

# PERL Basics In Pictures

by Paul Gruhn



**in pictures**

[www.inpics.net](http://www.inpics.net)

# PERL Basics In Pictures

## Copyright



This book is provided under a Creative Commons license at:  
[creativecommons.org/licenses/by-nc-nd/2.5/](http://creativecommons.org/licenses/by-nc-nd/2.5/)

You are free to download, copy, and share this electronic book with others.

However, it is illegal to sell this book, or change it in any way.

If you'd like to sell or change it, just contact us at [contact@inpics.net](mailto:contact@inpics.net).

## Trademarks and Disclaimer

Visibooks™ is a trademark of Visibooks, LLC. All brand and product names in this book are trademarks or registered trademarks of their respective companies.

Visibooks™ makes every effort to ensure that the information in this book is accurate. However, Visibooks™ makes no warranty, expressed or implied, with respect to the accuracy, quality, reliability, or freedom from error of this document or the products described in it. Visibooks™ makes no representation or warranty with respect to this book's contents, and specifically disclaims any implied warranties or fitness for any particular purpose. Visibooks™ disclaims all liability for any direct, indirect, consequential, incidental, exemplary, or special damages resulting from the use of the information in this document or from the use of any products described in it. Mention of any product does not constitute an endorsement of that product by Visibooks™. Data used in examples are intended to be fictional. Any resemblance to real companies, people, or organizations is entirely coincidental.

ISBN 1597061077

# Table of Contents

<b>Learning the Basics .....</b>	<b>1</b>
Install an FTP program .....	2
Create a simple script.....	14
Upload a script .....	20
Set script permissions.....	24
Run a script from a Web page.....	26
Insert comments.....	31
Format text output with HTML tags .....	34
<b>Working with Variables.....</b>	<b>41</b>
Employ single variables .....	43
Print quotation marks .....	54
Employ lists of variables .....	63
<b>Working with Numbers .....</b>	<b>71</b>
Perform calculations.....	72
Increment/decrement automatically .....	75
Generate random numbers .....	78

<b>Subroutines .....</b>	<b>83</b>
Create a subroutine .....	84
Parse form data with a subroutine .....	87
Parse form data.....	89
<b>Logic &amp; Loops.....</b>	<b>95</b>
Employ conditional logic .....	96
Employ looping.....	116
<b>Working With Files.....</b>	<b>123</b>
Create a text file .....	124
Display files.....	130
Append to files.....	132

# Learning the Basics

In this section, you'll learn how to:

- **Install an FTP program**
- **Create a simple script**
- **Upload a script**
- **Set script permissions**
- **Run a script from a Web page**
- **Insert comments**
- **Format text output with HTML tags**

# Install an FTP program

- 1.** Open your Web browser and go to:  
**[www.ipswitch.com](http://www.ipswitch.com)**
- 2.** Download and install WS\_FTP Home.

## **WS\_FTP**

FTP stands for File Transfer Protocol, a way to transfer files between computers over the Internet. If you have trouble configuring FrontPage to upload pages to a Web server, use an FTP program.

Using an FTP program is the most straightforward way to upload a Web site to a Web server. WS\_FTP is the most popular FTP program used to upload and download Web pages.

The Home version is free to use for 30 days, and can be downloaded at [www.ipswitch.com](http://www.ipswitch.com).

### 3. Open WS\_FTP Home.

The Connection Wizard should open.

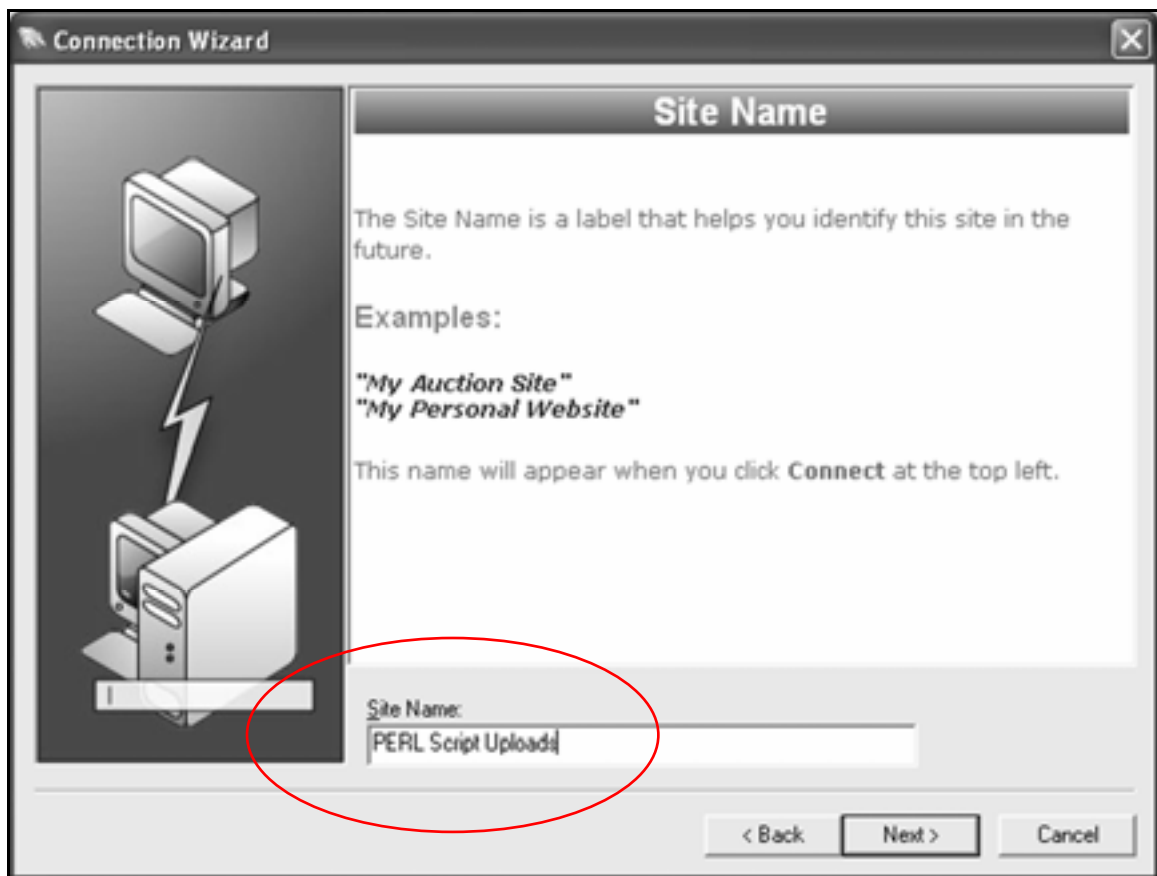


Click the  button.

**4.** When the **Site Name** screen appears, type:

### **Perl Script Uploads**

in the **Site Name** box.



Then click the  button.



- 5.** When the **Server Address** screen appears, type the host address of your server in the **Server Address** box.

It can be something like:

**www.inpics.net**

**washington.patriot.net**

**207.176.7.217**



Then click the **Next >** button.

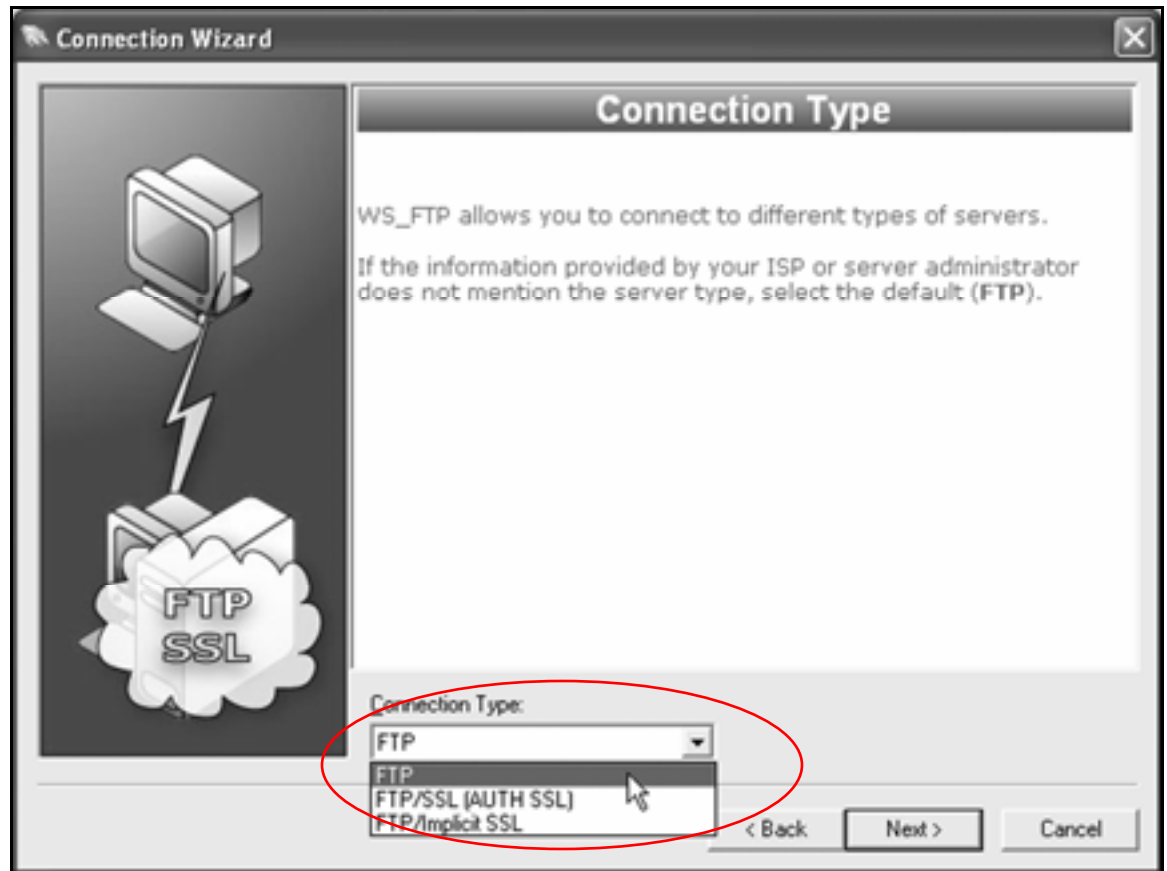
**Tip:** *You can get the Server Address of your Web site, as well as your username and password, from your Web server administrator.*

- 6.** When the **User Name and Password** screen appears, type in your username and password.



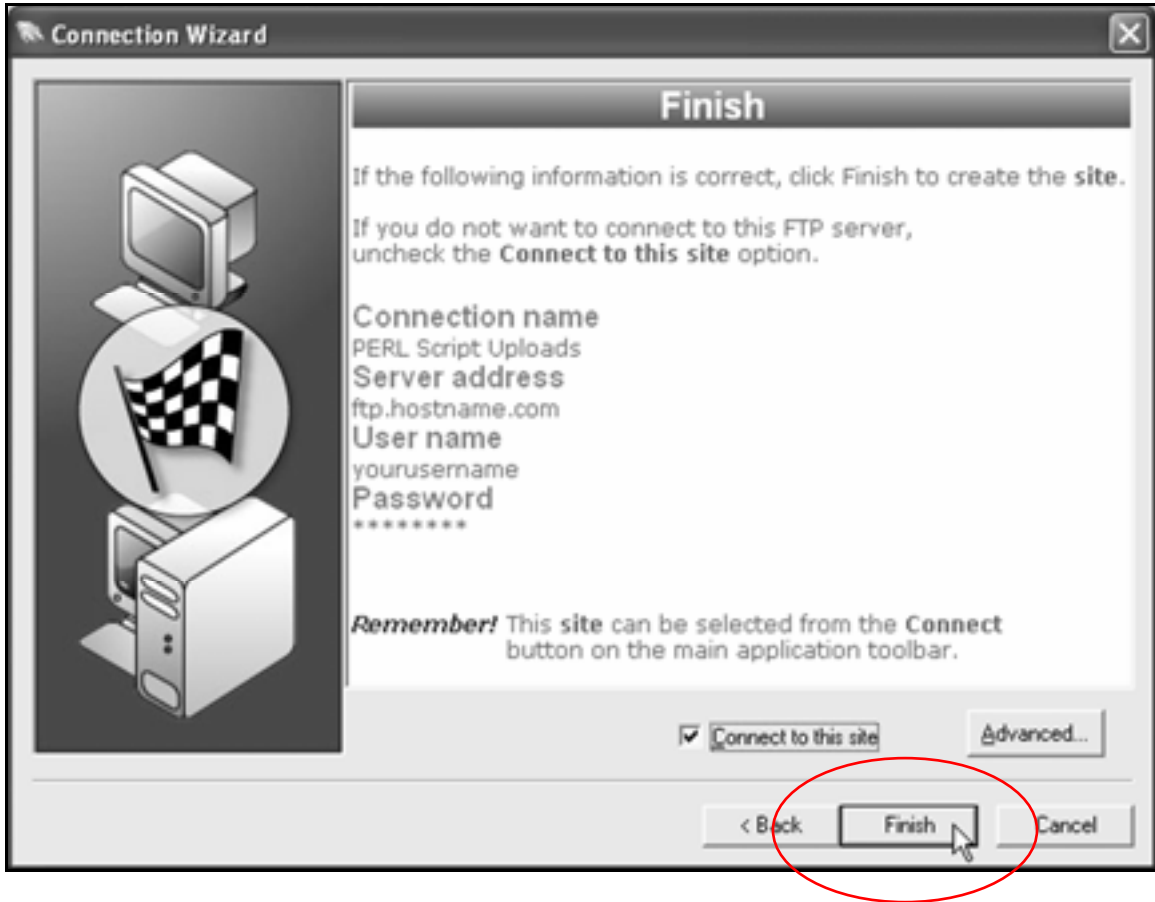
Then click the  button.

- 7.** When the **Connection Type** screen appears, leave the connection type set at **FTP**.

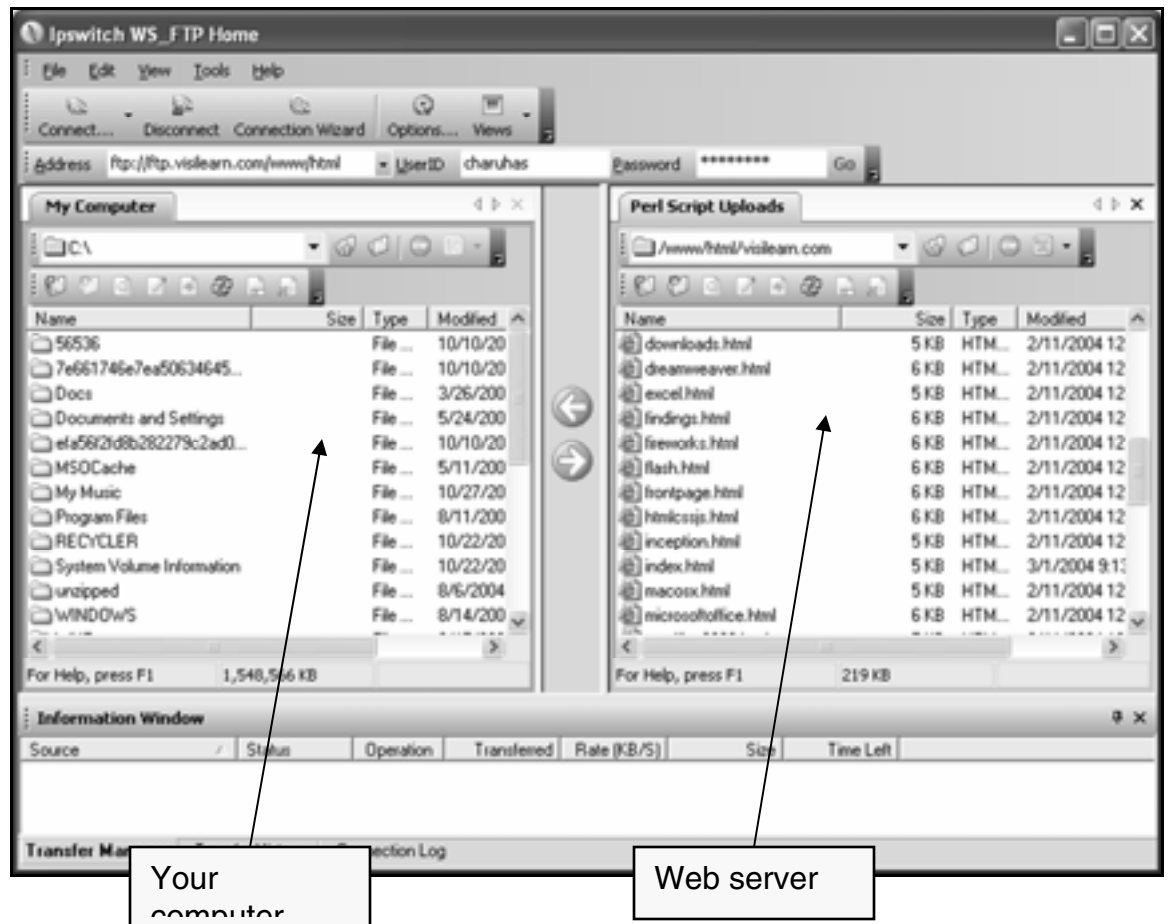


Then click the  button.

**8.** When the **Finish** screen appears, click the **Finish** button.

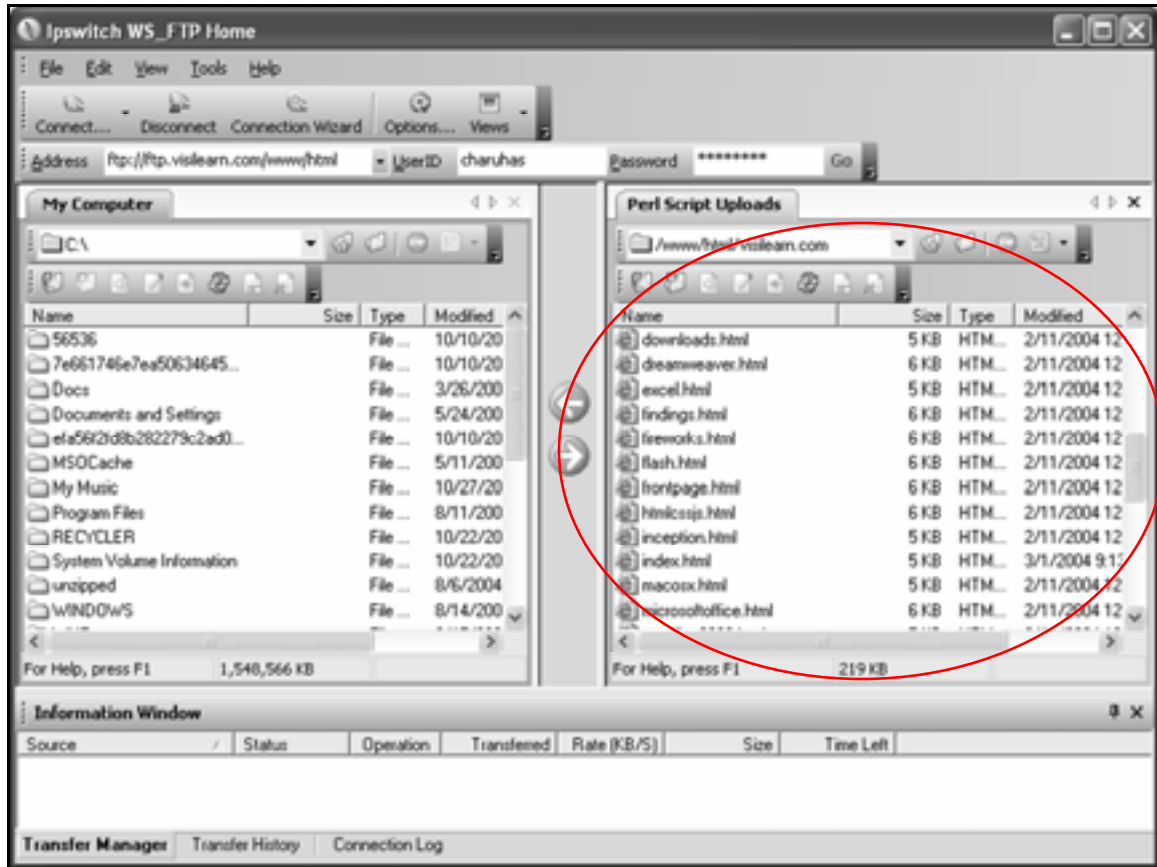


WS\_FTP should connect to your Web server:

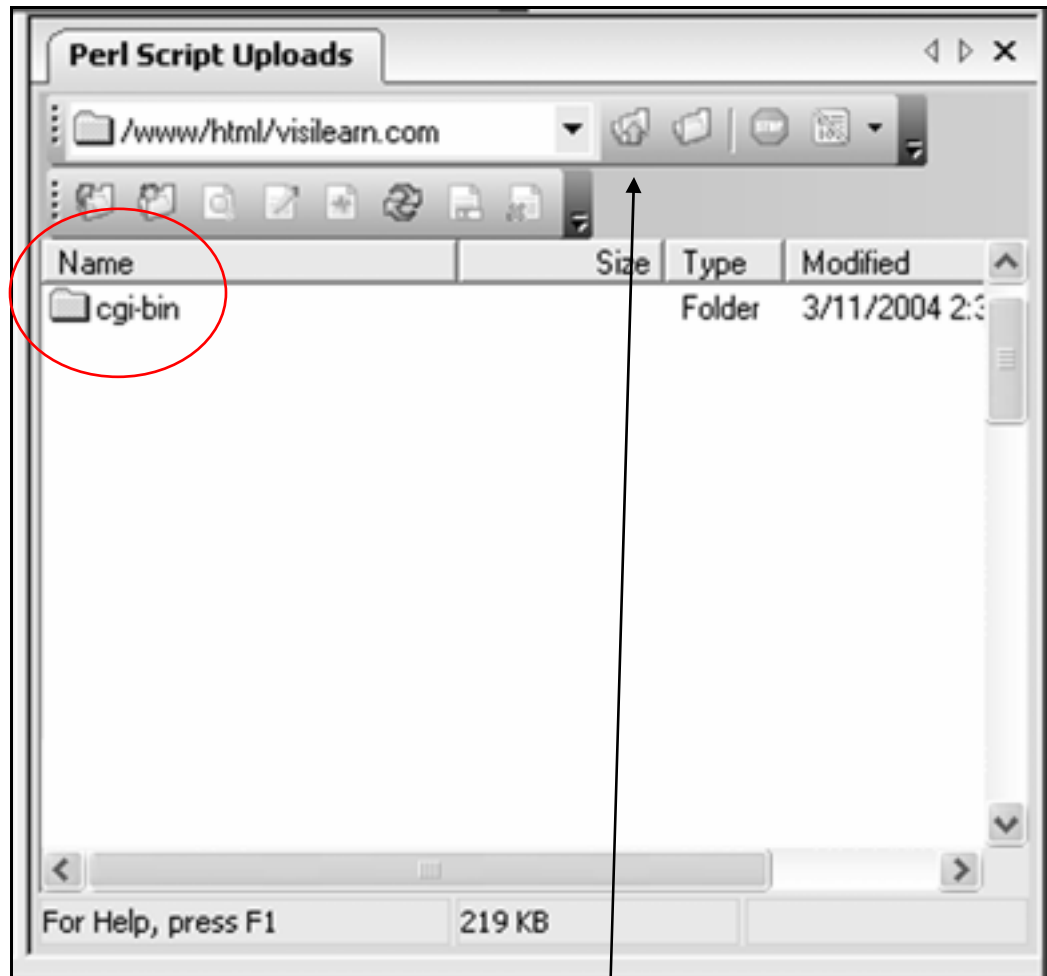


9. In the right-hand **Perl Script Uploads** pane, double-click on the **public\_html** folder, **html** folder, or the folder that contains your Web pages on the server.

You should now see the contents of your Web site on the server:




- 10.** In the right-hand **Perl Script Uploads** pane, navigate to the **cgi-bin** directory of your Web site.

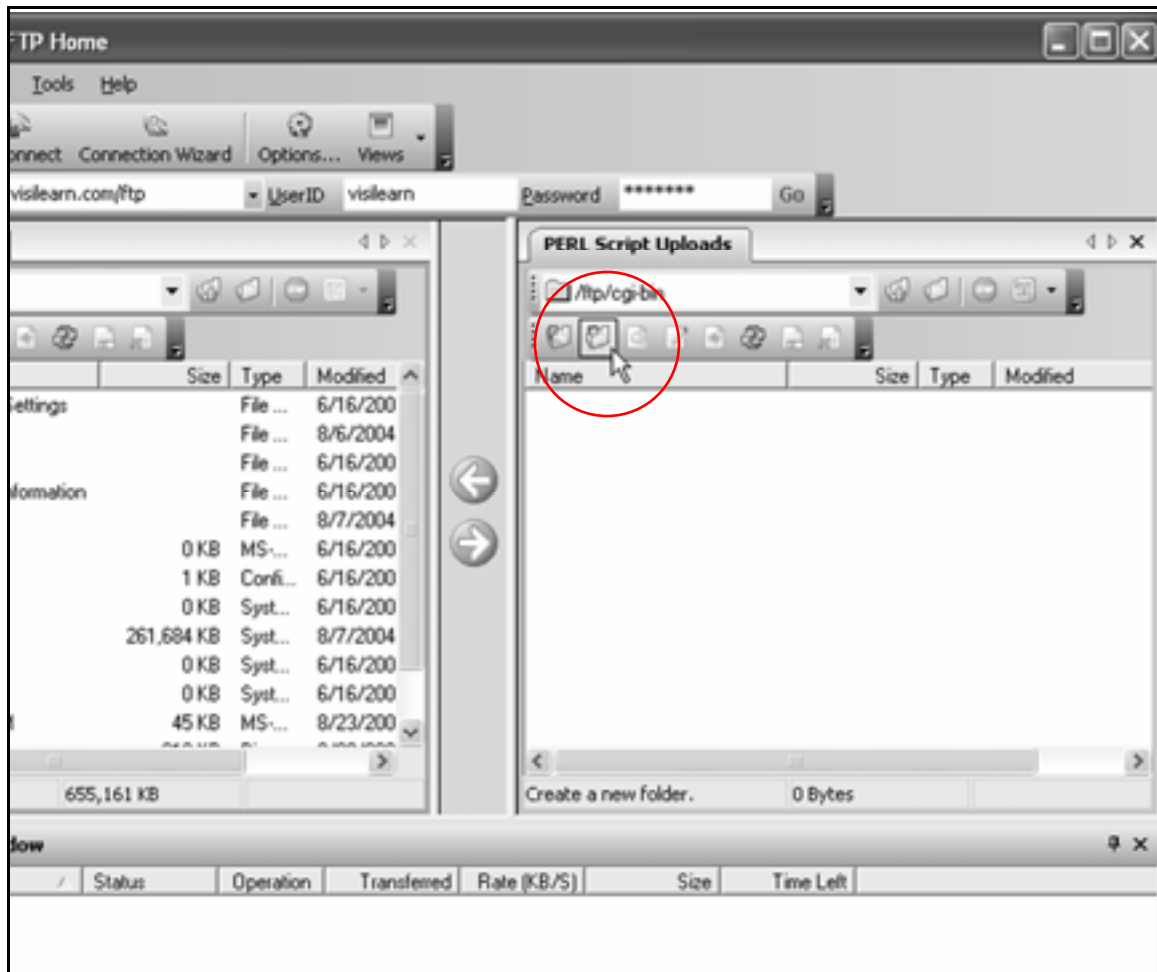


**Tip:** You may have to click the  icon to move up in the site hierarchy.

- 11.** Double-click the **cgi-bin** directory to open it.

**Tip:** Many Internet Service Providers require you to place all PERL scripts into a separate **cgi-bin** directory. This is a good security practice because it hides your scripts from the rest of the world.

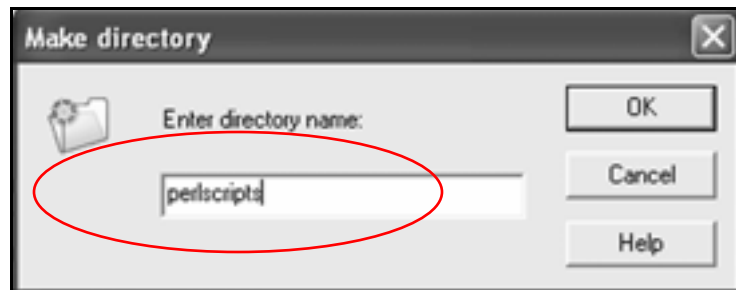
**12.** Click the  icon.





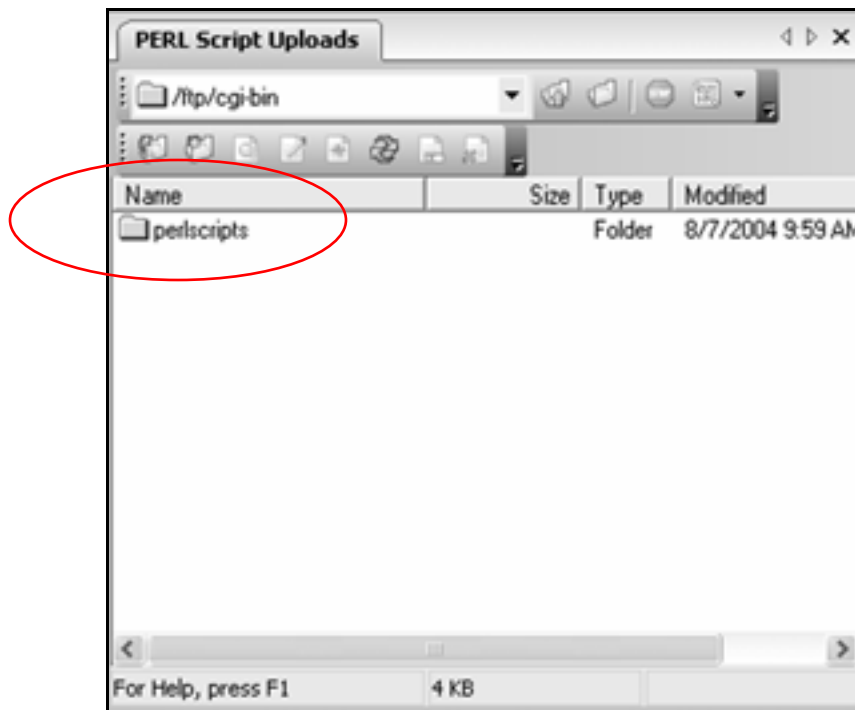
- 13.** When the **Make directory** window appears, type:  
**perlscripts**

in the textbox.



- 14.** Click the  button.

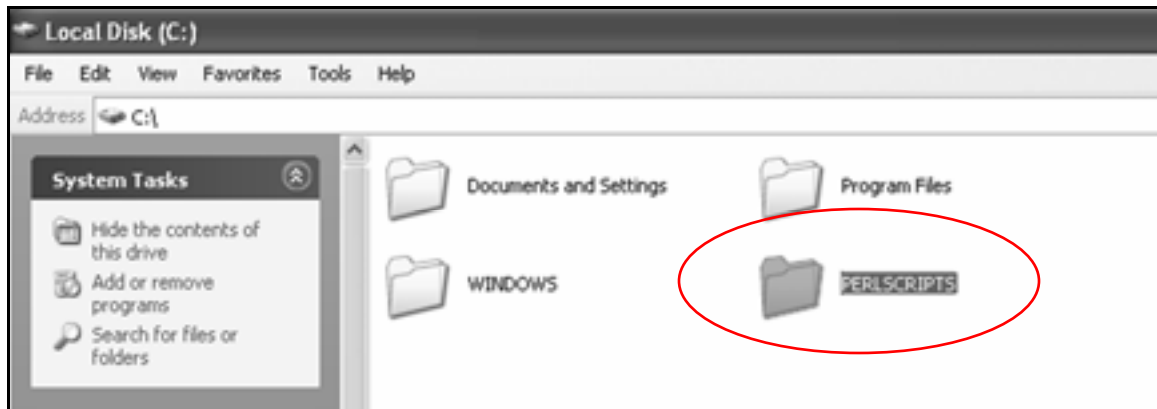
You should now see a directory called **perlscripts** in the right pane:



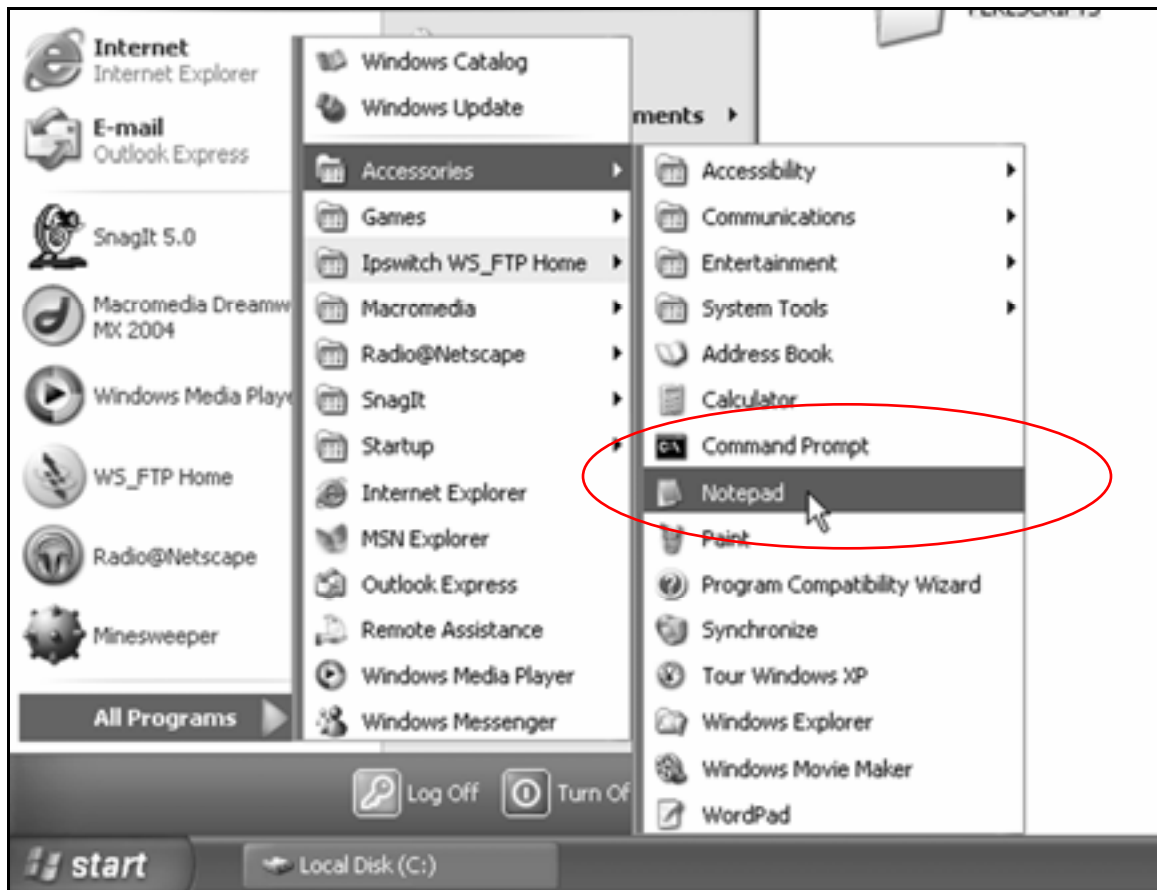
- 15.** Close WS\_FTP.

