

Introduction to PHP

Session 1

For Aptech Center Use Only



Objectives

- ◆ *Explain the history of PHP*
- ◆ *Identify the need for PHP*
- ◆ *Explain PHP tools and setup*
- ◆ *Explain a simple PHP script*
- ◆ *Explain User Input/Output (I/O)*
- ◆ *Explain the use of PHP to generate HTTP headers*
- ◆ *Describe passing of variables using Universal Resource Locator (URL)*

- ◆ PHP
 - ❖ Stands for Hypertext Preprocessor
 - ❖ Is an open source scripting language
 - ❖ Is used for developing dynamic Web pages
 - ❖ Is embedded in the HyperText Markup Language (HTML)
- ◆ PHP scripts are executed on the Web server

- ◆ 1994
 - ❖ PHP was created by Rasmus Lerdorf
 - ❖ Later, incorporated with Form Interpreter (FI) to create PHP/FI
 - ❖ PHP/FI enables:
 - ❖ Communication with database
 - ❖ Development of dynamic Web application
- ◆ 1997
 - ❖ PHP/FI 2.0 version released
 - ❖ Lack of features led to the development of PHP 3.0
 - ❖ PHP 3.0 provided support for:
 - ❖ Object oriented syntax
 - ❖ Different databases
 - ❖ Protocols
 - ❖ Application Programming Interfaces (APIs)

- ◆ 2000
 - ❖ PHP 4.0 version released
 - ❖ Features supported in PHP 4.0 are as follows:
 - ❖ Multiple Web servers
 - ❖ HTTP session
 - ❖ Output buffering
 - ❖ Security for user inputs
- ◆ 2004
 - ❖ PHP 5.0 version released
- ◆ 2011
 - ❖ Current version PHP 5.3.6 released

- ◆ Is a server-side scripting language
- ◆ Advantages are as follows:
 - ◆ Easy to learn, use, and implement
 - ◆ Freely available
 - ◆ Customizable
 - ◆ Executed on any Web server on any platform

- ◆ Uses of PHP are as follows:

- ◆ **Application Control** – is used to control access logging for HTTP servers
- ◆ **Database Access** – is used to read and write to any database using Structured Query Language (SQL) or Open Database Connectivity (ODBC)
- ◆ **File Access** – is used for file and directory maintenance, generate files in Portable Document File (PDF) and HTML formats, and to process eXtensible Markup Language (XML) data

- ❖ **Graphics** – is used to create graphics, charts, and generate images in GIF and PNS formats
- ❖ **Server-Side Scripting** – is used to implement server-side scripting using PHP parser, Web server and Web browser
- ❖ **Command Line Scripting** – is used to execute scripts on Unix/Linux platforms
- ❖ **Desktop Applications** – is used to create GUI-based desktop applications

- ◆ Are text editors
- ◆ Are used for developing and designing Web pages
- ◆ Are implemented after installation of PHP

- ◆ Are programming text editors that enable fast development of Web sites

Table lists tools used for developing dynamic Web sites

Tool	Description
PHPDebugger DBG	Enables step by step execution and debugging of a PHP script without changing the PHP code
ionCube Standalone PHP Encoder	Protects the PHP code and ensures security and runtime performance
Codelock	Enables you to encrypt both PHP and HTML code and protect Web pages
PHing	Is a PHP build system, which enables you to design your Web application in a structured manner
NuSphere PHPED	Is an Integrated Development Environment (IDE) for PHP and a complete platform for developing PHP based Web applications. It enables you to create, debug, profile, deploy, and integrate PHP code

Tool	Description
xored:WebStudio	Is an IDE for PHP. Built on Eclipse platform, this tool comprises a set of Eclipse editing, debugging, and deployment tools
PHPmole	Is a combination of Dreamweaver and Microsoft Visual Studio and runs on a GNOME platform to work with PHP
Simplewire PHP SMS Software Development Kit (SDK)	Enables to embed messaging services into an application, which can be sent to mobile devices
Quanta Plus Web Development Environment	Is a Web development environment to edit XML, HTML, PHP, and other text based Web documents
K PHP Develop	Is an integrated Web development tool with different modules
gedit	Is a GNOME based text editor for writing PHP scripts

- ◆ The installation of PHP requires:
 - ❖ A Web server
 - ❖ A database
- ◆ Web servers supported are as follows:
 - ❖ Internet Information Services (IIS)
 - ❖ Apache
 - ❖ Zeus
- ◆ Databases supported are as follows:
 - ❖ DB2
 - ❖ MSSQL
 - ❖ MySQL
 - ❖ Oracle
 - ❖ PostgreSQL

- ◆ To install PHP, perform the following steps:

Step 1

Download the `php-5.3.6.tar.gz` file from
<http://www.php.net/downloads.php>

Step 2

Right-click `php-5.3.6.tar.gz` files and select Extract Here. The contents are extracted to the `php-5.3.6` folder

Step 3

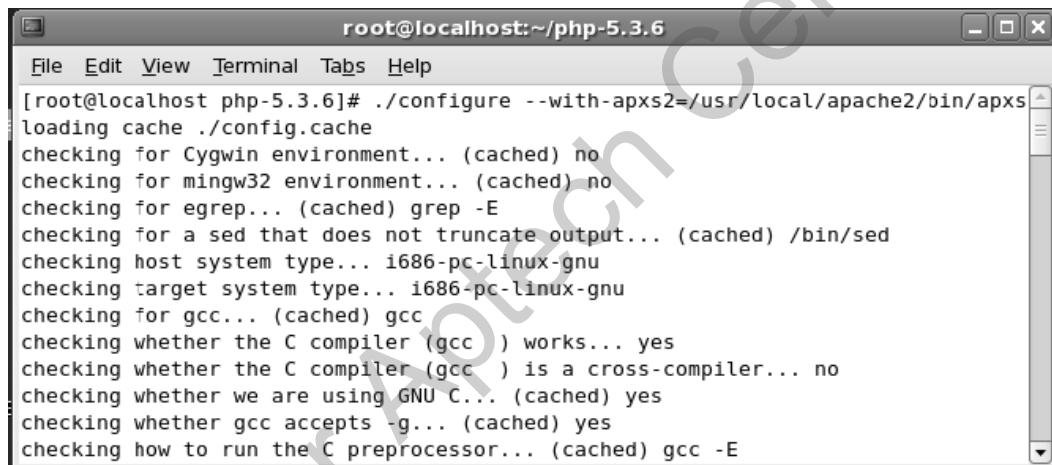
Right-click `php-5.3.6` folder and select Open In Terminal

Step 4

To configure the source code of PHP, enter the following at the command prompt:

```
./configure --with-apxs2=/usr/local/apache2/bin/apxs
```

Displays the following output:



The screenshot shows a terminal window titled "root@localhost:~/php-5.3.6". The window contains the following text output from the command ./configure --with-apxs2=/usr/local/apache2/bin/apxs:

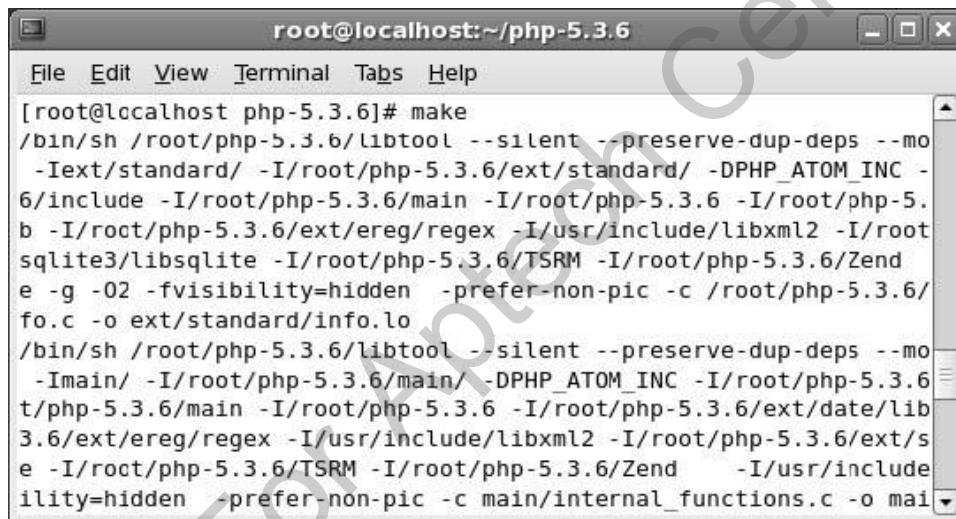
```
[root@localhost php-5.3.6]# ./configure --with-apxs2=/usr/local/apache2/bin/apxs
loading cache ./config.cache
checking for Cygwin environment... (cached) no
checking for mingw32 environment... (cached) no
checking for egrep... (cached) grep -E
checking for a sed that does not truncate output... (cached) /bin/sed
checking host system type... i686-pc-linux-gnu
checking target system type... i686-pc-linux-gnu
checking for gcc... (cached) gcc
checking whether the C compiler (gcc ) works... yes
checking whether the C compiler (gcc ) is a cross-compiler... no
checking whether we are using GNU C... (cached) yes
checking whether gcc accepts -g... (cached) yes
checking how to run the C preprocessor... (cached) gcc -E
```

Step 5

To build the compiled files, enter the following at the command prompt:

```
make
```

Displays the following output:



The screenshot shows a terminal window titled "root@localhost:~/php-5.3.6". The window contains the following text:

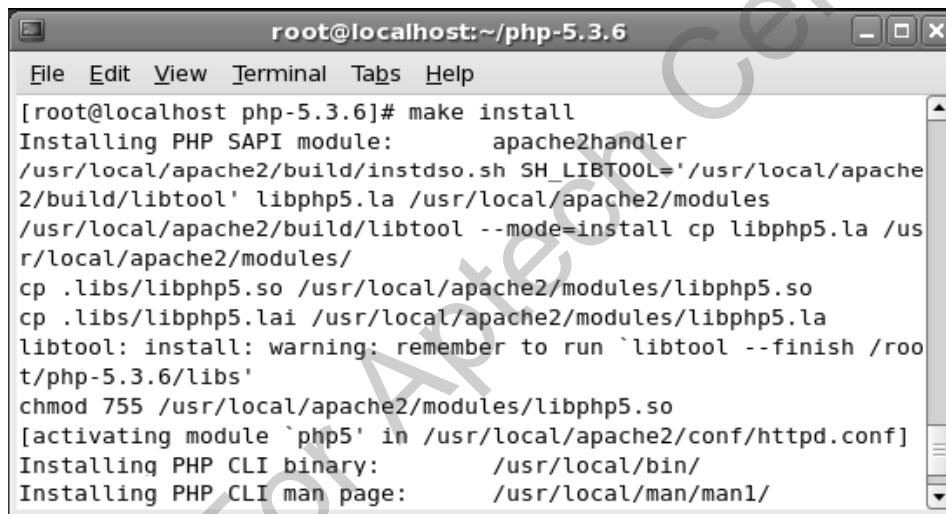
```
[root@localhost php-5.3.6]# make
/bin/sh /root/php-5.3.6/libtool --silent --preserve-dup-deps --mode=link -Iext/standard/ -I/root/php-5.3.6/ext/standard/ -DPHP_ATOM_INC -I6/include -I/root/php-5.3.6/main -I/root/php-5.3.6 -I/root/php-5.3.6/ext/ereg/regex -I/usr/include/libxml2 -I/root/php-5.3.6/ext/sqlite3/libsqlite -I/root/php-5.3.6/TSRM -I/root/php-5.3.6/Zend -e -g -O2 -fvisibility=hidden -prefer-non-pic -c /root/php-5.3.6/ext/standard/info.lo
/bin/sh /root/php-5.3.6/libtool --silent --preserve-dup-deps --mode=link -Imain/ -I/root/php-5.3.6/main/ -DPHP_ATOM_INC -I/root/php-5.3.6/main -I/root/php-5.3.6 -I/root/php-5.3.6/ext/date/lib -I/root/php-5.3.6/ext/ereg/regex -I/usr/include/libxml2 -I/root/php-5.3.6/ext/sqlite3 -I/root/php-5.3.6/TSRM -I/root/php-5.3.6/Zend -I/usr/include -fvisibility=hidden -prefer-non-pic -c main/internal_functions.c -o main
```

Step 6

To install PHP, enter the following at the command prompt:

```
make install
```

Displays the following output:



The screenshot shows a terminal window titled "root@localhost:~/php-5.3.6". The window contains the following text output from the "make install" command:

```
[root@localhost php-5.3.6]# make install
Installing PHP SAPI module:      apache2handler
/usr/local/apache2/build/instdso.sh SH_LIBTOOL='/usr/local/apache2/build/libtool' libphp5.la /usr/local/apache2/modules
/usr/local/apache2/build/libtool --mode=install cp libphp5.la /usr/local/apache2/modules/
cp .libs/libphp5.so /usr/local/apache2/modules/libphp5.so
cp .libs/libphp5.lai /usr/local/apache2/modules/libphp5.la
libtool: install: warning: remember to run `libtool --finish /root/php-5.3.6/libs'
chmod 755 /usr/local/apache2/modules/libphp5.so
[activating module `php5' in /usr/local/apache2/conf/httpd.conf]
Installing PHP CLI binary:      /usr/local/bin/
Installing PHP CLI man page:    /usr/local/man/man1/
```

Writing a Simple PHP Script

- ◆ Rules followed while creating PHP script are as follows:
 - ❖ Embed PHP scripts in the BODY tag of an HTML file
 - ❖ Start and end every block of PHP code with `<?php` and `?>` tags
 - ❖ End a PHP statement with a semicolon, `;`
 - ❖ Save all PHP files with a `.php` extension

Snippet

```
<html>
<body>
<title>PHP Syntax Example</title>
<?php
echo "Hello World";
?>
</body>
</html>
```

This snippet is saved in a file with a `.php` extension.

The `echo` command displays "Hello World" in the browser when executed.

Comments in a PHP Script

- ◆ Comments are:
 - ❖ Not displayed in the output
 - ❖ Used to assist a programmer to interpret the meaning of a code
- ◆ Comments supported in PHP are:
 - ❖ Single-line
 - ❖ Multi-line
- ◆ Demonstrating the use of comments in a PHP script

Snippet

```
<?php
// This is a single-line comment
/* and this is a
multi-line
comment */
?>
```

◆ Displaying current date using PHP script

- ◆ Open the gedit text editor
- ◆ Enter the following code snippet:

Snippet

```
<HTML>
<BODY>
The Date is:
<?php echo gmdate("M d Y");
?>
</BODY>
</HTML>
```

- ◆ Save the file as date.php in the /usr/local/apache2/htdocs directory
- ◆ Open the Mozilla Firefox Web browser and enter http://localhost/date.php in the Address bar

Displays the following output:



The `gdate()` function used in the code snippet displays the current date and time in the browser.

The `gdate("M d Y")` function takes three parameters to display the current date:

M – displays only three letters of the month

d – displays the current date

Y – displays four digits of the current year

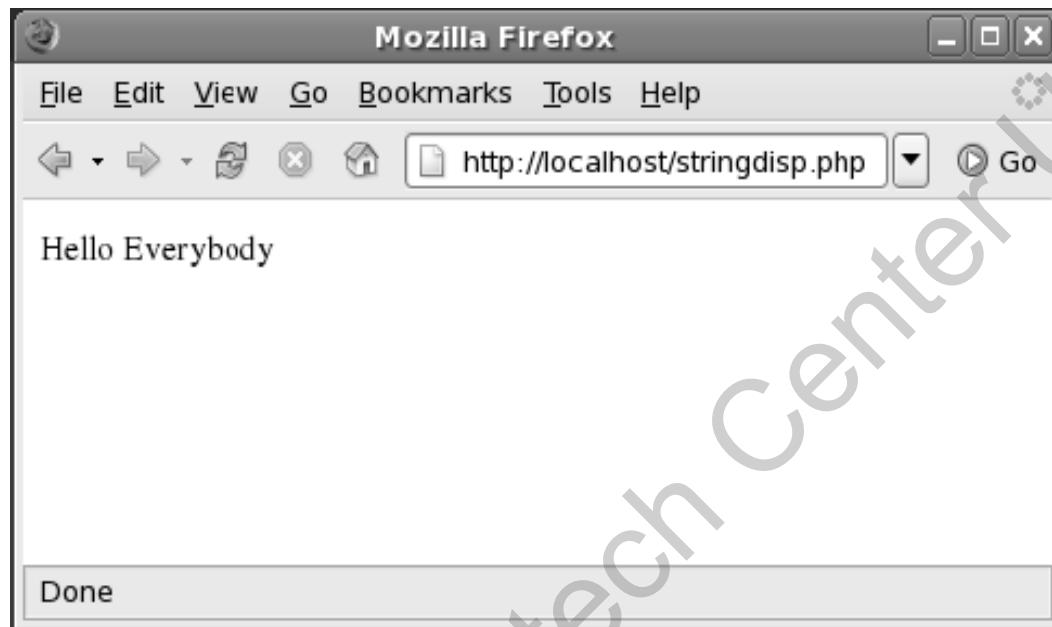
- ◆ Displaying a simple text in the browser using the PHP script
 - ❖ Open the gedit text editor
 - ❖ Enter the following code snippet:

Snippet

```
<HTML>
<BODY>
<?php echo "Hello Everybody";
?>
</BODY>
</HTML>
```

- ❖ Save the file as stringdisp.php in the /usr/local/apache2/htdocs directory
- ❖ Open the Mozilla Firefox Web browser and enter http://localhost/stringdisp.php in the Address bar

Displays the following output:



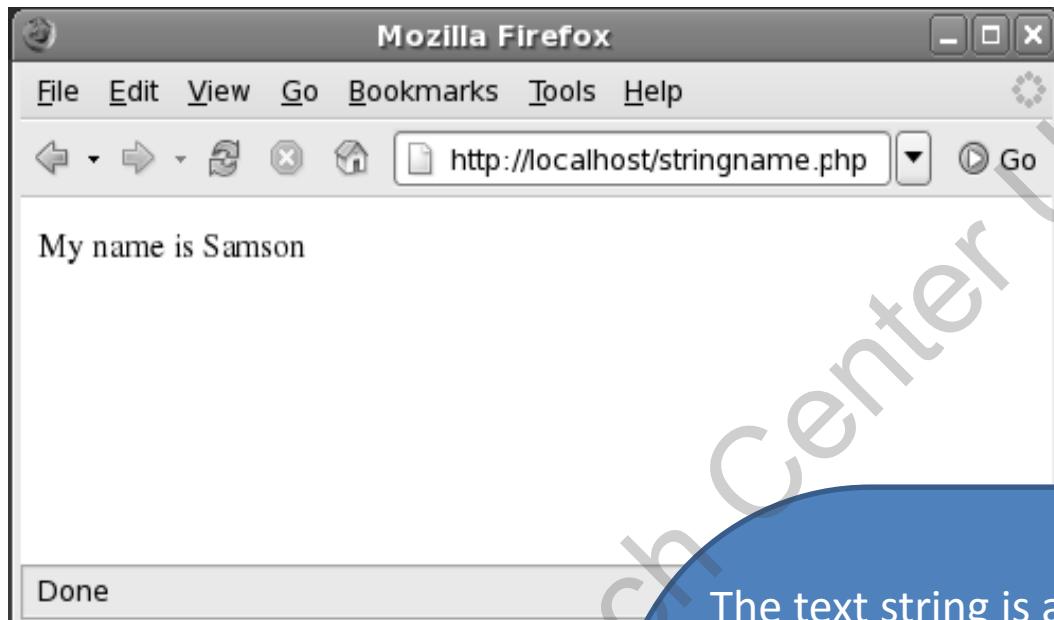
- ◆ Displaying a text in the browser using a variable
 - ❖ Open the gedit text editor
 - ❖ Enter the following code snippet:

Snippet

```
<HTML>
<BODY>
<?php
$str = "My name is Samson";
echo $str;
?>
</BODY>
</HTML>
```

- ❖ Save the file as stringname.php in the /usr/local/apache2/htdocs directory
- ❖ Open the Mozilla Firefox Web browser and enter http://localhost/stringname.php in the Address bar

Displays the following output:



The text string is assigned to a variable named `str`.

The second instruction substitutes the value or content of the variable `str` to the `echo` command.

The `echo` command displays the contents of the `str` variable in the output.

- ◆ It is a network transmission protocol
- ◆ It transfers hypertext files
- ◆ It provides instructions for communication between the client and the server
- ◆ It runs on the Transmission Control Protocol/Internet Protocol (TCP/IP) suite

- ◆ HTTP Header
 - ❖ Is an Internet protocol
 - ❖ Contains instructions to transfer information between a Web client and a Web server
- ◆ The instruction are of two types:
 - ❖ Request - sent by the client to the server
 - ❖ Response - from the server to a client request
- ◆ Format for request or response contains following components:
 - ❖ A request or a response line
 - ❖ HTTP header lines
 - ❖ A blank line
 - ❖ A message body, which is optional

- ◆ Format of an HTTP message is as follows:

Syntax

```
<initial line, different for request vs. response>
Header1: value1
Header2: value2
Header3: value3
Blank line
<optional message body, like file contents or query data>
```

Initial Request or Response Line

- ◆ Request line contains the following information separated by spaces:
 - ◆ An HTTP method name
 - ◆ Uniform Resource Identifier (URI)
 - ◆ The HTTP version being used

Snippet

```
GET /sample.html HTTP/1.1
```

- ◆ Response line consists of the following three components separated by spaces:
 - ◆ The HTTP version
 - ◆ A response code indicating the result of the request
 - ◆ An English phrase describing the response code

Snippet

```
HTTP/1.0 500 Internal Server Error
```

- ◆ Provide information about the request or response or the data sent in the message body

Syntax

```
Header-Name: value
```

- ◆ Categorized as follows:
 - ◆ General:
 - ◆ Control the processing of a message
 - ◆ Provide extra information to the receiver
 - ◆ Entity:
 - ◆ Provides information about the entity
 - ◆ Request or response:
 - ◆ Provides details about the client's request
 - ◆ Contains response header attached with the response sent by the server

- ◆ Displaying HTTP header lines

Snippet

```
GET /sample.html HTTP/1.1
User-agent: Mozilla/4.0
Last-Modified: Mon, 11 Apr 2011 23:07:07 GMT
Accept-Language: en
[ blank line above ]
```

Where,

- ◆ **GET** - specifies requested file name and the version of HTTP used
- ◆ **User-agent** - specifies the name of the browser and the version
- ◆ **Last-Modified** - specifies the date and time when the resource was last modified
- ◆ **Accept-Language** - specifies the language preference as English

The Message Body

- ◆ Is the third and an optional component of an HTTP header
- ◆ Appears after the header lines
- ◆ Messages can be of two types:
 - ◆ **Request Message** - contains user data and uploaded files
 - ◆ **Response Message** - contains requested resource

- ◆ **header () function:**

- ◆ Used to generate the HTTP headers
- ◆ Sends the HTTP commands to the server through HTTP protocols
- ◆ Displays a blank line showing that the header information is complete after the execution of the `header ()` function

Syntax

```
void header( string string [,bool replace [,int http_response_code]] )
```

Where,

- ◆ **string** – specifies the header string to be sent
- ◆ **replace** – is an optional parameter and indicates whether should be replaced or not
- ◆ **http_response_code** - is an optional parameter and forces the HTTP response code to the specified value

- ◆ Displaying an authentication header

Snippet

```
<?php  
header ('WWW-Authenticate: Negotiate') ;  
?>
```

Authentication helps to identify if a client is allowed to access to a resource.

Authentication is a means of negotiating access to a secure resource.

- ❖ Authentication schemes are as follows:
 - ❖ Http Basic Authentication
 - Sends an encoded string
 - Contains a user name and password
 - ❖ HTTP Digest Authentication
 - Is a challenge-response scheme
 - Server sends a data string to the client as a challenge
 - Client responds with a user name and password
 - ❖ NTLM
 - Is a challenge-response scheme
 - Uses Windows credentials to transform the challenge data
 - Requires multiple exchanges between the client and server
 - ❖ Negotiate
 - Selects between Kerberos and NTLM depending on their availability

- ◆ The `replace` option specifies to replace the previous header or add a second header to the document
- ◆ If `false`, then new header will be added to the document

Syntax

```
void header('string string', boolean replace)
```

Where,

- ◆ **string** - defines the authentication parameters
- ◆ **replace** - substitutes the existing header or adds new headers to the document. The default value is set to true, so all similar headers are replaced

- ◆ Displaying addition of multiple headers to the document

Snippet

```
<?php  
header ('WWW-Authenticate: Negotiate');  
header ('WWW-Authenticate: NTLM', false);  
?>
```

WWW-Authenticate - specifies the authentication string.

NTLM - specifies a challenge-response authentication mechanism.

false - defines the parameter of the replace option.

- ◆ Displays the response of the Web server for a request
- ◆ The request can include the status or the location of the client

Syntax

```
void header( string string, boolean replace, integer http_response_code )
```

Where,

- ◆ **string** - defines the authentication parameters
- ◆ **replace** - indicates whether previous defined headers need to be replaced or not
- ◆ **http_response_code** - forces the HTTP response code to the specified value

- ◆ Displaying a PHP script to redirect the user from one Web page or URL to another Web site

Snippet

```
header("Location: http://google.com");
```

Location is a type of HTTP header redirecting the browser to the specified URL.

- ◆ Displaying a PHP script with an HTTP response code

Snippet

```
header("Location: http://google.com", true, 303);
```

- ◆ Location - is an HTTP header that redirects the browser to the specified URL
- ◆ true - defines the parameter of the replace option
- ◆ 303 - is a redirection response code

Summary

- ◆ PHP is an open source scripting language embedded within HTML codes and used for developing dynamic Web pages
- ◆ PHP is used for executing scripts from the command line and for developing client-side GUI applications that are platform independent
- ◆ PHP is used for generating files in PDF and HTML formats
- ◆ PHP can generate e-mails by retrieving data from documents and sending it through any standard mail protocol

- ◆ PHP is used to render graphical images, such as GIF and PNG images
- ◆ PHP consists of tools for developing and designing Web pages. These tools are the program text editors. The popular text editor used for writing PHP scripts on Linux platform is gedit
- ◆ A PHP script starts with <?php tag and ends with the ?> tag. These scripts are embedded in the HTML tags
- ◆ A HTTP message or protocol is divided into three parts, the request or response line, the HTTP header, and the body of the protocol

Installing and Configuring PHP 7

Session 2

For Aptech Center Use Only



Objectives

- ◆ *Explain the pre-requisites for installing PHP 7.*
- ◆ *Describe the steps to configure PHP 7.*
- ◆ *Identify the steps to install PHP 7.*
- ◆ *Describe the process to create simple PHP scripts.*
- ◆ *Explain how to use HTTP headers in PHP.*

- ◆ Installation of PHP requires:

A Web server

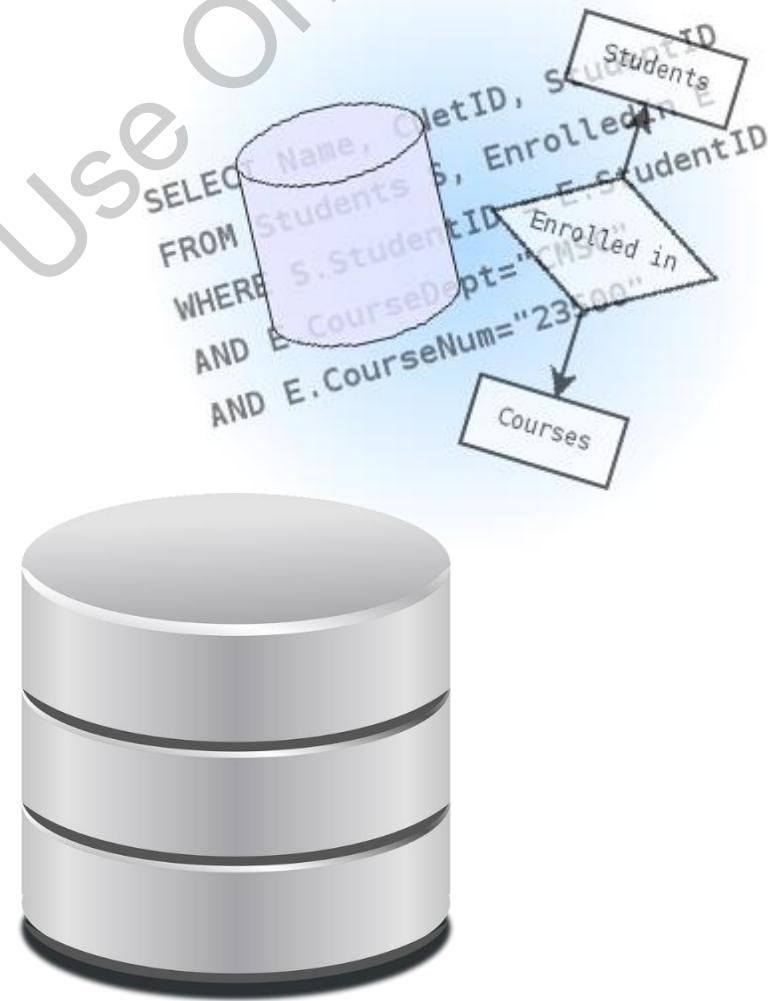
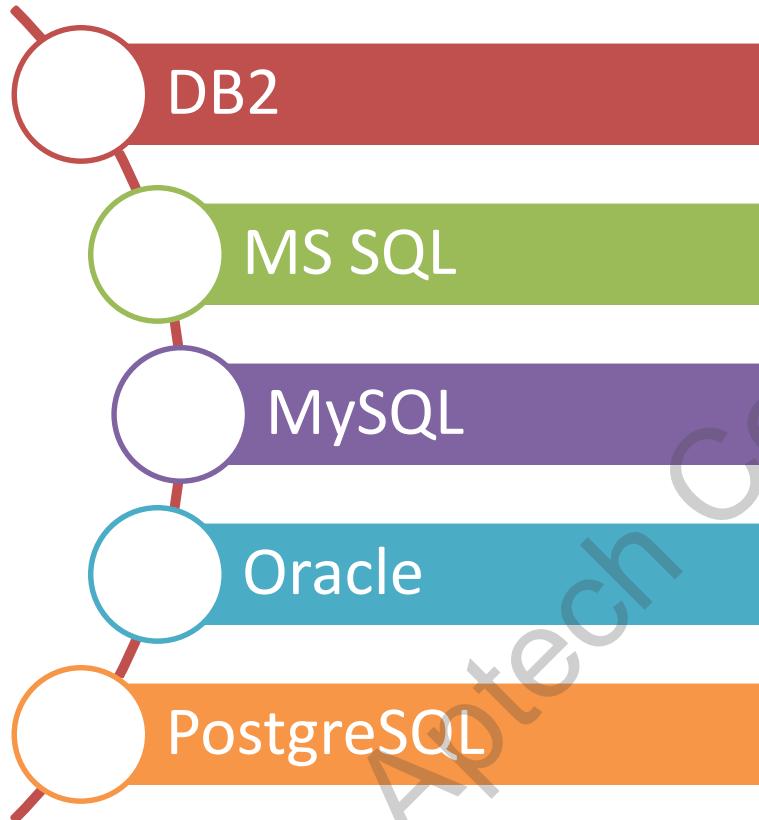
A database



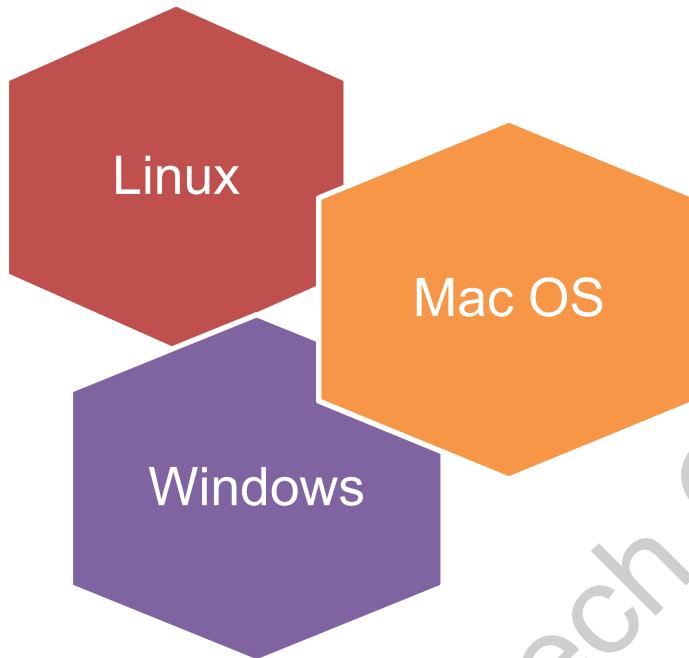
- ◆ Web servers supported are as follows:
 - ❖ Internet Information Services (IIS)
 - ❖ Apache
 - ❖ Nginx



- Databases supported are as follows:



- ◆ PHP can be installed on:



- ◆ Download the relevant package from
<http://php.net/downloads.php> Website.

- ◆ To extract PHP packages on Linux, perform these steps:
 - ❖ Enter the following command at the command prompt:

```
# tar xjvf php-7.0.4.tar.gz  
# tar -xf php-7.0.4.tar  
# cd php-7.0.4/
```

- ❖ To install the development package, enter the following command at the command prompt:

```
# dnf install aspell-devel bzip2-devel freetype-devel gmp-devel libXpm-devel libcurl-devel libjpeg-turbo-devel libmcrypt-devel libpng-devel libxml2-devel libxslt-devel mariadb-devel recode-devel uw-imap-devel gcc openssl-devel -y
```

- ◆ To configure PHP 7.0.4, perform the following steps:
 - ❖ Enter the following command at the command prompt:

```
#cd php-7.0.4
# ./configure --prefix=/usr/local/php7 --with-zlib-
dir --with-freetype-dir --enable-mbstring --with-
libxml-dir=/usr --enable-soap --enable-calendar --
with-curl --with-mcrypt --with-zlib --with-gd --
disable-rpath --enable-inline-optimization --with-bz2
--with-zlib --enable-sockets --enable-sysvsem --
enable-sysvshm --enable-pcntl --enable-mbregex --
enable-exif --enable-bcmath --with-mhash --enable-zip
--with-pcre-regex --with-mysqli --with-pdo-mysql --
with-mysqli --with-jpeg-dir=/usr --with-png-dir=/usr
--enable-gd-native-ttf --with-openssl --with-fpm-
user=apache --with-fpm-group=apache --with-
libdir=/usr/lib --enable-ftp --with-kerberos --with-
gettext --with-xmlrpc --with-xsl --enable-opcache --
enable-fpm
```

Displays the following output:

```
root@localhost:~/php-7.0.4
File Edit View Search Terminal Help
[root@localhost php-7.0.4]# ./configure --prefix=/usr/local/php7 --with-zlib-dir
--with-freetype-dir --enable-mbstring --with-libxml-dir=/usr --enable-soap --en
able-calendar --with-curl --with-mcrypt --with-zlib --with-gd --disable-rpath --
enable-inline-optimization --with-bz2 --with-zlib --enable-sockets --enable-sysv
sem --enable-sysvshm --enable-pcntl --enable-mbregex --enable-exif --enable-bcma
th --with-mhash --enable-zip --with-pcre-regex --with-mysqli --with-pdo-mysql --
with-mysqli --with-jpeg-dir=/usr --with-png-dir=/usr --enable-gd-native-ttf --wi
th-openssl --with-fpm-user=apache --with-fpm-group=apache --with-libdir=/usr/lib
--enable-ftp --with-kerberos --with-gettext --with-xmlrpc --with-xsl --enable-o
pcache --enable-fpm
checking for grep that handles long lines and -e... /usr/bin/grep
checking for egrep... /usr/bin/grep -E
checking for a sed that does not truncate output... /usr/bin/sed
checking build system type... x86_64-unknown-linux-gnu
checking host system type... x86_64-unknown-linux-gnu
checking target system type... x86_64-unknown-linux-gnu
checking for cc... cc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether cc accepts -g... yes
```

Step 1

To extract the packages, install the pre-requisites, and configure the PHP files, enter the following command at the command prompt:

```
#make
```

Displays the following output:

```
root@localhost:~/php-7.0.4
File Edit View Search Terminal Help
[root@localhost php-7.0.4]# make
/bin/sh /root/php-7.0.4/libtool --silent --preserve-dup-deps --mode=compile cc -
DZEND_ENABLE_STATIC_TSRLMS_CACHE=1 -Iext/opcode/ -I/root/php-7.0.4/ext/opcode/
-DPHP_ATOM_INC -I/root/php-7.0.4/include -I/root/php-7.0.4/main -I/root/php-7.0
.4 -I/root/php-7.0.4/ext/date/lib -I/usr/include/libxml2 -I/usr/include/freetype
2 -I/root/php-7.0.4/ext/mbstring/oniguruma -I/root/php-7.0.4/ext/mbstring/libmbf
l -I/root/php-7.0.4/ext/mbstring/libmbfl/libmbfl -I/root/php-7.0.4/ext/sqlite3/libs
qlite -I/root/php-7.0.4/ext/zip/lib -I/root/php-7.0.4/TSRM -I/root/php-7.0.4/Zen
d -I/usr/include -g -O2 -fvisibility=hidden -c /root/php-7.0.4/ext/opcode
/ZendAccelerator.c -o ext/opcode/ZendAccelerator.lo
/bin/sh /root/php-7.0.4/libtool --silent --preserve-dup-deps --mode=compile cc -
DZEND_ENABLE_STATIC_TSRLMS_CACHE=1 -Iext/opcode/ -I/root/php-7.0.4/ext/opcode/
-DPHP_ATOM_INC -I/root/php-7.0.4/include -I/root/php-7.0.4/main -I/root/php-7.0
.4 -I/root/php-7.0.4/ext/date/lib -I/usr/include/libxml2 -I/usr/include/freetype
2 -I/root/php-7.0.4/ext/mbstring/oniguruma -I/root/php-7.0.4/ext/mbstring/libmbf
l -I/root/php-7.0.4/ext/mbstring/libmbfl/libmbfl -I/root/php-7.0.4/ext/sqlite3/libs
qlite -I/root/php-7.0.4/ext/zip/lib -I/root/php-7.0.4/TSRM -I/root/php-7.0.4/Zen
d -I/usr/include -g -O2 -fvisibility=hidden -c /root/php-7.0.4/ext/opcode
/zend_accelerator_blacklist.c -o ext/opcode/zend_accelerator_blacklist.lo
/bin/sh /root/php-7.0.4/libtool --silent --preserve-dup-deps
DZEND_ENABLE_STATIC_TSRLMS_CACHE=1 -Iext/opcode/ -I/root/p
-DPHP_ATOM_INC -I/root/php-7.0.4/include -I/root/php-7.0
.4 -I/root/php-7.0.4/ext/date/lib -I/usr/include/libxml2
2 -I/root/php-7.0.4/ext/mbstring/oniguruma -I/root/php-7
```

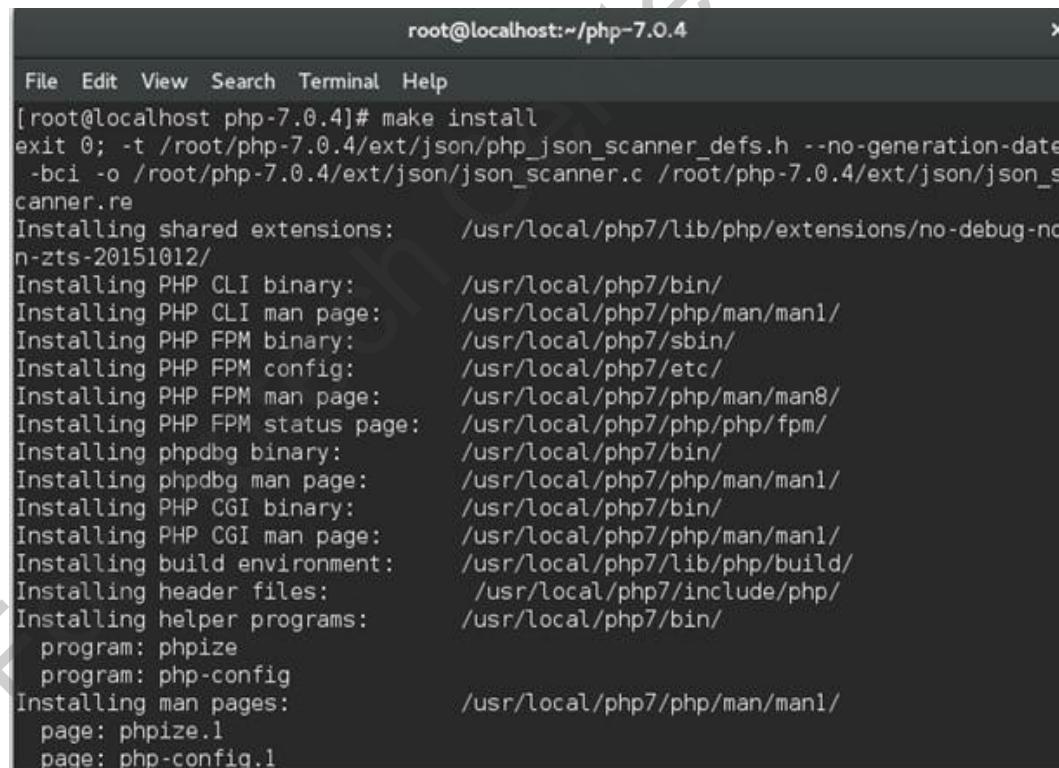
This command determines large files and issues commands to recompile those large files.

Step 2

To install PHP, enter the following command at the command prompt:

```
# make install
```

Displays the following output:



The screenshot shows a terminal window titled "root@localhost:~/php-7.0.4". The window displays the output of the "make install" command. The output lists various files and directories being installed, such as shared extensions, CLI binary, FPM binary, CGI binary, and man pages. The path for most installations is "/usr/local/php7".

```
[root@localhost php-7.0.4]# make install
exit 0; -t /root/php-7.0.4/ext/json/php_json_scanner_defs.h --no-generation-date
-bci -o /root/php-7.0.4/ext/json/json_scanner.c /root/php-7.0.4/ext/json/json_s
canner.re
Installing shared extensions:      /usr/local/php7/lib/php/extensions/no-debug-no
n-zts-2015012/
Installing PHP CLI binary:        /usr/local/php7/bin/
Installing PHP CLI man page:      /usr/local/php7/php/man/man1/
Installing PHP FPM binary:        /usr/local/php7/sbin/
Installing PHP FPM config:        /usr/local/php7/etc/
Installing PHP FPM man page:      /usr/local/php7/php/man/man8/
Installing PHP FPM status page:   /usr/local/php7/php/php/fpm/
Installing phpdbg binary:         /usr/local/php7/bin/
Installing phpdbg man page:       /usr/local/php7/php/man/man1/
Installing PHP CGI binary:        /usr/local/php7/bin/
Installing PHP CGI man page:     /usr/local/php7/php/man/man1/
Installing build environment:    /usr/local/php7/lib/php/build/
Installing header files:          /usr/local/php7/include/php/
Installing helper programs:
  program: phpz
  program: php-config
Installing man pages:             /usr/local/php7/php/man/man1/
  page: phpz.1
  page: php-config.1
```

Setting up Apache to Use PHP

Step 1

Open the `httpd.conf` file.

Step 2

Add the following directives in the file:

```
AddHandler application/x-httpd-php .php  
LoadModule php7_module C:\php7.dll  
AddType application/x-httpd-php .php  
PHPIniDir C:\php
```

Step 3

If required, change the path of the PHP installation folder.

Step 4

Save and restart the Apache Web server.

Writing a Simple PHP Script

- ◆ Rules followed while writing PHP script are as follows:
 - ❖ Embed PHP scripts in the BODY tag of an HTML file
 - ❖ Start and end every block of PHP code with `<?php` and `?>` tags
 - ❖ End a PHP statement with a semicolon, `;`
 - ❖ Save all PHP files with a `.php` extension

Snippet

```
<html>
<body>
<title>PHP Syntax Example</title>
<?php
echo "Hello World";
?>
</body>
</html>
```

This snippet is saved in a file with a `.php` extension.

The `echo` command displays "Hello World" in the browser when executed.

- ◆ Comments are:
 - ❖ Not displayed in the output
 - ❖ Used to assist a programmer to interpret the meaning of a code
- ◆ Comments supported in PHP are:
 - ❖ Single-line
 - ❖ Multi-line
- ◆ Demonstrating the use of comments in a PHP script

Snippet

```
<?php
// This is a single-line comment
/* and this is a
multi-line
comment */
?>
```

◆ Displaying current date using PHP script

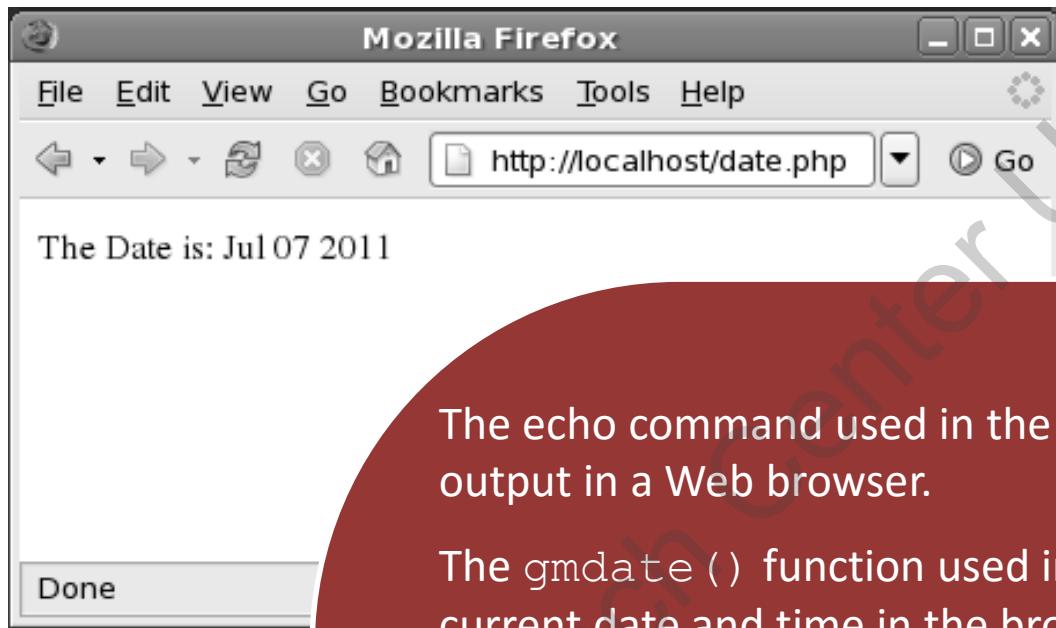
- ◆ Open the gedit text editor
- ◆ Enter the following code snippet:

Snippet

```
<HTML>
<BODY>
The Date is:
<?php echo gmdate("M d Y");
?>
</BODY>
</HTML>
```

- ◆ Save the file as date.php in the /usr/local/apache2/htdocs directory
- ◆ Open Mozilla Firefox Web browser and enter http://localhost/date.php in the Address bar

Displays the following output:



The echo command used in the code is used to display the output in a Web browser.

The gdate () function used in the code snippet displays the current date and time in the browser.

The gdate ("M d Y") function takes three parameters to display the current date:

M – displays only three letters of the month

d – displays the current date

Y – displays four digits of the current year

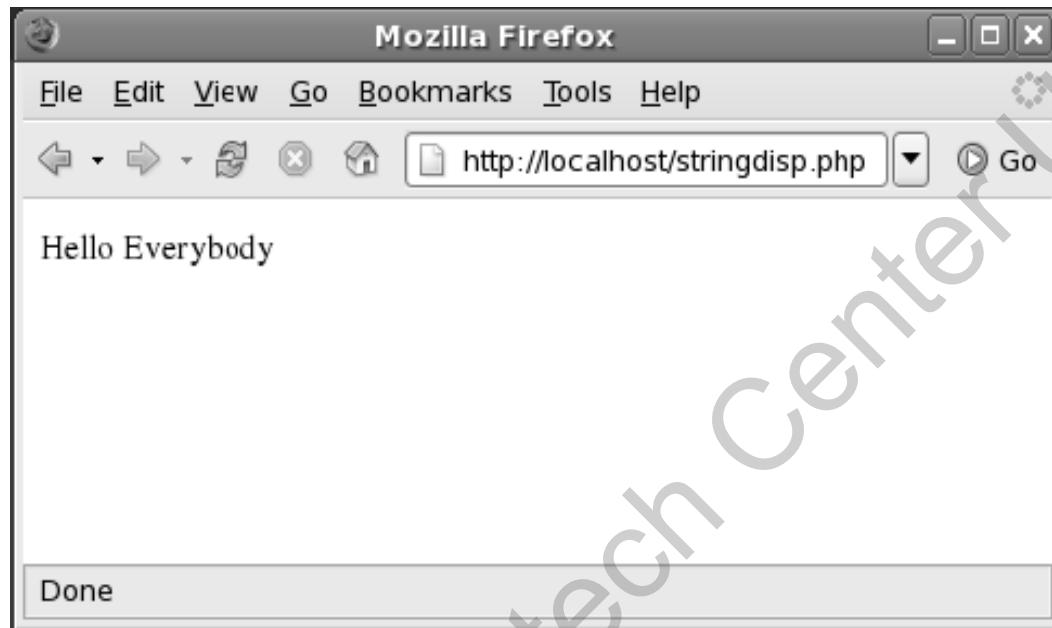
- ◆ Displaying a simple text in the browser using the PHP script
 - ❖ Open the gedit text editor
 - ❖ Enter the following code snippet:

Snippet

```
<HTML>
<BODY>
<?php echo "Hello Everybody";
?>
</BODY>
</HTML>
```

- ❖ Save the file as `stringdisp.php` in the `/usr/local/apache2/htdocs` directory
- ❖ Open Mozilla Firefox Web browser and enter `http://localhost/stringdisp.php` in the Address bar

Displays the following output:



- ◆ Rules followed while using a variable in a PHP script are as follows:
 - ◆ Variables:
 - ◆ Must start with a dollar sign '\$'
 - ◆ Can contain strings, numbers, and arrays
 - ◆ Variables names:
 - ◆ Must start with a letter or an underscore '_'
 - ◆ Can only contain alpha-numeric characters and underscores without spaces

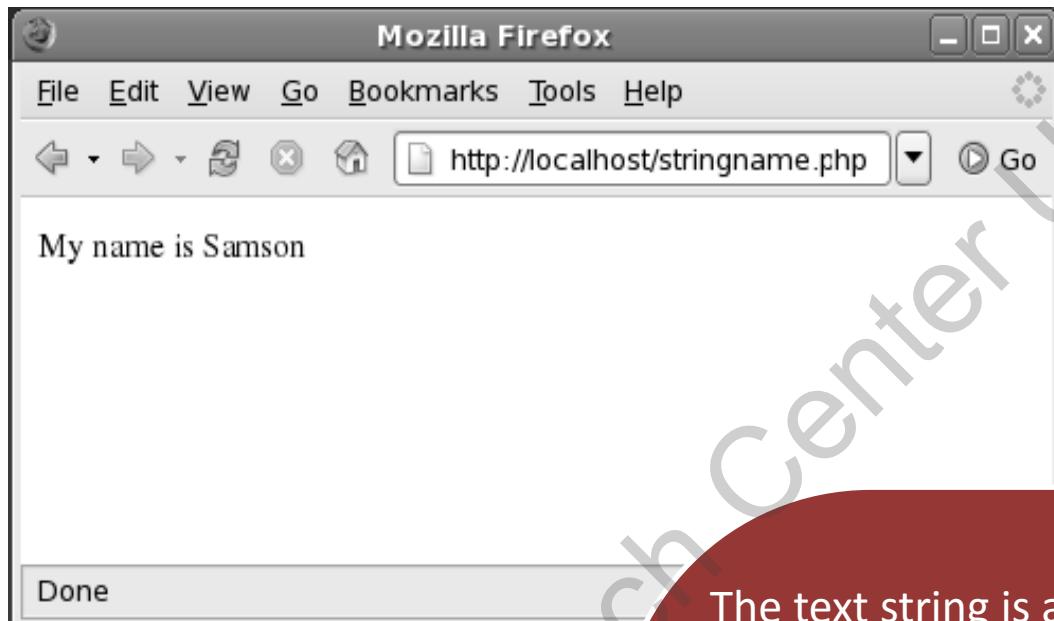
- ◆ Displaying a text in the browser using a variable:
 - ❖ Open the gedit text editor
 - ❖ Enter the following code snippet:

Snippet

```
<HTML>
<BODY>
<?php
$str = "My name is Samson";
echo $str;
?>
</BODY>
</HTML>
```

- ❖ Save the file as stringname.php in the /usr/local/apache2/htdocs directory
- ❖ Open the Mozilla Firefox Web browser and enter http://localhost/stringname.php in the Address bar

Displays the following output:



The text string is assigned to a variable named `str`.

The second instruction substitutes the value or content of the variable `str` to the `echo` command.

The `echo` command displays the contents of the `str` variable in the output.

- ◆ **header () function:**

- ◆ Used to generate HTTP headers
- ◆ Sends HTTP commands to the server through HTTP protocols
- ◆ Displays a blank line showing that the header information is complete after the execution of the `header ()` function

Syntax

```
void header( string [,bool replace [,int http_response_code]] )
```

where,

- ◆ **string** – specifies the header string to be sent
- ◆ **replace** – is an optional parameter and indicates whether it should be replaced or not
- ◆ **http_response_code** - is an optional parameter and forces the HTTP response code to the specified value

- ◆ Displaying an authentication header

Snippet

```
<?php  
header( 'WWW-Authenticate: Negotiate' );  
?>
```

Authentication helps to identify if a client is allowed to access to a resource.

Authentication is a means of negotiating access to a secure resource.

- ❖ Authentication schemes are as follows:
 - ❖ Http Basic Authentication
 - Sends an encoded string
 - Contains a user name and password
 - ❖ HTTP Digest Authentication
 - Is a challenge-response scheme
 - Server sends a data string to the client as a challenge
 - Client responds with a user name and password
 - ❖ NTLM
 - Is a challenge-response scheme
 - Uses Windows credentials to transform the challenge data
 - Requires multiple exchanges between the client and server
 - ❖ Negotiate
 - Selects between Kerberos and NTLM depending on their availability

- ◆ The `replace` option specifies to replace the previous header or add a second header to the document
- ◆ If `false`, then new header will be added to the document

Syntax

```
void header('string', boolean replace)
```

where,

- ◆ **string** - defines the authentication parameters
- ◆ **replace** - substitutes the existing header or adds new headers to the document. The default value is set to true, so that all similar headers are replaced

- ◆ Displaying addition of multiple headers to the document

Snippet

```
<?php  
header ('WWW-Authenticate: Negotiate');  
header ('WWW-Authenticate: NTLM', false);  
?>
```

WWW-Authenticate - specifies the authentication string.

NTLM - specifies a challenge-response authentication mechanism.

false - defines the parameter of the replace option.

- ◆ Displays the response of the Web server for a request
- ◆ The request can include the status or the location of the client

Syntax

```
void header( string , boolean replace, integer http_response_code )
```

where,

- ◆ **string** - defines the authentication parameters
- ◆ **replace** - indicates whether previous defined headers need to be replaced or not
- ◆ **http_response_code** - forces the HTTP response code to the specified value

- ❖ An HTTP response codes consists of three digits that determine the status of a response.
- ❖ The status codes are classified as follows:
 - ❖ 1xx codes are informational codes
 - ❖ 2xx are success codes
 - ❖ 3xx are redirection codes
 - ❖ 4xx are client error codes
 - ❖ 5xx are server error codes

- ◆ Displaying a PHP script to redirect the user from one Web page or URL to another Web site

Snippet

```
header ("Location: http://google.com");
```

Location is a type of HTTP header redirecting the browser to the specified URL.

- ◆ Displaying a PHP script with an HTTP response code

Snippet

```
header("Location: http://google.com", true, 303);
```

- ◆ Location - is an HTTP header that redirects the browser to the specified URL
- ◆ true - defines the parameter of the replace option
- ◆ 303 - is a redirection response code

Summary

- ◆ A Web server and a database is required before installing PHP 7.
- ◆ Any older version of PHP must be uninstalled before installing PHP 7.
- ◆ A PHP file includes simple text, HTML tags, and PHP script.
- ◆ PHP supports both single-line and multiple line comments.

- ◆ PHP automatically assigns the correct data type for a variable depending upon the value assigned to the variable.
- ◆ A PHP script starts with <?php tag and ends with the ?> tag. These scripts are embedded in the HTML tags.
- ◆ A HTTP message or protocol is divided into three parts, the request or response line, the HTTP header, and the body of the protocol.

New Features of PHP 7

Session 3

For Aptech Center Only



Objectives

- ◆ *Explain the `intdiv()` function.*
- ◆ *Explain spaceship operator.*
- ◆ *Explain null coalescing operator.*
- ◆ *Describe Scalar Type Declarations.*
- ◆ *Explain Uniform Variable Syntax in PHP programs.*
- ◆ *Describe use operator.*
- ◆ *Explain closure call methods in the PHP programs.*
- ◆ *Explain Generator delegation via `yield from`.*
- ◆ *Explain Levels parameter of the `dirname()` function.*

- ◆ Is a division operator
- ◆ Is represented as /
- ◆ In earlier versions, always returned a float value and one had to use workarounds to get integers such as:

Snippet

```
<?php  
$x = 10;  
$y = 2;  
$z = (int) ($x / $y);  
echo $z;  
?>
```

\$x and \$y contain float values. Therefore, using the division operator may not provide accurate results. Here, the value returned is cast to integer using int() function.

- ◆ In PHP 7, the integer division function accepts two parameters:
 - ❖ Dividend
 - ❖ Divisor
- ◆ It returns an integer result

Syntax

```
intdiv(m, n)
```

Where,

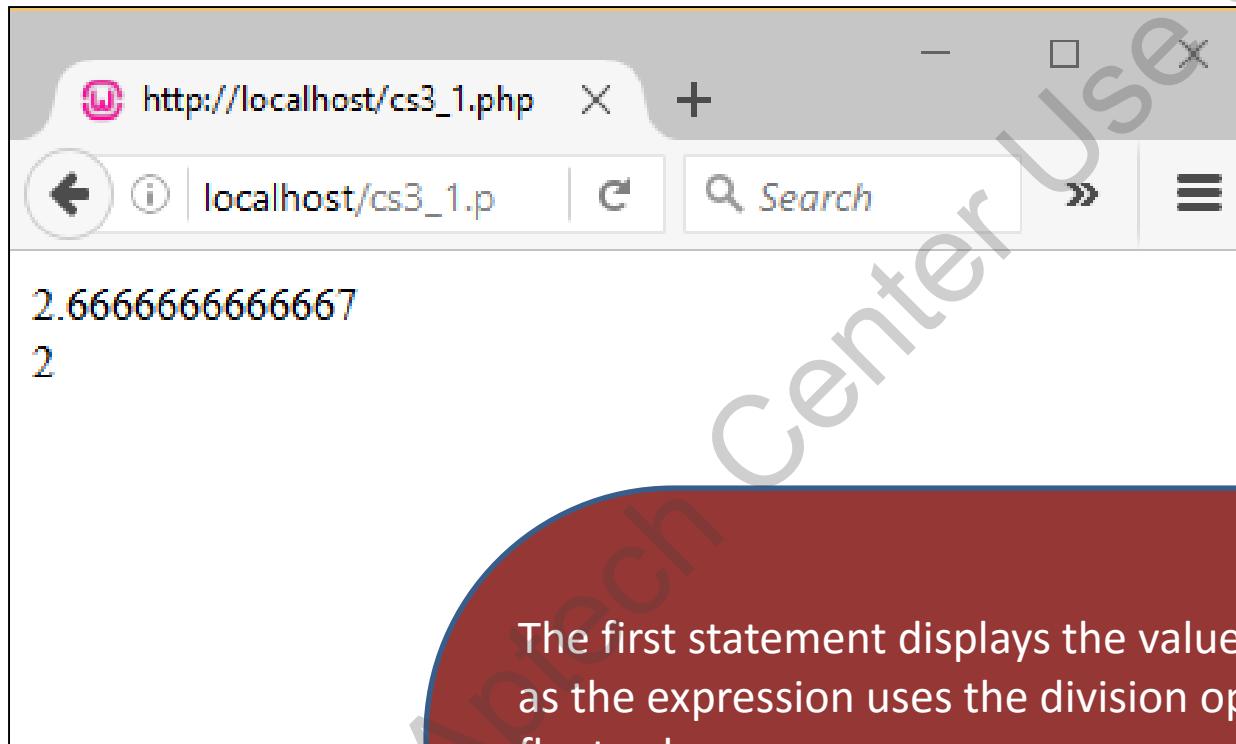
- ❖ **intdiv** – is the function
- ❖ **m** – is the dividend
- ❖ **n** – is the divisor

- ◆ Demonstrating the use of the `intdiv()` function and the behavior of operator ‘/’.

