Introduction to Databases

PHP Functions and Objects

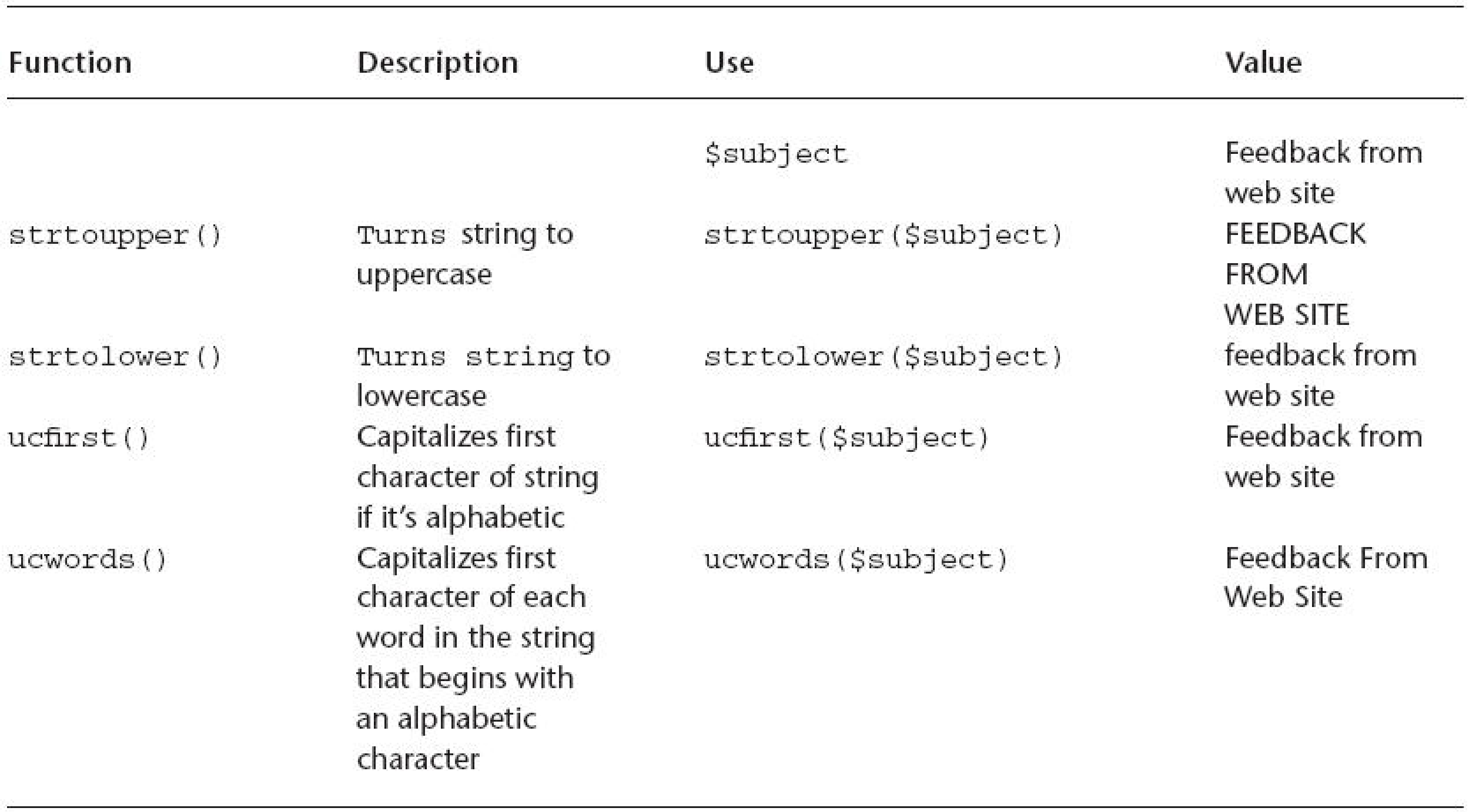
Key Concepts

- Formatting strings
- String functions
- Regular expressions
- require() and include()
- Functions
- Parameters
- Classes

Functions for Formatting Strings

- Trimming strings
  `trim(), ltrim(), rtrim()`
- HTML formatting
  `nl2br()`
  `Example:`
  ```
  <?php
  echo nl2br("foo isn't\n bar");
  ?>
  ```
  `Output:` foo isn't<br />
  bar
- Formatting for print or display
  `printf(), sprintf()`
  - Example:
  ```
  printf("Total amount of order is %.2f (with
    shipping of %.2f) ", $total, $total_shipping);
  ```

