INSERT PHP CODE INTO A WEB PAGE

Adding PHP code to an HTML document allows you to create dynamic, interactive Web pages. You can add PHP code to an existing Web page or to a new HTML document you create.

PHP code is inserted into HTML code using the `<?php` opening delimiter and the `?>` closing delimiter. The delimiters tell the Web server where the PHP code begins and ends. You can add multiple sections of PHP code to an HTML document. You must use an opening and closing delimiter for each section of code.

If the PHP code will generate output for display in a user's Web browser, you can use HTML tags to format the output. PHP code used to generate output must be inserted between the `<body>` and `</body>` tags.

When a Web server receives a request for a Web page containing PHP code, the Web server processes all the code found between the `<?php` and `?>` delimiters. The information generated by the PHP code is inserted into the Web page before the page is sent to a user's Web browser. Users who visit the page will not be able to see the PHP code, even if they display the source code for the page.

When saving a PHP page you have created, you must add the .php extension to the filename of the page. Some text editors do not recognize the .php extension, so you may have to enclose the filename in quotation marks, such as "index.php". The PHP page must be saved in the appropriate location on your Web server before the page can be displayed.

If you do not want to use the .php extension for your PHP pages, you can configure your Web server to use another extension, such as .phtml, .htm or .html. For more information, see page 10.

Extra

In addition to the `<?php` and `?>` delimiters, there are other delimiters you can use to insert PHP code into a Web page. If you have the `short_open_tag` setting enabled in your PHP configuration file, you can use the `<?` and `?>` delimiters. You can also use the ASP-style delimiters `<%` and `%>`. To use ASP-style delimiters, you need to enable the `asp_tags` setting in the PHP configuration file.

You can use the `<script>` tag to insert PHP code into a Web page. Using the `<script>` tag does not require any changes to the settings in the PHP configuration file.

Example:

```
<script language="php">
print "Thank you. Please visit again.";
</script>
```

Instead of the `print` function, you can use the `echo` command to generate output.

Example:

```
echo "Thank you. Please visit again.";
```

You can use the `<?=` and `?>` delimiters to insert a single `print` statement into an HTML document. Using the `<?=` and `?>` delimiters reduces the amount of code you have to type and can make your script easier to read.

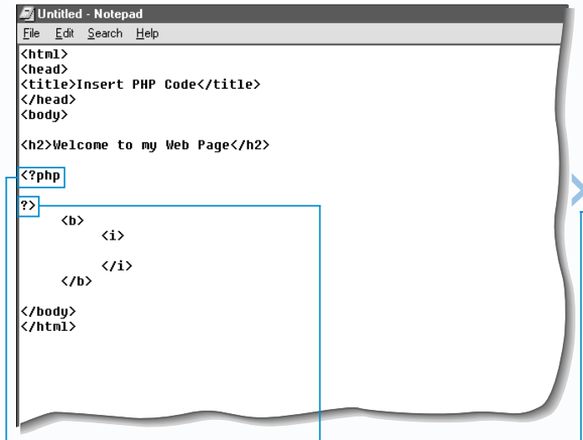
Example:

```
<?php
print $message;
?>
```

Can be typed as:

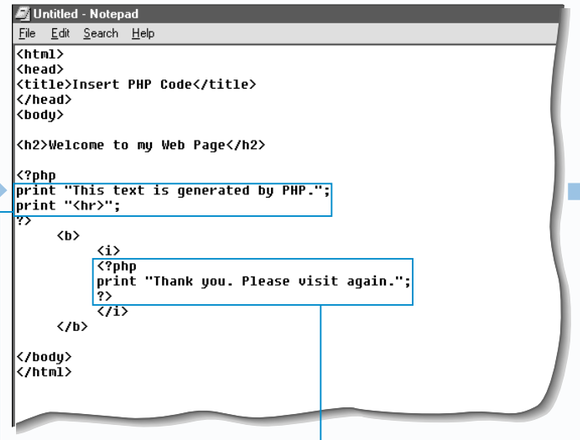
```
<?= $message ?>
```

INSERT PHP CODE INTO A WEB PAGE



1 Type **<?php** where you want to insert PHP code into a Web page.

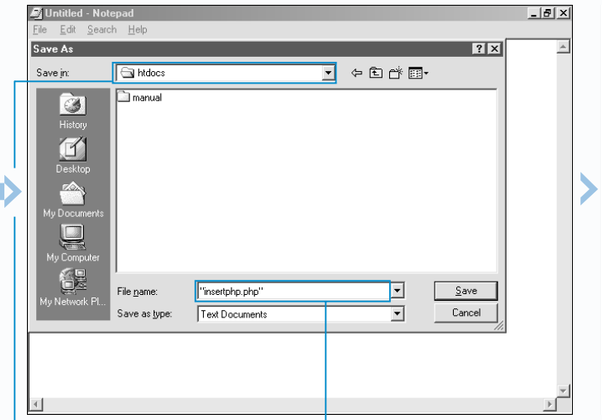
2 Type **?>** where you want to end the PHP code.



3 Between the opening and closing delimiters (`<?php` and `?>`), type the PHP code you want to insert into the Web page.

■ In this example, we use the `print` function to generate output.

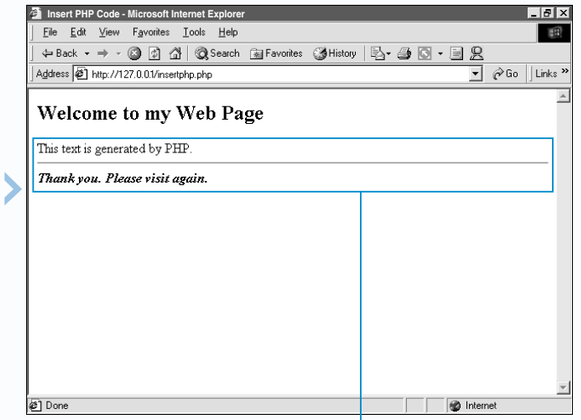
4 Repeat steps 1 to 3 for each section of PHP code you want to insert into the Web page.



SAVE AND DISPLAY A PHP PAGE

5 Save the page in the appropriate location on your Web server.

■ The filename must end with the .php extension. You may have to enclose the filename in quotation marks.



6 Display the PHP page in a Web browser.

■ The Web browser displays the result of inserting PHP code into a Web page.

ADD A COMMENT TO A PHP PAGE

Adding comments to your PHP code is a good programming practice and can help explain important or difficult sections of code. Web servers ignore comments in PHP code, so adding comments to a PHP page will not slow down the processing of the page.

Even relatively simple code may be difficult to understand without comments. For example, you may use a variable named $totalCost in your code. You could use a comment to explain whether the variable stores the total cost of all the products or only some of the products.

Besides describing the purpose of code, comments can contain information such as the name of the author and the author's contact information.

A single-line comment is preceded by // and can be included at the end of a line of code. A single-line comment can also have its own line in a PHP page. All the information following // to the end of the line will be ignored by the Web server.

To add a comment that spans multiple lines, use /* before the first line and */ at the end of the last line. While comments can be any size, very lengthy comments may make your code difficult to read. The most effective comments are descriptive and concise.

In addition to creating multi-line comments, you can use /* and */ to help troubleshoot a PHP page. For example, if your PHP code is generating an error, you can use /* and */ to comment out the suspect lines. If you confirm that the error lies somewhere else in the PHP code, you can simple remove /* and */ to restore the lines of code. Commenting out the lines saves you from having to erase the lines and type them again later. This technique is very effective for debugging large, complex sections of code. For more information about troubleshooting PHP code, see pages 208 to 219.

Extra

Instead of //, you can use the hash symbol (#) to indicate the start of a single-line comment. Both styles of adding comments are acceptable, but you should choose one style and use it consistently throughout your code.

Example:

```
<?php
// Display a welcome message
print "Welcome to my Web page.<br>";
# Display my name
print "My name is Martine.";
?>
```

When creating multi-line comments, you should be careful not to place one comment inside another comment. Nested comments will generate an error when the PHP page is processed.

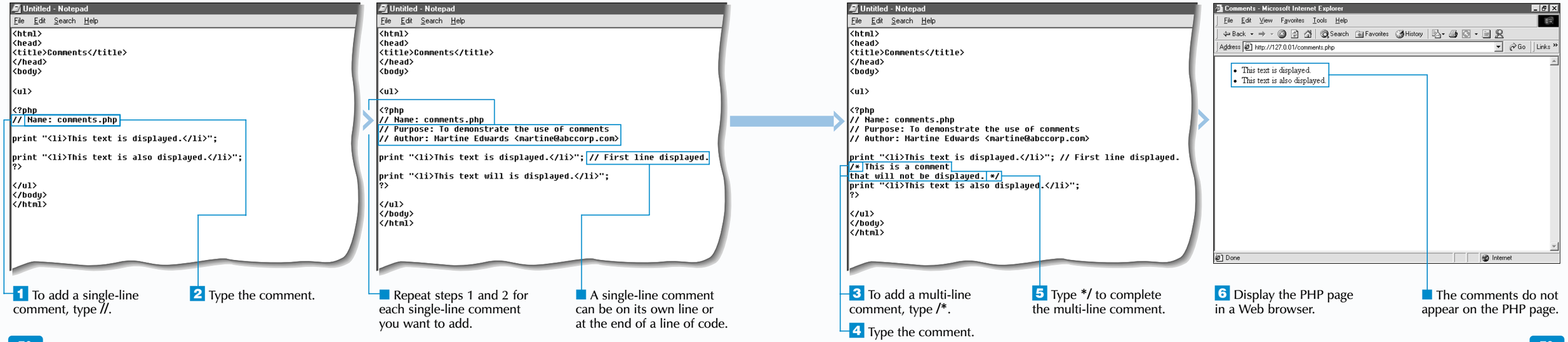
TYPE THIS:

```
<?php
/* Name: comments.php
   Purpose: To show how not to use delimiters
   Author: Martine Edwards <martine@abccorp.com>

/* Display a welcome message */
print "Welcome to my Web page.";

PHP will try to process these lines of comments
and generate an error */
?>
```

RESULT:

Parse error: parse error in **c:\web_apps\apache\htdocs\comments.php** on line **16**

ADD A COMMENT TO A PHP PAGE



1 To add a single-line comment, type //.

2 Type the comment.

■ Repeat steps 1 and 2 for each single-line comment you want to add.

■ A single-line comment can be on its own line or at the end of a line of code.

3 To add a multi-line comment, type /*.

4 Type the comment.

5 Type */ to complete the multi-line comment.

6 Display the PHP page in a Web browser.

■ The comments do not appear on the PHP page.

CREATE A NUMERIC VARIABLE

A variable is a name that represents a value. For example, you could have the variable $myAge represent the value 29. A numeric variable is often used to store a value that may change, such as the result of a calculation or data retrieved from a database.

To create a numeric variable, you use the assignment operator (=) to assign an initial value to a variable name. A numeric variable can store an integer, such as 1 or 3452, or a floating-point number, such as 9.3 or 0.0054. A numeric variable can also store a negative value or no value at all. To specify a negative value, you simply place a minus sign (-) directly in front of a number. A numeric variable with no value has a null value, which is not the same as a value of zero (0).

In PHP, a variable name must start with a dollar sign ($). The next character must be a letter or the underscore character (_) followed by any combination of letters and numbers.

A variable name can consist of multiple words. You can use a lowercase letter to begin the first word and then capitalize the first letter of each of the following words to make the name easy to read, such as $myDateOfBirth. The underscore character (_) can also be used to separate the words in a name, such as $my_age. You should choose one format and use it consistently to make your script easier for other people to understand.

Variable names are case sensitive. For example, the variable $myAge is different than the variable $MYAGE.

Several sections of PHP code can be inserted into one HTML document. The value of a variable created in one section of PHP code will be accessible in subsequent sections of the code.

Extra

Unlike other languages, such as C++ and Java, PHP is not a strongly typed language. In PHP, you do not need to explicitly declare the type of variable you want to create. For example, the same variable can be used to represent first an integer and then a floating-point number with no data type declarations.

Example:

```
$a = 1;
$a = $a + 1.5;
```

You can use *exponential notation* to specify a very large or very small number as the value of a variable. In this following example, $a is equal to 123000 and $b is equal to 0.0000123.

Example:

```
$a = 1.23e5;
$b = 1.23e-5;
```

When working with numbers, PHP assumes you are working in base 10, or *decimal notation*. PHP also supports base 8, or *octal notation*, and base 16, or *hexadecimal notation*. To specify a value in octal notation, you place a zero (0) before an octal number. To specify a value in hexadecimal notation, you place a zero (0) and the letter x before a hexadecimal number. When you display an octal or hexadecimal number, the result is shown in decimal notation.

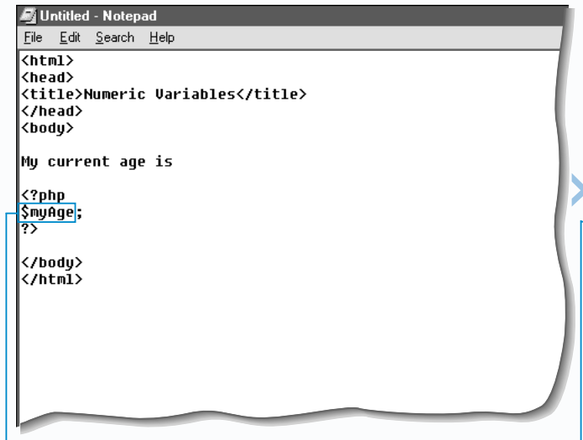
Example:

```
$a = 020;
print $a;
print "<br>";
$b = 0x10;
print $b;
```

Result:

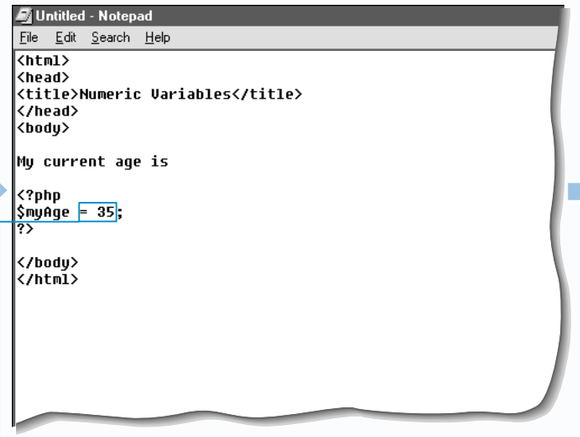
```
16
16
```

CREATE A NUMERIC VARIABLE



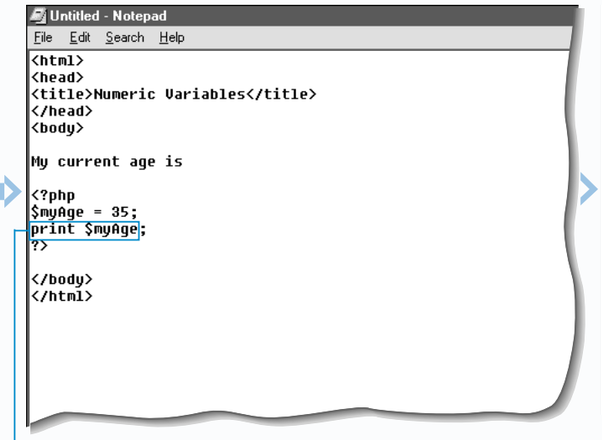
1 To create a numeric variable, type **$** followed by a name for the variable.