# Create a simple script

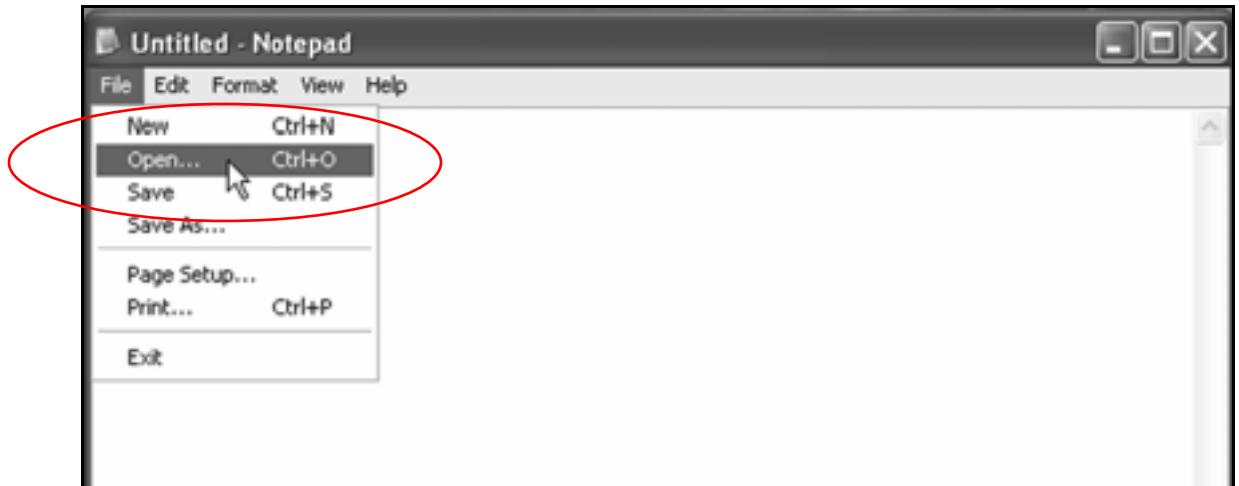
1. Create a folder called **PERLSCRIPTS** on your hard drive.



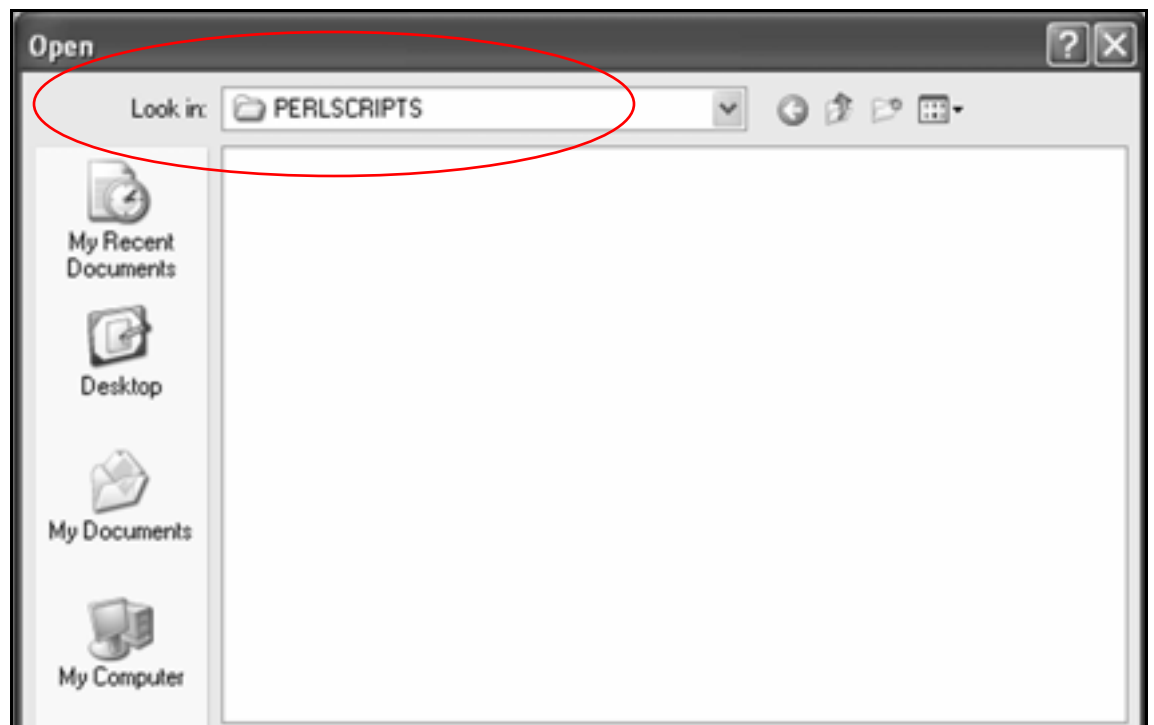
2. Open the Notepad program on your computer.



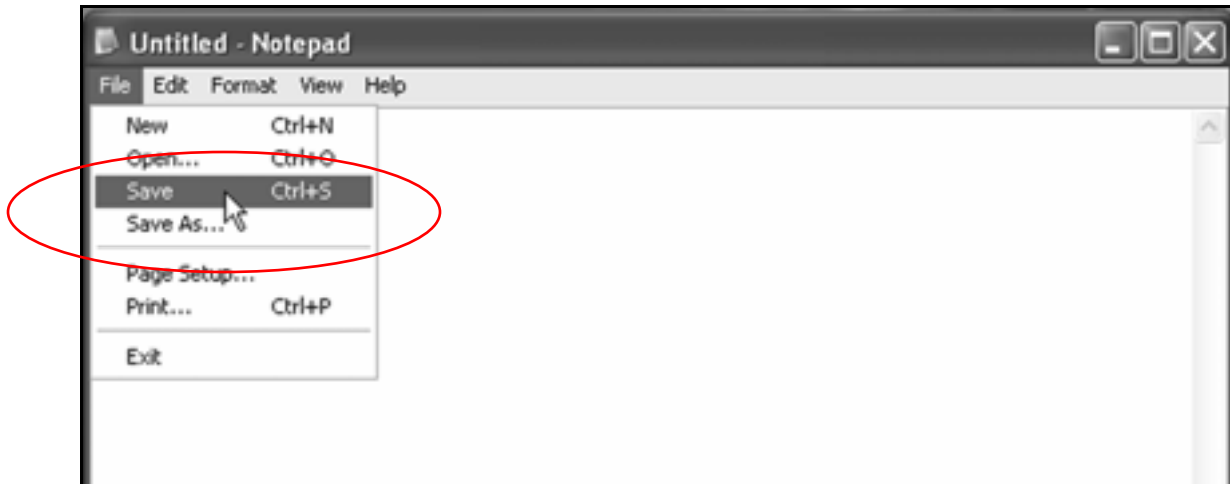
- 3.** Click **File**, then **Open**.



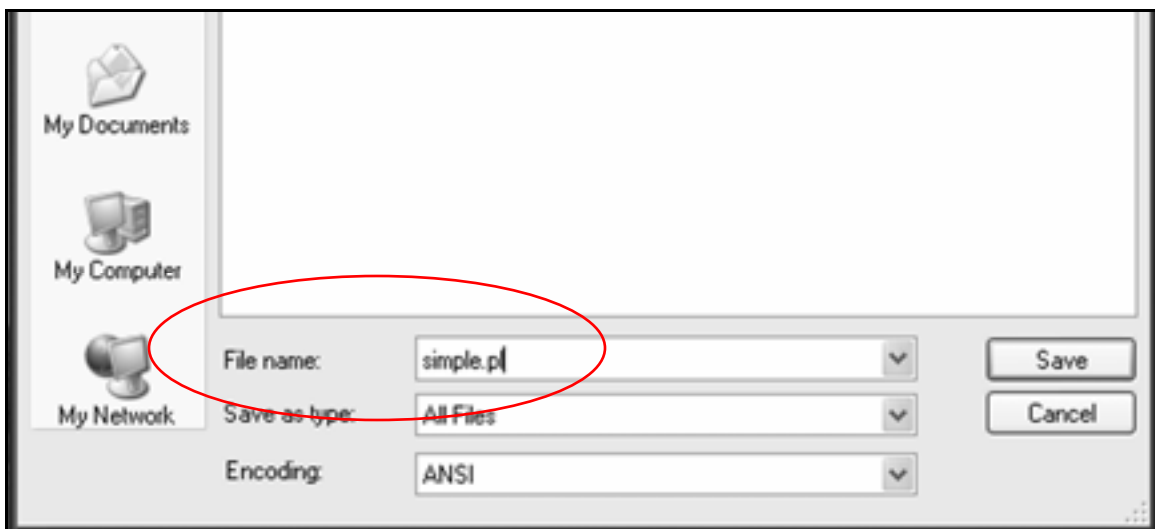
- 4.** When the **Open** window appears, navigate to the **PERLSCRIPTS** folder on your hard drive, then double-click it.  
It should appear in the **Look in** box.



**5.** Click **File**, then **Save**.



**6.** When the **Save As** window appears, type:  
**simple.pl**  
in the **File Name** textbox.



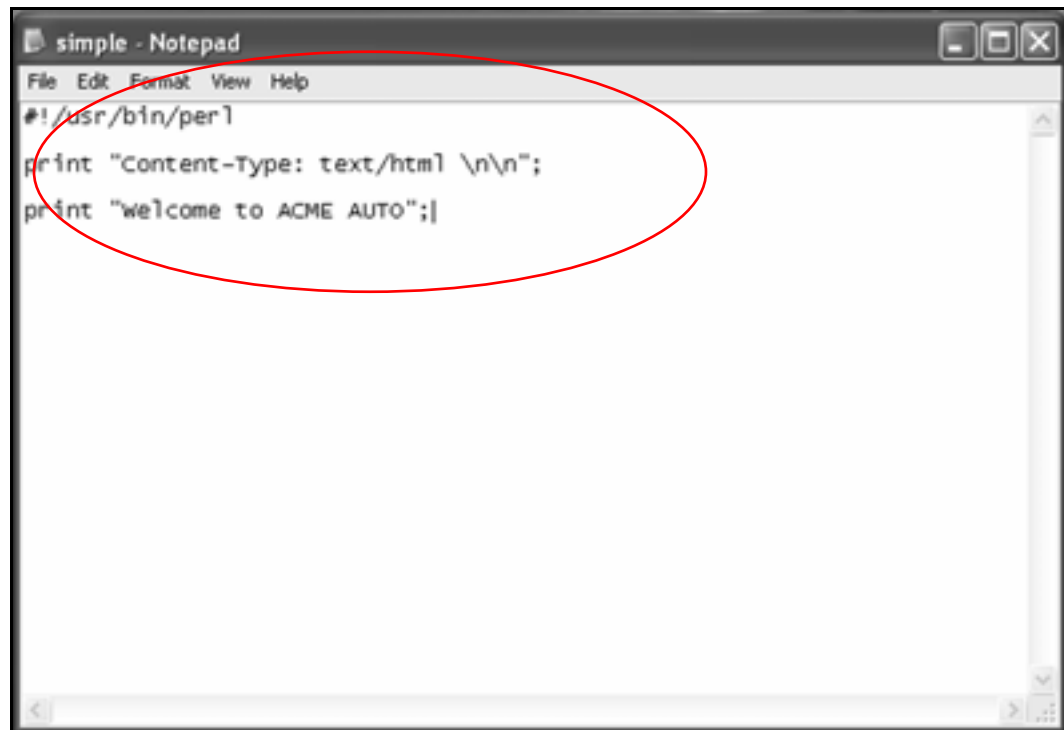
**7.** Click the  button.

**8.** In the blank document window, type:

```
#!/usr/bin/perl
```

```
print "Content-Type: text/html \n\n";
```

```
print "Welcome to ACME AUTO";
```



**Tip:** You're now typing commands to the Web server in the PERL language. Sometimes these commands are case-sensitive. Use lower-case for PERL commands—that is, everything not enclosed in quotation marks, like

```
"Content-Type: text/html \n\n."
```

Also, don't forget to type a semicolon (;) at the end of each line. For your commands to work, or "execute," they need a semicolon (;) at the end.

## 9. Save the script.

Here's what each line of this PERL script does:

- `#!/usr/bin/perl`

`#!/usr/bin/perl`

This first line states the location of the PERL module in your Web site. This module lets your Web server understand PERL commands.

Contact the company/person who runs your Web server to be sure you have the correct path to the PERL module.

In this case, the PERL module is in a directory on the Web server called `perl`, which is in the `bin` directory, which is contained within the `usr` directory.

This path **MUST** be the first line of all your PERL scripts.

- **(blank line)**

Before the next line of code is a blank line. You can use blank lines throughout your PERL scripts.

Blank lines allow you to group sections of code together, which makes scripts easier to read.

- `print "Content-Type: text/html \n\n";`  
`print "Content-Type: text/html \n\n";`

This `print` command tells the Web server to “print” a line of text to a Web browser.

```
print "Content-Type: text/html \n\n";
```

This line tells the web browser that what comes next is HTML, or text that can be read by a Web browser.

```
print "Content-Type: text/html \n\n";
```

The `\n` character tells the Web browser to print the HTML code that follows on a new line.

Since there are two new lines specified, a blank line is printed between this line of code and the next.

- `print "Welcome to ACME AUTO";`  
`print "Welcome to ACME AUTO";`

This `print` command prints the words between the quotes to the browser window.

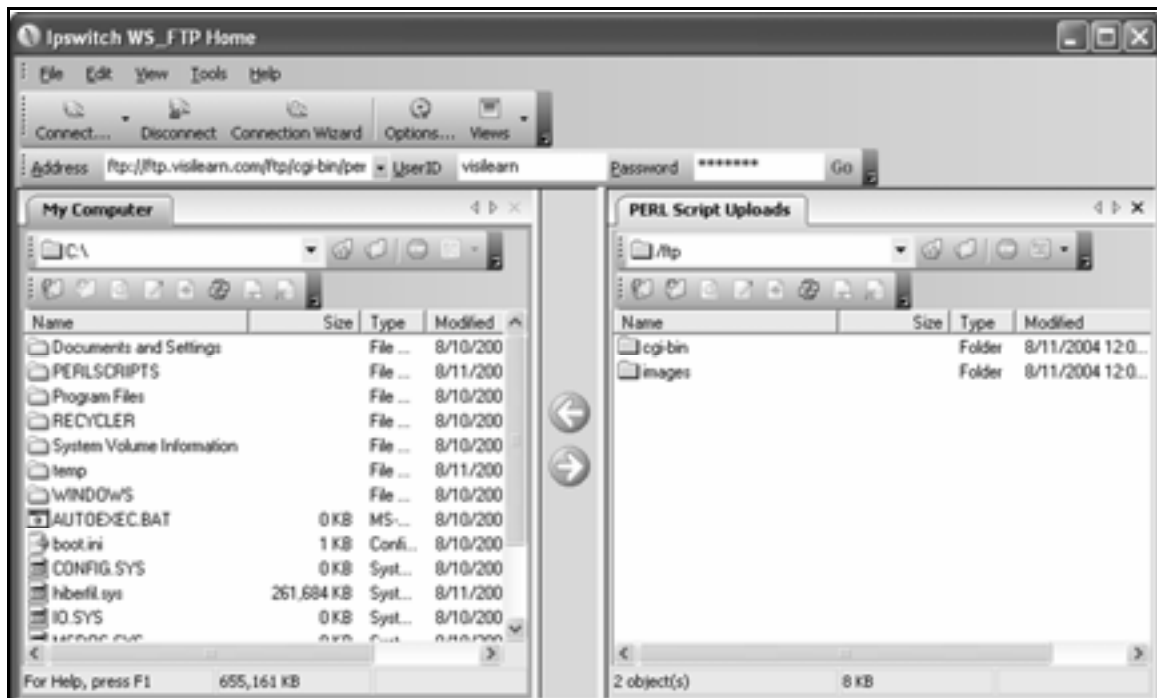
```
print "Welcome to ACME AUTO";
```

Remember: for a command string to execute, there must be a semicolon (;) at the end.

# Upload a script

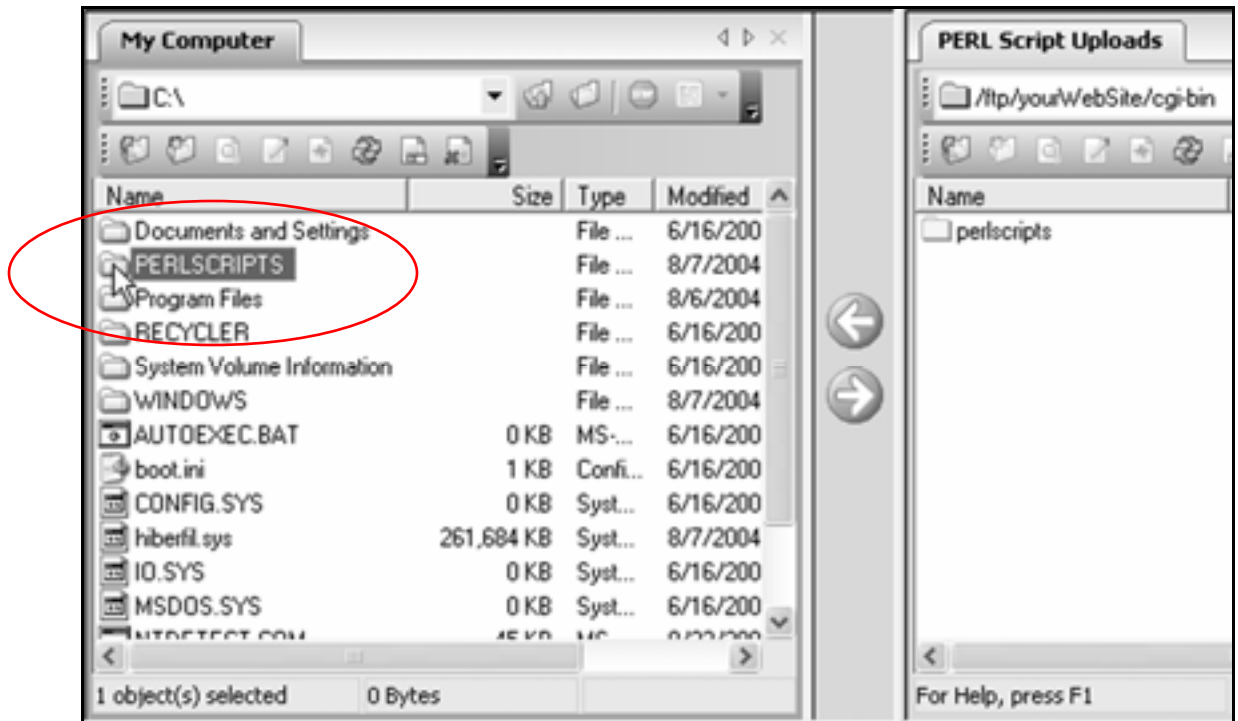
1. Open WS\_FTP and navigate to the home directory on your Web server.

It should look something like this:

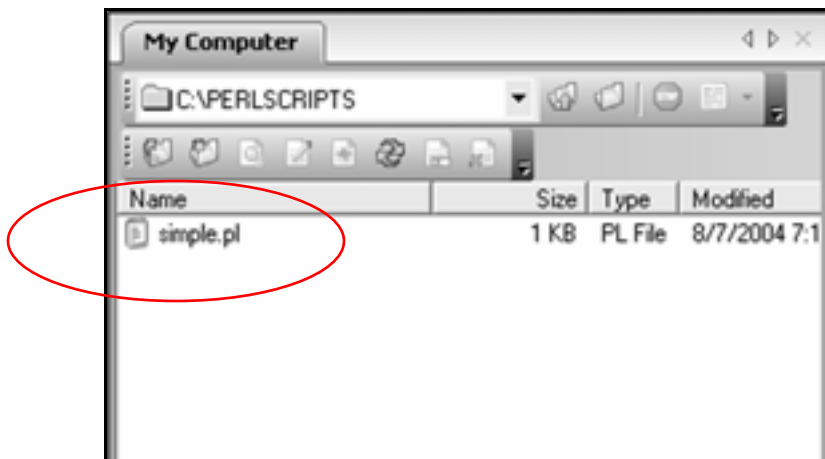




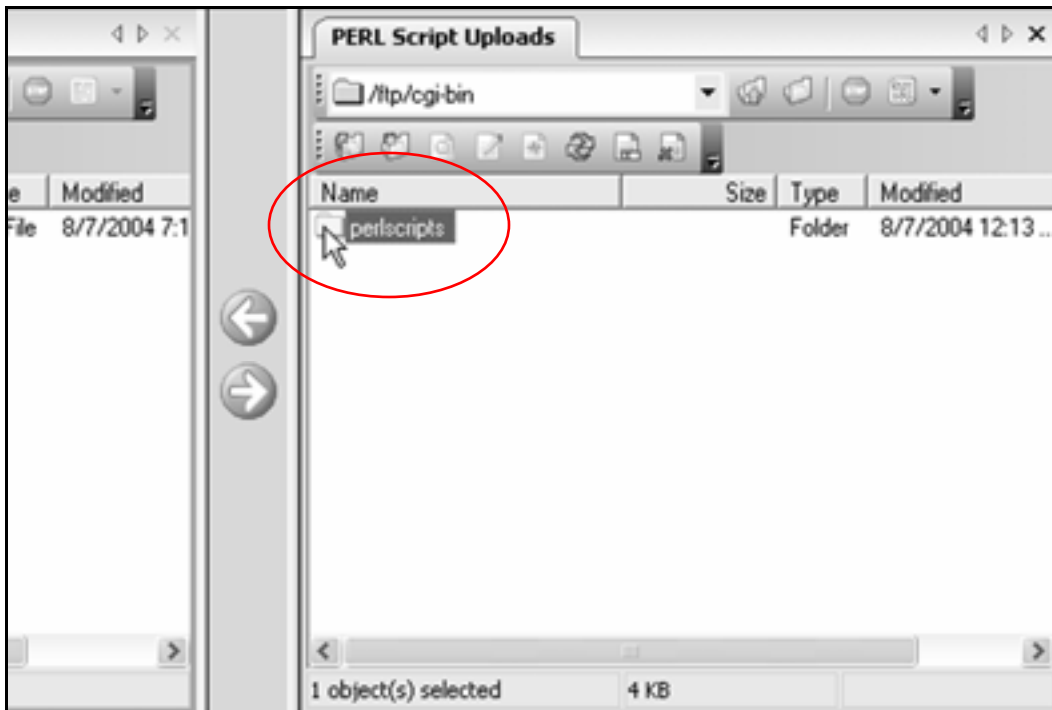
- 2.** In the left-hand **My Computer** pane, navigate to the **PERLSCRIPTS** folder on your computer.



- 3.** Double-click the **PERLSCRIPTS** folder.  
**simple.pl** should appear.

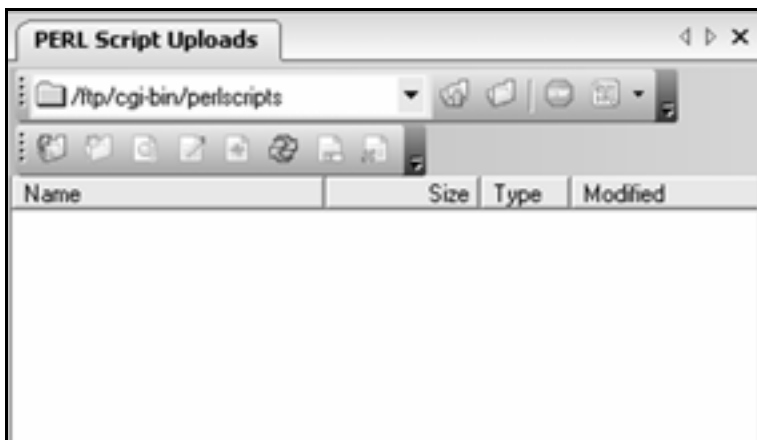


- 4.** In the right-hand **Perl Script Uploads** pane, navigate to the **cgi-bin** directory, then to the **perlscripts** directory in your Web site.

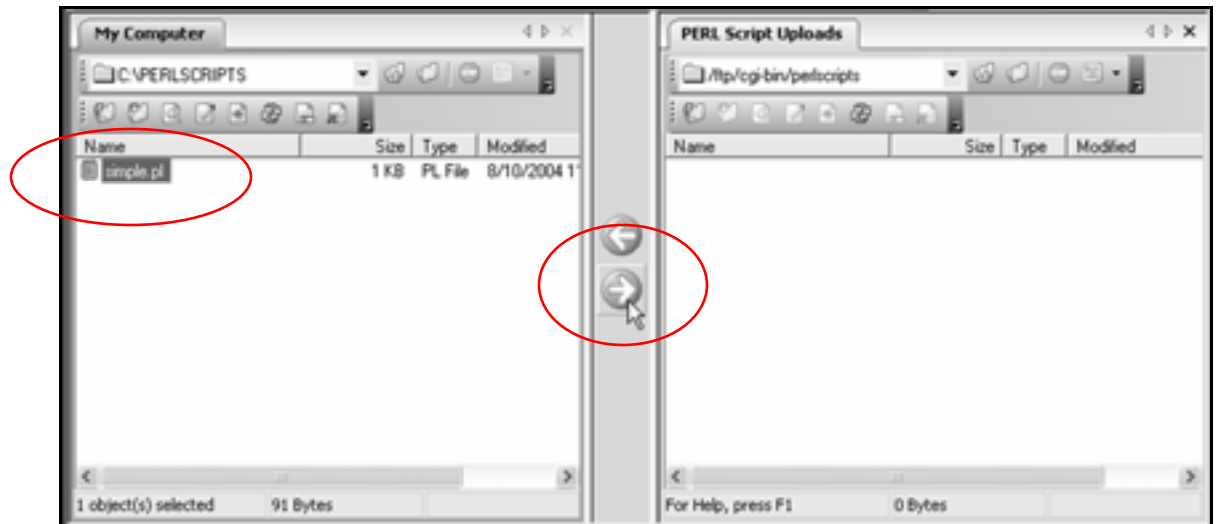


- 5.** Double-click the **perlscripts** directory.

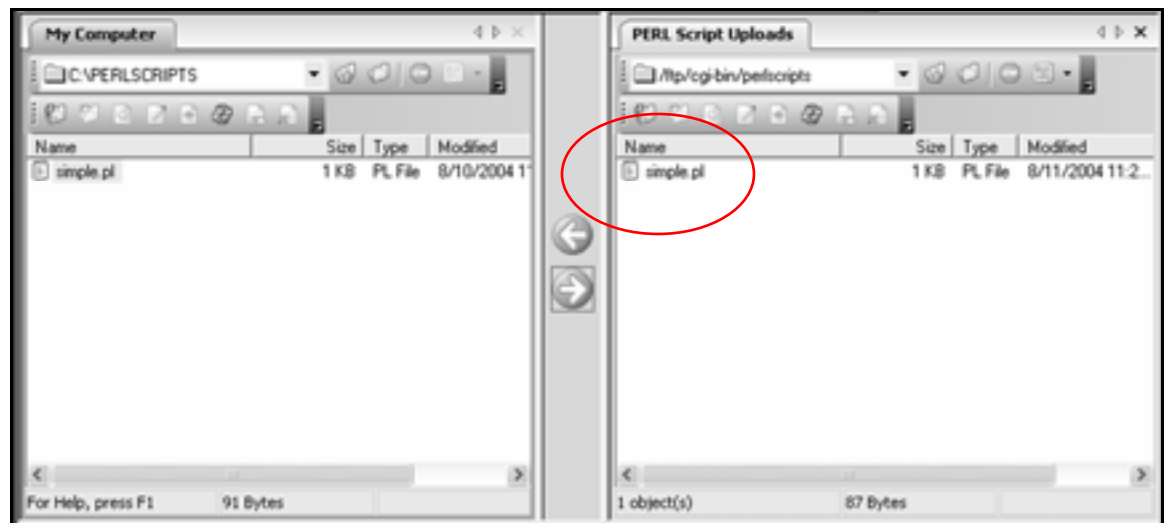
The pane should be blank:



6. Click **simple.pl** in the **My Computer** pane, then click the  button.

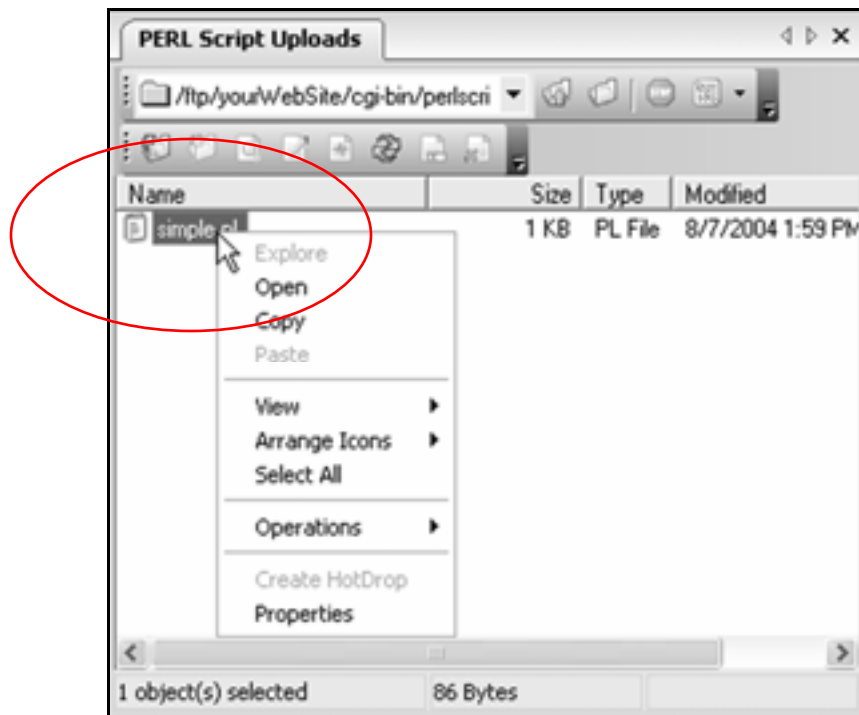


**simple.pl** should now appear in the **Perl Script Uploads** pane:

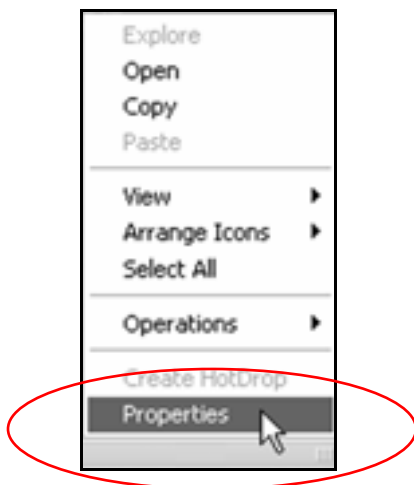


# Set script permissions

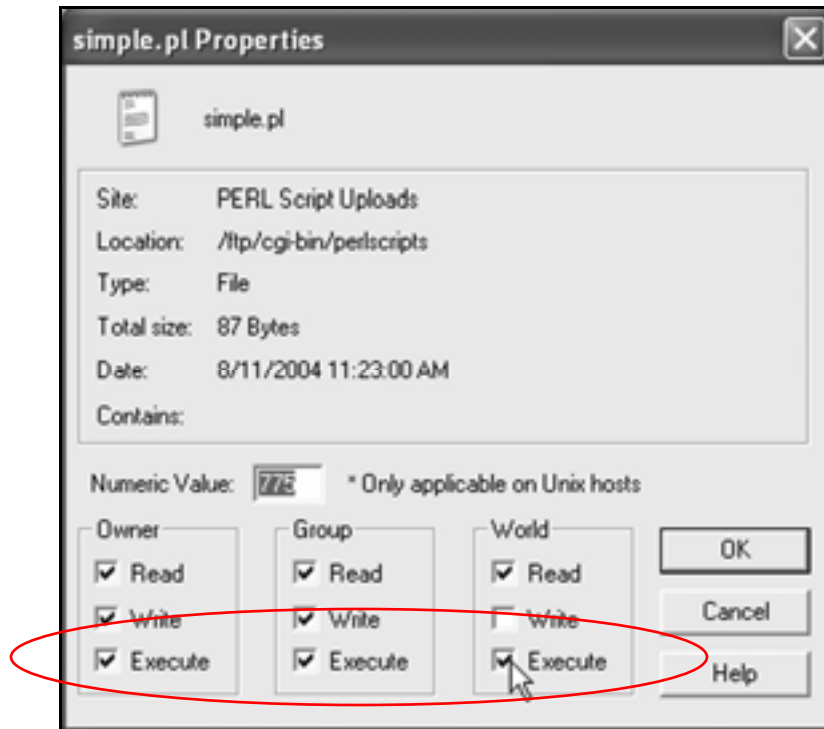
1. In the **Perl Script Uploads** pane, right-click **simple.pl**.



2. When the menu appears, click **Properties**.



- 3.** When the **simple.pl Properties** window appears, click all the **Execute** checkboxes.



- 4.** Click the **OK** button.

# Run a script from a Web page

1. Using Notepad, create a new Web page with this code:

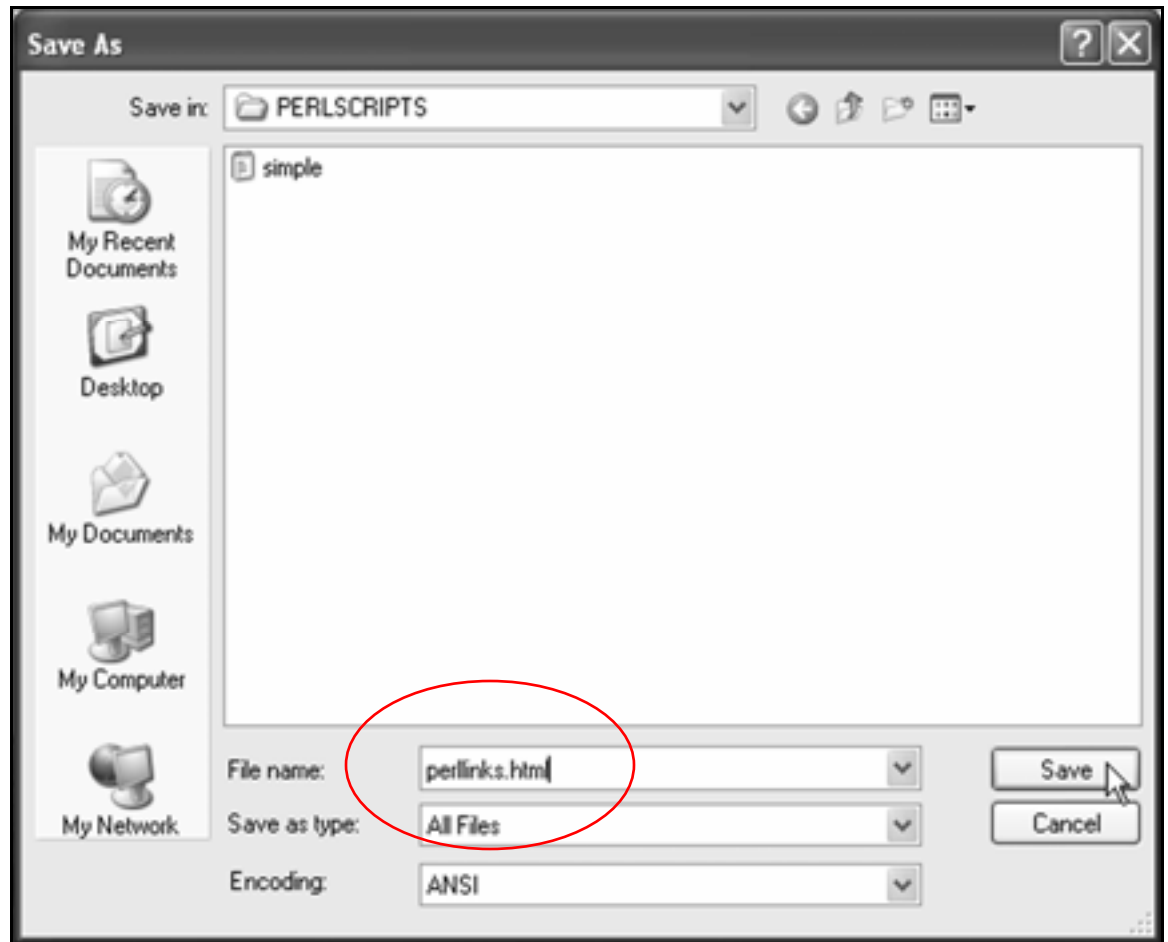
```
<html>
<head>
<title>Run your first PERL script</title>
</head>
<body>
```

Click on `<a href="http://www.yourwebsite.com/cgi-bin/perlscripts/simple.pl">this link</a>` to run your first PERL script.

