Biyani's Think Tank

Concept based notes

PHP Programming

(BCA Part-IV Sem.)

Smriti Verma
Asst. Professor
Deptt. of Information Technology
Biyani Girls College, Jaipur

BIYANI
Where you can trust

BIYANI
GROUP OF COLLEGES

Published by :
Think Tanks
Biyani Group of Colleges

Sector-3, Vidhyadhar Nagar,
Jaipur-302 023 (Rajasthan)
Ph : 0141-2338371, 2338591-95 • Fax : 0141-2338007
E-mail : acad@biyanicolleges.org
Website :www.gurukpo.com; www.biyanicolleges.org

First Edition: 2025

Leaser Type Setted by :
Biyani College Printing Department

Preface

I am glad to present this book, especially designed to serve the needs of the students. The book has been written keeping in mind the general weakness in understanding the fundamental concepts of the topics. The book is self-explanatory and adopts the "Teach Yourself" style. It is based on question-answer pattern. The language of book is quite easy and understandable based on scientific approach.

Any further improvement in the contents of the book by making corrections, omission and inclusion is keen to be achieved based on suggestions from the readers for which the author shall be obliged.

I acknowledge special thanks to Mr. Rajeev Biyani, *Chairman* & Dr. Sanjay Biyani, *Director* (*Acad.*) Biyani Group of Colleges, who are the backbones and main concept provider and also have been constant source of motivation throughout this Endeavour. They played an active role in coordinating the various stages of this Endeavour and spearheaded the publishing work.

I look forward to receiving valuable suggestions from professors of various educational institutions, other faculty members and students for improvement of the quality of the book. The reader may feel free to send in their comments and suggestions to the under mentioned address.

Author

Syllabus

UNIT-I

Introduction to PHP: Installation of PHP and MySQL, PHP configuration in IIS & Apache Web Server. Features of PHP, Writing PHP, Parsing PHP code, Embedding PHP and HTML Executing PHP and viewing in Browser.

Unit - II

Control Structures: Data types, Operators, PHP variables: static and global variables, Comments in PHP, Control Structures, Condition statements, If…Else, Switch, ? operator, Loops, While, Break Statement Continue. Do…While, For, For each, Exit, Die, Return. Arrays: Numeric, Associative and Multidimensional Arrays

UNIT-III

Strings: Creating and accessing String, Searching & Replacing String, Formatting String, String Related Library function, Pattern matching, Replacing text, Splitting a string with a Regular Expression

Functions: Defining a Function, Calling a Function, Parameter passing, Returning value from function

Form Data Handling: $_GET, $_POST, $_REQUEST Variables, Cookies handling, Session Management

UNIT-IV

Exception Handling: Understanding Exception and error, Try, catch, throw

File Handling: Opening and closing a file, Copying, renaming and deleting a file
Database Handling: Connection with MySql Database or ODBC, Performing basic database, operation (Insert, Delete, Update, Select, Truncate Alias, Order By), Setting query parameter.

Content

Chapter-1

Introduction of PHP

Q.1 Write a note on History of PHP?

Ans.: History of PHP: PHP is a general-purpose scripting language geared towards web development. It was originally created by Danish-Canadian programmer Rasmus Lerdorf in 1993 and released in 1995. The PHP reference implementation is now produced by the PHP Group.[11] PHP was originally an abbreviation of Personal Home Page, but it now stands for the recursive initialism PHP: Hypertext Preprocessor.

Q.2. What is PHP?

Ans.: PHP (recursive acronym for PHP: Hypertext Preprocessor) is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML. The language is used primarily for server-side scripting, although it can also be used for command-line scripting and, to a limited degree, desktop applications.

Q.3. What are the basic features of PHP Language?

Ans.: PHP Features: Here we list the basic features that make PHP a powerful and popular programming language :

1. **Simplicity:** PHP is particularly famous for its simplicity. It is organized and easy to learn. There is no need to include libraries in PHP like C. With a lot of pre-defined functions, PHP is easy to optimize as well.
2. **Flexibility:** PHP scripts can run on any device- mobile, tablet, or PC. It is very compatible with various databases. It can be easily embedded and integrated into HTML, XML, and JavaScript. Likewise, it is also compatible with almost all servers used today like Apache, IIS, etc.
3. **Objective oriented:** PHP supports object-oriented programming features like data encapsulation, inheritance, abstraction, polymorphism, etc. The Object-oriented programming feature was added in PHP5. This feature helps in building complex reusable web pages and makes PHP comparable to powerful object-oriented languages like Java and Python.

4. **Interpreted language:** PHP is an interpreted language, which means there is no need for compilation. Interpreters run through a program line by line and execute

the code. Since interpreters execute source code themselves, the code becomes platform-independent. Some other benefits of interpreted language include dynamic typing and short executable program size.

5. **Efficient:** PHP is a versatile, reliable, and efficient programming language. The memory management of PHP is very efficient. Great session management, eliminating unnecessary memory allocation, are some of the features that make PHP efficient.

6. **Fast Performance:** PHP scripts are usually faster than other scripting languages. Users can load their web pages faster.

7. **Free and open-source:** PHP is open-source, which means it can be downloaded and used freely. There is absolutely no hassle to acquire a license to use it and no payment is required to use it, so it is kind to your pocket too!

8. **Case-sensitive:** PHP is a partially case-sensitive language. Although functions names are not case-sensitive, other things in PHP are case-sensitive. The following things in PHP are case-sensitive:Variable names, Constructs (if, if-else, if-elseif, while, do-while)**,**Keywords (such as true and false)**,**User-defined functions and class names.

9. **Security:** PHP has many pre-defined functions for data encryption. PHP is designed specifically to be a more secure language for writing CGI (Computer-generated Imagery) programs. Security algorithms such as Sha1 (secure Hash algorithm 1) and MD5(Message digest 5) are used to encrypt the strings in PHP.

10. **Platform independent**: We can run PHP on any device and operating system (Microsoft Windows, macOS, Linux, RISC OS, or Unix). We can easily connect it with various databases and is also compatible with almost all web servers used today (Apache, IIS, and others). It supports a wide range of databases as well. Its cross-platform compatibility makes really popular among its users as it saves a lot of time and energy.

11. **Loosely typed language:** PHP supports variable declaration without declaring its data type.

12. **Real-time access monitoring:** PHP provides real-time information about users' access. It provides a summary of recent accesses of the user. PHP offers a secure user management system and prevents unrestricted access.

13. **Error reporting and handling:** PHP has many pre-defined functions and reporting constants that generate errors at runtime.PHP5 allows you to use semantics like try, throw and catch, like Java and C#.