- The dollar sign ($) must be followed by a letter or an underscore character (_).



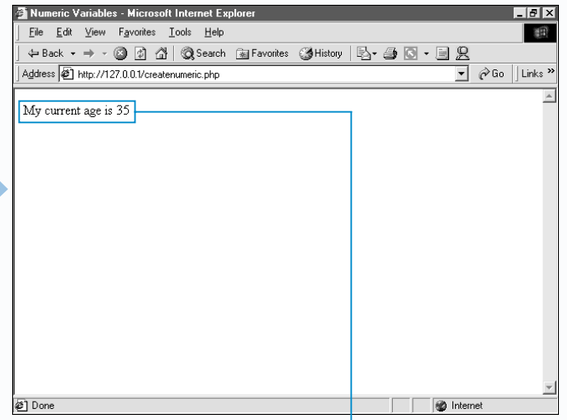
2 Type **=** followed by the value you want to assign to the variable.

- The value can be an integer or a floating-point number.



3 Type the code that uses the numeric variable you created.

- In this example, we use the `print` function to display the value of the variable.



4 Display the PHP page in a Web browser.

- The Web browser displays the value of the numeric variable.

CREATE A STRING VARIABLE

You can create a variable that stores a string value. A string is a collection of characters that are enclosed in single or double quotation marks. The information in a string is treated as a *scalar value*, or a single piece of information. The quotation marks identify the beginning and end of a string and allow PHP to work with the string as one piece of information.

To create a string variable, you use the assignment operator (=) to assign a string value to a variable name. A variable name must begin with a dollar sign ($) followed by a letter or the underscore character (_). Variable names can consist of multiple words and are case sensitive.

Escape sequences allow you to include special characters, such as tabs, newlines and carriage returns in a string and are often used to format text that will be displayed on a screen or stored in a file. When a string you assign to a variable is enclosed in double quotation marks, PHP will evaluate any escape sequences and variables included in the string. For example, a variable in a double-quoted string will be replaced with the value assigned to the variable when the string is processed.

When including a character required by the PHP programming language in a double-quoted string, you must precede the character with a backslash if you do not want the character to be evaluated. For example, when a dollar sign ($) is placed in a string, PHP will evaluate the dollar sign as the beginning of a variable name unless you use a backslash to escape the symbol.

When a string is enclosed in single quotation marks, PHP does not evaluate variables or escape sequences in the string, except for the \\ and \' escape sequences. You can use the \\ escape sequence to display a backslash and the \' escape sequence to display a single quotation mark in a single-quoted string.

Extra

The following is a list of several common escape sequences that can be enclosed in double-quoted strings.

ESCAPE SEQUENCE:	DESCRIPTION:
\n	Start a new line.
\r	Insert a carriage return.
\t	Insert a tab.
\\	Insert a backslash.
\"	Insert a double quotation mark.
\$	Insert a dollar sign.
\012	Use an octal ASCII value (example: 12).
\x0A	Use a hexadecimal ASCII value (example: 0A).

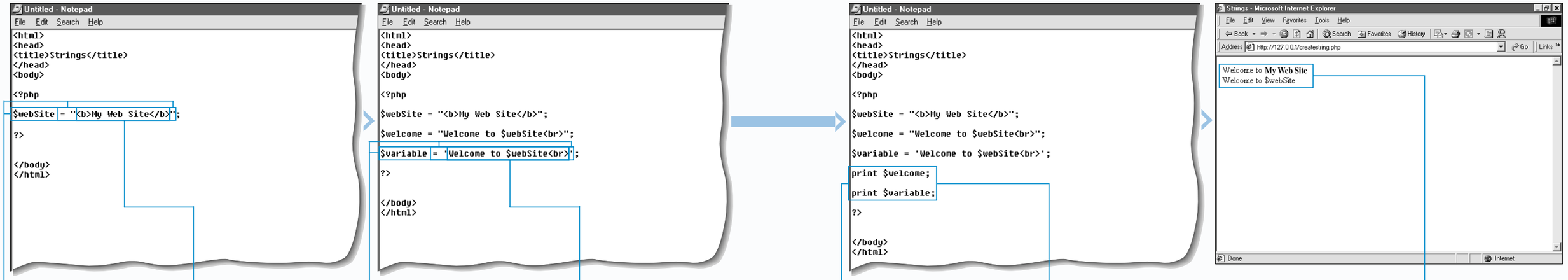
Strings can be joined together using the concatenation operator (.). The concatenation operator can be used to join two strings, such as `"Tom" . "Smith"`, or a combination of strings and variables, such as `$myString = "I am " . $myAge . " years old"`.

PHP is often used to output HTML code. When assigning an HTML tag, such as `<IMG SRC="my_image.gif" WIDTH="100" HEIGHT="100">` to a string variable, you must use escape sequences to include any special characters, such as quotation marks, within the tag. The HTML code within the string will be processed by the Web server and applied to the output for a client's Web browser.

Example:

```
$myImageTag = "<IMG SRC=\"my_image.gif\" WIDTH=\"100\" HEIGHT=\"100\">";
```

CREATE A STRING VARIABLE



1 To create a string variable using double quotation marks, type **$** followed by a name for the variable.

2 Type **= ""**.

3 Between the quotation marks, type the information you want the string to contain.

■ PHP will evaluate any variables and escape sequences you include in the string.

4 To create a string variable using single quotation marks, type **$** followed by a name for the variable.

5 Type **= ''**.

6 Between the quotation marks, type the information you want the string to contain.

■ PHP will not evaluate information you include in the string.

7 Type the code that uses the string variables you created.

■ In this example, we use the `print` function to display the values of variables.

8 Display the PHP page in a Web browser.

■ The Web browser displays the values of string variables.

CREATE A CONSTANT

A constant is a name that represents a value you want to remain the same throughout a PHP script. Regardless of where a constant is created in a PHP script, the constant can be accessed anywhere in the script.

You use the `define` function to create a constant in your script. You must first specify the name of the constant you want to create. Constant names must start with a letter or an underscore (_) character. To help distinguish constants from variables, you should use all uppercase letters in a constant name. The name of the constant must be enclosed in quotation marks when specified in the `define` function.

After naming a constant, you must specify the value you want to assign to the constant. A constant can store any string, integer or floating point value.

Once a constant has been created, it can be used within calculations in the same way you use a variable. Attempting to change the value of a constant when using the constant will generate an error.

Unlike variables, constants do not start with a dollar symbol, so PHP cannot determine when a constant is being used in a string. For example, in the string "This page was created by AUTHOR", PHP will interpret the constant AUTHOR not as a value, but as the word AUTHOR. You must separate a constant you are using from the rest of the string to allow the constant to be evaluated correctly.

PHP also allows you to access predefined constants from within PHP code. For information about predefined constants, see page 80.

Extra

It is good programming practice to define all your constants at the beginning of your PHP script. Keeping all your constants in the same location makes it easy to find and modify the constants.

In addition to a string or number, you can use an expression as the value for a constant. This is useful when you want to display a different value for the constant depending on the value of a variable. For example, you can create a constant that stores a personalized greeting based on the user name of the person visiting your Web site.

Example:

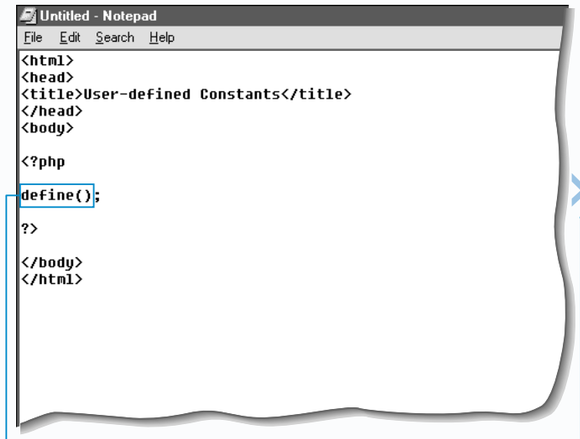
```
<?php
$userName = Lindsay;
define("GREETING", "Hello " . $userName);
print GREETING;
?>
```

You can use the `defined` function to verify whether a constant exists in your script. If the constant exists, the `defined` function will return a value of `1`. If the constant does not exist, the `defined` function will return a value of `0`. Using the `defined` function is useful for quickly determining if a constant exists in a long script that uses a large number of constants.

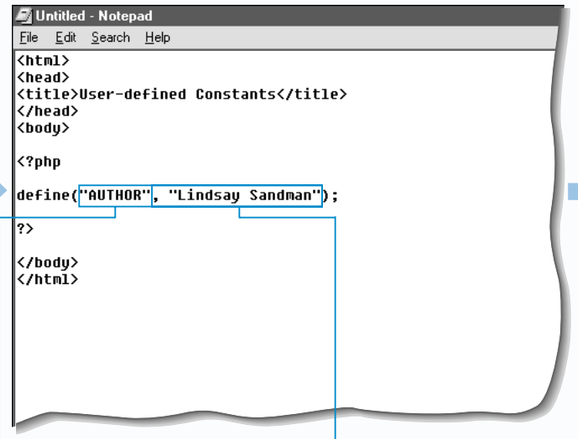
Example:

```
<?php
if (defined("AUTHOR"))
{
      print "Page created by: ";
      print AUTHOR;
}
else
{
      print "Author unknown";
}
?>
```

CREATE A CONSTANT



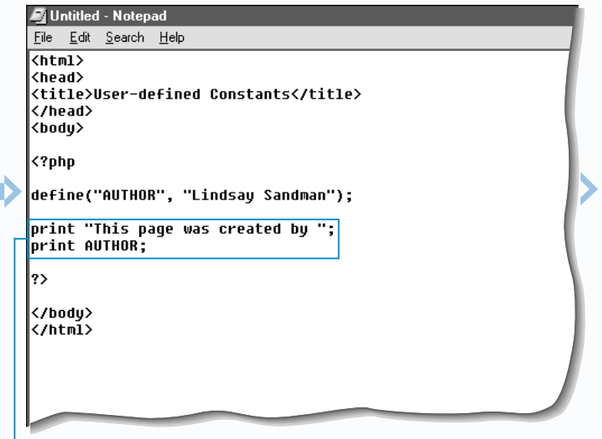
1 Type **define()** where you want to create a constant.



2 Between the parentheses, type the name of the constant you want to create, enclosed in quotation marks.

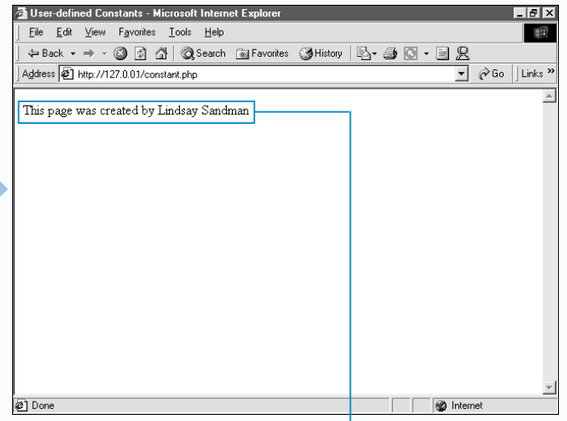
3 Type a comma followed by the value you want to assign to the constant.

- If the value for the constant is a string, you must enclose the text in quotation marks.



4 Type the code that uses the value of the constant.

- If you are using the constant with a string, make sure the constant is not enclosed within the quotation marks for the string.



5 Display the PHP page in a Web browser.

- The Web browser displays the result of using a constant.

INTRODUCTION TO OPERATORS

PHP provides numerous operators that can be used to assign values to variables, perform calculations, make comparisons and much more. An operator takes one or more arguments, called operands. A sequence of operands separated by one or more operators that produces a result in a PHP script is referred to as an expression.

Associativity

When an expression contains multiple operators that have the same precedence, the associativity of the operators determines which part of the expression PHP evaluates first. Operators can have left associativity, right associativity or can be non-associative.

If operators have left associativity, then the leftmost operator is processed first. For example, the result of the expression 5 - 3 + 2 is 4 rather than 0. The opposite holds true for operators that have right associativity.

Some operators have neither left nor right associativity and are known as non-associative. When a non-associative operator is used beside others with the same precedence, you must surround the operator and its operands with parentheses to ensure that it will return the desired result.

Order of Precedence

When an expression contains several operators, PHP processes the operators in a specific order, known as the order of precedence. The order of precedence ranks operators from highest to lowest precedence. Operators with higher precedence are evaluated before operators with lower precedence.

Parentheses

Regardless of the precedence and associativity of operators, you can use parentheses to dictate the order in which PHP should process operators. In an expression, PHP processes operators and operands enclosed in parentheses first.

The following table shows the associativity of operators and the order of precedence from highest to lowest.

Associativity	Operators	Associativity	Operators
Non-associative	`new`	Left	`|`
Right	`[`	Left	`&&`
Right	`! ~ ++ -- (int) (double) (string) (array) (object) @`	Left	`||`
		Left	`? :`
Left	`* / %`	Left	`= += -= *= /= .= %= &= |= ^= ~= <<= >>=`
Left	`+ - .`		
Left	`<< >>`	Right	`print`
Non-associative	`< <= > >=`	Left	`and`
Non-associative	`== != === !==`	Left	`xor`
Left	`&`	Left	`or`
Left	`^`	Left	`,`

TYPES OF OPERATORS

ARITHMETIC OPERATORS

You can use arithmetic operators to perform mathematical calculations.

+ (Addition)

Finds the sum of two values.

```
print 3 + 3;
```

Displays 6

* (Multiplication)

Multiplies two values.

```
print 2 * 3;
```

Displays 6

% (Modulus)

Divides one value by another value and returns only the remainder in the result.

```
print 7 % 5;
```

Displays 2

- (Subtraction)

Finds the difference between two values.

```
print 6 - 3;
```

Displays 3

/ (Division)

Divides one value by another value.

```
print 9 / 3;
```

Displays 3

CONCATENATION OPERATOR

You can use the concatenation operator (.) to combine values into a string. This is useful when you want to combine text with a variable that has text value.

. (Concatenation)

```
$temp = "hot";
$myString = "It is " . $temp . " today.";
print $myString;
```

Displays It is hot today.

INCREMENT AND DECREMENT OPERATORS

Increment and decrement operators are useful for creating counters in your script. These operators increase or decrease the value of an expression by one.

++value (Pre-increment)

Adds 1 to a value before the expression that uses the value is processed.

```
$a = 5;
print ++$a;
```

Displays 6

--value (Pre-decrement)

Subtracts 1 from a value before the expression that uses the value is processed.

```
$a = 5;
print --$a;
```

Displays 4

value++ (Post-increment)

Adds 1 to a value after the expression that uses the value is processed.

```
$a = 5;
print $a++;
print "<br>";
print $a;
```

Displays 5
6

value-- (Post-decrement)

Subtracts 1 from a value after the expression that uses the value is processed.

```
$a = 5;
print $a--;
print "<br>";
print $a;
```

Displays 5
4

INTRODUCTION TO OPERATORS

TYPES OF OPERATORS (CONTINUED)

ASSIGNMENT OPERATORS

The assignment operator (=) can be used to set one value to another value. You can also combine the assignment operator with arithmetic operators to more easily perform calculations. For example, the expression $b = $b + 12 can be written as $b += 12. Combining operators reduces the amount of code you need to type and can make your scripts easier to read.

