SOUTHERN ILLINOIS UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

SIU CARBONDALE

INTRODUCTION TO PHP

Origin and Uses of PHP

- Developed by Rasmus Lerdorf in 1994
- PHP is a server-side scripting language, embedded in XHTML pages
- PHP has good support for form processing
- PHP can interface with a wide variety of databases

Overview of PHP

- When a PHP document is requested, the web server will forward it to the PHP processor
- The result of processing is the response to the request
- Two modes of operation
    - Copy mode in which plain HTML is copied to the output
    - Interpret mode in which PHP code is interpreted
    - The client never sees PHP code, only the output HTML
- PHP has typical scripting language characteristics
    - Dynamic typing, untyped variables
    - Associative arrays
    - Pattern matching
    - Extensive libraries

General Syntactic Characteristics

- PHP code is contained between the tags <?php and ?>
- Code can be included with the PHP “include” directive:

```
include(“table2.inc”);
```

  - The code in an include file must be in <?php and ?> tags
- All variable names in PHP begin with $ and continue as usual for variables
- Variable names are case sensitive
- However *keywords* and *function names* are *not* case sensitive

PHP Syntax

- One line comments can begin with # or // and continue to the end of the line
- Multi-line comments can begin with /* and end with */
- PHP statements are terminated with semicolons;
- Curly braces are used to create compound statements
- Variables cannot be defined in a compound statement unless it is the body of a function!

Primitives, Operations, Expressions

- Four scalar types: boolean, integer, double, string
- Two compound types: array, object
- Two special types: resource and NULL
  - Resource can be any external resource, .e.g., FTP stream, link to database, reference to COM object, etc.

Variables

- Variables are not declared except in order to specify *scope* or *lifetime*
- A variable that has not been assigned a value is ***unbound*** and has the special value NULL
  - NULL is coerced to 0 if a number is needed, to the empty string “” if a string is needed
  - Both of these coercions count as Boolean FALSE
- The `error_reporting` function is used to set the level of problems that will be reported
  - For example, level 15 includes reporting *unbound* variable errors
  - The default reporting level is 7, which won’t report unbound errors

Integer Type

- PHP distinguishes between integer and floating point numeric types
- Integer is equivalent to the *long* type in C, that is, usually 32 bits

Double Type

- Literal double type numeric values include a period and/or the exponent sign: either e or E
- Double type values are stored internally as double precision floating point values

String Type

- Characters in PHP take one byte
- String literals are enclosed in single or double quotes
  - Double quoted strings have escape sequences interpreted and variables interpolated
  - Single quoted strings have neither
  - A literal $ sign in a double quoted string must be escaped with a backslash \
- Double-quoted strings can cover multiple lines, the included end of line characters are part of the string value

  → It may need be trimmed off

Boolean Type

- The Boolean type has two values :TRUE and FALSE
- Other type values are coerced as needed according to context, for example, in control expressions
  - The integer value 0, the empty string “” and the literal string “0” all count as false
  - NULL counts as false, too.
  - The double value 0.0 counts as false.
    - Beware, however, that double calculations rarely result in the exact value 0.0, and won’t be counted as false!

Arithmetic Operators and Expressions

- PHP supports the usual operators supported by the C/C++/Java family
- Integer divided by integer results in integer if there is no remainder but results in double if there is a remainder
  - 12/6 is 2
  - 12/5 is 2.4
- A variety of numeric functions is available: `floor, ceil, round, rand (and srand), abs, min, max`
  - **`rand` generates a pseudorandom number (between a range if given two params, otherwise default range)**
  - **`Min/max` require input of an array or 1+ augments**

String Operations

- String catenation is indicated with a period
- Characters are accessed in a string with a subscript enclosed in curly braces { }
- Many useful string functions are provided
  - `strlen` gives the length of a string
  - `strcmp` compares two strings and return < 0 if str1 is less than str2, > 0 if str1 is greater than str2 , and 0 if they are equal.
  - `chop` removes whitespace from the end of a string
  - `trim` remove whitespaces from both ends
  - `strtolower`
  - `strtoupper`

Scalar Type Conversions

- Implicit type conversions are performed in the context in which an expression appears
  - A string is converted to an integer if a numeric value is required and the string has only a sign followed by digits
  - A string is converted to a double if a numeric value is required and the string is a valid double literal (including a *period* or *e/E*)
- Type conversions can be forced in three ways
  - `(int)$sum` in the C style
  - `intval($sum)` using several conversion functions
  - `settype($x, “integer”)`

Scalar Type Conversions

- Type can be determined with the `gettype` function and with the `is_int` function and similar ones for other types

Assignment Operators

- The assignment operators used in C/C++/Java are supported in PHP

Output

- The `print` function is used to send data to output
  - `print` takes string parameters, coercion done as necessary
- The C `printf` function is also available
  - The first argument to printf is a string with interspersed format codes
  - A format code begins with % followed by a field width and a type specifier
  - Common type specifiers are s for string, d for integer and f for double
  - Field width is a single integer to specify the number of characters (minimum) used to display the value or two integers separated by a period to indicate total field width and decimal part of double values
  - `printf("x = %5d is %s\n", $x, $size);`

    Displays `$x` as an integer and `$size` as a string
- The example `today.php` uses the date function to dynamically generate a page with the current date

```
<!DOCTYPE html>
<!-- today.php - A trivial example to illustrate a php document -->
<html lang = "en">
  <head>
    <title> today.php </title>
    <meta charset = "utf-8" />
  </head>
  <body>
    <p>
      <?php
        print "<b>Welcome to my home page <br /> <br />";
        print "Today is:</b> ";
        print date("l, F jS");
        print "<br />";
      ?>
    </p>
  </body>
</html>
```