14. **Active community support:** PHP has got the backing and support of many users and volunteers across the globe. These volunteers contribute to many features and versions of PHP libraries. They also contribute to overcoming the language barrier by translating in different languages to help new programmers.

Q.3. Write basic syntax of PHP.

Ans.: A PHP file contains HTML tags and some PHP scripting code. 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.

Note: PHP statements ends with semicolon (;).
All PHP code goes between the php tag. It starts with <?php and ends with ?>. The syntax of PHP tag is given below:

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

Q4. What is data type? Explain different types of 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:
Scalar Types (predefined)
Compound Types (user-defined)
Special Types

PHP Data Types: Scalar Types
It holds only single value. There are 4 scalar data types in PHP.
boolean
integer
float
string

PHP Data Types: Compound Types
It can hold multiple values. There are 2 compound data types in PHP.
array
object

PHP Data Types: Special Types
There are 2 special data types in PHP.
resource
NULL

PHP Boolean
Booleans are the simplest data type works like switch. It holds only two values: TRUE (1) or FALSE (0). It is often used with conditional statements. If the condition is correct, it returns TRUE otherwise FALSE.

Example:

```
<?php
  if (TRUE)
     echo "This condition is TRUE.";
  if (FALSE)
```

```
    echo "This condition is FALSE.";
?>
```

Output:

This condition is TRUE.

PHP Integer

Integer means numeric data with a negative or positive sign. It holds only whole numbers, i.e., numbers without fractional part or decimal points.

Rules for integer:
An integer can be either positive or negative.
An integer must not contain decimal point.
Integer can be decimal (base 10), octal (base 8), or hexadecimal (base 16).
The range of an integer must be lie between 2,147,483,648 and 2,147,483,647 i.e., -2^31 to 2^31.
Example:

```
<?php
  $dec1 = 34;
  $oct1 = 0243;
  $hexa1 = 0x45;
  echo "Decimal number: " .$dec1. "</br>";
  echo "Octal number: " .$oct1. "</br>";
  echo "HexaDecimal number: " .$hexa1. "</br>";
?>
```

Output:

Decimal number: 34
Octal number: 163
HexaDecimal number: 69

PHP Float

A floating-point number is a number with a decimal point. Unlike integer, it can hold numbers with a fractional or decimal point, including a negative or positive sign.

Example:

```
<?php
  $n1 = 19.34;
  $n2 = 54.472;
  $sum = $n1 + $n2;
  echo "Addition of floating numbers: " .$sum;
?>
```

Output:

Addition of floating numbers: 73.812

PHP String

A string is a non-numeric data type. It holds letters or any alphabets, numbers, and even special characters.

String values must be enclosed either within single quotes or in double quotes. But both are treated differently. To clarify this, see the example below:

Example:

```
<?php
   $company = "Javatpoint";
   //both single and double quote statements will treat different
   echo "Hello $company";
   echo "</br>";
   echo 'Hello $company';
?>
```

Output:

```
Hello Javatpoint
Hello $company
```

PHP Array

An array is a compound data type. It can store multiple values of same data type in a single variable.

Example:

```
<?php
   $bikes = array ("Royal Enfield", "Yamaha", "KTM");
   var_dump($bikes);  //the var_dump() function returns the datatype and values
   echo "</br>";
   echo "Array Element1: $bikes[0] </br>";
   echo "Array Element2: $bikes[1] </br>";
   echo "Array Element3: $bikes[2] </br>";
?>
```

Output:

```
array(3) { [0]=> string(13) "Royal Enfield" [1]=> string(6) "Yamaha" [2]=> string(3) "KTM" }
Array Element1: Royal Enfield
Array Element2: Yamaha
Array Element3: KTM
```

You will learn more about array in later chapters of this tutorial.

PHP object

Objects are the instances of user-defined classes that can store both values and functions. They must be explicitly declared.

Example:

```
<?php
```

```
class bike {
    function model() {
        $model_name = "Royal Enfield";
        echo "Bike Model: " .$model_name;
      }
}
$obj = new bike();
$obj -> model();
?>
```

Output:

Bike Model: Royal Enfield
This is an advanced topic of PHP, which we will discuss later in detail.

PHP Resource
Resources are not the exact data type in PHP. Basically, these are used to store some function calls or references to external PHP resources. For example - a database call. It is an external resource.

This is an advanced topic of PHP, so we will discuss it later in detail with examples.

PHP Null
Null is a special data type that has only one value: NULL. There is a convention of writing it in capital letters as it is case sensitive.

The special type of data type NULL defined a variable with no value.

Example:

```
<?php
  $nl = NULL;
  echo $nl;  //it will not give any output
?>
```

Q5. What is operator? Explain different types of operators in PHP.

PHP Operator is a symbol i.e used to perform operations on operands. In simple words, operators are used to perform operations on variables or values. For example:

1. $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
- Assignment Operators
- Bitwise Operators
- Comparison Operators
- Incrementing/Decrementing Operators
- Logical Operators
- String Operators
- Array Operators
- Type Operators
- Execution Operators
- Error Control 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 "?:".

Arithmetic Operators

The PHP arithmetic operators are used to perform common arithmetic operations such as addition, subtraction, etc. with numeric values.

Operator	Name	Example	Explanation
+	Addition	$a + $b	Sum of operands
-	Subtraction	$a - $b	Difference of operands
*	Multiplication	$a * $b	Product of operands
/	Division	$a / $b	Quotient of operands
%	Modulus	$a % $b	Remainder of operands
**	Exponentiation	$a ** $b	$a raised to the power $b

The exponentiation (**) operator has been introduced in PHP 5.6.

Assignment Operators

The assignment operators are used to assign value to different variables. The basic assignment operator is "=".

Operator	Name	Example	Explanation
=	Assign	$a = $b	The value of right operand is assigned to the left operand.
+=	Add then Assign	$a += $b	Addition same as $a = $a + $b
-=	Subtract then Assign	$a -= $b	Subtraction same as $a = $a - $b
*=	Multiply then Assign	$a *= $b	Multiplication same as $a = $a * $b
/=	Divide then Assign (quotient)	$a /= $b	Find quotient same as $a = $a / $b
%=	Divide then Assign (remainder)	$a %= $b	Find remainder same as $a = $a % $b

Bitwise Operators

The bitwise operators are used to perform bit-level operations on operands. These operators allow the evaluation and manipulation of specific bits within the integer.

