

**THIRD YEAR B. Sc.  
COMPUTER SCIENCE  
SEMESTER-V**

**NEW SYLLABUS  
CBCS PATTERN**

# **WEB TECHNOLOGIES-I**

**Dr. A. B. NIMBALKAR**



SPPU New Syllabus

*A Book Of*  
**WEB TECHNOLOGIES - I**

**For T.Y.B.Sc. Computer Science : Semester – V**  
**[Course Code CS 353 : Credits - 2]**

**CBCS Pattern**

**As Per New Syllabus, Effective from June 2021**

**Dr. A. B. Nimbalkar**

*M.C.S., M.Phil. D.C.L, Ph.D. (Comp. Sci.)*  
Asst. Professor, Annasaheb Magar Mahavidyalaya  
Hadapsar, Pune

**Price ₹ 360.00**



**N5863**

---

**WEB TECHNOLOGIES - I****ISBN 978-93-5451-185-1****Second Edition : August 2022****© : Author**

The text of this publication, or any part thereof, should not be reproduced or transmitted in any form or stored in any computer storage system or device for distribution including photocopy, recording, taping or information retrieval system or reproduced on any disc, tape, perforated media or other information storage device etc., without the written permission of Author with whom the rights are reserved. Breach of this condition is liable for legal action.

Every effort has been made to avoid errors or omissions in this publication. In spite of this, errors may have crept in. Any mistake, error or discrepancy so noted and shall be brought to our notice shall be taken care of in the next edition. It is notified that neither the publisher nor the author or seller shall be responsible for any damage or loss of action to any one, of any kind, in any manner, there from. The reader must cross check all the facts and contents with original Government notification or publications.

**Published By :****NIRALI PRAKASHAN**

Abhyudaya Pragati, 1312, Shivaji Nagar,

Off J.M. Road, Pune – 411005

Tel - (020) 25512336/37/39

Email : niralipune@pragationline.com

**Polyplate****Printed By :****YOGIRAJ PRINTERS AND BINDERS**

Survey No. 10/1A, Ghule Industrial Estate

Nanded Gaon Road

Nanded, Pune - 411041

---

**DISTRIBUTION CENTRES****PUNE****Nirali Prakashan****(For orders outside Pune)**S. No. 28/27, Dhayari Narhe Road, Near Asian College  
Pune 411041, Maharashtra

Tel : (020) 24690204; Mobile : 9657703143

Email : bookorder@pragationline.com

**Nirali Prakashan****(For orders within Pune)**119, Budhwar Peth, Jogeshwari Mandir Lane  
Pune 411002, Maharashtra

Tel : (020) 2445 2044; Mobile : 9657703145

Email : niralilocal@pragationline.com

**MUMBAI****Nirali Prakashan**Rasdhara Co-op. Hsg. Society Ltd., 'D' Wing Ground Floor, 385 S.V.P. Road  
Girgaum, Mumbai 400004, Maharashtra

Mobile : 7045821020, Tel : (022) 2385 6339 / 2386 9976

Email : niralimumbai@pragationline.com

---

**DISTRIBUTION BRANCHES****DELHI****Nirali Prakashan**Room No. 2 Ground Floor  
4575/15 Omkar Tower, Agarwal Road  
Darya Ganj, New Delhi 110002  
Mobile : 9555778814/9818561840  
Email : delhi@niralibooks.com**BENGALURU****Nirali Prakashan**Maitri Ground Floor, Jaya Apartments,  
No. 99, 6<sup>th</sup> Cross, 6<sup>th</sup> Main,  
Malleswaram, Bengaluru 560003  
Karnataka; Mob : 9686821074  
Email : bengaluru@niralibooks.com**NAGPUR****Nirali Prakashan**Above Maratha Mandir, Shop No. 3,  
First Floor, Rani Jhansi Square,  
Sitabuldi Nagpur 440012 (MAH)  
Tel : (0712) 254 7129  
Email : nagpur@niralibooks.com**KOLHAPUR****Nirali Prakashan**New Mahadvar Road, Kedar Plaza,  
1<sup>st</sup> Floor Opp. IDBI Bank  
Kolhapur 416 012 Maharashtra  
Mob : 9850046155  
Email : kolhapur@niralibooks.com**JALGAON****Nirali Prakashan**34, V. V. Golani Market, Navi Peth,  
Jalgaon 425001, Maharashtra  
Tel : (0257) 222 0395  
Mob : 94234 91860  
Email : jalgaon@niralibooks.com**SOLAPUR****Nirali Prakashan**R-158/2, Avanti Nagar, Near Golden  
Gate, Pune Naka Chowk  
Solapur 413001, Maharashtra  
Mobile 9890918687  
Email : solapur@niralibooks.com

---

[marketing@pragationline.com](mailto:marketing@pragationline.com) | [www.pragationline.com](http://www.pragationline.com)Also find us on  [www.facebook.com/niralibooks](https://www.facebook.com/niralibooks)

## Preface ...

---

I take an opportunity to present this Text Book on "**Web Technologies - I**" to the students of Third Year B.Sc. (Computer Science) Semester-V as per the New Syllabus, June 2021.

The book has its own unique features. It brings out the subject in a very simple and lucid manner for easy and comprehensive understanding of the basic concepts. The book covers theory of Introduction to HTML, HTTP and PHP, Function and String, Arrays, Files and Database Handling and Handling Email with PHP.

A special word of thank to Shri. Dineshbhai Furia, and Mr. Jignesh Furia for showing full faith in me to write this text book. I also thank to Mr. Amar Salunkhe and Mr. Akbar Shaikh of M/s Nirali Prakashan for their excellent co-operation.

I also thank Ms. Chaitali Takle, Mr. Ravindra Walodare, Mr. Sachin Shinde, Mr. Ashok Bodke, Mr. Moshin Sayyed and Mr. Nitin Thorat.

Although every care has been taken to check mistakes and misprints, any errors, omission and suggestions from teachers and students for the improvement of this text book shall be most welcome.

## Author





# **Syllabus ...**

---

- |   |                      |
|---|----------------------|
| <b>1. Introduction to HTML, HTTP and PHP</b>  | <b>(10 Lectures)</b> |
| • Overview of HTML and Basic Tags, Creating Forms ,Tables, HTML5 Semantics  |                      |
| • CSS Basic Concept, Three ways to use CSS, Box Model, Navigation Bar   |                      |
| • Introduction to Web Server and Web Browser  |                      |
| • HTTP Basics   |                      |
| • PHP Basics: Use of PHP, Lexical Structure, Language Basics  |                      |
| <b>2. Function and String</b>   | <b>(8 Lectures)</b>  |
| • Defining and Calling a Function   |                      |
| • Default Parameters  |                      |
| • Variable Parameters, Missing Parameters   |                      |
| • Variable Function, Anonymous function   |                      |
| • Types of Strings in PHP   |                      |
| • Printing Functions  |                      |
| • Encoding and Escaping   |                      |
| • Comparing Strings   |                      |
| • Manipulating and Searching Strings  |                      |
| • Regular Expressions   |                      |
| <b>3. Arrays</b>  | <b>(6 Lectures)</b>  |
| • Indexed Vs Associative Arrays   |                      |
| • Identifying Elements of an Array  |                      |
| • Storing Data in Arrays  |                      |
| • Multidimensional Arrays   |                      |
| • Extracting Multiple Values  |                      |
| • Converting between Arrays and Variables   |                      |
| • Traversing Arrays   |                      |
| • Sorting   |                      |
| • Action on Entire Array  |                      |
| <b>4. Files and Database Handling</b>   | <b>(10 Lectures)</b> |
| • Working with Files and Directories  |                      |
| • Opening and Closing, Getting Information about File, Read/Write to File, Splitting Name and Path from File, Rename and Delete Files |                      |
| • Reading and Writing Characters in file  |                      |
| • Reading Entire File   |                      |

- Random Access to File Data
- Getting Information on File
- Ownership and Permissions
- Using PHP to Access a Database
- Relational Databases and SQL
- PEAR DB Basics
- Advanced Database Techniques

## 5. Handling Email with PHP

(2 Lectures)

- Email Background
- Internet Mail Protocol
- Structure of an Email Message
- Sending Email and Validation of Email\_id with PHP



# **Contents ...**

---

<b>1. Introduction to HTML, HTTP and PHP</b>	<b>1.1 – 1.128</b>
<b>2. Function and String</b>	<b>2.1 – 2.44</b>
<b>3. Arrays</b>	<b>3.1 – 3.36</b>
<b>4. Files and Database Handling</b>	<b>4.1 – 4.52</b>
<b>5. Handling Email with PHP</b>	<b>5.1 – 5.16</b>
<b>❖ Additional Programs</b>	<b>A.1 – A.16</b>

■ ■ ■



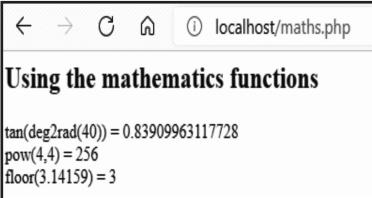
# Additional Programs

---

**Program 1:** Program for maths function.

```
<html>
  <head>
    <title>
      Using the mathematics functions
    </title>
  </head>
  <body>
    <h1>
      Using the math functions </h1>
    <?php
      echo "tan(deg2rad(40)) = ", tan(deg2rad(40)), "<br>";
      echo "pow(4,4) = ", pow(4,4), "<br>";
      echo "floor(3.14159) = ", floor(3.14159), "<br>";
    ?>
    </body>
</html>
```

**Output:**



**Program 2:** A program to check given number is odd or even input is taken from user.

```
<html>
  <body>
    <form method="post">
      Enter a number:
      <input type="number" name="number">
      <input type="submit" value="Submit">
    </form>
  </body>
</html>
<?php
  if($_POST)
  {
    $number = $_POST['number'];

```

```
//divide entered number by 2
//if the remainder is 0 then the number is even
//otherwise the number is odd

if(($number % 2) == 0)
{
    echo "$number is an Even number";
}
else
{
    echo "$number is Odd number";
}
?>
```

**Output:**

 <p>localhost/ag1/d1.php</p> <p>Enter a number: 344 <input type="text"/> Submit</p> <p>344 is an Even number</p>	 <p>localhost/ag1/d1.php</p> <p>Enter a number: 111 <input type="text"/> Submit</p> <p>111 is Odd number</p>
---	--

**Program 3:** Program to list the first 10 prime numbers.

```
<?php
$count = 0;
$num = 2;
while($count < 10 )
{
    $div_count=0;
    for($i=2; $i<$num; $i++)
    {
        if(($num%$i)==0)
        {
            $div_count++;
        }
    }
    if($div_count==0)
    {
        echo $num." ";
        $count=$count+1;
    }
    $num=$num+1;
}
?>
```

**Output:**

localhost/ag1/d1.php

2 3 5 7 11 13 17 19 23 29

**Program 4:** Program to check whether a number is prime or not.

```
<form method="post">
    Enter a Number: <input type="text" name="input"><br><br>
    <input type="submit" name="submit" value="Submit">
</form>
<?php
    if($_POST)
    {
        $input=$_POST['input'];
        for($i = 2; $i <= $input-1; $i++)
        {
            if($input % $i == 0)
            {
                $value= True;
            }
        }
        if(isset($value) && $value)
        {
            echo 'The Number '. $input . ' is not prime';
        } else
        {
            echo 'The Number '. $input . ' is prime';
        }
    }
?>
```

**Output:**

localhost/ag1/prim.php

Enter a Number: 9

Submit

The Number 9 is not prime

localhost/ag1/prim.php

Enter a Number: 11

Submit

The Number 11 is prime

**Program 5:** Program shows a form through which you can calculate factorial of number given by user.

```
<html>
    <head>
        <title>Factorial Program using loop in PHP</title>
    </head>
    <body>
        <form method="post">
            Enter the Number:<br>
            <input type="number" name="number" id="number">
            <input type="submit" name="submit" value="Submit" />
        </form>
        <?php
            if($_POST)
            {
                $fact = 1;
                //getting value from input text box 'number'
                $number = $_POST['number'];
                echo "Factorial of $number:<br><br>";
                //start loop
                for($i = 1; $i <= $number; $i++)
                {
                    $fact = $fact * $i;
                }
                echo $fact . "<br>";
            }
        ?>
    </body>
</html>
```

**Output:**

Enter the Number:  
5  
Submit

Factorial of 5:  
120

**Program 6:** Program to find Factorial of number using recursion method.

```
<html>
    <head>
        <title>Factorial Program using loop in PHP</title>
    </head>
    <body>
```

```

<form method="post">
    Enter the Number:<br>
    <input type="number" name="number" id="number">
    <input type="submit" name="submit" value="Submit" />
</form>
<?php
    function fact ($n)
    {
        if($n <= 1)
        {
            return 1;
        }
        else
        {
            return $n * fact($n - 1);
        }
    }
    $number = $_POST['number'];
    echo "<br><br>";
    echo "Factorial of $number is " .fact($number);
?
</body>
</html>

```

**Output:**

Enter the Number:  
5  
Submit

Factorial of 5 is 120

---

**Program 7:** Program to checks whether given number is Armstrong or not.

```

<?php
    $num=153;
    $total=0;
    $x=$num;
    while($x!=0)
    {
        $rem=$x%10;
        $total=$total+$rem*$rem*$rem;
        $x=$x/10;
    }

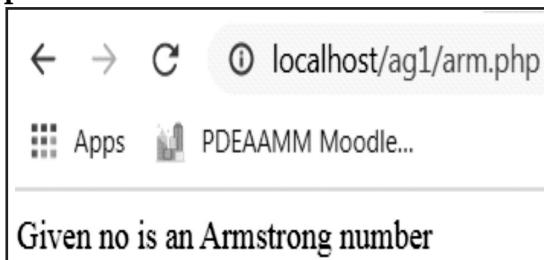
```