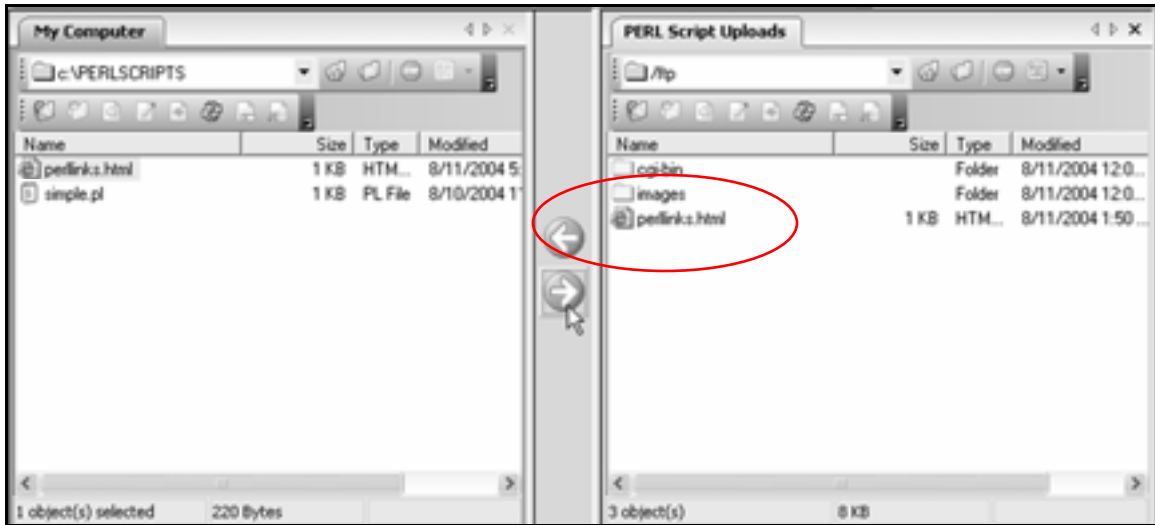
```
</body>
</html>
```



2. Save the Web page as **perllinks.html** in the **PERLSCRIPTS** folder on your computer.



- 3.** In WS\_FTP, upload **perllinks.html** into the home directory of your Web site.



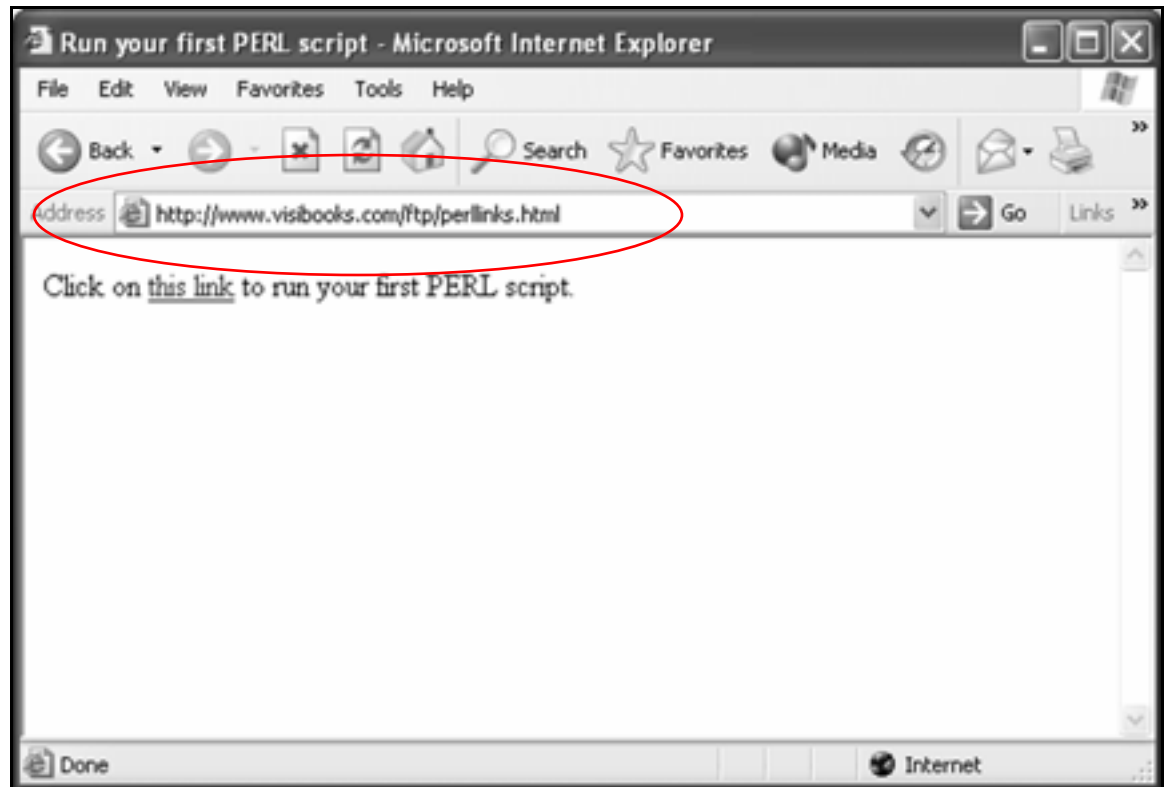
**Tip:** *Don't upload perllinks.html into the cgi-bin/perlscripts directory.*

*Put it in the home directory of your Web site, where the home page—index.html—resides.*

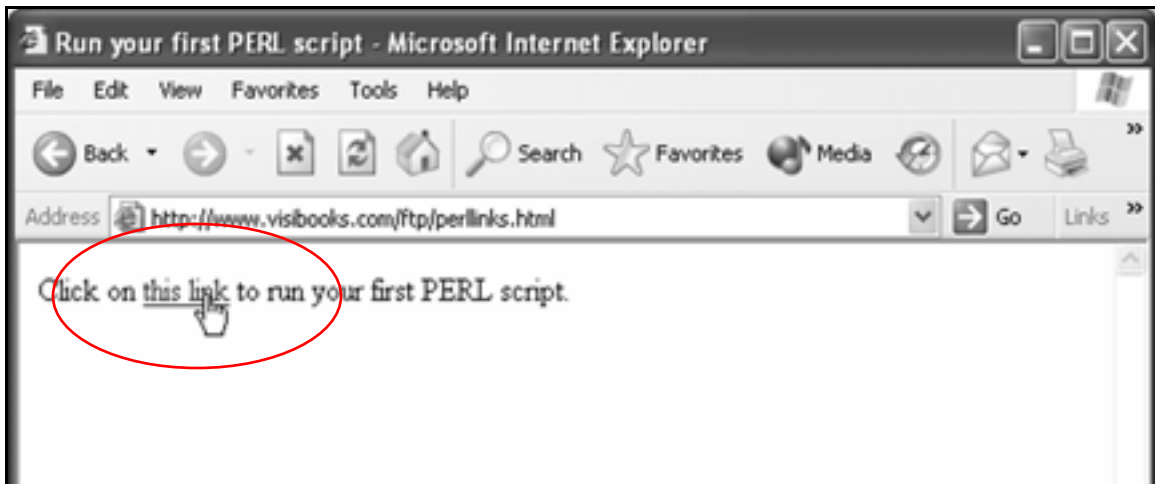


**4.** Open the Web browser and go to:

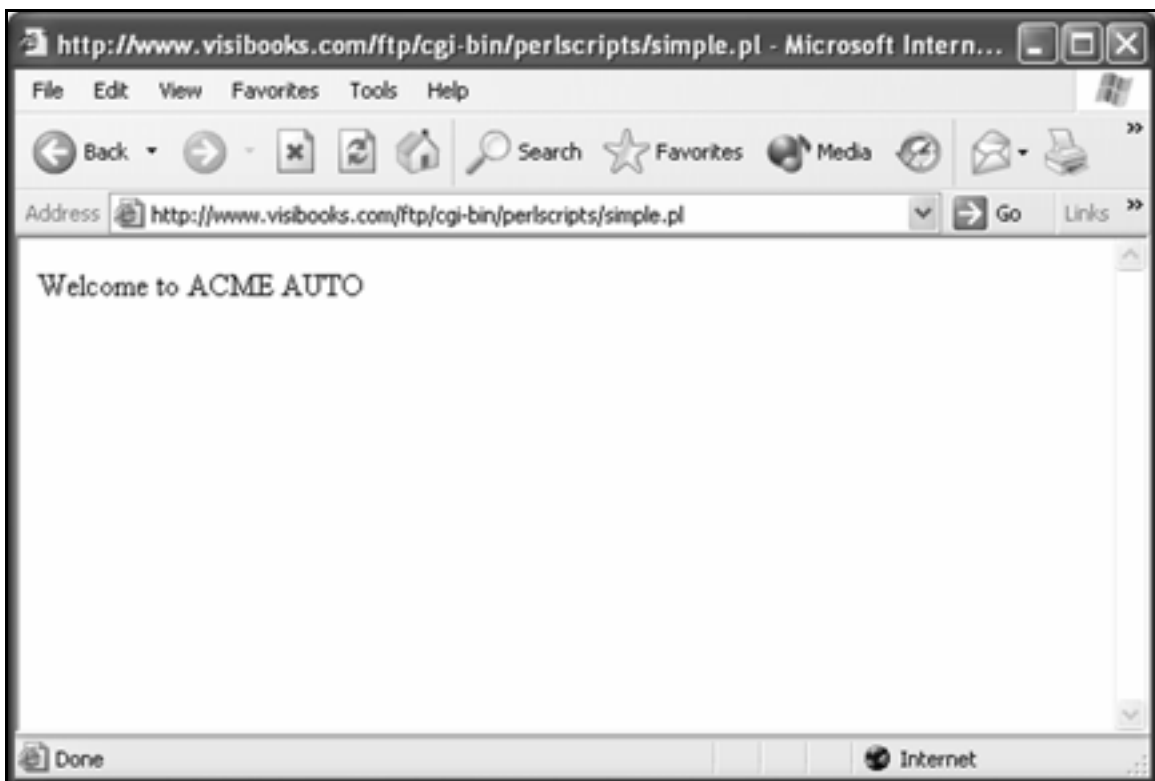
**[www.yourwebsite.com/perllinks.html](http://www.yourwebsite.com/perllinks.html)**



**5.** Click the link.



The output should look like this:



# Insert comments

1. Using Notepad, create a new script with this code:

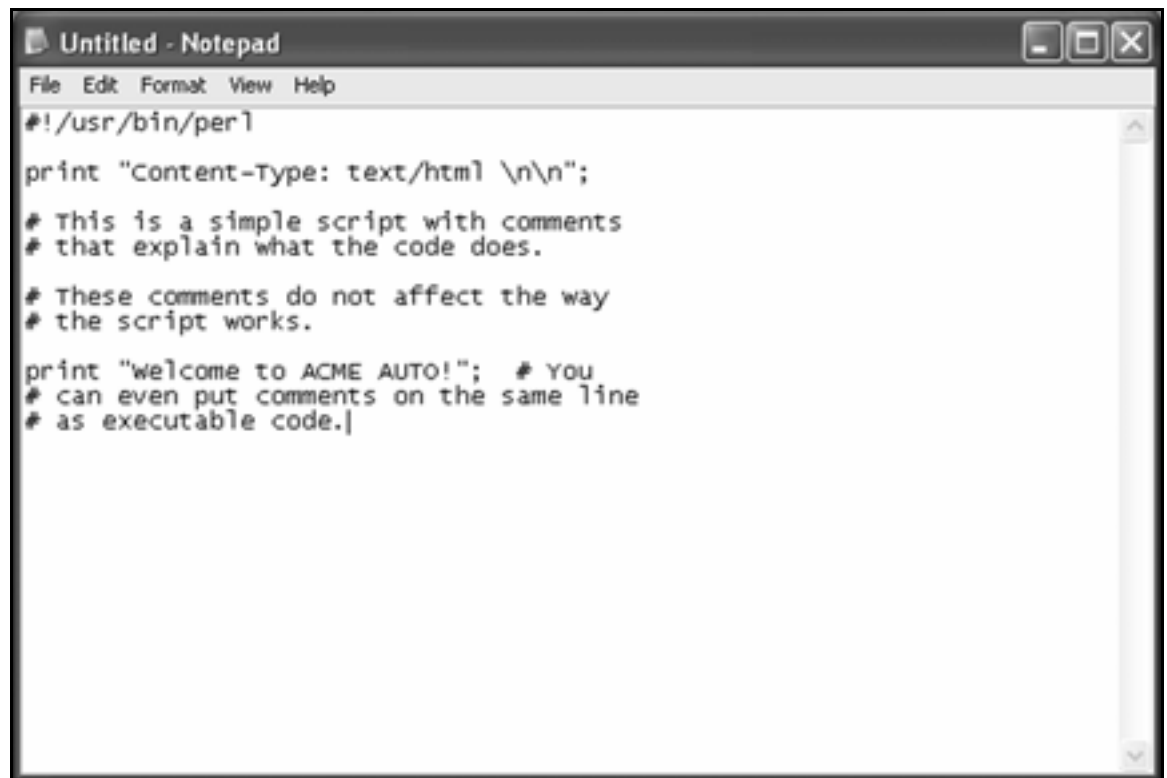
```
#!/usr/bin/perl

print "Content-Type: text/html \n\n";

# This is a simple script with comments
# that explain what the code does.

# These comments do not affect the way
# the script works.

print "Welcome to ACME AUTO!"; # You
# can even put comments on the same line
# as executable code.
```

A screenshot of a Notepad window titled "Untitled - Notepad". The window has a menu bar with "File", "Edit", "Format", "View", and "Help". The text area contains the same Perl script code as shown in the previous block. The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

```
Untitled - Notepad
File Edit Format View Help
#!/usr/bin/perl
print "Content-Type: text/html \n\n";
# This is a simple script with comments
# that explain what the code does.
# These comments do not affect the way
# the script works.
print "welcome to ACME AUTO!"; # You
# can even put comments on the same line
# as executable code.
```

**Tip:** *If you're writing a comment in a script and it wraps to the next line, it needs a new # character in front.*

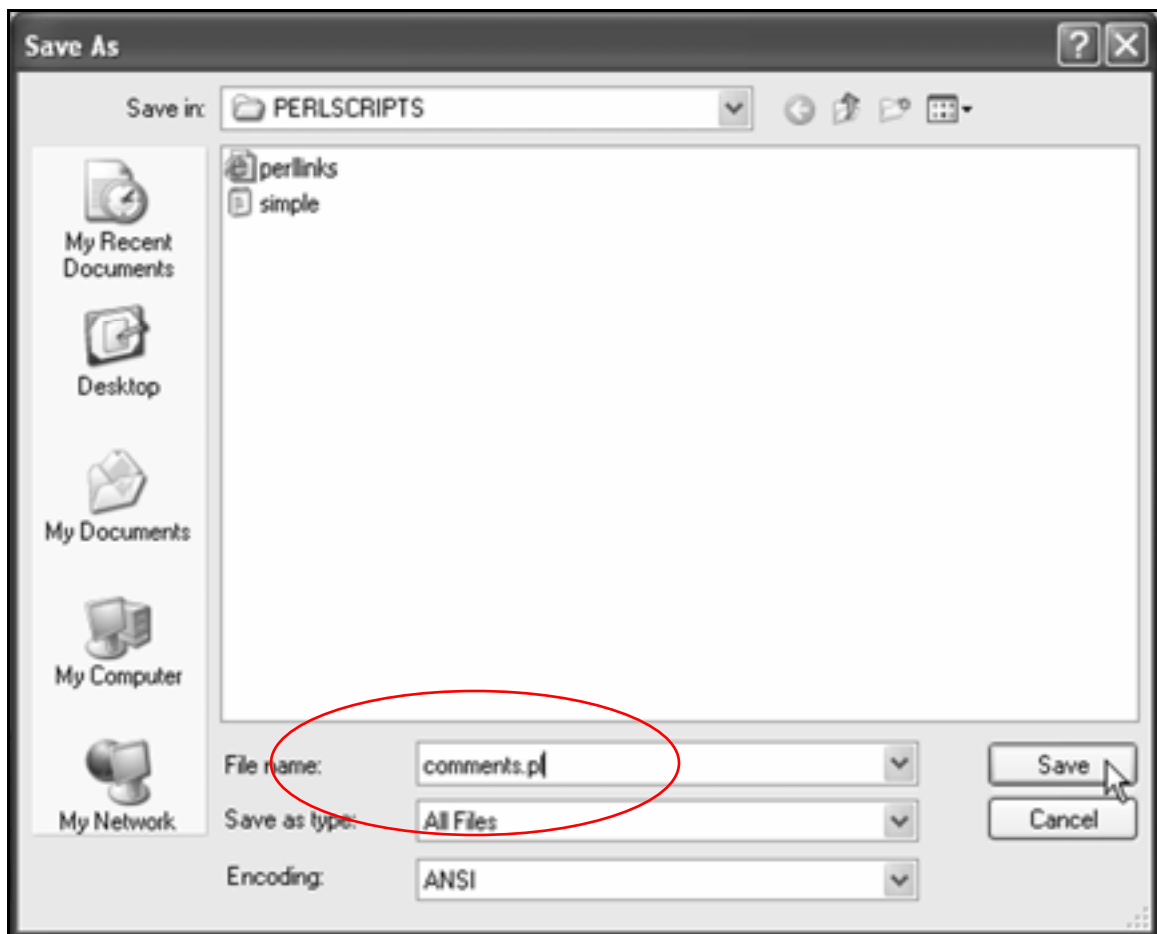
*Incorrect:*

```
# The second line lets a browser  
display the script output.
```

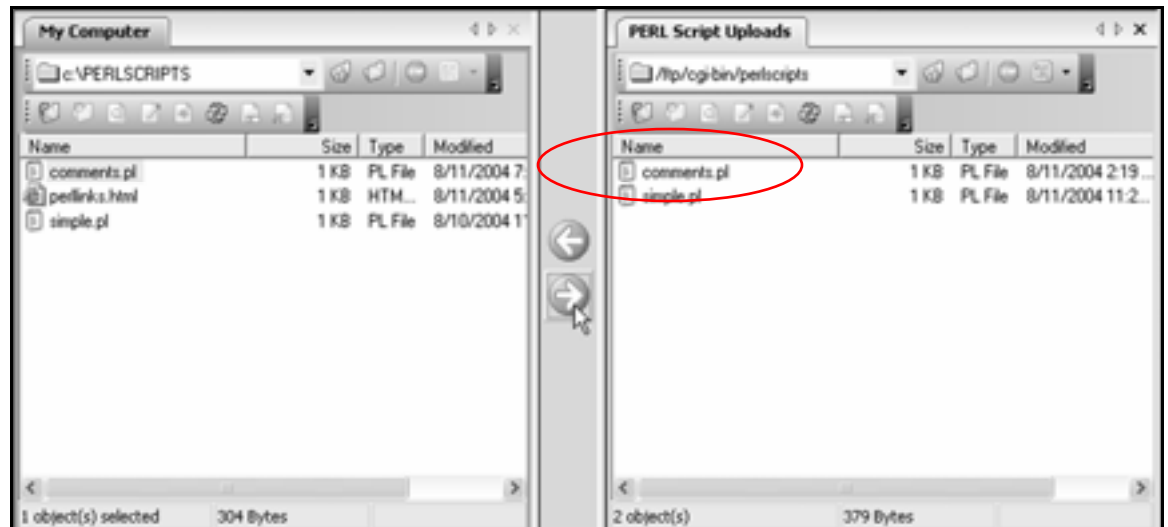
*Correct:*

```
# The second line lets a browser  
# display the script output.
```

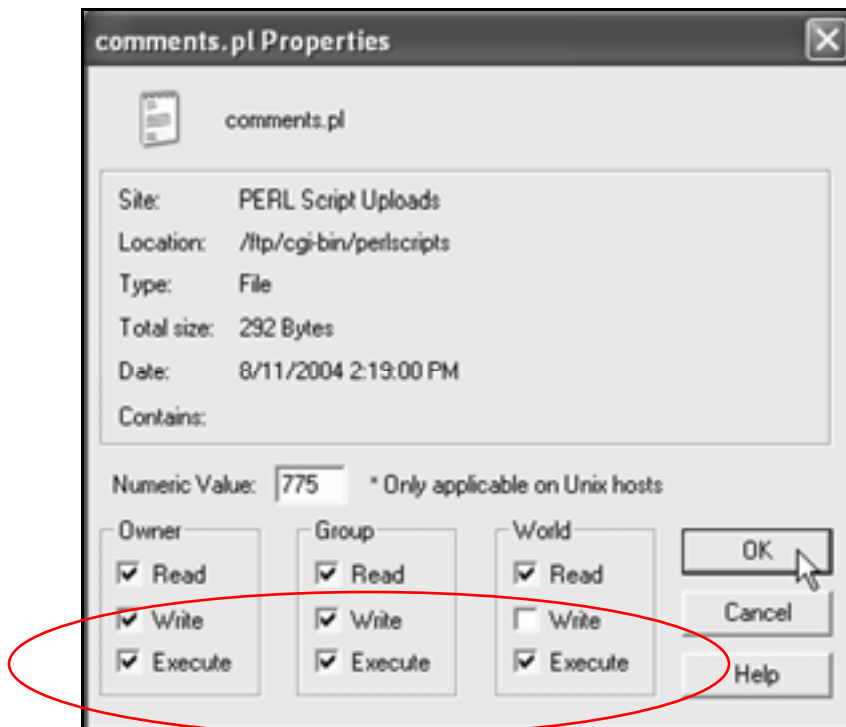
- 2.** Save this script as **comments.pl** in the **PERLSCRIPTS** folder on your computer.



- 3.** Open WS\_FTP and upload the **comments.pl** script to the **perlscripts** directory in your Web site.



- 4.** Set the script's permissions so that Owner, Group, and World can execute it.



# Format text output with HTML tags

1. In Notepad, create a new script with this code:

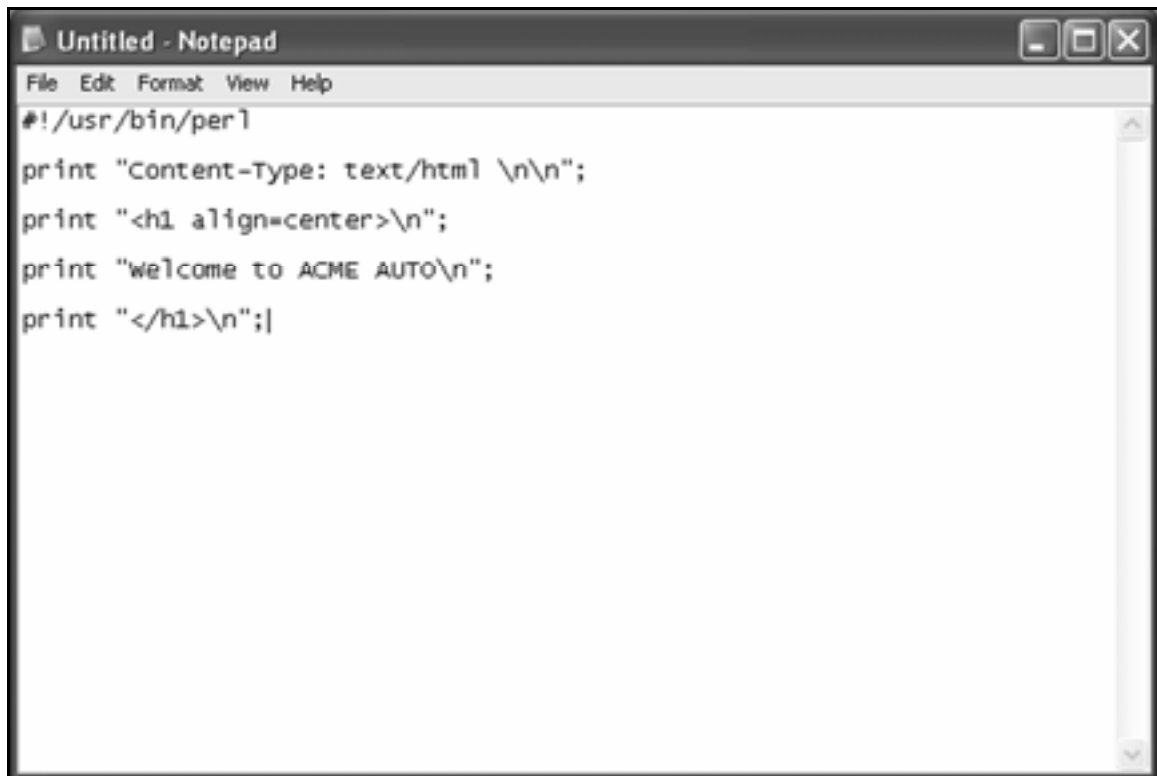
```
#!/usr/bin/perl

print "Content-Type: text/html \n\n";

print "<h1 align=center>\n";

print "Welcome to ACME AUTO\n";

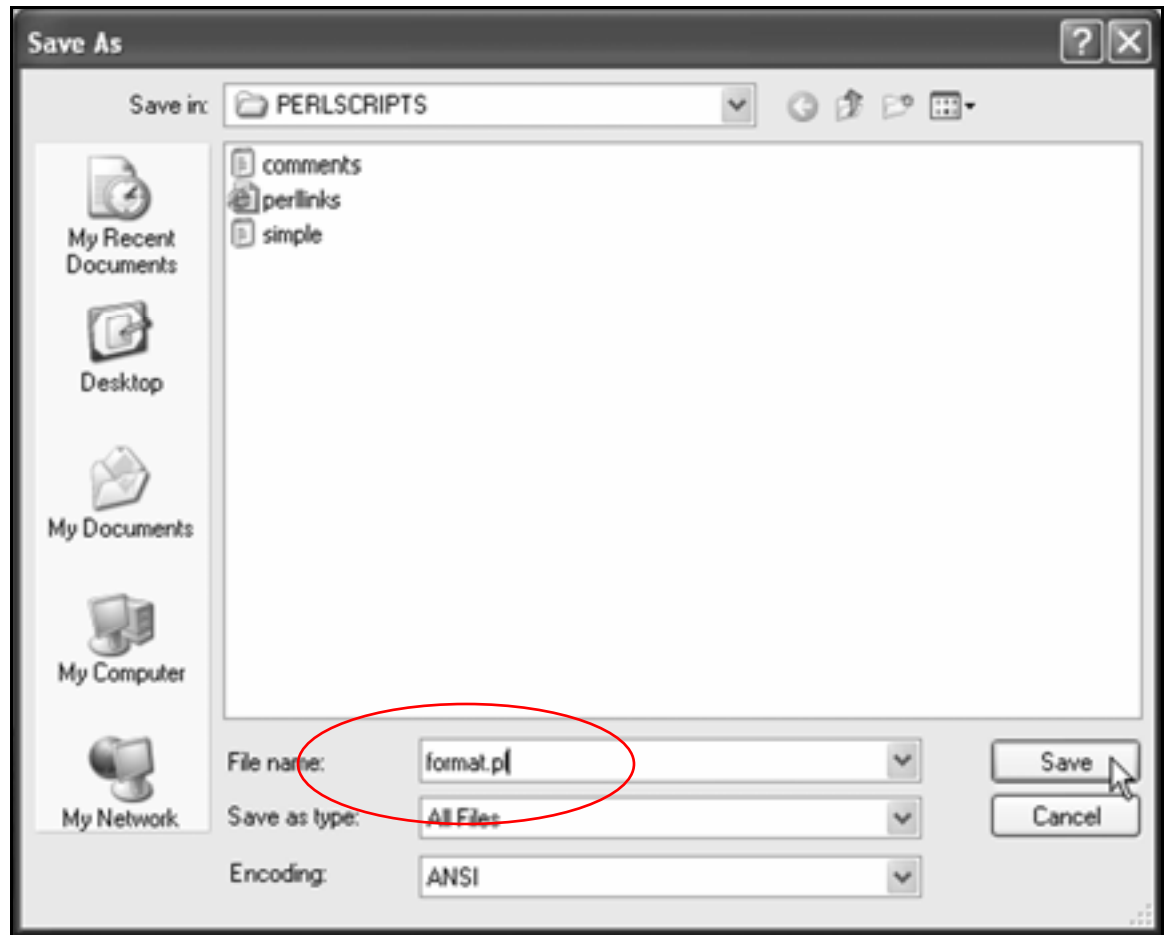
print "</h1>\n";
```

A screenshot of a Notepad window titled "Untitled - Notepad". The window has a menu bar with "File", "Edit", "Format", "View", and "Help". The text area contains the following Perl script code:

```
#!/usr/bin/perl

print "Content-Type: text/html \n\n";
print "<h1 align=center>\n";
print "welcome to ACME AUTO\n";
print "</h1>\n";|
```

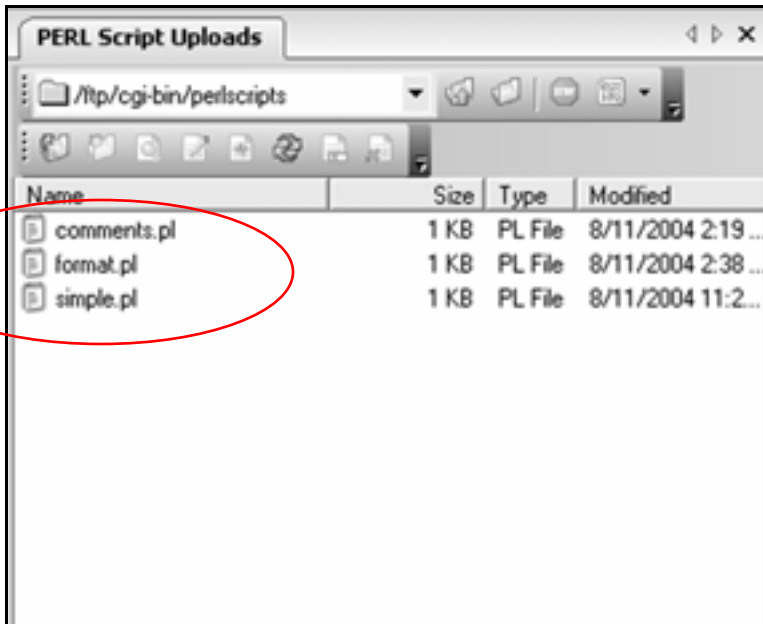
## 2. Save the script as **format.pl** in the **PERLSCRIPTS** folder.



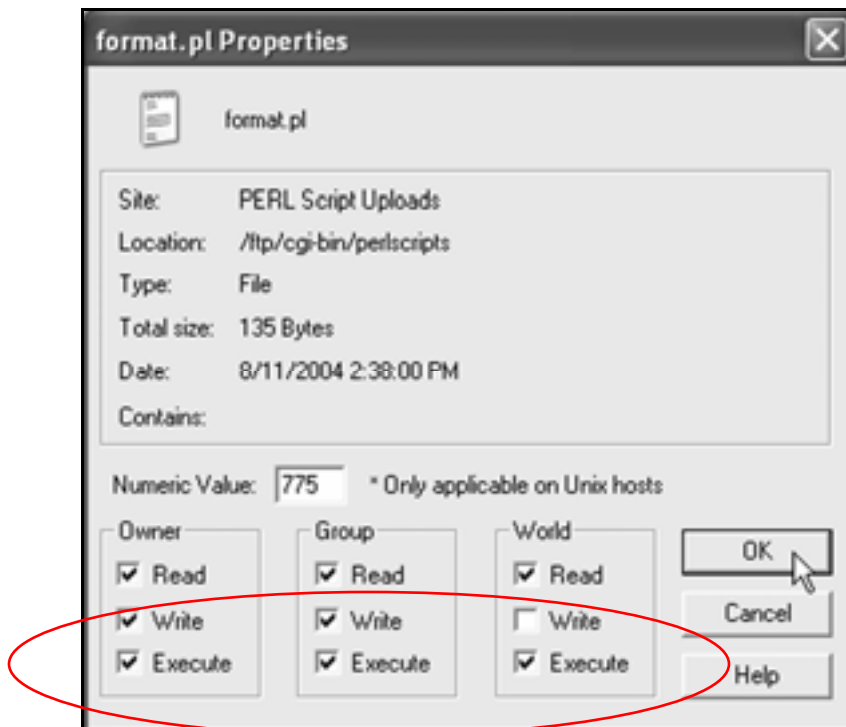
This PERL script also includes HTML tags that format the text it outputs to the browser window:

```
print "<h1 align=center>\n";  
  
print "Welcome to ACME AUTO\n";  
  
print "</h1>\n";
```

- 3.** Upload **format.pl** to the **perlscripts** directory in your Web site.

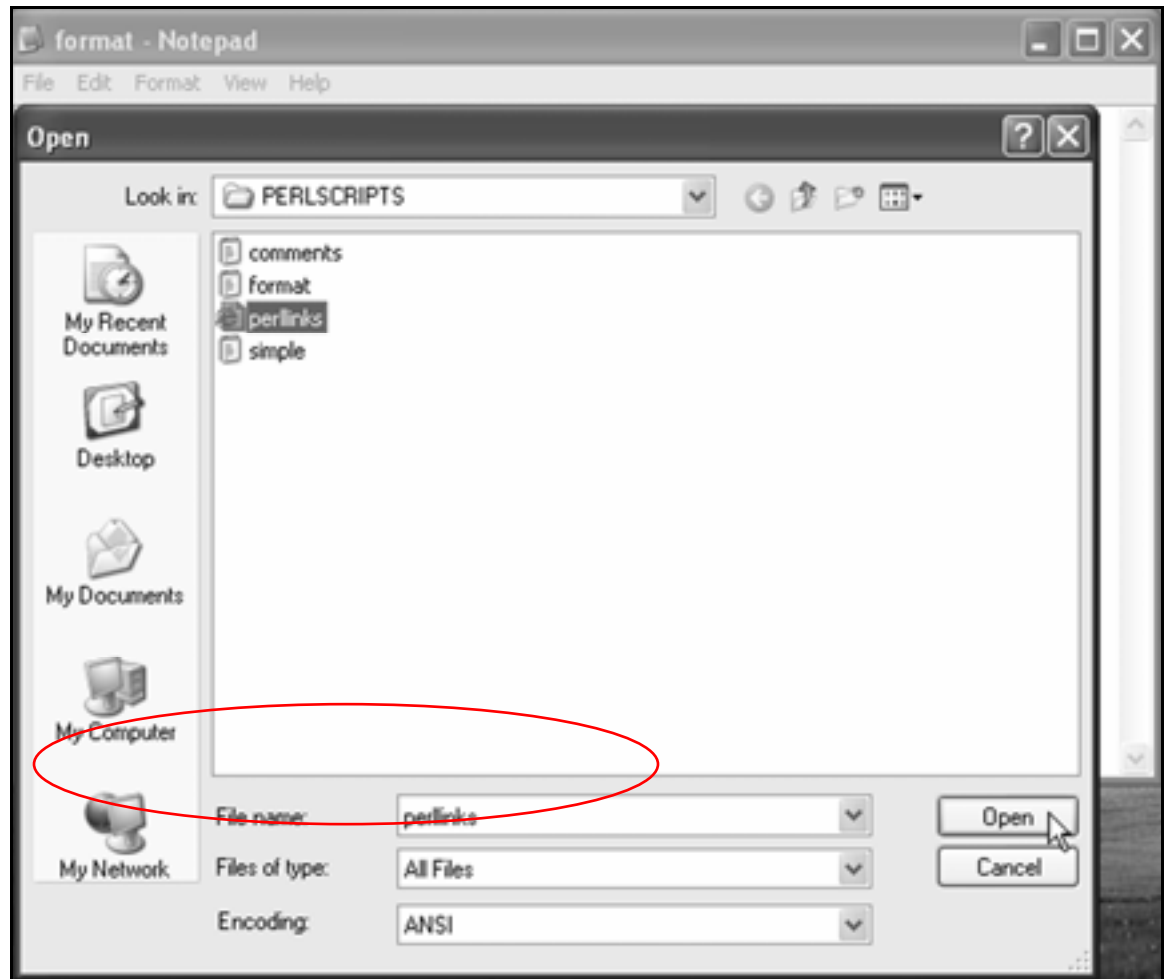


- 4.** Set its permissions so that anyone can execute it.





