Using Variables and Expressions in PHP

Session 3

◆ *Explain the use of identifiers*

◆ *Explain the data types in PHP*

◆ *Explain the use of variables and constants*

◆ *Explain the scope of variables in PHP*

◆ *Explain the use of HTTP environment variables*

- ## A variable

  - Is a named memory location

  - Stores different types of data

  - Contains data that keep on changing during execution

- ## An expression is a combination of:

  - Variables

  - Constants

  - Functions

  - Operators

Identifiers

- Are names given to various elements of a program such as variables, constants, arrays, functions, and classes

- Rules to be followed while naming identifiers are as follows:
  - It must begin with a letter
  - It must contain only letters (A to Z) or digits (0-9)
  - It must not have any special characters including blank space
  - An underscore (_) character can be used to add space in the identifier to make it more readable

- Valid identifiers names are `firstnum`, `lname`, `net_sal`, `add8num`, and `NewNum`

- Variable is an identifier whose value keeps changing throughout the execution of a program

- Variable has:
  - **Name** - refers to the variable
  - **Data type** - refers to the type of data that the variable can store

- Variables are used to store:
  - User information
  - Intermediate data such as calculated results
  - Values returned by the functions

◆ Rules to be followed for a variable are as follows:

- ❖ Its not mandatory to declare a variable before assignment

- ❖ A variable is automatically declared at the time of initialization

- ❖ A variable is of the same data type as the value stored in it

- ❖ Variable name is preceded by a dollar sign ($) and can only contain alpha-numeric characters and underscores

- ❖ Value to the variable is assigned with the equal to (=) operator, with the variable on the left side and the expression on the right side

- ❖ A variable created without any value assigned to it, takes the default value of `NULL`

- ❖ A variable is case-sensitive

- ◆ PHP supports primitive data types that are categorized as follows:

  - ◆ Scalar Types

    - ◆ Integer

    - ◆ Float

    - ◆ String

    - ◆ Boolean

  - ◆ Compound Types

    - ◆ Array

    - ◆ Object

  - ◆ Special Types

    - ◆ Resource

    - ◆ Null

  - ◆ Constants

- ## Integer
  - Stores whole numbers without decimal points
  - Range from -2,147,483,648 to +2,147,483,647

- ## Float
  - Data type size is 8 bytes
  - Range from -2.2E–308 to 1.8E+308
  - Can include a decimal point, +/- sign, and an exponential value

- ## String
  - Is a sequence of characters
  - Is enclosed within single quotes or double quotes

- ## Boolean
  - Stores one of the two values, `True` or `False`

- ## Array

  - Stores multiple values in one single variable

  - Element can be accessed using its index

  - Are classified as follows:

    - **Numeric array** - an array with a numeric index

    - **Associative array** - an array where each ID key is associated with a value

    - **Multidimensional array** - an array containing one or more arrays

- ## Object

  - Is an instance of a user-defined class

  - Is used for any object reference

◆ Resource

  ◈ Hold references to resources external to PHP, such as:

    ◈ Database connections

    ◈ Database query

    ◈ Open file

    ◈ Other external types

◆ NULL

  ◈ Is a variable with `NULL` value

  ◈ A variable is considered to be `NULL` in following cases:

    ◈ Variable has been assigned the constant value `NULL`

    ◈ Variable has not been set to any value yet

    ◈ Variable has been `unset()`

Initializing a variable
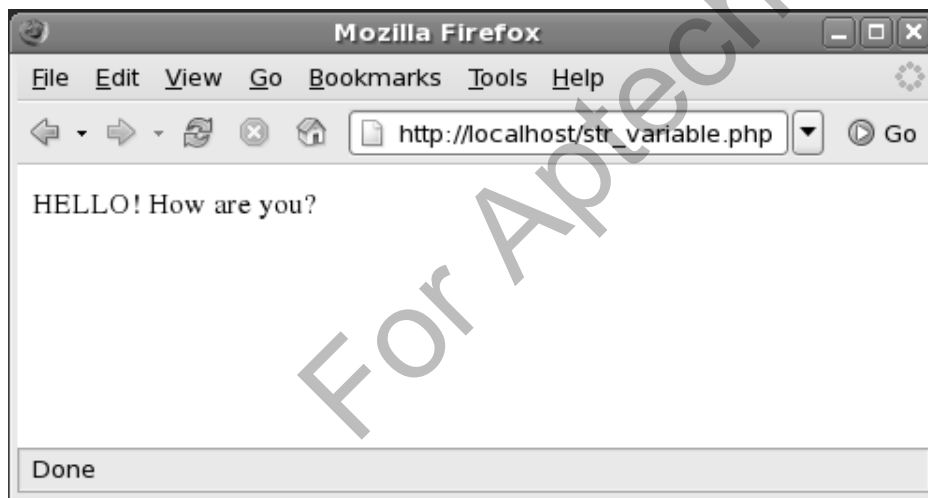
Syntax

```
$variable_name = value;
```