Operator	Name	Example	Explanation
&	And	$a & $b	Bits that are 1 in both $a and $b are set to 1, otherwise 0.
|	Or (Inclusive or)	$a | $b	Bits that are 1 in either $a or $b are set to 1
^	Xor (Exclusive or)	$a ^ $b	Bits that are 1 in either $a or $b are set to 0.
~	Not	~$a	Bits that are 1 set to 0 and bits that are 0 are set to 1
<<	Shift left	$a << $b	Left shift the bits of operand $a $b steps
>>	Shift right	$a >> $b	Right shift the bits of $a operand by $b number of places

Comparison Operators

Comparison operators allow comparing two values, such as number or string. Below the list of comparison operators are given:

Operator	Name	Example	Explanation
==	Equal	$a == $b	Return TRUE if $a is equal to $b
===	Identical	$a === $b	Return TRUE if $a is equal to $b, and they are of same data type
!==	Not identical	$a !== $b	Return TRUE if $a is not equal to $b, and they are not of same data type
!=	Not equal	$a != $b	Return TRUE if $a is not equal to $b
<>	Not equal	$a <> $b	Return TRUE if $a is not equal to $b
<	Less than	$a < $b	Return TRUE if $a is less than $b
>	Greater than	$a > $b	Return TRUE if $a is greater than $b
<=	Less than or equal to	$a <= $b	Return TRUE if $a is less than or equal $b
>=	Greater than or equal to	$a >= $b	Return TRUE if $a is greater than or equal $b
<=>	Spaceship	$a <=>$b	Return -1 if $a is less than $b Return 0 if $a is equal $b Return 1 if $a is greater than $b

Incrementing/Decrementing Operators

The increment and decrement operators are used to increase and decrease the value of a variable.

Operator	Name	Example	Explanation
++	Increment	++$a	Increment the value of $a by one, then return $a
		$a++	Return $a, then increment the value of $a by one
--	decrement	--$a	Decrement the value of $a by one, then return $a
		$a--	Return $a, then decrement the value of $a by one

Logical Operators

The logical operators are used to perform bit-level operations on operands. These operators allow the evaluation and manipulation of specific bits within the integer.

Operator	Name	Example	Explanation
and	And	$a and $b	Return TRUE if both $a and $b are true
Or	Or	$a or $b	Return TRUE if either $a or $b is true
xor	Xor	$a xor $b	Return TRUE if either $ or $b is true but not both
!	Not	! $a	Return TRUE if $a is not true
&&	And	$a && $b	Return TRUE if either $a and $b are true
||	Or	$a || $b	Return TRUE if either $a or $b is true

String Operators

The string operators are used to perform the operation on strings. There are two string operators in PHP, which are given below:

Operator	Name	Example	Explanation
.	Concatenation	$a . $b	Concatenate both $a and $b
.=	Concatenation and Assignment	$a .= $b	First concatenate $a and $b, then assign the concatenated string to $a, e.g. $a = $a . $b

Array Operators

The array operators are used in case of array. Basically, these operators are used to compare the values of arrays.

Operator	Name	Example	Explanation
+	Union	$a + $y	Union of $a and $b
==	Equality	$a == $b	Return TRUE if $a and $b have same key/value pair
!=	Inequality	$a != $b	Return TRUE if $a is not equal to $b
===	Identity	$a === $b	Return TRUE if $a and $b have same key/value pair of same type in same order
!==	Non-Identity	$a !== $b	Return TRUE if $a is not identical to $b
<>	Inequality	$a <> $b	Return TRUE if $a is not equal to $b

Chapter-2

Decision Making

Q1. What do you understand by decision making statements?

PHP allows us to perform actions based on some type of conditions that may be logical or comparative. Based on the result of these conditions i.e., either TRUE or FALSE, an action would be performed as asked by the user. It's just like a two- way path. If you want something then go this way or else turn that way. To use this feature, PHP provides us with four conditional statements:

if statement
if…else statement
if…elseif…else statement
switch statement

Q2. Explain if and if…else statements with example.

1. **if Statement**: This statement allows us to set a condition. On being TRUE, the following block of code enclosed within the if clause will be executed.
   Syntax :
   if (condition){
   // if TRUE then execute this code
   }
   Example:

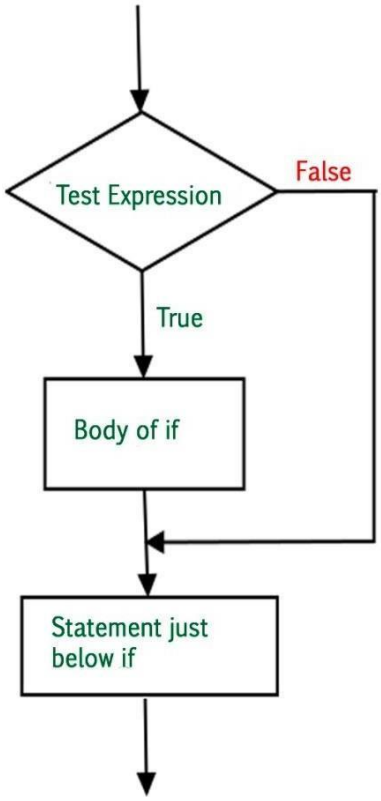
```
<?php

$x = 12;

if ($x > 0) {

    echo "The number is positive";

}
```

```
?>
```

Output:

```
The number is positive
```

Flowchart:



2. **if…else Statement**: We understood that if a condition will hold i.e., TRUE, then the block of code within if will be executed. But what if the condition is not TRUE and we want to perform an action? This is where else comes into play. If a condition is TRUE then if block gets executed, otherwise else block gets executed.

   Syntax:

```
if (condition) {
   // if TRUE then execute this code
}
else{
   // if FALSE then execute this code
}
```