---

```

if($num==$total)
{
    echo "Given no is an Armstrong number";
}
else
{
    echo "Given no is Not an Armstrong number";
}
?>

```

**Output:**

**Program 8:** Program to check whether the enter number is Palindrome or not (input from the user).

```

<html>
    <head>
        <title>Factorial Program using loop in PHP</title>
    </head>
    <body>
        <form method="post">
            Enter the Number:<br>
            <input type="number" name="number" id="number">
            <input type="submit" name="submit" value="Submit" />
        </form>
        <?php
            $input = $_POST['number'];
            function palindrome($n)
            {
                $number = $n;
                $sum = 0;
                while(floor($number))
                {
                    $rem = $number % 10;
                    $sum = $sum * 10 + $rem;
                    $number = $number/10;
                }
                return $sum;
            }
            $num = palindrome($input);

```

```

if($input==$num)
{
    echo "$input is a Palindrome number";
} else
{
    echo "$input is not a Palindrome";
}
?>
</body>
</html>

```

**Output:**

Enter the Number:



12321 is a Palindrome number

**Program 9:** Program to print the first 12 numbers of a Fibonacci series.

```

<?php
/* Print fiboancci series upto 12 elements. */
$num = 12;
echo "<h3>Fibonacci series using recursive function:</h3>";
echo "\n";
/* Recursive function for fibonacci series. */
function series($num)
{
    if($num == 0)
    {
        return 0;
    }else if( $num == 1)
    {
        return 1;
    } else
    {
        return (series($num-1) + series($num-2));
    }
}
/* Call Function. */
for($i = 0; $i < $num; $i++)
{
    echo series($i);
    echo "\n";
}

```

**Output:**

Fibonacci series using recursive function:

0 1 1 2 3 5 8 13 21 34 55 89

**Program 10:** Program for reverse the string.

```
<?php
    $string = "HELLO";
    $length = strlen($string);
    for($i=($length-1) ; $i >= 0 ; $i--)
    {
        echo $string[$i];
    }
?>
```

**Program 11:** Program for swapping of two numbers without using third variable.

```
<?php
    $a=50;
    $b=100;
    //using arithmetic operation
    $a=$a+$b;
    $b=$a-$b;
    $a=$a-$b;
    echo "Value of a: $a<br>";
    echo "Value of b: $b<br>";
?>
```

**Output:**

Value of a: 100  
Value of b: 50

**Program 12:** Program to print the following triangle pattern.

```
1
2 3
4 5 6
7 8 9 10
<?php
    $k=1;
    for($i=0;$i<4;$i++)
    {
        for($j=0;$j<=$i;$j++)
```

```

{
    echo $k." ";
    $k++;
}
echo "<br>";
}
?>

```

**Output:**

1
2 3
4 5 6
7 8 9 10

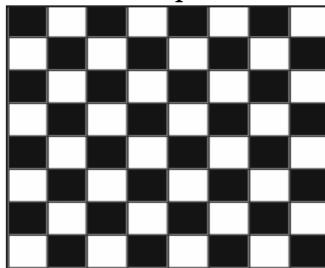
---

**Program 13:** Program to count the number of words in the string with string function.

```
<?php
$my_str = 'PHP is use for WebProgramming';
echo str_word_count($my_str);
?>
```

**Output:**

5

**Program 14:** Program using nested for loop that creates a chess board as shown below. Use table width="270px" and take 30px as cell height and width.

```

<html>
    <head>
        <title></title>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <h3>Chess Board using Nested For Loop</h3>
        <table width="270px" cellspacing="0px" cellpadding="0px"
            border="1px">
            <!-- cell 270px wide (8 columns x 60px) -->
<?php

```

---

```

for($row=1;$row<=8;$row++)
{
    echo "<tr>";
    for($col=1;$col<=8;$col++)
    {
        $total=$row+$col;
        if($total%2==0)
        {
            echo "<td height=30px width=30px bgcolor= #FFFFFF></td>";
        }
        else
        {
            echo "<td height=30px width=30px bgcolor=#000000></td>";
        }
    }
    echo "</tr>";
}
?>
</table>
</body>
</html>

```

---

**Program 15:** Program to develop a web page for the form validation.

```

<html>
    <head>
        <style>
            .error {color: #FF0000;}
        </style>
    </head>
    <body>
        <?php
            // define variables and set to empty values
            $nameErr = $emailErr = $genderErr = $websiteErr = "";
            $name = $email = $gender = $comment = $website = "";
            if ($_SERVER["REQUEST_METHOD"] == "POST") {
                if (empty($_POST["name"])) {
                    $nameErr = "Name is required";
                }else {
                    $name = test_input($_POST["name"]);
                }
                if (empty($_POST["email"])) {
                    $emailErr = "Email is required";
                }else {
                    $email = test_input($_POST["email"]);
                }
            }
        </?php>
    </body>
</html>

```

---

```

        // check if e-mail address is well-formed
        if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
            $emailErr = "Invalid email format";
        }
    }

    if (empty($_POST["website"])) {
        $website = "";
    }else {
        $website = test_input($_POST["website"]);
    }
    if (empty($_POST["comment"])) {
        $comment = "";
    }else {
        $comment = test_input($_POST["comment"]);
    }
    if (empty($_POST["gender"])) {
        $genderErr = "Gender is required";
    }else {
        $gender = test_input($_POST["gender"]);
    }
}

function test_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}

?>
<h2>Absolute classes registration</h2>
<p><span class = "error">* required field.</span></p>
<form method = "post" action = "<?php
    echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
    <table>
        <tr>
            <td>Name:</td>
            <td><input type = "text" name = "name">
                <span class = "error">* <?php echo $nameErr;?></span>
            </td>
        </tr>
    </table>

```

---

```
<tr>
    <td>E-mail: </td>
    <td><input type = "text" name = "email">
        <span class = "error">* <?php echo $emailErr;?></span>
    </td>
</tr>
<tr>
    <td>Time:</td>
    <td> <input type = "text" name = "website">
        <span class = "error">*<?php echo $websiteErr;?></span>
    </td>
</tr>
<tr>
    <td>Classes:</td>
    <td> <textarea name = "comment" rows = "5" cols =
        "40"></textarea></td>
</tr>
<tr>
    <td>Gender:</td>
    <td>
        <input type = "radio" name = "gender"
               value = "female">Female
        <input type = "radio" name = "gender"
               value = "male">Male
        <span class = "error">*<?php echo $genderErr;?></span>
    </td>
</tr>
<td>
    <input type = "submit" name = "submit" value = "Submit">
</td>
</table>
</form>
<?php
    echo "<h2>Your given values are as:</h2>";
    echo $name;
    echo "<br>";
    echo $email;
    echo "<br>";
    echo $website;
    echo "<br>";
```

```

        echo $comment;
        echo "<br>";

        echo $gender;
    ?>
</body>
</html>

```

**Output:**

Absolute classes registration

\* required field.

Name:  \*

E-mail:  \*

Time:

Classes:

Gender:  Female  Male \*

Your given values are as:

**Program 16:** Program for mail function.

```

<html>
    <head>
        <title>Sending HTML email using PHP</title>
    </head>
    <body>
        <?php
            $to = "xyz@somedomain.com";
            $subject = "This is subject";
            $message = "<b>This is HTML message.</b>";
            $message .= "<h1>This is headline.</h1>";
            $header = "From:abc@somedomain.com \r\n";
            $header .= "Cc:afgh@somedomain.com \r\n";
            $header .= "MIME-Version: 1.0\r\n";
            $header .= "Content-type: text/html\r\n";
            $retval = mail ($to,$subject,$message,$header);
            if( $retval == true ) {
                echo "Message sent successfully...";
```

```

        }else {
            echo "Message could not be sent...";
        }
    ?>
</body>
</html>

```

**Output:**


---

Message sent successfully...

---

**Program 17:** Program to check that emails are valid.

```

<html>
    <head>
        <style>
            .error {color: #FF0000;}
        </style>
    </head>
    <body>
        <?php
        // define variables and set to empty values
        $nameErr = $emailErr = $genderErr = $websiteErr = "";
        $name = $email = $gender = $comment = $website = "";
        if($_SERVER["REQUEST_METHOD"] == "POST") {
        if(empty($_POST["name"])) {
            $nameErr = "Name is required";
        } else {
            $name = test_input($_POST["name"]);
            // check if name only contains letters and whitespace
            if(!preg_match("/^[a-zA-Z ]*$/",$name)) {
                $nameErr = "Only letters and white space allowed";
            }
        }
        if(empty($_POST["email"])) {
            $emailErr = "Email is required";
        } else {
            $email = test_input($_POST["email"]);
            // check if e-mail address is well-formed
            if(!filter_var($email, FILTER_VALIDATE_EMAIL)) {
                $emailErr = "Invalid email format";
            }
        }
        if(empty($_POST["website"])) {
            $website = "";
        } else {
            $website = test_input($_POST["website"]);
        }
    }

```

---

```

// check if URL address syntax is valid
if (!preg_match("/\b(?:https?|ftp):\/\/|www\.)[-a-z0
9+@#\%?=~_|!:,.;]*[-a-z0-9+@#\%=~_|]/i",$website)) {
    $websiteErr = "Invalid URL";
}
}
if(empty($_POST["comment"])) {
    $comment = "";
} else {
    $comment = test_input($_POST["comment"]);
}
if(empty($_POST["gender"])) {
    $genderErr = "Gender is required";
} else {
    $gender = test_input($_POST["gender"]);
}
}
function test_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}
?>
<h2>PHP Form Validation Example</h2>
<p><span class="error">* required field</span></p>
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
Name: <input type="text" name="name">
<span class="error">* <?php echo $nameErr;?></span>
<br><br>
E-mail: <input type="text" name="email">
<span class="error">* <?php echo $emailErr;?></span>
<br><br>
Website: <input type="text" name="website">
<span class="error"><?php echo $websiteErr;?></span>
<br><br>
Comment: <textarea name="comment" rows="5" cols="40"></textarea>
<br><br>

```

```
Gender:  
<input type="radio" name="gender" value="female">Female  
<input type="radio" name="gender" value="male">Male  
<input type="radio" name="gender" value="other">Other  
<span class="error">*<?php echo $genderErr;?></span>  
<br><br>  
<input type="submit" name="submit" value="Submit">  
</form>  
<?php  
    echo "<h2>Your Input:</h2>";  
    echo $name;  
    echo "<br>";  
    echo $email;  
    echo "<br>";  
    echo $website;  
    echo "<br>";  
    echo $comment;  
    echo "<br>";  
    echo $gender;  
?  
    </body>  
</html>
```

**Output:**

The screenshot shows a web browser window with the URL <http://localhost:8080/emailvalidation.php>. The page title is "PHP Form Validation Example". The form includes fields for Name, E-mail, Website, and Comment, each marked with an asterisk (\*) indicating they are required. The E-mail field contains an invalid email address and displays an error message: "\* Invalid email format". Below the form, a "Your Input:" section shows the submitted data: "manisha" and "abc".



# Introduction to HTML, HTTP and PHP

## Objectives...

- To understand Basic Concepts in HTML
- To learn Basics of HTTP
- To study Basic Concepts of PHP
- To learn Language Basics and Lexical Structure of PHP

### 1.0 INTRODUCTION

- Web technologies refer to techniques and tools that are used to produce fully-featured Websites and Web applications.
- A Website is a collection of web pages. A document on the Web is called a Web page and is identified by a unique address called the Uniform Resource Locator (URL).
- A Web application (Web app) is an application program that is stored on a Web server and delivered over the Internet through a Browser.
- The two types of Web pages are static web page and dynamic web page. The content (data and information) in static web pages does not get changed until someone changed it manually while the content in dynamic web pages changes dynamically (runtime).
- The two distinct parts of a website are the frontend (refers to all those parts of a website that a user can see on their screen and interact with) and the backend (involves the hidden mechanisms where functions, methods, and data manipulation happens).
- The World Wide Web (WWW) consists of files, called pages or web pages, which contain information and links to resources throughout the Internet.
- In simple word, all publicly accessible websites collectively constitute the World Wide Web.
- A web page is an electronic document written in a computer language called HTML (HyperText Markup Language). HTML is the standard markup language for creating Web pages and Web applications.
- Web design is the process of creating websites. Website is a collection of related web pages that may contain text, images, audio and video.

- A web browser or a browser (client) is application software for accessing the World Wide Web. Some popular web browsers are Opera, Mozilla Firefox, Google Chrome, and Safari.
- When a user requests a web page from a particular website, the web browser retrieves the necessary content from a Web server (server) and then displays the page on the user's device like desktops, laptops, tablets, and smart phones.
- Web development, or web programming, refers to the design of software applications for a Web site. Web publishing or Online publishing, is the process of publishing content on the Internet.
- Web technology is a collective name for technologies primarily for the World Wide Web (WWW) or Web. These technologies includes Markup Language (HTML), Programming languages (JavaScript, Python, PHP etc.), Frameworks (Node.js, Angular.js, .NET, Drupal etc.), Databases (MongoDB, PostgreSQL, MySQL etc.) CSS (Cascading Style Sheet), Libraries (JQuery), Data formats (XML (Extensible Markup Language), JSON (JavaScript Object Notation)) and so on.
- In this chapter we will study HTML, CSS, HTTP and PHP technologies related to web technologies.

## 1.1 OVERVIEW OF HTML

- HTML stands for HyperText Markup Language. HTML is the standard markup language used for creating Web pages.
- HTML was created by Tim Berners-Lee in late 1991. HTML is used to create Web pages which are rendered over the Web.
- HTML5 is the latest version of HTML language. HTML5 is a markup language used for structuring and presenting content on the World Wide Web.

