KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)

(Established Under Section 3 of UGC Act, 1956)

Coimbatore - 641 021, India

FACULTY OF ARTS, SCIENCE AND HUMANITIES (FASH)

Department of CS,CA & IT

STAFF NAME : A.JEEVARATHINAM & Dr.S.MANJU PRIYA

SUBJECT NAME : PHP PROGRAMMING **SUB.CODE : 16CSU603A**

SEMESTER : VI **CLASS : III B.Sc (CS)**

UNIT I

S.NO	Lecture Duration (Hours)	Topics To Be Covered	Support Materials/ Pg.No
		Introduction	
1	1	PHP introduction, inventions and versions,	W1
		important tools and software requirements	
		PHP with other technologies, scope of PHP	
		Variables and Data Types	
2	1	Basic Syntax, PHP variables and constants	T3 : 4 - 55
3		Types of data in PHP , scopes of a variable	
		Expressions	T3 : 65 - 67
		PHP Operators	
4	1	Arithmetic, Assignment, Relational,	T4 : 28 - 37
		Logical operators, Bitwise , ternary	
5	1	MOD operator, Operator Precedence and associatively	
6	1	Recapitulation and Discussions of Important Questions	
		Total No of Hours Planned for Unit I	**6**

UNIT II

S.NO	Lecture Duration (Hours)	Topics To Be Covered	Support Materials/ Pg.No
		Handling HTML form with PHP:	
1	1	Capturing Form Data	T3 : 255 -267
		GET and POST form methods	
2	1	Dealing with multi value fields	
		Redirecting a form after submission	
		PHP conditional Events and Loops:	
3	1	PHP IF Else conditional statements (Nested IF and Else)	T3 : 75 - 81
4	1	Switch case, while	T3 : 82 - 89
5	1	For and Do While Loop	
6	1	Goto , Break ,Continue and exit	
7	1	Recapitulation and Discussion of Important Questions	
		Total No of Hours Planned for Unit II	**7**

UNIT III

S.NO	Lecture Duration (Hours)	Topics To Be Covered	Support Materials/ Pg.No
		PHP Functions:	
1	1	Function, Need of Function	T3 : 94 - 100
		declaration and calling of a function	
2	1	PHP Function with arguments,	
		Default Arguments in Function	
3	1	Function argument with call by value, call by reference	
4	1	Scope of Function Global and Local	
5	1	Recapitulation and Discussion of Important Questions	
		Total No of Hours Planned for Unit III	**5**

UNIT IV

S.NO	Lecture Duration (Hours)	Topics To Be Covered	Support Materials/ Pg.No
		String Manipulation and Regular Expression: (3L)	
		String Manipulation	
1	1	Creating and accessing String	T4 : 95 - 106
		Searching & Replacing String	
2	1	Formatting, joining and splitting String	
3	1	String Related Library functions	
		Regular Expression	
4	1	Use and advantage of regular expression over inbuilt function	T4 : 109 - 116
		Use of preg_match(), preg_replace()	
5	1	preg_split() functions in regular expression	
6	1	Recapitulation and Discussion of Important Questions	
		Total No of Hours Planned for Unit IV	**6**

UNIT V

S.NO	Lecture Duration (Hours)	Topics To Be Covered	Support Materials/ Pg.No
		Array:	
1	1	Anatomy of an Array	T3 : 119 - 131
		Creating index based and Associative array	
		Accessing array	
2	1	Looping with Index based array,	
		with associative array using each() and foreach()	
3	1	Some useful Library function	
4	1	Recapitulation and Discussion of Important Questions	
		Discussion of previous ESE question papers	
5	1	Discussion of previous ESE question papers	
6	1	Discussion of previous ESE question papers	
		Total No of Hours Planned for Unit V	**6**

Total Hours	**30**

S.NO	TEXT BOOKS
T1	Steven Holzner. (2007). PHP: The Complete Reference. New Delhi: McGraw Hill Education (India).
T2	Timothy Boronczyk., & Martin, E. Psinas. (2008). PHP and MYSQL (Create-Modify-Reuse). New Delhi: Wiley India Private Limited.
T3	Robin Nixon. (2014). Learning PHP, MySQL, JavaScript, CSS & HTML5 (3rd ed.). O'reilly.
T4	Luke Welling.,& Laura Thompson.(2008). PHP and MySQL Web Development (4th ed.). Addition Paperback, Addison-Wesley Professsional.

S.NO	WEB SITES
W1	https://www.JavaTPoint.com
W2	www.studytonight.com
W3	https://www.w3schools.com

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)

(Established Under Section 3 of UGC Act, 1956)

Coimbatore - 641 021, India

FACULTY OF ARTS, SCIENCE AND HUMANITIES (FASH)

Department of CS,CA & IT

III B.Sc CS **VI SEMESTER** **BATCH : 2016 - 2019**

16CSU603A **PHP PROGRAMMING** **3H – 3C**

Instruction Hours / week: L: 3 T: 0 P: 0 **Marks:** Int : **40** Ext : **60** Total: **100**

SCOPE

This course design focuses on the structure of the website including the information architecture, the layout or the pages and the conceptual design with branding. PHP helps the students for developing dynamic web pages.

COURSE OBJECTIVES

- To work with open source applications that deal with database and website development.
- Establish a working environment for PHP web page development
- Very familiar with GUI, coded, modified controls
- To understand the general concepts of PHP scripting language for the development of Internet websites.

COURSE OUTCOME

Upon successful completion of this course, a student should be able to:

- Understand the use of PHP with HTML.
- Understand the ability to post and publish a PHP website.
- Develop Database connectivity using MySQL.
- Develop Web Applications
- Create PHP programs that use various PHP library functions, and that manipulate files and directories.

UNIT-I

Introduction to PHP:

- PHP introduction, inventions and versions, important tools and software requirements (like Web Server, Database, Editors etc.)
- PHP with other technologies, scope of PHP
- Basic Syntax, PHP variables and constants
- Types of data in PHP , Expressions, scopes of a variable (local, global)
- PHP Operators: Arithmetic, Assignment, Relational, Logical operators, Bitwise , ternary and MOD operator.
- PHP operator Precedence and associatively

UNIT-II

Handling HTML form with PHP:

- Capturing Form Data
- GET and POST form methods
- Dealing with multi value fields
- Redirecting a form after submission

PHP conditional events and Loops:

- PHP IF Else conditional statements (Nested IF and Else)
- Switch case, while ,For and Do While Loop
- Goto , Break ,Continue and exit

UNIT-III

PHP Functions:

- Function, Need of Function , declaration and calling of a function
- PHP Function with arguments, Default Arguments in Function
- Function argument with call by value, call by reference
- Scope of Function Global and Local

UNIT-IV

String Manipulation and Regular Expression: (3L)

- Creating and accessing String , Searching & Replacing String
- Formatting, joining and splitting String , String Related Library functions
- Use and advantage of regular expression over inbuilt function
- Use of preg_match(), preg_replace(), preg_split() functions in regular expression

UNIT-V

Array:

- Anatomy of an Array ,Creating index based and Associative array ,ACSUessing array
- Looping with Index based array, with associative array using each() and foreach()
- Some useful Library function

SUGGESTED READINGS

1. Steven Holzner. (2007). PHP: The Complete Reference. New Delhi: McGraw Hill Education (India).

2. Timothy Boronczyk., & Martin, E. Psinas. (2008). PHP and MYSQL (Create-Modify-Reuse). New Delhi: Wiley India Private Limited.

3. Robin Nixon. (2014). Learning PHP, MySQL, JavaScript, CSS & HTML5 (3rd ed.). O'reilly.

4. Luke Welling.,& Laura Thompson.(2008). PHP and MySQL Web Development (4th ed.). Addition Paperback, Addison-Wesley Professsional.

5. David Sklar., & Adam Trachtenberg. PHP Cookbook: Solutions & Examples for PHP.

WEB SITES:

1. www. JavaTpoint.com
2. www.PHPTpoint.com
3. www.w3school.com
4. www.guru99.com
5. www.geeksforgeeks.com

CLASS : III B.Sc CS **BATCH : 2016 - 2019**

COURSE NAME : PHP PROGRAMMING **COURSE CODE : 16CSU603A**

UNIT I

Introduction to PHP:

- PHP introduction, inventions and versions, important tools and software requirements (like Web Server, Database, Editors etc.)
- PHP with other technologies, scope of PHP
- Basic Syntax, PHP variables and constants
- Types of data in PHP , Expressions, scopes of a variable (local, global)
- PHP Operators: Arithmetic, Assignment, Relational, Logical operators, Bitwis , ternary and MOD operator.
- PHP operator Precedence and associativity

Introduction to PHP:

PHP

PHP is a open source, interpreted and object-oriented scripting language i.e. executed at server side. It is used to develop web applications (an application i.e. executed at server side and generates dynamic page).

What is PHP?

- PHP is a server side scripting language.
- PHP is an interpreted language, i.e. there is no need for compilation.
- PHP is an object-oriented language.
- PHP is an open-source scripting language.
- PHP is simple and easy to learn language.

PHP Features

There are given many features of PHP.

- **Performance**: Script written in PHP executes much faster then 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.

History of PHP

The PHP code is well organized and can be easily embedded into HTML code. It works on all major operating systems like Linux, Windows, Unix and Mac OS, and it supports main web and enterprise servers like Apache, Netscape, Microsoft IIS, etc. Moreover, it is easier to troubleshoot problems in PHP when compared to other languages.

PHP was created by Rasmus Lerdorf in 1994 and was publicly released in June 1995. Back then, it was the abbreviated form of Personal Home Page tools. After two years, in 1997, it entered the public domain as PHP/F1 2.0. A year later, two programmers, Zeev Suraski and Andi Gutmans, rewrote the base of the original version and launched PHP 3.

PHP 4, which came out in 2000, incorporated a scripting engine named Zend Engine that was designed by Suraski and Gutmans. Three more major versions with some sub-versions were launched in the later years with the latest version 7.0 released in 2015.

Two decades after its inception, in the History of PHP has registered a phenomenal growth and is still going strong. Today, it controls over 80% of all the websites on the globe. This includes majors like Facebook, Wikipedia, and WordPress among others.

PHP 3 – Hits the Big Time

PHP 3.0 was the first version that closely resembles PHP as it exists today. Finding PHP/FI 2.0 still inefficient and lacking features they needed to power an eCommerce application they were developing for a university project, Andi Gutmans and Zeev Suraski of Tel Aviv, Israel, began yet another complete rewrite of the underlying parser in 1997. Approaching Rasmus online, they discussed various aspects of the current implementation and their redevelopment of PHP. In an effort to improve the engine and start building upon PHP/FI's existing user base, Andi, Rasmus, and Zeev decided to collaborate in the development of a new, independent programming language. This entirely new language was released under a new name, that removed the implication of limited personal use that the PHP/FI 2.0 name held. It was renamed simply 'PHP', with the meaning becoming a recursive acronym – PHP: Hypertext Preprocessor.

PHP 4 – Optimization, Scalability and More

By the winter of 1998, shortly after PHP 3.0 was officially released, Andi Gutmans and Zeev Suraski had begun working on a rewrite of PHP's core. The design goals were to improve the performance of complex applications and improve the modularity of PHP's code base. Such applications were made possible by PHP 3.0's new features and support for a wide variety of third-party databases and APIs, but PHP 3.0 was not designed to handle such complex applications efficiently.

PHP 5 – Object Orientation, Error Handling, and XML

PHP 5 was released in July 2004 after the long development and several pre-releases. It is mainly driven by its core, the *Zend Engine 2.0* with a new object model and dozens of other new features.

In the History of PHP development team includes dozens of developers, as well as dozens of others working on PHP-related and supporting projects, such as PEAR, PECL, and documentation, and an underlying network infrastructure of well over one-hundred individual web servers on six of the seven continents of the world. Though only an estimate based on statistics from previous years, it is safe to presume PHP is now installed on tens or even perhaps hundreds of millions of domains around the world.

PHP 7 – Makes Powering the web a whole lot better

Introducing PHP 7 – a revolution in the way we deliver applications that power everything from websites and mobile to enterprises and the cloud. This is the most important change for PHP since the release of PHP 5 in 2004, bringing explosive performance improvements, drastically reduced memory consumption, and a host of brand-new language features to make your apps soar.

Thanks to the new Zend Engine 3.0, your apps see up to 2x faster performance and 50% better memory consumption than PHP 5.6, allowing you to serve more concurrent users without adding any hardware. Designed and refactored for today's workloads, PHP 7 is the ultimate choice for web developers today.

PHP 7 is a major release of PHP programming language and is touted to be a revolution in the way web applications can be developed and delivered for mobile to enterprises and the cloud. This release is considered to be the most important change for PHP after the release of PHP 5 in 2004.

New Features

There are dozens of features added to PHP 7, the most significant ones are mentioned below −

- **Improved performance** − Having PHPNG code merged in PHP7, it is twice as fast as PHP 5.
- **Lower Memory Consumption** − Optimized PHP 7 utilizes the lesser resource.
- **Scalar type declarations** − Now parameter and return types can be enforced.
- **Consistent 64-bit support** − Consistent support for 64-bit architecture machines.
- **Improved Exception hierarchy** − Exception hierarchy is improved.
- **Many fatal errors converted to Exceptions** − Range of exceptions is increased covering many fatal errors converted as exceptions.
- **Secure random number generator** − Addition of new secure random number generator API.
- **Deprecated APIs and extensions removed** − Various old and unsupported APIs and extensions are removed from the latest version.
- **The null coalescing operator (??)** − New null coalescing operator added.
- **Return and Scalar Type Declarations** − Support for return type and parameter type added.
- **Anonymous Classes** − Support for anonymous added.
- **Zero cost asserts** − Support for zero cost asserts added.

PHP 7 uses new Zend Engine 3.0 to the History of PHP improve application performance almost twice and 50% better memory consumption than PHP 5.6. It allows serving more concurrent users without requiring any additional hardware. PHP 7 is designed and refactored considering today's workloads.

Advantage of PHP

If you are familiar with other server side language like ASP.NET or JSP you might be wondering what makes PHP so special, or so different from these competing alternatives well, here are some reasons:

1. Performance
2. Portability(Platform Independent)
3. Ease Of Use
4. Open Source

5. Third-Party Application Support
6. Community Support

Performance

Scripts written in PHP executives faster than those written in other scripting language, with numerous independent benchmarks, putting the language ahead of competing alternatives like JSP,ASP.NET and PERL.

The PHP 5.0 engine was completely redesigned with an optimized memory manager to improve performance, and is noticeable faster than previous versions.

In addition, third party accelerators are available to further improve performance and response time.

Portability

PHP is available for UNIX, MICROSOFT WINDOWS, MAC OS, and OS/2. PHP Programs are portable between platforms.

As a result, a PHP application developed on, say, Windows will typically run on UNIX without any significant issues.

This ability to easily undertake cross-platform development is a valuable one, especially when operating in a multi platform corporate environment or when trying to address multiple market segments.

Ease Of Use

"Simplicity is the ultimate sophistication", Said Leonardo da Vinci, and by that measure, PHP is an extremely sophisticated programming language.

Its syntax is clear and consistent, and it comes with exhaustive documentation for the 5000+ functions included with the core distributions.

This significantly reduces the learning curve for both novice and experienced programmers, and it's one of the reasons that PHP is favored as a rapid prototyping tool for Web-based applications.

Open Source

PHP is an open source project – the language is developed by a worldwide team of volunteers who make its source code freely available on the Web, and it may be used without payment of licensing fees or investments in expensive hardware or software .

This reduces software development costs without affecting either flexibility or reliability. The open-source nature of the code further means that any developer, anywhere , can inspect the code tree, spit errors, and suggest possible fixes, this produces a stable, robust product wherein bugs, once discovered, are rapidly resolved – sometimes within a few hours of discovery !.

Third-Party Application Support

One of PHP's Strengths has historically been its support for a wide range of different databases, including MySQL, PostgreSQL, Oracle, and Microsoft SQL Server.

PHP 5.3 Supports more than fifteen different database engines, and it includes a common API for database access.

XML support makes it easy to read and write XML documents though they were native PHP data structures, access XML node collections using Xpath, and transform XML into other formats with XSLT style sheets.

Community Support

One of the nice things about a community-supported language like PHP is the access it offers to the creativity and imagination of hundreds of developers across the world.

Within the PHP community, the fruits of this creativity may be found in PEAR, the PHP Extension and Application Repository and PECL, the PHP Extension Community Library, which contains hundreds of ready-,made widgets and extensions that developers can use to painlessly and new functionality to PHP.

Using these widgets is often a more time-and cost-efficient alternative to rolling your own code.

PHP vs. Java

It isn't correct to compare Java to PHP. Since PHP is a server-side scripting language whereas Java is a general-purpose language. In other words, PHP is only used as a server-side

language where Java is both for server-side and desktop programming language. Moreover, Java is compiled and strongly-typed language. On other hand, PHP is a dynamic typed language. Hence, only for server-side programming, the comparison between Java and PHP makes sense.

Install PHP

To install PHP, we will suggest you to install AMP (Apache, MySQL, PHP) software stack. It is available for all operating systems. There are many AMP options available in the market that are given below:

- **WAMP** for Windows (Windows Apache, MySQL, PHP)
- **LAMP** for Linux (Linux Apache, MySQL, PHP)
- **MAMP** for Mac(Mac Apache, MySQL, PHP)
- **SAMP** for Solaris (Solaris Apache, MySQL, PHP)
- **FAMP** for FreeBSD
- **XAMPP** (Cross, Apache, MySQL, PHP, Perl) for Cross Platform: It includes some other components too such as FileZilla, OpenSSL, Webalizer, Mercury Mail etc.

If you are on Windows and don't want Perl and other features of XAMPP, you should go for WAMP. In a similar way, you may use LAMP for Linux and MAMP for Macintosh.

PHP Syntax

PHP Environment Start – End

syntax is a way to representation of PHP script.

Basically it gives the primary idea to specify the code format.

It also specify the area of a written code.

There are three ways to start PHP environment.

The most commonly and effective PHP syntax:

```
<?php

echo "Welcome to the world of php";

?>
```

In the given Example
It begin with (< ?php) and End with(? >).
echo statement is used to print the given string. Mandatory closing in("ABCD")double quotes.
Terminate the script with semicolon because semicolon is known as terminator.

Short or short-open tags look like this:

```
<?
        echo "welcome to the world of php";
?>
```

In the given above example..
We use Short tags. start and end with () respectively.
Declare the statement in between (), to display the output on browser
short open tags smoothly works on **XAMPP server** but Face problem on **wamp server**

if you are using WAMP Server First you have to Set the **short_open_tag** setting in your **php.ini** file to on

HTML script tags:

```
<script language="PHP">
      echo "welcome to the world of php";
</script>
```

PHP is the server side Scripting language. So HTML script is also used to start PHP environment like other scripting language.

PHP Example

It is very easy to create a simple PHP example. To do so, create a file and write HTML tags + PHP code and save this file with .php extension.

All PHP code goes between php tag. A syntax of PHP tag is given below:

```
<?php
//your code here
?>
```

Let's see a simple PHP example where we are writing some text using PHP echo command.

File: first.php

```
<!DOCTYPE>
<html>
<body>
<?php
echo "<h2>Hello First PHP</h2>";
?>
</body>
</html>
```

Output:

Hello First PHP

PHP Echo

PHP echo is a language construct not a function, so you don't need to use parenthesis with it. But if you want to use more than one parameters, it is required to use parenthesis.