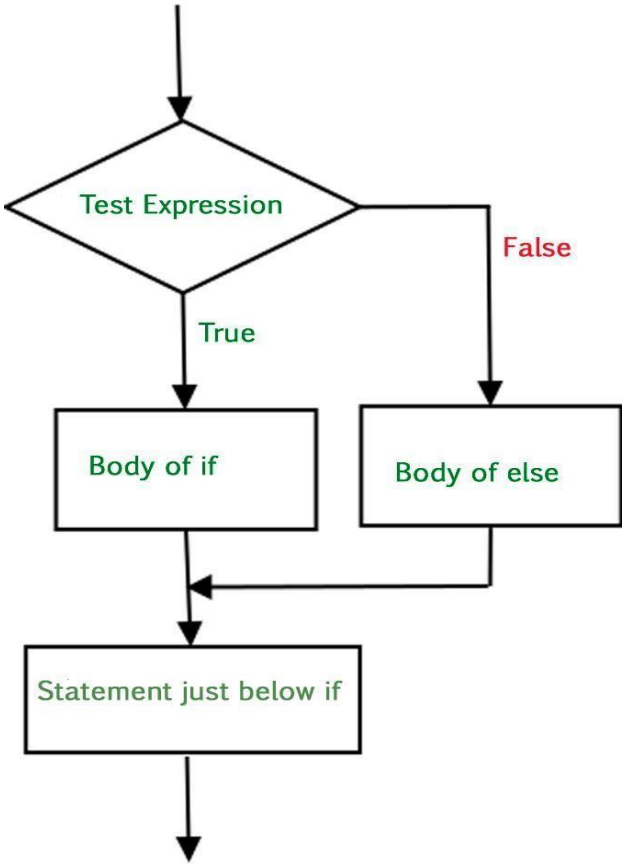
   Example:

```
<?php

$x = -12;

if ($x > 0) {

    echo "The number is positive";

}

else{

    echo "The number is negative";

}

?>
```

Output:

```
The number is negative
```

Flowchart:



Q3. What do you understand by switch statement? Give an example.

The "switch" performs in various cases i.e., it has various cases to which it matches the condition and appropriately executes a particular case block. It first evaluates an expression and then compares with the values of each case. If a case matches then the same case is executed. To use switch, we need to get familiar with two different keywords namely, break and default.
The break statement is used to stop the automatic control flow into the next cases and exit from the switch case.
The default statement contains the code that would execute if none of the cases match.

Syntax:

```
switch(n) {
   case statement1:
      code to be executed if n==statement1;
      break;
   case statement2:
      code to be executed if n==statement2;
      break;
   case statement3:
      code to be executed if n==statement3;
      break;
   case statement4:
      code to be executed if n==statement4;
      break;
   .....
   default:
      code to be executed if n != any case;
```

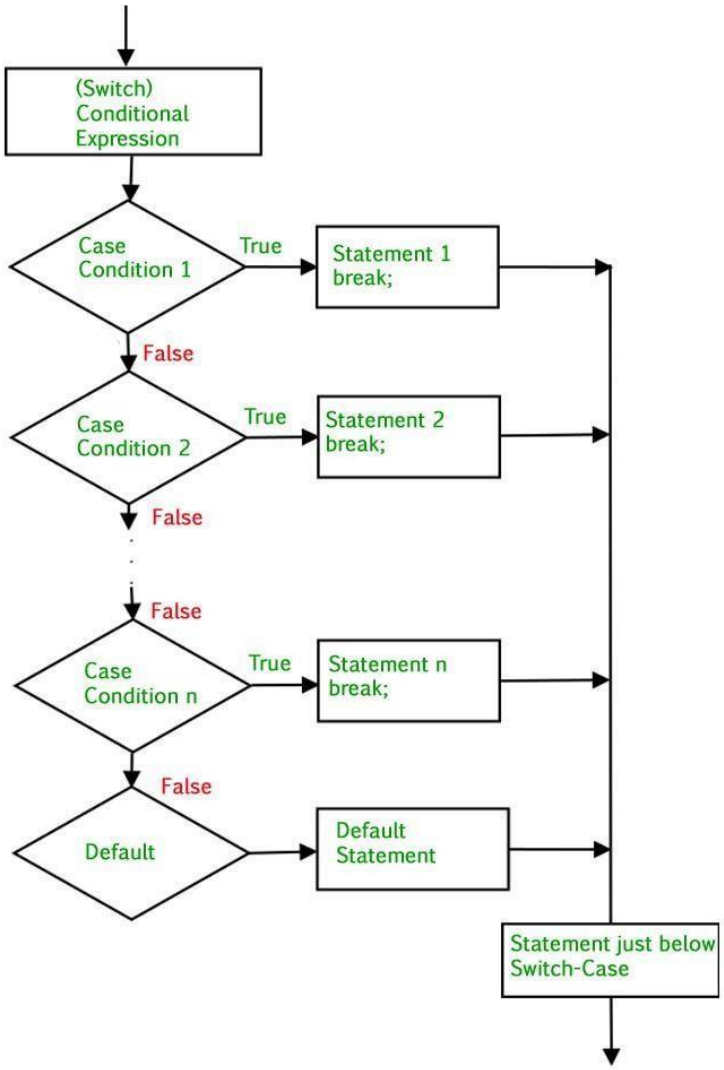
Example:

```
<?php
$n = "February";

switch($n) {
   case "January":
      echo "Its January";
      break;
   case "February":
      echo "Its February";
      break;
   case "March":
      echo "Its March";
      break;
   case "April":
      echo "Its April";
      break;
   case "May":
      echo "Its May";
```

```
        break;
    case "June":
        echo "Its June";
        break;
    case "July":
        echo "Its July";
        break;
    case "August":
        echo "Its August";
        break;
    case "September":
        echo "Its September";
        break;
    case "October":
        echo "Its October";
        break;
    case "November":
        echo "Its November";
        break;
    case "December":
        echo "Its December";
        break;
    default:
        echo "Doesn't exist";
}
?>
```

Output:
Its February

Flowchart



Q4. Describe an Array in PHP?

Arrays in PHP is a type of data structure that allows us to store multiple elements of similar data type under a single variable thereby saving us the effort of creating a different variable for every data. The arrays are helpful to create a list of elements of similar types, which can be accessed using their index or key. Suppose we want to store five names and print them accordingly. This can be easily done by the use of five different string variables. But if instead of five, the number rises to a hundred, then it would be really difficult for the user or developer to create so many different variables. Here array comes into play and helps us to store every element within a single variable and also allows easy access using an index or a key.
An array is created using an array() function in PHP.

There are basically three types of arrays in PHP:

Indexed or Numeric Arrays: An array with a numeric index where values are stored linearly.
Associative Arrays: An array with a string index where instead of linear storage, each value can be assigned a specific key.
Multidimensional Arrays: An array which contains single or multiple array within it and can be accessed via multiple indices.

Q5. What is Indexed Array? Give an example.

