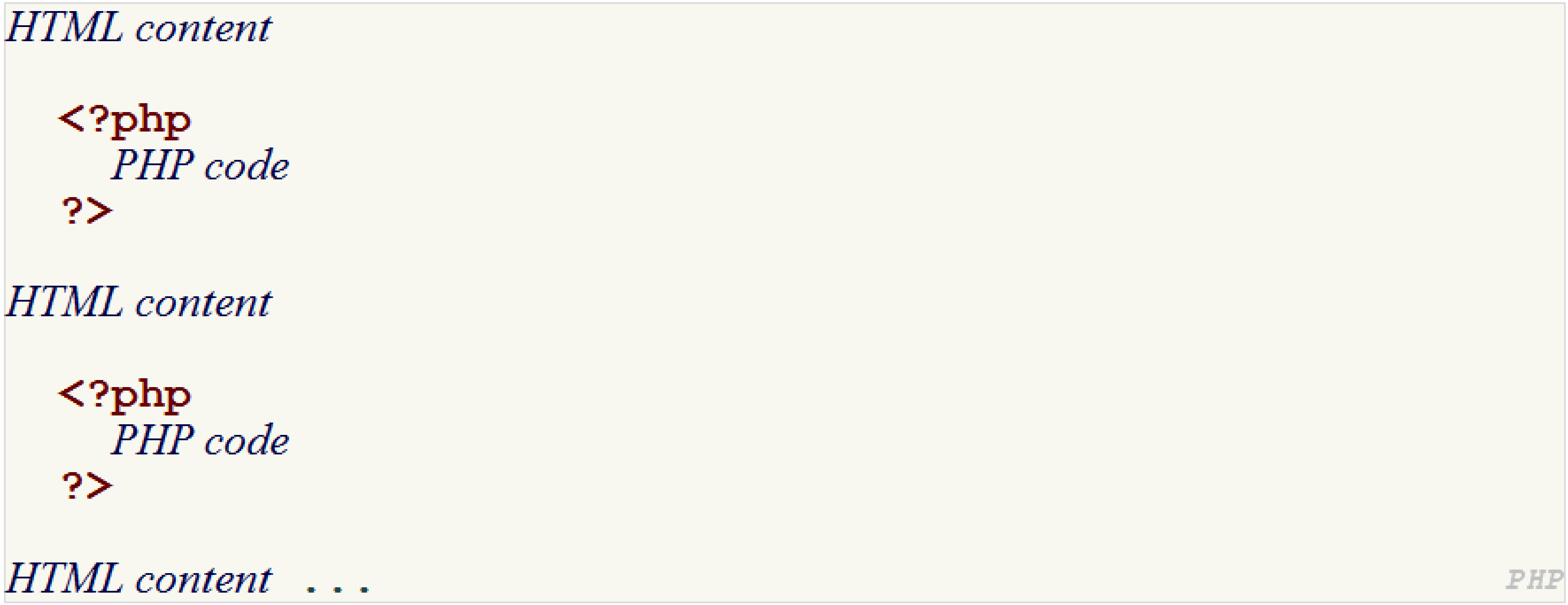
math functions

M_PI	**M_E**	**M_LN2**

math constants

- The syntax for method calls, parameters, returns is the same as Java and C

PHP Syntax Template



```
HTML content

   <?php
      PHP code
   ?>

HTML content

   <?php
      PHP code
   ?>

HTML content   . . .
```

PHP

- 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"

Summary

- Server-Side Basics
  - Dynamic web page
  - Server-side programming
- Introduction to PHP
  - Lifecycle of PHP Web Request
  - PHP code must be executed!

Summary

- 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

Exercises

- 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
- Write a PHP code snippet to calculate and output the first 20 Fibonacci numbers
- Write a PHP code snippet to figure out the day of today

Further Readings

- PHP home page: http://www.php.net/
- W3Schools PHP tutorial: http://www.w3schools.com/PHP/
- Practical PHP Programming: http://hudzilla.org/phpwiki/
- PHP Cookbook: http://commons.oreilly.com/wiki/index.php/PHP_Cookbook

Thank you!