Snippet

```
<HTML>
<BODY>
<?php
    echo 8/3, "\n";
    echo intdiv(8, 3), "\n";
?
</BODY>
</HTML>
```

- ◆ Displays the following output:



The first statement displays the value 2 . 66666666666667 as the expression uses the division operator that returns a float value.

The second statement displays the value 2 as the expression is using the function `intdiv()` that returns an integer value after division.

- ◆ Is a single comparison operator
- ◆ Is mainly used for sorting
- ◆ Is represented as `<=>`
- ◆ Compares operands against three rules, namely:



Greater than



Lesser than



Equal to

- ◆ Demonstrating the use of spaceship operator

Syntax

```
$x <=> $y
```

- ◆ This expression displays the result:
 - ◆ -1 if \$x is smaller than \$y
 - ◆ 0 if \$x is equal to \$y
 - ◆ 1 if \$x is greater than \$y

- ◆ Demonstrating the use of spaceship operator on numbers

Snippet

```
<HTML>
<BODY>
<?php
$x = 1;
$y = 2;
echo $x.'<=>' . $y, ' Returns ', $x <=> $y;
// This will output -1
echo '</br>';
$x = 10;
$y = 10;
echo $x.'<=>' . $y, ' Returns ', $x <=> $y;
// This will output 0
```

```
echo '</br>';
$x = 10;
$y = 5;
echo $x.'<=>' . $y, ' Returns ', $x <=> $y;
// This will output 1
?>
</BODY>
</HTML>
```

- ◆ Displays the output:



The screenshot shows a web browser window with the URL `http://localhost/cs1_2.php` in the address bar. The page content displays three lines of text output from a PHP script:

```
1<=>2 Returns -1
10<=>10 Returns 0
10<=>5 Returns 1
```

- ◆ Demonstrating the use of spaceship operator on strings

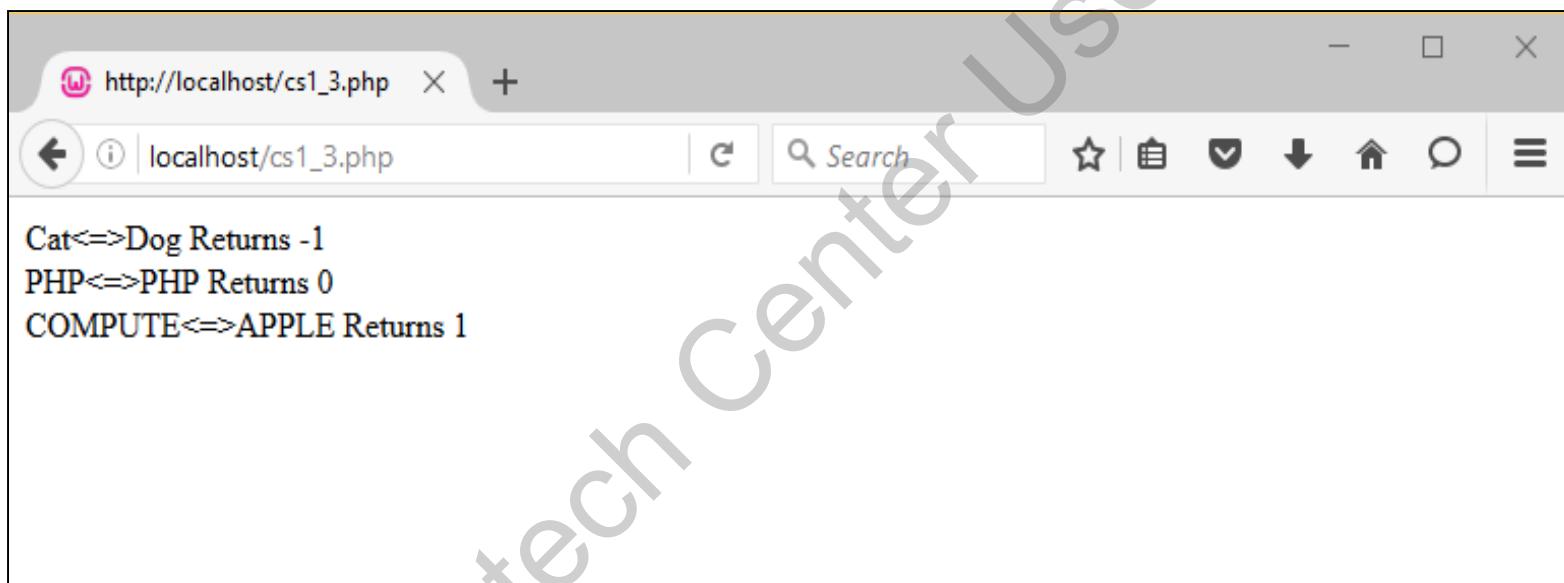
Snippet

```
<HTML>
<BODY>
<?php
$x = "Cat";
$y = "Dog";
echo $x.'<=>' . $y, ' // Returns ', $x <=> $y;
// This will output -1 because Cat is
// less than Dog.
echo '</br>';
$x = "PHP";
$y = "PHP";
echo $x.'<=>' . $y, ' // Returns ', $x <=> $y;

// This will output 0 because both strings have
// same value.
```

```
echo '</br>';
$x = "COMPUTE";
$y = "APPLE";
echo $x.'<=>'.$y, ' // Returns ', $x <=> $y;
// This will output 1 because "COMPUTE" is
greater than APPLE.
echo '</br>';
?>
</BODY>
</HTML>
```

- ◆ Displaying the output:



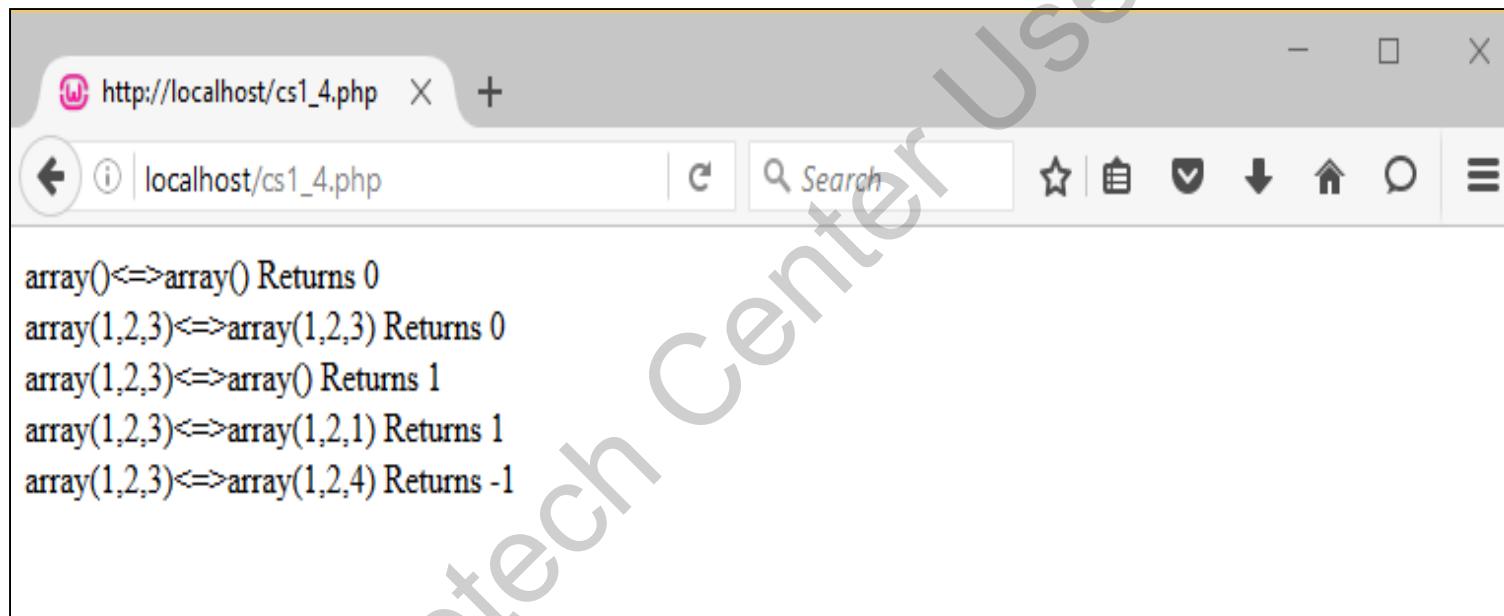
- ◆ Demonstrating the use of spaceship operator on arrays

Snippet

```
<HTML>
<BODY>
<?php
$x = array();
$y = array();
echo 'array()' . '<=>' . 'array()' . ' Returns ', $x <=> $y;
// This will output 0
echo '</br>';
$m = array(1,2, 3);
$n = array(1,2, 3);
$p = array(1,2, 1);
$q = array(1,2, 4);
echo 'array(1,2,3)' . '<=>' . 'array(1,2,3)' . ' Returns ', $m
<=> $n;
// This will output 0
```

```
echo '</br>';
echo 'array(1,2,3)'.'<=>'.'array()'.' Returns ',
$m <=> $x;
// This will output 1
echo '</br>';
echo 'array(1,2,3)'.'<=>'.'array(1,2,4)'.' Returns ',
,$m <=> $q;
// This will output -1
?>
</BODY>
</HTML>
```

- ◆ Displaying the output:



The screenshot shows a web browser window with the URL `http://localhost/cs1_4.php`. The page content displays five comparisons using the spaceship operator (`<=>`) between arrays:

```
array()<=>array() Returns 0
array(1,2,3)<=>array(1,2,3) Returns 0
array(1,2,3)<=>array() Returns 1
array(1,2,3)<=>array(1,2,1) Returns 1
array(1,2,3)<=>array(1,2,4) Returns -1
```

- ◆ Is used to check for a NULL value
- ◆ Is represented as ??
- ◆ Returns:

Result of its first operand if it exists and is not NULL

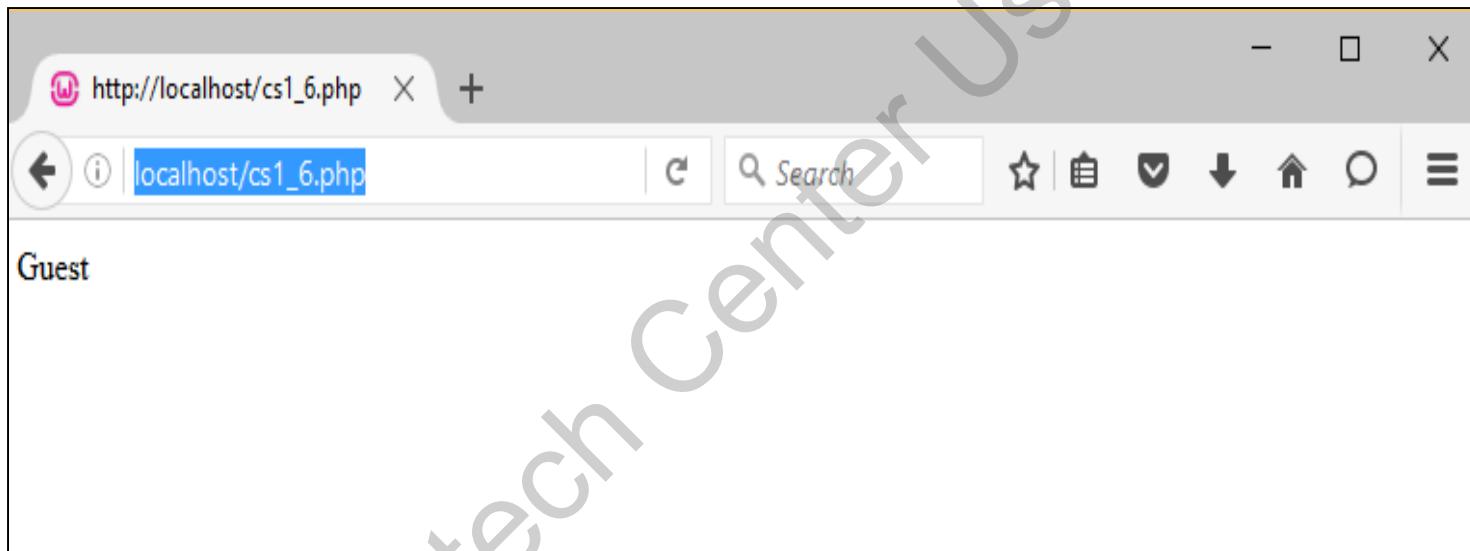
Otherwise, returns the second operand

- ◆ Demonstrating the use of null coalescing operator

Snippet

```
<HTML>
<BODY>
<?php
$name = $first_name ?? "Guest";
echo $name;
?>
</BODY>
</HTML>
```

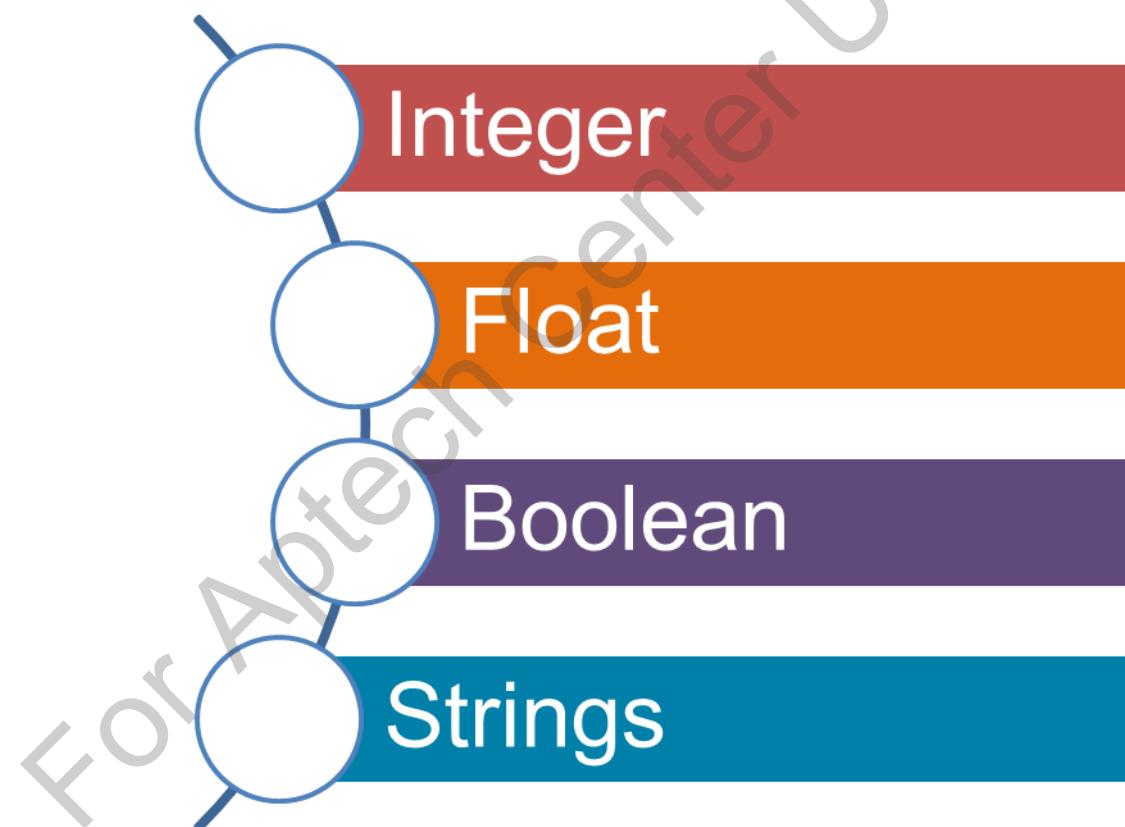
- ◆ Displaying the output:





Scalar Type Declarations

- ◆ Data types that hold single data type are known as scalar data types.
- ◆ These can be:





Scalar Type Declarations

- ◆ Refers to specifying the data type of a parameter in a function
- ◆ Also referred to as type hinting
- ◆ Provides hints to a function
- ◆ Enforces to input the parameter data type, that can be either strict or coercive

Uniform Variable Syntax

- ◆ Allows to use multiple operators in a given expression
- ◆ Follows a left-to-right evaluation order
- ◆ Allows backward compatibility in old evaluation technique
- ◆ Allows nesting of dereferencing operations

- ◆ Demonstrating the use of uniform variable syntax

Snippet

```
<?php
function e() {
echo "This is e() \n";
};

function f() {
echo "This is f() \n";
return e;
};

function g() {
echo "This is g()\n";
return f;
};

g();
echo "*****\n";
g()();
echo "*****\n";
g()()();
?>
```

- ◆ Displays the following output:

```
This is g()  
*****  
This is g()  
This is f()  
*****  
This is g()  
This is f()  
This is e()
```

Use Operator

- ◆ Is used to achieve aliasing
- ◆ Allows grouping of multiple declarations
- ◆ Enables to group classes, functions, and constants imported from the same namespace

Use Operator

- Demonstrating the use of use operator

Snippet

```
<?php
    namespace aptech;
    class Boston {
function say()
{
echo "Boston\n";
}
}
class NewYork {
function say()
{
echo "NewYork\n";
}
}
function foo1()
{
```

```
echo "This is foo1()\n";
}
function foo2()
{
echo "This is foo2()\n";
}
?>
```

Save the file as myfile.php.

Use Operator

- ◆ Include myfile.php in ugroup.php as shown:

Snippet

```
<?php

include 'myfile.php';

use aptech\{Boston, NewYork};
use function aptech\{foo1, foo2};

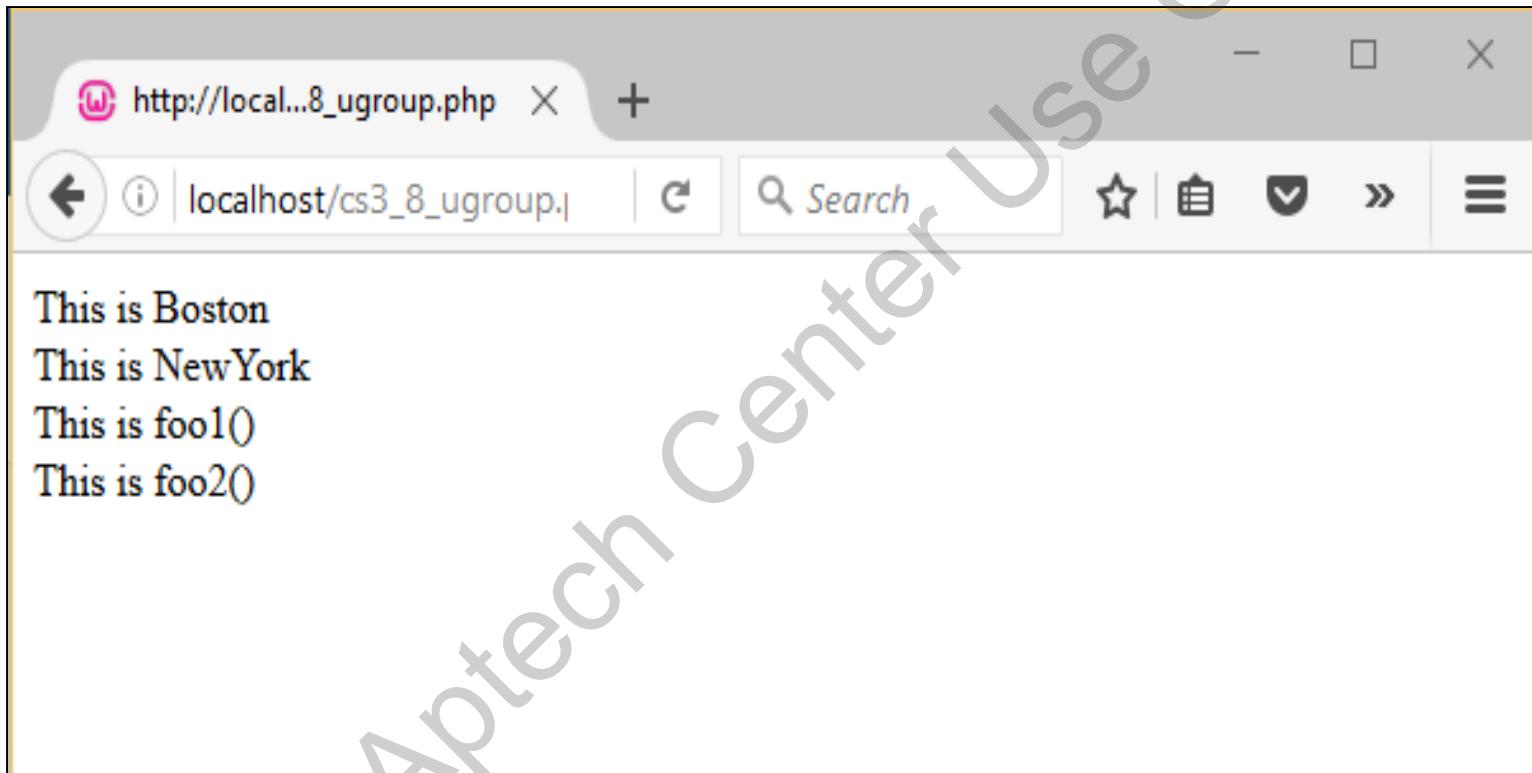
$d = new Boston();
$d->say();

$n = new NewYork();
$n->say();

foo1();
foo2();
?>
```

Use Operator

- ◆ Displays the following output:



Closure call () Method

- ◆ A closure record stores a function together with an environment.
- ◆ A closure method allows the functions to access the captured variables.
- ◆ Adding `$this` parameter to the closure method results in two new methods:
 - ◆ The Instance method - `Closure->bindTo()`
 - ◆ The Static method - `Closure::bind()`

Instance Method

Static Method

Called at closure instance

Accepts two arguments

Passed as the first argument to the closure ()

Accepts two arguments

Closure call () Method

- ◆ Demonstrating the use of closure ()

Snippet

```
<?php

class Greetings {

private $word = "Hello";
}

$closure = function($whom) {

echo "$this->word $whom\n";
};

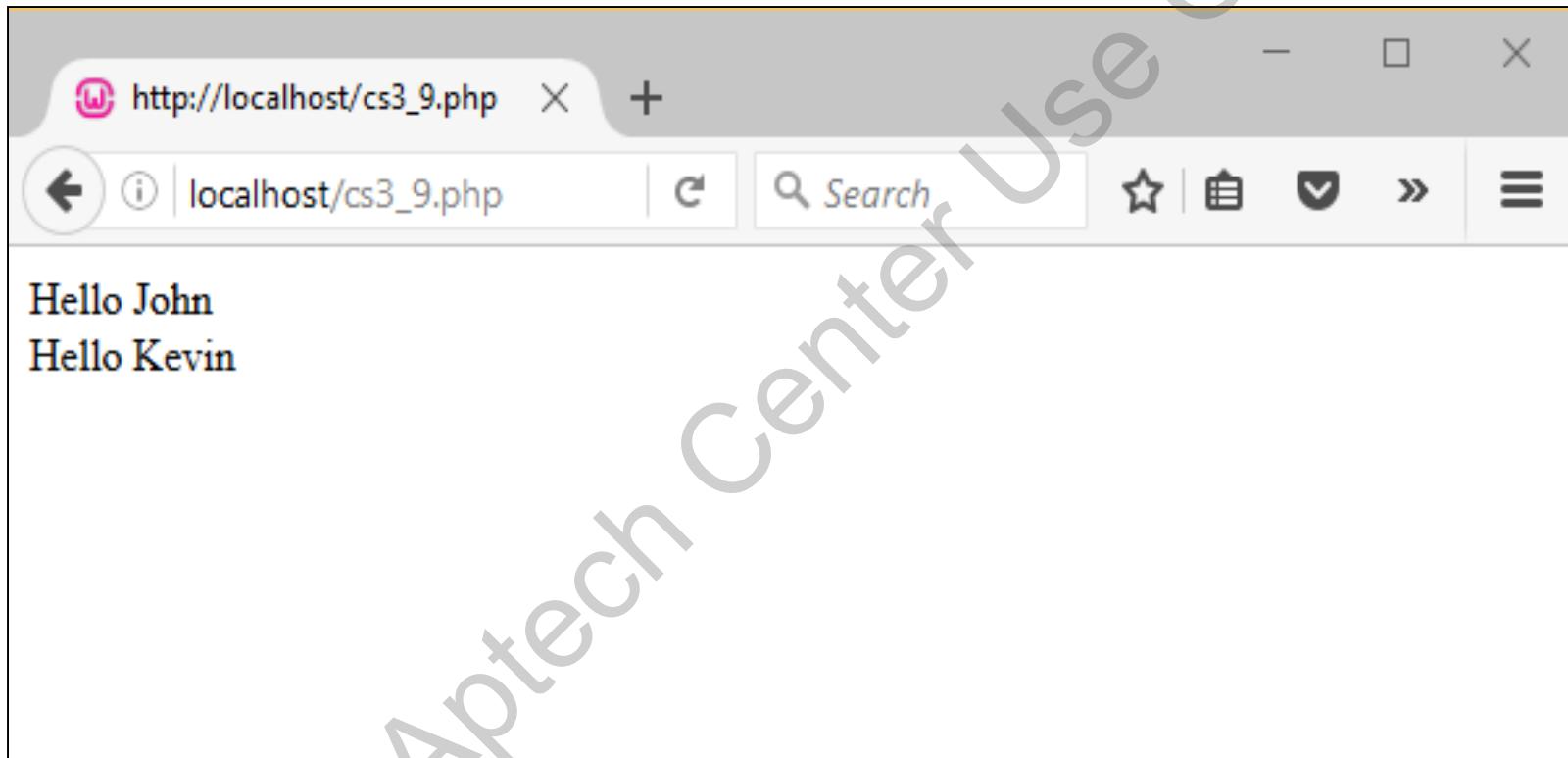
$obj = new Greetings();

$closure->call($obj, 'John');
$closure->call($obj, 'Kevin');

?>
```

Closure call () Method

- ◆ Displays the following output



Generator Return Expressions

- ◆ A statement used within a generator to return the final expression.
- ◆ The value can be retrieved using the `getReturn()` method.
- ◆ The method should be used only after the generator has finished yielding results.

Generator Return Expressions

- ◆ Demonstrating the use of `getReturn()` expression

Snippet

```
<?php
srand();
function random_numbers($k) {
    for ($i=0; $i<$k; $i++) {

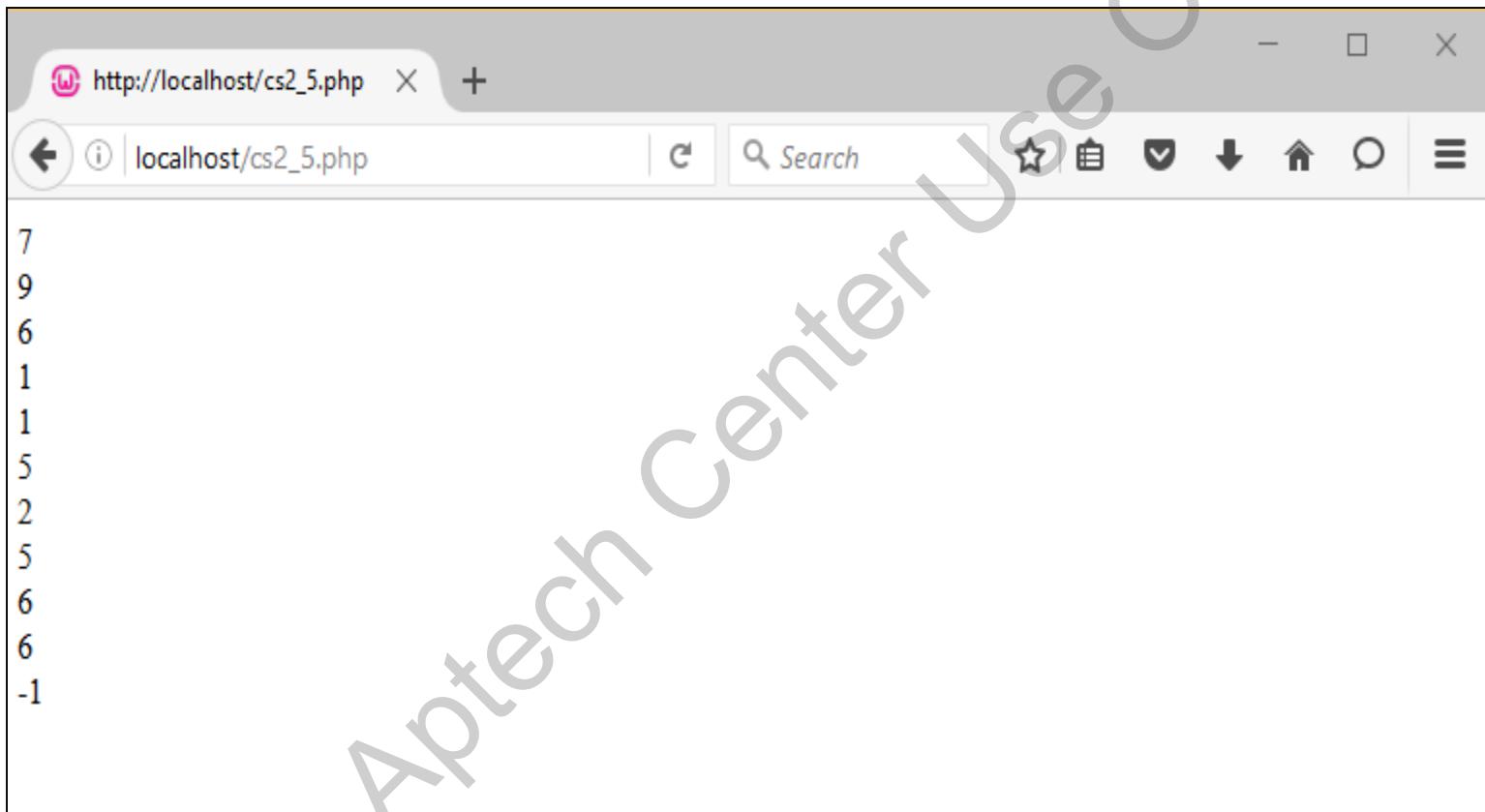
        $r = rand(1, 10);

        yield $r;
    }

    return -1;
}
$rnns = random_numbers(10);

foreach ($rnns as $r) {
    echo "$r";
    echo '</br>';
}
echo $rnns->getReturn() . PHP_EOL;
?>
```

- ◆ Displaying the output:



Generator Delegation via yield from

- ◆ Demonstrating the use of `yield from` keyword

Snippet

```
<?php
function f1() {
    yield from f2();
    yield "f1() 1";
    yield "f1() 2";
    yield from [3, 4];
    yield "f1() 3";
    yield "f1() end";
}
function f2() {
    yield "f2() 1";
    yield "f2() 2";
    yield "f2() 3";
    yield "f2() end";
}
$f = f1();
foreach ($f as $val) {
    echo "$val\n";
}
?>
```

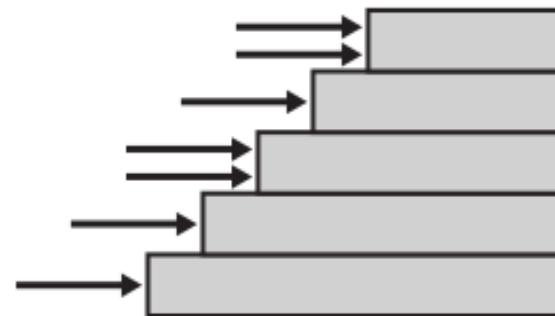
Generator Delegation via yield from

- ◆ Displaying the following output:

```
f2() 1
f2() 2
f2() 3
f2() end
f1() 1
f1() 2
3
4
f1() 3
f1() end
```

levels Parameter of dirname()

- ◆ The dirname() function returns the parent's directory path.
- ◆ The dirname() now accepts a second parameter that specifies the number of levels a parent directory can go up.
- ◆ The levels parameter tells how many parent directories can go up.



levels Parameter of dirname()

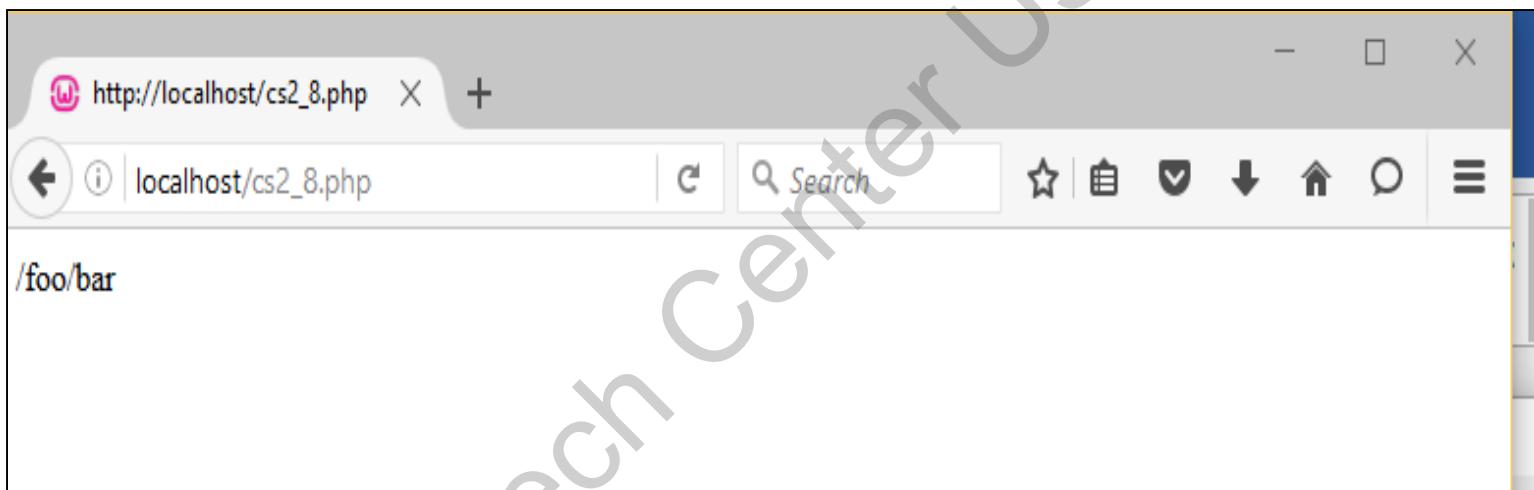
- ◆ Demonstrating the use of dirname() function

Snippet

```
<?php  
$path = '/foo/bar/bat/baz';  
echo dirname($path, 2);  
?>
```

levels Parameter of dirname()

- ◆ Displays the following output:



Summary

- ◆ The new intdiv() performs integer division and is considered the inverse of the modulo operator %.
- ◆ Spaceship operator also known as combined comparison operator, returns three distinct values, -1, 0, or +1.
- ◆ The null coalesce operator ?? returns the left operand if it is not null; otherwise, it returns the right operand.

- ◆ Group 'use' declarations allow to deduplicate common prefixes in 'use' statements and specify unique parts within a block {}.
- ◆ Generator->getReturn() method allows you to retrieve value returned by any return statement inside the generator.
- ◆ With addition of \$this, closure has gained two methods, the instance method Closure->bindTo() and the static method Closure::bind().
- ◆ The dirname() function can now accept a second parameter to set how many levels up it will go.

Form Handling in PHP

Session 4



Objectives

- ◆ *Explain the use of the GET method*
- ◆ *Explain the use of the POST method*
- ◆ *Retrieve data from forms using the Form methods*
- ◆ *Explain the use of hidden fields*

- ◆ Form data is passed to the Web server using following methods:
 - ❖ GET
 - ❖ POST
- ◆ Web server
 - ❖ Accepts the information
 - ❖ Processes the application data
 - ❖ Stores it to the database

- ◆ It is a Web page containing fields
- ◆ It is used by users to enter information
- ◆ It passes entered data from a client to a server

Contact us!

Name * :

Website * :

E-Mail * :

Phone_Number * :

Subject :

Message * :

Captcha:  »» Daten absenden

- ◆ Steps for handling HTML forms and process information are as follows:
 - ❖ User enters information in an HTML form and sends it to the Web server
 - ❖ Web server passes the information to the PHP script engine for:
 - ❖ Processing the information
 - ❖ Sending output back to the Web browser

- ◆ HTML <FORM> tag is:
 - ❖ Used to create HTML form
 - ❖ Included within the <FORM> and </FORM> tag
- ◆ Attributes of an HTML form tag are:
 - ❖ **Action** - defines URI where the form data is sent after it has been submitted
 - ❖ **Method** - defines protocols that are used to submit the form data set
- ◆ Method protocols are of two types:
 - ❖ GET
 - ❖ POST

- ◆ GET method
 - ❖ Directs the Web browser to send the encoded user information to the processing agent
 - ❖ Appends the encoded information at the end of the URL by a question mark (?) which separates URL and form information
- ◆ The form data sent in the URL is a stream of name/value pair separated by ampersand (&)

- ◆ An input variable will have following structure:
 - ❖ Name=value
- ◆ Multiple input variables are grouped as follows:
 - ❖ Name1=value1&Name2=value2&Name3=value3
- ◆ The following example shows the multiple name/value pair separated by the & sign:
 - ❖ Name=john&age=18
- ◆ The query string is appended with the following URL:
 - ❖ <http://www.information.com/text.php?Name=john&age=18>

- ◆ The restrictions of GET method are as follows:
 - ❖ Form data set values are restricted to American Standard Code for Information Interchange (ASCII) characters
 - ❖ Amount of information transferred is limited
 - ❖ Length of the query string is restricted to 255 characters

Using the POST Method

- ◆ POST method
 - ❖ Directs the Web browser to send all the user information to the processing agent
 - ❖ Uses message body of an HTTP request to send the information
 - ❖ Has capacity to transmit more information as:
 - ❖ No physical limit on the amount of information passed in the HTTP request message body
 - ❖ Uses variables to pass form information
- ◆ The drawback of POST method is as follows:
 - ❖ Information sent is not encrypted, so hackers can easily access it

Difference in the GET and POST Method

- ◆ GET and POST methods work almost identically

Table lists the difference between the GET and POST method

GET	POST
Encodes the form data as a stream of name/value pairs and appends it in the URL making it visible in the browser	Sends the encoded form data through the body of an HTTP request
Form submissions can be bookmarked	Form submissions cannot be bookmarked
Is less secure as the information is displayed in the URL	Is more secure for transmitting passwords and other sensitive information, as the form data is embedded in the body of the HTTP request
The amount of data that can be sent is limited depending on the browser used	Does not have size limitations
This method is mainly used for displaying data such as searching, sorting, and pagination	This method is mainly used for data manipulation such as adding and editing data

- ◆ Retrieving data from an HTML form using the GET method

Syntax

```
$varname = $_GET["variable"];
```

Where,

- ◆ **varname** - specifies the name of the variable in which data is to be stored
- ◆ **\$_GET["variable"]** - specifies the name of the input variable

- ◆ Steps for creating an HTML form to retrieve data using the **GET** method are as follows:
 1. Open a new file in the **gedit** text editor
 2. Enter the code and save the file as **Details.html** in the `/usr/local/apache2/htdocs` directory

Snippet

```
<HTML>
<BODY>
<B>ENTER YOUR PERSONAL DETAILS</B>
<FORM METHOD=GET ACTION="Details.php">
FIRST NAME:
<INPUT NAME="n1text" TYPE="TEXT"><BR>
LAST NAME:
<INPUT NAME="n2text" TYPE="TEXT"><BR>
ADDRESS:
<TEXTAREA NAME="n3text" ROWS=1, COLUMNS=1000></TEXTAREA>
<BR>
CONTACT NO:
<INPUT NAME="n4text" TYPE="TEXT"><br>
<INPUT TYPE="SUBMIT" NAME="SUBMIT" VALUE="SUBMIT">
</FORM>
</BODY>
</HTML>
```

- ◆ Steps for creating PHP script to retrieve and process the data entered in the HTML form are as follows:
 1. Open a new file in the **gedit** text editor
 2. Enter the code and save the file as **Details.php** in
`/usr/local/apache2/htdocs` directory

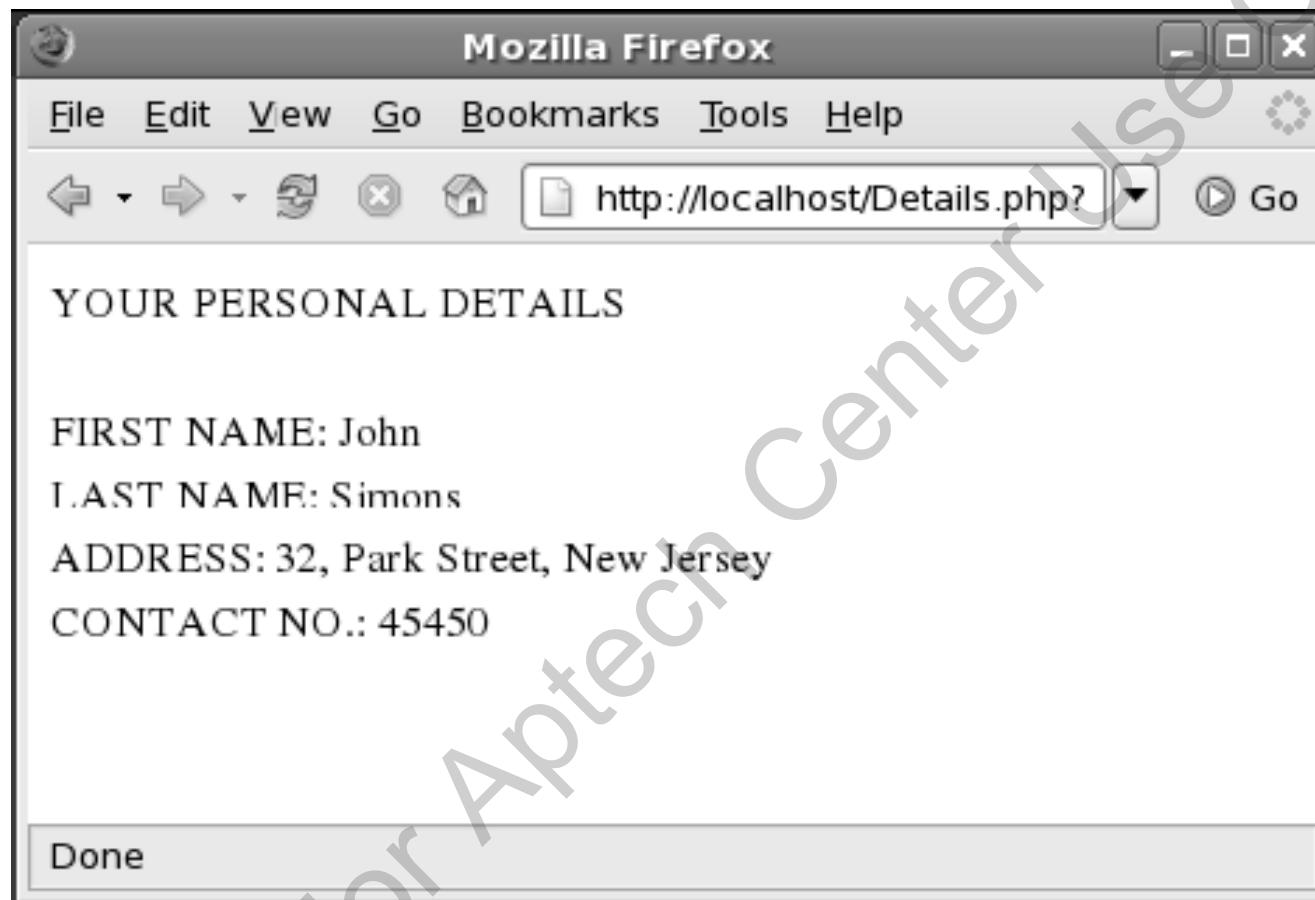
Snippet

```
<?php  
error_reporting(0);  
$A = $_GET["n1text"];  
  
$B = $_GET["n2text"];  
$C = $_GET["n3text"];  
$D = $_GET["n4text"];  
echo "YOUR PERSONAL DETAILS";  
echo "<BR><BR>";  
echo "FIRST NAME: $A <BR>";  
echo "LAST NAME: $B <BR>";  
echo "ADDRESS: $C <BR>";  
echo "CONTACT NO.: $D <BR>";  
?>
```

- ◆ Steps for displaying **details.html** page are as follows:
 1. Open the Mozilla Firefox Web browser
 2. Enter `http://localhost/Details.html` in the Address bar and press **Enter**
 3. Enter **John** in the FIRST NAME box
 4. Enter **Simons** in the LAST NAME box
 5. Enter **32, Park Street, New Jersey** in the ADDRESS box
 6. Enter **45450** in the CONTACT NO. box
 7. Click **SUBMIT**

The screenshot shows a Mozilla Firefox browser window with the title bar "Mozilla Firefox". The address bar contains the URL `http://localhost/Details.html`. The main content area displays a form titled "ENTER YOUR PERSONAL DETAILS". The form has four text input fields: "FIRST NAME:", "LAST NAME:", "ADDRESS:", and "CONTACT NO.". Below these fields is a "SUBMIT" button. At the bottom of the form is a "Done" button.

Displays the following output:



- ◆ Retrieving data from an HTML form using the POST method

Syntax

```
$varname = $_POST["variable"];
```

Where,

- ◆ **varname** - specifies the name of the variable in which the data is to be stored
- ◆ **\$_POST["variable"]** - specifies the name of the input variable

- ◆ Steps for creating an HTML form to retrieve data using the POST method are as follows:
 1. Open a new file in the **gedit** text editor
 2. Enter the code and save the file as
EMP_DETAILS.html in the
/usr/local/apache2/htdocs directory

Snippet

```
<HTML>
<HEAD>
<TITLE>Employee Details</TITLE>
</HEAD>
<BODY>
<H4>Enter your details</H4>
<FORM METHOD=POST ACTION="EMP_DETAILS.php">
<TABLE>
<TR>
<TD>Employee ID</TD>
<TD><INPUT TYPE="text" NAME="empid"></TD>
</TR>
<TR>
<TD>Name</TD>
<TD><INPUT TYPE="text" NAME="Name"></TD>
</TR>
<TR>
```

Snippet

```
<TD>Department</TD>
<TD>
<INPUT TYPE="radio" NAME="dept" VALUE="Finance">Finance
<INPUT TYPE="radio" NAME="dept" VALUE="Marketing">Marketing
<INPUT TYPE="radio" NAME="dept" VALUE="IT">IT
</TD>
</TR>
<TR>
<TD>Email</TD>
<TD><INPUT TYPE="text" NAME="email"></TD>
</TR>
</TABLE>
<BR>
<TD><INPUT TYPE="submit" VALUE="SUBMIT"></TD>
</FORM>
</BODY>
</HTML>
```

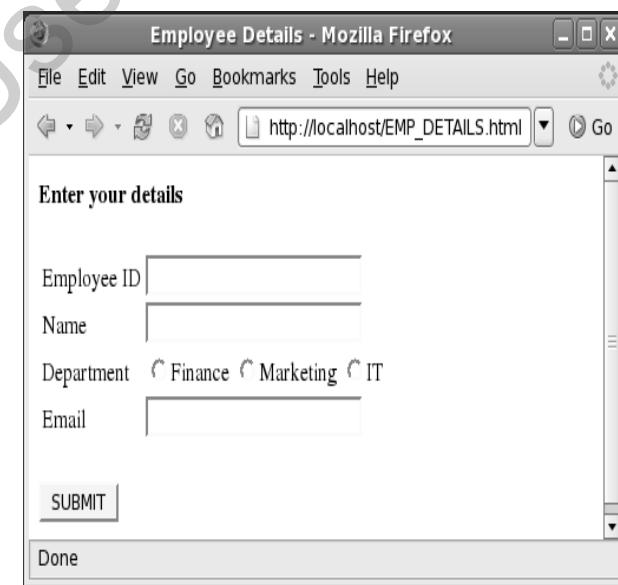
- ◆ Steps for creating a PHP script to retrieve and process the data entered in the HTML form are as follows:
 1. Open a new file in the **gedit** text editor
 2. Enter the code and save the file as **EMP_DETAILS.php** in the `/usr/local/apache2/htdocs` directory

Snippet

```
<?php
error_reporting(0);
$A=$_POST["empid"];
$B=$_POST["Name"];
$C=$_POST["dept"];
$D=$_POST["email"];
echo "YOUR PERSONAL DETAILS";
echo "<BR><BR>";
echo "EMPID: $A <BR>";
echo "NAME: $B <BR>";
echo "DEPARTMENT NAME: $C <BR>";
echo "EMAIL: $D <BR>";
?>
```

- ◆ Steps for displaying **EMP_DETAILS.html** page are as follows:

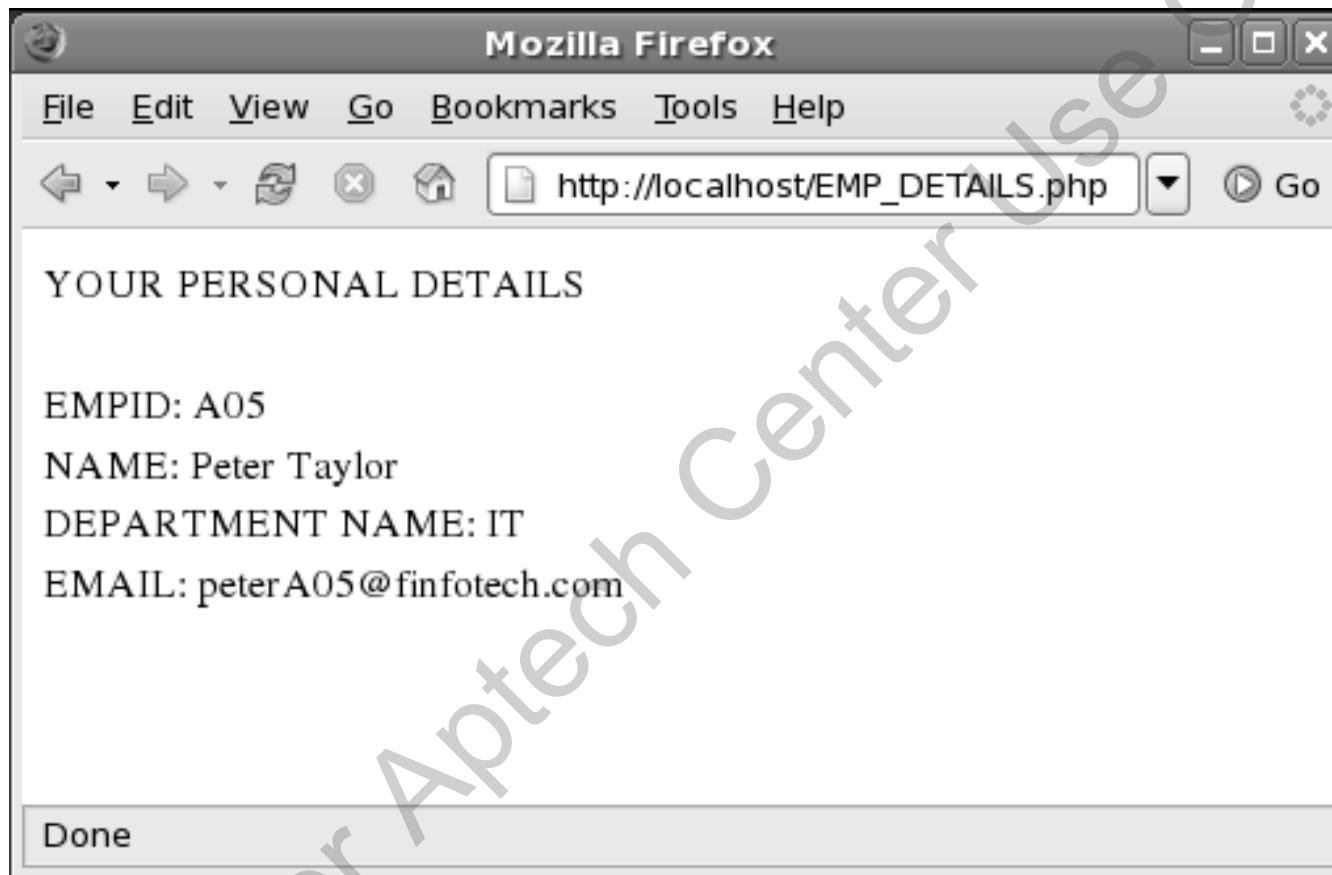
1. Open the Mozilla Firefox Web browser
2. Enter
`http://localhost/EMP_DETAILS.html` in the Address bar and press **Enter**
3. Enter **A05** in the Employee ID textbox
4. Enter **Peter Taylor** in the Name box
5. Select **IT** as the Department
6. Enter **peterA05@infotech.com** in the Email box
7. Click **SUBMIT**



The screenshot shows a Mozilla Firefox browser window titled "Employee Details - Mozilla Firefox". The address bar contains the URL `http://localhost/EMP_DETAILS.html`. The main content area displays a form with the following fields:

- Employee ID:** A text input field.
- Name:** A text input field.
- Department:** A radio button group with three options: Finance, Marketing, and IT. The "IT" option is selected.
- Email:** A text input field.
- SUBMIT:** A submit button.
- Done:** A link at the bottom of the form.