= (Assignment)

Sets one value to another value.

```
$a = 3;
print $a;
```

Displays 3

+=

Adds one value to another value.

```
$a = 3;
$a += 3;
print $a;
```

Displays 6

-=

Subtracts one value from another value.

```
$a = 3;
$a -= 1;
print $a;
```

Displays 2

***=**

Multiplies one value by another value.

```
$a = 3;
$a *= 2;
print $a;
```

Displays 6

/=

Divides one value by another value.

```
$a = 6;
$a /= 2;
print $a;
```

Displays 3

.=

Combines two values.

```
$a = "This is ";
$a .= "a test.";
print $a;
```

Displays This is a test.

LOGICAL OPERATORS

You can use logical operators to check if a statement in your script is true.

And

Checks if two or more statements are true.

```
$x = 7;
$y = 5;
if (($x == 7) and ($y == 5)) print "True";
```

Displays True

Or

Checks if at least one of two statements is true.

```
$x = 7;
$y = 5;
if (($x == 7) or ($y == 8)) print "True";
```

Displays True

Xor

Checks if only one of two statements is true.

```
$x = 7;
$y = 5;
if (($x == 7) xor ($y == 8)) print "True";
```

Displays True

!

Checks if a statement is not true.

```
$y = 5;
if (! ($y == 10)) print "True";
```

Displays True

&&

Checks if two or more statements are both true.

```
$x = 7;
$y = 5;
if (($x == 7) && ($y == 5)) print "True";
```

Displays True

||

Checks if at least one of two statements is true.

```
$x = 7;
$y = 5;
if (($x == 7) || ($y == 5)) print "True";
```

Displays True

BITWISE OPERATORS

Bitwise operators, such as &(AND) and |(OR), are used to compare and manipulate values at the binary level. This can be very useful when dealing with binary files such as image and sound files. You can find more information about bitwise operators at the www.php.net/manual/en/language.operators.bitwise.php Web site.

COMPARISON OPERATORS

You can use comparison operators to compare values in your script.

== (Equal to)

Checks if one value is equal to another value.

```
$x = 7;
$y = "7";
if ($x == $y) print $x . " is equal to " . $y;
```

Displays 7 is equal to 7

=== (Identical to)

Checks if one value is equal to and of the same type as another value.

```
$x = 7;
$y = 7;
if ($x === $y) print $x . " is identical to " . $y;
```

Displays 7 is identical to 7

!= (Not equal to)

Checks if one value is not equal to another value.

```
$x = 8;
$y = 4;
if ($x != $y) print $x . " is not equal to " . $y;
```

Displays 8 is not equal to 4

!== (Not Identical to)

Checks if two values are not equal or not of the same type.

```
$x = 8;
$y = 9;
if ($x !== $y) print $x . " is not identical to " . $y;
```

Displays 8 is not identical to 9

< (Less than)

Checks if one value is less than another value.

```
$x = 5;
$y = 9;
if ($x < $y) print $x . " is less than " . $y;
```

Displays 5 is less than 9

> (Greater than)

Checks if one value is greater than another value.

```
$x = 9;
$y = 5;
if ($x > $y) print $x . " is greater than " . $y;
```

Displays 9 is greater than 5

<= (Less than or equal to)

Checks if one value is less than or equal to another value.

```
$x = 5;
$y = 5;
if ($x <= $y) print $x;
```

Displays 5

>= (Greater than or equal to)

Checks if one value is greater than or equal to another value.

```
$x = 7;
$y = 5;
if ($x >= $y) print $x;
```

Displays 7

PERFORM A NUMERIC CALCULATION

You can perform calculations to manipulate numeric data in a PHP page. Calculations can be performed using literal values, such as 3 or 56.23, as well as variables, such as $quantity or $cost.

Calculations consist of operators and operands. Operators determine the type of calculation that will be performed. Operands are the values that are manipulated in a calculation. For example, the calculation 4 + 5 contains the operands 4 and 5 and the + (addition) operator.

The + operator is an example of an arithmetic operator. Arithmetic operators are commonly used when performing numeric calculations and also include the - (subtraction), * (multiplication), / (division) and % (modulus) operators. Other types of operators that can be used when performing numeric calculations include comparison, logical, incrementing and decrementing operators.
For more information about operators, see page 46.

You can use the assignment operator (=) to assign a calculation to a variable. When PHP processes a variable that contains a calculation, the calculation is performed and the result is stored in the variable. The stored result will only be available within the PHP page. Once the processing of the PHP page is complete, the variable ceases to exist.

When a calculation contains an operand that is a variable, the calculation will use the value of the variable at the time the calculation is performed. If the value of the variable later changes, the change will not affect the result of the calculation.

Extra

When a calculation contains several operators, such as 4 - 5 + 2 * 2, PHP processes the operators in a specific order, which is determined by operator precedence. For example, the * operator is processed before the - or + operators, since * has a higher precedence. If you want to override operator precedence, you can use parentheses to dictate the order in which PHP should process operators. PHP will process operators and operands enclosed in parentheses first.

Example:

```
$fahrenheit = 100;
$celsius = (($fahrenheit - 32) / 9) * 5);
```

Dividing a number by zero is a mistake commonly made when performing numeric calculations. If you divide a number by zero in PHP, a `Division by zero` error will result. The most common cause of this type of error is misspelling or mistyping a variable name in a calculation. When you misspell or mistype a variable name in a calculation, the variable will have a value of zero.

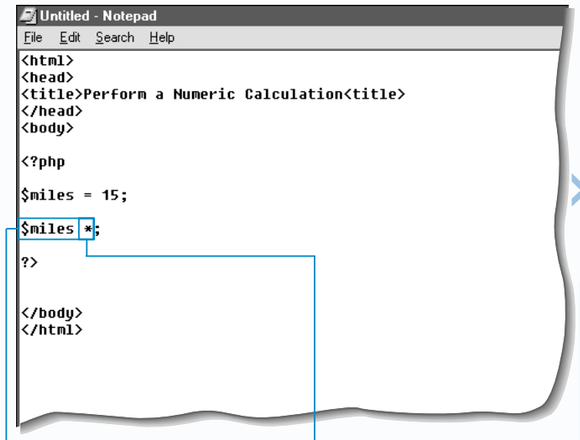
TYPE THIS:

```
$myNumber = 10;
print 1000 / $myNumbeR;
```

RESULT:

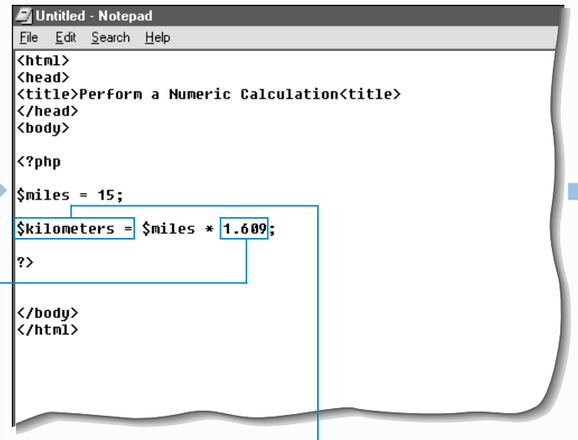
Warning: Division by zero in **c:\web_apps\apache\htdocs\calculation.php** on line **23**

PERFORM A NUMERIC CALCULATION



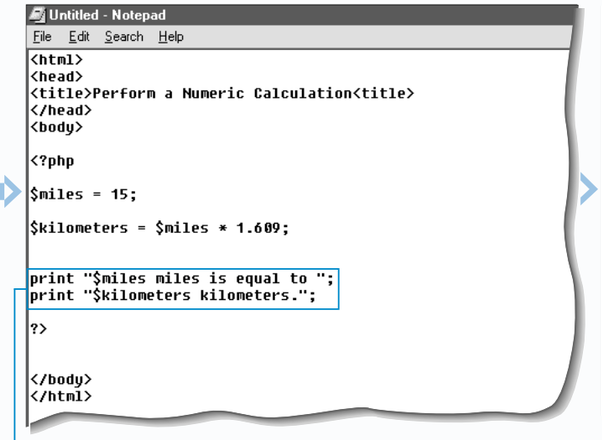
1 Type an operand you want to use in a calculation.

2 Type the operator for the type of calculation you want to perform.



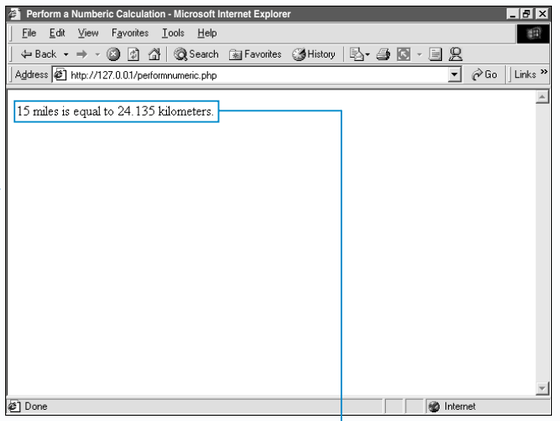
3 Type the second operand you want to use in the calculation.

4 To assign the calculation to a variable, type **$** followed by a name for the variable. Then type **=**.



5 Type the code that uses the result of the calculation.

■ In this example, we use the `print` function to display the result of the calculation.



6 Display the PHP page in a Web browser.

■ The Web browser displays the result of performing a numeric calculation.

USING THE IF STATEMENT

Using an `if` statement allows you to test a condition to determine whether the condition is true or false. When the condition is true, the section of code directly following the `if` statement is executed. For example, you can create a PHP page that displays a Good Morning message if a user views the page between 5:00 AM and 11:59 AM. If the condition is false, no code from the `if` statement is executed. A condition will be false if it evaluates to an empty string (""), a string zero ("0"), an integer zero (0) or a null value.

The condition you want to test must be enclosed in parentheses `()`. Unlike most PHP statements, the `if` statement does not end with a semicolon. Ending an `if` statement with a semicolon is a common programming mistake and will generate an error when the PHP page is processed.

A section of code you want to be executed must be enclosed in braces {} and is referred to as a statement block. To make your PHP code easier to read and understand, you should always indent the code in a statement block.

If you want an `if` statement to execute a statement block when the condition is false, you must include an `else` statement. Using an `if` statement with an `else` statement allows you to execute one of two sections of code, depending on the outcome of testing the condition. If the condition is true, the statement block directly following the `if` statement is executed. If the condition is false, the statement block directly following the `else` statement is executed. Using an `else` statement ensures that a section of code is executed regardless of the outcome of testing the condition.

Apply It

If you are going to execute only one line of code based on a condition, you can place the code to be executed on the same line as the `if` statement.

Example:

```
if ($currentTemp > $hot)
{
     print "It's hot.";
}
```

Can be typed as:

```
if ($currentTemp > $hot) print "It's hot.";
```

To test multiple conditions, you include `elseif` statements after the `if` statement. An `elseif` statement contains a condition that can be tested. If the condition specified in the `if` statement is false, PHP will test each `elseif` statement until a condition is true. PHP will then execute the block for the true `elseif` statement. If there are no true `elseif` statements, the `else` statement will be executed.

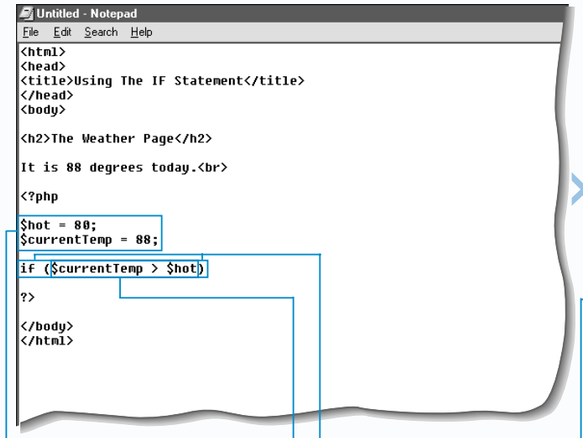
TYPE THIS:

```
$myAge = 32;
if ($myAge < 21)
{
     print "Child: \$2.00";
}
elseif ($myAge < 65)
{
     print "Adult: \$5.00";
}
else
{
     print "Senior: \$3.00";
}
```

RESULT:

Adult: $5.00

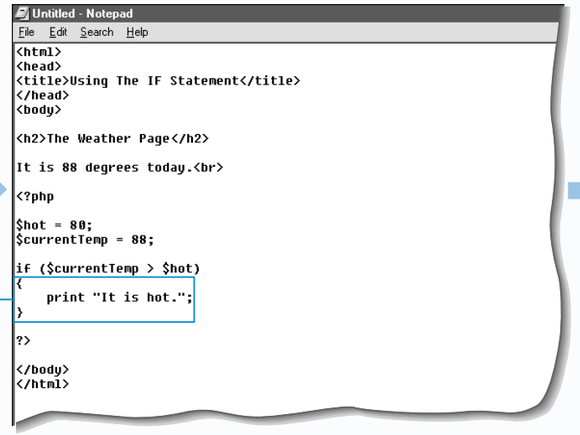
USING THE IF STATEMENT



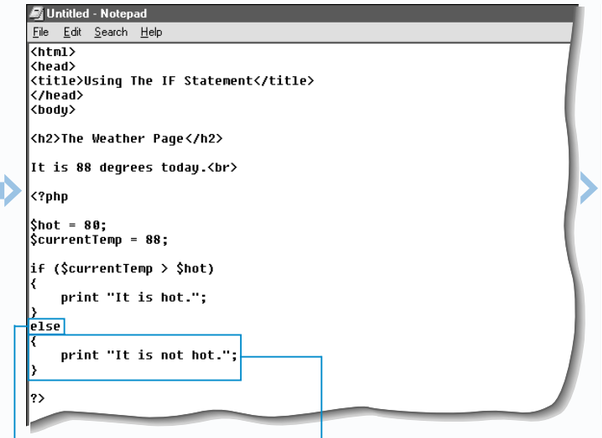
1 Type the code that creates the variables and assigns their values.

2 Type **if ()**.

3 Between the parentheses, type the condition you want to test.

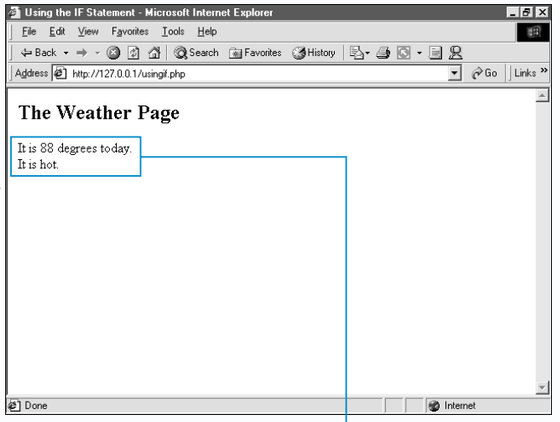


4 Type the code you want to execute if the condition you specified is true. Enclose the code in braces.



5 To use the `else` statement, type **else**.

6 Type the code you want to execute if the condition you specified is false. Enclose the code in braces.



7 Display the PHP page in a Web browser.

The Web browser displays the result of using the `if` statement.