Case Functions



Function	Description	Use	Value
		`$subject`	Feedback from web site
`strtoupper()`	Turns string to uppercase	`strtoupper($subject)`	FEEDBACK FROM WEB SITE
`strtolower()`	Turns string to lowercase	`strtolower($subject)`	feedback from web site
`ucfirst()`	Capitalizes first character of string if it's alphabetic	`ucfirst($subject)`	Feedback from web site
`ucwords()`	Capitalizes first character of each word in the string that begins with an alphabetic character	`ucwords($subject)`	Feedback From Web Site

Strings and Storage

- Escape characters that would cause problems in a database:
  `addslashes()`
- Remove escape characters for proper display
  `stripslashes()`

Joining and Splitting Strings

- Splitting strings

  `explode(), strtok(), substr()`

- Joining strings

  `implode(), join()`

Comparing Strings

- Useful for sorting strings

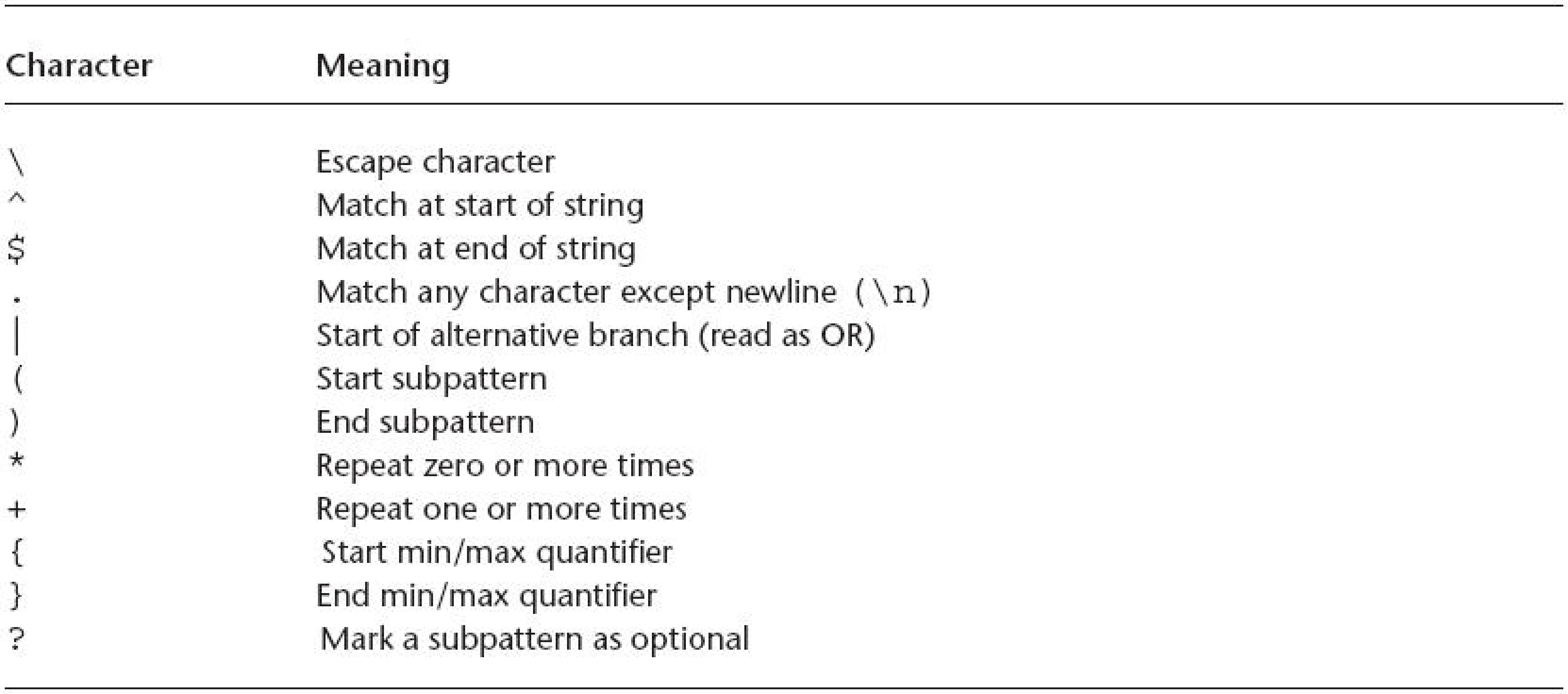
  `strcmp(), strcasecmp(), strnatcmp()`
- Testing string length