## 5. Open **perllinks.html** in Notepad.



**Tip:** *It's in the PERLSCRIPTS folder.*

*You may need to select All Files in the Files of type list.*

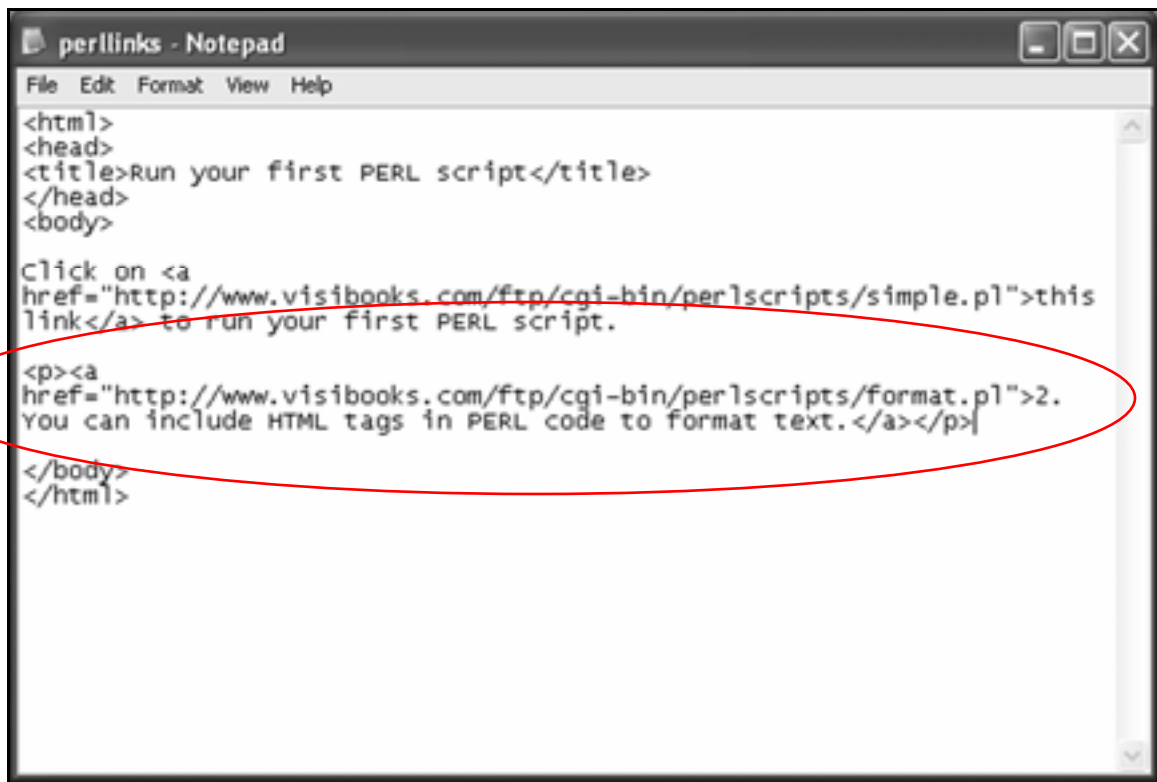
## 6. Add a link to see the output of `format.pl`:

```
<html>
<head>
<title>Run your first PERL script</title>
</head>
<body>
```

Click on `<a href="http://www.yourwebsite.com/cgi-bin/perlscripts/simple.pl">`this link`</a>` to run your first PERL script.

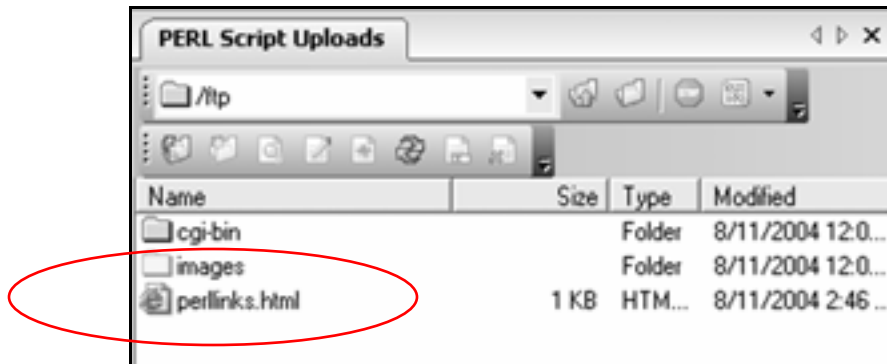
```
<p><a href="http://www.yourwebsite.com/cgi-bin/perlscripts/format.pl">2. You can include HTML tags in PERL code to format text.</a></p>
```

```
</body>
</html>
```



```
perllinks - Notepad
File Edit Format View Help
<html>
<head>
<title>Run your first PERL script</title>
</head>
<body>
Click on <a
href="http://www.visibooks.com/ftp/cgi-bin/perlscripts/simple.pl">this
link</a> to run your first PERL script.
<p><a href="http://www.visibooks.com/ftp/cgi-bin/perlscripts/format.pl">2.
You can include HTML tags in PERL code to format text.</a></p>
</body>
</html>
```

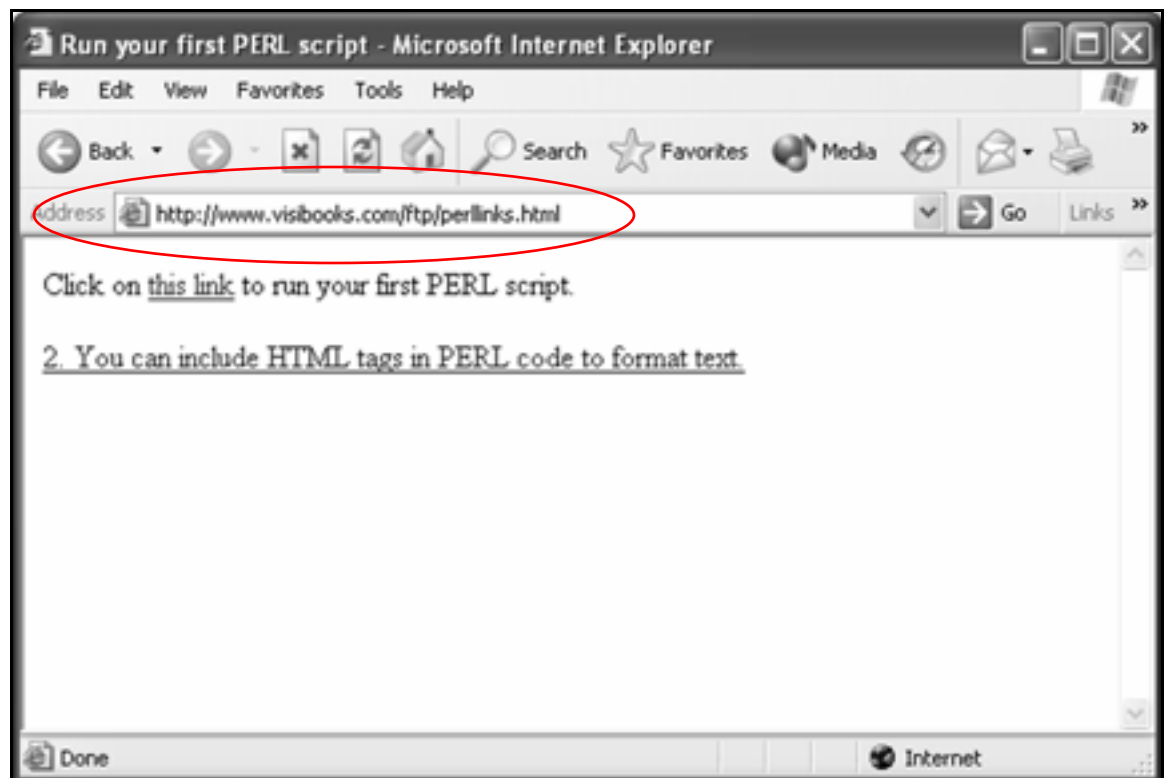
- 7.** Save **perllinks.html**, then use WS\_FTP to upload it to the home directory in your Web site.



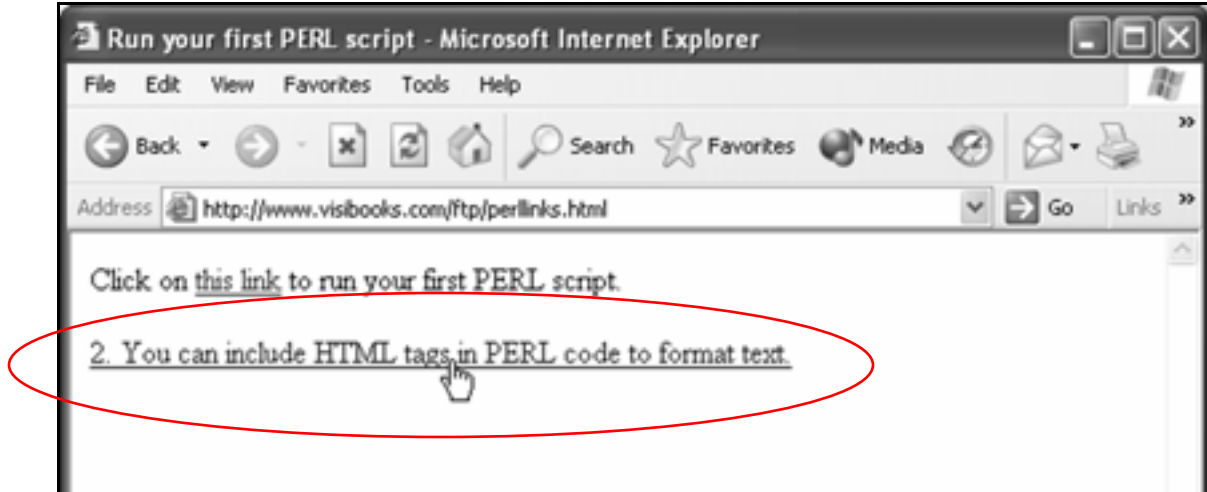
**Tip:** This is the same place **perllinks.html** was before. When WS\_FTP prompts you to replace the existing file, click the  button.

- 8.** Open the browser and go to:

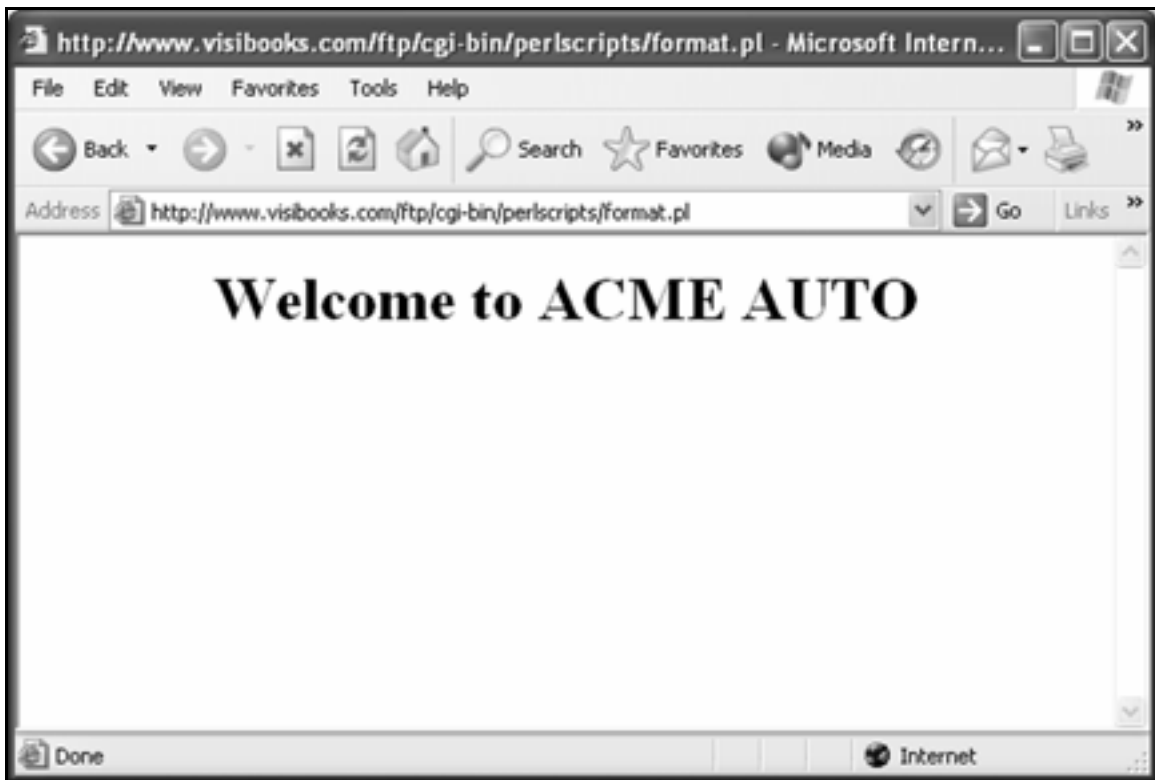
**[www.yourwebsite.com/perllinks.html](http://www.yourwebsite.com/perllinks.html)**



**9.** Click the second link.



The output should look like this:



**10.** Close Notepad and WS\_FTP.

# Working with Variables

In this section, you'll learn how to:

- **Employ single variables**
- **Print quotation marks**
- **Employ lists of variables**

## What's a variable?

A variable is a placeholder for information within a PERL script.

In PERL, a **Scalar variable** is a single piece of information. It always starts with a dollar sign.

*Example:* `$myname`

An **Array variable** is a list of information. It always starts with the “at” sign (@).

*Example:* `@months`

Variables are essential to all programming, and very useful. For example, you can use a Scalar variable to easily change “brown eyes” to “blue eyes” in a PERL script:

```
$eyecolor="brown"
```

As the old song says, “Don’t it make my \$eyecolor eyes blue...”

# Employ single variables

## Assign a number to a single variable

1. Open Notepad, then create a new script with this code:

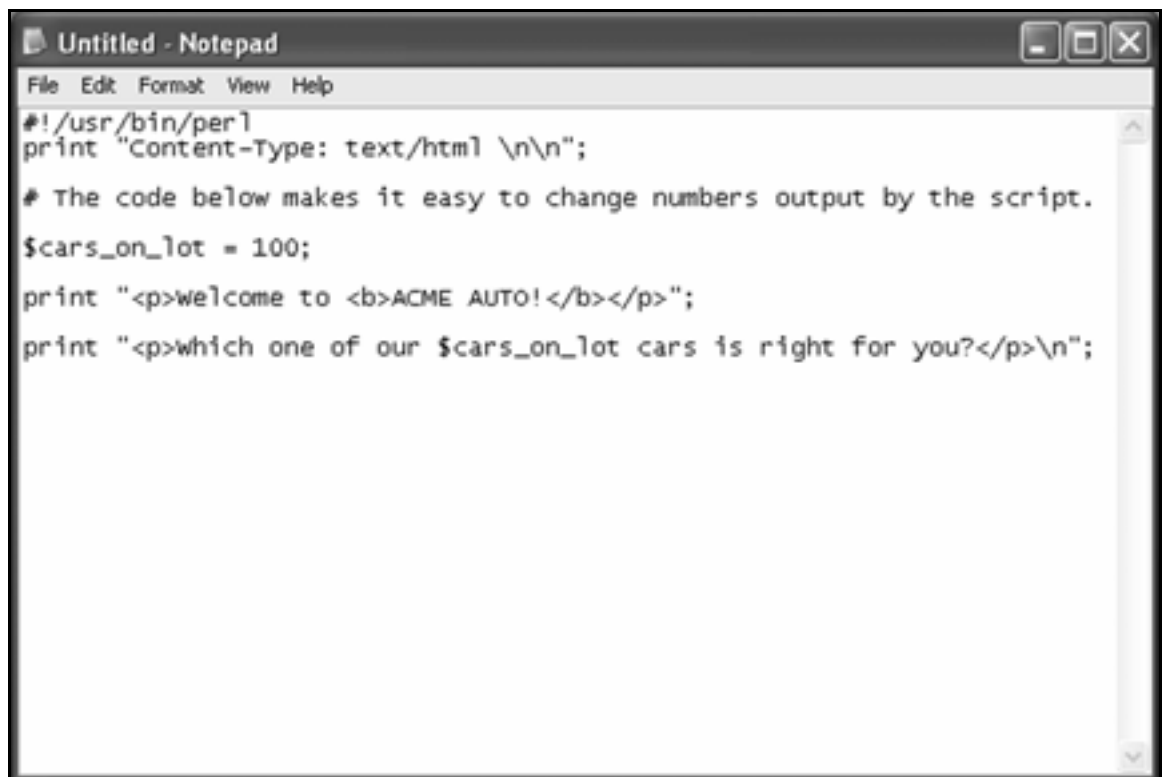
```
#!/usr/bin/perl
print "Content-Type: text/html \n\n";

# The code below makes it easy to change
# numbers output by the script.

$cars_on_lot = 100;

print "<p>Welcome to <b>ACME AUTO!</b></p>";

print "<p>Which one of our $cars_on_lot cars is
right for you?</p>\n";
```

A screenshot of a Notepad window titled "Untitled - Notepad". The window contains the same Perl script code as shown in the previous block. The code is displayed in a monospaced font, and the window includes a menu bar with "File", "Edit", "Format", "View", and "Help" options. The script code is: 

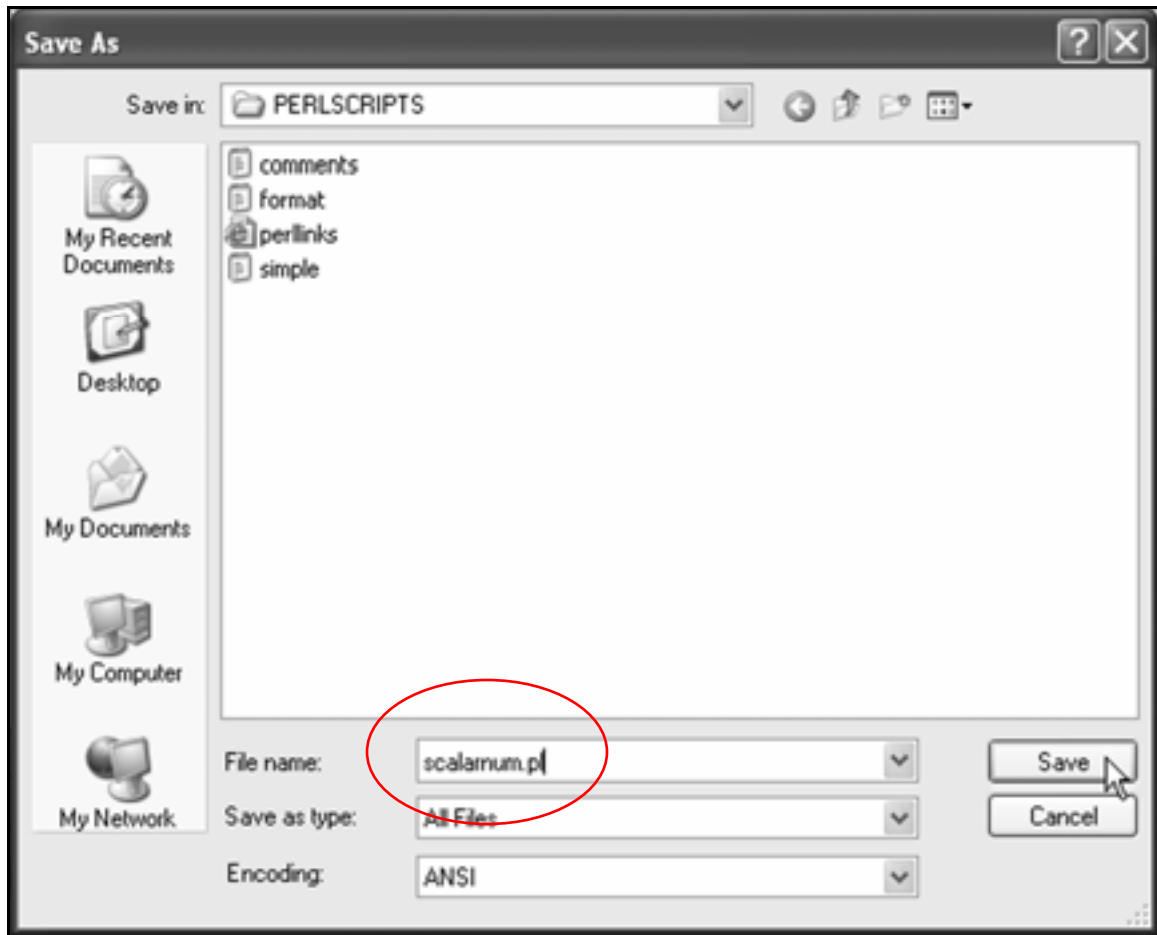
```
#!/usr/bin/perl
print "Content-Type: text/html \n\n";

# The code below makes it easy to change numbers output by the script.
$cars_on_lot = 100;

print "<p>welcome to <b>ACME AUTO!</b></p>";

print "<p>which one of our $cars_on_lot cars is right for you?</p>\n";
```

- 2.** Save the script as **scalarnum.pl** in the **PERLSCRIPTS** folder.



Here's what each line of the script does:

- `#!/usr/bin/perl`  
`print "Content-Type: text/html \n\n";`

These lines should look familiar. The first specifies the path to your Web server's PERL module. The second tells the browser that what comes after this line is HTML.



- `$cars_on_lot = 100;`

`$cars_on_lot` is the single (scalar) variable. Scalar variables start with a `$`.

The number `100` is assigned to the variable. The number is easy to change—that's why it's called a variable.

- `print "<p>Welcome to <b>ACME  
AUTO!</b></p>";`

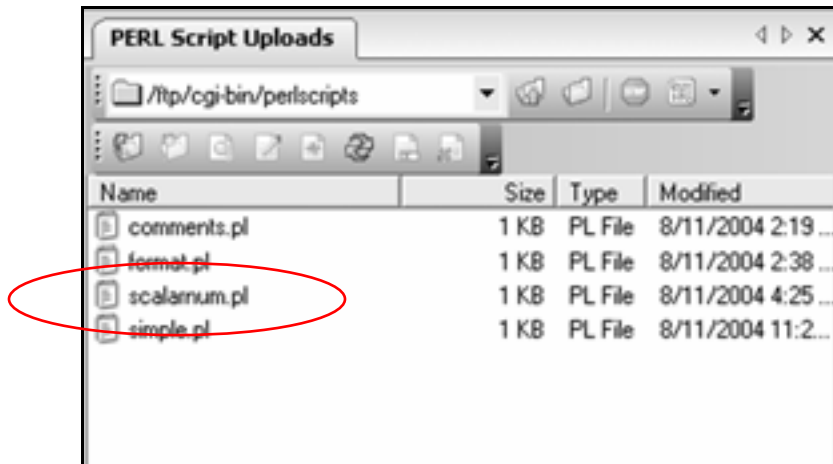
```
print "<p>Which one of our $cars_on_lot  
cars is right for you?</p>\n";
```

These lines should also look familiar. They're HTML code like we've used before, but with a difference: `$cars_on_lot`.

This variable tells the Web browser to get the number specified (`100`) and insert it here.

You'll see how it works in the following steps.

3. Open WS\_FTP, then upload **scalarnum.pl** to the **perlscripts** directory in your Web site.



4. Set its permissions so that anyone can execute it.
5. In Notepad, open **perllinks.html**.

## 6. Insert a new link to **scalarnum.pl**:

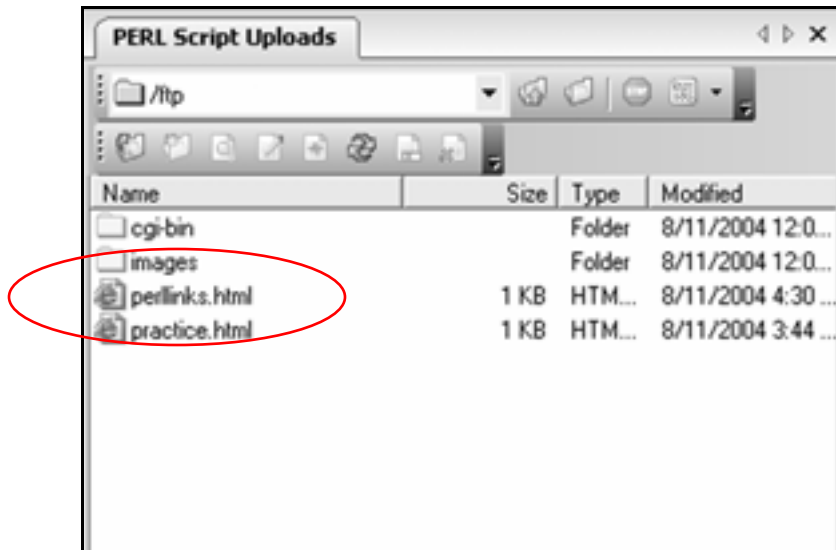
```
<p><a  
href="http://www.yourwebsite.com/cgi-  
bin/perlscripts/format.pl">2. You can include  
HTML tags in PERL code to format text.</a></p>
```

```
<p><a  
href="http://www.yourwebsite.com/cgi-  
bin/perlscripts/scalarnum.pl">3. Assign a  
number to a single variable.</a></p>
```



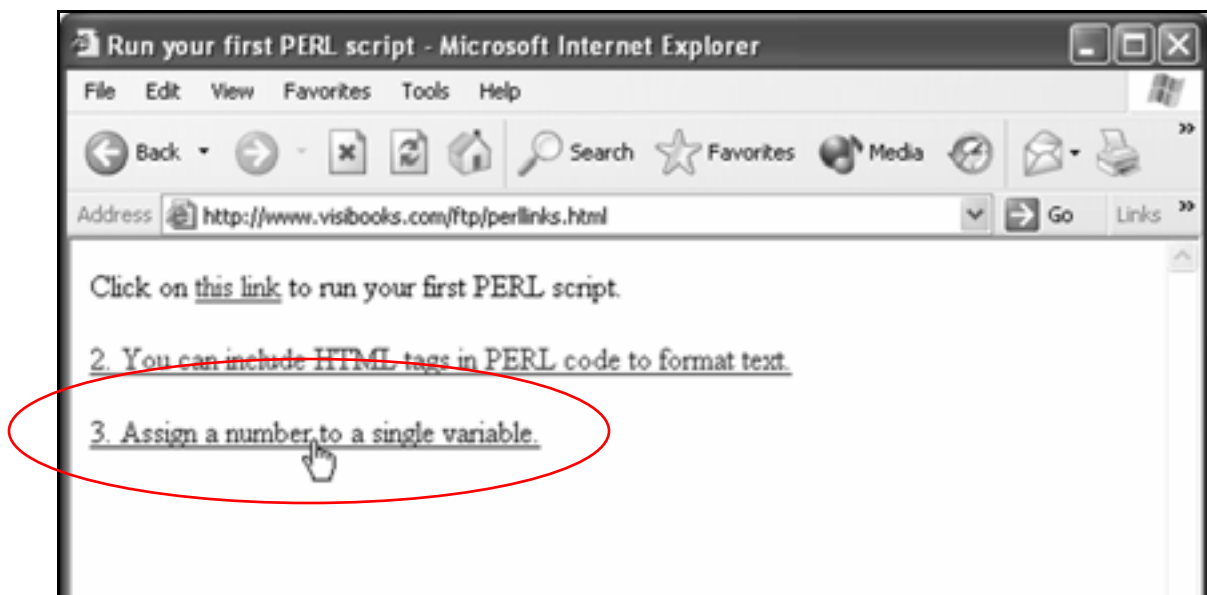
```
perllinks - Notepad  
File Edit Format View Help  
<html>  
<head>  
<title>Run your first PERL script</title>  
</head>  
<body>  
  
Click on <a  
href="http://www.visibooks.com/ftp/cgi-bin/perlscripts/simple.pl">this  
link</a> to run your first PERL script.  
  
<p><a  
href="http://www.visibooks.com/ftp/cgi-bin/perlscripts/format.pl">2.  
You can include HTML tags in PERL code to format text.</a></p>  
  
<p><a  
href="http://www.visibooks.com/ftp/cgi-bin/perlscripts/scalarnum.pl">3.  
Assign a number to a single variable.</a></p>  
</body>  
</html>
```

7. Save **perlinks.html**, then upload it to the home directory in your Web site.

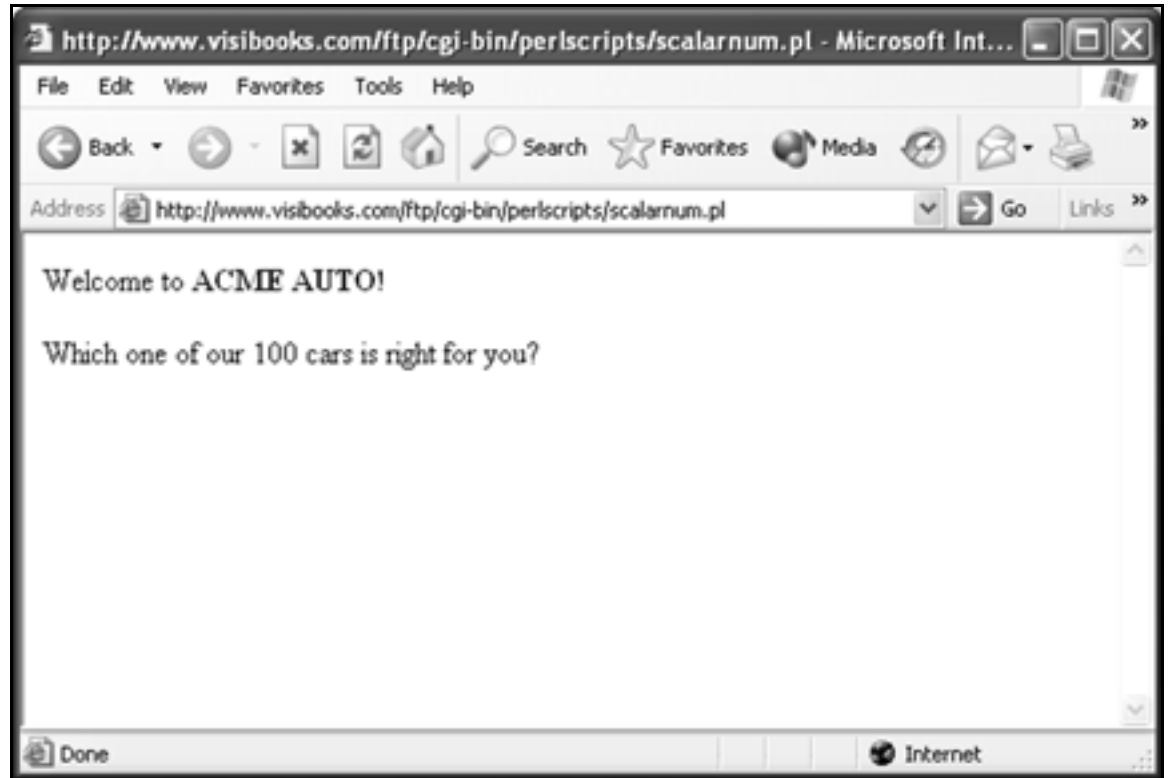


8. Using the browser, go to:  
**www.yourwebsite.com/perlinks.html**

9. Click the **Assign a number to a single variable** link.



The output should look like this:



# Assign text to a single variable

1. Using Notepad, create a new script with this code:

```
#!/usr/bin/perl
print "Content-Type: text/html \n\n";

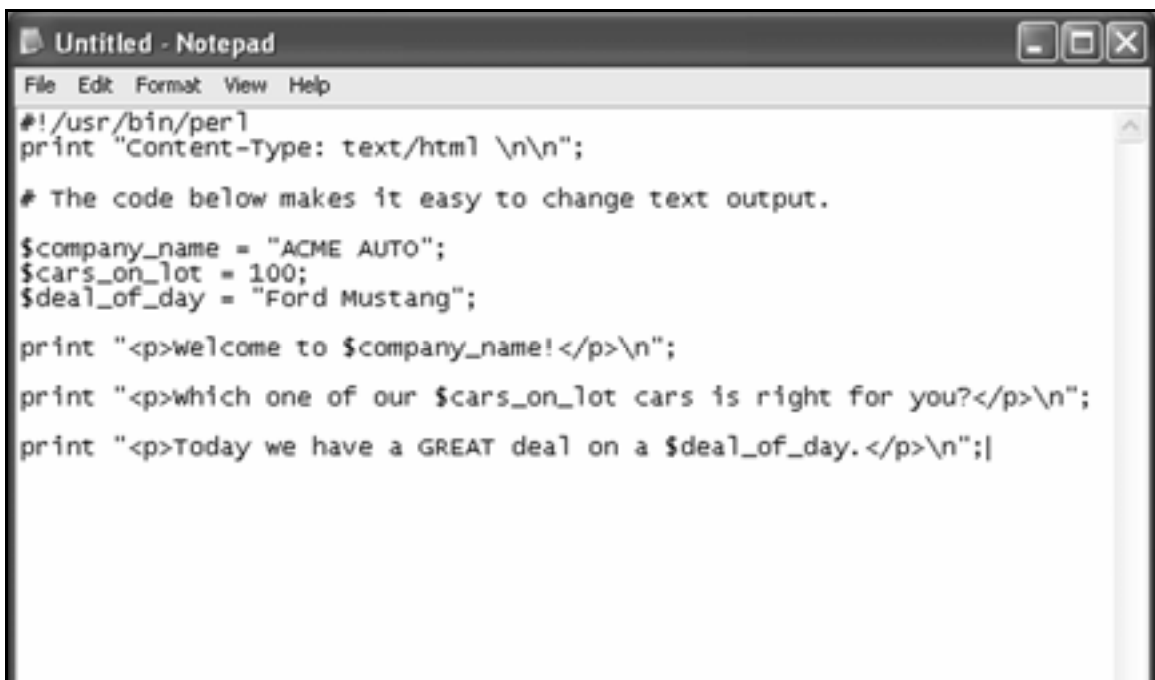
# The code below makes it easy to
# change text output.

$company_name = "ACME AUTO";
$cars_on_lot   = 100;
$deal_of_day   = "Ford Mustang";

print "<p>Welcome to $company_name!</p>\n";

print "<p>Which one of our $cars_on_lot cars is
right for you?</p>\n";

print "<p>Today we have a GREAT deal on a
$deal_of_day.</p>\n";
```

A screenshot of a Notepad window titled "Untitled - Notepad". The window contains the same Perl script code as shown in the previous block. The code is displayed in a monospaced font, and the window includes a menu bar with "File", "Edit", "Format", "View", and "Help" options. The script code is as follows:

```
#!/usr/bin/perl
print "Content-Type: text/html \n\n";

# The code below makes it easy to change text output.

$company_name = "ACME AUTO";
$cars_on_lot   = 100;
$deal_of_day   = "Ford Mustang";

print "<p>welcome to $company_name!</p>\n";
print "<p>which one of our $cars_on_lot cars is right for you?</p>\n";
print "<p>Today we have a GREAT deal on a $deal_of_day.</p>\n";|
```

## 2. Save the script as **scalartext.pl** in the **PERLSCRIPTS** folder.

Here's what the relevant lines in this script do:

- `$company_name = "ACME AUTO";`

Assigns the text `ACME AUTO` to the scalar variable `$company_name`.