Indexed or Numeric Arrays

These type of arrays can be used to store any type of elements, but an index is always a number. By default, the index starts at zero. These arrays can be created in two different ways as shown in the following example:

```
<?php

// One way to create an indexed array
$name_one = array("Zack", "Anthony", "Ram", "Salim", "Raghav");

// Accessing the elements directly
echo "Accessing the 1st array elements directly:\n";
echo $name_one[2], "\n";
echo $name_one[0], "\n";
echo $name_one[4], "\n";

// Second way to create an indexed array
$name_two[0] = "ZACK";
$name_two[1] = "ANTHONY";
$name_two[2] = "RAM";
$name_two[3] = "SALIM";
$name_two[4] = "RAGHAV";

// Accessing the elements directly
echo "Accessing the 2nd array elements directly:\n";
echo $name_two[2], "\n";
echo $name_two[0], "\n";
echo $name_two[4], "\n";

?>
```

Output:

```
Accessing the 1st array elements directly:
Ram
Zack
Raghav
Accessing the 2nd array elements directly:
RAM
ZACK
RAGHAV
```

Q6. Explain Associative Array.

These types of arrays are similar to the indexed arrays but instead of linear storage, every value can be assigned with a user-defined key of string type.

Example:

```
<?php

// Creating an Associative Array
$name_one = [
   "Zack" => "Zara",
   "Anthony" => "Any",
   "Ram" => "Rani",
   "Salim" => "Sara",
   "Raghav" => "Ravina",
];

// Looping through an array using foreach
echo "Looping using foreach: \n";
foreach ($name_one as $val => $val_value) {
   echo "Husband is " . $val . " and Wife is " . $val_value . "\n";
}

// Looping through an array using for
echo "\nLooping using for: \n";
$keys = array_keys($name_one);
$round = count($name_one);

for ($i = 0; $i < $round; ++$i) {
   echo $keys[$i] . " " . $name_one[$keys[$i]] . "\n";
}
?>
```

Output:

```
Accessing the elements directly:
zara
sara
any
Rani
Ravina
```

Q7. Describe about Multi-Dimensional Array with example.

Multi-dimensional arrays are such arrays that store another array at each index instead of a single element. In other words, we can define multi-dimensional arrays as an array of arrays. As the name suggests, every element in this array can be an array and they can also hold other sub-arrays within. Arrays or sub-arrays in multidimensional arrays can be accessed using multiple dimensions.

Example:

```
<?php
// Defining a multidimensional array
$favorites = array(
   "Dave Punk" => array(
      "mob" => "5689741523",
      "email" => "davepunk@gmail.com",
```

```
    ),
    "Dave Punk" => array(
        "mob" => "2584369721",
        "email" => "montysmith@gmail.com",
    ),
    "John Flinch" => array(
        "mob" => "9875147536",
        "email" => "johnflinch@gmail.com",
    )
);

// Using for and foreach in nested form
$keys = array_keys($favorites);
for($i = 0; $i < count($favorites); $i++) {
    echo $keys[$i] . "\n";
    foreach($favorites[$keys[$i]] as $key => $value) {
        echo $key . " : " . $value . "\n";
    }
    echo "\n";
}

?>
```

Output:

```
Dave Punk email-id is: davepunk@gmail.com
John Flinch mobile number is: 9875147536
```

□ □ □

Chapter-3

Strings

Q1. What is string in php and format specifiers of string?

Ans. In PHP, a string is a sequence of characters, where a character is a single unit of text. Strings can be created using single quotes (''), double quotes (" "), or the `heredoc` syntax. Here are examples of each:

```
<?php
// Single-quoted string
$singleQuotedString = 'Hello, World!';

// Double-quoted string
$doubleQuotedString = "Hello, World!";

// Heredoc syntax
$heredocString = <<<EOT
Hello, World!
This is a heredoc string.
EOT;
?>
```

In PHP, strings support various operations and functions for manipulation, comparison, and formatting. Format specifiers, often used in the context of sprintf() and printf() functions, are placeholders that indicate how a value should be formatted within a string. Here are some common format specifiers for strings:

1. `%s`: String
- Example: `printf("Hello, %s!", "World");`

2. `%d` or `%i`: Signed decimal number
- Example: `printf("The answer is %d.", 42);`

3. `%f`: Floating-point number
- Example: `printf("Pi is approximately %f.", 3.14159);`

4. `%c`: Single character
- Example: `printf("The first letter of the alphabet is %c.", 'A');`
5. `%b`: Binary representation of a number
- Example: `printf("%b", 10); // Prints binary representation of 10`

6. `%x` or `%X`: Hexadecimal representation of a number
- Example: `printf("The hexadecimal representation of 255 is %x.", 255);`

These format specifiers are commonly used with functions like `sprintf()` and `printf()` to format strings. For example:

```
<?php
$name = "John";
$age = 25;

// Using sprintf to format a string
$formattedString = sprintf("My name is %s and I am %d years old.", $name, $age);

// Printing the formatted string
echo $formattedString;
```

This will output: "My name is John and I am 25 years old."

Q2. Explain different string Library functions with examples.

Ans. PHP provides a variety of string functions that allow you to manipulate and work with strings. Here are some commonly used string functions along with examples:

1. strlen($string):Returns the length of the given string.

```
<?php
  $str = "Hello, World!";
  echo strlen($str); // Outputs 13
  ?>
```

2. strrev($string):Reverses the given string.

```
<?php
  $str = "Hello, World!";
echo strrev($str);  // Outputs "!dlroW ,olleH"
?>
```

3. strtolower($string):Converts all characters in a string to lowercase.

```
<?php
  $str = "Hello, World!";
  echo strtolower($str); // Outputs "hello, world!"
  ?>
```

4. strtoupper($string): Converts all characters in a string to uppercase.

```
<?php
  $str = "Hello, World!";
  echo strtoupper($str); // Outputs "HELLO, WORLD!"
  ?>
```

5. substr($string, $start, $length):Returns a portion of the string.

```
<?php
  $str = "Hello, World!";
  echo substr($str, 0, 5); // Outputs "Hello"
  ?>
```

6. strpos($haystack, $needle): Finds the position of the first occurrence of a substring.

```
<?php
  $str = "Hello, World!";
  echo strpos($str, "World"); // Outputs 7
  ?>
```

7. str_replace($search, $replace, $subject): Replaces all occurrences of a substring with another substring.

```
<?php
  $str = "Hello, World!";
  echo str_replace("World", "Universe", $str); // Outputs "Hello, Universe!"
  ?>
```

8. trim($string): Removes whitespace (or other characters) from the beginning and end of a string.

```
<?php
  $str = "   Hello, World!   ";
  echo trim($str); // Outputs "Hello, World!"
  ?>
```

9. explode($delimiter, $string):Splits a string into an array based on a delimiter.

```
<?php
  $str = "apple,orange,banana";
  $fruits = explode(",", $str);
  print_r($fruits); // Outputs Array ( [0] => apple [1] => orange [2] => banana )
  ?>
```

10. implode($glue, $pieces):Joins array elements with a string.

```
  <?php
   $fruits = array("apple", "orange", "banana");
   $str = implode(", ", $fruits);
   echo $str; // Outputs "apple, orange, banana"
   ?>
```

Q3. What do you understand by regular expressions in php.