  `strlen()`

Matching and Replacing Substrings

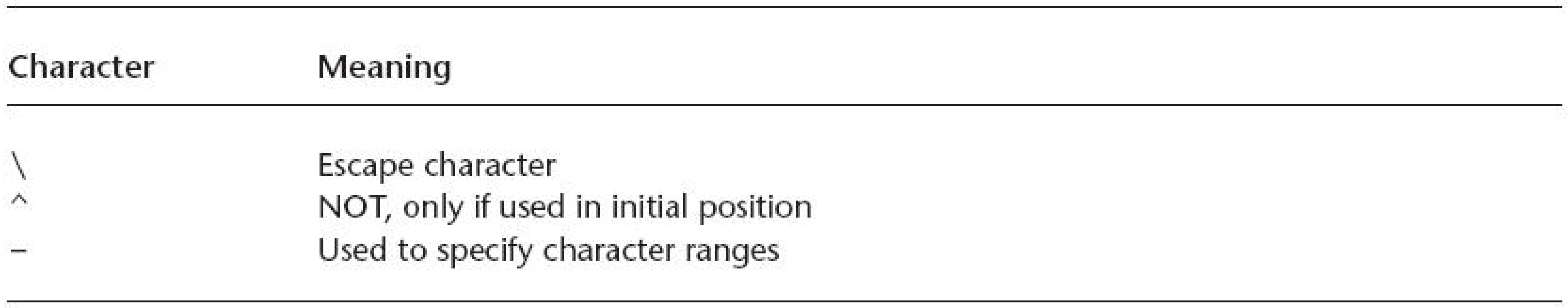
- Finding strings in strings
  `strstr(), strchr(), strrchr(), stristr()`
- Finding the position of a substring
  `strpos(), strrpos()`
- Replacing strings
  `str_replace(), substr_replace()`

Regular Expression Characters Used Outside Square Brackets



Character	Meaning
\	Escape character
^	Match at start of string
$	Match at end of string
.	Match any character except newline (`\n`)
|	Start of alternative branch (read as OR)
(	Start subpattern
)	End subpattern
*	Repeat zero or more times
+	Repeat one or more times
{	Start min/max quantifier
}	End min/max quantifier
?	Mark a subpattern as optional

Regular Expression Characters Used Inside Square Brackets



Character	Meaning
\	Escape character
^	NOT, only if used in initial position
–	Used to specify character ranges

Finding Strings Using Regular Expressions

```
if (!eregi('^[a-zA-Z0-9_\-\.]+@[a-zA_Z0-9]+\.[a-zA_Z0-9\-\.]+$', $email)) {
    echo "<p>That is not a valid email address.</p>".
          "<p>Please return to the previous page and try again.</p>";
    exit;
}
$toaddress = "feedback@example.com"; // the default value
if (eregi("shop|customer service|retail", $feedback))
    $toaddress = "retail@example.com";
} else if (eregi("deliver|fulfill", $feedback)) {
    $toaddress = "fulfillment@example.com";
} else if (eregi("bill|account", $feedback)) {
    $toaddress = "accounts@example.com";
}
if (eregi("bigcustomer\.com", $email)) {
    $toaddress = "bob@example.com";
}
```

Splitting a String Using a Regular Expression

```
string ereg_replace(string pattern, string replacement, string search);
```

```
array split(string pattern, string search[, int max]);
```