The syntax of PHP echo is given below:

```
void echo ( string $arg1 [, string $... ] )
```

PHP echo statement can be used to print string, multi line strings, escaping characters, variable, array etc.

PHP echo: printing string

File: echo1.php

```
<?php
echo "Hello by PHP echo";
?>
```

Output:

Hello by PHP echo

PHP echo: printing multi line string

File: echo2.php

```
<?php
echo "Hello by PHP echo
this is multi line
text printed by
PHP echo statement
";
?>
```

Output:

```
Hello by PHP echo this is multi line text printed by PHP echo statement
```

PHP echo: printing escaping characters

File: echo3.php

```
<?php
echo "Hello escape \"sequence\" characters";
?>
```

Output:

```
Hello escape "sequence" characters
```

PHP echo: printing variable value

File: echo4.php

```
<?php
$msg="Hello JavaTpoint PHP";
echo "Message is: $msg";
?>
```

Output:

```
Message is: Hello JavaTpoint PHP
```

PHP Print

Like PHP echo, PHP print is a language construct, so you don't need to use parenthesis with the argument list. Unlike echo, it always returns 1.

The syntax of PHP print is given below:

```
int print(string $arg)
```

PHP print statement can be used to print string, multi line strings, escaping characters, variable, array etc.

PHP print: printing string

File: print1.php

```
<?php
print "Hello by PHP print ";
print ("Hello by PHP print()");
?>
```

Output:

```
Hello by PHP print Hello by PHP print()
```

PHP print: printing multi line string

File: print2.php

```
<?php
print "Hello by PHP print
this is multi line
text printed by
PHP print statement
";
?>
```

Output:

```
Hello by PHP print this is multi line text printed by PHP print statement
```

PHP print: printing escaping characters

File: print3.php

```
<?php
print "Hello escape \"sequence\" characters by PHP print";
?>
```

Output:

```
Hello escape "sequence" characters by PHP print
```

PHP print: printing variable value

File: print4.php

```
<?php
$msg="Hello print() in PHP";
print "Message is: $msg";
?>
```

Output:

```
Message is: Hello print() in PHP
```

PHP Comments

A comment is non-executable lines. comment is used to write description for your own understanding.Browser doesn't read the comments.

There are two types of comments used in php

1. Single line comments :

Single line comment used for short explanations.

Declaration of Single line comment are two types

Either Begin with(#) Or backslash(//)

```
<?php
# This is the single line comment
# This is the next line comment
// This is also a single line comment.
?>
```

In the above Example.

*First and second line comments begin with hash(#).

*The third one is begin with(//).

If we check the output of the given example.

Browser show blank page. Because comments are always non-executable..

another Eg of Single line Comment

```
<?php
        $str= "welcome ";

//$str. =" student";
echo $str;
?>
```

Output

welcome

2. Multi-lines comments :

Multi lines comments used to comment multiple lines.

Here we can give comments in bulk

The bulk comments are enclose within (/*.....*/)

```
<?php
/*
This is a comment with multiline
Developer : sanjeev rai
view : Multiline Comments Demo
*/
?>
```

The all lines which is define in php evironment are Multiline comments.

it is non-executable.Because it enlose with Multiline comments statement.

another eg of Multi-line comments

```
<?php
/*
$str = "welcome ";
$str.= "users ";
*/
echo "Hello user how are you ? ";
?>
```

Output

Hello user how are you ?

PHP Variables

A variable in PHP is a name of memory location that holds data. A variable is a temporary storage that is used to store data temporarily.

In PHP, a variable is declared using $ sign followed by variable name.

Syntax of declaring a variable in PHP is given below:

```
$variablename=value;
```

PHP Variable: Declaring string, integer and float

Let's see the example to store string, integer and float values in PHP variables.

File: variable1.php

```
<?php
$str="hello string";
$x=200;
$y=44.6;
echo "string is: $str <br/>";
echo "integer is: $x <br/>";
echo "float is: $y <br/>";
?>
```

Output:

```
string is: hello string
integer is: 200
float is: 44.6
```

PHP Variable: Sum of two variables

File: variable2.php

```
<?php
$x=5;
$y=6;
$z=$x+$y;
echo $z;
?>
```

Output:

```
11
```

PHP Variable: case sensitive

In PHP, variable names are case sensitive. So variable name "color" is different from Color, COLOR, COLor etc.

File: variable3.php

```
<?php
$color="red";
echo "My car is " . $color . "<br>";
echo "My house is " . $COLOR . "<br>";
echo "My boat is " . $coLOR . "<br>";
?>
```

Output:

```
My car is red
Notice: Undefined variable: COLOR in C:\wamp\www\variable.php on line 4
My house is
Notice: Undefined variable: coLOR in C:\wamp\www\variable.php on line 5
My boat is
```

PHP Variable: Rules

PHP variables must start with letter or underscore only.

PHP variable can't be start with numbers and special symbols.

File: variablevalid.php

```
<?php
$a="hello";//letter (valid)
$_b="hello";//underscore (valid)

echo "$a <br/> $_b";
?>
```

Output:

```
hello
hello
```

File: variableinvalid.php

```
<?php
$4c="hello";//number (invalid)
$*d="hello";//special symbol (invalid)
```

```
echo "$4c <br/> $*d";
?>
```

Output:

```
Parse error: syntax error, unexpected '4' (T_LNUMBER), expecting variable (T_VARIABLE)
 or '$' in C:\wamp\www\variableinvalid.php on line 2
```

PHP: Loosely typed language

PHP is a loosely typed language, it means PHP automatically converts the variable to its correct data type.

var_dump() function

The var_dump() function is used to display structured information (type and value) about one or more variables.

```
var_dump(variable1, variabl2, ....variablen)
```

Example -1:

```
<?php
$var_name1=678;
$var_name2="a678";
$var_name3="678";
$var_name4="W3resource.com";
$var_name5=698.99;
$var_name6=+125689.66;
echo var_dump($var_name1)."<br>";
echo var_dump($var_name2)."<br>";
echo var_dump($var_name3)."<br>";
echo var_dump($var_name4)."<br>";
echo var_dump($var_name5)."<br>";
echo var_dump($var_name6)."<br>";
?>
```

Copy

Output :

```
int(678)
string(4) "a678"
string(3) "678"
string(14) "W3resource.com"
float(698.99)
float(125689.66)
```

PHP $ and $$ Variables

The **$var** (single dollar) is a normal variable with the name var that stores any value like string, integer, float, etc.

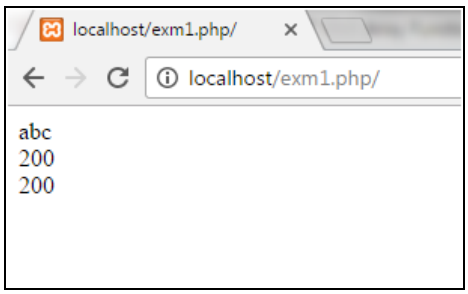
The **$$var** (double dollar) is a reference variable that stores the value of the $variable inside it.

To understand the difference better, let's see some examples.

Example 1

```
<?php
$x = "abc";
$$x = 200;
echo $x."<br/>";
echo $$x."<br/>";
echo $abc;
?>
```

Output:



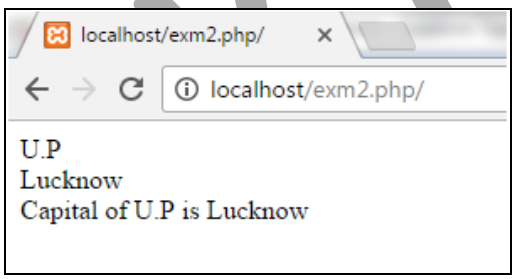
In the above example, we have assigned a value to the variable **x** as **abc**. Value of reference variable **$$x** is assigned as **200**.

Now we have printed the values **$x, $$x** and **$abc**.

Example2

```
<?php
 $x="U.P";
$$x="Lucknow";
echo $x. "<br>";
echo $$x. "<br>";
echo "Capital of $x is " . $$x;
?>
```

Output:



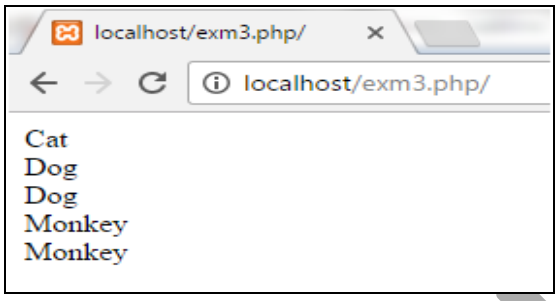
In the above example, we have assigned a value to the variable **x** as **U.P**. Value of reference variable **$$x** is assigned as **Lucknow.** Now we have printed the values **$x, $$x** and a string.

Example3

```
<?php
$name="Cat";
${$name}="Dog";
```

```
${${$name}}="Monkey";
echo $name. "<br>";
echo ${$name}. "<br>";
echo $Cat. "<br>";
echo ${${$name}}. "<br>";
echo $Dog. "<br>";
?>
```

Output:



In the above example, we have assigned a value to the variable name **Cat**. Value of reference variable **${$name}** is assigned as **Dog** and **${${$name}}** as **Monkey**.

Now we have printed the values as **$name, ${$name}, $Cat, ${${$name}}** and **$Dog.**

Super Global Variables in PHP

PHP super global variable is used to access global variables from anywhere in the PHP script.

PHP Super global variables is accessible inside the same page that defines it, as well as outside the page.
while local variable's scope is within the page that defines it.

The PHP super global variables are :

1) $_GET["FormElementName"]

It is used to collect value from a form(HTML script) sent with method='get'.
information sent from a form with the method='get' is visible to everyone(it display on the browser URL bar).

2) $_POST["FormElementName"]

It is used to collect value in a form with method="post". Information sent from a form is invisible to others.(can check on address bar)

3) $_REQUEST["FormElementName"]

This can be used to collect data with both post and get method.

4) $_FILES["FormElementName"]

: It can be used to upload files from a client computer/system to a server.

OR

$_FILES["FormElementName"]["ArrayIndex"]

: Such as File Name, File Type, File Size, File temporary name.

5) $_SESSION["VariableName"]

A session variable is used to store information about a single user, and are available to all pages within one application.

6) $_COOKIE["VariableName"]

A cookie is used to identify a user. cookie is a small file that the server embedded on user computer.

7) $_SERVER["ConstantName"]

$_SERVER holds information about headers, paths, and script locations.

eg.

$_SERVER["SERVER_PORT"],

$_SERVER["SERVER_NAME"],

$_SERVER["REQUEST_URI"]

EXAMPLE

```
<?php
extract($_POST);
if(isset($swap))
```

```
{
        //first number
        $x=$fn;
        //second number
        $y=$sn;

        //third is blank
        $z=0;

        //now put x's values in $z
        $z=$x;

        //and Y's values into $x
        $x=$y;

        //again store $z in $y
        $y=$z;


        //Print the reversed Values
        echo "<p align='center'>Now First numebr is : ". $x ."<br/>";
        echo "and Second number is : ". $y."</p>";
}
?>

<form method="post">
<table align="center">
        <tr>
                <td>Enter First number</td>
                <td><input type="text" name="fn"/></td>
        </tr>
        <tr>
                <td>Enter Second number</td>
                <td><input type="text" name="sn"/></td>
        </tr>
        <tr>
                <td colspan="2" align="center">
                <input type="submit" value="Swap Numbers" name="swap"/></td>
        </tr>
</table>
</form>
```

PHP Constants

PHP constants are name or identifier that can't be changed during the execution of the script. PHP constants can be defined by 2 ways:

1. Using define() function
2. Using const keyword

PHP constants follow the same PHP variable rules. For example, it can be started with letter or underscore only.

Conventionally, PHP constants should be defined in uppercase letters.

PHP constant: define()

Let's see the syntax of define() function in PHP.

define(name, value, case-insensitive)

name: specifies the constant name

value: specifies the constant value

case-insensitive: Default value is false. It means it is case sensitive by default.

Let's see the example to define PHP constant using define().

File: constant1.php

```
<?php
define("MESSAGE","Hello JavaTpoint PHP");
echo MESSAGE;
?>
```

Output:

```
Hello JavaTpoint PHP
```

File: constant2.php

```
<?php
define("MESSAGE","Hello JavaTpoint PHP",true);//not case sensitive
echo MESSAGE;
```

```
echo message;
?>
```

Output:

```
Hello JavaTpoint PHPHello JavaTpoint PHP
```

File: constant3.php

```
<?php
define("MESSAGE","Hello JavaTpoint PHP",false);//case sensitive
echo MESSAGE;
echo message;
?>
```

Output:

```
Hello JavaTpoint PHP
Notice: Use of undefined constant message - assumed 'message'
in C:\wamp\www\vconstant3.php on line 4
message
```

PHP constant: const keyword

The const keyword defines constants at compile time. It is a language construct not a function.

It is bit faster than define().

It is always case sensitive.

File: constant4.php

```
<?php
const MESSAGE="Hello const by JavaTpoint PHP";
echo MESSAGE;
?>
```

Output:

```
Hello const by JavaTpoint PHP
```

PHP Data Types

PHP data types are used to hold different types of data or values. PHP supports 8 primitive data types that can be categorized further in 3 types:

1. Scalar Types
2. Compound Types
3. Special Types

PHP Data Types: Scalar Types

There are 4 scalar data types in PHP.

1. boolean
2. integer
3. float
4. string

PHP Data Types: Compound Types

There are 2 compound data types in PHP.

1. array
2. object

PHP Data Types: Special Types

There are 2 special data types in PHP.

1. resource
2. NULL

Expressions

An expression is a combination of values, variables, operators, and functions that results in a value.

algebra: $y = 3(abs(2x) + 4)$

which in PHP would be: $y = 3 * (abs(2 * $x) + 4);

The value returned (*y*, or $y in this case) can be a number, a string, or a *Boolean value*

TRUE or FALSE? A basic Boolean value can be either TRUE or FALSE. For example, the expression "20 > 9" (20 is greater than 9) is TRUE, and the expression "5 == 6" (5 is equal to 6) is FALSE.

PHP Operators

PHP Operator is a symbol i.e used to perform operations on operands. For example:

```
$num=10+20;//+ is the operator and 10,20 are operands
```

In the above example, + is the binary + operator, 10 and 20 are operands and $num is variable.

PHP Operators can be categorized in following forms:

- Arithmetic Operators
- Comparison Operators
- Bitwise Operators
- Logical Operators
- String Operators
- Incrementing/Decrementing Operators
- Array Operators
- Type Operators
- Execution Operators
- Error Control Operators
- Assignment Operators

We can also categorize operators on behalf of operands. They can be categorized in 3 forms:

- Unary Operators: works on single operands such as ++, -- etc.
- Binary Operators: works on two operands such as binary +, -, *, / etc.
- Ternary Operators: works on three operands such as "?:".

PHP Arithmetic Operators

The arithmetic operators are used to perform common arithmetical operations, such as addition, subtraction, multiplication etc. Here's a complete list of PHP's arithmetic operators:

Operator	Description	Example	Result
+	Addition	$x + $y	Sum of $x and $y
-	Subtraction	$x - $y	Difference of $x and $y.
*	Multiplication	$x * $y	Product of $x and $y.
/	Division	$x / $y	Quotient of $x and $y
%	Modulus	$x % $y	Remainder of $x divided by $y

The following example will show you these arithmetic operators in action:

EXAMPLE:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>PHP Arithmetic Operators</title>
</head>
<body>
<?php
$x = 10;
$y = 4;
echo($x + $y);
echo "<br>";
echo($x - $y);
echo "<br>";
echo($x * $y);
echo "<br>";
echo($x / $y);
echo "<br>";
echo($x % $y);
?>
</body>
</html>
```

OUTPUT

14
6

40
2.5
2

PHP Assignment Operators

The assignment operators are used to assign values to variables.

Operator	Description	Example	Is The Same As
=	Assign	$x = $y	$x = $y
+=	Add and assign	$x += $y	$x = $x + $y
-=	Subtract and assign	$x -= $y	$x = $x - $y
*=	Multiply and assign	$x *= $y	$x = $x * $y
/=	Divide and assign quotient	$x /= $y	$x = $x / $y
%=	Divide and assign modulus	$x %= $y	$x = $x % $y

The following example will show you these assignment operators in action:

EXAMPLE

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>PHP Assignment Operators</title>
</head>
<body>
<?php
$x = 10;
echo $x;
echo "<br>";
$x = 20;
$x += 30;
echo $x;
```

```
echo "<br>";
 $x = 50;
$x -= 20;
echo $x;
echo "<br>";
 $x = 5;
$x *= 25;
echo $x;
echo "<br>";
 $x = 50;
$x /= 10;
echo $x;
echo "<br>";
 $x = 100;
$x %= 15;
echo $x;
?>
</body>
</html>
```

OUTPUT

```
10
50
30
125
5
10
```

PHP Comparison Operators

The comparison operators are used to compare two values in a Boolean fashion.

Operator	Name	Example	Result
==	Equal	$x == $y	True if $x is equal to $y
===	Identical	$x === $y	True if $x is equal to $y, and they are of the same type
!=	Not equal	$x != $y	True if $x is not equal to $y
<>	Not equal	$x <> $y	True if $x is not equal to $y
!==	Not identical	$x !== $y	True if $x is not equal to $y, or they are not of the same type
<	Less than	$x < $y	True if $x is less than $y
>	Greater than	$x > $y	True if $x is greater than $y
>=	Greater than or equal to	$x >= $y	True if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	True if $x is less than or equal to $y

The following example will show you these comparison operators in action:

EXAMPLE

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>PHP Comparison Operators</title>
</head>
<body>
<?php
$x = 25;
$y = 35;
$z = "25";
var_dump($x == $z);
echo "<br>";
var_dump($x === $z);
```

```
echo "<br>";
var_dump($x != $y);
echo "<br>";
var_dump($x !== $z);
echo "<br>";
var_dump($x < $y);
echo "<br>";
var_dump($x > $y);
echo "<br>";
var_dump($x <= $y);
echo "<br>";
var_dump($x >= $y);
?>
</body>
</html>
```

OUTPUT

```
bool(true)
bool(false)
bool(true)
bool(true)
bool(true)
bool(false)
bool(true)
bool(false)
```


PHP Incrementing and Decrementing Operators

The increment/decrement operators are used to increment/decrement a variable's value.

Operator	Name	Effect
++$x	Pre-increment	Increments $x by one, then returns $x

$x++	Post-increment	Returns $x, then increments $x by one
--$x	Pre-decrement	Decrements $x by one, then returns $x
$x--	Post-decrement	Returns $x, then decrements $x by one

The following example will show you these increment and decrement operators in action:

EXAMPLE

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>PHP Incrementing and Decrementing Operators</title>
</head>
<body>
<?php
$x = 10;
echo ++$x;
echo "<br>";
echo $x;
echo "<hr>";
$x = 10;
echo $x++;
echo "<br>";
echo $x;
echo "<hr>";
$x = 10;
echo --$x;
echo "<br>";
echo $x;
echo "<hr>";
```

```
$x = 10;
echo $x--;
echo "<br>";
echo $x;
?>
</body>
</html>
```

OUTPUT

```
11
11
```

```
10
11
```

```
9
9
```

```
10
9
```

PHP Logical Operators

The logical operators are typically used to combine conditional statements.