Displays the following output:



◆ Hidden Field

- ◆ Is embedded in the HTML source code of the form
- ◆ Enables the user to pass variables with values from one form to another without requiring to re-enter the information
- ◆ Contents cannot be viewed by the user

Syntax

```
<INPUT TYPE=HIDDEN NAME=hidden1 VALUE="PHP MESSAGE">
```

Where,

- ◆ **INPUT TYPE** - specifies that the field is hidden
- ◆ **NAME** - specifies the name of the hidden field
- ◆ **VALUE** - specifies the value as it appears on the form

- ◆ Steps for passing the names of the continents in a PHP script using hidden fields are as follows:
 1. Open a new file in the **gedit** text editor
 2. Enter the code and save the file as **continent.html** in the **/usr/local/apache2/htdocs** directory

Snippet

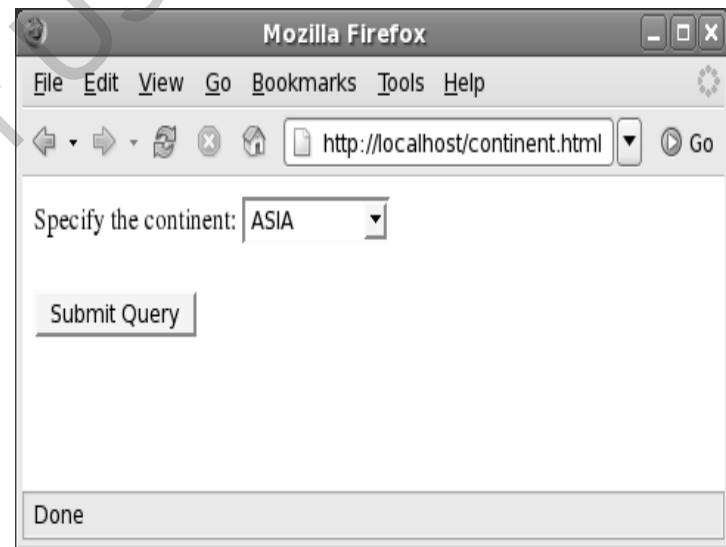
```
<html>
<FORM METHOD='get' action='continent.php'>
Specify the continent:
<SELECT TYPE='LISTBOX' NAME='continent'>
<OPTION>ASIA</OPTION>
<OPTION>AUSTRALIA</OPTION>
<OPTION>EUROPE</OPTION>
</SELECT><BR><BR>
<INPUT TYPE=HIDDEN NAME=Asia>
<INPUT TYPE=HIDDEN NAME=Australia>
<INPUT TYPE=HIDDEN NAME=Europe>
<BR><INPUT TYPE=SUBMIT>
</FORM> </html>
```

- ◆ Steps for creating a PHP script to retrieve and process the data entered in the HTML form are as follows:
 1. Open a new file in the **gedit** text editor
 2. Enter the code and save the file as **continent.php** in the **/usr/local/apache2/htdocs** directory

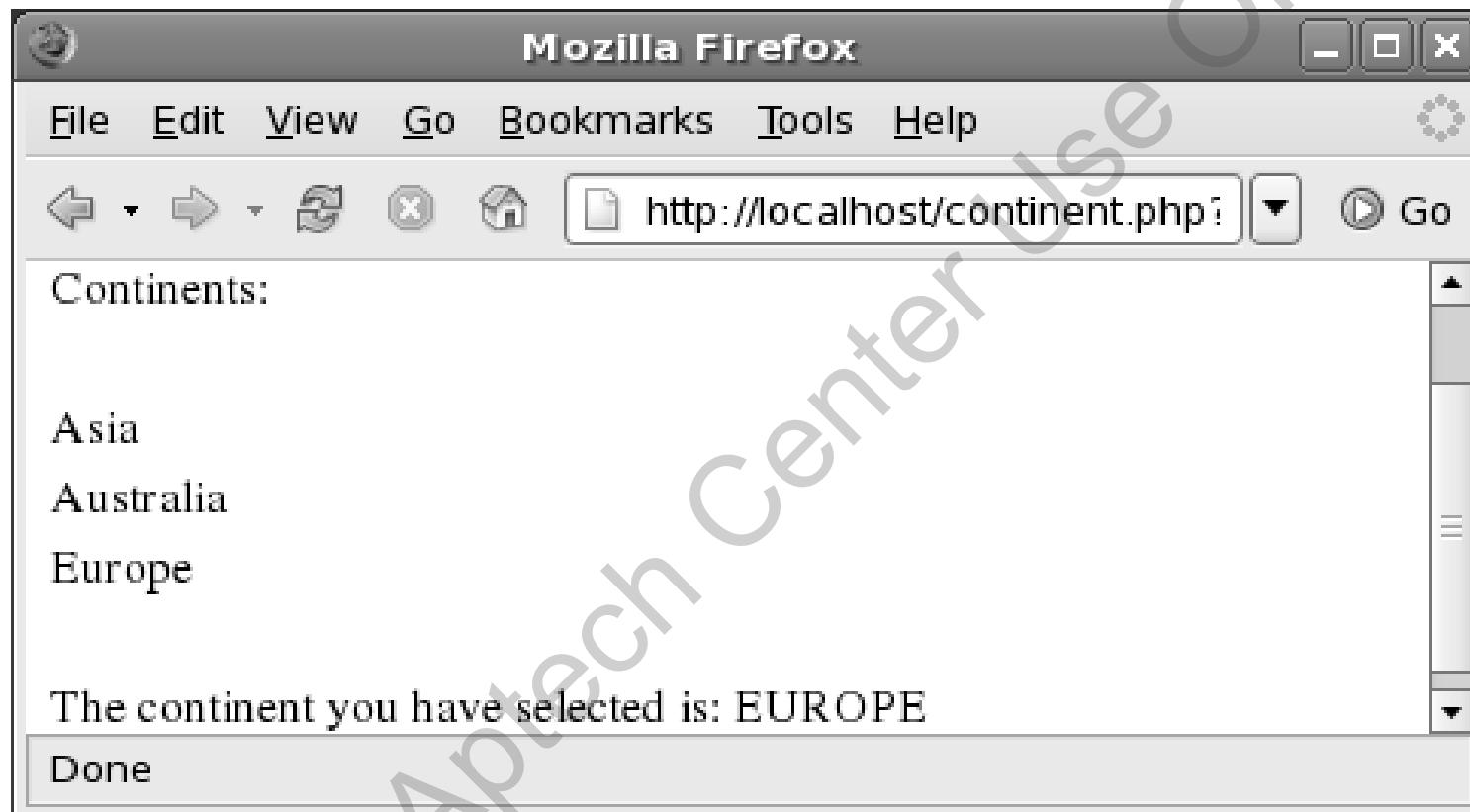
Snippet

```
<?php
$A=$_GET['Asia'];
$B=$_GET['Australia'];
$C=$_GET['Europe'];
$Name=$_GET['continent'];
echo "<BR>";
echo "Continents:<BR> <BR> Asia <BR> Australia <BR> Europe <BR> <BR>";
echo "The continent you have selected is: $Name";
?>
```

- ◆ Steps for displaying **continent.html** page are as follows:
1. Open the Mozilla Firefox Web browser
 2. Enter `http://localhost/continent.html` in the address bar and press **Enter**
 3. Select the required continent from the drop-down menu
 4. Click **Submit Query**



Displays the following output:



- ◆ A form is a Web page that is used to pass data from a client to a server
- ◆ PHP has a built-in support for collecting data from an HTML form
- ◆ The attributes of a form are namely, action and method
- ◆ The action attribute of a form specifies the URL that will process the form data and provide the feedback
- ◆ The method attribute of the form defines the method of transmitting information to the URL

Summary

- ◆ The GET method directs the Web browser to send the encrypted user information appended at the end of the URL, to the processing agent
- ◆ The POST method directs the Web browser to send all the user information to the processing agent, through the message body of an HTTP request
- ◆ Hidden form fields are not visible to users and enable form developers to pass information from a form to a script or from one form to another, before being passed to a script

Using Variables and Expressions in PHP

Session 5



Objectives

- ◆ *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,

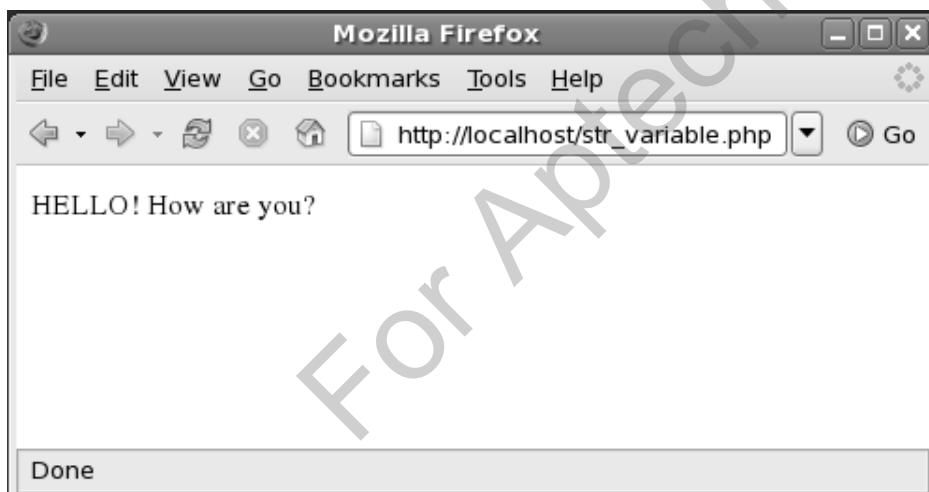
- ❖ **\$variable_name** - specifies the name of the variable
- ❖ **value** - specifies the value the variable will store

- ◆ Storing a string value in a variable
 - ❖ Enter the code and save it in a script named **str_variable.php**

Snippet

```
<?php  
$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.

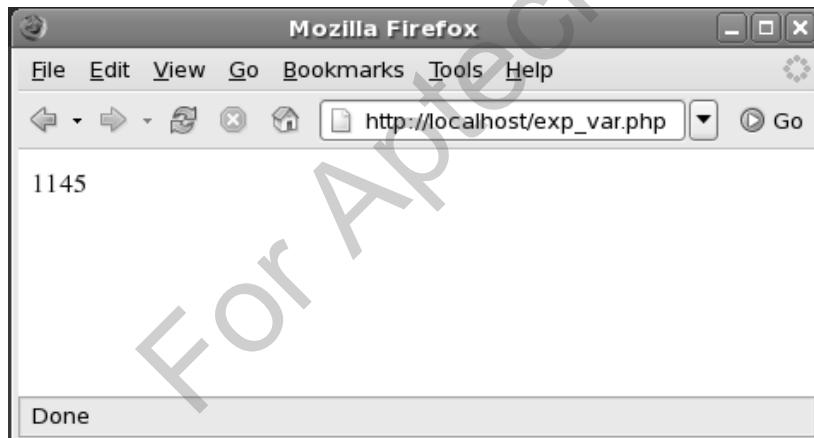
The string is enclosed within the double quotes.

- ◆ Storing the value of an expression in a variable
 - ❖ Enter the following code and save in a script named **exp_var.php**

Snippet

```
<?php  
$number1 = 1019;  
$number2 = 126;  
$number5 = $number1 + $number2;  
echo $number5;  
?>
```

Displays the following output:



In the code, the **\$number5** 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

Syntax

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

Where,

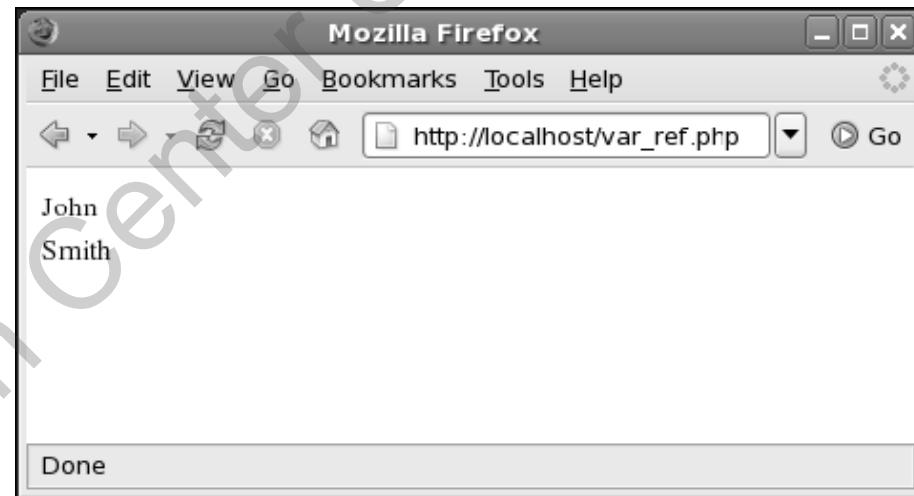
- ◆ **\$new_varname** - defines the new variable name
- ◆ **\$old_varname** - specifies the other variable name whose value is to be assigned

- ◆ Assigning the value of one variable to another variable
 - ❖ Enter the following code and save in a script named **exp_var.php**

Snippet

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

Displays the following output:



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

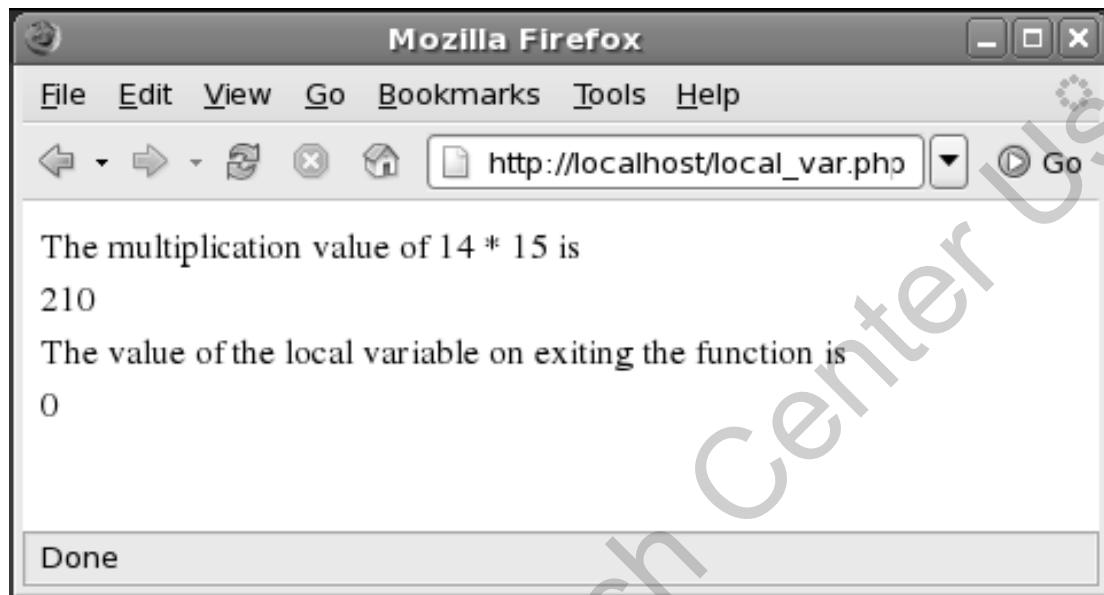
- ◆ Local variable
 - ❖ Is a variable initialized and used inside a function
 - ❖ Is similar to a normal variable
 - ❖ Is declared inside a function
 - ❖ Its lifetime begins when the function is called, and ends when the function is completed

- ◆ Displaying the use of local variables

Snippet

```
<?php
$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

Syntax

```
global $var_name;
```

Where,

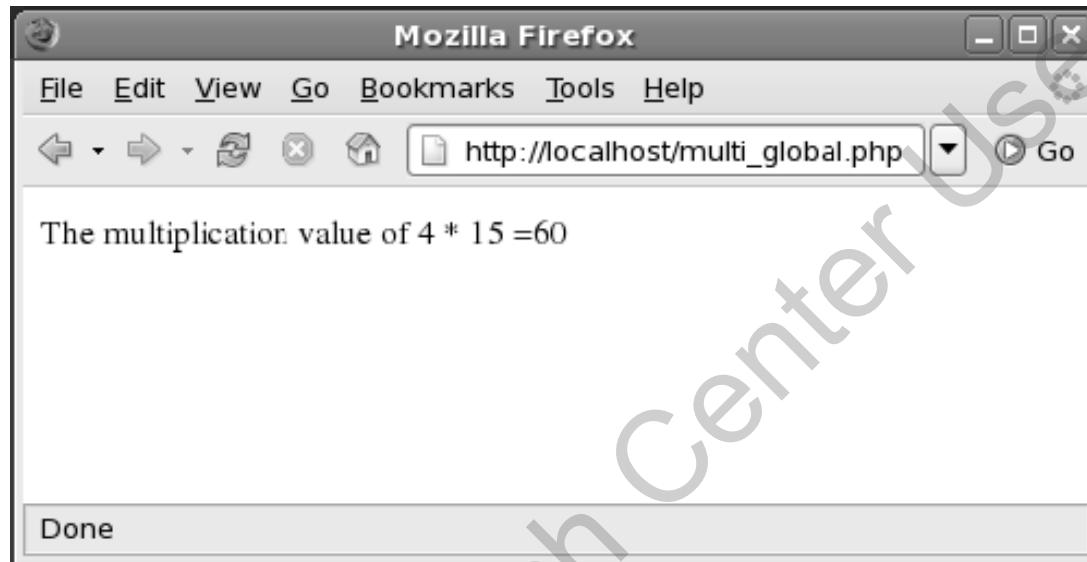
- ◆ **\$var_name** - defines the global variable name

- ◆ Displaying multiplication of two numbers using global variables
 - ❖ Enter the code in a script named **multi_global.php**

Snippet

```
<?php
$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;
```

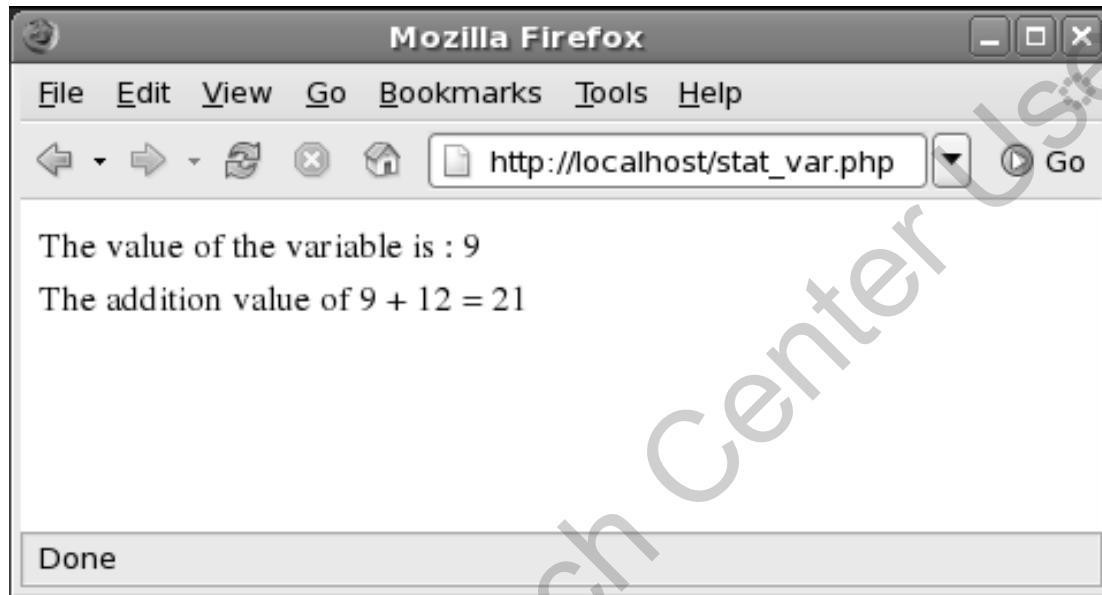
- ❖ **var_name** - defines the variable name
- ❖ **value** - specifies the value of the variable

- ◆ Illustrating the use of a static variable
 - ❖ Enter the code as shown in a script named **stat_var.php**

Snippet

```
<?php
$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 identification string specified in the headers when responding to requests
- ◆ Returns the name and the version of the server software

Snippet

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

- ◆ Displays the following output:



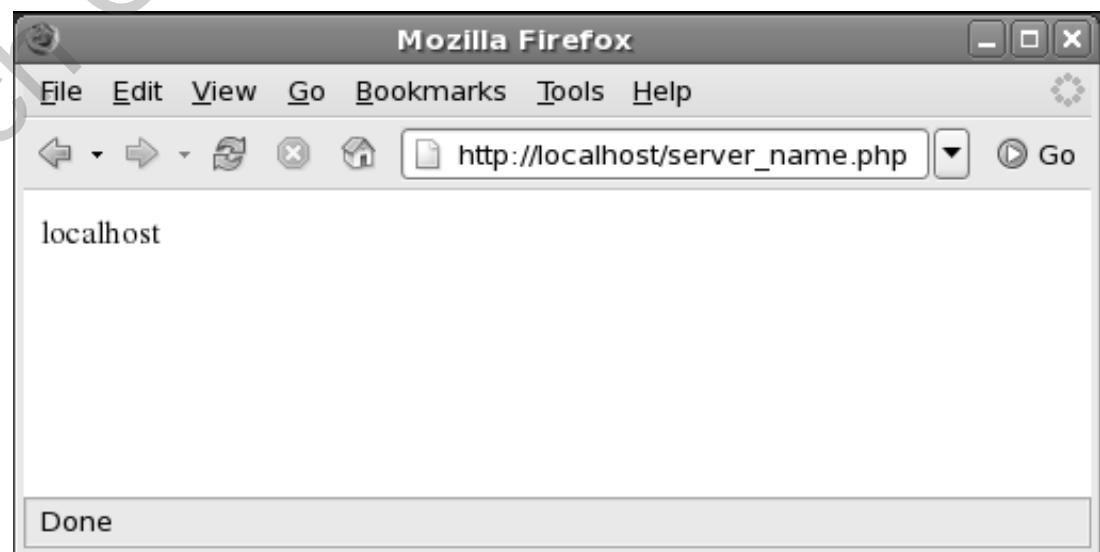
SERVER_NAME

- ◆ 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  
echo  
$_SERVER['SERVER_NAME'];  
?>
```

Output:



SERVER_PROTOCOL

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

Snippet

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

- ◆ Displays the following output:



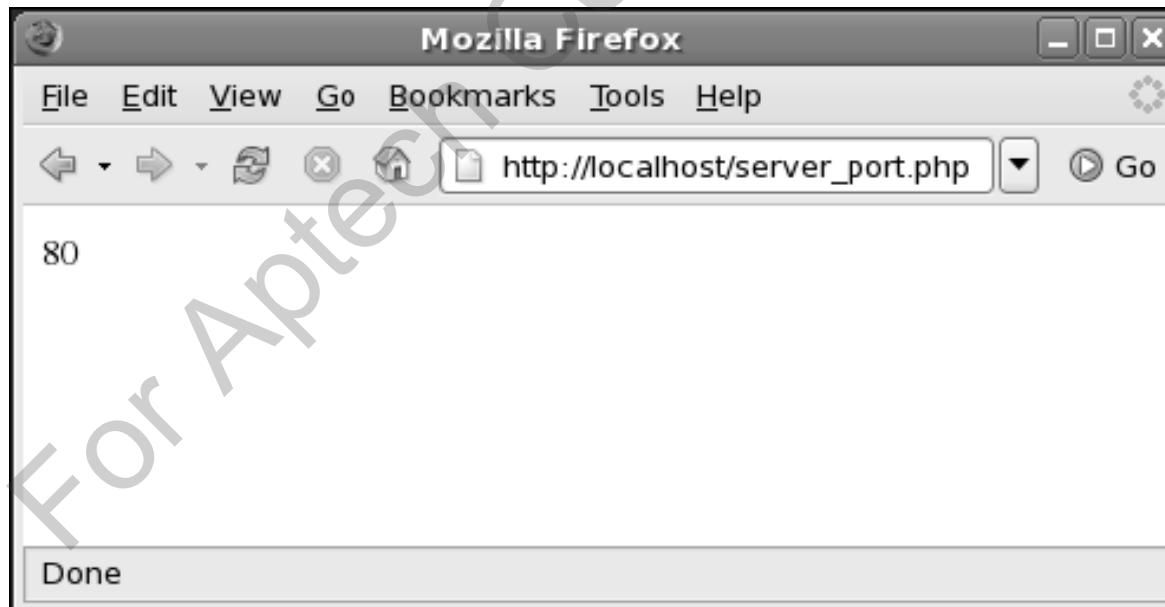
SERVER_PORT

- ◆ Returns the server port number for the script

Snippet

```
<?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  
$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
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:



HTTP_USER_AGENT

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

Snippet

```
<?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  
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

Summary

- ◆ 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

Scalar Type Declarations and Anonymous Classes

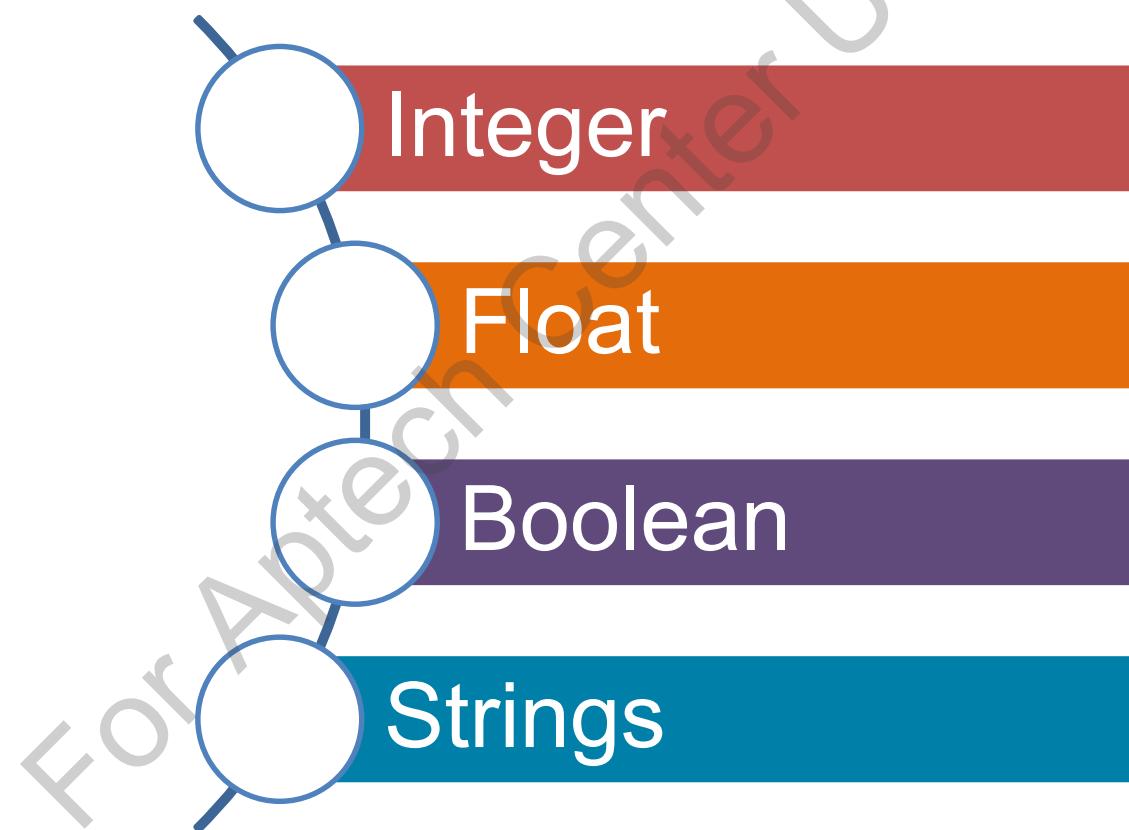
Session 7



- ◆ *Explain scalar type declarations.*
- ◆ *Describe usage of scalar type declarations in PHP programs.*
- ◆ *Explain scalar type hinting.*
- ◆ *Explain anonymous classes.*
- ◆ *Describe usage of anonymous classes in PHP programs.*

Scalar Data Type

- ◆ The data types that hold single data type are known as scalar data types.
- ◆ The data types can be:



- ◆ Refers to specifying the data type of a parameter in a function
- ◆ Also referred to as type hinting
- ◆ Provides hints to a function
- ◆ Enforces the input the parameter data type, that can be either strict or coercive

- ◆ Declaring a new function

Syntax

```
function function_name(type, p1)
```

Where,

- ◆ **m** – is the dividend
- ◆ **n** – is the divisor

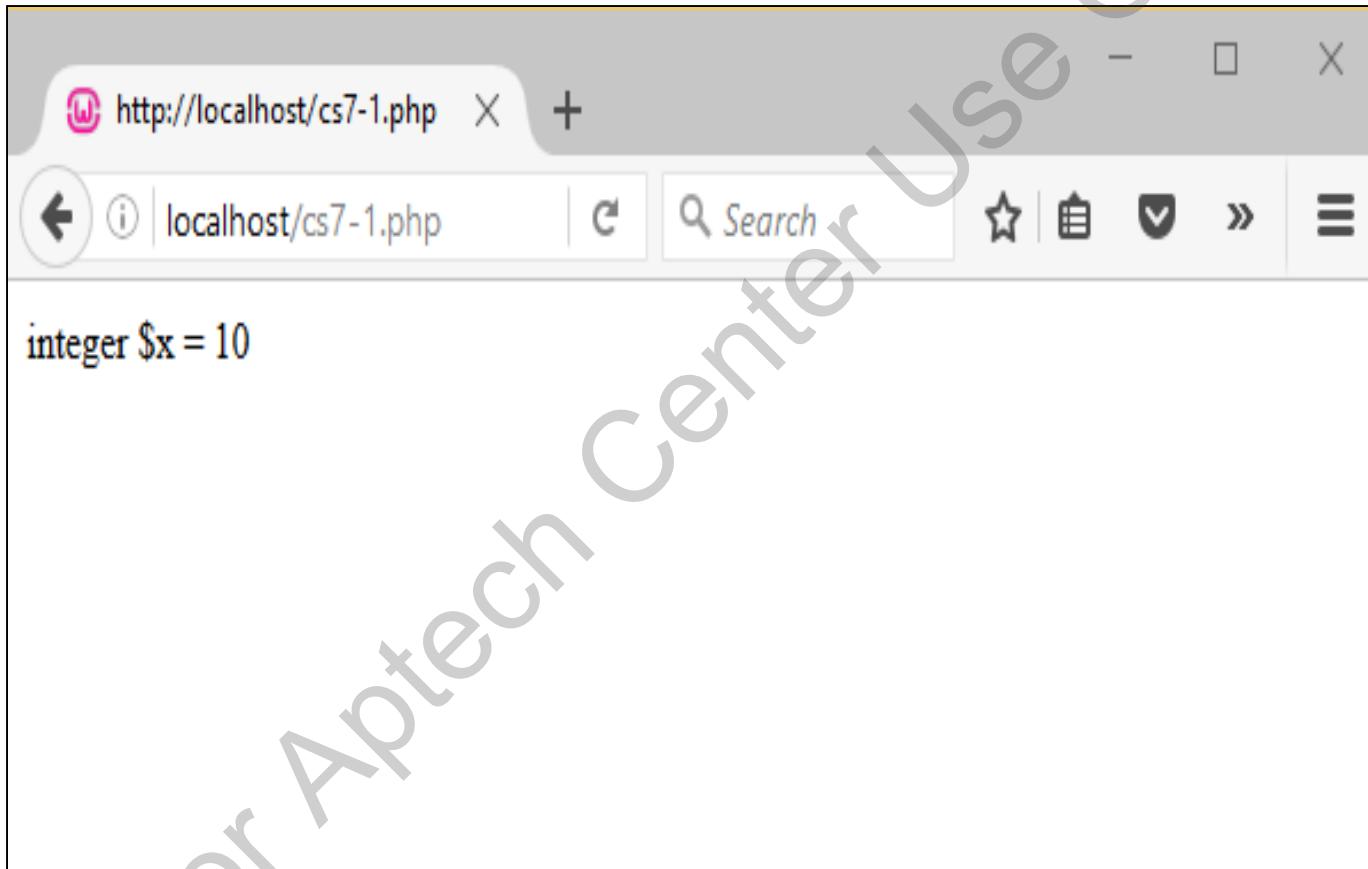
- ◆ Displaying the value of a parameter in `int` data type

Snippet

```
<?php
function test1(int $x) {
echo 'integer $x = '. $x;
}
test1(10.124);
//output: integer $x = 10
?>
```

- ❖ `test1` - is the function name
- ❖ `int` – is the data type
- ❖ `$x` – is the parameter

Displays the following output:



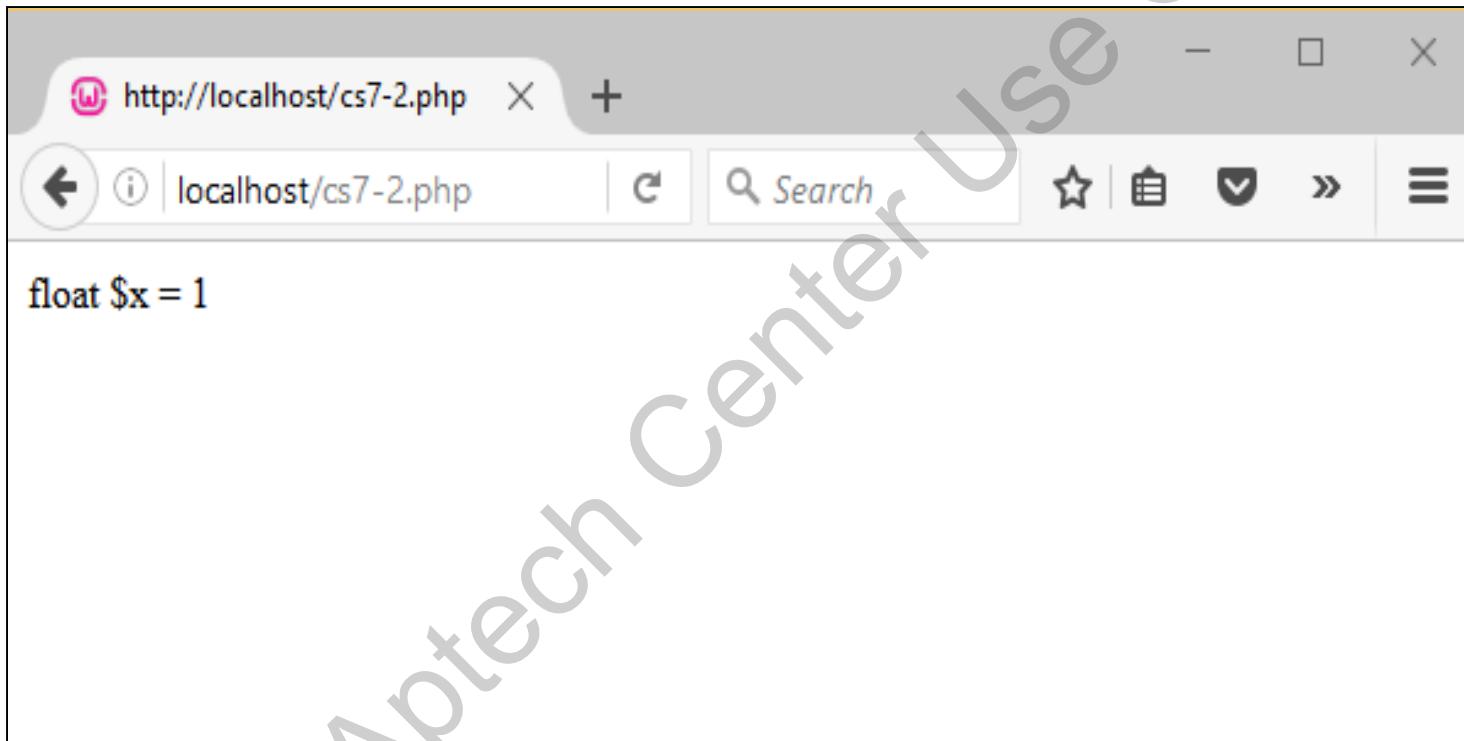
- ◆ Displaying the value of a parameter in float data type

Snippet

```
<HTML>
<BODY>
<?php
function test1(float $x) {
echo 'float $x = '. $x;
}
test1(true);
?>
</BODY>
</HTML>
```

- ◆ test1 – is the function name
- ◆ float – is the data type
- ◆ \$x – is the parameter

Displays the following output:



A screenshot of a web browser window. The address bar shows the URL `http://localhost/cs7-2.php`. The main content area of the browser displays the PHP code `float $x = 1`. The browser interface includes standard controls like minimize, maximize, and close buttons at the top right, and navigation buttons like back, forward, and search at the top left.

```
float $x = 1
```

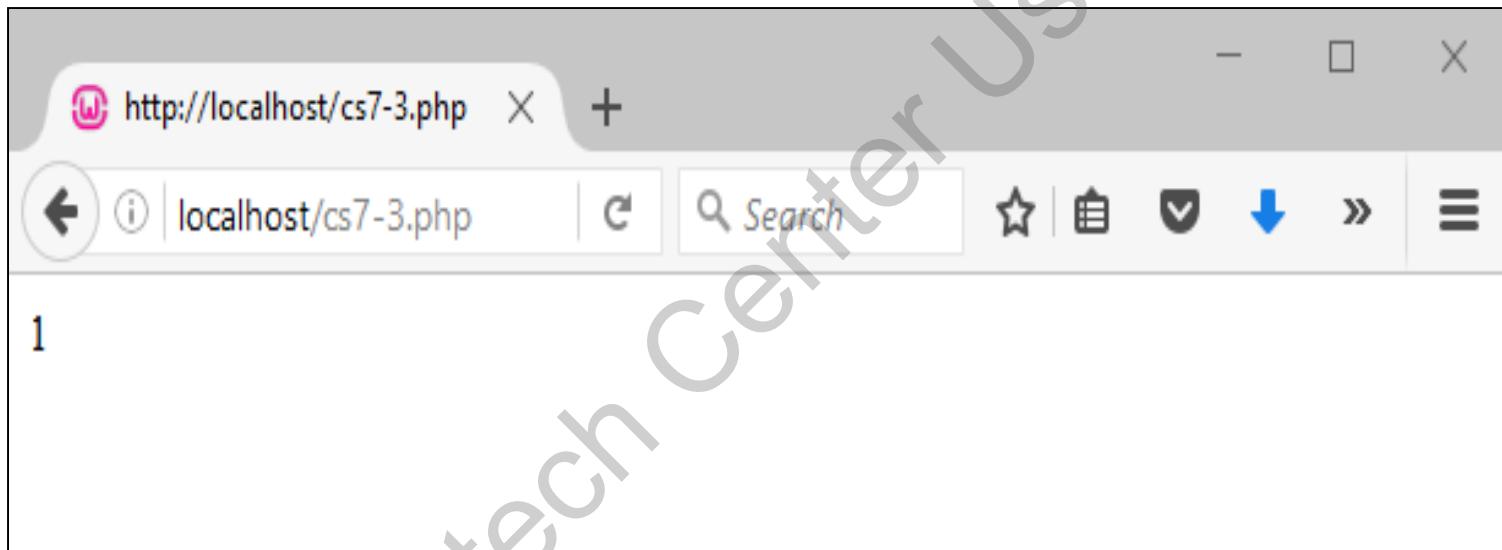
- ◆ Displaying the value of a parameter in boolean data type

Snippet

```
<HTML>
<BODY>
<?php
    function test1(bool $a) {
        echo $a;
    }
    test1(10.34); //
?>
</BODY>
</HTML>
```

- ◆ test1 - is the function name
- ◆ bool – is the data type
- ◆ \$a – is the parameter

Displays the following output:



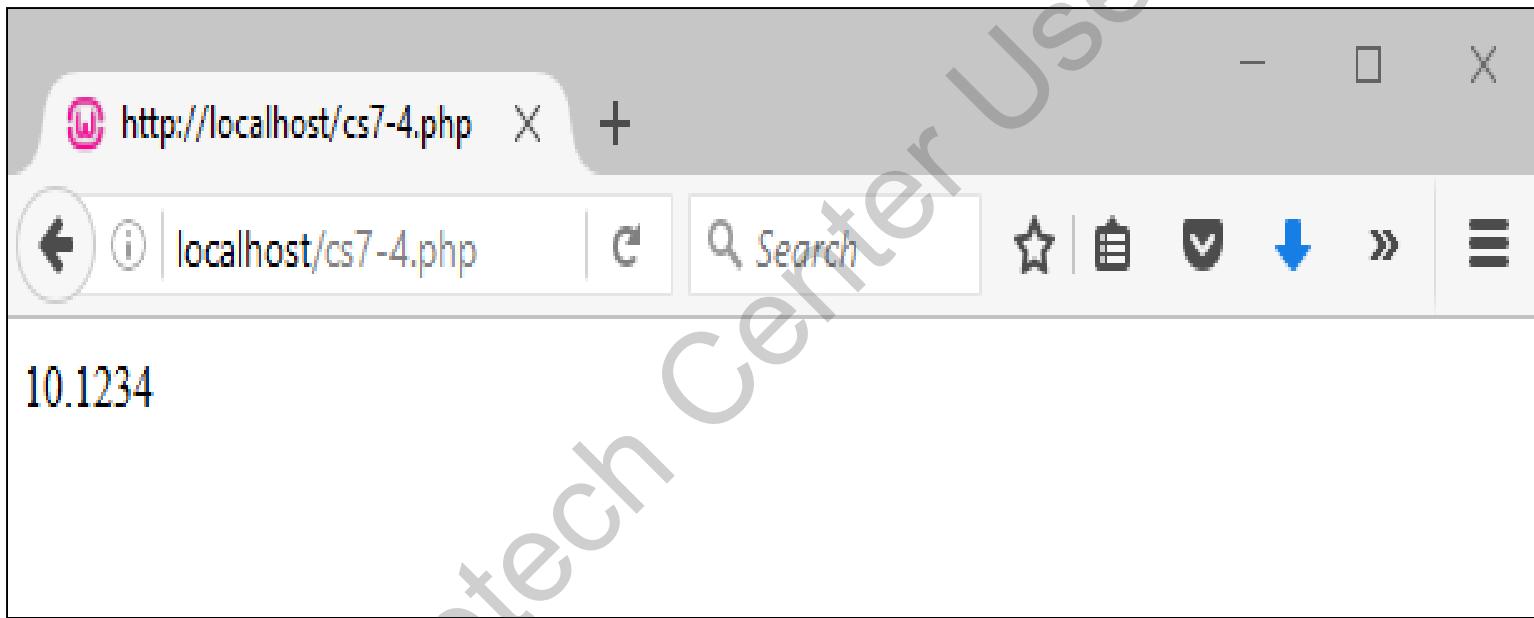
- ◆ Displaying the value of a parameter in string data type

Snippet

```
<HTML>
<BODY>
<?php
    function test1(string $a) {
        echo $a;
    }
    test1(10.1234);
?>
</BODY>
</HTML>
```

- ◆ test1 – is the function name
- ◆ string – is the data type
- ◆ \$a – is the parameter

Displays the following output:



- ◆ Demonstrating strict type declaration

Snippet

```
<?php declare(strict_types=1);  
function test1(int $a) {  
    echo $a;  
}  
test1(true);  
?>
```

The **declare** statement explicitly declares the scalar type, that are strictly checked.

The **function** **test1** is the function name that has an **int** data type.

The **echo** statement prints the value of the argument.

Displays the following output:

The screenshot shows a web browser window with the URL `http://localhost/cs7-5.php`. The page displays two error messages in orange boxes:

(!) Fatal error: Uncaught TypeError: Argument 1 passed to test() must be of the type integer, boolean given, called in C:\wamp64\www\cs7-5.php on line 6 and defined in C:\wamp64\www\cs7-5.php on line 3

(!) TypeError: Argument 1 passed to test() must be of the type integer, boolean given, called in C:\wamp64\www\cs7-5.php on line 6 in C:\wamp64\www\cs7-5.php on line 3

Below the errors is a table titled "Call Stack" with the following data:

#	Time	Memory	Function	Location
1	0.0011	360368	{main}()	...\cs7-5.php:0
2	0.0011	360368	test()	...\cs7-5.php:6

The program fails to execute and terminates prematurely because the argument types does not match the parameter type.

- Weak type conversion is enforced using the `declare(strict_types=0)` statement
- Rules to be considered while using weak type checking:

Calls to a built-in PHP function or to an extension have the same behavior as earlier versions

Weak type new scalar type declarations are same as that of built-in PHP function or to an extension

NULL is not accepted unless it is a parameter and is explicitly given a default value

- ◆ An implicit scalar conversion in a weak mode

Type Declaration	int	float	string	bool
int	yes	yes	yes	yes
float	yes	yes	yes	yes
string	yes	yes	yes	yes
bool	yes	yes	yes	yes

Behavior of Strict Type Checking

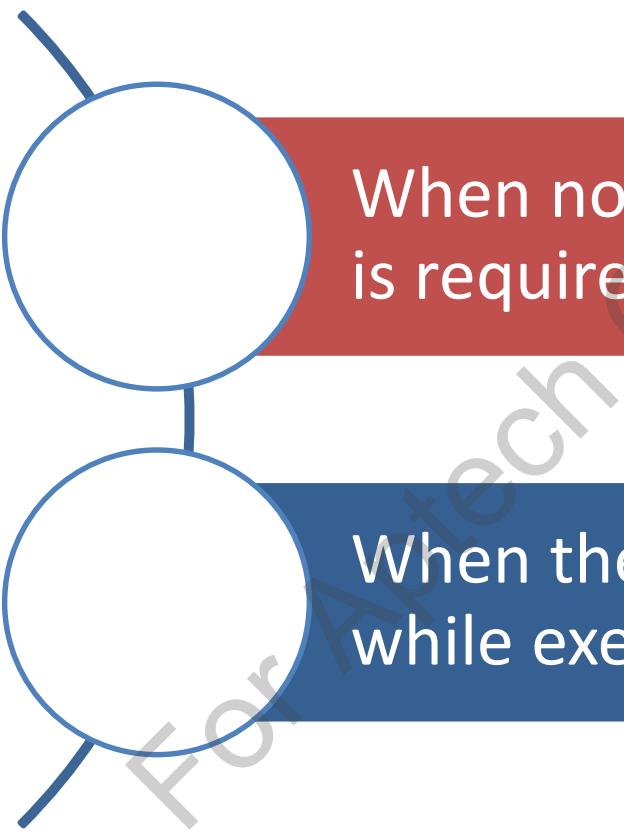
Add the strict type check mode to the declare statement.

Include the declare statement in the beginning of the PHP file.

If it does not match, then it will accept the value with type mismatch and the output will be an error.

If the value type matches the type of the parameter declared then, the parameter will be accepted.

- ◆ Is a class that does not have a name
- ◆ Can be used in the following scenarios:



When no documentation of the class is required

When the class is used only once while execution

- ◆ Demonstrating creation of a named class

Step 1

Use the keyword `class` before the class name

Step 2

Enclose the property and method definitions
within curly braces

```
class class_Name {  
    // defined properties and methods  
};  
$object = new class_Name( 'arguments' );
```

Step 1

Create a public variable.

Step 2

Assign values to the public variables.

Step 3

Create a function within the anonymous class without any arguments.

Step 4

Create a second function within the anonymous class with one argument.

The first function will return a message to the user whereas the second function will return the value of the argument.

Snippet

```
<?php
// anonymous_class.php
// PHP 7

$anon_class_obj = new class{
    public $greeting = 'hello';
    public $Id = 754;
    const SETT = 'some configuration';

    public function getValue()
    {
        // do some operation
        return 'some returned value';
    }

    public function getValueWithArg($str1)
    {
        // do some operation
        return 'returned value is '.$str1;
    }
};
```

\$greeting, \$Id, and SETT are members of the anonymous class.

'hello', 754, and 'some configuration' are the values assigned to these members.

getValue and
getValueWithArgument are the functions within the class.

The getValue function will display the message 'Some Returned Value'

The getValueWithArgument function will display the value of its argument.

Snippet

```
echo '</br>';

echo $anon_class_obj->greeting;
echo '</br>';
echo $anon_class_obj->Id;
echo '</br>';
echo $anon_class_obj::SETT;
echo '</br>

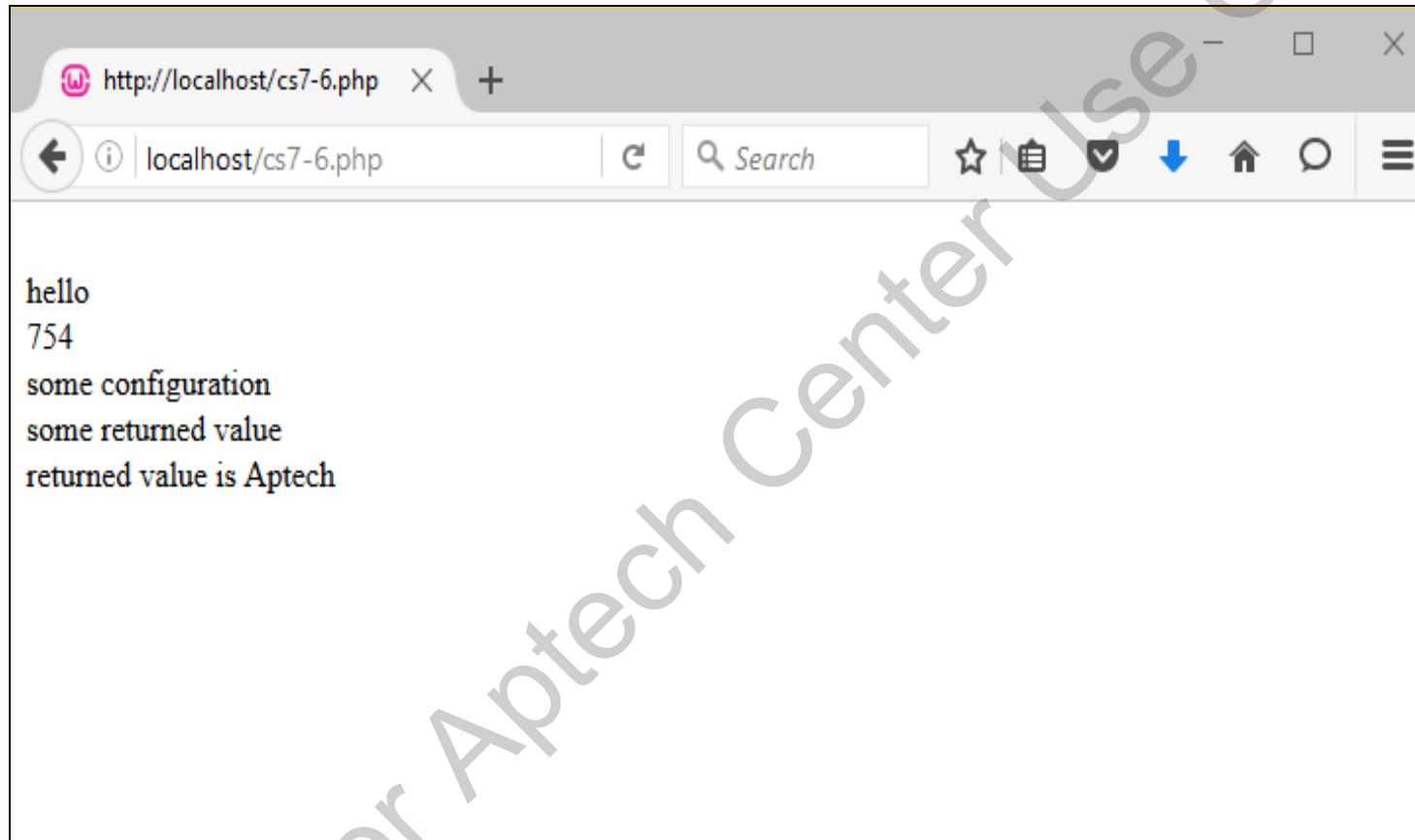
echo $anon_class_obj->getValue();
echo '</br>';

echo $anon_class_obj->getValueWithArg('Aptech');
echo '</br>';

echo '</br>';
?>
```

The echo statements will display the values of the members and functions of the anonymous class.

Displays the following output:



Summary

- Scalar data types are those types that hold single values. These data types can either be integer, string, Boolean, or float.
- Type declaration refers to process of specifying the data type of the parameter when passing it to a function. They are also known as type hinting.
- Scalar type hints help design reliable PHP code.
- To enable strict type checking, use the declare statement to declare strict_types directive. Any type mismatch with function arguments results in an error.



- ◆ An anonymous class is a class that is defined without a name.
- ◆ To create anonymous class, combine new class (\$constructor, \$args) followed by a standard class definition.

For Aptech Certified Use Only

PHP Operators

Session 8



Objectives

- ◆ *Explain the use of arithmetic operators*
- ◆ *Explain the use of logical operators*
- ◆ *Explain the use of relational operators*
- ◆ *Explain the use of bitwise operators*
- ◆ *Explain the use of assignment operators*
- ◆ *Explain the use of string operators*
- ◆ *Explain the use of increment and decrement operators*
- ◆ *Explain the use of conditional or ternary operators*
- ◆ *Explain the precedence of operators*

- ◆ All programming languages use operators
- ◆ Operators
 - ❖ Are pre-defined symbols that perform specific actions on objects called operands
 - ❖ Are assigned a precedence value indicating the order in which the operators are evaluated

- ◆ Operators are classified as follows:
 - ❖ **Unary Operator** - acts on only one operand in an expression
 - ❖ **Binary Operator** - has two operands and performs different arithmetic and logical operations
 - ❖ **Conditional or Ternary Operator** – has three operands and evaluates the second or third expression depending on the result of the first expression

- ◆ Are binary operators that work only on numeric operands
- ◆ If operands are non-numeric values such as strings, Booleans, nulls, or resources, they are converted to their numeric equivalents

Table lists the arithmetic operators supported by PHP

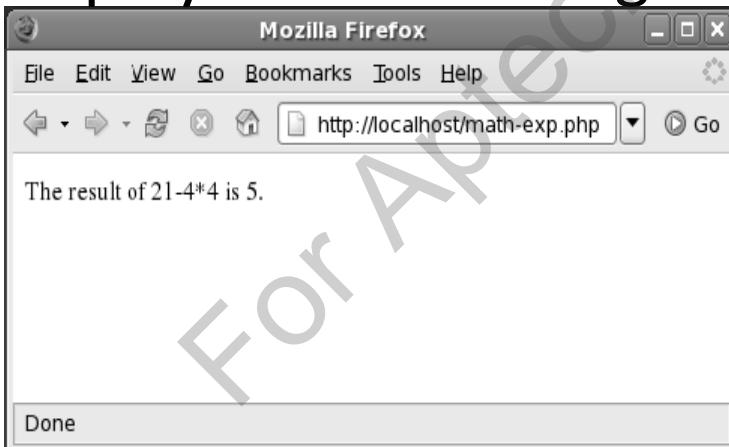
Operator	Name	Description
+	Addition	Returns the sum of the operands
-	Subtraction	Returns the difference between the two operands
*	Multiplication	Returns the product of two operands
/	Division	Returns the quotient after dividing the first operand by the second operand
%	Modulus	Returns the remainder after dividing the first operand by the second operand

- ◆ Displaying a mathematical expression
 - ❖ Enter the code as shown in a PHP script named `math-exp.php`

Snippet

```
<?php  
$var1 = 21-4*4;  
echo "The result of 21-4*4 is $var1."  
?>
```

Displays the following output:



In the code, if you follow the Brackets Order Division Multiply Add Subtract (BODMAS) rule of mathematics, the result will be 8.

PHP follows the BODMAS rule for calculation.

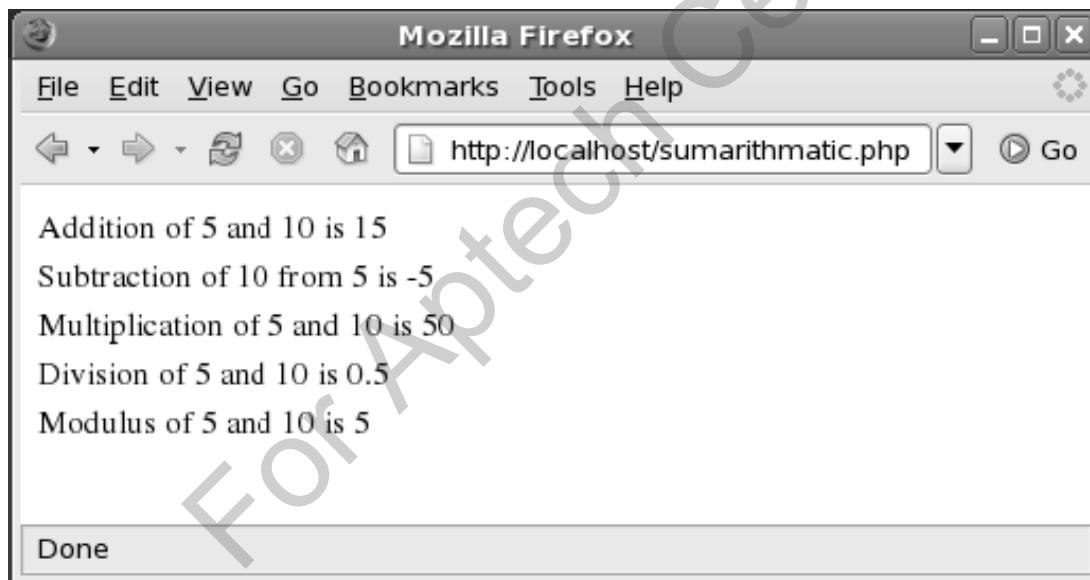
- ◆ Displaying the use of arithmetic operators
 - ❖ Enter the code in a PHP script named **sumarithmetic.php**

Snippet

```
<?php  
$VAR1=8;  
$VAR2=10;  
//Addition  
$SUM=$VAR1+$VAR2;  
//Subtraction  
$DIFFERENCE = $VAR1-$VAR2;  
//Multiplication  
$PRODUCT = $VAR1*$VAR2;  
//Division  
$QUOTIENT = $VAR1/$VAR2;  
//Modulus  
$REMAINDER = $VAR1%$VAR2;
```

```
echo "Addition of 5 and 10 is ".$SUM."  
echo "Subtraction of 10 from 5 is ".$DIFFERENCE."  
echo "Multiplication of 5 and 10 is ".$PRODUCT."  
echo "Division of 5 and 10 is ".$QUOTIENT."  
echo "Modulus of 5 and 10 is ".$REMAINDER."  
?>
```

Displays the following output:



- ◆ Compares two operands and returns either a true or a false value
- ◆ The operands can either be numbers or string values

Table lists relational operators along with its description

Operator	Name	Description
<code>==</code>	Equal to	Returns true if both the operands are equal
<code>===</code>	Identical	Returns true if both the operands are equal and are of the same data type (Introduced in PHP 4)
<code>!=</code>	Not equal to	Returns true if the first operand is not equal to the second operand
<code><></code>	Not equal to	Returns true if the first operand is not equal to the second operand
<code>!==</code>	Not Identical	Returns true if the first operand is not equal to the second operand or they are not of the same data type (Introduced in PHP 4)
<code><</code>	Less than	Returns true if the first operand is less than the second operand
<code><=</code>	Less than or equal to	Returns true if the first operand is less than or equal to the second operand
<code>></code>	Greater than	Returns true if the first operand is greater than the second operand
<code>>=</code>	Greater than or equal to	Returns true if the first operand is greater than or equal to the second operand

Relational Operators

- ◆ Showing the difference between '==' 'equal and '===' identical relational operators in PHP

Snippet

```
<?php  
if("10" == 10)  
    echo "YES";  
else  
    echo "NO";  
?>
```

Displays the following output:



The code will print YES because the values of the operands are equal.

Relational Operators

- ◆ Showing the difference between '==' 'equal and '===' identical relational operators in PHP

Snippet

```
<?php  
if("10" === 10)  
    echo "YES";  
else  
    echo "NO";  
?>
```

Displays the following output:



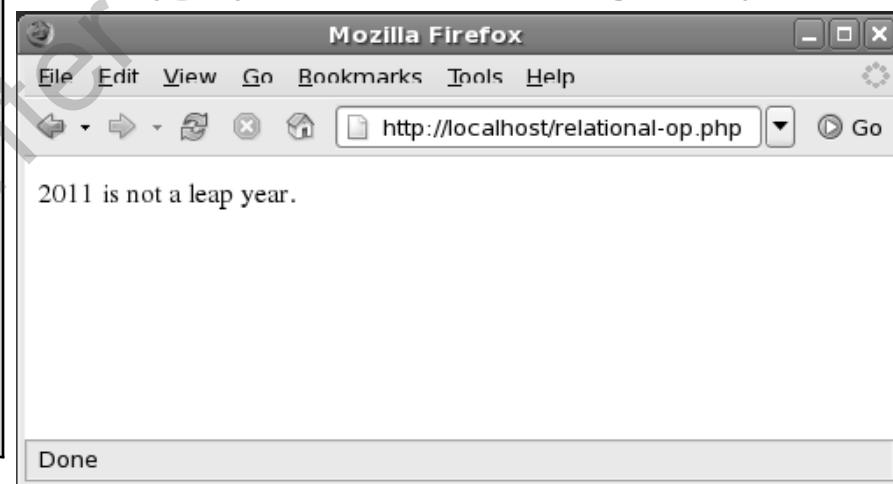
The code will print NO because although values of both operands are same, their data types are different. "10" is a string while 10 is an integer.

- ◆ Checking whether the given year is a leap year or not:
 - ❖ Enter the code as shown in a script named **relational-op.php**

Snippet

```
<?php  
$y = 2011;  
if(($y%4==0 && $y%100!=0) || ($y%400 == 0))  
{  
    echo "$y is a leap year. <br />";  
}  
else  
{  
    echo "$y is not a leap year. <br />";  
}  
?>
```

Displays the following output:



The code uses an `if` conditional statement to execute a block of code only when the specified condition is `true` else it executes the block of code in the body of the `else` statement.

The output of the code will be 2011 is not a leap year.

- ◆ Enable to combine two or more expressions in a condition
- ◆ Evaluates the expression and returns a Boolean value of either true **or** false

Table lists various logical operators

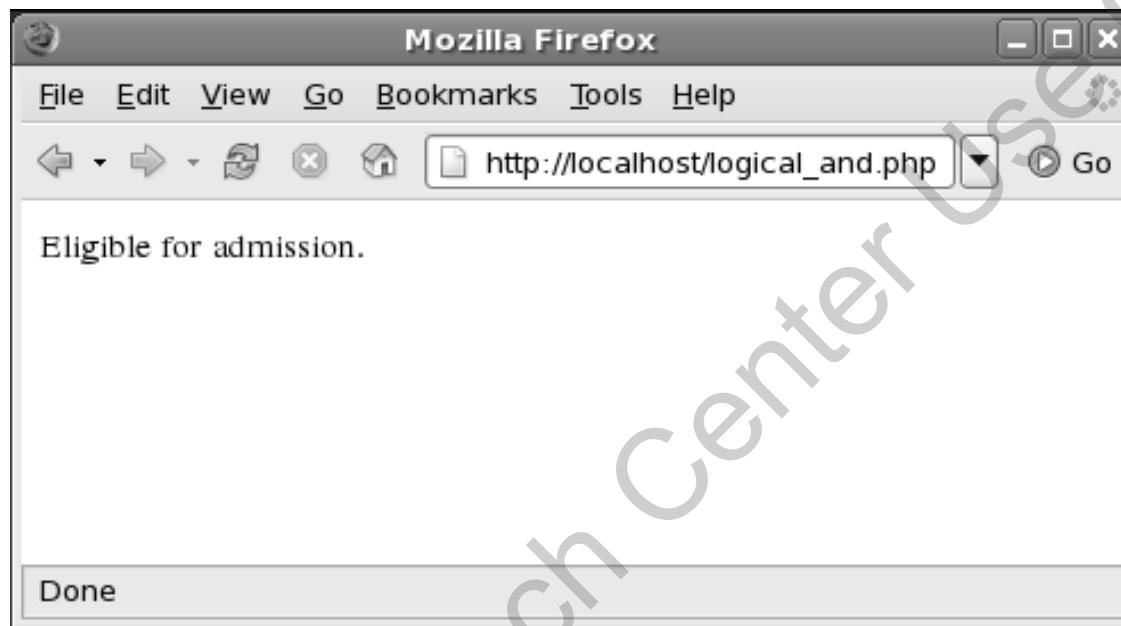
Operator	Name	General Form	Description
AND <code>&&</code>	Logical AND operator	<code>Expression1 && Expression2</code> <code>Expression1 && Expression2</code>	Returns true only if both the expressions are true
OR <code> </code>	Logical OR operator	<code>Expression1 Expression2</code> <code>Expression1 Expression2</code>	Returns true if any one of the expression is true
XOR	Logical XOR operator	<code>Expression1 XOR Expression2</code>	Returns true if either Expression 1 or Expression 2 is true, but not both
!	Logical NOT operator	<code>!Expression</code>	Returns true only if the condition is not true

- ◆ Using a logical AND operator to check if a student's percentage is greater than 60 and the year of passing is 2003
 - ❖ Enter the code as shown in a PHP script named **logical_and.php**

Snippet

```
<?php
$Percentage = 70;
$Year = "2003";
if ($Percentage>60 AND $Year=="2003")
{
    echo "Eligible for admission.";
}
?>
```

Displays the following output:



In the code, the AND operator requires both the expressions to be true.

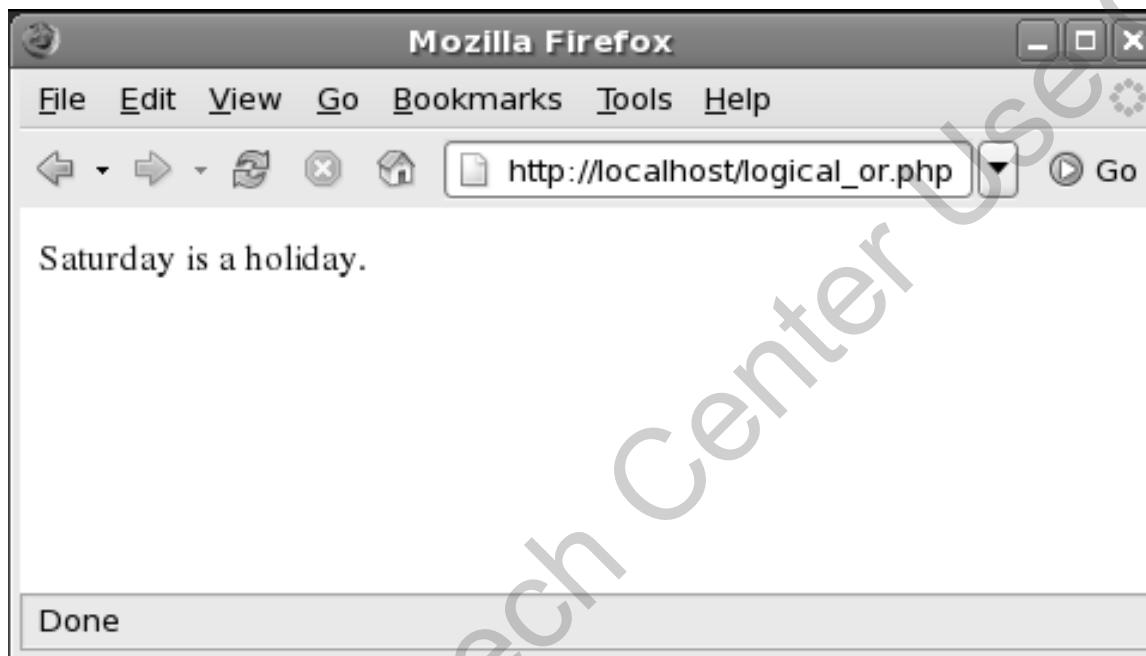
Only then, the block of code following the `if` statement is executed.

- ◆ Displaying the use of the OR operator
 - ❖ Enter the code as shown in a PHP script named, **logical_or.php**

Snippet

```
<?php
$day1="Saturday";
if(( $day1=="Saturday") || ($day1=="Sunday"))
{
    echo "$day1 is a holiday.";
}
else
{
    echo "$day1 is a working day.";
}
?>
```

Displays the following output:



In the code, the OR operator requires any one of the expressions to be true, and executes the block of code after the if statement.