USING THE FOR STATEMENT

Programmers often need to execute the same block of code several times. The `for` statement allows you to create a loop that repeats the execution of code a specific number of times. For example, you may want to create five line breaks on a Web page. Instead of typing the code that creates a line break five times, you can create a loop that executes the code to create a line break and then repeats the loop until the value of a counter reaches 5.

A `for` statement consists of three expressions. The first expression is called the initialization expression and is used to specify the initial value of the variable that will be used to control the processing of the loop. This variable is referred to as the iterator and is often used as a counter in a loop.

The second expression is known as the conditional expression. This expression allows you to specify a condition that evaluates the value of the iterator at the beginning of each loop. If the condition is true, the loop is executed and a block of code you specify is processed. If the condition is false, the block of code is not executed and the loop is ended. The block of code is placed between braces {} and is known as the body of the loop.

The third expression in a `for` statement is the re-initialization expression. This expression is used to modify the value of the iterator. Typically, the increment operator (++) is used in the re-initialization expression to increment the value of the iterator by one each time the loop is executed. The re-initialization expression should modify the iterator in a way that will eventually lead to the conditional expression being evaluated as false. Otherwise, the loop will be processed indefinitely.

Extra

The decrement operator (--) can be used in the re-initialization expression to decrease the value of the iterator by one each time the loop is processed. This allows you to create a `for` loop that counts backwards.

Example:

```
for ($i = 5; $i >= 0; $i--)
{
     print "$i<br>";
}
```

When creating a re-initialization expression, you are not limited to using the increment and decrement operators to modify the value of the iterator by 1. An iterator can be modified by any increment, including floating point numbers. For example, an iterator can be initialized with a value of 0 and then increased in increments of 0.5 until a value of 5 is reached.

Example:

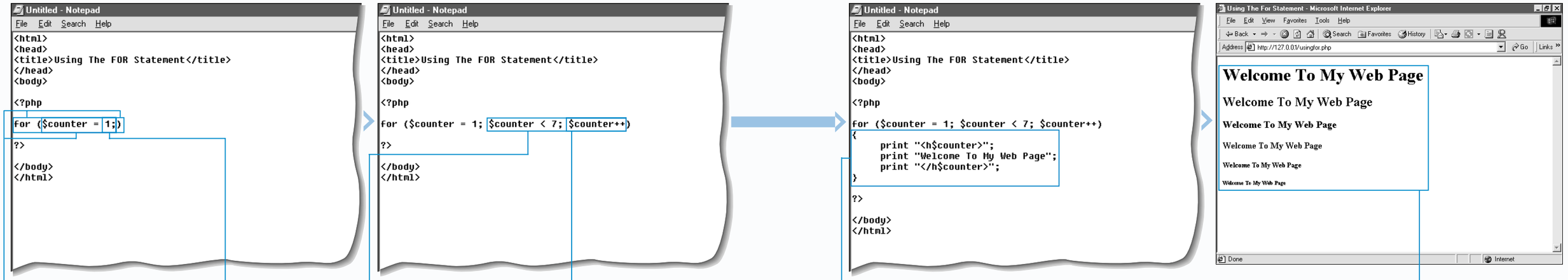
```
for ($i = 0; $i < 5; $i = $i + 0.5)
{
     print "$i<br>";
}
```

One or more expressions can be omitted from a `for` statement, though this may adversely affect the processing of the loop. For example, if a `for` statement does not include a conditional expression, PHP assumes the condition is always true and an infinite loop may be created. You should be careful not to create an infinite loop.

Example:

```
for ($i = 1; ; $i++)
{
     print "$i<br>";
}
```

USING THE FOR STATEMENT



1 Type **for** ().

2 Between the parentheses, type the name of the variable you want to use as the iterator followed by **=**.

3 Type the initial value you want to assign to the iterator followed by a semicolon.

4 Type the conditional expression you want to use to evaluate the value of the iterator followed by a semicolon.

5 Type the re-initialization expression you want to use to modify the value of the iterator each time the loop is executed.

6 Type the code you want to execute as long as the specified condition is true. Enclose the code in braces.

7 Display the PHP page in a Web browser.

The Web browser displays the result of executing the loop.

USING THE WHILE STATEMENT

The `while` statement allows you to create a conditional loop that will execute a section of code as long as a specified condition is true. Conditions often test the value of a variable. For example, you may want to process a pay statement for each of the 100 employees in a company. Instead of typing the code that will process a pay statement 100 times, you can create a loop that will repeat the pay statement for each employee. The condition would check how many pay statements have been processed. After the 100th pay statement has been processed, the condition would be evaluated as false and the loop would end.

The body of a `while` loop is enclosed in braces { } and contains the section of code to be executed, called the statement block. If the condition tests the value of a variable, the loop body will also contain code to alter the value of the variable. When the condition is true, the statement block is executed. When PHP reaches the end of the loop body, the condition is re-evaluated. If the condition is still true, the code in the loop body is executed again. If the condition is false, the statement block is not executed and the loop ends.

When creating a conditional loop, you must ensure that the condition being tested will be evaluated as false at some time. If the condition is always true, the code in the loop body will be executed indefinitely, creating an *infinite loop*. You should also keep in mind that there is typically a 30-second time limit for the processing of PHP code. You should thoroughly test your PHP code to prevent a PHP page from possibly timing out during processing, which could result in data loss. For information about the timeout settings for a PHP page, see page 219.

Extra

A `while` loop may be used to count the number of characters in a string or the number of lines in a file.

Example:

```
$word = "dynamic";
$x = 0;
while ($word[$x] != null)
{
      $x++;
}
print "The word <b>$word</b> has $x characters.<br>";
```

A `do..while` statement can be used to test a condition after the body of a loop has been executed. This is useful if you have a section of code that you want to execute at least once, regardless of how the condition is evaluated.

TYPE THIS:

```
$loopCounter = 0;
do
{
      print "This is line number ";
      print $loopCounter . "<br>";
      $loopCounter++;
} while ($loopCounter < 0);
```

RESULT:

This is line number 0

You can place a loop within another loop to create a nested loop.

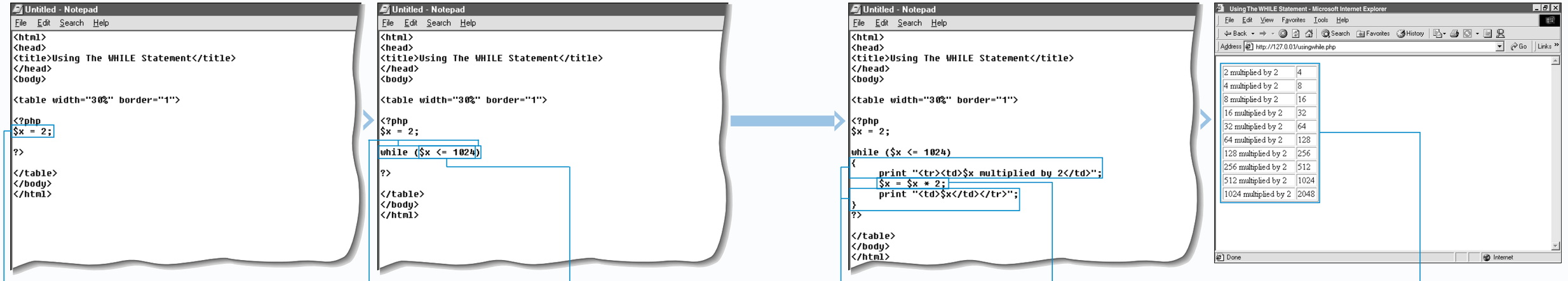
TYPE THIS:

```
$loopCounter = 0;
do
{
      print "This is line number";
      for ($x = 0; $x < 8; $x++)
      {
            print ".";
      }
      print "$loopCounter<br>";
      $loopCounter++;
} while ($loopCounter < 3);
```

RESULT:

This is line number.....0
This is line number.....1
This is line number.....2

USING THE WHILE STATEMENT



1 Type the code that creates a variable and assigns it a value.

2 Type **while ()**.

3 Between the parentheses, type the condition you want to evaluate.

4 Type the code you want to execute as long as the condition you specified is true. Enclose the code in braces.

5 Within the braces, type the code that will alter the value of the variable each time the loop is executed.

6 Display the PHP page in a Web browser.

■ The Web browser displays the result of using the `while` statement.

USING THE SWITCH STATEMENT

The `switch` statement allows you to execute a section of code, depending on the value of an expression. When a `switch` statement is executed, the value of the expression is compared to a number of possible choices, which are specified using `case` statements. If the value of the expression matches a `case` statement, the section of code following the `case` statement is executed. For example, you can create a `switch` statement that displays a specific message, depending on information entered by a user.

To use the `switch` statement, you must first specify the expression you want to use. For example, you can specify a variable whose value you want to check. After specifying the expression, you must create the `case` statements that contain the values the expression will be compared to. A `case` statement consists of the `case` keyword followed by a value and a colon (:). The value can be any scalar value, such as an integer or string, or an expression that evaluates to a scalar value.

The `switch` statement compares the value of the expression to each `case` statement in order, from top to bottom. To make your code more efficient, you should place the `case` statements that are most likely to match the expression first.

You can use the `break` statement to prevent the code for the remaining `case` statements from being executed after a match has been made. The `break` statement should be the last statement in the section of code that follows each `case` statement. Although the last `case` statement does not require a `break` statement, some programmers include it to be consistent. This can help you avoid accidentally leaving out the `break` statement if you later add another `case` statement to the `switch` statement.

Extra

You do not need to include code for each `case` statement in a `switch` statement. You can execute the same code for several `case` statements by grouping the `case` statements together and including the code after the last statement in the group. This saves you from having to retype the same code for several `case` statements.

Example:

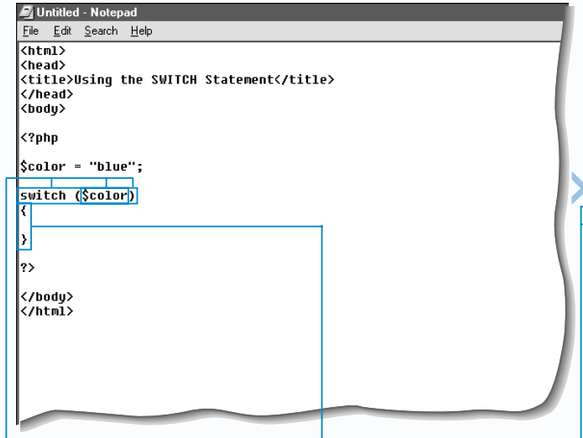
```
switch ($item)
{
      case "hat":
      case "shirt":
      case "pants":
            print "Clothing Category";
            break;
      case "fruit":
      case "vegetables":
            print "Food Category";
            break;
}
```

You can include a `default` statement in a `switch` statement if you want to execute specific code when none of the `case` statements match the specified expression. The `default` statement is placed last in the `switch` statement structure.

Example:

```
switch ($priority)
{
      case 1:
            print "Urgent";
            break;
      case 2:
            print "High Priority";
            break;
      default:
            print "Low Priority";
            break;
}
```

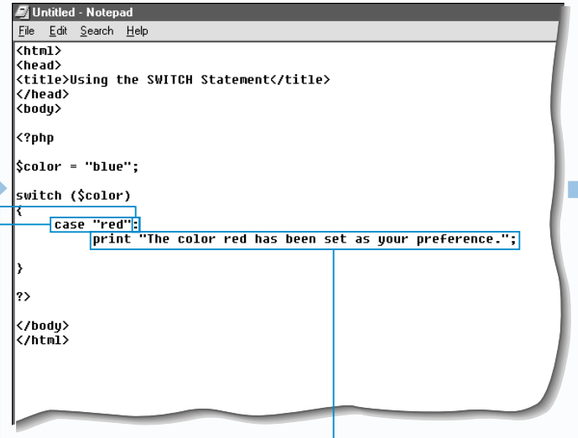
USING THE SWITCH STATEMENT



1 Type **switch** ().

2 Between the parentheses, type the expression you want the `switch` statement to use.

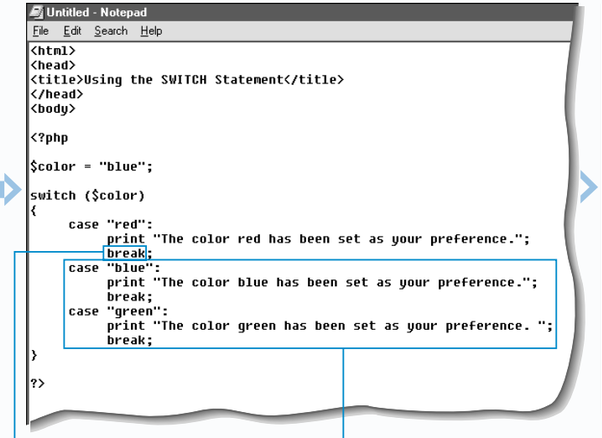
3 Type a pair of braces to hold the `case` statements.



4 Type **case** followed by a value the expression may contain.

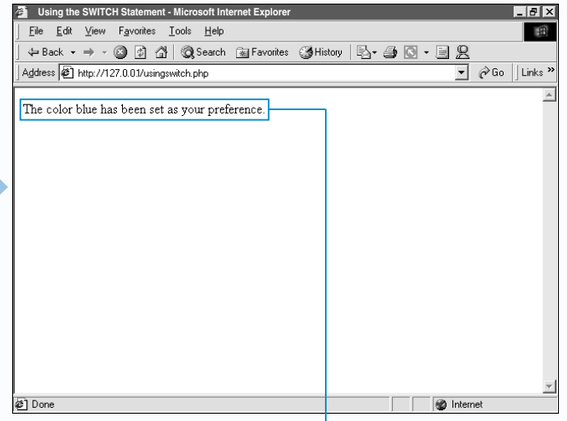
5 Type **:** to complete the `case` statement.

6 Type the code you want to execute if the value in the `case` statement matches the expression you specified in step 2.



7 Type **break** to prevent the code for the remaining `case` statements from being executed once a match has been made.

8 Repeat steps 4 to 7 for each value the expression may contain.



9 Display the PHP page in a Web browser.

■ The Web browser displays the result of using the `switch` statement.

CREATE AN ARRAY

An array is a variable that stores a set of related values. For example, an array could store the name of each day of the week. Using an array allows you to work with multiple values at the same time. Each value in an array is referred to as an element.

The first step in creating an array is to create a variable to store the array. You can then use the `array` keyword to assign a set of values, or elements, to the variable. The elements are enclosed in parentheses and separated by commas. String elements in an array must be enclosed in quotation marks.

In PHP, it is possible to store values of different data types in a single array. You may even store other arrays in an array. PHP's array-handling capabilities make it more flexible and intuitive than most other programming languages.

Each element in an array is uniquely identified by an index number, or key. PHP numbers the elements in array starting at zero (0). To access a specific element in an array, you enter the name of the array followed by the index number of the element you want to access enclosed in brackets, such as $days[1].

You can use an array element in a script as you would use a variable. The value of an element can be directly displayed to a Web browser using the `print` function or assigned to another variable that can then be processed further.

An element can be changed or a new element can be added to an array without affecting the other elements in the array. For more information about changing or adding elements in an array, see page 62.

Apply It

Creating a loop allows you to access all the elements in an array. You can use the `count` function to determine the total number of elements in the array and use the loop's counter variable as the index to access each element.

