Lecture #11

NEWM N510: Web-Database Concepts

PHP (3)

kharrazi@iupui.edu
http://www.info510.com

Review Last Lecture

- PHP: String Manipulations
- PHP: Array Manipulations

PHP in a Nutshell

1. PHP Intro
2. PHP Syntax
3. PHP *echo*
4. PHP Comment
5. PHP Variables
6. PHP String/Array Manipulation
7. PHP Conditions
8. PHP Loops
9. PHP Functions
10. PHP Cookies/Sessions
11. PHP SSI
12. PHP Forms
13. PHP/MySQL Integration

Lecture in a Nutshell

1. PHP Conditions
2. PHP Loops
3. PHP Functions

1. PHP Conditions

- Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this.

- In PHP we have two conditional statements:

  – *if (...else) statement* - use this statement if you want to execute a set of code when a condition is true (and another if the condition is not true)

  – *switch statement* - use this statement if you want to select one of many sets of lines to execute

PHP Conditions (cont.)

- Comparison Operators:

Operator	**Description**	**Example** ($x=7; $y = 12)	**Result**
==	is equal to	$x==$y;	false
!=	is not equal	$x!=$y;	true
>	is greater than	$x>$y;	false
<	is less than	$x<$y;	true
>=	is greater than or equal to	$x>=$y;	false
<=	is less than or equal to	$x<=$y;	true

PHP Conditions (cont.)

- Logical Operators:

Operator	**Description**	**Example** ($x=7; $y = 12)	**Result**
&&	and	$x==7 && $y==12;	true
		$x==7 && $y>20;	false
		$x>20 && $y>20;	false
||	or	$x==7 || $y==12;	true
		$x==7 || $y>20;	true
		$x>20 || $y==12;	true
		$x>20 || $y>20;	false
!	not	$x==7 && !($y>20);	true

PHP Conditions (cont.)

- *The If Statement*: If you want to execute some code if a condition is true and another code if a condition is false, use the if....else statement.

- [] syntax means that is optional.

```
if (condition)
{
        code to be executed if condition is true;
}
[else
{
        code to be executed if condition is false;
}]
```

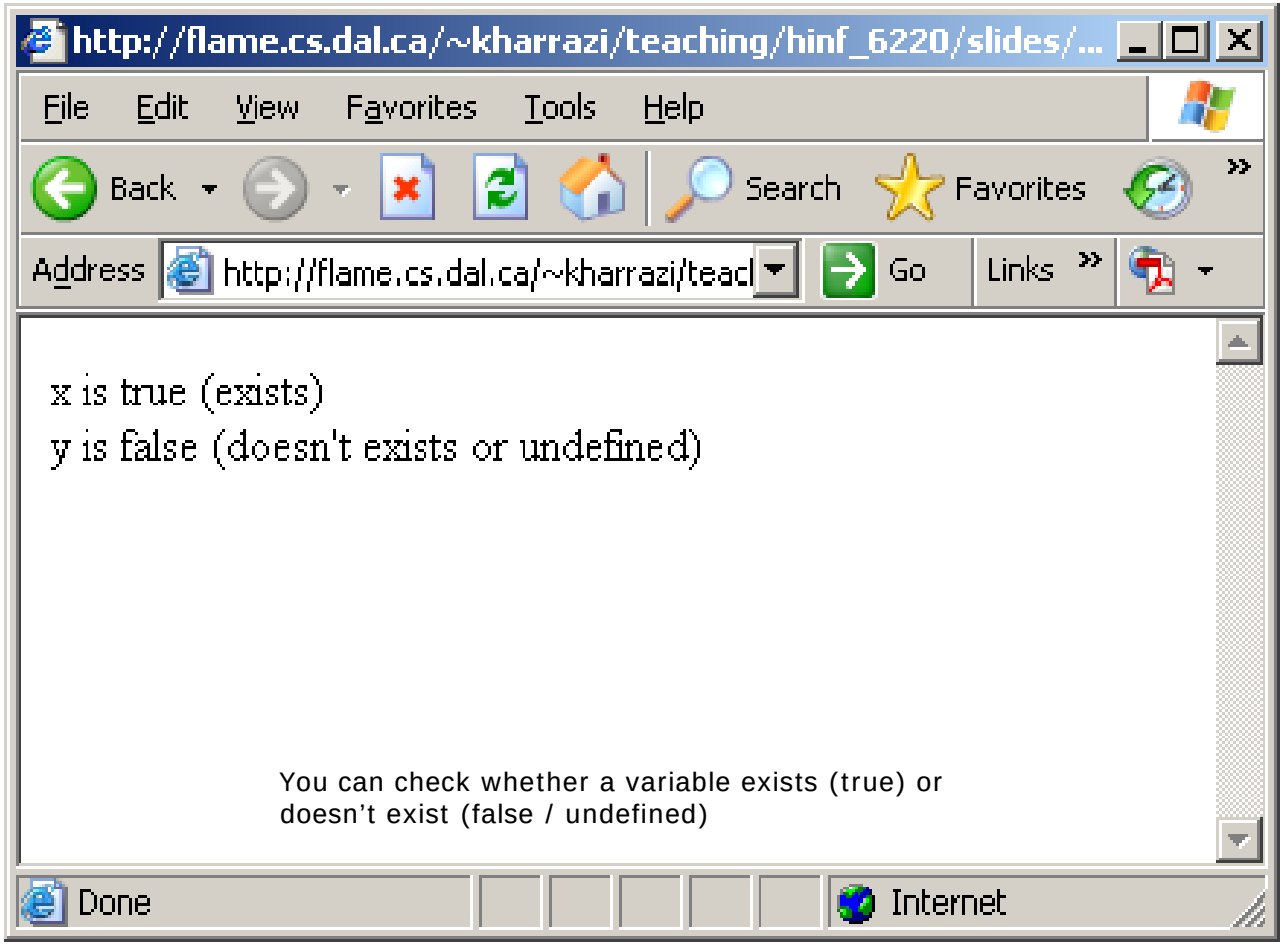
PHP Conditions (cont.)

if

```
<?php
    $x = true;
    // IF true conditional statement
    if ($x){
      echo "x is true (exists)";
    }
    // IF false conditional statement
    if (!$x){
      echo "x is false (doesn't exists or undefined)";
    }
    // IF false conditional statement
    if (!$y){
      echo "<br> y is false (doesn't exists or undefined)";
    }
?>
```

PHP Conditions (cont.)

if



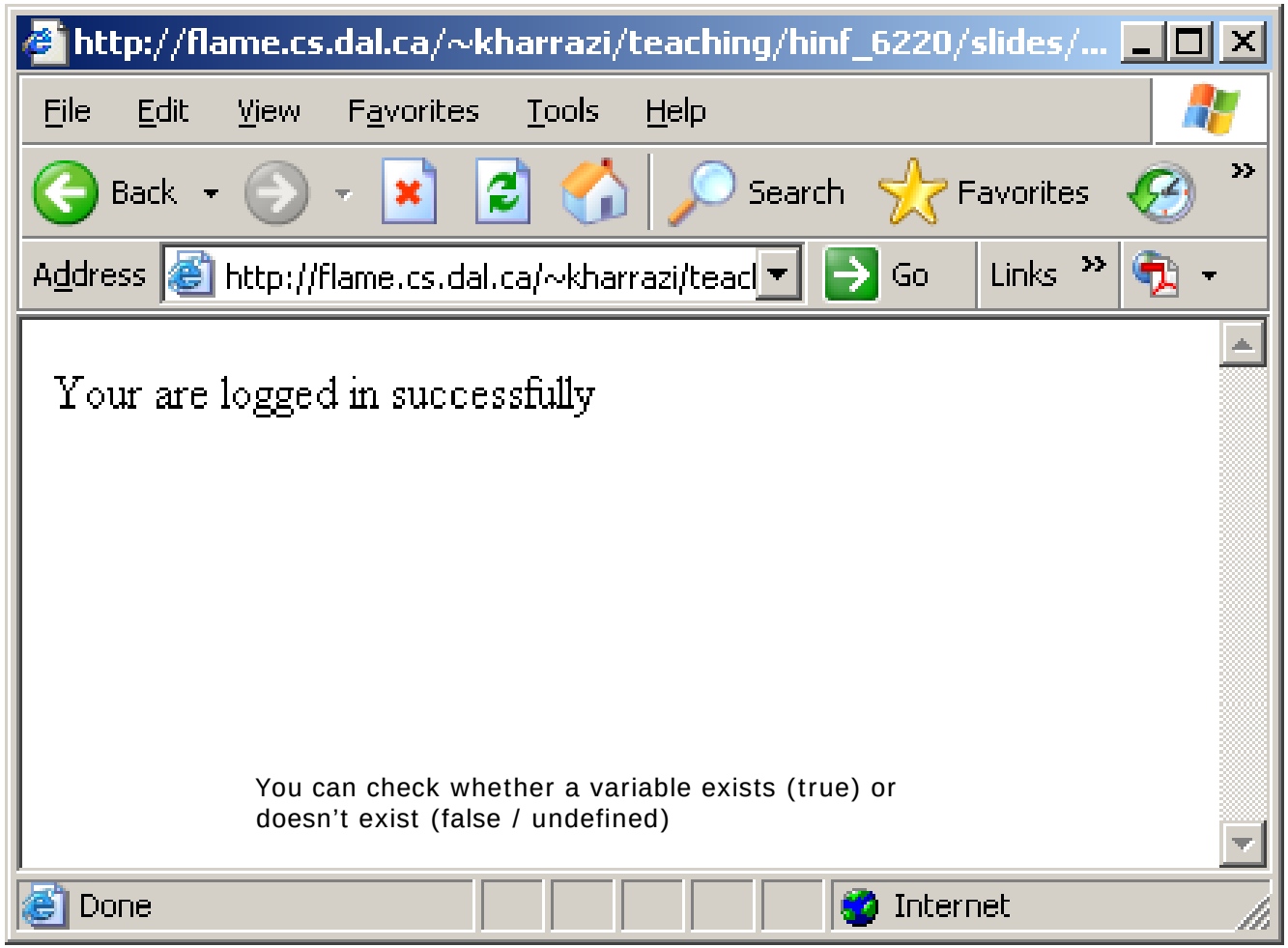
PHP Conditions (cont.)