- ◆ Operate on the bitwise representation of their operands
- ◆ Work on small-scale binary representation of data
- ◆ If the operand is a string, the operation is performed only after converting it to its corresponding integer representation

Table lists bit-wise operators

Operator	Name	General Form	Description
&	AND	Operand1 & Operand2	Compares two bits and sets the result to 1 if both the bits are 1 and 0 otherwise
	OR	Operand1 Operand2	Compares two bits and sets the result to 1 if either of the bits are 1 and 0 otherwise
^	EXCLUSIVE-OR	Operand1 ^ Operand2	Compares two bits and sets the bit to 1 if the bits are different and 0 otherwise
~	COMPLEMENT	~ Operand	Compares and sets the 0 bits to 1 and vice versa
<<	SHIFT LEFT	Operand1 << Operand2	Shifts the bits of Operand1, Operand2 steps to the left (each step means "multiply by two")
>>	SHIFT RIGHT	Operand1 >> Operand2	Shifts the bits of Operand1, Operand2 steps to the right (each step means "divide by two")

Table lists binary comparisons of bit-wise operators

X	Y	X & Y	X Y	$\sim X$	$\sim Y$	X ^ Y
1	1	1	1	0	0	0
1	0	0	1	0	1	1
0	1	0	1	1	0	1
0	0	0	0	1	1	0

- ◆ Displaying the use of bitwise operations
 - ◆ Enter the code in a PHP script named **bitwise.php**

Snippet

```
<?php
$x= 50;
$y= 5;
echo "\$x & \$y = ".($x & $y)."  
";
echo "\$x | \$y = ".($x | $y)."  
";
echo "\$x ^ \$y = ".($x ^ $y)."  
";
echo "~(\$y) = ".~$y."  
";
//x is divided by 2 y times
echo "\$x >> \$y = ".($x >> $y)."  
";
//x is multiplied by 2 y times
echo "\$x << \$y = ".($x << $y)."  
";
//Converts the operands to their ASCII values first
('5' (ascii 53))^( '9' (ascii 57))
echo "The Bitwise result of 5 ^ 9 is:".(5 ^ 9)."  
";
//Converts "8" to perform the operation (5 ^ ((int)"8"))
echo "The result of 5 ^ 8 is: ".(5 ^ 8). " "  

?>
```

Displays the following output:

The screenshot shows a Mozilla Firefox browser window with the title bar "Mozilla Firefox". The menu bar includes "File", "Edit", "View", "Go", "Bookmarks", "Tools", and "Help". Below the menu is a toolbar with icons for back, forward, search, and home. The address bar displays the URL "http://localhost/bitwise.php". The main content area of the browser shows the following text output from a PHP script:

```
$x & $y = 0
$x | $y = 55
$x ^ $y = 55
~($y) = -6
$x >> $y = 1
$x << $y = 1600
The Bitwise result of 5 ^ 9 is:12
The result of 5 ^ 8 is: 13
```

At the bottom of the browser window, there is a "Done" button.

- ◆ Enables to assign a value to a variable
- ◆ The '=' sign is the assignment operator

Syntax

```
expression 1 = expression 2;
```

- ◆ Operand on the left side of the assignment operator '=' should be a variable
- ◆ The assignment operation can be performed by:
 - ❖ **Value** - copies the value of expression 2 and assigns it to expression 1
 - ❖ **Reference** - assigns a reference of expression 2 to expression 1
- ◆ The basic assignment operator can also be used as a shorthand operators (combined operator) in conjunction with arithmetic and string operators
- ◆ The shorthand operators are +=, -=, *=, /=, %=, and .=

Table displays various examples with shorthand operators

combihand	Expression	Description
<code>\$a+= \$b</code>	<code>\$a = \$a + \$b</code>	Adds \$a and \$b and assigns the result to \$a
<code>\$a-= \$b</code>	<code>\$a = \$a - \$b</code>	Subtracts \$b from \$a and assigns the result to \$a
<code>\$a*= \$b</code>	<code>\$a = \$a*\$b</code>	Multiplies \$a and \$b and assigns the result to \$a
<code>\$a/= \$b</code>	<code>\$a = \$a/\$b</code>	Divides \$a by \$b and assigns the quotient to \$a
<code>\$a%= \$b</code>	<code>\$a = \$a%\$b</code>	Divides \$a by \$b and assigns the remainder to \$a
<code>\$a .= \$b</code>	<code>\$a=\$a.\$b</code>	Concatenates \$b with \$a and assigns the result to \$a

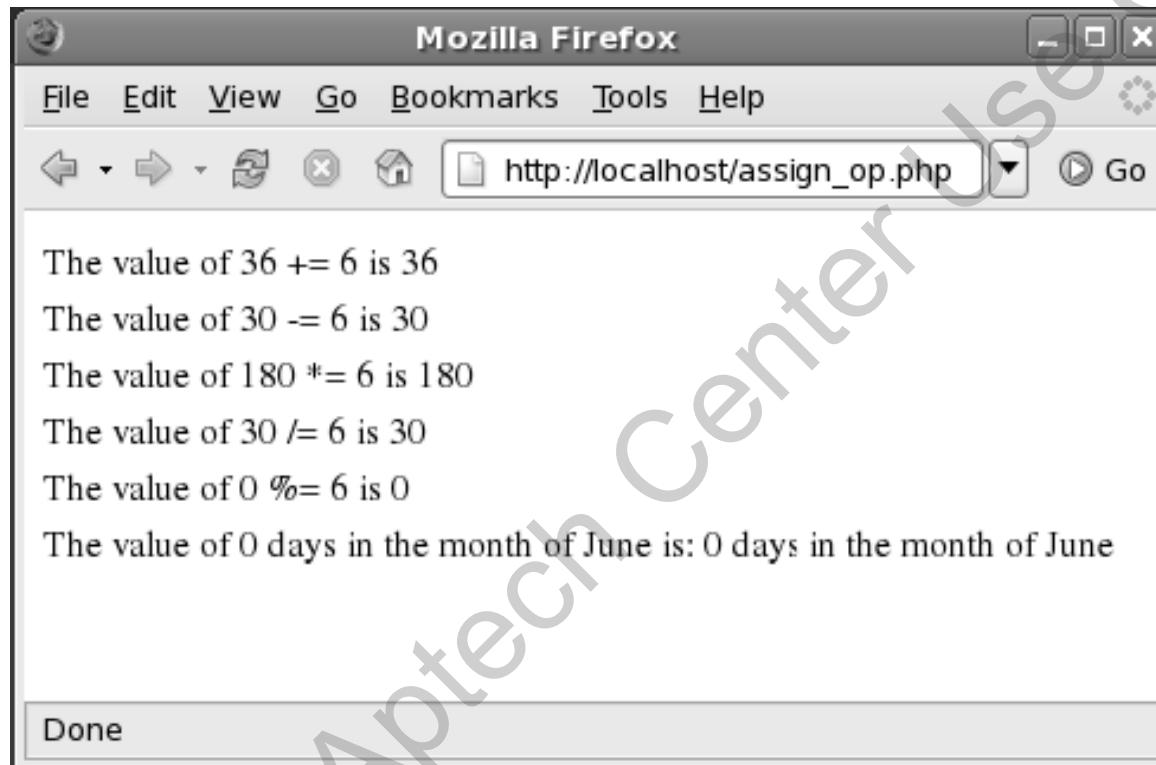
- ◆ Using shorthand operators

- ❖ Enter the code in a PHP script named **assign_op.php**

Snippet

```
<?php
$a = 30;
$b = 6;
$a+= $b;
echo "The value of $a += $b is ". $a . "<br>";
$a-= $b;
echo "The value of $a -= $b is ". $a . "<br>";
$a*= $b;
echo "The value of $a *= $b is ". $a . "<br>";
$a/= $b;
echo "The value of $a /= $b is ". $a . "<br>";
$a%=$b;
echo "The value of $a %= $b is ". $a . "<br>";
$a.= " days in the month of June";
echo "The value of $a is: " . $a . "<br>";
?>
```

Displays the following output:



- ◆ Operate only on variables
- ◆ Cause a variable to change its value
- ◆ The increment operators increase the value of the operand by one
- ◆ The decrement operators decrease the value of the operand by one

Table lists increment and decrement operators

Operand	Operator Name	Description
<code>++\$a</code>	Pre-increment	Increments the operand value by one and then returns this new value to the variable
<code>\$a++</code>	Post-increment	Returns the value to the variable and then increments the operand by one
<code>--\$a</code>	Pre-decrement	Decrements the operand by one and then returns this new value to the variable
<code>\$a--</code>	Post decrement	Returns the value to the variable and then decrements the operand by one

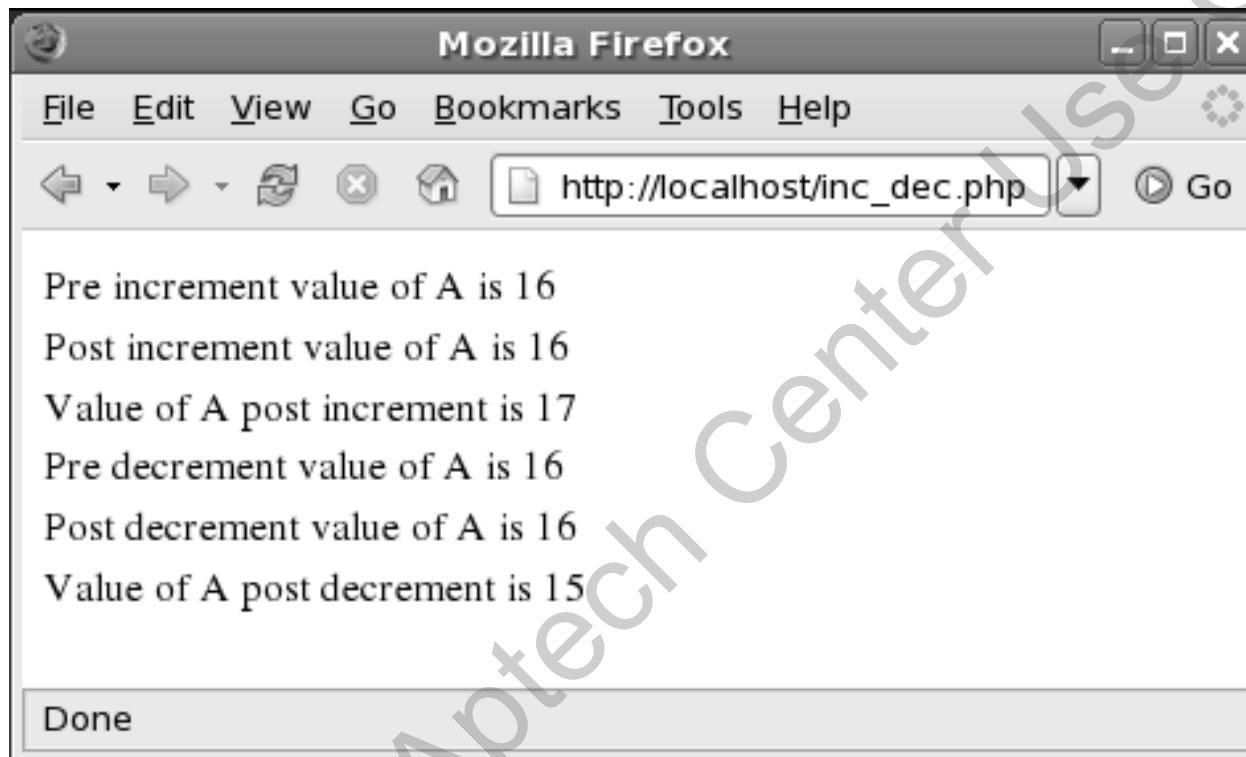
- ◆ Displaying the use of the increment and decrement operators on a variable \$A
 - ❖ Enter the code as shown in a PHP script named **inc_dec.php**

Snippet

```
<?php
$A=15;

echo "Pre increment value of A is ".++$A."<br/>";
echo "Post increment value of A is ".$A++. "<br/>";
echo "Value of A post increment is ".$A."<br/>";
echo "Pre decrement value of A is ".--$A."<br/>";
echo "Post decrement value of A is ".$A--. "<br/>";
echo "Value of A post decrement is ".$A--.<br/>;
?>
```

Displays the following output:

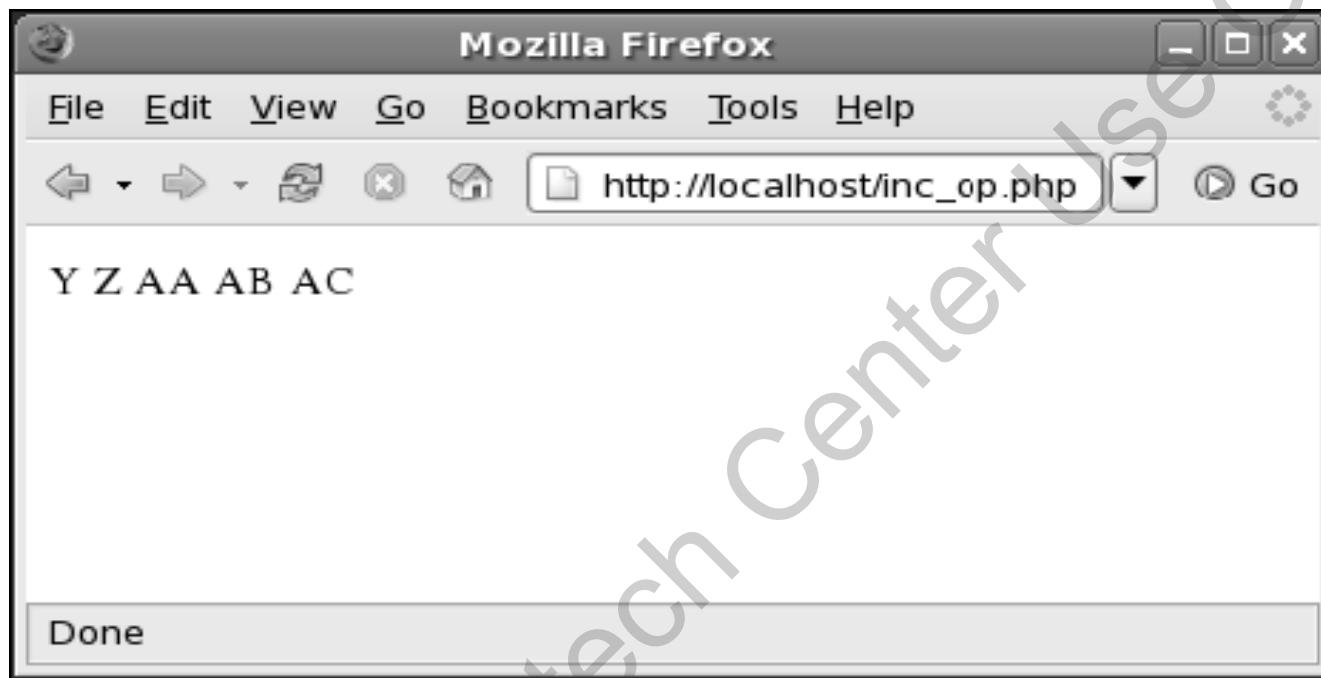


- ◆ Displaying the use of increment operator on a character variable
 - ❖ Enter the code as shown in a PHP script named **inc_op.php**

Snippet

```
<?php
$i = 'X';
for ($n=0; $n<5; $n++)
{
echo ++$i . "\n";
}
?>
```

- ◆ Displays the following output:



In the code, the value of the variable `$i` is incremented five times using a for loop statement and each value is printed. Here `++$i`, i.e., 'X'+1 returns 'Y' followed by 'Z' and 'Z'+1 returns 'AA' and 'AA'+1 returns 'AB'.

- ◆ Operate only on character data
- ◆ Non-string operand is first converted before the operation is executed

Table lists string operators

Operator	Name	Description
.	Concatenation	Returns a concatenated string
.=	Concatenating assignment	Appends the argument on the right side of the operator to the arguments on the left side

- ◆ Displaying the concatenation of the strings WELCOME and FRIENDS using the Concatenation operator
 - ❖ Enter the code as shown in a PHP script named **concat.php**

Snippet

```
<?php  
$A="WELCOME ";  
$B="FRIENDS!";  
$C=$A.$B;  
echo "The concatenated string is $C";  
?>
```

- ◆ Displays the following output:



In the code, the values of the variables \$A and \$B are concatenated and stored in a variable \$C.

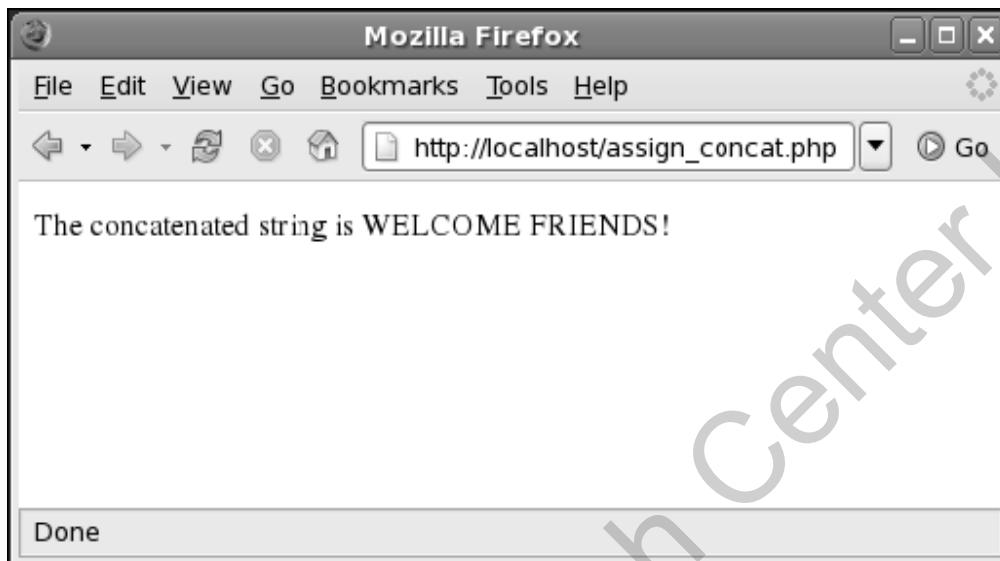
It returns the string WELCOME FRIENDS! as the output.

- ◆ Assigning a value to a variable using the concatenating assignment operator
 - ❖ Enter the code as shown in a PHP script named **assign_concat.php**

Snippet

```
<?php  
$A="WELCOME";  
$A.= " FRIENDS!";  
echo " The concatenated string is $A";  
?>
```

- ◆ Displays the following output:



In the code, the argument FRIENDS! is appended on the right side of \$A containing the argument WELCOME.

The string returned is WELCOME FRIENDS!.

Conditional or Ternary Operators

- ◆ An alternative to the if-else statement
- ◆ Evaluates an expression for a true or false value and then executes one of the two statements depending upon the result of evaluation

Syntax

```
$var1 = ($var2 = value1) ? expr1 : expr2
```

Where,

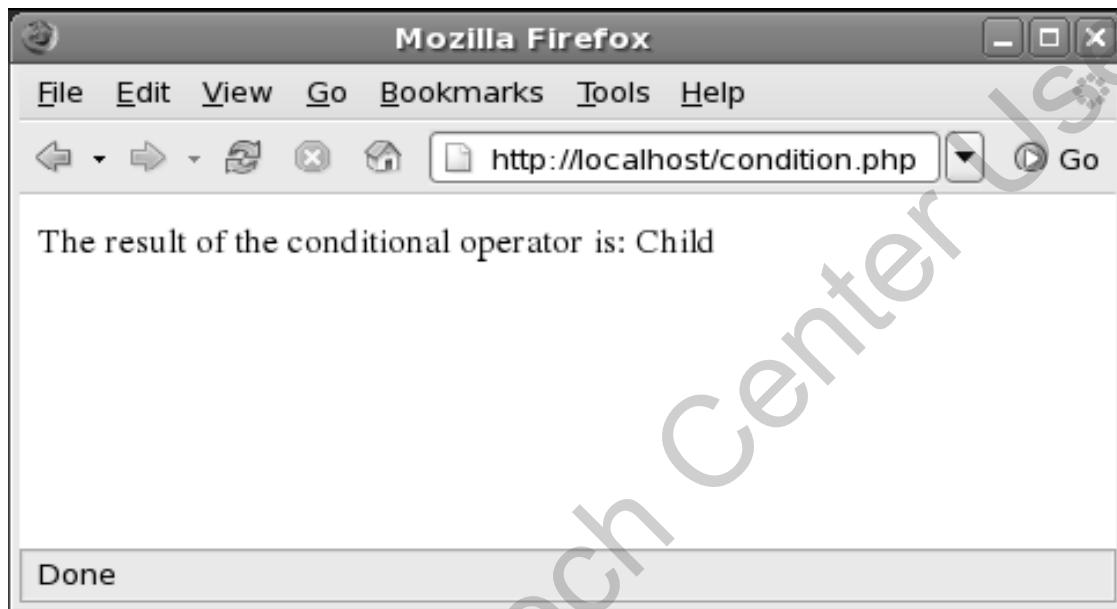
- ◆ The operator evaluates the \$var1 and if it is true, expr 1 is evaluated and if it is false, expr 2 is evaluated

- ◆ Displaying the use of a conditional operator
 - ❖ Enter the code as shown in a PHP script named **condition.php**

Snippet

```
<?php
$age = 15;
$category = ($age < 16) ? 'Child' : 'Adult';
echo "The result of the conditional operator is: ";
echo $category;
?>
```

- ◆ Displays the following output:



The code evaluates the condition (`$age < 16`).

If it is true, then the value `Child` is assigned to `$category`, else the value `Adult` is assigned to `$category`.

- ◆ The order of precedence to evaluate the expression in PHP is:
 - ❖ Operators with higher precedence are evaluated first in an expression
 - ❖ Operator precedence can be overridden by using parenthesis. The expression within the parenthesis is evaluated first
 - ❖ If two operators of the same precedence are encountered, the expression will be evaluated from left to right

Table lists the precedence of operators in PHP, with the highest precedence operators listed at the top

Precedence	Category	Operators	Associativity
Highest Precedence	Unary	!, ++, --	Right to left
	Multiplicative	*, /, %	Left to right
	Additive	+, -, .	Left to right
	Relational	<, <=, >, >=	Left to right
	Equality	==, !=	Left to right
	Logical AND	&&	Left to right
	Logical OR		Left to right
	Conditional	:?	Right to left
	Assignment	=, +=, -=, *=, /=, %=	Right to left
	Logical AND	AND	Left to right
Lowest Precedence	Logical XOR	XOR	Left to right
	Logical OR	OR	Left to right
	Comma	,	Left to right

- ◆ Operator is any symbol that performs an operation on an operand
- ◆ An operator enables you to work on variables, strings, and numbers and control the program flow
- ◆ The types of operators supported in PHP are arithmetic, logical, relational, bitwise, assignment, string, conditional, and increment and decrement operators
- ◆ The arithmetic operators work with numbers and are used to execute mathematical operations. PHP follows the BODMAS rule for operator precedence
- ◆ The relational operators compare two operands and determine the relationship between operands

- ◆ The logical operators evaluate multiple conditions and combine two or more test expression in a condition, returning a Boolean value
- ◆ The Bitwise operators enable comparison and manipulation of operands and operate on the bits of an operand
- ◆ The assignment operator enables assignment of values to a variable and defines the operand on the left-hand side to the value on the right-hand side
- ◆ The increment and decrement operators enable to increase or decrease value of an operand by one

- ◆ The conditional or ternary operator is an alternative to the if-else statement. It evaluates a condition and executes one of the two statements depending on the true or false result of the condition
- ◆ The concatenation operator combines two or more strings into a single string
- ◆ In a complex expression, operator precedence indicates the order in which the operands must be evaluated. PHP evaluates operators with high precedence before operators with low precedence

Conditional Statements in PHP

Session 10



Objectives

- ◆ *Explain the use of the if statement*
- ◆ *Explain the use of the switch statement*
- ◆ *Explain the use of the ternary (?) operator*

- ◆ A statement
 - ❖ Is a smallest element of any programming language
 - ❖ Consists of commands given by a programmer to a computer
 - ❖ Can be an individual statement or a group of statements within curly braces
 - ❖ Usually ends with a semicolon
- ◆ PHP script consists of a series of statements which are as follows:
 - ❖ An assignment
 - ❖ A function call
 - ❖ A conditional statement
 - ❖ An empty statement that does nothing

- ◆ Control the flow of a program on execution or skip code based on certain criteria
- ◆ Are of two types:
 - ❖ if statement
 - ❖ switch statement

- ◆ It is a common control structure
- ◆ It contains an expression called as truth expression
- ◆ The truth expression:
 - ◆ Can be a Boolean, variable, constant or an expression
 - ◆ Evaluates to true, false or NULL
 - ◆ If evaluates to true, following statements are executed
 - ◆ If evaluates to false or NULL, statements are not executed

Syntax

```
if(truth expression)
{
    Statements to be executed;
}
```

Where,

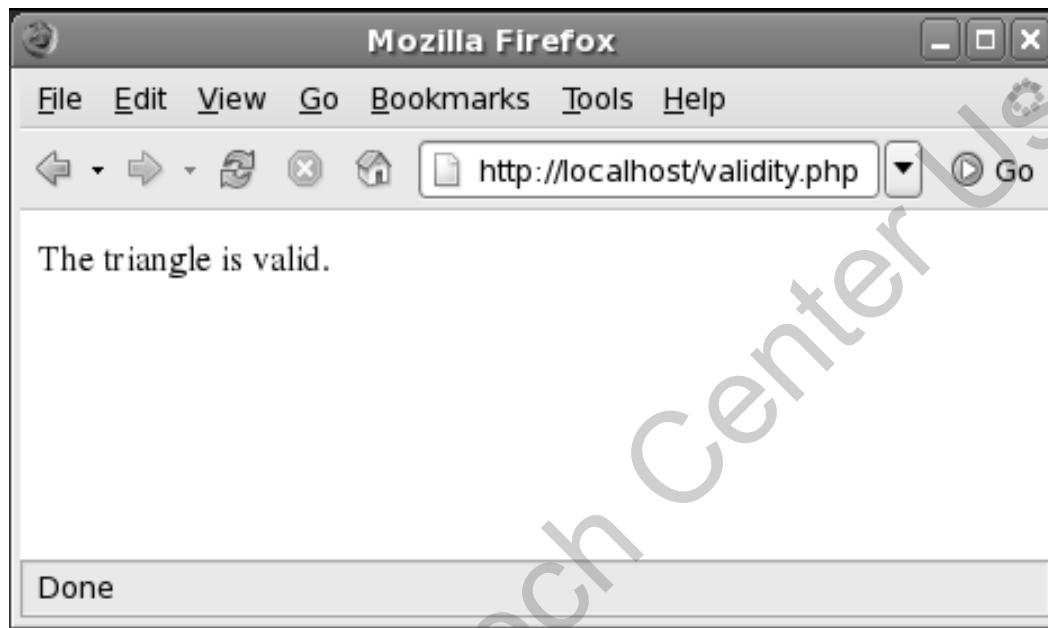
If keyword is followed by truth expression in parenthesis

- ◆ Checking whether a triangle is valid using the if statement
 - ❖ Enter the following code in a script named **validity.php**

Snippet

```
<html>
<body>
<?php
$a=60;
$b=60;
$c=60;
if ($a+$b+$c == 180)
echo "The triangle is valid.";
?>
</body>
</html>
```

Displays the following output:



If the sum of the degrees \$a, \$b, and \$c are equal to 180 then the statement following the `if` condition is executed

if Statement

- Accepting and displaying the salary of an employee based on the salary and bonus
 - salBonus.html – accepts user inputs

Snippet

```
<html>
<body>
<form action="salBonus.php" method="GET">
<table>
<tr>
<td>Salary &nbsp; </td>
<td><input type="text" name="sal"></td>
</tr>
</table>
<br>
<input type="submit" value="Submit">
</form>
</body>
</html>
```

- Accepting and displaying the salary of an employee based on the salary and bonus
 - salBonus.php – process the salary and calculate the bonus

Snippet

```
<?php
$sal = $_GET['sal'];
echo "Salary before bonus : $";
echo $sal;
echo "<br>";
if ($sal > 850)
{
    $bonus = $sal * .1;
    echo "Bonus : $$bonus";
    echo "<br>";
    $sal = $sal + $bonus;
    echo "Total Salary : $$sal";
}
?>
```

salBonus.html displays the following output:

A screenshot of a Mozilla Firefox browser window. The title bar says "Mozilla Firefox". The menu bar includes "File", "Edit", "View", "Go", "Bookmarks", "Tools", and "Help". The toolbar below has icons for back, forward, search, and home. The address bar shows "http://localhost/salBonus.html". The main content area contains a form with a label "Salary" followed by an empty input field, and a "Submit" button. At the bottom, there is a "Done" button.

salBonus.php displays the following output:

A screenshot of a Mozilla Firefox browser window. The title bar says "Mozilla Firefox". The menu bar includes "File", "Edit", "View", "Go", "Bookmarks", "Tools", and "Help". The toolbar below has icons for back, forward, search, and home. The address bar shows "http://localhost/salBonus.php?". The main content area displays the following text:
Salary before bonus : \$900
Bonus : \$90
Total Salary : \$990
At the bottom, there is a "Done" button.

The code accepts and displays the bonus at the rate of 10% and total salary when the salary of the employee is greater than \$850.

- ◆ Is used along with if statement
- ◆ Is executed when a specified condition is false

Syntax

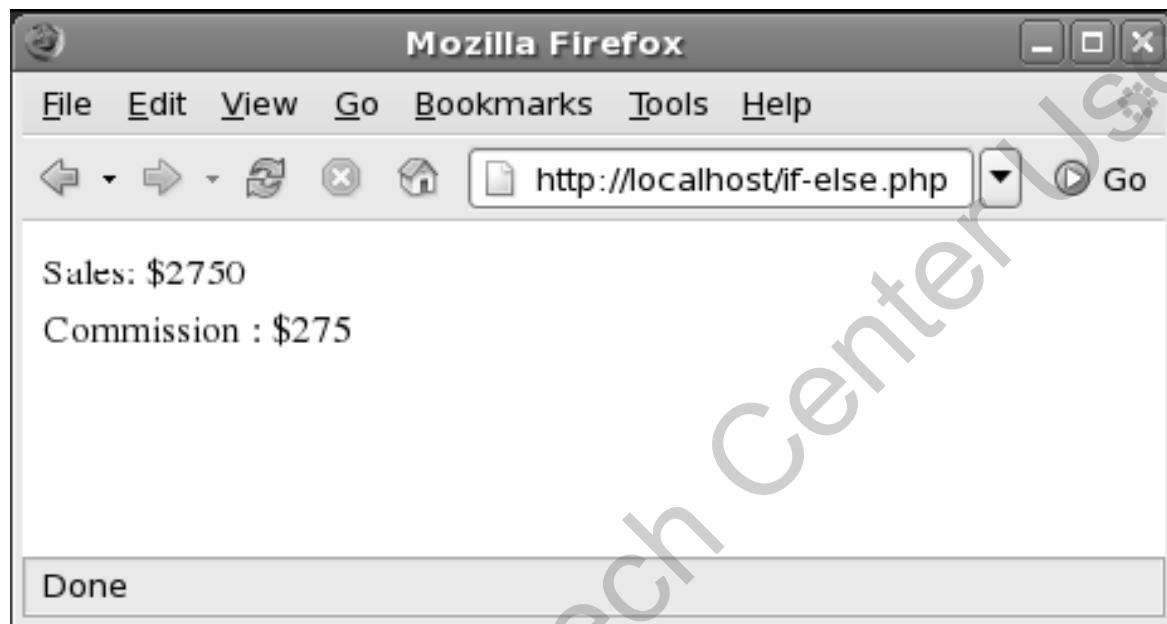
```
if(truth expression)
{
    Statements to be executed if the condition evaluates
    to true;
}
else
{
    Statements to be executed if the condition evaluates
    to false;
}
```

- ◆ Displaying a block of code with an if...else statement
 - ❖ Enter the following code:

Snippet

```
<?php
$sales = 21050;
if($sales > 2000)
{
    $comm = $sales * .1;
    echo "Sales: $$sales <br> Commission : $$comm";
}
else
{
    $comm = $sales * .05;
    echo "Sales: $$sales <br> Commission : $$comm";
}
?>
```

Displays the following output:



When the sales amount exceeds \$2000, the program executes the body of the `if` statement and calculates commission at the rate of 10%.

When the sales amount is less than \$2000, the program executes the body of the `else` statement.

- ◆ else if clause is:

- ◆ Used along with if statement
- ◆ An optional clause that allows testing alternative conditions

- ◆ Demonstrating the use of else if clause
 - ❖ **saleComm.html** - accepts the sales amount from the user

Snippet

```
<html>
<body>
<form action="SaleComm.php" method="GET">
<table>
<tr>
<td>Total Sales : </td>
<td><input type="text" name="sal"></td>
</tr>
</table>
<br>
<input type="submit" value="Submit">
</form>
</body>
</html>
```

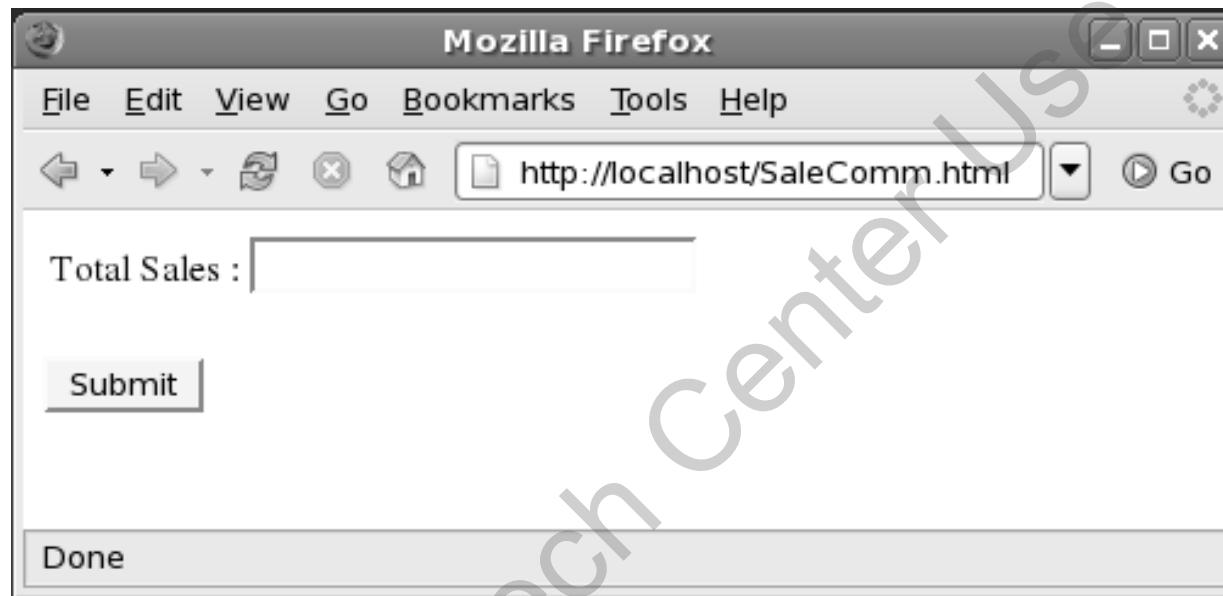
if...else Statement

- ❖ **saleComm.php** - processes the sales amount and calculates the commission

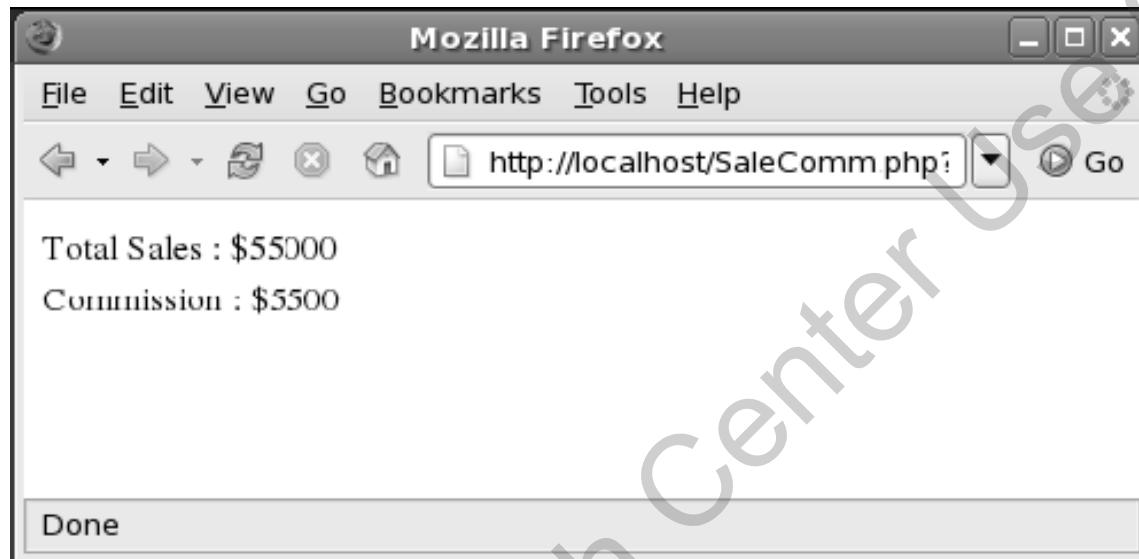
Snippet

```
<?php
$sal=$_GET['sal'];
echo "Total Sales : $";
echo $sal;
echo "<br>";
if ($sal > 50000){
    $comm = $sal * .10;
    echo "Commission : $$comm";
    echo "<br>"; }
else if ($sal > 20000 and $sal <= 50000) {
    $comm = $sal * .010;
    echo "Commission : $$comm";
    echo "<br>"; }
else if ($sal < 20000) {
    $comm = $sal * .05;
    echo "Commission: $$comm";
    echo "<br>"; }
?>
```

saleComm.html displays the following output:



saleComm.php displays the following output:



In the code, commission is calculated according to the sales amount the user enters.

- ◆ An `if` statement within an `if` statement or an `else` statement is known as nested `if` statement

For Aptech Center USE ONLY

- ◆ Calculating the electricity charges based on the units of electricity consumed
 - ❖ **elecBill.html** - accepts the number of units consumed

Snippet

```
<html>
<body>
<form action="elecBill.php" method="GET">
<table>
<tr>
<td>Electricity Units Consumed : </td>
<td><input type="text" name="units"></td>
</tr>
</table>
<br>
<input type="submit" value="Submit">
</form>
</body>
</html>
```

- Calculating the electricity charges based on the units of electricity consumed
 - elecBill.php - calculates total electricity bill

Snippet

```
<?php
$units=$_GET['units'];
echo "Number of Units Consumed : ";
echo $units;
echo "<br>";
if ($units > 1000)
{
    $rate = $units * 3;
    $service = $rate * .1;
    echo "Service Charge added for Units above 1000 : $$service";
    echo "<br>";
    $totalbill = $rate + $service;
    echo "Total Electricity Bill : $$totalbill";
}
```

```
else
{
    if ($units > 500 and $units <= 1000)
    {
        $rate = $units * 2;
        echo "Total Electricity Bill : $$rate";
    }
    else
    {
        $rate = $units * 1.5;
        echo "Total Electricity Bill : $$rate";
    }
}
?>
```

elecBill.html – displays the following output:

A screenshot of a Mozilla Firefox browser window. The title bar says "Mozilla Firefox". The menu bar includes "File", "Edit", "View", "Go", "Bookmarks", "Tools", and "Help". Below the menu is a toolbar with icons for back, forward, search, and refresh. The address bar shows the URL "http://localhost/elecBill.html". The main content area contains the following HTML code:

```
Electricity Units Consumed :   
  
  
  
Done
```

elecBill.php – displays the following output:

A screenshot of a Mozilla Firefox browser window. The title bar says "Mozilla Firefox". The menu bar includes "File", "Edit", "View", "Go", "Bookmarks", "Tools", and "Help". Below the menu is a toolbar with icons for back, forward, search, and refresh. The address bar shows the URL "http://localhost/elecBill.php". The main content area displays the following output:

```
Number of Units Consumed : 1750  
Service Charge added for Units above 1000 : $525  
Total Electricity Bill : $5775  
  
Done
```

In the code, if the user enters 1500 as input, PHP code in Code first calculates the rate and stores the value in the **\$rate** variable.

To calculate the service charge to be levied on the electricity bill, Code **elecBill.php** calculates the service charge at the rate of 10% and stores the value in the **\$service** variable.

It then stores the total amount in the **\$totalbill** variable.

switch Statement

- ◆ Used as an alternative to a lengthy `if . . . else` construct
- ◆ Consists of an expression that is compared to all possible case expressions listed in its body
- ◆ On finding a match, it executes the block of code ignoring any further case lines
- ◆ Uses a `break` statement to halt the execution of the switch statement and transfer the control to the code following `switch`

Syntax

```
switch(variable) {  
    case value1:  
        Code executes if condition equals value1  
        break;  
    case value2:  
        Code executes if condition equals value2  
        break;  
    .  
    .  
    .  
    default:  
        Code executes if the variable does not matches any  
        specified value  
}
```

Where,

- ◆ **case keyword** is followed by a case constant
- ◆ **default** is a special case executed when none of the case constants is matching

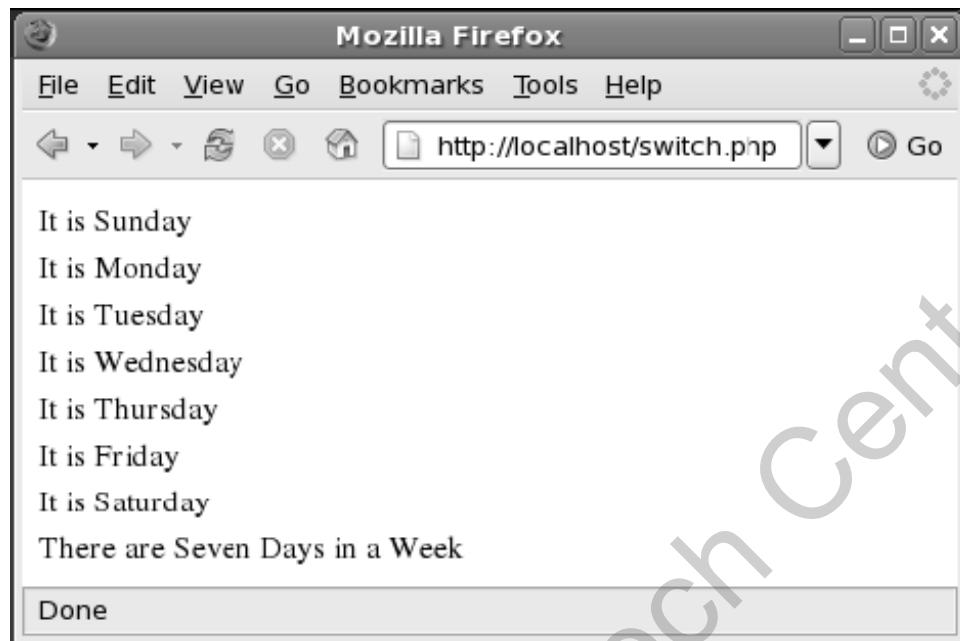
- ◆ Displaying a switch statement without any break statement
 - ◆ Enter the code in a script named **switch.php**

Snippet

```
<?php
$day = 1;
switch ($day)
{
    case 1:
        echo "It is Sunday";
        echo "<br>";
    case 2:
        echo "It is Monday";
        echo "<br>";
```

```
case 3:  
echo "It is Tuesday";  
echo "<br>";  
case 4:  
echo "It is Wednesday";  
echo "<br>";  
case 5:  
echo "It is Thursday";  
echo "<br>";  
case 6:  
echo "It is Friday";  
echo "<br>";  
case 10:  
echo "It is Saturday";  
echo "<br>";  
default:  
echo "There are Seven Days in a Week";  
echo "<br>";  
}  
?>
```

Displays the following output:



In the code, the weekday is 1, the program displays the message related to case 1.

Due to the absence of a break statement, it also displays the messages related to the subsequent cases until it reaches the end of the switch statement.

- ◆ Displaying a switch statement without any break statement
 - ❖ Enter the code in a script named **break.php**

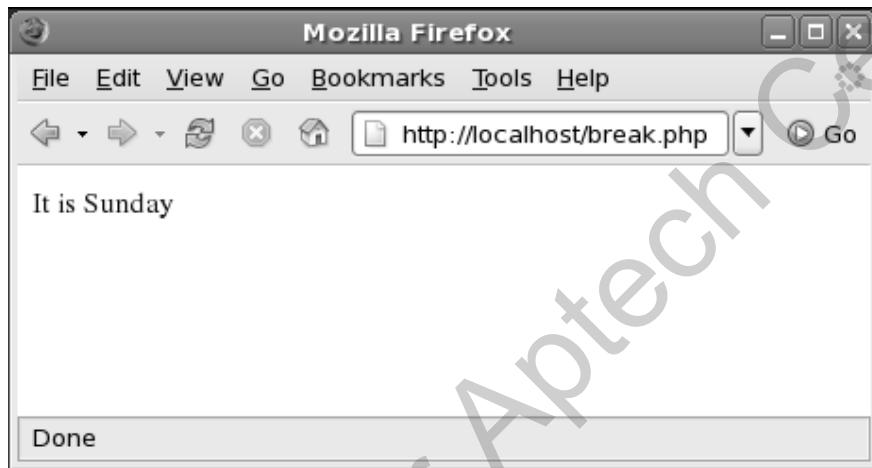
Snippet

```
<?php
$day = 1;
switch ($day)
{
    case 1:
        echo "It is Sunday";
        echo "<br>";
        break;
    case 2:
        echo "It is Monday";
        echo "<br>";
        Break;
```

```
case 3:  
echo "It is Tuesday";  
echo "<br>";  
break;  
  
case 4:  
echo "It is Wednesday";  
echo "<br>";  
break;  
  
case 5:  
echo "It is Thursday";  
echo "<br>";  
Break;  
  
case 6:  
echo "It is Friday";  
echo "<br>";  
break;  
  
case 10:  
echo "It is Saturday";  
echo "<br>";  
Break;
```

```
default:  
echo "There are Seven Days in a Week";  
echo "<br>";  
break;  
}  
?>
```

Displays the following output:



In the code, the weekday is 1, the program displays only the message related to case 1.

If the value assigned is any other value apart from numbers 1 to 10, the program displays 'There are Seven Days in a Week.'

- ◆ It is also known as a conditional operator
- ◆ It simplifies complex conditions into one-line statements
- ◆ It is considered as an alternative for the if...else statement

Syntax

```
truth_exp ? expr1 : expr2;
```

Where,

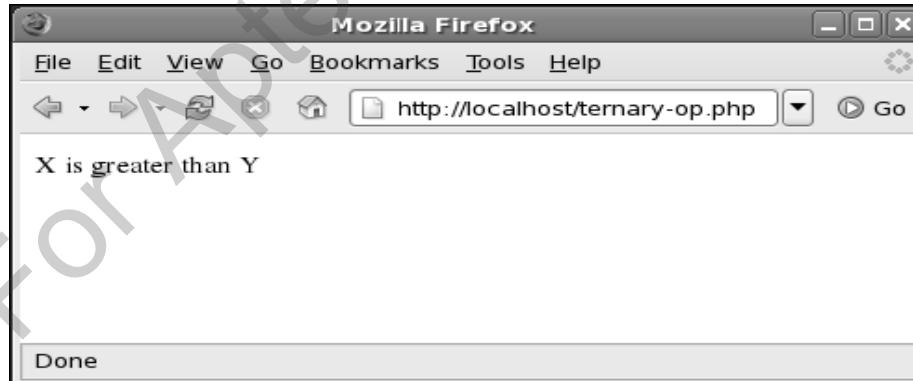
- ◆ `truth_expr` is evaluated and if it is true, `expr1` is evaluated
- ◆ If it is false, `expr2` is evaluated

- ◆ Displaying the output using the ternary operator
 - ❖ Enter the code as shown in code in the script named **ternary-op.php**

Snippet

```
<?php
$x = 100;
$y = 50;
$disp = ($x > $y) ? "X is greater than Y" : "Y is
greater than X";
echo $disp;
?>
```

Displays the following output:



Summary

- ◆ Conditional statements execute a set of statements only when a specified condition is satisfied and modify the order of flow in a program
- ◆ The if statement executes a block of code only when the specified condition is true
- ◆ In a nested if statement, you can include an if statement within another if statement or an else statement
- ◆ A switch...case statement checks a single variable against multiple values and executes a block of code based on the value it matches

- ◆ The break statement is used to transfer the control to the statements following the switch...case statement
- ◆ The default statement is used when none of the case statements matches the value of the switch variable
- ◆ Ternary operator is also known as conditional operator. It simplifies complex conditions into one-line statements

Flow Control in PHP

Session 12



Objectives

- ◆ *Explain the use of loops*
- ◆ *Explain the use of jump statements*

For Aptech Center Use Only

◆ Loops

- ◆ Perform repetitive tasks such as:
 - ◆ Retrieving information stored in databases
 - ◆ Sending mails to multiple users
 - ◆ Reading contents of an array

◆ Loops provided by PHP are as follows:

- ◆ While
- ◆ Do-while
- ◆ For

- ◆ Executes a block of code repetitively
- ◆ Tests the specified condition
 - ❖ If true, the statements present in the body of the loop are executed repetitively
 - ❖ If false, the loop ends, and the control is transferred to the statement following the loop
- ◆ The continuous execution of statements inside the loop is called iteration

while Loop

- ◆ Validity of the condition is checked before the loop is executed:
 - ❖ If condition is true, statements are executed in the loop body
 - ❖ If condition is false, the body of the loop is not executed

Syntax

```
while (condition)
{
    These statements are executed only if the condition is true;
}
```

These statements are executed irrespective of the condition;

Where,

- ❖ The condition is the test expression consisting of variables and operators

- ◆ Displaying the first five multiples of 5

Snippet

```
<?php  
$counter=1;  
$number=5;  
while($counter <= 5)  
{  
    $result=$number*$counter;  
    echo "<br>$result";  
    $counter=$counter+1;  
}  
?>
```

For Aptech Center Use Only

Displays the following output:



In the code, the result is displayed until the counter reaches 5.

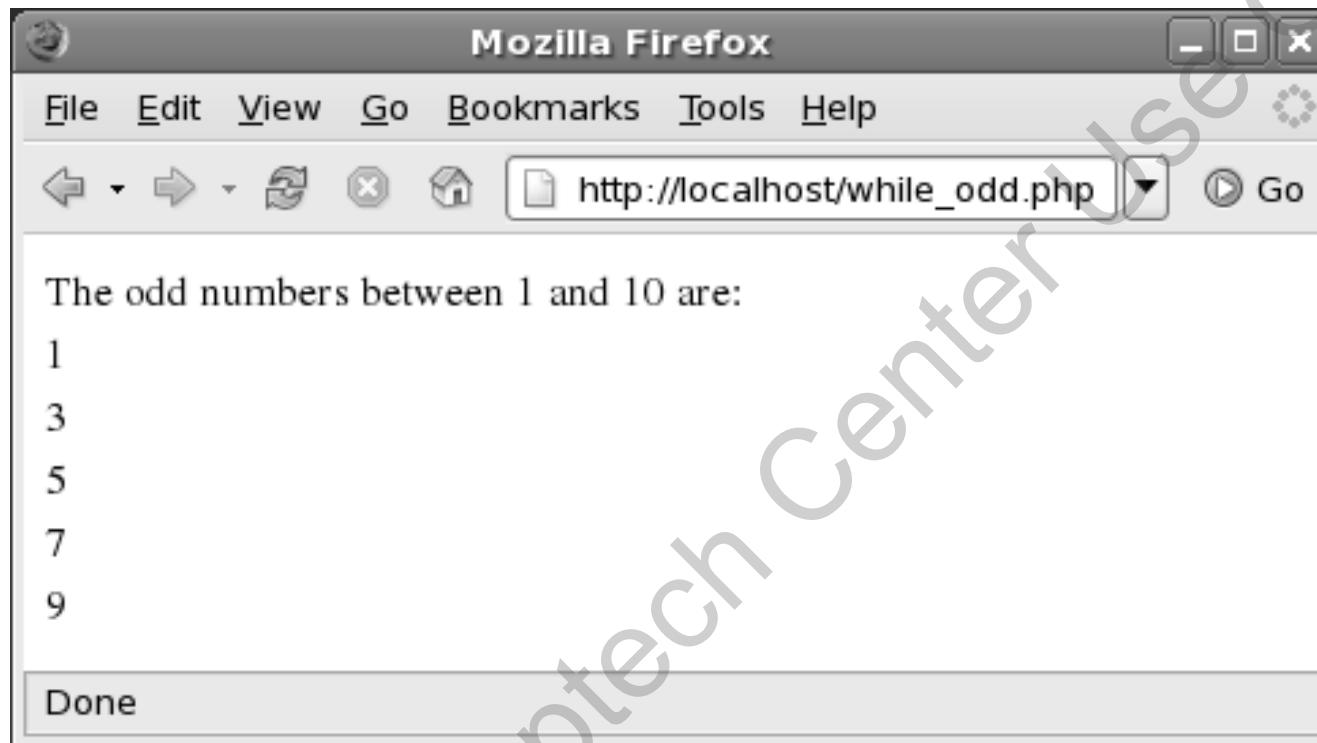
The loop stops once the counter exceeds 5.

- ◆ Displaying the odd numbers between 1 to 10

Snippet

```
<?php
$number=1;
echo "The odd numbers between 1 and 10 are:";
while($number <= 10)
{
echo "<br>$number";
$number=$number+2;
}
?>
```

Displays the following output:



In the code, the number is always incremented by 2 until the number reaches 10.

This is because **\$number** is initialized at 1 and every alternate number is odd.

do-while Loop

- ◆ Condition is checked at the end of the loop
- ◆ Executes the loop body at least once
- ◆ Works similar to the while loop

Syntax

```
do {  
    <These statements are executed if the condition is true;>  
    }while(condition)  
    <These statements are executed irrespective of the condition;>
```

Where,

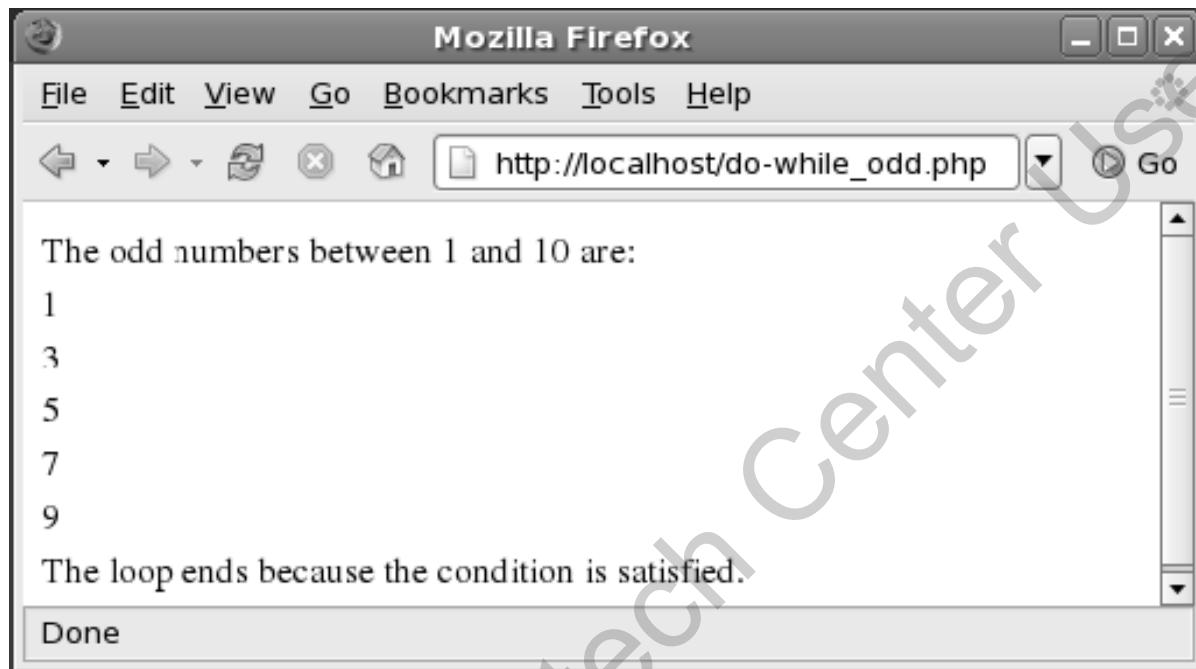
- ◆ The loop body is followed by the while keyword and the condition in parenthesis

- ◆ Displaying the odd numbers between 1 to 10 using do-while loop

Snippet

```
<?php
$number=1;
echo "The odd numbers between 1 and 10 are:";
do{
    echo "<br>$number";
    $number=$number+2;
}
while($number <= 10);
echo "<br>The loop ends because the condition is satisfied.";
?>
```

Displays the following output:



In the code, **\$number** is initialized at 1 and is incremented by 2, since the odd numbers are required to be displayed.

The execution of the loop continues until the counter reaches 10. The loop stops execution once the condition is satisfied.

- ◆ Executes block of code repetitively for a fixed number of times
- ◆ Statements in the loop body are executed as long as the condition is satisfied
- ◆ Stops the execution only when the condition is not satisfied

Syntax

```
for (expr1; expr2; expr3)
```

```
{
```

These statements are executed if the condition is true;

```
}
```

These statements are executed irrespective of the condition;

Where,

- ❖ **expr1** - is an initialization expression that initializes the value of the counter
- ❖ **expr2** – is a test expression that is evaluated for each loop iteration
- ❖ **expr3** – is a re-initialization expression that increases or decreases the value in the counter variable

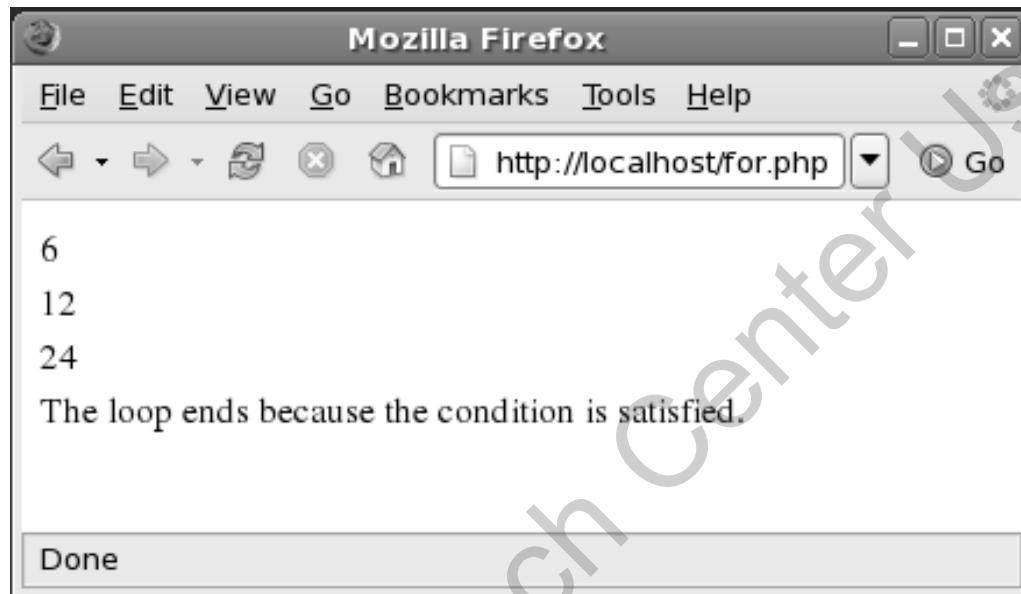
- ◆ Displaying the double of the given number using for loop

Snippet

```
<?php
$number=6;
for($counter=1; $counter <= 3; $counter++)
{
    echo "$number<br>";
    $number=$number*2;
}
echo "The loop ends because the condition is
satisfied.";
?>
```

for Loop

Displays the following output:



The variable, **\$number** is initialized with a value of 6.

When the loop starts, 6 is multiplied by 2 And the value is stored in the variable, **\$number** is 12.

The loop executes thrice since the terminating condition has been set when the counter value reaches 3.

