SRINIVAS UNIVERSITY

**CITY CAMPUS PANDESHWAR MANGALURU- 575001.**

**COLLEGE OF COMPUTER & INFORMATION SCIENCE**

# BACKGROUND STUDY MATERIAL

LAMP TECHNOLOGY

**V SEMESTER B.C.A**



**Sandhya Bangera**

|  |  |  |
| --- | --- | --- |
|  | | |
| **Paper :**  17BCASD51  **Theory/Week:**4 Hours  **Credits:**4 | **LAMP Technology** | **Hours:** 40  **IA :** 50  **Exam:** 50 |
| **Unit - I** |  | **8hrs** |
| **Linux Operating System:** Linux Operating System Concepts and Architecture; Overview of the Linux Kernel, User Space, Kernel Space; Processes and Daemons, Process Control; **Overview of Linux Administration**; Linux File system, User, Group and Resource Management; File system Permissions, Access Permissions and Security,  **Apache web Server:** Introduction, MIME types and CGI Files, Global Environment, Authentication and log files, PHP and the web server Architecture model, Overview of PHP capabilities, CGI vs. Shared Object Model. | | |
| **Unit – II** |  | **8hrs** |
| **MYSQL:** Introduction**,** MySQL Administration; Commands – Show database, create database, Use command, describe command, Insert command, Select command, Update command, delete command, Creating databases; Defining tables and column ; Entering Data; Understanding the datatype, Using Autoincrement.  **PHP:** Introduction**,** PHP syntax overview, Commenting in PHP code, Benefits of PHP, Combining HTML and PHP, PHP – Variables; Variables naming, Local variables, Function parameters, Global variables, Static variables, Constants. | | |
| **Unit – III** |  | **8hrs** |
| **Fundamentals of PHP:** Datatypes, and Operators; Decision making: if, else, elseif, switch statements, Flow control and loops types: while, do while, for, foreach, breaking out of loop, the continue statement.  creating  **PHP Date function, Arrays and Strings**  Date functions, Arrays and variable handing functions, arrays, Numerically indexed array, Associative array, and multidimensional arrays, array functions, Strings and strings operations and functions,. | | |
| **Unit – IV** |  | **8hrs** |
| **Working with form:** Using HTML Forms, PHP – GET and POST methods, The  $\_Request variable, Creating a dynamic HTML Form with PHP, PHP file Inclusion- Include(), require().  **PHP Function:** Creating PHP function, PHP function with parameters, Sessions- Starting php session, destroying session, Cookies- Setting cookies, accessing cookies, Sending emails, File Upload. | | |
| **Unit – V** |  | **8hrs** |
| **PHP and MySQL:** Areas where PHP and MySQL are used together, Connecting to a | | |
|  | | |

|  |
| --- |
| MySQL Database, Closing Database connection, Create MySQL Database Using PHP, Selecting a Database, Creating Database Tables, Delete MySQL Database Using PHP.  **Accessing MySQL database from web with PHP :** Querying the database, Retrieving the query results, Insert Data to MySQL Database, Retrieving Data from MySQL Database, Updating Data to MySQL Database, Deleting Data from MySQL Database, Disconnecting from the database. |
| **Text Books** |
| Lee , **Open Source Development with LAMP : Using Linux, Apache, MySQL, Perl and PHP**, Pearson Education, 2006 |
| Timothy Boronczyk, et al, **Beginning PHP6, Apache, MySQL Web Development**, Wiley India Pvt Ltd,2009 |
| Julie C Meloni, **Teach Yourself PHP, MySQL and Apache All-in-One**, SAMS, 2008 |
| **Reference Books** |
| W. Jason Gilmore, **Beginning PHP and MySQL: From Novice to Professional**, 4th Edition, Apress, 2010 |
| Aleksa Vukotic, James Goodwill, **Apache Tomcat 7**, Apress, 2011 |
| Richard Petersen, **Linux Complete Reference, 6th Edition,** Tata McGraw Hill Education Private Limited |
|  |
|  |

|  |
| --- |
| **TEACHING PLAN**  **UNIT- 1 8 hrs.**  Session 1 Linux Operating System Concepts and Architecture  Session 2. Overview of the Linux Kernel; User Space, Kernel Space; Processes and Daemons, Process Control  Session 3 Linux File system  Session 4 File system Permissions, Access Permissions and Security Session 5 MIME types and CGI Files,  Session 6 Global Environment, Authentication and log files Session 7 PHP and the web server Architecture model Session 8 CGI vs. Shared Object Model.  **UNIT-2 8 hrs.**  Session 9 Introduction  Session 10 MySQL Administration; Commands  Session 11 Creating databases; Defining tables and column ; Entering Data; Session 12 Understanding the datatype, Using Autoincrement  Session 13 PHP syntax overview  Session 14 Benefits of PHP, commenting in php. Session 15 Combining HTML and PHP  Session 16 Variables, scope of variables, constants.  **UNIT-3 8 hrs.**  Session 17 Datatypes in PHP  Session 18 Operators- Assignment, logical, Arithmetic. Session 19 Decision making : if, else, elseif, switch statements  Session 20 Flow control and loops types: while, do while, for, foreach, breaking out of loop, the continue statement.  Session 21 Date function  Session 22 Arrays and variable handling functions  Session 23 Numerical, Associative and multidimensional arrays |

Session 24 Strings and String functions

**UNIT-4 8 hrs.**

Session 25 Using HTML Forms

Session 26 PHP – GET and POST methods, The $\_Request variable Session 27 Creating a dynamic HTML Form with PHP

Session 28 PHP file Inclusion- Include(), require().

Session 29 Creating PHP function, PHP function with parameters Session 30 Sessions- Starting php session, destroying session Session 31 Cookies- Setting cookies, accessing cookies

Session 32 Sending emails, File Upload.

**UNIT-5 8hrs**

Session 33 Areas where PHP and MySQL are used together

Session 34 Connecting to a MySQL Database, Closing Database connection Session 35 Create MySQL Database Using PHP, Selecting a Database Session 36 Creating Database Tables, Delete MySQL Database Using PHP Session 37 Querying the database, Retrieving the query results

Session 38 Insert Data to MySQL Database, Retrieving Data from MySQL Database Session 39 Updating Data to MySQL Database

Session 40 Deleting Data from MySQL Database, Disconnecting from the database.

**CONTENT**

# UNIT I

1. **Linux Operating System:**

Linux Operating System Concepts and Architecture Overview of the Linux Kernel

User Space, Kernel Space Processes and Daemons Process Control

**Overview of Linux Administration**

Linux File system

User, Group and Resource Management File system Permissions

Access Permissions and Security

1. **Apache web Server:**

Introduction

MIME types and CGI Files Global Environment Authentication and log files

PHP and the web server Architecture model Overview of PHP capabilities

CGI vs. Shared Object Model.

# UNIT II

1. **MYSQL:**

Introduction

MySQL Administration

Commands – Show database, create database, Use command, describe command. Insert command, Select command, Update command, delete command.

Creating databases. Defining tables and column Entering Data.

Understanding the datatype. Using Autoincrement.

1. **PHP:**

Introduction.

PHP syntax overview.

Commenting in PHP code. Benefits of PHP.

Combining HTML and PHP. PHP – Variables; Variables naming.

Local variables, Function parameters, Global variables, Static variables. Constants

# UNIT III

1. **Fundamentals of PHP:**

Datatypes, and Operators.

Decision making: if, else, elseif, switch statements.

Flow control and loops types while, do while, for, foreach.

breaking out of loop, the continue statement.

1. **PHP Date function, Arrays and Strings**

Date functions

Arrays and variable handing functions arrays Numerically indexed array

Associative array. Multidimensional arrays Array functions.

Strings and strings operations and functions,.

**UNIT IV**

1. **Working with form:**

Using HTML Forms

PHP – GET and POST methods The $\_Request variable

Creating a dynamic HTML Form with PHP PHP file Inclusion- Include(), require().

1. **PHP Function:**

Creating PHP function

PHP function with parameters

Sessions- Starting php session, destroying session. Cookies- Setting cookies, accessing cookies.

Sending email. File Upload.

# UNIT V

1. **PHP and MySQL:**

Areas where PHP and MySQL are used together Connecting to a MySQL Database.

Closing Database connection. Create MySQL Database Using PHP Selecting a Database

Creating Database Tables

Delete MySQL Database Using PHP.

1. **Accessing MySQL database from web with PHP :**

Querying the database Retrieving the query results Insert Data to MySQL Database,

Retrieving Data from MySQL Database, Updating Data to MySQL Database Deleting Data from MySQL Database Disconnecting from the database.

|  |
| --- |
| **UNIT I**  **CHAPTER 1**  **LINUX OPERATING SYSTEM**  1.1 **Linux Operating System Concepts and Architecture**  The Linux kernel is composed of five main subsystems that communicate using procedure calls. The architecture of the kernel is one of the reasons that Linux has been successfully adopted by many users. In particular, the Linux kernel architecture was designed to support a large number of volunteer developers. Further, the subsystems that are most likely to need enhancements were architected to easily support extensibility.    **Figure 1.1:** *Decomposition of Linux System into Major Subsystems*  The Linux operating system is composed of four major subsystems:   1. **User Applications** -- the set of applications in use on a particular Linux system will be different depending on what the computer system is used for, but typical examples include a word-processing application and a web-browser. 2. **O/S Services** -- these are services that are typically considered part of the operating system (a windowing system, command shell, etc.); also, the programming interface to the kernel (compiler tool and library) is included in this subsystem. 3. **Linux Kernel** -- this is the main area of interest in this paper; the kernel abstracts and mediates access to the hardware resources, including the CPU. 4. **Hardware Controllers** -- this subsystem is comprised of all the possible physical devices in a Linux installation; for example, the CPU, memory hardware, hard disks, and network hardware are all members of this subsystem.   1.2 **Over view of the Linux Kernel**  The Linux kernel is composed of five main subsystems:   1. The [Process Scheduler](http://oss.org.cn/ossdocs/linux/kernel/a1/index.html#Toc_3_1) (SCHED) is responsible for controlling process access to the CPU. The scheduler enforces a policy that ensures that processes will have fair access to the CPU, while ensuring that necessary hardware actions are performed by the kernel on time. 2. The [Memory Manager](http://oss.org.cn/ossdocs/linux/kernel/a1/index.html#Toc_3_2) (MM) permits multiple process to securely share the machine's main memory system. In addition, the memory manager supports virtual memory that allows Linux to support processes that use more memory than is available in the system. Unused |

|  |
| --- |
| memory is swapped out to persistent storage using the file system then swapped back in when it is needed.   1. The [Virtual File System](http://oss.org.cn/ossdocs/linux/kernel/a1/index.html#Toc_3_3) (VFS) abstracts the details of the variety of hardware devices by presenting a common file interface to all devices. In addition, the VFS supports several file system formats that are compatible with other operating systems. 2. The [Network Interface](http://oss.org.cn/ossdocs/linux/kernel/a1/index.html#Toc_3_4) (NET) provides access to several networking standards and a variety of network hardware. 3. The [Inter-Process Communication](http://oss.org.cn/ossdocs/linux/kernel/a1/index.html#Toc_3_5) (IPC) subsystem supports several mechanisms for process-to-process communication on a single Linux system.    1. **User space and Kernel space**   *System* [*memory*](http://www.linfo.org/memory.html)in [Linux](http://www.linfo.org/linuxdef.html) can be divided into two distinct regions: *kernel space* and [*user*](http://www.linfo.org/user_space.html)[*space*](http://www.linfo.org/user_space.html). Kernel space is where the [*kernel*](http://www.linfo.org/kernel.html) (i.e., the core of the operating system) *executes* (i.e., runs) and provides its *services*. User space is that set of memory locations in which *user processes* (i.e., everything other than the kernel) run.  Memory consists of [*RAM*](http://www.linfo.org/ram.html)(random access memory) cells, whose contents can be *accessed* (i.e., read and written to) at extremely high speeds but are retained only temporarily (i.e., while in use or, at most, while the power supply remains on). Its purpose is to hold programs and data that are currently in use and thereby serve as a high speed intermediary between the CPU (central processing unit) and the much slower [*storage*](http://www.linfo.org/storage.html), which most commonly consists of one or more hard disk drives (HDDs).  A [*process*](http://www.linfo.org/process.html) is an executing instance of a program. One of the roles of the kernel is to manage individual user processes within this space and to prevent them from interfering with each other.  Kernel space can be accessed by user processes only through the use of *system calls*. System calls are requests in a Unix-like operating system by an *active process* for a service performed by the kernel, such as *input/output* (I/O) or process creation. An active process is a process that is currently progressing in the CPU, as contrasted with a process that is waiting for its next turn in the CPU. I/O is any program, operation or device that transfers data to or from a CPU and to or from a peripheral device (such as disk drives, keyboards, mice and printers).   * 1. **Processes and Demons**   A ***process*** is an *executing* (i.e., running) [instance](http://www.linfo.org/instance.html) of a [*program*.](http://www.linfo.org/program.html) Processes are also frequently referred to as *tasks*.  A program is an [*executable file*](http://www.linfo.org/executable.html) that is held in [*storage*](http://www.linfo.org/storage.html). Storage refers to devices or media that can retain data for relatively long periods of time (e.g., years or even decades), such as [hard](http://www.linfo.org/hdd.html) [disk drives](http://www.linfo.org/hdd.html) (HDDs), optical disks and magnetic tape. This contrasts with [*memory*](http://www.linfo.org/memory.html), whose contents can be accessed (i.e., read and written to) at extremely high speeds but which are retained only temporarily (i.e., while in use or only as long as the power supply remains on).  A program is a passive entity until it is launched, and a process can be thought of as a program in action. Processes are dynamic entities in that they are constantly changing as their machine code instructions are executed by the CPU. Each process consists of (1) *system resources* that are allocated to it, (2) a section of memory, (3) security attributes (such as its *owner* and its set of *permissions*) and (4) the processor *state*.  An alternative definition of a process is the execution *context* of a running program, i.e., all of the activity in the current *time slot* in the CPU. A time slot, also called a *time slice* or a *quantum*, is the length of time that each process is permitted to run in the CPU until it is *preempted* (i.e., replaced) by another process in a *time sharing* [operating system.](http://www.linfo.org/operating_systems_list.html)  A ***daemon*** is a type of [program](http://www.linfo.org/program.html) on [Unix-like](http://www.linfo.org/unix-like.html) [operating systems](http://www.linfo.org/operating_systems_list.html) that runs unobtrusively in the background, rather than under the direct control of a user, waiting to be activated by the occurrence of a specific event or condition. |

|  |
| --- |
| Unix-like systems typically run numerous daemons, mainly to accommodate requests for services from other computers on a network, but also to respond to other programs and to hardware activity. Examples of actions or conditions that can trigger daemons into activity are a specific time or date, passage of a specified time interval, a file landing in a particular directory, receipt of an e-mail or a Web request made through a particular communication line. It is not necessary that the perpetrator of the action or condition be aware that a daemon is *listening*, although programs frequently will perform an action only because they are aware that they will implicitly arouse a daemon.  Daemons are usually instantiated as [*processes*](http://www.linfo.org/process.html). A process is an *executing* (i.e., running) instance of a program. Processes are managed by the [*kernel*](http://www.linfo.org/kernel.html)(i.e., the core of the operating system), which assigns each a unique *process identification number* (PID).  There are three basic types of processes in [Linux](http://www.linfo.org/linuxdef.html): interactive, batch and daemon. Interactive processes are run interactively by a user at the [*command line*](http://www.linfo.org/command_line.html)(i.e., all-text mode). Batch processes are submitted from a queue of processes and are not associated with the command line; they are well suited for performing recurring tasks when system usage is otherwise low. Daemons are recognized by the system as any processes whose *parent process* has a PID of one, which always represents the process *init*. init is always the first process that is started when a Linux computer is *booted up* (i.e., started), and it remains on the system until the computer is turned off. init *adopts* any process whose *parent* process *dies* (i.e., terminates) without waiting for the *child* process's status. Thus, the common method for launching a daemon involves *forking* (i.e., dividing) once or twice, and making the parent (and grandparent) processes die while the child (or grandchild) process begins performing its normal function.  Some daemons are launched via [*System V*](http://www.linfo.org/system_v.html) *init scripts*, which are *scripts* (i.e., short programs) that are run automatically when the system is booting up. They may either survive for the duration of the session or be regenerated at intervals. Many daemons are now started only as required and by a single daemon, *xinetd* (which has replaced *inetd* in newer systems), rather than running continuously. xinetd, which is referred to as a *TCP/IP super server*, itself is started at boot time, and it listens to the *ports* assigned to the processes listed in the  */etc/inetd.conf* or in */etc/xinetd.conf* configuration file. Examples of daemons that it starts include *crond* (which runs scheduled tasks), *ftpd* (file transfer), *lpd* (laser printing), *rlogind* (remote login), *rshd* (remote command execution) and *telnetd* (telnet). In addition to being launched by the operating system and by application programs, some daemons can also be started manually. Examples of commands that launch daemons include *binlogd* (which logs binary events to specified files), *mysqld* (the MySQL databse server) and *apache* (the Apache web server). In many Unix-like operating systems, including Linux, each daemon has a single *script* (i.e., short program) with which it can be terminated, restarted or have its status checked. The handling of these scripts is based on [*runlevels*](http://www.linfo.org/runlevel_def.html). A runlevel is a configuration or operating state of the system that only allows certain selected processes to exist. Booting into a different runlevel can help solve certain problems, including repairing system errors. The term *daemon* is derived from the daemons of Greek mythology, which were supernatural beings that ranked between gods and mortals and which possessed special knowledge and power1. For example, Socrates claimed to have a daemon that gave him warnings and advice but never coerced him into following it. He also claimed that his daemon exhibited greater accuracy than any of the forms of divination practiced at the time.  The word *daemon* was first used in a computer context at the pioneering Project MAC (which later became the MIT Laboratory for Computer Science) using the IBM 7094 in 1963. This |

|  |
| --- |
| usage was inspired by Maxwell's daemon of physics and thermodynamics, which was an imaginary agent that helped sort molecules of different speeds and worked tirelessly in the background. The term was then used to describe background processes which worked tirelessly to perform system chores. The first computer daemon was a program that automatically made tape backups. After the term was adopted for computer use, it was rationalized as an [acronym](http://www.linfo.org/acronym_list.html) for *Disk And Execution MONitor*.  1.5 **Process Control**  Under Linux, the *ptrace* system call is supported for process control, and it works as in 4.3BSD. To obtain process and system information, Linux also provides a */proc* filesystem, but with very different semantics. Under Linux, */proc* consists of a number of files providing general system information, such as memory usage, load average, loaded module statistics, and network statistics. These files are generally accessed using *read* and *write* and their contents can be parsed using *scanf*. The */proc* filesystem under Linux also provides a directory entry for each running process, named by process ID, which contains file entries for information such as the command line, links to the current working directory and executable file, open file descriptors, and so forth. The kernel provides all of this information on the fly in response to *read* requests. This implementation is not unlike the */proc* filesystem found in Plan 9, but it does have its drawbacks--for example, for a tool such as *ps* to list a table of information on all running processes, many directories must be traversed and many files opened and read. By comparison, the *kvm* routines used on other UNIX systems read kernel data structures directly with only a few system calls. Obviously, each implementation is so vastly different that porting applications which use them can prove to be a real task. It should be pointed out that the SVR4 */proc* filesystem is a very different beast than that found in Linux, and they may not be used in the same context. Arguably, any program which uses the *kvm* routines or SVR4 */proc* filesystem is not really portable, and those sections of code should be rewritten for each operating system.  **Overview of Linux Administration**  2.1 **Linux File system**  General A simple description of the UNIX system, also applicable to Linux, is this: "On a UNIX system, everything is a file; if something is not a file, it is a process." This statement is true because there are special files that are more than just files (named pipes and sockets, for instance), but to keep things simple, saying that everything is a file is an acceptable generalization. A Linux system, just like UNIX, makes no difference between a file and a directory, since a directory is just a file containing names of other files. Programs, services, texts, images, and so forth, are all files. Input and output devices, and generally all devices, are considered to be files, according to the system. In order to manage all those files in an orderly fashion, man likes to think of them in an ordered tree-like structure on the hard disk, as we know from MS-DOS (Disk Operating System) for instance. The large branches contain more branches, and the branches at the end contain the tree's leaves or normal files. For now we will use this image of the tree, but we will find out later why this is not a fully accurate image. Sorts of files Most files are just files, called *regular* files; they contain normal data, for example text files, executable files or programs, input for or output from a program and so on. While it is reasonably safe to suppose that everything you encounter on a Linux system is a file, there are some exceptions.   * *Directories*: files that are lists of other files. * *Special files*: the mechanism used for input and output. Most special files are in /dev. * *Links*: a system to make a file or directory visible in multiple parts of the system's file tree. |

|  |  |
| --- | --- |
| * *(Domain) sockets*: a special file type, similar to TCP/IP sockets, providing inter- process networking protected by the file system's access control. * *Named pipes*: act more or less like sockets and form a way for processes to communicate with each other, without using network socket semantics.   As a user, you only need to deal directly with plain files, executable files, directories and links. The special file types are there for making your system do what you demand from it and are dealt with by system administrators and programmers.  For convenience, the Linux file system is usually thought of in a tree structure. On a standard Linux system you will find the layout generally follows the scheme presented below.  **Figure 2-1. Linux file system layout**  This is a layout from a RedHat system. Depending on the system admin, the operating system and the mission of the UNIX machine, the structure may vary, and directories may be left out or added at will. The names are not even required; they are only a convention. The tree of the file system starts at the trunk or *slash*, indicated by a forward slash (/). This directory, containing all underlying directories and files, is also called the *root directory* or "the root" of the file system. Directories that are only one level below the root directory are often preceded by a slash, to indicate their position and prevent confusion with other directories that could have the same name. When starting with a new system, it is always a good idea to take a look in the root directory. Let's see what you could run into: | |
| **DIRECTORY** | **CONTENT** |
| /bin | Common programs, shared by the system, the system administrator and the users. |
| /boot | The startup files and the kernel, vmlinuz. In some recent distributions also grub data. Grub is the GRand Unified Boot loader and is an attempt to get rid of the many different boot-loaders we know today. |
|  | |

|  |  |
| --- | --- |
| **DIRECTORY** | **CONTENT** |
| /dev | Contains references to all the CPU peripheral hardware, which are represented as files with special properties. |
| /etc | Most important system configuration files are in /etc, this directory contains data similar to those in the Control Panel in Windows |
| /home | Home directories of the common users. |
| /initrd | (on some distributions) Information for booting. Do not remove! |
| /lib | Library files, includes files for all kinds of programs needed by the system and the users. |
| /lost+found | Every partition has a lost+found in its upper directory. Files that were saved during failures are here. |
| /misc | For miscellaneous purposes. |
| /mnt | Standard mount point for external file systems, e.g. a CD-ROM or a digital camera. |
| /net | Standard mount point for entire remote file systems |
| /opt | Typically contains extra and third party software. |
| /proc | A virtual file system containing information about system resources. More information about the meaning of the files in proc is obtained by entering the  command **man proc** in a terminal window. The file proc.txt discusses the virtual file system in detail. |
| /root | The administrative user's home directory. Mind the difference between /, the root directory and /root, the home directory of the *root* user. |
| /sbin | Programs for use by the system and the system administrator. |
| /tmp | Temporary space for use by the system, cleaned upon reboot, so don't use this for saving any work! |
| /usr | Programs, libraries, documentation etc. for all user-related programs. |
| /var | Storage for all variable files and temporary files created by users, such as log files, the mail queue, the print spooler area, space for temporary storage of files downloaded from the Internet, or to keep an image of a CD before burning it. |
| **Table 2-1. Subdirectories of the root directory User Group and Resource Management**  A *user* is anyone who uses a computer. In this case, we are describing the names which represent those users. It may be Mary or Bill, and they may use the names Dragonlady or Pirate in place of their real name. All that matters is that the computer has a name for each account it creates, and it is this name by which a person gains access to use the computer. Some system services also run using restricted or privileged user accounts.  Managing users is done for the purpose of security by limiting access in certain specific ways. The superuser (root) has complete access to the operating system and its configuration; it is intended for administrative use only. Unprivileged users can use the [su](https://wiki.archlinux.org/index.php/Su) and [sudo](https://wiki.archlinux.org/index.php/Sudo) programs for controlled privilege escalation. Any individual may have more than one account, as long as they use a different name for each account they create. Further, there are some reserved names which may not be used such as "root". Users may be grouped together into a "group", and users may be added to an existing group to utilize the privileged access it grants. Linux/Unix | |

|  |
| --- |
| operating systems have the ability to multitask in a manner similar to other operating systems. However, Linux's major difference from other operating systems is its ability to have multiple users. Linux was designed to allow more than one user access to the system at the same time. In order for this multiuser design to work properly, there needs to be a method to protect users from each other. This is where permissions come in to play.  **File system Permissions**  [Read, Write & Execute Permissions](https://library.linode.com/using-linux/users-and-groups#sph_id2)  Permissions are the "rights" to act on a file or directory. The basic rights are read, write, and execute. Read - A readable permission allows the contents of the file to be viewed. A read permission on a directory allows you to list the contents of a directory. Write - A write permission on a file allows you to modify the contents of that file. For a directory, the write permission allows you to edit the contents of a directory (e.g. add/delete files). Execute - For a file the executable permission allows you to run the file and execute a program or script. For a directory, the execute permission allows you to change to a different directory and make it your current working directory. Users usually have a default group, but they may belong to several additional groups.  [**Viewing File Permissions**](https://library.linode.com/using-linux/users-and-groups#sph_id3)  To view the permissions on a file or directory, issue the command ls -l <directory/file>. Remember to replace the information in the **< >** with the actual file or directory name. Below is sample output for the ls command: -rw-r--r-- 1 root root 1031 Nov 18 09:22 /etc/passwd  The first ten characters show the access permissions. The first dash (-) indicates the type of file (d for directory, s for special file, and - for a regular file). The next three characters (**rw-**) define the owner's permission to the file. In this example, the file owner has read and write permissions only. The next three characters (**r--**) are the permissions for the members of the same group as the file owner (which in this example is read only). The last three characters (**r-**  **-**) show the permissions for all other users and in this example it is read only.  **Access permissions and security**  The Unix operating system (and likewise, Linux) differs from other computing environments in that it is not only a *multitasking* system but it is also a *multi-user* system as well. The computer would support many users at the same time. In order to make this practical, a method had to be devised to protect the users from each other. After all, you could not allow the actions of one user to crash the computer, nor could you allow one user to interfere with the files belonging to another user. Linux uses the same permissions scheme as Unix. Each file and directory on your system is assigned access rights for the owner of the file, the members of a group of related users, and everybody else. Rights can be assigned to read a file, to write a file, and to execute a file (i.e., run the file as a program). To see the permission settings for a file, we can use the ls command as follows:  **[me@linuxbox me]$ ls -l some\_file**  **-rw-rw-r-- 1 me me 1097374 Sep 26 18:48 some\_file**  We can determine a lot from examining the results of this command:   * The file "some\_file" is owned by user "me" * User "me" has the right to read and write this file * The file is owned by the group "me" * Members of the group "me" can also read and write this file |

|  |
| --- |
| * Everybody else can read this file   Let's try another example. We will look at the bash program which is located in the /bin directory:  **[me@linuxbox me]$ ls -l /bin/bash**  **-rwxr-xr-x 1 root root 316848 Feb 27 2000 /bin/bash**  Here we can see:   * The file "/bin/bash" is owned by user "root" * The superuser has the right to read, write, and execute this file * The file is owned by the group "root" * Members of the group "root" can also read and execute this file * Everybody else can read and execute this file   In the diagram below, we see how the first portion of the listing is interpreted. It consists of a character indicating the file type, followed by three sets of three characters that convey the reading, writing and execution permission for the owner, group, and everybody else.    **chmod**  The chmod command is used to change the permissions of a file or directory. To use it, you specify the desired permission settings and the file or files that you wish to modify.  It is easy to think of the permission settings as a series of bits (which is how the computer thinks about them). Here's how it works:  rwx rwx rwx = 111 111 111  rw- rw- rw- = 110 110 110  rwx = 111 000 000  and so on...  rwx = 111 in binary = 7 rw- = 110 in binary = 6 r-x = 101 in binary = 5 |

|  |  |
| --- | --- |
| r-- = 10  Now, if single di For exa but wan  **[me@li** Here is are use  *Value* | 0 in binary = 4  you represent each of the three sets of permissions (owner, group, and other) as a git, you have a pretty convenient way of expressing the possible permissions settings. mple, if we wanted to set some\_file to have read and write permission for the owner, ted to keep the file private from others, we would:  **nuxbox me]$ chmod 600 some\_file**  a table of numbers that covers all the common settings. The ones beginning with "7" d with programs (since they enable execution) and the rest are for other kinds of files.  *Meaning* |
| *777* | *(rwxrwxrwx)* No restrictions on permissions. Anybody may do anything. Generally not a desirable setting. |
| *755* | *(rwxr-xr-x)* The file's owner may read, write, and execute the file. All others may read and execute the file. This setting is common for programs that are used by all users. |
| *700* | *(rwx------)* The file's owner may read, write, and execute the file. Nobody else has any rights. This setting is useful for programs that only the owner may use and must be kept private from others. |
| *666* | *(rw-rw-rw-)* All users may read and write the file. |
| *644* | *(rw-r--r--)* The owner may read and write a file, while all others may only read the file. A common setting for data files that everybody may read, but only the owner may change. |
| *600*  **Directo** The chm most wa Howeve file listi  *Value 777*  *755*  *700*  **Commo** | *(rw-------)* The owner may read and write a file. All others have no rights. A common setting for data files that the owner wants to keep private.  **ry permissions**  od command can also be used to control the access permissions for directories. In ys, the permissions scheme for directories works the same way as they do with files. r, the execution permission is used in a different way. It provides control for access to ng and other things. Here are some useful settings for directories:  *Meaning*  *(rwxrwxrwx)* No restrictions on permissions. Anybody may list files, create new files in the directory and delete files in the directory. Generally not a good setting.  *(rwxr-xr-x)* The directory owner has full access. All others may list the directory, but cannot create files nor delete them. This setting is common for directories that you wish to share with other users.  *(rwx------)* The directory owner has full access. Nobody else has any rights. This setting is useful for directories that only the owner may use and must be kept private from others.  **n file system commands** |