Relational Operators

- PHP has the usual comparison operators:
  - >, < <=, >=, == and !=
- PHP also has the identity operator ===

Relational Operators

- The regular comparisons will force conversion of values as needed
  - Comparing a string with a number (other than with ===) will result in the string converting to a number if it can be. Otherwise the number is converted to a string
  - If two strings are compared (other than with ===) and the strings can both be converted to numeric values, the conversion will be done and the converted values compared
  - Use `strcmp` to avoid conversion to numeric values

Boolean Operators

- PHP supports &&, || and ! as in C/C++/Java
- There are lower precedence versions of the operations too:

  `and` and `or` are also provided
- The `xor` operator is provided as well

Selection Statements

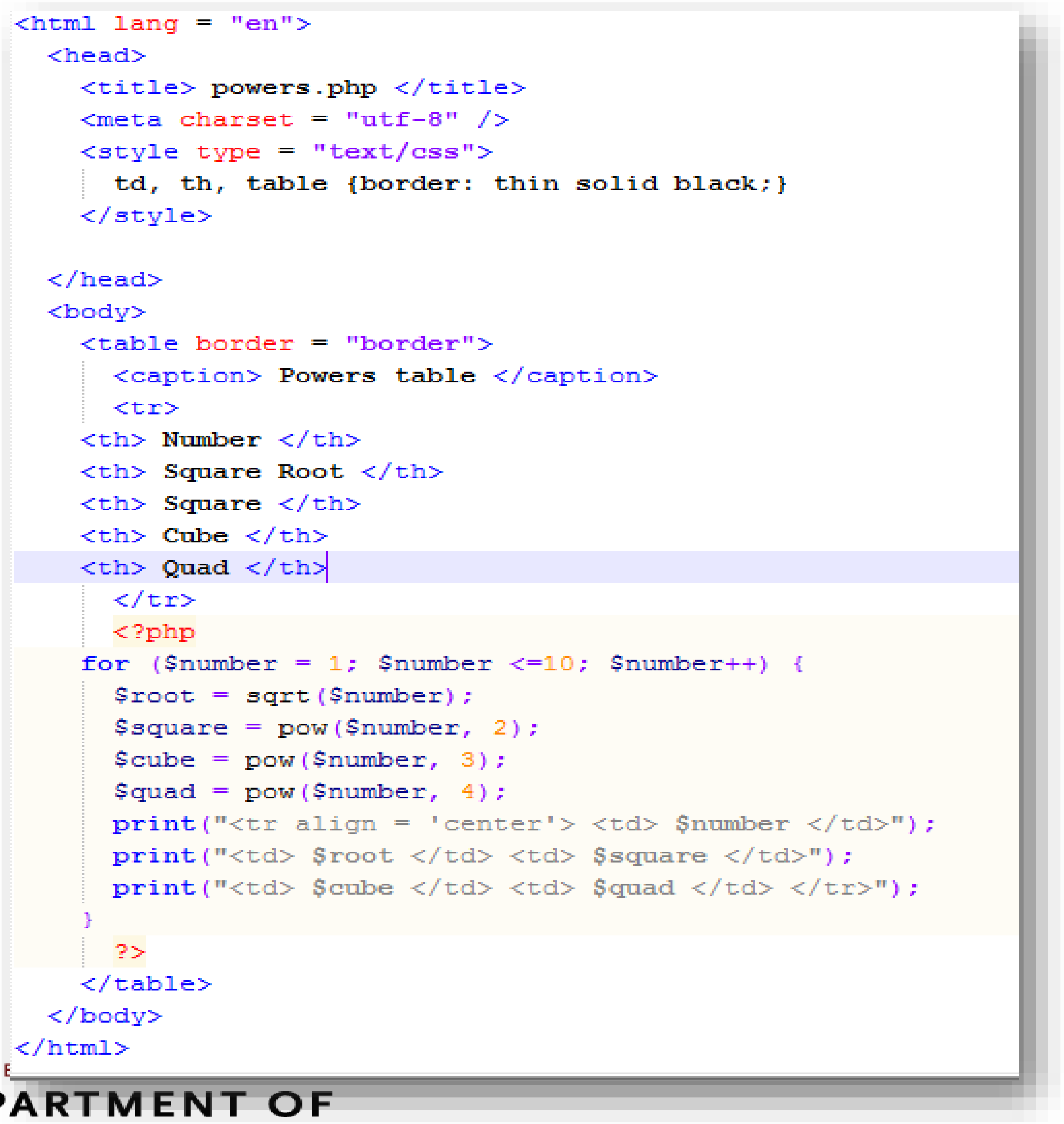
- PHP provides if with almost the same syntax as C/C++/Java
  - The only difference is the elseif (can have multiple)
- The switch statement is provided with syntax and semantics similar to C/C++/Java
  - The *case expressions* are coerced before comparing with the *control expression*
  - `break` is necessary to prevent execution from flowing from one case to the next

Switch: Syntax

```
switch (expr) {       // expr is the control expression
  case c1:            // c1 is a case expression
        statements // do these if expr == c1
        break;
  case c2:
        statements // do these if expr == c2
        break;
  case c2:
  case c3:
  case c4:            //  Cases can simply fall thru.
        statements // do these if expr ==       any of the c's
        break;
  . . .
  default:
        statements // do these if expr != any above
}
```