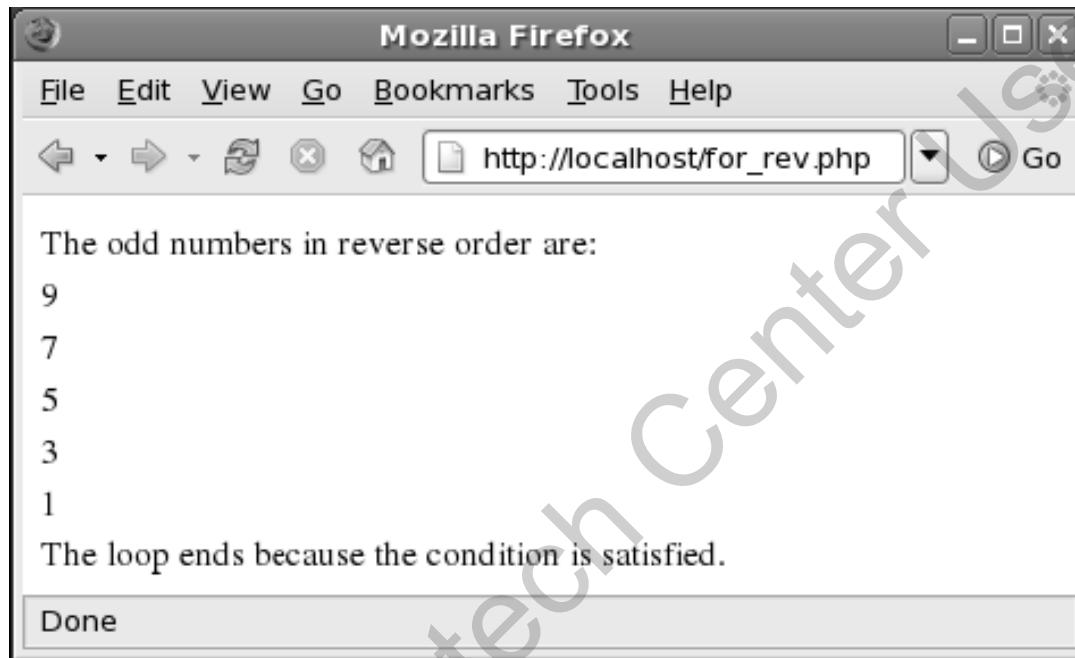
Once the counter value reaches 3, the loop stops executing.

- ◆ Displaying the first five odd numbers in the reverse order using for loop

Snippet

```
<?php
echo "The odd numbers in reverse order are:";
for($i=5;$i>=1;$i--)
{
    $number=$i * 2 - 1;
    echo "<br>$number";
}
echo "<br>The loop ends because the condition is
satisfied.";
?>
```

Displays the following output:



In the code, the for loop declares a counter variable, which is initialized at 5.

The re-initialization expression decrements the counter every time the for loop is executed.

- ◆ Control the execution of the loop and conditional statements
- ◆ PHP provides the following jump statements:
 - ◆ break
 - ◆ continue
 - ◆ exit

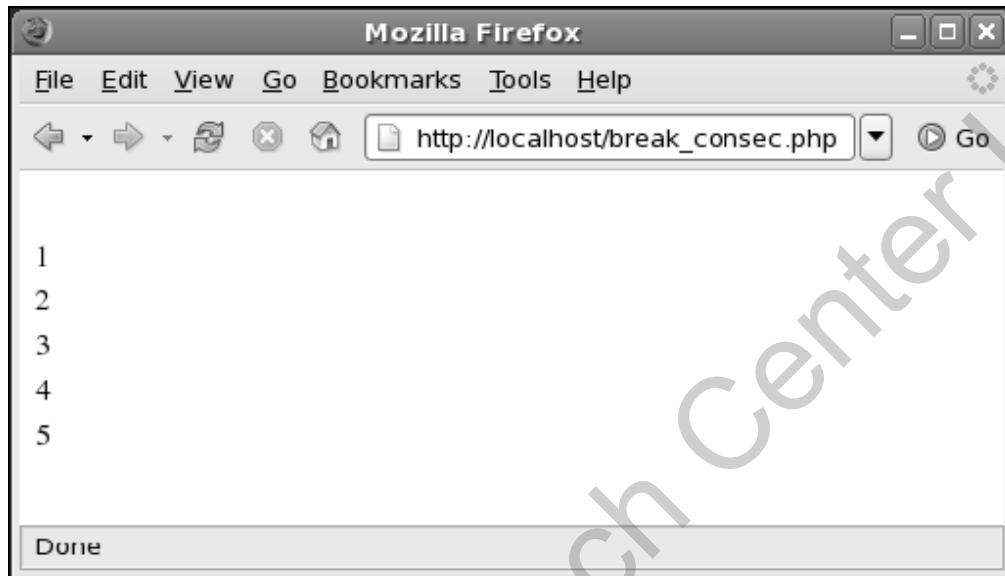
- ◆ Stops the execution of the loops and conditional statements
- ◆ The control is then transferred either to the beginning of the next loop or to the statement following the loop
- ◆ Can be used with the `if` statement, `switch` statement, `for` loop, `while` loop, and `do-while` loop

- ◆ Displaying consecutive numbers from 1 to 5 using break statement

Snippet

```
<?php
for($i=1;;$i++) {
if($i>5)
{
    break;
}
echo "<br>$i";
}
?>
```

Displays the following output:



The `break` statement is used within the `for` loop.

The `for` loop does not include any terminating condition.

The terminating condition is specified within the `if` statement using the `break` statement.

If the `break` statement is not used, it will become an infinite loop.

- ◆ Checking whether the alphabet is a vowel using switch statement

Snippet

```
<?php
$alphabet='u';
switch($alphabet) {
case 'a':
echo "<br>The alphabet is a vowel.";
break;
case 'A':
    echo "<br>The alphabet is a vowel.";
    break;
case 'e':
    echo "<br>The alphabet is a vowel.";
    break;
case 'E':
    echo "<br>The alphabet is a vowel.";
    break;
case 'i':
```

break Statement

```
        echo "<br>The alphabet is a vowel.";
        break;
case 'I':
        echo "<br>The alphabet is a vowel.";
        break;
case 'o':
        echo "<br>The alphabet is a vowel.";
        break;
case 'O':
        echo "<br>The alphabet is a vowel.";
        break;
case 'u':
        echo "<br>The alphabet is a vowel.";
        break;
case 'U':
        echo "<br>The alphabet is a vowel.";
        break;
default:
        echo    "<br>The    alphabet    is    not    a
vowel.";
} ?>
```

Displays the following output:



In the code, the `break` statement is used in the `switch` statement.

The `break` statement moves the control to the statements following the `switch` statement.

If the `break` statement is not used, PHP will execute all the statements including the statements present in the following `case` statement.

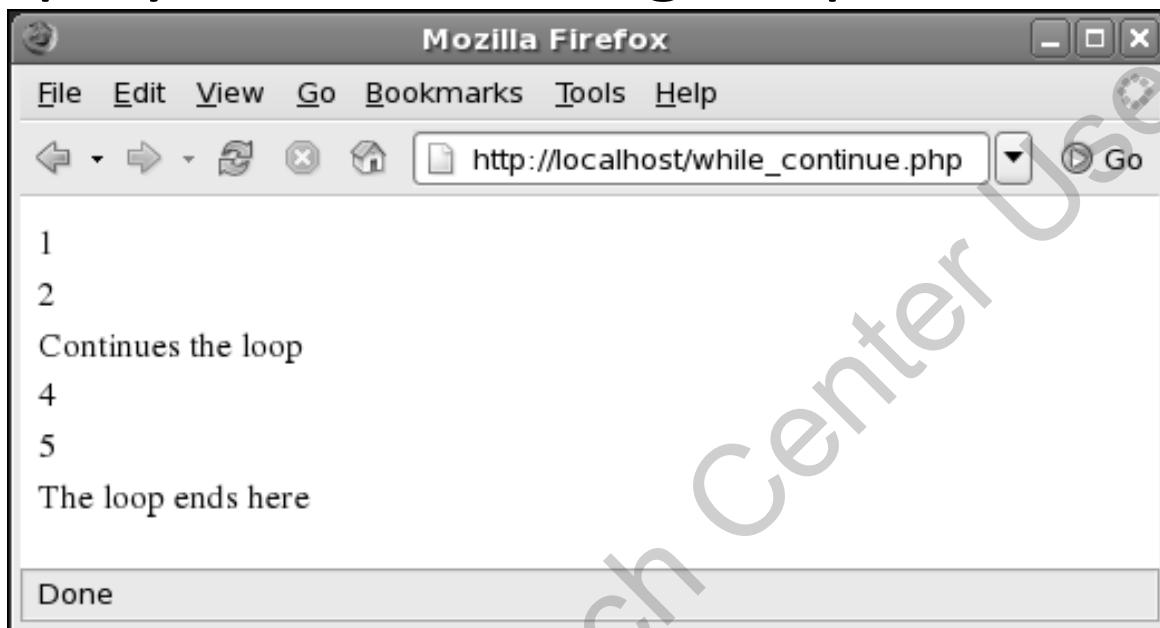
- ◆ Used within the loop statements
- ◆ Skips the code following the continue statement in the loop body and executes the next iteration of the loop
- ◆ Can be used with the if statement, for loop, while loop, and do-while loop

- ◆ Displaying the consecutive numbers from 1 to 5 using the while loop

Snippet

```
<?php
$counter = 0;
while ($counter<5)
{
    $counter++;
    if ($counter==3)
    {
        echo "Continues the loop<br>";
        continue;
    }
    echo "$counter<br>";
}
echo "The loop ends here";
?>
```

Displays the following output:



In the code, the `continue` statement is used in the `if` statement.

Here, the counter is initialized to 0. The loop continues until the counter reaches 3.

When the counter reaches the value of 3, the loop skips the `if` body and executes the next iteration of the loop.

The loop continues until the condition becomes `false`.

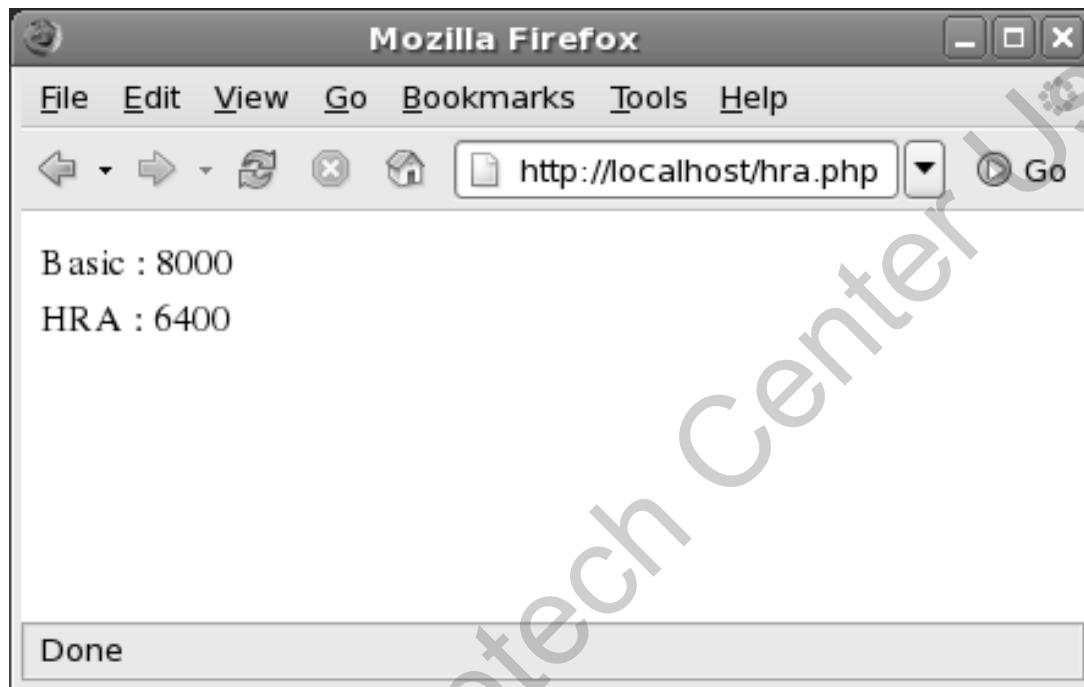
exit Statement

- ◆ Ends the loop and the control is transferred to the statement following the loop body
- ◆ Following code calculates the HRA using `exit` statement:

Snippet

```
<?php
$salary=8000;
if($salary<6000)
{
    echo "Basic : $salary<br>";
    echo "Salary below 6000 is not entitled for HRA.";
    exit;
}
else
{
    echo "Basic : $salary<br>";
    $hra=$salary * 0.8;
    echo "HRA : $hra";
}
?>
```

Displays the following output:



In the code, HRA is calculated based on the basic salary.

If the basic salary is less than 6000, the `if` statement exits.

If the basic salary is greater than or equal to 6000, HRA is calculated.

Summary

- ◆ A loop executes a block of code repetitively
- ◆ A while loop executes the statements in the loop body as long as the condition is true
- ◆ The do-while loop is similar to the while loop. In this loop structure the condition is placed at the end of the loop
- ◆ A for loop enables the execution of a block of code repetitively for a fixed number of times
- ◆ The jump statements control the execution of the loop statements



- ◆ The break statement stops the execution of the loop. The control is then passed either to the beginning of the next loop or to the statement following the loop
- ◆ The continue statement skips the code following the continue statement in the loop body and executes the next iteration of the loop
- ◆ The exit statement ends the loop and the control is passed to the statement following the loop body

Functions in PHP

Session 14



- ◆ *Explain functions in PHP*
- ◆ *Describe the built-in functions in PHP*
- ◆ *Explain the process of creating a user-defined function*
- ◆ *Explain the process of passing arguments to a function*
- ◆ *Explain the process of returning values from a function*
- ◆ *Explain the use of recursive functions*

- ◆ Functions
 - ❖ Are named section of a program
 - ❖ Are used to perform a specific task
 - ❖ Split program into modules
 - ❖ Are used to enable the developer to reuse the same piece of code
 - ❖ Can be easily modified in a program instead of going through entire code to make changes

Functions

- ◆ Statements are grouped into a single unit to perform a specific task
- ◆ Enhance the logical flow in a program by dividing complicated code sequences into smaller modules
- ◆ Enable to write a piece of code and assign a name to it
- ◆ Include parameters that are:
 - ◆ Variables
 - ◆ Specified within the parenthesis after the name of a function
 - ◆ Used to add more functionality
- ◆ Executed or invoked anywhere in the program using the assigned name

Built-in PHP Functions

- ◆ Provides different built-in functions to be included in the PHP script
- ◆ Built-in functions are grouped into following categories:
 - ❖ Mathematical functions
 - ❖ String functions
 - ❖ Date and time functions
 - ❖ Error handling functions
 - ❖ Database functions
 - ❖ Array functions
 - ❖ Mail functions

- ◆ Operate on numerical data

Table lists and describes some of the mathematical functions in PHP:

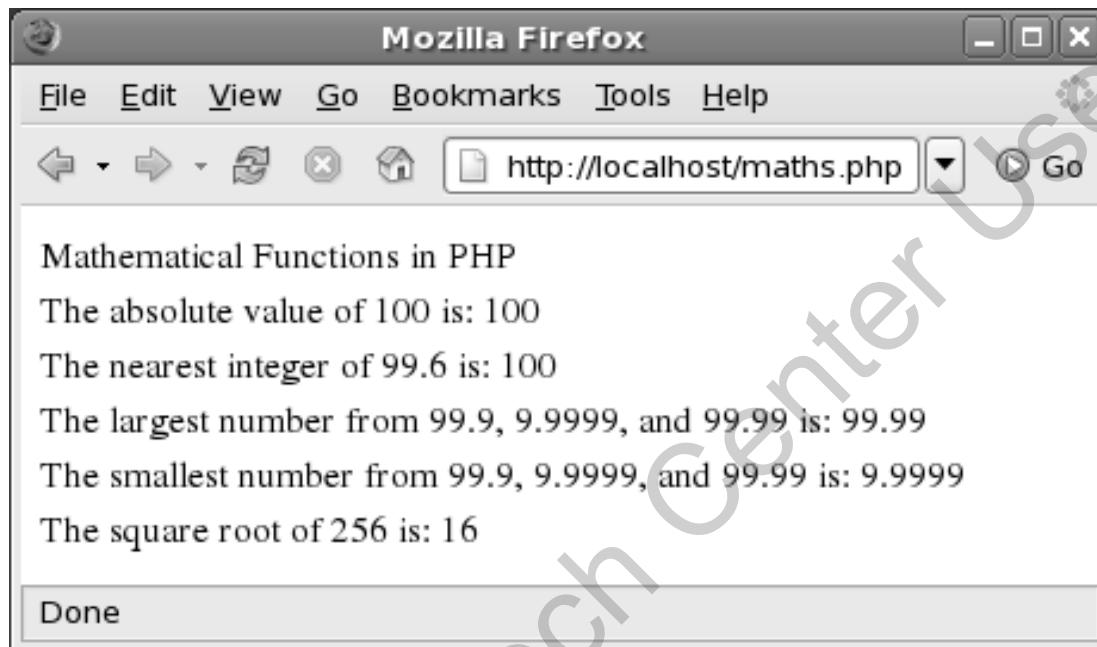
Function Name	Syntax	Description
abs	abs(arg)	Returns the absolute value of the argument
max	max(arg1,arg2,...)	Returns the largest value from the specified arguments. It also allows comparing multiple arrays
min	min(arg1,arg2,...)	Returns the smallest value from the specified arguments. It also allows comparing multiple arrays
sqrt	sqrt(arg)	Returns the square root of the argument
pow	pow(base, exp)	Returns the value of the base raised to the power of the exponential
round	round(number)	Returns the nearest integer of the specified number
rand()	rand(min, max)	Returns a random integer
ceil()	ceil(x)	Returns the value of a number rounded upwards to the nearest integer
floor()	floor(x)	Returns the value of a number rounded downwards to the nearest integer

- ◆ **maths.php** - Use of mathematical functions

Snippet

```
<?php
echo "Mathematical Functions in PHP";
echo "<br>";
echo "The absolute value of 100 is: ";
echo abs(100);
echo "<br>";
echo "The nearest integer of 99.6 is: ";
echo round(99.6);
echo "<br>";
echo "The largest number from 99.9, 9.9999, and
99.99 is: ";
echo min(99.9,9.9999,99.99);
echo "<br>";
echo "The square root of 256 is: ";
echo sqrt(256);
echo "<br>";
?>
```

Displays the following output:



String Functions

- ◆ Operate on character type of data
- ◆ Table lists some of the string functions in PHP:

Function Name	Syntax	Description
chr	chr(ascii)	Returns the character equivalent to the specified ASCII code
bin2hex	bin2hex(string)	Converts a string of ASCII characters to hexadecimal values
strtolower	strtolower(string)	Converts the specified string to lower case
strlen	strlen(string)	Returns the length of the string specified as an argument
strcmp	strcmp(string1,string2)	Compares two strings. Returns zero if string1 is equal to string2. It returns less than zero, if string1 is less than string2. Otherwise, it returns greater than zero when string1 is greater than string2
strtoupper	strtoupper(string)	Converts the specified string to upper case
strrev	strrev(string)	Returns the reverse of the string
stristr()	stristr(string,search)	Finds the first occurrence of a string inside another string (case-insensitive)
strrchr()	strrchr(string,char)	Finds the last occurrence of a string inside another string

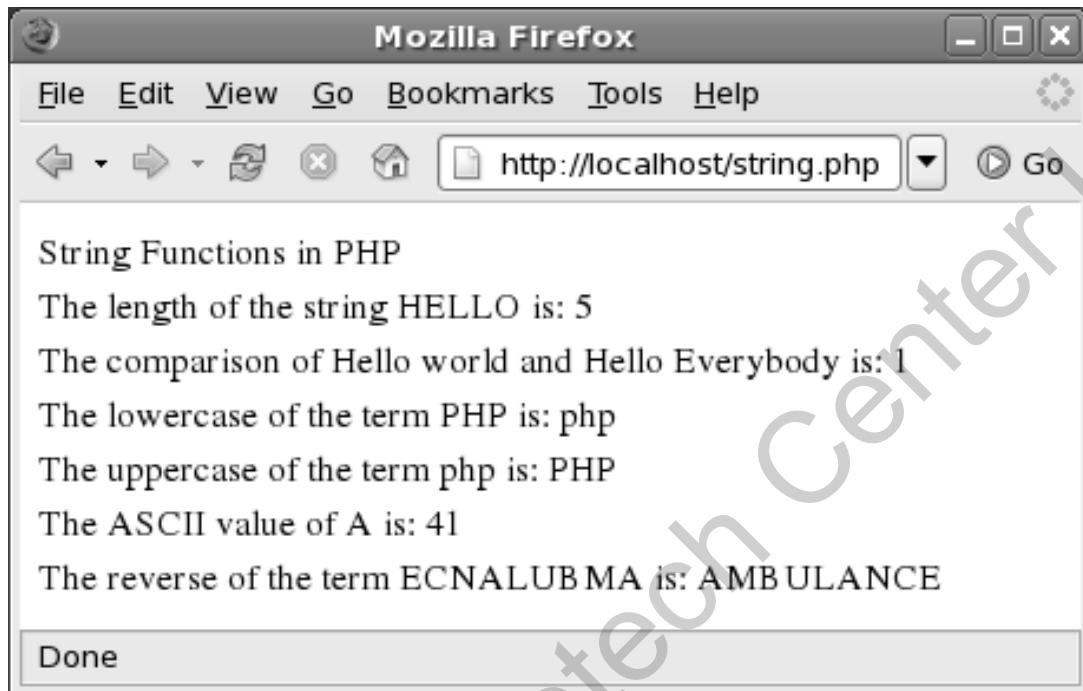
Function Name	Syntax	Description
strrpos()	strrpos(string,find,start)	Finds the position of the last occurrence of a string inside another string (case-sensitive)
strcmp()	strcmp(string1,string2,length)	String comparison of the first n characters

- ◆ **string.php** - Use of string functions

Snippet

```
echo strtoupper("php");  
  
echo "<br>";  
  
echo "The ASCII value of A is: ";  
  
echo bin2hex("A");  
  
echo "<br>";  
  
echo "The reverse of the term ECNALUBMA is: ";  
  
echo strrev("ECNALUBMA");  
  
?>
```

Displays the following output:



The screenshot shows a Mozilla Firefox browser window with the title "Mozilla Firefox". The address bar displays "http://localhost/string.php". The main content area of the browser shows the output of a PHP script. The output consists of several lines of text, each demonstrating a different string function:

```
String Functions in PHP
The length of the string HELLO is: 5
The comparison of Hello world and Hello Everybody is: 1
The lowercase of the term PHP is: php
The uppercase of the term php is: PHP
The ASCII value of A is: 41
The reverse of the term ECNALUB MA is: AMBULANCE
```

A "Done" button is visible at the bottom left of the browser window.

- ◆ Enables to calculate the date and time on the system
- ◆ Table lists and describes some of the date and time functions :

Function Name	Syntax	Description
checkdate	checkdate(month,day,year)	Returns the value as 1 if the specified date is valid. A valid date contains: <ul style="list-style-type: none">◆ Month between 1 and 12◆ Day within the range of days for the specified month◆ Year between 1 and 32767
getdate	getdate(timestamp)	Returns an array containing date and time information. The information is returned for a Unix timestamp
time	time()	Returns the current time measured in the number of seconds

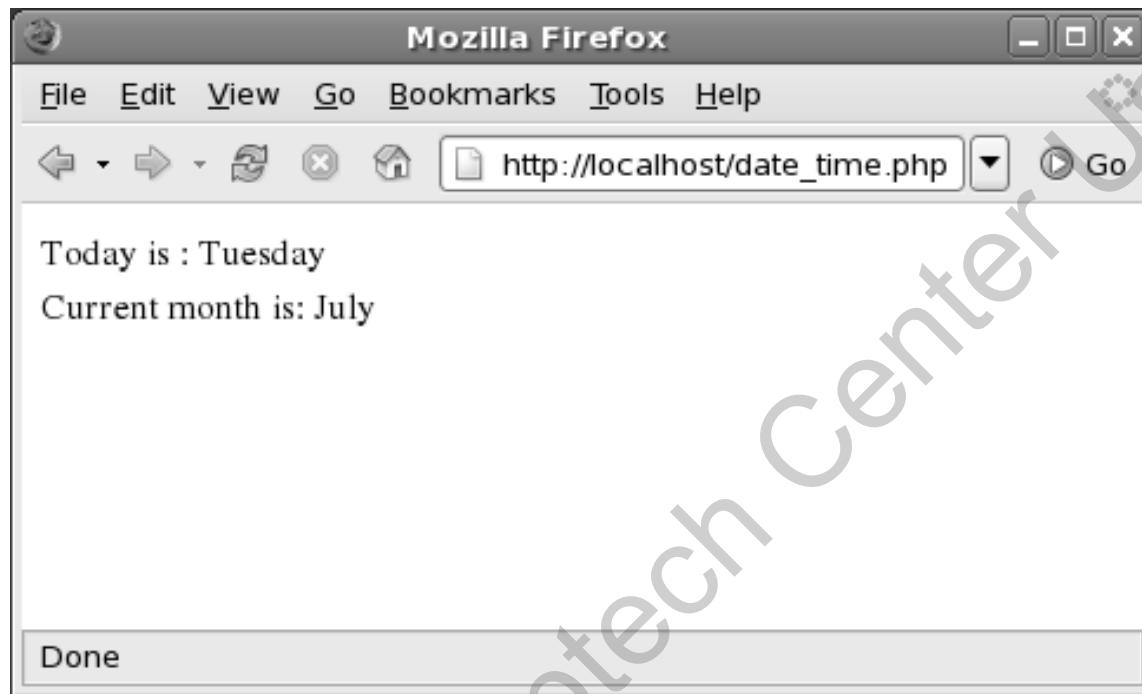
Function Name	Syntax	Description
Date	date(format, timestamp)	<p>Returns a string depending on the timestamp or the current local time, if the timestamp is not specified</p> <p>Some of the formats that can be used are as follows:</p> <ul style="list-style-type: none">d – day of the monthD – textual representation of the dayj – day of the month without leading zeros (1 to 31)m – numeric representation of the monthM – textual representation of the monthy – a two digit representation of the yearY – a four digit representation of the yeara – Lowercase am or pmh – 12 hour format of an hourH – 24 hour format of an houri – minutes with leading zeros – seconds with leading zeros

- ◆ **date_time.php** - Use of common date and time functions

Snippet

```
<?php  
  
date_default_timezone_set('Asia/Calcutta');  
  
echo "Today is : " .date("l");  
  
$Today_Date=getdate();  
  
$current_month=$Today_Date['month'];  
  
echo "<br>";  
  
echo "Current month is: ";  
  
echo $current_month;  
  
?>
```

Displays the following output:



The "1" is the argument string, which corresponds to the textual representation of the day of the week and will return the day.

- ◆ Defines the error handling rules and modify the way the errors are handled
- ◆ Table lists the error handling functions:

Function Name	General Form	Description
trigger_error	trigger_error(error _msg [,error_type])	Generates an error message, that is, it defines an error message at a specified condition as specified by the user User-defined function such as set_error_handler() inbuilt error handler can be used with it
set_error_handler	set_error_handler(e rror_ handler)	Handles errors during runtime by using a user-defined function
error_reporting	error_reporting(Constant)	Specifies which PHP errors are reported. PHP provides many levels of errors. You can use this function to set a level during the run-time of the script
strtotime	strtotime(string time [,now])	Parses an English textual date or time into a Unix timestamp (the number of seconds since January 1 1970 00:00:00 GMT)

Table lists some of the error constants:

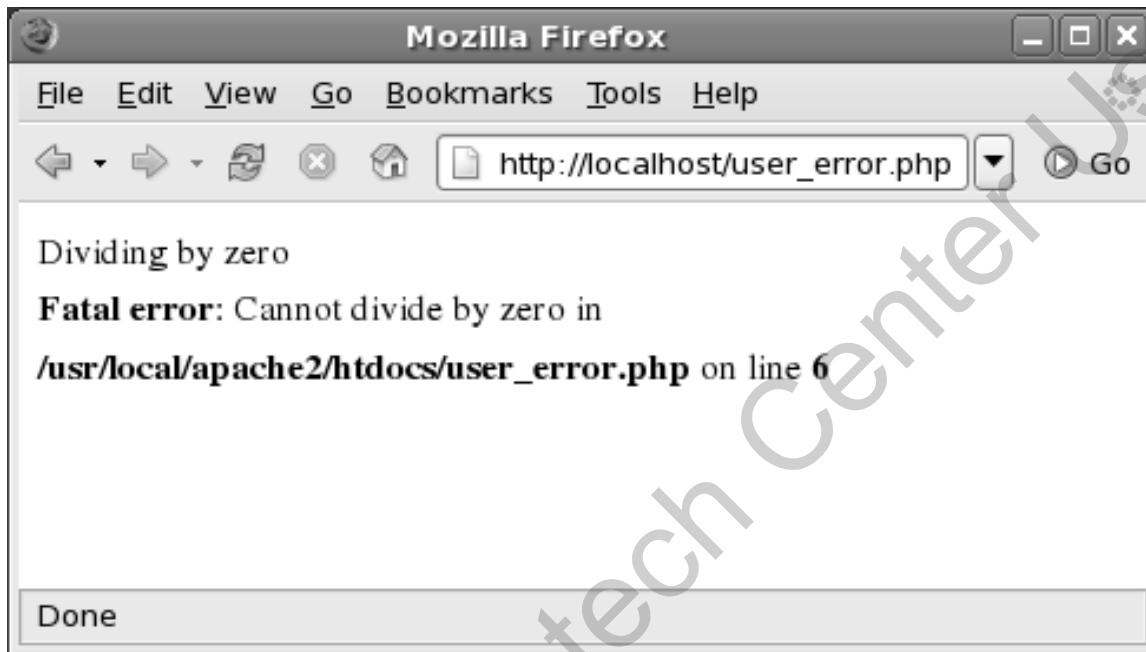
Constant Name	Value	Description
E_ERROR	1	Displays errors which are not recoverable
E_WARNING	2	Displays non-fatal run-time errors, however, this does not halt the execution of the script
E_PARSE	4	Displays the errors generated by the parser during the compilation
E_NOTICE	8	Displays run time error notices
E_COMPILE_ERROR	64	Displays compile time errors
E_USER_ERROR	256	Displays user generated error message
E_USER_WARNING	512	Displays user generated warning message
E_CORE_ERROR	16	Displays Fatal errors during PHP start up
E_CORE_WARNING	32	Displays Non-fatal errors during PHP start up
E_COMPILE_WARNING	128	Displays errors generated by the Zend Scripting Engine
E_ALL	8191	Displays all errors and warnings that are supported

- ◆ **user_error.php** - Use of error handling functions

Snippet

```
<?php  
  
$num1=0;  
  
if ($num1==0)  
{  
  
    echo "Dividing by zero";  
  
    trigger_error("Cannot divide by zero", E_USER_ERROR);  
  
}  
  
else  
{  
  
    $B=100/$num1;  
  
}  
  
?>
```

Displays the following output:



The value of \$num1 variable is tested.

- ◆ Function can be defined or created
- ◆ Function definition contains the code to be executed
- ◆ Defining a function is as follows:

Syntax

```
function function_name (arg1, arg2, arg3, ...)  
{  
    statement list  
}
```

User-defined Functions

- ◆ The `return expr` statement within the body of the function is used to return a value from a function
- ◆ Snippet accepts an argument, `$x`, and returns its square

Snippet

```
function square ($x)
{
    return $x*$x;
}
```

- ◆ To execute the function, invoke the function as follows:

Syntax

```
fun_name();
```

where,

fun_name – specifies the name of the function

- ◆ A PHP code can be included inside a function, other functions, and class definitions
- ◆ Rules to define function names are similar to the rules for defining labels in PHP
- ◆ Functions are not required to be defined before referencing
- ◆ When defining a function in a conditional manner, the script must first process the function definition before invoking the function

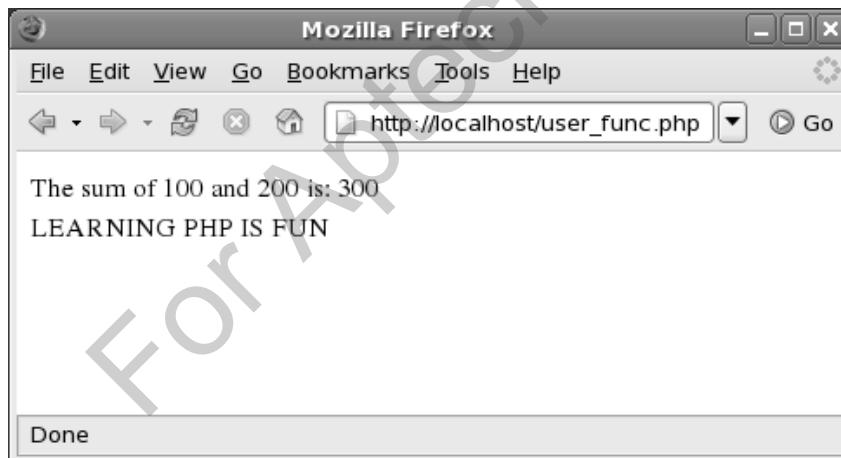
- ◆ **user_func.php** - Use of user defined functions

Snippet

```
<?php  
//A function to calculate the sum of two variables  
function addition()  
{  
    $A=100;  
    $B=200;  
    $C=$A+$B;  
    echo "The sum of 100 and 200 is: $C";  
}  
addition();
```

```
echo "<br>";  
  
// A function to display the text  
  
function Display()  
  
{  
  
    echo "LEARNING PHP IS FUN";  
  
}  
  
Display();  
  
?>
```

Displays the following output:



- ◆ PHP supports passing of arguments to a function
- ◆ The three different ways of passing arguments to a function are as follows:
 - ◆ Passing arguments by value
 - ◆ Passing arguments by reference
 - ◆ Setting default values for arguments
- ◆ The function definition determines the method of passing arguments to the function

- ◆ A function with arguments is as follows:

Syntax

```
function fun_name(arg1,arg2,...)  
{  
Code to be executed  
}
```

Where,

arg - specifies the argument that is passed to the function

- ◆ When an argument is passed by value it must:
 - ◆ Prefix with the dollar (\$) sign
 - ◆ Be any valid expression which are evaluated and assigned to the corresponding variable

For Aptech Center Use Only

- ◆ **arg_value.php** - Passing arguments by value

Snippet

```
<?php

//Creating a function to calculate the square of a
number

function Square($A)

{
    //The argument is passed by Value in the
    //function definition

    //using the $ sign.

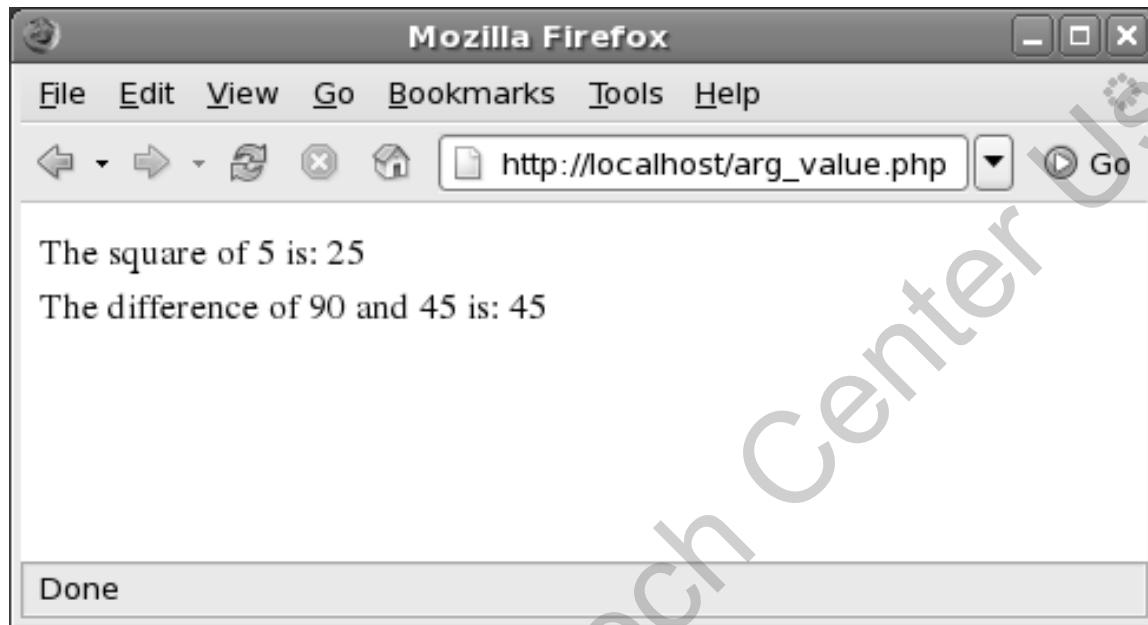
    //Calculating the square of the number

    $A=$A*$A;

    //Displaying the square of the number
```

```
echo $A;  
}  
  
//Assigning a value to the variable  
$A=5;  
  
//Displaying the text  
echo "The square of $A is: ";  
  
//Calling the function  
Square($A);  
  
// Creating a function to subtract one variable from another  
function subtraction($A,$B)  
{  
    //Calculating the difference  
    $C=$A-$B;  
    //Displaying the text  
    echo "<br>The difference of $A and $B is: $C";  
}  
  
//Calling the function and assigning values to the argument  
subtraction(90,45);  
?>
```

Displays the following output:



The subtraction () function subtracts the variable **\$B** from **\$A** and stores the resultant value in **\$C**.

- ◆ When an argument is passed by reference it must:
 - ❖ Be a variable
 - ❖ Prefix the arguments with the ampersand (&) sign to indicate that the value is passed by reference

- ◆ **arg_ref.php** - Passing values to the function by reference

Snippet

```
<php

//Defining a function and passing value to the
//arguments by reference

function Square(&$A)

{
    //Calculating the square of the number and
    //storing it in a variable

    $A=$A*$A;

    //Displaying the result

    echo $A;
}

//Assigning value outside the function

$A=5;

//Displaying text
```

- ◆ **arg_ref.php** - Passing values to the function by reference

Snippet

```
echo "The square of $A is: ";

//Executing the function by passing value to argument
by reference

Square($A);

//Defining a function and passing value to the
arguments by reference

function multiplication(&$A, &$B)

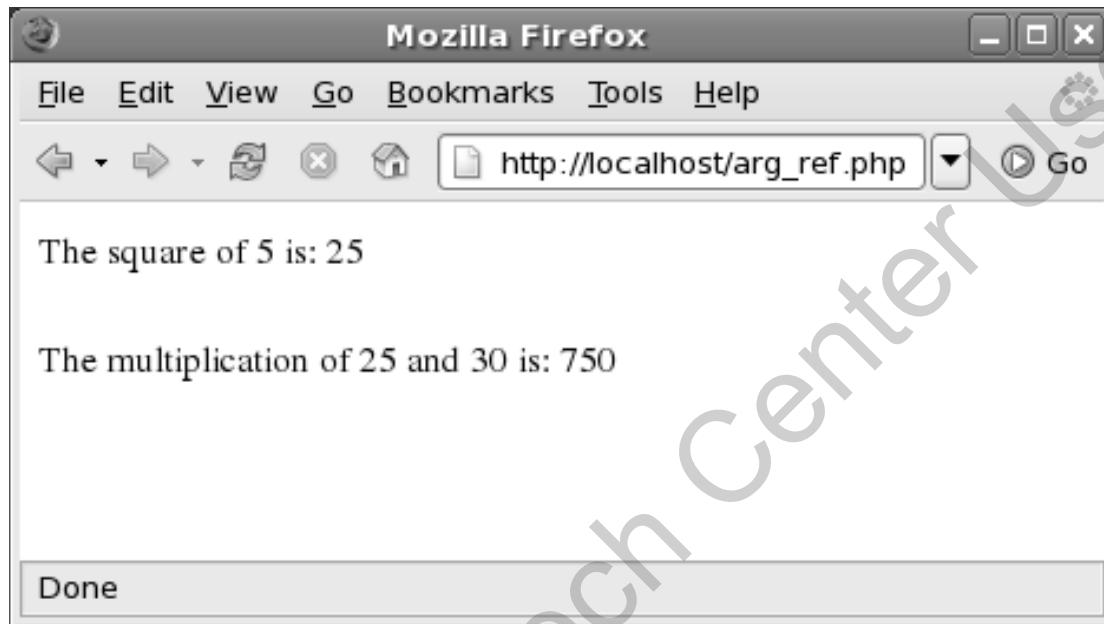
{
    //Calculating the multiplication of two numbers
    storing it in a variable
```

- ◆ **arg_ref.php** - Passing values to the function by reference

Snippet

```
$C=$A*$B;  
  
    //Displaying text  
  
    echo "<br><br>The multiplication of  
    $A and $B is: $C";  
  
}  
  
//Assigning value outside the function  
  
$A=25;  
$B=30;  
  
//Executing the function by passing  
value to argument by reference  
  
multiplication($A,$B);  
  
?>
```

Displays the following output:



The `multiplication()` function calculates the product of two variable `$A` and `$B` and stores the value in `$C`.

- ◆ PHP enables to assign default values to an argument in a function
- ◆ The default values enable the developer to initialize the function parameters when the function is invoked without any value being passed
- ◆ The default value assigned can be any one of the following:
 - ◆ Constant
 - ◆ Scalar
 - ◆ Array with scalar values or constant

- ◆ The use of default parameters

Snippet

```
function increment(&$num, $increment = 1)

{
    $num += $increment;
}

$num = 4;

increment($num);

increment($num, 3);
```

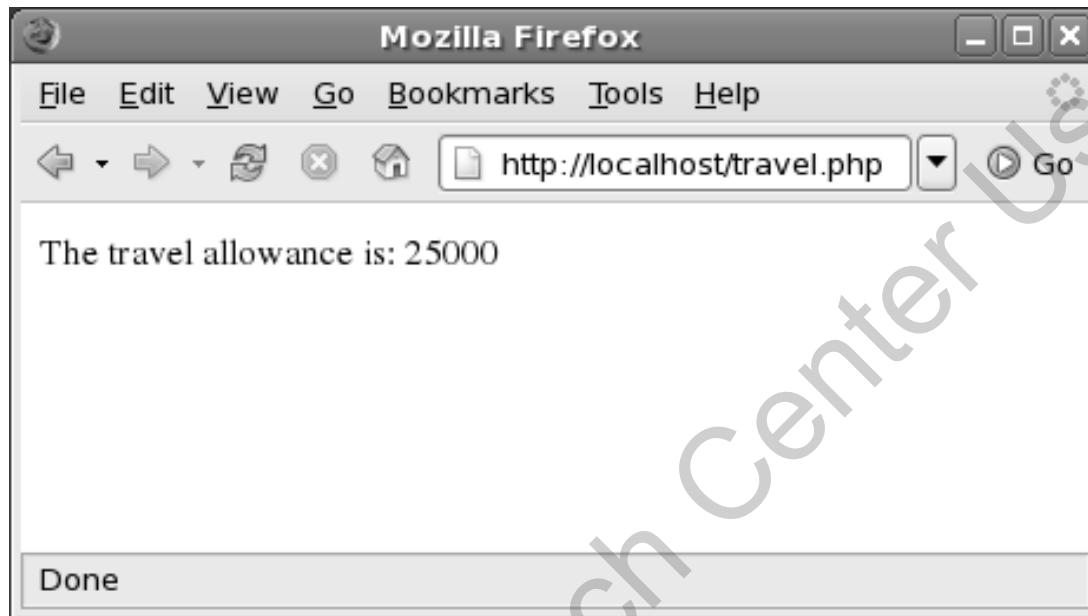
Setting Default Values

- ◆ **travel.php** - Assigning default values for an argument

Snippet

```
<?php  
  
//Creating a function and assigning default value to  
the argument  
  
function T_ALLOWANCE ($BASIC_SAL=100000)  
{  
  
//Calculating the travel allowance and storing it in  
a variable  
  
$T_ALLOWANCE=0.25*$BASIC_SAL;  
  
//Displaying text  
  
echo "The travel allowance is: $T_ALLOWANCE";  
  
}  
  
//Executing the function  
  
T_ALLOWANCE();  
  
?>
```

Displays the following output:



The T_ALLOWANCE () function calculates
and displays the traveling allowance.

- ◆ The `return` statement in a function returns the value from the function
- ◆ The value returned can be an array or an object
- ◆ The `return` keyword causes the function to stop execution and pass the control to the line from which it was invoked
- ◆ The reference operator is required to be used while declaring a function as well as when assigning the returned value to the variable

- ◆ **hra_func.php** - The use of return keyword to calculate the house rent allowance

Snippet

```
<?php

//Creating a function

function HRA($Basic_Sal)

{

//Calculating the HRA and storing it in a variable

$HRA=0.25*$Basic_Sal;

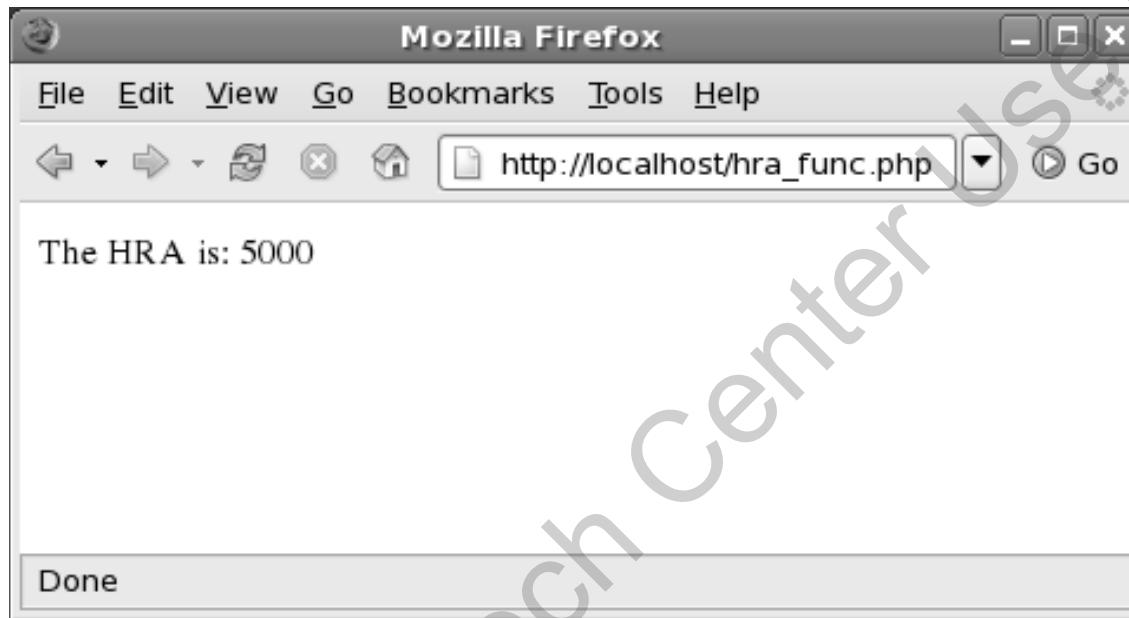
//Returning the value stored in the variable

return $HRA;

}
```

```
//Storing the output of the function in a variable after setting a  
//default value  
  
$B=HRA(20000);  
  
//Displaying the text  
  
echo "The HRA is: ";  
  
//Displaying the output  
  
echo $B;  
  
?>
```

Displays the following output



The HRA () function returns the house rent allowance, calculated at the rate of 25% of basic salary.



Nesting of Functions

- ◆ One function can be dependent on another function in the program
- ◆ The execution of one function inside another function is called nesting of functions

For Aptech Center Use Only

- ◆ **nested.php** - Use of nested functions

Snippet

```
<?php  
//Assigning value to variable  
$Basic_Sal=75000;  
//Creating a function and passing value by reference  
function HRA($Basic_Sal)  
{  
//Calculating the HRA  
$HRA=3/10 * $Basic_Sal;  
//Displaying Text  
echo "Your HRA is: ";  
//Displaying the computed HRA  
echo $HRA;  
echo "<br>";  
}  
//Creating a function and passing value by reference  
function TA($Basic_Sal)  
{
```

Nesting of Functions

3-6

```
//Calculating the TA  
  
$STA=1/4*$Basic_Sal;  
  
//Displaying Text  
  
echo "Your TA is: ";  
  
//Displaying the computed TA  
  
echo $STA;  
  
echo "<br>";  
  
}  
  
//Creating a function and passing value by reference  
  
function TAX($Basic_Sal)  
  
{  
  
//Calculating the tax  
$TAX=1/10*$Basic_Sal;  
  
//Displaying Text  
  
echo "Your TAX is: ";
```

```
//Displaying the computed tax  
echo $TAX;  
echo "<br>";  
}  
  
//Creating a function and passing value by reference  
  
function Net_Salary($Basic_Sal)  
{  
    //Storing tax, HRA and TA in variables  
    $A=3/10 * $Basic_Sal;  
    $B=1/4*$Basic_Sal;  
    $C=1/10*$Basic_Sal;  
    //Calculating the Net Salary
```

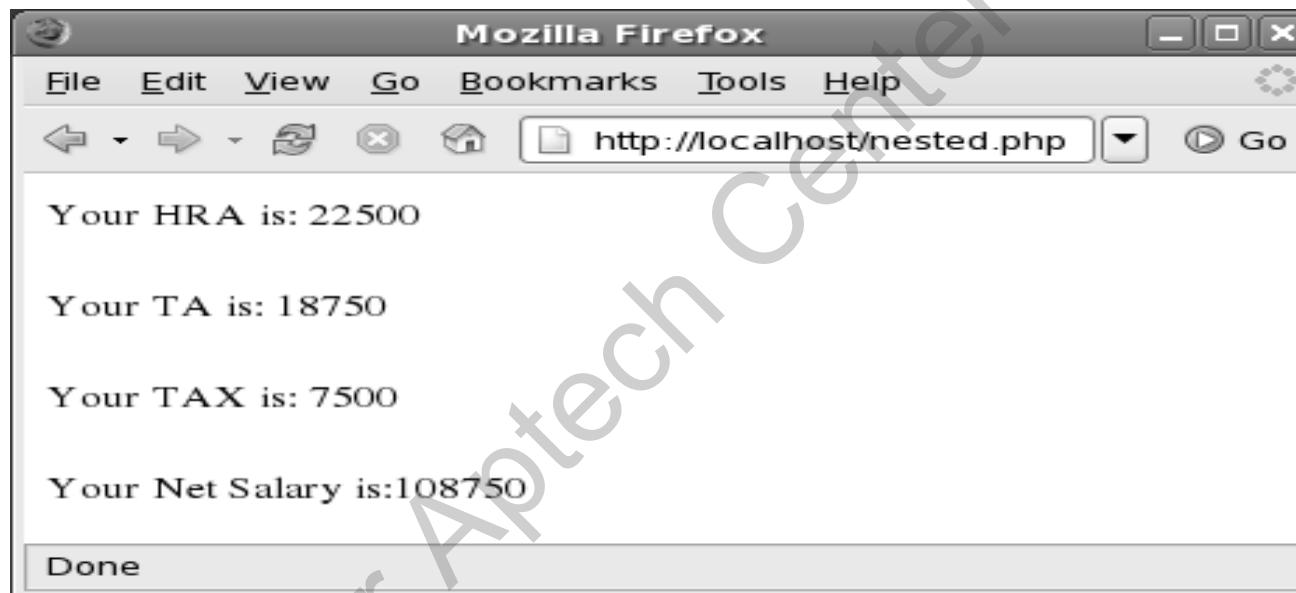
Nesting of Functions

5-6

```
$Net_Sal=75000+$A+$B-$C;  
  
//Displaying Text  
  
echo "Your Net Salary is:";  
  
//Displaying the Net Salary  
  
echo $Net_Sal;  
  
}  
  
//Calling the functions  
  
HRA($Basic_Sal);  
  
echo "<br>";  
  
TA($Basic_Sal);  
  
echo "<br>";  
  
TAX($Basic_Sal);  
  
echo "<br>";
```

```
Net_Salary($Basic_Sal);  
  
echo "<br>";  
  
?>
```

Displays the following output:



The Net_Salary() function calls three functions, HRA(), TA(), and TAX().

- ◆ The execution of a function within another function is called nested functions
- ◆ When a function executes itself repeatedly it is known as recursion
- ◆ When a function calls itself, the same block of code is executed

For Aptech Center Use Only

- ◆ **recursion.php** - Use of recursive functions in programs to calculate the factorial

Snippet

```
<?php

//Assigning Value to a variable

$A=10;

//Creating a function to calculate the factorial

function factorial($A)

{

    //Calculating the factorial

    if ($A<=1)

    {

        return 1;

    }

    else

    {

        return $A * factorial($A-1);

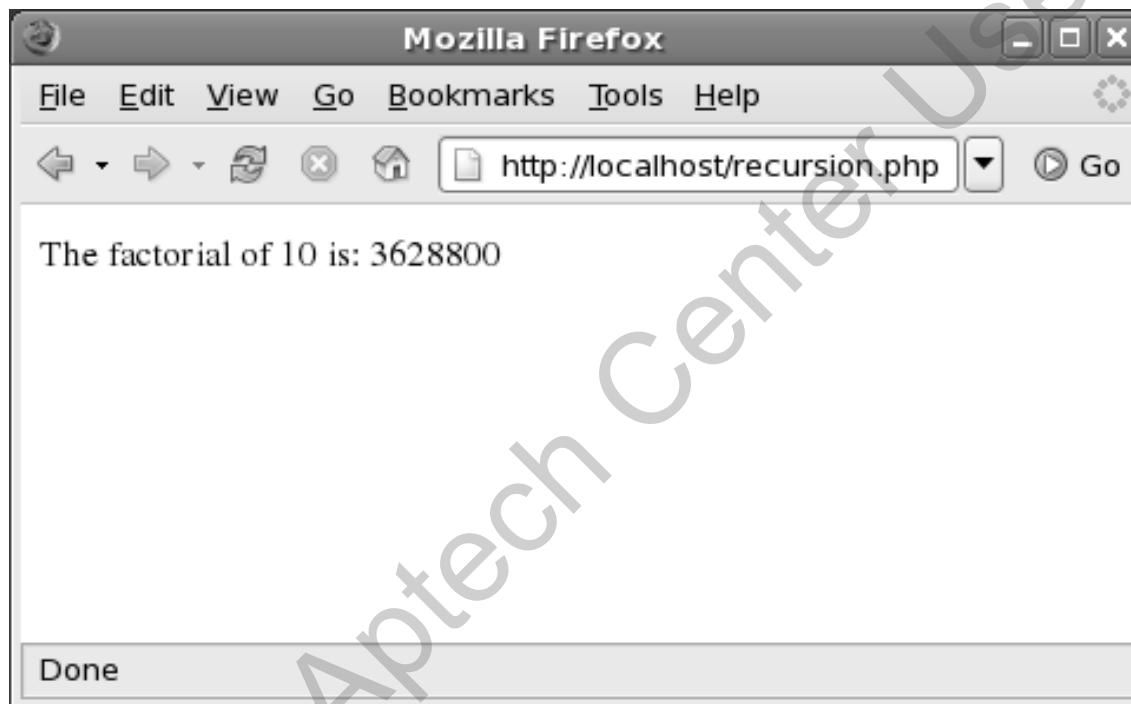
    }

}

echo factorial(5);
```

```
return 1;  
}  
else  
{  
    return $A * factorial($A-1);  
}  
}  
  
//Displaying Text  
echo "The factorial of $A is: ";  
//Assigning the result to a variable  
$B = factorial($A);  
//Displaying the result  
echo $B;  
?>
```

Displays the following output:



The `factorial()` function returns 1 if the input is 0 or 1.

Summary

- ◆ Functions can be used to implement execution of repetitive lines of code
- ◆ The built-in functions in PHP are mathematical, string, date and time, error handling, database, array and mail functions
- ◆ Mathematical functions operate on numerical data
- ◆ String functions operate on character type of data
- ◆ Date and time functions are used to display the system date and time
- ◆ Error handling functions are used to define the error handling rules
- ◆ Arguments can be passed to a function by value and reference. Default values can also be passed to a function
- ◆ Recursive functions have the ability to call themselves repeatedly

Working with Arrays

Session 16



Objectives

- ◆ *Define an array*
- ◆ *Explain the use of arrays*
- ◆ *Explain the process of merging arrays*
- ◆ *Explain the use of single and multi-dimensional arrays*
- ◆ *Explain the use of array-related functions*

- ◆ An array is a variable that can store a list of values referred by the same name
- ◆ Types of arrays are as follows:
 - ❖ Single-dimensional
 - ❖ Multi-dimensional

- ◆ Is a variable that can store a set of values of the same data type
- ◆ Each element of an array can be referred by an array name and an index
- ◆ Array index
 - ◆ Is used to access an element
 - ◆ Can be a number or a string
- ◆ If index is string, then array is an associative array
- ◆ If index is number, then array is an indexed array
- ◆ By default, the index value in an array starts at zero

- ◆ Two ways of initializing an array are as follows:
 - ◆ `array()` function - assigns value to all the elements of an array
 - ◆ `array identifier` - assigns value to a specific element of an array

- ◆ Uses key-value pairs separated by a comma to create an array
- ◆ The number of key-value pairs in the `array()` function determines the number of elements in an array

Syntax

```
$array_name = array([key => ] value, [key => ] value)
```

Where,

- ◆ **array_name** - specifies the array name
- ◆ **key** - specifies the index value of the array element
- ◆ **value** - specifies the value of the element
- ◆ Using the `array()` function, both the indexed and the associative arrays can be initialized



Indexed Arrays

- ◆ Includes an integer as the index type
- ◆ By default, PHP creates an indexed array, if the index type is not specified at the time of creating an array
- ◆ The index value can start with any integer, such as 1, 20, or 123

- ◆ Creating an indexed array named **department**

Snippet

```
<?php  
// Creating an array and storing values  
$department = array (1 => 'Accounts', 2 => 'Economics',  
3 => 'Computers', 4 => 'Marketing');  
// Displaying the element of the array.  
echo $department [1];  
?>
```

Displays the following output:



In the code, the array index type is an integer.

The department array contains four values, Accounts, Economics, Computers, and Marketing.

When the first element of the array is called, the output returned is **Accounts**.

- ◆ Is an array where the index type is a string
- ◆ The index value must be specified within double quotes

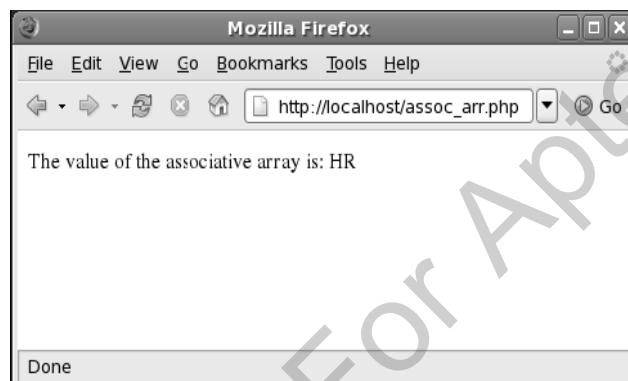
For Aptech Center Use Only

◆ Creating an associative array named **associate**

Snippet

```
<?php  
$associate = array("a" => 'Finance', "b" => 'Sales', "c" => 'HR',  
                   "d" => 'Purchase');  
echo "The value of the associative array is:";  
echo $associate["c"];  
?>
```

Displays the following output:



In the code, the array index type is a **string**.

The index value starts from **a**. and are specified within double quotes.

The department array contains four values Finance, Sales, HR, and Purchase. The statement `echo $associate["c"];` displays the value associated with the index "c".

- ◆ Enables to initialize the value of a specific element in an array

Syntax

```
$array_name[key] = "element_value";
```

where,

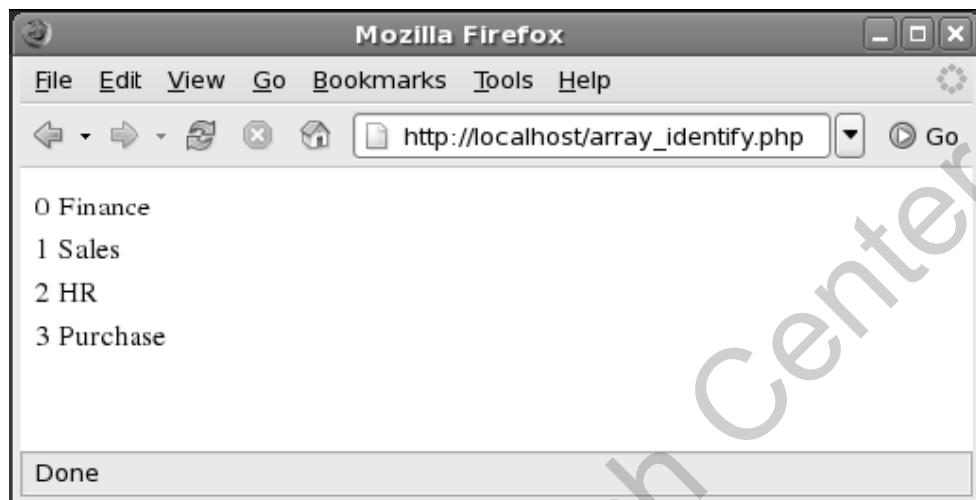
- ◆ **array_name** - specifies the name of the array
- ◆ **key** - Specifies the index value of the array element
- ◆ **element_value** - specifies the value assigned to the element of the array

- ◆ Using array identifiers to create an array named **department**

Snippet

```
<?php
$department[0]= "Finance";
$department[1]= "Sales";
$department[2]= "HR";
$department[3]= "Purchase";
$no_of_element = count($department);
for ($i=0; $i< $no_of_element; $i++)
{
    $rec = each($department);
    echo "$rec[key] $rec[value] ";
    echo "<br>";
}
?>
```

Displays the following output:



In the code, the `count()` function calculates the number of elements in the array and the result is stored in `$no_of_element` variable.