Ans. Regular expressions, often referred to as regex or regexp, are powerful sequences of characters that define a search pattern. In PHP, regular expressions are used for pattern matching within strings. They enable you to search, match, and manipulate text based on specific criteria. Regular expressions are widely used in tasks such as data validation, searching and replacing, and text processing.

In PHP, regular expressions are implemented through a set of functions, mainly provided by the PCRE (Perl Compatible Regular Expressions) library. Some of the key functions for working with regular expressions in PHP include `preg_match()`, `preg_match_all()`, `preg_replace()`, and others.

Here are some key concepts related to regular expressions in PHP:

1. Pattern Matching:

(i) preg_match() : Checks if a pattern exists in a given string.

```
<?php
$str = "Hello, World!";
    if (preg_match("/World/", $str)) {
       echo "Pattern found!";
    } else {
       echo "Pattern not found.";
    }
    ?>
```

2. Global Pattern Matching:

(ii) preg_match_all(): Finds all occurrences of a pattern in a string.

```
<?php
   $str = "apple, orange, banana";
   preg_match_all("/\w+/", $str, $matches);
   print_r($matches);
   ?>
```

3. Pattern Replacement:

(iii)preg_replace(): Replaces a pattern with a specified string.

```
<?php
   $str = "Hello, World!";
   $newStr = preg_replace("/World/", "Universe", $str);
   echo $newStr; // Outputs "Hello, Universe!"
   ?>
```

4. Regular Expression Patterns:

Regular expressions use a specific syntax to define patterns. For example, `\w` represents any word character, and `+` indicates one or more occurrences.

```
   <?php
   $pattern = "/\d{2,4}/"; // Matches 2 to 4 digits
   ?>
```

5. Modifiers:

Regular expressions in PHP can include modifiers that affect how the pattern matching is performed. For example, the `i` modifier makes the pattern case-insensitive.

```
<?php
   $pattern = "/world/i"; // Case-insensitive match
   ?>
```

- **Regular expressions can be complex, but they provide a flexible and efficient way to work with text data.**

Q4. Describe function of regular expressions in php.

Ans. Regular expressions in PHP serve as a powerful tool for working with text data. They provide a flexible and concise way to define patterns and perform operations such as searching, matching, and replacing within strings. The primary functions of regular expressions in PHP include:

1. Pattern Matching:

You can use regular expressions to check if a specific pattern exists in a string. This is commonly done with the `preg_match()` function.

```
<?php
   $str = "Hello, World!";
   if (preg_match("/World/", $str)) {
      echo "Pattern found!";
   } else {
      echo "Pattern not found.";
   }
   ?>
```

2. Global Pattern Matching:

Regular expressions can find all occurrences of a pattern in a string using the `preg_match_all()` function.

```
<?php
 $str = "apple, orange, banana";
   preg_match_all("/\w+/", $str, $matches);
   print_r($matches);
   ?>
```

3. Pattern Replacement:

With `preg_replace()`, you can replace occurrences of a pattern with a specified string.

```
<?php
   $str = "Hello, World!";
   $newStr = preg_replace("/World/", "Universe", $str);
   echo $newStr; // Outputs "Hello, Universe!"
   ?>
```

4. Validation:

Regular expressions are commonly used for input validation. For instance, you can use them to check if an email address or a phone number is in a valid format.

```
   <?php
   $email = "user@example.com";
   if (preg_match("/^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/", $email)) {
      echo "Valid email address!";
   } else {
      echo "Invalid email address.";
   }
   ?>
```

5. Text Extraction:

- Regular expressions are useful for extracting specific information from a string, especially when the data follows a consistent pattern.

```
   <?php
   $text = "Date: 2022-01-01";
   preg_match("/Date: (\d{4}-\d{2}-\d{2})/", $text, $matches);
   echo "Extracted date: " . $matches[1];
   ?>
```

6. Tokenization:

Regular expressions can assist in breaking down a string into meaningful tokens or segments. This is particularly useful in lexical analysis or parsing.

```
   <?php
   $sentence = "The quick brown fox";
   $words = preg_split("/\s+/", $sentence);
   print_r($words);
   ?>
```

7. Case-Insensitive Matching, Multi-line Matching, etc.:

Regular expressions support various modifiers that allow you to control aspects such as case sensitivity, multi-line matching, and more.

```
<?php
$pattern = "/world/i"; // Case-insensitive match
?>
```

- **Regular expressions provide a versatile and efficient way to work with textual data, enabling developers to handle complex string manipulation tasks with relative ease. While they may initially seem complex, mastering regular expressions can significantly enhance your ability to work with strings in PHP.**

.

Chapter-4

Functions

Q1. What is function in php. Give example.
In PHP, a function is a block of reusable code that performs a specific task. Functions help organize code, make it more modular, and allow you to reuse logic throughout your program.

Here's a simple example of a PHP function:

```
php
<?php
// Define a function named "greet"
function greet($name) {
    echo "Hello, $name!";
}

// Call the function with an argument
greet("John");
?>
```

In this example, the greet function takes a parameter $name and echoes a greeting using that parameter. When you call the function with greet("John");, it outputs "Hello, John!".

Q2. Explain different types of functions with examples.
In PHP, functions can be categorized into several types based on their behavior and purpose. Here are some common types:

1. *Built-in Functions:*
   - These are functions that come pre-defined in PHP.
   - Example: strlen(), strtolower(), array_merge()

php
<?php
$length = strlen("Hello, world!"); // Returns the length of the string
?>

2. *User-Defined Functions:*
  - Functions created by the user to perform a specific task.
  - Example:

php
<?php
// User-defined function
function addNumbers($a, $b) {
  return $a + $b;
}

// Call the function
$result = addNumbers(5, 10); // $result now holds the value 15
?>

3. *Anonymous Functions (Closures):*
  - Functions without a name; defined on the fly and assigned to a variable.
  - Example:

php
<?php
$multiply = function($x, $y) {
  return $x * $y;
};

$result = $multiply(3, 4); // $result now holds the value 12
?>

Q3. What Is Recursive Function?