Loop Statements

- PHP provides the while and for and do-while as in JavaScript
- The for loop is illustrated in the example `powers.php`
- This example also illustrates a number of mathematical functions available in PHP
- Next slide



```
<html lang = "en">
  <head>
    <title> powers.php </title>
    <meta charset = "utf-8" />
    <style type = "text/css">
      td, th, table {border: thin solid black;}
    </style>

  </head>
  <body>
    <table border = "border">
      <caption> Powers table </caption>
      <tr>
    <th> Number </th>
    <th> Square Root </th>
    <th> Square </th>
    <th> Cube </th>
    <th> Quad </th>
      </tr>
      <?php
    for ($number = 1; $number <=10; $number++) {
      $root = sqrt($number);
      $square = pow($number, 2);
      $cube = pow($number, 3);
      $quad = pow($number, 4);
      print("<tr align = 'center'> <td> $number </td>");
      print("<td> $root </td> <td> $square </td>");
      print("<td> $cube </td> <td> $quad </td> </tr>");
    }
      ?>
    </table>
  </body>
</html>
```

Arrays

- Arrays in PHP combine the characteristics of regular arrays and hashes
  - An array can have elements indexed numerically. These are maintained in order
  - The same array can also have elements indexed by strings, and they are not maintained in any particular order
- The elements of an array are key/value pairs

Array Creation

- Two ways of creating an array
  - Assigning a value to an element of an array
  - Using the array function (as fellows)
- Create a numerically indexed array
  - `array(23, ‘xiv’, “bob”, 777);`
- Create an array with string indexes (associative array)
  - `array(“x” => “xerxes”, “y” => “ytrbium”)`

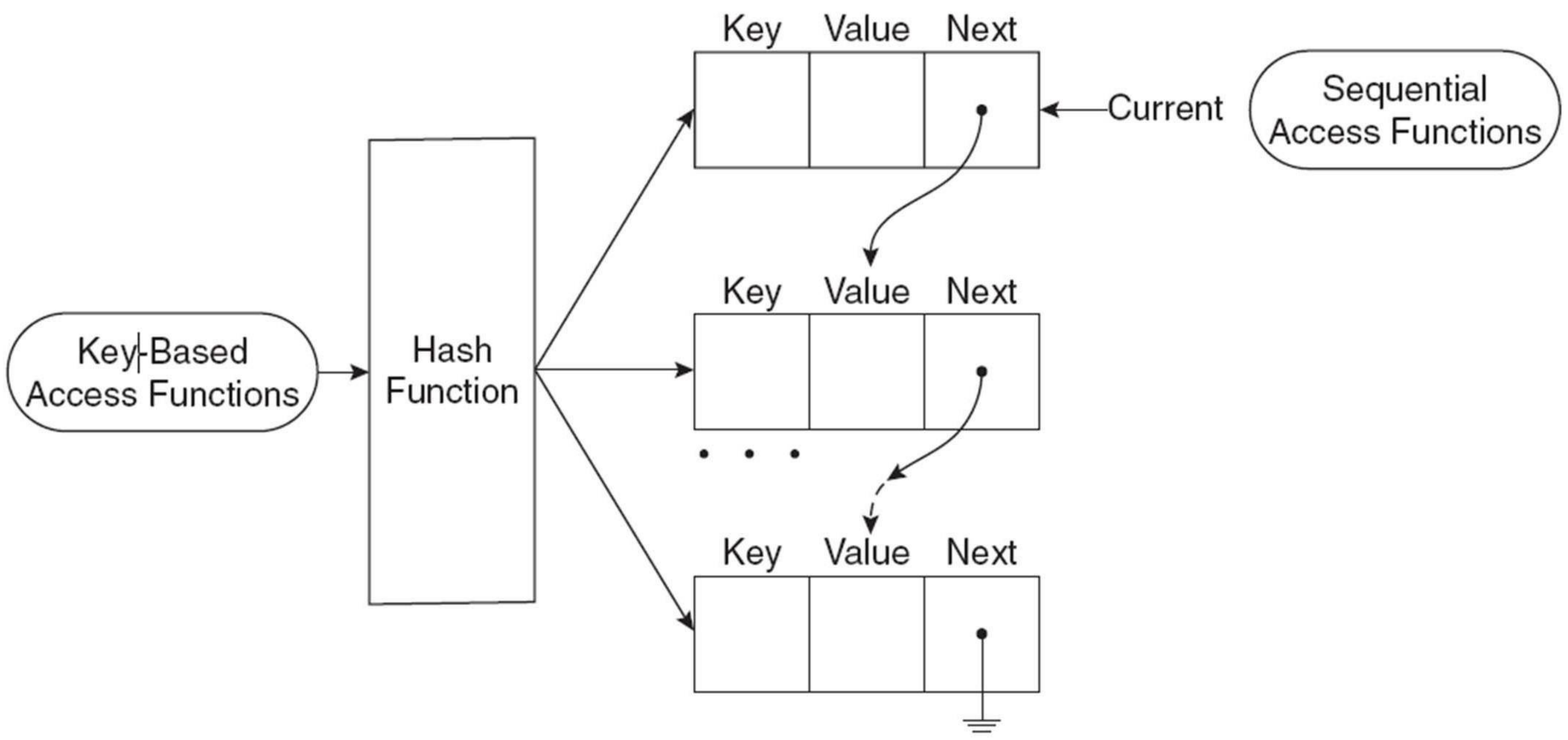
Accessing Array Elements

- Array elements are accessed by using a subscript in square brackets as with most programming languages
- An array can be assigned to a list of variables
  - `list($x, $y, $z) = array($y, $z, $x)`