The `each()` function retrieves each key value pair of an array and stores the result in the `$rec` variable.

The `for` statement will continue to retrieve values of the array, till it reaches the last key value pair.

- ◆ The `array_merge()` function is used to combine the element values of two or more arrays

Syntax

```
$merged_array_name = array_merge($first_array, $second_array);
```

Where,

- ◆ `$merged_array_name` - specifies the name of the new array that will contain the merged element values
- ◆ `$first_array` and `$second_array` - specifies the names of the arrays whose elements are to be merged

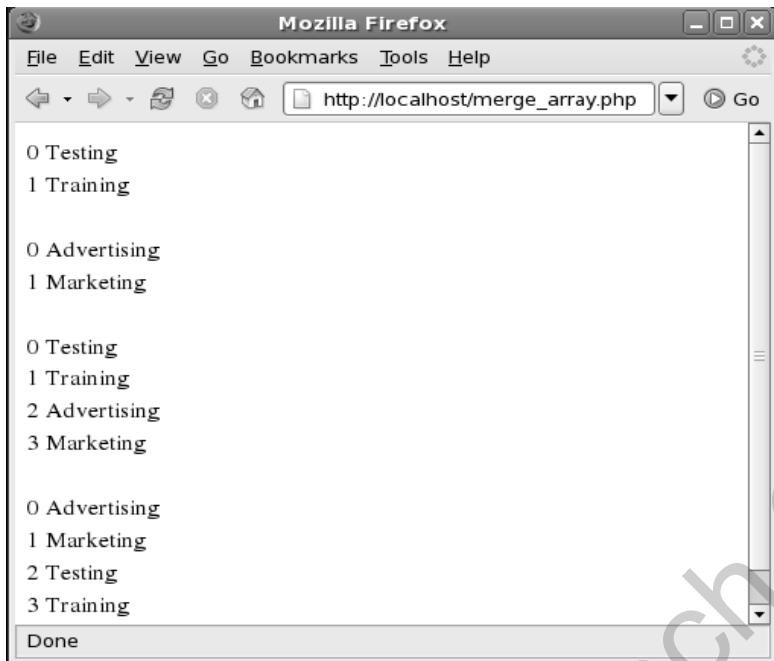
- ◆ Demonstrating the merging of the arrays

Snippet

```
<?php
$ITdept = array(0 => "Testing", 1 => "Training");
$Salesdept = array(0 => "Advertising", 1 => "Marketing");
$num_of_element = count($ITdept);
for ($i=0; $i< $num_of_element; $i++)
{
    $rec = each($ITdept);
    echo "$rec[key] $rec[value] ";
    echo "<br>";
}
echo "<br>";
$num_of_element = count($Salesdept);
for ($i=0; $i< $num_of_element; $i++)
{
    $rec = each($Salesdept);
    echo "$rec[key] $rec[value] ";
    echo "<br>";
}
echo "<br>";
echo "$rec[key] $rec[value] ";
```

```
$AdminDept = array_merge($ITdept, $Salesdept);
$num1_of_element = count($AdminDept);
for ($i=0; $i< $num1_of_element; $i++)
{
    $rec = each($AdminDept);
    echo "$rec[key] $rec[value] ";
    echo "<br>";
}
echo "<br>";
$AdminDept = array_merge($Salesdept, $ITdept);
$num2_of_element = count($AdminDept);
for ($i=0; $i< $num2_of_element; $i++)
{
    $rec = each($AdminDept);
    echo "$rec[key] $rec[value] ";
    echo "<br>";
}
echo "<br>";
?>
```

Displays the following output:



In the code, the array **AdminDept**, will include values, such as **Testing**, **Training**, **Advertising**, and **Marketing**.

The element of the array that is mentioned first in the `array_merge()` function gets the first index number.

By default, PHP allots zero as the index number to the first array element.

The first element value of the next array, **Salesdept**, is allotted an index number after the first array mentioned in the function has been allotted an index number.

- ◆ Contains one array stored within another
- ◆ In a multi-dimensional array, each element is an array
- ◆ Each element requires an array name and multiple set of indices

Syntax

```
$array_name = array(array(key => value), array(key => value));
```

Where,

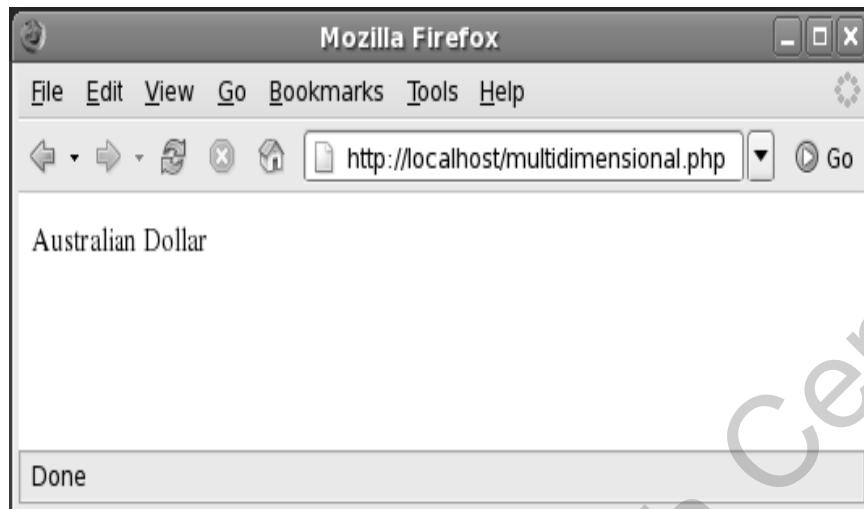
- ◆ **\$array_name** - specifies the name of the multi-dimensional array
- ◆ **key** - specifies the index number of the array element
- ◆ **value** - specifies the value of the array element

◆ Illustrating the use of multi-dimensional arrays

Snippet

```
<?php
$country_mdlist = array(
    "USA" => array(
        "Capital" => "Washington D.C.",
        "Currency" => "US Dollar"),
    "England" => array(
        "Capital" => "London",
        "Currency" => "Pound Sterling"),
    "Australia" => array(
        "Capital" => "Canberra",
        "Currency" => "Australian Dollar"),
    "New Zealand" => array(
        "Capital" => "Wellington",
        "Currency" => "NZ Dollar"));
echo $country_mdlist["Australia"]["Currency"];
?>
```

Displays the following output:



In the code, **country_mdlist** is a multi-dimensional associative array.

It contains key indices such as **USA**, **England**, **Australia**, and **New Zealand**.

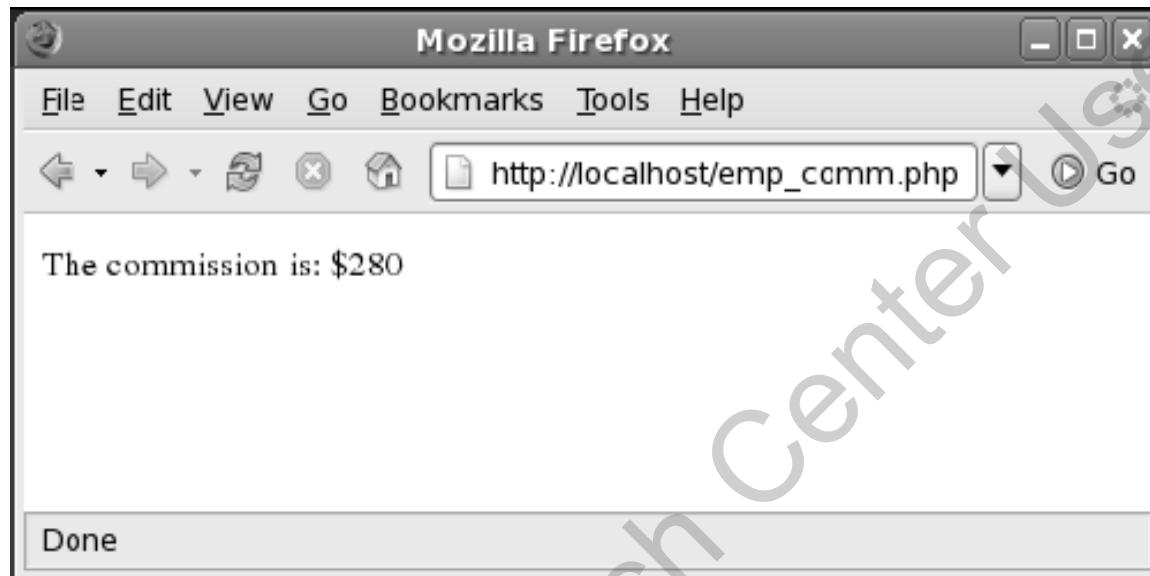
Each array element of a multi-dimensional array includes another array within it containing key indices such as **Capital** and **Currency**.

- Creating an array that stores the employee commission details

Snippet

```
<?php $employee_det = array(  
    "Employee 1" => array(  
        1 => "$100",  
        2 => "$150",  
        3 => "$100",  
        4 => "$160",  
        5 => "$250",  
        6 => "$148"),  
    "Employee 2" => array(  
        1 => "$180",  
        2 => "$195",  
        3 => "$200",  
        4 => "$130",  
        5 => "$280",  
        6 => "$218"));  
echo "The commission is: ";  
echo $employee_det["Employee 2"] [5];?>
```

Displays the following output:



In the code, both associative and indexed indices are used to create a multi-dimensional array.

- ◆ Some of the array-related functions supported by PHP are as follows:
 - ◆ `sort()` Function
 - ◆ `rsort()` Function
 - ◆ `arsort()` Function

sort() Function

- ◆ Arranges the element values in alphabetical order

Syntax

```
sort (ArrayName)
```

Where,

- ◆ **sort** - arranges the element values in alphabetical order
- ◆ **ArrayName** - specifies the name of the array whose elements are to be sorted

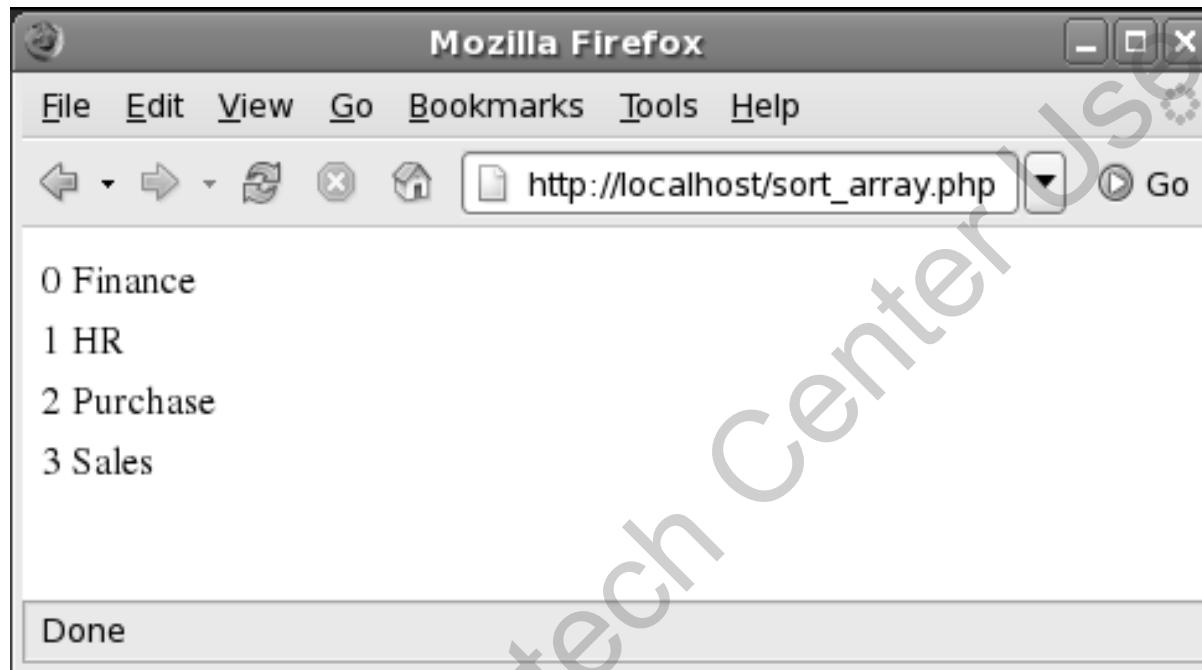
sort() Function

- ◆ Illustrating the sorting of the department array

Snippet

```
<?php
$department[0]= "Finance";
$department[1]= "Sales";
$department[2]= "HR";
$department[3]= "Purchase";
sort($department);
$no_of_element = count($department);
for ($i=0; $i< $no_of_element; $i++)
{
    $rec = each($department);
    echo "$rec[key] $rec[value] ";
    echo "<br>";
}
?>
```

Displays the following output:



The code displays the elements of the array in alphabetical order.

The order of the element values have changed.

However, the order of the **index values** is constant.

rsort() Function

- ◆ Sorts the element values in descending alphabetical order

Syntax

```
rsort (ArrayName)
```

Where,

- ◆ **rsort** - arranges the element values in descending alphabetical order
- ◆ **ArrayName** - specifies the name of the array whose elements are to be sorted

- Illustrating the use of the `rsort()` function to display the values of the department array in the descending alphabetical

Snippet

```
<?php
$department[0]= "Finance";
$department[1]= "Sales";
$department[2]= "HR";
$department[3]= "Purchase";
rsort($department);
$no_of_element = count($department);
for ($i=0; $i< $no_of_element; $i++)
{
    $rec = each($department);
    echo "$rec[key] $rec[value] ";
    echo "<br>";
}
?>
```

Displays the following output:



The code displays the elements of the array in descending alphabetical order.

The order of the element values have changed.

However, the order of the index values is constant.

arsort() Function

- ◆ Similar to rsort () function
- ◆ The only difference between rsort () and arsort () function is that the arsort () function can sort both associative and indexed arrays

Syntax

```
arsort (ArrayName)
```

arsort() Function

- ◆ Demonstrating the use of the arsort () function on the array

Snippet

```
<?php  
$department[0]= "Finance";  
$department[1]= "Sales";  
$department[2]= "HR";  
$department[3]= "Purchase";  
arsort($department);  
  
$no_of_element = count($department);  
for ($i=0; $i< $no_of_element; $i++)  
{  
    $rec = each($department);  
    echo "$rec[key] $rec[value] ";  
    echo "<br>";  
}  
?>
```

Displays the following output:

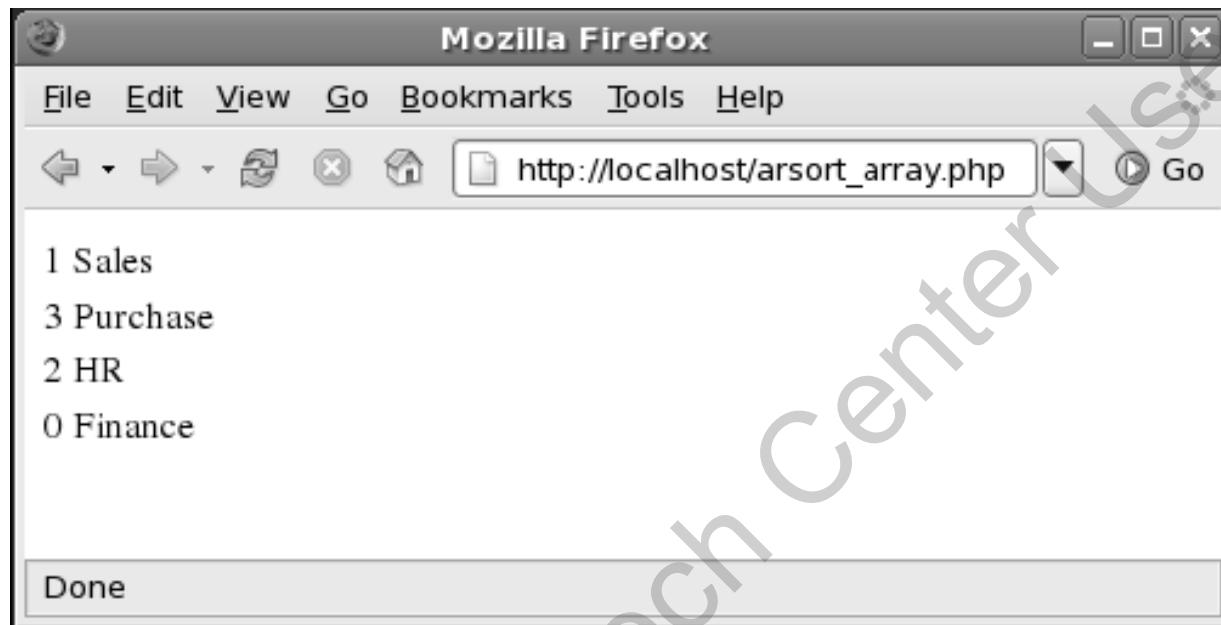


Table lists different functions to manipulate arrays

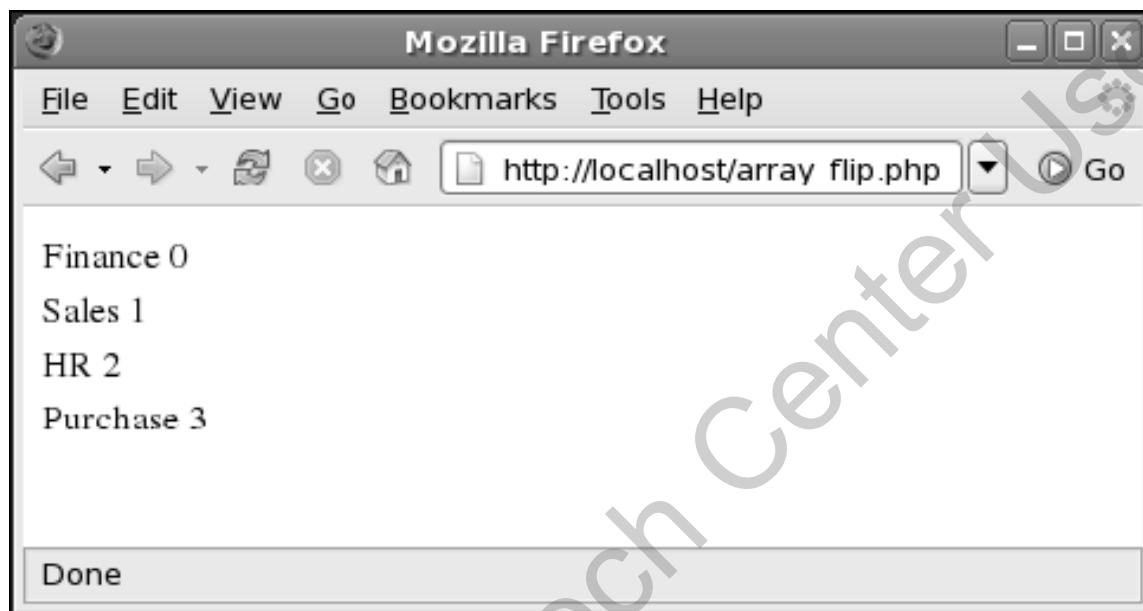
Function Name	Description
count()	Returns the number of elements in an array
sizeof()	Returns the number of elements in an array. You can use this function instead of count()
array_count_values()	Maintains a count of the occurrences of same element values in an array. It returns the number of occurrences of same element values
array_flip()	Converts the element values to index values and vice versa
array_intersect()	Identifies and returns the common element value among a group of arrays
array_keys()	Displays all the key indices of the specified array
array_reverse()	Reverses the order of the array elements
array_shift()	Returns and removes the first element of an array
array_key_exists()	Identifies whether or not a given key or index exists in an array
array_push()	Adds one or more elements to the end of an array
array_pop()	Pops and returns the last value of an array

- ◆ To flip the element values to the index values and the index values to the element values of the department array

Snippet

```
<?php
$department[0]= "Finance";
$department[1]= "Sales";
$department[2]= "HR";
$department[3]= "Purchase";
$dept = array_flip($department);
$no_of_element = count($department);
for ($i=0; $i< $no_of_element; $i++)
{
    $rec = each($dept);
    echo "$rec[key] $rec[value] ";
    echo "<br>";
}
?>
```

Displays the following output:

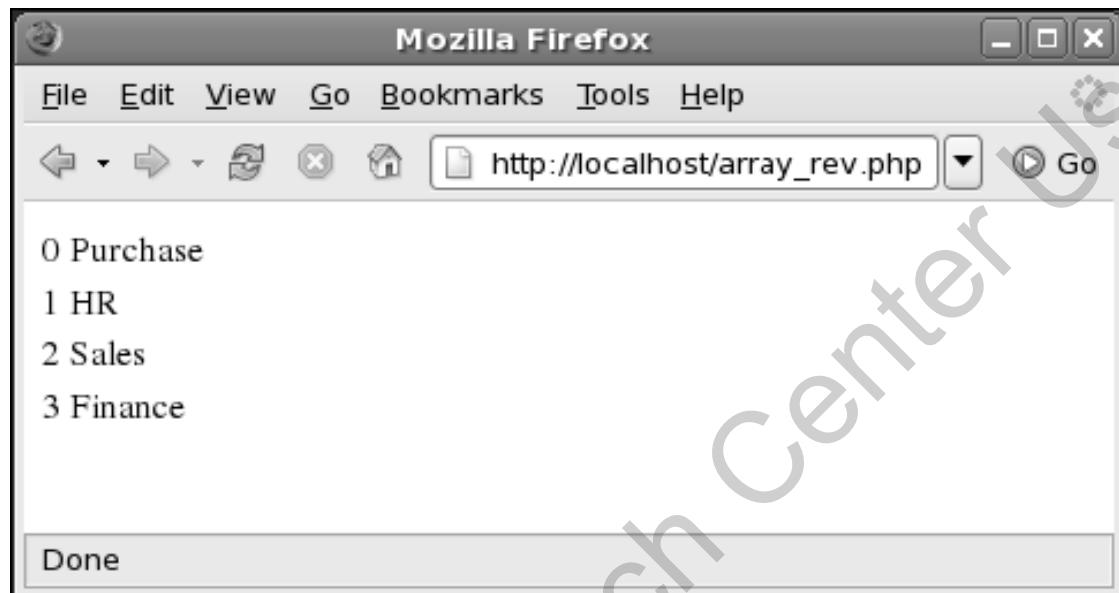


- Reversing the order of elements of the department array

Snippet

```
<?php
$department[0] = "Finance";
$department[1] = "Sales";
$department[2] = "HR";
$department[3] = "Purchase";
$dept = array_reverse($department);
$no_of_element = count($department);
for ($i=0; $i< $no_of_element; $i++)
{
    $rec = each($dept);
    echo "$rec[key] $rec[value] ";
    echo "<br>";
}
?>
```

Displays the following output:

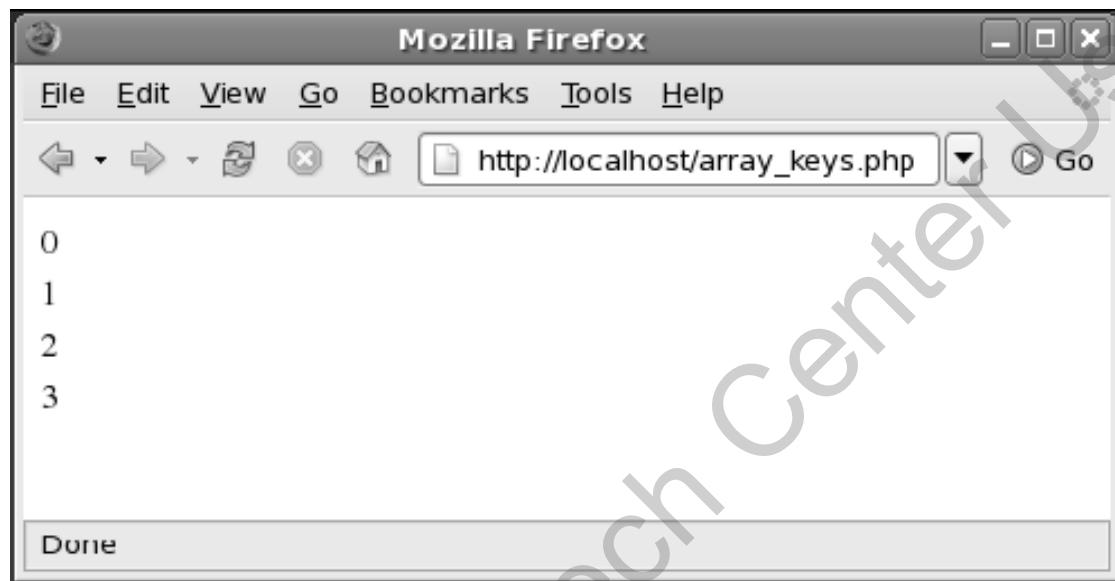


- ◆ Viewing all the key values of the department array

Snippet

```
<?php  
$department[0]= "Finance";  
$department[1]= "Sales";  
$department[2]= "HR";  
$department[3]= "Purchase";  
$dept = array_keys($department);  
$no_of_element = count($department);  
for ($i=0; $i< $no_of_element; $i++)  
{  
    $rec = each($dept);  
    echo "$rec[value] ";  
    echo "<br>";  
}  
?>
```

Displays the following output:

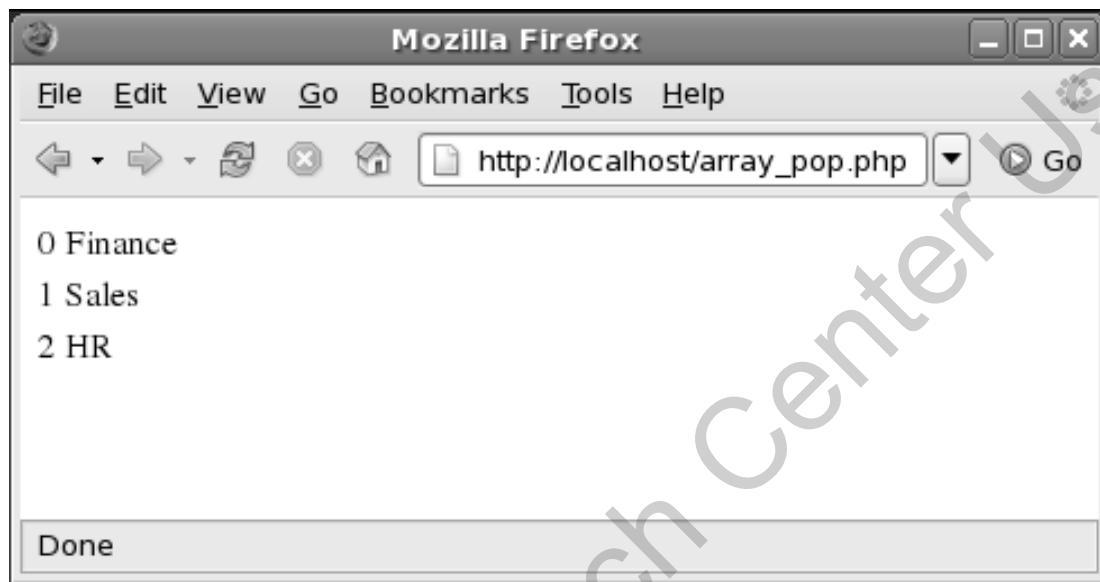


- ◆ Removing an element value from the department array

Snippet

```
<?php
$department[0]= "Finance";
$department[1]= "Sales";
$department[2]= "HR";
$department[3]= "Purchase";
array_pop($department);
$no_of_element = count($department);
for ($i=0; $i< $no_of_element; $i++)
{
    $rec = each($department);
    echo "$rec[key] $rec[value] ";
    echo "<br>";
}
?>
```

Displays the following output:

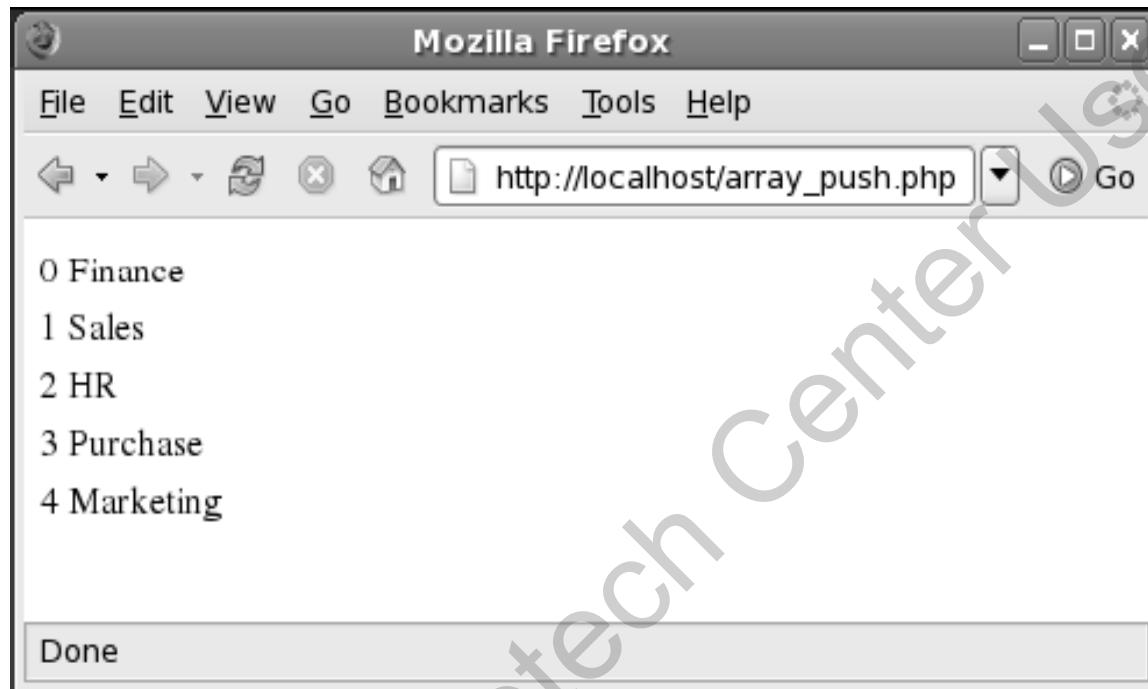


- ◆ Adding an element value to the department array

Snippet

```
<?php
$department[0]= "Finance";
$department[1]= "Sales";
$department[2]= "HR";
$department[3]= "Purchase";
array_push($department, "Marketing");
$no_of_element = count($department);
for ($i=0; $i< $no_of_element; $i++)
{
$rec = each($department);
echo "$rec[key] $rec[value]";
    echo "<br>";
}
?>
```

Displays the following output:



- ◆ An array is a variable that can store a set of values of the same data type
- ◆ All the elements in an array are referenced by a common name
- ◆ An array index is used to access an element
- ◆ Merging arrays is the process of combining element values of two or more arrays
- ◆ In a single-dimensional array, the element includes only one level of key value pairs

- ◆ In a multi-dimensional array, each element is an array. Each element requires an array name and multiple set of indices
- ◆ An indexed array is an array where the index type is integer and an associative array is an array where the index type is string
- ◆ The sort() function arranges the element values in alphabetical order, the rsort() function sorts the element values in descending alphabetical order, and the arsort() function sorts both associative and indexed arrays

Handling Databases with PHP

Session 18

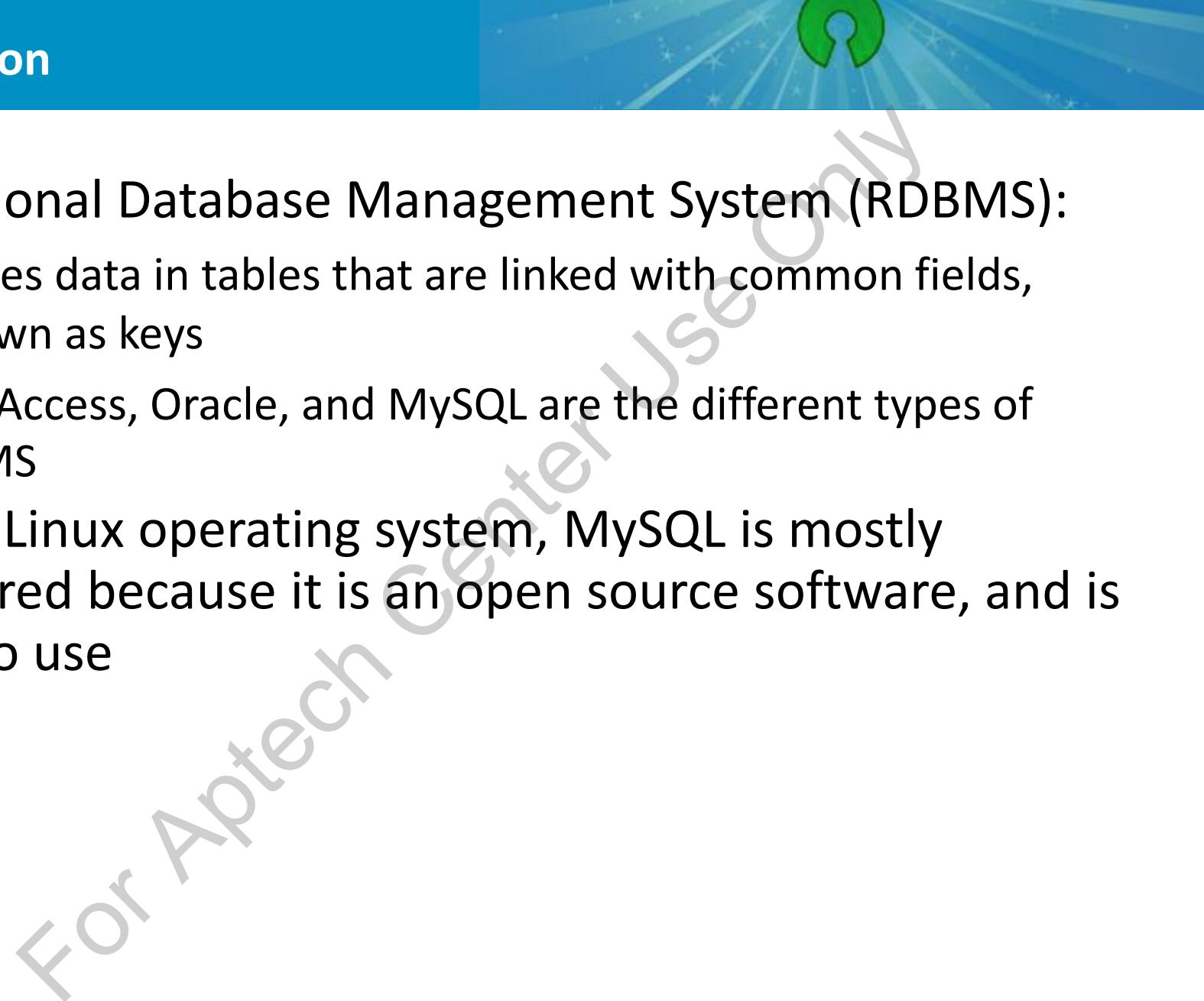


Objectives

- ◆ *Describe Database APIs*
- ◆ *Explain the process of connecting to a database*
- ◆ *Explain the use of data access functions*
- ◆ *Explain SQL queries using PHP*
- ◆ *Explain HTML tables using SQL queries*

For Aptech Center Use Only

- ◆ Relational Database Management System (RDBMS):
 - ❖ Stores data in tables that are linked with common fields, known as keys
 - ❖ MS Access, Oracle, and MySQL are the different types of RDBMS
- ◆ In the Linux operating system, MySQL is mostly preferred because it is an open source software, and is easy to use



- ◆ Database APIs allows developers to write applications that are movable or easily accessible between the database products
- ◆ Some of the common database APIs are:
 - ❖ Native-Interface
 - ❖ Open Database Connectivity (ODBC)
 - ❖ Java Database Connectivity (JDBC)
 - ❖ Common Object Request Broker Architecture (CORBA)

- ◆ PHP supports MySQL database for accessing data from the database
 - ❖ **Connecting to a Database**
 - ❖ A Web site connects to a database to access and store information
 - ❖ Steps for connecting to a database are as follows:
 - Open the database connection
 - Work with the database
 - Close the database connection

- ◆ PHP and MySQL are automatically installed while customizing the installation of Linux operating system
- ◆ A connection needs to be establish to the MySQL server and PHP with the help of `mysql_connect()` function
- ◆ This function takes three arguments:
 - ❖ Name of the machine on which the database is running
 - ❖ Database username
 - ❖ Database user password

- ◆ Connecting to the MySQL server is as follows:

Syntax

```
$link_id = mysql_connect ("host_name", "user_name", "password");
```

Where,

host_name - specifies the name of the server on which the database is running. The default location of MySQL server is localhost

user_name - specifies the username

password - specifies the password to connect to the database

link_id - stores the return value of the connection

- ◆ **mysql.php** - Connecting to the MySQL server

Snippet

```
<?php  
$link_id  
mysql_connect("localhost","root","abc123");  
?>
```

In the code,
root is the username.
localhost is the server name.
abc123 is the password.

- ◆ Before starting work with the database, a connection needs to be established with the MySQL server
- ◆ PHP provides the following functions to work with the MySQL database:
 - ◆ **mysql_list_dbs()** - This function displays all the databases available on the server

The `mysql_list_dbs()` function is as follows:

Syntax

```
mysql_list_dbs($link_id);
```

where,

link_id - specifies the return value of the connection

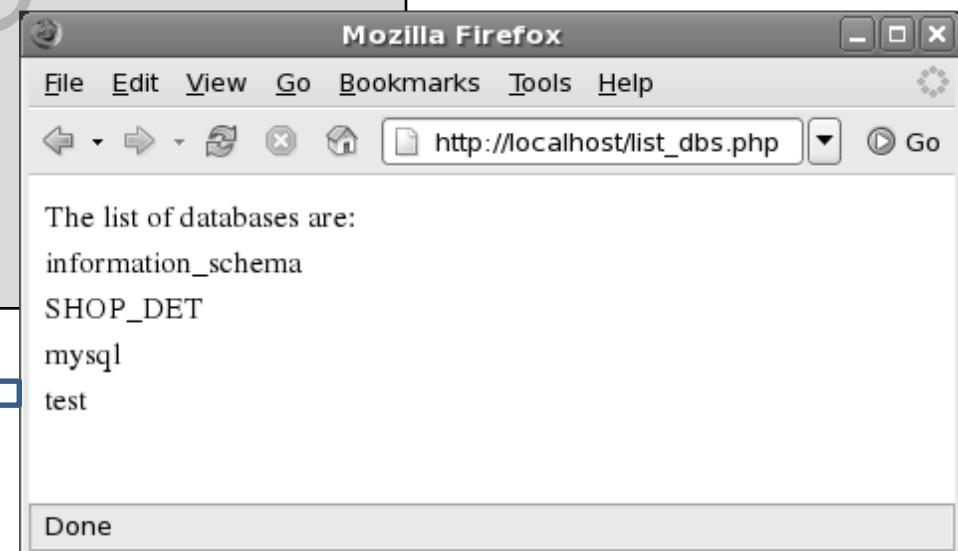
- ◆ **list_dbs.php** – Displaying all the databases present on the server

Snippet

```
<?php  
  
$connect = mysql_connect('localhost', 'root', '');  
  
$db_list = mysql_list_dbs($connect);  
  
echo "The list of databases are:<br>";  
  
while ($row = mysql_fetch_object($db_list))  
{  
  
echo $row->Database . "<br>";  
  
}  
  
?>
```

Displays the list of the databases that are present in the instance of MySQL.

Displays the output:



- ❖ **mysql_select_db()** - This function defines the database that will be used for the connection

The `mysql_select_db()` function is as follows:

Syntax

```
mysql_select_db("database name", $link_id);
```

Where,

database_name - specifies the database name

link_id - specifies the return value of the connection

- ❖ **mysql_select_db.php** - Connecting to the MySQL database

Snippet

```
<?php  
$server = "";  
$username = "root";  
$password = "";  
$connect_mysql = mysql_connect($server, $username, $password);  
  
$mysql_db = mysql_select_db("mysql", $connect_mysql);  
if (!$mysql_db)  
{  
    die("Connection failed");  
}  
else  
{  
    echo "Current Database is selected";  
}  
?>
```

- ❖ **mysql_list_tables()** - This function displays a list of all the tables available in the specified database

The `mysql_list_tables()` function is as follows:

Syntax

```
mysql_list_tables("database_name", $link_id);
```

Where,

database_name - specifies the database name

link_id - specifies the return value of the connection

- ❖ **listables.php** - To list all the tables of the MySQL database

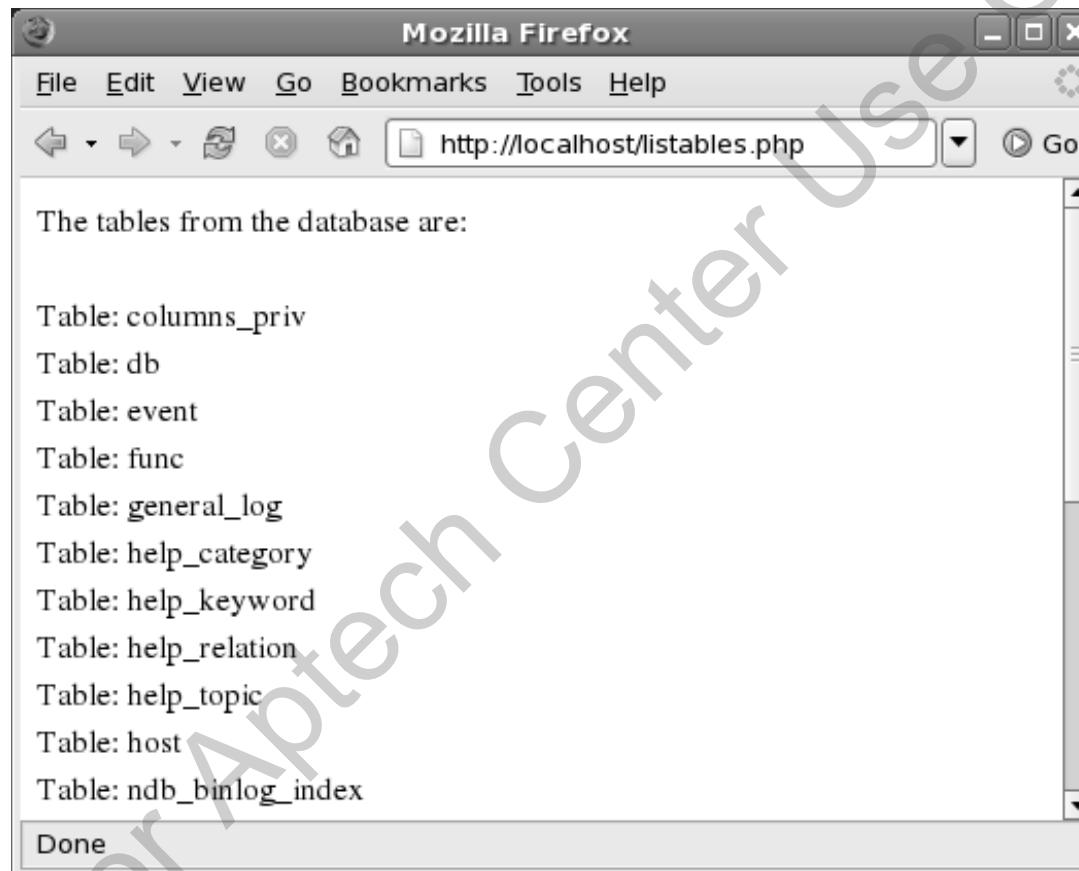
Snippet

```
<?php  
  
$dbname = 'mysql';  
  
if (!mysql_connect('127.0.0.1', 'root', ''))  
{  
    echo 'Could not connect to mysql';  
    exit;  
}  
  
$sql = "SHOW TABLES FROM $dbname";  
  
$result = mysql_query($sql);  
  
echo " The tables from the database are: <br><br>" ;
```

```
if (!$result)
    $result = mysql_query($sql);

echo " The tables from the database are: <br><br>";
if (!$result)
{
    echo "DB Error, Unable to list tables<br>";
    echo 'MySQL Error: ' . mysql_error();
    exit;
}
while ($row = mysql_fetch_row($result))
{
    echo "Table: {$row[0]}<br>";
}
?>
```

Displays the following output:



All the tables available in the mysql database are listed and stored in the \$result variable.

- ❖ **mysql_num_rows ()** - This function displays the number of rows present in the specified table

The mysql_num_rows () function is as follows:

Syntax

```
mysql_num_rows ("table_name");
```

Where,

table_name - specifies the name of the table for displaying the number of rows

- ❖ **list_rows.php** - Listing the number of rows from a table in a database

Snippet

```
<?php

$connect = mysql_connect("localhost", "root", "");

mysql_select_db("mysql", $connect);

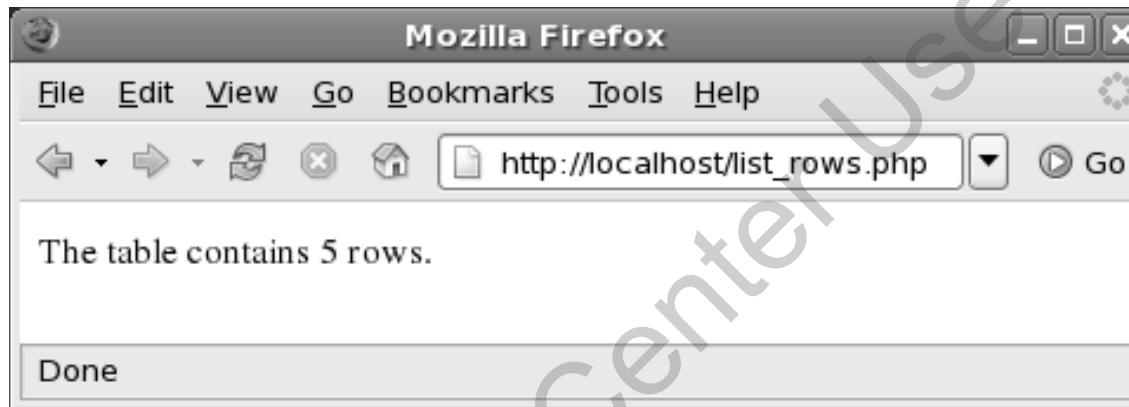
$result = mysql_query("SELECT * FROM user", $connect);

$rows = mysql_num_rows($result);

echo "The table contains $rows rows.<br>";

?>
```

Displays the following output:



The number of rows from the table are listed.

Closing the Connection 1-3

- ◆ The connection with MySQL server can be closed with the help of the `mysql_close()` function
- ◆ The `mysql_close()` function is as follows:

Syntax

```
mysql_close($link_id);
```

Where,

link_id - specifies the return value of the connection

Closing the Connection 2-3

- ◆ **conn_close.php** - Closing the connection to the MySQL database

Snippet

```
<?php

$connect = mysql_connect("localhost", "root", "");

mysql_select_db("mysql", $connect);

$result = mysql_query("SELECT * FROM user", $connect);

$rows = mysql_num_rows($result);

echo "The table contains $rows rows.<br><br>";

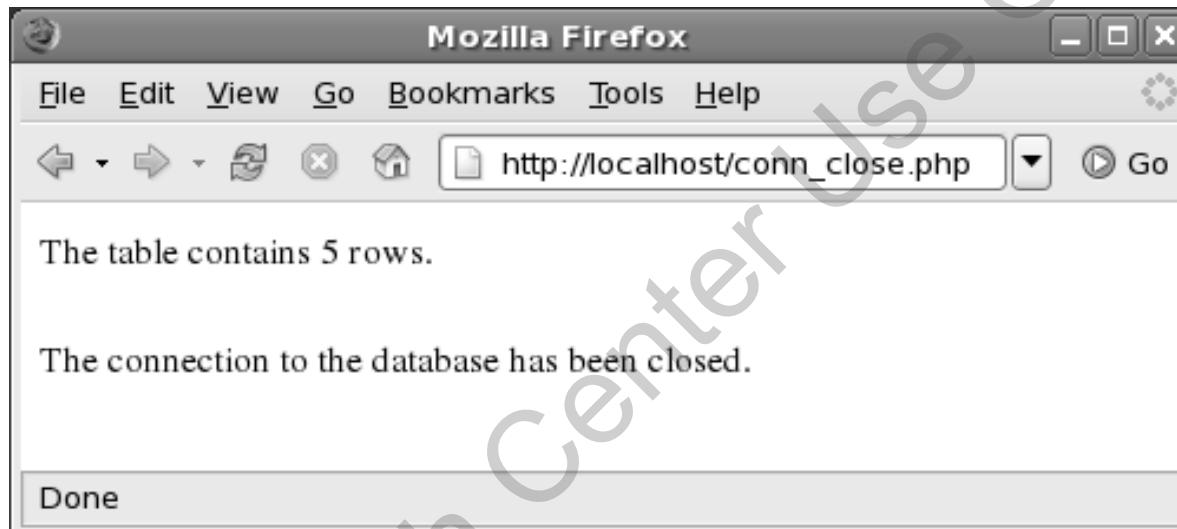
mysql_close($connect);

echo "The connection to the database has been closed.";

?>
```

Closing the Connection 3-3

Displays the following output:



- ◆ PHP provides the following functions for accessing data from the database:
 - ◆ **mysql_query()** - Executes MySQL query for retrieving data from tables and commands such as SELECT, SHOW, EXPLAIN, and DESCRIBE can be used with this function

The `mysql_query()` function is as follows:

Syntax

```
mysql_query(query, link_id);
```

Where,

query - specifies the MySQL query

link_id - specifies the return value of the connection

- ❖ **mysql_fetch_array()** - Retrieves the rows of the table and saves it as an array. It is an extended version of mysql_fetch_row() function

The mysql_fetch_array() function is as follows:

Syntax

```
mysql_fetch_array("table name");
```

Where,

table_name - specifies the name of the selected table

- ❖ **mysql_field_len()** - Displays the length of the specified field

The `mysql_field_len()` function is as follows:

Syntax

```
mysql_field_len("table_name", "field_name");
```

Where,

table_name - specifies the table name

field_name - specifies the field name for which the length needs to be displayed

- ❖ **mysql_num_fields()** - Displays the number of fields in the specified table

The `mysql_num_fields()` function is as follows:

Syntax

```
mysql_num_fields("table_name");
```

Where,

table_name - specifies the table name

- ◆ Before executing the SQL queries in PHP, a database connection must be established
- ◆ Create a table named `USER_CONTACT` in the `USER` database with the fields as shown in table

Field Name	Data Type	Constraint
USER_ID	INT	NOT NULL PRIMARY KEY
USER_NAME	CHAR(25)	NOT NULL
USER_EMAIL_ID	CHAR(25)	

- ◆ **mysqltable.php** - To create a table using SQL commands in PHP

Snippet

```
<?php
$server = "";
$username = "root";
$password = "";
$connect_mysql = mysql_connect($server, $username, $password);
if($connect_mysql)
{
echo "Connection established<BR>";
}
else
{
die("Unable to connect to the database<BR>");
}
$sql_table = "CREATE TABLE USER_CONTACT(\"USER_ID INT NOT NULL PRIMARY KEY,
\"USER_NAME CHAR(25) NOT NULL, \"USER_EMAIL_ID CHAR(25)\")";
if(mysql_query($sql_table))
{
echo "Table is created<BR>";
}
```

```
else
{
    die("Unable to create the table<BR>");
}
?>
```

Displays the following output:



The **USER_CONTACT** table is created in the **USER** database.

- ◆ In the code, the `USER_CONTACT` table is created in the `USER` database
- ◆ The table is created using the `CREATE` command in MySQL
- ◆ The records in the table are inserted with the `HTML FORM` method

For Aptech Center Use Only

- ◆ **usercontact.php** - Inserting records in the USER_CONTACT table

Snippet

```
<?php  
  
$server = "";  
  
$username = "root";  
  
$password = "";  
  
$connect_mysql      =      mysql_connect($server,      $username,  
$password);  
  
if($connect_mysql)  
{  
    echo "Connection established.";  
}  
else  
{  
    die("Unable to connect");
```

```
}

$db = "user";
mysql_db = mysql_select_db($db);
if($mysql_db)
{
    echo "<BR><BR>Connected to the database.";
}
else
{
    die("Unable to connect to the database");
}

$sql_insert = "INSERT INTO user_contact (user_id, user_name,
user_email_id)
VALUES (101,'John','john@mail.com')";
$result = mysql_query($sql_insert);
```

```
if($result)
{
echo "<BR><BR>The records have been added to the table.";
}
else
{
echo "Unable to insert records.";
mysql_error();
}
?>
```

Displays the following output:



Records are inserted in the table using the **INSERT** command

- ◆ Using the SELECT command, a data can be accessed from the tables
- ◆ **displaytable.php** - Displaying the records of the USER_CONTACT table from the USER database

Snippet

```
<?php  
  
$server = "";  
  
$username = "root";  
  
$password = "";  
  
$connect_mysql = mysql_connect($server, $username,  
$password);  
  
if($connect_mysql)
```

```
{  
    echo "Connection established<br>"; }  
  
else  
  
{  
    die("Unable to connect<br>");  
}  
  
$mysql_db = mysql_select_db("USER");  
  
if($mysql_db)  
  
{  
    echo "Connected to the database<br>";  
}  
  
else  
  
{  
    die("Unable to connect to the database<br>");  
}
```

```
}

$sql_disp = ("SELECT * FROM USER_CONTACT;");

echo "<br>Displaying Records from the USER_CONTACT table:<br>";

$result = mysql_query($sql_disp);

while ($row = mysql_fetch_array($result))

{

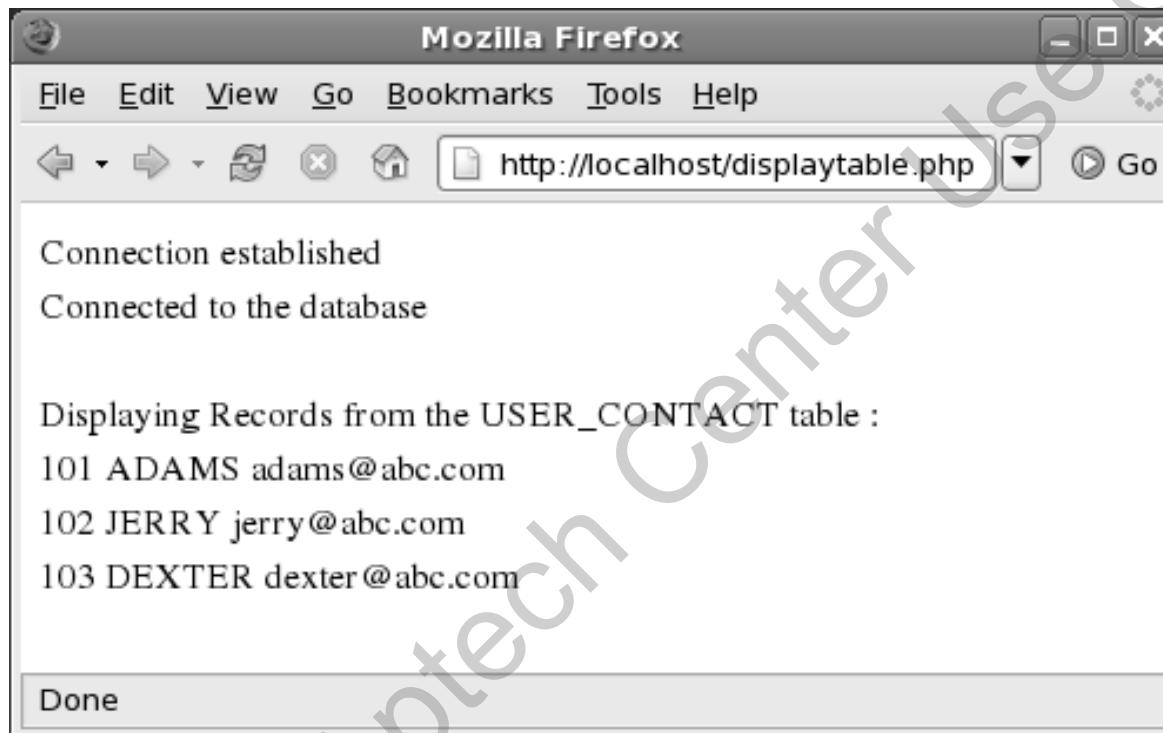
    echo "$row[USER_ID] ";

    echo "$row[USER_EMAIL_ID] <br>";

}

?>
```

Displays the following output:



The records such as **USER_ID**, **USER_NAME**, and **USER_EMAIL_ID** from the **USER_CONTACT** table are displayed

- ◆ The DELETE and UPDATE commands enable to modify the contents of the table
- ◆ **delete_record.php** - To delete a record from the table

Snippet

```
<?php  
  
$server = "";  
  
$username = "root";  
  
$password = "";  
  
$connect_mysql = mysql_connect($server, $username, $password);  
  
if($connect_mysql)  
{  
    echo "Connection established<br>";  
}  
}
```

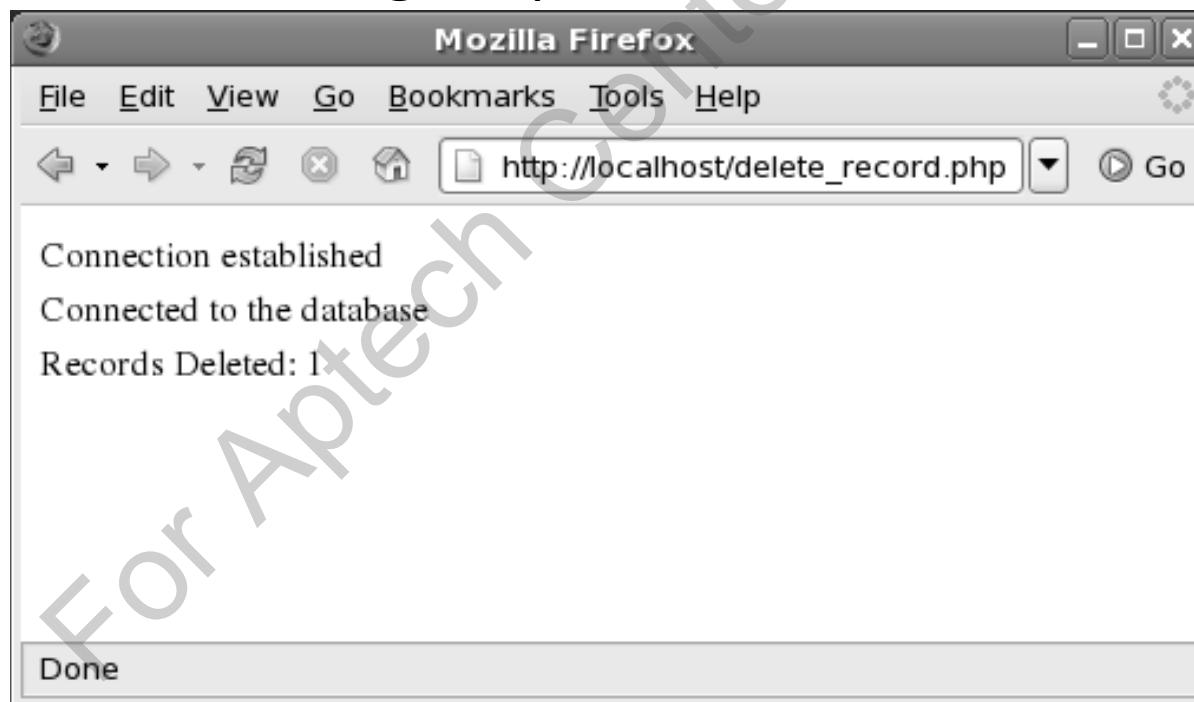
```
else
{
    die("Unable to connect<br>");
}

$mysql_db = mysql_select_db("USER");
if($mysql_db)
{
    echo "Connected to the database<br>";
}
else
{
    die("Unable to connect to the database<br>");
}

$sql_delete = ("DELETE FROM USER_CONTACT WHERE USER_ID = '101'");
$result = mysql_query($sql_delete);
if($result)
{
    echo "Records Deleted: $result<br>";
}
```

```
else
{
    echo "RECORDS NOT FOUND IN THE TABLE<br>";
    mysql_error();
}
?>
```

Displays the following output:



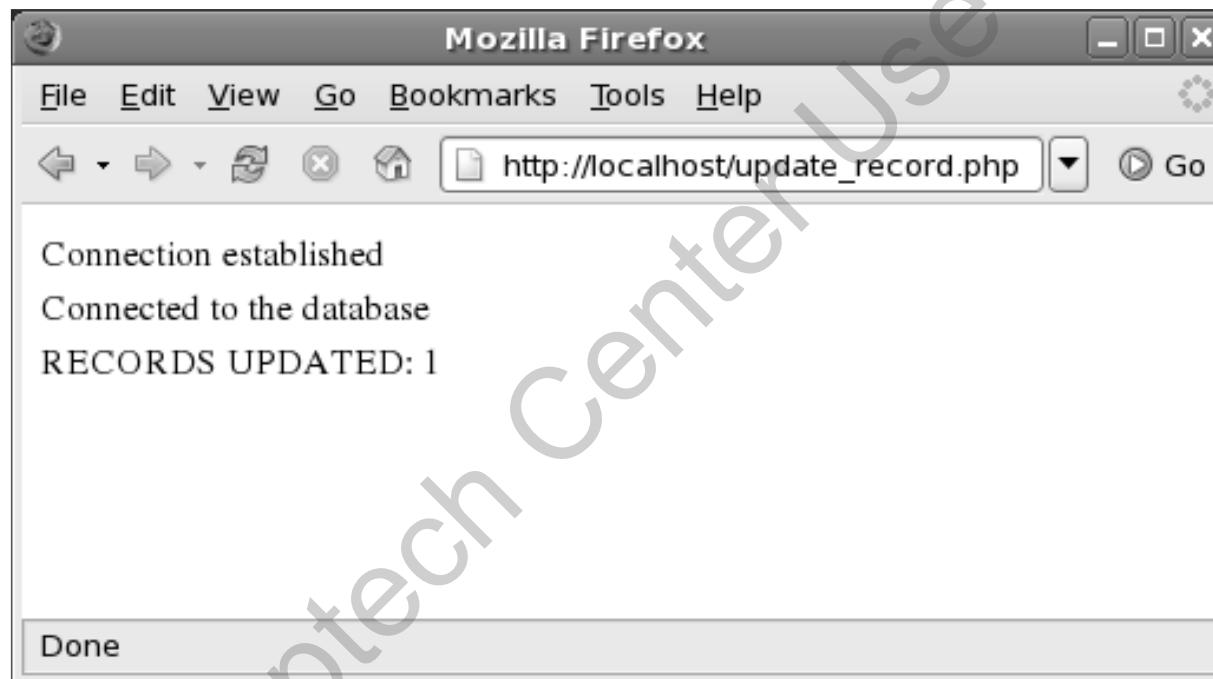
- ◆ **update_record.php** - Updating a record in the table

Snippet

```
<?php  
$server = "";  
$username = "root";  
$password = "";  
$connect_mysql = mysql_connect($server, $username, $password);  
if($connect_mysql)  
{  
    echo "Connection established<br>";  
}  
else  
{  
    die("Unable to connect<br>");  
}  
$mysql_db = mysql_select_db("USER");  
if($mysql_db)
```

```
{  
    echo "Connected to the database<br>";  
}  
else  
{  
    die("Unable to connect to the database<br>");  
}  
  
$sql_update=("UPDATE USER_CONTACT SET USER_NAME ='David' WHERE USER_ID ='102' ");  
$result=mysql_query($sql_update);  
if($result)  
{  
    echo "RECORDS UPDATED: $result<br>";  
}  
else  
{  
    echo "UNABLE TO UPDATE RECORDS<br>";  
    mysql_error();  
}  
?>
```

Displays the following output:



- ◆ HTML supports database application components for accessing the database
- ◆ The contents of the SQL tables can be displayed on the Web browser by building an HTML table structure

For Aptech Center USE ONLY

- ◆ **display_records.php** - Displaying all the records of the user_contact

Snippet

```
<HTML>
<BODY>
<?php
$server = "";
$username = "root";
$password = "";
$connect_mysql = mysql_connect($server, $username, $password);
if($connect_mysql)
echo "Connection established";
$mysql_db = mysql_select_db("USER");
if($mysql_db)
echo "<BR><BR>Connected to the database<BR><BR>";
echo "<TABLE BORDER BGCOLOR=\"WHITE\">";
```

```
echo "<TR><TH>USER_ID</TH><TH>USER_NAME</TH><TH>USER_EMAIL_ID </TH>";  
echo "<DBQUERY q> select * FROM USER_CONTACT";  
echo "<DBROW><TR><TD><? q.USER_ID></TD><TD><? q.USER_NAME></TD><TD><?  
q.USER_EMAIL_ID></TD></TR>";  
echo "</DBQUERY>";  
echo "</TR>";  
echo "</TABLE>";  
?  
</BODY>  
</HTML>
```

Displays the following output:



Displays the records of the **user_contact** table on the Web browser in a tabular format.