|  |  |  |
| --- | --- | --- |
| **Command** | **Meaning** |  |
| **cat file(s)** | Send content of file(s) to standard output. |  |
| **chmod *mode* file(s)** | Change access permissions on file(s) |  |
| **cp sourcefile targetfile** | Copy sourcefile to targetfile. |  |
| **echo *string*** | Display a line of text |  |
| **file filename** | Determine file type of filename. |  |
| **locate *searchstring*** | Print all accessible files matching the search pattern. |  |
| **ls file(s)** | Prints directory content. |  |
| **mkdir newdir** | Make a new empty directory. |  |
| **mv oldfile newfile** | Rename or move oldfile. |  |
| **Pwd** | Print the present or current working directory. |  |
| **rm file** | Removes files and directories. |  |
| **rmdir file** | Removes directories. |  |
| **wc file** | Counts lines, words and characters in file. |  |
|  | | |

|  |
| --- |
| **Chapter 2 APACHE**  **What is apache?**  Apache is a remarkable piece of application software. It is the most widely used Web Server application in the world with more than 50% share in the commercial web server market. Apache is the most widely used Web Server application in Unix-like operating systems but can be used on almost all platforms such as Windows, OS X, OS/2, etc. The word, Apache, has been taken from the name of the Native American tribe ‗*Apache’*, famous for its skills in warfare and strategy making.  It is a modular, process-based web server application that creates a new thread with each simultaneous connection. It supports a number of features; many of them are compiled as separate modules and extend its core functionality, and can provide everything from server side programming language support to authentication mechanism. Virtual hosting is one such feature that allows a single Apache Web Server to serve a number of different websites  **What is Virtual Host?**  An Apache web server can host multiple websites on the **SAME** server. You do not need separate server machine and apache software for each website. This can achieved using the concept of **Virtual Host** or **VHost.** Any domain that you want to host on your web server will have a separate entry in apache configuration file.  Types of Apache Virtualhost   1. Name-based Virtual host 2. Address-based or IP based virtual host and. Name-based Virtual Host   Name based virtual hosting is used to host multiple virtual sites on a single IP address.    In order to configure name based virtual hosting, you have to set the IP address on which you are going to receive the Apache requests for all the desired websites. You can do this by NameVirutalHost directive within the apache configuration i.e. **httpd.conf/apache2.conf file.** IP-based Virtual host In order to setup IP based virtual hosting, you need more than one IP address configured on your server. So, the number of vhost apache will depend on number of IP address configured on your server. If your server has 10 IP addresses, you can create 10 IP based virtual hosts. |

|  |
| --- |
| **3.3 MIME types and CGI Files Manipulating the Apache2 HTTPD service**  Starting the Apache2 HTTPD service: #/usr/local/apache2/bin/apachectl start Stopping the Apache2 HTTPD service:  # /usr/local/apache2/bin/apachectl stop  The main configuration file for configuring Apache is httpd.conf, which contains directives written in plain text. The location of this file is set at compile-time. When required this may be overridden with the -f command line flag. The srm.conf and access.conf files can also be used to configure Apache. Other configuration files may be added and their instructions passed to Apache by using the Include directive followed the path that points to where the additional configuration files are. Any directive may be placed in any of these configuration files. Apache is pretty flexible like that. Changes made to the main configuration files are only recognized by Apache when it is Started or restarted. If any configuration file is actually a directory, Apache will enter that directory and parse any files (and sub-directories) found there as configuration files. Apache works best if there is only one configuration file used and all its directives are placed in that file (i:e. httpd.conf). Apache also reads a file containing document mime types. This filename is set by the Types Config directive, and is mime.types by default. Apache configuration files contain one directive per line. If a directive must continue onto the next line use back-slash '\' as the last character on the previous line. Directives in configuration files are case-insensitive, but arguments to directives are often case-sensitive. Any line beginning with a hash (#) character is ignored. Blank lines and white spaces before a directive are ignored. Configuration files can be checked for syntax errors without starting the server by using apachectl configest or the -t command line option. Some important entries in httpd.conf file.  **Global Environment**  Server Type standalone The two server types are standalone and inetd. Usually it is always standalone. Setting it to inetd causes a new server to be started to handle every incoming HTTP request, which will die as soon as the request is served. This makes things slower because of re-reading the configuration file and the overhead of server startup with every request.  Server Root /etc/httpd  This sets the absolute path to the server directory. Generally, the argument 'of Server Root should be the path to where Apache is installed. |

|  |
| --- |
| **3.4 Authentication and log files**  ErrorLog /logs/error\_log  Custom Log /logs/access\_log common  The various directives that end in Log control indicate whether log files exist at all. Directives also indicate exactly where the log files exist in the Linux file system.  LockFile /var/run/httpd.lock  The LockFile directive sets the path to the lockfile used when Apache is compiled. This directive should normally be left at its default value.  PidFile /var/run/httpd.pid  It is a file in which the server should record its process identification number when it starts. ScoreBoard File /logs/apache\_runtime\_status  It is a file used to store internal server process information. Not air architectures require this But if local architecture does (This will be known because this file will be created when Apache run) then ensure that no two invocations of Apache share the same Scoreboardfile. Timeout 300  Indicates the number of seconds before Apach.ereceives and senqs a time out. KeepAlive On  Indicates whether or not to allow persistent connections (more than one request per connection  Set it to Off to deactivate. MaxKeepAliveRequests 100  Indicates the maximum number of requests to allow during a persistent connection. Set it to allow an unlimited amount. It is recommended that this number is set high, for maximum performance.  KeepAliveTimeout 15  Indicates the number of seconds to wait for the next request from the same client on the same connection.  MinSpareServers 5  MaxSpareServers 20  Indicates the Server-pool size regulation. Rather than making a user guess how many server processes are required, Apache dynamically adapts to the load it sees - that is, it tries to maintain enough server processes to handle the current load, plus a few spare servers to handle transient load spikes (e.g., multiple simultaneous requests from a single Netscape browser).  It doesthis by periodically checking how many servers are waiting for a request If there are fewer than MinSpareServers, it creates a new spare. If there are more than MaxSpareServers, some of the spares die off. The default values are probably OK for most sites.  Number of servers to start initially should be a reasonable ballpark figure. StartServers 8  Port 80  This indicates the port on which the server should run on. User and Group  The Web server's user and group values are denoted by the User and Group settings. This should be set to User ID and Group ID that the server will use to process requests. If the server runs as root, some hacker could exploit the privilege. Normally, people want to run Apache as an underprivileged user for security reasons. On Linux, this can be done, by setting both to nobody.  ServerAdmin root@localhost  Accepts the Email address, where problems with the server should be e-mailed. This address appears on some server-generated pages, such as error documents. |

|  |
| --- |
| ServerName localhost  This sets the hostname the server will return. Set the name of the server using the ServerName directive. This is especially useful when the computer has multiple names or IP addresses.  DocumentRoot /var/www/html  This indicates the absolute path of the document tree, which is the top directory from which Apache will serve files. The DocumentRoot is the root of the Web tree and it defaults to  /usr/local/apache2/htdocs. Assuming that Apache is installed in/usr/local/apche2/, this can be changed if required.  5.2 **PHP and the web server architecture model**  The most commonly used framework on the Internet, for building interactive, database driven websites is L.A.M.P.P. as mentioned earlier this is an acronym for Linux, Apache, MySQL, PHP and PERL. Here the operating system of the framework is Linux. Common flavors being RedHat, Mandrake, SuSE, Debian and so on. The Web server is Apache. MySQL is a production quality, Linux based, RDBMS in which user information is stored and finally, PHP and PERL are the programming  Environment of choice that acts as a go between Apache Web server and the MySQL database engine, which protects user information.  Apache is the Web server responsible for responding to requests received from cIient browsers for information. MySQL is the database in which such information is stored. PHP and PERL are the middleware, programming environment of choice that can:   1. Respond to such information requests being processed by the Web server Apache 2. Access the MySQL database tables where the information requested is stored 3. Convert this to HTML 4. Return this HTML to the client browser via Apache Web server Decomposing the server side architecture.     Now that the request/response paradigm of the Internet and the framework on which this paradigm can be implemented is known, it is necessary to actually create such a framework on a Linux box to work on. |

|  |
| --- |
| 5.3 **Over view of PHP capabilities**  This will involve the installation (and configuration, where applicable) of: Linux as an O/S. (RedHat Linux 9 is the 0/5 of choice)   * ApacheWeb Server * MySQL * PHP * PERL   Thus servicing the client's request for information. All this is because a Web server (Apache2) cannot communicate directly with a database management system (MySQL) hence PHP and PERL program codes plays the role of mediator.  5.4 **CGI vs Shared object Model**  Common Gateway Interface (CGI) is a basic way to create dynamic web pages. CGI is a standard for communication between a client and the server. CGI scripts can be written in almost any language. Perl is well suited to the types of text processing common for many tasks, such as search engines and forms interfaces. Other benefits of Perl include portability, ease of programming and overall computational power and performance. And to top it off, the Perl module CGI.pm is a useful way to make Perl CGI script writing quick and easy.  CGI scripts can do simple things that require no input from the client, such as displaying the current time or a random banner when a web page is accessed. Or they can do more complicated tasks involving posted form data from the client, such as entering a credit card number, searching a database and returning the information, and filling out a form.  Figure shown below depicts what happens during the request and execution of a CGI program. The web server recognizes a CGI request by the location of the thing requested (or by the file name extension). For instance, if we load the URL [*www.example.com/cgi-bin/a.cgi*](http://www.example.com/cgi-bin/a.cgi)into the browser, the web server contacted, [www.example.com.](http://www.example.com/) receives a request such as the following:  GET /cgi-bin/a.cgi HTTP/1.0  The server notices that the directory that contains the thing requested is cgi -bin. It is configured to take the object requested, here a. cgi, which is a program located on the server, and execute it as a stand-alone program. The program generates standard output (in Perl, we would use print()). This output is in an important format: a header, a blank line, and the body.    Figure: CGI model  The header is a very important piece of information that is sent back to the browser because it tells the browser how to render the data that follows. The primary piece of information that is sent in the header is the Content- l2e. If the header contains content-type: text/plain, the browser displays the data that follows as plain text. If the header contains Content type: |

|  |
| --- |
| text/html, the browser treats the data that follows as HTML and renders it appropriately. And this is what is really important: Programs must output the header, then a blank line, and then the content to be displayed. The blank line is essential-it tells the browser that the header is complete and the body is about to begin. It is easy to see the header, blank line, and body output from a CGI program by using a shell and telnetting to a server. The following code is an example of connecting to a CGI program named test. cgi, which simply prints the content type, the blank line, and some important text:  $ telnet www.not\_a\_real\_web\_server.com 80 Trying 1.299.299.1  Connected to www.not\_a\_real\_web\_server.com (1.299.299.1) Escape character is '-'.  GET /cgi-bin/test.cgi HTTP/l.0  HTTP/I.1 200 OK  Date: Thu, 17 Jan 2002 19:57:05 GMT  Server: Acme Web Server Version 0.001b  Connection: close Content-Type: text/plain  There's more than one way to do it. Connection closed by foreign host.  When the server accepts the connection, it tells the client so. Then we see the HTTP request: followed by a blank line. The webserver prints some header stuff, including the content type that the program prints, followed by a blank line that the program prints, followed by an important philosophical assertion in the next-to-last line. Then, had we used a browser instead of a telnet session ,it would have taken the information in the header and the body and rendered it appropriately based on the content type in the header. Again, that blank line that separates the header and the body is important. If it is not present, a server error will result. |

|  |
| --- |
| **Multiple Choice Questions**  **1.** The programming interface to the kernel is included in which subsystem of operating system.   1. User Applications 2. **O/S Services** 3. Linux Kernel 4. Hardware Controllers   2. Memory hardware is an example of which subsystem of operating system   1. User Applications 2. O/S Services 3. Linux Kernel 4. **Hardware Controllers**   3. Give the fullform of VFS   1. [**Virtual File System**](http://oss.org.cn/ossdocs/linux/kernel/a1/index.html#Toc_3_3) 2. Visual File System 3. Virus File System 4. Valid File System   4. Full form of IPC   1. [**Inter-Process Communication**](http://oss.org.cn/ossdocs/linux/kernel/a1/index.html#Toc_3_5) 2. Intra -Process Communication 3. [Inter-Process Command](http://oss.org.cn/ossdocs/linux/kernel/a1/index.html#Toc_3_5) 4. Intra- Process Command   5. Which of the following is responsible for controlling process access to the CPU   1. [Memory Manager](http://oss.org.cn/ossdocs/linux/kernel/a1/index.html#Toc_3_2) 2. [Inter-Process Communication](http://oss.org.cn/ossdocs/linux/kernel/a1/index.html#Toc_3_5) 3. [Virtual File System](http://oss.org.cn/ossdocs/linux/kernel/a1/index.html#Toc_3_3) 4. [**Process Scheduler**](http://oss.org.cn/ossdocs/linux/kernel/a1/index.html#Toc_3_1)   6. Give the fullform of PID   1. **P**[**rocess identification number**](http://www.linfo.org/pid.html) 2. Page identification number 3. [Process identical number](http://www.linfo.org/pid.html) 4. Program identical name   7. Special files are in which folder   1. **/dev** 2. /bin 3. /lib 4. /tmp 5. Which of the following value represent ―No restrictions on permissions‖ on **files a) *777***   b) *755*  c) *700*  d) *666*   1. Which of the following value represent ―The file's owner may read, write, and execute the **file**‖.   a) *777*  b) *755*  **c) *700***  d) *666*   1. Which of the following value represent ―The directory owner has full access. Nobody else has any rights.‖ |

|  |
| --- |
| a) *777*  b) *755*  **c) *700***  d) *666*  11. Which of the following command is used to ―Prints directory content‖.   1. Cat 2. chmod 3. **ls** 4. mkdir   12. Which of the following command is used Removes directories.   1. Cat 2. chmod 3. **rmdir** 4. rm   13. Which of the following has extra and third party software.   1. **/opt** 2. /root 3. /sbin 4. /tmp   **14.** Which directory is used for miscellaneous purposes.   1. **/misc** 2. /root 3. /sbin 4. /tmp 5. How is rwx represented in binary form   a) 111 111 111  b) 110 110 110  c) **111 000 000**  d) 000 000 000   1. The main configuration file for configuring Apache is The main configuration file for configuring Apache is 2. **httpd.conf** 3. srm.conf 4. access.conf 5. apache.conf   17. . If a directive must continue onto the next line which of the following must be used as the last character on the previous line   1. **back-slash '\'** 2. colon (:) 3. semicolon (;) 4. underscore (\_)   **18.** Any line beginning with a character is ignored   1. **hash (#)** 2. colon (:) 3. semicolon (;) 4. underscore (\_)   19. Maximum number of requests to allow during a persistent connection is indicated via   1. **MaxKeepAliveRequests** 2. KeepAliveTimeout 3. MinSpareServers |

|  |
| --- |
| **d)** MaxSpareServers  20. Full form of CGI   1. Correct Gateway Interface 2. **Common Gateway Interface** 3. Common Gateway Interconnect 4. Correct Gateway Interconnect   21. IP based Virtual host is also called as   1. Name based Virtual host 2. **Address-based Virtual host** 3. File based virtual host 4. Time based virtual host 5. If your server has 10 IP addresses, how many IP based virtual hosts can be created?   **a) 10**  b) 20  c) 30  d) 40   1. Which of the following indicates whether or not to allow persistent connections 2. **KeepAlive** 3. MinSpareServers 4. MaxSpareServers 5. MaxKeepAliveRequests   24. In which of the following file the server should record its process identification number   1. LockFile 2. **PidFile** 3. ScoreBoard File 4. StartServers   25. Which of the following file is used to store internal server process information   1. LockFile 2. PidFile 3. **ScoreBoard File** 4. StartServers   **Long Answer Questions**   1. Explain the architecture of linux operating system 2. Briefly explain different subsystem of linux kernel 3. Describe the 2 distinct region of system memory 4. What are different file system permission 5. What is the different setting for directory permissions? 6. List and explain 8 common filesystem commands 7. Write a note on Manipulating the Apache2 HTTPD service 8. What is virtual host, Differentiate IP based Virtual Hosts and Name based Virtual Hosts 9. Write a note on PHP and the web server architecture model 10. Describe the different authentication and log files 11. Explain how the Web server's user and group values are denoted. 12. Write a note on CGI Model. |

|  |
| --- |
| **UNIT II CHAPTER 3 MYSQL**  **Introduction**  WHAT IS A DATABASE?  Databases have terminology all their own. Several terms are also used interchangeably:  ‰ Database: A database is an organized collection of data. In MySQL you often create separate databases for each of your projects. ‰ Table: A table is a collection of similar information. In MySQL you might have a Customers table that contains data about your customers, a Products table that has data about your products, an Order Headers table that contains header and totals about your orders, and an Order Details table that contains the line items on the orders. ‰ Row: Inside a table, you have rows. Each row is a related set of data. In the Customers table, each customer is in a row.  ‰ Record: A record is another word for a row.  ‰ Column: Inside your table you also have columns. Columns are the types of information you are storing in your table. For instance in the Customers table, name, street, and city would all be columns.  ‰ Field: A ﬁeld is another word for a column. Sometimes used to refer to a speciﬁc row‘s column.  ‰ Value: A value is what is in a given cell. In the Customers table, for instance, you would have a row for George Smith where the value of the cell in the name column is ―George Smith.‖  Relationship: A relationship is a link between two tables.  %For instance, an Order Details table would link to the Order Headers so that you can associate the line items with the correct order. ‰ Key: A key is a ﬁeld, or ﬁelds, that link the tables. In the Order Details table you have an order number ﬁeld that matches an order number ﬁeld in the Order Headers table. The order number ﬁeld is a key or key ﬁeld.  ‰ Index: An index is an internal system that a database system uses to locate information more quickly. In MySQL you can specify that certain columns, usually keys, are indexes.  6.3 **MYSQL administration Commands-**  **$mysql –u root**  This is a command used to connect to the MYSQL as a root user. This root user is different from linux root user. Using this connection the administrator has all the privileges in the MYSQL. He can create or modify the databases, users, various privileges to the users etc.  In order to connect to MYSQL first of all the service should be started. The command to start the MYSQL service is  # chkconfig mysqld on # /etc/init.d/mysqld start  Once the root user is connected the prompt is shown as follows Mysql> |

|  |
| --- |
| **THE SHOW DATABASE AND CREATE DATABASE COMMAND**  To show the database created inside MYSQL the show command is used. Syntax of the same is as follows  Mysql> SHOW DATABASES;  This query will show all the databases present inside. To create a databse the following command is used Mysql> CREATE DATABASE *name of the database*  SQL commands and sub commands (in the previous example, CREATE is a command; DATABASE is its subcommand) are case-insensitive. The name of the database (and table and field) are case sensitive. It's a matter of style whether one uses uppercase or lowercase, but traditionally the SQL commands are distinguished by uppercase.  One way to think of a database is as a container for related tables. A table is a collection of rows, each row holding data for one record. Each record containing chunks of information called fields.  **USE Command**  Before anything can be done with the newly created database, MySQL has to connect to it. That‘s done with the USE command.  Mysql>USE name of the database;  **CREATE TABLE and SHOW TABLE Command**  Each table within the database must be defined and created. This is done with the CREATE TABLE command.  Create a table named age\_information to contain an individual is first name, last name, and age. MySQL needs to know what kind of data can be stored in these fields. In this case, the first name and the last name are character strings of up to 20 characters each, and the age is an integer:  The Syntax of the same is :  **mysql> CREATE TABLE age\_information (**  **-> lastname CHAR(20),**  **-> firstname CHAR(20),**  **-> age INT**  **-> );**  **Query OK, 0 rows affected (0.00 see)**  It appears that the table was created properly (it says OK after all), but this can be checked by executing the SHOWTABLES command. If an error is made, the table can be removed with DROP TABLE.  When a database in MySQL is created, a directory is created with the same name as the database (people, in this example):  SHOW TABLES is used to show all the tables present in the database. The syntax of the same is  **Mysql> SHOW TABLES;**  **The DESCRIBE Command** |

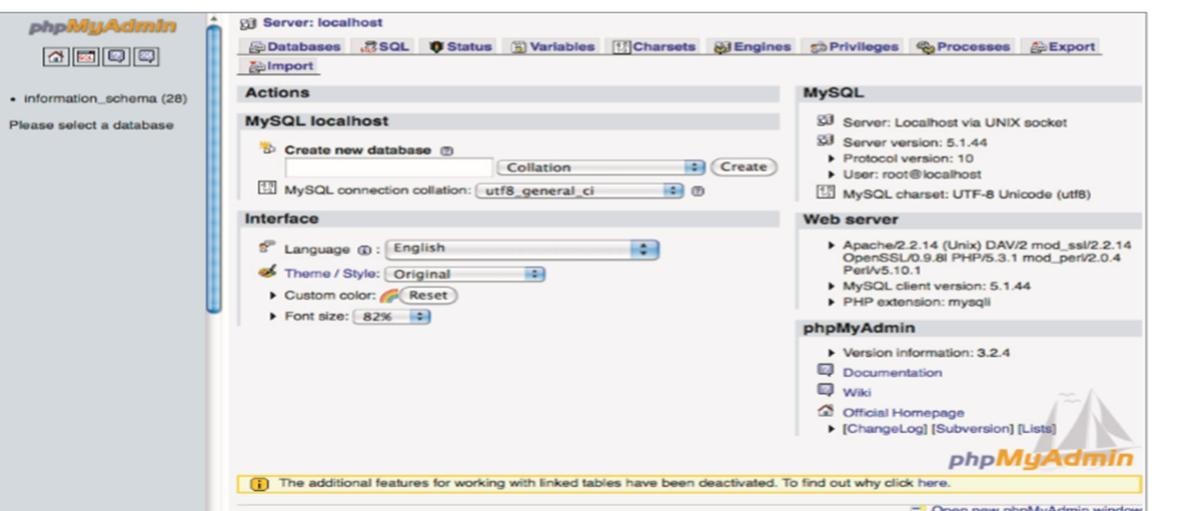
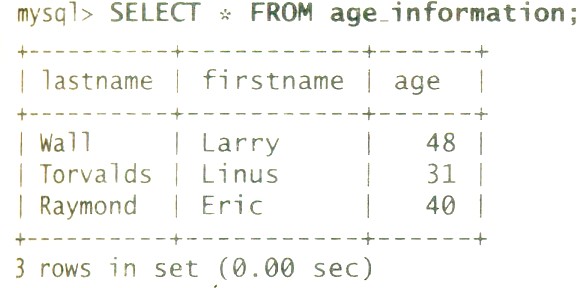
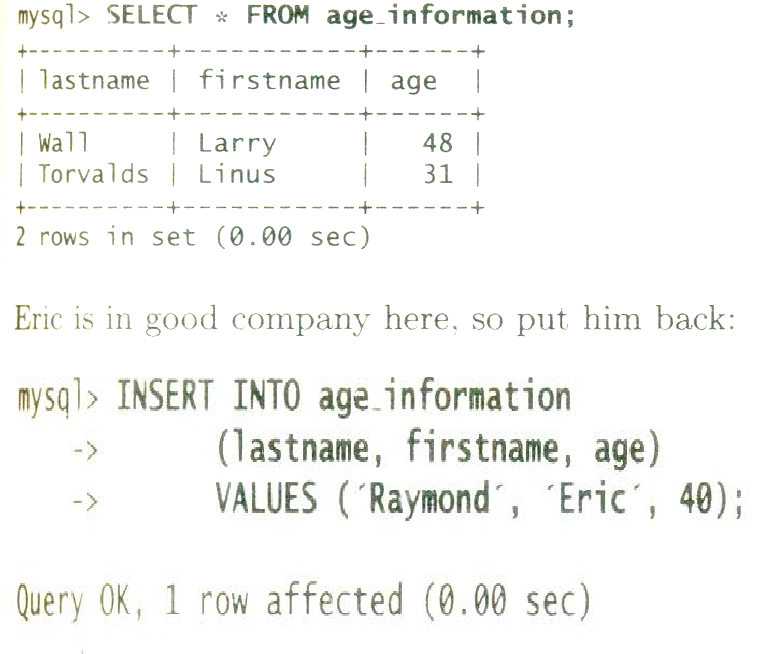
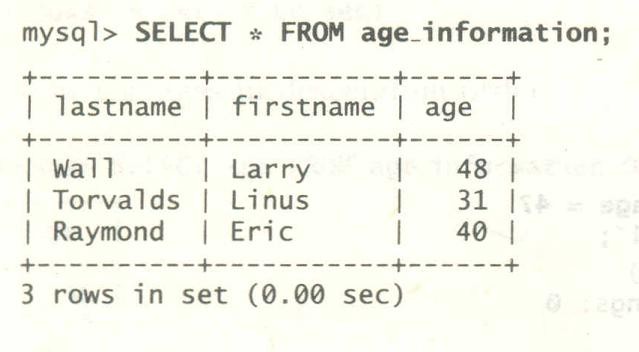
|  |
| --- |
| The DESCRIBE command gives information about the fields in a table. The fields created earlier-lastname, firstname, and age-appear to have been created correctly.    The command SHOW COLUMNS FROM age\_information· gives the same information as DESCRIBE age\_ information; but DESCRIBE involves less typing.  **INSERT Command**  This command is used to insert the records into the table created using CREATE command. The syntax of the same is  Mysql> INSERT INTO name of the table VALUES (the necessary values); mysql> INSERT INTO age\_information  -> (1astname , firstname, age)  -> VALUES ('Wall', 'Larry', 46);  Query OK, 1 row affected (0.00 sec)  The syntax of the command is INSERT INTO, followed by the table in which to insert, a list within parentheses of the fields into which information is to be inserted, and the qualifier VALUES followed by the list of values in parentheses in the same order as the respective fields.  **SELECT command**  SELECT selects records from the database. When this command is executed from the command line, MySQL prints all the records that match the query.  The simplest use of SELECT is shown in this example    The \* means "show values for all fields in the table"; FROM specifies the table from which to extract the information.  The previous output shows that the record for Larry Wall was added successfully. There are many ways to use the SELECT command-it's very flexible.  First, the table based on 1astname: |

|  |
| --- |
| **The UPDATE Command**  Since the database is about people, information in it can change (people are unpredictable like that). For instance, although a person's birthday is static, their age changes. To change the value in an existing record, we can UPDATE the table. Let's say the fictional Larry Wall has turned 47: |

|  |
| --- |
| Be sure to use that WHERE clause; otherwise if we had only entered UPDATE age\_information SET age = 47, all the records in the database would have been given the age of 47!  Although this might be good news for some people in these records (how often have the old- timers said "Oh, to be 47 years old again"-OK, probably not), it might be shocking news to others.  This method works, but it requires the database to know that Larry is 46, turning 47. Instead of keeping track of this, for Larry's next birthday we simply increment his age: |

Follow these steps as illustrated in Figure 19-5 to create a database called test

LAMP TECHNOLOGY 25



## The DELETE Command

Sometimes we need to delete a record from the table (don't assume the worst-perhaps the person just asked to be removed from a mailing list, which was opt-in in the first place, of course). This is done with the DELETE command:

mysql> DELETE FROM age\_information WHERE lastname = Raymond'; Query OK, 1 row affected (0.00 sec)