Functions for Dealing with Arrays

- The **`unset`** function can remove an array or an element
- The **`array_keys`** function returns a list of the keys of an array
- The **`array_values`** returns a list of values in an array
- The **`array_key_exists`** tests if a given key is actually present in a given array
- **`is_array`** determines if its argument is an array
- **`implode`** converts an array of strings <u>to a single s</u>tring, separating the parts with a specified string
- **`explode`** converts a string into a list/array of strings by separating the string at specified characters

Logical Internal Array Structure



Sequential Access to Array Elements

- PHP maintains a marker in each array – the current pointer
  - Several functions in PHP manipulate the current pointer
  - The pointer starts at the first element when the array is created
- The `next` function moves the pointer to the next element and returns the value there
- The `each` function move the pointer to the next element and returns the key/value pair at the previous position
  - The key and value can be accessed using the “key” and “value” on the key/value pair
- Both functions (next and each) return false if no more elements are available
- `prev` moves the pointer back backwards
- `reset` moves the pointer to the beginning of the array

Arrays as Stacks

- PHP provides the `array_push` function that appends its arguments to a given array
- The function `array_pop` removes the last element of a given array and returns it

Iterating through an Array

- The `foreach` statement has two forms for iterating through an array

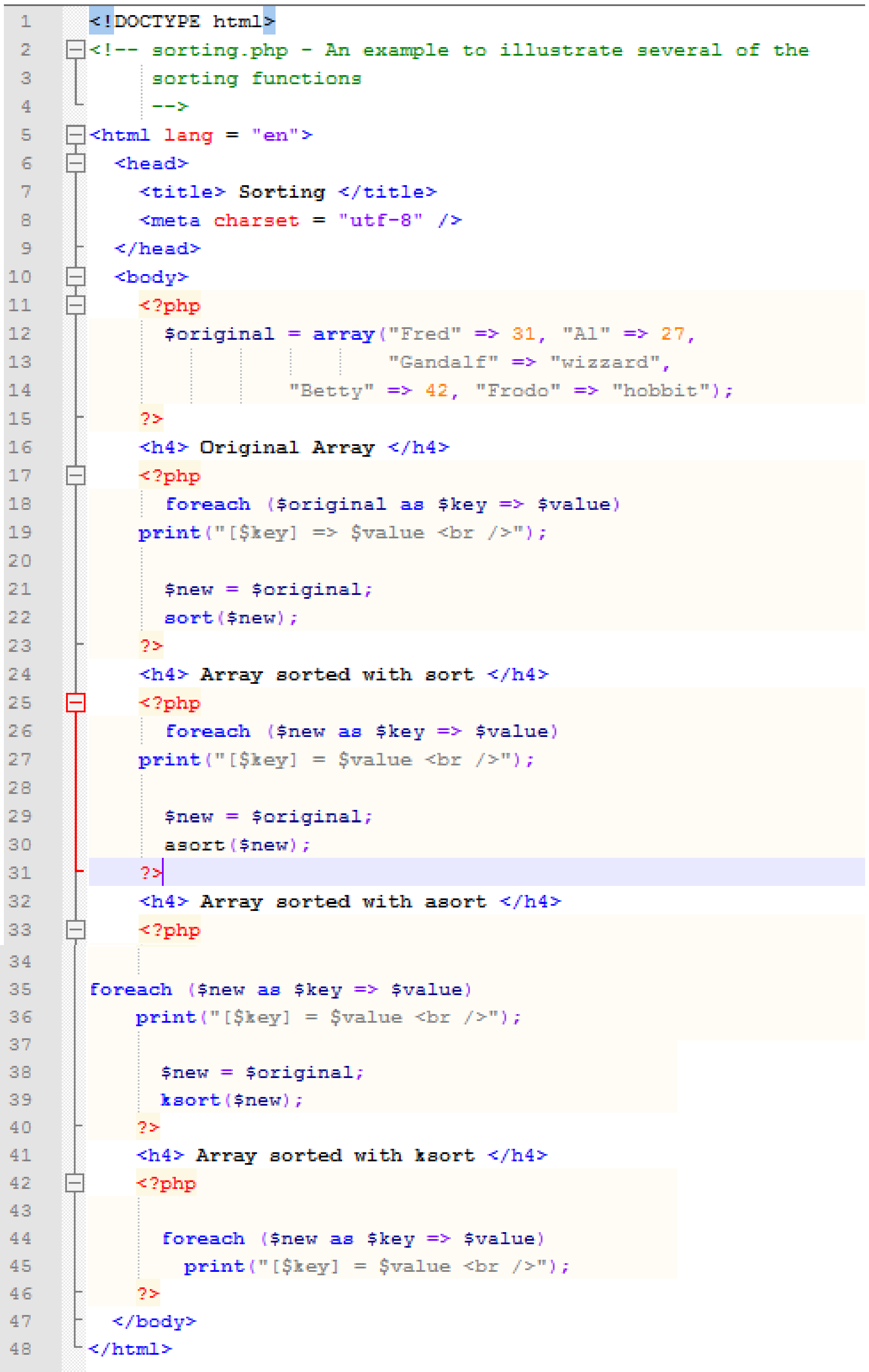
```
foreach (array as scalar_variable) loop body

foreach (array as key => value) loop body
```

- The first version assigns each value in the array to the scalar_variable in turn
- The second version assigns each key to *key* and the associated value to *value* in turn
- In this example, each day and temperature are printed

```
$lows = array("Mon" => 23, "Tue" => 18, "Wed" => 27);
foreach ($lows as $day => $temp)
print("The low temperature on $day was $temp <br />");
```