### Features of HTML:

1. HTML is a very **easy and simple language** and can be easily understood and modified.
2. It is a **markup language** so it provides a flexible way to design web pages along with the text.
3. It is **platform-independent** because it can be displayed on any platform like Windows, Linux and Macintosh etc.
4. It facilitates the programmer to add Graphics, Videos, and Sound to the web pages which makes it more **attractive and interactive**.
5. **Canvas** feature allows a web developer to render graphics on the fly. As with video, there is no need for a plug in.
6. It is very easy to make **effective presentation** with HTML because it has a lot of formatting tags.
7. It supports **scripting languages** to support dynamic web applications (e.g., forms that react to user input).

- An HTML file is a text file with extension .htm or .html. To create the HTML source document, we need an HTML Editor. An HTML editor is a program for editing HTML, the markup of a web page.
- An HTML editor is a computer program used for creating and editing Web pages. There are two main varieties of HTML editors i.e. text and WYSIWYG (What You See Is What You Get) editors.
- A text editor is a type of program used for editing plain text files. A text-based HTML editor includes Notepad on Windows, GNU Emacs on UNIX/Linux, or SimpleText on the Macintosh.
- The WYSIWYG editors provide an editing interface which resembles how the page will be displayed in a web browser like Adobe Dreamweaver, KomodoIDE, EditPlus etc.
- Other professional text editors for HTML include Notepad++, Sublime Text, Vim, Atom, and Visual Studio Code and so on.

#### HTML Page Structure:

- Fig. 1.1 shows page structure of HTML. A web page is also known as HTML page or HTML document.
- A basic HTML page starts with the Document Type Declaration also called DOCTYPE declaration before the `<html>` tag. The `<!DOCTYPE>` declaration represents the document type.
- The Document Type Declaration is not an HTML tag; it is an instruction to the web browser about what version of HTML the page is written in.
- The current version of HTML is HTML5 and it makes use of the only one declaration `<!DOCTYPE html>`.

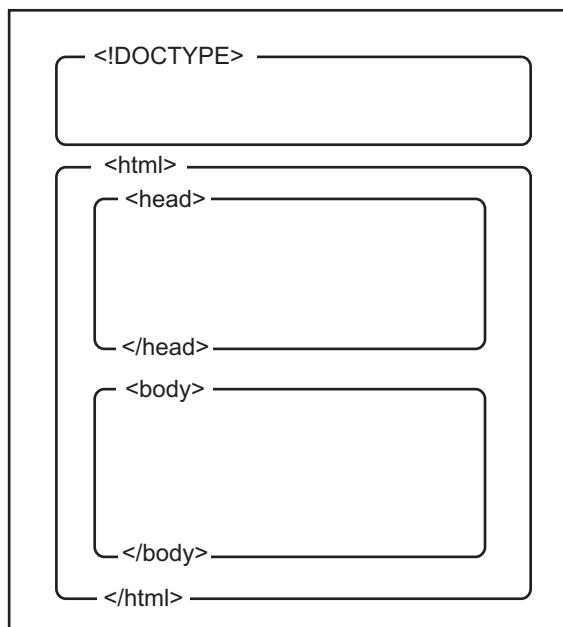


Fig. 1.1: HTML Web Page Structure

- The entire HTML document is contained within an opening `<html>` tag and a closing `</html>` tag.
- Within the `<html>` tag, the document is divided into a head and a body. The `<head>` tag contains descriptive information about the document itself, such as its title, the style sheet(s) it uses and so on.
- The `<body>` tag contains the entire information or content about the web page and its behavior. It also contains the information that we want to display on a Web page.

### Example for creating an html document:

```
<!DOCTYPE html>
<html>
  <head>
    <title>
      </title>
    <body>
      <h1>Nirali Prakashan</h1>
      <p>Textbook Publication Firm.</p>
    </body>
  </head>
</html>
```

### Output:

**Nirali Prakashan**

Textbook Publication Firm.

### 1.1.1 HTML Basic Tags

- HTML documents are simply a text file made up of HTML elements and these elements are defined using HTML tags.
- An element is the basic building block of HTML and is typically made up of two tags namely, an opening or start tag, some contents and a closing or end tag as shown in Fig. 1.2.

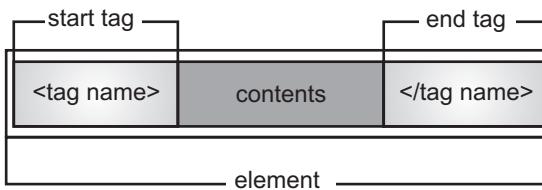
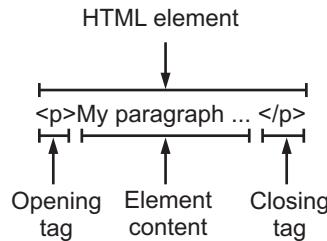


Fig. 1.2: Element in HTML

- Fig. 1.3 shows an example of HTML element. In this example, the HTML element is a paragraph `<p>` opening and closing tags with “My paragraph...” content that will display on the web page.

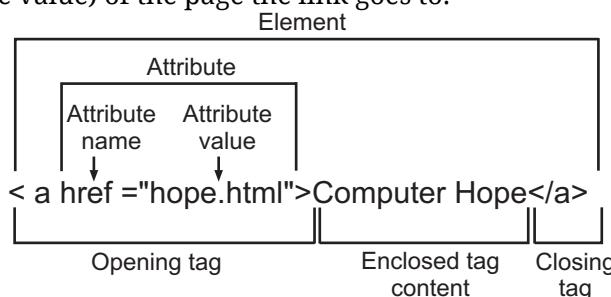


**Fig. 1.3: Example of an Element in HTML**

- Tags contain elements which provide instructions for how information will be processed or displayed on a web page.
- HTML tags are labels we use to mark up the beginning and end of an element. All tags have the same format and they begin with a less-than sign "<" and end with a greater-than sign ">" also known as angle brackets.
- Generally speaking, there are two kinds of tags i.e., an opening tag like <html> and a closing tag like </html>.
- A closing tag must include forward slash "/" and it controls the appearance, layout and flow of the web page. The information within an element's opening and closing tags is its content.

**Syntax of Tag:** <tag\_name> Content... </tag\_name>

- A tag contains three parts i.e. element (identification of tag), attribute and value as shown in Fig. 1.4.
- The additional information supplied to an HTML tag is known as attributes of a tag. Attributes are written immediately following the tag, separated by a space like <font face="arial"> Welcome </font>.
- Multiple attributes can be associated with a tag, also separated by a space like <font face = "arial" size = 12> Welcome </font>. The face and size are multiple attributes of the <font> tag.
- In Fig. 1.4 the anchor <a> tag defines a hyperlink in which the href attribute specifies the URL (attribute value) of the page the link goes to.



**Fig. 1.4: Example of an Element, Tag, Attribute in HTML**

- HTML tags can be of following two types:
  - Paired Tags:** In paired tag, first tag is called the opening tag like <b> and the second tag is called the closing tag like </b>. Paired tag is also called as container tag. In other words, a tag is said to a paired tag if it along with a companion tag or closing tag appears at the end.

2. **Singular Tags:** The singular tag is also known as a stand-alone tag or empty tag. The stand-alone tag does not have companion tag or closing tag. For example <br> tag will insert a line break. This tag does not require any companion tag or a closing tag.

### Basic HTML Tags:

- HTML contains following basic tags:

1. **<html> Tag:** The <html> tag represents the root of an HTML document. The <html> tag is the container that contains all other HTML elements (except for the <!DOCTYPE> declaration which is located before the opening HTML tag). The <html> tag tells the browser that this is an HTML document.

**Syntax:** <html>.....</html>

Attribute	Description
1. Manifest	This attribute specifies the address of the document's application cache manifest and the value must be a valid URL.

2. **<title> Tag:** The <title> tag is used for declaring the title, or name, of the HTML document. The title is usually displayed in the browser's title bar (at the top) or the <title> tag defines a title in the browser toolbar. It is also displayed in browser bookmarks and search results.

**Syntax:** <title>.....</title>

3. **<head> Tag:** The <head> tag is a container for all the head elements and must include a title for the document, and can include scripts, styles, meta information, and so on.

**Syntax:** <head>.....</head>

4. **<body> Tag:** The <body> tag defines the document's body. The body tag is placed between the </head> and the </html> tags. The <body> tag contains all the contents of an HTML document, such as text, hyperlinks, images, tables, lists, etc.

**Syntax:** <body>.....</body>

### Attributes of <body> Tag:

Attribute	Value	Description
1. alink	color	This attribute specifies the color of an active link in a document.
2. background	URL	This attribute specifies a background image for a document.
3. bgcolor	color	This attribute specifies the background color of a document.

*contd. ...*

4. link	color	This attribute specifies the color of unvisited links in a document.
5. text	color	This attribute specifies the color of the text in a document.
6. vlink	color	This attribute specifies the color of visited links in a document.

**Example for basic HTML tags:**

```
<!DOCTYPE html>
<html>
  <head>
    <title> Example of HTML Basic Tags</title>
  <body bgcolor = "orange">
    <p>Document content goes... here.</p>
  </body>
  </head>
</html>
```

**Output:**

Document content goes... here.

**HTML Text Formatting Tags:**

- The HTML tags are used for formatting text are called as text formatting tags. Following are the text formatting tags used in HTML:

Tag	Description
1. <b>	The <b> tag defines bold text. Anything that appears in a <b>...</b> element is displayed in bold. <b>Syntax:</b> <b>.....</b>
2. <em>	The <em> tag is used for indicating emphasis. The <em> tag surrounds the word/term being emphasised. <b>Syntax:</b> <em>.....</em>
3. <i>	The content of the <i> tag is usually displayed in italic. The <i> tag can be used to indicate a technical term, a phrase from another language, a thought etc. <b>Syntax:</b> <i>.....</i>
4. <small>	The <small> tag defines smaller text (and other side comments). The content of the <small> element is displayed one font size smaller than the rest of the text surrounding it. <b>Syntax:</b> <small>.....</small>

*contd. ...*

5. <code>&lt;strong&gt;</code>	The <code>&lt;strong&gt;</code> tag is used for indicating strong importance for its contents. The strong tag surrounds the emphasized word/phrase. <b>Syntax:</b> <code>&lt;strong&gt;.....&lt;/strong&gt;</code>
6. <code>&lt;sub&gt;</code>	The <code>&lt;sub&gt;</code> tag defines subscript text. Subscript text appears half a character below the baseline. Subscript text can be used for chemical formulas, like H <sub>2</sub> O. <b>Syntax:</b> <code>&lt;sub&gt;.....&lt;/sub&gt;</code>
7. <code>&lt;sup&gt;</code>	The <code>&lt;sup&gt;</code> tag defines superscript text. Superscript text appears half a character above the baseline like 10 <sup>5</sup> . <b>Syntax:</b> <code>&lt;sup&gt;.....&lt;/sup&gt;</code>
8. <code>&lt;ins&gt;</code>	The <code>&lt;ins&gt;</code> tag defines a text that has been inserted into a document. <b>Syntax:</b> <code>&lt;ins&gt;.....&lt;/ins&gt;</code>
9. <code>&lt;del&gt;</code>	The <code>&lt;del&gt;</code> tag defines text that has been deleted from a document. <b>Syntax:</b> <code>&lt;del&gt;.....&lt;/del&gt;</code>
10. <code>&lt;u&gt;</code>	The <code>&lt;u&gt;</code> tag usually results in the text being underlined. Anything that appears in a <code>&lt;u&gt;...&lt;/u&gt;</code> element is displayed with underline, <b>Syntax:</b> <code>&lt;u&gt;.....&lt;/u&gt;</code>
11. <code>&lt;strike&gt;</code>	Anything that appears in a <code>&lt;strike&gt;</code> tag is displayed with strikethrough, which is a thin line through the text like, <u>strikethrough</u> . <b>Syntax:</b> <code>&lt;strike&gt;.....&lt;/strike&gt;</code>
12. <code>&lt;big&gt;</code>	The content of the <code>&lt;big&gt;</code> element is displayed one font size larger than the rest of the text surrounding it. <b>Syntax:</b> <code>&lt;big&gt;.....&lt;/big&gt;</code>

### Example for text formatting tags:

```
<!DOCTYPE html>
<html>
  <head>
    <title> Example of HTML Text Formatting Tags </title>
  </head>
  <body>
    <p>Nirali Prakashan</p>
    <font size="4">Wecome to <b>Nirali Prakashan.</b> <br/></font>
    <font size="2"><i>Nirali Prakashan</i> is a textbook publication
      firm.<br/></font>
    <font face="verdana" color="black">Nirali Prakashan <u>publised
      more than 3500 book titles.</u></font>
  </body>
</html>
```

### Output:

Nirali Prakashan

Welcome to **Nirali Prakashan**.

Nirali Prakashan is a textbook publication firm.

**Nirali Prakashan publised more than 3500 book titles.**

### HTML Block Level Tags:

- Most HTML tags are defined as block level elements. Block level tags normally start (and end) with a new line when displayed in a browser.
  - <!----> Comment Tag:** The comment tag is used to insert comments in the source code in a HTML document. Comments are not displayed in the browsers. Comments can be inserted into the HTML code to make it more readable and understandable.
  - Heading <h1> to <h6> Tags:** The HTML <h1> to <h6> tags are used to define HTML headings. <h1> defines the most important heading, while <h6> defines the least important heading.

**Syntax:**

```
<h1>.....</h1>
<h2>.....</h2>
<h3>.....</h3>
<h4>.....</h4>
<h5>.....</h5>
<h6>.....</h6>
```

**Attributes for Header Tags:**

Attribute	Value	Description
1. align	left center right justify	This attribute specifies the alignment of a heading.

- <p> Tag:** HTML documents are divided into paragraphs. The HTML <p> tag is used for defining a paragraph.