- Functions that call themselves, often used for tasks that can be broken down into simpler, similar sub-tasks.
- Example:

```
php
<?php
// Recursive function to calculate factorial
function factorial($n) {
    if ($n <= 1) {
        return 1;
    } else {
        return $n * factorial($n - 1);
    }
}

$result = factorial(5); // $result now holds the value 120
?>
```

These are just a few types of functions in PHP. Each type serves a specific purpose and contributes to making your code more organized and efficient.

Chapter-5

Exception Handling in PHP

Q.1. What is an Exception?

OR

Explain how Exceptions are handled using try-catch Block?

OR

What is a Finally Block?

Ans.: The term *exception* is shorthand for the phrase "exceptional event."

Definition : An *exception* is an event, which occurs during the execution of a program that disrupts the normal flow of the program's instructions.

When an error occurs within a method, the method creates an object and hands it off to the runtime system. The object, called an *exception object*, contains information about the error, including its type and the state of the program when the error occurred. Creating an exception object and handing it to the runtime system is called *throwing an exception*. After a method throws an exception, the runtime system attempts to find something to handle it.

Some of the predefined exception classes are :

ArithmeticException,

ArrayIndexOutOfBoundException,

IOException etc.

The try Block : The first step in constructing an exception handler is to enclose the code that might throw an exception within a try block. In general, a try block looks like the following.

```
try {
   code
}
```

catch and finally blocks . . .

The segment in the example labeled *code* contains one or more legal lines of code that could throw an exception.

A catch Block : A catch block defined by the keyword catch ―catches‖ the exception ―thrown‖ by the try block and handles it appropriately.The catch block is added immediately after the try block.

The general form is :

.....

.....

```
try
{
        statement;
}
catch(Exception type e)
{
        statement;
}
.....
.....
```

Multiple catch Statements : It is possible to have more than one catch statements in the catch block.

e.g.

.....

.....

```
try
{
        statement;
}
catch(Exception-Type-1 e)
{
```

```
        statement;
    }
    catch(Exception-Type-2 e)
    {
        statement;
    }
    .
    .
    .
    .
    .
    catch(Exception –Type-N e)
    {
        statement;
    }
    ……..
    ……..
```

Using Finally Statement : PHP supports another statement known as finally statement that can be used to handle an exception that is not caught by any of the previous catch statements. Finally block can be used to handle any exception generated within a try block. It may be immediately after the try block or after the last catch block.

When a finally block is defined, this is guaranteed to execute, regardless of whether or not in exception is thrown.

Throwing our own Exceptions : There may be times when we would like to throw our own exceptions. We can do this by using the keyword throw as follows :

throw new Throwable_subclass;

e.g. throw new Arithmetic Exception();

Now, let's see an example in PHP:

```
<?php
function divide($numerator, $denominator)
{
   if ($denominator == 0)
{
      throw new Exception("Cannot divide by zero!");
 }
```

```
    return $numerator / $denominator;
}

try
 {
    $result = divide(10, 0);
    echo "Result: $result";
}
catch (Exception $e)
{
    echo "Error: " . $e->getMessage();
}
finally
{
       echo " This will always be executed.";
}
?>
```

Another example is

Example:

```
  <?php

  function checkAge($age)
{
      if ($age < 18)
  {
          throw new Exception("Age must be 18 or older.");
  }
      echo "You are eligible.";
 }
  try
 {
      checkAge(15);
  }
 catch (Exception $e)
 {
      echo "Error: " . $e->getMessage();
 }
 ?>
```

The `finally` block is used in conjunction with `try` and `catch` blocks to specify a block of code that will be executed regardless of whether an exception is thrown or not.

- This block is optional and provides a way to perform cleanup operations or ensure that certain code is always executed.

Example:

```
<?php

function divide($numerator, $denominator)
```

```
{
try
{
        if ($denominator == 0)
 {
           throw new Exception("Cannot divide by zero!");
  }

        $result = $numerator / $denominator;
        echo "Result: $result";
 }
catch (Exception $e)
{
        echo "Error: " . $e->getMessage();
 }
finally
 {
        echo " This will always be executed.";
  }
}
?>
```

Both `throw` and `finally` play essential roles in structured exception handling, allowing developers to handle errors gracefully and manage resources effectively.

□ □ □

Chapter-6

File Handling

Q1. What do you understand by File handling? Explain how to open a file and close a file in PHP.

PHP File System allows us to create file, read file line by line, read file character by character, write file, append file, delete file and close file.

PHP Open File - fopen()

The PHP fopen() function is used to open a file.

Syntax

resource fopen (string $filename , string $mode [, bool $use_include_path = false [, resource $context]])

Example

```
<?php
$handle = fopen("c:\\folder\\file.txt", "r");
?>
```

PHP Close File - fclose()

The PHP fclose() function is used to close an open file pointer.

Syntax

ool fclose (resource $handle)

Example

```
<?php
fclose($handle);
?>
```

Q2. Write a code to write in a file and read a file.

The PHP fread() function is used to read the content of the file. It accepts two arguments: resource and file size.

Syntax

string fread (resource $handle , int $length)

Example

```
<?php
$filename = "c:\\myfile.txt";
$handle = fopen($filename, "r");//open file in read mode

$contents = fread($handle, filesize($filename));//read file

echo $contents;//printing data of file
fclose($handle);//close file
?>
```

Output

hello php file

PHP Write File - fwrite()

The PHP fwrite() function is used to write content of the string into file.

Syntax

int fwrite (resource $handle , string $string [, int $length])

Example

```
<?php
$fp = fopen('data.txt', 'w');//open file in write mode
fwrite($fp, 'hello ');
fwrite($fp, 'php file');
fclose($fp);
```

```
echo "File written successfully";
?>
```

Output

File written successfully

Q4. How to delete a file in PHP. Give example.

PHP Delete File - unlink()
The PHP unlink() function is used to delete file.

Syntax

bool unlink (string $filename [, resource $context])

Example

```
<?php
unlink('data.txt');

echo "File deleted successfully";
?>
```

□ □ □

Chapter-7

Database Handling in PHP

Q1. How to establish a connection between database and PHP frontend?

The collection of related data is called a database. XAMPP stands for cross-platform, Apache, MySQL, PHP, and Perl. It is among the simple light-weight local servers for website development.

Requirements: XAMPP web server procedure:

Start XAMPP server by starting Apache and MySQL.
Write PHP script for connecting to XAMPP.
Run it in the local browser.
Database is successfully created which is based on the PHP code.
In PHP, we can connect to the database using XAMPP web server by using the following path.