TYPE THIS:

```
$vegetables = array ("broccoli", "carrot",
                     "zucchini", "eggplant");
for ($x = 0; $x < count($vegetables); $x++)
{
      print $vegetables[$x] . "<br>\n";
}
```

RESULT:

broccoli
carrot
zucchini
eggplant

You can use the `foreach` statement to create a loop that allows you to work with each consecutive element in an array without having to specify each element's index. In the following example, each element in the $vegetables array is assigned to the $value variable.

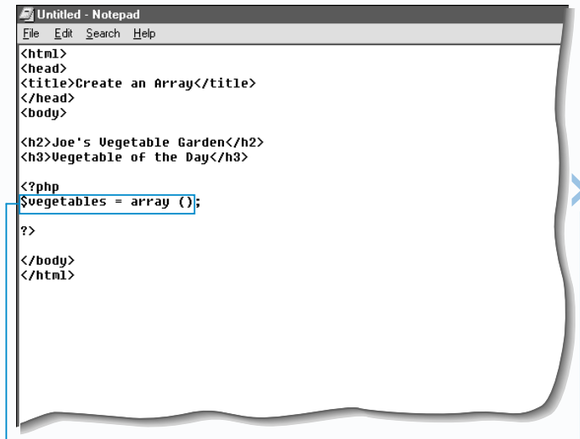
TYPE THIS:

```
$vegetables = array ("broccoli", "carrot",
                     "zucchini", "eggplant");
foreach ($vegetables as $value)
{
      print "The vegetable is: $value<br>\n";
}
```

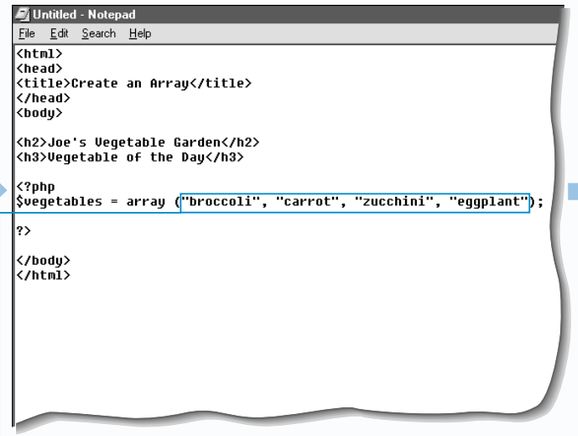
RESULT:

The vegetable is: broccoli
The vegetable is: carrot
The vegetable is: zucchini
The vegetable is: eggplant

CREATE AN ARRAY

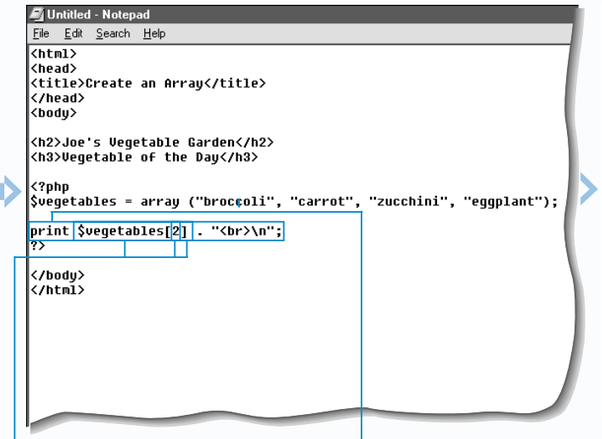


1 Type a name for the array you want to create followed by **= array ()**.



2 Between the parentheses, type each element you want to add to the array separated by a comma.

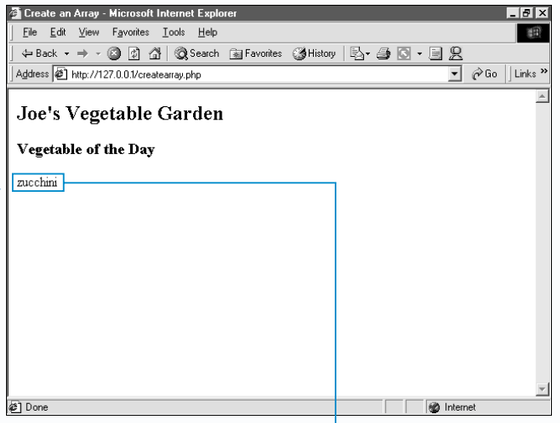
■ If the value for an element is a string, you must enclose the string in quotation marks.



3 To access an array element, type the name of the array followed by [].

4 Between the brackets, type the index of the element you want to access.

5 Type the code that uses the element in the array.



6 Display the PHP page in a Web browser.

■ The Web browser displays an element in the array.

WORK WITH ARRAYS

Once you have created an array, you can work with the elements in the array. For example, you can change or add elements in an array.

To change an element in an array, you must first specify the name of the array followed by a set of brackets. You must then specify the index number of the element you want to change. Keep in mind that PHP usually numbers the elements in an array starting at zero (0).

PHP allows you to change the value of an element in an array by assigning the element a new value. When changing the value of an element, make sure you do not make unintended changes to the data, such as inserting text data into an array you want to contain only numbers.

You use a similar method to add a new element to an existing array. When adding a new element, you do not need to specify an index number. PHP will assign the new element an index number one greater than the current largest index number in the array.

You may also explicitly specify the index number of a new element you add to an array. If you specify an index number that is more than one greater than the current largest index number, PHP will not create elements to fill the gap between the index numbers. Any new elements later added to the array will be placed after the new largest index number.

If a new element is added to an array that does not yet exist, a new array will automatically be created. It is better programming practice, however, to first create an array before adding new elements to the array.

Extra

You can create an empty array and then later add elements to the array. This is useful when you want to filter array elements or dynamically add elements to an array using data obtained from a database or a file. To create an empty array, use the `array` keyword without placing any elements within the parentheses.

Example:

```
$unfiltered = array ("apple", "orange", "not fruit",
                     "grape", "banana", "not fruit",
                     "not fruit", "cherry");
$filtered = array ();
foreach ($unfiltered as $value)
{
      if ($value != "not fruit")
      {
      $filtered[] = $value;
      }
}
foreach ($filtered as $value)
{
      print "$value<br>";
}
```

You can create an array with an index that starts with a number other than 0. The => operator is used to explicitly assign a starting index to the first element in the array. The next elements in the array will then be indexed accordingly in sequence.

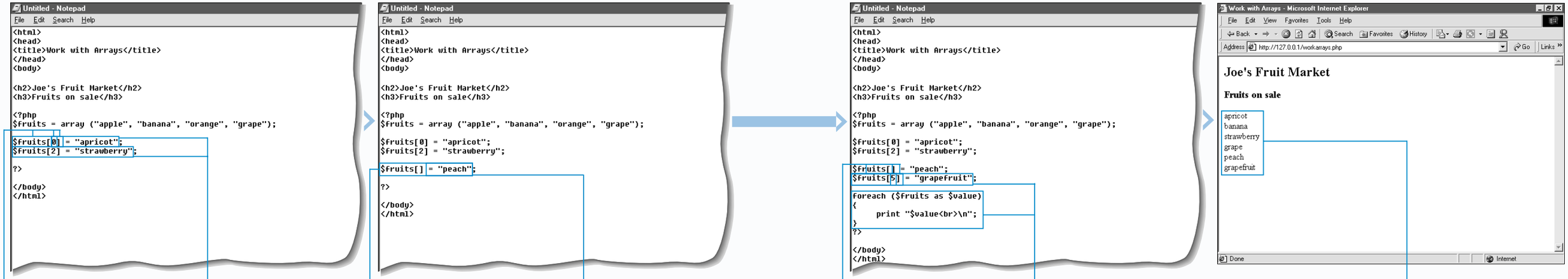
TYPE THIS:

```
$startWithTen = array (10 => "ten", "eleven",
                       "twelve", "thirteen");
foreach ($startWithTen as $key => $value)
{
      print "key: $key, value: $value<br>\n";
}
```

RESULT:

```
key: 10, value: ten
key: 11, value: eleven
key: 12, value: twelve
key: 13, value: thirteen
```

WORK WITH ARRAYS



CHANGE AN ELEMENT

1. To change an element in an array, type the name of the array followed by [].
2. Between the brackets, type the index number of the element you want to change.
3. Type = followed by the value you want to assign to the element.
4. Repeat steps 1 to 3 for each element you want to change.

ADD AN ELEMENT

5. To add an element to an array, type the name of the array followed by [].
6. Type = followed by the value you want to assign to the new element.
7. To add an element with a specific index number to an array, type the name of the array followed by [].
8. Between the brackets, type the index number you want to assign to the new element.
9. Type = followed by the value you want to assign to the element.
10. Type the code that generates the results of your changes.
11. Display the PHP page in a Web browser.

The Web browser displays the results of changing and adding elements in an array.

USING A FUNCTION

A function, sometimes referred to as a subroutine, contains a block of code that performs a specific task, such as displaying a message. Using functions makes it easy to re-use sections of code in a PHP page. For example, you may have a function that displays a warning message when a client enters invalid data into a form. Instead of retyping the section of code that displays the message for each field in the form, you can simply re-use the function.

Functions also allow you to group lines of code into smaller, more manageable sections. This makes it easier to understand and troubleshoot the code.

A function is created using the `function` keyword followed by the name of the function and a set of parentheses. A function name must start with a letter or an underscore (_) character. The name should identify the purpose of the function or describe the action the function performs.

It is common to have a function name begin with a capital letter. If the name consists of multiple words, you can capitalize the first letter of each word to make the name easy to read, such as AddTotalCost. Function names are case-insensitive, but you should be consistent in naming functions to help make your scripts easier for other people to understand.

The block of code used in a function is enclosed in braces { }. The block of code will not be executed until the function is called in the script. To call a function, you type the name of the function followed by a set of parentheses where you want to execute the code.

PHP has many useful built-in functions, such as the `print` function. To avoid unnecessary work, you should check whether there is an existing function that suits your needs before defining your own function.

Apply it

You can call a function from within another function. When calling other functions from within a function, you should be careful not to generate a loop that causes the functions to continuously call each other.

TYPE THIS:

```
function WelcomeMessage()
{
     print "Welcome to my Web page!<br>";
}
function DisplayIntroduction()
{
     WelcomeMessage();
     print "My name is Martine.<br>";
}
DisplayIntroduction();
```

RESULT:

Welcome to my Web page!
My name is Martine.

When creating a function, you should keep in mind that any variables created within the function can be accessed only within that function. The function's variables will not be accessible to other functions or other sections of code in the PHP page.

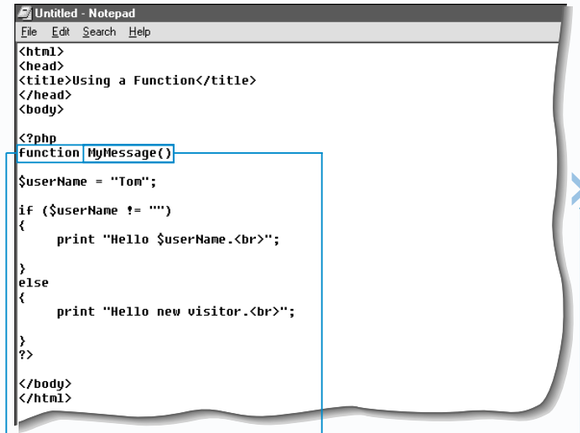
TYPE THIS:

```
function SetPageTitle()
{
     $pageTitle = "Martine's Home Page";
}
function DisplayWelcome()
{
     print "Welcome to $pageTitle<br>";
}
DisplayWelcome();
```

RESULT:

Welcome to

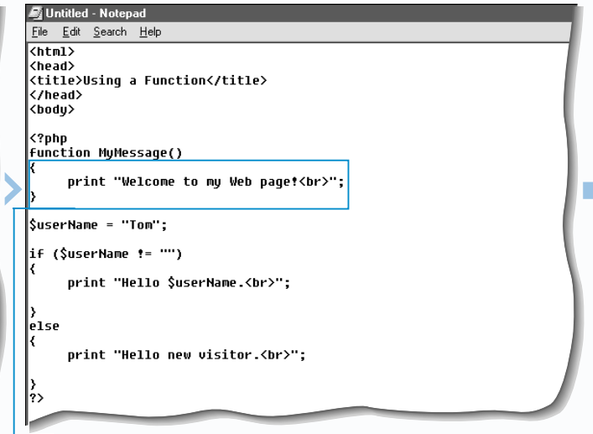
USING A FUNCTION



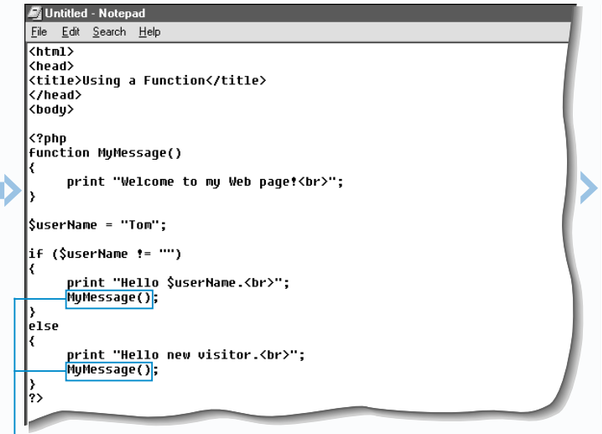
CREATE A FUNCTION

1 To create a function, type **function**.

2 Type a name for the function followed by **()**.

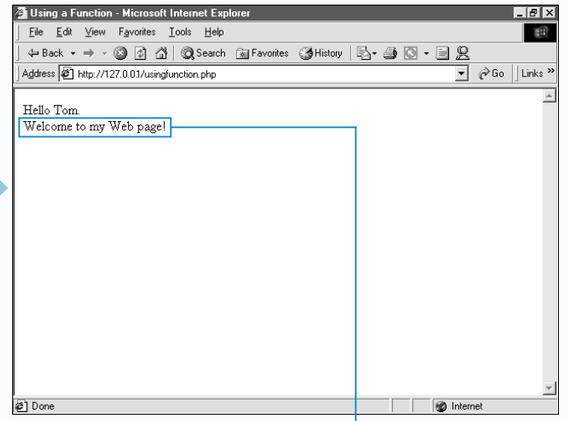


3 Type the code you want to execute when the function is called. Enclose the code in braces.



CALL A FUNCTION

4 To call a function, type the function name followed by **()**.



5 Display the PHP page in a Web browser.

The Web browser displays the result of the function.

RETURN A VALUE FROM A FUNCTION

Although a function may simply display a value using the `print` function, it may be more useful to have the function return a value to the code that calls the function. A value returned from a function may be the result of a calculation or procedure or may indicate whether a process was successfully completed. For a function to return a value, you must include a `return` statement.

A `return` statement consists of the `return` keyword followed by the value you want the function to return and is usually placed at the end of a function, after the code that performs an operation and generates or calculates the value to be returned. The execution of a function will terminate once the `return` statement is processed.

A function can return a value of any single data type, such as strings, integers, floating-point numbers and boolean values. A function typically returns only one value. If you need to have a function return multiple values, you should first place the values in an array and then return the array using the `return` keyword.

When a function that returns a value is called, the value can be directly displayed to a Web browser using the `print` function. It may be more useful, however, to assign the value returned from a function to a variable that can then be processed further by the PHP script.