Sorting Arrays

- The `sort` function sorts the values in an array and makes a numerically indexed and sorted array (original key/value associations are permanently lost!!).
- The function `asort` sorts the values in an array but keeps the original *key/value* associations
- The function `ksort` is similar to asort but sorts by keys
- The functions `rsort`, `arsort` and `krsort` are similar but sort in reverse order
- The example `sorting.php` illustrates the various sort functions



```
<!DOCTYPE html>
<!-- sorting.php - An example to illustrate several of the
     sorting functions
     -->
<html lang = "en">
  <head>
    <title> Sorting </title>
    <meta charset = "utf-8" />
  </head>
  <body>
    <?php
      $original = array("Fred" => 31, "Al" => 27,
                           "Gandalf" => "wizzard",
                   "Betty" => 42, "Frodo" => "hobbit");
    ?>
    <h4> Original Array </h4>
    <?php
      foreach ($original as $key => $value)
    print("[$key] => $value <br />");

      $new = $original;
      sort($new);
    ?>
    <h4> Array sorted with sort </h4>
    <?php
      foreach ($new as $key => $value)
    print("[$key] = $value <br />");

      $new = $original;
      asort($new);
    ?>
    <h4> Array sorted with asort </h4>
    <?php

foreach ($new as $key => $value)
    print("[$key] = $value <br />");

      $new = $original;
      ksort($new);
    ?>
    <h4> Array sorted with ksort </h4>
    <?php

      foreach ($new as $key => $value)
        print("[$key] = $value <br />");
    ?>
  </body>
</html>
```

General Characteristics of Functions

- **Function syntax**

```
function name([parameters]) {
...
}
```

- The parameters are **optional**, but parentheses needed!
- Function names are not case sensitive
- A return statement causes the function to immediately terminate and return a value if provided
- A function that reaches the end of the body without meeting a return, returns **no** value!

Parameters

- A formal parameter, specified in a function declaration, is simply a variable name
- If more actual parameters are supplied in a call than there are formal parameters, the extra values are ignored
- If more formal parameters are specified than there are actual parameters in a call then the extra formal parameters receive no value (i.e., NULL)
- PHP defaults to pass by value
  - Putting an ampersand in front of a formal parameter **forces** pass-by- reference
  - An ampersand can also be appended to the actual parameter (which must be a variable name, and **forces** pass-by-reference**)**

The Scope of Variables

- A variable defined in a function is, by default, local to the function
- A global variable of the same name is not visible in the function
- Declaring a variable within a function with the *global* declaration means to **use** the global variable of that name, e.g.,

```
global $big_sum;   // within a function

// you have to indeed have such a global var
```

Lifetime of Variables

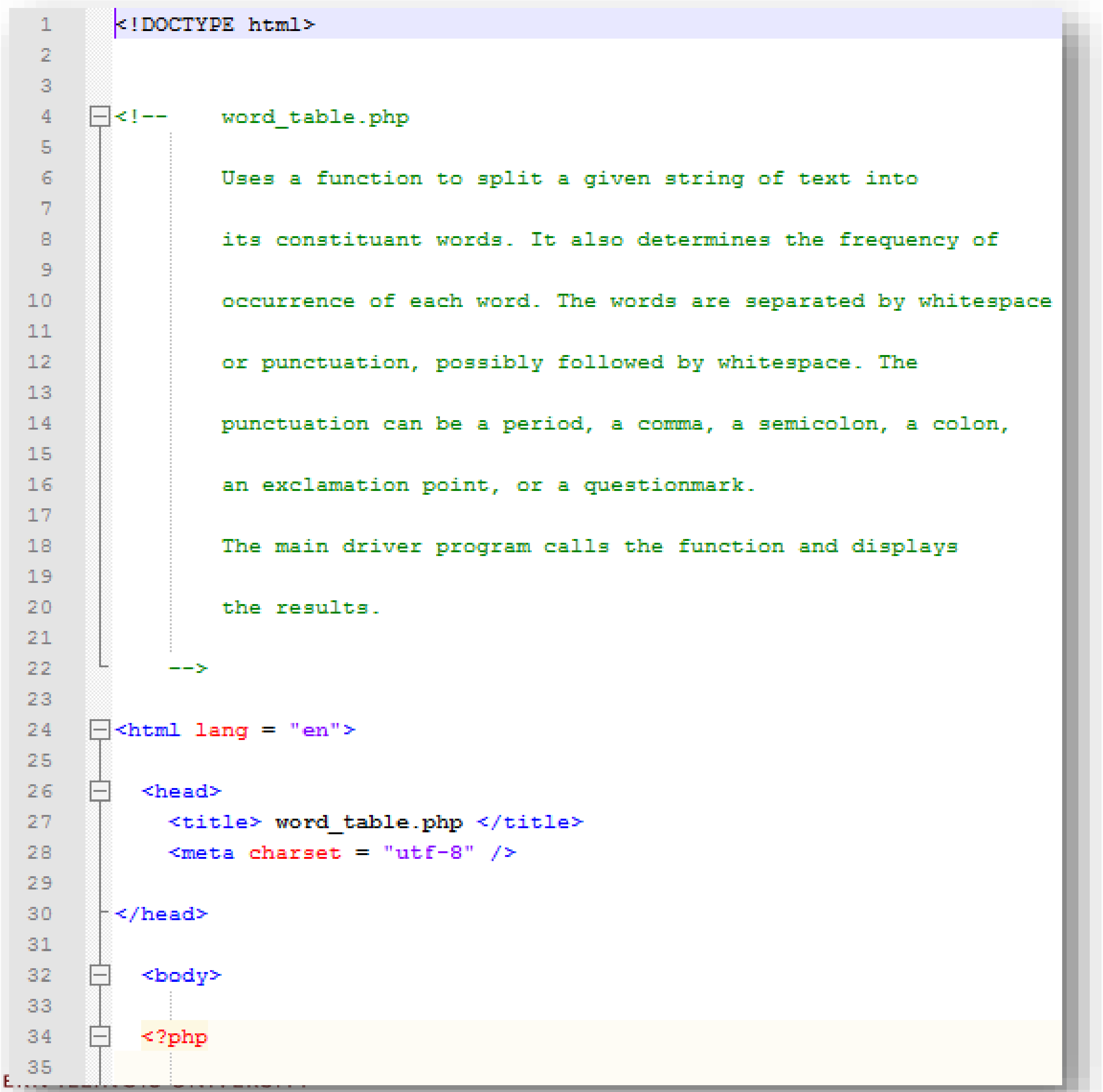
- The lifetime of a local variable is from the time the function begins to execute to the time the function returns
- Declaring a (local) variable with the *static* keyword means that the lifetime is from the **first use** of the variable to the end of the execution of the **entire script** (i.e., the browser leaves the doc)
  - Similar to that in C/C++
  - In this way a function can retain some 'history'
- The lifetime of a global variable is the entire execution