if

```php
<?php
    $username = "Peter";

    // IF true conditional statement
    if ($username){
      echo "Your are logged in successfully";
    }

    // IF false conditional statement
    if (!$username){
      echo "Your are not logged in";
    }
?>
```

PHP Conditions (cont.)

if



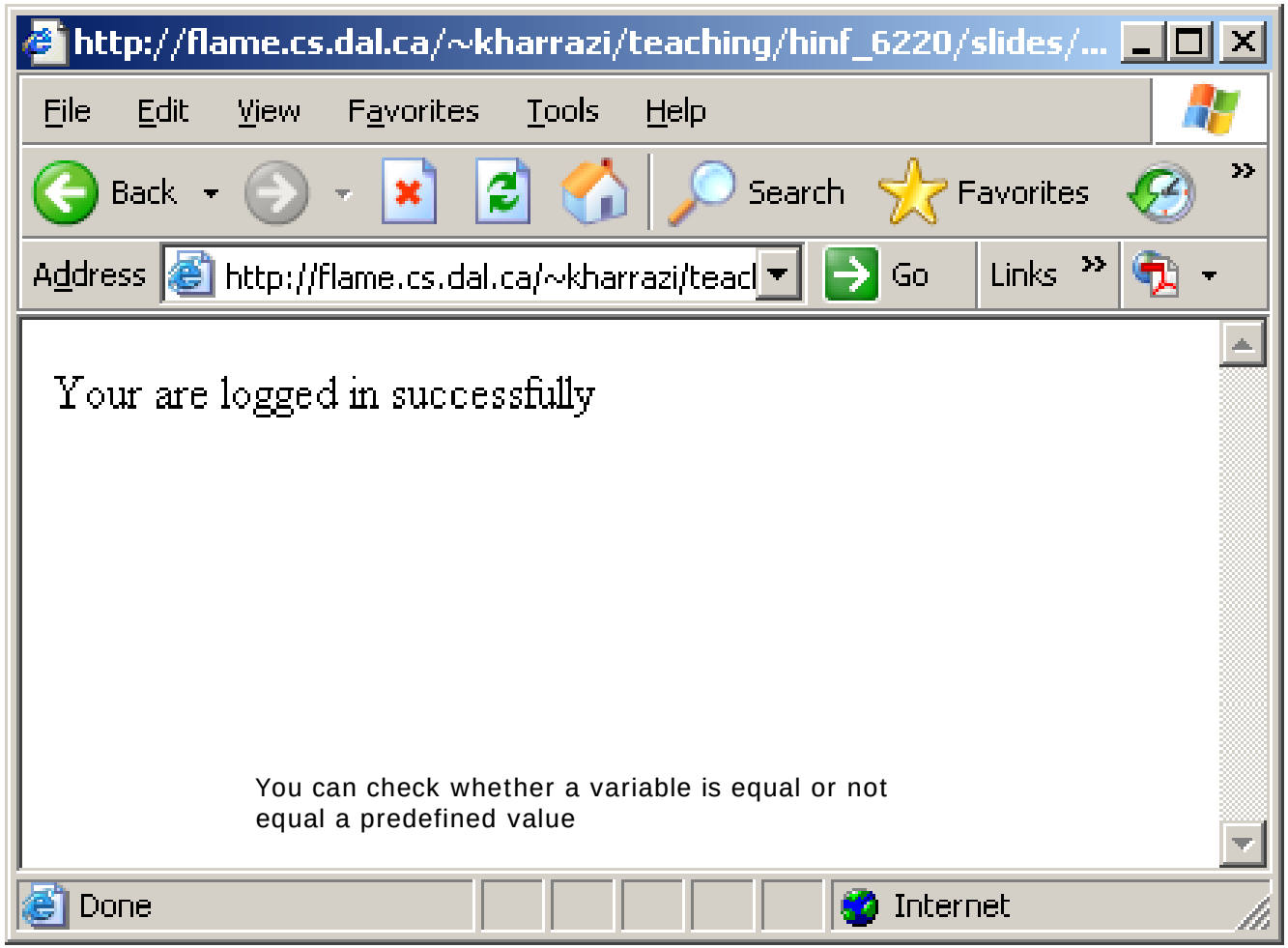
PHP Conditions (cont.)

if

```php
<?php
    $username = "Peter";

    // IF true conditional statement
    if ($username == "Peter"){
      echo "Your are logged in successfully";
    }

    // IF false conditional statement
    if ($username != "Peter"){
      echo "Your are not logged in";
    }
?>
```

PHP Conditions (cont.)

if



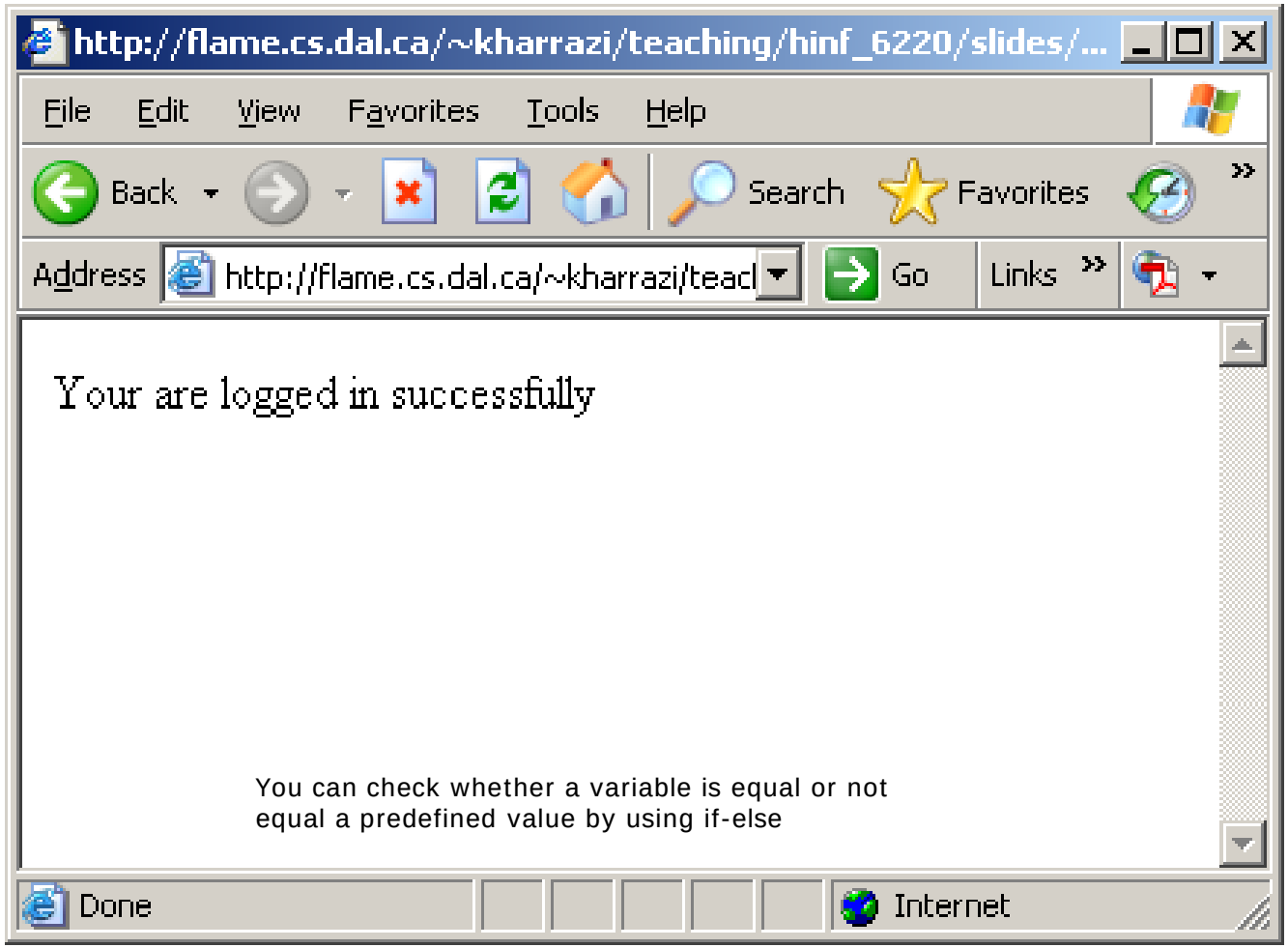
PHP Conditions (cont.)

if/else

```
<?php
     $username = "Peter";

     // IF true/false conditional statement
     if ($username == "Peter"){
       echo "Your are logged in successfully";
     }else{
       echo "Your are not logged in";
     }
?>
```

PHP Conditions (cont.)