The `return` keyword may also be used to simply denote the end of a function. This is useful for functions that will not return a value. In such cases, the `return` keyword is used alone and is not followed by any value or variable.

Apply It

A function can have more than one `return` statement. This is commonly found in functions that use conditional statements. Although a function can have more than one `return` statement, only one `return` statement will be executed. When a `return` statement is processed, the execution of the function is terminated.

TYPE THIS:

```
function CheckAge()
{
      $age = 19;
      if ($age >= 18)
      {
            return "You may vote.<br>";
      }
      else
      {
            return "You may not vote.<br>";
      }
}
print CheckAge();
```

RESULT:

You may vote.

In PHP, most built-in functions return a boolean value. This is useful for helping to prevent your PHP scripts from crashing or generating errors. For example, the `is_numeric` function will return a value of true if the argument passed to the function is a number or a numeric string. Otherwise, the function returns a value of false.

Example:

```
$x = 8;
$y = "Hello";
if (is_numeric($y) == TRUE)
{
      print "Answer: " . $x / $y . ".<br>";
}
else
{
      print "Cannot perform division.<br>";
}
```

RETURN A VALUE FROM A FUNCTION

Untitled - Notepad

File Edit Search Help

```
<html>
<head>
<title>Return a Value from a Function</title>
</head>
<body>

<h2>Bill's Bargain Warehouse</h2>
Thank you for your order!<br>

<?php
Function TotalPrice()
{
     $retailPrice = 15.00;
     $tax = 0.07;
     $totalPrice = $retailPrice + ($retailPrice * $tax);
}

?>

</body>
</html>
```

1 Create the function you want to return a value from.

Untitled - Notepad

File Edit Search Help

```
<html>
<head>
<title>Return a Value from a Function</title>
</head>
<body>

<h2>Bill's Bargain Warehouse</h2>
Thank you for your order!<br>

<?php
function TotalPrice()
{
     $retailPrice = 15.00;
     $tax = 0.07;
     $totalPrice = $retailPrice + ($retailPrice * $tax);
     return $totalPrice;
}

?>

</body>
</html>
```

2 In the body of the function, type **return** followed by the information you want the function to return.

■ In this example, we return a variable that contains the result of a calculation.

Untitled - Notepad

File Edit Search Help

```
<html>
<head>
<title>Return a Value from a Function</title>
</head>
<body>

<h2>Bill's Bargain Warehouse</h2>
Thank you for your order!<br>

<?php
function TotalPrice()
{
     $retailPrice = 15.00;
     $tax = 0.07;
     $totalPrice = $retailPrice + ($retailPrice * $tax);
     return $totalPrice;
}

print "The total price is \$" . TotalPrice() . "<br>\n";
?>

</body>
</html>
```

3 Type the code that calls the function and uses the value returned from the function.

Return a Value from a Function - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Search Favorites History

Address http://127.0.0.1/returnvalue.php Go Links

Bill's Bargain Warehouse

Thank you for your order!
The total price is $16.05

Done Internet

4 Display the PHP page in a Web browser.

■ The Web browser displays the result of returning a value from a function.

PASS ARGUMENTS TO A FUNCTION

Passing arguments to a function you have created can make the function more powerful by allowing you to use one function throughout a script to process different data.

In the `function` statement, you define one or more variables between the parentheses that follow the function name. Each variable represents a value, or argument, you will pass to the function. Each time you call the function in the PHP page, you will specify the arguments you want to pass to the function.

You can use the variables you create in the `function` statement within the body of the function. The variables will exist only within the function and will not be available once the function has finished processing. This means that you will not be able to retrieve the value of a variable in a function from outside the function. The function can use the values that are passed to it to perform a calculation and then send the result back to the code used to call the function. The result can be used in many ways in a PHP page. For example, you can assign the result to another variable or display the result on the screen.

You can pass one or more arguments to a function, but the number of arguments you pass must match the number of variables you created to store the arguments. An argument you pass can be a variable or any alphanumeric value, but the argument must be appropriate for the purpose of the function. For example, you cannot pass a text value to a function if the value will be used in a mathematical calculation. Text values passed to a function must be enclosed in quotation marks.

Extra

You can specify a default value for an argument that will be passed to a function. When creating the variables that will store arguments you pass, you can assign a value you want to use as a default argument to a variable. A variable that has a default value assigned must be last in the list of variables in the `function` statement.

When you pass arguments to the function, you do not need to specify an argument for the variable that has a default value assigned. The variable will automatically use the default argument. You will need to pass an argument to the variable only if you want to use a different argument.

TYPE THIS:

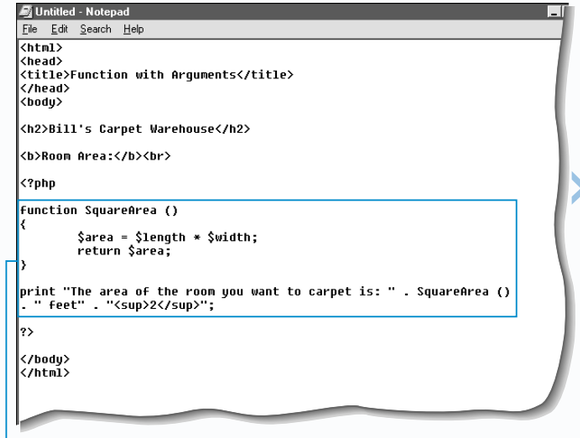
```
function MyMessage ($message, $size = 3)
{
     print "<font size=\"$size\">$message</font><br>";
}

MyMessage ("Welcome to My Web Site");
MyMessage ("Welcome to My Web Site", 5);
```

RESULT:

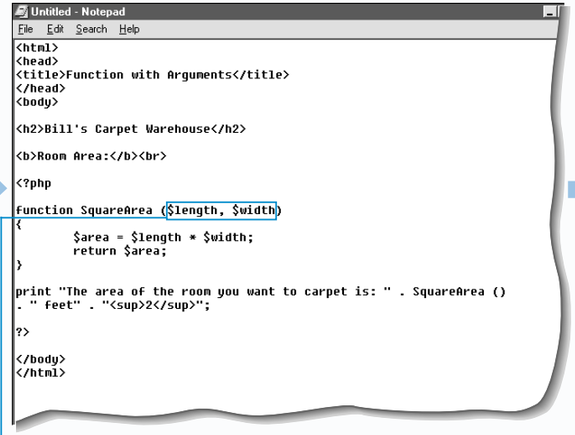
Welcome to My Web Site

Welcome to My Web Site

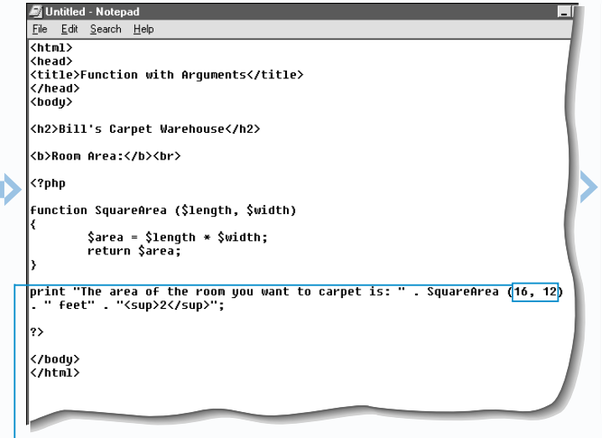
PASS ARGUMENTS TO A FUNCTION



1 Create and call the function you want to pass arguments to.

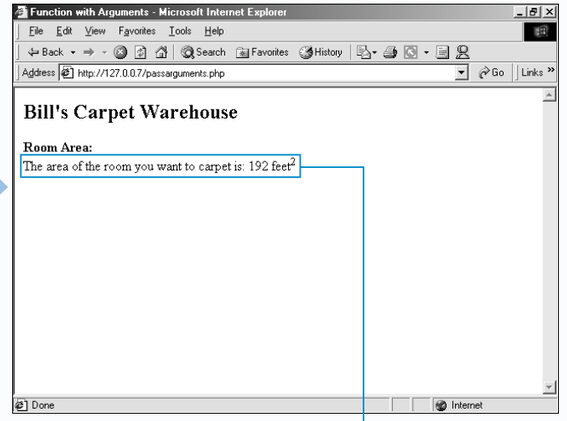


2 Between the parentheses in the code that creates the function, type the names of the variables you want to store the arguments that will be passed to the function. Separate each variable name with a comma.



3 Between the parentheses in the code that calls the function, type the arguments you want to pass to the function. Separate each argument with a comma.

■ The number of arguments you pass must match the number of variables you created in step 2.



4 Display the PHP page in a Web browser.

■ The Web browser displays the result of passing arguments to the function.

PASS ARGUMENTS TO A FUNCTION BY REFERENCE

By default, when an argument you pass to a function is a variable created outside the function, only the value of the variable is passed to the function. Any changes the function makes to the value do not affect the variable in the PHP page. If you want to use a function to change the value of a variable in the PHP page, you must pass the variable to the function by reference. Passing an argument by reference passes an entire variable to the function, rather than simply passing the value of the variable.

Before you can pass arguments to a function, you must create one or more variables between the parentheses that follow the function name in the `function` statement. Each variable represents an argument you will pass to the function. The variables can then be used in the body of the function, allowing the function to work with the arguments that are passed to it. The variables you create in the `function` statement exist only within the function and are not available outside the function.

To pass an argument to a function by reference, you preface the argument with an ampersand (&) in the code that calls the function. An argument you pass by reference must be a variable. You can pass one or more arguments to a function by reference, but the number of arguments you pass must match the number of variables you created to store the arguments.

When the function is called, the arguments you pass by reference are assigned to the variables you created in the `function` statement. When the variables are modified within the function, the values of the arguments they contain are also modified. The results of the function are then sent back to the code used to call the function.

Extra

If you want an argument passed to a function to always be passed by reference, you can type an ampersand (&) in front of the variable you create to store the argument in the `function` statement. This saves you from having to type an ampersand each time you pass an argument to the function.

TYPE THIS:

```
function TimesThree(&$number)
{
    $number *= 3;
}

$myNumber = 5;

print "The original number is: $myNumber<br>";
TimesThree($myNumber);
print "The new number is: $myNumber<br>";
```

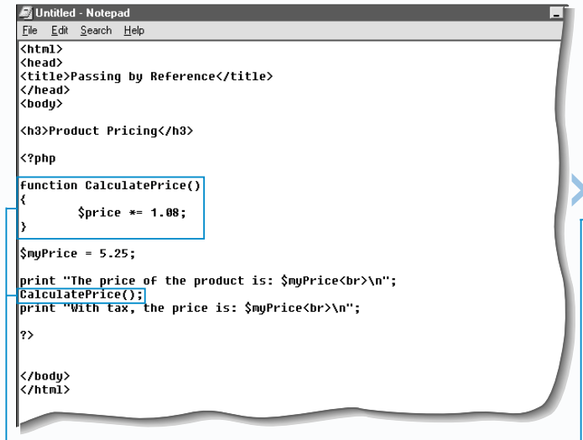
RESULT:

```
The original number is: 5
The new number is: 15
```

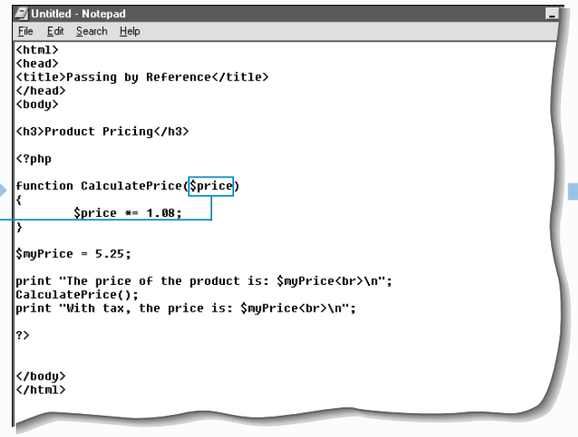
Unlike variables, whose scope does not automatically extend to functions, constants you create in a PHP page are automatically accessible to the functions you create. This means that you do not need to pass a constant to a function before the constant can be used in the function. Constants are normally created at the beginning of a script.

It is considered good programming practice to create all of your functions near the beginning of a script. This can make your code easier to understand and maintain. You can then call the functions wherever you want to execute the code in the PHP page.

PASS ARGUMENTS TO A FUNCTION BY REFERENCE

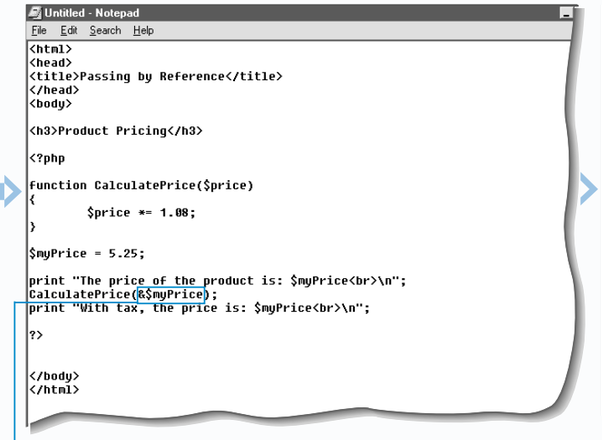


1 Create and call the function you want to pass arguments to by reference.



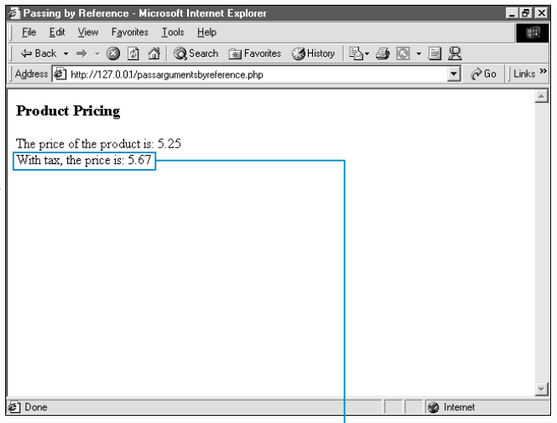
2 Between the parentheses in the code that creates the function, type the name of the variable you want to store the argument that will be passed to the function.

- If multiple arguments will be passed to the function, type the names of all the variables you want to store the arguments. Separate each variable name with a comma.



3 Between the parentheses in the code that calls the function, type an ampersand followed by the name of a variable you want to pass to the function.

- You can repeat step 3 for each argument you want to pass by reference, separating each argument with a comma. The number of arguments you pass must match the number of variables you created in step 2.



4 Display the PHP page in a Web browser.

- The Web browser displays the result of passing an argument to the function by reference.

UNDERSTANDING VARIABLE SCOPE

The scope of a variable determines which parts of a script can access the variable and use its value.

If you create a variable in the body of a function, the value of the variable will only be available in the function. Variables created in a function are referred to as local variables. By default, a function can access only local variables. This is an important safety feature in PHP because it prevents variables created outside a function from interfering with variables of the same name created within the function.

To allow a function to use a variable that was created outside of the function, you must declare the variable as global within the function. To declare a variable as global, you use the keyword `global` followed by the name of the variable. The global declaration must be made before the variable can be used. Declaring a variable as global within a function does not create a new variable, but simply allows the function to access and use the value of the variable. Any changes made to a global variable inside a function also affect the value of the variable outside of the function.