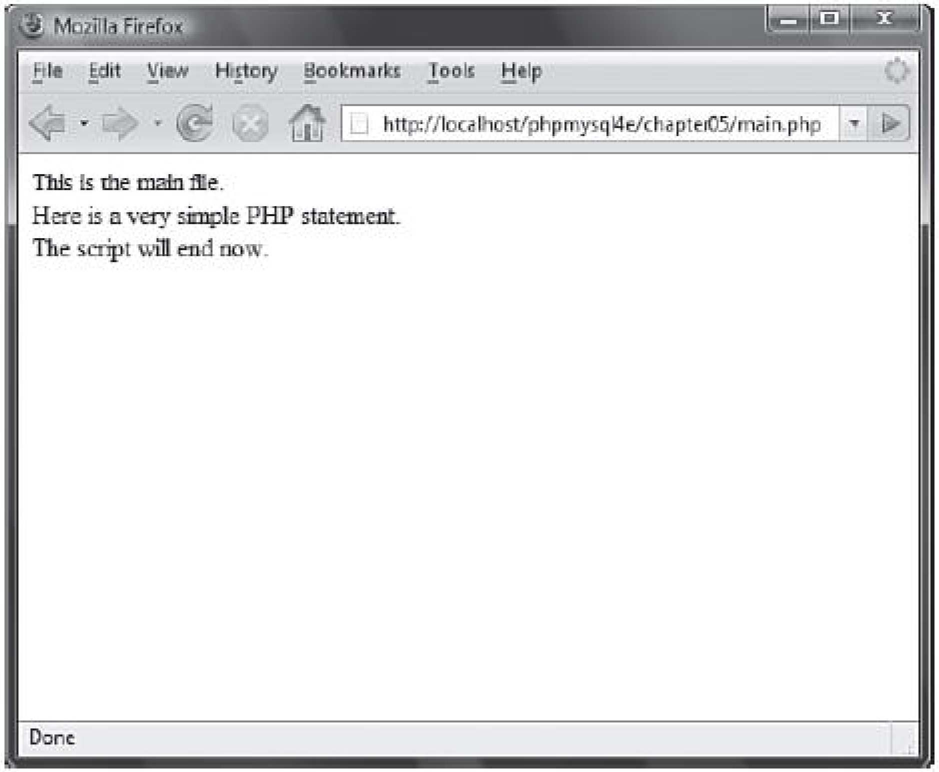
```
$address="user@example.com";
$arr = split("\.|@",$address);
while(list($key, $value) = each($arr))
   echo "<br/>".$value;
```

```
user
example
com
```

Reusing Code with require()

```
<?php
  echo 'Here is a very simple PHP statement.<br />';
?>
```

```
<?php
  echo 'This is the main file.<br />';
  require( 'reusable.php' );
  echo 'The script will end now.<br />';
?>
```


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/phpmysql4e/chapter05/main.php
Done

```
This is the main file.
Here is a very simple PHP statement.
The script will end now.
```

Uses for require() and include()

- Reusing PHP scripts
- Adding a PHP-generated header to every page
- Adding a PHP-generated footer to every page
- Using classes

Using HTML Within a Function

```
<?php
  function my_function() {
?>
  My function was called
  <?php
  //Execute more PHP code
  }
?>
```

Using Parameters with a PHP Function

```
<?php
    $people = array("Peter", "Joe", "Glenn", "Cleveland");

    echo current($people) . "<br />";
    echo next($people) . "<br />";

    echo reset($people);
    ?>
```

Output:

```
Peter
Joe
Peter
```

Passing Parameters by Reference

```
function increment (&$value, $amount=1)
{
  $value = $value + $amount;
}

increment($value);
```

Returning Values from Functions

```
function larger ($x, $y) {
  if ((!isset($x)) || (!isset($y)))
     return false;
  else if ($x>=$y)
     return $x;
  else
     return $y;
}
```

Creating a Class

```
class classname
{
  public $attribute1;
  function operation1()
  {
  }
}
```

Instantiating a Class

```
class classname
{
  function classname($param="default")
  {
     echo "Constructor called with
  parameter ".$param."<br/>";
  }
}
$a = new classname("First");
$b = new classname("Second");
$c = new classname();
```

Class Attributes and Functions

```
class classname
{
  public $attr;
  function operation($param)
  {
     $this->attr = $param
  }
}
$a = new classname();
$a->attr = "value";
$a->operation(10);
```

Access Modifiers

- public
  - Default value
  - Accessible outside class
- private
  - Accessible only within class
- protected
  - Accessible only with class or subclass

Inheritance

```
class A
{
  public $attr1;
  function op1()
  {
  }
}
```

```
class B extends A
{
  public $attr2;
  function op2()
  {
  }
}
$myB = new B();
$myB->attr1 = 5;
$myB->attr2 = 10;
```

Overriding

```
class A
{
  public $attr1=0;
  function op1()
  {
    return $this->attr1;
  }
}
```

```
class B extends A
{
  public $attr1=2;
  function op1()
  {
    return ($this->attr1 * 2);
  }
}
$myA = new A();
$myB = new B();
echo $myA->op1();
echo $myB->op1();
```

The final Keyword

Prevent inheritance

```
final class A()
{
}
```

Prevent override

```
class A
{
  public $attr;
  final function op()
  {
     echo "Something<br/>";
  }
}
```

Interfaces

Interface definition

```
interface Displayable
{
  function display();
}
```

Class implements interface

```
class A implements
  Displayable
{
  function display()
  {
     //code
  }
}
```