if/else



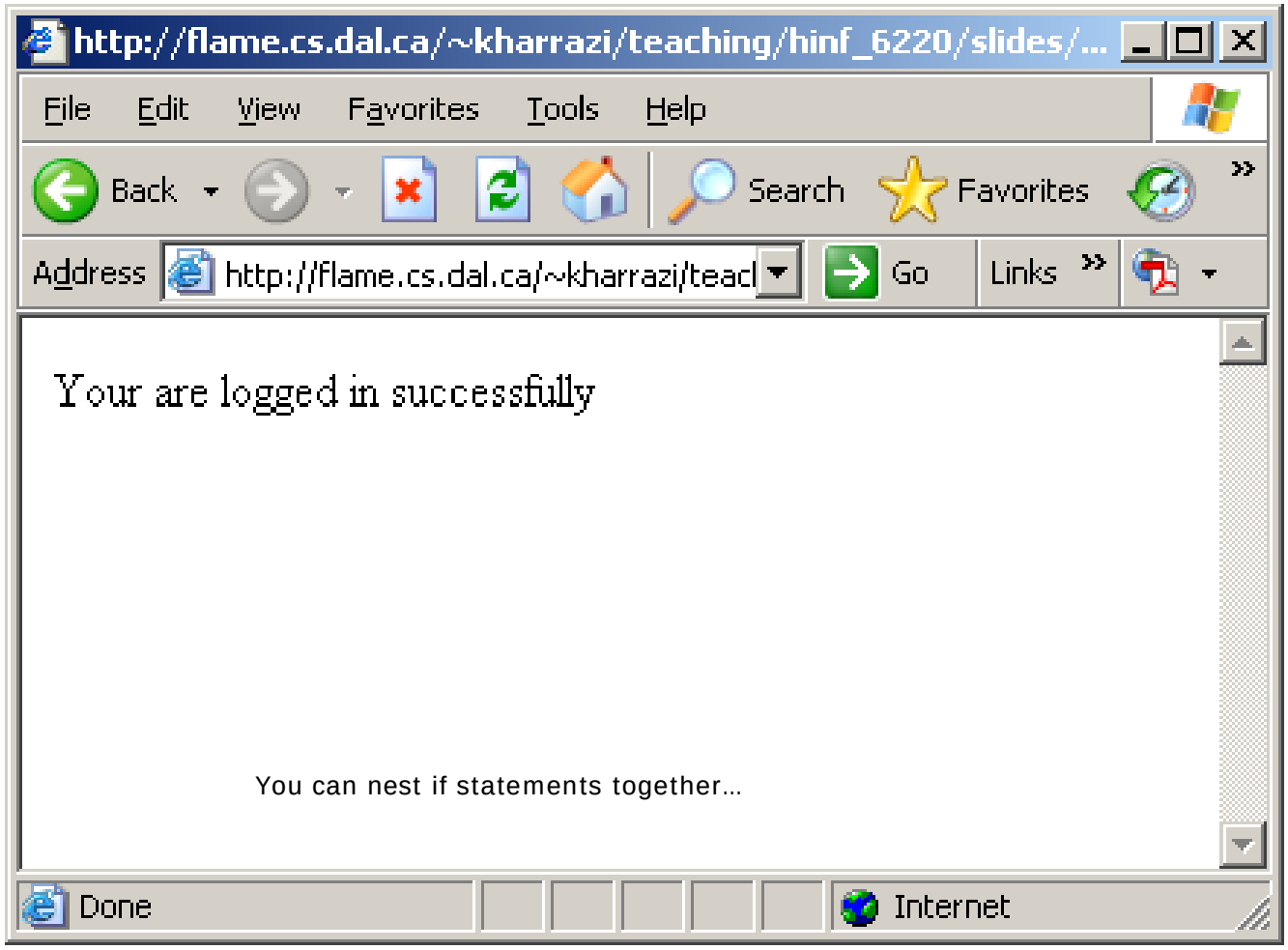
PHP Conditions (cont.)

Nested if

```
<?php
     $username = "Peter";
     $password = "abc";

     // Nested IF conditional statements
     if ($username == "Peter"){
       if ($password == "abc"){
            echo "Your are logged in successfully";
       }
     }
?>
```

PHP Conditions (cont.)

Nested if



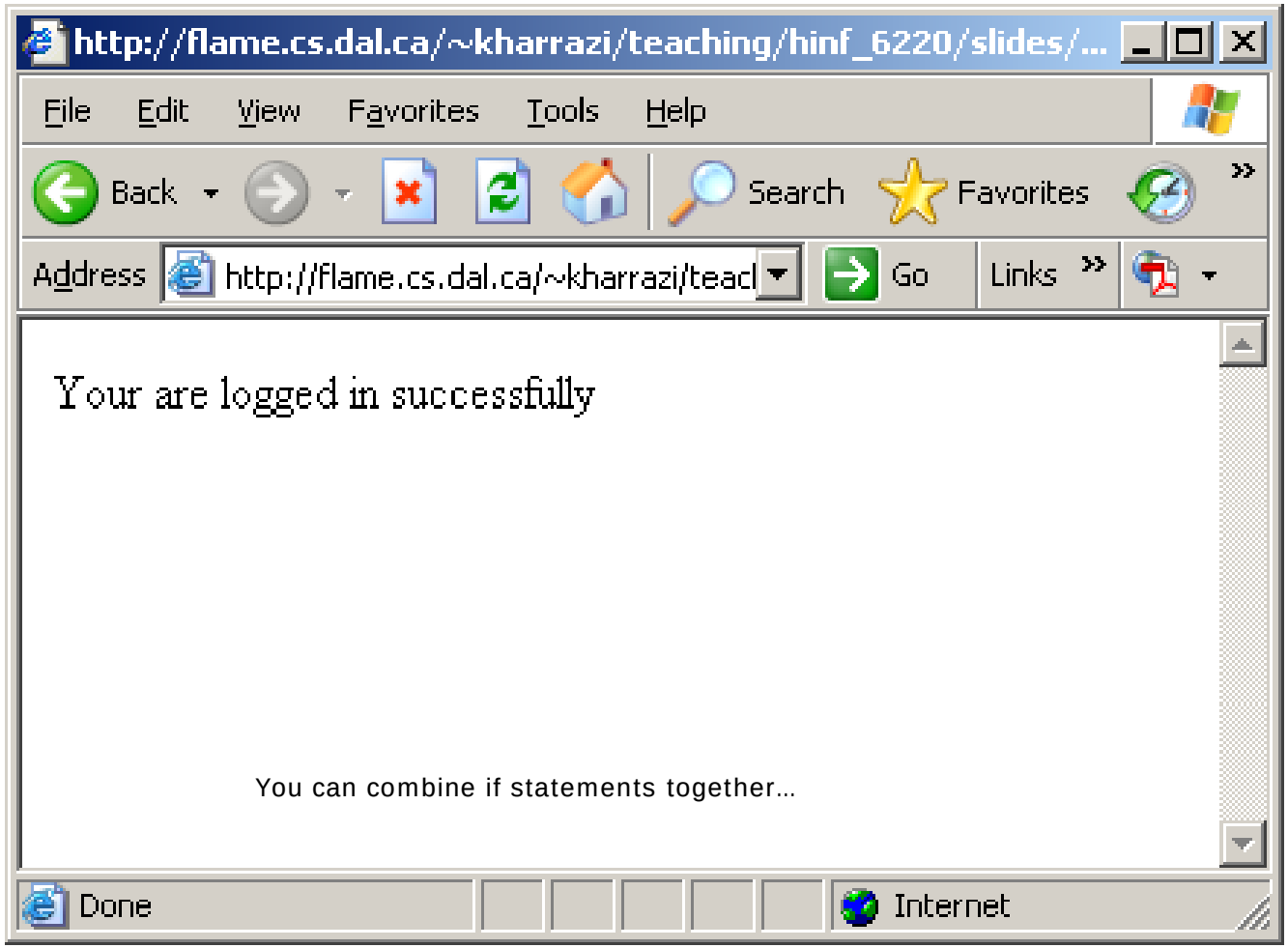
PHP Conditions (cont.)

Combined if

```
<?php
     $username = "Peter";
     $password = "abc";

     // Combined IF conditional statements
     if (($username == "Peter") && ($password == "abc")){
       echo "Your are logged in successfully";
     }
?>
```

PHP Conditions (cont.)

Combined if



PHP Conditions (cont.)