Variables have a limited lifetime in PHP. The lifetime of a variable refers to the length of time the value of the variable exists in memory. A variable created in a PHP script exists only for the duration of the script. When the script is finished processing, the value of the variable ceases to exist. A variable created in a function exists only while the function is being executed. The value of a local variable ceases to exist when the function terminates.

Apply It

You can have PHP retain the value of a local variable after a function has finished executing and use the value the next time the function is called. To retain the value of a local variable, you use the `static` keyword when creating the variable. The initial value you assign to the variable is used the first time the function is called. The next time the function is called, PHP will use the value assigned to the variable the last time the function was executed.

The value of a static local variable is retained only for the duration of the script. Once the script is finished processing, the value ceases to exist.

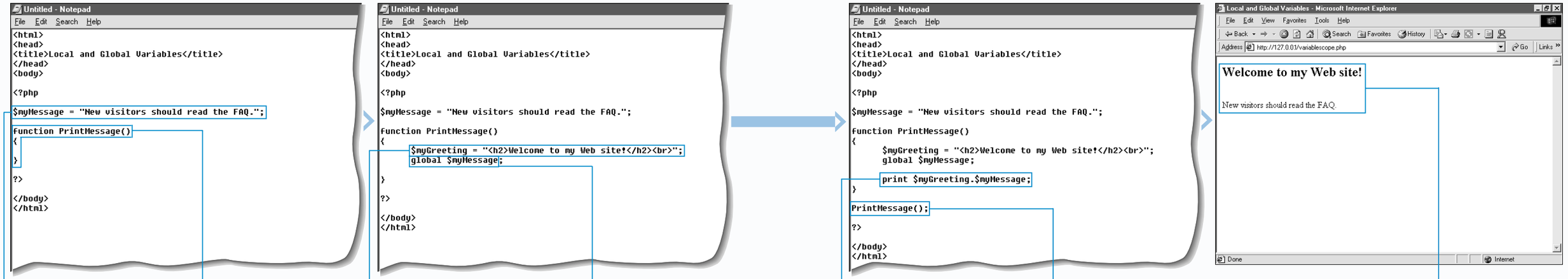
TYPE THIS:

```
function Multiply($number)
{
      static $container = 1;
      $container *= $number;
      return $container;
}
$myNumber = Multiply(2);
print "1 times 2 is: $myNumber<br>\n";

$myNumber = Multiply(4);
print "...times 4 is: $myNumber<br>\n";

$myNumber = Multiply(3);
print "...times 3 is: $myNumber<br>\n";
```

RESULT:

```
1 times 2 is: 2
...times 4 is: 8
...times 3 is: 24
```

UNDERSTANDING VARIABLE SCOPE



1 Type the code that creates a variable and assigns it a value.

2 Type the code that creates a function.

3 To create a local variable, type the code that creates a variable and assigns it a value in the body of the function.

4 To declare a variable as global in the body of the function, type **global** followed by the name of the variable.

5 Type the code that uses the local variable and the global variable.

6 Type the code that calls the function.

7 Display the PHP page in a Web browser.

The Web browser displays the result of using a local variable and a global variable.

USING DYNAMIC VARIABLES

PHP allows you to use dynamic variables, which are also referred to as variable variables, in your scripts. A dynamic variable is a variable whose name is stored in another variable.

A variable consists of two parts—the variable operator ($) and the name of the variable. A regular variable uses a literal string as the name of the variable. However, you can create a dynamic variable by using the name of an existing string variable as the name of the dynamic variable.

Dynamic variables are especially useful for accessing the values of a series of variables that you have previously created dynamically in your script, such as field names in a form. If you do not know the exact number of variables in the form, but you know that the names of the variables all begin with "Field_", you can create a dynamic variable that will access all the variables named "Field_" in the form.

You first create a string variable that has the same name as the dynamic variable you want to use. The string variable should contain the name of the variable you want to access. To create the dynamic variable, you type the variable operator ($), followed by the string variable you created. The result is a variable that has two variable operators ($), such as $$varName.

A dynamic variable acts as the equivalent of the original variable that stores the name of the variable you want to access. You can work with the dynamic variable in the same way you would work with a regular variable.

Extra

You can access functions dynamically, just as you would access dynamic variables. To call a function dynamically, you first assign the function name to a string variable. When you want to call the function, you can use the variable name followed by parentheses instead of the function name.

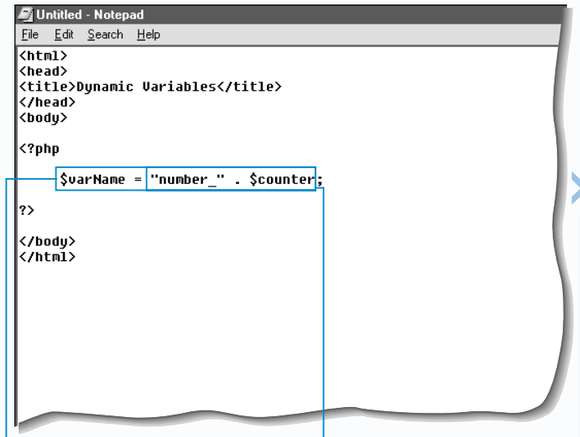
TYPE THIS:

```
function FontFour($message)
{
     print "<font size=\"4\">$message</font>";
}
function FontFive($message)
{
     print "<font size=\"5\">$message</font>";
}
function FontSix($message)
{
     print "<font size=\"6\">$message</font>";
}
$size = "Four";
if ($size != "")
{
     $functionName = "Font" . $size;
}
else
{
     $functionName = "FontSix";
}
$functionName("Hello there! How are you?");
```

RESULT:

Hello there! How are you?

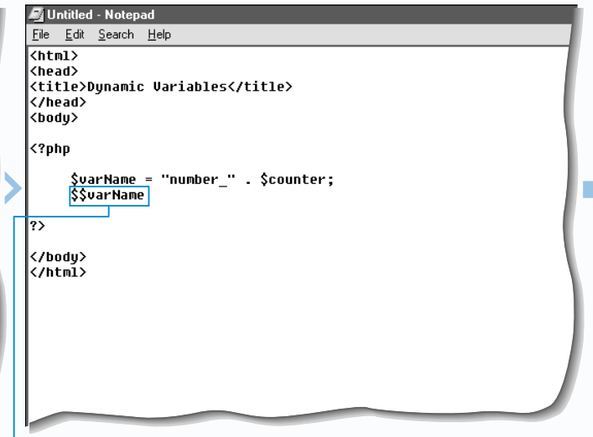
USING DYNAMIC VARIABLES



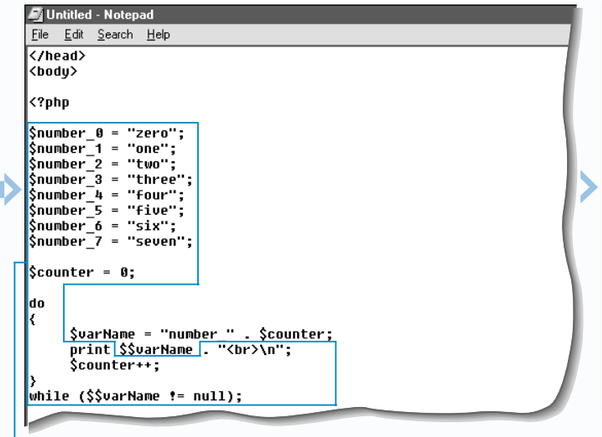
1 To create a string variable that will contain the name of the variable you want to access, type **$** followed by a name for the variable. Then type **=**.

2 Type the value or expression the variable will store, including the name of the variable you want to access.

■ String values must be enclosed in quotation marks.

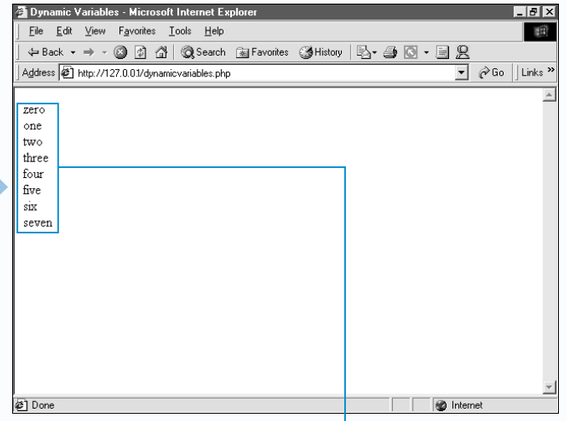


3 To create a dynamic variable, type **$** followed by the string variable you created in step 1.



4 Type the code that uses the dynamic variable.

■ In this example, the dynamic variable is used in a loop that accesses a series of string variables and displays their values.



5 Display the PHP page in a Web browser.

■ The Web browser displays the result of using a dynamic variable.

WORK WITH DATA TYPES OF VARIABLES

When creating variables in PHP, you do not have to explicitly declare the type of variable you want to create. The data type of a variable is determined by the value assigned to the variable. For example, if you assign an integer value to the $number variable, then $number will use the `integer` data type. This distinguishes PHP from other languages, such as C++ and Java, which require variables to have assigned data types.

If you are working with different types of data, PHP will automatically perform the required data type conversions. For example, if you add the string "3" to the integer 2, PHP will convert the string to an integer and return the integer 5 as the result. When PHP automatically converts a value to a new data type, it affects only how the value is evaluated. The value of the variable is not changed.

The `gettype` function allows you to determine the current data type of a variable. The `gettype` function takes the name of the variable whose data type you want to determine as its argument.

You can use the `settype` function to force PHP to always evaluate a variable using a specific data type. The `settype` function takes two arguments—the name of the variable whose data type you want to set and the data type you want the variable to use. The data types you can specify include `integer`, `double`, `string`, `boolean`, `array` and `object`. The `double` data type refers to double-precision floating-point numbers.

When you use the `settype` function, PHP will attempt to convert the value of the variable to the specified type. If the conversion is successful, the `settype` function will return a value of true. A value of false will be returned if the conversion was not successful. When a value is converted to a new data type, data may be lost. For example, if you convert a floating-point number to an integer, the fractional part of the number will be lost.

Extra

Type casting is another way to explicitly set the data type of a variable. You can cast a variable to the `integer`, `double`, `string`, `boolean`, `array` or `object` data type. You must first create the variable you want to cast. You then create a second variable to store the data type you want to use, enclosed in parentheses, and the first variable.

TYPE THIS:

```
$uncasted = 1.2345;
$casted = (integer)$uncasted;
print "Data type of the uncasted variable is: " . gettype($uncasted);
print "<br>";
print "Data type of the casted variable is: " . gettype($casted);
```

RESULT:

```
Data type of the uncasted variable is: double
Data type of the casted variable is: integer
```

When you use a string in a numeric calculation, PHP examines the beginning of the string. If the string begins with valid numeric data, PHP will use the numeric data and truncate the rest of the string. If the string begins with anything other than numeric data, the value of the string will be zero (0).

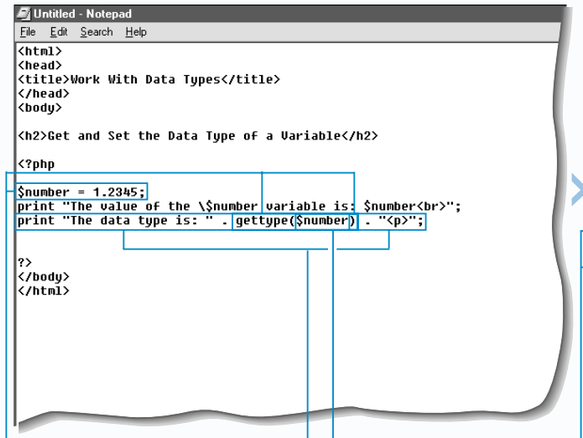
TYPE THIS:

```
$a = "100 US Dollars";
print "\$a = " . ($a + 0) . "<br>";

$b = "US Dollars: 100";
print "\$b = " . ($b + 0);
```

RESULT:

```
$a = 100
$b = 0
```

WORK WITH DATA TYPES OF VARIABLES

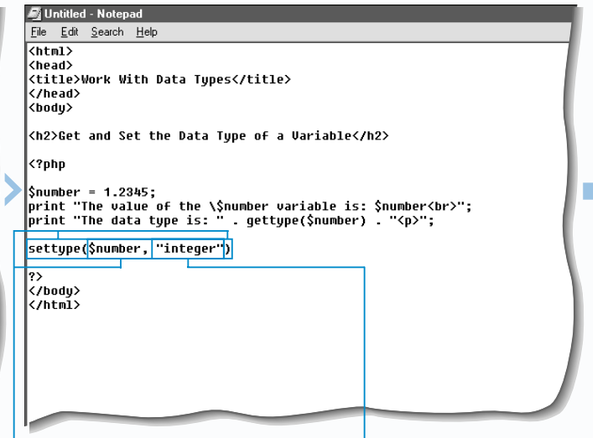


1 Type the code that creates a variable and assigns it a value.

2 To determine the data type of a variable, type **gettype**().

3 Between the parentheses, type the name of the variable whose data type you want to determine.

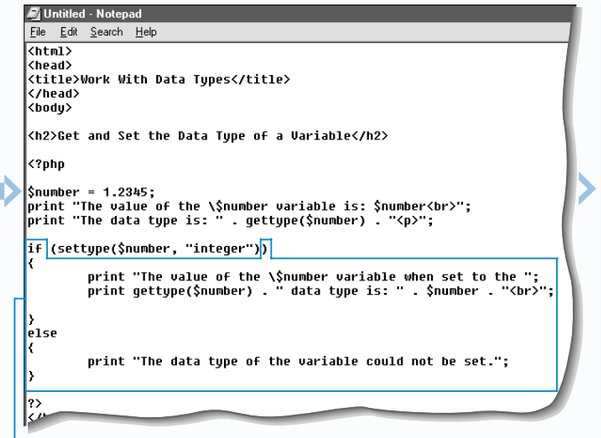
4 Type the code that uses the `gettype` function.



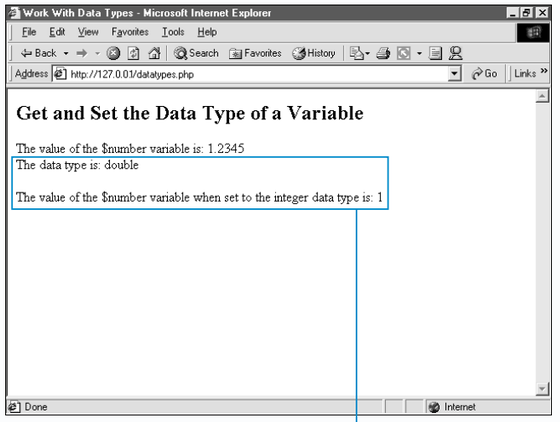
5 To set the data type of a variable, type **settype**().

6 Between the parentheses, type the name of the variable whose data type you want to set followed by a comma.

7 Type the data type you want to set for the variable, enclosed in quotation marks.



8 Type the code that uses the `settype` function.



9 Display the PHP page in a Web browser.

The Web browser displays the results of working with the data types of variables.

GET INFORMATION ABOUT A VARIABLE

There are a number of functions you may use to test whether a variable stores data of a certain type. Testing the value of data stored in a variable is useful before using a variable in a calculation. If you attempt a calculation with the wrong type of variable, an error may occur.