Creating Database

|  |
| --- |
| 1. Type test in the input box labeled Create New Database. (If you do not see that input box, click the house icon to return to the Home page ﬁrst.) 2. Click the down arrow in Collation and select utf8\_general\_ci. 3. Click the Create button.   Next comes a list of the tables in this database. There are no tables yet, so you need to create one now:   1. Enter table1 as the name. 2. Enter 2 for the number of ﬁelds (columns) to create. 3. Click the Go button and a window similar to Figure 19-7 is presented. 4. Fill in the ﬁeld ID by giving it the name id, selecting PRIMARY from the Index drop- down, and clicking the AUTO\_INCREMENT checkbox. Selecting Primary for the index tells MySQL that this is a main ﬁeld that is used to identify records in the table. MySQL creates an index for the ﬁeld in order to retrieve the data more quickly. Flagging the ﬁeld as auto\_ increment means that MySQL automatically creates a unique sequential number in this ﬁeld when a record is added. 5. Fill in the ﬁeld description by giving it the name description, changing the Type drop-down to TEXT, and clicking the Null checkbox to allow nulls to exist. Allowing nulls to exist means that the ﬁeld is not required. Your window looks like Figure 19-8. 6. Click the Save button to add these two columns to table1 as shown in Figure 19-9. |

|  |
| --- |
| The table table1 is now listed in the left column. To work with a speciﬁ c table, click that table on the left. If there are records in the table, the Browse tab is activated; otherwise, as in this case, the Structure tab is active as it was in Figure 19-7. You add or change columns (not the data in the columns) in the Structure tab. You can add a column either at the end of the table, at the beginning of the table, or after a given column as shown in Figure 19-10.  After you click the Go button, the window where you create your new column displays as shown in Figure 19-11. This example creates an integer column called code, where any new rows default to 42 if not set to a different number. |

|  |  |  |
| --- | --- | --- |
|  | | |
|  | |  |
| Clicking the Save button adds the column to table1 as shown in Figure 19-12. | | |
|  |  | |
| **Entering Data**  You now have a database called test, which contains a single table called table1, which has three ﬁelds: id, description, and code. At this point you can enter data into your database. To do so, be sure that the table is selected either by clicking the table name on the left or checking the breadcrumb at the top of the window. Clicking the Insert tab opens a window with forms to enter two records. Figure 19-13 shows the forms ﬁlled in and ready to be saved. The ﬁrst ﬁeld is id, which is the primary key that is ﬂagged as an auto\_increment ﬁeld. When left blank, MySQL automatically assigns the next number in sequence. The next ﬁeld is description, which is a text ﬁeld where text can be entered. The code ﬁeld displays the default of 42 to start, but can that can be changed to a different number. To save both the records, click the Go button in either form. The program jumps to the SQL tab where it displays a status message and the SQL command used to add the records as shown in Figure 19-14. Now that you have records in the table, clicking the Browse tab displays those records as shown in Figure 19-15. To delete the records in a table, but leave the structure intact, click the Empty tab. You are asked if you want to TRUNCATE TABLE. Click the OK button to continue with the deletion of the records. To delete the entire table, including the structure, click the Drop tab. You are asked if you want to DROP TABLE. Click the OK button to continue to delete the table completely. | | |

|  |
| --- |
| **UNDERSTANDING DATA TYPES**  Just as in PHP, MySQL has different data types for the ﬁelds. The data types in MySQL are stricter than in PHP and there is not a one-to-one correlation.  **Strings**  There are two types of strings in MySQL. The ﬁrst is text strings, which have character sets and collations. This is the type of string that you use most often. Text strings are further deﬁned as follows:  ‰ CHAR: This is the character data type. You deﬁne exactly how many characters are stored. For example, if you want a ﬁeld to be exactly six characters long, you deﬁne it as CHAR(6). If you pass it data that is less than that, it pads with spaces at the end. If you pass it more, the extra characters are truncated. Whether you are truncating blanks or non-blank characters and what error reporting you have set determines what, if any, errors you see. You can go all the |

|  |
| --- |
| way up to 255 characters. Note that some character sets require more than 1 byte to store some characters. The size limits for the text strings are based on the number of characters, not the number of bytes. Trailing spaces are removed when you retrieve the data.  ‰ VARCHAR: This data type has a variable number of characters. You specify the maximum number of characters, up to 65,535. If you have a ﬁ eld that could contain up to 50 characters but would likely contain less, you deﬁ ne it as VARCHAR(50). There is a little overhead when using VARCHAR rather than CHAR because 1 or 2 bytes are used to store the length. Trailing spaces are not removed when you retrieve the data.  ‰ TEXT: There are four TEXT types — TINYTEXT, TEXT, MEDIUMTEXT, and LONGTEXT. Like VARCHAR, the TEXT types contain a variable number of characters. The difference between the four types is the maximum number of characters. The type deﬁ nes the maximum number of characters; you do not. See Table 21-1.  TABLE 21-1:  TEXT Type Sizes  TEXT TYPE MAXIMUM CHARACTERS TINYTEXT 255  TEXT 64K  MEDIUMTEXT 16M  LONGTEXT 4G  The second type of string is binary strings, which have no character sets or collations. Character strings contain text, whereas binary strings contain raw data such as images and other media. The binary types are subdivided in the same way that the text strings are, but the size limits are based on the number of bytes, not the number of characters. ‰ BINARY: This is the binary data type. You deﬁne exactly how many bytes are stored. For example, if you want a ﬁ eld to be exactly 6 bytes long, you deﬁne it as BINARY(6). You can go all the way up to 255 bytes. ‰ VARBINARY: This data type has a variable number of bytes. You specify the maximum number of bytes, up to 65,535. If you have a ﬁeld that could contain up to 50 bytes but would likely contain less, you deﬁne it as VARBINARY(50). There is a little overhead when using VARBINARY rather than CHAR because 1 or 2 bytes are used to store the length. ‰ BLOB: There are four BLOB types — TINYBLOB, BLOB, MEDIUMBLOB, and LONGBLOB. Like VARBINARY, the BLOB types contain a variable number of bytes. The difference between the four types is the maximum number of bytes. The type deﬁnes the maximum number of bytes; you do not. See Table 21-2.  TABLE 21-2: BLOB Type Sizes  BLOB TYPE MAXIMUM BYTES  TINYBLOB 255  BLOB 64K  MEDIUMBLOB 16M  LONGBLOB 4G  **Numeric**  As in PHP, numbers that do not have a decimal point are integers. MySQL has different integer types that are based on the size of the integer. Additionally, if the integer is SIGNED  — that is, has both negative and positive values — the range starts in the negative numbers. If |

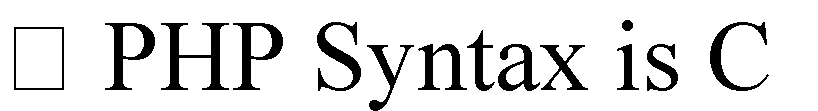
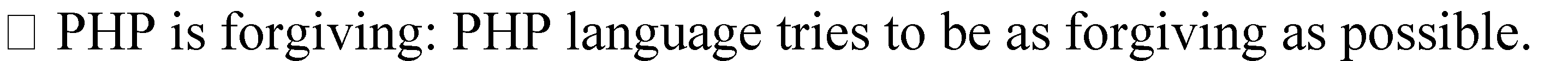
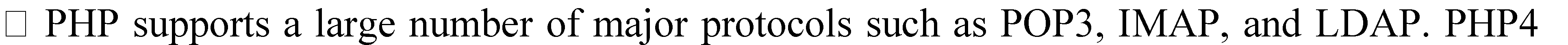
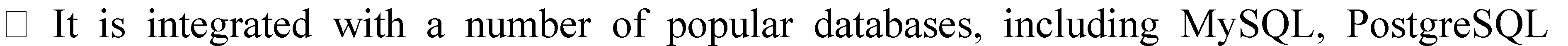
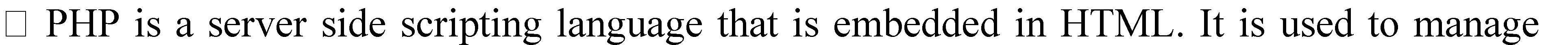
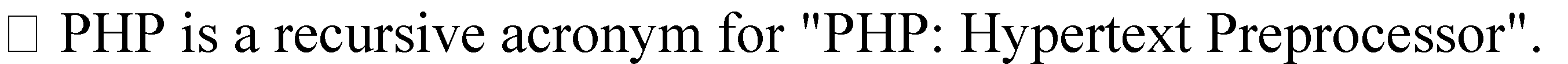
|  |
| --- |
| the ﬁeld is ﬂagged as UNSIGNED, the values start at 0 and go twice as high, as you see in Table 21-3.  TABLE 21-3: Integer Types  INTEGER TYPE RANGE IF SIGNED RANGE IF UNSIGNED  TINYINT –128 to 127 0 to 255  SMALLINT –32,768 to 32,767 0 to 65,535  MEDIUMINT –8,388,608 to 8,388,607 0 to 16,777,215  INT (or INTEGER) –2,147,483,648 to 2,147,483,647 0 to 4,294,967,295  BIGINT –9,223,372,036,854,775,808 0 to 18,446,744,073,709,551,615  to 9,223,372,036,854,775,807  You may see these types written with a length such as TINYINT(1) or TINYINT(4). This refers to the number of digits to be displayed. It does not affect the value that is stored or the space needed to store the value. If you need decimals you use either a ﬂoating-point data type or a ﬁxed-point data type. The ﬂoating-point types are similar to PHP ﬂoating-point types. FLOAT uses 4 bytes of storage and DOUBLE (also called DOUBLE PRECISION or REAL) takes 8. MySQL allows you to specify the total number of digits and the number of digits after the decimal point. So to specify a number between –999.9999 and 999.9999 you use FLOAT(7,4). MySQL rounds the decimal when storing it rather than truncating it if it is too long.  **Date and Time**  MySQL stores dates and times in the format of YYYY-MM-DD HH:MM:SS, unlike PHP. Your MySQL server dictates where you can store invalid dates or whether all invalid dates should be converted to zeros.  ‰ DATETIME contains the date and the time. It has a range from the year 1000 through the year 9999.  ‰ DATE contains just the date value.  ‰ You can use TIMESTAMP to automatically contain the initial value or automatically update when something changes on the row. It has a range from 1970 through early 2038. It stores all values as of the UTC time zone.  ‰ TIME displays the time portion of a date or an elapsed time.  ‰ YEAR displays the year. It can be either YEAR(2) or YEAR(4) for two- or four-digit representation of the year. It has a range from the year 1901 through 2155. Two digits between 00 and 69 are converted to 2000 through 2069 and 70 to 99 are converted to 1970  through 1999.  Other Data Types MySQL has a data type ENUM that restricts the ﬁeld to values from an enumerated list of values. Although you can use numbers as values, it is not recommended because errors can easily occur. Numbers can be misinterpreted as an index of a value instead of the value itself.  ENUM(‗small‘, ‗medium‘, ‗large‘)  Although you can put your business logic here, if there‘s any chance it might change, it would be better to put the value checking in your program where it would be easier to make changes. MySQL has two other data types that you may come across:  ‰ SET includes zero or more values from a deﬁned list.  ‰ BIT stores data at the bit level using binary values. |

|  |
| --- |
| **USING AUTO\_INCREMENT**  The primary key for a table should have the following characteristics:  ‰ Unique  ‰ Not Null  ‰ Not optional  ‰ Never needs to be changed  ‰ Does not violate security policies  In addition, a short simple key that can be retrieved quickly helps performance. It can be difﬁcult to ﬁnd a data ﬁeld that meets all of these requirements. For that reason, tables are often given artiﬁcial keys — arbitrary keys that have no meaning other than to be a primary key. MySQL supports this policy with the AUTO\_INCREMENT attribute. You assign this attribute to a ﬁeld and MySQL generates a unique sequential number for each new row. You can assign AUTO\_ INCREMENT to either an integer or a ﬂoating-point data type, though an integer is the most common. Make sure that the data type you choose is large enough to hold the highest number you need. The following snippet of code shows the typical speciﬁcations for an artiﬁcial primary key. The name of the ﬁeld is id; it is an integer data type that is unsigned, is a required ﬁeld, will be automatically ﬁlled by MySQL, and is assigned as the primary key.  `id` INT UNSIGNED NOT NULL AUTO\_INCREMENT PRIMARY KEY,  The table keeps track of the next number to be assigned. It starts with 1 unless you tell it differently when you create the table. In Lesson 22, you learn how to add data to the tables. When you add the data, if you do not assign a value to id for new rows, or you assign a NULL or 0, a value is automatically assigned. MySQL has a function, LAST\_INSERT\_ID(), that contains the last AUTO\_INCREMENT value. PHP also has a function that can retrieve this number if you need it. |

**Introduction:**

# CHAPTER 4 INTRODUCTION TO PHP

PHP started out as a small open source project that evolved as more and more people found out how useful it was. Rasmus Lerdorf unleashed the first version of PHP way back in 1994.



dynamic content, databases, session tracking, even build entire e-commerce sites.

,

Oracle, Sybase, Informix, and Microsoft SQL Server.

added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.

-Like.

## Features of PHP

The basic features of PHP can be divided into 5 major sections:

## Performance

**Open Source Software Platform Independent Compatibility Embedded**

There are given many features of PHP.

* **Performance**: Script written in PHP executes much faster than those scripts written in other languages such as JSP & ASP.
* **Open Source Software**: PHP source code is free available on the web, you can developed all the version of PHP according to your requirement without paying any cost.
* **Platform Independent**: PHP are available for WINDOWS, MAC, LINUX & UNIX operating system. A PHP application developed in one OS can be easily executed in other OS also.
* **Compatibility**: PHP is compatible with almost all local servers used today like Apache, IIS etc.
* **Embedded**: PHP code can be easily embedded within HTML tags and script.

## PHP coding

**Escaping from HTML:**

Here, a file is analysed and simply parsed until a special tag is reached. The entire text is interpreted as PHP code.

## Instruction separation:

The instructions in PHP code are separated exactly the same way as in Perl or C. Each statement is terminated with a semicolon. In PHP, the closing tag suggests the end of a statement. The syntax is as follows:

<?php

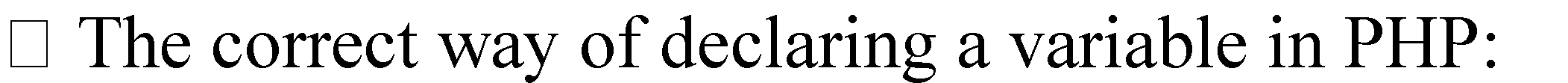
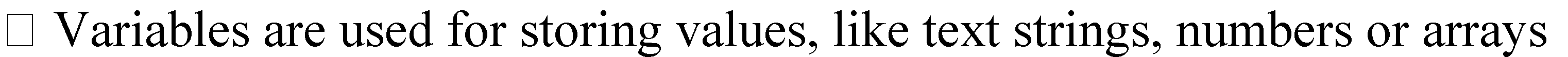
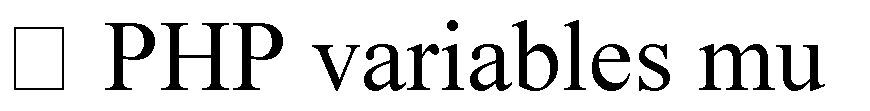
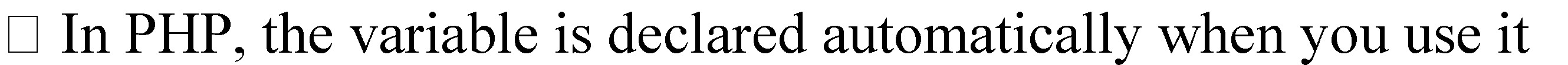
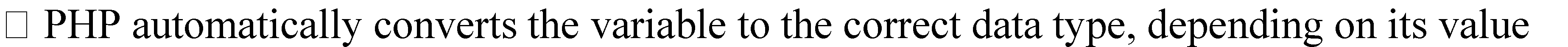
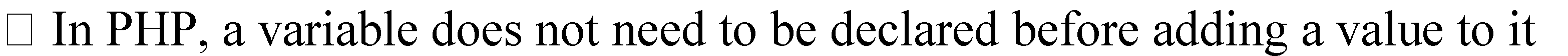
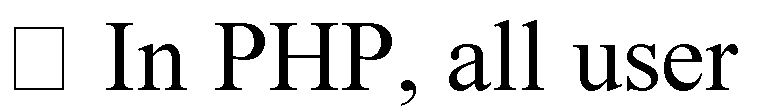
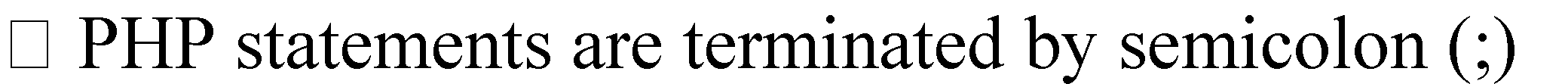
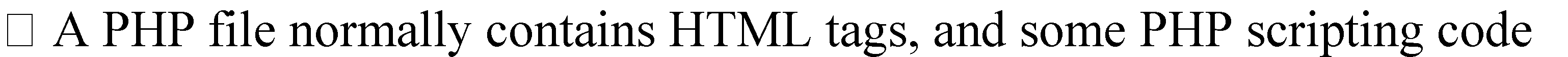
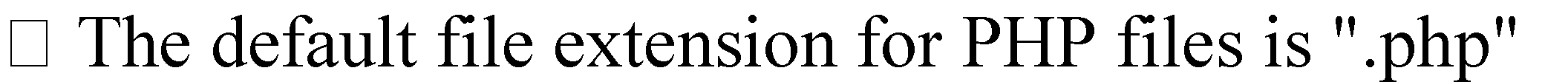
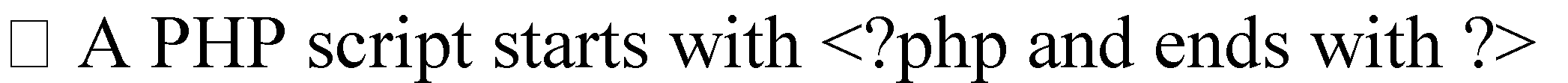
echo ―This is used for testing‖;

?>

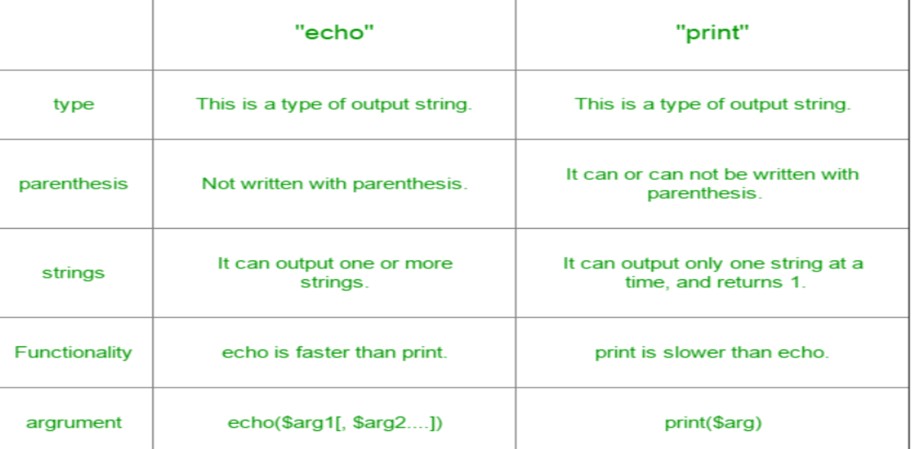
## Comments:

In PHP, C and C++ style comments are supported along with Unix shell like comments. The comment comes at the end of a line or in a PHP block of code.

## Basic Syntax



|  |
| --- |
| -defined functions, classes, and keywords (e.g. if, else, while, echo, etc.) are  not case-sensitive  You can also begin a scripting block with (<?) and end with (?>). This is just a shortened version. It is always advisable to use the standard form of (<?php) in place of the shortened form (<?) as the former is clearer and generally supported.  Let‘s see how we can declare some PHP code: We can declare PHP code in three different forms:  ― <?  PHP Here we insert PHP codes  ?>  ― <?php  PHP Here we insert PHP codes php  ?>  ― <script language=‖php‖> PHP Here we insert PHP codes  </script>  As in an HTML file, PHP files also have HTML tags in addition to some PHP script code.  Some important examples are given as  below using the text string ―Hello World‖ and sending it to the browser.  <html>  <body>  <?php echo ―Hello World‖; ?>  </body>  </html>  Here, each line of PHP code ends with a semicolon. This semicolon actually acts as a separator between two sets of instructions. Echo and Print are the two basic statements available to output text with PHP. Since, PHP scripts are basically embedded in an HTML document, you have the freedom to shift between HTML and PHP.  **PHP is a Loosely Typed Language**  st begin with a ―$‖ sign  $var\_name = value;  **PHP | echo and print**  There are two basic ways to get output in PHP:-   1. echo 2. print   **PHP echo statement**  In PHP ‗echo‘ statement is a language construct and not a function, so it can be used without paranthesis. But we are allowed to use paranthesis with echo statement when we are using more than one arguments with it. The end of echo statement is identified by the semi-colon (‗;‘). |



We can use ‗echo‘ to output strings or variables. Below are some of the usage of echo statement in PHP:

**Displaying Strings**: We can simply use the keyword echo followed by the string to be displayed within quotes. Below example shows how to display strings with PHP:

<?php

echo "Hello,This is a display string example!";

?>

Output:

Hello,This is a display string example! **Displaying Strings as multiple arguments**: We can pass multiple string arguments to the echo statement instead of single string argument, separating them by comma (‗,‘) operator. For example, if we have two strings say ―Hello‖ and ―World‖ then we can pass them as (―Hello‖,‖World‖). Below example shows how to do this:

<?php

echo "Multiple ","argument ","string!";

?>

## PHP print statement

The PHP **print** statement is similar to the echo statement and can be used alteranative to echo at many times.It is also language construct and so we may not use parenthesis : print or print(). The main difference between the **print** and **echo** statement is that print statement can have only one agrument at a time and thus can print a single string. Also, print statement always returns a value 1.Like echo, print statement can also be used to print strings and variables. Below are some examples of using print statement in PHP:

* **Displaying String of Text**: We can display strings with print statement in the same way we did with echo statements. The only difference is we can not display multiple strings separated by comma(,) with a single print statement. Below example shows how to display strings with the help of PHP print statement:-

<?php

print "Hello, world!";

?>

* Output:
* Hello, world!

## Comparison between Echo and Print in PHP:

LA 35

MP TECHNOLOGY

|  |
| --- |
| **Using Comments in PHP**  Comments in PHP are similar to comments that are used in HTML. The PHP comment syntax always begins with a special character sequence and all text that appears between the start of the comment and the end will be ignored by the browser. In HTML a comment's main purpose is to serve as a note to you, the web developer or to others who may view your website's source code. However, PHP's comments are different in that they will not be displayed to your visitors. The only way to view PHP comments is to open the PHP file for editing. This makes PHP comments only useful to PHP programmers. In case you forgot what an HTML comment looked like, see our example below. HTML Code:  <!--- This is an HTML Comment -->  **PHP Comment Syntax:**  **Single Line Comment**  While there is only one type of comment in HTML, PHP has two types. The first type we will discuss is the single line comment. The single line comment tells the interpreter to ignore everything that occurs on that line to the right of the comment. To do a single line comment type "//" and all text to the right will be ignored by PHP interpreter. PHP Code:  <?php  echo "Hello World!"; // This will print out Hello World!  echo "<br />Psst...You can't see my PHP comments!"; // echo "nothing";  // echo "My name is Humperdinkle!"; ?>  **Display**:  Hello World!  Psst...You can't see my PHP comments!  Notice that a couple of our echo statements were not evaluated because we commented them out with the single line comment. This type of line commenting is often used for quick notes about complex and confusing code or to temporarily remove a line of PHP code.  **PHP Comment Syntax:**  **Multiple Line Comment**  Similiar to the HTML comment, the multi-line PHP comment can be used to comment out large blocks of code or writing multiple line comments. The multiple line PHP comment begins with " /\* " and ends with " \*/ ".  **PHP Code:**  <?php  /\* This Echo statement will print out my message to the place in which I reside on. In other words, the World. \*/  echo "Hello World!";  /\* echo "My name is Humperdinkle!";  echo "No way! My name is Uber PHP Programmer!"; \*/  ?>  **Display:**  Hello World!  **Good Commenting Practices**  One of the best commenting practices that I can recommend to new PHP programmers is....USE THEM!! So many people write complex PHP code and are either too lazy to write good comments or believe the commenting is not needed. However, do you really believe that |

|  |
| --- |
| you will remember exactly what you were thinking when looking at this code a year or more down the road? Let the comments permeate your code and you will be a happier PHPer in the future. Use single line comments for quick notes about a tricky part in your code and use multiple line comments when you need to describe something in greater depth than a simple note.  **Benefits of PHP**  PHP has several important benefits such as:   * It is not restricted to HTML output * It provides cross-platform functionality * PHP converses with several network protocols * It is compatible with a wide variety of databases * Strong text processing facilities are available * It supports most current web servers   PHP can be used effectively on different operating systems such as Linux, Microsoft Windows, many Unix variants (like Solaris, HP-UX and OpenBSD), Mac OS X, RISC OS, and many others. As discussed earlier, in the present scenario, PHP supports most web servers. It works as a CGI processor in servers supporting the CGI standard. Each PHP script remains enclosed between two PHP tags commanding the server to recognise the information as PHP.  As PHP is a server-side language, its scripts only run on the operating web server. They never run in the user‘s browser. With PHP installed in your computer, you can use both procedural programming and object oriented programming (OOP). In some recent PHP versions, not every OOP feature is mentioned. Some code libraries and large applications have been written by using just the OOP codes.  Why Should You Care about What Your Code Looks Like?  **It’ s important to follow good coding practices for three reasons:**   * **For efficiency:**   The easier your code is to read and follow, the easier it will be to keep track of where you are within your code, and the quicker it will be to pick up where you left off after a break   * **For debugging:**   Knowing where your problem lies is a major debugging tool. If comments are used correctly, you can easily follow your own logic, and if you have line numbers and consistent formatting, you can easily scan your document to pinpoint a trouble area.   * **For future expansions and modifications:**   Using comments in your code is especially important for future changes because it ‘ s difficult to remember the logic behind code that was written years or even just months ago. Also, if you are working on code that involves a team, if everyone is using the same coding style, it will be much easier to make changes or additions to someone else ‘ s work down the road.  **Creating Your First Program**  You can ‘ t get much simpler than this first program, but try it out to get a feel for what the results look like. The PHP statement echo , seen in the example that follows, is one of the most commonly used PHP functions and one that you will undoubtedly become intimate with. It is used to send text (or variable values or a variety of other things) to the browser.  Using echo  Try using echo to see what results you achieve.  1. Enter the following program in your favorite text editor (Notepad, Simple Text, or whatever you choose), and save it as firstprog.php . Regardless of your editor, make sure |

|  |
| --- |
| you save it in a plaintext format to avoid parsing problems. If you ‘ re using Notepad, double - check to ensure that the file is not saved as firstprog.php.txt by default.  Simple php program  Create a php webpage and print ―hello world‖.  <Html>  <Head>  <Title> My Simple Program </Title>  </head>  <Body>  < ?php  echo ―I‘m a lumberjack.‖;  ? >  </Body>  </Html>  2. Open this program using your browser. Your resulting screen should look like the below    **Integrating H TML with PHP**  You will be better able to see how easily you can use HTML in the PHP program with the following practical example.  In this example, you ‘ ll use some PHP and HTML together.  1 . Modify the highlighted lines of firstprog.php :  < html >  < head >  < title > My First PHP Program < /title >  < /head >  < body >  < ?php  echo ― < h1 > I‘m a lumberjack. < /h1 > ‖; echo ― < h2 > And I‘m okay. < /h2 > ‖;  ? >  < /body >  < /html >  2.Save your file, and reload the page. Your screen should now look something like the one in |

|  |
| --- |
| **PHP - Variables**  A variable is a means of storing a value, such as text string "Hello World!" or the integer value 4. A variable can then be reused throughout your code, instead of having to type out the actual value over and over again. In PHP you define a variable with the following form:   * $variable\_name = Value;   If you forget that dollar sign at the beginning, it will not work. This is a common mistake for new PHP programmers!  Let‘s look at the syntax of a PHP variable:  $variable\_name = Value; Example:  <?php  $learning = ―Learning Variable!‖;  $x\_numeral = 8;  $first\_name = ‗John‘;  $lastName = ‗Denver‘;  $nextNumeral = 16;  ?>  **PHP Variable Naming Conventions**  There are a few rules that you need to follow when choosing a name for your PHP variables.   * PHP variables must start with a letter or underscore "\_". * PHP variables may only be comprised of alpha-numeric characters and underscores. a-z, A- Z, 0-9, or \_ . * Variables with more than one word should be separated with underscores. $my\_variable * Variables with more than one word can also be distinguished with capitalization.   $myVariable  **Outputting a String -Echo**  To output a string, use the PHP echo function. You can place either a string variable or you can use quotes, like we do below, to create a string that the echo function will output. PHP Code:  <?php  $myString = "Hello!"; echo $myString;  echo "<h5>I love using PHP!</h5>";  ?>  **Display:**  Hello!  I love using PHP!  In the above example we output "Hello!" without a hitch. The text we are outputting is being sent to the user in the form of a web page, so it is important that we use proper HTML syntax! |

In our second echo statement we use echo to write a valid Header 5 HTML statement. To do this we simply put the <h5> at the beginning of the string and closed it at the end of the string. **Variable Scope:**

* Local variables
* Function parameters
* Global variables
* Static variables.

## Local Variables

A variable declared in a function is considered local; that is, it can be referenced solely in that function. Any assignment outside of that function will be considered to be an entirely different variable from the one contained in the function –

Example

<?php

$x = 4; assignx();

echo ("\$x outside of function is $x. <br />"); function assignx ()

{

$x = 0;

echo ( "\$x inside function is $x. <br />");

}

?>

Output:

$x inside function is 0.

$x outside of function is 4.