- *The Switch Statement*: If you want to select one of many blocks of code to be executed, use the Switch statement.

```
switch (expression)
{
case label1:
  code to be executed if expression = label1;
  break;
case label2:
  code to be executed if expression = label2;
  break;
default:
  code to be executed
  if expression is different
  from both label1 and label2;
}
```

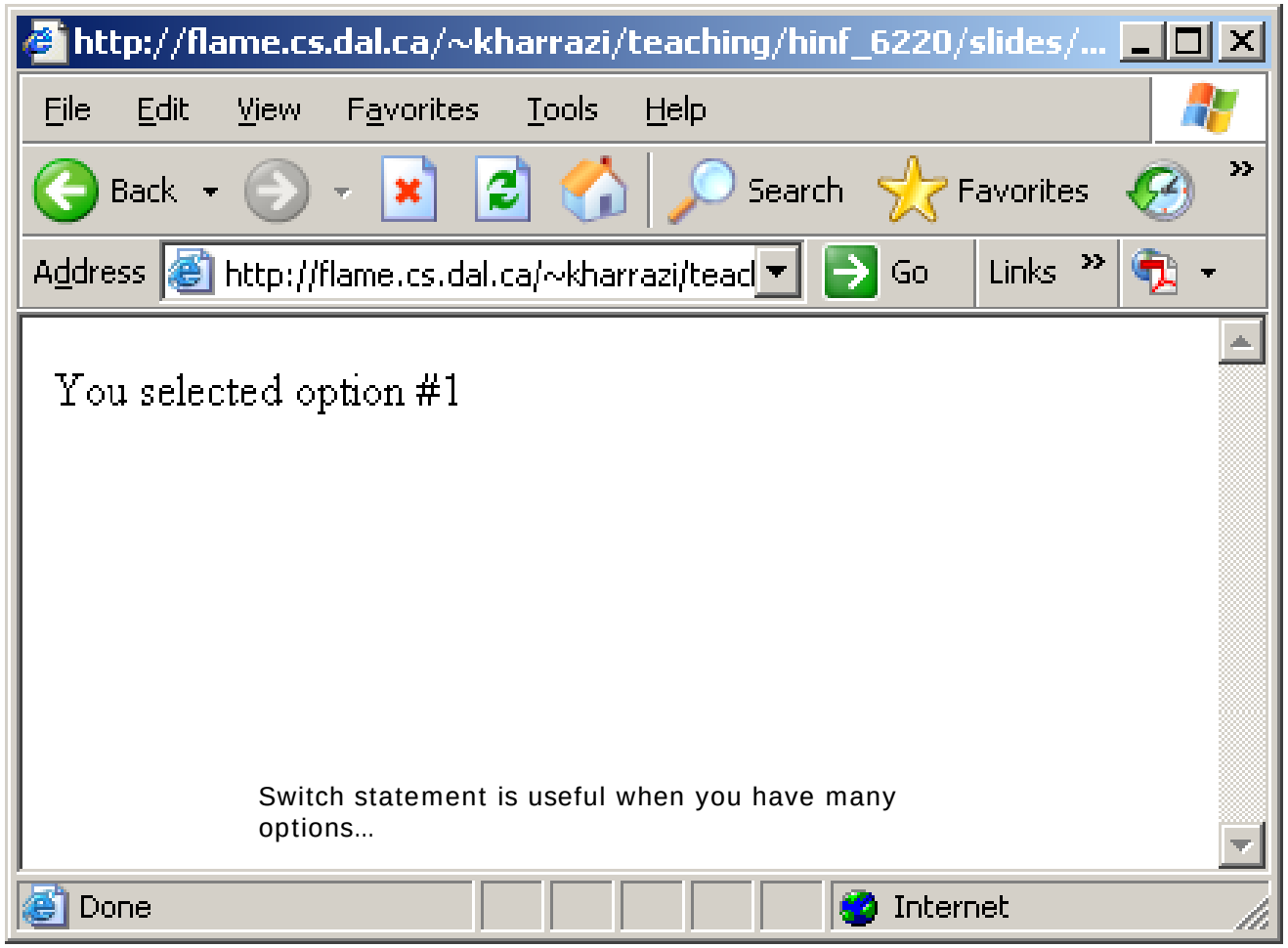
PHP Conditions (cont.)

switch

```
<?php
      $option = 1;
      // SWITCH conditional statements
      switch ($option)
      {
      case 1:
        echo "You selected option #1";
        break;
      case 2:
        echo "You selected option #2";
        break;
      case 3:
        echo "You selected option #3";
        break;
      default:
        echo "You didn't select any options";
      }
?>
```

PHP Conditions (cont.)

switch



2. PHP Loops

- Very often when you write code, you want the same block of code to run a number of times. You can use looping statements in your code to perform this.

- In PHP we have the following looping statements:

  - *while* - loops through a block of code if and as long as a specified condition is true
  - *do...while* - loops through a block of code once, and then repeats the loop as long as a special condition is true
  - *for* - loops through a block of code a specified number of times
  - *foreach* - loops through a block of code for each element in an array

PHP Loops (cont.)

- *The while Statement*: The while statement will execute a block of code if and as long as a condition is true.

- [] syntax means that is optional.

```
while (condition)
code to be executed;
```

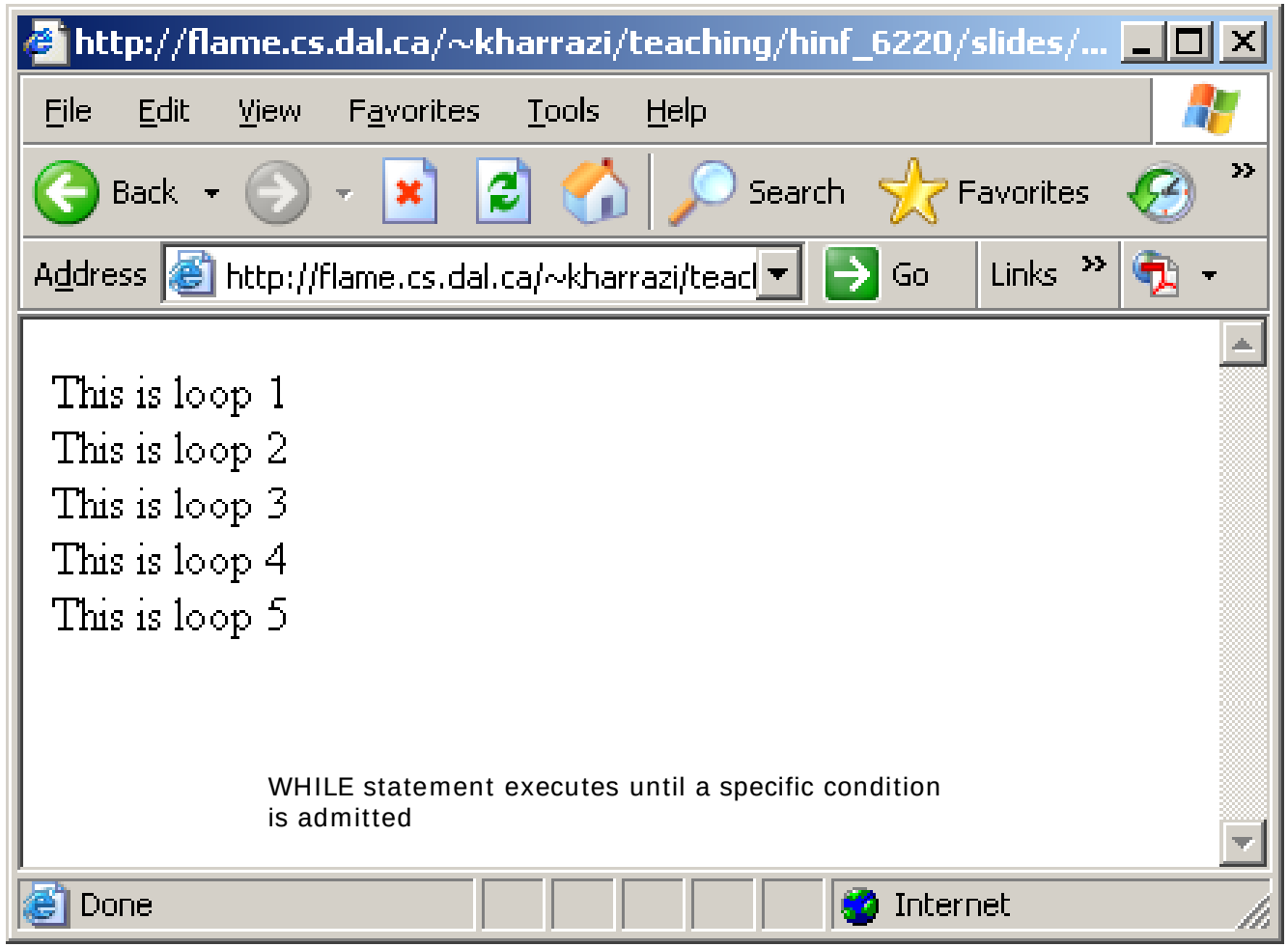
PHP Loops (cont.)

while

```
<?php
     $i=1;

     // WHILE loop statement
     while($i <= 5)
     {
       echo "This is loop " . $i . "<br />";
       $i++;
     }
?>
```

PHP Loops (cont.)

while



PHP Loops (cont.)

- *The do...while Statement*: The do...while statement will execute a block of code at least once - it then will repeat the loop as long as a condition is true.

- [] syntax means optional.

```
do
{
code to be executed;
}
while (condition);
```