- `$deal_of_day = "Ford Mustang";`

Assigns the text `Ford Mustang` to the scalar variable `$deal_of_day`.

- `$cars_on_lot = 100;`

Assigns the number `100` to the scalar variable `$cars_on_lot`.

- `print "<p>Welcome to  
$company_name!</p>\n";`

Prints the words “Welcome to” to the browser window, then inserts the text assigned to the scalar variable `$company_name` (“ACME AUTO”).

- `print "<p>Which one of our $cars_on_lot  
cars is right for you?</p>\n";`

Prints words to the browser window, inserting the number assigned to the scalar variable `$cars_on_lot` (100).

- `print "<p>Today we have a GREAT deal on a $deal_of_day.</p>\n";`

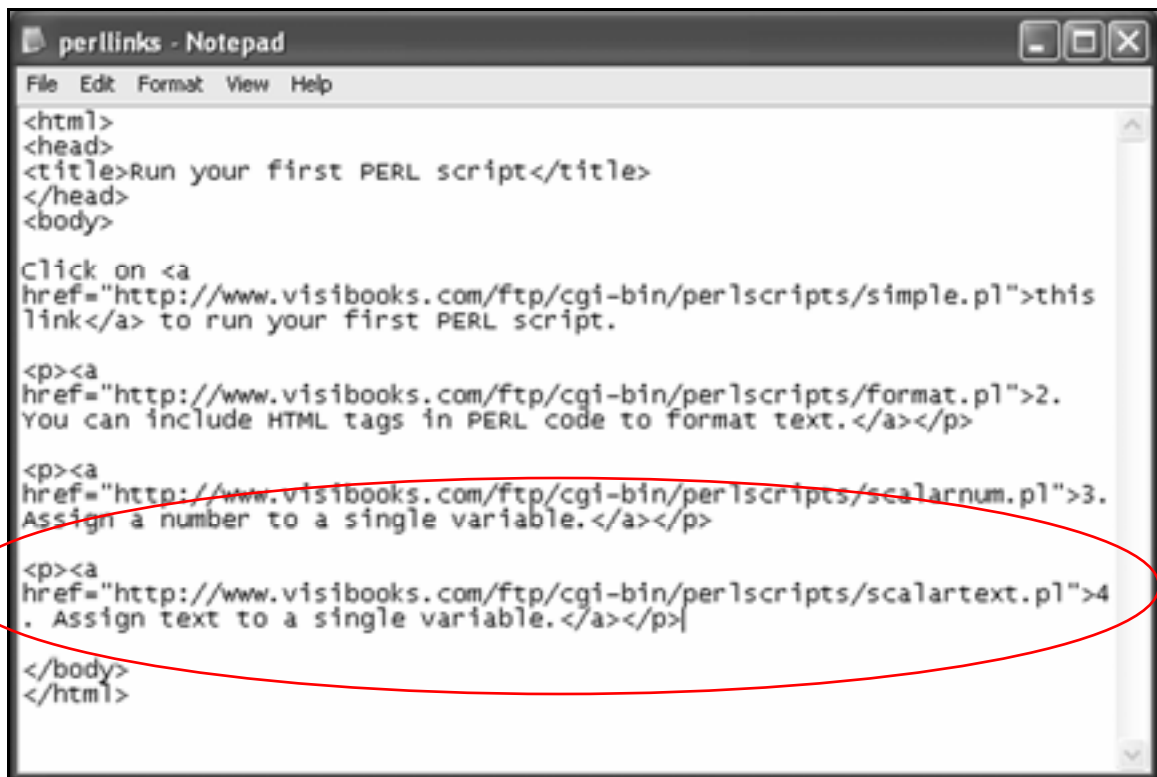
Prints words to the browser window, then inserts the text assigned to the scalar variable `$deal_of_day` ("Ford Mustang").

**3.** Upload **scalartext.pl** to the **perlscripts** directory in your Web site and set its permissions so that anyone can execute it.

**4.** In Notepad, open **perllinks.html**.

**5.** Insert a new link to **scalartext.pl**:

```
<p><a  
href="http://www.yourwebsite.com/cgi-  
bin/perlscripts/scalartext.pl">4. Assign text  
to a single variable.</a></p>
```

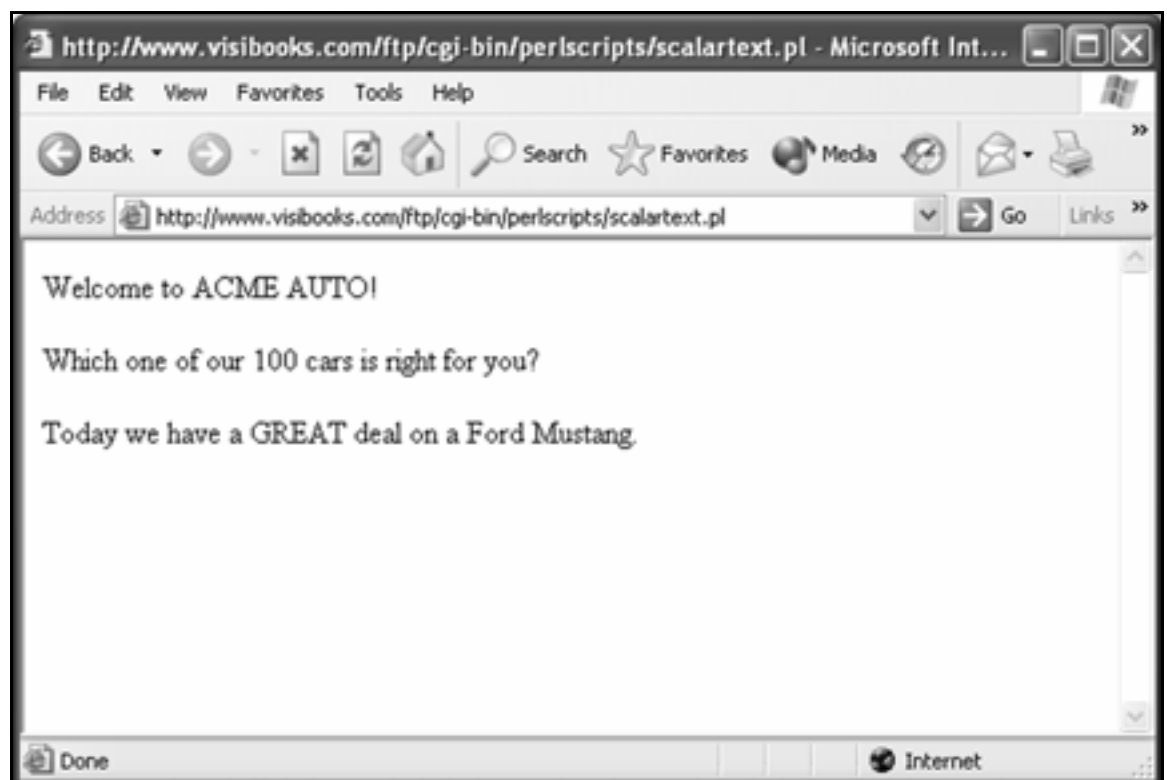


```
perllinks - Notepad  
File Edit Format View Help  
<html>  
<head>  
<title>Run your first PERL script</title>  
</head>  
<body>  
  
Click on <a  
href="http://www.visibooks.com/ftp/cgi-bin/perlscripts/simple.pl">this  
link</a> to run your first PERL script.  
  
<p><a  
href="http://www.visibooks.com/ftp/cgi-bin/perlscripts/format.pl">2.  
You can include HTML tags in PERL code to format text.</a></p>  
  
<p><a  
href="http://www.visibooks.com/ftp/cgi-bin/perlscripts/scalarnum.pl">3.  
Assign a number to a single variable.</a></p>  
  
<p><a  
href="http://www.visibooks.com/ftp/cgi-bin/perlscripts/scalartext.pl">4  
. Assign text to a single variable.</a></p>  
  
</body>  
</html>
```



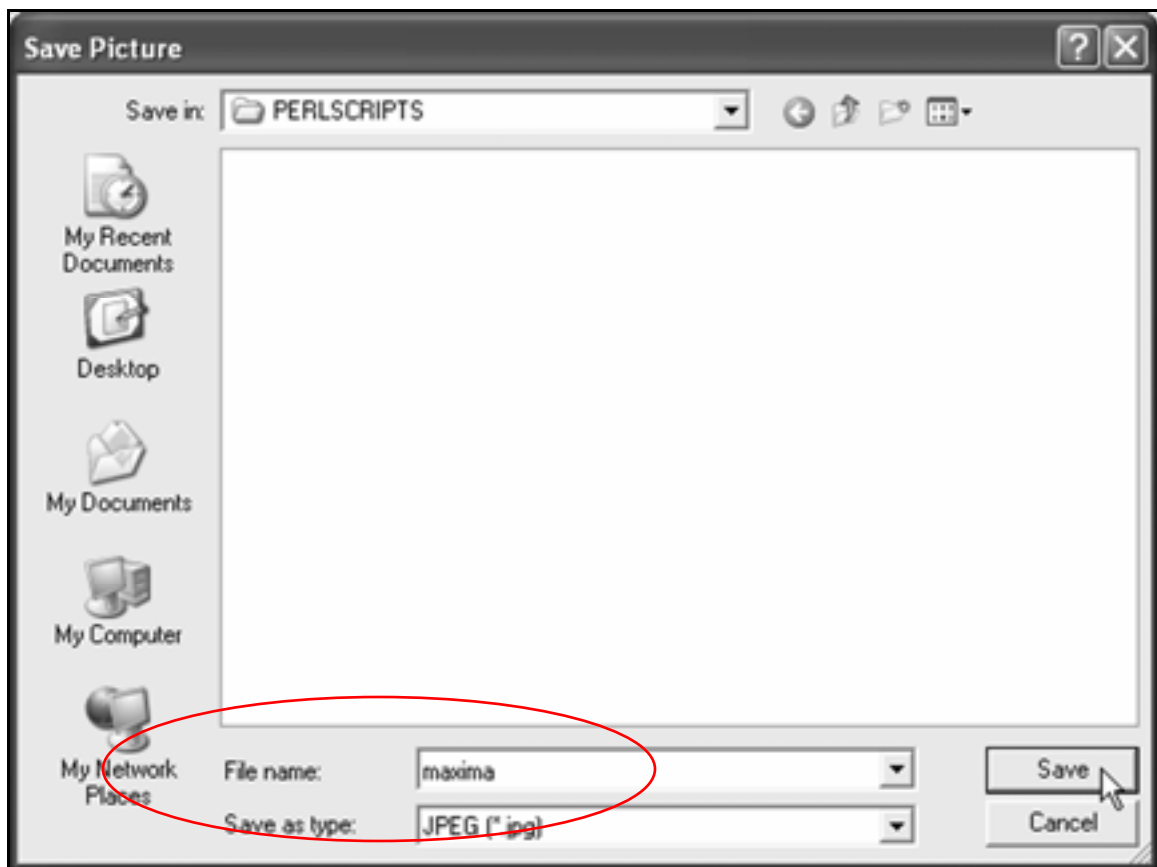
6. Save **perllinks.html**, then upload it to the home directory in your Web site.
7. Using the browser, go to:  
**www.yourwebsite.com/perllinks.html**
8. Click the **Assign text to a single variable** link.

The output should look like this:

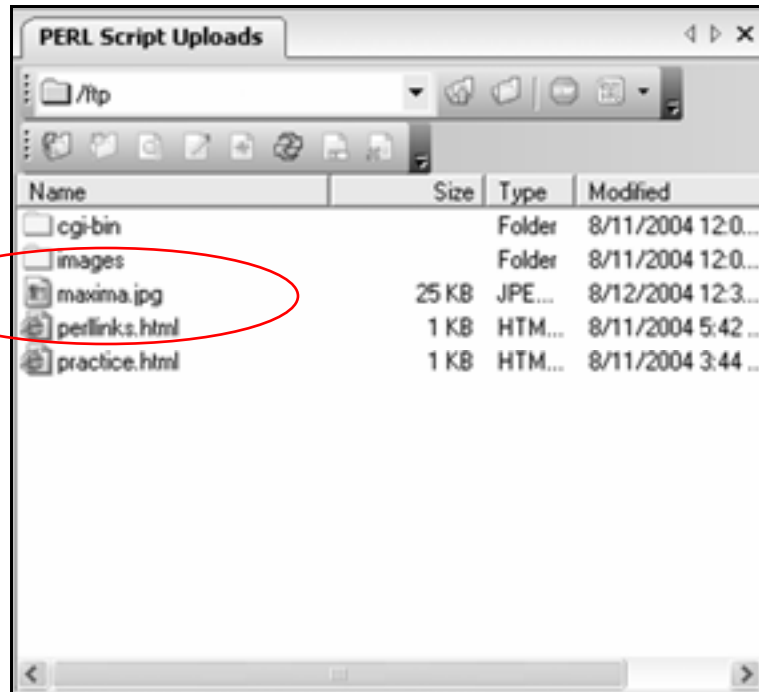


# Print quotation marks

1. In the browser, go to:  
[www.inpics.net/books/perl](http://www.inpics.net/books/perl)
2. Right-click **maxima.jpg**, then save it in the **PERLSCRIPTS** folder on your computer.



- 3.** Upload **maxima.jpg** to the home directory in your Web site.



- 4.** In Notepad, create a new script with this code:

```
#!/usr/bin/perl
print "Content-Type: text/html \n\n";

# The code below uses a variable to
# display a photo.

$cars_on_lot = 100;

$deal_of_day = "Nissan Maxima";

$pic_of_day =
"http://www.yourwebsite.com/maxima.jpg";

print "<p>Welcome to <b>ACME AUTO!</b></p>\n";

print "<p>Which one of our $cars_on_lot cars is
right for you?</p>\n";
```

```
print "<p>Today we have a <b>GREAT</b> deal on  
a $deal_of_day car:</p>\n";  
  
print "<img src=\"$pic_of_day\">\n";
```

**Tip:** Remember to change the `www.yourwebsite.com` address in `$pic_of_day = "http://www.yourwebsite.com/maxima.jpg"` to your actual Web site address.

- 5.** Save the script as **qmarks.pl** in the **PERLSCRIPTS** folder.
- 6.** Upload **qmarks.pl** to the **perlscripts** directory in your Web site and set its permissions so that anyone can execute it.
- 7.** In Notepad, open **perllinks.html**.
- 8.** Insert a new link to **qmarks.pl**:

```
<p><a  
href="http://www.yourwebsite.com/cgi-  
bin/perlscripts/qmarks.pl">5. Print quotation  
marks.</a></p>
```

- 9.** Save **perllinks.html**, then upload it to the home directory in your Web site.
- 10.** Using the browser, go to:  
  
**www.yourwebsite.com/perllinks.html**

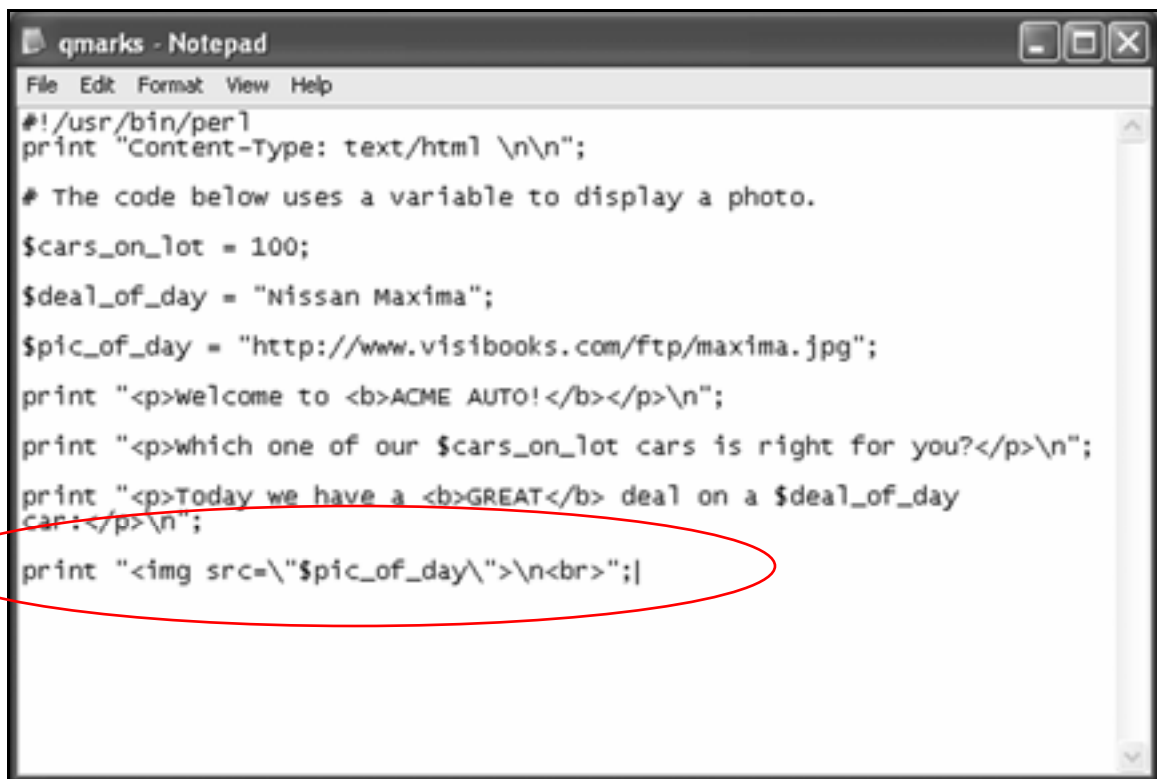
**11.** Click the [Print quotation marks](#) link.

The output should look something like this:



- 12.** In Notepad, edit **qmarks.pl** to enclose the **\$pic\_of\_day** variable in **\** characters:

```
print "<img src=\"\$pic_of_day\">\n<br>";
```



```
qmarks - Notepad
File Edit Format View Help
#!/usr/bin/perl
print "Content-Type: text/html \n\n";
# The code below uses a variable to display a photo.
$cars_on_lot = 100;
$deal_of_day = "Nissan Maxima";
$pic_of_day = "http://www.visibooks.com/ftp/maxima.jpg";
print "<p>welcome to <b>ACME AUTO!</b></p>\n";
print "<p>which one of our $cars_on_lot cars is right for you?</p>\n";
print "<p>Today we have a <b>GREAT</b> deal on a $deal_of_day
car:</p>\n";
print "<img src=\"\$pic_of_day\">\n<br>";
```

- 13.** Save **qmarks.pl** and upload it to the **perlscripts** directory again.
- 14.** Reload **perllinks.html** in your Web browser.

**15.** Click the [Print quotation marks](#) link again.

Its output should look like this:



**Tip:** *Since the HTML `<img>` tag requires the use of two double-quotation marks*

```

```

*enclose them in `\` characters to let the Web server know that you want to print a double-quote to the screen. Otherwise, the Web server will think you want the double-quotes to start and end a text string in a PERL command.*

*`\` is called an “escape character.” Escape characters are used to print characters, such as double-quotes, that the Web server might otherwise think were part of a PERL command or text string.*

# Print with double vs. single quotes

1. Create a new script with this code:

```
#!/usr/bin/perl
print "Content-Type: text/html \n\n";

# Printing with Double Quotes (")
# vs. Single Quotes (')

$cars_on_lot = 100;

print "<p>Welcome to <b>ACME AUTO!</b></p>\n";

# double quotes

print "<p>Which one of our $cars_on_lot cars is
right for you?</p>\n";

# single quotes

print '<p>Which one of $cars_on_lot is right
for you?</p>\n';
```

2. Save the script as **quotes.pl** in the **PERLSCRIPTS** folder.

Here's what the relevant lines in this script do:

- `print "<p>Which one of our $cars_on_lot cars is right for you?</p>\n";`

By using “double quotes” in the above print statement, the number assigned to the scalar variable `$cars_on_lot` (100) is printed to the browser window.



- `print '<p>Which one of $cars_on_lot is right for you?</p>\n';`

By using 'single quotes' in the above print statement, the text `$cars_on_lot` is printed to the browser window along with the words that surround it.

**3.** Upload `quotes.pl` to the `perlscripts` directory in your Web site, then set its permissions so that anyone can execute it.

**4.** Open `perllinks.html` and insert a new link to `quotes.pl`:

```
<p><a  
href="http://www.yourwebsite.com/cgi-  
bin/perlscripts/quotes.pl">6. Double vs. single  
quotes.</a></p>
```

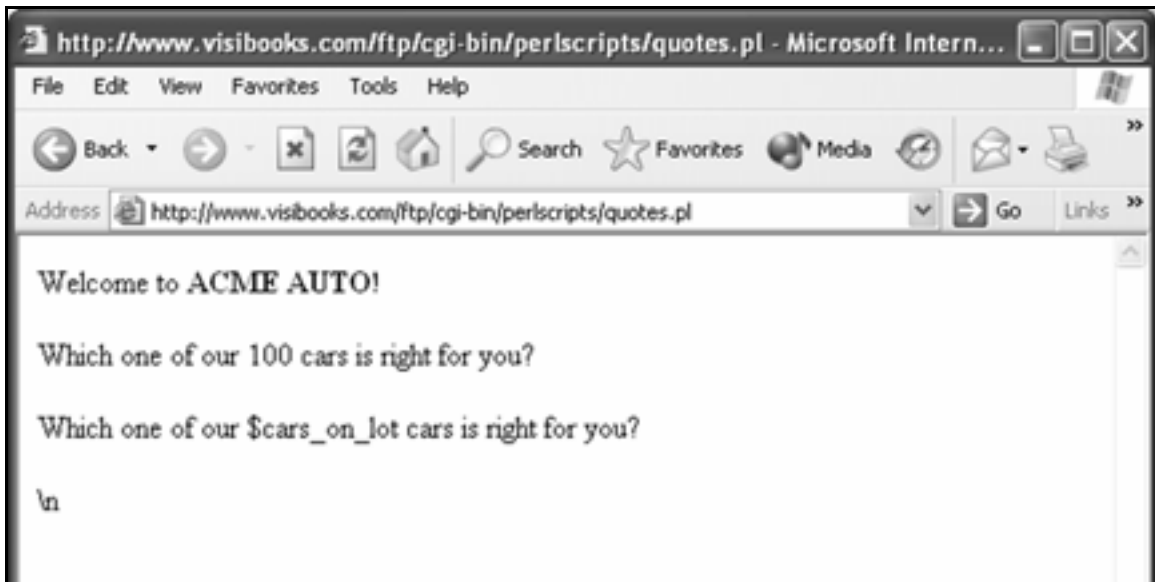
**5.** Save `perllinks.html`, then upload it to the home directory in your Web site.

**6.** In the browser, go to:

`www.yourwebsite.com/perllinks.html`

## 7. Click the Double vs. single quotes link.

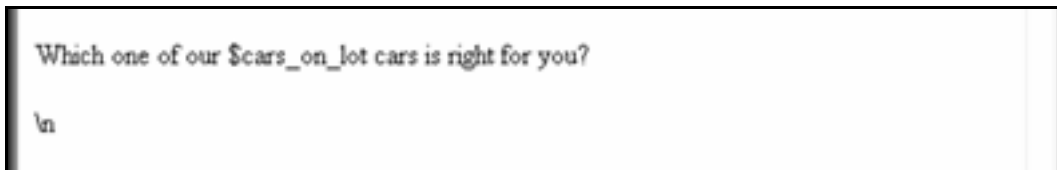
The output should look like this:



**Tip:** *Using single quotes (') with the print function*

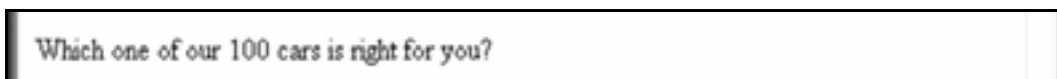
```
print '<p>Which one of $cars_on_lot is right  
for you?</p>\n' ;
```

*prints literally everything in between the two quotation marks.*



*If you want to display the value of a variable, use double quotes ("):*

```
print "<p>Which one of our $cars_on_lot cars is  
right for you?</p>\n";
```



# Employ lists of variables

## Create lists of number variables

1. Create a new script with this code:

```
#!/usr/bin/perl
print "Content-Type: text/html \n\n";

# This script demonstrates how to
# create a numeric array.

@AcmeInventory = (178,286,387);

print @AcmeInventory;

print "<p>We just created a list of numbers
using an array variable!";
```

2. Save the script as **numberlist.pl** in the **PERLSCRIPTS** folder.

Here's what the relevant lines in this script do:

- `@AcmeInventory = (178,286,387);`

The numbers 178, 286, and 387 are assigned to the array variable `@AcmeInventory`.

- `print @AcmeInventory;`

The numbers assigned the array variable `@AcmeInventory` are printed to the browser window: 178, 286, and 387.

**3.** Upload **numberlist.pl** to the **perlscripts** directory in your Web site, then set its permissions so that anyone can execute it.

**4.** Open **perllinks.html** and insert a new link to **numberlist.pl**:

```
<p><a  
href="http://www.yourwebsite.com/cgi-  
bin/perlscripts/numberlist.pl">7. Create a list  
of numbers.</a></p>
```

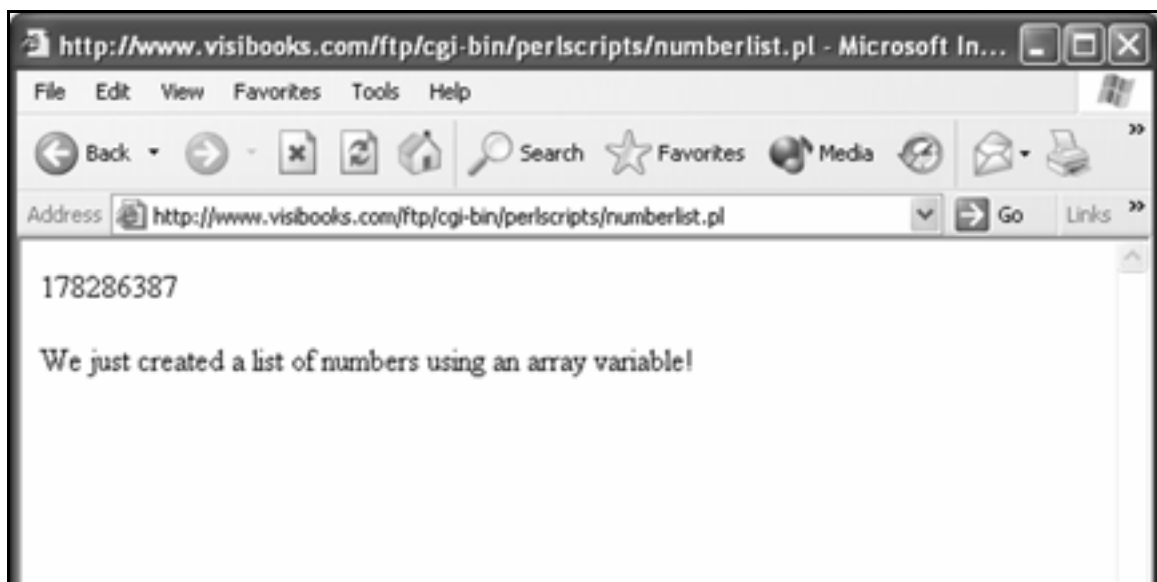
**5.** Save **perllinks.html**, then upload it to the home directory in your Web site.

**6.** In the browser, go to:

**www.yourwebsite.com/perllinks.html**

**7.** Click the **Create a list of numbers** link.

The output should look like this:



# Create lists of text variables

1. Create a new script with this code:

```
#!/usr/bin/perl
print "Content-Type: text/html \n\n";

# This script demonstrates how to
# create a text array.

@AcmeCars = ("Ford", "Dodge", "Chevy");

print "@AcmeCars";

print "<p>We have just created a text
array!</p>";
```

2. Save the script as **textlist.pl** in the **PERLSCRIPTS** folder.

Here's what the relevant lines in this script do:

- `@AcmeCars = ("Ford", "Dodge", "Chevy");`

The words "Ford", "Dodge", and "Chevy" are assigned to the array variable `@AcmeCars`.

- `print "@AcmeCars";`

The words assigned to the array value `@AcmeCars` are printed to the browser window:

**Ford Dodge Chevy**

3. Upload **textlist.pl** to the **perlscripts** directory in your Web site, then set its permissions so that anyone can execute it.

4. Open **perllinks.html** and insert a new link to **textlist.pl**:

```
<p><a  
href="http://www.yourwebsite.com/cgi-  
bin/perlscripts/textlist.pl">8. Create a list  
of text.</a></p>
```

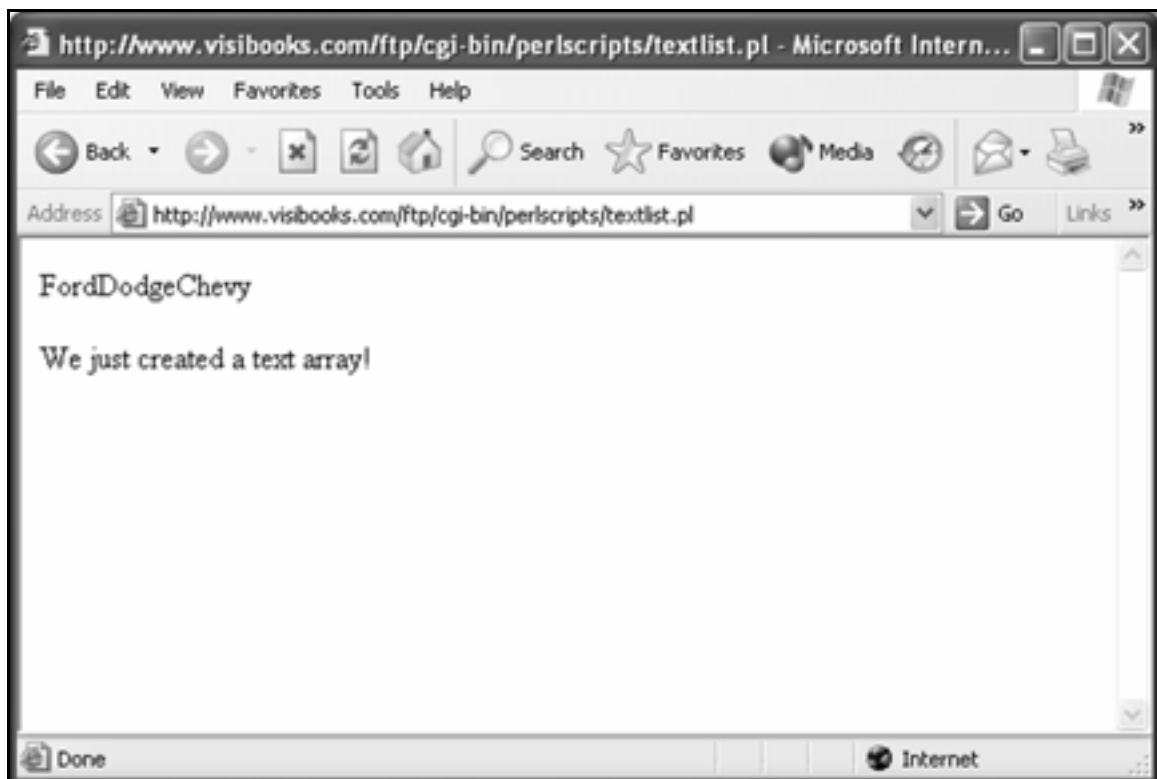
5. Save **perllinks.html**, then upload it to the home directory in your Web site.

6. In the browser, go to:

**www.yourwebsite.com/perllinks.html**

7. Click the **Create a list of text** link.

The output should look like this:



# Print an element in a list of variables

1. Create a new script with this code:

```
#!/usr/bin/perl
print "Content-Type: text/html \n\n";

# This script demonstrates how to
# print an element in an array

@AcmeCars = ("Ford", "Dodge", "Chevy");

print "<p>* $AcmeCars[0] * is the first element
in the text array.</p>";

print "<p>* $AcmeCars[2] * is the third element
in the text array.</p>";
```

**Tip:** *In PERL, numbering of array variables starts at 0 not 1.*

*This can be confusing, but it's common to many programming languages.*

2. Save the script as **printelement.pl** in the **PERLSCRIPTS** folder.

Here's what the relevant lines in the script do:

- `print "<p>* $AcmeCars[0] * is the first element in the text array.</p>";`

The word “Ford” is printed to the browser window because it’s the first word in the `@AcmeCars` array list—the zero position.

Notice that `$`—signifying a single (scalar) value—is used when printing a single element of an array. Basically, `$AcmeCars[0]` means: give me the single (scalar) value, in the zero position in the `@AcmeCars` array list.

- `print "<p>* $AcmeCars[2] * is the third element in the text array.</p>";`

The word “Chevy” is printed because it’s in the number 2 position in the `@AcmeCars` array. It’s the third item in the array list—the number two position.

- 3.** Upload `printelement.pl` to the `perlscripts` directory in your Web site, then set its permissions so that anyone can execute it.

- 4.** Open `perllinks.html` and insert a new link to `printelement.pl`:

```
<p><a
href="http://www.yourwebsite.com/cgi-
bin/perlscripts/printelement.pl">9. Print
elements in a text array.</a></p>
```

- 5.** Save `perllinks.html`, then upload it to the home directory in your Web site.