## Function Parameters

Function parameters are declared after the function name and inside parentheses. They are declared much like a typical variable would be −

<?php

// multiply a value by 10 and return it to the caller function multiply ($value) {

$value = $value \* 10; return $value;

}

$retval = multiply (10);

Print "Return value is $retval\n";

?>

This will produce the following result − Return value is 100

## Global Variables

In contrast to local variables, a global variable can be accessed in any part of the program. However, in order to be modified, a global variable must be explicitly declared to be global in the function in which it is to be modified. This is accomplished, conveniently enough, by placing the keyword **GLOBAL** in front of the variable that should be recognized as global. Placing this keyword in front of an already existing variable tells PHP to use the variable having that name. Consider an example −

|  |
| --- |
| <?php  $somevar = 15; function addit() { GLOBAL $somevar;  $somevar++;  print "Somevar is $somevar";  }  addit();  ?> |
| This will produce the following result − Somevar is 16  **Static Variables**  The final type of variable scoping that I discuss is known as static. In contrast to the variables declared as function parameters, which are destroyed on the function's exit, a static variable will not lose its value when the function exits and will still hold that value should the function be called again.  You can declare a variable to be static simply by placing the keyword STATIC in front of the variable name. |
| <?php  function keep\_track() { STATIC $count = 0;  $count++; print $count; print "<br />";  }  keep\_track(); keep\_track(); keep\_track();  ?> |
| This will produce the following result − 1  2  3  **Constants**  A constant is a name or an identifier for a simple value. A constant value cannot change during the execution of the script. By default, a constant is case-sensitive. By convention, constant identifiers are always uppercase. A constant name starts with a letter or underscore, followed by any number of letters, numbers, or underscores. If you have defined a constant, it can never be changed or undefined.  To define a constant you have to use define() function and to retrieve the value of a constant, you have to simply specifying its name. Unlike with variables, you do not need to have a constant with a $. You can also use the function constant() to read a constant's value if you wish to obtain the constant's name dynamically.  **constant() function**  As indicated by the name, this function will return the value of the constant. |

|  |
| --- |
| This is useful when you want to retrieve value of a constant, but you do not know its name,  i.e. It is stored in a variable or returned by a function.  **constant() example** |
| <?php define("MINSIZE", 50);  echo MINSIZE;  echo constant("MINSIZE"); // same thing as the previous line  ?> |
| Only scalar data (boolean, integer, float and string) can be contained in constants.  **Differences between constants and variables are**   * There is no need to write a dollar sign ($) before a constant, where as in Variable one has to write a dollar sign. * Constants cannot be defined by simple assignment, they may only be defined using the define() function. * Constants may be defined and accessed anywhere without regard to variable scoping rules. * Once the Constants have been set, may not be redefined or undefined. |

|  |
| --- |
| **Multiple Choice Questions**  1. Which one of the following databases has PHP supported almost since the beginning?   1. Oracle Database 2. SQL 3. SQL+ 4. **MySQL**   **2.** Which one of the following statements is used to create a table?   1. **CREATE TABLE table\_name (column\_name column\_type);** 2. CREATE table\_name (column\_type column\_name); 3. CREATE table\_name (column\_name column\_type); 4. CREATE TABLE table\_name (column\_type column\_name);   **3. PHP is an example of scripting language.**   1. **Server-side** 2. Client-side 3. Browser-side 4. In-side   **4. Which of the following is not true?**   1. PHP can be used to develop web applications. 2. PHP makes a website dynamic 3. PHP applications can not be compile 4. **PHP can not be embedded into html.**   5. Which of the following is not used to begin php code   1. <?php 2. <? 3. **<php** 4. ― <script language=‖php‖> 6. PHP is a -------------------- 5. **Loosely typed language** 6. Tightly typed language 7. Server typed language 8. Client typed language   7. Full form of PHP   1. **PHP Hypertext Pre-processor** 2. Hypertext Pre-processor 3. Plain Hypertext Pre-processor 4. Parsed Hypertext Pre-Processor   8. Php code ends with   1. **;** 2. : 3. .   d) ,  9. Which of the following statement is not true about echo?   1. Not written within parenthesis 2. Can output more than 1 string 3. **Slower than print** 4. Echo($arg1[,$arg2……])   10. PHP variables must begin with a sign |

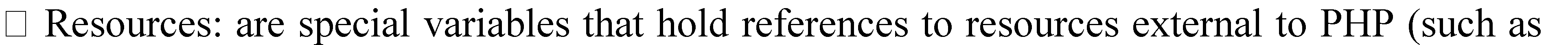
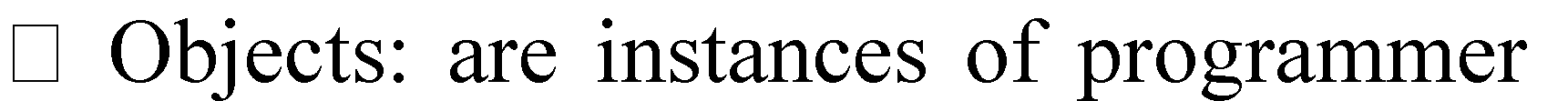
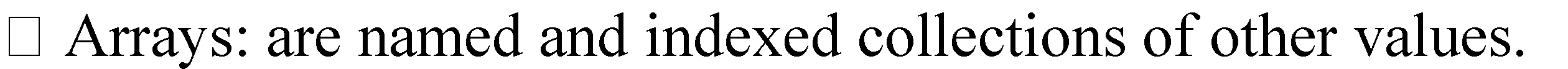
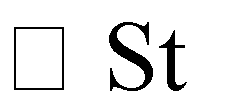
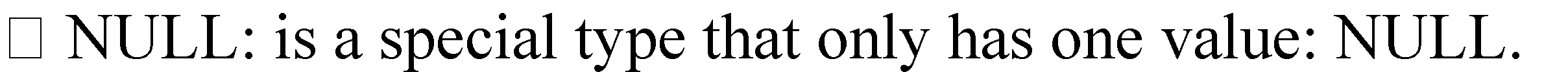
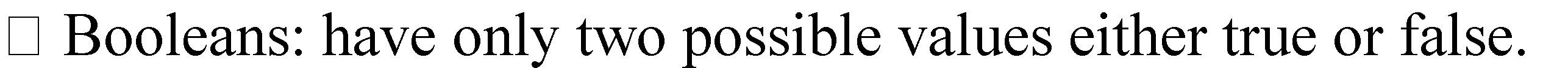
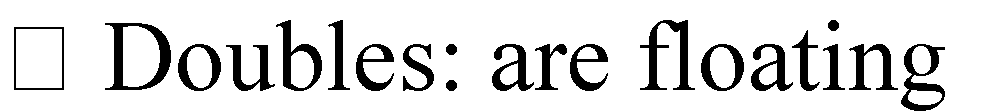
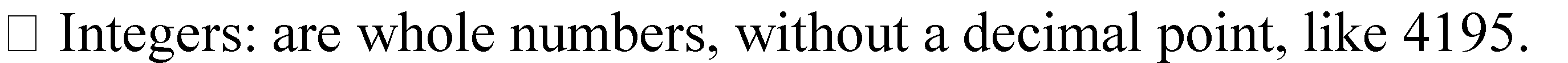
|  |  |
| --- | --- |
| 1. **$** 2. @ 3. & 4. #   11. Constants are defined using which of the following function   1. Include 2. Require 3. **Define** 4. Main   12. Which of the following command gives information about the fields in a table?   1. **Describe** 2. Create 3. Use 4. Select   13. **Which of the following statements prints in PHP?**   1. Out 2. Write 3. **Echo** 4. Display   14. Which of the following is used to add multi line comment in PHP a) {/ \}  b) //  **c) /\* \*/**  d) {{ }}  15. What will be the output of the following php code? |  |
| < ?php  $num = "1";  $num1 = "2";  print $num+$num1 ;  ?> | |
| **a) 3**  b) 1+2  c) Error  d) 12  16. What will be the output of the following PHP code? |  |
| **<?php**  $num = 1;  $num1 = 2;  print $num . "+". $num1;  **?>** |  |
| a)3 **b)1+2** c)1.+.2  d) Error  17. **Which of the following is not the scope of Variable in PHP?**  A. Local |  |

|  |
| --- |
| 1. Global 2. Static 3. **Extern**   18. What is the range of unsigned TINYBIT Integer  **a) 0 to 255**  b) -128 to 127  c) 10 to 265  d) 8 to 263  19. Which among the following datatypes is not case sensitive   1. VARCHAR 2. TINYBLOB 3. TINYTEXT 4. **TINYBIT**   20. Which of the following command is used to show the database created inside MYSQL.   1. **SHOW** 2. USE 3. CREATE 4. SELECT   21. Which of the following command is used to connect to MYSQL   1. SHOW 2. **USE** 3. CREATE 4. SELECT   22. Each Table within the database is been defined and created by which of the following command   1. SHOW 2. USE 3. **CREATE** 4. SELECT   23. Which of the following command is used to print all the records that match the query?   1. SHOW 2. USE 3. CREATE 4. **SELECT**   24. Which of the following is scalar data?   1. Boolean 2. Integer 3. Float 4. **Array**   25. Which of the following is not true about constant?   1. There is no need to write a dollar sign ($) before a constant 2. **Constants can be defined by simple assignment** 3. Once the Constants have been set, may not be redefined or undefined. 4. Constants may be defined and accessed anywhere without regard to variable scoping rules.   **Long Questions**   1. What is the use of CREATE TABLE   and SHOW TABLE Command, Give the syntax?   1. Give the syntax of the following |

|  |
| --- |
| 1. Insert 2. Select 3. Update 4. Delete 5. What are steps involved to create table   using MySQL   1. Explain how the text string is further   defined?   1. Write a note on date and time datatype   in MySQL   1. What is the use of autoincrement in   MySQL   1. Explain the different features of PHP 2. With a simple code explain the basic   syntax of PHP   1. Explain php echo statement with   example   1. With syntax explain print statement 2. Explain 2 types of comments available   in php   1. Write a note on scope of variable in   php   1. Write a note on constants, how is it   different from variable |

|  |
| --- |
| **UNIT III FUNDAMENTALS OF PHP**  **Echoing Variables and Text Strings**  All variable names must start with the $ character. After that there must be a letter. The rest of the name can consist of both letters and numbers. The following are three examples of valid names for a variable: $name, $address2 and $colour\_30. The \_ (underscore) character is often used in variable names. It is used as a replacement for space, since space is an illegal character in a variable name.  Once a variable is named it needs to be assigned a value to be used in any program. Assigning a value to variable is done as follows:  $name= 'Jitesh';  The variable name is on the left of the = sign (i.e. assignment operator) and the value of the variable is on its right. Here 18 is assigned as a value to the variable $age:  $age= 18;  There are several different types of variables. The integer and string types have already been demonstrated.  **NUMBERS**  Dealing with numbers is easy with PHP. Just use them as required. All the normal rules about precedence apply:  $a = 4;  $b = 7;  $c=2+3\*$a+5\*$b; ( **This is evaluated as 2+(3'\*4)+(5\*7) = 49** )  Everything works as expected. Any variable is automatically substituted with its value. This is done before the new value is stored in the variable on the left side of the = sign. This means that something like this can be done:  $a = 5;  $b = 10;  $a= (**2 \* $a + 5)/$b; ( Evaluated as $a (2\*5+5) /10= 1.5**)  **STRINGS**  To assign a string to a variable, the string must be enclosed in quotes. Either single quotes(' ) or double quotes ( " ) can be used. The kind of quotes chosen depends on the string being worked with in a given situation.  There are some differences between the two types of quotes. The following code should demonstrate the differences:  $first\_name = 'Chhaya';  $greeting1 = "Hello, my first name is $first\_name. "; echo $greeting1;  $Iast\_name = 'Bankar';  $greeting2 = 'Hello, my last name is $Iast\_name. '; |

|  |
| --- |
| echo $greeting2;  This code produces the following output:  **Hello, my first name is Chhaya. Hello, my last name is $Iast\_name.**  When double quotes are used, PHP performs what is known as variable expansion. That means that PHP expands (substitutes) $first\_name with it's value. The result is that the string stored in $greetingl is Hello, my name is Chhaya with variable expansion being done on the variable $firsCname. When assigning a value to the variable $greeting2, using single quotes, PHP does not do any variable expansion hence $greeting2 ends up with Hello, my last name is  $Iast\_name, (i.e. anything enclosed in single quotes is treated as a string constant and not to be changed in anyway by PHP)  Double quotes also expand other special characters such as newline characters (\n) This means that the following string:  echo ―Chhaya\nBankar";  will look like this when rendered in the client browser:  Chhaya Bankar  The echo statement just outputs the string after it to the Web browser. As said earlier there's no trace of the PHP code, but notice that there's a newline character between the two words. The newline character can only be seen in the source code of the HTML page, because the Web browser treats the newline character as a normal space. Notice that there's no space before or after the \n character.  This is because the ( \ ) backslash is used to indicate that the next character is special. In PHP code spec write ( \ \ ) when a single ( \ ) is required. The backslash ( \ ) is called an escape character, i.e. it makes characters escape from their normal role.  Its only when expressions are evaluated (i.e. such as in assignments and echo statements) that the difference between single and double quotes comes into play. After the assignment is done, no one can tell how the string was produced. Consider this code.  $first\_name = 'Chhaya';  $Iast\_name = "Bankar";  $varl = "$firsCname $lasCname";  $var2= 'Chhaya Bankar';  The two variables $varl and $var2 will both contain the string Chhaya Bankar after the evaluation, PHP will not make any distinction between the values stored in the two variables. To concatenate (add together) strings use the ( • ) period character (i.e. a dot or full stop) this:  $first\_name = 'Chhaya';  $Iast\_name = 'Bankar';  $furr\_name = $first\_name.$last\_name;  Now $full\_name will contain the string "ChhayaBankar". That's probably not what is wanted it would be better with a space between the two words. To do this, execute the following code:  $full\_name = ‗$first\_name ' , ‗ $last\_name‘; |



Notice that the space between the variable $first\_name and the dot is ignored, it's the string with the space that's important. Use the dot (i.e. period character) each time two strings have to be concatenated.

But it is also seen how this could be solved using variable substitution. This is often easier to read, just remember to use double-quotes around the string. So this code gives the same result:

$full\_name = "$first\_name $Iast\_name";

## Data Types:

PHP has a total of eight data types which we use to construct our variables:

-point numbers, like 3.14159 or 49.1.

rings: are sequences of characters, like 'PHP supports string operations.'

-defined classes, which can package up both other kinds of values and functions that are specific to the class.

database connections).

The first five are simple types, and the next two (arrays and objects) are compound - the compound types can package up other arbitrary values of arbitrary type, whereas the simple types cannot.

‗Data Types‘ can be divided into two groups: ‗Core Data Type‘ and ‗Special Data Type‘.

The ‗Core data type‘ group includes Integer, Float/Double, String and Boolean. The ‗Special data types includes Null, Array, Object and Resources.

**Integers**: As discussed earlier, Integers are whole numbers. It does not include precision. Negative values are also regarded as Integers.

Example: -32, 32, 986, 1245, etc.

**Floating-Point Number or Double**: Fractional numbers are grouped as Floating point numbers or Double data type. Simply put, Double variables hold numbers with decimal points. Example: 123.56, 5.6, etc.

The syntax of the ‗Floating-Point Number or Double is as follows:

<?php

$a = 1.234;

$b = 1.2e3;

$c = 7E-10;

?>

## Boolean:

They have only two possible values either true or false. PHP provides a couple of constants especially for use as Booleans: TRUE and FALSE, which can be used like so:

if (TRUE)

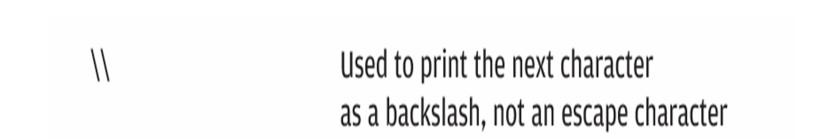
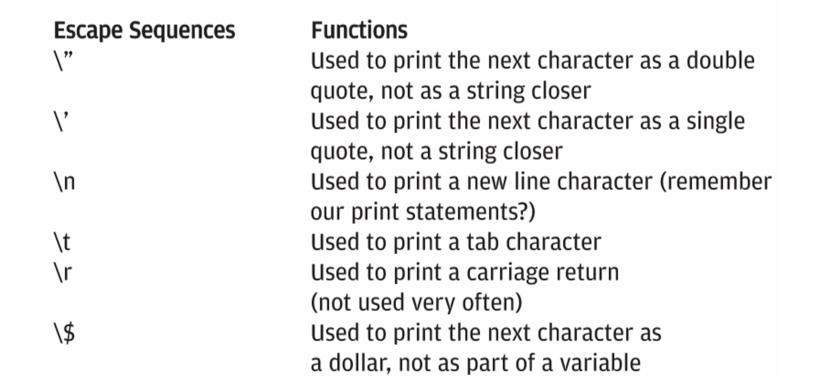
echo ("This will always print<br>"); else

echo ("This will never print<br>");

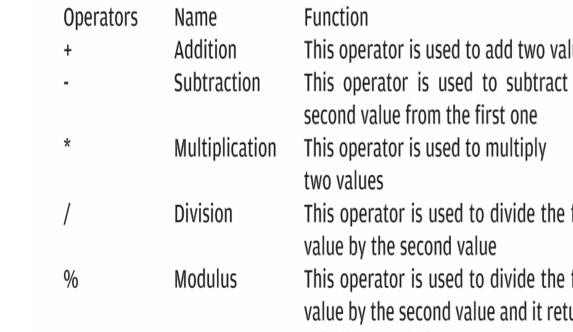
## NULL:

NULL is a special type that only has one value: NULL. To give a variable the NULL value, simply assign it like this:

$my\_var = NULL;



|  |
| --- |
| The special constant NULL is capitalized by convention, but actually it is case insensitive; you could just as well have typed:  $my\_var = null;  **String Literals:**  We have already discussed that Strings hold both words and sentences. These are always inserted within quotation marks. If it starts with a single quotation mark then it must end with the same. If a single quotation is inserted at the beginning of a string then it can not be closed with a double quotation mark. If the quotation marks are inserted in a code without any characters, then it will be treated as ‗Null‘ string. A numeric character is treated as a string if it is inserted within quotation marks. For example, if the number 9 is inserted in a PHP code, then it will be treated as a number. On the other hand, if 9 is inserted in a PHP code, then it will be treated as a string.  Example: ―It is an example of a string with double quotes‖ ‗It is an example of a string with single quotes‘ ―It is also an example of ‗a string‘ where the single quote will be ignored‖ ‗It is an example of ―a string‖ where the double quote will be ignored‘ ―4‖ ― ― (Null string)  **Escape Sequences:**  You can achieve the same effect in double-quoted strings by using the escape character, which, in PHP, is a backslash \. Escape sequences, the combination of the escape character \ and a letter, are used to signify that the character after the escape character should be treated specially. For example, if you wanted to have the string "And then he said, "That is amazing!", which was true", you would need escape characters because you have double quotes inside double quotes. Here is a list of the escape sequences in PHP:  **Operators**  In PHP, variables and values are performed by Operators, that is, they operate on variables and values in PHP. Look at the following expression: $z = $x + $y;  In the above expression x and y are two numbers. It is clear from the above expression that it would add x with y and the sum is z. The plus sign (+) inserted between x and y is an operator (Arithmetic Operator).  Operators used in PHP are categorically grouped in various sections:   1. Assignment Operators 2. Arithmetic Operators 3. Comparison Operators 4. String Operators 5. Logical Operators 6. Conditional Operators   **Assignment operators**  You can use the Assignment Operator to assign a value to a variable. Often a variable is assigned a value of another variable. In this case assignment operators are used. The equal character (=) is used here. Look at the following expression: |



xample 01:

adding = 2 + 4;

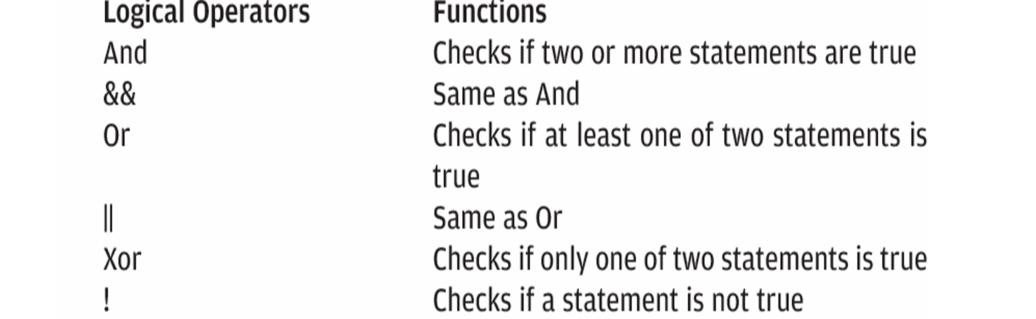
$minus = 6 - 2;

$multiply = 5 \* 3;

divide = 15 / 3;

i.e. 5.

|  |  |
| --- | --- |
| Example:  $first\_var = 5;  $second\_var = $first\_var;  Here the values of both ‗$first\_var‘ and ‗$second\_var‘ variables are assigned the same value | |
|  |  |
| **Arithmetic operators**  Look at the various Arithmetic Operators:  E  $  $  $percent = 5 % 2;  echo ―Result adding: 2 + 4 = ―.$adding.‖<br />‖; echo ―Result minus: 6 - 2 = ―.$minus.‖<br />‖; echo ―Result multiply: 5 \* 3 = ―.$multiply.‖<br />‖; echo ―Result divide: 15 / 3 = ―.$divide.‖<br />‖; echo ―Result percent: 5 % 2 = ― . $percent;  The output of the above program is as follows:  Result adding: 2 + 4 = 6  Result minus: 6 - 2 = 4  Result multiply: 5 \* 3 = 15  Result divide: 15 / 3 = 5  Result percent: 5 % 2 = 1.  **Comparison operators :**  The ‗Comparison Operators‘ verify the relationship between a variable and its value. These operators are usually inserted within a conditional statement and it returns boolean values like true and false. Look at the various types of Comparison Operators: | |



**ariable incrementing and decrementing** rementing and Decrementing operators are unary erators. Incrementing Operators

|  |
| --- |
| **String operators**  There are two types of ‗String Operators‘:   * The   Concatenating Operator (‗.‘) and   * The   Concatenating Assignment Operator (‗.=‘).  The Concatenating Operator joins the right and the left string into a single string. The Concatenating Assignment Operators add the argument that is placed on the right side of the equal operator with the argument placed on the left side of the ‗equal‘ operator.  Example:  $first\_string = ―Welcome‖;  $second\_string = ― Jack‖;  $third\_string = $first\_string . $second\_string; echo $third\_string . ―!‖;  The output of the above program is as follows:  Welcome Jack!  **Logical operator**  **V**  Inc op  Increment operator increases the value of its operand by 1. The operand(variable) must be a integer value.  We have two types of Increment operator, |

|  |
| --- |
| 1. Pre-increment operator   ++$var (Pre-increment : Increments $var by 1, then returns $var)   1. Post-increment operator   $var++ (Post-increment : Returns $var, then increments $var by one) Example 1: **Pre-increment operator**  <?php  $i=2;  //Increments $i by 1, then returns $i echo ++$i ."<br>";  echo $i;  ?>  **Output**  3  3  Example 2: **Post-increment operator**  <?php  $i=2;  //Returns $i, then increments $i by one echo $i++ ."<br>"; echo $i;  ?>  Output 2  3  We also have two types of Decremental Operator's,   1. Pre-decrement Operator   ++$var (Pre-decrement : Decrements $var by 1, then returns $var)   1. Post-decrement Operator   $var++ (Post-decrement : Returns $var, then decrements $var by one) Example 3: **Pre-decrement operator**  <?php  $i=2;  Decrements $i by 1, then returns $i echo --$i ."<br>";  echo $i;  ?>  Output 1  1  Example 2: **Post-decrement Operator**  <?php  $i=2;  //Returns $i, then decrements $i by one echo $i-- ."<br>";  echo $i;  ?>  Output 2  1  **The ? : Operator** |

For example

|  |
| --- |
| Let's check **conditional operator ? :** which can be used to replace **if...else** statements. It has the following general form:  Exp1 ? Exp2: Exp3;  Where Exp1, Exp2, and Exp3 are expressions. Notice the use and placement of the colon.  The value of a ? expression is determined like this: Exp1 is evaluated. If it is true, then Exp2 is evaluated and becomes the value of the entire ? expression. If Exp1 is false, then Exp3 is evaluated and its value becomes the value of the expression.  **Precedence of PHP Operators:**  Operator precedence determines the grouping of terms in an expression. This affects how an expression is evaluated. Certain operators have higher precedence than others; for example, the multiplication operator has higher precedence than the addition operator:  x = 7 + 3 \* 2;  Here x is assigned 13, not 20 because operator \* has higher precedence than + so it first get multiplied with 3\*2 and then adds into 7.  Here operators with the highest precedence appear at the top of the table, those with the lowest appear at the bottom. Within an expression, higher precedence operators will be evaluated first.  **PHP Control structure**  The ‗Control Structure‘ controls the program flow of PHP. It can also check whether a block of code is executed or not. The syntax of the ‗Control Structure‘ is as follows:  <?php  if (expression) statement  ?>  Let‘s look at various types of ‗Control Structure‘:   * if * elseif/else if * switch * while * do-while * for |

|  |
| --- |
| * foreach * if: It is used for conditional execution of code fragments. It returns Boolean values (true/false).   Look at the syntax of ‗if‘ Control Structure:  if (expr) statement Example:  <?php  if ($x > $y)  {  echo ―x is bigger than y‖;  }  ?>  else: If an expression in the ‗if‘ statement returns false, then the ‗else‘ ‗Control Structure‘ is used.  Example:  <?php  if ($x > $y)  {  echo ―x is bigger than y‖;  }  else  {  echo ―x is NOT bigger than y‖;  }  ?>  **elseif/else if:**  It is a combination of ‗if‘ and ‗else‘ Control Structure. If the ‗if‘ Control Structure‘ returns a  ‗false‘ value, then a different statement is executed by using the ‗else‘ Control Structure‘. Example 1:  <?php  $t = date("H");  if ($t < "10") {  echo "Have a good morning!";  } elseif ($t < "20") {  echo "Have a good day!";  } else {  echo "Have a good night!";  }  ?>  Example 2:  <?php //Change the messages to what you want.  $afternoon = "Good afternoon! ";  $evening = "Good evening! ";  $late = "Working late? ";  $morning = "Good morning! ";  $friday= "Get ready for the weekend! ";  //Get the current hour |

|  |
| --- |
| $current\_time =date('G');  //Get the current day  $current\_day = date('l'); echo $current\_time; echo $current\_day;  //12 p.m. - 4 p.m.  if ($current\_time >= 12 && $current\_time <= 16)  {  echo $afternoon;  }  // 5 p.m. to 11 p.m.  elseif ($current\_time >= 17 && $current\_time <= 24)  {  echo $evening;  }  //12 a.m. - 5 a.m.  elseif ($current\_time >= 1 && $current\_time <= 5)  {  echo $late;  }  // 6 a.m. to 11 a.m.  elseif ($current\_time >= 6 && $current\_time <= 11)  {  echo $morning;  }  //If it's Friday, display a message if ($current\_day == "Friday")  {  echo $friday;  }  ?>  **switch:**  This Control Structure is similar to a series of ‗if‘ statements.  If you want to select one of many blocks of code to be executed, use the Switch statement. The switch statement is used to avoid long blocks of if..elseif..else code.  Syntax  switch (expression)  {  case label1: code to be executed if expression = label1; break;  case label2: code to be executed if expression = label2; break;  default: code to be executed if expression is different from both label1 and label2; |

|  |
| --- |
| }  Example  <?php  $favcolor= "red";  switch ($favcolor) { case "red":  echo "Your favorite color is red!"; break;  case "blue":  echo "Your favorite color is blue!"; break;  case "green":  echo "Your favorite color is green!"; break;  default:  echo "Your favorite color is neither red, blue, nor green!";  }  ?>  **PHP Loop Types while:**  The ‗while‘ Control Structure executes the nested statements repetitively until the ‗while‘ statement returns a false value. The syntax of ‗while‘ control structure is as follows:  while (expr) statement Example  <?php  $x = 1;  while($x <= 5)  {  echo "The number is: $x <br>";  $x++;  }  ?>  **do-while :**  It is very much similar to the ‗while‘ Control Structure. The only difference is that here the truth expression is checked at the end of every repetition. Look at the syntax of ‗do-while‘:  <?php  $i = 0; do  {  echo $i;  } while ($i > 0);  ?>  **for:**  This is one of the complex loops in PHP. The syntax of the ‗for‘ control structure is as follows:  for (expr1; expr2; expr3) statement Example |

|  |
| --- |
| <html>  <head>  <title>STAR</title>  </head>  <body>  <h2>PYRAMIND</h2>  <?php if(isset($\_POST["Submit"]))  {  $val = $\_POST["val"];  $symbol = $\_POST["symbol"];  }  else if(isset($\_POST["clear"]))  {  $val = "";  $symbol = "";  }  ?>  <form method="POST">  Enter the loop Number <input type="text" name="val" value=" <?php echo $val; ?>" />  </br>  Enter the Symbol <input type="text" name="symbol" value=" <?php echo $symbo; ?>"/>  </br>  <input type="submit" name="Submit">  <input type="submit" name="clear" value="Reset" >  </form>  <?php  if($val and $symbol)  {  if(is\_numeric($val))  {  for($i=1;$i<=$val;$i++)  {  for($j=1;$j<=$i;$j++)  {  echo "&nbsp ".$symbol;  }  echo "<br />";  }  }  else  {  echo "Enter loop in Digits";  }  }  ?>  </body>  </html>  **foreach:** |

|  |
| --- |
| The foreach statement is used to loop through arrays. For each pass the value of the current array element is assigned to $value and the array pointer is moved by one and in the next pass next element will be processed.  Syntax  foreach (array as value)  {  code to be executed;  }  Example  Try out following example to list out the values of an array.  <html>  <body>  <?php  $array = array( 1, 2, 3, 4, 5); foreach( $array as $value )  {  echo "Value is $value <br />";  }  ?>  </body>  </html>  This will produce following result:  Value is 1  Value is 2  Value is 3  Value is 4  Value is 5  **The break statement**  The PHP break keyword is used to terminate the execution of a loop prematurely.  The break statement is situated inside the statement block. If gives you full control and whenever you want to exit from the loop you can come out. After coming out of a loop immediate statement to the loop will be executed.  Example  In the following example condition test becomes true when the counter value reaches 3 and loop terminates.  <html>  <body>  <?php  $i = 0;  while( $i < 10)  {  $i++;  if( $i == 3 )break;  }  echo ("Loop stopped at i = $i" );  ?>  </body>  </html>  This will produce following result: Loop stopped at i = 3 |

## The continue statement

The PHP continue keyword is used to halt the current iteration of a loop but it does not terminate the loop.