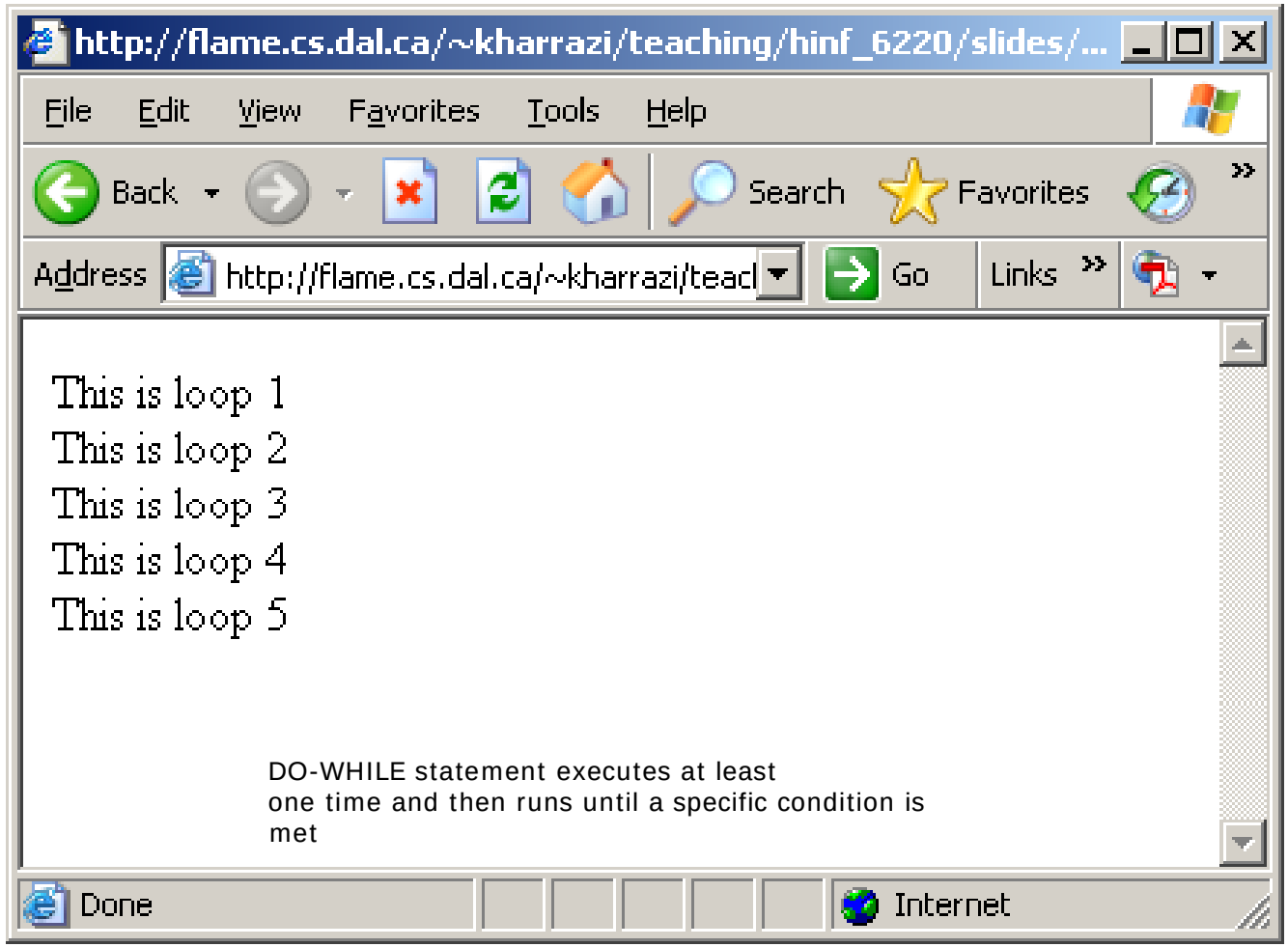
PHP Loops (cont.)

do...while

```
<?php
     $i = 0;

     // DO_WHILE loop statement
     do
     {
       $i++;
       echo "This loop is " . $i . "<br />";
     }
     while ($i < 5);
?>
```

PHP Loops (cont.)

do...while



PHP Loops (cont.)

- *The for Statement*: The for statement is used when you know how many times you want to execute a statement or a list of statements.

- [] syntax means optional.

```
for (initialization; condition; increment)
{
    code to be executed;
}
```

PHP Loops (cont.)

- The for statement has three parameters. The first parameter is for initializing variables, the second parameter holds the condition, and the third parameter contains any increments required to implement the loop. If more than one variable is included in either the initialization or the increment section, then they should be separated by commas. The condition must evaluate to true or false.

```
for (initialization; condition; increment)
{
    code to be executed;
}
```

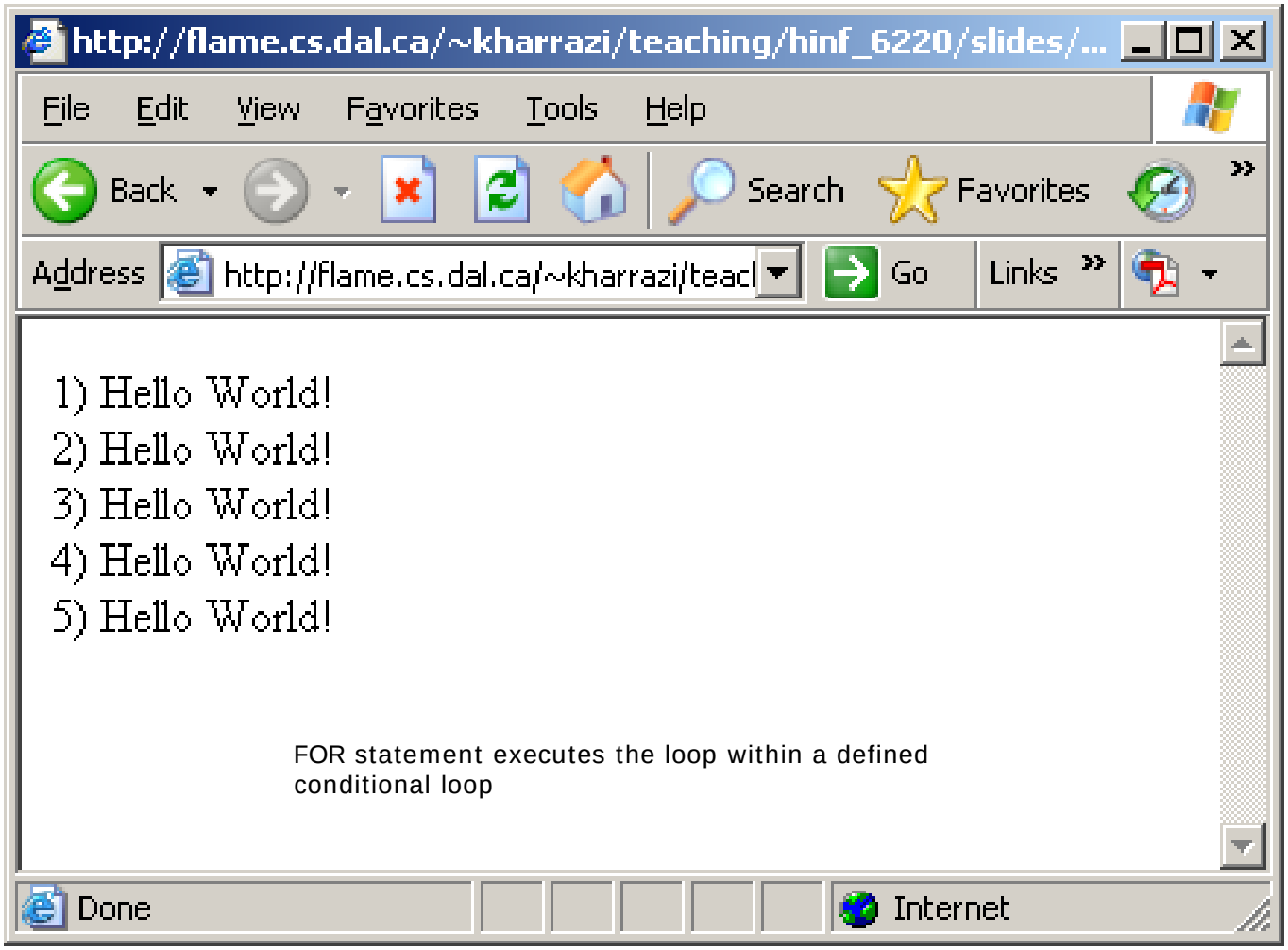
PHP Loops (cont.)

for

```
<?php

    // FOR loop statement
    for ($i=1; $i<=5; $i++)
    {
    echo $i . ") Hello World! <br />";
    }
?>
```

PHP Loops (cont.)

for



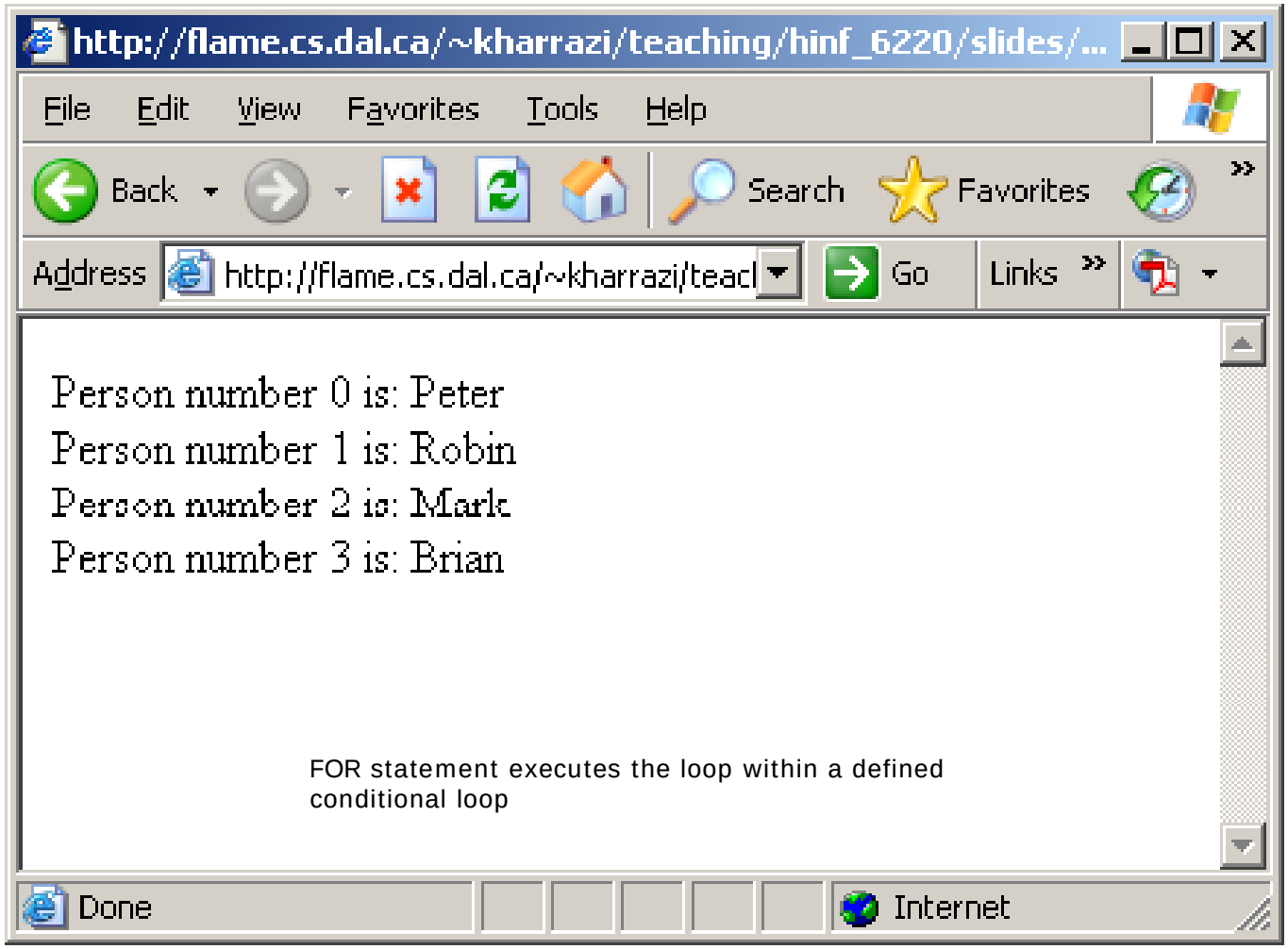
PHP Loops (cont.)