"localhost/phpmyadmin"

PHP code to create a database:

PHP

```
<?php

// Server name must be localhost
$servername = "localhost";

// In my case, user name will be root
$username = "root";

// Password is empty
$password = "";

// Creating a connection
$conn = new mysqli($servername,
        $username, $password);

// Check connection
if ($conn->connect_error) {
   die("Connection failure: "
      . $conn->connect_error);
}
```

```
// Creating a database named geekdata
$sql = "CREATE DATABASE geekdata";
if ($conn->query($sql) === TRUE) {
   echo "Database with name geekdata";
} else {
   echo "Error: " . $conn->error;
}

// Closing connection
$conn->close();
?>
```

Save the file as "data.php" in htdocs folder under XAMPP folder.
Finally the database is created and connected to PHP.

Q2. How to create a table in database using PHP code?

In relational databases, and flat file databases, a table is a set of data elements using a model of vertical columns and horizontal rows, the cell being the unit where a row and column intersect. A table has a specified number of columns, but can have any number of rows.

The CREATE TABLE statement is used to create a table in MySQL.
Syntax :

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "newDB";

// checking connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}

// sql code to create table
$sql = "CREATE TABLE employees(
      id INT(2)  PRIMARY KEY,
      firstname VARCHAR(30) NOT NULL,
      lastname VARCHAR(30) NOT NULL,
      email VARCHAR(50)
      )";

if ($conn->query($sql) === TRUE) {
   echo "Table employees created successfully";
} else {
   echo "Error creating table: " . $conn->error;
}

$conn->close();
?>
```

Q3. Write a code to insert data into table.

INSERT INTO statement is used to insert new rows in a database table. Let's see the syntax how to insert into table, considering database already exists.

SYNTAX :

INSERT INTO TABLE_NAME (column1, column2, column3, ... columnN)
VALUES (value1, value2, value3, ...valueN);
Here, column1, column2, column3, …columnN are the names of the columns in the table into which you want to insert the data.

```
<?php
$link = mysqli_connect("localhost", "root", "", "newdb");

if ($link == = false) {
   die("ERROR: Could not connect. ".mysqli_connect_error());
}

$sql = "INSERT INTO mytable (first_name, last_name, age)
        VALUES('ram', 'singh', '25') ";
   if (mysqli_query($link, $sql))
{
   echo "Records inserted successfully.";
}
else
{
   echo "ERROR: Could not able to execute $sql. "
      .mysqli_error($link);
}

mysqli_close($link);
? >
```

Q4. How to delete a row in table using PHP?
The DELETE query is used to delete records from a database table.
It is generally used along with the "Select" statement to delete only those records that satisfy a specific condition.

```
<?php
$link = mysqli_connect("localhost", "root", "", "Mydb");

if($link === false){
   die("ERROR: Could not connect. " . mysqli_connect_error());
}

$sql = "DELETE FROM Data WHERE ID=201";
if(mysqli_query($link, $sql)){
   echo "Record was deleted successfully.";
}
else{
   echo "ERROR: Could not able to execute $sql. "
                        . mysqli_error($link);
```

```
}
mysqli_close($link);
?>
```

Q5. Write a code to update data in table using PHP.

The MySQL UPDATE query is used to update existing records in a table in a MySQL database.

It can be used to update one or more field at the same time.
It can be used to specify any condition using the WHERE clause.

Syntax :

The basic syntax of the Update Query is –

```
<?php
$link = mysqli_connect("localhost", "root", "", "Mydb");

if($link === false){
   die("ERROR: Could not connect. "
            . mysqli_connect_error());
}

$sql = "UPDATE data SET Age='28' WHERE id=201";
if(mysqli_query($link, $sql)){
   echo "Record was updated successfully.";
} else {
   echo "ERROR: Could not able to execute $sql. "
                     . mysqli_error($link);
}
mysqli_close($link);
?>
```

This question paper contains 2 printed pages. Roll No. .....

306/336-B

B.C.A. (Part-III) EXAMINATION - 2022

(Faculty of Science)

(Three-Year Scheme of 10+2+3 Pattern)

ADVANCE TECHNOLOGY OF PROGRAMMING THROUGH PHP

Time Allowed : Three Hours Maximum Marks : 100

Answer of all the questions (short answer as well as descriptive) are to be given in the main answer-book only. Answers of short answer type questions must be given in sequential order. Similarly all the parts of one question of descriptive part should be answered at one place in the answer-book. One complete question should not be answered at different places in the answer-book.

Write your roll number on question paper before start writing answers of questions.

*Question paper consists of **three** Parts.*

All three** parts are **compulsory.

PART-I : *(Very Short Answer) consists 10 questions of 2 marks each. Maximum limit for each question is upto 40 words.*

PART-II : *(Short Answer) consists 5 questions of 4 marks each. Maximum limit for each question is upto 80 words.*

PART-III : *(Long Answer) consists 5 questions of 12 marks each with internal choice.*

PART - I

1. Attempt **all** questions. **10x2=20**

   (a) What does scripting language mean in programming ?

   (b) What are the rules for naming a PHP variable ?

   (c) What is the purpose of break and continue statement ?

   (d) Write in brief about foreach( ) statement with syntax.

   (e) What is the difference between single quoted string and double quoted string ?

   (f) Write the use of str_split( ) function with syntax.

   (g) What do you mean by dynamic website ?

   (h) What do you mean by exception handling ?

   (i) Which function you use to read a file ? Write its syntax also.

   (j) How can you connect the webpage with database ?

PART - II

2. Differentiate between client-side and server-side scripting. 4

3. What is the meaning of the associative array ? Explain with a suitable example. 4

4. What are the uses of explode( ) and implode( ) functions ? Explain with an example. 4

5. Write the difference between Get( ) and Post( ) function. 4

6. Using an appropriate example, demonstrate how to use fread( ), fgets( ), and fgetc( ) in a file. 4

PART - III

7. Describe the various types of operators available in PHP. **12**

OR

Write short note on : **(6+6)=12**

(a) Features of PHP

(b) Data types in PHP

8. What is an array ? Describe the types of array available in PHP. **(2+10)=12**

OR

Write a program in PHP to find the given number is Prime or not. **12**

9. Differentiate between Call by Value and Call by Reference function in PHP. **(6+6)=12**

OR

Explain any six string handling library function with example. **(6x2)=12**

10. What is session ? Why do we use it ? Where does PHP store session data ? Through an example describe how to maintain session on website. **(1+2+2+7)=12**

OR

What is exception ? Why do we handle exception ? Explain about it through an example. **(2+2+8)=12**

11. How can you perform open, read, write, close operations on file in PHP ? Explain with suitable example. **(4+8)=12**

OR

Write a program in PHP to store and access the data using database. **12**

- o O o -