Variable functions

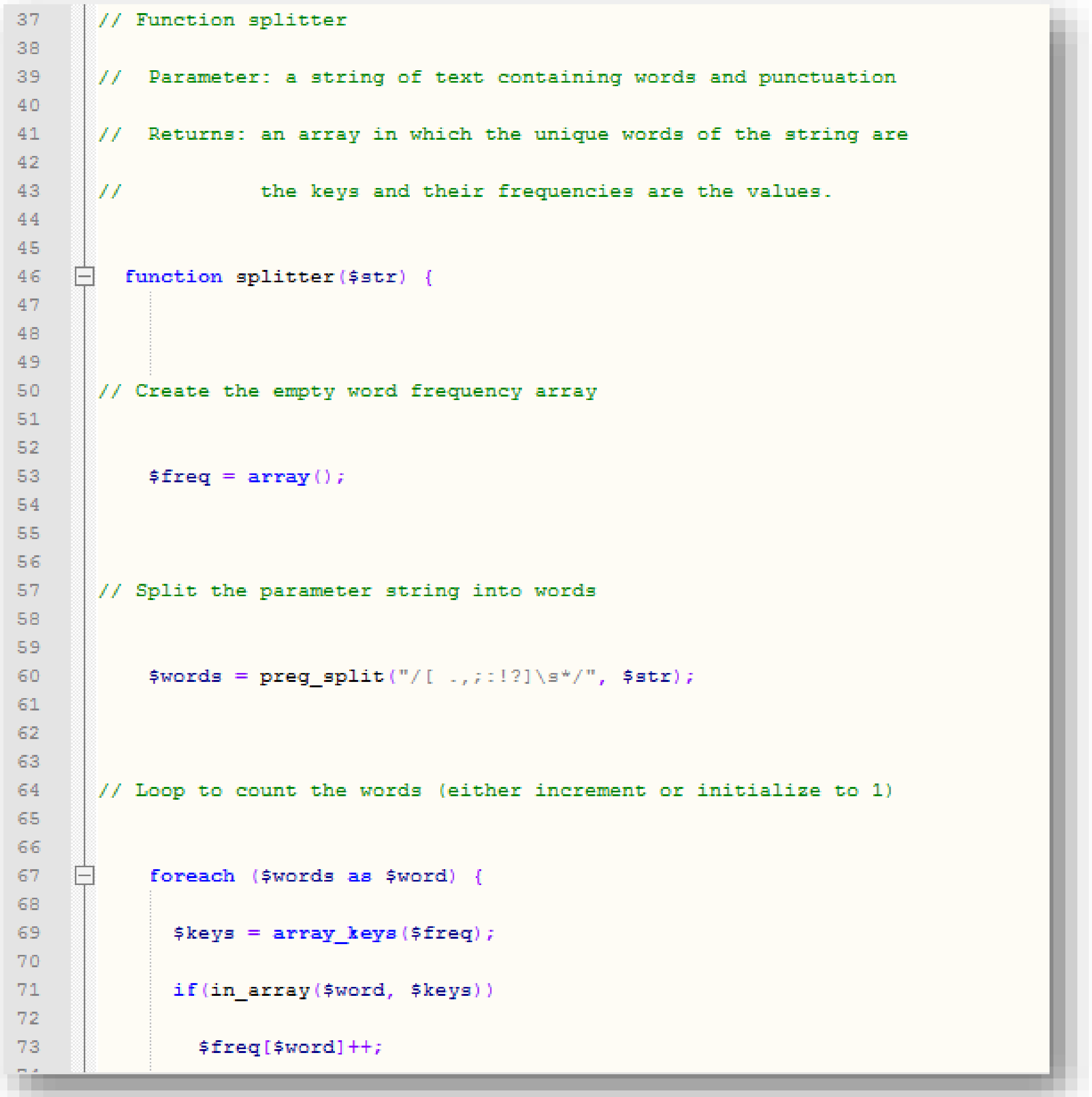
- If a variable name has parentheses appended, PHP will look for a function with the same name as whatever the variable evaluates to, and will then attempt to execute it.
  - Among other things, this can be used to implement ***callbacks***, ***function tables***, and so forth.
- Variable functions won't work with language constructs such as echo(), print(), unset(), isset(), empty(), include(), require() and the like.
  - What can we do if we really need do so?
    - Utilize wrapper functions to wrap up these functions.