for

```php
<?php

    $x = array ("Peter", "Robin", "Mark", "Brian");

    // FOR loop statement
    for ($i=0; $i<count($x) ; $i++)
    {
      echo "Person number $i is: " . $x[$i] . " <br />";
    }

?>
```

PHP Loops (cont.)

for



PHP Loops (cont.)

- *The foreach Statement*: Loops over the array given by the parameter. On each loop, the value of the current element is assigned to $value and the array pointer is advanced by one - so on the next loop, you'll be looking at the next element.

- [] syntax means optional.

```
foreach (array as value)
{
    code to be executed;
}
```

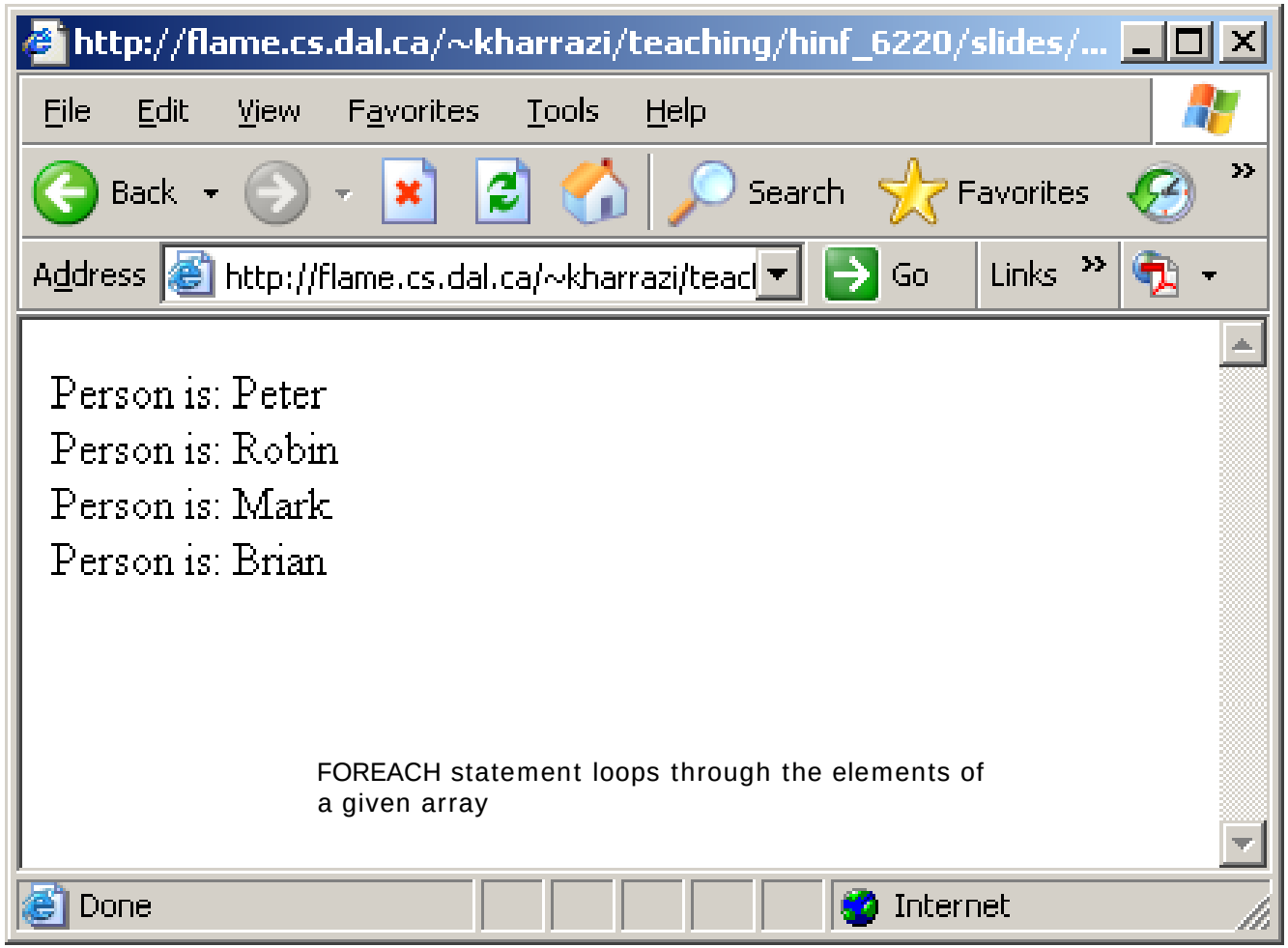
PHP Loops (cont.)

foreach

```
<?php

    $x = array ("Peter", "Robin", "Mark", "Brian");

    // FOREACH loop statement
    foreach ($x as $v)
    {
      echo "Person is: " . $v . " <br />";
    }
?>
```

PHP Loops (cont.)

foreach



3. PHP Functions

- A function may be defined by the users or be pre-defined by built-in PHP functions. Both user-defined and PHP-defined functions should be called in order to execute their contents.
- [] syntax means optional.

```
function name ([var1, var2, ..., var n])
{
    code to be executed;
}
.
.
.
name ([x1, x2, x3, ..., x n]);
```

PHP Functions (cont.)

- Built-in functions are available to all PHP scripts, but if you declare your own functions, they are only available to the script(s) in which they were declared.

- Within a function, curly braces enclose the code that performs the task you require. Between these braces, you can have anything that is legal elsewhere in a PHP script including function calls, declarations of new variables or functions.

- PHP does not support function overloading, so your function cannot have the same name as any built-in function or an existing user-defined function.

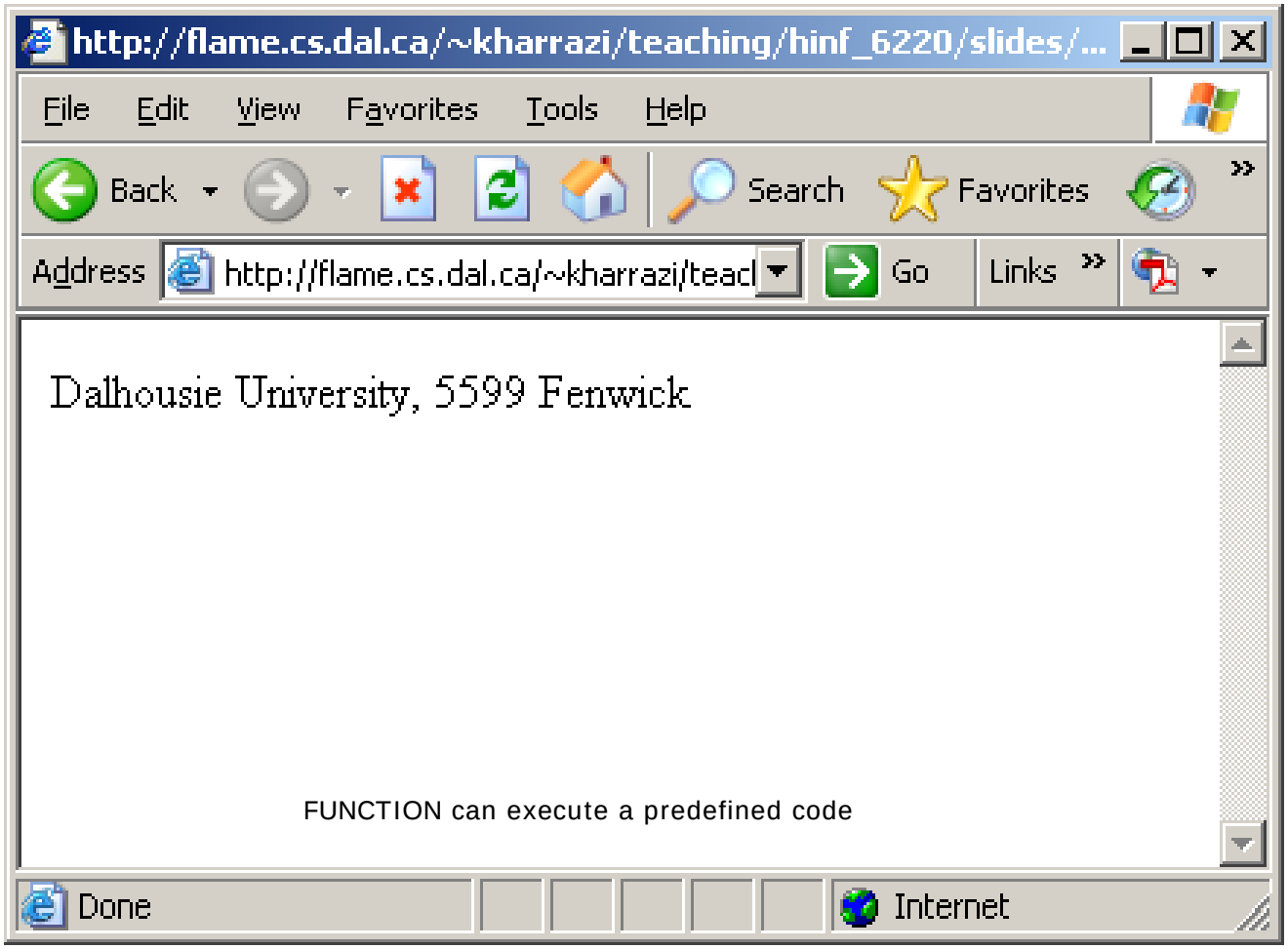
PHP Functions (cont.)