Just like the break statement the continue statement is situated inside the statement block containing the code that the loop executes, preceded by a conditional test. For the pass encountering continue statement, rest of the loop code is skipped and next pass starts.

Example

In the following example loop prints the value of array but for which condition becomes true it just skip the code and next value is printed.

<?php

$x=1;

echo 'List of odd numbers between 1 to 10 <br />'; while ($x<=10)

{

if (($x % 2)==0)

{

$x++; continue;

}

else

{

echo $x.'<br />';

$x++;

}

}

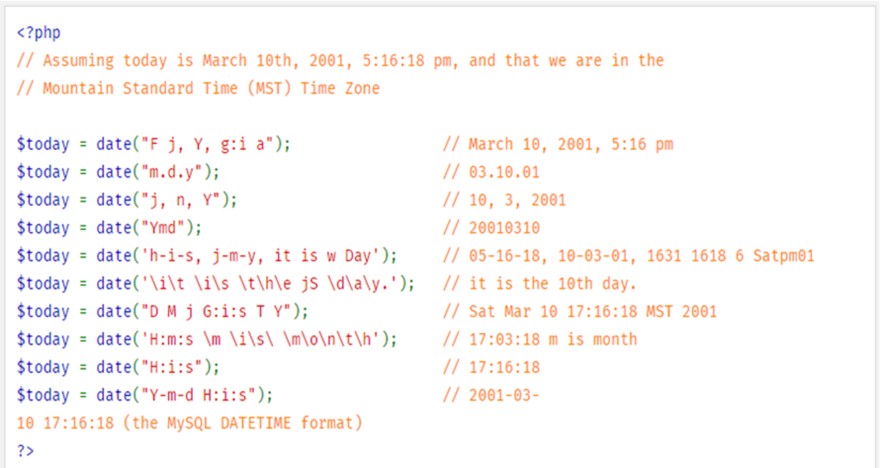
This will produce following result List of odd numbers between 1 to 10 1

3

5

7

9



|  |
| --- |
| **CHAPTER 6 DATE FUNCTION ()**  In PHP the date() function is used to format a timestamp or a date. It arranges a timestamp into a readable date and time. Using the date/ time functions, you can format date and time on the server. However, these functions entirely depend on the server settings.  Here you can use the syntax, date (format, timestamp).  Look at the table below:  Format: This parameter is essential. It assigns the timestamp format.  Timestamp: This Parameter is optional. This takes the Date or/and time that you want to format. If no value is provided then the current time is used for formatting.  In the date function, the first parameter specifies about formatting date and time. Several letters are used to represent date and time formats. Some commonly used letters are given below:   * d - Represents day of a month (01-31) * D - Represents day in three letter text format * m - Represents month, as a number (01-12) * M - Represents month in three letter text format * Y - Represents year in four digits * y - Represents year in two digits * a – ―am‖ or ―pm‖ * A – ―AM‖ OR ―PM‖ * F – Full name of the month ( January- December) * g – Hours in 12- hour format without leading zero (1-12) * G – Hours in 24- hour format without leading zero (0-23) * h – Hours in 12- hour format with leading zero (01-12) * H – Hours in 24- hour format with leading zero (00-23) * i – minutes with leading zero(00-59) * j – day of month without leading zero * l – the full name of the day (Monday- Sunday) * n – the month as a number, without leading zero(1-12) * s – the seconds with leading zero(00-59) * t – the number of days in a given month (28-31) * w – the day of the week as a number (0-6, 0=Sunday) |

|  |
| --- |
| **Arrays**  In PHP, arrays are ordered data maps and are used to store, manage and operate a set of variables. To put it simply, an array is a data structure that holds multiple data within a single identifier. There are two parts in an Array - Values and Keys. While Values contain information to be stored, Keys are used to identify these values. It is allocated to a single variable. It holds significant information, popularly termed as Array Elements. This information can be used for a number of times in a program. Either non negative Integers or Strings are used as Keys. The arrays that use non-negative Integers as Keys are termed as Scalar Arrays. These are Associative Arrays that use Strings as keys. An Array may contain different Array(s) popularly known as Multidimensional Arrays.  The syntax of an Array is as follows:  $array[key] = value;  Look at the simple example below:  Example:  $student\_array[0] = ―Rohit‖;  $student\_array[1] = ―Rahul‖;  $student\_array[2] = ―Sourav‖;  $student\_array[3] = ―Abdul‖;  In the above example, the names of the students (Rohit, Rahul, Sourav and Abdul) are the Values and the numeric characters (0, 1, 2 and 3) are the Keys of this array.  **Array function**  **array** — Create an array  **count** — Count all elements in an array  **current** — Return the current element in an array  [**prev ()**](https://www.php.net/manual/en/function.prev.php) - Rewind the internal array pointer  [**next ()**](https://www.php.net/manual/en/function.next.php) - Advance the internal pointer of an array  [**end ()**](https://www.php.net/manual/en/function.end.php) - Set the internal pointer of an array to its last element  <?php  $transport = array('foot', 'bike', 'car', 'plane');  $mode = current($transport); // $mode = 'foot';  $mode = next($transport); // $mode = 'bike';  $mode = current($transport); // $mode = 'bike';  $mode = prev($transport); // $mode = 'foot';  $mode = end($transport); // $mode = 'plane';  $mode = current($transport); // $mode = 'plane';  ?>  **sizeof()**  The sizeof() function returns the number of elements in an array. The sizeof() function is an alias of the count() function.  <?php  $cars=array("Volvo","BMW","Toyota"); echo sizeof($cars);  ?>  Output:  3  **Pos()**  The pos() function returns the value of the current element in an array. |

|  |
| --- |
| This function is an alias of the [current()](https://www.w3schools.com/php/func_array_current.asp) function.  <?php  $people = array("Peter", "Joe", "Glenn", "Cleveland"); echo pos($people) . "<br>";  ?>  Output Peter  **Reset()**  The reset() function moves the internal pointer to the first element of the array.  <?php  $people = array("Peter", "Joe", "Glenn", "Cleveland");  echo next($people) . "<br>"; echo reset($people);  ?>  Output Joe Peter  **Variable function**  **isset** — Determine if a variable is declared and is different than **NULL**  <?php  $var = '';  // This will evaluate to TRUE so the text will be printed. if (isset($var)) {  echo "This var is set so I will print.";  }  ?>  **empty** — Determine whether a variable is empty  <?php  $var = 0;  // Evaluates to true because $var is empty if (empty($var)) {  echo '$var is either 0, empty, or not set at all';  }  // Evaluates as true because $var is set if (isset($var)) { |

|  |
| --- |
| echo '$var is set even though it is empty';  }  ?>  There are three different kind of arrays and each array value is accessed using an ID c which is called array index.   * **Numeric array** − An array with a numeric index. Values are stored and accessed in linear fashion. * **Associative array** − An array with strings as index. This stores element values in association with key values rather than in a strict linear index order. * **Multidimensional array** − An array containing one or more arrays and values are accessed using multiple indices   **Numeric Array**  These arrays can store numbers, strings and any object but their index will be represented by numbers. By default array index starts from zero.  **Example**  Following is the example showing how to create and access numeric arrays.  Here we have used **array()** function to create array. This function is explained in function reference.  <html>  <body>  <?php  $numbers[0] = "one";  $numbers[1] = "two";  $numbers[2] = "three";  $numbers[3] = "four";  $numbers[4] = "five";  foreach( $numbers as $value ) { echo "Value is $value <br />";  }  ?>  </body>  </html> **Output** Value is one Value is two Value is three Value is four Value is five  **Associative Arrays**  The associative arrays are very similar to numeric arrays in term of functionality but they are different in terms of their index. Associative array will have their index as string so that you can establish a strong association between key and values.  Example  <!DOCTYPE html>  <html>  <body>  <?php  $age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43"); echo "Peter is " . $age['Peter'] . " years old.";  ?> |

|  |
| --- |
| </body>  </html>  **OUTPUT**  Peter is 35 years old.  **Multidimensional Arrays**  A multi-dimensional array each element in the main array can also be an array. And each element in the sub-array can be an array, and so on. Values in the multi-dimensional array are accessed using multiple index.  Example  In this example we create a two dimensional array to store marks of three students in three subjects −  This example is an associative array, you can create numeric array in the same fashion.  <html>  <body>  <?php  $marks = array( "mohammad" => array (  "physics" => 35,  "maths" => 30,  "chemistry" => 39  ),  "qadir" => array ( "physics" => 30,  "maths" => 32,  "chemistry" => 29  ),  "zara" => array ( "physics" => 31,  "maths" => 22,  "chemistry" => 39  )  );  /\* Accessing multi-dimensional array values \*/ echo "Marks for mohammad in physics : " ; echo $marks['mohammad']['physics'] . "<br />"; echo "Marks for qadir in maths : ";  echo $marks['qadir']['maths'] . "<br />"; echo "Marks for zara in chemistry : "  echo $marks['zara']['chemistry'] . "<br />";  ?>  </body>  </html>  This will produce the following result – |
| Marks for mohammad in physics : 35 |
|  |

|  |
| --- |
| Marks for qadir in maths : 32 Marks for zara in chemistry : 39 |
| **Strings and string operations**  A string is a sequence of characters, like "Hello world!". PHP uses several functions related to string operations  **The PHP strlen() function**  The strlen() function returns the length of a string, in characters. The example below returns the length of the string "Hello world!": strlen() is often used in loops or other functions, when it is important to know when a string ends. (i.e. in a loop, we might want to stop the loop after the last character in a string).  <?php  echo strlen("Hello world!"); // outputs 12  ?>  Output:  12  **Count The Number of Words in a String**  The PHP str\_word\_count() function counts the number of words in a string:  Example  <?php  echo str\_word\_count("Hello world!"); // outputs 2  ?>  Outputs:  2  **Reverse a String**  The PHP strrev() function reverses a string:  Example  <?php  echo strrev("Hello world!"); // outputs !dlrow olleH  ?>  Outputs:  !dlrow olleH  **Search For a Specific Text Within a String**  The PHP strpos() function searches for a specific text within a string.  If a match is found, the function returns the character position of the first match. If no match is found, it will return FALSE.  The example below searches for the text "world" in the string "Hello world!":  Example  <?php  echo strpos("Hello world!", "world"); // outputs 6  ?>  Outputs:  6  **Replace Text Within a String** |

|  |
| --- |
| The PHP str\_replace() function replaces some characters with some other characters in a string.  The example below replaces the text "world" with "Dolly":  Example  <?php  echo str\_replace("world", "Dolly", "Hello world!"); // outputs Hello Dolly!  ?>  Outputs:  Hello Dolly!  **PHP strtolower() function**  The strtolower() function returns string in lowercase letter.  **Syntax**  string strtolower ( string $string )  **Example**  <?php  $str="My name is KHAN";  $str=strtolower($str);  echo $str;  ?>  Output:  my name is khan  **2) PHP strtoupper() function**  The strtoupper() function returns string in uppercase letter.  **Syntax**  string strtoupper ( string $string )  **Example**  <?php  $str="My name is KHAN";  $str=strtoupper($str);  echo $str;  ?>  Output:  MY NAME IS KHAN  **strcasecmp()**  The strcasecmp() function is used to compare two case sensitive strings. This function was introduced in PHP3. Look at the example below:  <html>  <body>  <?php  $text1 ="Good morning";  $text2 = "Good morning"; |

|  |
| --- |
| if (strcasecmp($text1, $text2) == 0) {  echo '$text1 is equal to $text2 in a case-insensitive string comparison'; }  ?>  </body>  </html> Output  $text1 is equal to $text2 in a case-insensitive string comparison  **substr\_count()**  The substr\_count() function is used to count the number of times a substring appears in a string. This function was introduced in PHP4.  <html>  <body>  <?php print substr\_count("Hello how are you and what are you doing now?", "are"); ?>  </body>  </html> Output 2 |

|  |
| --- |
| **Multiple choice questions**   1. **Which of the following is not a valid example of variable declaration?**    1. $name    2. $address2    3. $colour\_30.    4. **$2name** 2. **Give the output for the following code**   **$first\_name = 'Chhaya';**  **$greeting1 = "Hello, my first name is $first\_name. "; echo $greeting1;**   * 1. **Hello, my first name is Chhaya.**   2. Hello, my last name is $Iast\_name.   3. $greeting1   4. Hello, my first name is  1. To concatenate 2 strings which of the following character is used    1. **.**   b. ,   1. $ 2. \*   4. are named and indexed collections of other values.   1. Strings 2. **Arrays** 3. Objects 4. Resources 5. hold references to resources external to PHP    1. Strings    2. Arrays    3. Objects    4. **Resources** 6. Which of the following is not a ―special datatype‖?    1. **Boolean**    2. Null    3. Array    4. Object 7. Which of the following is used to print the next character as a dollar not as a part of variable?    1. \d    2. $   c. \\$  **d. \$**   1. Which of the following represents conditional operator    1. **?:**    2. ??    3. ?;    4. ?.   9. Which keyword is used to halt the current iteration of a loop but it does not terminate the loop |

|  |
| --- |
| 1. **Continue** 2. Break 3. Halt 4. Stop 5. Which Letter is used to represent ―AM‖ in Date    1. **A**    2. a    3. D    4. d 6. Which of the following is used to Set the internal pointer of an array to its last element?    1. count    2. current    3. next    4. **end** 7. Which of the following determines if a variable is declared and is different than   **NULL**   * 1. **Isset**   2. Empty   3. Current   4. Set  1. An array with string index is called as    1. Numeric array    2. **Associative array**    3. Multidimensional array    4. Indexed Array   14. PHP‘s numerically indexed array begin with position   1. 1 2. 2 3. **0** 4. -1 5. Which of the following function is used to get the value of the previous element in an array?    1. last()    2. before()    3. **prev()**    4. previous() 6. Multidimensional arrays are simple arrays that have    1. One dimensional    2. **Many arrays stored in them**    3. No indexes    4. 1 element 7. For finding nonempty elements in array we use    1. is\_array ( ) function    2. sizeof ( ) function    3. array\_count ( ) function    4. **count ( ) function** 8. When we simply want iteration through looping an array values we can use    1. current ( ) |

|  |
| --- |
| 1. **foreach ( )** 2. next ( ) 3. prev() 4. Count ( ) function is identical to    1. is\_array ( ) function    2. in\_array ( ) function    3. **sizeof ( )**    4. isset ( ) function 5. How does the identity operator === compare two values?    1. **It converts them to a common compatible data type and then compares the resulting values**    2. It returns True only if they are both of the same type and value    3. If the two values are strings, it performs a lexical comparison    4. It bases its comparison on the C strcmp function exclusively 6. Which of the following is not a global variable?    1. $\_POST    2. $GLOBALS    3. $\_REQUEST    4. **$NUM** 7. loop will execute at least once.    1. While    2. **Do while**    3. For    4. Foreach   23. Which among the following has highest precedence over others?  a. ==  **b. ++**  c. &&  d. +   1. Which of the following is unary operator?    1. **--**    2. -    3. \*    4. && 2. are used in PHP to performs what is known as variable expansion    1. **Double quotes**    2. Single quotes    3. Comma    4. Semicolon   Long Question   * + 1. What are the different datatypes available in PHP     2. What is Escaping Characters? Write brief notes on Escaping Characters.     3. Explain the following with example        1. Assignment operator        2. Logical operator  1. With appropriate example explain the difference between post increment and pre increment 2. With syntax and example explain, else if and switch statements in PHP |

|  |
| --- |
| 1. Why do we need break and continue statements, explain with example? 2. Explain date () function with its syntax. List and explain any eight format which can be used with it 3. What is an array give its syntax? Explain Reset () and Sizeof () function with example 4. Define numeric and associative array with example 5. With example, explain multi-dimensional array 6. With syntax and example explain the following:    1. String length    2. Count the Number of Words in a String    3. Reverse a String    4. Search for a Specific Text Within a String 7. Explain   the following with example, and give respective output.   * 1. substr\_c   ount()   * 1. strcasec   mp() |

**PHP Forms Building form**

# UNIT IV WORKING WITH FORM

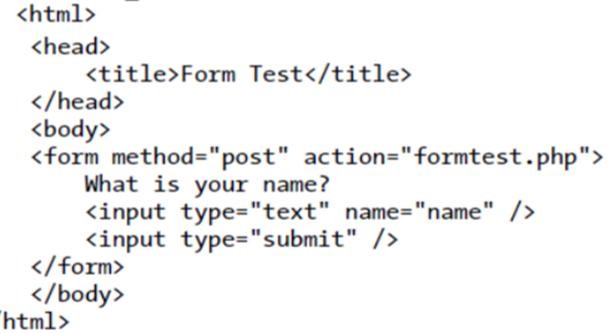
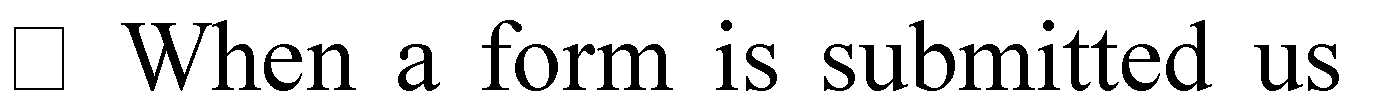
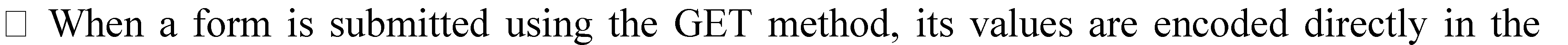
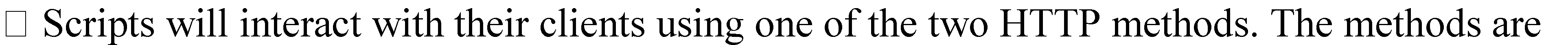
GET and POST

query string portion of the URL

query string portion of the URL

ing the POST method, its values will not be displayed the

An interactive web site requires user input, which is generally gathered through forms. As in the paper - based world, the user fills in a form and submits its content for processing. In a web application, the processing isn ‘ t performed by a sentient being; rather, it is performed by a PHP script. Thus, the script requires some sort of coded intelligence. When you fill in a paper form, you generally use a means to deliver its content (the postal service is one example) to a known address (such as a mail - order bookstore). The same logic applies to online forms. The data from an HTML form is sent to a specific location and processed.



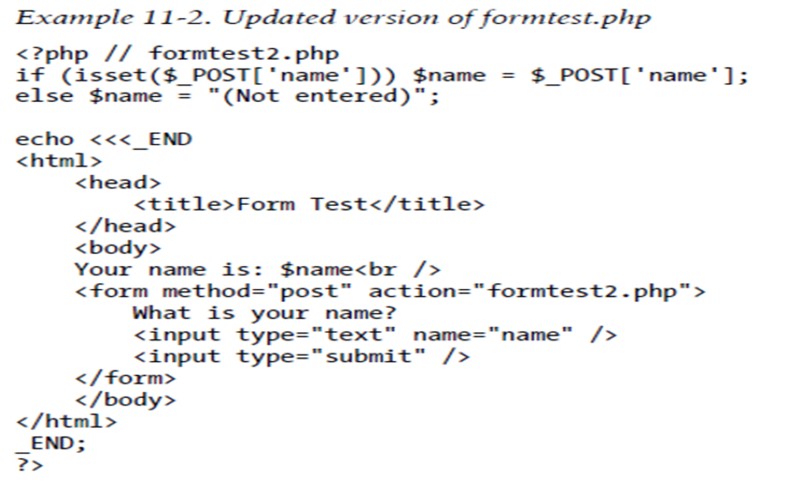
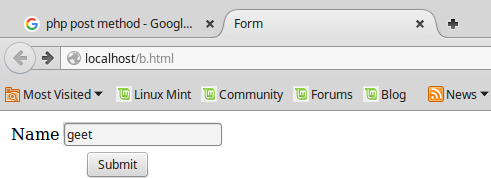
The form element is rather simple in HTML. It states where and how it will send the contents of the elements it contains once submitted. It is after that point that PHP comes into play. PHP uses a set of simple yet powerful expressions that, when combined, provide you with the means to do virtually anything you want. The PHP script receives the data from the form and uses it to perform an action such as updating the contents of a database, sending an e - mail, testing the data format, and so on.

Handling forms is a multipart process. First a form is created, into which a user can enter the required details. This data is then sent to the web server, where it is interpreted, often with some error checking. If the PHP code identifies one or more fields that require reentering, the form may be redisplayed with an error message. When the code is satisfied with the accuracy of the input, it takes some action that usually involves the database, such as entering details about a purchase.

To build a form, you must have at least the following elements:

* An opening <form> and closing </form> tag
* A submission type specifying either a GET or POST method
* One or more input fields
* The destination URL to which the form data is to be submitted

|  |
| --- |
| Inside this multiline output is some standard code for commencing an HTML document, displaying its title, and starting the body of the document. This is followed by the form, which is set to send its data using the POST method to the PHP program formtest.php, which is the name of the program itself. The rest of the program just closes all the items it opened: the form, the body of the HTML document  This is a simple variation of the commonly used ― hello world ‖ program, allowing you to take your first step into interactivity.   1. Create form1.html with your favorite text editor. 2. Enter the following code:   <html>  <head>  <title> Form </title>  </head>  <body>  <form action="b.php" method="post">  <table>  <tr>  <td> Name </td>  <td> <input type="text" name="name"/ > </td>  </tr> <tr>  <td colspan="2" style="text-align: center;" >  <input type="submit" name="submit" value="Submit" / > </td>  </tr>  </table>  </form>  </body>  </html>   1. Create another empty file named formprocess1.php , and enter the following code:   <html>  <head>  <title> Form </title>  <body>  <?php  echo ' <h1> Hello ' . $\_POST['name'] . '! </h1> ';  ?> |



</body>

</html>

1. Open form1.html in your browser.
2. Type your name in the name text box and click the Submit button.

## FORM Element

First, we ‘ ll introduce the first HTML element you ‘ ll need: form . It delimits the form ‘ s area in the page and holds the fields you want your web site users to fill in.

< form action=‖formprocess1.php‖ method=‖post‖ >

< !--form controls go here-- >

< /form >

Notice that the form element has an ending tag and two attributes. The first attribute ( action ) is the recipient page address (the form - processing script). The second attribute ( method ) is the way in which you will send the data to the recipient. You may recall that there are two separate ways of sending a form to its processing script: the POST and the GET methods.

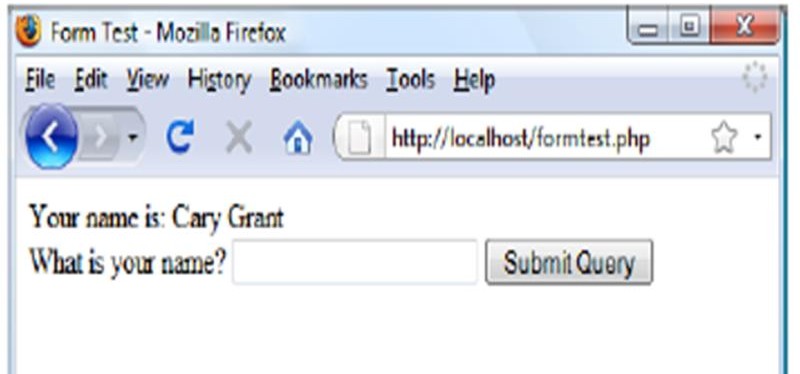
The POST method takes the data from the form fields and sends it through an HTTP header. In this case, the data cannot be seen in the URL. The GET method gets the data from the form fields, encodes it, and appends it to the destination URL.

## Retrieving Submitted Data

Example 11-1 is only one part of the multipart form handling process. If you enter a name and click on the Submit Query button, absolutely nothing will happen other than the form being redisplayed. So, now it‗s time to add some PHP code to process the data submitted by the form.Example 11-2 expands on the previous program to include data processing. Type it in (or modify *formtest.php* by adding in the new lines), save it as *formtest2.php*, and try the program for yourself.

LAMP 75

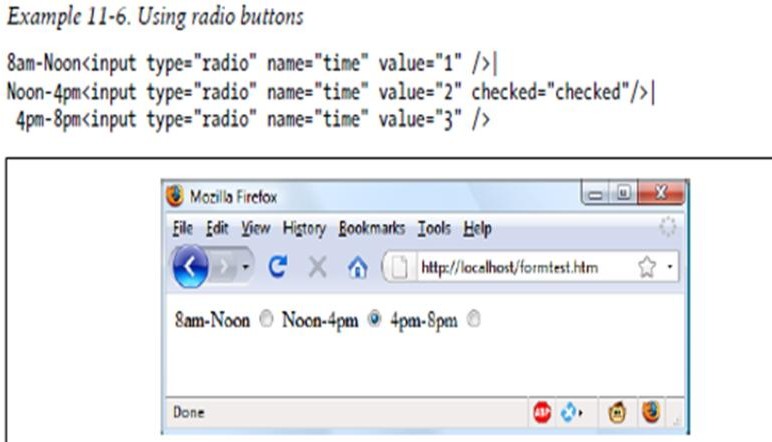
TECHNOLOGY



|  |  |  |
| --- | --- | --- |
|  | The only changes are a couple of lines at the start that check the $\_POST associative array for the field name submitted. The previous chapter introduced the $\_POST associative array, which contains an element for each field in an HTML form. In Example 11-2, the input name used was name and the form method was POST, so element name of the  $\_POST array contains the value in $\_POST['name'].  The PHP isset function is used to test whether $\_POST['name'] has been assigned a value. If nothing was posted, the program assigns the value ―(Not entered)‖; otherwise, it stores the value that was entered. Then a single line has been added after the <body> statement to display that value, which is stored in $name.  **Input Text Boxes**  Probably the type of input you will use most often is the text box. It accepts a wide range of alphanumeric text and other characters in a single-line box. The general format of a text box input is:  <input type="text" name="*name*" size="*size*" maxlength="*length*" value="*value*" /> | |
|  | |  |
|  | **Textarea**  When you need to accept input of more than a single line of text, use a text area. This is | |



|  |
| --- |
| similar to a text box but, because it allows multiple lines, it has some different parameters.Its general format looks like this:  <textarea name="*name*" cols="*width*" rows="*height*" wrap="*type*">  </textarea>  The first thing to notice is that <textarea> has its own tag and is not a subtype of the  <input> tag. It therefore requires a closing </textarea> to end input.  Instead of a default parameter, if you have default text to display, you must put it before the closing </textarea>, like this:  <textarea name="*name*" cols="*width*" rows="*height*" wrap="*type*"> This is some default text.  </textarea>  It will then be displayed and be editable by the user. To control the width and height, use the cols and rows parameters. Both use the character spacing of the current font to determine the size of the area. If you omit these values, a default input box will be created that will vary in dimensions depending on the browser used, so you should always define them to be certain about how your form will appear.  **Checkboxes**  When you want to offer a number of different options to a user, from which he can select one or more items, checkboxes are the way to go. The format to use is:  <input type="checkbox" name="*name*" value="*value*" checked="checked" />  If you include the checked parameter, the box is already checked when the browser is displayed (the string you assign to the parameter doesn‗t matter; the parameter just has to be present). If you don‗t include the parameter, the box is shown unchecked. Here is an example of an unchecked box:  I Agree <input type="checkbox" name="agree" />  If the user doesn‗t check the box, no value will be submitted. But if he does, a value of ―on‖ will be submitted for the field named agree. If you prefer to have your own value submitted instead of the word ―on‖ (such as the number 1), you could use the following  I Agree <input type="checkbox" name="agree" value="1" />  On the other hand, if you wish to offer a newsletter to your readers when submitting a form, you might want to have the checkbox already checked as the default value: Subscribe? <input type="checkbox" name="news" checked="checked" />  **To get value of multiple checked checkboxes**, name attribute in HTML input type=‖checkbox‖ tag must be initialize with an array, to do this write **[ ]** at the end of it‘s  name attribute : |
| **<form action="#" method="post">**  **<input type="checkbox" name="check\_list[]" value="C/C++"><label>C/C++</label><br/>** |
|  |



## <input type="checkbox" name="check\_list[]" value="Java"><label>Java</label><br/>

**<input type="checkbox" name="check\_list[]" value="PHP"><label>PHP</label><br/>**

## <input type="submit" name="submit" value="Submit"/>

**</form>**

## <?php

**if(isset($\_POST['submit'])){//to run PHP script on submit if(!empty($\_POST['check\_list'])){**

## // Loop to store and display values of individual checked checkbox. foreach($\_POST['check\_list'] as $selected){

**echo $selected."</br>";**

## }

**}**

## }

**?>**

## Radio buttons

Radio buttons are named after the push-in preset buttons found on many older radios, where any previously depressed button pops back up when another is pressed. They are used when you want only a single value to be returned from a selection of two more options. All the buttons in a group must use the same name and, because only a single value is returned, you do not have to pass an array.