**Syntax:** <p>.....</p>

**Attributes for Paragraph Tag:**

Attribute	Value	Description
1. align	left right center justify	Align attribute specifies the alignment of the text within a paragraph.

- <br> Tag:** The HTML <br> tag is used for specifying a line break. The <br> tag is an empty tag. In other words, it has no end tag. **Syntax:** <br/>
- <center> Tag:** The <center> tag is used to center-align text.

**6. Non breaking Spaces:** Suppose we were to use the phrase "14 Angry Men." Here we would not want a browser to split the "14" and "Angry" across two lines. A good example of this technique appears in the movie "14 Angry Men." In cases where we do not want the client browser to break text, we should use a non-breaking space entity (&nbsp;) instead of a normal space. For example, when coding the "14 Angry Men" paragraph, we would use something similar to the following code:

```
<p>A good example of this technique appears in the movie  
"14 Angry Men."</p>
```

**7. <div> Tag:** The <div> tag defines a division or a section in an HTML document. The <div> tag is used to group block-elements to format them with CSS.

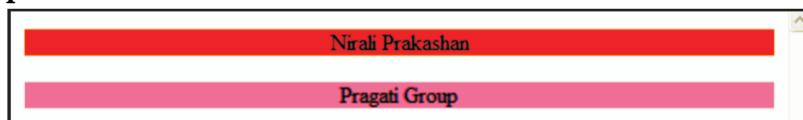
**Syntax:** <div>.....</div>

Attribute	Value	Description
1. align	left right center justify	Align attribute specifies the alignment of the content inside a <div> element

**Example for <div> tag:**

```
<!DOCTYPE html>  
<html>  
  <head>  
    <title> Example of HTML div Tag </title>  
  </head>  
  <body>  
    <div style="background-color:orange;text-align:center">  
      <p>Nirali Prakashan</p>  
    </div>  
    <div style="background-color:pink;text-align:center">  
      <p>Pragati Group</p>  
    </div>  
  </body>  
</html>
```

**Output:**



**8. <span> Tag:** The <span> tag is used for grouping and applying styles to inline elements.

**Syntax:** <span>.....</span>

**Example for <span> tag:**

```
<!DOCTYPE html>
<html>
  <head> span tag
    <title> HTML span tag Example</title>
  </head>
  <body>
    <p>The <span style="color:red">span tag is used with inline elements</span> and the <span style="color:purple">div tag</span> is used with block-level content.</p>
  </body>
</html>
```

**Output:**

span tag

The span tag is used with inline elements and the div tag is used with block-level content.

9. **<hr> Tag:** The HTML <hr> tag is used for specifying a horizontal rule in an HTML document. The <hr> tag is used for creating a horizontal line which separate content (or define a change) in an HTML page.

**Syntax:** <hr/>

**Attributes for <hr> Tag:**

Attribute	Value	Description
1. align	left center right	This attribute specifies the alignment of a <hr> element.
2. noshade	noshade	This attribute specifies that a <hr> element should render in one solid color (no-shaded), instead of a shaded color.
3. size	pixels	This attribute specifies the height of a <hr> element.
4. width	pixels %	This attribute specifies the width of a <hr> element.

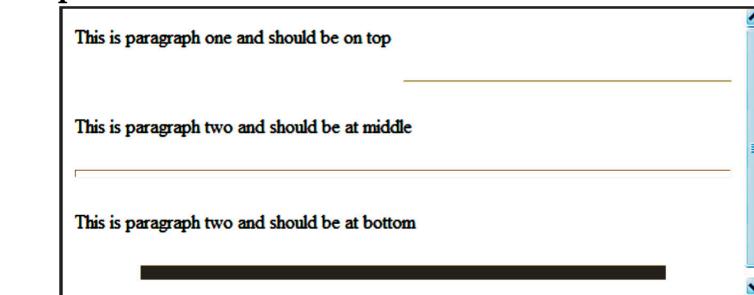
**Example for <hr> tag:**

```
<!DOCTYPE html>
<html>
  <head>
    <title> Example for HTML <hr> Tag</title>
  </head>
  <body>
    <p>This is paragraph one and should be on top</p>
```

```

<hr align="right" width="50%">
<p>This is paragraph two and should be at middle</p>
<hr size="7">
<p>This is paragraph two and should be at bottom</p>
<hr width="80%" size="12" noshade>
</body>
</html>

```

**Output:**

10. **<font> Tag:** Fonts play very important role in making a website more user friendly and increasing content readability. The <font> tag specifies the font face, font size, and font color of text.

**Syntax:** <font>.....</font>

**Attributes for <font> tag:**

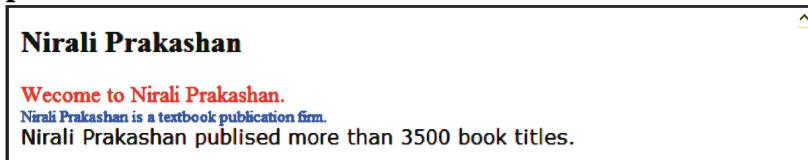
Attribute	Value	Description
1. color	rgb(x,x,x) #xxxxxx colorname	This attribute specifies the color of text.
2. face	font_family	This attribute specifies the font of text.
3. size	number	This attribute specifies the size of text.

**Example for <font> tag:**

```

<!DOCTYPE html>
<html>
  <head>
    <title> Example for HTML font Tag </title>
  </head>
  <body>
    <p><h2>Nirali Prakashan</h2></p>
    <font size="4" color="red">Wecome to Nirali Prakashan. <br/></font>
    <font size="2" color="blue">Nirali Prakashan is a textbook
                                publication firm.<br/></font>
    <font face="verdana" color="black">Nirali Prakashan publised more
                                than 3500 book titles.</font>
  </body>
</html>

```

**Output:****HTML "Computer Output" or "Code Formatting" Tags:**

- HTML normally uses variable letter size and spacing. This is not what we want when displaying computer code for this purpose Computer Code Elements (tags) are used like `<kbd>`, `<samp>`, `<code>` etc. and all these tags are displayed in fixed letter size and spacing.
- **HTML Citations, Quotations and Definition Tags** includes `<abbr>`, `<address>`, `<bdo>`, `<blockquote>`, `<q>`, `<cite>` etc.

**Hyperlinks in HTML:**

- The real power of HTML is its ability to link one document to the other document. A link is a connection from one web resource (documents) to another.
- A webpage can contain various links that take us directly to other pages and even specific parts of a given page. These links are known as hyperlinks.
- A hypertext link, or hyperlink or link, contains a reference to a specific Web page that we can click to open that Web page.
- Hyperlinks allow visitors to navigate between Web sites by clicking on words, phrases, and images.

**HTML `<a>` Anchor Tag:**

- The HTML `<a>` tag is used for creating a hyperlink to another web page. The `<a>` tag can be used in two ways:
  1. To create a link to another document, by using the `href` attribute, and
  2. To create a bookmark inside a document, by using the `name` attribute.
- The `<a>` tag defines a hyperlink, which is used to link from one page to another. By default, links will appear as follows in all browsers:
  1. An unvisited link is underlined and blue,
  2. A visited link is underlined and purple, and
  3. An active link is underlined and red.

**Syntax:** `<a href="Document URL">.....</a>`

Attributes	Value	Description
1. <code>charset</code>	<code>char_encoding</code>	This attribute specifies the character-set of a linked document.
2. <code>coords</code>	<code>coordinates</code>	This attribute specifies the coordinates of a link.
3. <code>href</code>	<code>URL</code>	This attribute specifies the URL of the page the link goes to.

*contd. ...*

4. hreflang	language_code	This attribute specifies the language of the linked document.
5. media	media_query	This attribute specifies what media/device the linked document is optimized for.
6. name	section_name	This attribute specifies the name of an anchor.
7. rel	alternate author bookmark help license next nofollow noreferrer prefetch prev search tag	This attribute specifies the relationship between the current document and the linked document.
8. rev	text	This attribute specifies the relationship between the linked document and the current document.
9. shape	default rect circle poly	This attribute specifies the shape of a link.
10. target	_blank _parent _self _top framename	This attribute specifies where to open the linked document.
11. type	MIME_type	This attribute specifies the MIME type of the linked document.

#### Example for <a> tag:

```
<!DOCTYPE html>
<html>
    <head>
        <title> Example for Hyperlink in HTML </title>
    </head>
    <body>
        <p> <a href="default.asp">HTML</a> This is a link to a page on this
           website.</p>
```

```

<p> <a href="http://www.google.com/">Google</a> This is a link to a
website on the World Wide Web.</p>
</body>
</html>

```

**Output:**

How to create hyperlinks

HTML This is a link to a page on this website.

Google This is a link to a website on the World Wide Web.

**Concept of URL:**

- Each and every web page has a unique address, called a Uniform Resource Locator (URL) that identifies its location on the Internet.

**Parts of URL:**

- URL has several parts i.e., the protocol, the domain name, the directory or folder, and the filename and its extension. Below is an example of URL and the corresponding parts.



- The first part of URL is called protocol in the URL above the protocol being used is the https protocol which enables the web browsing. HTTP protocol allows the computers in the World Wide Web to talk to each other. It provides a set of instructions for precise information exchange.
- The domain name is the second part of URL. It is a text name that corresponds to the IP address of the server that serves the web site.
- Text name is easier to remember that is why domain name is used instead of the IP address. In our example the domain name is `www.niraliprakashan.com`.
- There are domain names that ends with .com (for commercial), .edu (for education), .org (for organization), .info (for information) and many more.
- Another part of the URL identifies the directory name or the folder name. It is the actual location of the file you want to access.
- When creating links make sure that the directory exists and contains the file we want to link to. In the given example the name of the directory or folder is `html`.
- A directory may consist of one or more sub directories. The last part of the URL is the file name is `index.html`. In creating links, make sure that the file exists and make sure to use the appropriate file extension such as `.htm` or `.html`.
- We may address a URL in one of the following two ways:
  1. **Absolute:** An absolute URL is the complete address of a resource. For example,

`https://www.niraliprakashan.com/html/html_text_links.htm`

2. **Relative:** A relative URL indicates where the resource is in relation to the current page. Given URL is added with the `<base>` element to form a complete URL. For example, `/html/html_text_links.htm`. There are two types of relative URLs:
  - (i) **In Document Relative URLs**, the document address is given in relation with the originating document.
  - (ii) **In Server Relative URLs**, the document address is given in relation with the server on which the document is present.

### Lists in HTML:

- In HTML, we can list out the items, subjects or menu in the form of a list. HTML gives us three different types of lists as explained below:
  1. **Unordered Lists:** An unordered list is a collection of related items that have no special order or sequence. Unordered list is created by using `<ul>` tag. Each item in the list is marked with a bullet. The bullet itself comes in three flavors: squares, discs, and circles.
  2. **Ordered Lists:** The Ordered list is created by using `<ol>` tag. Each item in the list is marked with a number. The numbering starts at one and is incremented by one for each successive ordered list element tagged with `<li>` tag.
  3. **Definition Lists:** The definition list is the ideal way to present a glossary, list of terms, or other name/value list. Definition List makes use of `<dl>`, `<dt>` and `<dd>`tags.
- Metadata is data (information) about data. Meta elements are typically used to specify page description, keywords, author of the document, last modified and other metadata.
- The `<meta>` tag is used for declaring metadata for the HTML document. The `<meta>` tag always goes inside the head element.

**Syntax:** `<meta name = string content = string>`

1. **<li> List Tag:** The `<li>` tag defines a list item. The `<li>` tag is used in ordered lists (`<ol>`), unordered lists (`<ul>`), and in menu lists (`<menu>`).

**Syntax:** `<li>.....</li>`

Attributes	Value	Description
1. <code>type</code>	1 A a I i disc square circle	This attribute specifies which kind of bullet point will be used.
2. <code>value</code>	number	This attribute specifies the value of a list item.

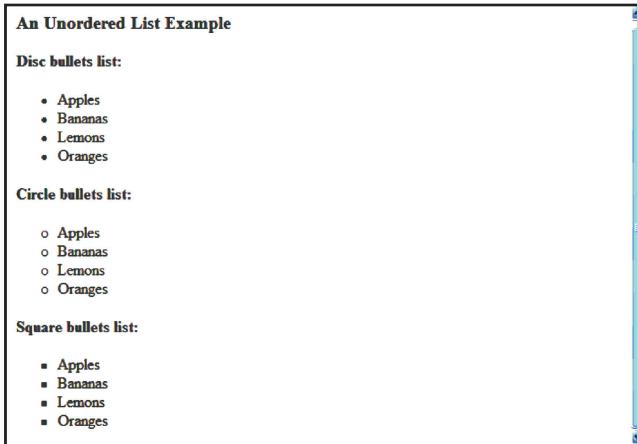
2. **<ul> Unordered List Tag:** An unordered list starts with the <ul> tag. Each list item starts with the <li> tag. The <ul> tag defines an unordered (bulleted) list.

**Syntax:** <ul>.....</ul>

Attributes	Value	Description
1. compact	compact	This attribute specifies that the list should render smaller than normal.
2. type	disc square circle	This attribute specifies the kind of marker to use in the list.