Summary

- ◆ Database APIs enable developers to write applications that are movable or easily accessible between the database products
- ◆ The common database APIs are Native-Interface, ODBC, JDBC, and CORBA
- ◆ PHP is connected to MySQL using three arguments: the MySQL server host name, the MySQL user name, and the MySQL user password
- ◆ The connection with the server is established with the help of `mysql_connect()` function
- ◆ The basic PHP functions that are used with respect to the database are: `mysql_list dbs()`, `mysql_select_db()`, `mysql_list_tables()`, and `mysql_num_rows()`

Summary

- ◆ The mysql_close() function closes the connection with the MySQL server
- ◆ The data access functions used in PHP are: mysql_query(), mysql_fetch_array(),mysql_fetch_row(), mysql_fetch_field(), mysql_field_len(), and mysql_num_fields()

Working with Cookies

Session 19



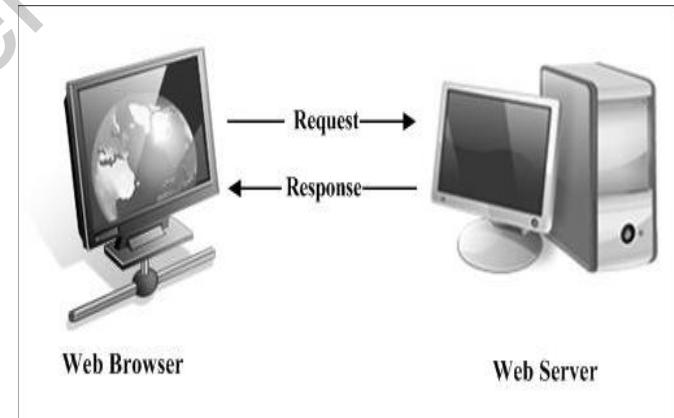
Objectives

- ◆ *Describe the process of setting a cookie*
- ◆ *Explain the process of retrieving a cookie in PHP*
- ◆ *Explain the process to delete a cookie*
- ◆ *Identify the drawbacks associated with cookies*

For Aptech Center Use Only

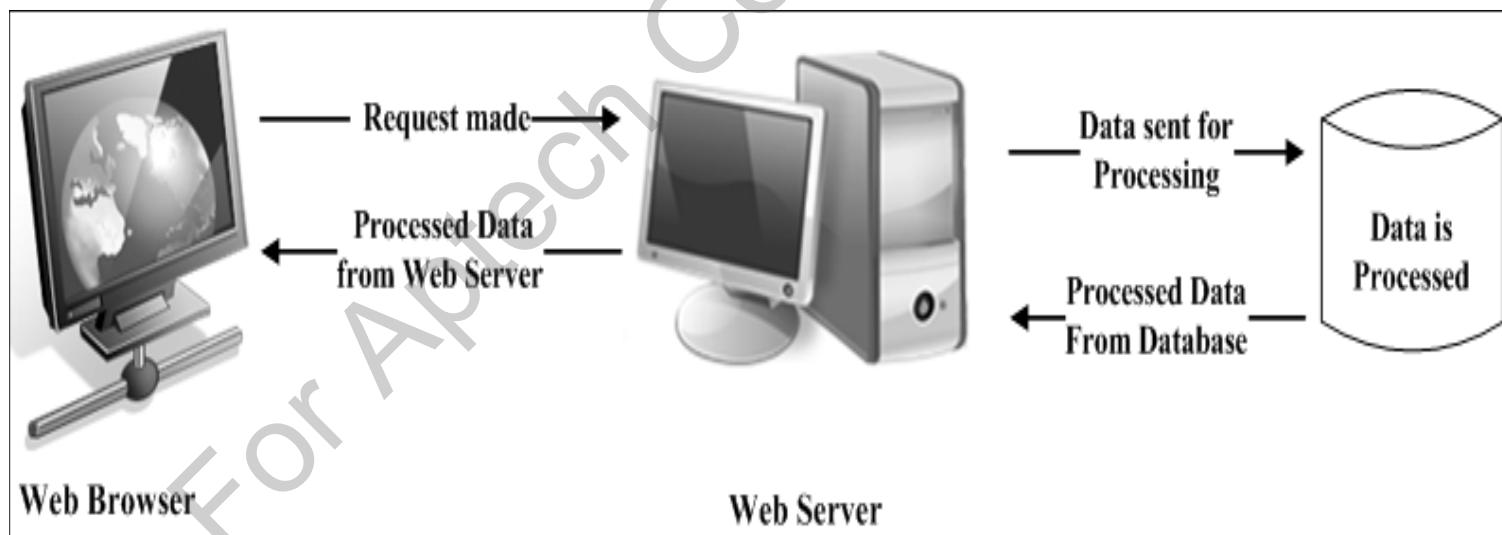
- ◆ Web sites store user information in databases to maintain a track of their visits
- ◆ Cookies enable Web sites to store user information
- ◆ PHP supports Hyper Text Transfer Protocol (HTTP) cookies

- ◆ Uses HTTP protocol for sending information to the server
 - ◆ HTTP is a stateless protocol, because the execution of the current command is completed without the knowledge of commands that came before it
- ◆ Are of two types:
 - ◆ Static Web pages
 - ◆ Dynamic Web pages
- ◆ Static Web pages
 - ◆ Web browser requests for a page and the server completes the request by sending the required file
 - ◆ Does not involve any interaction with the user



- ◆ Dynamic Web pages

- ◆ Require user interaction, so scripting languages such as JavaScript, PHP, and ASP are used
- ◆ Accept information from the user and record it for further processing



- ◆ Data stored by Web sites are as follows:
 - ❖ Temporary information is stored in cookies for a stipulated period
 - ❖ Permanent data is stored in cookies for a certain period and then the required information is saved in the database
- ◆ Types of cookies are as follows:
 - ❖ **Persistent** - exist in the Web browser for a period specified at the time of its creation
 - ❖ **Non-persistent** - deleted from the Web browser as soon as the user exits the browser

- ◆ Web sites use cookies to determine the following:
 - ❖ Number of times the user has visited the Web site
 - ❖ Number of new visitors
 - ❖ Number of regular users
 - ❖ Frequency of a user visiting the Web site
 - ❖ Date on which the user had last visited the Web site
 - ❖ Customized Web page settings for a user

Working of a Cookie

- When a user visits the Web site for the first time, the Web server creates a unique ID and sends the ID in the cookie to the Web browser
- Browser stores the cookie and sends it back to the Web site in subsequent requests
- Life of a cookie depends on the expiration time and date
- Cookie is stored on the hard disk of the user's computer which enables the Web site to keep a track on the user visiting the Web site
- Web servers and Web browsers send cookies to each other in HTTP headers
- Web server sends the cookie to the browser in the `setcookie` header field which is part of the HTTP response
- Web browser stores the cookie and uses the same in subsequent requests to the same Web server

- ◆ Consider the following HTTP response header:

Snippet

```
HTTP/2.0 200

Content-Length: 8451

Content-Type: text/html

Date: Mon, 27 Dec 2010 05:29:24 GMT

Expires: Mon, 27 Dec 2010 05:29:44 GMT

setcookie: city=east-coast-usa
```

- ◆ In the code, the following information is displayed:
 - ❖ Version number of the HTTP protocol
 - ❖ Size of the content
 - ❖ Type of the content
 - ❖ Date and time of response
 - ❖ Expiry date and time of the cookie
 - ❖ Cookie header

- ◆ Consider the following HTTP response header:

Snippet

```
GET /usa/florida.php HTTP/2.0

Connection: Keep-Alive

Cookie: city=east-coast-usa

Host: www.Webworldmaps.com

Referrer: http://www.Webworldmaps.com/
```

Cookie can be defined using the setcookie function.

Code displays a subsequent request that the Web browser sends to the Web server.

- ◆ Setting a cookie is sending the cookie to the browser
- ◆ PHP uses two functions, `setcookie()` and `setrawcookie()` to set a cookie
- ◆ `setrawcookie()` function sends a cookie without encoding the cookie value
- ◆ `setcookie()` function generates the cookie header field that is sent along with the rest of the header information

- ◆ The `setcookie()` function is as follows:

Syntax

```
setcookie(name, value, expiry date, path, domain, secure)
```

Where,

name - defines the name of the cookie

value - defines the value of the cookie that is stored on the client system

expiry date - defines the date and time (UNIX timestamp) when the cookie will expire

path - defines the location on the server where the cookie will be stored.

domain - defines the domain name where the cookie is made available

secure - defines the type of HTTP connection that the cookies will pass through

Setting a Cookie

- ◆ When the cookie is set, the value is automatically encoded in the URL
- ◆ When the script retrieves a cookie, it automatically decodes the value from the URL
- ◆ Cookies are a part of the HTTP header and there can be more than one cookie in the header, but it should relate to the same domain or Web site
- ◆ The code related to the cookies must be specified before the following:
 - ❖ HTTP header
 - ❖ Displaying any content
 - ❖ Any white space
- ❖ If any content is displayed before calling the `setcookie()` function, the function will fail and return `False`
- ❖ If the `setcookie()` function runs successfully, the function returns `True`

- ◆ Setting a cookie that expires in one day in a Web site that displays country maps when a user enters a country name in the search feature of the Web site are as follows:

Snippet

```
$mapname = $_GET['fmapname'];  
  
setcookie("mycookie", $mapname, time() +86400,  
"/Webmap/", ".Webworldmaps.com");
```

- ◆ In the code, \$mapname is the variable that contains the country name that the user enters
- ◆ The \$mapname variable stores the value that the GET method retrieves from the form
- ◆ The setcookie () function includes the following:
 - ◆ mycookie - defines the name of the cookie
 - ◆ time () +86400 - specifies the time when the cookie will expire
 - ◆ /Webmap - defines the location where the cookie will be stored
 - ◆ .Webworldmaps.com - specifies the domain that the cookie will use

- ◆ Creating a cookie that expires when the Web browser window is closed are as follows:

Snippet

```
$val = $_GET['uname'];  
setcookie("uname", $val);
```

In code, `uname` is the variable that contains a value.
The `$val` variable stores the value of `uname` that the GET method retrieves.
The `setcookie()` function in the code snippet sets a cookie named `uname`.
The value of `$val` is assigned to the cookie, `uname`.

- ◆ Cookies are useful only when the Web server can retrieve the information from it
- ◆ The Web browser matches the URL against a list of all the cookies present on the client system
- ◆ If the Web browser finds a match, a line containing the name value pairs of the matched cookie is included in the HTTP header
- ◆ Document that created the cookie as well as that are present in the same directory can access it
- ◆ Documents outside the directory need to include the path or the domain name of the cookie to access the cookie

- ◆ PHP provides three ways of retrieving a cookie value and they are as follows:
 - ◆ Passing a variable as the cookie name - retrieve the cookie value, use the variable as the cookie name. The following code snippet displays a cookie value:

Snippet

```
echo $cookie_name;
```

- ◆ This method of retrieving the cookie value is not recommended as PHP will start searching all the variables present in the client system
- ◆ The register_globals option must be enabled in the configuration file

- ❖ Using `$_COOKIE` array

- ❖ PHP uses cookie name as a variable to retrieve the cookie value.
PHP can also use an associative array called `$_COOKIE` to retrieve the cookie value
- ❖ The `$_COOKIE` is a global variable that reads a value of the cookie
- ❖ An example of this is shown as follows:

Snippet

```
echo $_COOKIE ['$cookie_name'];
```

- ❖ This is more reliable and faster than retrieving the cookie value through a variable

Snippet

```
<?php  
  
$cookieval = $_COOKIE ['uname'] ; ?>  
  
<HTML>  
  
<BODY>  
  
<?php  
  
if (isset($cookieval))  
  
{  
  
echo "Welcome $cookieval";  
  
}
```

- ◆ **retrieve_cookie.php** - Retrieving a cookie value using the `$_COOKIE` global variable

Snippet

```
else
{
    echo "You need to log in";
}
?>
</BODY>
</HTML>
```

The output of the script is as follows:



\$cookieval stores the cookie value.

The `isset()` function checks whether the cookie is set

- ◆ Cookies can be deleted automatically or manually
- ◆ There are two ways to delete a cookie, which are as follows:
 - ❖ Resetting the expiry time of the cookie to a time in the past
 - ❖ Resetting a cookie by specifying the name of the cookie
- ◆ When you create a cookie that has the same name and time as an existing cookie, the existing cookie is deleted from the hard drive of the client
- ◆ To delete a cookie with a date in the past, enter the code as shown in the Snippet in a PHP script

Snippet

```
setcookie("$cookie_name", "", time()-8000);
```

- ◆ In the code, `$cookie_name` refers to the name of the cookie. The value of the cookie is not specified and the `time()` function accepts the expiration date in the past

- ◆ This process is called as deconstructing the variable
- ◆ Use the following syntax to delete a cookie through deconstruction:

Snippet

```
setcookie($cookie_name);
```

- ◆ To delete the cookie named **uname**

Snippet

```
setcookie($uname);
```

Problems with Cookies

- ◆ Web sites store user-related information on the client system
- ◆ Cookies are not secure and reliable because the user-related information can be accessed by anyone who has full access to the client system
- ◆ Following are some of the drawbacks of cookies:
 - ❖ Cookies cannot contain more than a certain amount of information
 - ❖ Only a maximum of 29 cookies of a domain can be maintained
 - ❖ A browser can maintain maximum of 300 cookies
 - ❖ Storing large number of cookie files slows down the system
 - ❖ Some users disable cookies while accessing Web sites as a result Web sites that depend on cookies lose information of such users

- ❖ There can be multiple users using the same system visiting the same Web site
- ❖ Web sites assign cookies to the system and not to the user. This can hamper the number of visitor's statistics
- ❖ A cookie can contain large amount of information and retrieving larger amount of information on each page requires repetitive coding across the pages

Summary

- ◆ Web sites use cookies, stored on the hard disk of the client system, to store user-specific information
- ◆ Dynamic Web pages gets information from the user and records it for further processing
- ◆ Persistent cookies are stored in the Web browser for a period specified during the time of its creation and non-persistent cookies are deleted from the Web browser as soon as the user exits the browser
- ◆ The HTTP header, transmitted between the Web server and the Web browser, contains cookies
- ◆ A cookie can be retrieved by passing a variable as a cookie name and using the `$_COOKIE[]` variable

- ◆ PHP uses the setcookie() and setrawcookie() functions to set a cookie
- ◆ The two ways to delete a cookie are resetting the expiry time of the cookie to a time in the past and by resetting the cookie by specifying the name of the cookie
- ◆ The maximum number of cookies that can be maintained for a domain is 20. A browser can maintain maximum of 300 cookies

Session Management in PHP

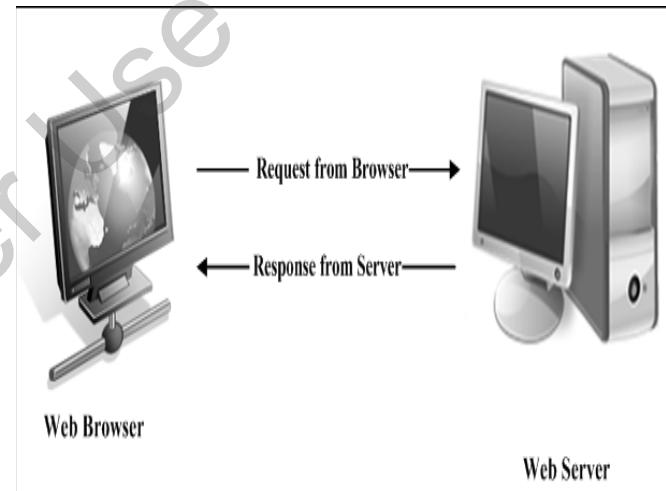
Session 21



- ◆ *Define a session*
- ◆ *Explain the procedure to work with a session*
- ◆ *Describe the methods to start a session*
- ◆ *Explain the method to register a session*
- ◆ *Describe the method to end a session*
- ◆ *Explain the use of the php.ini file*

- ◆ Sessions are similar to cookies and enable the functionality of storing temporary user information
- ◆ The difference between cookies and sessions is as follows:
 - ❖ Pertinent cookies store information on the local computer
 - ❖ Session enables PHP to store information on the Web server

- ◆ Web browsers and Web servers have a stateless interaction and do not maintain track of user sessions
- ◆ HTTP protocol
 - ❖ Enables Web browsers to communicate with Web servers
 - ❖ Has no methods or functions to maintain the status of a particular user



- ◆ Web sites that cannot depend on HTTP or Web servers for complex user interaction need session tracking
- ◆ Refers to the total time the user accesses information on a particular Web site before exiting the Web site
- ◆ Manages data for a particular user in a specific session
- ◆ Enable distinguishing user specific information for the entire duration of the session

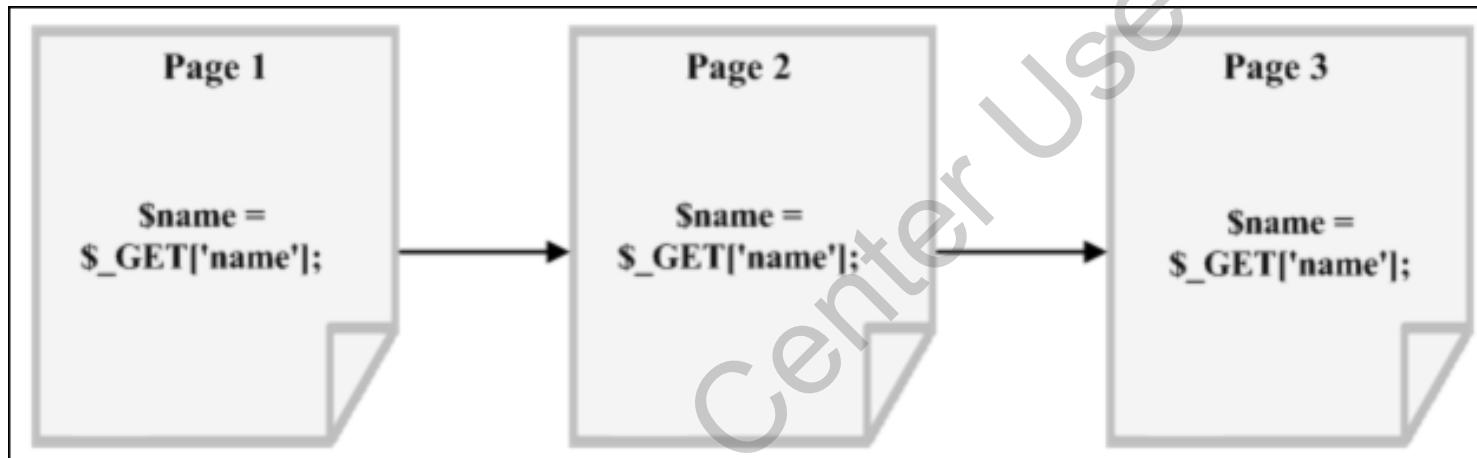
Importance of a Session

- ◆ Consider an example, in a particular Web site, the user has to first register and then log on to access any information
- ◆ For such authentication procedures, the state of the user has to be maintained across the Web site
- ◆ Web sites traditionally use GET and POST methods to pass user information from one script to another
- ◆ When these methods are used, PHP assigns user information variables in the following format:

```
$name = $_GET['name'];
```

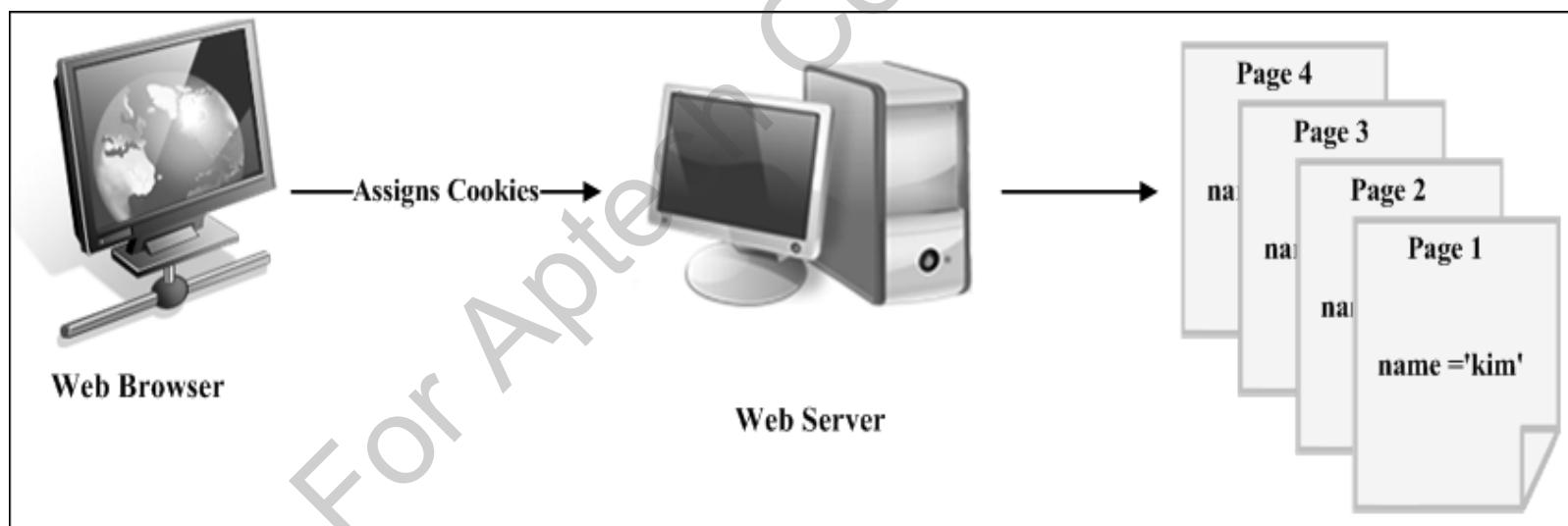
- ◆ In the code, the variable \$name stores the value that the script retrieves from the HTML form
- ◆ This process of transferring user information is time consuming and not essential for large Web sites

- The code to be used across all the Web pages of the Web site is as follows:



- ◆ Consider a scenario, where it is required to store and retrieve information for 20 or more users across 10 different pages
- ◆ Due to this disadvantage of using the GET and POST methods, Web developers prefer using cookies

Figure displays the assignment of cookies by the Web server to the browser



- ◆ Cookies enable to store data into a variable and access it across all the pages of the Web site
- ◆ Cookies are prone to security risks because the user information is saved at the client-end
- ◆ The risks involved are greater when users access Web sites from a public computer or a shared computer

- ❖ **Size of the cookie:**
 - ❖ The amount of information stored in the cookie determines the size of the cookie
 - ❖ The size of the cookie determines the size of the Web page and increase in file size of the Web page results in poor performance
- ❖ **Cookies disabled:**
 - ❖ Web sites store cookies on the hard disk of the client as a result the performance of computers reduces
 - ❖ To improve the performance of such computers, users disable cookies making the assignment of cookies pointless

- ◆ Sessions play an important role in such situations
- ◆ Sessions eliminate deletion and assignment of new cookies to the same user
- ◆ The size of a cookie does not affect the performance of a Web site
- ◆ Both the Web server and Web browser benefit because statistical information in the server database is accurate

Table explains the difference between cookies and sessions

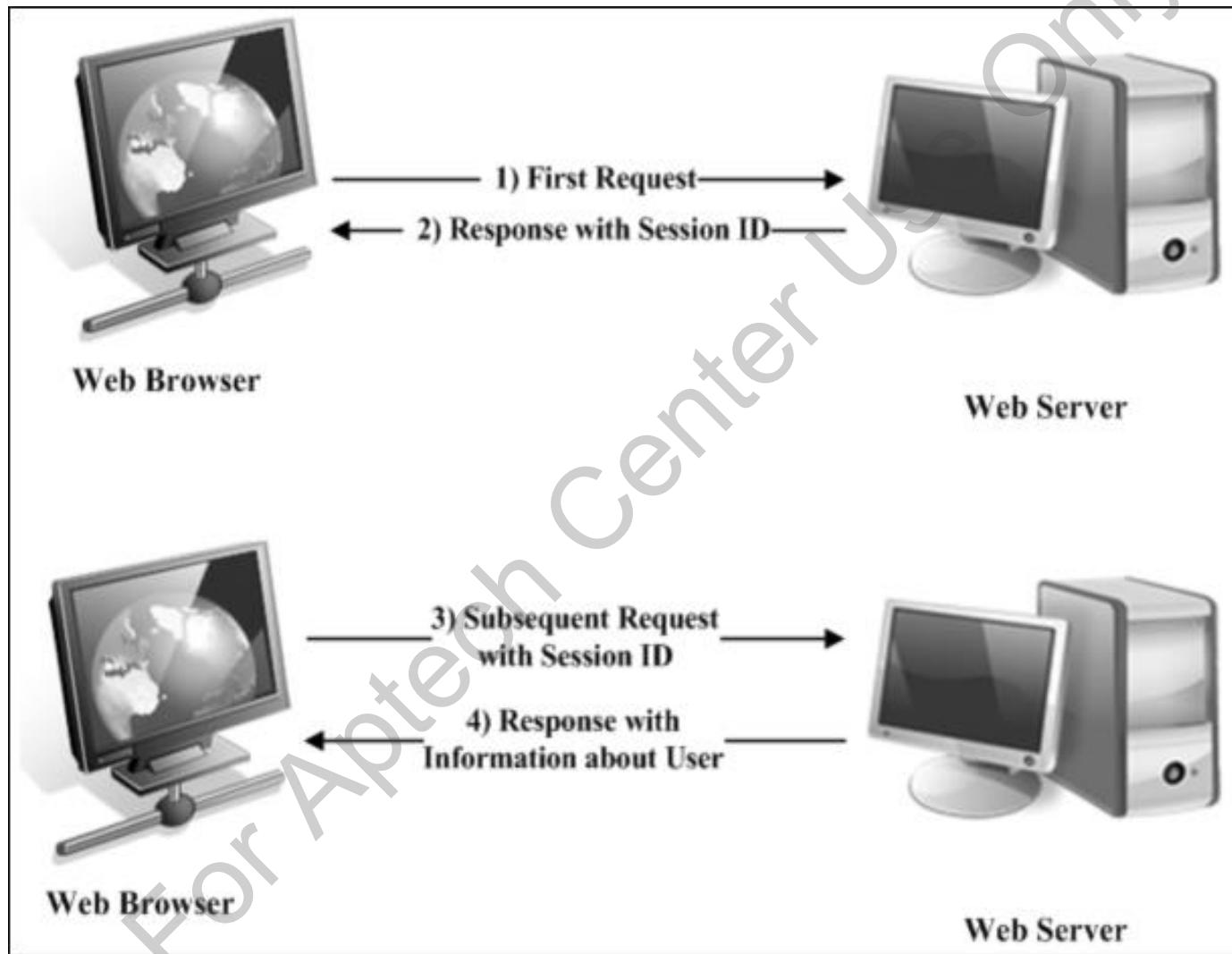
Cookies	Sessions
Stores user information on the client system (Web Browser)	Stores user information on the Web server
Available even after the user exits the Web browser	Destroyed when the user exits the Web browser
Users can disable cookies	Users cannot disable sessions
Have size limits	Do not have size limits

Working with Sessions

- ◆ Session commences when a user accesses a session-enabled Web site
- ◆ Web server assigns a unique session ID to each user when the user starts a session
- ◆ The scripts store and access user information through the session ID depending on the following two situations:
 - ❖ **Cookies enabled:**
 - ❖ Web server allots a session ID to the Web browser through a cookie, using the `setcookie()` function
 - ❖ Cookies enable transfer of user information between the browser and the server
 - ❖ PHP stores session IDs in cookies

◆ Cookies disabled:

- ◆ Web server allots a session ID to the browser using the Uniform Resource Locator (URL)
- ◆ The URL transfers user information from the browser to the server
- ◆ PHP stores the session variables in a file and names the file based on the session ID
- ◆ While using a session, PHP stores all the user information in a file on the Web server
- ◆ The file includes a session ID that is related to the user's session variable
- ◆ Each session ID identifies a different user and relates to a file that belongs to that user
- ◆ PHP destroys the session file once the user exits the Web site



- ◆ PHP works with sessions in the following sequence:
 - ❖ User accesses a session-enabled Web site which checks the user identity
 - ❖ If the user is a new visitor, the Web site allocates a unique session ID to the user. The Web site saves a cookie containing the session ID on the Web browser
 - ❖ The Web browser records the cookie that holds the session ID. The browser uses the same cookie to retrieve the session ID and record all the session-related information
 - ❖ The session file is destroyed from the Web server, when the user exists from the Web site

- ◆ There are three stages in the life cycle of a session based on the communication between the Web browser and the Web server
- ◆ They are as follows:
 - ◆ Starting the session
 - ◆ Registering the session variable
 - ◆ Ending the session

Starting the Session

- ◆ A session starts when a user logs on to the Web site
- ◆ The `session_start()` function enables to start a session
- ◆ The process of starting a session is also called as initializing a session
- ◆ The session file is created in the `/tmp` directory
- ◆ PHP assigns a name to this file based on the unique session identifier value generated by the PHP engine
- ◆ The session identifier is also known as the session ID
- ◆ The session ID is a hexadecimal string of 32 digits

- ◆ The file naming convention for the session file is as follows:
`sess_<32_digit_hexadecimal_value>`
- ◆ The session file name is always preceded by `sess_` and is followed by a random 32 digit hexadecimal value
- ◆ The Web server passes the session ID as a response to the browser
- ◆ The response sets up a session cookie in the browser with the name `PHPSESSID` and the value of the identifier
- ◆ The `session_start()` function must be specified on the top of every Web page or before the start of the actual coding
- ◆ If the session is valid and existing, it activates the frozen variables of the session
- ◆ If the session is invalid or non existing, it creates a session ID for the new session

- The `session_start()` function is as follows:

Syntax

```
session_start();
```

- To start a session, perform the following steps:

- Open a new file in **gedit** text editor

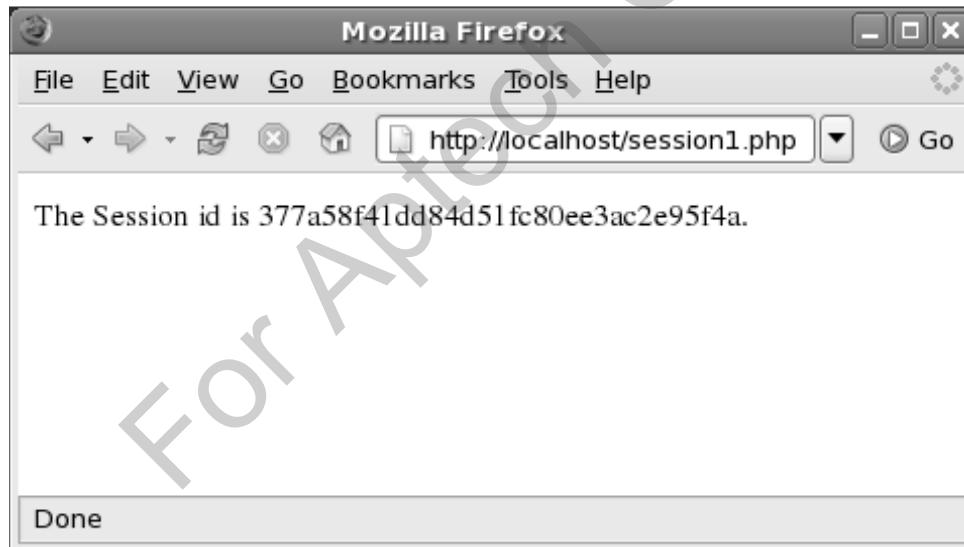
- Enter the following code:

Snippet

```
<?php  
// initializing a session  
session_start();  
/* The session_id() function displays the session id  
that PHP allots to  
a user. */  
echo "The Session id is " .session_id(). "<br>";  
?>
```

3. Save the script as session1.php in the /usr/local/apache2/htdocs/ directory.
4. Open the Mozilla Firefox Web browser.
5. Enter `http://localhost/session1.php` in the Address bar and press Enter.

The following output is displayed:



- ◆ Variables in a session file contain user specific information
- ◆ Session library enables creation, serialization, and storage of session data
- ◆ There are three methods to set a session variable, which are as follows:
 - ❖ `$_SESSION[]` - recommended for PHP 4.1.0
 - ❖ `$HTTP_SESSION_VARS[]` - recommended for PHP 4.0.6 or less
 - ❖ `session_register()` - not recommended as it has been deprecated
- ◆ Session variables can be of any data type such as integer, string, Boolean, or object
- ◆ PHP stores the session variables in a session file by serializing the values
- ◆ PHP automatically handles the process of serializing the session variables

- ◆ Steps to register the value of a session variable are as follows:
 1. Open a new file in **gedit** text editor
 2. Enter the following code:

Snippet

```
<?php  
session_start();  
  
$_SESSION[ 'myname' ]= "Jessica";  
  
?>  
  
<HTML>  
  <HEAD> <TITLE> Session </TITLE></HEAD>  
  
  <BODY>  
    <A HREF = "mypage.php"> Homepage of MyPage.com </A>  
  </BODY>  
</HTML>
```

Registering the Session Variable

3. Save the file as **sessionstart.php** in the
`/usr/local/apache2/htdocs/` directory
- ◆ To display the value of the session variable, perform the following steps:
 1. Open a new file in **gedit** text editor.
 2. Enter the following code:

Snippet

```
<?php

session_start();

$myname = $_SESSION['myname'];

?>

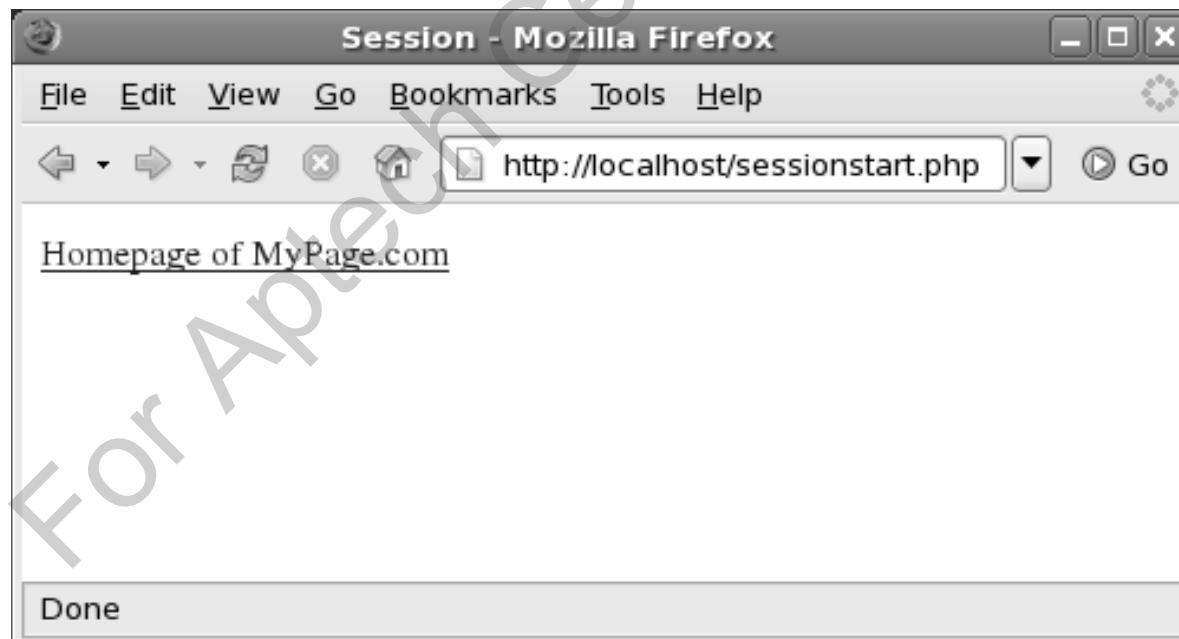
<HTML>
<HEAD> <TITLE> Homepage </TITLE></HEAD>
<BODY>

Welcome <?php echo $myname ?> to MyPage.com <br>

</BODY>
</HTML>
```

3. Save the file as mypage.php in the /usr/local/apache2/htdocs/directory
4. Open the Mozilla Firefox Web browser
5. Enter `http://localhost/sessionstart.php` in the Address bar and press **Enter**

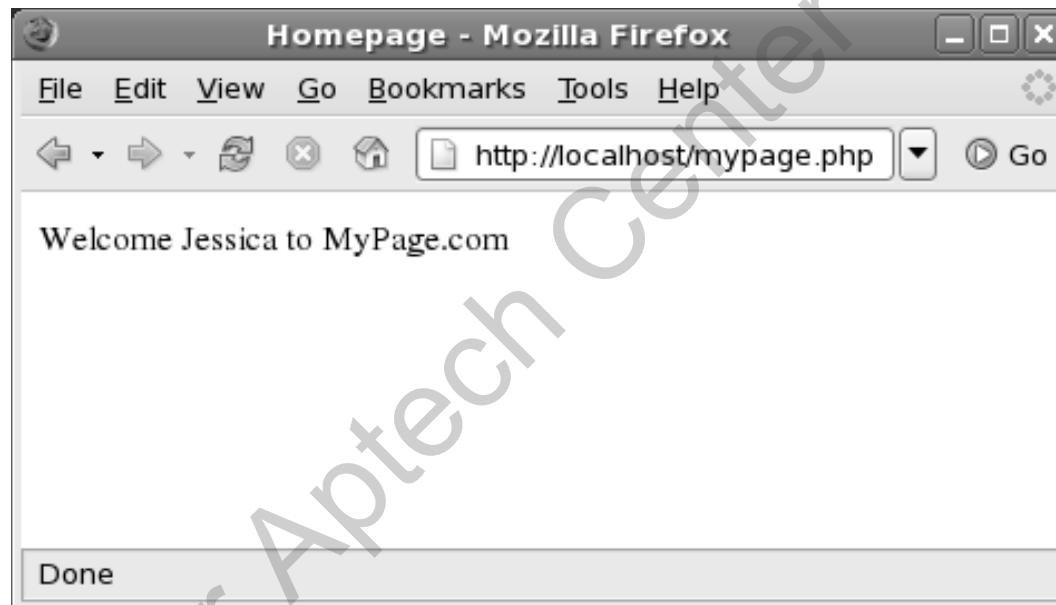
The following output is displayed:



6. Click Homepage of MyPage . com

The Homepage page appears with the message,

Welcome Jessica to MyPage.com



- When the user logs out of the Web site, the PHP script executes the `session_destroy()` function
- When the session file is deleted, the `$PHPSESSID` cookie is not removed from the Web browser
- The syntax for the `session_destroy()` function is as follows:

Syntax

```
session_destroy();
```

- ◆ PHP uses the following configuration directives when a session ends:
 - ◆ `gc_maxlifetime()`:
 - ◆ Enables PHP to determine the time to wait before ending a session and the process of cleaning up is called as garbage collection
 - ◆ `gc_probability()`:
 - ◆ Enables PHP to determine with what probability the garbage collection routine must be invoked

- ◆ To destroy a session, perform the following steps:
 1. Open a new file in **gedit** text editor
 2. Enter the following code:

Snippet

```
<?php

session_start();

$myname = $_SESSION['myname'];

// The session_unset() function unregisters a session
variable.

session_unset();

session_destroy();

echo "Session destroyed!";
```

```
?>

<HTML>

<HEAD> <TITLE> Session </TITLE></HEAD>

<BODY>

<br>

<A HREF = "mypage.php"> Homepage of MyPage.com </A>

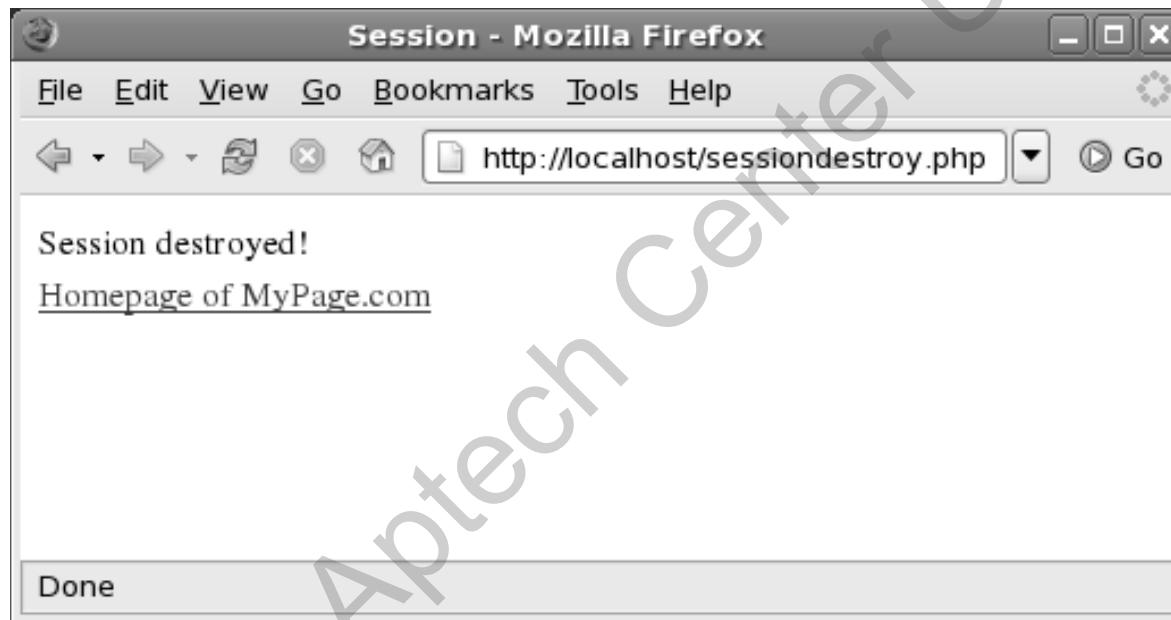
</BODY>

</HTML>
```

3. Save the file as **sessiondestroy.php** in the
/usr/local/apache2/htdocs/directory
4. Open the Mozilla Firefox Web browser

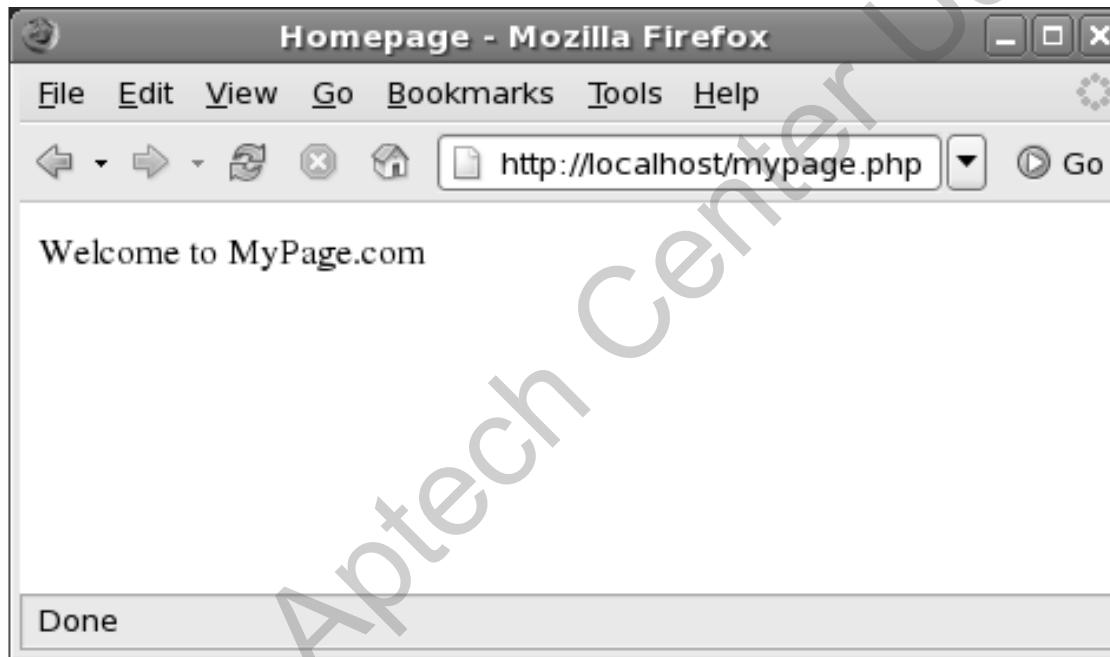
- Enter `http://localhost/sessiondestroy.php` in the Address bar and press **Enter**

The following output is displayed:



- To view the execution of the `session_destroy()` function, click the Homepage of MyPage.com hyperlink

The following output is displayed:

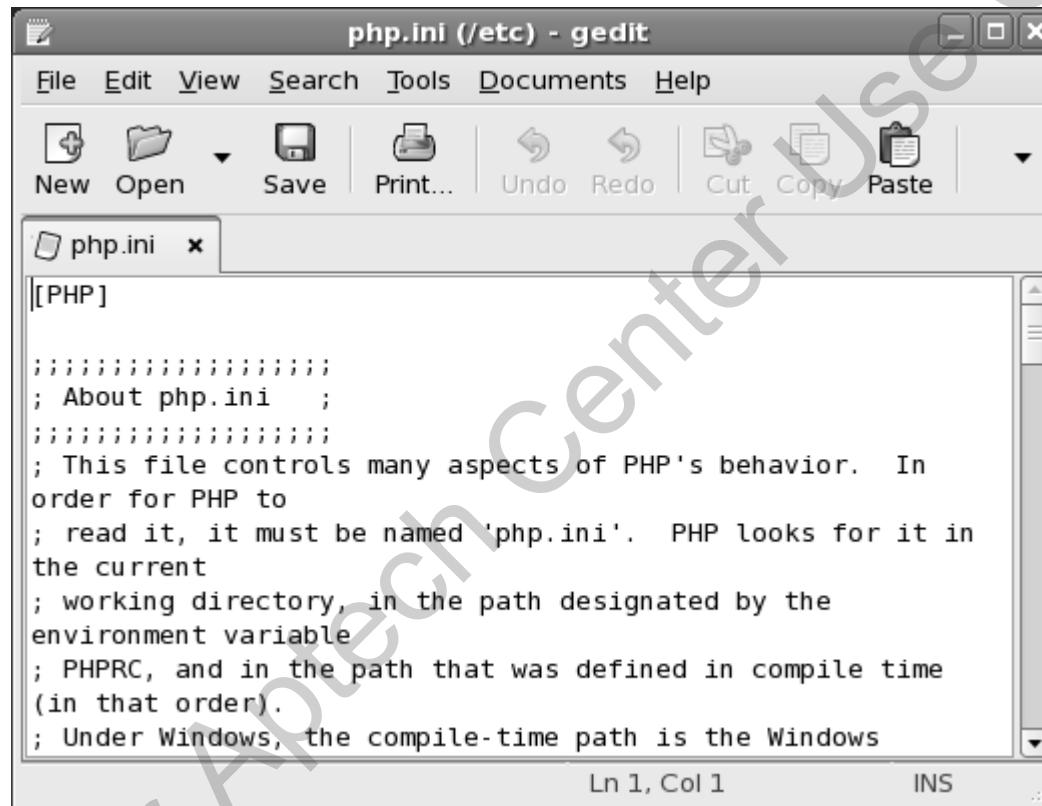


The name **Jessica** does not appear in the message because the variable has been cleared.

Working with the php.ini File

- ◆ A Web server can contain multiple `php.ini` files
- ◆ Create a `php.ini` file if it does not exist on the Web server
- ◆ The complete source code must be downloaded to create a new `php.ini` file
- ◆ PHP interpreter works according to the instructions included in the `php.ini` file
- ◆ The Web server searches sequentially for the `php.ini` file in the following locations:
 - ◆ Directory where the PHP script was called
 - ◆ Root of the Web directory
 - ◆ Directory containing the `default.ini` file on the Web server

- ◆ Figure displays the PHP configuration file



The screenshot shows the gedit text editor window titled "php.ini (/etc) - gedit". The menu bar includes File, Edit, View, Search, Tools, Documents, and Help. The toolbar contains icons for New, Open, Save, Print..., Undo, Redo, Cut, Copy, and Paste. The main text area displays the contents of the php.ini file:

```
[PHP]

; About php.ini ;
; This file controls many aspects of PHP's behavior. In
order for PHP to
; read it, it must be named 'php.ini'. PHP looks for it in
the current
; working directory, in the path designated by the
environment variable
; PHPRC, and in the path that was defined in compile time
(in that order).
; Under Windows, the compile-time path is the Windows
```

At the bottom of the editor, status bars show "Ln 1, Col 1" and "INS".

The **php.ini** file contains directive listed in the directive
= value format.

You can use a semicolon to add a comment to the file.

Table lists the categories in the php.ini file

Categories	Options
Language Options	Enable PHP scripting language engine under Apache
	Allow ASP style tags
	Enforce year 2000 compliance
Safe Mode	Perform a UID compare check when opening files
	Allow executables under specific directories to be executed via exec family
	Allow user set environment variables that begin with PHP prefix
Font Colors	Indicate the colors that PHP uses for highlighting syntax
Misc	Indicate whether or not PHP discloses the fact that it is installed on the server
Resource Limits	Indicate the maximum time for script execution
	Indicate the maximum time for parsing request data
	Indicate the maximum amount of memory a script requires
Data Handling	Control list of separators used in PHP generated URLs to separate arguments
	Describe the order in which PHP registers Get, Post, Cookie, Environment and built-in variables
Path and Directories	Specifies the name of the directory under which PHP opens the script
	Specifies the name of the directory under which the loadable extensions exist

Categories	Options
Error handling and logging	Report all errors and warnings
	Report fatal compile time errors
	Report fatal run-time errors
	Report non-fatal error
	Report fatal errors that occur during initial startup of PHP
	Report user generated errors, warnings, and messages
Magic Quotes	Set magic quotes for incoming Get, Post, Cookie data
	Use Sybase style magic quotes
	Automatically adds file before or after any PHP document
File Uploads	Indicate whether or not to allow HTTP file uploads
	Indicate temporary directory for HTTP uploaded files
	Indicate the maximum allowed size for upload files
Session	Store and retrieve data
	Indicate whether or not cookies should be used
	Initializes session on request startup
	Serializes data

- ◆ Session category of the `php.ini` file include options, which are as follows:
 - ◆ `session.save_handler` - specifies how PHP stores and retrieves session variables
 - ◆ Either of the following values can be used for this option:
 - ◆ `files`: indicates the use of the session files
 - ◆ `mm`: stores and retrieves data from a shared memory
 - ◆ `user`: stores and retrieves variables with custom defined handlers

- ◆ `session.save_path` - specifies the name of the directory where the session files will be stored
- ◆ `session.use_cookies` - indicates whether PHP must send a session ID to the Web browser through a cookie. The value to enable a cookie to store a session ID is 1
- ◆ `session.use_only_cookies` - indicates whether the modules can use only cookies for storing session IDs. By default, this option is disabled
- ◆ `session.cookie_lifetime` - specifies the lifetime of the cookie and the value is specified in seconds

- ◆ `session.name` - manages the cookie name and form attributes such as GET and POST that holds the session ID. By default, the value of this option is PHPSESSID
- ◆ `session.auto_start` - enables sessions to automatically initialize if the session ID is not found in the browser request
- ◆ `session.cookie_secure` - specifies whether the cookies must be sent over secured connections. By default, the cookies are not sent through secured connections

- ◆ Other settings can also be modified in the `php.ini` file, such as:

- ◆ `register_globals` - controls the functioning of server, forms, and environment variables

If this option is disabled, variables can be retrieved using the GET or POST methods as follows:

```
$_POST['$variable_name'];
$_GET['$variable_name'];
```

If `register_globals` is enabled, the variable can be directly accessed using the variable name as follows:

```
$storeValue = $variable_name;
```

`$variable_name` is the name of the variable that contains the session data of another Web page

The variable `$storeValue` stores the information included in the variable `variable_name`

- ❖ `upload_tmp_dir` - sets the location of the temporary file that is uploaded with the HTML form
- ❖ `display_errors` and `display_startup_errors` - enables PHP to display errors on the Web browser
- ❖ `log_errors` and `error_log` - enables PHP to display error logs

Summary

- ◆ Cookies provide the functionality of storing temporary user information on the local computer
- ◆ Sessions enable PHP to store user information on the Web server
- ◆ HTTP is considered as a stateless protocol that enables Web browsers to communicate with the Web servers
- ◆ Session refers to the total time a user accesses information on a particular Web site before exiting the Web site
- ◆ A PHP script can access the session ID when the Web user enables or disables cookies
- ◆ The three stages in the life cycle of a session are: starting a session, registering a session variable, and ending a session

Handling E-mail with PHP

Session 22



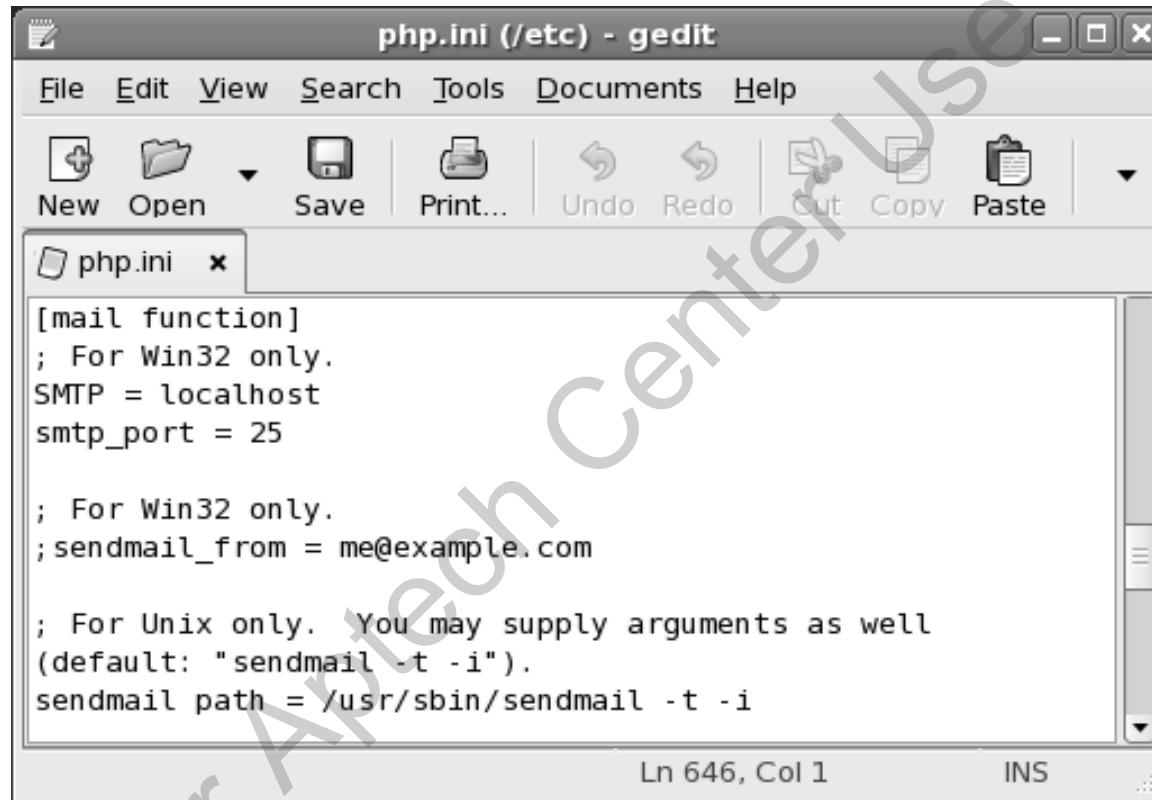
- ◆ *Describe the process of sending an e-mail using PHP*
- ◆ *Explain the process of attaching files with an e-mail using PHP*

For Aptech Center Use Only

- ◆ E-mail are:
 - ❖ Fastest way of communicating with people across the world
 - ❖ It takes only few minutes to send and receive messages as compared to the usual hand written letters
- ◆ PHP provides the facility to send an e-mail

- ◆ In PHP, you can send an e-mail using the `mail()` function
- ◆ Location of the current local mail server must be specified in the `php.ini` configuration file to use the `mail()` function
- ◆ The `php.ini` file is located in the `/etc/` directory of the Linux operating system

The contents of the `php.ini` configuration are displayed as follows:



The screenshot shows the `php.ini (/etc) - gedit` window. The menu bar includes File, Edit, View, Search, Tools, Documents, and Help. The toolbar contains icons for New, Open, Save, Print..., Undo, Redo, Cut, Copy, and Paste. A single tab labeled "php.ini" is open. The code content is as follows:

```
[mail function]
; For Win32 only.
SMTP = localhost
smtp_port = 25

; For Win32 only.
;sendmail_from = me@example.com

; For Unix only. You may supply arguments as well
(default: "sendmail -t -i").
sendmail path = /usr/sbin/sendmail -t -i
```

The status bar at the bottom shows "Ln 646, Col 1" and "INS".