Here, the second option, Noon–4pm, has been selected by default. Providing a default choice ensures that at least one delivery time will be selected, which users can change to one of the other two options if they prefer. Had one of the items not been already checked, the user might forget to select an option and no value would be submitted at all for the delivery time. By default, radio buttons are round



|  |  |  |
| --- | --- | --- |
|  | **Select**  The select tag lets you create a drop-down list of options, offering either single or multiple selections. It conforms to the following syntax:  <select name="*name*" size="*size*" multiple="multiple">  The parameter size is the number of lines to display. Clicking on the display causes a list to drop down showing all the options. If you use the multiple parameter, the user can select multiple options from the list by pressing the Ctrl key when clicking. So, to ask a user for his favorite vegetable from a choice of five, you might use HTML like  that in Example 11-7, which offers a single selection  *Example 11-7. Using select*  Vegetables <select name="veg" size="1">  <option value="Peas">Peas</option>  <option value="Beans">Beans</option>  <option value="Carrots">Carrots</option>  <option value="Cabbage">Cabbage</option>  <option value="Broccoli">Broccoli</option>  </select>  **PHP $\_GET Variable**  In PHP, the ***$\_GET*** variable is used to collect values from HTML forms using method *get*. Information sent from an HTML form with the GET method is displayed in the browser's address bar, and it has a limit on the amount of information to send.  Example: |  |
|  | <html>  <body>  <form action="registration.php" method="get"> Name: <input type="text" name="name"> Email: <input type="text" name="email">  <input type="submit">  </form>  </body>  </html> |  |
|  | When the user clicks on the "Submit button", the URL will be something like this: |  |

|  |  |  |
| --- | --- | --- |
|  | get_img  *registration.php* looks like this: |  |
|  | <html>  <body>  Welcome <?php echo $\_GET["name"]; ?>!  Your email address is <?php echo $\_GET["email"]; ?>  </body>  </html> |  |
|  | **When to use method="get"?**   * The variable names and values will be visible in URL if HTML forms submitted by the GET method. * The GET method is restricted to send up to *2048 characters* only. * When you submit sensitive information like passwords then should not use this method. * GET method can't be used, to send binary data like images and Word documents. * GET method data can be accessed using PHP QUERY\_STRING environment variable. * PHP $\_GET associative array is used to access all the sent information by GET method.   **PHP $\_POST Variable**  In PHP, the ***$\_POST*** variable is used to collect values from HTML forms using method *post*. Information sent from a form with the POST method is invisible and has no limits on the amount of information to send.  **Note:** However, there is an 8 MB max size for the POST method, by default (can be changed by setting the post\_max\_size in the php.ini file).  Example: |  |
|  | <html>  <body>  <form action="registration.php" method="post"> Name: <input type="text" name="name"> Email: <input type="text" name="email">  <input type="submit">  </form>  </body>  </html> |  |
|  | When the user clicks on the "Submit button", the URL will be something like this:  post_img  *registration.php* looks like this: Example: |  |

|  |  |  |
| --- | --- | --- |
|  | <html>  <body>  Welcome <?php echo $\_POST["name"]; ?>!  Your email address is <?php echo $\_POST["email"]; ?>  </body>  </html> |  |
|  | **When to use method="post"?**   * The POST method does not have any restriction on data size to be sent. The POST method can be used to send ASCII as well as binary data.   The data sent by POST method goes through HTTP header, so security depends on HTTP protocol. By using Secure HTTP, you can make sure that your information is secure.  PHP $\_POST associative array is used to access all the sent information by POST method. Variables are not visible in the URL so users can't bookmark your page.  **Disadvantages of Using the GET Method**   * The GET method is not suitable for passing sensitive information such as the username and password, because these are fully visible in the URL query string as well as potentially stored in the client browser‘s memory as a visited page.   Because the GET method assigns data to a server environment variable, the length of the URL is limited. So, there is a limitation for the total data to be sent.  **Disadvantages of using the POST Method**   * Since the data sent by the POST method is not visible in the URL, so it is not possible to bookmark the page with specific query.   POST requests are never cached  POST requests do not remain in the browser history.  **Creating a dynamic HTML Form with PHP Dealing with forms**  When building a web site, sooner or later a form will be needed. There are a number of different forms that can be used to input information. Some require entering information using keyboard, while others require selecting one or more choices by clicking with a mouse. Yet others simply involve a hidden form value that is embedded in the form itself and is not intended to be modified by the user.  Forms are the best way of gathering data from visitors. Forms can be as small as a login form, or as large as a membership subscription form asking for lots of input. Obviously, the bigger the form, the more information there is to process, and the greater the challenge.  It is possible to have multiple forms on the same page, so there must be some way to distinguish one form from the other. Furthermore, there must be a way to tell the client's browser what to once the user initiates some form action (usually by clicking a button). Both of these needs retaken care of by enclosing form entities as follows:  <FORM Action="some\_action" Method="post">  ... form entities ...  </FORM>  Two important elements make up this enclosure: |  |

|  |
| --- |
| * Action - The value passed to the keyword Action specifies what script should process the form at the Web server side.   Method - The method specifies how the data captured by the will be sent to the script. There are two possible methods:   * The get method sends all of the form information embedded into the URL. This method is rarely used, due to various language and length restrictions.   The post method sends all of the form information in the request body. This method is usually preferred over get.  A powerful feature of PHP is the way it handles HTML forms. The basic concept that is that any element in an HTML form will automatically be available to PHP scripts located at the Web server.  Here is an example of a basic HTML form:  <FORM Action="action.php" Method="post"> Your name: <INPUT Type="text" Name="name"> Your age: <INPUT Type="text" Name="age">  <INPUT Type="submit" Value="Submit">  </FORM>  This is a simple HTML form with no special tags of any kind. When a user fills in this form and hits the submit button, the data captured by the form elements is dispatched to action.php by the client browser located at the Web server.  In this file the code spec could be like:  Hi <?php echo $\_POST['name']; ?>.  You are <?php echo $\_POST['age']; ?> years old. Hi Hansel. You are 22 years old.  The $\_POST['name'] and LPOST['age'] variables are automatically set by PHP. In the above codespec, the $\_POST auto global variable contains all POST data that has been introduced. Notice the method in the HTML form is POST, if the method in the HTML form was GET then the form information would be contained in the $\_GET auto global instead.  The $\_REQUEST auto global variable can also be used, if the source of the requested data unknown. The LREQUEST auto global variable contains the merged information of GET,POST and COOKIE data.  So, using $\_POST['variable'] is the best option. However, nothing works if the web site, hosting company, puts off the track\_vars option in the php.ini file.  For example, if a page has a form to allow users to pass their comments to the web master, the should use POST. If Reload or Refresh is clicked on a page that is reached through a POST, it almost always is an error, same comment should not be posted twice, which is why these pages are not book marked or cached.  Use the GET method with an HTML form when the form provides something to the Web server, it does not actually change anything at the Web server. One disadvantage of the POST method is that the URL cannot be book marked.  **The include and require statements are identical, except upon failure:**   * require will produce a fatal error (E\_COMPILE\_ERROR) and stop the script * include will only produce a warning (E\_WARNING) and the script will continue |

|  |
| --- |
| So, if you want the execution to go on and show users the output, even if the include file is missing, use the include statement. Otherwise, in case of FrameWork, CMS, or a complex PHP application coding, always use the require statement to include a key file to the flow of execution. This will help avoid compromising your application's security and integrity, just in- case one key file is accidentally missing.  Including files saves a lot of work. This means that you can create a standard header, footer, or menu file for all your web pages. Then, when the header needs to be updated, you can only update the header include file.  **Syntax**  include '*filename*'; or  require '*filename*';  **Example 1**  Assume we have a standard footer file called "footer.php", that looks like this:  <?php  echo "<p>Copyright &copy; 1999-" . date("Y") . " W3Schools.com</p>";  ?>  To include the footer file in a page, use the include statement:  <!DOCTYPE html>  <html>  <body>  <h1>Welcome to my home page!</h1>  <p>Some text.</p>  <p>Some more text.</p>  <?php include 'footer.php';?>  </body>  </html>  The output will be  **Welcome to my home page!**  Some text. Some more text.  Copyright © 1999-2014 W3Schools.com  **PHP include vs. require**  The require statement is also used to include a file into the PHP code. However, there is one big difference between include and require; when a file is included with the **include** statement and PHP cannot find it, the script will continue to execute:  Example:  <!DOCTYPE html>  <html>  <body>  <h1>Welcome to my home page!</h1>  <?php include 'nofile.php'; echo ―Hello‖;  ?>  </body> |

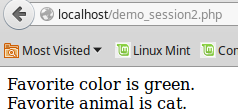
|  |
| --- |
| </html>  The output will be  **Welcome to my home page!**  Hello  If we do the same example using the **require** statement, the echo statement will not be executed because the script execution dies after the require statement returned a fatal error: The output will be  **Welcome to my home page!**  **Chapter VI - PHP functions**  **Functions in PHP**  In all programming and scripting languages, a function is a block of code that is used repetitively in a program. It saves time while developing a web page. In PHP, the concept of function is the same as in other languages. There are some in-built functions in PHP. Besides that, we can define functions as per our requirements. These are called ‗User Defined Functions‘.  Look at the elements of a function:  function: all function declarations begin with the word ‗function‘.  Name of the function: names to a function are usually assigned in accordance with its utility.  Opening and Closing parentheses (()): the opening and closing parentheses are an integral part of a function and you can insert both the opening and closing parentheses together, just after the name of the function. As the dollar sign ($) indicates the existence of a variable, these parentheses indicate the existence of a function.  Opening and Closing curly braces ({}): the opening curly brace ({) indicates the beginning of the function code and the closing curly brace marks the termination of a function.  Example:  <html>  <body>  <?php  function DisplayTitle()  {  echo ―Learning Function‖;  }  DisplayTitle();  ?>  </body>  </html>  In this example, PHP codes are embedded in HTML. Here, we have used a function  ‗DisplayTitle()‘. This function starts with the word function and indicates that the character inserted just after this word is a function. It displays the title of the tutorial. Any one who will go through this will understand the purpose of this function from its name.  **User defined function**  Defining Function |

|  |
| --- |
| The general syntax for a function is:  function *function\_name*([*parameter* [, ...]])  {  // *Statements*  }  The first line of the syntax indicates that:  A definition starts with the word function.  Following that is a name, which must start with a letter or underscore, followed by any number of letters, numbers, or underscores.  The parentheses are required.  One or more parameters, separated by commas, are optional (indicated by the square brackets, which are not part of the function syntax).  Function names are case-insensitive, so all of the following strings can refer to the print function: PRINT, Print, and PrInT.  The opening curly brace starts the statements that will execute when you call the function; a matching curly brace must close it. These statements may include one or more return statements, which force the function to cease execution and return to the calling code. If a value is attached to the return statement, the calling code can retrieve it, as we‗ll see next. **Function scope**  The origin from where a function can be accessed is called the function scope. A function, once declared, can be accessed from any section of a program. A variable scope will be local to a function, if defined within a function. Use the global key word while using a variable defined in the body part of the program.   1. **Local variables** A variable declared within a PHP function is local and can only be accessed within that function. (the variable has local scope):   <?php  $a = 5; // global scope function myTest()  {  echo $a; // local scope  }  myTest();  ?>  The script above will not produce any output because the echo statement refers to the local scope variable $a, which has not been assigned a value within this scope.  You can have local variables with the same name in different functions, because local variables are only recognized by the function in which they are declared. Local variables are deleted as soon as the function is completed.   1. **Global variables** Global scope refers to any variable that is defined outside of any function.   Global variables can be accessed from any part of the script that is not inside a function. To access a global variable from within a function, use the global keyword:  <?php |

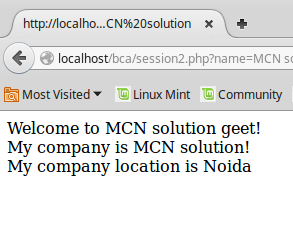
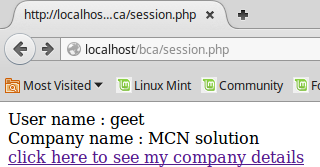
|  |
| --- |
| $a = 5; $b = 10; function myTest()  {  global $a, $b;  $b = $a + $b;  }  myTest(); echo $b;  ?>  The script above will output 15.  1. ***S***  ***tatic variables***  Another important feature of variable scoping is the *static* variable. A static variable exists only in a local function scope, but it does not lose its value when program execution leaves this scope. Normally, when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job. To do this, use the static keyword when you first declare the variable:  **Example demonstrating need for static variables**  **Example**  <?php  function myTest() { static $x = 0; echo $x;  $x++;  }  myTest();  myTest();  myTest();  ?>  Output 0  1  2  **Function arguments and return values**  In PHP, the codes are passed by both objects and arrays. We have two models to pass data in a program: ‗Pass by value‘ and ‗Pass by reference‘.  **Pass by value:** This indicates passing the variables. Here the variables are sent as an argument to a defined function. The assignment operator is used to assign it to a different variable. The receiving function or the variable gets a copy of the value of the variable. Look at the codes below: |
| <?php  function calculate($a){  $a++;  echo $a."<br/>";  }  $a=5; |
|  |

|  |
| --- |
| calculate($a); echo $a;  ?> |
| Output 6  5  **Pass by Reference:**  ‗Pass by Reference‘ is a unique feature of OOPs (Object Oriented Programming languages). It creates a new indicator but indicates to the same variable. As the ‗Pass by Value‘ creates a copy of a variable, the ‗Pass by Reference‘ creates a different name of the same variable. The ampersand sign (&) is used while passing by reference. It is always inserted just after an assignment operator. Look at the code below:  $x =& $y;  Here, the ampersand operator is used to create a reference to the variable $y. Look how this code is executed:  $x = (& $y);  In order to define arguments to pass to a function, we need to insert a list of names. These names must be inserted within parentheses in the statements of the function. Look at the code below:  function Passfunction ($argument1, $argument2) |
| <?php  function calculate(&$a){  $a++;  }  $a=5; calculate($a); echo $a;  ?> |
| Output 6  **Returning results from a function**  In PHP, a function returns a specific result when a code is called. Here the ‗return‘ keyword is used. Look at the following syntax:  return $abc;  While processing a return statement, the function is terminated. Here the value of the variable  $abc is returned to a code that is called. The values must be specified in the variable. Using one return statement, a single variable can be returned.  **What is a PHP Session?**  When you work with an application, you open it, do some changes, and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.  Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc). By default, session variables last until the user closes the browser.  So; Session variables hold information about one single user, and are available to all pages in |

|  |
| --- |
| one application.  **Start a PHP Session**  A session is started with the session\_start() function.  Session variables are set with the PHP global variable: $\_SESSION.  Now, let's create a new page called "demo\_session1.php". In this page, we start a new PHP session and set some session variables:  Example  <?php  // Start the session session\_start();  ?>  <!DOCTYPE html>  <html>  <body>  <?php  // Set session variables  $\_SESSION["favcolor"] = "green";  $\_SESSION["favanimal"] = "cat"; echo "Session variables are set.";  ?>  </body>  </html>  **Get PHP Session Variable Values**  Next, we create another page called "demo\_session2.php". From this page, we will access the session information we set on the first page ("demo\_session1.php").  Notice that session variables are not passed individually to each new page, instead they are retrieved from the session we open at the beginning of each page (session\_start()).  Also notice that all session variable values are stored in the global $\_SESSION variable:  <?php session\_start();  ?>  <!DOCTYPE html>  <html>  <body>  <?php  // Echo session variables that were set on previous page  echo "Favorite color is " . $\_SESSION["favcolor"] . ".<br>"; echo "Favorite animal is " . $\_SESSION["favanimal"] . ".";  ?>  </body>  </html> |



|  |
| --- |
| **Destroy a PHP Session**  To remove all global session variables and destroy the session, use session\_unset() and  session\_destroy():  <?php session\_start();  ?>  <!DOCTYPE html>  <html>  <body>  <?php  // remove all session variables session\_unset();  // destroy the session session\_destroy();  ?>  </body>  </html>  **Example 2:**  **Session.php**  <?php session\_start();  $\_SESSION['username']= 'geet';  ?>  <html>  <head>  <title> Session  </title>  </head>  <body>  <?php  $my = "MCN solution"; echo "User name : ";  echo $\_SESSION['username']; echo "</br>";  echo "Company name : "; echo $my;  echo "</br>";  echo "<a href=\"session2.php?name=$my\">"; echo 'click here to see my company name'; echo '</a>'  ?> |



</body>

</html>

## session2.php

<?php session\_start();

?>

<html>

<head><title></title>

</head>

<body>

<?php

echo 'Welcome to MCN solution '; echo $\_SESSION['username']; echo '!<br>';

echo 'My company is '; echo $\_GET['name']; echo '!<br>';

$loca = 'Noida';

echo 'My company location is '; echo $loca;

echo '</a>'

?>

</body>

</html>

## Output:

**Cookie**

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

Setting Cookies With PHP

A cookie is created with the setcookie() function. Syntax

<?php

setcookie(cookie\_name, cookie\_value, [expiry\_time], [cookie\_path], [domain], [secure], [httponly]);

?> HERE,

* Php―setcookie‖ is the PHP function used to create the cookie.

|  |
| --- |
| * ―cookie\_name‖ is the name of the cookie that the server will use when retrieving its value from the $\_COOKIE array variable. It‘s mandatory. * ―cookie\_value‖ is the value of the cookie and its mandatory * ―[expiry\_time]‖ is optional; it can be used to set the expiry time for the cookie such as 1 hour. The time is set using the PHP time() functions plus or minus a number of seconds greater than 0 i.e. time() + 3600 for 1 hour. * ―[cookie\_path]‖ is optional; it can be used to set the cookie path on the server. The forward slash ―/‖ means that the cookie will be made available on the entire domain. Sub directories limit the cookie access to the subdomain. * ―[domain]‖ is optional, it can be used to define the cookie access hierarchy i.e. [www.cookiedomain.com](http://www.cookiedomain.com/) means entire domain while [www.sub.cookiedomain.com](http://www.sub.cookiedomain.com/) limits the cookie access to [www.sub.cookiedomain.com](http://www.sub.cookiedomain.com/) and its sub domains. *Note it’s possible to have a subdomain of a subdomain as long as the total characters do not exceed 253 characters.* * ―[secure]‖ is optional, the default is false. It is used to determine whether the cookie is sent via https if it is set to true or http if it is set to false. * ―[Httponly]‖ is optional. If it is set to true, then only client side scripting languages   i.e. [JavaScript](https://www.guru99.com/interactive-javascript-tutorials.html) cannot access them.  The code below shows the implementation of the above example ―cookies.php‖.  <?php  setcookie("user\_name", "Guru99", time()+ 60,'/'); // expires after 60 seconds echo 'the cookie has been set for 60 seconds';  ?>  **Output:**  the cookie has been set for 60 seconds  1. The browser issues a request to retrieve the main page, *index.html*, at the website  [*http://www.webserver.com*.](http://www.webserver.com/) The first header specifies the file and the second header specifies |

|  |
| --- |
| the server.   1. When the web server at *webserver.com* receives this pair of headers, it returns some of its own. The second header defines the type of content to be sent (*text/html*) and the third one sends a cookie with the name *name* and the value *value*. Only then are the contents of the web page transferred. 2. Once the browser has received the cookie, it will then return it with every future request made to the issuing server until the cookie expires or is deleted. So, when the browser requests the new page */news.html*, it also returns the cookie *name* with the value *value*. 3. Because the cookie has already been set, when the server receives the request to send */news.html*, it does not have to resend the cookie, but just returns the requested page.   **SEND FORM INFORMATION VIA MAIL**  While the idea of simply displaying the entered user information to the browser is appealing, it doesn‘t do too much for us in the sense of actually processing the user input in a meaningful way. One way to process this information could be to send it to a particular email address, for example, the site administrator's.  This can be done by creating a small form, which captures some user information like Name, Email Address, Products Purchased, Comments etc and a php script which actually transfers the information captured via the form to a desired email address mentioned in the script.  Now that the basic form creation is in place, have a look at a form that will accept user information and send the information captured via email to the web master:  <HTML> <BODY>  <FORMAction = "emailinfo.php" Method = "post">  <b>Please take a moment to tell us what you think about our web site:</b><p>  <b>Your Name:</b><BR>  <INPUT Type="text" Name="name" Size="15" Maxlength="25" Value=""><BR>  <b>Your Email Address: </b><BR>  <INPUT Type = "text" Name="email" Size="15" Maxlength="75" value=""><BR>  <b>I frequently purchase the following products from our site:</b><BR>  <TEXTAREA Name="products" Rows="3" Cols="40"></TEXTAREA><BR>  <b>Comments: </b><BR>  <TEXTAREA Name="comments" Rows="3" Cols="40"></TEXTAREA><BR>  <INPUT Type="submit" Value="Submit">  </FORM>  </BODY> </HTML>  The output will look like |

|  |
| --- |
| **Mail() function**  The mail() function is used to mail information to a given recipient. Syntax:  boolean mail(string recipient, string subject, string message [, string addl\_headers])  The recipient is the person to whom the mail has to be sent, the subject is, of course, the subject of the email. The message is the textual body of the email, and the optional input parameter addl\_headers is used to supply any additional header information (such as HTML formatting) that is sent along with the email.  The codespec for a emailinfo.php is  <?php  // Change -$recipient to be the recipient of the form information  $recipient= ["admin@ivanbayross.c](mailto:admin@ivanbayross.com)o[m](mailto:admin@ivanbayross.com)";  // Email subj ect  $subject= "User Comments from LPOST['name']";  // Extra email headers  $headers= "From: LPOST['email'] ";  // Create a message  $message= "Messagefrom LPOST['name'] \n\n Most PurchasedProduct:  $POST['products'] \n\n Comments: \n\n LPOST['comments']";  // Send the email or produce an error  mail($recipient, $subject, $message, $headers) or die("Could not send email!");  // Send the user an appropriate message  print "Thank you LPOST['name'] for taking a moment to send us your comments!";  ?>  Output:  Thank you Sharanam Shah for taking a moment to send us your comments! |

|  |
| --- |
| **File Upload**  **Creating an upload script**  There is one global PHP variable called **$\_FILES**. This variable is an associate double dimension array and keeps all the information related to uploaded file. Five things are stored in the $\_FILES array when a file is uploaded, So if the value assigned to the input's name attribute in uploading form was **file**, then PHP would create following variables −   * **$\_FILES['file']['tmp\_name']** − the uploaded file in the temporary directory on the web server. * **$\_FILES['file']['name']** − the actual name of the uploaded file. * **$\_FILES['file']['size']** − the size in bytes of the uploaded file. * **$\_FILES['file']['type']** − the MIME type of the uploaded file. * **$\_FILES['file']['error']** – Error code resulting from file upload   **Upload1.html**  <html>  <body>  <form action="upload.php" enctype="multipart/form-data" method="post"> Your File Name <input type="file" name="file"/><br/>  <input type="submit" value="Upload" name="upload"/>  </form>  </body>  **upload.php**  <?php  if ($\_POST['upload'] )  {  if ( move\_uploaded\_file($\_FILES["file"]["tmp\_name"],  "/var/www/" . $\_FILES["file"]["name"]))  {  echo "uploaded successfully!"; echo"</br>";  echo "Upload: " . $\_FILES["file"]["name"] . "<br>";  echo "Type: " . $\_FILES["file"]["type"] . "<br>";  echo "Size: " . ($\_FILES["file"]["size"] / 1024) . " kB<br>"; echo "Stored in: " . $\_FILES["file"]["tmp\_name"];  } else {  print "Upload failed!";  } |

|  |
| --- |
| }  ?>  **Output** |

|  |
| --- |
| **UNIT IV**  1. Which predefined variables are used to retrieve information from forms?   1. $GET 2. **$\_GET** 3. $\_SET 4. GET   2. When you use the $\_GET variable to collect data, the data is visible to   1. none 2. only you 3. **everyone** 4. selected few   3. When you use the $\_POST variable to collect data, the data is visible to   1. none 2. **only you** 3. everyone 4. selected few   4. Which variable is used to collect form data sent with both the GET and POST methods?   1. $BOTH 2. $\_BOTH 3. $REQUEST 4. **$\_REQUEST**   5. Which one of the following should not be used while sending passwords or other sensitive information?   1. **GET** 2. POST 3. REQUEST 4. NEXT   6. The is a superglobal variable that stores information passed into the  script through HTTP cookies.   1. $COOKIE 2. **$\_COOKIE** 3. $SET\_COOKIE 4. $ADD\_COOKIE   7. What is the default type of ‗type‘ attribute of <input> element?   1. **Text** 2. Password 3. Numerals 4. Special Characters 5. **In PHP, cookies are set by using the**    1. **setcookie ( ) function**    2. set ( ) function    3. cookie\_set ( ) function    4. cookie() 6. variable is used to set session    1. **$\_SESSION**    2. $\_SETSESSION    3. SETSESSION    4. SESSION 7. How do we remove all session variables    1. **session\_unset();**    2. session\_destroy();    3. session\_delete(); |

|  |
| --- |
| d. session\_stop();   1. Which symbol is used for **Pass by Reference**    1. **&**    2. |    3. %    4. $   12. How do you get information from a form that is submitted using the "get" method? |
| **a. $\_GET[];** |
| b. Request.Form; |
| c. Request.QueryString; |
| d. $\_POST[];   1. Which of the following is the right one to define the function    1. function {function body}    2. datatype functionname(parameters){function body}    3. function {function body}    4. **function functionname (parameters){function body}** 2. Maximum length of get method   **a. 2048**  b. 2000  c. 2100  d. 3000   1. In PHP default behavior for user defined functions is    1. **Call-by-value**    2. Call-by-reference    3. Call-by-type    4. Call-by-loc   17. Which of the following methods would you use for sending an email?   1. **mail($to,$subject,$body)** 2. sendmail($to,$subject,$body) 3. mail(to,subject,body) 4. sendmail(to,subject,body)   18. In your PHP application you need to open a file. You want the application to and continue execution, in case the file is not found. The ideal function to be used   1. **include()** 2. require() 3. nowarn() 4. getFile(false)   19. Sessions doesn‘t allow you to   1. store persistent user preference on a site 2. save user authentication information from page to page 3. create multipage forms   **d) Doesn’t save user authentication information from page to page**   1. will produce a fatal error    1. **Require**    2. Include    3. Get    4. Post 2. Which of the following is a mandatory parameter of setcookie    1. expiry\_time    2. cookie\_path |

issue a warning is:

|  |
| --- |
| 1. **cookie\_name** 2. domain 3. can be used to define the cookie access hierarchy    1. expiry\_time    2. cookie\_path    3. cookie\_name    4. **domain** 4. A definition of function starts with .    1. **Function**    2. Datatype    3. Function name    4. void 5. things are stored in the $\_FILES array when a file is uploaded?    1. **5**    2. 8    3. 7    4. 9 6. $\_FILES is an array    1. **Associative**    2. Numeric    3. Indexed    4. Boolean   **Long Answer Questions**   1. What are the elements required to build PHP form explain with example? 2. Write a simple php form with at least 4 input fields to display values obtained from input fields 3. Explain the use of any 2 form elements in php with appropriate example. 4. Why do we use the get method, explain the variable used to fetch the values passed by get method, give the drawback of the same? 5. Write a note on POST method and $\_POST variable. 6. Why do we need include and require statements? Differentiate include and require statements in PHP. With php script. 7. Explain function declaration in PHP. Give example 8. Write a note on scope of variable 9. What is session? How the session is created explain with example. 10. Give the syntax of cookie, explain the parameters in it. 11. Explain mail() function with its syntax. Give example. 12. How to create a upload script in PHP, explain the different variables in it. |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| | | 1 | | | Peter | | | Parker | | | [peterparker@mail.com](mailto:peterparker@mail.com) | | |
| | | 2 | | | John | | | Rambo | | | [johnrambo@mail.com](mailto:johnrambo@mail.com) | | |

|  |
| --- |
| **UNIT V CHAPTER 9**  **PHP AND MYSQL**  **Built in database functions**  For many people, the main reason to learn a scripting language like PHP is to interact with databases. PHP and MySQL make up what must be the best combination for data-driven Web sites on the planet. MySQL is a small, compact database server ideal for small and not so small applications. In addition to supporting standard SQL (ANSI), it compiles on a number of platforms and has multithreading abilities on Unix servers, which make for great performance. For non-Unix people, MySQL can be run as a service on Windows NT and as a normal process in Windows 95/98 machines.  In addition to being free, the PHP-MySQL combination is also cross-platform, which means application can be developed in Windows and copied to a Unix platform. Also, PHP can be run as an external CGI process, a standalone script interpreter, or an embedded Apache module. The ability to efficiently store and retrieve large amounts of information has contributed enormously to the success of the Internet. Usually this information storage is implemented through the use of a database. Sites such as Yahoo, Bazee, and Ebay depend heavily on the reliability of their databases for storing enormous amounts of information.  It is actually surprising how useful a database can be when used with a website. There area huge variety of things that can be done when both interact with each other, from displaying simple lists to running a complete website from a database.  **What is MySQL**  MySQL is one of the most popular relational database system being used on the Web today. It is freely available and easy to install, however if you have installed Wampserver it already there on your machine. MySQL database server offers several advantages:   * MySQL is easy to use, yet extremely powerful, fast, secure, and scalable. * MySQL runs on a wide range of operating systems, including UNIX or Linux, Microsoft Windows, Apple Mac OS X, and others. * MySQL supports standard SQL (Structured Query Language). * MySQL is ideal database solution for both small and large applications. * MySQL is developed, and distributed by Oracle Corporation. * MySQL includes data security layers that protect sensitive data from intruders.   MySQL database stores data into tables like other relational database. A table is a collection of related data, and it is divided into rows and columns.  Each row in a table represents a data record that are inherently connected to each other such as information related to a particular person, whereas each column represents a specific field such as *id*, *first\_name*, *last\_name*, *email*, etc. The structure of a simple MySQL table that contains person's general information may look something like this:  + + + + +  | id | first\_name | last\_name | email |  + + + + + |

|  |
| --- |
| | 3 | Clark | Kent | [clarkkent@mail.com](mailto:clarkkent@mail.com) |  | 4 | John | Carter | [johncarter@mail.com](mailto:johncarter@mail.com) |  | 5 | Harry | Potter | [harrypotter@mail.com](mailto:harrypotter@mail.com) |  + + + + +  **Areas of PHP and MySQL being used together are:**   * Banner Rotation: Here each banner is a PHP script. Using database random banners are picked up and shown to the visitor. In addition to that a track of the number of times the banner has been viewed can be kept. Banners can be added, changed or edited just by making changes in the database and the script picks the correct banners for all the pages on the site. * Forums: Hundreds of forums (message boards) on the internet are run using PHP and MySQL. These are much more efficient than other systems that create a page for each message and offer a wide variety of options. All the pages in the forum can be updated by changing one script. * Databases: One quite obvious example is sites, which get all there information from a database. For example Script Avenue is run by a few scripts, which gain all their information from a large database. All the different script categories can be accessed in one script by just changing the URL to access a different part of the database. * Web sites: For a large website and changing the design can take a very long time to update and upload all the pages. With PHP and MySQL the whole website could be just one or two PHP scripts. These would access a MySQL database to get the information for the pages. To update the website's design just change one page.   1. Talking to MySQL Databases with SQL   SQL, the Structured Query Language, is a simple, standardized language for communicating with relational databases like MySQL. With SQL you can perform any database-related task, such as creating databases and tables, saving data in database tables, query a database for specific records, deleting and updating data in databases.  Look at the following standard SQL query that returns the email address of a person whose first name is equal to 'Peter' in the *persons* table:  SELECT email FROM persons WHERE first\_name="Peter"  If you execute the SQL query above it will return the following record: [peterparker@mail.com](mailto:peterparker@mail.com)  Ways of Connecting to MySQL through PHP  In order to store or access the data inside a MySQL database, you first need to connect to the MySQL database server. PHP offers two different ways to connect to MySQL server: **MySQLi** (Improved MySQL) and **PDO** (PHP Data Objects) extensions.  While the PDO extension is more portable and supports more than twelve different databases, MySQLi extension as the name suggests supports MySQL database only. MySQLi extension however provides an easier way to connect to, and execute queries on, a MySQL database server. Both PDO and MySQLi offer an object-oriented API, but MySQLi also offers a procedural API which is relatively easy for beginners to understand.   * 1. **Connecting to MySQL Database Server**   In PHP you can easily do this using the mysqli\_connect() function. All communication between PHP and the MySQL database server takes place through this connection. Here're the basic syntaxes for connecting to MySQL using MySQLi and PDO extensions: |