#### Example for <ul> tag:

```
<!DOCTYPE html>
<html>
    <head>
        <h3>An Unordered List Example</h3>
    <body>
        <h4>Disc bullets list:</h4>
        <ul type="disc">
            <li>Apples</li>
            <li>Bananas</li>
            <li>Lemons</li>
            <li>Oranges</li>
        </ul>
        <h4>Circle bullets list:</h4>
        <ul type="circle">
            <li>Apples</li>
            <li>Bananas</li>
            <li>Lemons</li>
            <li>Oranges</li>
        </ul>
        <h4>Square bullets list:</h4>
        <ul type="square">
            <li>Apples</li>
            <li>Bananas</li>
            <li>Lemons</li>
            <li>Oranges</li>
        </ul>
    </body>
    </head>
</html>
```

**Output:**

- **Nested List:** One list inside another list is known as nesting of list or nested list. We can nest an unordered list as follows:

```
<!DOCTYPE html>
<html>
    <body>
        <h4>A nested List:</h4>
        <ul>
            <li>Coffee</li>
            <li>Tea
                <ul>
                    <li>Black tea</li>
                    <li>Green tea
                        <ul>
                            <li>China</li>
                            <li>Africa</li>
                        </ul>
                    </li>
                </ul>
            </li>
            <li>Milk</li>
        </ul>
    </body>
</html>
```

**Output:**

```

A nested List:

- Coffee
- Tea
  - Black tea
  - Green tea
    - China
    - Africa
- Milk

```

3. **<ol> Ordered List Tag:** An ordered list starts with the <ol> tag. Each list item starts with the <li> tag. Each item in the list is marked with a number.

**Syntax:** <ol>.....</ol>

Attributes	Value	Description
1. compact	compact	This attribute specifies that the list should render smaller than normal.
2. reversed	reversed	This attribute specifies that the list order should be descending like 9,8,7...
3. start	number	This attribute specifies the start value of an ordered list.
4. type	1 A a I i	This attribute specifies the kind of marker to use in the list.

**Example for <ol> tag:**

```

<!DOCTYPE html>
<html>
  <head>
    <body>
      <h3>An Ordered List Example</h3>
      <h4>Numbered list:</h4>
      <ol>
        <li>Apples</li>
        <li>Bananas</li>
        <li>Lemons</li>
      </ol>
    </body>
  </html>

```

```
<h4>Letters list:</h4>
<ol type="A">
    <li>Apples</li>
    <li>Bananas</li>
    <li>Lemons</li>
</ol>
<h4>Lowercase letters list:</h4>
<ol type="a">
    <li>Apples</li>
    <li>Bananas</li>
    <li>Lemons</li>
</ol>
<h4>Roman numbers list:</h4>
<ol type="I">
    <li>Apples</li>
    <li>Bananas</li>
    <li>Lemons</li>
</ol>
<h4>Lowercase Roman numbers list:</h4>
<ol type="i">
    <li>Apples</li>
    <li>Bananas</li>
    <li>Lemons</li>
</ol>
</body>
</head>
</html>
```

**Output:**

An Ordered List Example

Numbered list:

1. Apples
2. Bananas
3. Lemons

Letters list:

- A. Apples
- B. Bananas
- C. Lemons

Lowercase letters list:

- a. Apples
- b. Bananas
- c. Lemons

Roman numbers list:

- I. Apples
- II. Bananas
- III. Lemons

Lowercase Roman numbers list:

- i. Apples
- ii. Bananas
- iii. Lemons

- 4. Definition Lists:** A definition list is a list of items, with a description of each item. The `<dl>` tag defines a definition list. The `<dl>` tag is used to provide a list of items with associated definitions.

**Syntax:** `<dl>.....</dl>`

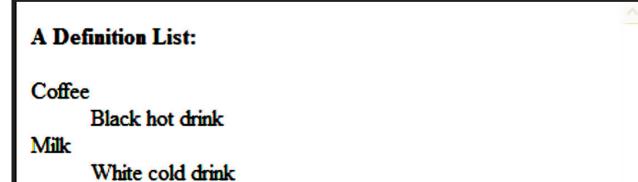
Definition list makes use of following two tags:

- (i) `<dt>`: The `<dt>` tag is used inside `dl`. It marks up a term whose definition is provided by the next `dd`. The `dt` tag may only contain text-level markup.
- (ii) `<dd>`: The `<dd>` tag is used inside a `dl` definition list to provide the definition of the text in the `dt` tag. It may contain block elements but also plain text and markup.

#### Example for definition list:

```
<!DOCTYPE html>
<html>
  <body>
    <h4>A Definition List:</h4>
    <dl>
      <dt>Coffee</dt>
        <dd>Black hot drink</dd>
      <dt>Milk</dt>
        <dd>White cold drink</dd>
    </dl>
  </body>
</html>
```

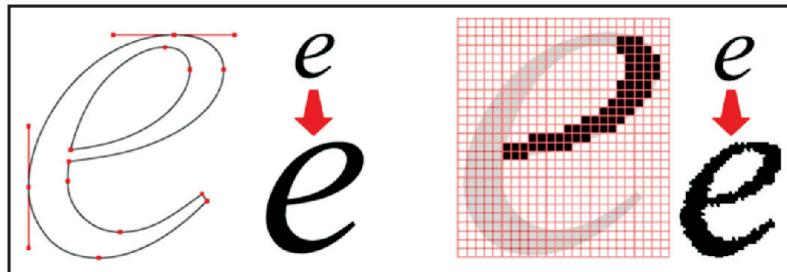
#### Output:



#### Images in HTML:

- It is true that one single image is worth than thousands of words. So as a web developer we should have clear understanding on how to use images in the web pages.
- Images are used in many ways to enhance the look of a Web page and make it more interesting and colorful.
- There are basically following two types of graphic programs for images:
  1. **Bitmap (or Raster) images** are stored as a series of tiny dots called pixels. Each pixel is actually a very small square that is assigned a color, and then arranged in a pattern to form the image. Bitmap graphics can be edited by erasing or changing the color of individual pixels using a program such as Adobe Photoshop.

2. **Vector images** are not based on pixel patterns, but instead use mathematical formulas to draw lines and curves that can be combined to create an image from geometric objects such as circles and polygons. Vector images are edited by manipulating the lines and curves that make up the image using a program such as Adobe Illustrator.



(a) Vector Image

(b) Bitmap Image

Fig. 1.5

#### Types of Images:

- Some common types of images used in HTML are listed below:
  1. **JPEG (Joint Photographic Experts Group)** is the most popular among the image formats used on the Web. JPEG files usually have a filename extension of .jpg or .jpeg.
  2. **GIF (Graphic Interchange Format)** images are limited to 256 colors; they are cross-platform, which means any computer can view them. GIF files have the .gif extension.
  3. **BMP (Bitmap)** is the standard Windows image format on DOS and Windows compatible computers. The BMP format supports RGB (Red, Green, Blue) indexed-colors, grayscale, and Bitmap color modes. BMP files have the .bmp extension.
  4. **PDF (Portable Document Format)** is used by Adobe Acrobat. PDF files can represent both vector and bitmap graphics. The Photoshop PDF format supports RGB, indexed-colors, CMYK (Cyan, Magenta, Yellow and Black), grayscale and Bitmap. PDF file has .pdf extension.
  5. **TIFF (Tagged-Image File Format)** is used to exchange files between applications and computer platforms. Virtually all paint programs, image editing, and page layout applications support TIFF file format. TIFF files have the .tif or .tiff extension.
  6. **PNG (Portable Network Graphics)** pronounced "ping" was developed as an alternative to GIF. PNG files support 24-bit images and produces background transparency without jagged edges. PNG files have the .png extension.

#### HTML <img> Image Tag:

- In HTML, images are defined with the <img> tag. The <img> tag is empty, which means that it contains attributes only, and has no closing tag.

- We will insert any image in our web page by using <img> tag.

**Syntax:**

```

```

Attributes	Value	Description
1. alt	text	This attribute specifies an alternate text for an image.
2. src	URL	This attribute specifies the URL of an image.
3. align	top bottom middle left right	This attribute specifies the alignment of an image according to surrounding elements.
4. border	pixels	This attribute specifies the width of the border around an image.
5. crossorigin	anonymous use-credentials	This attribute allow images from third-party sites that allow cross-origin access to be used with canvas.
6. height	pixels	This attribute specifies the height of an image.
7. hspace	pixels	This attribute specifies the whitespace on left and right side of an image.
8. ismap	ismap	This attribute specifies an image as a server-side image-map.
9. longdesc	URL	This attribute specifies the URL to a document that contains a long description of an image.
10. src	URL	This attribute specifies the URL of an image.
11. usemap	#mapname	This attribute specifies an image as a client-side image-map.
12. vspace	pixels	This attribute specifies the whitespace on top and bottom of an image.
13. width	pixels	This attribute specifies the width of an image.

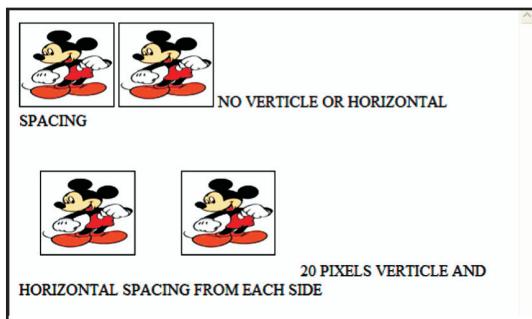
**Example for image:**

```
<!DOCTYPE html>
<html>
    <head>
        <meta name="GENERATOR" content="MICROSOFT FRONTPAGE 4.0">
        <meta name="PROGID" content="FRONTPAGE.EDITOR.DOCUMENT">
        <title>IMAGE SPACING</title>
    </head>
    <body>
        <p>
```

```

<img src = "http://www.umnet.com/pic/diy/screensaver/8/mickey-mouse-87422.jpg"
      border=1 width="90" height="80">
<img src = "http://www.umnet.com/pic/diy/screensaver/8/mickey-mouse-87422.jpg"
      border=1 width="90" height="80"> NO VERTICLE OR HORIZONTAL SPACING</p>
<p>
<img src = "http://www.umnet.com/pic/diy/screensaver/8/mickey-mouse-87422.jpg"
      width = "90" height = "80" border=1 HSPACE = "20" VSPACE = "20">
<img src = "http://www.umnet.com/pic/diy/screensaver/8/mickey-mouse-87422.jpg"
      width = "90" height = "80" border=1 HSPACE = "20" VSPACE = "20" > 20 PIXELS
      VERTICLE AND HORIZONTAL SPACING FROM EACH SIDE</p>
</body>
</html>

```

**Output:****Image as a Link:**

- Anything can be a link i.e., text or images. To make an image into a link we simply put the image tag inside the tag for a link. The tag would look like this:

```
<a href="http://www.anycartoonsite.com"></a>
```

**Example for image as link:**

```

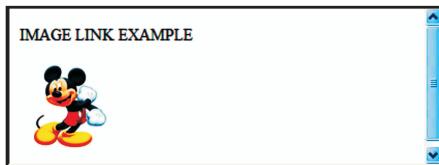
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="CONTENT-TYPE" CONTENT="TEXT/HTML;
                                              CHAR SET = WINDOWS-1252">
    <meta name="GENERATOR" CONTENT="MICROSOFT FRONTPAGE 4.0">
    <meta name="PROGID" CONTENT="FRONTPAGE.EDITOR.DOCUMENT">
    <title>IMAGELINK</title>
  </head>

```

```

<body>
    IMAGE LINK EXAMPLE
    <p>
        <a href = "HTTP://WWW.YAHOO.COM" ><img src = "http://images4.wikia.
        nocookie.net/_cb20120710211159/cartoons/images/8/80/Mickey_Mouse.jpg"
        border = "0" width = "90" height = "80"> </a>
    </p>
</body>
</html>

```

**Output:****Frames in HTML:**

- With frames, we can display more than one HTML document in the same browser window. Each HTML document is called a frame, and each frame is independent of the others. A collection of frames in the browser window is known as a frameset.
- The frameset, frame, and noframes tags are supported in previous versions of HTML such as HTML 4.01 and not supported in HTML5.

**Example for Vertical frames:**

```

<!DOCTYPE html>
<html>
    <frameset cols="50%,30%,20%">
        <frame src="frame1.html" />
        <frame src="frame2.html" />
        <frame src="frame3.html" />
    </frameset>
</html>

```

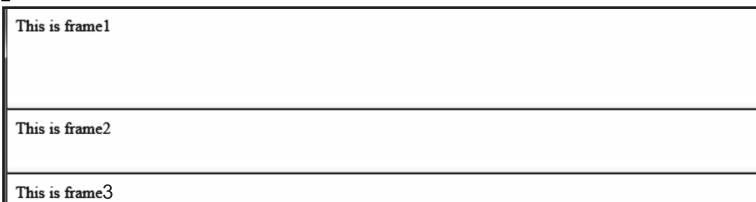
**Output:**

This is frame1	This is frame2	This is frame3
----------------	----------------	----------------

### Example for Horizontal frames:

```
<!DOCTYPE html>
<html>
  <frameset rows="50%,30%,20%">
    <frame src="frame1.html" />
    <frame src="frame2.html" />
    <frame src="frame3.html" />
  </frameset>
</html>
```

### Output:



- The `<iframe>` tag specifies an inline frame. An inline frame is used to embed another document within the current HTML document.
- The `<iframe>` tag defines a rectangular region within the document in which the browser displays a separate document, including scrollbars and borders.
- HTML5 has added some new attributes, and several HTML 4.01 attributes are removed from HTML5.

**Syntax:** `<iframe> ... </iframe>`

### Attributes for `<iframe>` tag:

Attributes	Value	Description
1. height	pixels	This attribute specifies the height of an <code>&lt;iframe&gt;</code> .
2. name	text	This attribute specifies the name of an <code>&lt;iframe&gt;</code> .
3. sandbox	allow-forms allow-pointer-lock allow-popups allow-same-origin allow-scripts allow-top-navigation	This attribute enables an extra set of restrictions for the content in an <code>&lt;iframe&gt;</code> .
4. src	URL	This attribute specifies the address of the document to embed in the <code>&lt;iframe&gt;</code> .
5. srcdoc	HTML_code	This attribute specifies the HTML content of the page to show in the <code>&lt;iframe&gt;</code>
6. width	pixels	This attribute specifies the width of an <code>&lt;iframe&gt;</code> .