Where,

◈ **$variable_name** - specifies the name of the variable

◈ **value** - specifies the value the variable will store

◆ Storing a string value in a variable

◈ Enter the code and save it in a script named **str_variable.php**

Snippet

```php
<?php
$message = "HELLO! How are you?";
echo $message;
?>
```

Displays the following output:

In the code, the **$message** variable will be declared as the string variable because the value assigned to it is of string data type.
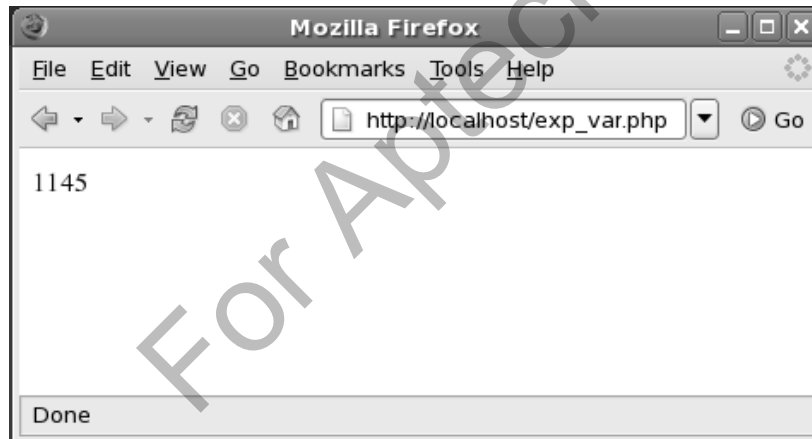
The string is enclosed within the double quotes.

◆ Storing the value of an expression in a variable

  ◈ Enter the following code and save in a script named **exp_var.php**

Snippet

```php
<?php
$number1 = 1019;
$number2 = 126;
$number3 = $number1 + $number2;
echo $number3;
?>
```

Displays the following output:

In the code, the `$number3` variable will be declared as an integer variable and the value of the addition expression `($number1 + $number2)` is assigned to it.

◆ To assign a value to a variable by referencing another variable

<div style="background: navy; color: white;">Syntax</div>

```
$new_varname = & $old_varname
```

Where,

◈ **$new_varname** - defines the new variable name

◈ **$old_varname** - specifies the other variable name whose value is to be assigned

◆ **Assigning the value of one variable to another variable**

  ◈ Enter the following code and save in a script named **exp_var.php**

Snippet

```php
<?php
$Fname = "John";
$Lname = "Smith";
$name =& $Fname;
echo $name;
echo "<br>";
echo $Lname;
echo "<br>";
?>
```