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

|  |
| --- |
| *Syntax: MySQLi, Procedural way* |
| $link = mysqli\_connect("hostname", "username", "password", "database"); |
| *Syntax: MySQLi, Object Oriented way* |
| $mysqli = new mysqli("hostname", "username", "password", "database"); |
| *Syntax: PHP Data Objects (PDO) way* |
| $pdo = new PDO("mysql:host=hostname;dbname=database", "username", "password"); |
| The *hostname* parameter in the above syntax specify the host name (e.g. localhost), or  IP address of the MySQL server, whereas the *username* and *password* parameters specifies the credentials to access MySQL server, and the *database* parameter, if provided will specify the default MySQL database to be used when performing queries.  **Closing the MySQL Database Server Connection**  The connection to the MySQL database server will be closed automatically as soon as the execution of the script ends. You can disconnect from the MySQL database anytime using another PHP function **mysql\_close()**. This function takes a single parameter, which is a connection returned by the **mysql\_connect()** function.  Syntax  bool mysql\_close ( resource $link\_identifier );  If a resource is not specified, then the last opened database is closed. This function returns true if it closes the connection successfully otherwise it returns false.  <?php  /\* Attempt MySQL server connection. Assuming you are running MySQL server with default setting (user 'root' with no password) \*/  $link = mysqli\_connect("localhost", "root", "");  // Check connection if($link === false){  die("ERROR: Could not connect. " . mysqli\_connect\_error());  }  // Print host information  echo "Connect Successfully. Host info";  // Close connection mysqli\_close($link);  ?>  **4.4** Creating MySQL Database Using PHP  PHP uses **mysql\_query** function to create or delete a MySQL database. This function takes two parameters and returns TRUE on success or FALSE on failure.  Syntax  bool mysql\_query( sql, connection );  **sql**  Required - SQL query to create or delete a MySQL database  **connection**  Optional - if not specified, then the last opened connection by mysql\_connect will be used.  <html>  <center> |

|  |
| --- |
| <h3>Display All</h3>  <?php  $dbhost = 'localhost';  $dbuser = 'root';  $dbpass = 'password';  $conn = mysql\_connect("localhost",$dbuser, $dbpass) or die("conn to db failed!");  if(! $conn ) {  die('Could not connect: ' . mysql\_error());  }  echo 'Connected successfully<br />';  $sql = 'CREATE DATABASE TUTORIALS';  $retval = mysql\_query( $sql, $conn );  if(! $retval ) {  die('Could not create database: ' . mysql\_error());  }  echo "Database TUTORIALS created successfully\n"; mysql\_close($conn);  ?>  </tr>  </table>  </html>  OUTPUT    Creating Tables inside MySQL Database Using PHP  Now it's time to create some tables inside the database that will actually hold the data. A table organizes the information into rows and columns.  To begin with, the table creation command requires the following details −   * Name of the table * Name of the fields * Definitions for each field Syntax   Here is a generic SQL syntax to create a MySQL table − CREATE TABLE table\_name (column\_name column\_type);  Now, we will create the following table in the **TUTORIALS** database. create table tutorials\_tbl(  tutorial\_id INT NOT NULL AUTO\_INCREMENT, tutorial\_title VARCHAR(100) NOT NULL, tutorial\_author VARCHAR(40) NOT NULL, submission\_date DATE,  PRIMARY KEY ( tutorial\_id )  );  Here, a few items need explanation −   * Field Attribute **NOT NULL** is being used because we do not want this field to be NULL. So, if a user will try to create a record with a NULL value, then MySQL will raise an error. |

|  |
| --- |
| * Field Attribute **AUTO\_INCREMENT** tells MySQL to go ahead and add the next available number to the id field. * Keyword **PRIMARY KEY** is used to define a column as a primary key. You can use multiple columns separated by a comma to define a primary key.   The SQL [CREATE TABLE](https://www.tutorialrepublic.com/sql-tutorial/sql-create-table-statement.php) statement is used to create a table in database.  Let's make a SQL query using the CREATE TABLE statement, after that we will execute this SQL query through passing it to the PHP mysqli\_query() function to finally create our table.  <html>  <center>  <h3>Display All</h3>  <?php  $dbhost = 'localhost';  $dbuser = 'root';  $dbpass = 'password';  $conn = mysql\_connect("localhost",$dbuser, $dbpass) or die("conn to db failed!");  if(! $conn ) {  die('Could not connect: ' . mysql\_error());  }  echo 'Connected successfully<br />';  $sql = "CREATE TABLE tutorials\_tbl( tutorial\_id INT NOT NULL AUTO\_INCREMENT, tutorial\_title VARCHAR(100) NOT NULL, tutorial\_author VARCHAR(40) NOT NULL,  submission\_date DATE, PRIMARY KEY ( tutorial\_id )); ";  mysql\_select\_db( 'TUTORIALS' );  $retval = mysql\_query( $sql, $conn ); if(! $retval ) {  die('Could not create table: ' . mysql\_error());  }  echo "Table created successfully\n"; mysql\_close($conn);  ?>  </body>  </tr>  </table>  </html>  **OUTPUT**  **Selecting a MySQL Database Using PHP Script**  PHP provides function  **mysql\_select\_db** to select a database. It returns TRUE on success or FALSE on failure.  Syntax  bool mysql\_select\_db( db\_name, connection ); |

|  |
| --- |
| **db\_name**  Required − MySQL Database name to be selected  **connection**  Optional − if not specified, then the last opened connection by mysql\_connect will be used. Example  <html>  <center>  <h3>Display All</h3>  <?php  $dbhost = 'localhost';  $dbuser = 'root';  $dbpass = 'password';  $conn = mysql\_connect("localhost",$dbuser, $dbpass) or die("conn to db failed!"); if(! $conn ) {  die('Could not connect: ' . mysql\_error());  }  echo 'Connected successfully'; mysql\_select\_db( 'TUTORIALS' );  mysql\_close($conn);  ?>  </body>  </tr>  </table>  </html>  **OUTPUT**    **Deleting a Database**  If a database is no longer required then it can be deleted forever. You can use pass an SQL command to **mysql\_query** to delete a database.  Example  Try out following example to drop a database.  <?php  $dbhost = 'localhost:3036';  $dbuser = 'root';  $dbpass = 'rootpassword';  $conn = mysql\_connect($dbhost, $dbuser, $dbpass);  if(! $conn ) {  die('Could not connect: ' . mysql\_error());  }  $sql = 'DROP DATABASE test\_db';  $retval = mysql\_query( $sql, $conn ); if(! $retval ) {  die('Could not delete database db\_test: ' . mysql\_error());  }  echo "Database deleted successfully\n"; |

mysql\_close($conn);

?>

## Mysql Function in php Connect

Connection to Mysql database can be established by using mysql\_connect function. We can check the success of the function by checking the result. We will get a true result in case connection is established.This function takes three parameters, first one is hostname then user-id and them password. We can give the port number along with the hostname also

Here is the function to connect to mysql database mysql\_connect ("$servername","$dbuser","$dbpassword"); **Select Database**

The mysql\_select\_db() function sets the active MySQL database. This function returns TRUE on success, or FALSE on failure.

Syntax mysql\_select\_db(database,connection)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| database | Required. Specifies the database to select. |
| connection | Optional. Specifies the MySQL connection. If not specified, the last connection opened by mysql\_connect() or mysql\_pconnect() is used. |

Example

<?php

$con = mysql\_connect("localhost", "peter", "abc123"); if (!$con)

{

die('Could not connect: ' . mysql\_error());

}

$db\_selected = mysql\_select\_db("test\_db", $con); if (!$db\_selected)

{

die ("Can\'t use test\_db : " . mysql\_error());

}

mysql\_close($con);

?>

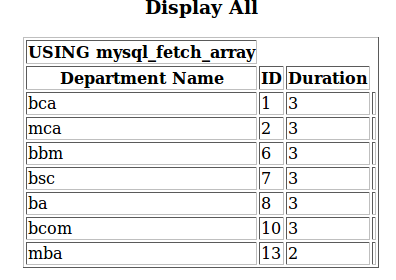
## Close

To close a mysql connection we can use mysql\_close() function. It can takes a optional parameter as link and closes it. If no link identifier is specified then last opened connection is closed. It is not necessary to use mysql\_close() function as all connections are closed at the end of the script execution. Here is the function

mysql\_close()

## Accessing MySQL database from web with PHP : Querying the database:

mysql\_db\_query — Queries a MySQL database.



**mysql\_query()** executes *query* on *database* using the MySQL server connection referenced by *connection* . If no connection handle is specified in the *connection* argument, the last opened connection will be used by default. If no connection is open, **mysql\_query()** attempts to connect to a MySQL database by calling [**mysql\_connect()**](http://www.phpdig.net/ref/rn41re774.html) without arguments.

The value returned depends on the query made. **SELECT**, **DESCRIBE**, **EXPLAIN**, and **SHOW** queries return a MySQL result handle if successful or FALSE on failure. Note that the previous types of queries are considered to have failed only if they're malformed

mysql\_query ("SELECT \* FROM database.table");

## Retriving Query results

*mysql\_fetch\_row()* function returns a record by taking a result identifier. Here it returns the set of data as an array. To get the value from the array we have to use array offset staring from 0. Each call to this mysql\_fetch\_row function returns the next record. Here one record is returned at a time and returns false if there is no more record to return. So we can easily use this function inside a while loop and display all the records. Each time at the starting of the while loop the mysql\_fetch\_row() function is checked and if true then the loop is executed and the records are displayed. Here is one example and we have used our student table for this.

Syntax mysql\_fetch\_row(data)

data- Required. Specifies which data pointer to use. The data pointer is the result from the mysql\_query() function

Examples

<html>

<center>

<h3>Display All</h3>

<?php

$connect = mysql\_connect("localhost","college","college","college") or die("conn to db failed!"); mysql\_select\_db ("college") or die ("Db not found" );

$result = mysql\_query("SELECT \* FROM department");

?>

<table border=1>

<tr><th> USING mysql\_fetch\_array

<tr><th>Department Name</th><th>ID </th> <th>Duration</th>

</tr>

<?php while($nt=mysql\_fetch\_row($result)){

echo "<tr><td>$nt[0]<td>$nt[1]<td>$nt[2]<td>$nt[3] "; echo "</tr>";

}

?>

</tr></table></html>

Here is the result of above code

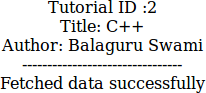
LAMP TECHNOLOGY 106



|  |  |  |  |
| --- | --- | --- | --- |
| **Inserting Data into a MySQL Database Table**  The INSERT INTO statement is used to insert new rows in a database table.  Let's make a SQL query using the INSERT INTO statement with appropriate values, after that we will execute this insert query through passing it to the PHP mysqli\_query() function to insert data in table.  <html>  <center>  <h3>Display All</h3>  <?php  $connect = mysql\_connect("localhost","root","password") or die("conn to db failed!"); mysql\_select\_db ("TUTORIALS") or die ("Db not found" );  $sql = "INSERT INTO tutorials\_tbl VALUES('', 'C++', 'Balaguru Swami')";  $result=mysql\_query($sql) or die(mysql\_error()); if($result)  {  ?>  <SCRIPT language=JavaScript> alert("Record Inserted Successfully...");  </script>  <?php  }  ?>  </body>  </html>  **Inserting Multiple Rows into a Table**  You can also insert multiple rows into a table with a single insert query at once. To do this, include multiple lists of column values within the INSERT INTO statement, where column values for each row must be enclosed within parentheses and separated by a comma.  Let's insert few more rows into the *persons* table, like this:  <?php  /\* Attempt MySQL server connection. Assuming you are running MySQL server with default setting (user 'root' with no password) \*/  $link = mysqli\_connect("localhost", "root", "", "demo");  // Check connection } | | |  |
|  | if($link === false){ |  |
|  | die("ERROR: Could not connect. " . mysqli\_connect\_error()); | // Attempt insert  query execution |
|  | $sql = "INSERT INTO persons (first\_name, last\_name, email) VALUES | | |
| ('John',  'Rambo', 'johnrambo@mail.com'), | | |  |
| ('Clark', 'Kent', 'clarkkent@mail.com'), | |  |
| ('John', 'Carter', 'johncarter@mail.com'), | | |
| ('Harry', 'Potter', 'harrypotter@mail.com')"; | |  |
| if(mysqli\_query($link, $sql)){ | | |
| echo "Records added successfully."; | |  |
| } else{ | | |
| echo "ERROR: Could not able to execute $sql. " . mysqli\_error($link); | |  |
|  | | |

|  |  |
| --- | --- |
| }  // Close connection | |
| mysqli\_close($link); |  |
| ?>  Now, go to phpMyAdmin (<http://localhost/phpmyadmin/)> and check out the *persons* table data inside *demo* database. You will find the value for the *id* column is assigned automatically by incrementing the value of previous *id* by 1.  **Insert Data into a Database from an HTML Form**  In the previous section, we have learned how to insert data into database from a PHP script. Now, we'll see how we can insert data into database obtained from an HTML form. Let's create an HTML form that can be used to insert new records to *persons* table.  **db\_connect.php**  <?php  $connect = mysql\_connect("localhost","root","password","bca") or die("conn to db failed!"); mysql\_select\_db ("bca") or die ("Db not found" );  ?>  **employee.php**  <html>  <head>  <title>Add New Record in MySQL Database</title>  </head>  <body>  <?php require('db\_connect.php');  $emp\_name = $\_POST['ename'];  $emp\_address = $\_POST['eadd'];  $emp\_salary = $\_POST['sal'];  $sql = "INSERT INTO emp VALUES('$emp\_name','$emp\_address','$emp\_salary')";  $result=mysql\_query($sql) or die(mysql\_error()); if($result)  {  echo ‖ Inserted‖;  }  else  {  echo ‖Not Inserted‖;  }  ?>  <form method = "post" action = "<?php $\_PHP\_SELF ?>">  <table width = "400" border = "0" cellspacing = "1" cellpadding = "2">  <tr>  <td width = "100">Employee Name</td>  <td><input name = "ename" type = "text"></td>  </tr>  <tr>  <td width = "100">Employee Address</td>  <td><input name = "eadd" type = "text" ></td>  </tr> | |

|  |
| --- |
| <tr>  <td width = "100">Employee Salary</td>  <td><input name = "sal" type = "text" ></td>  </tr>  <tr>  <td width = "100"> </td>  <td> </td>  </tr>  <tr>  <td width = "100"> </td>  <td>  <input name = "add" type = "submit" id = "add" value = "Add Employee">  </td>  </tr>  </table>  </form>  </body>  </html>  **Selecting data from database**  The SQL **SELECT** command is used to fetch data from the MySQL database. You can use this command at mysql> prompt as well as in any script like PHP.  Syntax  Here is generic SQL syntax of SELECT command to fetch data from the MySQL table − SELECT field1, field2,...fieldN  FROM table\_name1, table\_name2... [WHERE Clause]  [OFFSET M ][LIMIT N]   * You can use one or more tables separated by comma to include various conditions using a WHERE clause, but the WHERE clause is an optional part of the SELECT command. * You can fetch one or more fields in a single SELECT command. * You can specify star (\*) in place of fields. In this case, SELECT will return all the fields. * You can specify any condition using the WHERE clause. * You can specify an offset using **OFFSET** from where SELECT will start returning records. By default, the offset starts at zero. * You can limit the number of returns using the **LIMIT** attribute.   **Filtering the records Where clause**  We have seen the SQL **SELECT** command to fetch data from a MySQL table. We can use a conditional clause called the **WHERE Clause** to filter out the results. Using this WHERE clause, we can specify a selection criteria to select the required records from a table.  Syntax  The following code block has a generic SQL syntax of the SELECT command with the WHERE clause to fetch data from the MySQL table −  SELECT field1, field2,...fieldN table\_name1, table\_name2... [WHERE condition1 [AND [OR]] condition2.....   * You can use one or more tables separated by a comma to include various conditions using a WHERE clause, but the WHERE clause is an optional part of the SELECT command. * You can specify any condition using the WHERE clause. * You can specify more than one condition using the **AND** or the **OR** operators. * A WHERE clause can be used along with DELETE or UPDATE SQL command also to |



|  |
| --- |
| specify a condition.  The **WHERE** clause works like an **if condition** in any programming language. This clause is used to compare the given value with the field value available in a MySQL table. If the given value from outside is equal to the available field value in the MySQL table, then it returns that row.  <html>  <center>  <h3>Display All</h3>  <?php  $conn = mysql\_connect("localhost","root","password") or die("conn to db failed!");  $sql = 'SELECT tid, title, author FROM tutorials\_tbl where tid="2"'; mysql\_select\_db('TUTORIALS');  $retval = mysql\_query( $sql, $conn );  if(! $retval ) {  die('Could not get data: ' . mysql\_error());  }  while($row = mysql\_fetch\_array($retval)) { echo "Tutorial ID :{$row['tid']} <br> ".  "Title: {$row['title']} <br> ".  "Author: {$row['author']} <br> ".  " <br>";  }  echo "Fetched data successfully\n"; mysql\_close($conn);  ?>  </body>  </html> OUTPUT  **LIMIT clause**  In MySQL the **LIMIT** clause is used with the SELECT statement to  restrict the number of rows in the result set. This feature is very helpful for optimizing the page loading time as well as to enhance the readability of a website. For example you can divide the large number of records in multiple pages using pagination, where limited number of records will be loaded on every page from the database when a user request for that page by clicking on pagination link. The Limit Clause accepts one or two arguments which are offset and count.The value of both the parameters can be zero or positive integers.  **Offset:**It is used to specify the offset of the first row to be returned.  **Count:**It is used to specify the maximum number of rows to be returned.  The Limit clause accepts one or two parameters, whenever two parameters are specified, the first is the offset and the second denotes the count whereas whenever only one parameter is specified, it denotes the number of rows to be returned from the beginning of the result set. |

|  |
| --- |
| **Syntax:** |
| SELECT column1, column2, ... FROM table\_name  LIMIT offset, count; |
| To retrieve the first three rows from the table ―Data‖, we will use the following query: |
| SELECT \* FROM Data LIMIT 3; |
| To retrieve the rows 2-3(inclusive) from the table ―Data‖, we will use the following query: |
| SELECT \* FROM Data LIMIT 1, 2; |
| **Updating**  There may be a requirement where the existing data in a MySQL table needs to be modified. You can do so by using the SQL **UPDATE** command. This will modify any field value of any MySQL table.  Syntax  The following code block has a generic SQL syntax of the UPDATE command to modify the data in the MySQL table −  UPDATE table\_name SET field1 = new-value1, field2 = new-value2 [WHERE Clause]  You can update one or more field altogether.  You can specify any condition using the WHERE clause. You can update the values in a single table at a time.  The WHERE clause is very useful when you want to update the selected rows in a tab  Eg: $result=mysql\_query("update user set fname='$fname',lname='$lname',age= '$age' where id='$id' ");  **Example**  <html>  <center>  <h3>Display All</h3>  <?php  $conn = mysql\_connect("localhost","root","password") or die("conn to db failed!");  $sql = 'SELECT tid, title, author FROM tutorials\_tbl where tid="2"'; if(! $conn ) {  die('Could not connect: ' . mysql\_error());  }  $sql = 'UPDATE tutorials\_tbl SET title="Learning JAVA" WHERE tid=2';  mysql\_select\_db('TUTORIALS');  $retval = mysql\_query( $sql, $conn ); if(! $retval ) {  die('Could not update data: ' . mysql\_error());  }  echo "Updated data successfully\n"; mysql\_close($conn);  ?> |

|  |
| --- |
| </body>  </html>  **Updating value from form db\_connect.php**  <?php  $connect = mysql\_connect("localhost","root","password","bca") or die("conn to db failed!"); mysql\_select\_db ("bca") or die ("Db not found" );  ?>  **update.php**  <?php  // php code to Update data from mysql database Table if(isset($\_POST['update']))  {  require('db\_connect.php');  // get values form input text and number  $id = $\_POST['id'];  $fname = $\_POST['fname'];  $lname = $\_POST['lname'];  $age = $\_POST['age'];  // mysql query to Update data  $result=mysql\_query("update user set fname='$fname',lname='$lname',age= '$age' where id='$id' ");  echo "ll"; if($result)  {  echo 'Data Updated';  }else{  echo 'Data Not Updated';  }  mysqli\_close($connect);  }  ?>  <!DOCTYPE html>  <html>  <head>  <title> PHP UPDATE DATA </title>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  </head>  <body>  <form action="update.php" method="post">  ID To Update: <input type="text" name="id" required><br><br>  New First Name:<input type="text" name="fname" required><br><br> New Last Name:<input type="text" name="lname" required><br><br> New Age:<input type="text" name="age" required><br><br>  <input type="submit" name="update" value="Update Data">  </form> |

|  |
| --- |
| </body>  </html>  **Deleting data from database**  If you want to delete a record from any MySQL table, then you can use the SQL command **DELETE FROM**. You can use this command at the mysql> prompt as well as in any script like PHP.  Syntax  The following code block has a generic SQL syntax of the DELETE command to delete data from a MySQL table.  DELETE FROM table\_name [WHERE Clause]   * If the WHERE clause is not specified, then all the records will be deleted from the given MySQL table. * You can specify any condition using the WHERE clause. * You can delete records in a single table at a time.   The WHERE clause is very useful when you want to delete selected rows in a table. Eg: $sql = 'DELETE FROM tutorials\_tbl WHERE tid = 1';  <html>  <center>  <h3>Display All</h3>  <?php  $conn = mysql\_connect("localhost","root","password") or die("conn to db failed!"); if(! $conn ) {  die('Could not connect: ' . mysql\_error());  }  $sql = 'DELETE FROM tutorials\_tbl WHERE tid = 1'; mysql\_select\_db('TUTORIALS');  $retval = mysql\_query( $sql, $conn );  if(! $retval ) {  die('Could not delete data: ' . mysql\_error());  }  else  echo "Deleted data successfully\n"; mysql\_close($conn);  ?>  **Deleting value from database via form db\_connect.php**  <?php  $connect = mysql\_connect("localhost","root","password","bca") or die("conn to db failed!"); mysql\_select\_db ("bca") or die ("Db not found" );  **?>**  **delete.php**  <html>  <head>  <title>Delete a Record from MySQL Database</title> |

|  |
| --- |
| </head>  <body>  <?php  require('db\_connect.php'); if(isset($\_POST['delete']))  {  $emp\_id = $\_POST['emp\_id'];  $result=mysql\_query("DELETE FROM employees WHERE id = $emp\_id") ;  }  ?>  <form method = "post" action = "<?php $\_PHP\_SELF ?>">  <table width = "400" border = "0" cellspacing = "1" cellpadding = "2">  <tr>  <td width = "100">Employee ID</td>  <td><input name = "emp\_id" type = "text" id = "emp\_id"></td>  </tr>  <tr>  <td width = "100"> </td>  <td> </td>  </tr>  <tr>  <td width = "100"> </td>  <td>  <input name = "delete" type = "submit" id = "delete" value = "Delete">  </td>  </tr>  </table>  </form>  </body>  </html>  **Different ways to fetch the records from database Fetch Row**  *mysql\_fetch\_row()* function returns a record by taking a result identifier. Here it returns the set of data as an array. To get the value from the array we have to use array offset staring from 0. Each call to this mysql\_fetch\_row function returns the next record. Here one record is returned at a time and returns false if there is no more record to return. So we can easily use this function inside a while loop and display all the records. Each time at the starting of the while loop the mysql\_fetch\_row() function is checked and if true then the loop is executed and the records are displayed.  <?  $query=mysql\_query("select \* from student"); echo mysql\_error();  echo "<b>id,name,class,mark</b><br>"; while($nt=mysql\_fetch\_row($query)){ echo "$nt[0],$nt[1],$nt[2],$nt[3] <br>";  }  ?>  **Fetch assoc mysql\_fetch\_assoc**  The function returns an associative array of strings that corresponds to the fetched row, or FALSE |

|  |
| --- |
| if there are no more rows. The associativity array tells us about the key value pair, whereas the key tells about any column name and the value tells about the row value.  Here we can map the column name as key and value as row. For example. Key is ID and value is corresponding name.  **<?php**  $dbhost = 'localhost';  $dbuser = 'root';  $dbpass = 'password';  $conn = mysql\_connect("localhost",$dbuser, $dbpass,"bca") or die("conn to db failed!"); mysql\_select\_db("bca");  $result = mysql\_query("SELECT \* FROM login"); while ($row = mysql\_fetch\_assoc($result)) {  echo $row["id"]; echo $row["uname"];  }  ?>  **Output 1001**  **geet Fetch array**  mysql\_fetch\_array--Fetch a result row as an associative array, a numeric array, or both Description  array **mysql\_fetch\_array** ( result )  Parameters  *result*  The result **resource** that is being evaluated. This result comes from a call to **mysql\_query()**. Returns an array that corresponds to the fetched row, or FALSE if there are no more rows. The type of returned array depends on how *result\_type* is defined. By using MYSQL\_BOTH (default), you'll get an array with both associative and number indices. Using MYSQL\_ASSOC, you only get associative indices (as **mysql\_fetch\_assoc()** works), using MYSQL\_NUM, you only get number indices (as **mysql\_fetch\_row()** works).  Parameters  *result*  The result resource that is being evaluated. This result comes from a call to mysql\_query()  <?php  $dbhost = 'localhost';  $dbuser = 'root';  $dbpass = 'password';  $conn = mysql\_connect("localhost",$dbuser, $dbpass,"bca") or die("conn to db failed!"); mysql\_select\_db("bca");  $result = mysql\_query("SELECT \* FROM login"); while ($row = mysql\_fetch\_array($result)) {  echo $row[0]; echo $row["uname"];  }  ?>  **Output**  1001  geet  **Execute Query**  This function mysql\_query() is used to pass a sql query to mysql database. We can write any sql query like insert , select , update , delete etc and pass the query to mysql database. The result of |

|  |
| --- |
| the execution of the query will be known by monitoring the status. We will get true or false based on the success or failure status of the query. We can specify and connecting link as an optional parameter. If no connecting link is specified it assumes default connection.  <?Php  $query="select name,class,mark from student";  $status=mysql\_query($query) or die( "query failed"); if($status){echo "Query is successful";}  else {echo "Qyery failed";}  ?>  **Error**  We can display error message in case of an error generated by mysql query. This meaning full error message gives idea one the problem or bugs in the script. We can print the error message by using mysql function mysql\_error(). This function returns the error message associated with most recently executed query.  $querry = mysql\_query("SELECT new\_field FROM student"); echo "Error message = ".mysql\_error(); |

|  |
| --- |
| **Multiple Choice questions**  1. [Which Statement Is Used To Insert A New Data In A Database?](https://www.onlineinterviewquestions.com/mysql-mcq/#_blank)   1. INSERT NEW 2. Add New 3. **INSERT INTO** 4. UPDATE   [2. Which Of These Commands Will Delete A Table Called XXX If You Have Appropriate](https://www.onlineinterviewquestions.com/mysql-mcq/#_blank) [Authority:](https://www.onlineinterviewquestions.com/mysql-mcq/#_blank)   1. **DROP TABLE XXX** 2. DELETE XXX WHERE Confirm = "YES" 3. DROP XXX 4. DROP XXX WHERE Confirm = "YES"   3. [MySQL Runs On Which Operating Systems?](https://www.onlineinterviewquestions.com/mysql-mcq/#_blank)   1. Unix And Linux Only 2. **Unix, Linux, Windows And Others** 3. Linux And Mac OS-X Only 4. Any Operating System At All   4. [Which Of The Following Is Available In MySQL:](https://www.onlineinterviewquestions.com/mysql-mcq/#_blank)   1. CREATE VIEW 2. CREATE SCHEMA 3. CREATE TRIGGER 4. **CREATE DATABASE**   5. [Which Of These Is Not A Valid Name For A Column](https://www.onlineinterviewquestions.com/mysql-mcq/#_blank)   1. To 2. **From** 3. Far 4. Near |

1. Which of the following is NOT available in MySQL:

## FETCH

1. LIKE
2. JOIN
3. SELECT
4. To use MySQL on your computer, you'll need?
5. FTP and Telnet

## Some sort of client program to access the databases

1. A Browser
2. Perl, PHP or Java
3. MySQL is

|  |
| --- |
| a) Protocol |
| b) A Programming language |
| c) A technique for writing reliable programs |
| **d) A Relational Database Management System** |

1. Which of the following is used to delete an entire MYSQL database?
2. mysql\_drop\_database
3. mysql\_drop\_entiredb

## mysql\_drop\_db

1. mysql\_drop\_dbase
2. Which of the following commands should be used to create a database named

―student‖?