Operator	Name	Example	Result
and	And	$x and $y	True if both $x and $y are true
or	Or	$x or $y	True if either $x or $y is true
xor	Xor	$x xor $y	True if either $x or $y is true, but not both
&&	And	$x && $y	True if both $x and $y are true
||	Or	$x || $y	True if either $$x or $y is true
!	Not	!$x	True if $x is not true

The following example will show you these logical operators in action:

EXAMPLE

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>PHP Logical Operators</title>
</head>
<body>
<?php
$year = 2014;
// Leap years are divisible by 400 or by 4 but not 100
if(($year % 400 == 0) || (($year % 100 != 0) && ($year % 4 == 0))){
   echo "$year is a leap year.";
} else{
   echo "$year is not a leap year.";
}
?>
</body>
</html>
```

OUTPUT

2014 is not a leap year.

PHP String Operators

There are two operators which are specifically designed for strings.

Operator	Description	Example	Result
.	Concatenation	$str1 . $str2	Concatenation of $str1 and $str2
.=	Concatenation assignment	$str1 .= $str2	Appends the $str2 to the $str1

The following example will show you these string operators in action:

EXAMPLE:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>PHP String Operators</title>
</head>
<body>
<?php
$x = "Hello";
$y = " World!";
echo $x . $y; // Outputs: Hello World!
echo "<br>";
$x .= $y;
echo $x; // Outputs: Hello World!
?>
</body>
</html>
```

OUTPUT

Hello World!
Hello World!

PHP Array Operators

The array operators are used to compare arrays:

Operator	Name	Example	Result
+	Union	$x + $y	Union of $x and $y
==	Equality	$x == $y	True if $x and $y have the same key/value pairs
===	Identity	$x === $y	True if $x and $y have the same key/value pairs in the same order and of the same types

!=	Inequality	$x != $y	True if $x is not equal to $y
<>	Inequality	$x <> $y	True if $x is not equal to $y
!==	Non-identity	$x !== $y	True if $x is not identical to $y

The following example will show you these array operators in action:

EXAMPLE

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>PHP Array Operators</title>
</head>
<body>
<?php
$x = array("a" => "Red", "b" => "Green", "c" => "Blue");
$y = array("u" => "Yellow", "v" => "Orange", "w" => "Pink");
$z = $x + $y; // Union of $x and $y
var_dump($z);
echo "<hr>";
var_dump($x == $y);
echo "<br>";
var_dump($x === $y);
echo "<br>";
var_dump($x != $y);
echo "<br>";
var_dump($x <> $y);
echo "<br>";
var_dump($x !== $y);
?>
```

```
</body>

</html>
```

OUTPUT

```
array(6) { ["a"]=> string(3) "Red" ["b"]=> string(5) "Green" ["c"]=> string(4) "Blue" ["u"]=>
string(6) "Yellow" ["v"]=> string(6) "Orange" ["w"]=> string(4) "Pink" }
```

```
bool(false)
bool(false)
bool(true)
bool(true)
bool(true)
```

PHP Spaceship Operator(PHP7)

PHP 7 introduces a new spaceship operator (<=>) which can be used for comparing two expressions. It is also known as combined comparison operator.

The spaceship operator returns 0 if both operands are equal, 1 if the left is greater, and -1 if the right is greater. It basically provides three-way comparison as shown in the following table:

Operator	**<=> Equivalent**
$x < $y	($x <=> $y) === -1
$x <= $y	($x <=> $y) === -1 \|\| ($x <=> $y) === 0
$x == $y	($x <=> $y) === 0
$x != $y	($x <=> $y) !== 0
$x >= $y	($x <=> $y) === 1 \|\| ($x <=> $y) === 0
$x > $y	($x <=> $y) === 1

The following example will show you how spaceship operator actually works:

EXAMPLE

```
<!DOCTYPE html>

<html lang="en">

<head>

   <title>PHP Spaceship Operators</title>

</head>
```

```
<body>
<?php
// Comparing Integers
echo 1 <=> 1; // Outputs: 0
echo "<br>";
echo 1 <=> 2; // Outputs: -1
echo "<br>";
echo 2 <=> 1; // Outputs: 1
echo "<hr>";
 // Comparing Floats
echo 1.5 <=> 1.5; // Outputs: 0
echo "<br>";
echo 1.5 <=> 2.5; // Outputs: -1
echo "<br>";
echo 2.5 <=> 1.5; // Outputs: 1
echo "<hr>";
 // Comparing Strings
echo "x" <=> "x"; // Outputs: 0
echo "<br>";
echo "x" <=> "y"; // Outputs: -1
echo "<br>";
echo "y" <=> "x"; // Outputs: 1
?>
</body>
</html>
```

OUTPUT

0
-1
1

0
-1
1

0
-1
1

The Ternary Operator

The ternary operator provides a shorthand way of writing the *if...else* statements. The ternary operator is represented by the question mark (?) symbol and it takes three operands: a condition to check, a result for true, and a result for false.

Using the ternary operator the same code could be written in a more compact way:

```
<?php echo ($age < 18) ? 'Child' : 'Adult'; ?>
```

The ternary operator in the example above selects the value on the left of the colon (i.e. 'Child') if the condition evaluates to true (i.e. if $age is less than 18), and selects the value on the right of the colon (i.e. 'Adult') if the condition evaluates to false.

The Modulus Operator in PHP

In mathematics the modulus operation is used to find the remainder of division between two numbers .the expression “5 modulus 3” would evaluate to 2 because 5 divided by 3 leaves a quotient of 1 and a remainder of 2

```
// By using the modulus operator
for($i = 2012 ; $i < 2025 ; $i++)
{
if(($i%4) == 0)
{
$val = "Leap Year <br>" ;
} else {
$val = "Not Leap Year<br>" ;
}
echo $i , ' ---> '. $val  ;
}
```

PHP Operators Precedence

- Let's see the precedence of PHP operators with associativity.

Operators	Additional Information	Associativity
clone new	clone and new	non-associative
[	array()	left
**	arithmetic	right
++ -- ~ (int) (float) (string) (array) (object) (bool) @	increment/decrement and types	right
instanceof	types	non-associative
!	logical (negation)	right
* / %	arithmetic	left
+ - .	arithmetic and string concatenation	left
<< >>	bitwise (shift)	left
< <= > >=	comparison	non-associative
== != === !== <>	comparison	non-associative
&	bitwise AND	left
^	bitwise XOR	left
|	bitwise OR	left
&&	logical AND	left
||	logical OR	left
?:	ternary	left
= += -= *= **= /= .= %= &= |= ^= <<= >>=	assignment	right

=>		
and	logical	left
xor	logical	left
or	logical	left
,	many uses (comma)	left

POSSIBLE QUESTIONS

PART B: 2 MARK QUESTIONS

1. What is PHP?
2. List some of the features of PHP7.
3. What is the difference between "echo" and "print" in PHP?
4. How a variable is declared in PHP?
5. What is the difference between $message and $$message?
6. What are the ways to define a constant in PHP?
7. How many data types are there in PHP?
8. How to do single and multi line comment in PHP?
9. What are super global variables?

PART C: 8 MARK QUESTIONS

1. Explain the features and versions of PHP.
2. Discuss super global variables in PHP.
3. How will you define a constant in PHP? Explain with example.
4. Discuss arithmetic and comparison operators with example.
5. Illustrate ternary and logical operators with example.

KARPAGAM ACADEMY OF HIGHER EDUCATION

Department of Computer Science

III B.Sc(CS) **(BATCH 2016-2019)** **VI SEMESTER**

PHP PROGRAMMING (16CSU603 A)

PART-A OBJECTIVE TYPE/ MULTIPLE CHOICE QUESTIONS

UNIT I

S.NO	QUESTIONS	OPT 1	OPT 2	OPT 3	OPT 4	ANSWER
1	What does PHP stand for? i) Personal Home Page ii) Hypertext Preprocessor iii) Pretext Hypertext Processor iv) Preprocessor Home Page	Both i) and iii)	Both ii) and iv)	Only ii)	Both i) and ii)	Both i) and ii)
2	Who is known as the father of PHP?	Rasmus Lerdorf	Willam Makepiece	Drek Kolkevi	List Barely	Rasmus Lerdorf
3	PHP is an example of _____ scripting language	Server-side	Client-side	Browser-side	In-side	Server-side
4	PHP files have a default file extension of_____	.html	.xml	.php	.ph	.php
5	What will be the output of the following PHP code ? < ?php echo $red ; ?>	0	Nothing	True	Error	Nothing
6	PHP scripts are enclosed within _____	<php> . . . </php>	<?php . . . ?>	?php . . . ?php	) . . .	<?php . . . ?>
7	Which of the following variables is not a predefined variable?	$get	$ask	$request	$post	$ask

8	Which of the below symbols is a newline character?	\r	\n	/n	/r	\n
9	If $a = 12 what will be returned when ($a == 12) ? 5 : 1 is executed?	12	1	Error	5	5
10	What will be the output of the following PHP code ? < ?php Echo "Hello world </br> I am learning PHP"; ?>	Hello world	Hello world I am learning PHP	Hello world I am learning PHP	Error	Hello world I am learning PHP

11	What will be the output of the following PHP code ?
	<?php
	$a = 10;
	echo ++$a;
	echo $a++;
	echo $a;
	echo ++$a;
	?>
		11111213	11121213	11111212	11111112	**11111213**

12	A PHP script should start with ___ and end with ___	< php >	< ? php ?>	<? ?>	<?php ?>	<? ?>
13	Which of the following is/are a PHP code editor? i) Notepad ii) Notepad++ iii) Adobe Dreamweaver iv) PDT	Only iv)	all	i), ii) and iii)	Only iii)	all
14	Which of the following must be installed on your computer so as to run PHP script? i) Adobe Dreamweaver ii) PHP iii) Apache iv) IIS	ll of the mentione	Only ii)	ii) and iii)	ii), iii) and iv)	ii), iii) and iv)

15	Which version of PHP introduced Try/catch Exception?	PHP 4	PHP 5	PHP 5.3	PHP 6	PHP 5
16	We can use ___ to comment a single line? i) /? ii) // iii) # iv) /* */	Only ii)	i), iii) and iv)	ii), iii) and iv)	Both ii) and iv)	ii), iii) and iv)
17	What will be the output of the following php code? <?php $num = 1; $num1 = 2; print $num . "+". $num1; ?>	3	1+2	1.+.2	Error	1+2
18	What will be the output of the following php code? <?php $num = "1"; $num1 = "2"; print $num+$num1; ?>	3	1+2	1.+.2	Error	3
19	Which of following variables can be assigned a value to it? i) $3hello ii) $_hello iii) $this iv) $This	ALL	Only ii)	ii), iii) and iv)	ii) and iv)	ii) and iv)

20	Which of the following PHP statements will output Hello World on the screen? i) echo (“Hello World”); ii) print (“Hello World”); iii) printf (“Hello World”); iv) sprintf (“Hello World”);	i) and ii)	i), ii) and iii)	ALL	i), ii) and iv)	i), ii) and iii)
21	What will be the output of the following PHP code? <?php $color = "maroon"; $var = $color[2]; echo "$var"; ?>	a	Error	$var	r	r
22	Which of the following is not true?	PHP can be used to develop web applications.	PHP makes a website dynamic	PHP applications can not be compile	PHP can not be embedded into html.	PHP can not be embedded into html.
23	In PHP language PEAR stands for	A PHP Enhancement	PHP Event and Application	PHP Extension and Application	None of these above	PHP Extension and Application
24	PHP does not have an built in support for which one of the following ?	JPEG	GIF	MPEG	PDF	MPEG
25	PHP configuration settings are maintained in	A pws-php5cgi.reg	B php.ini	httpd.conf	D httpd-info.conf	php.ini
26	During PHP installation which function creates a HTML page to display records how PHP was installed ?	phpconf()	phpinfo()	phprec()	phpdisplay()	phpinfo()
27	Select the incorrect statement about PHP programming language	Classes are case-insensitive	Functions rae case-insensitive	Variables are case-insensitive	Constants are case-sensitive	Variables are case-insensitive
28	In PHP programming literal is a	Class	Function	Data value	None	Data value

29	Which class name is reserveed in PHP ?	stdClass	nameClass	newClass	None	stdClass
30	When you need to obtain the ASCII value of a character which of the following function you apply in PHP?	chr( );	asc( );	ord( );	val( );	ord( );
31	Which of the following function returns a text in title case from a variable?	ucwords($var)	upper($var)	toupper($var)	ucword($var)	ucwords($var)
32	Which of the following function returns the number of characters in a string variable?	count($variable)	len($variable)	trcount($variable	strlen($variable)	strlen($variable)
33	What will be the output of the following PHP code ? 1. <?php 2. echo "This", "was", "a", "bad", "idea"; 3. ?>	This, was, a, bad, idea	This was a bad idea	Thiswasabadide a	Error	Thiswasabadidea
34	What will be the output of the following PHP code ? 1. <?php 2. $one = 1; 3. print($one); 4. print $one; 5. ?>	1	11	10	error	11
35	What will be the output of the following PHP code ? 1. <?php 2. define("GREETING", "PHP is a scripting language"); 3. echo $GREETING; 4. ?>	$GREETING	no output	PHP is a scripting language	GREETING	no output

36	What will be the output of the following PHP code ? 1. <?php 2. $x = "test"; 3. $y = "this"; 4. $z = "also"; 5. $x .= $y .= $z ; 6. echo $x; 7. echo $y; 8. ?>	testthisthisalso	testthis	estthisalsothisals	error at line 4	testthisalsothisalso
37	What will be the output of the following PHP code ? 1. <?php 2. $y = 2; 3. $w = 4; 4. $y *= $w /= $y; 5. echo $y, $w; 6. ?>	80.5	44	82	42	42
38	What will be the output of the following PHP code ? 1. <?php 2. $a = 'a' ; 3. print $a * 2; 4. ?>	192	2	error	0	0
39	How many basic data types are offered by PHP?	5	6	7	8	8
40	Which of following type conversion behavior is offered by PHP?	Array to boolean	Null to number	Resource to string	All of them	All of them
41	Which one is not a data type in PHP?	Resources	Objects	Null	Void	Void

UNIT-II

Handling HTML form with PHP: Capturing Form Data ,GET and POST form, methods, Dealing with multi value fields, Redirecting a form after submission

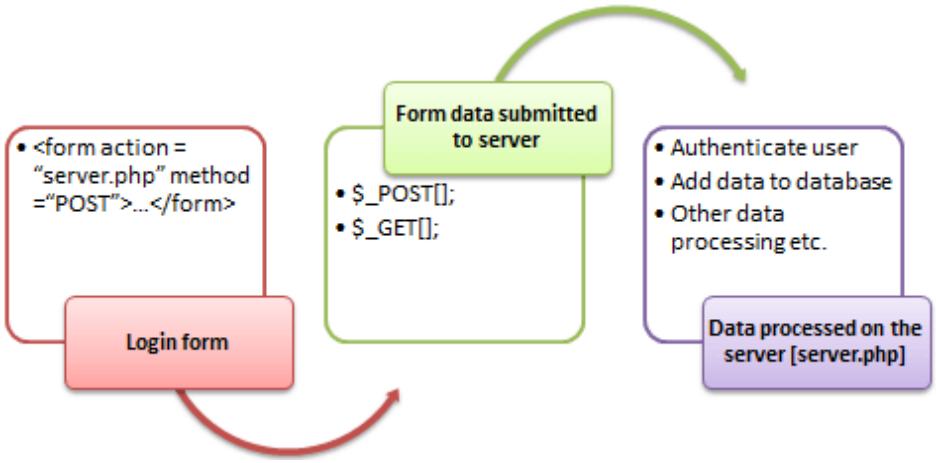
PHP conditional events and Loops: PHP IF Else conditional statements (Nested IF and Else) Switch case, while ,For and Do While Loop , Goto , Break ,Continue and exit

What is Form?

When you login into a website or into your mail box, you are interacting with a form.

Forms are used to get input from the user and submit it to the web server for processing.

The diagram below illustrates the form handling process.



A form is an HTML tag that contains graphical user interface items such as input box, check boxes radio buttons etc.

The form is defined using the <form>...</form> tags and GUI items are defined using form elements such as input.

When and why we are using forms?

- Forms come in handy when developing flexible and dynamic applications that accept user input.
- Forms can be used to edit already existing data from the database

Create a form

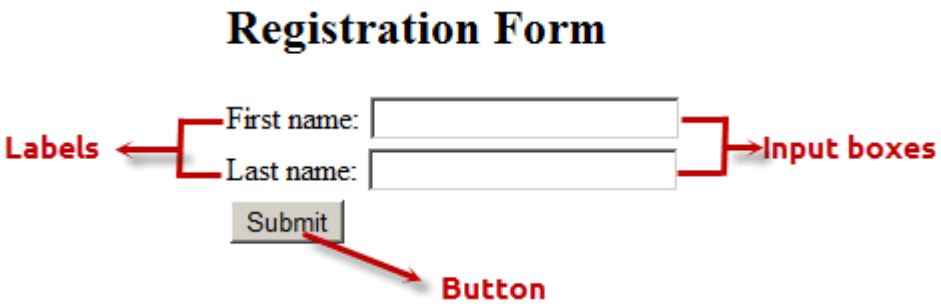
We will use HTML tags to create a form. Below is the minimal list of things you need to create a form.

- Opening and closing form tags <form>…</form>
- Form submission type POST or GET
- Submission URL that will process the submitted data
- Input fields such as input boxes, text areas, buttons,checkboxes etc.

The code below creates a simple registration form

```
<html>
<head>
        <title>Registration Form</title>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
<h2>Registration Form</h2>
<form action="registration_form.php" method="POST"> First name:
<input type="text" name="firstname"><br> Last name:
<input type="text" name="lastname">
<input type="hidden" name="form_submitted" value="1" />
<input type="submit" value="Submit">
</form>
</body>
</html>
```

Viewing the above code in a web browser displays the following form.



HERE,

- <form…>…</form> are the opening and closing form tags
- action="registration_form.php" method="POST"> specifies the destination URL and the submission type.
- First/Last name: are labels for the input boxes
- <input type=”text”…> are input box tags
-
 is the new line tag
- <input type="hidden" name="form_submitted" value="1"/> is a hidden value that is used to check whether the form has been submitted or not
- <input type="submit" value="Submit"> is the button that when clicked submits the form to the server for processing

Submitting the form data to the server

The action attribute of the form specifies the submission URL that processes the data. The method attribute specifies the submission type.

PHP POST method

- This is the built in PHP super global array variable that is used to get values submitted via HTTP POST method.
- The array variable can be accessed from any script in the program; it has a global scope.
- This method is ideal when you do not want to display the form post values in the URL.
- A good example of using post method is when submitting login details to the server.

It has the following syntax.

```
<?php
```

```
$_POST['variable_name'];
?>
```

HERE,

- "$_POST[…]" is the PHP array
- "'variable_name'" is the URL variable name.

PHP GET method

- This is the built in PHP super global array variable that is used to get values submitted via HTTP GET method.
- The array variable can be accessed from any script in the program; it has a global scope.
- This method displays the form values in the URL.
- It's ideal for search engine forms as it allows the users to book mark the results.

It has the following syntax.