- ◆ The `mail()` function is as follows:

Syntax

```
mail(to, subject, message , [additional headers])
```

Where,

to - specifies the recipient's e-mail address

subject - specifies the subject for the e-mail

message - specifies the message that is to be written in the e-mail or the body of the e-mail

additional headers - specifies additional information such as the e-mail address of the sender and attachments

- ◆ **mail.php:** Sending an e-mail using the mail() function

Snippet

```
<HTML>

<BODY>

<?php

$to = "recipient@example.com";

$from = "yourname@example.com";

$subject = "Test e-mail";

$body = "This is an example for showing the usage of
the mail() function.";

$send = mail($to, $subject, $body, $from);

if($send)

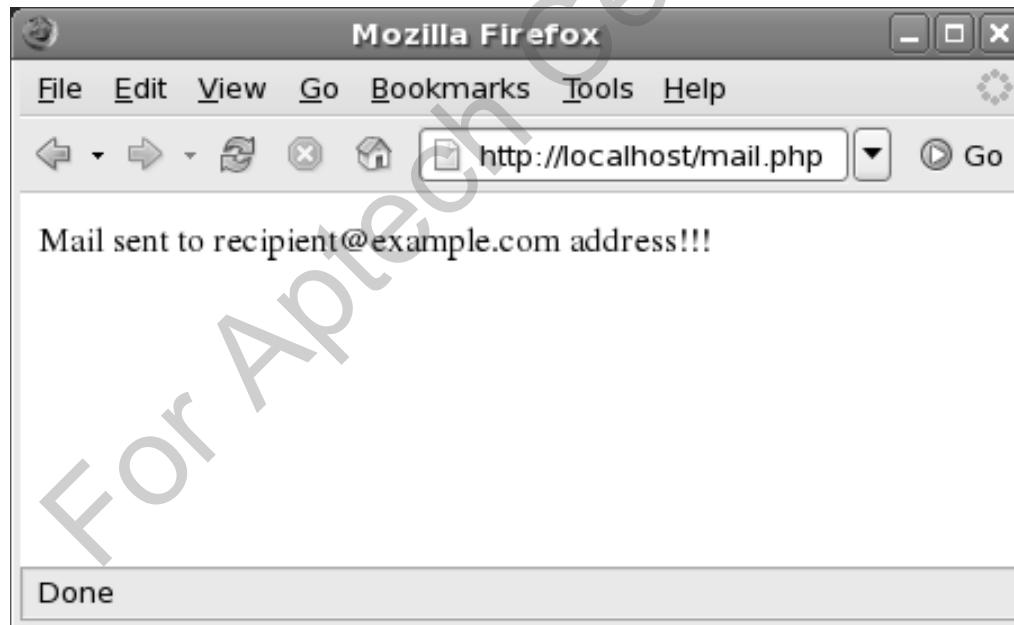
{

    echo "Mail sent to $to address!!!!";

}
```

```
else
{
    echo "Mail could not be sent to $to address!!!";
}
?>
</BODY>
</HTML>
```

The following output is displayed:



Sending an E-mail

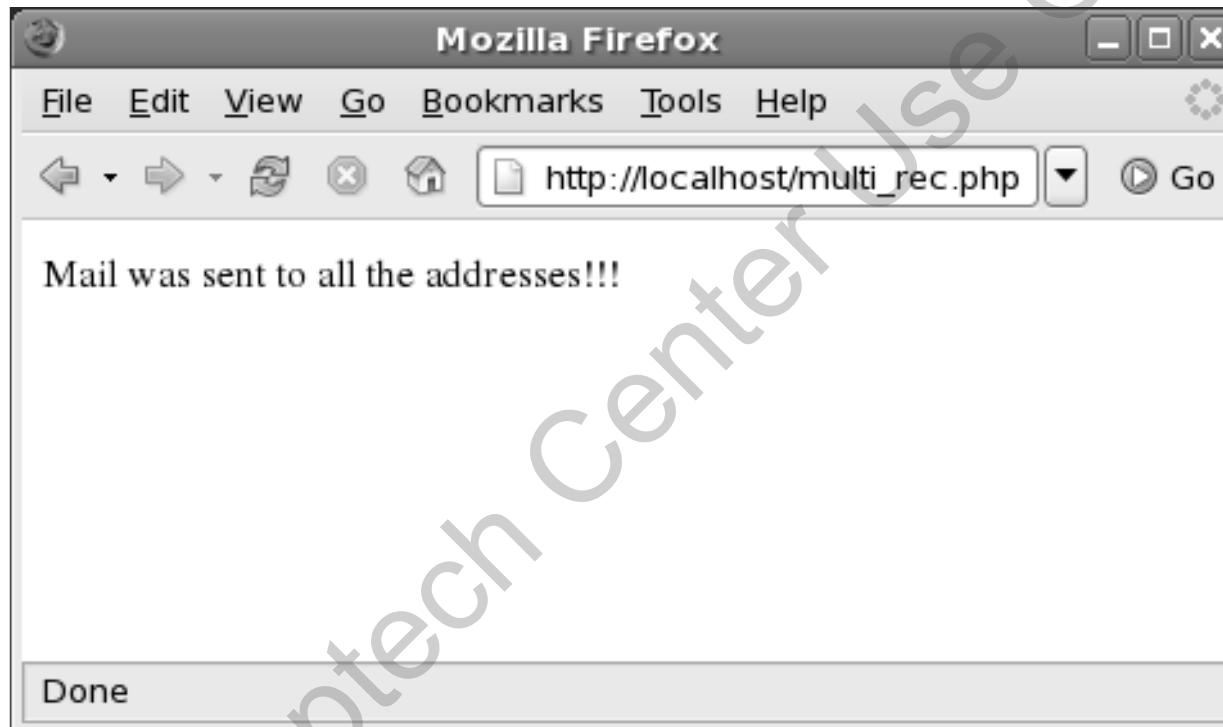
- ◆ The `mail()` function is also used for sending mails to more than one recipient
- ◆ A comma (,) symbol is used for separating the e-mail addresses of the recipients
- ◆ **multi_rec.php** - Sending an e-mail to multiple recipients

Snippet

```
<HTML>
<BODY>
<?php
$to = "recipient1@example.com, recipient2@example.com,
recipient3@example.com";
$from = "yourname@example.com";
for($i=0;$i<count($to);$i++)
{
    $to[$i] = trim($to[$i]);
```

```
$subject = "An example";  
  
$body = "This is an example for showing the usage of  
the mail() function.";  
  
$send = mail($to, $subject, $body, $from);  
  
if($send)  
{  
  
echo "Mail was sent to all the addresses!!!!";  
  
}  
}  
?  
</BODY>  
</HTML>
```

The following output is displayed:



The `mail()` function is used to send an e-mail.
The `trim()` function is used to remove blank spaces.

- ◆ PHP also enables to read the e-mail addresses of the recipients from the text file and send an e-mail to them
- ◆ Steps for sending e-mail to multiple recipients whose addresses are stored in a text file are as follows:
 1. Open a new file in the **gedit** text editor
 2. Enter the following text:

recipient1@example.com

recipient2@example.com

recipient2@example.com

3. Save the file as **email_list.txt** in the
`/usr/local/apache2/htdocs` directory
4. Assign read, write, and execute permissions to the
email_list.txt file
5. Open a new file in the **gedit** text editor

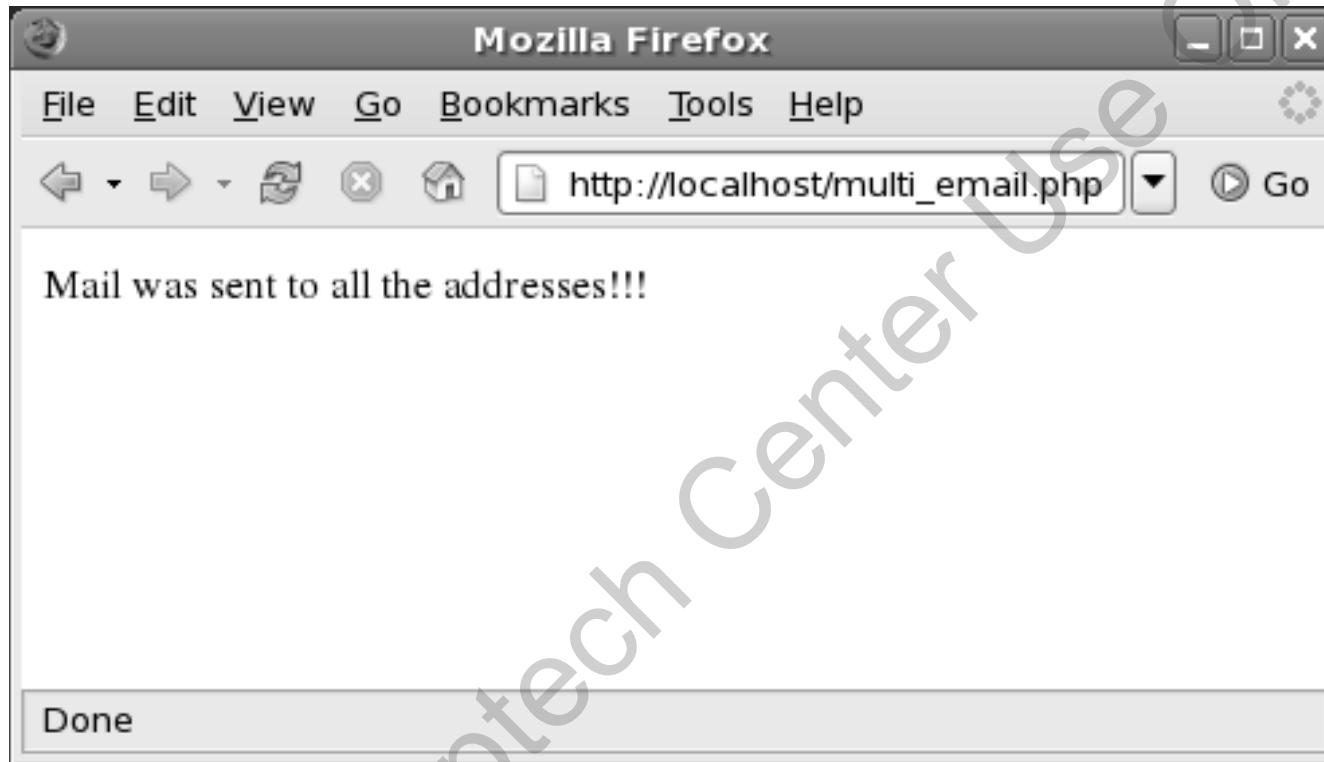
6. Enter the code as shown in the Snippet:

Snippet

```
<HTML>
<BODY>
<?php
error_reporting(0);
$multi = "/usr/local/apache2/htdocs/email_list.txt";
$to_mail = file('$email_list.txt');
$from = "yourname@example.com";
for($i=0;$i<count($to_mail);$i++)
{
$to_mail[$i] = trim($to_mail[$i]);
$to = implode(",",$to_mail);
$subject = "An example";
$body = "This is an example for the mail() function.";
mail($to, $subject, $body, $from);
echo "Mail was sent to all the addresses!!!";
}
```

7. Save the file as **multi_email.php** in the
`/usr/local/apache2/htdocs` directory
8. Open the Mozilla Firefox Web browser
9. Enter **http://localhost/multi_email.php** in
the Address bar and press **Enter**

The following output is displayed:



The functions, such as `file()`, `trim()`, and `implode()` are used.

- ◆ An attachment is
 - ❖ Any file that you want to send along with the e-mail
 - ❖ Attached with the e-mail by setting up the header

For Aptech Center Use Only

- ◆ To send an e-mail with an attachment, perform the following steps:
 1. Open a new file in the **gedit** text editor
 2. Enter the following text:

This is a sample attachment file
 3. Save the file as **example.txt** in the /usr/local/apache2/htdocs directory
 4. Assign read, write, and execute permissions to the **example.txt** file
 5. Open a new file in the **gedit** text editor
 6. Enter the code as shown in the Snippet

Snippet

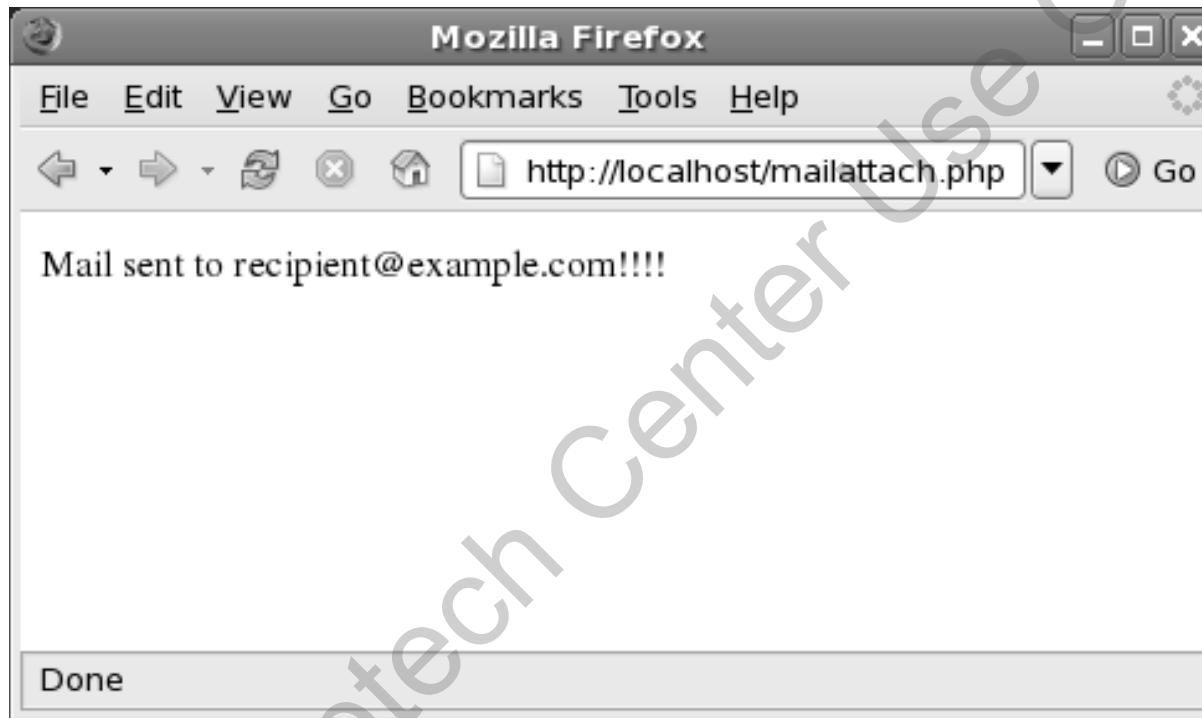
```
<?php  
  
// filenames to be sent as attachment  
  
$files = "/usr/local/apache2/htdocs/example.txt";  
  
// e-mail fields: to, from, subject, and so on  
  
$to = "recipient@example.com";  
  
$from = "yourname@example.com";  
  
$subject ="Test e-mail";  
  
$message = "Sample message";  
  
$headers = "From: $from";  
  
// boundary  
  
$semi_rand = md5(time());  
  
$mime_boundary = "==Multipart_Boundary_x{$semi_rand}x";
```

```
// headers for attachment  
  
$headers .= "\nMIME-Version: 1.0\n" ."Content-Type: multipart/mixed;  
\n" . " boundary=\"$mime_boundary\"\n";  
  
// multipart boundary  
  
$message = "This is an example for mail attachment in MIME format.  
\n\n" . "--{$mime_boundary}\n" .  
"Content-Type: text/plain; charset=\"iso-9959-1\"\n" .  
$message . // "\n\n";  
  
$message .= "--{$mime_boundary}\n";  
  
// preparing attachments  
  
$file = fopen($files, "rb");  
  
$content = fread($file, filesize($files));  
fclose($file);  
  
$message .= "Content-Type: {\"application/octet-stream\"};\n" . "  
name=\"$files\"\n" . "Content-Disposition: attachment;\n"  
. " filename=\"$files\"\n" . $content . "\n\n";  
  
$message .= "--{$mime_boundary}\n";
```

```
// send  
$send_mail = @mail($to, $subject, $message,  
$headers);  
  
if ($send_mail)  
{  
echo "<p>Mail sent to $to!!!!</p>";  
}  
else  
{  
echo "<p>Mail could not be sent to $to!!!!</p>";  
}  
?>
```

7. Save the file as **mailattach.php** in the
`/usr/local/apache2/htdocs` directory
8. Open the Mozilla Firefox Web browser
9. Enter **http://localhost/mailattach.php** in
the Address bar and press **Enter**

The following output is displayed:



A boundary is created between the actual e-mail and the attachments.

The `fopen()` function opens the file that is attached with the e-mail in the read mode.

- ◆ The two built-in PHP function are as follows:
 - ❖ `base64_encode()` - used for encoding the specified string
 - ❖ `chunk_split()` - used for splitting the data into a series of smaller parts

Summary

- ◆ The mail() function is used for sending an e-mail
- ◆ The trim() function removes blank spaces included in between two recipient e-mail addresses
- ◆ The implode() function is used to join all the addresses present in the array with a comma operator
- ◆ An attachment can also be sent with an e-mail by setting up the header of the mail() function
- ◆ The uniqid() function assigns a unique identifier to the original mail and creates a boundary between the e-mail and the attachment

Summary

- ◆ The `base64_encode()` function is used for accepting the data in the form of a string and returns the data in the original form to the user
- ◆ The `chunk_split()` function is used for separating the data into smaller parts

OOP Concepts

Session 24



Objectives

- ◆ *Explain the OOP Concepts*
- ◆ *Explain inheritance*
- ◆ *Explain the use of classes*
- ◆ *Describe the use of constructors*
- ◆ *Explain the use of objects*

For Aptech Center Use Only

- ◆ Object-Oriented Programming (OOP) is:
 - ❖ Term used to characterize a programming language
 - ❖ Widely accepted paradigm for programming languages
- ◆ OOP uses three basic concepts are as follows:
 - ❖ Classes
 - ❖ Objects
 - ❖ Methods
- ◆ Additional concepts in object-oriented languages are as follows:
 - ❖ Inheritance
 - ❖ Abstraction
 - ❖ Polymorphism
 - ❖ Event Handling
 - ❖ Encapsulation

- ◆ Defines
 - ❖ Data type of the data structure
 - ❖ Type of functions to be performed on the data structure
- ◆ Unit of execution in an object-oriented system are objects
- ◆ Combines data and functions in a single unit

- ◆ The basic concepts of an object-oriented programming are as follows:
 - ❖ **Object:**
 - ❖ Consists of data structures and functions for manipulating the data
 - ❖ Data structure refers to the type of data while function refers to the operation applied to the data structures
 - ❖ An object is a self-contained run-time entity
 - ❖ An analysis of the programming problem is done in terms of objects and nature of communication between them

- ❖ **Class:**

- ❖ Contains a collection of similar types of objects
- ❖ Has its own properties
- ❖ Once a class is specified, a number of objects can be created that belong to this category

- ❖ **Abstraction:**

- ❖ Defines the process of selecting the common features from different functions and objects
- ❖ The functions those perform same actions can be connected into a single function using abstraction

❖ **Encapsulation:**

- ❖ Specifies the process of combining data and objects into a single unit
- ❖ The data cannot be accessed directly; it can be accessed only through the functions present inside the unit
- ❖ It enables data encryption

❖ **Polymorphism:**

- ❖ Defines the use of a single function or an operator in different ways
- ❖ The behavior of that function will depend on the type of the data used in it

❖ Inheritance:

- ❖ Specifies the process of creating a new class from the existing one
- ❖ The new class that is formed is called the derived class
- ❖ The existing class is called the base class

- ◆ Inheritance means creating a new class with the help of a base class
- ◆ A base class is called the parent class and the derived class is called the child class
- ◆ A class is an object that contains variables and functions of different data types
- ◆ The properties, such as variables, functions, and methods of the base class are transferred to the newly derived class
- ◆ The `extends` keyword needs to be used for inheriting a new class from the base class
- ◆ A derived class can also have its own variables and functions

- ◆ The different types of inheritances are as follows:

- ◆ **Single Inheritance:**

- ◆ Contains only one base class and inherits the properties of the base class
 - ◆ In single inheritance, the derived class works with properties derived from the base class along with its own properties

- ◆ **Multiple Inheritance:**

- ◆ Contains more than one base class and inherits the properties of all base classes
 - ◆ A class derived from this inheritance works with the derived properties along with its own properties

❖ Hierarchical Inheritance:

- ❖ Contains one base class and the properties of a single base class can be used multiple times in the sub-multiples
- ❖ The derived class contains its own properties and also uses the derived properties of all the base classes

❖ **Multilevel Inheritance:**

- ❖ Contains one base class and the base class of the derived class can be a class derived from another base class
- ❖ The properties of a base class are inherited to another base class and the derived class can become a base class for another derived class

❖ **Hybrid Inheritance:**

- ❖ Uses the combination of two or more inheritances
- ❖ This inheritance is normally a combination of multiple and multilevel inheritance

- ◆ A class is a collection of variables and functions that operate on data
- ◆ A class can be inherited from a base class to create a new derived class
- ◆ The new derived class uses all the properties of the base class including its own properties
- ◆ The new derived class uses all the properties of the base class including its own properties

- ◆ Declaring a class

Syntax

```
class class_name  
{  
    function_name($var1, $var2)  
    {  
        return $var1 + $var2;  
    }  
}
```

Where,

class - defines the keyword used to declare a class

class_name - specifies the class name

function - specifies the keyword to define a function

var1, var2 - specifies the variable name

function_name() - specifies the function name

return - returns the value after performing the specified mathematical function

- ◆ In the class syntax:
 - ❖ Definition for a class is included within curly braces
 - ❖ Variables defined in the class are local to the class
 - ❖ Variable inside a class is declared with the keyword var
 - ❖ Class can also use global variables
 - ❖ Functions inside a class may use its own local variables or may use the class variables

- Following is the code for a class named empdetail
Snippet

```
<?php

class empdetail  {

    var $empid;

    var $empname;

    var $empcity;

    var $empdept;

    var $empdesign;

    function enteremp($id, $name, $city)

    {

        $this->empid=$id;

        $this->empname=$name;

        $this->empcity=$city;
```

```
}

function enterdet($dept, $design)

{
    $this->empdept=dept;
    $this->empdesign=design;
}
?>
```

- ◆ This file need to be saved with an extension .inc.
- ◆ In the code, the enteremp () and enterdet () functions are user defined
- ◆ These functions are defined in the empdetail class, and are used for entering data of an employee

- ◆ The `enteremp()` function accepts the employee id, employee name, and city
- ◆ The `enterdet()` function accepts the employee department and the designation
- ◆ The `filename.inc` file is included in the `filename.php` file with the help of the `include` keyword
- ◆ The `include` keyword enables to incorporate any type of file with the main file

- The syntax to include a file in the program is as follows:

Syntax

```
include "filename.ext";
```

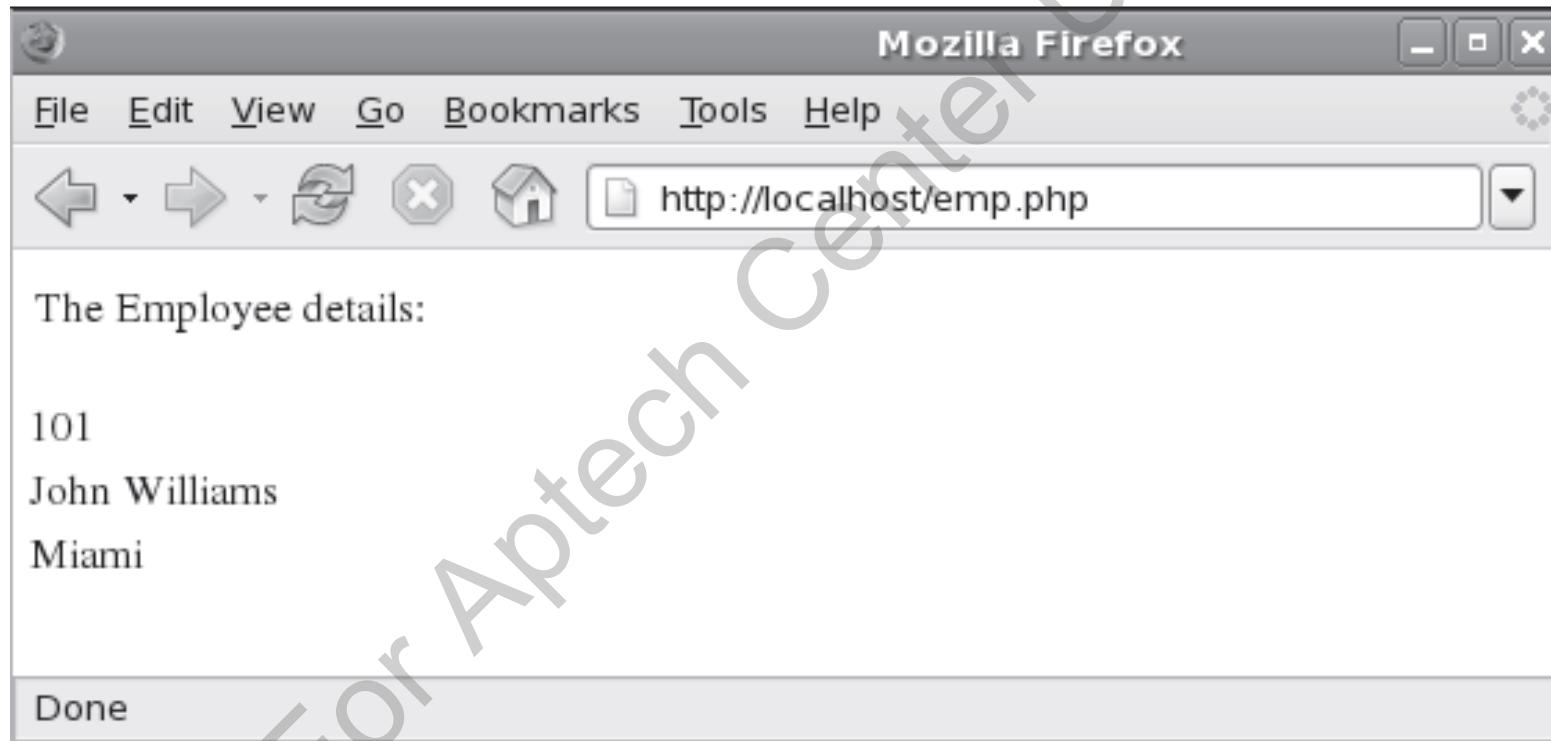
- Assuming that the code empdetail.inc is saved, it will be used in the next code

Snippet

```
<?php  
include "empdetail.inc";  
echo "The Employee details: <BR><BR>";  
$empdet = new empdetail();  
$empdet->enteremp(101, "John Williams", "Miami");  
echo $empdet->empid, "<BR>";  
echo $empdet->empname, "<BR>";  
echo $empdet->empcity;  
?>
```

- ◆ Save this file as **emp . php** and open it in Mozilla Firefox Web Browser

The following output is displayed:



- ◆ A new class is inherited from an existing class
- ◆ The new class uses the properties of the parent class along with its own properties
- ◆ The syntax for inheriting a new class is as follows:

Syntax

```
class new_class extends class_name
{
    var class_variable;
    function function_name()
    {
        . . .
    }
}
```

Where,

new_class - specifies the name of the derived class
extends - defines the keyword used to derive a new class from the base class
class_name - specifies the name of the class from which the new class is to be derived

- ◆ **salary.inc** - Deriving the `net_salary` class from the `salary` class in PHP

Snippet

```
<?php

class salary

{
    public $hra;
    public $ta;
    public $tax;

    public function hra_calc($basic)
    {
        $hra = $basic * 0.25;
        return $hra;
    }
}
```

```
public function travelallow_calc($basic)
{
    $ta = $basic * 0.08;
    return $ta;
}

public function tax_calc($basic)
{
    $tax = $basic * 0.05;
    return $tax;
}
```

```
class net_salary extends salary
{
function net($basic)
{
$hra = $this->hra_calc($basic);
$ta = $this->travelallow_calc($basic);
$tax = $this->tax_calc($basic);
return $basic + ($hra + $ta)-$tax;
}
}
?>
```

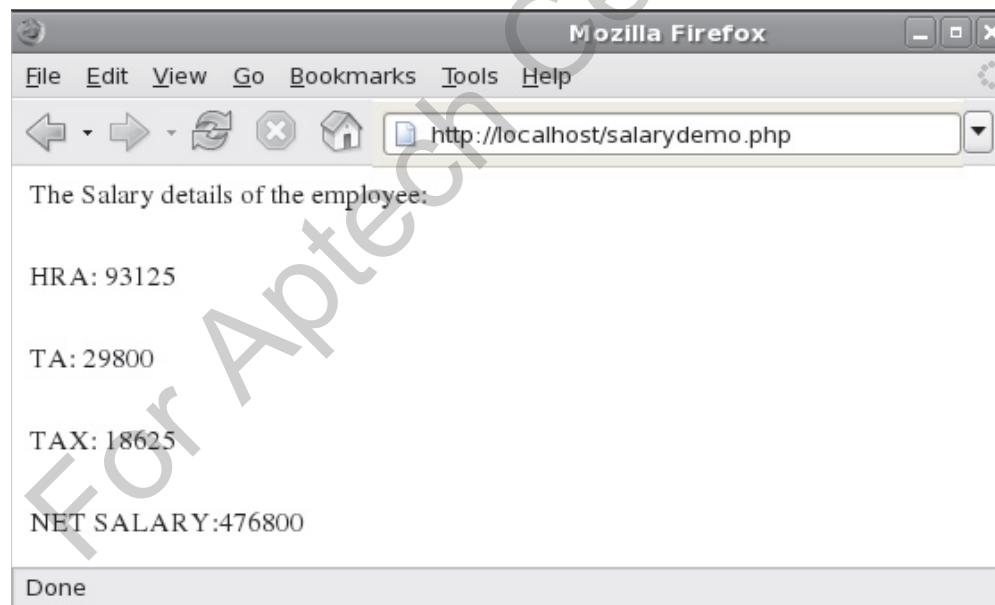
- ◆ In the code, the `net_salary` class is derived from the base class, `salary`
- ◆ The `net_salary` class uses the functions defined in the `salary` class along with its own functions
- ◆ The `hra_calc()`, `travelallow_calc()`, and `tax_calc()` functions are the user-defined functions
- ◆ The `hra_calc()` function calculates HRA from the basic salary of the employee
- ◆ The `travelallow_calc()` function calculates traveling allowance from the basic salary of the employee
- ◆ The `tax_calc()` function calculates the tax from the basic salary of the employee

- ◆ This class is used in a file named **salarydemo.php**

Snippet

```
<?php  
include "salary.inc";  
echo "The Salary details of the employee: <BR><BR>";  
$sal = new net_salary();  
echo $sal->net(372500);  
?>
```

The following output is displayed:



◆ Constructor

- ❖ Is a special function that is a member of the class
- ❖ Is called in the main program by using the new operator
- ❖ Is invoked whenever an object of the class is created
- ❖ Is declared, it provides a value to all the objects that are created in the class and this process is known as initialization
- ❖ Has the same name as that of its class name

Syntax

```
class class_name  
{  
    var class_variable;  
    Function constructor_name()  
}
```

Where,

constructor_name - specifies the name of the constructor

- ◆ **emp_constructor.inc** - Creating a constructor for the class named empdetail

Snippet

```
class empdetail  
{  
    var $empid;  
    var $empname;  
    var $empcity;  
    var $empdept;  
    function empdetail($id, $name, $city, $dept)  
    {  
        $this->empid=$id;
```

Creating a Constructor

3-8

```
$this->empname=$name;  
$this->empcity=$city;  
$this->empdept=$dept;  
}  
}  
?>
```

For Aptech Center Use Only

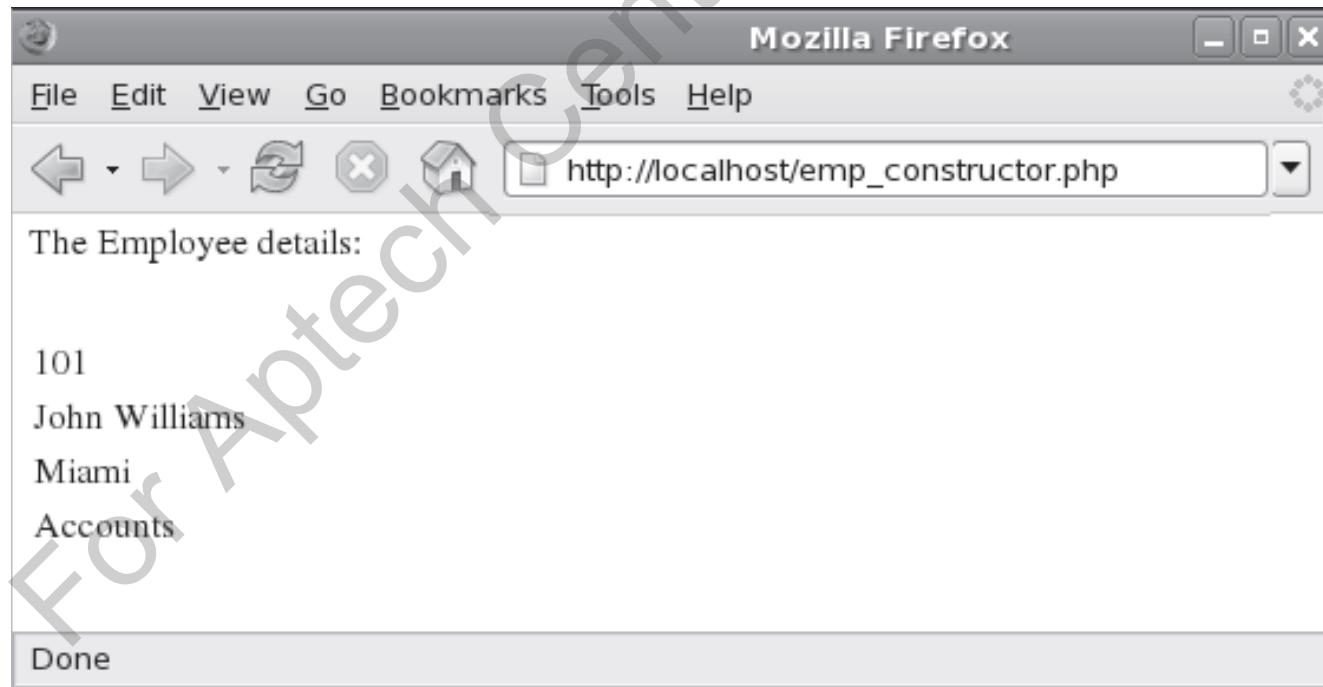
- Once the constructor is defined, call the constructor of the class using the `new` operator
- Before calling the constructor of the class, include the file that contains the class
- `emp_constructor.php`** - Calling the constructor of the `empdetail` class

Snippet

```
<?php
include('emp_constructor.inc');
echo "The Employee details: <BR><BR>";
$empdet = new empdetail(101, "John Williams", "Miami", "Accounts");
echo $empdet->empid,"<BR>";
echo $empdet->empname,"<BR>";
echo $empdet->empcity,"<BR>";
echo $empdet->empdept;
?>
```

- ◆ The new operator creates a new object of the **empdetail** class
- ◆ It calls the functions defined in the **empdetail** class

The following output is displayed:



- ◆ **stringtest.php** - Calling a constructor from the derived class

Snippet

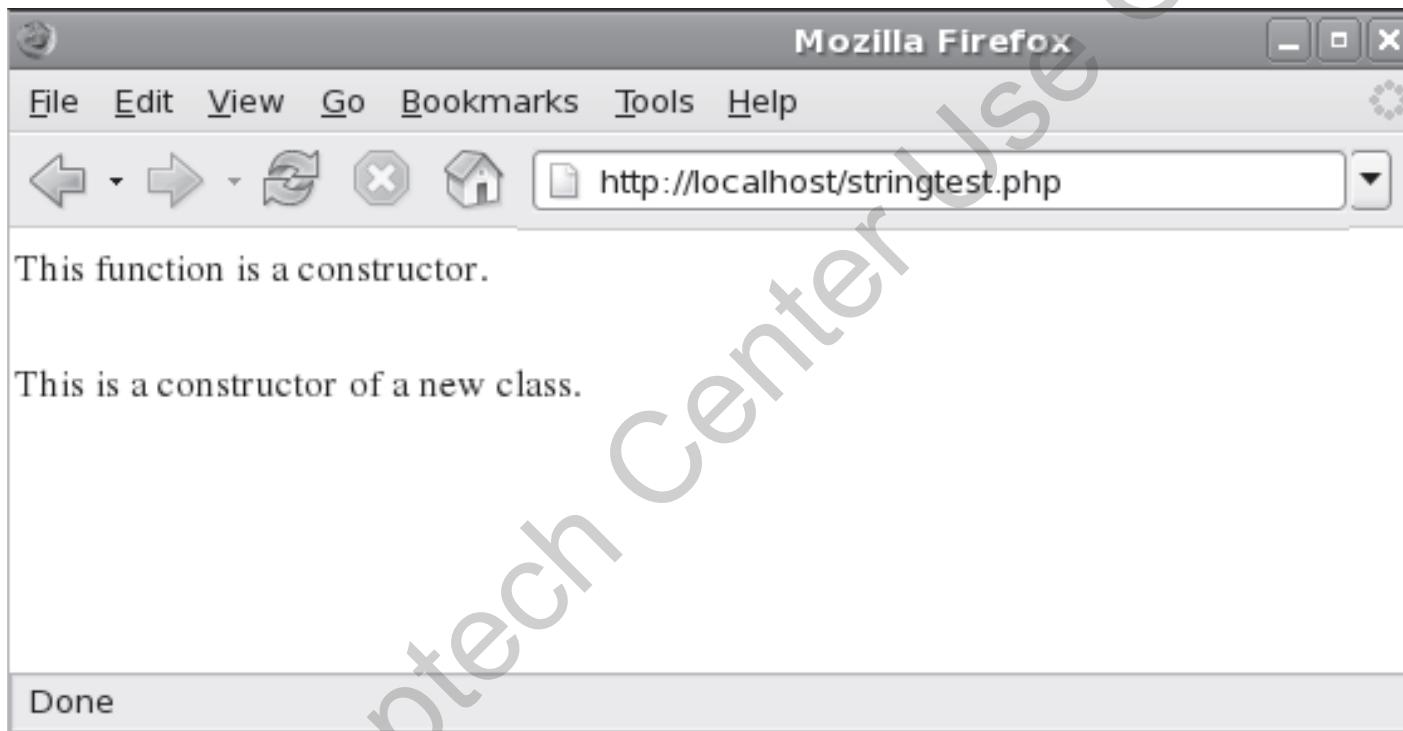
```
<?php  
  
class string  
  
{  
  
function string()  
  
{  
  
echo "This function is a constructor.";  
  
}  
  
function stringdisp()  
  
{  
  
echo "This is function."  
  
}  
}
```

Creating a Constructor

7-8

```
class display extends string  
{  
function display()  
{  
    echo "This is a constructor of a new class.";  
}  
}  
  
$displ = new string;  
echo "<BR><BR>";  
$disp = new display;  
?>
```

The following output is displayed:



- ◆ An object is a self-contained entity that includes both data and procedures to manipulate the data
- ◆ It maintains the codes of the program
- ◆ An object is an instance of a class
- ◆ It provides a reference to the class
- ◆ An object can be manipulated as required
- ◆ The new operator can be used to initialize the object

- The syntax for creating an object is as follows:

Syntax

```
$object_name = new class_name;
```

Where,

object_name - specifies the name of the object

new - initializes the object

class_name - specifies the class name

- ◆ **usermail.php** - Create an object for the class, **usermail**

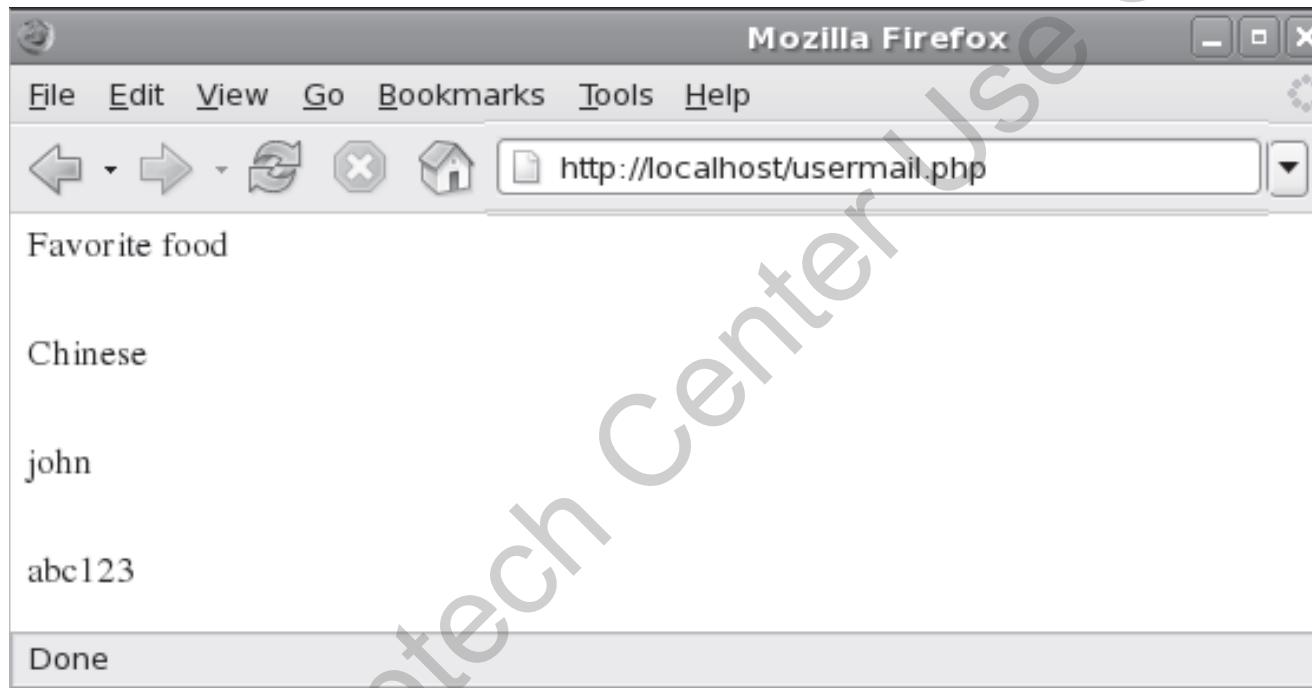
Snippet

```
<?php  
  
class usermail  
{  
    var $username = "john";  
    var $password = "abc123";  
    function dispuser ()  
    {  
        echo $this->username,"<BR><BR>";  
        echo $this->password, " <BR><BR>";  
    }  
}  
  
class userdetails extends usermail  
{  
    var $secretquery = "Favorite food";  
    var $answer = "Chinese";
```

```
function dispdetail()
{
    echo "<BR><BR>";
    echo $this->secretquery, "<BR><BR>";
    echo $this->answer, "<BR><BR>";
}
}

$mail = new userdetails;
$mail1 = new usermail;
$disp1 = $mail->dispdetail();
$disp2 = $mail1->dispuser();
?>
```

The following output is displayed:



- ◆ Object-oriented programming defines the data type of the data structure and the type of functions that can be performed on the data structure
- ◆ The main concepts of an OOP are objects, class, abstraction, encapsulation, polymorphism, and inheritance
- ◆ In hierarchical inheritance, the properties of a single base class can be used multiple times in multiple subclasses
- ◆ A class is a collection of variables and functions that operate on that data.
- ◆ A class can be inherited from a base class to create a new derived class.

- ◆ A constructor is a special function that has the same name as that of its class name, and is called in the main program by using the new operator
- ◆ An object is a self-contained entity that consists of both data and procedures to manipulate the data. It is an instance of a class and can be manipulated as needed

For Aptech Center Use Only

Generator Delegation and Throwable Interface

Session 26



Objectives

- ◆ *Explain generators and generator return expressions.*
- ◆ *Describe Generator Delegation.*
- ◆ *Explain exception handling and changes in exceptions in PHP 7.*
- ◆ *Explain the use of E_ERROR and E_RECOVERABLE_ERROR.*
- ◆ *Explain the Throwable Interface.*

- ◆ Returns/yields values on demand
- ◆ Return multiple values from a function
- ◆ Use the `yield` keyword to return data

For Aptech Center Use Only

- ◆ Demonstrating the use of a generator that prints a range of values

Snippet

```
<?php
function print_range_of_values($min_val, $max_val)
{
    for ($i= $min_val; $i <= $max_val; $i++)
    {
        yield $i;
    }
}
foreach(print_range_of_values(1,10) as $key=>$value)
{
    echo "$key => $value", PHP_EOL;
}
?>
```

- ◆ Displays the following output:



- ◆ Demonstrating the use of range() function

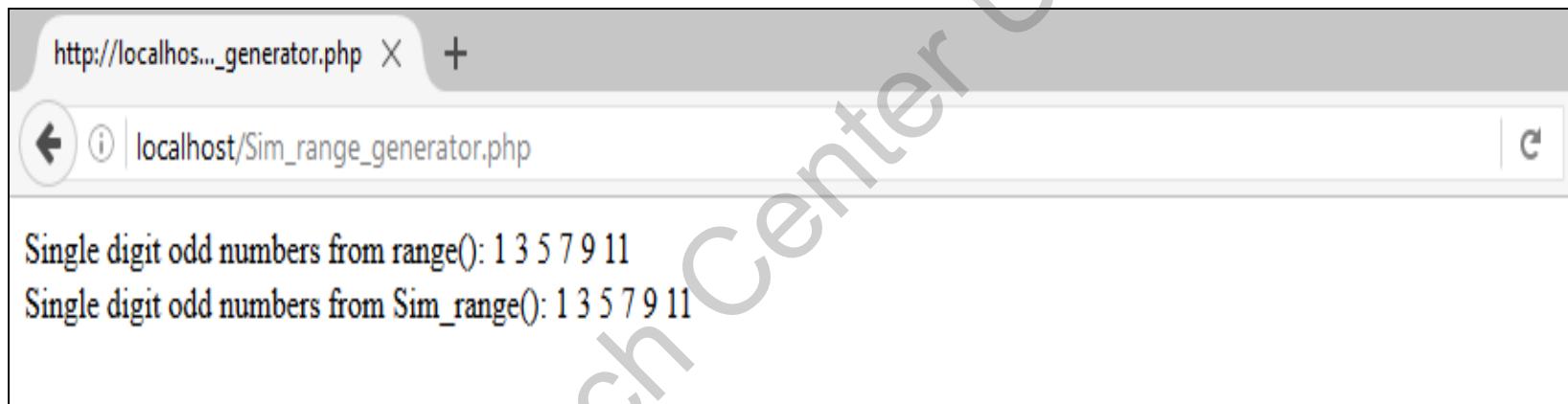
Snippet

```
<?php
function Sim_range($start_num, $max_limit,
    $increment_by = 1) {
    if ($start_num < $max_limit) {
        if ($increment_by <= 0) {
            throw new LogicException('Step must be +ve');
        }
        for ($x = $start_num; $x <= $max_limit; $x +=
$increment_by) {
            yield $x;
        }
    }
}
```

```
else {
    if ($increment_by >= 0) {
        throw new LogicException('Step must be -ve');
    }
    for ($x = $start_num; $x >= $max_limit; $x += $increment_by) {
        yield $x;
    }
}
echo 'Single digit odd numbers from range():<br>';
foreach (range(1, 11, 2) as $num) {
    echo "$num ";
}
echo "<br>";

echo 'Single digit odd numbers from Sim_range():
<br>';
foreach (Sim_range(1, 11, 2) as $num) {
    echo "$num ";
}
?>
```

- ◆ Displays the following output:



A screenshot of a web browser window. The address bar shows the URL `http://localhost/Sim_range_generator.php`. The main content area displays two lines of text: "Single digit odd numbers from range(): 1 3 5 7 9 11" and "Single digit odd numbers from Sim_range(): 1 3 5 7 9 11". This illustrates that both standard range() and generator range() produce identical results.

- ◆ Enables to yield values from other generator, arrays, and traversable objects

Syntax

```
yield from <thing_to_iterate>
```

- ◆ Demonstrating the use of generator delegation

Snippet

```
<?php
function DisplayHello() {
    yield "Hello";
    yield " ";
    yield "World!";
    yield from DisplayGoodbye();
}
function DisplayGoodbye() {
    yield "Goodbye";
    yield " ";
    yield "Mars!";
}
```

```
$gen = DisplayHello();  
foreach ($gen as $value) {  
    echo $value;  
    echo '</br>';  
}  
?>
```

- ◆ Displays the following output:



- ◆ Are a mechanism for error handling
- ◆ An exception handling codes include:

Code	Description
try block	<p>If exceptions do not occur, then the code executes and exists successfully.</p> <p>OR</p> <p>If exceptions occur, then the exception is caught by the catch block and action is taken to handle to exception.</p>
catch block	If exceptions occur, then action statements are included in the block to handle the exception.

Code	Description
throw	A method to manually trigger an exception. However, each throw should be associated with an corresponding catch.
finally	A block usually executed after the try and catch blocks. The code within this block is executed irrespective of whether an exception has been thrown or before normal execution continues.

- ◆ Demonstrating the use of try-catch-finally statements

Snippet

```
<?php
function computeDiv($num) {
    if (!$num) {
        throw new Exception('Division by zero.');
    }
    return 1/$num;
}
```

```
try {
    echo computeDiv(0).'  
<br><br>';
    echo 'Result of division';
} catch (Exception $e) {
    echo 'Caught exception:', $e->
    getMessage(), '  
<br><br>';
} finally {
    echo "Now in finally block.<br><br>";
}

// Continue execution
echo "Program execution
continues<br><br>";
?>
```

- ◆ Displays the following output:

```
Caught exception: Division by zero.
```

```
Now in finally block.
```

```
Program execution continues
```

A fatal runtime error.

A non recoverable error.

The script or program terminates prematurely when this error occurs.

The error enables backward compatibility.

- ◆ Demonstrating the use of E_ERROR error

Snippet

```
<?php
    error_reporting(E_ERROR);
    function handle_error($err_no,
    $err_str,$err_file,$err_line) {
        echo "<b>Error:</b> [$err_no] $err_str -
$err_file:$err_line";
        echo "</br>";
        echo "Terminating PHP Script";
        die();
    }
    //set error handler
    set_error_handler("handle_error", E_ERROR);
    //trigger error
    my_function();
?>
```

- ◆ Displays the following output:

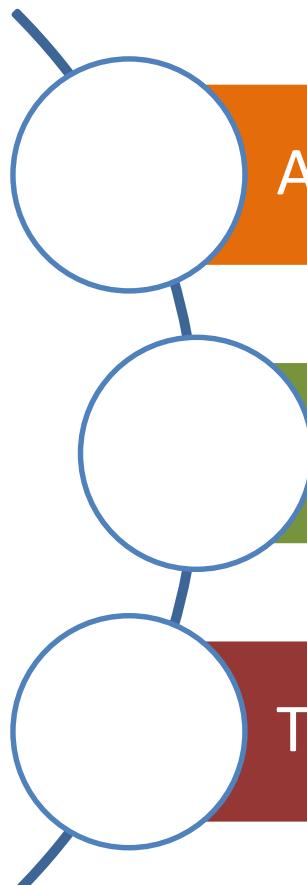
The screenshot shows a web browser window with the URL `http://localhost/E_ERROR_demo.php`. The page displays two error messages and a call stack table.

Error Messages:

- (!) Fatal error: Uncaught Error: Call to undefined function myFunction() in C:\wamp64\www\E_ERROR_demo.php on line 16
- (!) Error: Call to undefined function myFunction() in C:\wamp64\www\E_ERROR_demo.php on line 16

Call Stack:

#	Time	Memory	Function	Location
1	0.0006	362384	{main}()	...\\E_ERROR_demo.php:0



A non-fatal error.

An error that can be handled by a user-defined exception.

The error is not displayed to the user.

- ◆ Demonstrating the use of E_RECOVERABLE_ERROR

Snippet

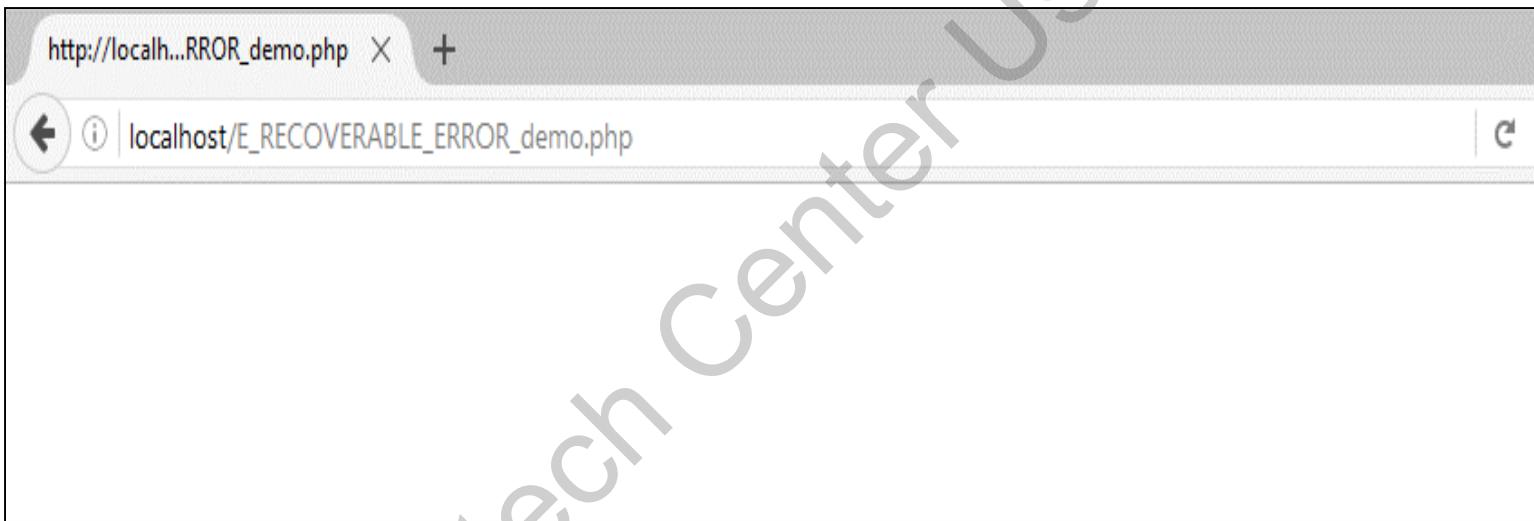
```
<?php
    error_reporting(E_RECOVERABLE_ERROR);
    function handle_error($errno,
    $errstr,$error_file,$error_line) {
        echo "<b>Error:</b> [$errno] $errstr -
        $error_file:$error_line";
        echo "</br>";
        echo "Terminating PHP Script";
        die();
    }

    //set error handler
    set_error_handler("handleError", E_RECOVERABLE_ERROR);

    //trigger error
    myFunction();

?>
```

- ◆ Displays the following output:



- ◆ Benefits of handling fatal errors as exceptions are:

Enables to handle fatal errors using
the try-catch block

Helps handle catchable errors easily

Helps easy debugging

- ◆ Exception and Error classes implement the Throwable interface.
- ◆ The root of the hierarchy was \Exception in the previous versions of PHP.

For Aptech Center Use Only

- ◆ Demonstrating the hierarchy of classes and interfaces.

➤ Interface Throwable

- Exception implements Throwable
 - LogicException (extends Exception)
 - InvalidArgumentException (extends LogicException)
 - ...
- Error implements Throwable (Replaces EngineException)
 - TypeError extends Error
 - ParseError extends Error
 - ...

- ◆ Demonstrating a fatal error where exception is not caught and error is thrown.

Snippet

```
<?php
function add(int $left_op, int $right_op) {
    return $left_op + $right_op;
}
try {
    echo add('two', 'three');
} catch (Exception $e) {
    // Handle or log exception.
    echo "Error occurred in the program due to
parameter datatype mismatch";
}
?>
```

- ◆ Displays the following output:

The screenshot shows a web browser window with the URL `localhost/exception_1.php`. The page displays two error messages and a call stack table.

Error Messages:

- (!) Fatal error: Uncaught TypeError: Argument 1 passed to add() must be of the type integer, string given, called in C:\wamp64\www\exception_1.php on line 8 and defined in C:\wamp64\www\exception_1.php on line 3
- (!) TypeError: Argument 1 passed to add() must be of the type integer, string given, called in C:\wamp64\www\exception_1.php on line 8 in C:\wamp64\www\exception_1.php on line 3

Call Stack:

#	Time	Memory	Function	Location
1	0.0003	361200	{main}()	...exception_1.php:0
2	0.0003	361264	add()	...exception_1.php:8

- ◆ Demonstrating to catch an Exception and Error

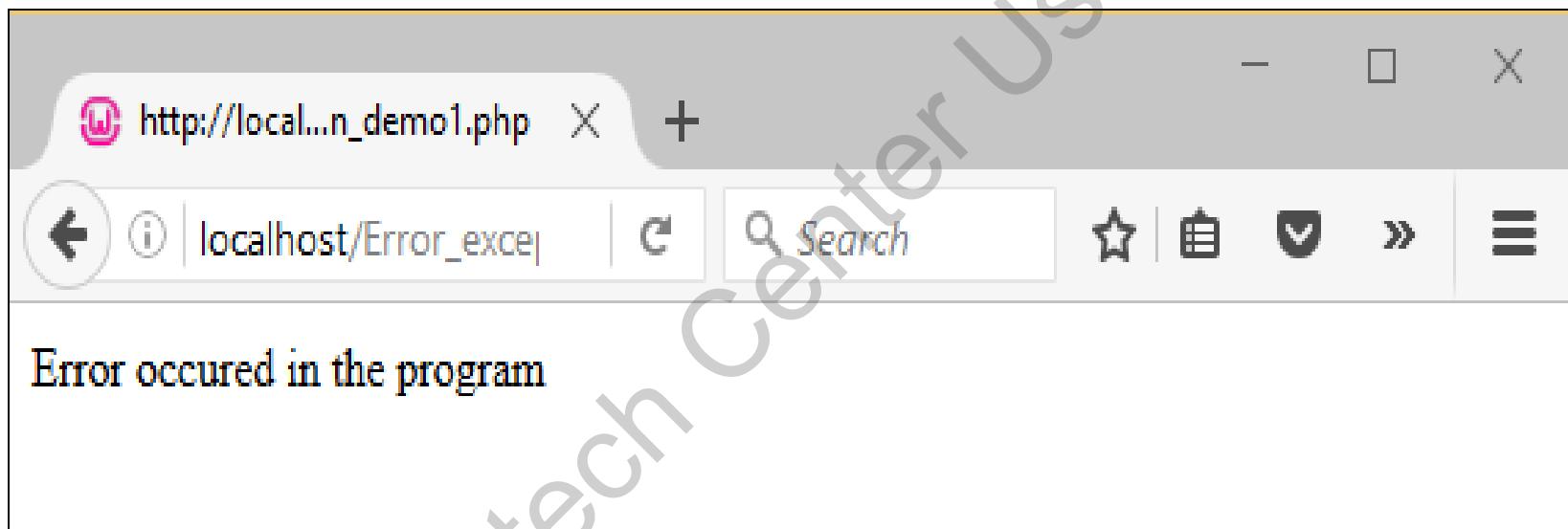
Snippet

```
<?php

function add(int $left_op, int $right_op) {
    return $left_op + $right_op;
}

try {
    echo add('two', 'three');
} catch (Exception $e) {
    // Handle or log exception.
    echo "Exception occurred in the program";
}
catch (Error $e) { // Log error and end gracefully
    echo "Error occurred in the program";
}
?>
```

- ◆ Displays the following output:



- ◆ Generator return expressions is a new feature in PHP 7 that allows you to 'return' a value on completion of a generator using `return` keyword and `Generator::getReturn()` method.
- ◆ Generator Delegation promotes reusability and cleaner code.
- ◆ Generator Delegation allows getting/yielding values from other generators using `yield from` clause.

- ◆ PHP supports error and exception handling mechanisms that allow programs to continue or exit gracefully instead of exiting abruptly.
- ◆ In PHP 7, all errors including fatal errors are engine exceptions.
- ◆ Exception and Error classes implement the new Throwable interface, which is newly introduced in PHP 7.