### Example for inline frame:

```
<!DOCTYPE html>
<html>
<body>
    <iframe src="http://www.pragati.com">
        <p>Your browser does not support iframes.</p>
    </iframe>
</body>
</html>
```

### Output:



## 1.1.2 Tables in HTML

- Tables are made up of rows and columns. The HTML table model allows arranging data or information like text, images, links, forms, etc. into rows and columns of cells.
- The tables represents tabular data i.e., information/data presented in a two-dimensional table comprised of rows and columns of cells containing data

### HTML <table> Tag:

- The <table> tag defines an HTML table. An HTML table consists of the <table> element and one or more <tr>, <th>, and <td> elements.
- The <tr> element defines a table row, the <th> element defines a table header, and the <td> element defines a table cell.

**Syntax:** <table>.....</table>

Attributes	Value	Description
1. align	left center right	This attribute specifies the alignment of a table according to surrounding text.
2. bgcolor	rgb(x,x,x) #xxxxxx colorname	This attribute specifies the background color for a table.

**contd. ...**

3. border	pixels	This attribute specifies the width of the borders around a table.
4. cellpadding	pixels	This attribute specifies the space between the cell wall and the cell content.
5. cellspacing	pixels	This attribute specifies the space between cells.
6. frame	void above below hsides lhs rhs vsides box border	This attribute specifies which parts of the outside borders that should be visible.
7. rules	none groups rows cols all	This attribute specifies which parts of the inside borders that should be visible.
8. summary	text	This attribute specifies a summary of the content of a table.
9. width	pixels %	This attribute specifies the width of a table.

### Example for <table> tag:

```
<!DOCTYPE html>
<html>
  <body>
    <p>
      Each table starts with a table tag.
      Each table row starts with a tr tag.
      Each table data starts with a td tag.
    </p>
    <h4>One column:</h4>
    <table border="1">
      <tr>
        <td>100</td>
      </tr>
    </table>
```

```

<h4>One row and three columns:</h4>
<table border="1">
  <tr>
    <td>100</td>
    <td>200</td>
    <td>300</td>
  </tr>
</table>

<h4>Two rows and three columns:</h4>
<table border="1">
  <tr>
    <td>100</td>
    <td>200</td>
    <td>300</td>
  </tr>
  <tr>
    <td>400</td>
    <td>500</td>
    <td>600</td>
  </tr>
</table>
</body>
</html>

```

**Output:**

Each table starts with a table tag. Each table row starts with a tr tag. Each table data starts with a td tag.						
<b>One column:</b>						
<table border="1"><tr><td>100</td></tr></table>	100					
100						
<b>One row and three columns:</b>						
<table border="1"><tr><td>100</td><td>200</td><td>300</td></tr></table>	100	200	300			
100	200	300				
<b>Two rows and three columns:</b>						
<table border="1"><tr><td>100</td><td>200</td><td>300</td></tr><tr><td>400</td><td>500</td><td>600</td></tr></table>	100	200	300	400	500	600
100	200	300				
400	500	600				

- Table heading can be defined using `<th>` tag or the `<th>` tag defines a header cell in an HTML table. An HTML table has two kinds of cells:
  - Header Cells:** Contains header information (created with the `<th>` element), and
  - Standard Cells:** Contains data (created with the `<td>` element).

- The `<tr>` tag defines a row in an HTML table. A `<tr>` element contains one or more `<th>` or `<td>` elements.
- The `<td>` tag defines a standard cell in an HTML table. The `<td>` tag is used to mark up individual cells inside a table row.
- The `<caption>` tag defines a table caption. The `<caption>` tag is used to provide a caption for a table. This caption can either appear above or below the table.

### Example for various table tags:

```
<!DOCTYPE html>
<html>
<body>
<table border="1">
<caption>Monthly Savings</caption>
<tr>
<th>Month</th>
<th>Savings</th>
</tr>
<tr>
<td>January</td>
<td>Rs. 1000</td>
</tr>
<tr>
<td>February</td>
<td> Rs. 1500</td>
</tr>
</table>
</body>
</html>
```

### Output:

Monthly Savings	
Month	Savings
January	Rs. 1000
February	Rs. 1500

- Tables can be divided into three portions namely a header, a body, and a foot. The head and foot are rather similar to headers and footers in a word-processed document that remain the same for every page, while the body is the main content of the table.

- The three elements for separating the head, body, and foot of a table are:
    - **<thead>**: To create a separate table header.
    - **<tbody>**: To indicate the main body of the table.
    - **<tfoot>**: To create a separate table footer.
- 

**Example:**

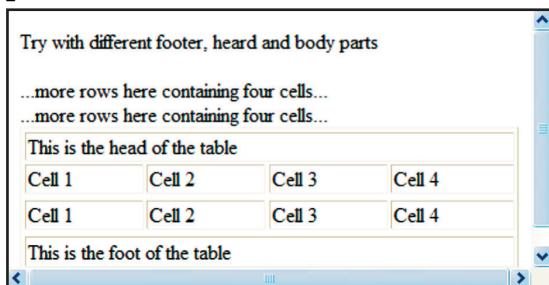
```
<!DOCTYPE html>
<html>
    <head>
        <title>Practice HTML table footer, header and body</title>
    </head>
    <body>
        <p>Try with different footer, heard and body parts</p>
        <table border="1" width="100%">
            <thead>
                <tr>
                    <td colspan="4">This is the head of the table</td>
                </tr>
            </thead>
            <tfoot>
                <tr>
                    <td colspan="4">This is the foot of the table</td>
                </tr>
            </tfoot>
            <tbody>
                <tr>
                    <td>Cell 1</td>
                    <td>Cell 2</td>
                    <td>Cell 3</td>
                    <td>Cell 4</td>
                </tr>
                <tr>
                    ...more rows here containing four cells...
                </tr>
            </tbody>
```

---

```

<tbody>
    <tr>
        <td>Cell 1</td>
        <td>Cell 2</td>
        <td>Cell 3</td>
        <td>Cell 4</td>
    </tr>
    <tr>
        ...more rows here containing four cells...
    </tr>
</tbody>
</table>
</body>
</html>

```

**Output:**

- Cellspacing attribute defines the width of the border, while cellpadding attribute represents the distance between cell borders and the content within.

**Example for cellspacing and cellpadding:**

```

<!DOCTYPE html>
<html>
    <head>
        <title>HTML Table Cellpadding</title>
    </head>
    <body>
        <table border = "1" cellpadding = "10" cellspacing = "10">
            <tr>
                <th>Name</th>
                <th>Salary</th>
            </tr>
            <tr>

```

```

        <td>Ramesh Salvi</td>
        <td>5000</td>
    </tr>
    <tr>
        <td>Amar Dube</td>
        <td>7000</td>
    </tr>
</table>
</body>
</html>

```

**Output:**

Name	Salary
Ramesh Salvi	5000
Amar Dube	7000

- We will use colspan attribute if we want to merge two or more columns into a single column. Similar way we will use rowspan if we want to merge two or more rows.

**Example for colspan and rowspan attributes:**

```

<!DOCTYPE html>
<html>
    <head>
        <title>Practice HTML table spans</title>
    </head>
    <body>
        <p>Merge two or more rows or columns and then see the result:</p>
        <table border="1">
            <tr>
                <th>Column 1</th>
                <th>Column 2</th>
                <th>Column 3</th>
            </tr>
            <tr><td rowspan="2">Row 1 Cell 1</td>
                <td>Row 1 Cell 2</td><td>Row 1 Cell 3</td></tr>
                <tr><td>Row 2 Cell 2</td><td>Row 2 Cell 3</td></tr>
            <tr><td colspan="3">Row 3 Cell 1</td></tr>
        </table>
    </body>
</html>

```

**Output:**

Merge two or more rows or columns and then see the result:

Column 1	Column 2	Column 3
Row 1 Cell 1	Row 1 Cell 2	Row 1 Cell 3
Row 2 Cell 1	Row 2 Cell 2	Row 2 Cell 3
Row 3 Cell 1		

**Nested Tables:**

- We can use one table inside another table is called as nesting of tables (or nested tables). Not only tables we can use almost all the tags inside table data tag <td>.

**Example for nested tables:**

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML Nested table</title>
  </head>
  <body>
    <table border="1">
      <tr>
        <td>
          <table border="1">
            <tr>
              <th>Name</th>
              <th>Salary</th>
            </tr>
            <tr>
              <td>Ramesh Salvi</td>
              <td>5,700</td>
            </tr>
            <tr>
              <td>Amar Dubey</td>
              <td>7,500</td>
            </tr>
          </table>
        </td>
        <td>
          <ul>
            <li>This is another cell</li>
            <li>Using list inside this cell</li>
          </ul>
        </td>
      </tr>
    </table>
  </body>
</html>
```

```

</td>
</tr>
<tr>
    <td>Row 2, Column 1</td>
    <td>Row 2, Column 2</td>
</tr>
</table>
</body>
</html>

```

**Output:**

Name	Salary	<ul style="list-style-type: none"> <li>• This is another cell</li> <li>• Using list inside this cell</li> </ul>
Ramesh Salvi	5,700	
Amar Dubे	7,500	
Row 2, Column 1		Row 2, Column 2

**1.1.3 HTML Forms**

- Forms are an essential part of the Internet, as they provide a way for websites to capture or collect information from users and to process requests.
- HTML forms are one of the main elements of interaction between a user and a website. An HTML form is used to collect user input. The user input is most often sent to a server for processing.
- A form is a group of controls that the user interacts with and sends the result to specific files as designed by an application developer.
- A form can contain input elements like text fields, checkboxes, radio-buttons, submit buttons and more. A form can also contain select lists, textarea, fieldset, legend, and label elements.
- The HTML `<form>` element is used to create an HTML form for user input.

**Syntax:**

```

<form>
.
.
.
input elements ...
.
.
.
</form>

```

**Attributes:**

Attributes	Value	Description
1. accept	file_type	This attribute specifies a comma-separated list of file types that the server accepts (that can be submitted through the file upload).
2. accept-charset	character_set	This attribute specifies the character encodings that are to be used for the form submission.
3. action	URL	This attribute specifies where to send the form-data when a form is submitted.
4. autocomplete	on off	This attribute specifies whether a form should have autocomplete on or off.
5. enctype	application/x-www-form-urlencoded multipart/form-data text/plain	This attribute specifies how the form-data should be encoded when submitting it to the server (only for method="post").
6. method	get post	This attribute specifies the HTTP method to use when sending form-data.
7. name	text	This attribute specifies the name of a form.
8. novalidate	novalidate	This attribute specifies that the form should not be validated when submitted.
9. target	_blank _self _parent _top	<p>This attribute specifies where to display the response that is received after submitting the form.</p> <p>Target to open the given URL:</p> <ul style="list-style-type: none"> <li>_blank : The target URL will open in a new window.</li> <li>_self : The target URL will open in the same frame as it was clicked.</li> <li>_parent : The target URL will open in the parent frameset.</li> <li>_top : The target URL will open in the full body of the window.</li> </ul>

- Users interact with forms through named controls. A control's "control name" is given by its name attribute. The <form> element can contain one or more of the form elements: <input>, <textarea>, <button>, <select>, <option>, <fieldset>, <label> etc.

#### HTML <input> Tag:

- The most important form element is the <input> tag which defines an input control.
- The <input> tag are used within a <form> element to declare input controls that allow users to input data. An input field can vary in many ways, depending on the type attribute.

**Syntax:** <input type= " " >

#### Attributes:

Attributes	Value	Description
1. accept	audio/* video/* image/* MIME_type	This attribute specifies the types of files that the server accepts (only for type="file").
2. align	left right top middle bottom	This attribute specifies the alignment of an image input (only for type="image").
3. alt	text	This attribute specifies an alternate text for an image (only for type="image").
4. checked	checked	This attribute specifies that an <input> element should be preselected when the page loads (for type="checkbox" or type="radio").
5. disabled	disabled	This attribute specifies that an <input> element should be disabled.
6. maxlength	number	This attribute specifies the maximum number of characters allowed in an <input> element.
7. name	name	This attribute specifies the name of an <input> element.
8. readonly	readonly	This attribute specifies that an input field should be read-only.
9. size	number	This attribute specifies the width, in characters, of an <input> element.

*contd. ...*

10. src	URL	This attribute specifies the URL of the image to use as a submit button (only for type="image").
11. type	button checkbox file hidden image password radio reset submit text	This attribute specifies the type of <input> element.
12. value	text	This attribute specifies the value of an <input> element.

- Various input fields are explained below:

### 1. Text Field:

Text fields are one line areas that allow the user to input text.

**Syntax:** <input type = "text">

Following is the list of attributes for <input> tag.

- (i) **type:** Indicates the type of input control you want to create. This element is also used to create other form controls such as radio buttons and checkboxes.
- (ii) **name:** Used to give the name part of the name/value pair that is sent to the server, representing each form control and the value the user entered.
- (iii) **value:** Provides an initial value for the text input control that the user will see when the form loads.
- (iv) **size:** Allows you to specify the width of the text-input control in terms of characters.
- (v) **maxlength:** Allows you to specify the maximum number of characters a user can enter into the text box.

---

#### Example for text field:

```
<!DOCTYPE html>
<html>
  <head>
    <body>
      <title>Concepts of Form</title>
      <form action="/cgi-bin/hello_get.cgi" method="get">
```

---

```

First name: <input type="text" name="first_name" />
<br>
Last name: <input type="text" name="last_name" />
<br>
<input type="submit" value="submit" />
</form>
</body>
</head>
</html>

```

**Output:**

The screenshot shows a simple HTML form. It contains two text input fields, one labeled "First name:" and another labeled "Last name:". Below these fields is a blue rectangular button labeled "submit". To the right of the input fields is a vertical scroll bar.

2. **Password Field:** Password fields are similar to text fields. The difference is that what is entered into a password field shows up a dot on the screen. This is, of course, to prevent others from reading the password on the screen.

**Syntax:** <input type = "password">

Attribute	Explanation
1. size=	Characters shown.
2. maxlength=	Max characters allowed.
3. name=	Name of the field.
4. value=	Initial value in the field.
5. align=	Alignment of the field.
6. tabindex=	Tab order of the field.

**Example for password field:**

```

<!DOCTYPE html>
<html>
<head>
    <title>MyPage</title>
</head>
<body>
    <form name="myform" action=http://www.mydomain.com/myformhandler.cgi
          method="POST">

```

```

<div align="center">
    Enter Password: <input type="password" size="25">
    <br><br>
</div>
</form>
</body>
</html>

```

**Output:**

Enter Password:

- 
3. **<label> Tag:** The <label> tag defines a label for an <input> element.
  4. **<fieldset> Tag:** <fieldset> tag defines a border around elements in a form. The <fieldset> tag is used to group related elements in a form. The <fieldset> tag draws a box around the related elements.

**Attributes:**

Attribute	Value	Description
1. disabled	disabled	This attribute specifies that a group of related form elements should be disabled.
2. form	form_id	This attribute specifies one or more forms the field set belongs to.
3. name	text	This attribute specifies a name for the field set.

**Example for <fieldset> tag:** The <legend> tag defines a caption for a <fieldset> element.