```
<?php
$_GET['variable_name'];
?>
```

HERE,

- "$_GET[…]" is the PHP array
- "'variable_name'" is the URL variable name.

When to use GET?

- Information sent from a form with the GET method is **visible to everyone** (all variable names and values are displayed in the URL). GET also has limits on the amount of information to send. The limitation is about 2000 characters. However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.
- GET may be used for sending non-sensitive data.

  Note: GET should NEVER be used for sending passwords or other sensitive information!

When to use POST?

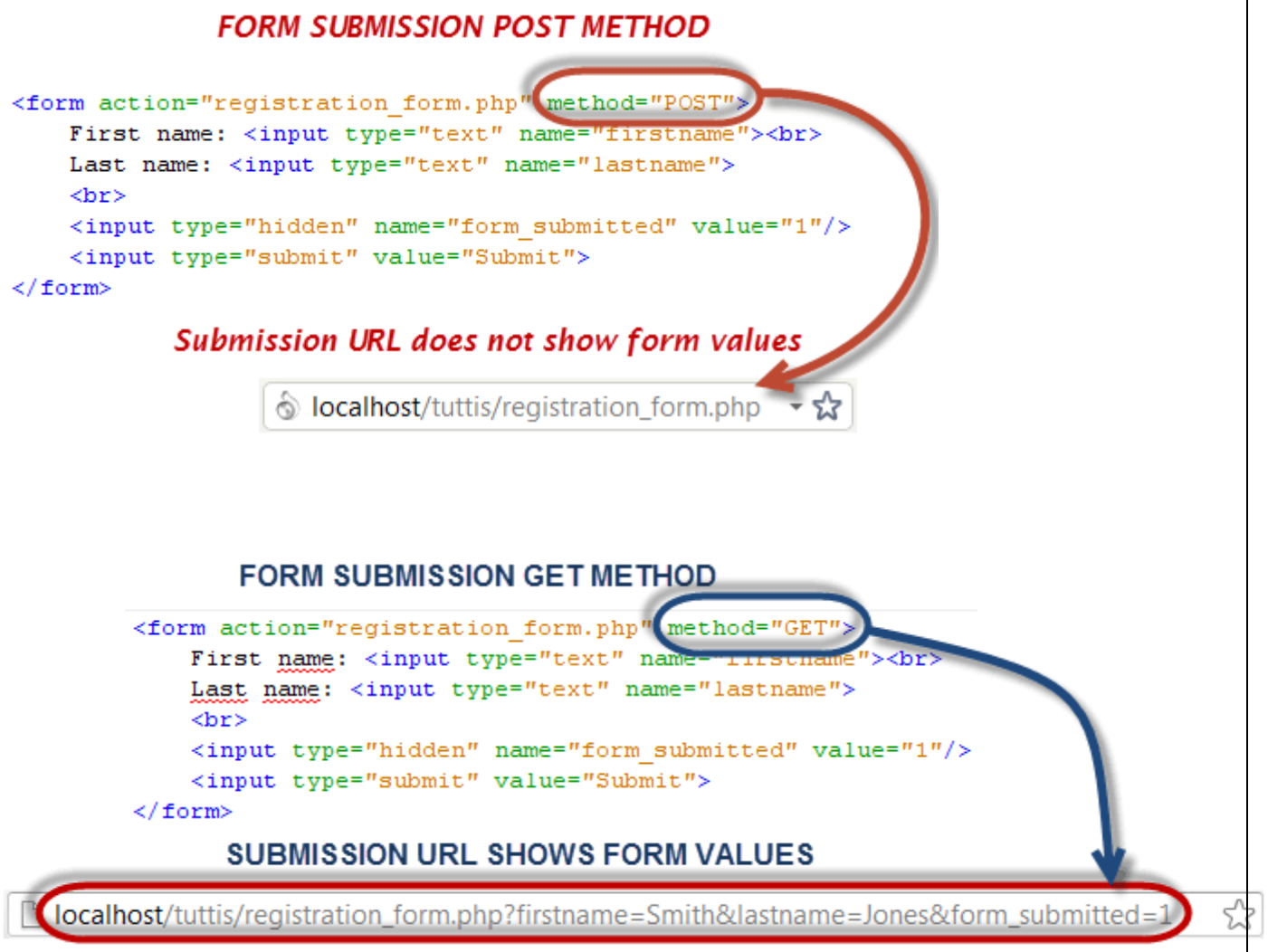
- Information sent from a form with the POST method is **invisible to others** (all names/values are embedded within the body of the HTTP request) and has **no limits** on the amount of information to send.
- Moreover POST supports advanced functionality such as support for multi-part binary input while uploading files to server.
- However, because the variables are not displayed in the URL, it is not possible to bookmark the page.
- Developers prefer POST for sending form data.

GET vs POST Methods

POST	GET
Values not visible in the URL	Values visible in the URL
Has not limitation of the length of the values since they are submitted via the body of HTTP	Has limitation on the length of the values usually 255 characters. This is because the values are displayed in the URL. Note the upper limit of the characters is dependent on the browser.
Has lower performance compared to Php_GET method due to time spent encapsulation the Php_POST values in the HTTP body	Has high performance compared to POST method dues to the simple nature of appending the values in the URL.
Supports many different data types such as string, numeric, binary etc.	Supports only string data types because the values are displayed in the URL
Results cannot be book marked	Results can be book marked due to the

	visibility of the values in the URL

The below diagram shows the difference between get and post



Processing the registration form data

The registration form submits data to itself as specified in the action attribute of the form.

When a form has been submitted, the values are populated in the $_POST super global array.

We will use the PHP isset function to check if the form values have been filled in the $_POST array and process the data.

We will modify the registration form to include the PHP code that processes the data. Below is the modified code

```
<html>
<head>
        <title>Registration Form</title>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
<?php if (isset($_POST['form_submitted'])): ?> //this code is executed when the form is submitted
<h2>Thank You <?php echo $_POST['firstname']; ?></h2>
<p>You have been registered as
<?php echo $_POST['firstname'] . ' ' . $_POST['lastname']; ?>
</p>
<p>Go <a href="/registration_form.php">back</a> to the form</p>
<?php else: ?>
<h2>Registration Form</h2>
<form action="registration_form.php" method="POST">
        First name:
<input type="text" name="firstname">
<br> Last name:
<input type="text" name="lastname">
        <input type="hidden" name="form_submitted" value="1" />
<input type="submit" value="Submit">
</form>
<?phpendif; ? >
</body>
</html>
```

HERE,

- <?php if (isset($_POST['form_submitted'])): ?> checks if the form_submitted hidden field has been filled in the $_POST[] array and display a thank you and first name message.

  If the form_fobmitted field hasn't been filled in the $_POST[] array, the form is displayed.

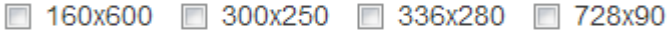
PHP Multi-Values Form Fields

There are form fields that contain multiple values such as multi-select list box:

```
1 <select multiple="multiple"name="formats">
2 <option value="image">image</option>
3 <option value="flash">flash</option>
4 <option value="video">video</option>
5 <option value="HTML5">HTML5</option>
6 </select>
```

and checkboxes with the same name but different values:

```
1 <input type="checkbox"name="sizes"value="160x600"> 160x600
2 <input type="checkbox"name="sizes"value="300x250"> 300x250
3 <input type="checkbox"name="sizes"value="336x280"> 336x280
4 <input type="checkbox"name="sizes"value="728x90">728x90
```



When you submit those fields to the web server, those form fields send multiple values, rather than a single value. So how do you handle these multi-values fields in PHP? It's kind of tricky. You need to add square brackets ([]) after the field name of the multi-valued form field. When PHP sees this sign ([]), it creates a nested array of values within the $_GET or $_POST array based on the method you use in the form.

The name of the multi-select list box should be formats[] and the name of the checkboxes should be sizes[]

Validating multi-valued form fields

When you submit the form that contains multi-valued form fields to the web server without selecting any item from a multi-select list box and or checking any checkbox, nothing is sent to the web server at all. Therefore you cannot handle the multi-values from fields like single-value form fields as follows:

```
1 // validate the formats fields
2 if(count($_POST['formats']>0)){
3 // ...
4 }
```

Because the formats do not exist in the $_POST array, you will get an error message. To check if a multi-values field in submitted with value, you use the isset() function:

```
1 <?php
2
3 // validate the formats fields
4 if(isset($_POST['formats'])){
5 $formats=$_POST['formats'];
6 //
7 }
```

To get value of a checked checkbox :

```
<form action="#" method="post">
<input type="checkbox" name="gender" value="Male">Male</input>
<input type="checkbox" name="gender" value="Female">Female</input>
<input type="submit" name="submit" value="Submit"/>
</form>
<?php
if (isset($_POST['gender'])){
echo $_POST['gender']; // Displays value of checked checkbox.
}
?>
```

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>";
}
}
}
?>
```

In our example, there is a form contains some checkboxes, User checks them and when he/she hits submit button, multiple values of checkboxes will be display.

Our example's complete HTML and PHP codes are given below.

HTML Codes: php_checkbox.php

Given below our complete HTML codes.

```
<!DOCTYPE html>
<html>
<head>
<title>PHP: Get Values of Multiple Checked Checkboxes</title>
<link rel="stylesheet" href="css/php_checkbox.css" />
</head>
<body>
<div class="container">
<div class="main">
<h2>PHP: Get Values of Multiple Checked Checkboxes</h2>
<form action="php_checkbox.php" method="post">
<label class="heading">Select Your Technical Exposure:</label>
```

```
<input type="checkbox" name="check_list[]" value="C/C++"><label>C/C++</label>
<input type="checkbox" name="check_list[]" value="Java"><label>Java</label>
<input type="checkbox" name="check_list[]" value="PHP"><label>PHP</label>
<input type="checkbox" name="check_list[]" value="HTML/CSS"><label>HTML/CSS</label>
<input type="checkbox" name="check_list[]" value="UNIX/LINUX"><label>UNIX/LINUX</label>
<input type="submit" name="submit" Value="Submit"/>
<!----- Including PHP Script ----->
<?php include 'checkbox_value.php';?>
</form>
</div>
</div>
</body>
</html>
```

PHP Codes: checkbox_value.php

In the below script, we used for each loop to display individual value of checked checkboxes, we have also used a counter to count number of checked checkboxes.

```
<?php
if(isset($_POST['submit'])){
if(!empty($_POST['check_list'])) {
// Counting number of checked checkboxes.
$checked_count = count($_POST['check_list']);
echo "You have selected following ".$checked_count." option(s): <br/>";
// Loop to store and display values of individual checked checkbox.
foreach($_POST['check_list'] as $selected) {
echo "<p>".$selected ."</p>";
}
```

```
echo "<br/><b>Note :</b><span>Similarily, You Can Also Perform CRUD Operations using These
Selected Values.</span>";
}
else{
echo "<b>Please Select Atleast One Option.</b>";
}
}
?>
```

CSS File: php_checkbox.css

Styling HTML elements.

```
/* Below line is used for online Google font */
@import url(http://fonts.googleapis.com/css?family=Droid+Serif);
div.container{
width: 960px;
height: 610px;
margin:50px auto;
font-family:'Droid Serif', serif;
}
div.main{
width: 308px;
margin-top: 35px;
float:left;
border-radius: 5px;
Border:2px solid #999900;
padding:0px 50px 20px;
}
p{
```

```
margin-top: 5px;
margin-bottom: 5px;
color:green;
font-weight: bold;
}
h2{
background-color: #FEFFED;
padding: 25px;
margin: 0 -50px;
text-align: center;
border-radius: 5px 5px 0 0;
}
hr{
margin: 0 -50px;
border: 0;
border-bottom: 1px solid #ccc;
margin-bottom:25px;
}
span{
font-size:13.5px;
}
label{
color: #464646;
text-shadow: 0 1px 0 #fff;
font-size: 14px;
font-weight: bold;
}
.heading{
```

```
font-size: 17px;
}
b{
color:red;
}
input[type=checkbox]{
margin-bottom:10px;
margin-right: 10px;
}
input[type=submit]{
padding: 10px;
text-align: center;
font-size: 18px;
background:linear-gradient(#ffbc00 5%, #ffdd7f 100%);
border: 2px solid #e5a900;
color: #ffffff;
font-weight: bold;
cursor: pointer;
text-shadow: 0px 1px 0px #13506D;
width: 100%;
border-radius: 5px;
margin-bottom: 15px;
}
input[type=submit]:hover{
background:linear-gradient(#ffdd7f 5%, #ffbc00 100%);
}
```

PHP Redirect after a Form Submission

Redirection is done by outputting a Location using PHP using the header() function.

Here's how to redirect to a page called thanks.html:

```
header( "Location: thanks.html" );
```

Don't output any content to the browser via echo() or print(), or by including HTML markup outside the <?php ... ?> tags before calling header().

Example

Here's a quick example of a form handler script that redirects to a thank - you page:

```
<?php //from   ww  w  .j  a va2  s .c om
if ( isset( $_POST["submitButton"] ) ) {
    // (deal with the submitted fields here)
header("Location: thanks.html" );
exit;
  } else {
displayForm();
  }
functiondisplayForm() {
  ?>
<!DOCTYPE html5>
<html>
<body>
<form action="index.php" method="post">
<label for="firstName">First name</label>
<input type="text" name="firstName" id="firstName" value="" />
<label for="lastName">Last name</label>
<input type="text" name="lastName" id="lastName" value="" />
<input type="submit" name="submitButton" id="submitButton" value= "Send Details" />
```

```
</form>
</body>
</html>
<?php
  }
  ?>
```

Conditional statement in PHP

PHP lets programmers evaluate different conditions during of a program and take decisions based on whether these conditions evaluate to true of false.

These conditions, and the actions associated with them, are expressed by means of a programming construct called a conditional statement.PHP supports different types of conditional statements

If Else

PHP if else statement is used to test condition. There are various ways to use if statement in PHP.

- if
- if-else
- if-else-if
- nested if

If Statement

PHP if statement is executed if condition is true.

Syntax

```
if(condition){
//code to be executed
}
```

Example

```
1. <?php
2. $num=12;
3. if($num<100){
4. echo "$num is less than 100";
5. }
6. ?>
```

Output:

```
12 is less than 100
```

If-else Statement

PHP if-else statement is executed whether condition is true or false.

Syntax

```
1. if(condition){
2. //code to be executed if true
3. }else{
4. //code to be executed if false
5. }
```

Example

```
1. <?php
2. $num=12;
3. if($num%2==0){
4. echo "$num is even number";
5. }else{
6. echo "$num is odd number";
7. }
8. ?>
```

Output:

```
12 is even number
```

Nested if else in PHP

The **if-else-if-else** statement lets you chain together multiple **if-else** statements, thus allowing the programmer to define actions for more than just two possible outcomes.

Example: Check given character is vowel or consonant

```
<?php
 $char=$_POST['ch'];
        if($char=="a")
        {
        echo $char." is vowel";
        }
        else if($char=="e")
        {
        echo $char." is vowel";
        }
        else if($char=="i")
        {
          echo $char." is vowel";
        }
        else if($char=="o")
        {
          echo $char." is vowel";
        }
        else if($char=="u")
        {
          echo $char." is vowel";
```

```
        }
        else
        {
        echo $char. "is consonent";
        }
?>
        <body>
<form method="post">
Enter Your number<input type="text" name="ch"/><hr/>
<input type="submit"/>
</form>
</body>
```

PHP Switch

PHP switch statement is used to execute one statement from multiple conditions. It works like PHP if-else-if statement.

Syntax

```
1.  switch(expression){
2.  case value1:
3.   //code to be executed
4.   break;
5.  case value2:
6.   //code to be executed
7.   break;
8.  ......
9.  default:
10.  code to be executed if all cases are not matched;
```

```
11.     }
```

PHP Switch Example

```
1.      <?php
2.      $num=20;
3.      switch($num){
4.      case 10:
5.      echo("number is equals to 10");
6.      break;
7.      case 20:
8.      echo("number is equal to 20");
9.      break;
10.     case 30:
11.     echo("number is equal to 30");
12.     break;
13.     default:
14.     echo("number is not equal to 10, 20 or 30");
15.     }
16.     ?>
```

Output:

```
number is equal to 20
```

PHP For Loop

PHP for loop can be used to traverse set of code for the specified number of times.

It should be used if number of iteration is known otherwise use while loop.

Syntax

```
1.      for(initialization; condition; increment/decrement){
```

```
2.  //code to be executed
3.  }
```

Example

```
1.  <?php
2.  for($n=1;$n<=10;$n++){
3.  echo "$n<br/>";
4.  }
5.  ?>
```

Output:

```
1
2
3
4
5
6
7
8
9
10
```

Example: Program to print your name 5 times

```
<?php
$name="rexx";
for ($i=1; $i<=5; $i++)
        {

echo "My Name is: ".$name."<br/>";
        }
```

```
?>
```

Example: Program to print pattern

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

Output

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

Example: Program to print pattern

```
<?php
for($i=1;$i<=5;$i++)
{
for($k=5;$k>$i;$k--)
{
//print one space throgh html ;
echo" ";
}
```

```
for($j=1;$j<=$i;$j++)
{
echo"*";
}
echo"<br/>";
}
?>
```

Output

```
  *
 * *
* * *
* * * *
* * * * *
```

PHP Nested For Loop

We can use for loop inside for loop in PHP, it is known as nested for loop.In case of inner or nested for loop, nested for loop is executed fully for one outer for loop. If outer for loop is to be executed for 3 times and inner for loop for 3 times, inner for loop will be executed 9 times (3 times for 1st outer loop, 3 times for 2nd outer loop and 3 times for 3rd outer loop).

Example

```
1.  <?php
2.  for($i=1;$i<=3;$i++){
3.  for($j=1;$j<=3;$j++){
4.  echo "$i  $j<br/>";
5.  }
6.  }
7.  ?>
```

Output:

```
1 1
1 2
1 3
2 1
2 2
2 3
3 1
3 2
3 3
```

PHP For Each Loop

PHP for each loop is used to traverse array elements.

Syntax

```
1. foreach( $array as $var ){
2.  //code to be executed
3. }
4. ?>
```

Example

```
1. <?php
2. $season=array("summer","winter","spring","autumn");
3. foreach( $season as $arr ){
4.   echo "Season is: $arr<br />";
5. }
6. ?>
```

Output:

```
Season is: summer
```

```
Season is: winter
Season is: spring
Season is: autumn
```

While loop in PHP

The **while** loop executes a block of code while a condition is true.

Syntax:

```
while (condition)
        {
        code to be executed;
        }
```

Example:

```
<?php
$i=1;
        while($i<=5)
        {
          echo "The number is " . $i . "<br>";
          $i++;
        }
 ?>
```

Output

```
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
```

PHP While Loop Example

```
1.  <?php
2.  $n=1;
3.  while($n<=10){
4.  echo "$n<br/>";
5.  $n++;
6.  }
7.  ?>
```

Output:

```
1
2
3
4
5
6
7
8
9
10
```

PHP Nested While Loop

We can use while loop inside another while loop in PHP, it is known as nested while loop.

In case of inner or nested while loop, nested while loop is executed fully for one outer while loop. If outer while loop is to be executed for 3 times and nested while loop for 3 times, nested while loop will be executed 9 times (3 times for 1st outer loop, 3 times for 2nd outer loop and 3 times for 3rd outer loop).