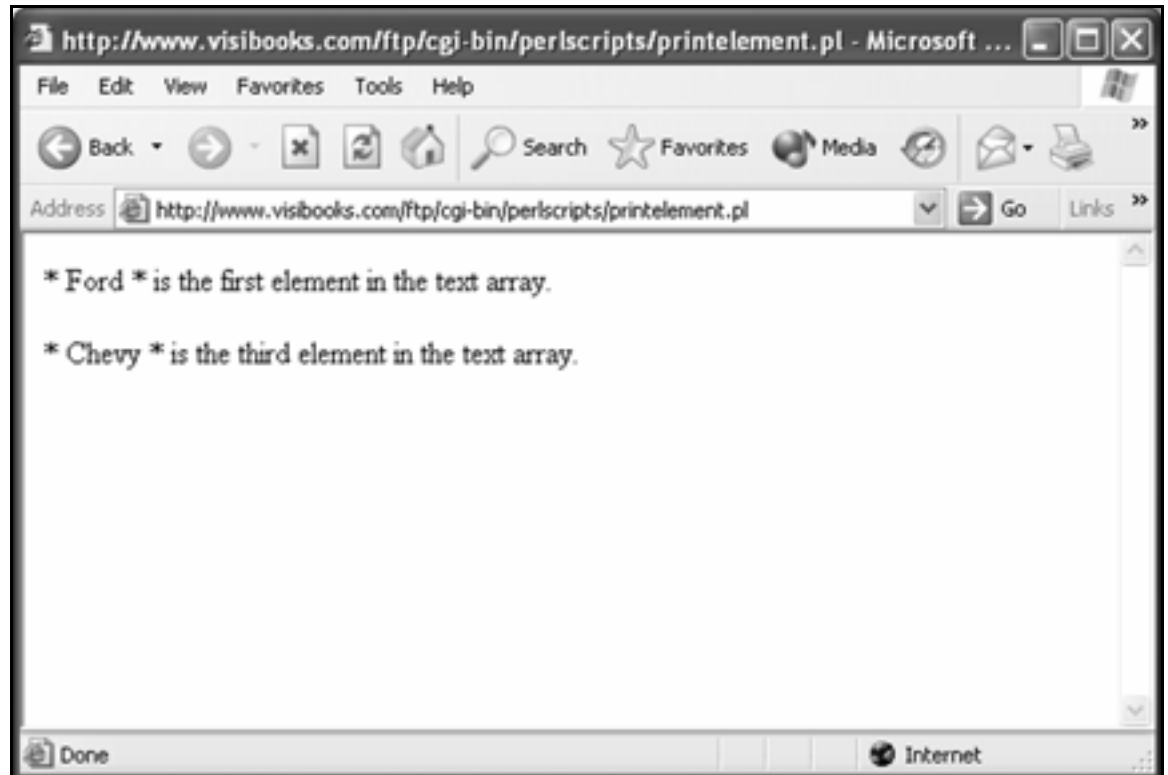


**6.** In the browser, go to:

**[www.yourwebsite.com/perllinks.html](http://www.yourwebsite.com/perllinks.html)**

**7.** Click the **Print elements in a text array** link.

The output should look like this:



**1.**

# Working with Numbers

In this section, you'll learn how to:

- **Perform calculations**
- **Increment/decrement automatically**
- **Generate random numbers**

# Perform calculations

1. Create a new script with this code:

```
#!/usr/bin/perl
print "Content-Type: text/html \n\n";

$var1 = 5;
$var2 = 2;

$answer = $var1 + $var2 ;

print "$var1 plus $var2 equals $answer.\n";
```

2. Save the script as **add.pl** in the **PERLSCRIPTS** folder.

Here's what the relevant lines in this script do:

- `$answer = $var1 + $var2 ;`

Adds the scalar variables `$var1` and `$var2` together, then assigns the sum to a scalar variable called `$answer`.

Since `$var1` is 5, and `$var2` is 2, `$answer` has a value of 7.

3. Upload **add.pl** to the **perlscripts** directory in your Web site, then set its permissions so that anyone can execute it.

4. Open **perllinks.html** and insert a new link to **add.pl**:

```
<p><a  
href="http://www.yourwebsite.com/cgi-  
bin/perlscripts/add.pl">10. Add five plus  
two.</a></p>
```

5. Save **perllinks.html**, then upload it to the home directory in your Web site.

6. In the browser, go to:

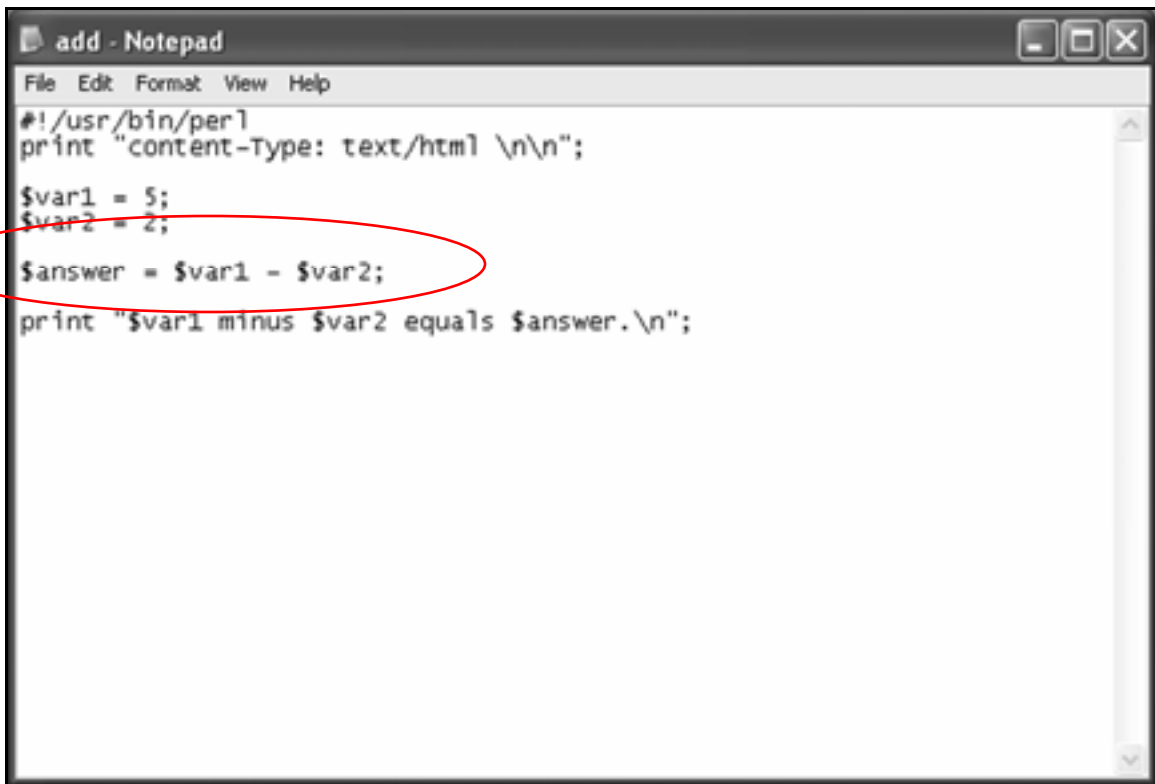
**[www.yourwebsite.com/perllinks.html](http://www.yourwebsite.com/perllinks.html)**

Click the **Add five plus two** link.

The output should look like this:



**Tip:** To subtract, just change the + sign in the script above to a - sign.



```
add - Notepad
File Edit Format View Help
#!/usr/bin/perl
print "content-type: text/html \n\n";
$var1 = 5;
$var2 = 2;
$answer = $var1 - $var2;
print "$var1 minus $var2 equals $answer.\n";
```

To multiply, just change it to a \* sign.

To divide, change it to /.

# Increment/decrement automatically

1. Create a new script with this code:

```
#!/usr/bin/perl
print "Content-Type: text/html \n\n";

$cars_on_lot = 10;

print "We have $cars_on_lot cars.\n<br>";

print "We got another new car.\n<br>";

$cars_on_lot++;

print "Now we have $cars_on_lot cars!\n<p>";

print '<b>$cars_on_lot++</b> is the same to
PERL as <b>$cars_on_lot + 1.</b>';
```

2. Save the script as **autoplus.pl** in the **PERLSCRIPTS** folder.

Here's what the relevant lines in this script do:

- `$cars_on_lot++;`

The auto incrementer (++) adds 1 to the `$cars_on_lot` variable.

- `print '<b>$cars_on_lot++</b> is the same to PERL as <b>$cars_on_lot = $cars_on_lot + 1</b>';`

Prints the literal text: `$cars_on_lot++` is the same to PERL as `$cars_on_lot + 1`.

**3.** Upload **autoplus.pl** to the **perlscripts** directory in your Web site, then set its permissions so that anyone can execute it.

**4.** Open **perllinks.html** and insert a new link to **autoplus.pl**:

```
<p><a  
href="http://www.yourwebsite.com/cgi-  
bin/perlscripts/autoplus.pl">11. Advance a  
number by 1 automatically.</a></p>
```

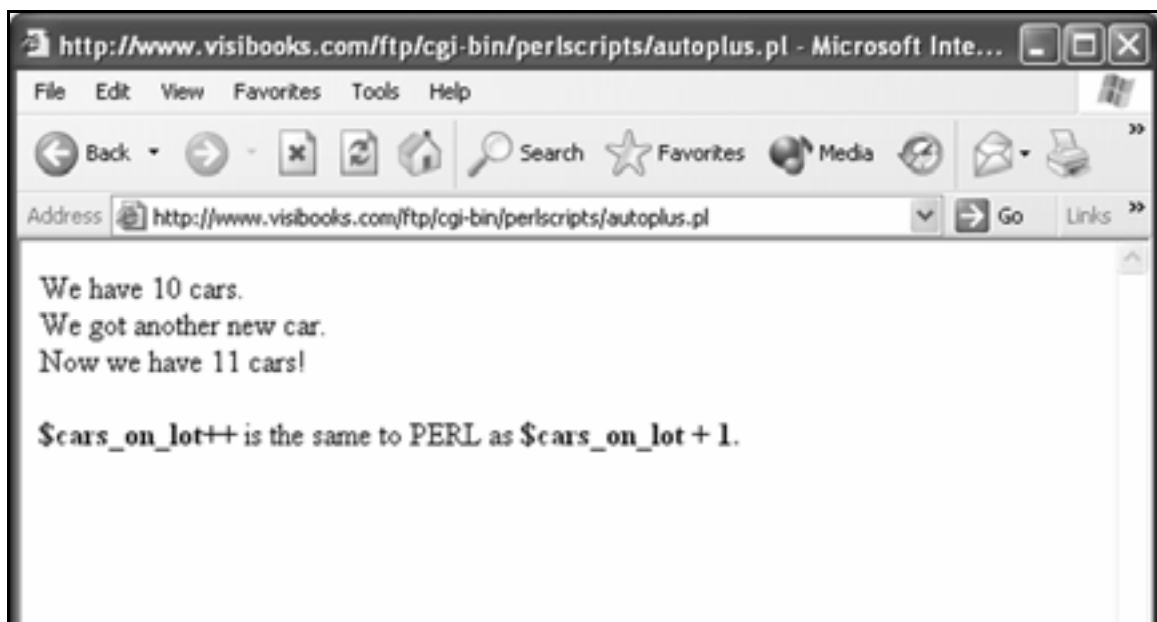
**5.** Save **perllinks.html**, then upload it to the home directory in your Web site.

**6.** In the browser, go to:

**[www.yourwebsite.com/perllinks.html](http://www.yourwebsite.com/perllinks.html)**

**7.** Click the **Advance a number by 1 automatically** link.

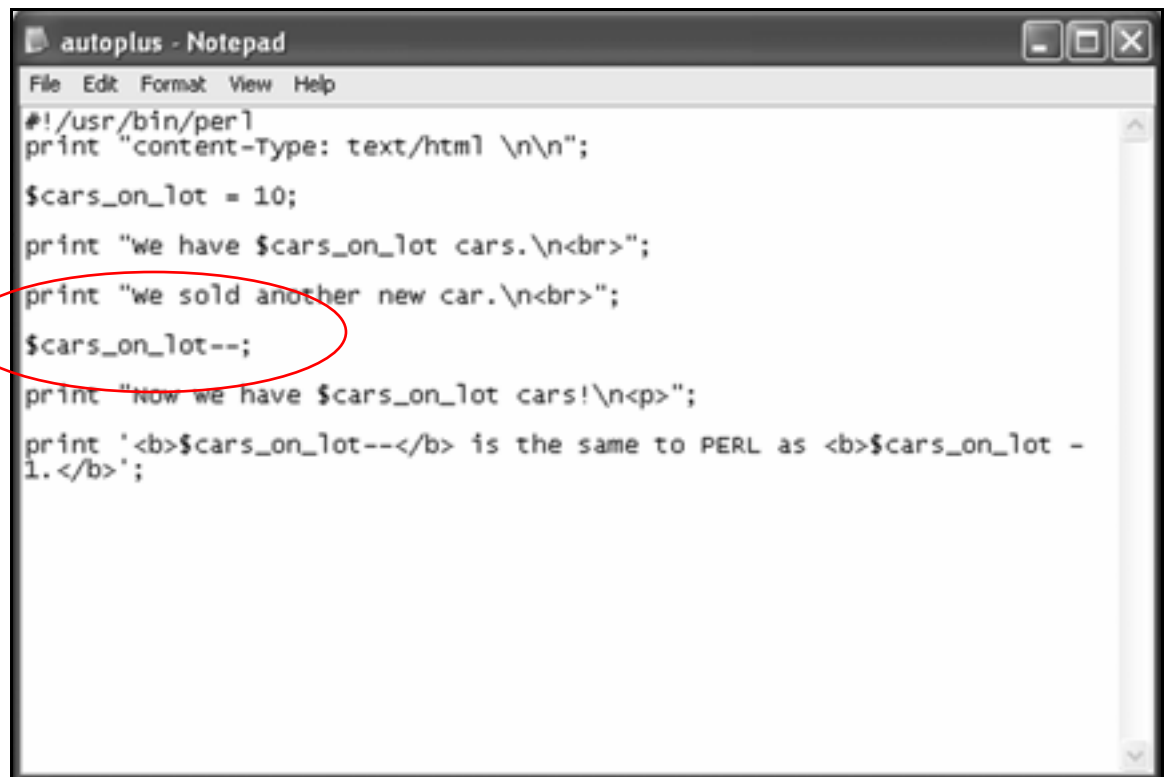
The output should look like this:





**Tip:** To automatically decrement by one, change the auto incremter in the script above (++) to an auto decremter:

--



```
File Edit Format View Help
#!/usr/bin/perl
print "content-type: text/html \n\n";

$cars_on_lot = 10;
print "we have $cars_on_lot cars.\n<br>";
print "we sold another new car.\n<br>";
$cars_on_lot--;
print "now we have $cars_on_lot cars!\n<p>";
print '<b>$cars_on_lot--</b> is the same to PERL as <b>$cars_on_lot -
1.</b>';
```

# Generate random numbers

1. Create a new script with this code:

```
#!/usr/bin/perl
print "Content-Type: text/html \n\n";

$random_number = rand(10);

print "<p>Your Acme Auto Lucky Number from 1 to
10 is $random_number.</p>\n";

$random_integer = int(rand(10)) + 1;

print "<p>Your Acme Auto Lucky Integer from 1
to 10 is $random_integer.</p>\n";

print "Click the Reload button on your browser
to get a new random number.";
```

## 2. Save the script as **random.pl** in the **PERLSCRIPTS** folder.

Here's what the relevant lines in this script do:

- 
- `$random_number = rand(10);`

Assigns a computer-generated random number to the scalar variable `$random_number`.

Because the number 10 is inside the parenthesis:

```
rand(10)
```

the computer-generated number will be between 0 and 9.999999999999999.

If you'd used the number 100—`rand(100)` it would be between 0 and 99.999999999999999.

- `print "<p>Your Acme Auto Lucky Number from 1 to 10 is $random_number .</p>\n";`

Prints **“Your Acme Auto Lucky Number from 1 to 10 is 8.77515995948674.”**

Because it is a random number, the number on your screen will be different.

- `$random_integer = int(rand(10)) + 1;`

Creates a random number:

```
$random_integer = int(rand(10)) + 1;
```

Turns it into an integer:

```
$random_integer = int(rand(10)) + 1;
```

An integer is a whole number, without decimal points, like 1, 2, 3.

Makes sure the number generated is between 1 and 10, and not 0 and 10:

```
$random_integer = int(rand(10)) + 1;
```

Then assigns it to the scalar variable

```
$random_integer:
```

```
$random_integer = int(rand(10)) + 1;
```

- `print "<p>Your Acme Auto Lucky Integer from 1 to 10 is $random_integer.</p>\n";`

Prints “**Your Acme Auto Lucky Integer from 1 to 10 is 5.**”

Of course, your lucky number may be different than 5, because it’s a random number being generated.

- 3.** Upload **random.pl** to the **perlscripts** directory in your Web site, then set its permissions so that anyone can execute it.

4. Open **perllinks.html** and insert a new link to **random.pl**:

```
<p><a  
href="http://www.yourwebsite.com/cgi-  
bin/perlscripts/random.pl">12. Generate random  
numbers.</a></p>
```

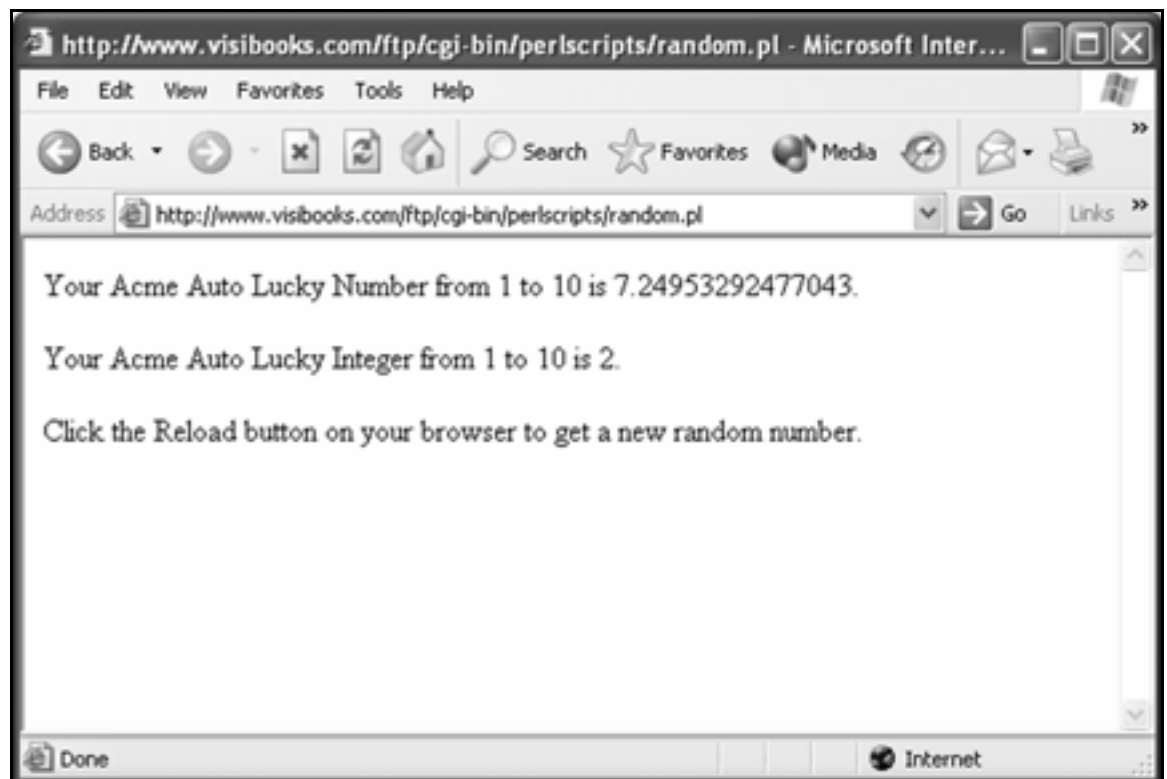
5. Save **perllinks.html**, then upload it to the home directory in your Web site.

6. In the browser, go to:

**[www.yourwebsite.com/perllinks.html](http://www.yourwebsite.com/perllinks.html)**

7. Click the **Generate random numbers** link.

The output should look like this:





# Subroutines

In this section, you'll learn how to:

- **Create a subroutine**
- **Parse form data with a subroutine**

# Create a subroutine

## What is a Subroutine?

A subroutine is block of reusable code that you create within your program.

Instructions within the subroutine can be called or executed from the main program more then once. This makes redundant tasks simpler.

1. Create a new script with this code:

```
#!/usr/bin/perl
print "Content-Type: text/html \n\n";

print "Program starts.\n";

&bigHeader;

print "Program ends.\n";

# subroutines below this line

sub bigHeader {
    print "<h1>Welcome to Acme Auto!</h1>\n";
}
```



## 2. Save the script as **subsimple.pl** in the PERLSCRIPTS folder.

Here's what the relevant lines in this script do:

- `&bigHeader;`

`bigHeader` is the name of the subroutine.

The `&` sign before the name of the subroutine tells the Web server to execute the subroutine.

- `sub bigHeader {`

`sub` defines this as a subroutine.

`bigHeader` is the name of the subroutine. You can use any name you want, but it should describe what the subroutine does.

`{` marks the beginning of what the subroutine does.

- `print "<h1>Welcome to Acme Auto</h1>\n";`

This is what the subroutine does: it prints the phrase “**Welcome to Acme Auto!**” to the browser window in large, bold type.

This is a very simple subroutine, but you can put as much PERL code as you want in a subroutine.

- `}`

`}` marks the end of the subroutine.

## 3. Upload **subsimple.pl** to the **perlscripts** directory in your Web site, then set its permissions so that anyone can execute it.

4. Open **perllinks.html** and insert a new link to **subsimple.pl**:

```
<p><a  
href="http://www.yourwebsite.com/cgi-  
bin/perlscripts/subsimple.pl">13. Execute a  
subroutine.</a></p>
```

5. Save **perllinks.html**, then upload it to the home directory in your Web site.

6. In the browser, go to:

**[www.yourwebsite.com/perllinks.html](http://www.yourwebsite.com/perllinks.html)**

7. Click the **Execute a subroutine** link.

The output should look like this:



# Parse form data with a subroutine

## Create a form

1. Create a new Web page with this code:

```
<html>
<head>
<title>Dream Car</title>
</head>

<body>

<form method="post"
action="http://www.yourwebsite.com/cgi-
bin/perlscripts/dreamcar.pl">

<h2>What's my dream car?</h2>

Make: <input type="text" name="make"><br>

Model: <input type="text" name="model"><br>

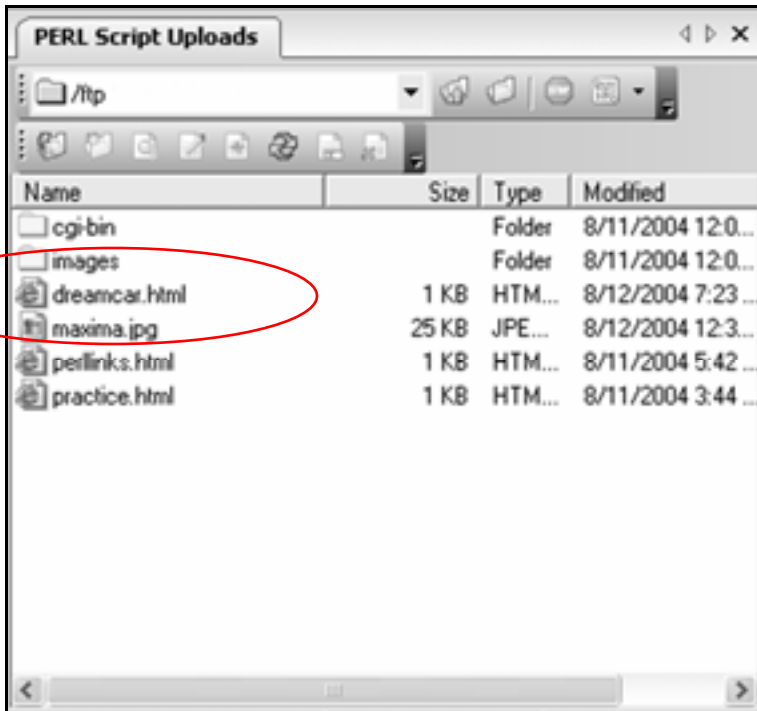
<input type="submit" value="Submit">

</form>

</body>
</html>
```

2. Save the page as **dreamcar.html** in the **PERLSCRIPTS** folder.

**3.** Upload it to the home directory in your Web site.



# Parse form data

1. Create a new script with this code:

```
#!/usr/bin/perl
print "Content-type: text/html\n\n";

&getFormData;

print "<h2>Here's my dream car:</h2>";

print "Make: $request{'make'}";
print "<br>\n";
print "Model: $request{'model'}";

# Subroutine below this line.

sub getFormData {

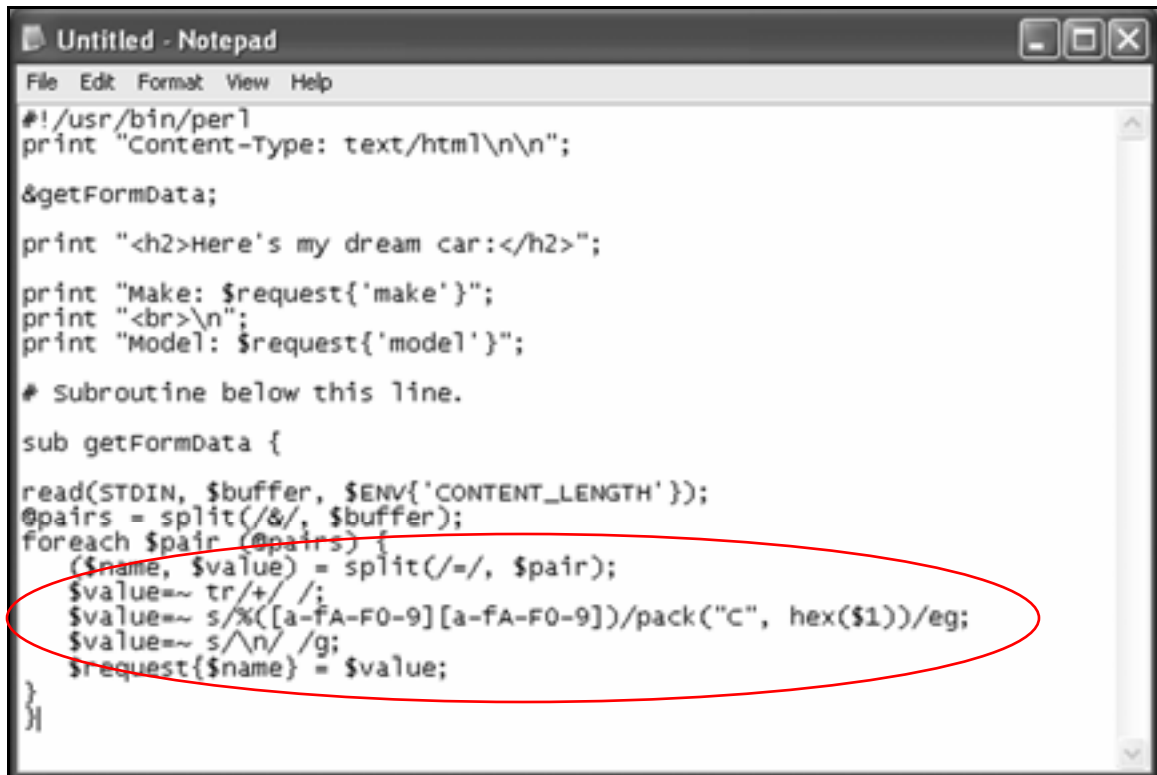
read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});
@pairs = split(/&/, $buffer);
foreach $pair (@pairs) {
    ($name, $value) = split(/=/, $pair);
    $value =~ tr/+//;
    $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
    $value =~ s/\n//g;
    $request{$name} = $value;
}
}
```

**Tip:** *When you write the code*

```
$value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
```

*in the script above, you must write it on a single line, or the script will not work.*

*It must look like this:*



```
Untitled - Notepad
File Edit Format View Help
#!/usr/bin/perl
print "Content-Type: text/html\n\n";
&getFormData;
print "<h2>Here's my dream car:</h2>";
print "Make: $request{'make'}";
print "<br>\n";
print "Model: $request{'model'}";
# Subroutine below this line.
sub getFormData {
read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});
@pairs = split(/&/, $buffer);
foreach $pair (@pairs) {
($name, $value) = split(/=/, $pair);
$value =~ tr/+//;
$value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
$value =~ s/\n//g;
$request{$name} = $value;
}
}
```

## 4. Save the script as **dreamcar.pl** in the **PERLSCRIPTS** folder.

Here's what each line of the script means:

- `&getFormData;`

Executes the subroutine `getFormData` to pull information from the form input boxes in **dreamcar.html**.

- ```
print "Make: $request{'make'}";
print "<br>\n";
print "Model: $request{'model'}";
```

This requests the information that `getFormData` pulled from the form input boxes in the Web page **dreamcar.html**:

```
print "Model: $request{'model'}";
```

Remember, one of the input boxes in **dreamcar.html** is named "model:"

```
<input type="text" name="model">
```

Then it prints it to the browser window:

```
print "Model: $request{'model'}";
```

- `sub getFormData {`

Defines `getFormData` as a subroutine, then starts it.

- ```
read (STDIN, $buffer,
ENV{'CONTENT_LENGTH'});
    @pairs = split(/&/, $buffer);
    foreach $pair (@pairs) {
        ($name, $value) = split(/=/, $pair);
        $value =~ tr/+/ /;
        $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
        $value =~ s/\n/ /g;
        $request{$name} = $value;
    }
```

This is the `getFormData` subroutine.

It takes the raw text from the form input boxes and “parses” it, or puts it in a form that PERL can use.

This subroutine uses environmental variables and regular expressions that are beyond the scope of this book.

Don’t worry about not understanding it: after you’ve finished this book, you can move on to other, more advanced PERL books that explain parsing subroutines like this one in detail.

**Tip:** *Remember that when you write the code*

```
$value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
```

*in the script above, you must write it on a single line, or the script will not work.*

- }

Ends the `getFormData` subroutine.



5. Upload **dreamcar.pl** to the **perlscripts** directory in your Web site, then set its permissions so that anyone can execute it.
6. In the browser, go to:  
**www.yourwebsite.com/dreamcar.html**
7. Fill in the form boxes, then click the  button.

The output should look something like this:





# Logic & Loops

In this section, you'll learn how to:

- **Employ conditional logic**
- **Employ looping**

# Employ conditional logic

## If statements

1. Create a new Web page with this code:

```
<html>
<head>
<title>If Statements</title>
</head>

<body>

<h2>Acme Logon Page</h2>

<form method="POST"
action="http://www.yourwebsite.com/cgi-
bin/perlscripts/if.pl">

<h2>Enter Password:</h2>

Password: <input type="password" name=
"password"><p>

<input type="submit" value="Submit">

</form>

</body>
</html>
```

2. Save the page as **if.html** in the **PERLSCRIPTS** folder.

- 3.** Upload it to the home directory in your Web site.
- 4.** Create a new script with this code:

```
#!/usr/bin/perl
print "Content-Type: text/html \n\n";

&getFormData;

$GoodPassword = 'acme';

if ($request{'password'} eq $GoodPassword){
    print "<b>Acme Password verified!</b>\n";
}

sub getFormData {

read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});
@pairs = split(/&/, $buffer);
foreach $pair (@pairs) {
    ($name, $value) = split(/=/, $pair);
    $value =~ tr/+//;
    $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
    $value =~ s/\n//g;
    $request{$name} = $value;
}
}
```

- 5.** Save the script as **if.pl** in the **PERLSCRIPTS** folder, then upload it to the **perlscripts** directory in your Web site.

Here's what the relevant lines in this script do:

- `$GoodPassword = 'acme' ;`

Assigns the value “acme” to the variable `$GoodPassword`.

- `if ($request{'password'} eq $GoodPassword) {`

Compares the password word typed in the text box on `if.html` to the password assigned to the variable `$GoodPassword`.

If they're the same, then the code between the curly braces is executed:

```
print "<b>Acme Password  
verified!</b><br>\n" ;
```

**Tip:** *The command `eq` is used to compare to text variables. Don't use the `=` sign—that's for comparing numbers.*

- 6.** Upload **if.pl** to the **perlscripts** directory in your Web site, then set its permissions so that anyone can execute it.

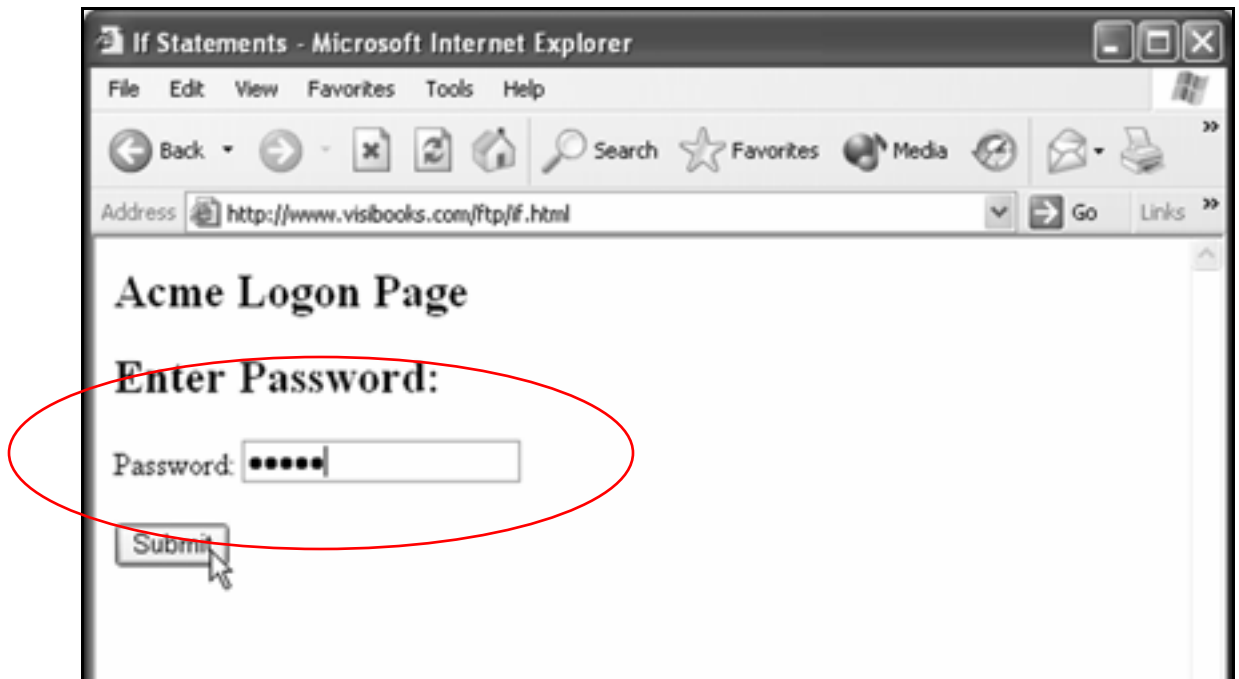
- 7.** In the browser, go to:

**[www.yourwebsite.com/if.html](http://www.yourwebsite.com/if.html)**

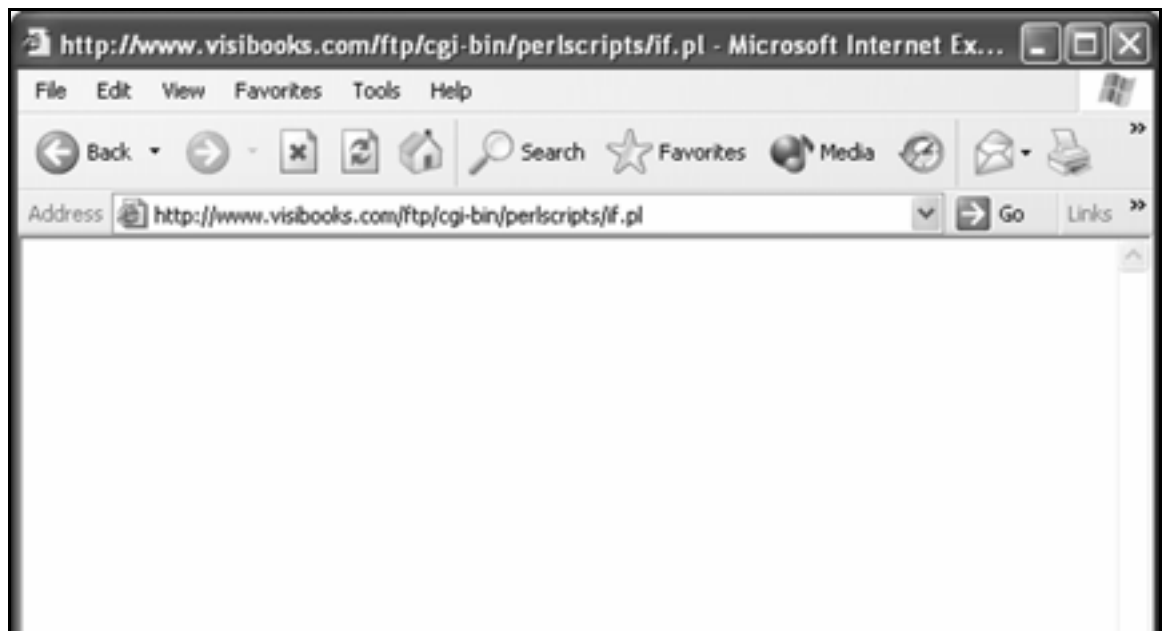
**8.** In the Password box, type:

**pizza**

then click the  button.