Pattern Matching

- PHP provides both POSIX regular expressions and Perl regular expressions
  - These are generally the same but differ in certain details
- The `preg_match` function matches a pattern, given as a string (Perl style), with a string, e.g.,
- If (preg_match(“/^PHOP/”, $str)) print “\$str begins with PHP <br />”;
- The `preg_split` function splits a string into an array of strings based on a pattern describing the separators
- → example `word_table.php` illustrating pattern matching in PHP

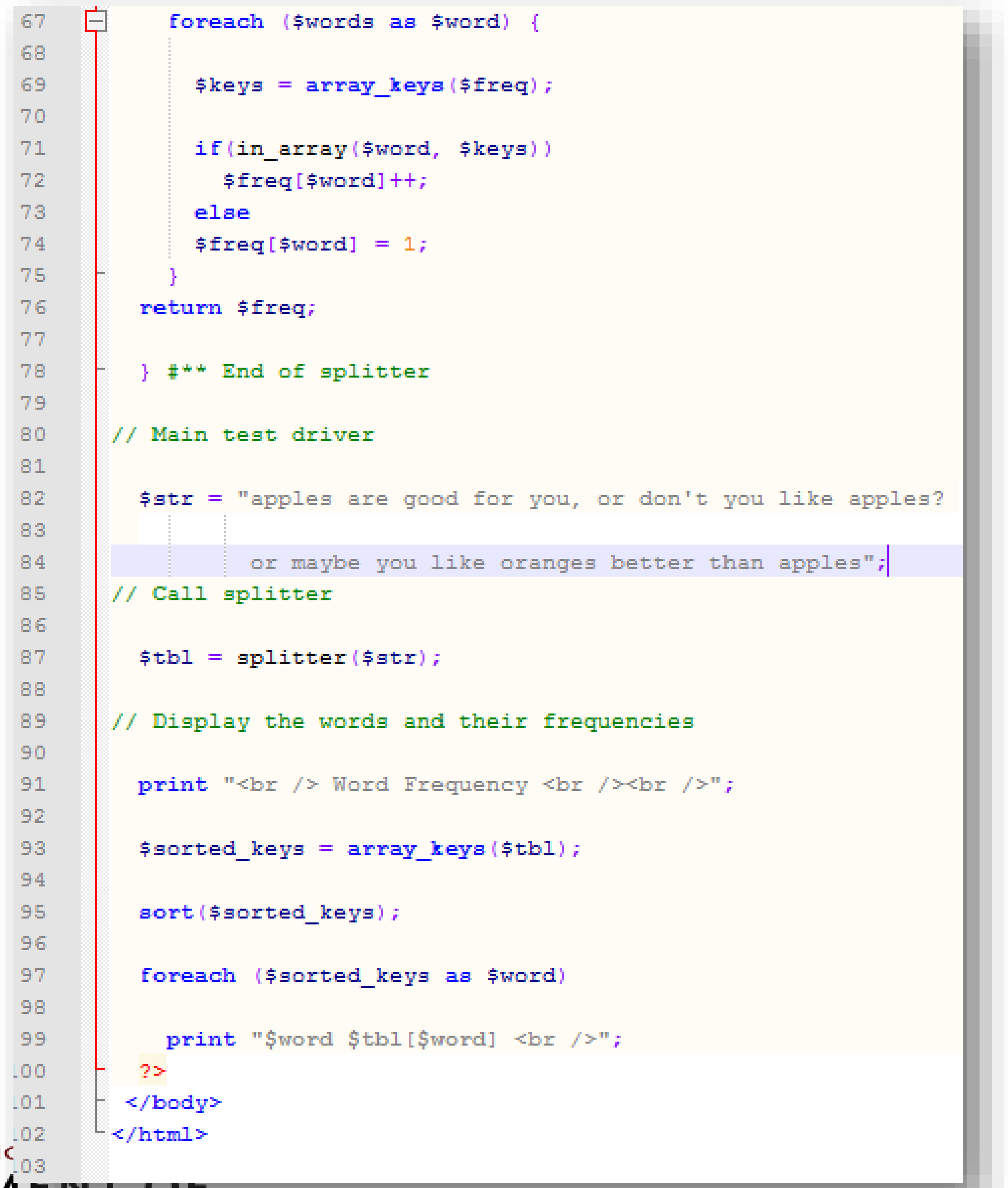


```
<!DOCTYPE html>


<!--    word_table.php

        Uses a function to split a given string of text into

        its constituant words. It also determines the frequency of

        occurrence of each word. The words are separated by whitespace

        or punctuation, possibly followed by whitespace. The

        punctuation can be a period, a comma, a semicolon, a colon,

        an exclamation point, or a questionmark.

        The main driver program calls the function and displays

        the results.

    -->

<html lang = "en">

  <head>
    <title> word_table.php </title>
    <meta charset = "utf-8" />

</head>

  <body>

  <?php

```



```php
// Function splitter

//  Parameter: a string of text containing words and punctuation

//  Returns: an array in which the unique words of the string are

//           the keys and their frequencies are the values.


  function splitter($str) {



// Create the empty word frequency array


    $freq = array();



// Split the parameter string into words


    $words = preg_split("/[ .,;:!?]\s*/", $str);



// Loop to count the words (either increment or initialize to 1)


    foreach ($words as $word) {

      $keys = array_keys($freq);

      if(in_array($word, $keys))

        $freq[$word]++;
```