Example

```
1.  <?php
2.  $i=1;
```

```
3.  while($i<=3){
4.  $j=1;
5.  while($j<=3){
6.  echo "$i  $j<br/>";
7.  $j++;
8.  }
9.  $i++;
10. }
11. ?>
```

Output:

```
1 1
1 2
1 3
2 1
2 2
2 3
3 1
3 2
3 3
```

PHP do while loop

PHP do while loop can be used to traverse set of code like php while loop. The PHP do-while loop is guaranteed to run at least once.

It executes the code at least one time always because condition is checked after executing the code.

Syntax

```
do{
//code to be executed
```

}**while**(condition);

Example

```
<?php
$n=1;
do{
echo "$n<br/>";
$n++;
}while($n<=10);
?>
```

Output:

```
1
2
3
4
5
6
7
8
9
10
```

PHP Break

PHP break statement

breaks the execution of current for, while, do-while, switch and for-each loop. If you use break inside inner loop, it breaks the execution of inner loop only.

Syntax

jump statement;

break;

PHP Break: inside loop

Let's see a simple example to break the execution of for loop if value of i is equal to 5.

```
<?php
for($i=1;$i<=10;$i++){
echo "$i <br/>";
if($i==5){
break;
}
}
?>
```

Output:

```
1
2
3
4
5
```

PHP Break: inside inner loop

The PHP break statement breaks the execution of inner loop only.

```
<?php
for($i=1;$i<=3;$i++){
 for($j=1;$j<=3;$j++){
  echo "$i   $j<br/>";
```

```
 if($i==2 && $j==2){
  break;
 }
 }
}
?>
```

Output:

```
1 1
1 2
1 3
2 1
2 2
3 1
3 2
3 3
```

PHP Break: inside switch statement

The PHP break statement breaks the flow of switch case also.

```
<?php
$num=200;
switch($num){
case 100:
echo("number is equals to 100");
break;
case 200:
echo("number is equal to 200");
break;
case 50:
```

```
echo("number is equal to 300");
break;
default:
echo("number is not equal to 100, 200 or 500");
}
?>
```

Output:

```
number is equal to 200
```

PHP continue Statement

Sometimes a situation arises where we want to take the control to the beginning of the loop (for example for, while, do while etc.) skipping the rest statements inside the loop which have not yet been executed.

The keyword continue allow us to do this. When the keyword continue executed inside a loop the control automatically passes to the beginning of loop. Continue is usually associated with the if.

Example:

In the following example, the list of odd numbers between 1 to 10 have printed. In the while loop we test the remainder (here $x%2) of every number, if the remainder is 0 then it becomes an even number and to avoid printing of even numbers continue statement is immediately used and the control passes to the beginning of the loop.

```
<?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++;
}
}
```

Output

List of odd numbers between 1 to 10

1

3

5

7

9

GOTO statement

PHP also supports operator **goto** - which is the operator of unconditional transition. It used to go into another area of the code. A place where you must go to the program, is indicated by the label (a simple identifier), followed by a colon. For the transition after the **goto** statement is put the desired label .

Example

```
<?php
for($i = 0; $i < 5; $i ++)
  {
if($x == 2)
```

```
    {
goto end;
    }
echo $i . " ";
  }
end:
 ?>
// output 0 1
```

Two Mark Questions:

1. What is form?
2. Give the difference beween GET and POST.
3. Mention the uses of GET method.
4. What are conditional statements?
5. Give the syntax of Switch statement in PHP.
6. Write a simple example for “for loop” statement**.**

Six Mark Question

1. Write a PHP script to create multivalued form fields.
2. Write a PHP script to validate a Email form.
3. IIlustrate the usage of redirection form using a simple script.
4. Write a PHP program to iilustrate the working of if-else and nested if else statements
5. Explain the concepts of goto, break, continue statements with example.
6. Differentiate while and do-while statements with example.

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)

(Established Under Section 3 of UGC Act 1956)

Coimbatore – 641 021.

ONE MARK QUESTIONS

DEPARTMENT OF CS, CA & IT

STAFF NAME: Dr.S.MANJU PRIYA & A.JEEVARATHINAM **SUBJECT NAME: PHP PROGRAMMING**

SUB.CODE: 16CSU603A **UNIT II** **SEMESTER: VI**

S.NO	Question	Choice1	Choice2	Choice3	Choice4	Ans
1	_____ are used to get input from the user and submit it to the web server for processing	Reports	Forms	client	database	Forms
2	The form is defined using the _____ tags	<fm…fm>	<frm>… </frm>	<form>… </form>	<for>… </for>	<form>… </form>
3	The _____attribute of the form specifies the submission URL that processes the dat	action	server	event	perform	action
4	_____ is the built in PHP super global array variable that is used to get values	POST	GET	$SET	$PS_SET	POST

5	____is the new line tag	 	<bk>	<break>	<end>	
6	_____ is an array of variables passed to the current script via the URL parameters	$SET	$_GET	POST	GET	$_GET
7	_____is an array of variables passed to the current script via the HTTP POST method	POST	GET	$_GET	$PS_SET	$_GET
8	Information sent from a form with the ____ method is visible to everyone	POST	GET	$_GET	$PS_SET	GET
9	____may be used for sending non-sensitive data	GET	$_GET	$PS_SET	POST	GET
10	Information sent from a form with the ____method is invisible to others	POST	$_GET	$PS_SET	GET	POST
11	The limitation of the GET method is about _____ characters	1000	2000	3000	4000	2000
12	____should NEVER be used for sending passwords or other sensitive information	POST	GET	$_GET	$PS_SET	GET
13	The POST method supports only _____data types because the values are displayed in the URL	boolean	logical	string	integers	string
14	The PHP uses _____function to check if the form values have been filled in the $_POST array	isset	ischeck	isfunction	isform	isset
15	Redirection is done by outputting a Location using PHP using the _____ function	footer()	header()	action()	move()	header()
16	___statements allows you to display output in both the condition	if-else	switch	if-else-if-else	for each	if-else
17	The ____statement is similar to a series of if statements on the same expression	while	switch	if-else-if-else	for each	switch
18	PHP___ statement is executed if condition is true	if-else-if-else	for each	if	while	if

19	_____allows the programmer to define actions for more than just two possible outcomes.	if	if-else-if-else	for each	while	if-else-if-else
20	_____statement works like PHP if-else-if statement.	switch	for	for each	while	switch
21	PHP____loop can be used to traverse set of code for the specified number of times	do while	for	for each	while	for
22	___ loop should be used if number of iteration is know	for	for each	while	do while	for
23	PHP _____loop is used to traverse array elements	for	for each	while	do while	for each
24	The____ loop executes a block of code while a condition is true	for	goto	while	continue	while
25	The PHP do-while loop is guaranteed to run at least __	once	twice	thrice	five times	once
26	PHP ____statement breaks the execution of current lo	goto	break	continue	while	break
27	The PHP break statement breaks the execution of____ loop only	inner	outer	whole prog	none of the above	inner
28	The PHP break statement breaks the flow of _____statements also	switch	goto	break	continue	switch
29	When the keyword _____executed inside a loop the control automatically passes to the beginning of loop	goto	break	switch	continue	continue
30	____ is the operator of unconditional transition	goto	break	continue	switch	goto
31	Which two predefined variables are used to retrieve information from forms?	$GET & $SE	$_GET & $_SET	$__GET & $__SET	GET & SET	$_GET & $_SET

32	When you use the $_POST variable to collect data, the data is visible to____	none	only you	everyone	selected few	only you
33	Which variable is used to collect form data sent with both the GET and POST methods?	$BOTH	$_BOTH	$REQUEST	$_REQUEST	$_REQUEST
34	Which one of the following should not be used while sending passwords or other sensitive information?	GET	POST	REQUEST	NEXT	GET
35	Which of the following method sends input to a script via a URL?	GET	POST	REQUEST	NEXT	GET
36	Which of the following is not PHP Loops?	for	do for	while	switch	do for
37	Most complicated looping structure is	for	do for	while	switch	for
38	For using multiple possible branches based on a single value which branching style do you prefer?	for stateme	switch statements	for each loo	if else	switch statements
39	Two broad types of control structures is branching and	swtiching	unlooping	Looping	ordering	Looping
40	Two main structures of branching are	if and switc	if and Do wh	if and for	if and for each	if and switch
41	Loops that iterate for fixed number of times is called	Bounded loops	loops	For loops	While loops	Bounded loops
42	Which loop evaluates condition expression as Boolean, if it is true, it executes statements and when it is false it will terminate?	do loop	for each	switch	While loop	While loop
43	Which two predefined variables are used to retrieve information from forms?	$_GET & $_SET	$__GET & $__SET	$GET & $SET	GET & SET	$_GET & $_SET

PHP Functions:

- Function, Need of Function , declaration and calling of a function
- PHP Function with arguments, Default Arguments in Function
- Function argument with call by value, call by reference
- Scope of Function Global and Local

PHP Functions

PHP function is a piece of code that can be reused many times. It can take input as argument list and return value. There are thousands of built-in functions in PHP.

In PHP, we can define **Conditional function**, **Function within Function** and **Recursive function** also.

Advantage of PHP Functions

Code Reusability: PHP functions are defined only once and can be invoked many times, like in other programming languages.

Less Code: It saves a lot of code because you don't need to write the logic many times. By the use of function, you can write the logic only once and reuse it.

Easy to understand: PHP functions separate the programming logic. So it is easier to understand the flow of the application because every logic is divided in the form of functions.

PHP User-defined Functions

We can declare and call user-defined functions easily. Let's see the syntax to declare user-defined functions.

Syntax

```
function functionname(){
//code to be executed
}
```

PHP Functions Example

File: function1.php

```
<?php
function sayHello(){
echo "Hello PHP Function";
}
sayHello();//calling function
?>
```

Output:

```
Hello PHP Function
```

PHP Function Arguments

We can pass the information in PHP function through arguments which is separated by comma.

PHP supports **Call by Value** (default), **Call by Reference**, **Default argument values** and **Variable-length argument list**.

Let's see the example to pass single argument in PHP function.

File: functionarg.php

```
<?php
function sayHello($name){
echo "Hello $name<br/>";
}
sayHello("Sonoo");
sayHello("Vimal");
sayHello("John");
?>
```

Output:

```
Hello Sonoo
Hello Vimal
Hello John
```

Let's see the example to pass two argument in PHP function.

File: functionarg2.php

```
<?php
```

```
function sayHello($name,$age){
echo "Hello $name, you are $age years old<br/>";
}
sayHello("Sonoo",27);
sayHello("Vimal",29);
sayHello("John",23);
?>
```

Output:

```
Hello Sonoo, you are 27 years old
Hello Vimal, you are 29 years old
Hello John, you are 23 years old
```

PHP Call By Reference

Value passed to the function doesn't modify the actual value by default (call by value). But we can do so by passing value as a reference.

By default, value passed to the function is call by value. To pass value as a reference, you need to use ampersand (&) symbol before the argument name.

Let's see a simple example of call by reference in PHP.

File: functionref.php

```
<?php
function adder(&$str2)
{
   $str2 .= 'Call By Reference';
}
$str = 'Hello ';
adder($str);
echo $str;
?>
```

Output:

```
Hello Call By Reference
```

PHP Function: Default Argument Value

We can specify a default argument value in function. While calling PHP function if you don't specify any argument, it will take the default argument. Let's see a simple example of using default argument value in PHP function.

File: functiondefaultarg.php

```
<?php
function sayHello($name="Sonoo"){
echo "Hello $name<br/>";
}
sayHello("Rajesh");
sayHello();//passing no value
sayHello("John");
?>
```

Output:

```
Hello Rajesh
Hello Sonoo
Hello John
```

PHP Function: Returning Value

Let's see an example of PHP function that returns value.

File: functiondefaultarg.php

```
<?php
function cube($n){
return $n*$n*$n;
}
echo "Cube of 3 is: ".cube(3);
?>
```

Output:

```
Cube of 3 is: 27
```

PHP Call By Value

PHP allows you to call function by value and reference both. In case of PHP call by value, actual value is not modified if it is modified inside the function.

Example 1

In this example, variable $str is passed to the adder function where it is concatenated with 'Call By Value' string. But, printing $str variable results 'Hello' only. It is because changes are done in the local variable $str2 only. It doesn't reflect to $str variable.

```
<?php
function adder($str2)
{
   $str2 .= 'Call By Value';
}
$str = 'Hello ';
adder($str);
echo $str;
?>
```

Output:

```
Hello
```

Example 2

Let's understand PHP call by value concept through another example.

```
<?php
function increment($i)
{
   $i++;
}
$i = 10;
increment($i);
echo $i;
?>
```

Output:

```
10
```

PHP Call By Reference

In case of PHP call by reference, actual value is modified if it is modified inside the function. In such case, you need to use & (ampersand) symbol with formal arguments. The & represents reference of the variable.

Example 1

In this example, variable $str is passed to the adder function where it is concatenated with 'Call By Reference' string. Here, printing $str variable results 'This is Call By Reference'. It is because changes are done in the actual variable $str.

```
<?php
function adder(&$str2)
{
   $str2 .= 'Call By Reference';
}
$str = 'This is ';
adder($str);
echo $str;
?>
```

Output:

```
This is Call By Reference
```

Example 2

Let's understand PHP call by reference concept through another example.

```
<?php
function increment(&$i)
{
   $i++;
}
$i = 10;
increment($i);
echo $i;
?>
```

Output: 11

PHP Default Argument Values Function

PHP allows you to define C++ style default argument values. In such case, if you don't pass any value to the function, it will use default argument value.

Let' see the simple example of using PHP default arguments in function.

Example 1

```
<?php
function sayHello($name="Ram"){
echo "Hello $name<br/>";
}
sayHello("Sonoo");
sayHello();//passing no value
sayHello("Vimal");
?>
```

Output:

```
Hello Sonoo
Hello Ram
Hello Vimal
```

Example 2

```
<?php
function greeting($first="Sonoo",$last="Jaiswal"){
echo "Greeting: $first $last<br/>";
}
greeting();
greeting("Rahul");
greeting("Michael","Clark");
?>
```

Output:

```
Greeting: Sonoo Jaiswal
Greeting: Rahul Jaiswal
Greeting: Michael Clark
```

Example 3

```
<?php
```

```
function add($n1=10,$n2=10){
$n3=$n1+$n2;
echo "Addition is: $n3<br/>";
}
add();
add(20);
add(40,40);
?>
```

Output:

```
Addition is: 20
Addition is: 30
Addition is: 80
```

PHP Variable Length Argument Function

PHP supports variable length argument function. It means you can pass 0, 1 or n number of arguments in function. To do so, you need to use 3 ellipses (dots) before the argument name.

The 3 dot concept is implemented for variable length argument since PHP 5.6.

Let's see a simple example of PHP variable length argument function.

```
<?php
function add(...$numbers) {
   $sum = 0;
   foreach ($numbers as $n) {
      $sum += $n;
   }
   return $sum;
}

echo add(1, 2, 3, 4);
?>
```

Output:

```
10
```

PHP Recursive Function

PHP also supports recursive function call like C/C++. In such case, we call current function within function. It is also known as recursion.

It is recommended to avoid recursive function call over 200 recursion level because it may smash the stack and may cause the termination of script.

Example 1: Printing number

```
<?php
function display($number) {
   if($number<=5){
    echo "$number <br/>";
    display($number+1);
   }
}

display(1);
?>
```

Output:

```
1
2
3
4
5
```

Example 2 : Factorial Number

```
<?php
function factorial($n)
{
   if ($n < 0)
      return -1; /*Wrong value*/
   if ($n == 0)
      return 1; /*Terminating condition*/
   return ($n * factorial ($n -1));
}

echo factorial(5);
```

```
?>
```

Output:

```
120
```

PHP Functions Scope

Variables declared outside of functions and classes are global. global variables are available else where in the script.

Function variables are self-contained and do not affect variables in the main script.

Variables from the main script are not implicitly made available inside functions.

Example

Take a look at this example:

```
<?PHP
function foo() {
  $bar = "java2s.com";
}
$bar = "PHP";
foo();
print $bar;
?>
```

The code above generates the following result.

```
PHP
```


Execution of the script starts at the $bar = "PHP" line, and then calls the foo() function.

foo() sets $bar to java2s.com, then returns control to the main script where $bar is printed out.

Function foo() is called, and, having no knowledge that a $bar variable exists in the global scope, creates a $bar variable in its local scope.

Once the function ends, all local scopes are gone, leaving the original $bar variable intact.

PHP Global Variables

A global variable can be accessed anywhere in your script, whether inside or outside a function.

In PHP, all variables created outside a function are, in a sense, global in that they can be accessed by any other code in the script that's not inside a function.

To use such a variable inside a function, write the word global followed by the variable name inside the function ' s code block.

```
<?PHP//from  w ww.j av  a  2 s. c o m
      $myGlobal = "Hello there!";

      function hello() {
        global $myGlobal;
        echo "$myGlobal\n";
      }

      hello(); // Displays "Hello there!"
?>
```

The code above generates the following result.

```
Hello there!
```

hello() function accesses the $myGlobal variable by declaring it to be global using the global statement. The function can then use the variable to display the greeting.

Example 1

We don't need to have created a variable outside a function to use it as a global variable. Take a look at the following script:

```
<?PHP//w ww. j  a v a2  s.c o m
      function setup() {
        global $myGlobal;
```

```
    $myGlobal = "Hello there!";
  }

  function hello() {
    global $myGlobal;
    echo "$myGlobal\n";
  }

  setup();
  hello(); // Displays "Hello there!"
?>
```

The code above generates the following result.

```
Hello there!
```

In this script, the setup() function is called first. It declares the $myGlobal variable as global, and gives it a value.

Then the hello() function is called. It too declares $myGlobal to be global, which means it can now access its value previously set by setup() and display it.

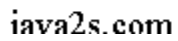
Example 2

The $GLOBALS array can access global variables within functions. All variables declared in the global scope are in the $GLOBALS array, which you can access anywhere in the script. Here is a demonstration:

```
<?PHP
function foo() {
  $GLOBALS['bar'] = "java2s.com";
}
$bar = "PHP";
foo();
```

```
print $bar;
?>
```

The code above generates the following result.



We can read variables in the same way:

```
$localbar = $GLOBALS['bar'];
```

PHP GLOBAL keyword allow a variable to be accessed locally.

```
function myfunc() {
   GLOBAL $foo, $bar, $baz;
   ++$baz;
}
```

The code above reads the global variables $foo, $bar, and $baz. The ++$baz line will increment $baz by 1, and this will be reflected in the global scope.

Note

We can also declare more than one global variable at once on the same line, just separate the variables using commas:

```
function myFunction() {
  global $oneGlobal, $anotherGlobal;
}
```

Be careful with global variables. If you modify the value of a global variable in many different places within your application, it can make it hard to debug your code.

Generally speaking, you should avoid using global variables unless it's strictly necessary.

CLASS : III B.Sc CS **BATCH : 2016 - 2019**

COURSE NAME : PHP PROGRAMMING **COURSE CODE : 16CSU603A**

UNIT III

POSSIBLE QUESTIONS

PART B: 2 MARK QUESTIONS

1. What is the use of header() function in PHP?
2. What is the use of header() function in PHP?
3. What is the use of count() function in PHP?
4. How is it possible to return a value from a function?
5. What is the use of Var_dump?
6. What's the difference between unset () and unlink ()?
7. Define functions with its syntax.