functions

```php
<?php

    // Declare or define the function
    function show_address(){
      echo "Dalhousie University, 5599 Fenwick";
    }

    // Call the function
    show_address();
?>
```

PHP Functions (cont.)

functions



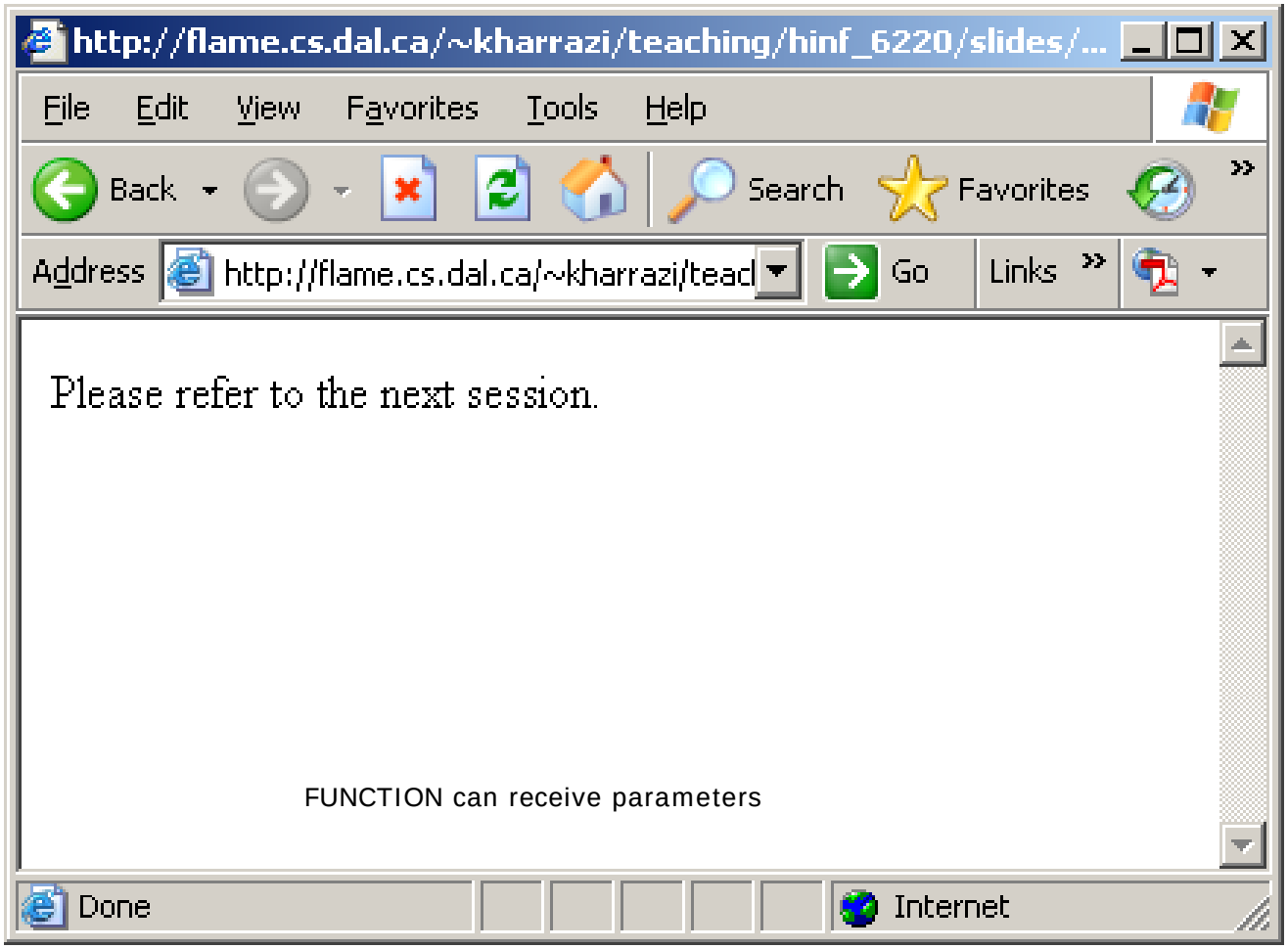
PHP Functions (cont.)

functions

```php
<?php

    // Declare or define the function
    function show_me($x){
      echo $x;
    }

    // Call the function
    show_me("Please refer to the next session.");
?>
```

PHP Functions (cont.)

functions



PHP Functions (cont.)

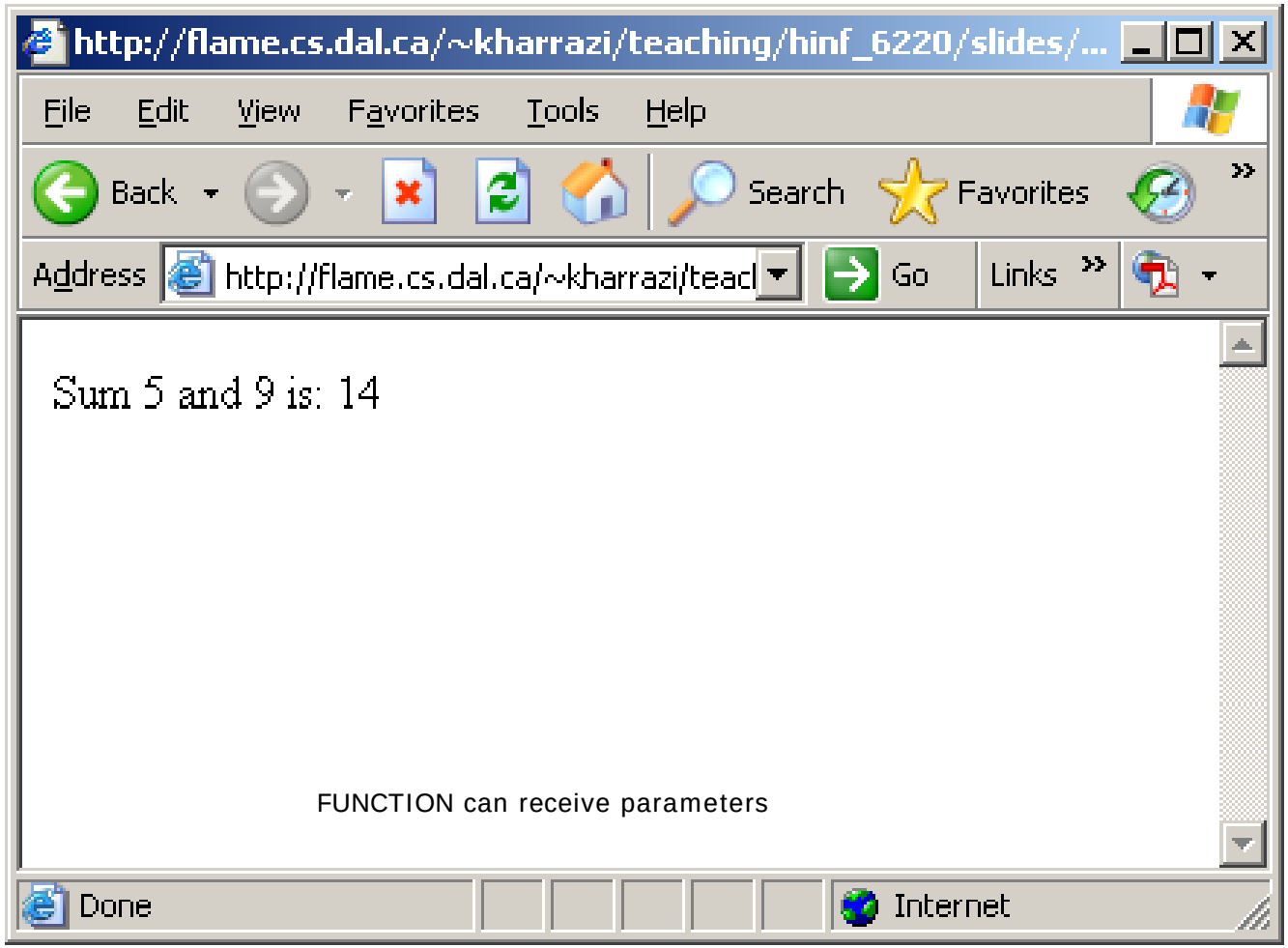
functions

```php
<?php

    // Declare or define the function
    function sum($x, $y){
      return $x+$y;
    }

    // Call the function
    echo "Sum 5 and 9 is: " . sum(5, 9);
?>
```

Scope of $x and $y is limited to the function only

PHP Functions (cont.)

functions



PHP Functions (cont.)