The output should look like this:

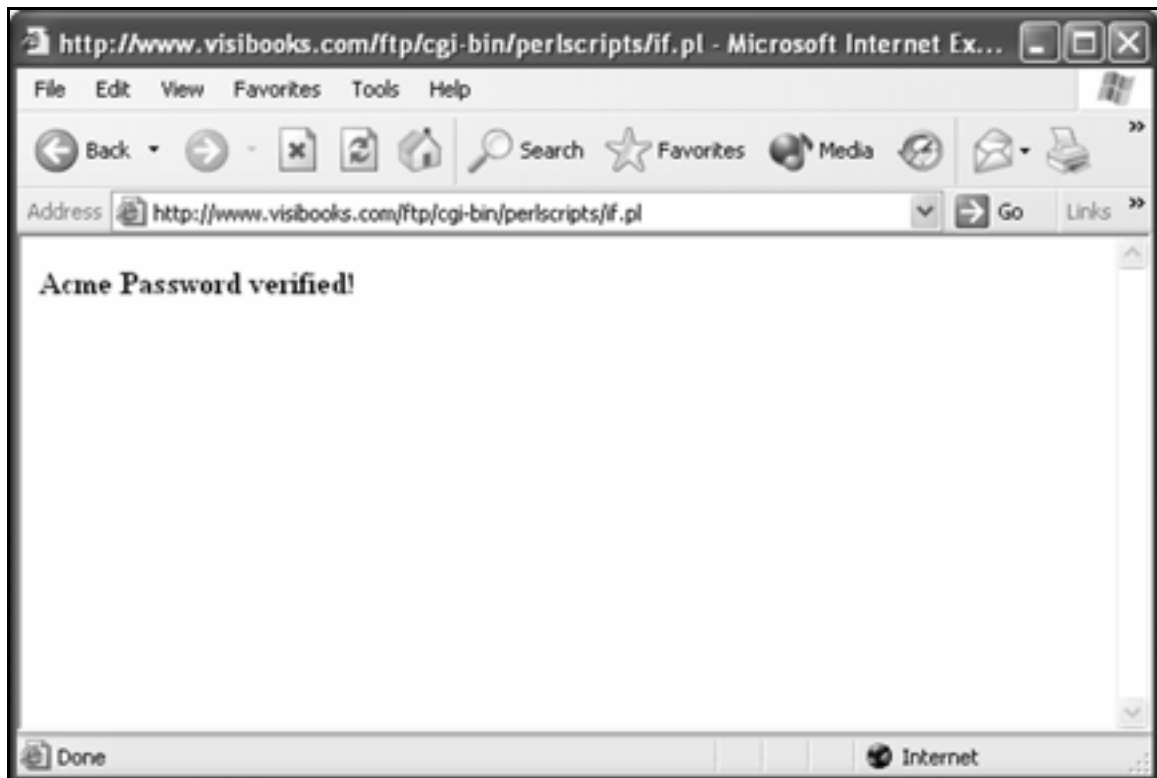


**9.** Go back to **if.html** and in the Password box, type:

**acme**

then click the  button.

The output should look like this:





## If/else statements

- 1.** In the Web page **if.html**, change the action of its `<form>` tag to use a script called **ifelse.pl**:

```
<form method="post"
action="http://www.yourwebsite.com/cgi-
bin/perlscripts/ifelse.pl">
```

- 2.** Save the page as **ifelse.html** in the **PERLSCRIPTS** folder, then upload it to the home directory in your Web site.

- 3.** In the script **if.pl**, change its code from this:

```
if ($request{'password'} eq $GoodPassword) {
print "<b>Acme Password verified!</b>\n";
}
```

To this:

```
if ($request{'password'} eq $GoodPassword) {
print "<b>Acme Password verified!</b>\n";
}

else {
print "<b>Acme Password incorrect.</b>\n";
}
```

4. Save the script as **ifelse.pl** in the **PERLSCRIPTS** folder.

Here's what the relevant lines in this script do:

- `if ($request{'password'} eq $GoodPassword) {`

Compares the password word typed in the text box on if.html to the password assigned to the variable `$GoodPassword`.

If they're the same, then the code between the curly braces is executed.

- `else {`

If the two values are NOT the same then the else condition is executed. The else condition is the block of code in between the curly braces after the word 'else.'

**Tip:** *Think of it this way:*

```
if (this is true) { then do this }  
  
else { do this }
```

5. Upload **ifelse.pl** to the **perlscripts** directory in your Web site, then set its permissions so that anyone can execute it.

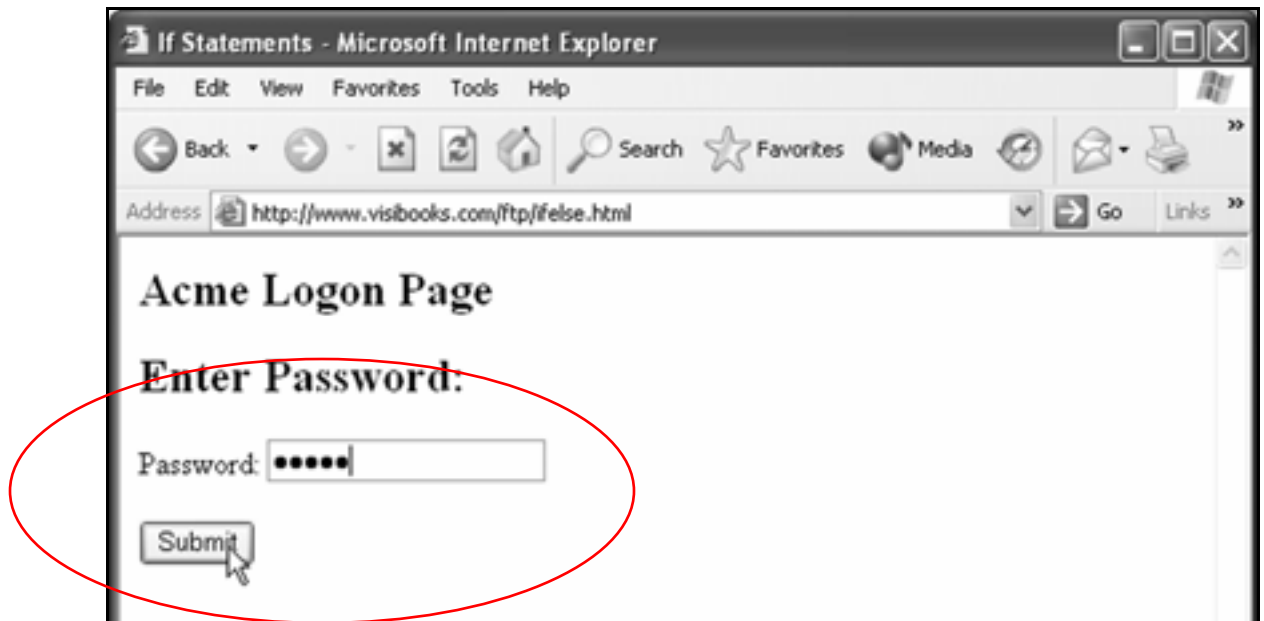
6. In the browser, go to:

**[www.yourwebsite.com/ifelse.html](http://www.yourwebsite.com/ifelse.html)**

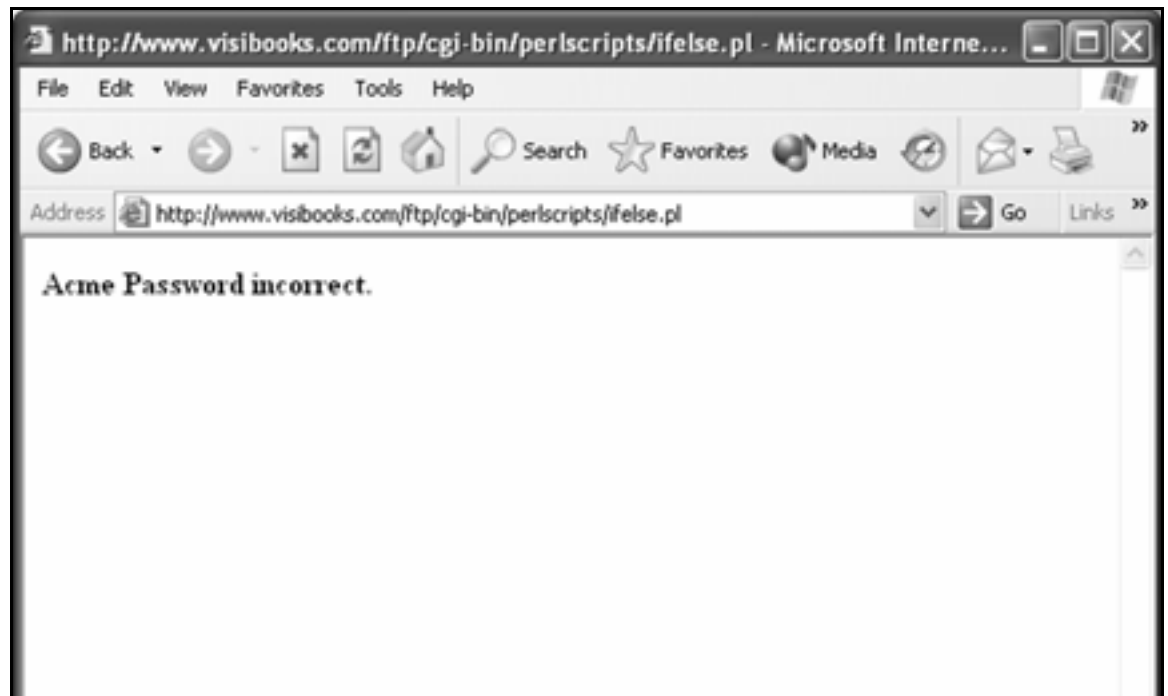
**7.** In the Password box, type:

**pizza**

then click the  button.

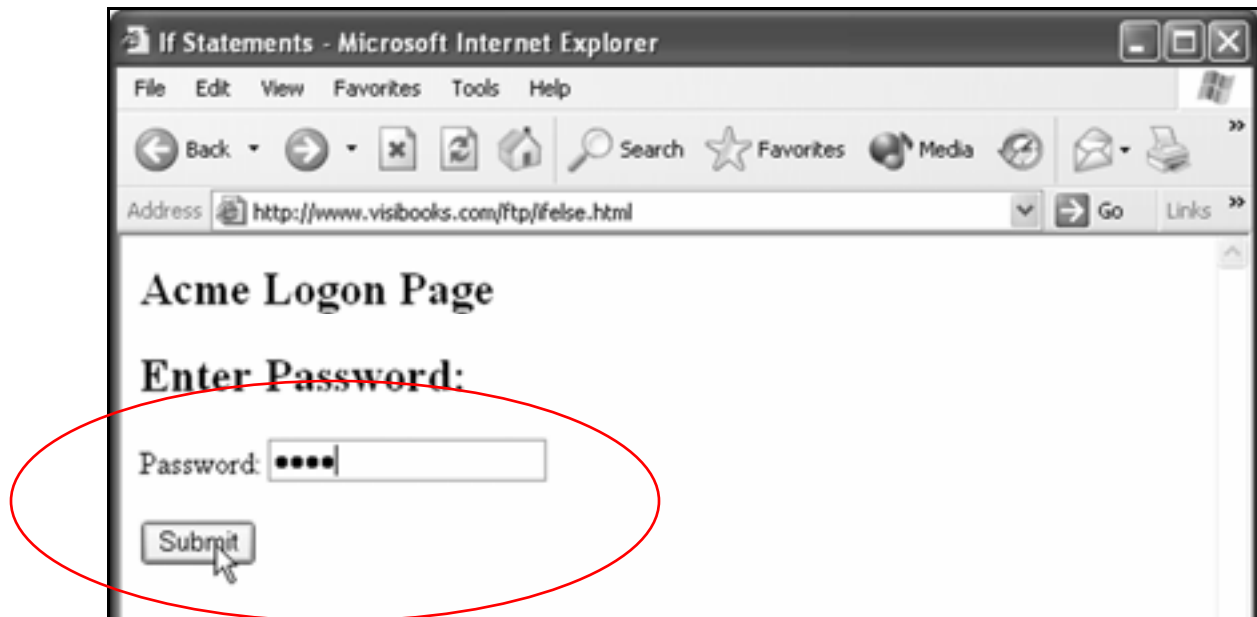


The output should look like this:

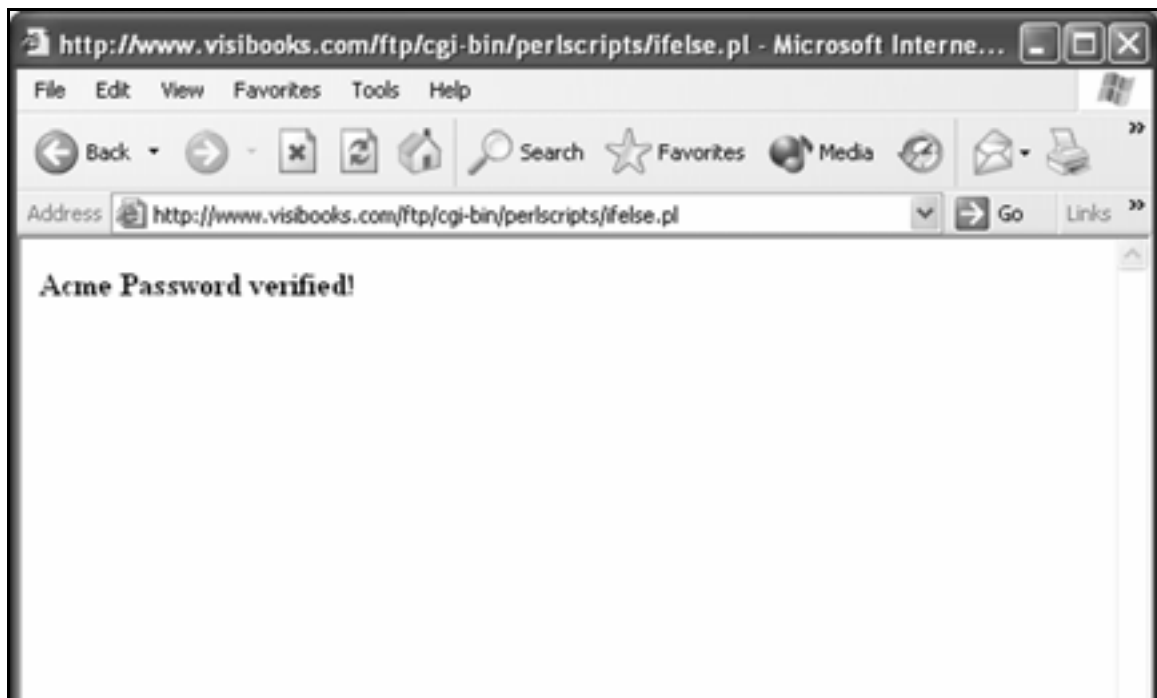


- 8.** Go back to **ifelse.html** and In the Password box, type:  
**acme**

then click the  button.



The output should look like this:



# The OR operator

1. Create a new Web page with this code:

```
<html>
<head>
<title>The OR Operator</title>
</head>

<body>

<form method="post"
action="http://www.yourwebsite.com/cgi-
bin/perlscripts/or.pl">

<h2>Enter Acme Auto User Name</h2>

User Name: <input type="text" name= "username">

<input type="submit" value="Submit">

</form>

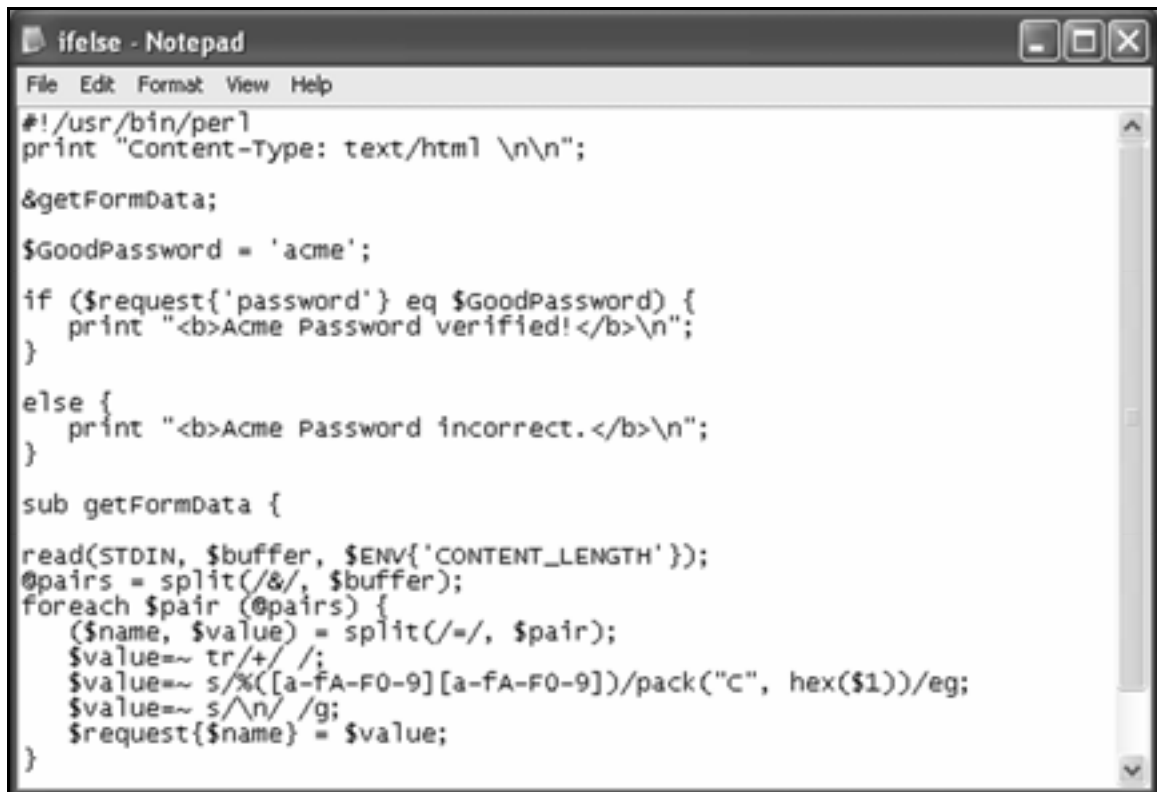
</body>
</html>
```

2. Save the page as **or.html** in the **PERLSCRIPTS** folder, then upload it to the home directory in your Web site.

### 3. Change the code in `ifelse.pl` from this:

```
$GoodPassword = 'acme';

if ($request{'password'} eq $GoodPassword) {
print "<b>Acme Password verified!</b>\n";
}
else {
print "<b>Acme Password incorrect.</b>\n";
}
```



```
ifelse - Notepad
File Edit Format View Help
#!/usr/bin/perl
print "Content-Type: text/html \n\n";

&getFormData;

$GoodPassword = 'acme';

if ($request{'password'} eq $GoodPassword) {
    print "<b>Acme Password verified!</b>\n";
}

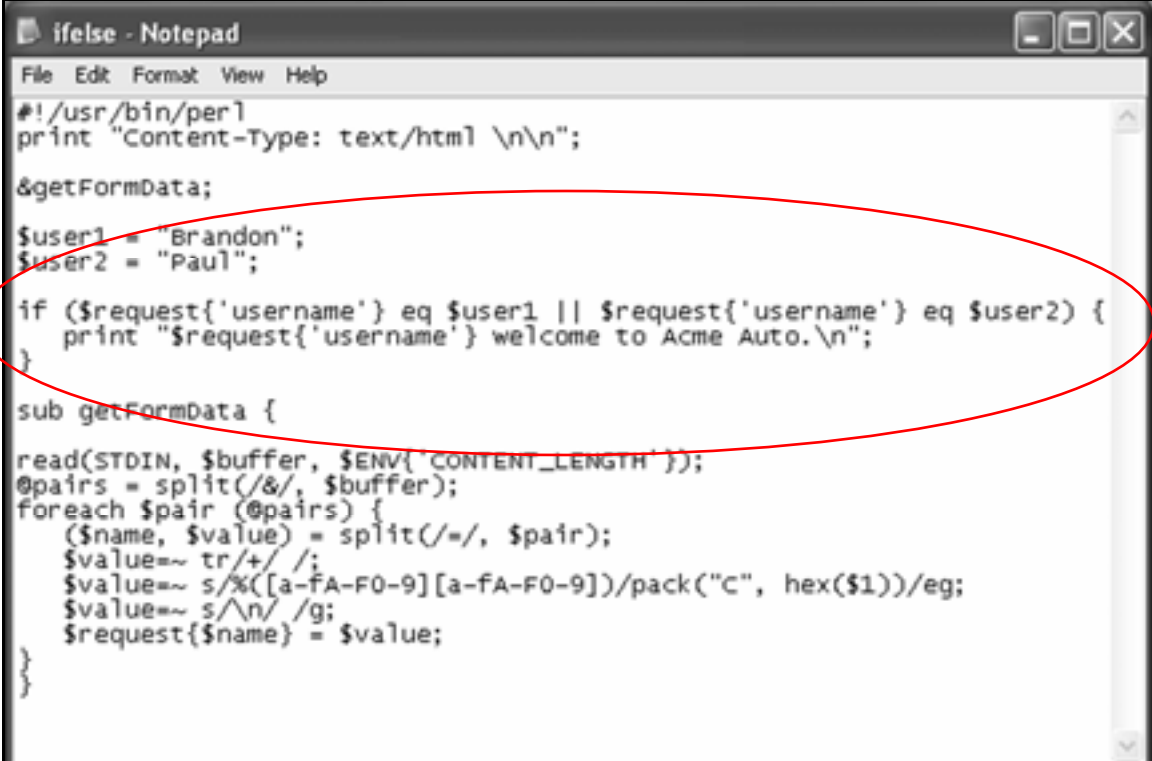
else {
    print "<b>Acme Password incorrect.</b>\n";
}

sub getFormData {
    read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});
    @pairs = split(/&/, $buffer);
    foreach $pair (@pairs) {
        ($name, $value) = split(/=/, $pair);
        $value =~ tr/+//;
        $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("c", hex($1))/eg;
        $value =~ s/\n//g;
        $request{$name} = $value;
    }
}
```

To this:

```
$user1 = "Brandon";  
$user2 = "Paul";
```

```
if ($request{'username'} eq $user1 ||  
$request{'username'} eq $user2) {  
print "$request{'username'} welcome to Acme  
Auto.\n";  
}
```



```
ifelse - Notepad  
File Edit Format View Help  
#!/usr/bin/perl  
print "Content-Type: text/html \n\n";  
&getFormData;  
$user1 = "Brandon";  
$user2 = "Paul";  
if ($request{'username'} eq $user1 || $request{'username'} eq $user2) {  
    print "$request{'username'} welcome to Acme Auto.\n";  
}  
sub getFormData {  
    read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});  
    @pairs = split(/&/, $buffer);  
    foreach $pair (@pairs) {  
        ($name, $value) = split(/=/, $pair);  
        $value =~ tr/+//;  
        $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("c", hex($1))/eg;  
        $value =~ s/\\n//g;  
        $request{$name} = $value;  
    }  
}
```

4. Save the script as **or.pl** in the **PERLSCRIPTS** folder.

Here's what the relevant lines in this script do:

- `if ($request{'username'} eq $user1 || $request{'username'} eq $user2)`

Uses the OR operator:

```
||
```

to compare two conditions.

It is asking the question, “is condition one true, OR is condition two true?” Is the entered user name either Brandon OR Paul?

If the entered user name is either Brandon or Paul, then the block of code in between the curly braces is executed.

5. Upload **or.pl** to the **perlscripts** directory in your Web site, then set its permissions so that anyone can execute it.

6. In the browser, go to:

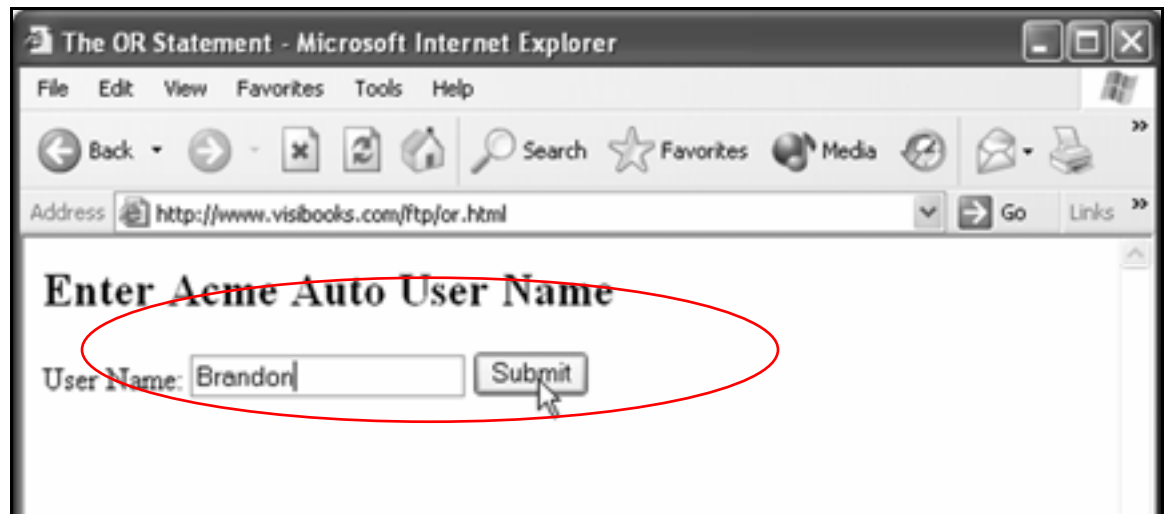
**[www.yourwebsite.com/or.html](http://www.yourwebsite.com/or.html)**



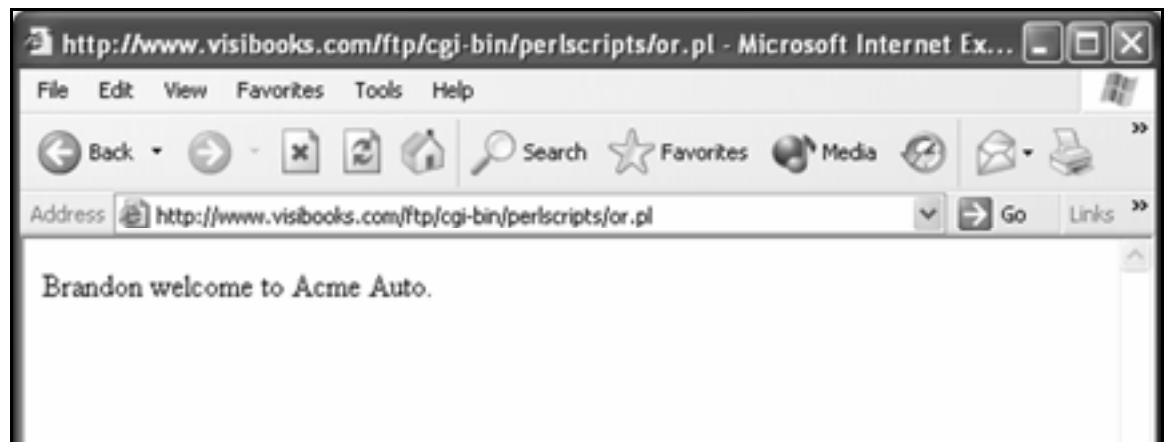
**7.** In the User Name box, type:

**Brandon**

then click the  button.



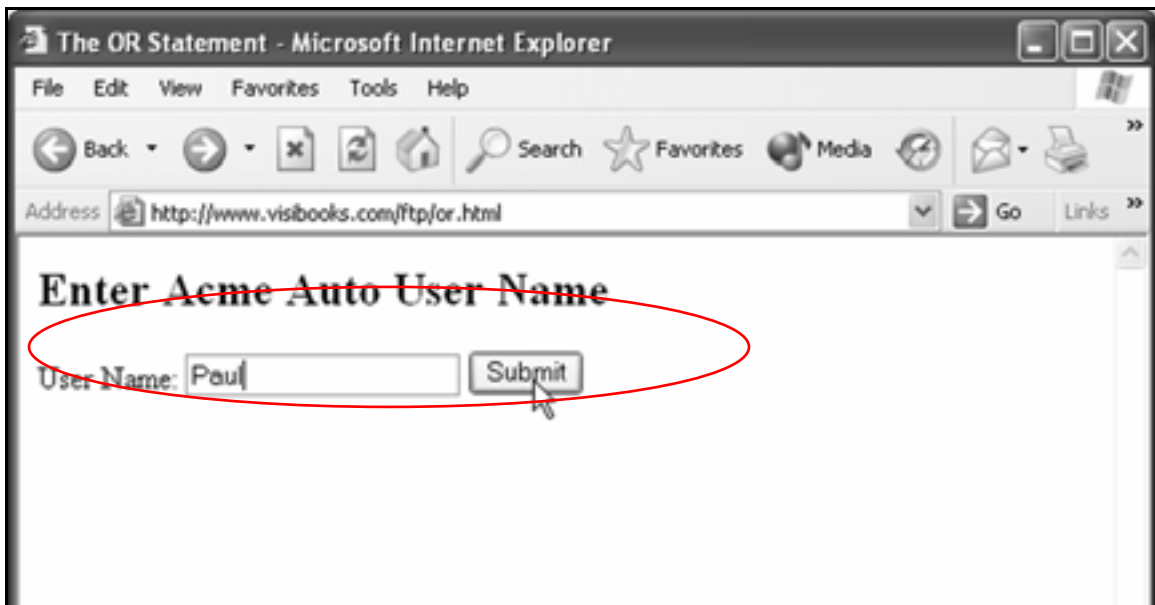
The output should look like this:



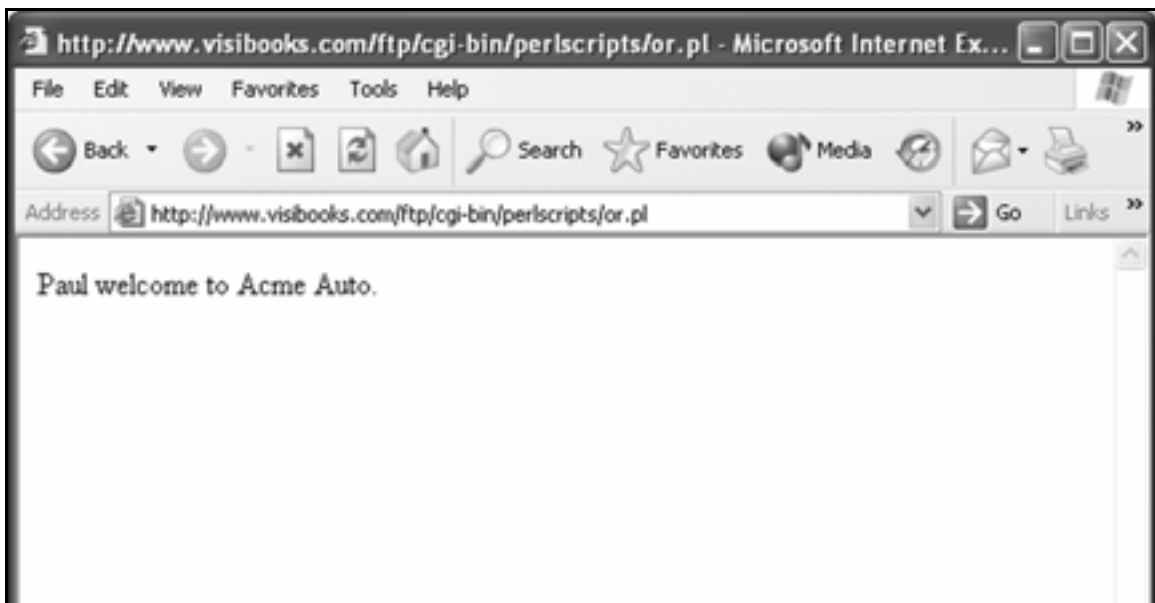
**8.** Go back to **or.html** and in the User Name box, type:

**Paul**

then click the  button.



The output should look like this:



# The AND operator

1. Create a new Web page with this code:

```
<html>
<head>
<title>The AND Operator</title>
</head>
<body>

<form method="post" action="
http://www.yourwebsite.com/cgi-
bin/perlscripts/and.pl">

<h2>Acme Logon Page</h2>

<h3>Enter User Name & Password</h3>

User Name: <input type="text" name=
"username"><br>

Password: <input type="password"
name="password"><br>

<input type="submit" value="Submit">

</form>

</body>
</html>
```

2. Save the page as **and.html** in the **PERLSCRIPTS** folder, then upload it to the home directory in your Web site.

### 3. Change the code in `or.pl` from this:

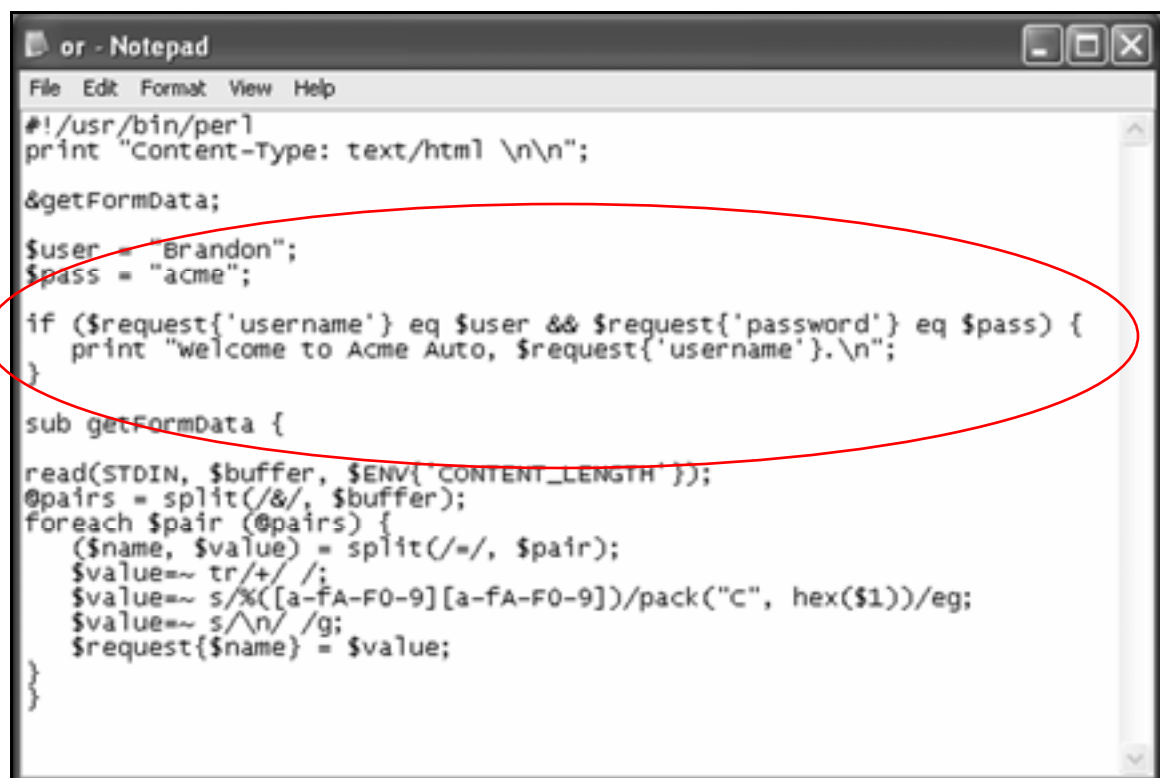
```
$user1 = "Brandon";
$user2 = "Paul";

if ($request{'username'} eq $user1 ||
    $request{'username'} eq $user2) {
    print "$request{'username'} welcome to Acme
    Auto.\n";
}
```

To this:

```
$user = "Brandon";
$pass = "acme";

if ($request{'username'} eq $user &&
    $request{'password'} eq $pass) {
    print "Welcome to Acme Auto,
    $request{'username'}.\n";
}
```



```
or - Notepad
File Edit Format View Help
#!/usr/bin/perl
print "Content-Type: text/html \n\n";

&getFormData;

$user = "Brandon";
$pass = "acme";

if ($request{'username'} eq $user && $request{'password'} eq $pass) {
    print "Welcome to Acme Auto, $request{'username'}.\n";
}

sub getFormData {
    read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});
    @pairs = split(/&/, $buffer);
    foreach $pair (@pairs) {
        ($name, $value) = split(/=/, $pair);
        $value =~ tr/+//;
        $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
        $value =~ s/\n//g;
        $request{$name} = $value;
    }
}
```

4. Save the script as **and.pl** in the **PERLSCRIPTS** folder.

Here's what the relevant lines in this script do:

- `if ($request{'username'} eq $user && $request{'pass'} eq $pass)`

Uses the AND operator:

`&&`

to compare two conditions.