PART C: 8 MARK QUESTIONS

1. Explain in detail about function in PHP.
2. Differentiate call by value and reference with suitable example.
3. Explain functions with arguments with example.
4. Discuss default arguments in function with example.
5. Illustrate the usage of super and global variables in functions with proper example.

KARPAGAM ACADEMY OF HIGHER EDUCATION

Department of Computer Science

III B.Sc(CS) **(BATCH 2016-2019)** **VI SEMESTER**

PHP PROGRAMMING (16CSU603 A)

PART-A OBJECTIVE TYPE/ MULTIPLE CHOICE QUESTIONS

UNIT III

S.NO	QUESTIONS	OPT 1	OPT 2	OPT 3	OPT 4	ANSWER
1	Which one of the following PHP functions can be used to build a function that accepts any number of arguments	func_get_argv()	func_get_argc()	get_argv()	get_argc()	func_get_argc()
2	Which one of the following PHP functions can be used to find files?	glob()	file()	fold()	get_file()	glob()
3	The filesize() function returns the file size in ___.	bits	bytes	kilobytes	gigabytes	bytes
4	Which one of the following PHP function is used to determine a file's last access time?	fileltime()	filectime()	fileatime()	filetime()	fileatime()
5	Which one of the following function is capable of reading a file into an array?	file()	arrfile()	arr_file()	file_arr()	file()
6	The function func_num_args() returns	the number of arguments passed to the function	the total length of the arguments	the number of lines in the program	the number of variables used in the program	the number of arguments passed to the function
7	All these func_num_args ( ), func_get_arg ( ), func_get_args ( ), functions are introduced in	PHP1,	PHP2	PHP3	PHP4	PHP4

8	PHP function arguments are modified in function definition and	In a function call	In execution time	In deceleration time	None of them	In a function call
9	How many ways that PHP offers to modify arguments of a function?	1	2	3	4	2
10	Baseconvert ( ) function converts a	String argument into floats	String argument into numbers	String argument into boolean	String argument into arrays	String argument into numbers
11	A function in PHP which starts with __ (double underscore) is know as..	Magic Function	Inbuilt Function	Default Function	User Defined Function	Magic Function
12	Which one of the following is the right way of defining a function in PHP?	function { function body }	data type functionName(parameters) { function body }	functionName(parameters) { function body }	function fumctionName(parameters) { function body }	function fumctionName(parameters) { function body }
13	Type Hinting was introduced in which version of PHP?	PHP 4	PHP 5	PHP 5.3	PHP 6	PHP 5

14	What will happen in this function call?
	<?php
	function calc($price, $tax)
	{
	$total = $price + $tax;
	}
	$pricetag = 15;
	$taxtag = 3;
	calc($pricetag, $taxtag);
	?>
		Call By Value	Call By Reference	Default Argument Value	Type Hinting	Call By Value

15	Which function sets the file filename's last-modified and last-accessed times?	sets()	set()	touch()	touched()	touch()
16	Which of the following are valid function names? i) function() ii) €() iii) .function() iv) $function()	Only ii)	None	All	iii) and iv)	Only ii)
17	Which one of the following function outputs the contents of a string variable to the specified resource?	filewrite()	filewrite()	filewrites()	fwrites()	filewrite()
18	Which one of the following function is capable of reading a specific number of characters form a file?	fgets()	fget()	fileget()	filegets()	fgets()
19	Which function is used to determine whether a variable is empty or not?	check()	empty()	determine()	fill()	empty()
20	Which function is used to determine whether a variable is set or not?	unset()	isset()	set()	none	isset()
21	In PHP default behavior for user defined functions is	Call By Value	Call By Reference	call by type	none	Call By Value
22	To define a function with default arguments, simply you need to turn formal parameter names into	An assignment expression	An boolean expression	A function	none	An assignment expression
23	____ scope refers to any variable that is defined outside of any function	Local	Global	Static	parameters	Global
24	A parameter is a_____variable whose value is passed to the function by the calling code.	Local	Global	Static	parameters	local

25	A static variable is again a variable with_____scope	Global	Static	parameters	local	local
26	The _____ represents reference of the variable.	**$**	&	*	**@**	&
27	actual value is not modified if it is modified inside the function.	Call By Value	Call By Reference	call by type	none	Call By Value
28	The____ dot concept is implemented for variable length argument since PHP 5.6.	2	3	4	5	3
29	_____ means passing the value directly to a function	Call By Reference	Call by value	call by type	none	Call By Value
30	_____ means passing the address of a variable where the actual value is stored.	Call By Reference	Call by value	call by type	none	Call by reference
31	_____ are specified after the function name, inside the parentheses	reference	Arguments	type	values	Arguments
32	Which of the following function is used to redirect a page?	header()	redirect()	submit()	exit()	header()

33	What will be the output of the following PHP code?
	?php
	function a()
	{
	function b()
	{
	echo 'I am b';
	}
	echo 'I am a';
	}
		I am b	I am bI am a	Error	I am a Error	I am a Error

34	What will happen in this function call?
	<?php
	function calc($price,$tax)
	{
	$total = $price + $tax;
	}
	$pricetag = 15;
		Call By Value	Call By Reference	ault Argument Va	Type Hinting	Call By Value

35	What will be the output of the following PHP code?
	<?php
	function calc($price, $tax="")
	{
	$total = $price + ($price * $tax);
	echo "$total";
	}
	calc(42);
	?>
		Error	0	42	94	42

36	Which of the following are valid function names?	€()	function()	.function()	$function()	function()
37	Which function is useful when you want to output the executed command's result?	out_cmm()	out_system()	cmm()	system()	system()
38	Trace the function that does continue the script execution even if the file inclusion fails	include()	require()	both of above	None of above	include()
39	Which of the following function is used for terminate the script execution in PHP?	break()	quit()	die()	stop()	die()
40	Which of the following is used to check if a function has already been defined?	bool function_exists(functionname)	bool f_exists(functionname)	bool func_exist(functioname)	none	bool function_exists(functionname)

UNIT-IV

String Manipulation and Regular Expression: (3L) Creating and accessing String , Searching & Replacing String ,Formatting, joining and splitting String , String Related Library functions ,Use and advantage of regular expression over inbuilt function ,Use of preg_match(), preg_replace(), preg_split() functions in regular expression

String Manipulation

What is String in PHP

A string is a sequence of letters, numbers, special characters and arithmetic values or combination of all. PHP only supports a 256-character set.
PHP does not support Unicodes. PHP string can be as large as 2GB. The simplest way to create a string is to enclose the string literal (i.e. string characters) in single quotation marks ('), like this:

```
$my_string = 'Hello World';
```

Creating and accessing String

There are 4 ways to specify string in PHP.

- single quoted
- double quoted
- heredoc syntax
- newdoc syntax (since PHP 5.3)

Single Quoted PHP String

We can create a string in PHP by enclosing text in a single quote. It is the easiest way to specify string in PHP.

Example:

```
<?php

$str='Hello text within single quote';
```

```
echo $str;

?>
```

Output:

```
Hello text within single quote
```

We can store multiple line text, special characters and escape sequences in a single quoted PHP string.

```
<?php

$str1='Hello text

multiple line

text within single quoted string';

$str2='Using double "quote" directly inside single quoted string';

$str3='Using escape sequences \n in single quoted string';

echo "$str1 <br/> $str2 <br/> $str3";

?>
```

Output:

```
Hello text multiple line text within single quoted string
Using double "quote" directly inside single quoted string
Using escape sequences \n in single quoted string
```

Example:

```
<?php

$num1=10;

$str1='trying variable $num1';

$str2='trying backslash n and backslash t inside single quoted string \n \t';

$str3='Using single quote \'my quote\' and \\backslash';

echo "$str1 <br/> $str2 <br/> $str3";

?>
```

Output:

```
trying variable $num1
trying backslash n and backslash t inside single quoted string \n \t
Using single quote 'my quote' and \backslash
```

Double Quoted PHP String

In PHP, we can specify string through enclosing text within double quote also. But escape sequences and variables will be interpreted using double quote PHP strings.

```
<?php

$str="Hello text within double quote";

echo $str;

?>
```

Output:

```
Hello text within double quote
```

Now, you **can't use double quote directly** inside double quoted string.

Example:

```
<?php

$str1="Using double "quote" directly inside double quoted string";

echo $str1;

?>
```

Output:

```
Parse error: syntax error, unexpected 'quote' (T_STRING) in C:\wamp\www\string1.php on line
2
```

We **can store multiple line text, special characters and escape sequences** in a double quoted PHP string.

Example:

```
<?php

$str1="Hello text

multiple line

text within double quoted string";

$str2="Using double \"quote\" with backslash inside double quoted string";

$str3="Using escape sequences \n in double quoted string";

echo "$str1 <br/> $str2 <br/> $str3";

?>
```

Output:

```
Hello text multiple line text within double quoted string
```

Using double "quote" with backslash inside double quoted string

Using escape sequences in double quoted string

In double quoted strings, **variable will be interpreted**.

Example:

```
<?php

$num1=10;

echo "Number is: $num1";

?>
```

Output:

```
Number is: 10
```

An example to clarify the differences between single and double quoted strings:

```
<?php
$my_str = 'World';
echo "Hello, $my_str!<br>";     // Displays: Hello World!
echo 'Hello, $my_str!<br>';     // Displays: Hello, $my_str!

echo '<pre>Hello\tWorld!</pre>'; // Displays: Hello\tWorld!
echo "<pre>Hello\tWorld!</pre>"; // Displays: Hello   World!
echo 'I\'ll be back';          // Displays: I'll be back
?>
```

Strings that are delimited by double quotes (as in "this") are preprocessed in both the following two ways by PHP −

- Certain character sequences beginning with backslash (\) are replaced with special characters
- Variable names (starting with $) are replaced with string representations of their values.

The escape-sequence replacements are −

- \n is replaced by the newline character
- \r is replaced by the carriage-return character
- \t is replaced by the tab character
- \$ is replaced by the dollar sign itself ($)
- \" is replaced by a single double-quote (")
- \\ is replaced by a single backslash (\)

Heredoc PHP String

Heredoc is a robust way to create string in PHP with more lines but without using quotations. Heredoc is rarely used as the day by day usage is more complicated as creating strings with quotes or double quotes. Besides this the not properly used heredoc can lead to problems in your code.

However if you want to use it you can do it in the following way:

```
<?php
   $str = <<<DEMO
This is a
demo message
with heredoc.
DEMO;

   echo $str;
?>
```

Output:

This is a demo message with heredoc.

As you see the heredoc starts with the <<< operator and an identifier. After it you can type your text in more lines as if it were a double quoted string. It means that you can use variables inside the heredoc. If you are ready with your text you only need to write the identifier again in a new line as follows:

```
<?php
    $name = "Max";
    $str = <<<DEMO
Hello $name! <br/>
This is a
demo message
with heredoc.
DEMO;

    echo $str;
?>
```

Output:

Hello Max! This is a demo message with heredoc.

Nowdocs PHP String

Nowdocs are to single-quoted strings what heredocs are to double-quoted strings. A nowdoc is specified similarly to a heredoc, but *no parsing is done* inside a nowdoc.

A nowdoc is identified with the same *<<<* sequence used for heredocs, but the identifier which follows is enclosed in single quotes, e.g. *<<<'EOT'*. All the rules for heredoc identifiers also apply to nowdoc identifiers, especially those regarding the appearance of the closing identifier.

Only difference between heredoc's and nowdoc's syntax is that nowdoc's starting identifier is surrounded by single quotes.

```
<?php
$name = "Mike";
//Heredoc example
echo <<<EOT
My name is $name
I love PHP.
EOT;
echo "<br>";
//Nowdoc example
echo <<<'EOT'
My name is $name
I love PHP.
EOT;
?>
```

Output

My name is Mike I love PHP.
My name is $name I love PHP.

String access in PHP

Characters within `string`s may be accessed and modified by specifying the zero-based offset of the desired character after the `string` using square `array` brackets. Think of a `string` as an `array` of characters for this purpose. The

functions `substr()` and `substr_replace()` can be used when you want to extract or replace more than 1 character.

```
Example
<?php
// Get the first character of a string
$str = 'This is a test.';
$first = $str[0];

// Get the third character of a string
$third = $str[2];

// Get the last character of a string.
$str = 'This is still a test.';
$last = $str[strlen($str)-1];

// Modify the last character of a string
$str = 'Look at the sea';
$str[strlen($str)-1] = 'e';

?>
```

Searching & Replacing String

PHP provides various string functions to access and manipulate strings.

Counting of the number of words in a String

Another function which enables display of the number of words in any specific string is str_word_count(). This function is also useful in validation of input fields.

Syntax

Str_word_count(string)

Example

```
<?php
echo str_word_count("Welcome to Cloudways");//will return the number of words in a string
?>
```

Output

3

Finding Text Within a String

Strpos() enables searching particular text within a string. It works simply by matching the specific text in a string. If found, then it returns the specific position. If not found at all, then it will return "False". Strops() is most commonly used in validating input fields like email.

Syntax

Strpos(string,text);

Example

```
<?php
echo strpos("Welcome to Cloudways","Cloudways");
?>
```

Output

11

Replacing text within a string

Str_replace() is a built-in function, basically used for replacing specific text within a string.

Syntax

Str_replace(string to be replaced,text,string)

Example

```
<?php
echo str_replace(“cloudways”, “the programming world”, “Welcome to cloudways”);
?>
```

Output

Welcome to the programming world

Repeating a String

PHP provides a built-in function for repeating a string a specific number of times.

Syntax

Str_repeat(string,repeat)

Example

```
<?php
echo str_repeat(“=”,13);
?>
```

Output

==========

Comparing Strings

You can compare two strings by using strcmp(). It returns output either greater than zero, less than zero or equal to zero. If string 1 is greater than string 2 then it returns greater than zero. If string 1 is less than string 2 then it returns less than zero. It returns zero, if the strings are equal.

Syntax

Strcmp(string1,string2)

Example

```
<?php
echo strcmp(“Cloudways”,”CLOUDWAYS”);
echo “<br>”;
echo strcmp(“cloudways”,”cloudways”);//Both the strings are equal
echo “<br>”;
echo strcmp(“Cloudways”,”Hosting”);
echo “<br>”;
echo strcmp(“a”,”b”);//compares alphabetically
echo “<br>”;
echo strcmp(“abb baa”,”abb baa caa”);//compares both strings and returns the result in terms of number of characters.
?>
```

Output

```
1
0
-1
-1
-4
```

Displaying part of String

Through substr() function you can display or extract a string from a particular position.

Syntax

substr(string,start,length)

Example

```
<?php
echo substr("Welcome to Cloudways",6)."<br>";
echo substr("Welcome to Cloudways",0,10)."<br>";
?>
```

Output

e to Cloudways
Welcome to
s

Removing white spaces from a String

Trim() is dedicated to remove white spaces and predefined characters from a both the sides of a string.

Syntax

trim(string,charlist)

Example

```
<?php
$str = "Wordpess Hosting";
```

```
echo $str . “<br>”;
echo trim(“$str”,”Wording”);
?>
```

Output

Wordpess Hosting

pess Host

Formatting Strings in PHP

`sprintf()` Function

The sprintf() function writes a formatted string to a variable.

The arg1, arg2, ++ parameters will be inserted at percent (%) signs in the main string. This function works "step-by-step". At the first % sign, arg1 is inserted, at the second % sign, arg2 is inserted, etc.

Note: If there are more % signs than arguments, you must use placeholders. A placeholder is inserted after the % sign, and consists of the argument- number and "\$".

Syntax

sprintf(*format,arg1,arg2,arg++*)

Parameter	Description
format	Required. Specifies the string and how to format the variables in it. Possible format values: %% - Returns a percent sign

	%b - Binary number %c - The character according to the ASCII value %d - Signed decimal number (negative, zero or positive) %e - Scientific notation using a lowercase (e.g. 1.2e+2) %E - Scientific notation using a uppercase (e.g. 1.2E+2) %u - Unsigned decimal number (equal to or greather than zero) %f - Floating-point number (local settings aware) %F - Floating-point number (not local settings aware) %g - shorter of %e and %f %G - shorter of %E and %f %o - Octal number %s - String %x - Hexadecimal number (lowercase letters) %X - Hexadecimal number (uppercase letters) Additional format values. These are placed between the % and the letter (example %.2f): + (Forces both + and - in front of numbers. By default, only negative numbers are marked) ' (Specifies what to use as padding. Default is space. Must be used together with the width specifier. Example: %'x20s (this uses "x" as padding) - (Left-justifies the variable value) [0-9] (Specifies the minimum width held of to the variable value) .[0-9] (Specifies the number of decimal digits or maximum string length) Note: If multiple additional format values are used, they must be in the same order as above.
arg1	Required. The argument to be inserted at the first %-sign in the format string

arg2	Optional. The argument to be inserted at the second %-sign in the format string
arg++	Optional. The argument to be inserted at the third, fourth, etc. %-sign in the format string

Example

Replace the percent (%) sign by a variable passed as an argument:

```
<?php
$number = 9;
$str = "Beijing";
$txt = sprintf("There are %u million bicycles in %s.",$number,$str);
echo $txt;
?>
```

Example

Using the format value %f:

```
<?php
$number = 123;
$txt = sprintf("%f",$number);
echo $txt;
?>
```

Example

Use of placeholders:

```
<?php
$number = 123;
```

```
$txt = sprintf("With 2 decimals: %1\$.2f
<br>With no decimals: %1\$u",$number);
echo $txt;
?>
```

Example

A demonstration of all possible format values:

```
<?php
$num1 = 123456789;
$num2 = -123456789;
$char = 50; // The ASCII Character 50 is 2

// Note: The format value "%%" returns a percent sign
echo sprintf("%%b = %b",$num1)."<br>"; // Binary number
echo sprintf("%%c = %c",$char)."<br>"; // The ASCII Character
echo sprintf("%%d = %d",$num1)."<br>"; // Signed decimal number
echo sprintf("%%d = %d",$num2)."<br>"; // Signed decimal number
echo sprintf("%%e = %e",$num1)."<br>"; // Scientific notation (lowercase)
echo sprintf("%%E = %E",$num1)."<br>"; // Scientific notation (uppercase)
echo sprintf("%%u = %u",$num1)."<br>"; // Unsigned decimal number (positive)
echo sprintf("%%u = %u",$num2)."<br>"; // Unsigned decimal number (negative)
echo sprintf("%%f = %f",$num1)."<br>"; // Floating-point number (local settings aware)
echo sprintf("%%F = %F",$num1)."<br>"; // Floating-point number (not local sett aware)
echo sprintf("%%g = %g",$num1)."<br>"; // Shorter of %e and %f
echo sprintf("%%G = %G",$num1)."<br>"; // Shorter of %E and %f
echo sprintf("%%o = %o",$num1)."<br>"; // Octal number
echo sprintf("%%s = %s",$num1)."<br>"; // String
```

```
echo sprintf("%%x = %x",$num1)."<br>"; // Hexadecimal number (lowercase)
echo sprintf("%%X = %X",$num1)."<br>"; // Hexadecimal number (uppercase)
echo sprintf("%%+d = %+d",$num1)."<br>"; // Sign specifier (positive)
echo sprintf("%%+d = %+d",$num2)."<br>"; // Sign specifier (negative)
?>
```

Example : A demonstration of string specifiers:

```
<?php
$str1 = "Hello";
$str2 = "Hello world!";

echo sprintf("[%s]",$str1)."<br>";
echo sprintf("[%8s]",$str1)."<br>";
echo sprintf("[%-8s]",$str1)."<br>";
echo sprintf("[%08s]",$str1)."<br>";
echo sprintf("[%'*8s]",$str1)."<br>";
echo sprintf("[%8.8s]",$str2)."<br>";
?>
```

Joining and splitting String

PHP `join()` Function

Join array elements with a string. The join() function returns a string from the elements of an array.