The `is_int`, `is_integer`, and `is_long` functions are used to verify if a variable stores an integer. You use the `is_float`, `is_real` and `is_double` functions to check whether a variable stores a floating-point number. The `is_numeric` function is used to test if a variable stores a number or a numeric string.

The `is_bool` function is used to determine if a variable is a boolean. To check if a variable stores a string, you use the `is_string` function. The `is_array` function is used to test if a variable stores an array, and the `is_object` function is used to test whether a variable is an object. You use the `is_resource` function to check if a variable is an integer that represents a system resource, such as an open file or an open database connection.

The functions that test the type of a variable return a value of true if the value stored in the variable is of the indicated type, otherwise, the functions return a value of false.

There are other useful functions you may use to get information about a variable, such as whether a variable has been assigned a value or whether a variable holds a zero (0) value. The `isset` function returns a value of true if a value has been assigned to a variable. The `empty` function returns a value of false if a variable contains a non-zero or non-empty value.

Extra

You may use the `print_r` function to view information about a variable you specify in a form that can be easily read. This function is especially useful for viewing the contents of arrays and objects. When using the `print_r` function, you may want to use the `<pre>` tag to format the result.

TYPE THIS:

```
$info = array ("Tom", 24, 65.5);
print "<pre>";
print_r($info);
print "</pre>";
```

RESULT:

```
Array
(
    [0] => Tom
    [1] => 24
    [2] => 65.5
)
```

The `var_dump` function is used to display information about a variable in an easy-to-read format and shows the data types of all the values being displayed. You may want to use the `<pre>` tag to format the result of the `var_dump` function.

TYPE THIS:

```
$info = array ("Tom", 24, 65.5);
print "<pre>";
var_dump($info);
print "</pre>";
```

RESULT:

```
array(3) {
  [0]=>
  string(3) "Tom"
  [1]=>
  int(24)
  [2]=>
  float(65.5)
}
```

GET INFORMATION ABOUT A VARIABLE

Untitled - Notepad

```
<html>
<head>
<title>Get Information About a Variable</title>
</head>
<body>

<?php
$name = "Martine";
$age;
$email = "";

if (is_string($name) == TRUE)
{
    print "Welcome $name.<br>";
}
else
{
    print "Welcome new visitor.<br>";
}

?>

</body>
</html>
```

1 Type the code that creates variables and assigns their values.

2 To check if a variable stores a value of a specific type, type the function you want to use followed by ().

3 Between the parentheses, type the name of the variable you want to check.

4 Type the code that uses the function.

Untitled - Notepad

```
<?php
$name = "Martine";
$age;
$email = "";

if (is_string($name) == TRUE)
{
    print "Welcome $name.<br>";
}
else
{
    print "Welcome new visitor.<br>";
}

if (isset($age) == TRUE)
{
    print "You are $age years old.<br>";
}
else
{
    print "Your age was not set.<br>";
}

?>
```

5 To check if a value has been assigned to a variable, type **isset()**.

6 Between the parentheses, type the name of the variable you want to check.

7 Type the code that uses the `isset` function.

Untitled - Notepad

```
    print "Welcome $name.<br>";
}
else
{
    print "Welcome new visitor.<br>";
}

if (isset($age) == TRUE)
{
    print "You are $age years old.<br>";
}
else
{
    print "Your age was not set.<br>";
}

if (empty($email) == TRUE)
{
    print "Your e-mail address is blank.<br>";
}
else
{
    print "Your e-mail address is $email.<br>";
}
?>
```

8 To check if a variable stores an empty or 0 value, type **empty()**.

9 Between the parentheses, type the name of the variable you want to check.

10 Type the code that uses the `empty` function.

Get Information About a Variable - Microsoft Internet Explorer

Welcome Martine.
Your age was not set.
Your e-mail address is blank.

11 Display the PHP page in a Web browser.

■ The Web browser displays the results of retrieving information about the variables.

ACCESS SYSTEM CONSTANTS

PHP comes with several built-in system constants that you can use in your scripts. A constant is a name that represents a value that does not change. Because PHP sets the value of a system constant, these constants do not have to be declared and can be used in every script you create.

You can also create your own constants. This allows you to set the value of the constant. For information about defining and using constants, see page 44.

System constants are usually used to store information about the environment in which a PHP script is running. For example, the `PHP_VERSION` constant stores the version of PHP being used on the computer and the `__FILE__` constant stores the filename of the script being processed.

To access the value of a system constant in a PHP page, you specify the name of the constant in the PHP code. Unlike variables, constants do not start with a dollar symbol, so PHP cannot determine when a constant is being used in a string. To ensure that a constant is evaluated correctly in a string, you must separate the constant from the rest of the string.

You should use all uppercase letters when specifying the name of a system constant. The constants `TRUE` and `FALSE`, however, are also commonly referenced using all lowercase letters. `TRUE` and `FALSE` are the most frequently used system constants. You can test the values of the `TRUE` or `FALSE` system constant by assigning the constant to a variable and using the variable in an `if` statement.

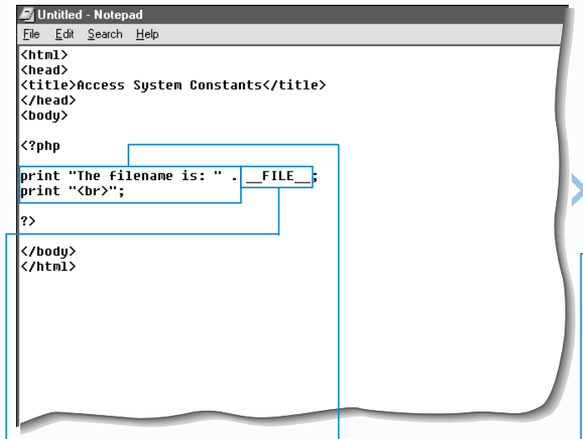
You cannot modify the value of a PHP system constant. You also cannot use the name of a system constant as the name or value of a constant you create. If you attempt to do either of these tasks in your script, the script will generate an error.

Extra

Common PHP System Constants

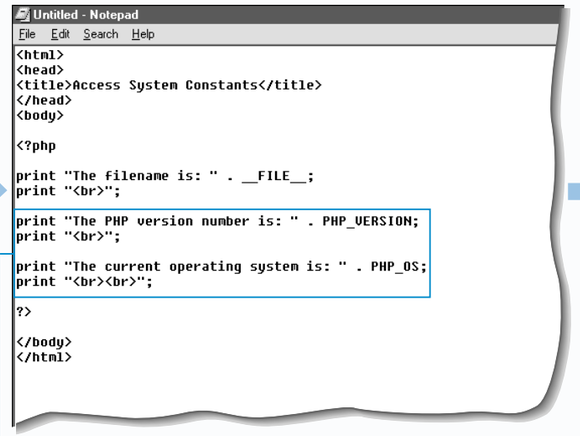
CONSTANT:	DESCRIPTION:
`__FILE__`	Returns the filename of the script being processed.
`__LINE__`	Returns the number of the line being processed in the script.
`PHP_VERSION`	Returns the version number of PHP being used.
`PHP_OS`	Returns the name of the operating system running PHP.
`TRUE`	Stores a value of true.
`FALSE`	Stores a value of false.
`E_ERROR`	Indicates a fatal error that is not related to parsing.
`E_WARNING`	Indicates that an error has occurred, but PHP will continue processing the script.
`E_PARSE`	Indicates a fatal parsing error caused by invalid syntax in the script.
`E_NOTICE`	Indicates that an error may have occurred, but PHP will continue processing the script.
`E_ALL`	Represents all the `E_*` constants. If used with the `error_reporting` function, all problems noticed by PHP will be reported.
`NULL`	Represents no value.

ACCESS SYSTEM CONSTANTS

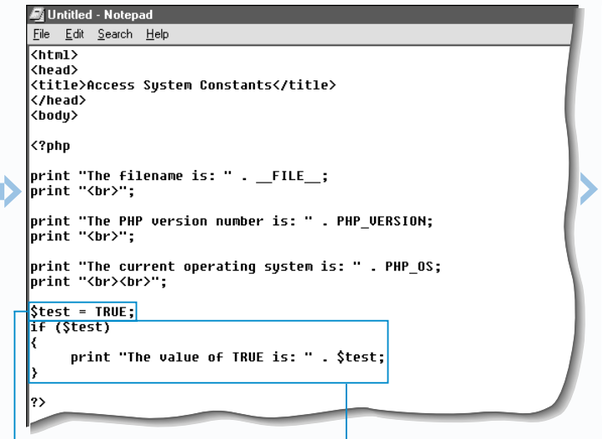


1 To access the value of a system constant, type the name of the constant.

2 Type the code that uses the value of the constant.

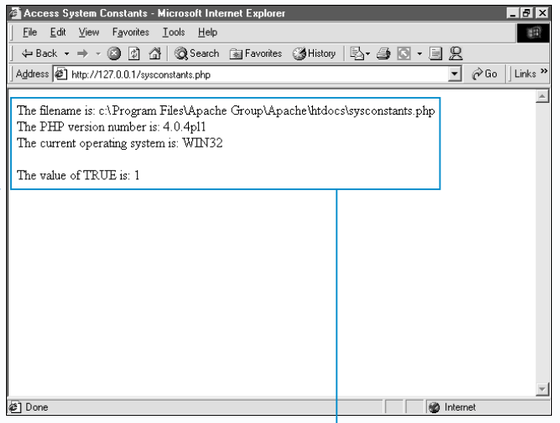


3 Repeat steps 1 and 2 for each system constant you want to access.



4 To test the value of a system constant, type the code that assigns the value of the constant to a variable.

5 Type the code that uses the value of the constant.



6 Display the PHP page in a Web browser.

■ The Web browser displays the result of accessing the values of system constants.

INCLUDE A FILE

The `require` and `include` statements allow you to use one file in several different PHP pages. This can save you time when you need to include the same information in multiple pages. For example, if you have a copyright notice you want to display on all your PHP pages, you can create a file that contains the copyright notice and then use a `require` or `include` statement to include the information on all your pages. If you change the code in the file, all the PHP pages that include the file will be updated.

You must first create the file you want to include. The file can contain plain text or HTML code, such as a table, header or footer. The file can also contain PHP code enclosed within the PHP delimiters (`<?php` and `?>`), such as variables or functions. You can save the file with the .inc extension on the Web server.

To include the file in a PHP page, you must add a `require` or `include` statement in the PHP page. The filename specified in the `require` or `include` statement must be enclosed in quotation marks. The contents of the required or included file can then be used in your script.

The `require` and `include` statements differ mainly in the way PHP processes each statement.

When a script is processed, PHP replaces instances of the `require` statement with the contents of the file to be included, even if the line containing the `require` statement is never executed. Unlike the `require` statement, the `include` statement is processed only when the statement is executed and returns a boolean value of true or false, depending on whether the statement is executed correctly. If you want to include a file based on a condition being true, you should use an `include` statement instead of a `require` statement.

Extra

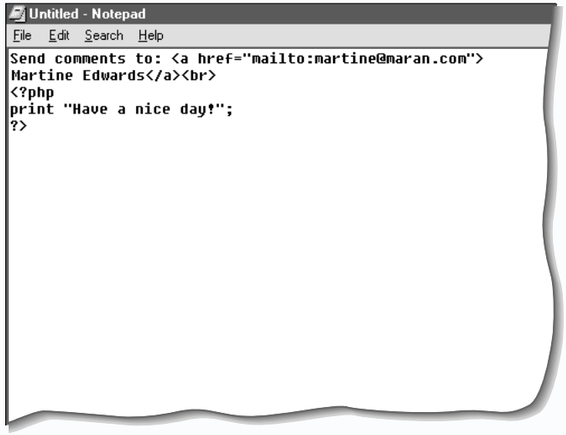
Using the `require` and `include` statements allows you to break code into manageable sections and then include the code in PHP pages as needed. For example, you can organize constants and functions you create into include files that you can then insert into your PHP pages. You can also nest include files within other include files. This is especially useful for complex PHP pages that contain long sections of code.

Some files should be included only once when a script is processed. For example, if a file with a function declaration has been included multiple times, an error will occur after the `require` or `include` statement is processed the first time, because a function can be declared only once in a script. To ensure that a file is included only once, you can use the `require_once` or `include_once` statement when you include the file in a PHP page. After the file has been included once, PHP will ignore any new statements that include the file.

Example:

```
<?php
require_once("hello.inc");
?>
```

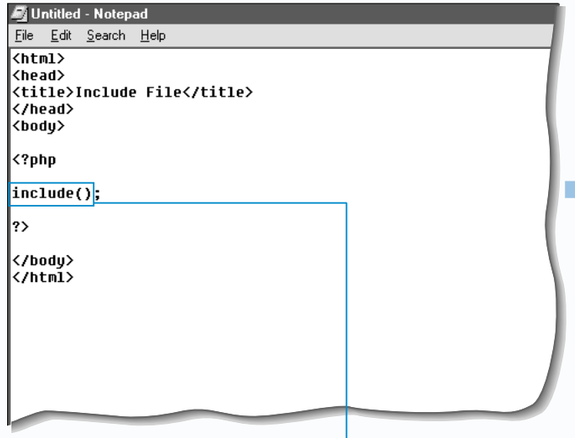
INCLUDE A FILE



CREATE A FILE TO INCLUDE

1 In a text editor, create a file you want to include in several PHP pages.

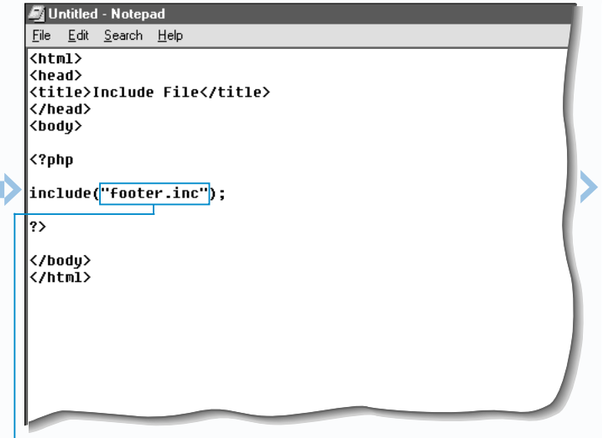
2 Save the file on the Web server.



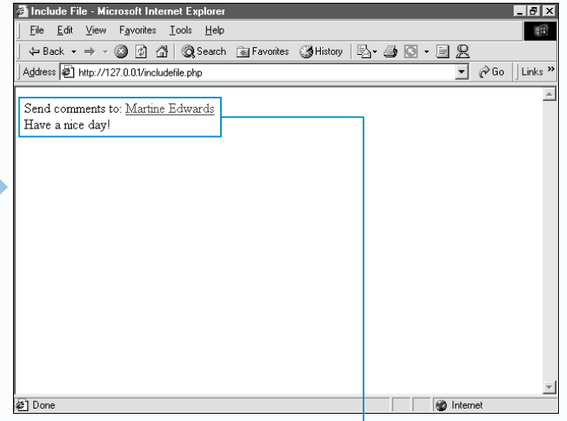
INCLUDE A FILE

1 Display the code for the PHP page in which you want to include a file.

2 Type **include()** or **require()** where you want to include a file.



3 Between the parentheses, type the name of the file you want to include, enclosed in quotation marks.



4 Display the PHP page in a Web browser.

■ The Web browser displays the result of including a file.