Displays the following output:
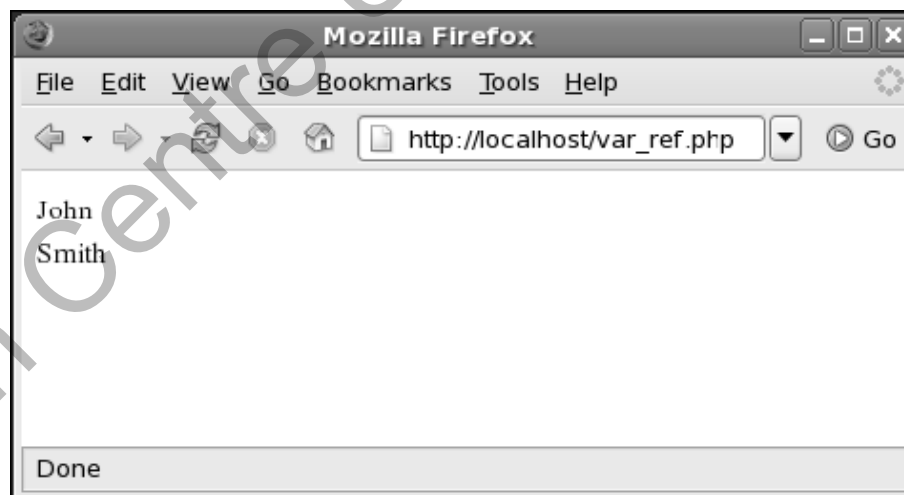
In the code, the variables, **$Fname and $Lname** are string variables.

The reference of the **$Fname** variable is assigned to the **$name** variable using the equal to and ampersand symbols.

◆ Constants

◈ Are identifiers containing values that do not change throughout the execution of a program

◈ Are case-sensitive

◈ Are static, meaning constants once defined cannot be changed

◈ Are declared using the `define()` function

◈ Are accessed from anywhere in the script

◆ **Declaring a Constant using** `define()` **function**

Syntax

```
define(string_name, mixed_value)
```

Where,

- ◈ **string_name -** defines the variable name for the constant

- ◈ **mixed_value -** specifies a numeric or string value

◆ Declaring the constant, **NAME**, containing a string value

  ◈ Enter the code as shown, in a script named **dec_con.php**

Snippet

```php
<?php
//Enable error reporting
error_reporting(-1);
define("NAME", "John Smith");
echo NAME;
echo "<br>";
echo name;
echo "<br>";
?>
```

Displays the following output:

```
Mozilla Firefox                           _ □ ✕
File  Edit  View  Go  Bookmarks  Tools  Help

◀ ▾  ▶ ▾  ⟳  ⊗  ⌂  [ http://localhost/dec_con.php  ] ▾  ▶ Go

John Smith

Notice: Use of undefined constant name - assumed 'name' in
/usr/local/apache2/htdocs/dec_con.php on line 6
name


Done
```

The code snippet declares a constant "**NAME**" and assigns a string value to it.

On executing the statement, `echo NAME`, the string value stored in the constant will be displayed.

The declaration of the constant is case sensitive. Therefore, the statement `echo name` displays the text name.

◆ Is the context within which a variable is defined

◆ Is the lifetime of a variable from the time the variable is created to the time the execution of the script ends

◆ Different scopes that a variable can have are as follows:

  ◈ Local

  ◈ Global

  ◈ Static
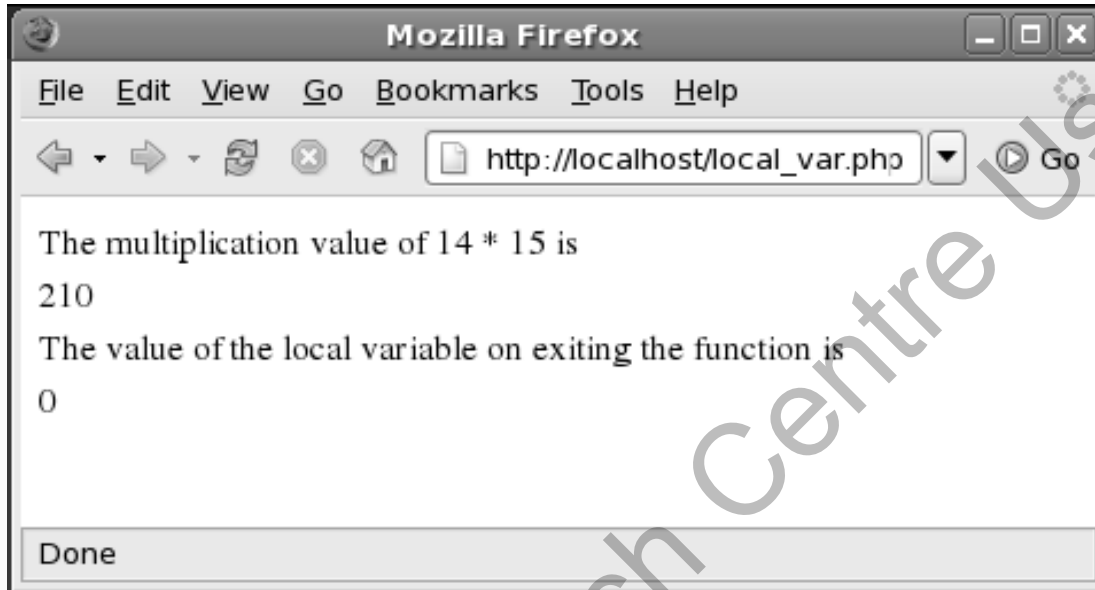
◆ Local variable

 ◈ Is a variable initialized and used inside a function

 ◈ Is similar to a normal variable

 ◈ Is declared inside a function

 ◈ Its lifetime begins when the function is called, and ends when the function is completed

◆ **Displaying the use of local variables**

Snippet

```php
<?php
$num2 = 0;
echo "The multiplication value of 14 * 15 is <br>";
function multiply()
{
$num1=14;
$num2=15;
$num2=$num1 * $num2;
echo $num2;
}
multiply();
echo " <br> The value of the local variable on exiting the function is
<br>";
echo $num2;
?>
```

Displays the following output:

The variable **num1** and **num2** are declared as local variables inside the **multiply()** function

The **multiply()** function is executed when PHP script invokes the function

- ## Global Variable

  - ◈ Is a variable retaining its value throughout with the lifetime of a Web page

  - ◈ Is declared within the function using global keyword

  - ◈ Is accessed from any part of the program

- ## Declaring a global variable

  <div style="background:navy;color:white;padding:4px;">Syntax</div>

  ```
  global $var_name;
  ```

  Where,

  - ◈ **$var_name** - defines the global variable name

◆ Displaying multiplication of two numbers using global variables

  ◈ Enter the code in a script named **multi_global.php**

Snippet

```php
<?php
$var1 = 4;
$var2 = 15;
function multiply()
{
global $var1, $var2;
$var2 = $var1 * $var2;
echo $var2;
}
echo "The multiplication value of 4 * 15 =";
multiply();
?>
```

Displays the following output:

The variables **var1** and **var2** are initialized outside the function and declared as global variables within the **multiply()** function

◆ Static Variables

- ◈ Retains its value even after the function terminates

- ◈ Are only accessible from within the function they are declared and their value remains intact between function calls

- ◈ Can be initialized during declaration and the static declarations are resolved during compile time

- ◈ Are commonly used in the recursive functions

Syntax

```
static $var_name = value;
```

Where,

- ◈ **var_name** - defines the variable name

- ◈ **value** - specifies the value of the variable

◆ Illustrating the use of a static variable

   ◈ Enter the code as shown in a script named **stat_var.php**

Snippet

```php
<?php
$var1;
function sum()
{
        static $var1 = 9;
        $var2 = $var1 + 12;
echo "The value of the variable is : $var1<br>";
echo "The addition value of 9 + 12 = ";
echo "$var2<br>";
}
sum();
?>
```