- Variables declared *inside a function* are in scope from the statement in which they are declared to the closing brace at the end of the function. This is called function scope. These variables are called local variables.

- Variables declared *outside of functions* are in scope from the statement in which they are declared to the end of the file, but not inside functions. This is called global scope. These variables are called global variables.

- The special *superglobal* variables are visible both inside and outside functions.

PHP Functions (cont.)

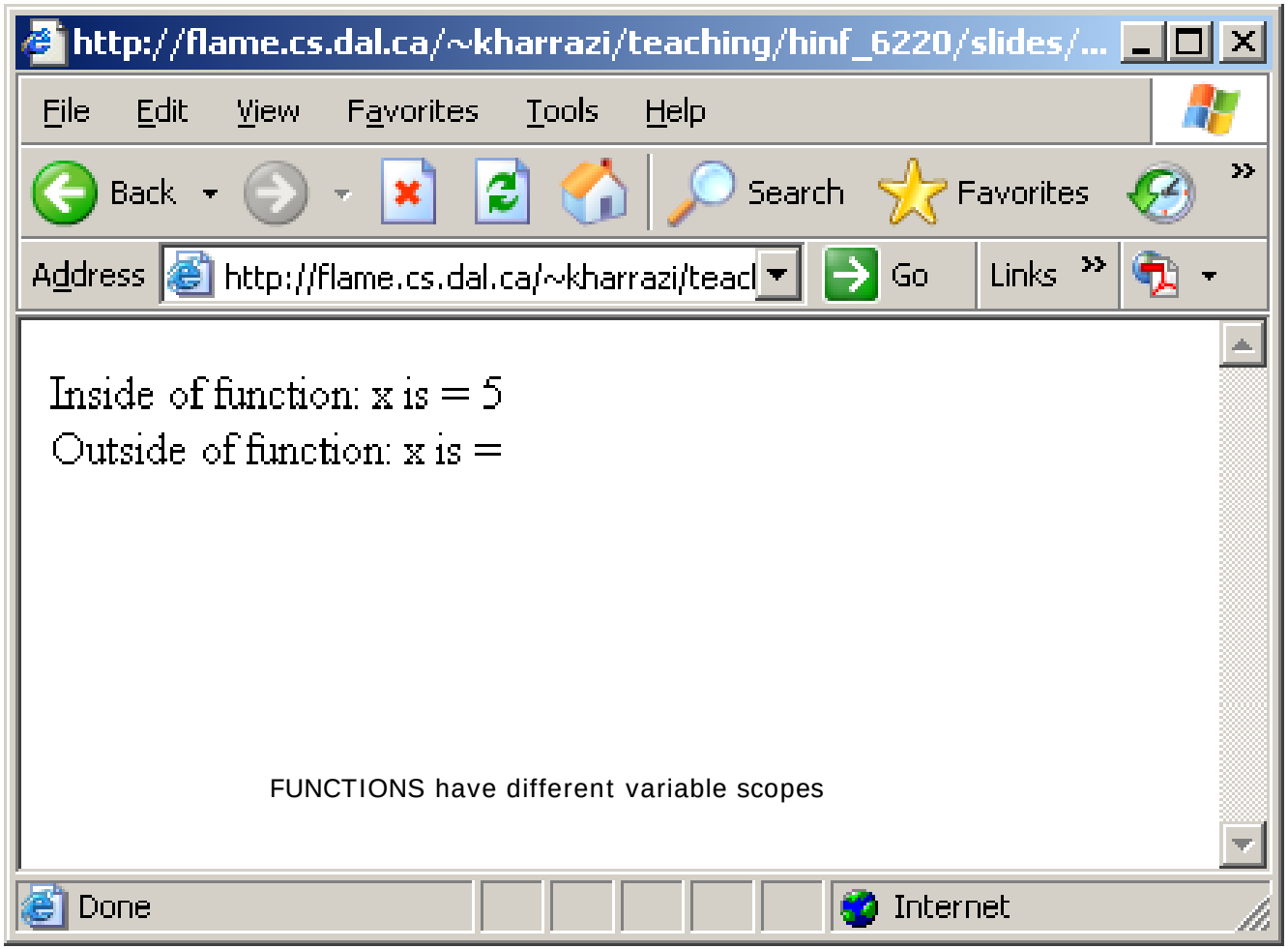
functions

```php
<?php

    // Declare or define the function
    function something(){
      $x = 5;
      echo "Inside of function: x is = " . $x;
    }

    // Call the function
    something();
    echo "<br>";
    echo "Outside of function: x is = " . $x;
?>
```

Scope of $x is limited to the function only

PHP Functions (cont.)

functions



PHP Functions (cont.)

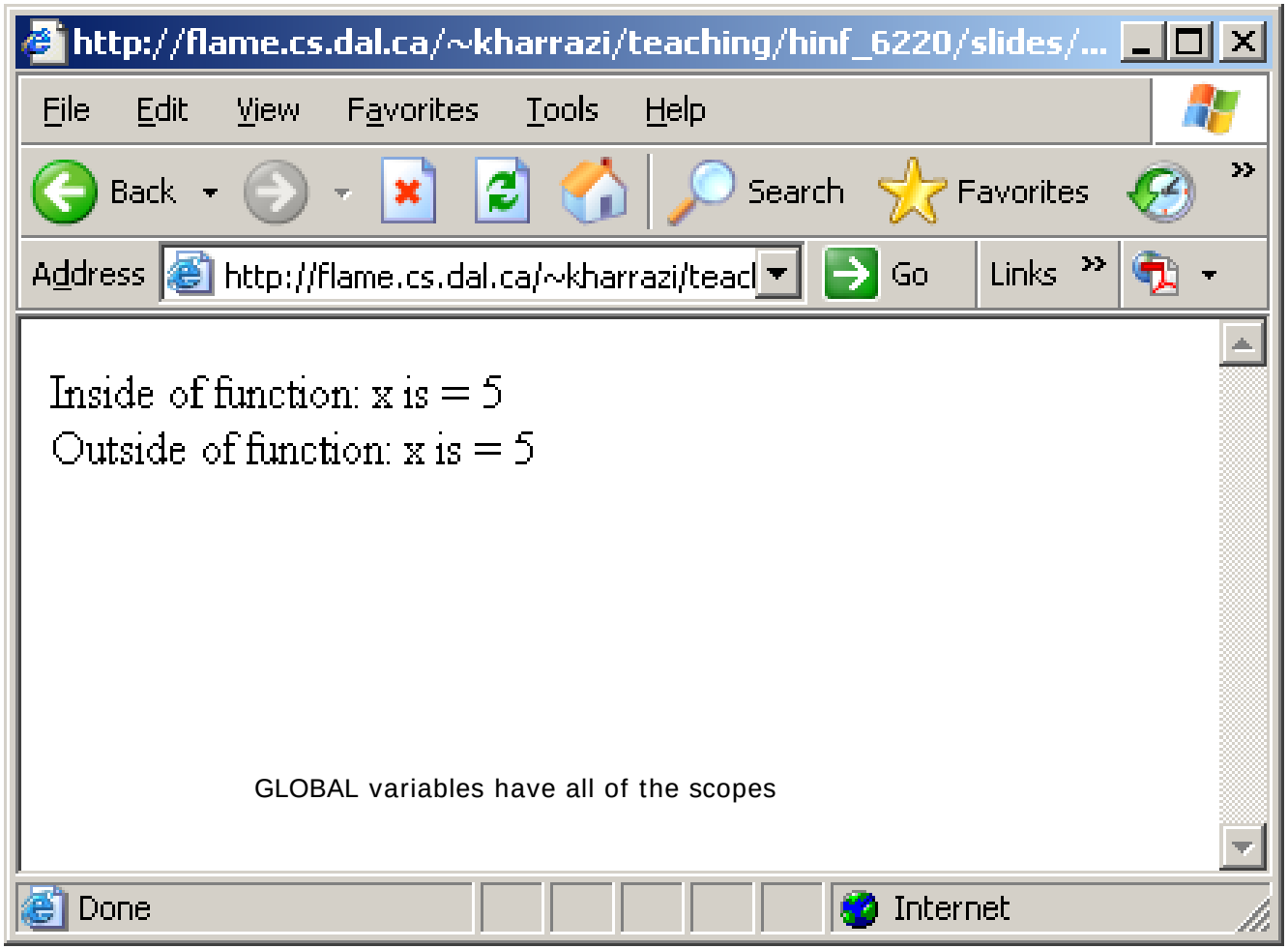
functions

```php
<?php

    // Declare or define the function
    function something(){
      global $x;
      $x = 5;
      echo "Inside of function: x is = " . $x;
    }

    // Call the function
    something();
    echo "<br>";
    echo "Outside of function: x is = " . $x;
?>
```

Scope of $x is NOT limited to the function only

PHP Functions (cont.)

functions



Summary

1. PHP Conditions
2. PHP Loops
3. PHP Functions

Next Sessions

- PHP Cookies / Sessions
- PHP SSI
- PHP Forms
- PHP/MySQL Integration

Exercise

- Please refer to the available text file in the slides section for this session on the course website:

- http://info510.com/core/public_page.php?page_name=slides