It's asking:

Is the word entered in the textbox named **username** the same as the word assigned to the variable **\$user**?

AND

Is the word entered in the textbox named **password** the same as the word assigned to the variable **\$pass**?

If both these things are true, then execute the code in the curly braces.

5. Upload **and.pl** to the **perlscripts** directory in your Web site, then set its permissions so that anyone can execute it.

6. In the browser, go to:

**[www.yourwebsite.com/and.html](http://www.yourwebsite.com/and.html)**

**7.** In the User Name box, type:

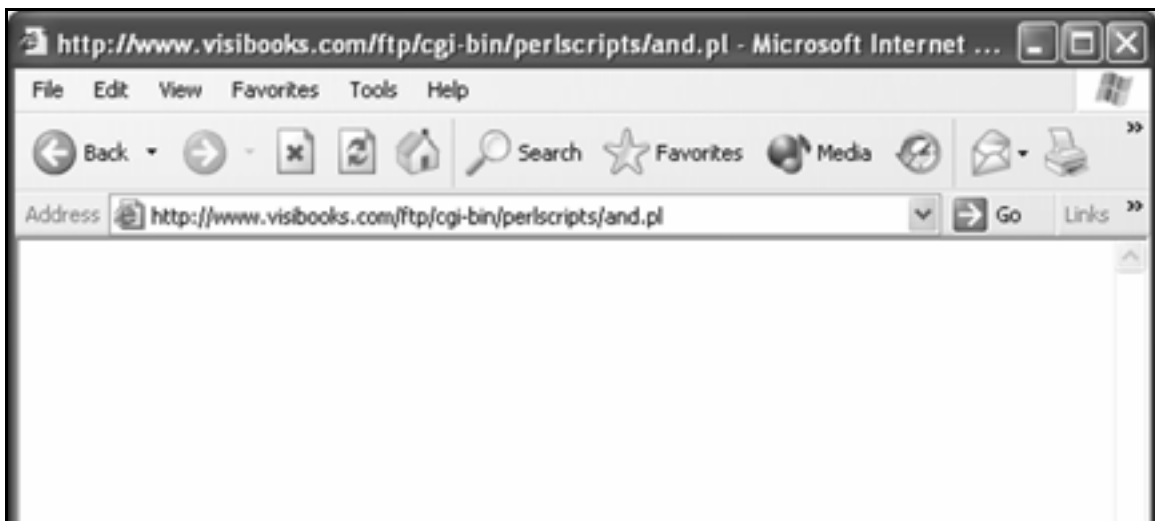
**Brandon**

**8.** In the Password box, type:

**asdf**

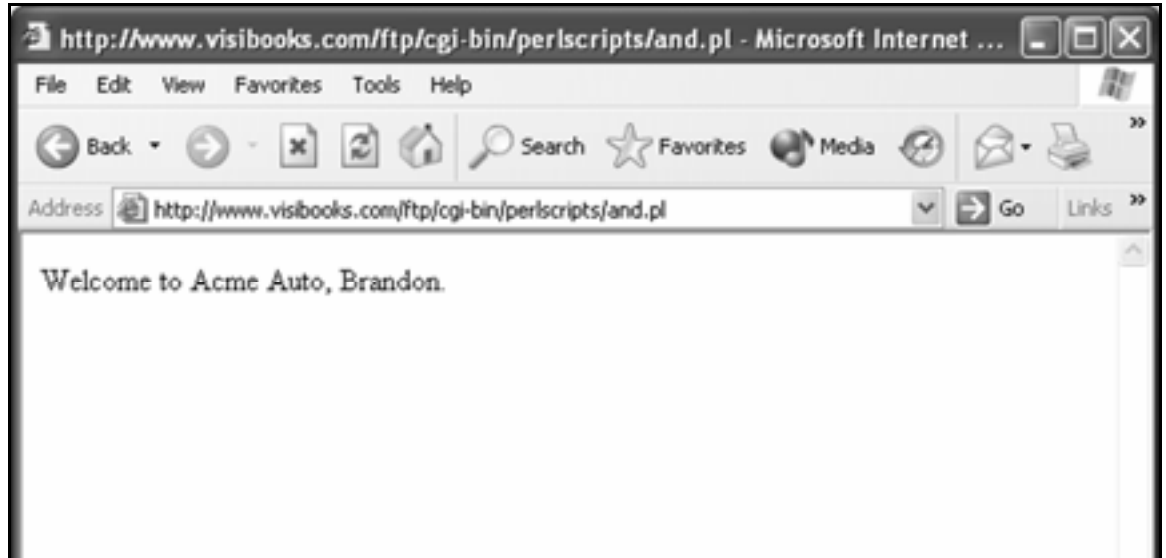
then click the  button.

The output should look like this:



9. View **and.html** in the browser, enter **Brandon** as the User Name and **acme** as the Password, then click the  button.

The output should look like this:



# Employ looping

## Print a list of elements

1. Create a new script with this code:

```
#!/usr/bin/perl
print "Content-Type: text/html \n\n";

# This script demonstrates how to print
# an array using a foreach loop.

@AcmeCars = ("Ford", "Dodge", "Chevy");

print "<p>The text array contains:</p>";

foreach $thisCar (@AcmeCars){
    print "$thisCar<br>\n";
}
```

2. Save the script as **printlist.pl** in the **PERLSSCRIPTS** folder.

Here's what the relevant lines in the script do:

- `@AcmeCars = ("Ford", "Dodge", "Chevy");`

Creates the array variable `@AcmeCars` and places the “Ford”, “Dodge”, and “Chevy” values into the array.



- `foreach $thisCar (@AcmeCars) {`

`foreach` tells the Web server to “loop” through the `@AcmeCars` array, going through each value in the array, one by one.

`$thisCar` is a scalar variable: it tells the Web server to pull out each separate element in the `@AcmeCars` array.

- 3.** Upload `printlist.pl` to the `perlscripts` directory in your Web site, then set its permissions so that anyone can execute it.

- 4.** Open `perllinks.html` and insert a new link to `printlist.pl`:

```
<p><a  
href="http://www.yourwebsite.com/cgi-  
bin/perlscripts/printlist.pl">14. Print a list  
of elements using a loop.</a></p>
```

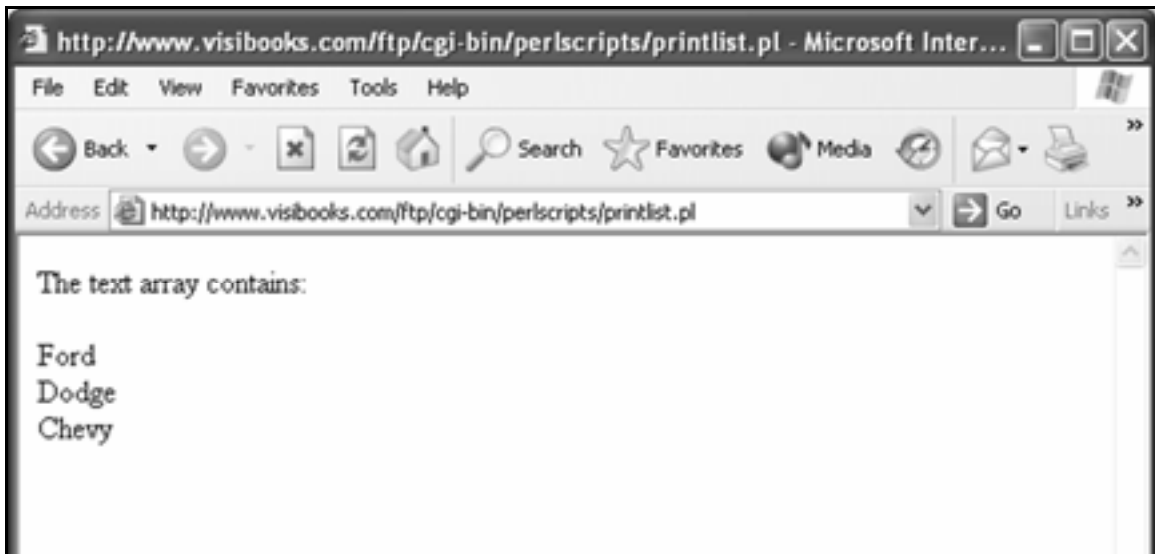
- 5.** Save `perllinks.html`, then upload it to the home directory in your Web site.

- 6.** In the browser, go to:

**`www.yourwebsite.com/perllinks.html`**

**7.** Click the **Print a list of elements using a loop** link.

The output should look like this:



# Print elements in an HTML table

1. Create a new script with this code:

```
#!/usr/bin/perl
print "Content-Type: text/html \n\n";

# This script demonstrates how to print
# array elements within a table.

@AcmeCars = ("Ford", "Dodge", "Chevy");

print "<html><head><title>Table
Example</title></head><body>\n";

print "<table border=1 bgcolor=yellow>\n";

foreach $thisCar (@AcmeCars){
    print "<tr><td>$thisCar</td></tr>\n";
}

print "</table></body></html>";
```

## 2. Save the script as **tablelist.pl** in the **PERLSCRIPTS** folder.

Here's what the relevant lines in this script do:

- `print "<html><head><title>Table Example</title></head><body>\n";`

Prints the HTML tags that begin a Web page.

- `print "<table border=1 bgcolor=yellow>\n";`

Prints the tags that set up an HTML table.

- `foreach $thisCar (@AcmeCars) {`

This line tells the Web server to pull out each separate element (`$thisCar`) in the `@AcmeCars` array, then start doing something to `$thisCar`.

- `print "<tr><td>$thisCar</td></tr>\n";`

Tells the Web server to create a table cell for each element, with the elements inside the cells.

- `}`

Tells the Web server to stop doing things with each element in the `@AcmeCars` array.

- `print "</table></body></html>";`

Prints HTML tags that complete the table and Web page.

## 3. Upload **tablelist.pl** to the **perlscripts** directory in your Web site, then set its permissions so that anyone can execute it.

- 4.** Open **perllinks.html** and insert a new link to **tablelist.pl**:

```
<p><a  
href="http://www.yourwebsite.com/cgi-  
bin/perlscripts/tablelist.pl">15. Print a list  
in a table.</a></p>
```

- 5.** Save **perllinks.html**, then upload it to the home directory in your Web site.

- 6.** In the browser, go to:

**[www.yourwebsite.com/perllinks.html](http://www.yourwebsite.com/perllinks.html)**

- 7.** Click the **Print a list in a table** link.

The output should look like this:





# Working With Files

In this section, you'll learn how to:

- **Create a text file**
- **Display files**
- **Append to files**

# Create a text file

1. Create a new Web page with this code:

```
<html>
<head>
<title>Create Text File</title>
</head>

<body>

<h2>Today's Thought</h2>

<form name="thought"
action="http://www.yourwebsite.com/cgi-
bin/perlscripts/textwriter.pl" method="post">

<input type="hidden" name="filename"
value="textthought.txt">

<textarea name="comments" rows=3 cols=50
wrap></textarea>

<p><input type="submit" value="Create Thought"
name="submit">

</form>

</body>
</html>
```

2. Save the page as **textwriter.html** in the **PERLSCRIPTS** folder, then upload it to the home directory in your Web site.



### 3. Create a new script with this code:

```
#!/usr/bin/perl
print "Content-type: text/html\n\n";

&getFormData;

$mycomments = $request{"comments"};
$myfile      = $request{"filename"};

open(MYFILE, ">$myfile");
print MYFILE "$mycomments";
close(MYFILE);

print "<p>The $myfile file is created with the
following thought:</p>";

print "<p>$mycomments</p>";
print "<p>$myfile</p>";

print "<a
href=\"http://www.yourwebsite.com/textwriter.ht
ml\">Enter a new thought</a><br>\n";

print "<a href=\"$myfile\">View the $myfile
text file</a>\n";

sub getFormData {

read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});
@pairs = split(/&/, $buffer);
foreach $pair (@pairs) {
    ($name, $value) = split(/=/, $pair);
    $value =~ tr/+//;
    $value =~ s/%([a-fA-F0-9][a-fA-F0-
9])/pack("C", hex($1))/eg;
    $value =~ s/\n//g;
    $request{$name} = $value;
}
}
```

}

#### 4. Save the file as **textwriter.pl** in the **PERLSCRIPTS** folder.

Here's what the relevant lines in this script do:

- `$mycomments = $request{"comments"};`

Requests the text entered in the text area named `comments` in **textwriter.html**, then assigns it to the variable **\$mycomments**.

- `$myfile = $request{"filename"};`

The form in **textwriter.html** has a hidden text field named **filename**. Here the script requests the value assigned to it in the form—**textthought.txt**—then assigns that value to the variable **\$myfile**.

- `open (MYFILE, ">$myfile");`

The PERL command **open** opens the **MYFILE** file variable. Then, the `>` sign tells the Web server that the value of **\$myfile**—**textthought.txt**—can be overwritten.

If **textthought.txt** does not exist, the Web server will create the file. If it does exist, then the old file will be completely overwritten with the new data.

**Tip:** *When using the `open ()` command, there are three ways a file can be opened:*

- > *Overwrites an existing file—all previous data is lost*
- >> *Appends data to the end of an existing file*
- < *Used for reading data as an input file*

- `print MYFILE "$mycomments";`

Puts, or “prints,” the text associated with the `$mycomments` variable (the text entered in the `comments` textbox in the form) into the file assigned to the `MYFILE` file variable—`textthought.txt`.

Since `textthought.txt` doesn't exist yet, a new text file called `textthought.txt` is created to hold the text coming in from `$mycomments`.


- `close (MYFILE) ;`

The `close` command tells the Web server that you're done using the `MYFILE` file variable.

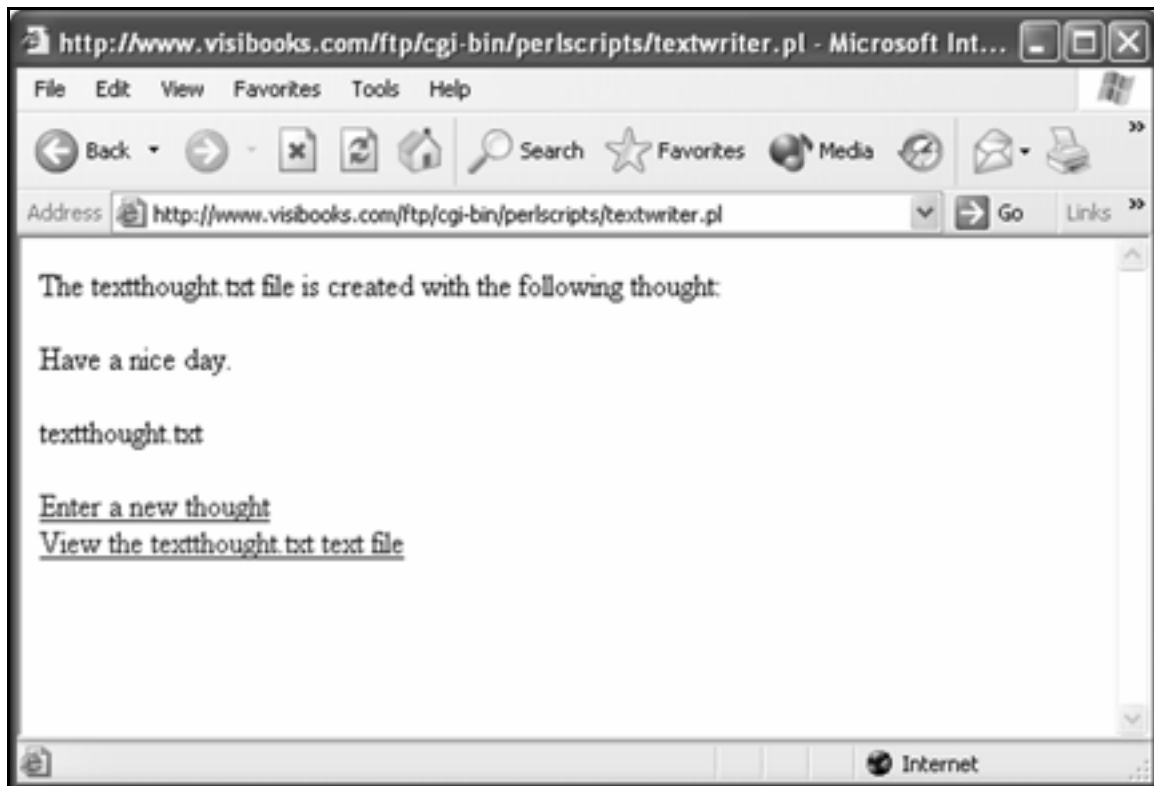
**5.** Upload `textwriter.pl` to the `perlscripts` directory in your Web site, then set its permissions so that anyone can execute it.

**6.** In the browser, go to:


**`www.yourwebsite.com/textwriter.html`**

7. Enter some text in the comments area, then click the  button.

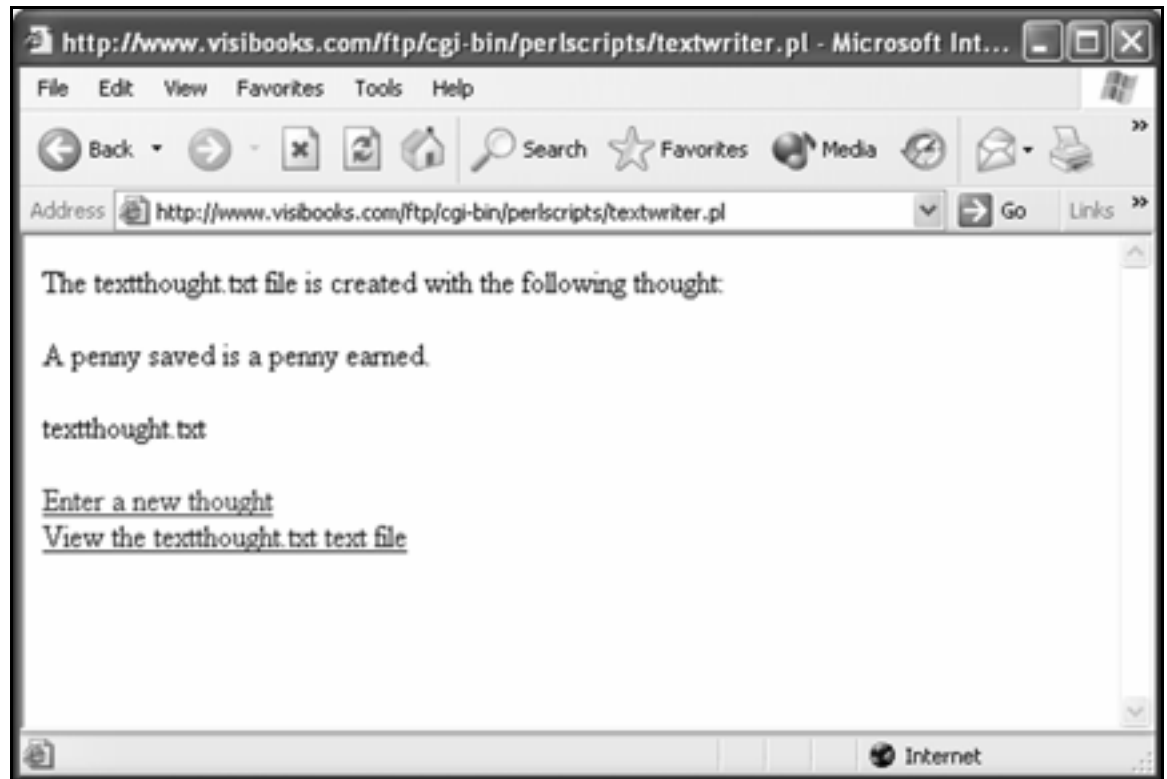
The output should look something like this:



**8.** Return to the **textwriter.html** page.

Enter different text in the comments area, then click the  button again.

The output should be different.



# Display files

1. Create a new script with this code:

```
#!/usr/bin/perl
print "Content-type: text/html\n\n";

open (THISFILE,"textthought.txt");

foreach $line(<THISFILE>) {
print "$line<br>\n";
}

close (THISFILE);
```

2. Save the file as **textviewer.pl** in the **PERLSCRIPTS** folder, upload it to the **perlscripts** directory in your Web site, then set its permissions so that anyone can execute it.

Here's what the relevant lines in this script do:

- `open (THISFILE,"textthought.txt")`

Opens the file **textthought.txt**, and assigns the text in it to the file variable **THISFILE**.

- `foreach $line (<THISFILE>) {`

The `foreach` loop loops through every line in the **THISFILE** variable. In other words, it goes through each line of the **textthought.txt** file.

At each line, it assigns the value of that line to the scalar variable `$line`.

- `print "$line<br>\n";`

Prints the text in each line of **textthought.txt** as the script loops through them.

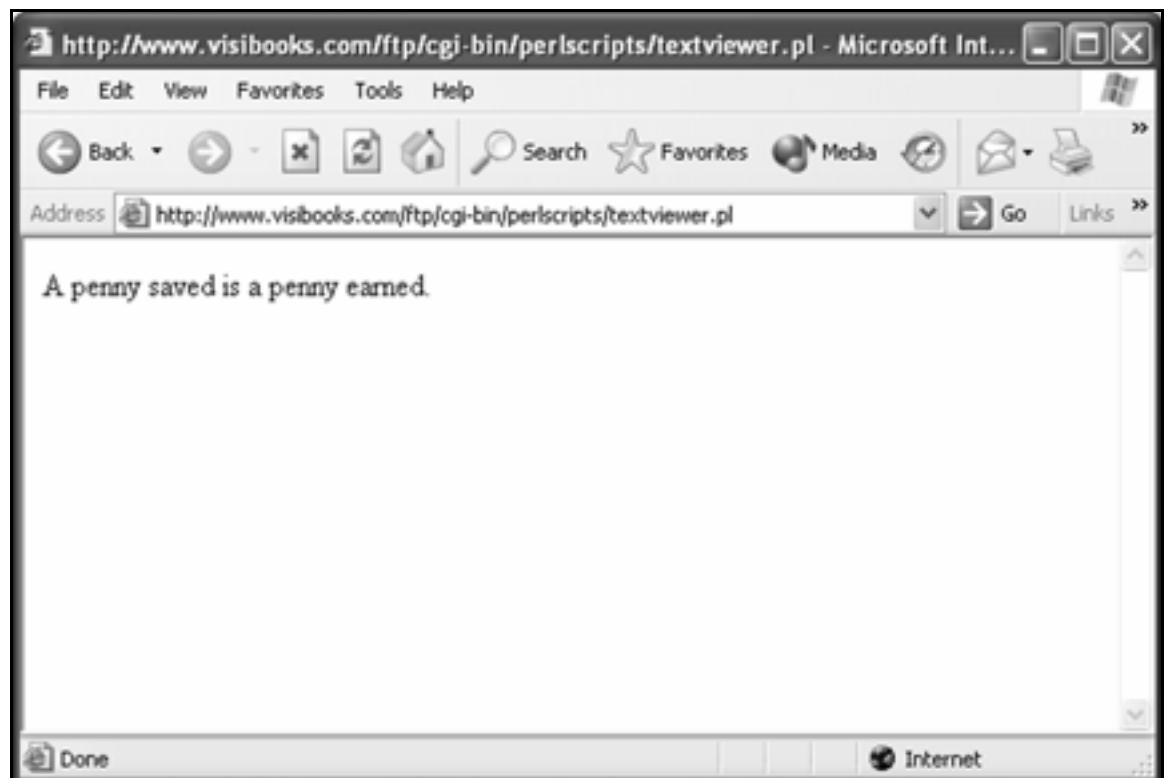
```
close (THISFILE);
```

Once the `foreach` loop is finished, the file variable that stands for **textthought.txt** file is closed.

**3.** In the browser, go to:

**[www.yourwebsite.com/cgi-bin/perlscripts/textviewer.pl](http://www.yourwebsite.com/cgi-bin/perlscripts/textviewer.pl)**

The output should look something like this:



# Append to files

1. Create a new Web page with this code:

```
<html>
<head>
<title>Append to files</title>
</head>

<body>

<h2>Add to Today's Thought</h2>

<form name="thought"
action="http://www.yourwebsite.com/cgi-
bin/perlscripts/textappender.pl" method="post">

<input type="hidden" name="filename"
value="textthought.txt">

<textarea name="comments" rows=3 cols=50
wrap></textarea>

<br><input type="submit" value="Update
Thought">

</form>

</body>
</html>
```

2. Save the page as **textappender.html** in the **PERLSCRIPTS** folder, then upload it to the home directory in your Web site.



**3.** Open `textwriter.pl` and replace this code:

```
open (MYFILE, ">$myfile");
print MYFILE "$mycomments";
close (MYFILE);

print "<p>The $myfile file is created with the
following thought:</p>";

print " p>$mycomments</p>";
print "<p>$myfile</p>";

print "<a href=\"textwriter.html\">Enter a new
thought</a><br>\n";

print "<a href=\"$myfile\">View the $myfile
text file</a>\n";
```

with this code:

```
open (MYFILE, ">>$myfile");

print MYFILE "$mycomments";

close (MYFILE);

print "<br>The $myfile file is updated with the
following text: <p>$mycomments</p>";
```

4. Save the file as **textappender.pl** in the **PERLSCRIPTS** folder, then upload it to the **perlscripts** directory in your Web site.

Set its permissions so that anyone can execute it.

Here's what the relevant lines in this script do:

- 
- `open (MYFILE, ">>$myfile") ;`

Opens the the **MYFILE** variable (**textthought.txt**), then tells the Web server, with the >> sign, to add comments entered in **textappender.html** to **textthought.txt**.

- `print MYFILE "$mycomments" ;`

Puts, or “prints,” the text associated with the **\$mycomments** variable (the text entered in the **comments** textbox in the form) into the file assigned to the **MYFILE** file variable—**textthought.txt**.

However, because **MYFILE** was opened for append (>>), the old text remains in **textthought.txt**, and the new text is added to the end of it.

- `close (MYFILE) ;`

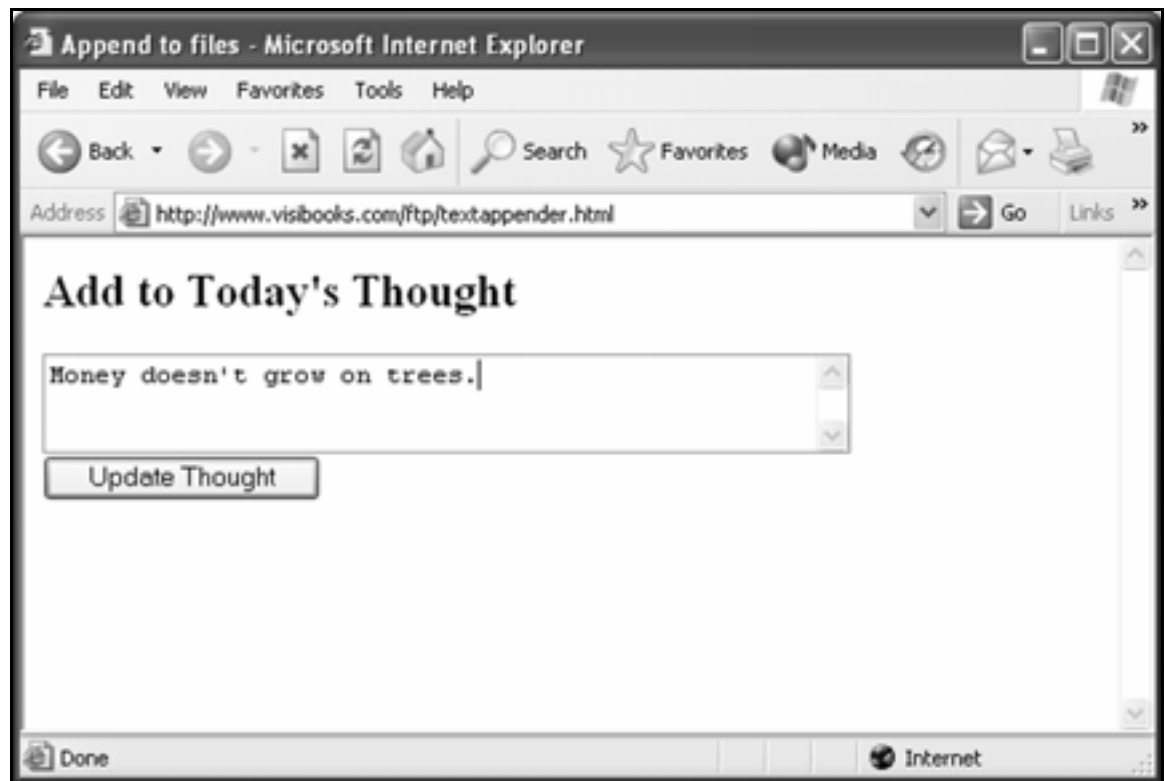
After **MYFILE** has been appended, it's closed.

5. In the browser, go to:

**[www.yourwebsite.com/textappender.html](http://www.yourwebsite.com/textappender.html)**

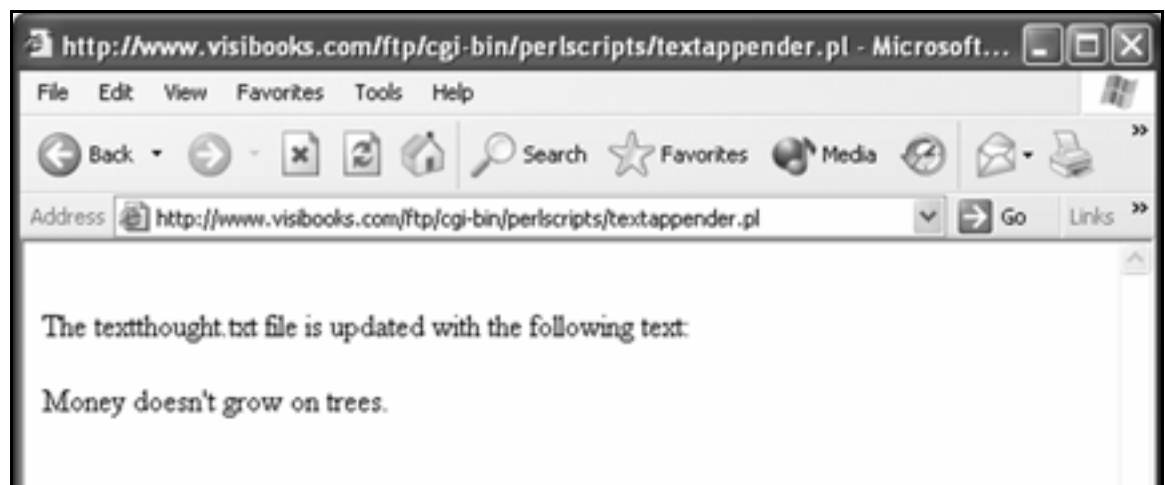
**6.** In its comments box, type:

**Money doesn't grow on trees.**



**7.** Click the  button.

The output should look like this:





# Modifying scripts

## Modifying downloaded scripts

Just about any basic script you'd need has already been written by someone else. Do a Web search on "perl scripts" and you'll find many sites with good, useful scripts. Most are either free or inexpensive to use.

Now that you're familiar with PERL, you can download an existing script that generally does what you want, then modify it to meet your specific needs.

Below is an example of a script downloaded from a Web site. It takes form data from a Web page and sends it to an email address. Modify a few sections, and you can use it on your Web site.

```
#!/usr/bin/perl -w
```

```
read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});  
@pairs = split(/&/, $buffer);  
foreach $pair (@pairs) {  
    ($name, $value) = split(/=/, $pair);  
    $value =~ tr/+//;  
    $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C",  
hex($1))/eg;  
    $value =~ s/\n//g;  
    $request{$name} = $value;  
}
```

```
$myemail = "you\@yourserver.com";
```

Parsing code.  
Takes text from form  
inputs and puts it in  
a format PERL can  
work with.

Email address  
to which you'd  
like the form  
input sent.

```
$maillocation = "/usr/sbin/sendmail";
```

```
$name="$request{'name'}" ;  
$email="$request{'email'}" ;
```

Script requests text from two form inputs on a Web page. One input is named "name," the other is named "email."

Location of the email program on your Web server.

If the "name" input is left blank, print the HTML text below.

```
if ($name eq "") {  
  print "Content-type: text/html\n\n";  
  print "<HTML>  
  <HEAD>  
  <TITLE>Enter name</TITLE>  
  </HEAD>  
  <BODY>Please enter your name.</BODY>  
  </HTML>\n";  
  exit;  
  }
```

If the "email" input is left blank, print the HTML text below.

```
if ($email eq "") {  
  print "Content-type: text/html\n\n";  
  print "<HTML>  
  <HEAD>  
  <TITLE>Enter email</TITLE>  
  </HEAD>  
  <BODY>Please enter your email address.</BODY>  
  </HTML>\n";  
  exit;  
  }
```

If both inputs are filled in, print the "Thanks!" message.

```
print "Content-type: text/html\n\n";  
print "<HTML>  
<HEAD>  
<TITLE>Thanks for subscribing</TITLE>  
</HEAD>  
<BODY>Thanks!</BODY>
```

```
</HTML>\n";
```

```
open (MAIL, "| $maillocation") || die "aw, cant  
use $maillocation";
```

```
print MAIL "To: $myemail\n";
```

```
print MAIL "From: $email\n";
```

```
print MAIL "Subject: Info from form\n";
```

```
print MAIL "\n";
```

```
print MAIL "Here's the info:\n\n";
```

```
print MAIL "Name: $name\n";
```

```
print MAIL "Email: $email\n";
```

```
close (MAIL);
```

Open the email program at \$maillocation on the server.

After the email program is open, have it print an email with the form date, then send it to \$myemail.





# Where To Get In Pictures Books

If you liked this book, and would like to buy more like it, visit:

**[www.inpics.net](http://www.inpics.net)**

In Pictures offers more than 20 titles on subjects such as:

- **Computer Basics**
- **Microsoft Office**
- **Desktop Linux**
- **OpenOffice.org**
- **Web Site Layout**
- **Web Graphics**
- **Web Programming**

In Pictures: computer books based on pictures, not text.



[www.inpics.net](http://www.inpics.net)