```

<!DOCTYPE html>
<html>
    <body>
        <form>
            <fieldset>
                <legend><b>User Information:</b></legend>
                Name: <input type="text"><br>
                Email: <input type="text"><br>
                Date of birth: <input type="text">
            </fieldset>
        </form>
    </body>
</html>

```

**Output:**

User Information:

Name:	<input type="text"/>
Email:	<input type="text"/>
Date of birth:	<input type="text"/>

5. **Hidden Field:** Hidden fields are similar to text fields. The difference is that the hidden field does not show on the page. Therefore the visitor cannot type anything into a hidden field, which leads to the purpose of the field.

**Syntax:** <input type = "hidden">

Attribute	Explanation
1. name=	Name of the field.
2. value=	Value of the field.

**Example for hidden field:**

```
<!DOCTYPE html>
<html>
  <head>
    <title>MyPage</title>
  </head>
  <body>
    <form name="myform" action=http://www.mydomain.com/myformhandler.cgi
          method="POST">
      <div align="center">
        <input type="text" size="25" value="Enter your name here!">
        <input type="hidden" name="Language" value="English">
      <br><br>
    </div>
    </form>
  </body>
</html>
```

**Output:**

6. **<textarea> Tag:** It defines a multi-line text input control. The size of a text area is specified by the cols and rows attributes.

**Syntax:** <textarea>.....</textarea>

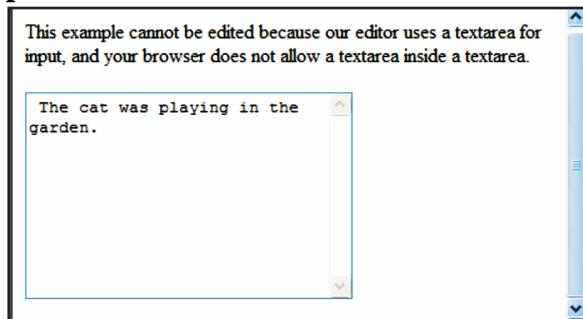
Attributes for <textarea> tag are given below:

- (i) **name**: The name of the control. This is used in the name/value pair that is sent to the server.
  - (ii) **rows**: Indicates the number of rows of text area box.
  - (iii) **cols**: Indicates the number of columns of text area box.
- 

#### Example for <textarea> tag:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Concepts of Form</title>
    <body>
        <p> This example cannot be edited because our editor uses a textarea
            for input, and your browser does not allow a textarea inside a
            textarea. </p>
        <textarea rows="10" cols="30"> The cat was playing in the garden.
        </textarea>
    </body>
</head>
</html>
```

#### Output:



- 
- 7. **Radio Buttons:** Radio buttons are used when only one option is required to be selected. They are created using <input> tag.

**Syntax:** <input type="radio">

Following is the list of important radiobox attributes:

- (i) **type**: Indicates that you want to create a radiobox.
  - (ii) **name**: Name of the control.
  - (iii) **value**: Used to indicate the value that will be sent to the server if this option is selected.
  - (iv) **checked**: Indicates that this option should be selected by default when the page loads.
-

**Example for radio button:**

```
<!DOCTYPE html>
<html>
    <head>
        <title>Practice HTML input radiobox tag</title>
    </head>
    <body>
        <form action="/cgi-bin/radiobutton.cgi" method="post">
            <input type="radio" name="subject" value="maths" /> Maths
            <input type="radio" name="subject" value="physics" /> Physics
            <input type="submit" value="Select Subject" />
        </form>
    </body>
</html>
```

**Output:**

A screenshot of a web browser window. Inside the window, there is a form containing two radio buttons. The first radio button is labeled "Maths" and the second is labeled "Physics". Both radio buttons have a blue outline and are currently unselected. To the right of the radio buttons is a blue rectangular button with the text "Select Subject" in white.

- 
8. **Checkbox Control:** Checkboxes are used when more than one option is required to be selected. They are created using `<input>` tag.

**Syntax:** `<input type="checkbox">`

Following is the list of important checkbox attributes:

- (i) `type`: Indicates that you want to create a checkbox.
  - (ii) `name`: Name of the control.
  - (iii) `value`: The value that will be used if the checkbox is selected. More than one checkbox should share the same name only if you want to allow users to select several items from the same list.
  - (iv) `checked`: Indicates that when the page loads, the checkbox should be selected.
- 

**Example for checkboxes:**

```
<!DOCTYPE html>
<html>
    <head>
        <title>Practice HTML input checkbox tag</title>
    </head>
    <body>
```

```

<p>Try with different checkbox and different attributes</p>
<form action="" method="get">
    <input type="checkbox" name="maths" value="on" /> Maths
    <input type="checkbox" name="physics" value="on" /> Physics
    <input type="reset" value="Select Subject" />
</form>
</body>
</html>

```

**Output:**

Try with different checkbox and different attributes

Maths  Physics

9. **<select> Tag:** The <select> tag is used to create a drop-down list. Drop-down list is used when we have many options available to be selected but only one or two will be selected. The <option> tags inside the <select> element define the available options in the list.

**Syntax:** <select>.....</select>

Attributes	Value	Description
1. disabled	disabled	This attribute specifies that a drop-down list should be disabled.
2. multiple	multiple	This attribute specifies that multiple options can be selected at once.
3. name	name	This attribute defines a name for the drop-down list.
4. size	number	This attribute specifies defines the number of visible options in a drop-down list.

**Example for <select> tag:**

```

<!DOCTYPE html>
<html>
    <head>
        <title>Practice HTML input selectbox tag</title>
    </head>
    <body>
        <p>Try with different selectbox and different attributes</p>

```

```

<form action="" method="get">
    <select name="dropdown">
        <option value="Maths" selected>Maths</option>
        <option value="Physics">Physics</option>
    </select>
    <input type="reset" value="reset" />
</form>
</body>
</html>

```

**Output:**

Try with different selectbox and different attributes

- 10. Creating Button:** The `<button>` tag defines a push button. We can create clickable button using `<input>` tag.

**Syntax:** `<button>.....</button>`

Attributes	Value	Description
1. <code>disabled</code>	<code>disabled</code>	This attribute specifies that a button should be disabled.
2. <code>name</code>	<code>name</code>	This attribute specifies the name for a button.
3. <code>type</code>	<code>button</code> <code>reset</code> <code>submit</code>	This attribute specifies the type of button. The submit value creates a button that automatically submits a form. The reset value creates a button that automatically resets form controls to their initial values. The button value creates a button that is used to trigger a client-side script when the user clicks that button.
4. <code>value</code>	<code>text</code>	This attribute specifies the initial value for a button.

**Example for `<button>` tag:**

```

<!DOCTYPE html>
<html>
    <body>
        <form name="input" action="html_form_action.asp" method="get">
            First name: <input type="text" name="FirstName"
                value="Mickey" /><br />
            Last name: <input type="text" name="LastName" value="Mouse" /><br />
                <input type="submit" value="Submit" />
        </form>
        <p>If you click the "Submit" button, the form-data will be sent
            to a page called "html_form_action.asp".</p>
    </body>
</html>

```

**Output:**

First name: Mickey  
 Last name: Mouse

If you click the "Submit" button, the form-data will be sent to a page called "html\_form\_action.asp".

**Example:**

```
<!DOCTYPE html>
<html>
  <body>
    <form action="/cgi-bin/hello_get.cgi" method="get">
      First name:
      <input type="text" name="first_name" /><br>
      Last name:
      <input type="text" name="last_name" />
      <input type="submit" value="Submit" />
      <input type="reset" value="Reset" />
    </form>
  </body>
</html>
```

**Output:**

First name:   
 Last name:

- 11. <option> Tag:** It defines an option in a select list. The <option> tag go inside a <select> element.

**Syntax:** <option>.....</option>

Attributes	Value	Description
1. disabled	disabled	This attribute specifies that an option should be disabled.
2. label	text	This attribute specifies a shorter label for an option.
3. selected	selected	This attribute specifies that an option should be pre-selected when the page loads.
4. value	text	This attribute specifies the value to be sent to a server.

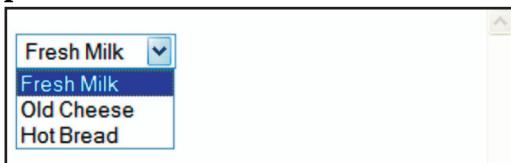
**Example for <option> tag:**

```
<!DOCTYPE html>
<html>
  <body>
    <form name="myform" action="nextpage.html" method="post" >
```

```

<select size=1 name="mydropdown">
    <option value="milk">Fresh Milk</option><br>
    <option value="cheese">Old Cheese</option><br>
    <option value="bread">Hot Bread</option>
</select>
</form>
</body>
</html>

```

**Output:****Example for form using various controls:**

```

<!DOCTYPE html>
<html>
    <head>
        <h3><b> EX. FORM FOR RADIO BUTTON, CHECKBOXES & PULL DOWN MENU
        </b></h3>
    </head>
    <body>
        <form action="nextpage.html" method ="post">
            <h3>Enter your information:</h3>
            <b><i>Enter your name:</i></b>
            <input type=text size="25"><br>
            <b><i>Your age is in between:</i></b><br>
            <input type="radio" name="radios" value="radio1" checked>15-20 yrs<br>
            <input type="radio" name="radios" value="radio2">20-25 yrs<br>
            <input type="radio" name="radios" value="radio3">25 yrs and above<br>
            <br><b><i>
            Your Hobbies
            </i></b>
            <input type=checkbox name="option" value="read">Reading novels <br>
            <input type="checkbox" name="option" value="write" checked> Writing <br>
            <input type="checkbox" name="option" value="sing">Singing <br>
            <br><br><b><i> Enter your city: </i></b>
            <select>
                <option value=pune> Pune </option>
                <option value=mumbai> Mumbai</option>
                <option value=a'nagar> Ahmednagar </option>

```

```

</select><br><br>
<input type="submit" name="button" value="Submit data">
</form>
</body>
</html>

```

**Output:**

EX. FORM FOR RADIO BUTTON, CHECKBOXES & PULL DOWN MENU

Enter your information :

Enter your name :

Your age is in between :

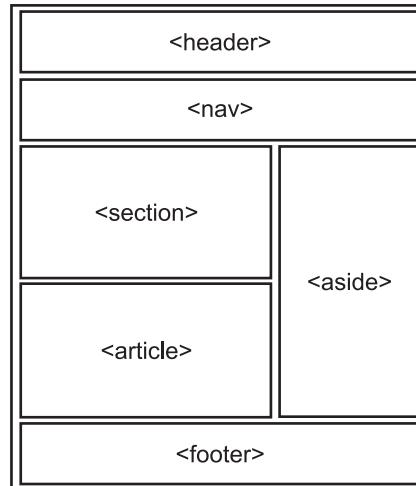
15-20 yrs  
 20-25 yrs  
 25 yrs and above

Your Hobbies  Reading novels  
 Writing  
 Singing

Enter your city :

**1.1.4 HTML5 Semantics**

- HTML5 is the latest version of HTML. Semantics is the study of the meanings of words and phrases in a language.
- A semantic element clearly describes its meaning to both the browser and the developer.
- HTML5 offers new semantic elements to define different parts of a web page as shown in Fig. 1.6. HTML5 semantic elements are supported in all modern browsers.

**Fig. 1.6**

- Various semantic elements in HTML5 are explained below:
- 1. HTML5 <article> Element:**
- The <article> element is used to represent an article. The <article> element specifies independent, self-contained content.

- Examples of article content could include a forum post, a newspaper article, a blog entry, or a user-submitted comment.

**Syntax:** <article>....</article>

---

**Example for <article> element:**

```
<!DOCTYPE html>
<html>
  <body>
    <article>
      <h1>Google Chrome</h1>
      <p>Google Chrome is a free, open-source web browser developed
         by Google.</p>
    </article>
    <p><strong>Note:</strong> The article tag is not supported in
       Internet Explorer 8 and earlier versions.</p>
  </body>
</html>
```

**Output:**

**Google Chrome**

Google Chrome is a free, open-source web browser developed by Google.

**Note:** The article tag is not supported in Internet Explorer 8 and earlier versions.

---

**2. HTML5 <nav> Element:**

- The <nav> element is new in HTML5. The HTML <nav> element is used for declaring a navigational section of the HTML document.
- The <nav> element defines a set of navigation links.