SOUTHERN ILLINOIS UNIVERSITY
DEPARTMENT OF
COMPUTER SCIENCE



```
    foreach ($words as $word) {

      $keys = array_keys($freq);

      if(in_array($word, $keys))
        $freq[$word]++;
      else
      $freq[$word] = 1;
    }
  return $freq;

  } #** End of splitter

// Main test driver

  $str = "apples are good for you, or don't you like apples?

          or maybe you like oranges better than apples";
// Call splitter

  $tbl = splitter($str);

// Display the words and their frequencies

  print "<br /> Word Frequency <br /><br />";

  $sorted_keys = array_keys($tbl);

  sort($sorted_keys);

  foreach ($sorted_keys as $word)

    print "$word $tbl[$word] <br />";
  ?>
 </body>
</html>
```

SOUTHERN ILLINC
DEPARTMENT OF
COMPUTER SCIENCE

Form Handling

- The values from forms can be accessed in PHP using the $_POST and $_GET arrays
  - Some web servers allow more direct access, though this has security implications

→ example `popcorn3.html` and `popcorn3.php` that implements the popcorn order form using PHP (from textbook)

  - The **`printf`** function is used to get two decimal places printed for currency values

Opening and Closing Files

- The PHP function **`fopen`** is used to create a <u>file handle</u> for accessing a file given by name
  - A second argument to `fopen` gives the <u>mode</u> of access
    - “r”, “r+”, “w”, “w+”, “a”, “a+” (detailed later)
  - The `fopen` function <u>returns a file handle</u>
  - Every open file has a *current* pointer to a point in the file
  - Normally input and output operations occur at the current pointer position
- The **`file_exists`** function tests if a file, given by name, exists
- The function **`fclose`** closes a file handle

File Use Indicators: Access Modes

Mode	Description
"r"	Read only. The file pointer is initialized to the beginning of the file.
"r+"	Read and write an existing file. The file pointer is initialized to the beginning of the file; if a read operation precedes a write operation, the new data is written just after where the read operation left the file pointer.
“w"	Write only. Initializes the file pointer to the beginning of the file; creates the file if it does not exist.
"w+"	Read and write. Initializes the file pointer to the beginning of the file; creates the file if it does not exist. Always initializes the file pointer to the beginning of the file before the first write, destroying any existing data.
"a"	Write only. If the file exists, initializes the file pointer to the end of the file; if the file does not exist, creates it and initializes the file pointer to its beginning.
"a+"	Read and write a file, creating the file if necessary; new data is written to the end of the existing data

Reading from a File

- The `fread` function reads a given number of bytes from a file given by a file handle
  - The entire file can be read by using the `filesize` function to determine the number of bytes in the file
- The `file` function returns an array of lines from a file named as a parameter
  - No explicit open and close are required for using this function, it does not use a file handle parameter either
- The `file_get_contents` func returns the content of a named file as a single string
- The `fgetc` function returns a single character
- The `feof` function returns TRUE if the last character read was the end of file marker

Writing to a File

- If a file handle is open for writing or appending, then the `fwrite` function can be used to write bytes to the file
- The `file_put_contents` function writes a given string parameter to a named file, not a file handler

Locking Files

- The `flock` function will lock a named file
- The function takes a second parameter giving the mode of the lock:
  - 1 specifies **others can read**
  - 2 specifies **no other access** is allowed
  - 3 removes the lock

Cookies

- HTTP is a *stateless* protocol, that is, the server treats each request as completely separate from any other
- This, however, makes some applications difficult
  - A shopping cart is an object that must be maintained across numerous cycles of requests and responses
- The mechanism of cookies can be used to help maintain state by storing some information on the browser system
- A cookie is simply a key/value pair
  - This key/value pair is sent along with any request made by the browser to the same server
- A cookie has a lifetime which specifies a time at which the cookie will be deleted from the browser

Cookies and Security

- Cookies are only returned to the server where they were created
- Cookies can be used to determine usage patterns that might not otherwise be ascertained by a server
- Browsers generally allow users to limit how cookies are used
  - Browsers usually allow users to remove all cookies currently stored by the browser
- Systems that <u>depend on cookies</u> will fail if the browser refuses to store them!!!

PHP Support for Cookies

- PHP provides the `setcookie` function to set a cookie in a response
  - The first parameter is the cookie's name
  - The second, optional, parameter gives the cookie's value
  - The third, optional, parameter gives the expiration
- The cookie must be set before setting content type and before providing any other output
- The `$_COOKIES` array provides access to cookies in the HTTP request (sent back from browsers)
- Example

  setcookie ("voted", "yes", time()+8888); // in seconds

Session Tracking

- Some applications need to keep track of a session
- Sessions are identified internally in PHP via a session id
    - A session consists of key/value pairs, too
    - Well, what is the difference?
- A session can be initialized or retrieved by using the
- `session_start` function
    - First time call will assign a session id (**implicitly**)
    - This function retrieves `$_SESSION`, an array containing the key/value pairs in the current request
-
    - E.g.,
- if (!IsSet($_SESSION["page_number"])) $_SESSION["page_nubmer"] = 1;

A Session Example

```
<?php
// page1.php

session_start();
    if (!IsSet($_SESSION[“page_number”]))
       $_SESSION[“page_number”] = 1;
    $page_num = $_SESSION[“page_number”];
    print("You have now visited $page_num page(s)”);
    print(" pages <br />");
    $_SESSION[“page_number”]++;
     …
?>
```

Lab (Parts 1 and 2)

- Write a XHTML code to create a form that collects favorite popular songs, including the name of the song, the composer and the performing artist or group. This document must call one PHP script when the form is submitted and another to request a current list of survey results.

Part 1. Write a PHP scripts that collect the data from the form and save it in a file.

Part 2. Write a PHP scripts that produces the current result from the form.