Displays the following output:
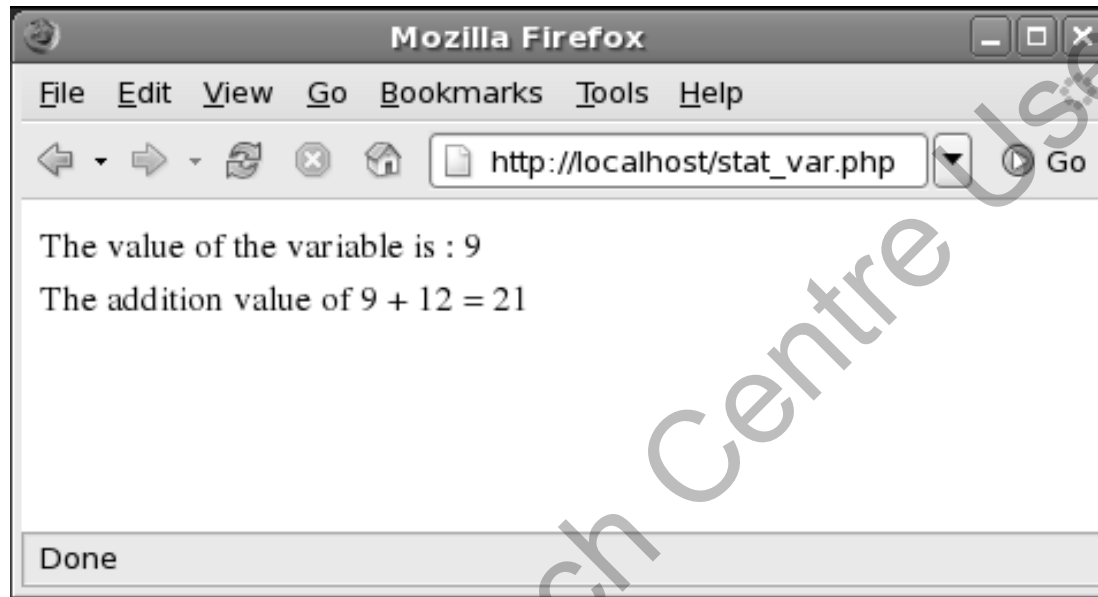
In the code, the first time the function **sum()** is called, the static variable **$var1** is set to zero, and incremented to display 1 as output.

The value of **$var1** is maintained for subsequent calls. Therefore, the next function call will increment the value of $var1 by one and print 2.

The variable, **var1** is declared as a static variable. The variable **var1** is local to the **sum()** function but retains its value throughout the program.

◆ Environment variables are:

  ◈ System-defined variables

  ◈ Similar to the user-defined variables and begin with dollar ($) sign

◆ Environment variables provide information about:

  ◈ Transactions between the client and the server

  ◈ HTTP request or response

◆ Environment variables are as follows:

- ◈ `SERVER_NAME`
- ◈ `SERVER_PROTOCOL`
- ◈ `SERVER_PORT`
- ◈ `$_COOKIE`
- ◈ `HTTP_USER_AGENT`
- ◈ `HTTP_ACCEPT`
- ◈ `HTTP_FROM`

SERVER_SOFTWARE

- Server identification string specified in the headers when responding to requests

- Returns the name and the version of the server software

```php
<?php
echo $_SERVER['SERVER_SOFTWARE'];
?>
```
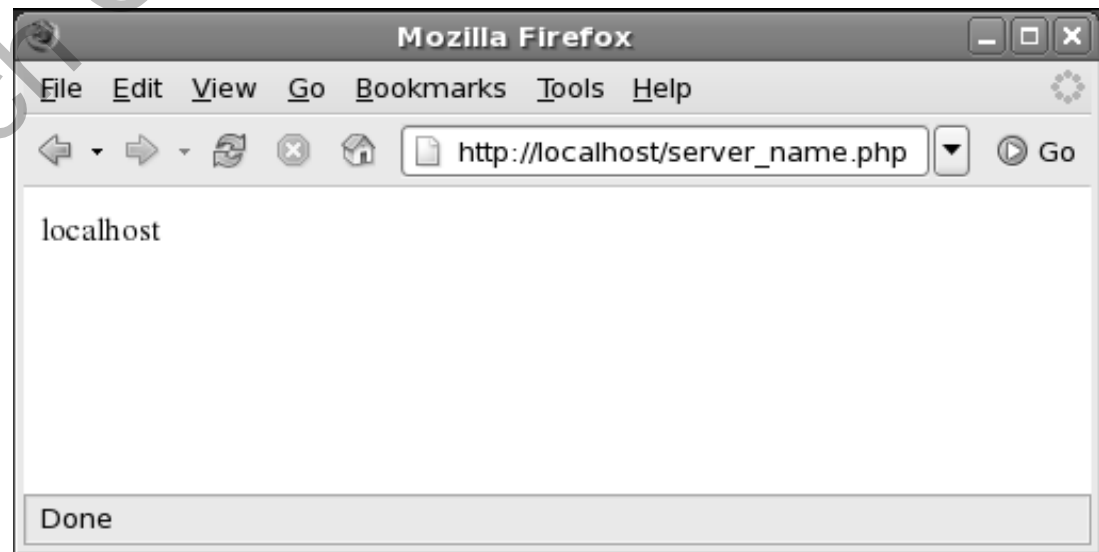
- Displays the following output:

SERVER_NAME

◆ Returns the name of the server host under which the current script is executing

◆ The host name can be:

◈ The Internet Protocol (IP) address or

◈ The Domain Name System (DNS) name of the server

Snippet
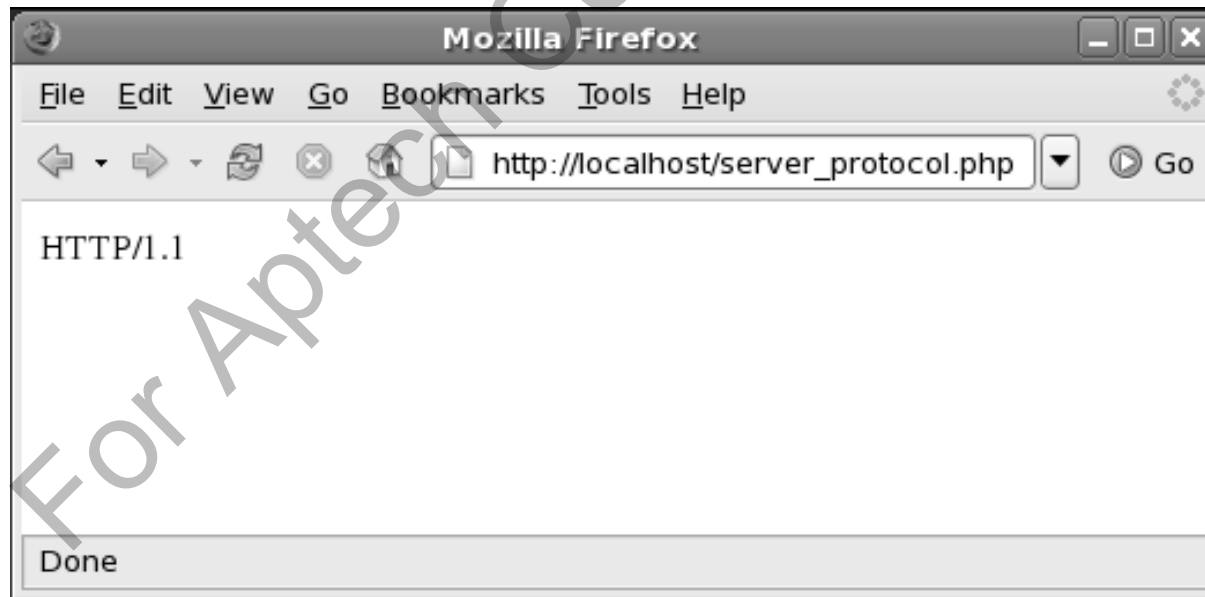
```php
<?php
echo
$_SERVER['SERVER_NAME'];
?>
```

Output:

- Returns the name and version number of the protocol via which the page was requested

Snippet

```php
<?php
echo $_SERVER['SERVER_PROTOCOL'];
?>
```

- Displays the following output:

◆ Returns the server port number for the script

Snippet

```php
<?php
echo $_SERVER['SERVER_PORT'];
?>
```
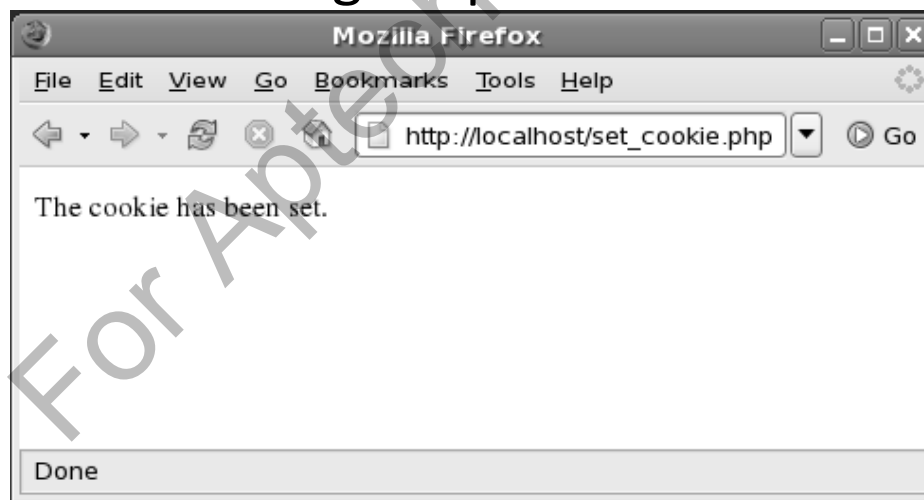
◆ Displays the following output:

◆ Returns the content of the recently used cookie which are:

◈ Saved as a text file

◈ Sent back to the server when the browser requests to display a page

Snippet

```php
<?php
$Month = 86400 + time();
setcookie('Name', 'Jerry', $Month);
echo "The cookie has been set.";
?>
```
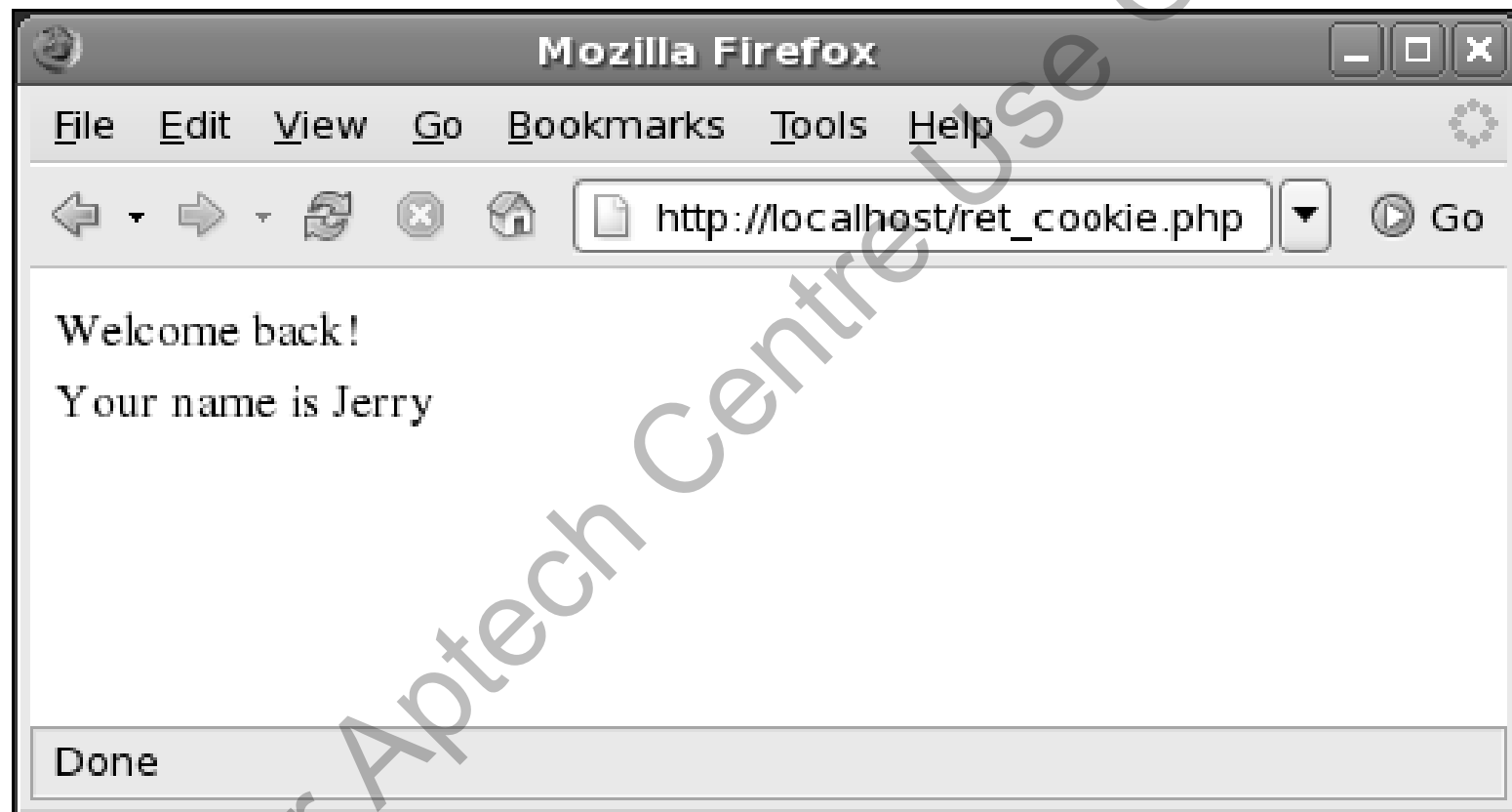
◆ Displays the following output:

◆ Retrieving the value of the cookie

Snippet

```php
<?php
if(isset($_COOKIE['Name']))
        {
        $last = $_COOKIE['Name'];
        echo "Welcome back! <br> Your name is ". $last;
        }
else
        {
        echo "Welcome to our site!";
        }
?>
```

Displays the following output:

Mozilla Firefox

File   Edit   View   Go   Bookmarks   Tools   Help

http://localhost/ret_cookie.php

Welcome back!
Your name is Jerry

Done

◆ The following code returns the name of the browser to the client:

Snippet

```php
<?php
echo $_SERVER['HTTP_USER_AGENT'];
?>
```

Displays the following output:

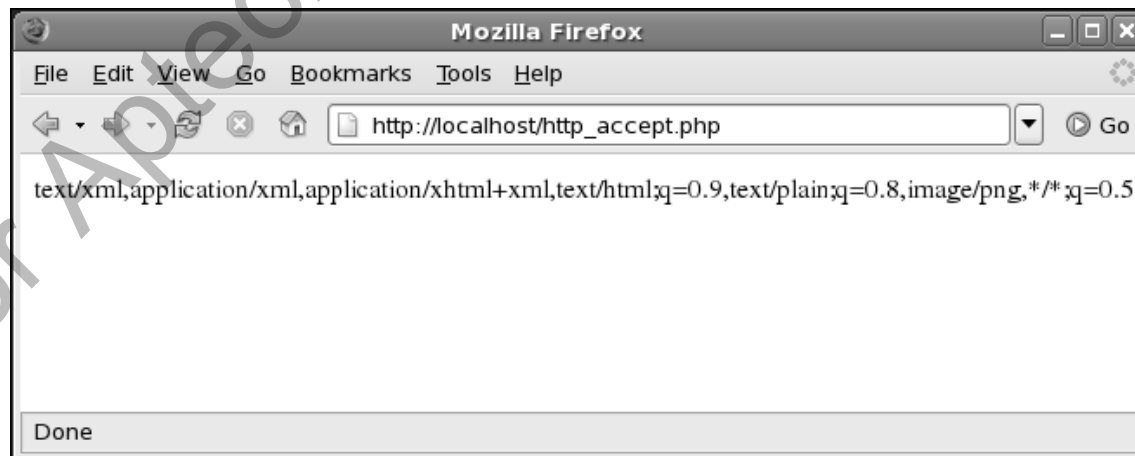- ◆ Returns the contents of the Accept: header if there is a current request

- ◆ Lists the media types the client will accept

Snippet

```php
<?php
echo $_SERVER['HTTP_ACCEPT'];
?>
```

Displays the following output:

- Identifiers are names given to different elements of a program such as, variables, constants, arrays, and classes

- A variable is an identifier whose value keeps changing. Variables are used to store data values. A dollar ($) symbol must be included before a variable name

- Constants are identifiers whose values do not change throughout the execution of a program. They are declared using the define() function

- The scope of a variable defines the availability of the variable in a program in which it is declared. The different scopes of a variable are local, global, and static

- A variable initialized and used inside a function is called a local variable. When a variable retains its value throughout the lifetime of the Web page, it is called the global variable. It is declared using the keyword global within the function

- The static variable retains its value even after the function terminates. It is declared using the keyword static with the variable name

- Environment variables are system-defined variables that can be used in any PHP script

- The $_COOKIE environment variable returns the content of the recently used cookie