1. CREATE student

## CREATE DATABASE student

1. DATABASE /student
2. DATABSE student
3. A SELECT command without a WHERE clause returns?
4. All the records from a table that match the previous WHERE clause

## All the records from a table, or information about all the records

1. SELECT is invalid without a WHERE clause
2. Nothing
3. What does SQL stand for?
4. Strong question Language

## Structured Query language

1. Structured question language
2. Strong query language
3. With SQL, how do you select a column named "FirstName" from a table named "Persons"?
4. Select Persons.Firstname
5. Extract firstname from persons

## Select firstname from persons

1. Select firstname
2. With SQL, how do you select all the columns from a table named "Persons"?
3. Select persons
4. select [all] from persons

## Select \* from persons

1. Select \*.persons
2. The result of select command is

|  |
| --- |
| 1. File 2. Report 3. **Table** 4. Form   16. Full form of **PDO**   1. **PHP Data Objects** 2. Permanent Data Object 3. Public Data Object 4. Private Data Object   17. Output on executing mysql\_fetch\_row()   1. **Array** 2. String 3. Number 4. Object   18. Using SELECT will return all the fields.   1. **\*** 2. .   c) +  d) &  19. Which of the following is not true about MYSQL   1. Fast 2. Easy 3. **Not Secure** 4. Scalable   20. Which function returns an error massage?   1. **mysql\_error();** 2. mysql\_errors(); 3. mysql–error(); 4. mysql–er();   21. What is function used for close mysql database connection?   1. mysql=close() 2. **mysql\_close()** 3. mysql->die() 4. mysqli->end()   22. Output of **mysql\_query is**   1. **Boolean** 2. Array 3. String 4. Object   23. Offset value starts from   1. **0** 2. 1 3. 2 4. 3   24. if limit clause has only 1 parameter its considered as   1. offset 2. **limit** 3. same value is considered for both offset and limit 4. error   25. Which of the following takes both numeric and string value as its index  a) mysql\_fetch\_array() |

|  |
| --- |
| 1. **mysql\_fetch\_assoc()** 2. mysql\_fetch\_row() 3. mysql\_fetch()   **Long Answer Questions**   1. What are the different areas where PHP and MySQL are being used together? 2. Define MySQL? What are the different advantages of MySQL 3. Explain different ways to connect to database with appropriate syntaxes 4. What are the different functions related to database used in PHP? 5. Write a php script to create and delete MySQL Database 6. What are the different items used while creating tables, explain with a simple example 7. How to Insert values into database using MySQL 8. Write a note on Select command in php script with where clause 9. With syntax, Explain Update and Delete command 10. Write a note on Limit Clause 11. What are the different methods to fetch the record from database? 12. With appropriate example explain following: 13. Function to execute the query 14. Function to display the error |

|  |
| --- |
| **LAMP Question Bank UNIT I**  1. The programming interface to the kernel is included in which subsystem of operating system.   1. User Applications 2. **O/S Services** 3. Linux Kernel 4. Hardware Controllers   2. Memory hardware is an example of which subsystem of operating system   1. User Applications 2. O/S Services 3. Linux Kernel 4. **Hardware Controllers**   3. Give the full form of VFS   1. [**Virtual File System**](http://oss.org.cn/ossdocs/linux/kernel/a1/index.html#Toc_3_3) 2. Visual File System 3. Virus File System 4. Valid File System   4. Full form of IPC   1. [**Inter-Process Communication**](http://oss.org.cn/ossdocs/linux/kernel/a1/index.html#Toc_3_5) 2. Intra -Process Communication 3. [Inter-Process Command](http://oss.org.cn/ossdocs/linux/kernel/a1/index.html#Toc_3_5) 4. Intra- Process Command   5. Which of the following is responsible for controlling process access to the CPU   1. [Memory Manager](http://oss.org.cn/ossdocs/linux/kernel/a1/index.html#Toc_3_2) 2. [Inter-Process Communication](http://oss.org.cn/ossdocs/linux/kernel/a1/index.html#Toc_3_5) 3. [Virtual File System](http://oss.org.cn/ossdocs/linux/kernel/a1/index.html#Toc_3_3) 4. [**Process Scheduler**](http://oss.org.cn/ossdocs/linux/kernel/a1/index.html#Toc_3_1)   6. Give the full form of PID   1. **P**[rocess identification number](http://www.linfo.org/pid.html) 2. Page identification number 3. [Process identical number](http://www.linfo.org/pid.html) 4. Program identical name   7. Special files are in which folder   1. **/dev** 2. /bin 3. /lib 4. /tmp 5. Which of the following value represent ―No restrictions on permissions‖ on **files**   ***a) 777***  *b) 755*  *c) 700*  *d) 666*   1. Which of the following value represent ―The file's owner may read, write, and execute the   **file**‖.  *a) 777*  *b) 755*  ***c) 700***  *d) 666*   1. Which of the following value represent ―The directory owner has full access. Nobody else |

|  |
| --- |
| has any rights.‖  *a) 777*  *b) 755*  ***c) 700***  *d) 666*  11. Which of the following command is used to ―Prints directory content‖.   1. Cat 2. chmod 3. **ls** 4. mkdir   12. Which of the following command is used Removes directories.   1. Cat 2. chmod 3. **rmdir** 4. rm   13. Which of the following has extra and third party software.   1. **/opt** 2. /root 3. /sbin 4. /tmp   14. Which directory is used for miscellaneous purposes.   1. **/misc** 2. /root 3. /sbin 4. /tmp 5. How is rwx represented in binary form   a) 111 111 111  b) 110 110 110  **c) 111 000 000**  d) 000 000 000   1. The main configuration file for configuring Apache is The main configuration file for configuring Apache is 2. **httpd.conf** 3. srm.conf 4. access.conf 5. apache.conf   17. . If a directive must continue onto the next line which of the following must be used as the last character on the previous line   1. **back-slash '\'** 2. colon (:) 3. semicolon (;) 4. underscore (\_)   18. Any line beginning with a character is ignored   1. **hash (#)** 2. colon (:) 3. semicolon (;) 4. underscore (\_)   19. Maximum number of requests to allow during a persistent connection is indicated via   1. **MaxKeepAliveRequests** 2. KeepAliveTimeout 3. MinSpareServers 4. MaxSpareServers |

|  |
| --- |
| 20. Full form of CGI   1. Correct Gateway Interface 2. **Common Gateway Interface** 3. Common Gateway Interconnect 4. Correct Gateway Interconnect   21. IP based Virtual host is also called as   1. Name based Virtual host 2. **Address-based Virtual host** 3. File based virtual host 4. Time based virtual host 5. If your server has 10 IP addresses, how many IP based virtual hosts can be created?   **a) 10**  b) 20  c) 30  d) 40   1. Which of the following indicates whether or not to allow persistent connections 2. **KeepAlive** 3. MinSpareServers 4. MaxSpareServers 5. MaxKeepAliveRequests   24. In which of the following file the server should record its process identification number   1. LockFile 2. **PidFile** 3. ScoreBoard File 4. StartServers   25. Which of the following file is used to store internal server process information   1. LockFile 2. PidFile 3. **ScoreBoard File** 4. StartServers   **Long Answer Questions**   1. Explain the architecture of linux operating system 2. Briefly explain different subsystem of linux kernel 3. Describe the 2 distinct region of system memory 4. What are different file system permission 5. What is the different setting for directory permissions? 6. List and explain 8 common filesystem commands 7. Write a note on Manipulating the Apache2 HTTPD service 8. What is virtual host, Differentiate IP based Virtual Hosts and Name based Virtual Hosts 9. Write a note on PHP and the web server architecture model 10. Describe the different authentication and log files 11. Explain how the Web server's user and group values are denoted. 12. Write a note on CGI Model |

|  |
| --- |
| **UNIT II**  1. Which one of the following databases has PHP supported almost since the beginning?   1. Oracle Database 2. SQL 3. SQL+ 4. **MySQL**   2. Which one of the following statements is used to create a table?   1. **CREATE TABLE table\_name (column\_name column\_type);** 2. CREATE table\_name (column\_type column\_name); 3. CREATE table\_name (column\_name column\_type); 4. CREATE TABLE table\_name (column\_type column\_name);   **3. PHP is an example of scripting language.**   1. **Server-side** 2. Client-side 3. Browser-side 4. In-side   **4. Which of the following is not true?**   1. PHP can be used to develop web applications. 2. PHP makes a website dynamic 3. PHP applications can not be compile 4. **PHP can not be embedded into html.**   5. Which of the following is not used to begin php code   1. <?php 2. <? 3. **<php** 4. ― <script language=‖php‖> 6. PHP is a -------------------- 5. **Loosely typed language** 6. Tightly typed language 7. Server typed language 8. Client typed language   7. Full form of PHP   1. **PHP Hypertext Pre-processor** 2. Hypertext Pre-processor 3. Plain Hypertext Pre-processor 4. Parsed Hypertext Pre-Processor   8. Php code ends with   1. **;** 2. : 3. .   d) ,  9. Which of the following statement is not true about echo?   1. Not written within parenthesis 2. Can output more than 1 string 3. **Slower than print** 4. Echo($arg1[,$arg2……])   10. PHP variables must begin with a sign   1. **$** 2. @ |

|  |  |
| --- | --- |
| 1. & 2. #   11. Constants are defined using which of the following function   1. Include 2. Require 3. **Define** 4. Main   12. Which of the following command gives information about the fields in a table?   1. **Describe** 2. Create 3. Use 4. Select   13. **Which of the following statements prints in PHP?**   1. Out 2. Write 3. **Echo** 4. Display   14. Which of the following is used to add multi line comment in PHP a) {/ \}  b) //  **c) /\* \*/**  d) {{ }}  15. What will be the output of the following php code? |  |
| < ?php  $num = "1";  $num1 = "2";  print $num+$num1 ;  ?> | |
| **a) 3**  b) 1+2  c) Error  d) 12  16. What will be the output of the following PHP code? |  |
| **<?php**  $num = 1;  $num1 = 2;  print $num . "+". $num1;  **?>** |  |
| a)3 **b)1+2** c)1.+.2  d) Error  17. **Which of the following is not the scope of Variable in PHP?**   1. Local 2. Global 3. Static 4. **Extern**   18. What is the range of unsigned TINYBIT Integer |  |

|  |
| --- |
| **a) 0 to 255**  b) -128 to 127  c) 10 to 265  d) 8 to 263  19. Which among the following datatypes is not case sensitive   1. VARCHAR 2. TINYBLOB 3. TINYTEXT 4. **TINYBIT**   20. Which of the following command is used to show the database created inside MYSQL.   1. **SHOW** 2. USE 3. CREATE 4. SELECT   21. Which of the following command is used to connect to MYSQL   1. SHOW 2. **USE** 3. CREATE 4. SELECT   22. Each Table within the database is been defined and created by which of the following command   1. SHOW 2. USE 3. **CREATE** 4. SELECT   23. Which of the following command is used to print all the records that match the query?   1. SHOW 2. USE 3. CREATE 4. **SELECT**   24. Which of the following is scalar data?   1. Boolean 2. Integer 3. Float 4. **Array**   25. Which of the following is not true about constant?   1. There is no need to write a dollar sign ($) before a constant 2. **Constants can be defined by simple assignment** 3. Once the Constants have been set, may not be redefined or undefined. 4. Constants may be defined and accessed anywhere without regard to variable scoping rules.   **Long Question Answers**   1. What is the use of CREATE TABLE and SHOW TABLE Command, Give the syntax? 2. Give the syntax of the following    1. Insert    2. Select    3. Update    4. Delete 3. What are steps involved to create table using MySQL 4. Explain how the text string is further defined? 5. Write a note on date and time datatype in MySQL 6. What is the use of autoincrement in MySQL 7. Explain the different features of PHP |

|  |
| --- |
| 1. With a simple code explain the basic syntax of PHP 2. Explain php echo statement with example 3. With syntax explain print statement 4. Explain 2 types of comments available in php 5. Write a note on scope of variable in php 6. Write a note on constants, how is it different from variable   **UNIT III**   1. **Which of the following is not a valid example of variable declaration?**    1. $name    2. $address2    3. $colour\_30.    4. **$2name** 2. **Give the output for the following code**   **$first\_name = 'Chhaya';**  **$greeting1 = "Hello, my first name is $first\_name. "; echo $greeting1;**   * 1. **Hello, my first name is Chhaya.**   2. Hello, my last name is $Iast\_name.   3. $greeting1   4. Hello, my first name is  1. To concatenate 2 strings which of the following character is used    1. **.**   b. ,   1. $ 2. \*   4. are named and indexed collections of other values.   1. Strings 2. **Arrays** 3. Objects 4. Resources 5. hold references to resources external to PHP    1. Strings    2. Arrays    3. Objects    4. **Resources** 6. Which of the following is not a ―special datatype‖?    1. **Boolean**    2. Null    3. Array    4. Object 7. Which of the following is used to print the next character as a dollar not as a part of variable?    1. \d    2. $   c. \\$  **d. \$**   1. Which of the following represents conditional operator    1. **?:**    2. ?? |

|  |
| --- |
| 1. ?; 2. ?. 3. Which keyword is used to halt the current iteration of a loop but it does not terminate the loop    1. **Continue**    2. Break    3. Halt    4. Stop 4. Which Letter is used to represent ―AM‖ in Date    1. **A**    2. a    3. D    4. d 5. Which of the following is used to Set the internal pointer of an array to its last element?    1. count    2. current    3. next    4. **end** 6. Which of the following determines if a variable is declared and is different than **NULL**    1. **Isset**    2. Empty    3. Current    4. Set 7. An array with string index is called as    1. Numeric array    2. **Associative array**    3. Multidimensional array    4. Indexed Array   14. PHP‘s numerically indexed array begin with position   1. 1 2. 2 3. **0** 4. -1 5. Which of the following function is used to get the value of the previous element in an array?    1. last()    2. before()    3. **prev()**    4. previous() 6. Multidimensional arrays are simple arrays that have    1. One dimensional    2. **Many arrays stored in them**    3. No indexes    4. 1 element 7. For finding nonempty elements in array we use    1. is\_array ( ) function    2. sizeof ( ) function    3. array\_count ( ) function    4. **count ( ) function** 8. When we simply want iteration through looping an array values we can use    1. current ( ) |

|  |
| --- |
| 1. **foreach ( )** 2. next ( ) 3. prev() 4. Count ( ) function is identical to    1. is\_array ( ) function    2. in\_array ( ) function    3. **sizeof ( )**    4. isset ( ) function 5. How does the identity operator === compare two values?    1. **It converts them to a common compatible data type and then compares the resulting values**    2. It returns True only if they are both of the same type and value    3. If the two values are strings, it performs a lexical comparison    4. It bases its comparison on the C strcmp function exclusively 6. Which of the following is not a global variable?    1. $\_POST    2. $\_GLOBALS    3. $\_REQUEST    4. **$NUM** 7. loop will execute at least once.    1. While    2. **Do while**    3. For    4. Foreach   23. Which among the following has highest precedence over others?  a. ==  **b. ++**  c. &&  d. +   1. Which of the following is unary operator?    1. **--**    2. -    3. \*    4. && 2. are used in PHP to performs what is known as variable expansion    1. **Double quotes**    2. Single quotes    3. Comma    4. Semicolon |

|  |
| --- |
| **Long Answer Questions**   1. What are the different datatypes available in PHP 2. What is Escaping Characters? Write brief notes on Escaping Characters. 3. Explain the following with example    1. Assignment operator    2. Logical operator 4. With appropriate example differentiate post increment and pre increment. 5. With syntax and example explain, else if and switch statements in PHP 6. Why do we need break and continue statements, explain with example? 7. Explain date () function with its syntax. List and explain any eight format which can be used with it 8. What is an array give its syntax? Explain Reset () and Sizeof () function with example 9. Define numeric and associative array with example 10. With example, explain multi-dimensional array 11. With syntax and example explain the following:     1. String length     2. Count the Number of Words in a String     3. Reverse a String     4. Search for a Specific Text Within a String 12. Explain the following with example, and give respective output.     1. substr\_count()     2. **strcasecmp(**   **UNIT IV**   * + 1. Which predefined variables are used to retrieve information from forms?  1. $GET 2. **$\_GET** 3. $\_SET 4. GET   2. When you use the $\_GET variable to collect data, the data is visible to   1. none 2. only you 3. **everyone** 4. selected few   3. When you use the $\_POST variable to collect data, the data is visible to   1. none 2. **only you** 3. everyone 4. selected few   4. Which variable is used to collect form data sent with both the GET and POST methods?   1. $BOTH 2. $\_BOTH 3. $REQUEST 4. **$\_REQUEST**   5. Which one of the following should not be used while sending passwords or other sensitive information?   1. **GET** 2. POST 3. REQUEST 4. NEXT   6. The is a superglobal variable that stores information passed into the  script through HTTP cookies. |

|  |
| --- |
| 1. $COOKIE 2. **$\_COOKIE** 3. $SET\_COOKIE 4. $ADD\_COOKIE   7. What is the default type of ‗type‘ attribute of <input> element?   1. **Text** 2. Password 3. Numerals 4. Special Characters 5. **In PHP, cookies are set by using the**    1. **setcookie ( ) function**    2. set ( ) function    3. cookie\_set ( ) function    4. cookie() 6. variable is used to set session    1. **$\_SESSION**    2. $\_SETSESSION    3. SETSESSION    4. SESSION 7. How do we remove all session variables    1. **session\_unset();**    2. session\_destroy();    3. session\_delete();    4. session\_stop(); 8. Which symbol is used for **Pass by Reference**    1. **&**    2. |    3. %    4. $   12. How do you get information from a form that is submitted using the "get" method? |
| **a. $\_GET[];** |
| b. Request.Form; |
| c. Request.QueryString; |
| d. $\_POST[];   1. Which of the following is the right one to define the function    1. function {function body}    2. datatype functionname(parameters){function body}    3. function {function body}    4. **function functionname (parameters){function body}** 2. Maximum length of get method   **a. 2048**  b. 2000  c. 2100  d. 3000   1. In PHP default behavior for user defined functions is    1. **Call-by-value**    2. Call-by-reference    3. Call-by-type    4. Call-by-loc   17. Which of the following methods would you use for sending an email? |

|  |
| --- |
| 1. **mail($to,$subject,$body)** 2. sendmail($to,$subject,$body) 3. mail(to,subject,body) 4. sendmail(to,subject,body)   18. In your PHP application you need to open a file. You want the application to issue a warning and continue execution, in case the file is not found. The ideal function to be used is:   1. **include()** 2. require() 3. nowarn() 4. getFile(false)   19. Sessions doesn‘t allow you to   1. store persistent user preference on a site 2. save user authentication information from page to page 3. create multipage forms   **d) Doesn’t save user authentication information from page to page**   1. will produce a fatal error    1. **Require**    2. Include    3. Get    4. Post 2. Which of the following is a mandatory parameter of setcookie    1. expiry\_time    2. cookie\_path    3. **cookie\_name**    4. domain 3. can be used to define the cookie access hierarchy    1. expiry\_time    2. cookie\_path    3. cookie\_name    4. **domain** 4. A definition of function starts with .    1. **Function**    2. Datatype    3. Function name    4. void   24. things are stored in the $\_FILES array when a file is uploaded?   1. **5** 2. 8 3. 7 4. 9 5. $\_FILES is an array    1. **Associative**    2. Numeric    3. Indexed    4. Boolean |

|  |
| --- |
| **Long Answer Questions**   1. What are the elements required to build PHP form explain with example? 2. Write a simple php form with at least 4 input fields to display values obtained from input fields 3. Explain the use of any 2 form elements in php with appropriate example. 4. Why do we use the get method, explain the variable used to fetch the values passed by get method, give the drawback of the same? 5. Write a note on POST method and $\_POST variable. 6. Why do we need include and require statements? Differentiate include and require statements in PHP. With php script. 7. Explain function declaration in PHP. Give example 8. Write a note on scope of variable 9. What is session? How the session is created explain with example. 10. Give the syntax of cookie, explain the parameters in it. 11. Explain mail() function with its syntax. Give example. 12. How to create a upload script in PHP, explain the different variables in it.   **UNIT V**  **1.** Which Statement Is Used To Insert A New Data In A Database?   1. INSERT NEW 2. Add New 3. **INSERT INTO** 4. UPDATE   **2.** Which Of These Commands Will Delete A Table Called XXX If You Have Appropriate Authority:   1. **DROP TABLE XXX** 2. DELETE XXX WHERE Confirm = "YES" 3. DROP XXX 4. DROP XXX WHERE Confirm = "YES"   **3.** MySQL Runs On Which Operating Systems?   1. Unix And Linux Only 2. **Unix, Linux, Windows And Others** 3. Linux And Mac OS-X Only 4. Any Operating System At All   **4.** Which Of The Following Is Available In MySQL:   1. CREATE VIEW 2. CREATE SCHEMA 3. CREATE TRIGGER 4. **CREATE DATABASE**   **5.** Which Of These Is Not A Valid Name For A Column   1. To 2. **From** 3. Far 4. Near   6. Which of the following is NOT available in MySQL:   1. **FETCH** 2. LIKE 3. JOIN 4. SELECT   7. To use MySQL on your computer, you'll need?  a) FTP and Telnet |

## Some sort of client program to access the databases

1. A Browser
2. Perl, PHP or Java
3. MySQL is

|  |
| --- |
| a) Protocol |
| b) A Programming language |
| c) A technique for writing reliable programs |
| **d) A Relational Database Management System** |

1. Which of the following is used to delete an entire MYSQL database?
2. mysql\_drop\_database
3. mysql\_drop\_entiredb

## mysql\_drop\_db

1. mysql\_drop\_dbase
2. Which of the following commands should be used to create a database named

―student‖?

1. CREATE student

## CREATE DATABASE student

1. DATABASE /student
2. DATABSE student
3. A SELECT command without a WHERE clause returns?
4. All the records from a table that match the previous WHERE clause

## All the records from a table, or information about all the records

1. SELECT is invalid without a WHERE clause
2. Nothing
3. What does SQL stand for?
4. Strong question Language

## Structured Query language

1. Structured question language
2. Strong query language
3. With SQL, how do you select a column named "FirstName" from a table named "Persons"?
4. Select Persons.Firstname
5. Extract firstname from persons

## Select firstname from persons

1. Select firstname
2. With SQL, how do you select all the columns from a table named "Persons"?
3. Select persons
4. select [all] from persons

## Select \* from persons

1. Select \*.persons
2. The result of select command is
3. File
4. Report

## Table

1. Form
2. Full form of **PDO**

## a) PHP Data Objects

|  |
| --- |
| 1. Permanent Data Object 2. Public Data Object 3. Private Data Object   17. Output on executing mysql\_fetch\_row()   1. **Array** 2. String 3. Number 4. Object   18. Using SELECT will return all the fields.   1. **\*** 2. .   c) +  d) &  19. Which of the following is not true about MYSQL   1. Fast 2. Easy 3. **Not Secure** 4. Scalable   20. Which function returns an error massage?   1. **mysql\_error();** 2. mysql\_errors(); 3. mysql–error(); 4. mysql–er();   21. What is function used for close mysql database connection?   1. mysql=close() 2. **mysql\_close()** 3. mysql->die() 4. mysqli->end()   22. Output of **mysql\_query is**   1. **Boolean** 2. Array 3. String 4. Object   23. Offset value starts from   1. **0** 2. 1 3. 2 4. 3   24. if limit clause has only 1 parameter its considered as   1. offset 2. **limit** 3. same value is considered for both offset and limit 4. error   25. Which of the following takes both numeric and string value as its index   1. mysql\_fetch\_array() 2. **mysql\_fetch\_assoc()** 3. mysql\_fetch\_row() 4. mysql\_fetch() |

|  |
| --- |
| **Long Answer Questions**   1. What are the different areas where PHP and MySQL are being used together? 2. Define MySQL? What are the different advantages of MySQL 3. Explain different ways to connect to database with appropriate syntaxes 4. What are the different functions related to database used in PHP? 5. Write a php script to create and delete MySQL Database 6. What are the different items used while creating tables, explain with a simple example 7. How to Insert values into database using MySQL 8. Write a note on Select command in php script with where clause 9. With syntax, Explain Update and Delete command 10. Write a note on Limit Clause 11. What are the different methods to fetch the record from database? 12. With appropriate example explain following: 13. Function to execute the query 14. Function to display the error |

|  |  |
| --- | --- |
|  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | | | |
| **Reg.No** |  |  |  |  |  |  |  |  |  |  |  |  |
| SRINIVAS UNIVERSITY  V SEMESTER BCA DEGREE EXAMINATION NOVEMBER-2019  SUBJECT NAME: LAMP Technology  PAPER CODE: 17BCAAM51 / 17BCACC51/ 17BCASD51  Time: 2 Hours Max. Marks: 50  ***Instructions: Answer any 10 Questions from PART-A and One Full Questions from Each UNIT in PART-B***  **PART – A (10x1 = 10)**  1.  a) The programming interface to the kernel is included in which subsystem of operating system.   1. User Applications 2. O/S Services 3. Linux Kernel 4. Hardware Controllers   b) Memory hardware is an example of which subsystem of operating system   1. User Applications 2. O/S Services 3. Linux Kernel 4. Hardware Controllers   c) Give the fullform of VFS   1. [Virtual File System](http://oss.org.cn/ossdocs/linux/kernel/a1/index.html#Toc_3_3) 2. Visual File System 3. Virus File System 4. Valid File System   d) Which one of the following databases has PHP supported almost since the beginning?   1. Oracle Database 2. SQL 3. SQL+ 4. MySQL   e) Which one of the following statements is used to create a table?   1. CREATE TABLE table\_name (column\_name column\_type); 2. CREATE table\_name (column\_type column\_name); 3. CREATE table\_name (column\_name column\_type); 4. CREATE TABLE table\_name (column\_type column\_name);   f) PHP is an example of scripting language.   1. Server-side 2. Client-side 3. Browser-side 4. In-side   g) To concatenate 2 strings which of the following character is used  A. .  B. ,   1. $ 2. \*   h) are named and indexed collections of other values.  A. Strings | | | | | | | | | | | | |

|  |
| --- |
| 1. Arrays 2. Objects 3. Resources   i) What is function used for close mysql database connection?   1. mysql=close() 2. mysql\_close() 3. mysql->end() 4. mysql->die()   j) [Which Statement Is Used To Insert A New Data In A Database?](https://www.onlineinterviewquestions.com/mysql-mcq/#collapseUnfiled33)   1. INSERT NEW 2. Add New 3. INSERT INTO 4. UPDATE   k) In PHP, cookies are set by using the   1. setcookie ( ) function 2. set ( ) function 3. cookie\_set ( ) function 4. cookie()   l) How do we remove all session variables   1. session\_unset(); 2. session\_destroy(); 3. session\_delete(); 4. session\_stop(); |

|  |
| --- |
| **UNIT I**  2. ( 4 + 4 )   1. Explain the architecture of linux operating system 2. What is virtual host, Differentiate IP based Virtual Hosts and Name based Virtual Hosts   **OR**  3. ( 4 + 4 )   1. Briefly explain different subsystem of linux kernel 2. Write a note on Manipulating the Apache2 HTTPD service   **UNIT II**  4. ( 4 + 4 )   1. Write a note on date and time datatype in MySQL 2. With a simple code explain the basic syntax of PHP   **OR**  5. ( 4 + 4 )   1. What is the use of autoincrement in MySQL 2. Explain the different features of PHP   **UNIT III**  6. ( 4 + 4 )   1. With syntax and example explain, else if and switch statements in PHP 2. What is an array give its syntax? Explain Reset () and Sizeof () function with example   **OR**  7. ( 4 + 4 )   1. Why do we need break and continue statements, explain with example? 2. Explain date () function with its syntax. List and explain any eight format which can be used with it   **UNIT IV**  8. ( 4 + 4 )   1. Write a note on POST method and $\_POST variable. 2. Why do we need include and require statements? Differentiate include and require statements in PHP. With php script.   **OR**  **9.** ( 4 + 4 )   1. What are the elements required to build PHP form explain with example? 2. Explain mail() function with its syntax. Give example.   **UNIT V**  10. ( 4 + 4 )   1. Write a php script to create and delete MySQL Database 2. Write a note on Select command in php script with where clause   **OR**  **11.** ( 4 + 4 )   1. How to Insert values into database using MySQL 2. What are the different items used while creating tables, explain with an example |