The join() function is an alias of the `implode()` function.

Note: The join() function accept its parameters in either order. However, for consistency with explode(), you should use the documented order of arguments.

Note: The separator parameter of join() is optional. However, it is recommended to always use two parameters for backwards compatibility.

Syntax

join(*separator,array*)

Parameter	Description
separator	Optional. Specifies what to put between the array elements. Default is "" (an empty string)
array	Required. The array to join to a string

```
<?php
$arr = array('Hello','World!','Beautiful','Day!');
echo join(" ",$arr);
?>
```

Separate the array elements with different characters:

```
<?php
$arr = array('Hello','World!','Beautiful','Day!');
echo join(" ",$arr)."<br>";
echo join("+",$arr)."<br>";
echo join("-",$arr)."<br>";
echo join("X",$arr);
?>
```

`explode()` Function

The explode() function breaks a string into an array.

Note: The "separator" parameter cannot be an empty string.

Note: This function is binary-safe.

Syntax

explode(*separator,string,limit*)

Parameter	Description
separator	Required. Specifies where to break the string
string	Required. The string to split
limit	Optional. Specifies the number of array elements to return. Possible values: Greater than 0 - Returns an array with a maximum of *limit*element(s) Less than 0 - Returns an array except for the last *-limit*elements() 0 - Returns an array with one element

Example: Break a string into an array:

```
<?php
$str = "Hello world. It's a beautiful day.";
print_r (explode(" ",$str));
?>
```

Example: Using the limit parameter to return a number of array elements:

```
<?php
$str = 'one,two,three,four';

// zero limit
print_r(explode(',',$str,0));

// positive limit
print_r(explode(',',$str,2));

// negative limit
print_r(explode(',',$str,-1));
?>
```

String Related Library functions

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

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

PHP ucfirst() function

The ucfirst() function returns string converting first character into uppercase. It doesn't change the case of other characters.

Syntax

string ucfirst (string $str)

Example

```
<?php
$str="my name is KHAN";
$str=ucfirst($str);
echo $str;
?>
```

Output:

My name is KHAN

PHP lcfirst() function

The lcfirst() function returns string converting first character into lowercase. It doesn't change the case of other characters.

Syntax

```
string lcfirst ( string $str )
```

Example

```
<?php
$str="MY name IS KHAN";
$str=lcfirst($str);
echo $str;
?>
```

Output:

mY name IS KHAN

PHP ucwords() function

The ucwords() function returns string converting first character of each word into uppercase.

Syntax

string ucwords (string $str)

Example

```
<?php
$str="my name is Sonoo jaiswal";
$str=ucwords($str);
echo $str;
?>
```

Output:

My Name Is Sonoo Jaiswal

PHP strrev() function

The strrev() function returns reversed string.

Syntax

string strrev (string $string)

Example

```
<?php
$str="my name is Sonoo jaiswal";
$str=strrev($str);
echo $str;
?>
```

Output:

lawsiaj oonoS si eman ym

PHP strlen() function

The strlen() function returns length of the string.

Syntax

int strlen (string $string)

Example

```
<?php
$str="my name is Sonoo jaiswal";
$str=strlen($str);
echo $str;
?>
```

Output:

24

Regular Expressions

What is a Regular Expressions?

Regular expressions are powerful pattern matching algorithm that can be performed in a single expression. Regular expressions use arithmetic operators such as (+,-,^) to create complex expressions. Regular expressions help you accomplish tasks such as validating email addresses, IP address etc.

Why to use regular expressions

Regular expressions simplify identifying patterns in string data by calling a single function. This saves us coding time. When validating user input such as email address, domain names,

telephone numbers, IP addresses, Highlighting keywords in search results, When creating a custom HTML template.

Regular expressions can be used to identify the template tags and replace them with actual data.

Let's now look at the commonly used regular expression functions in PHP.

preg_match – this function is used to perform a pattern match on a string. It returns true if a match is found and false if a match is not found.

preg_split – this function is used to perform a pattern match on a string and then split the results into a numeric array

preg_replace – this function is used to perform a pattern match on a string and then replace the match with the specified text.

Below is the syntax for a regular expression function such as preg_match,preg_split or

preg_replace.

```
<?php
function_name('/pattern/',subject);
?>
```

HERE,

"function_name(...)" is either preg_match, preg_split or preg_replace.

"/.../" The forward slashes denote the beginning and end of our regular expression

"'/pattern/'" is the pattern that we need to matched

"subject" is the text string to be matched against

Let's now look at practical examples that implement the above regular expression functions in PHP.

PHP Preg_match

The first example uses the preg_match function to perform a simple pattern match for the word guru in a given URL.

The code below shows the implementation for the above example.

```
<?php
$my_url = "www.guru99.com";
if (preg_match("/guru/", $my_url))
{
        echo "the url $my_url contains guru";
}
else
{
        echo "the url $my_url does not contain guru";
}
?>
```

Let's examine the part of the code responsible for our output "*preg_match('/guru/', $my_url)*" HERE,

- "preg_match(...)" is the PHP regular expression function
- "'/guru/'" is the regular expression pattern to be matched
- "$my_url" is the variable containing the text to be matched against.

PHP Preg_split

Let's now look at another example that uses the preg_split function.

We will take a string phrase and explode it into an array; the pattern to be matched is a single space.

The text string to be used in this example is "I Love Regular Expressions".

The code below illustrates the implementation of the above example.

```
<?php

$my_text="I Love Regular Expressions";

$my_array  = preg_split("/ /", $my_text);

print_r($my_array );

?>
```

PHP Preg_replace

Let's now look at the preg_replace function that performs a pattern match and then replaces the pattern with something else.

The code below searches for the word guru in a string.

It replaces the word guru with the word guru surrounded by css code that highlights the background colour.

```
<?php
$text = "We at Guru99 strive to make quality education affordable to the masses. Guru99.com";

$text = preg_replace("/Guru/", '<span style="background:yellow">Guru</span>', $text);

echo $text;

?>
```

Two mark Questions

1. How do you create strings n PHP?
2. Mention some of the inbuilt functions in PHP.
3. Give the syntax for heredoc.
4. Mention the functions for formatting a string.
5. What are regular expressions?
6. How do you split the strings?
7. What is the purpose of join()?
8. How do you access a string in arrays?

Six mark Questions:

1. Write a PHP script illustrating the formatting of strings.
2. Explain the inbuilt functions in strings with examples
3. Describe the role of regular expression.
4. Write a PHP script to illustrate the replacing and searching of strings.

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)

(Established Under Section 3 of UGC Act 1956)

Coimbatore – 641 021.

ONE MARK QUESTIONS

DEPARTMENT OF CS, CA & IT

STAFF NAME: Dr.S.MANJU PRIYA & A.JEEVARATHINAM **SUBJECT NAME: PHP PROGRAMMING**

SUB.CODE: 16CSU603A **UNIT IV** **SEMESTER: VI**

S.NO	Question	Choice1	Choice2	Choice3	Choice4	Ans
1	A _____ is a sequence of letters, numbers, special characters and arithmetic values or combination of all	array	string	structures	functions	string
2	PHP only supports a ____character set	256	128	64	32	256
3	PHP string can be as large as ____	less than 100MB	1KB	2GB	less than 2GB	2GB
4	_____the easiest way to specify string in PHP.	single quoted	double quoted	heredoc	newdoc	single quoted

5	We can store multiple line text, special characters and escape sequences in a _____ PHP string.	single quoted	double quoted	heredoc	newdoc	double quoted
6	In_____ strings, variable will be interpreted	single quoted	double quoted	heredoc	newdoc	double quoted
7	_____is used to create string in PHP with more lines but without using quotations	single quoted	double quoted	heredoc	newdoc	heredoc
8	heredoc starts with the ____operator	<<	##	<<<	""	<<<
9	no parsing is done inside a ____	nowdoc	single quoted	double quoted	heredoc	nowdoc
10	_____function enables to display of the number of words in any specific string	str_word()	count()	str_word_ count()	word()	str_word_ count()
11	____enables searching particular text within a string	pos()	stringpos()	strpos()	position()	strpos()
12	_____function is used for replacing specific text within a string	replace()	string_rep()	text_rep()	str_replace()	str_replace()
13	____ function is used for repeating a string a specific number of times	str_ repeat()	string_ rep()	text_rep()	repeat()	str_repeat()
14	You can compare two strings by using ____	strcmp()	comp()	string_ cmp()	str_cmp()	strcmp()
15	Through _____ function you can display or extract a string from a particular position	substr()	extract()	sub()	sub_ext()	substr()
16	____is dedicated to remove white spaces and predefined characters from a both the sides of a string.	remove()	sub()	white()	Trim()	Trim()
17	The ____ function writes a formatted string to a variable	printf()	echo()	sprintf()	format()	sprintf()

18	A placeholder is inserted after the____ sign in a sprint()	&	%	*	$	%
19	The _____function returns a string from the elements of an array.	explote()	join()	split()	array()	join()
20	The join() function is an alias of the _____ function	implode()	explote()	split()	array()	implode()
21	The____ function breaks a string into an array	split()	array()	break()	explode()	explode()
22	The _____ parameter cannot be an empty string	separator	function	string	format	separator
23	The_____ function returns string in lowercase letter	lower()	strtolower()	str_low()	string()	strtolower()
24	The _____ function returns string in uppercase letter	upper()	ucase()	strtoupper()	string()	strtoupper()
25	The_____ function returns string converting first character into uppercase	upper()	ucase()	strtoupper()	ucfirst()	ucfirst()
26	The _____ function returns string converting first character into lowercase	lower()	strlower()	lcfirst()	string()	lcfirst()
27	The _____ function returns string converting first character of each word into uppercase	ucwords()	lcfirst()	string()	upper()	ucwords()
28	The ____function returns reversed string	reverse()	strrev()	str_rev()	string()	strrev()
29	The ____ function returns length of the string	strlen()	length()	str_len()	lg()	strlen()
30	Regular expressions use _____ operators to create complex expressions	binary	assignment	arithmetic	logical	arithmetic

31	____can be used to identify the template tags and replace them with actual data.	string fucntions	Regular expressions	arrays	assocaitive arrays	Regular expressions
32	_____function is used to perform a pattern match on a string	preg_ match	preg_ split	preg_ replace	pre_exp	preg_match
33	____function is used to perform a pattern match on a string and then split the results into a numeric array	preg_ match	preg_ split	preg_ replace	pre_exp	preg_ split
34	____function is used to perform a pattern match on a string and then replace the match with the specified text	preg_ match	preg_ split	preg_ replace	pre_exp	preg_ replace
35	PHP does not support _____	Unicodes	EBIDIC	binary	ASCII	Unicodes
36	Escape sequences and variables will be interpreted using _____PHP strings.	double quote	single quoted	heredoc	newdoc	double quote
37	Strings that are delimited by _____ are preprocessed	single quotes	heredoc	double quotes	newdoc	double quotes
38	____is replaced by the carriage-return character	\c	\r	\v	\b	\r
39	____is replaced by a single backslash (\)	\\	@	%	*	\\
40	In sprint() the arg1, arg2, ++ parameters will be inserted at ____ signs in th		%			%
41	____ represents binary number	%bin	%binary	%b	%zero	%b
42	_____can be used when you want to extract or replace more than 1 character	repalce()	substr_ replace()	ext()	sub()	substr_ replace()

43	_____function is also useful in validation of input fields	validate()	str_word_ count()	check()	count()	str_word_ count()

CLASS : III B.Sc CS **BATCH : 2016 - 2019**

COURSE NAME : PHP PROGRAMMING **COURSE CODE : 16CSU603A**

UNIT V

Array:

- Anatomy of an Array ,Creating index based and Associative array ,Accessing array
- Looping with Index based array, with associative array using each() and foreach()
- Some useful Library function

PHP Arrays

PHP array is an ordered map (contains value on the basis of key). It is used to hold multiple values of similar type in a single variable.

Advantage of PHP Array

Less Code: We don't need to define multiple variables.

Easy to traverse: By the help of single loop, we can traverse all the elements of an array.

Sorting: We can sort the elements of array.

PHP Array Types

There are 3 types of array in PHP.

1. Indexed Array
2. Associative Array
3. Multidimensional Array

PHP Indexed Array

PHP indexed array is an array which is represented by an index number by default. All elements of array are represented by an index number which starts from 0.

PHP indexed array can store numbers, strings or any object. PHP indexed array is also known as numeric array.

Definition

There are two ways to define indexed array:

1st way:

```
$size=array("Big","Medium","Short");
```

2nd way:

```
$size[0]="Big";
$size[1]="Medium";
$size[2]="Short";
```

PHP Indexed Array Example

File: array1.php

```
<?php
$size=array("Big","Medium","Short");
echo "Size: $size[0], $size[1] and $size[2]";
?>
```

Output:

```
Size: Big, Medium and Short
```

File: array2.php

```
<?php
$size[0]="Big";
$size[1]="Medium";
$size[2]="Short";
echo "Size: $size[0], $size[1] and $size[2]";
?>
```

Output:

```
Size: Big, Medium and Short
```

Traversing PHP Indexed Array

We can easily traverse array in PHP using foreach loop. Let's see a simple example to traverse all the elements of PHP array.

File: array3.php

```
<?php
$size=array("Big","Medium","Short");
foreach( $size as $s )
{
  echo "Size is: $s<br />";
}
?>
```

Output:

```
Size is: Big
Size is: Medium
Size is: Short
```

Count Length of PHP Indexed Array

PHP provides count() function which returns length of an array.

```
<?php
$size=array("Big","Medium","Short");
echo count($size);
?>
```

Output:

```
3
```

Loop Through an Indexed Array

To loop through and print all the values of an indexed array, you could use a for loop, like this:

Example

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
$arrlength = count($cars);

for($x = 0; $x < $arrlength; $x++) {
  echo $cars[$x];
  echo "<br>";
```

```
}
?>
```

OUTPUT

```
Volvo
BMW
Toyota
```

PHP Associative Array

PHP allows you to associate name/label with each array elements in PHP using => symbol. Such way, you can easily remember the element because each element is represented by label than an incremented number.

Definition

There are two ways to define associative array:

1st way:

```
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
```

2nd way:

```
$salary["Sonoo"]="550000";
$salary["Vimal"]="250000";
$salary["Ratan"]="200000";
```

Example

File: arrayassociative1.php

```
<?php
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
echo "Sonoo salary: ".$salary["Sonoo"]."<br/>";
echo "Vimal salary: ".$salary["Vimal"]."<br/>";
echo "Ratan salary: ".$salary["Ratan"]."<br/>";
?>
```

Output:

```
Sonoo salary: 550000
Vimal salary: 250000
Ratan salary: 200000
```

File: arrayassociative2.php

```
<?php
$salary["Sonoo"]="550000";
$salary["Vimal"]="250000";
$salary["Ratan"]="200000";
echo "Sonoo salary: ".$salary["Sonoo"]."<br/>";
echo "Vimal salary: ".$salary["Vimal"]."<br/>";
echo "Ratan salary: ".$salary["Ratan"]."<br/>";
?>
```

Output:

```
Sonoo salary: 550000
Vimal salary: 250000
Ratan salary: 200000
```

Traversing PHP Associative Array

By the help of PHP for each loop, we can easily traverse the elements of PHP associative array.

```
<?php
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
foreach($salary as $k => $v) {
echo "Key: ".$k." Value: ".$v."<br/>";
}
?>
```

Output:

```
Key: Sonoo Value: 550000
Key: Vimal Value: 250000
Key: Ratan Value: 200000
```

Loop Through an Associative Array

To loop through and print all the values of an associative array, you could use a foreach loop, like this:

Example

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

foreach($age as $x => $x_value) {
   echo "Key=" . $x . ", Value=" . $x_value;
   echo "<br>";
}
?>
```

OUT PUT

Key=Peter, Value=35
Key=Ben, Value=37
Key=Joe, Value=43

The PHP foreach Loop

The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.

Syntax

```
foreach ($array as $value) {
   code to be executed;
}
```

For every loop iteration, the value of the current array element is assigned to $value and the array pointer is moved by one, until it reaches the last array element.

The following example demonstrates a loop that will output the values of the given array ($colors):

Example

```
<?php
$colors = array("red", "green", "blue", "yellow");
```

```
foreach ($colors as $value) {
  echo "$value <br>";
}
?>
```

OUTPUT

red
green
blue
yellow

PHP each() Function

Definition and Usage

The each() function returns the current element key and value, and moves the internal pointer forward.

This element key and value is returned in an array with four elements. Two elements (1 and Value) for the element value, and two elements (0 and Key) for the element key.

Related methods:

- current() - returns the value of the current element in an array
- end() - moves the internal pointer to, and outputs, the last element in the array
- next() - moves the internal pointer to, and outputs, the next element in the array
- prev() - moves the internal pointer to, and outputs, the previous element in the array
- reset() - moves the internal pointer to the first element of the array

Syntax

each(*array*)

Example

Return the current element key and value, and move the internal pointer forward:

```
<?php
$people = array("Peter", "Joe", "Glenn", "Cleveland");
print_r (each($people));
?>
```

OUTPUT

Array ([1] => Peter [value] => Peter [0] => 0 [key] => 0)

PHP Multidimensional Array

PHP multidimensional array is also known as array of arrays. It allows you to store tabular data in an array. PHP multidimensional array can be represented in the form of matrix which is represented by row * column.

Definition

```
$emp = array
 (
 array(1,"sonoo",400000),
 array(2,"john",500000),
 array(3,"rahul",300000)
 );
```

PHP Multidimensional Array Example

Let's see a simple example of PHP multidimensional array to display following tabular data. In this example, we are displaying 3 rows and 3 columns.

Id	Name	Salary
1	sonoo	400000
2	john	500000
3	rahul	300000

File: multiarray.php

```
<?php
$emp = array
 (
 array(1,"sonoo",400000),
 array(2,"john",500000),
 array(3,"rahul",300000)
 );

for ($row = 0; $row < 3; $row++) {
 for ($col = 0; $col < 3; $col++) {
  echo $emp[$row][$col]."  ";
 }
 echo "<br/>";
}
?>
```

Output:

```
1 sonoo 400000
2 john 500000
3 rahul 300000
```

PHP Array Functions

PHP provides various array functions to access and manipulate the elements of array. The important PHP array functions are given below.

1) PHP array() function

PHP array() function creates and returns an array. It allows you to create indexed, associative and multidimensional arrays.

Syntax

```
array array ([ mixed $... ] )
```

Example

```
<?php
```

```
$season=array("summer","winter","spring","autumn");
echo "Season are: $season[0], $season[1], $season[2] and $season[3]";
?>
```

Output:

```
Season are: summer, winter, spring and autumn
```

2) PHP array_change_key_case() function

PHP array_change_key_case() function changes the case of all key of an array.

Note: It changes case of key only.

Syntax

```
array array_change_key_case ( array $array [, int $case = CASE_LOWER ] )
```

Example

```
<?php
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
print_r(array_change_key_case($salary,CASE_UPPER));
?>
```

Output:

```
Array ( [SONOO] => 550000 [VIMAL] => 250000 [RATAN] => 200000 )
```

Example

```
<?php
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
print_r(array_change_key_case($salary,CASE_LOWER));
?>
```

Output:

```
Array ( [sonoo] => 550000 [vimal] => 250000 [ratan] => 200000 )
```

3) PHP array_chunk() function

PHP array_chunk() function splits array into chunks. By using array_chunk() method, you can divide array into many parts.

Syntax

```
array array_chunk ( array $array , int $size [, bool $preserve_keys = false ] )
```

Example

```
<?php
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
print_r(array_chunk($salary,2));
?>
```

Output:

```
Array (
[0] => Array ( [0] => 550000 [1] => 250000 )
[1] => Array ( [0] => 200000 )
)
```

4) PHP count() function

PHP count() function counts all elements in an array.

Syntax

```
int count ( mixed $array_or_countable [, int $mode = COUNT_NORMAL ] )
```

Example

```
<?php
$season=array("summer","winter","spring","autumn");
echo count($season);
?>
```

Output:

```
4
```

5) PHP sort() function

PHP sort() function sorts all the elements in an array.

Syntax

```
bool sort ( array &$array [, int $sort_flags = SORT_REGULAR ] )
```

Example

```
<?php
$season=array("summer","winter","spring","autumn");
sort($season);
foreach( $season as $s )
{
  echo "$s<br />";
}
?>
```

Output:

```
autumn
spring
summer
winter
```

6) PHP array_reverse() function

PHP array_reverse() function returns an array containing elements in reversed order.

Syntax

```
array array_reverse ( array $array [, bool $preserve_keys = false ] )
```

Example

```
<?php
$season=array("summer","winter","spring","autumn");
$reverseseason=array_reverse($season);
foreach( $reverseseason as $s )
{
```

```
 echo "$s<br />";
}
?>
```

Output:

```
autumn
spring
winter
summer
```

7) PHP array_search() function

PHP array_search() function searches the specified value in an array. It returns key if search is successful.

Syntax

```
mixed array_search ( mixed $needle , array $haystack [, bool $strict = false ] )
```

Example

```
<?php
$season=array("summer","winter","spring","autumn");
$key=array_search("spring",$season);
echo $key;
?>
```

Output:

```
2
```

8) PHP array_intersect() function

PHP array_intersect() function returns the intersection of two array. In other words, it returns the matching elements of two array.

Syntax

```
array array_intersect ( array $array1 , array $array2 [, array $... ] )
```

Example

```
<?php
$name1=array("sonoo","john","vivek","smith");
$name2=array("umesh","sonoo","kartik","smith");
$name3=array_intersect($name1,$name2);
foreach( $name3 as $n )
{
 echo "$n<br />";
}
?>
```

Output:

```
sonoo
smith
```

POSSIBLE QUESTIONS

PART B: 2 MARK QUESTIONS

1. How to access the first element of an array in PHP?
2. Differentiate foreach() and for().
3. List the advantage of PHP Array.
4. What are the different PHP functions to sort an array?
5. How many types of arrays are available in PHP?
6. What is the use of array_chunk() function in PHP?
7. Which PHP function counts all the values of an array?

PART C: 8 MARK QUESTIONS

1. Explain about Associative array in PHP with example.
2. Illustrate Array functions in PHP with proper example
3. Discuss Index based array in PHP.
4. Explain each() and foreach() with proper example

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)

(Established Under Section 3 of UGC Act 1956)

Coimbatore – 641 021.

ONE MARK QUESTIONS

DEPARTMENT OF CS, CA & IT

STAFF NAME: Dr.S.MANJU PRIYA & A.JEEVARATHINAM **SUBJECT NAME: PHP PROGRAMMING**

SUB.CODE: 16CSU603A **UNIT V** **SEMESTER: VI**

S.NO	Question	Choice1	Choice2	Choice3	Choice4	Ans
1	An ____ is a special variable, which can hold more than one value at a time	literal	array	keywords	index	array
2	You can access the array values by referring to an____.	counter	keywords	index number	variable	index number
3	The _____function is used to create an array	array()	arr()	new()	create()	array()
4	_____arrays are with a numeric index	associative	indexed	multiple	declarative	indexed

5	_____ arrays are with named keys	associative	indexed	multiple	declarative	associative
6	_____arrays containing one or more arrays	associative	indexed	declarative	Multidimensional	Multidimensional
7	In index array , the index always starts at ___	0	1	2	3	0
8	The ___ function is used to return the length of an array	length()	term()	index()	count()	count()
9	To loop through and print all the values of an indexed array, you could use a____loop	do	foreach	for	do while	for
10	To loop through and print all the values of an associative array, you could use a ____ loop	do	foreach	do while	for	foreach
11	we can define ____ arrays as array of arrays.	associative	indexed	declarative	Multidimensional	Multidimensional
12	Array elements can be accessed using the ___ syntax	index[key]	array[key]	element [key]	access[key]	array[key]
13	To handle a multiple-item return value from an array, you can use the___function	return	list	display	array	list
14	___ arrays are similar to Map in java	associative	indexed	multiple	declarative	associative
15	We can use ____ function to print the human readable form of the array.	form_r()	print_r()	display_r()	read_r()	print_r()
16	_____splits an array into chunks of arrays	array_ chunk()	split()	chunk()	split_chunk()	array_chunk()
17	____ compare arrays, and returns the differences	compare()	arr_cmp()	array_diff()	difference()	array_diff()

18	____fills an array with values	fills()	array_fill()	values()	insert()	array_fill()
19	____Filters the values of an array using a callback function	filter()	fill()	array_filter()	sort()	array_filter()
20	_____Exchanges all keys with their associated values in an array	flip()	exchange()	array()	array_flip()	array_flip()
21	____function sends each value of an array to a user-made function, which returns new values	map()	array_map()	array()	fill()	array_map()
22	__function merges one or more arrays into one array	merge()	join()	array_merge()	array_join()	array_merge()
23	___ function deletes the last element of an array	pop()	array_pop()	delete()	last()	array_pop()
24	____ function returns one or more random keys from an array	rand()	array_rand()	random()	array_random()	array_rand()
25	____returns an array as a string, using a user-defined function	reduce()	arr_reduce()	array_ reduce()	string()	array_reduce()
26	____returns an array in the reverse order	reverse()	array_ reverse()	arr_rev()	order()	array_reverse()
27	____searches an array for a given value and returns the key	search()	arr_search()	array_ search()	value()	array_search()
28	____Returns selected parts of an array	slice()	split()	arracy_split()	array_slice()	array_slice()
29	____Returns the sum of the values in an array	sum()	array_sum()	total()	array_total()	array_sum()
30	_____Removes duplicate values from an array	unique()	array_ unique()	duplicate()	array_duplicate()	array_unique()

31	_____Create array containing variables and their values	create()	variable()	compact()	value()	compact()
32	_____Returns the current key and value pair from an array	do()	for()	each()	while()	each()
33	_____Sets the internal pointer of an array to its last element	last()	end()	pointer()	element()	end()
34	_____Checks if a specified value exists in an array	exists()	check()	in_array()	array()	in_array()
35	____Fetches a key from an array	key()	value()	array()	exists()	key()
36	Which function will return true if a variable is an array or false if it is not?	this_array()	in_array()	do_array()	is_array()	is_array()
37	Which function can be used to move the pointer to the previous array position?	previous()	prev()	last()	before()	prev()
38	In multidimensional arrays rather than a single key they values are stored in_____	A. Sequence of key values	2 keys	linear style	3 keys	Sequence of key values
39	PHP arrays are also called as	Vector arrays	Perl arrays	Hashes	folders	Hashes
40	PHP indexed array is also known as____ array.	numeric	binary	string	digital	numeric
41	PHP allows you to associate name/label with each array elem	@	%	=>	#	=>
42	_____moves the internal pointer to the first element of the a	first()	reset()	previous()	next()	reset()

Reg .No…….

[16CSU603A]

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)

(Established Under Section 3 of UGC Act, 1956)

COIMBATORE – 641 021

(For the candidates admitted from 2016 onwards)

B.Sc DEGREE EXAMINATION

Sixth Semester

First Internal Examination – December 2018

COMPUTER SCIENCE

PHP PROGRAMMING

Date : 16 –12 – 18 (FN) **Time** : 2:00 Hrs

Class : III.B.Sc CS (A & B) **Maximum** : 50

SECTION A – (20*1 = 20 Marks)
Answer All Questions

1. What does PHP stand for?
   - i) Personal Home Page
   - ii) Hypertext Preprocessor
   - iii) Pretext Hypertext Processor
   - iv) Preprocessor Home Page

   a) Both i) and iii) b) Both ii) and iv) c) Only ii) d) Both i) and ii)

2. Who is known as the father of PHP?

   a) Rasmus Lerdorf b) Willam Makepiece
   c) Drek Kolkevi d) List Barely

3. PHP is an example of _____ scripting language.

   a) Server-side b) Client-side c) Browser-side d) In-side

4. PHP files have a default file extension of_____

   a) .html b) .xml c) .php d) .ph

5. What will be the output of the following PHP code ?

```
< ?php
echo $red ;
?>
```

a) 0 b) Nothing c) True d) Error

6. PHP scripts are enclosed within _____

a) <php> . . . </php> b) <?php . . . ?>

c)?php . . . ?php d) . . .

7. Which of the following variables is not a predefined variable?

a) $get b) $ask c) $request d) $post

8. Which of the below symbols is a newline character?

a) \r b) \n c) /n d) /r

9. If $a = 12 what will be returned when ($a == 12) ? 5 : 1 is executed?

a) 12 b) 1 c) Error d) 5

10. What will be the output of the following PHP code ?

```
< ?php
Echo "Hello world </br> I am learning PHP";
?>
```

a) Hello world b) Hello world I am learning PHP

c) Hello world I am learning PHP d) Error

11. What will be the output of the following PHP code ?

```
<?php
$a = 10;
echo ++$a;
echo $a++;
echo $a;
echo ++$a;
?>
```

a) 11111213 b) 11121213 c) 11111212 d) 11111112

12. Select the incorrect statement about PHP programming language

a) Classes are case-insensitive b) Functions rae case-insensitive

c) Variables are case-insensitive d) Constants are case-sensitive

13. Which of the following function returns the number of characters in a string variable?

a) count($variable) b) len($variable)
c) strcount($variable) d) strlen($variable)

14. The PHP uses _____function to check if the form values have been filled in the $_POST array

a) ischeck b) isset c) check d) isfill

15. Information sent from a form with the ____ method is visible to everyone.

a)POST b) GET c) PS_POST d) PS_VISIBLE

16. ___is the new line tag.

a) <bk> b) <new> c) <newline> d)

17. The ___attribute of the form specifies the submission URL that processes the data a) submit

b) check c) action d) process d) form

18. The form is defined using the _____ tags

a) <form>...</form> b) <f>...</f> c) <fm>...</fm> d) <forms>...</forms>

19. _____are used to get input from the user and submit it to the web server for processing.

a) Reports b) Forms c) Client d) Functions

20. Which variable is used to collect form data sent with both the GET and POST methods?

a) $BOTH b) $_BOTH c) $REQUEST d) $_REQUEST

SECTION B – (3*2 = 6 Marks)

Answer All Questions

21. What are super global variables?

22. Give the difference between GET and POST methods.

23. List the data types in PHP.

SECTION C – (3*8 = 24 Marks)

Answer the Questions

24. a) Explain the features and versions of PHP.

[OR]

b) How will you define a constant in PHP? Explain with example.

25. a) Discuss arithmetic and comparison operators with example.

[OR]

b) Illustrate ternary and logical operators with example.

26. a) Write a PHP script program to illustrate the usage of GET and POST methods.

[OR]

b) Explain the process of redirecting a form after submission in PHP

Reg .No…….

[16CSU603A]

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)

(Established Under Section 3 of UGC Act, 1956)

COIMBATORE – 641 021

(For the candidates admitted from 2016 onwards)

B.Sc DEGREE EXAMINATION

Sixth Semester

Second Internal Examination – February 2019

COMPUTER SCIENCE

PHP PROGRAMMING

Date : 05 –02 – 19 (FN) **Time** : 2:00 Hrs

Class : III.B.Sc CS (A & B) **Maximum** : 50

SECTION A – (20*1 = 20 Marks)
Answer All Questions

1. ___statements allows you to display output in both the condition.
   a) if-else b) switch c) if-else-if-else d) for each

2. PHP____loop can be used to traverse set of code for the specified number of times.
   a) do while b) for c) for each d) while

3. PHP ____statement breaks the execution of current loop.
   a goto b) break c) continue d) while

4. ____ is the operator of unconditional transition.
   a) goto b) break c) continue d) switch

5. ____ scope refers to any variable that is defined outside of any function
   a) Local b) Global c) Static d) parameters

6. The _____ represents reference of the variable.
   a) $ b) & c) * d) @

7. Two broad types of control structures is branching and_____

a) swtiching b) unlooping c) Looping d) ordering

8. _____ are specified after the function name, inside the parentheses

a) reference b) Arguments c) type d) values

9. The____ dot concept is implemented for variable length argument since PHP 5.6.

a) 2 b) 3 c) 4 d) 5

10. A static variable is again a variable with_____scope

a) Global b) Static c) parameters d) local

11. The PHP do-while loop is guaranteed to run at least ____times.

a)once b)twice c)thrice d) five

12. Most complicated looping structure is

a)for b) do c)for…while d) switch

13. PHP function arguments are modified in function definition and

a) In a function call b) In execution time

c) In deceleration time d) None of them

14. In PHP default behavior for user defined functions is

a) Call By Value b) Call By Reference c) call by type d) none

15. The PHP break statement breaks the execution of____ loop only.

a) inner b) outer c) whole program d) none

16. PHP___ statement is executed if condition is true

a)if-else-if-else b) for each c) if d)while

17. _____ means passing the address of a variable where the actual value is stored.

a) Call By Reference b) Call by value c) call by type d) none

18. To define a function with default arguments, simply you need to turn formal parameter names into

a) An assignment expression b) An boolean expression

c) A function d) none

19. ___ loop should be used if number of iteration is known.

a) for b) for each c) while d) do while

20. Which of the following are valid function names?

a) €() b) function() c) .function() d) $function()

SECTION B – (3*2 = 6 Marks)
Answer All Questions

21. Give the syntax for if-else statement.

22. Define functions with its syntax.

23. What is the use of Var_dump?

SECTION C – (3*8 = 24 Marks)
Answer the Questions

24. a) Explain while and Do While Loop statements with example.

(OR)

b) Illustrate with a suitable program about switch and for statements.

25. a) Define function. How to define user defined function? Explain with example.

(OR)

b) Mention the usage of super and global variables in functions with proper example.

26. a) Differentiate call by value and call by reference with suitable example

(OR)

b) Explain functions with arguments with example Program.

Reg .No…….

[16CSU603A]

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)

(Established Under Section 3 of UGC Act, 1956)

COIMBATORE – 641 021

(For the candidates admitted from 2016 onwards)

B.Sc DEGREE EXAMINATION

Sixth Semester

Third Internal Examination – March 2019

COMPUTER SCIENCE

PHP PROGRAMMING

Date : 12 –03 – 19 (FN) **Time** : 2:00 Hrs

Class : III.B.Sc CS (A & B) **Maximum** : 50

SECTION A – (20*1 = 20 Marks)
Answer All Questions

1. In_____ strings, variable will be interpreted.
   a) single quoted b) double quoted c) heredoc d) newdoc

2. _____function is used to perform a pattern match on a string.
   a) preg_match b) preg_split c) preg_replace d) pre_exp

3. _____arrays are with a numeric index
   a) associative b) indexed c) multiple d) declarative

4. ____Returns selected parts of an array
   a) slice() b) split() c) array_split() d) array_slice()

5. PHP string can be as large as ____
   a) less than 100MB b) 1KB c) 2GB d) < 2GB

6. _____is used to create string in PHP with more lines but without using quotations.
   a) single quoted b) double quoted c) heredoc d) newdoc

7. ____enables searching particular text within a string.
a) pos() b) stringpos() c) strpos() d) position()

8. _____Checks if a specified value exists in an array
a) exists() b) check() c) in_array() d) array()

9. _____ arrays are with named keys
a) associative b) indexed c) multiple d) declarative

10. You can compare two strings by using ____
a)strcmp() b)comp() c)string_cmp() d) str_cmp()

11. The _____ parameter cannot be an empty string
a) separator b) function c) string d) format

12. An ____ is a special variable, which can hold more than one value at a time
a) literal b) array c) keywords d) index

13. In index array , the index always starts at ___
a)0 b) 1 c)2 d) 3

14. ___ arrays are similar to Map in java
a) associative b) indexed c) multiple d) declarative

15. heredoc starts with the ____operator.
a) << b)## c)<<< d) " "

16. Regular expressions use _____ operators to create complex expressions
a) binary b) assignment c) arithmetic d) logical

17. ____Returns the sum of the values in an array.
a) sum() b) array_sum( ) c) total() d) array_total()

18. _____moves the internal pointer to the first element of the array
a) first() b) reset() c) previous() d) next()

19. Array elements can be accessed using the ___ syntax
a) index[key] b) array[key] c) element[key] d) access[key]

20. The ____ function writes a formatted string to a variable
a) printf() b) echo() c) sprintf() d) format()

SECTION B – (3*2 = 6 Marks)
Answer All Questions

21. Mention the different ways to create a string in PHP.
22. Define index array with example.
23. Give the use of join() in PHP.

SECTION C – (3*8 = 24 Marks)
Answer the Questions

24. a) Discuss about the string library functions with example.

(OR)

b) Explain in detail about regular expression with suitable example

25. a) Explain each() and foreach() with proper example.

(OR)

b) Illustrate Array functions in PHP with proper example

26. a) With a suitable program, explain about searching and replacing strings

(OR)

b) Why PHP arrays are called associative arrays? Explain with example.

Total No. of Questions: 07

B.sc.(IT) (2015 & Onwards) (Sem. – 4)

PROGRAMMING IN PHP

M Code: 74083

Subject Code: BSIT-401

Paper ID: [74083]

Time: 3 Hrs. **Max. Marks: 60**

INSTRUCTIONS TO CANDIDATES:

1. **SECTION-A is COMPULSORY consisting of TEN questions carrying TWO marks each.**
2. **SECTION-B contains SIX questions carrying TEN marks each and students have to attempt any FOUR questions.**

SECTION A

1. Explain the following:
   a) GET method
   b) Google Caffeine
   c) WYSIWYG
   d) Array
   e) Cookies
   f) Outer join
   g) For loop
   h) String
   i) Function
   j) Create Image

SECTION B

2. Explain various data types in PHP.
3. What is a join? Explain its types.
4. What is the difference between function overloading and overriding? Explain.
5. How web pages are formed using CSS?
6. Explain various date and time functions in PHP.
7. What is RDBMS? How to create a connection between PHP and My SQL?