**Syntax:** <nav>...</nav>

---

**Example for <nav> element:**

```
<!DOCTYPE html>
<html>
  <body>
    <nav>
      <a href="/html/">Photoshop</a> |
      <a href="/html/">HTML</a> |
      <a href="/css/">CSS</a> |
      <a href="/js/">JavaScript</a> |
```

```

<a href="/html/">Dreamweaver</a> |
<a href="/html/">Ajax</a> |
<a href="/jquery/">jQuery</a>
</nav>
<p><strong>Note:</strong> The nav tag is not supported in
Internet Explorer 8 and earlier versions.</p>
</body>
</html>
```

**Output:****3. HTML5 <section> Element:**

- The <section> element is used to represent a section within an article.
- The <section> element defines sections in a document, such as chapters, headers, footers, or any other sections of the document.

**Syntax:** <section>...</section>

**Example for <section> element:**

```

<!DOCTYPE html>
<html>
<body>
<section>
<h1>NIRALI PRAKASHAN</h1>
<p>Nirali Prakashan specializes in quality text books from Std II to Postgraduate levels. These books are written as per the syllabus of Pune, Mumbai, Shivaji, Goa, North Maharashtra & Marathwada Universities, by eminent and experienced authors in their subjects. Besides textbooks, these books are also used as reference books by academicians and researchers, because of the quality of contents. Most of the books are also prescribed as basic texts by several universities and thus, are widely used by the student community. The subjects covered are Management, Engineering, Pharmacy and Computer Science, Arts and Commerce, to name a few.
Apart from publishing, Nirali has developed a wide network of dealers & distributors to make their books readily available to a large section of student community, all over India.
```

```

With its success in publishing, in a little over 25 years,
Nirali has now been able to extend its activities abroad. </p>
</section>
<section>
    <h1>CONTACT US </h1>
    <p>Pune Office:<br>
        Abhyudaya Pragati, 1312, Shivaji Nagar,<br>
        Off. J.M. Road Pune 411005,<br>
        Maharashtra, India<br>
        Tel: (+91-020) 25512336 / 7 / 9<br>
        Fax: (+91-020) 25511379<br>
        Email: niralipune@pragationline.com</p>
</section>
<p><strong>Note:</strong> The section tag is not supported
in Internet Explorer 8 and earlier versions.</p>
</body>
</html>

```

## Output:

### NIRALI PRAKASHAN

Nirali Prakashan specializes in quality text books from Std II to Postgraduate levels. These books are written as per the syllabus of Pune, Mumbai, Shivaji, Goa, North Maharashtra & Marathwada Universities, by eminent and experienced authors in their subjects. Besides textbooks, these books are also used as reference books by academicians and researchers, because of the quality of contents. Most of the books are also prescribed as basic texts by several universities and thus, are widely used by the student community. The subjects covered are Management, Engineering, Pharmacy and Computer Science, Arts and Commerce, to name a few. Apart from publishing, Nirali has developed a wide network of dealers & distributors to make their books readily available to a large section of student community, all over India. With its success in publishing, in a little over 25 years, Nirali has now been able to extend its activities abroad.

### CONTACT US

Pune Office:  
 Abhyudaya Pragati, 1312, Shivaji Nagar,  
 Off. J.M. Road Pune 411005,  
 Maharashtra, India  
 Tel: (+91-020) 25512336 / 7 / 9  
 Fax: (+91-020) 25511379  
 Email: niralipune@pragationline.com

**Note:** The section tag is not supported in Internet Explorer 8 and earlier versions.

#### 4. HTML5 <header> Element:

- It specifies a header for a document or section. The <header> element should be used as a container for introductory content.

**Syntax:** <header>...</header>

#### 5. HTML5 <footer> Element:

- It specifies footer for a document or section. A <footer> element should contain information about its containing element.
- Footer typically contains the author of the document, copyright information, links to terms of use, contact information, etc.

**Syntax:** <footer>...</footer>

**Example for <header> and <footer> elements:**

```
<!DOCTYPE html>
<html>
    <head></head>
    <body>
        <header>
            <span style="color:brown;font-style:italic;">
                Nirali Prakashan: Publication Firm</span>
            <hr>
            <h1>Engineering Books</h1>
            <h4>Computer <br>
                IT <br>
                Civil <br></h4>
        </header>
        <nav></nav>
        <main>
            <article>
                <p>Nirali Prakashan is a Text book Publication firm
                    since 30 years.</p>
            </article>
        </main>
        <footer>
            <hr>
            © 2016 Nirali Prakashan
        </footer>
    </body>
</html>
```

**Output:**

**6. HTML5 <aside> Element:**

- The <aside> element defines some content aside from the content it is placed in.
- The <aside> element is used to represent content that is related to the surrounding content within an article or web page, but could still stand alone in its own right. This type of content is often represented in sidebars.

**Syntax:** <aside>...</aside>

---

**Example for <aside> element:**

```
<!DOCTYPE html>
<html>
  <body>
    <p>My family and I visited Queenstown </p>
    <aside>
      <h4> Queenstown </h4>
      <p>The Queenstown is a city of New Zealand.</p>
    </aside>
    <p><strong>Note:</strong> The aside tag is not supported in
       Internet Explorer 8 and earlier versions.</p>
  </body>
</html>
```

**Output:**

My family and I visited Queenstown

**Queenstown**

The Queenstown is a city of New Zealand.

**Note:** The aside tag is not supported in Internet Explorer 8 and earlier versions.

---

**7. HTML5 <details> Element:**

- The HTML <details> element specifies additional details that the user can view or hide on demand.
- It can be used in conjunction with the HTML5 <summary> element to provide a heading that can be clicked on to expand/collapse the details as required.

**Syntax:** <details>...</details>

**8. HTML5 <summary> Element:**

- The <summary> element specifies a summary/caption that can be used in conjunction with the HTML5 <details> element.
- This summary/caption can be clicked on to expand/collapse the details as required. The <summary> element, if used, should be the first child of the <details> element.

**Syntax:** <summary>...</summary>

---

### Example for <details> and <summary> elements:

```
<!DOCTYPE html>
<html>
  <body>
    <details>
      <summary>Copyright 1999-2014.</summary>
      <p> - by Nirali Prakashan.</p>
      <p>All content and graphics on this web site are the property
         of the company Nirali Prakanan.</p>
    </details>
    <p><b>Note:</b> The details tag is not supported in Internet
       Explorer.</p>
  </body>
</html>
```

### Output:

► Copyright 1999-2014.  
**Note:** The details tag is not supported in Internet Explorer.

▼ Copyright 1999-2014.  
- by Nirali Prakashan.  
All content and graphics on this web site are the property of the company Nirali Prakanan.  
**Note:** The details tag is not supported in Internet Explorer.

### Example for HTML5 semantics:

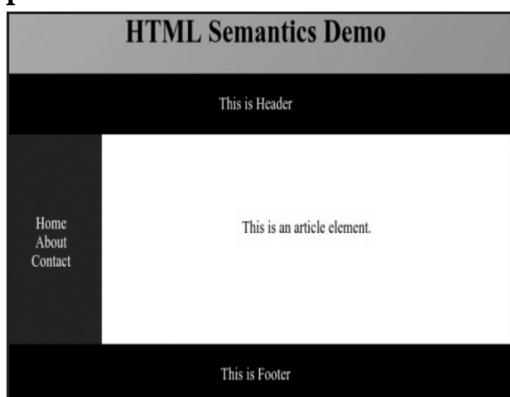
```
<!DOCTYPE html>
<html>
  <style>
    * {
      box-sizing: border-box;
    }
    body {
      color: #000;
      background-color: #8BC6EC;
      background-image: linear-gradient(135deg, #8BC6EC 0%, #9599E2 100%);
      text-align: center;
    }
    header {
      background-color: #000;
      padding: 20px;
      text-align: center;
      color: white;
    }
  </style>
</html>
```

```
nav {  
    float: left;  
    width: 20%;  
    height: 200px;  
    background: #282828;  
    padding: 60px 10px;  
}  
nav ul {  
    list-style-type: none;  
    padding: 0;  
}  
nav ul li a {  
    text-decoration: none;  
    color: #fff;  
}  
article {  
    float: left;  
    padding: 80px 10px;  
    width: 80%;  
    background-color: #fff;  
    height: 200px;  
    text-align: center;  
}  
section:after {  
    content: "";  
    display: table;  
    clear: both;  
}  
footer {  
    background-color: #000;  
    padding: 20px;  
    text-align: center;  
    color: white;  
}  
</style>  
<body>  
    <h1>HTML Semantics Demo</h1>  
    <header>This is Header</header>
```

```

<section>
  <nav>
    <ul>
      <li><a href="#">Home</a></li>
      <li><a href="#">About</a></li>
      <li><a href="#">Contact</a></li>
    </ul>
  </nav>
  <article>This is an article element.</article>
</section>
  <footer>This is Footer</footer>
</body>
</html>

```

**Output:****Embedding Multimedia in HTML:**

- Multimedia is a combination of more than one media like text, graphics, images, audio, or video, which is used for presenting, sharing, and disseminating the information.
- The `<embed>` tag allows us to add Multimedia like sound, music and video files to the web pages.

**Syntax:** `<embed>...</embed>`

**Attributes:**

Attribute	Description
1. align	This attribute determines how to align the object. It can be set to either center, left or right.
2. autostart	This Boolean attribute indicates if the media should start automatically. You can set it either true or false.

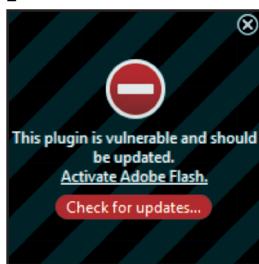
*contd. ...*

3. loop	This attribute specifies if the sound should be played continuously (set loop to true), a certain number of times (a positive value) or not at all (false).
4. playcount	This attribute specifies the number of times to play the sound. This is alternate option for loop if you are usiong IE.
5. hidden	This attribute specifies if the multimedia object should be shown on the page. A false value means no and true values means yes.
6. width	Width of the object in pixels.
7. height	Height of the object in pixels.
8. name	A name used to reference the object.
9. src	URL of the object to be embedded.
10. volume	This attribute controls volume of the sound. Can be from 0 (off) to 100 (full volume).

#### Example for <embed> tag:

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML embed Tag</title>
  </head>
  <body>
    <embed src="/html/yourfile.swf" width="200" height="200" >
    <noembed>
    </noembed>
  </embed>
  </body>
</html>
```

#### Output:



- The HTML <source> tag is used to specify multiple media resources on media elements such as <audio> and <video>.
- The <object> tag defines an embedded object within an HTML document. Use this element to embed multimedia (like audio, video, Java Applets, ActiveX, PDF, and Flash) in the web pages.
- The HTML <audio> tag is used to specify audio on an HTML document.

**Syntax:** <audio>...</audio>

Attribute	Value	Description
1. autoplay	autoplay	This attribute specifies that the audio will start playing as soon as it is ready.
2. controls	controls	This attribute specifies that audio controls should be displayed (such as a play/pause button etc).
3. loop	loop	This attribute specifies that the audio will start over again, every time it is finished.
4. muted	muted	This attribute specifies that the audio output should be muted.
5. preload	auto metadata none	This attribute specifies if and how the author thinks the audio should be loaded when the page loads.
6. src	URL	This attribute specifies the URL of the audio file.

**Example for <audio> tag:**

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Example</title>
  </head>
  <audio src="/music/good_enough.mp3" controls>
    <p>If you are reading this, it is because your browser does not support
       the audio element.</p>
  </audio>
</html>
```

**Output:**



- The <video> tag is used to specify video on an HTML document. The <video> tag specifies video, such as a movie clip or other video streams.

**Syntax:** <video>...</video>

**Attributes:**

Attribute	Value	Description
1. autoplay	autoplay	This attribute specifies that the video will start playing as soon as it is ready.
2. controls	controls	This attribute specifies that video controls should be displayed (such as a play/pause button etc).
3. height	pixels	This attribute sets the height of the video player.
4. loop	loop	This attribute specifies that the video will start over again, every time it is finished.
5. muted	muted	This attribute specifies that the audio output of the video should be muted.
6. poster	URL	This attribute specifies an image to be shown while the video is downloading, or until the user hits the play button.
7. preload	auto metadata none	This attribute specifies if and how the author thinks the video should be loaded when the page loads.
8. src	URL	This attribute specifies the URL of the video file.
9. width	pixels	This attribute sets the width of the video player.

**Example for <video> tag:**

```
<!DOCTYPE html>
<html>
  <body>
    <video width="320" height="240" controls>
      <source src="movie.mp4" type="video/mp4">
      <source src="movie.ogg" type="video/ogg">
      Your browser does not support the video tag.
    </video>
    <p><strong>Note:</strong> The video tag is not supported in Internet Explorer 8 and earlier versions.</p>
  </body>
</html>
```

**Output:**

## 1.2 CSS (CASCADING STYLE SHEET)

- CSS is a standard language used for describing the presentation (i.e. the layout and formatting) of the web pages.
- The purpose of CSS is beautifying web pages and increase look and feel of them. A Cascading Style Sheet (CSS) is a set of rules that define the layout of web pages or the way the content is presented.
- Cascading style sheets is a language used to describe the look and formatting of a web document written in a markup language.
- CSS's most common use is to style web pages written in HTML. These styles are set CSS properties.

**Advantages of CSS:**

1. **CSS Saves Time:** We can write CSS once and then reuse same sheet in multiple HTML pages. Using CSS we can define a style for each HTML element and apply it to as many Web pages as we want which saves time for code writing.
2. **Pages Load Faster:** If we are using CSS, we do not need to write HTML tag attributes every time. Just write one CSS rule of a tag and apply to all the occurrences of that tag. So less code means faster download times.
3. **Easy Maintenance:** To make a change, simply change the style, and all elements in all the web pages will be updated automatically.
4. **Superior Styles to HTML:** CSS has a much wider array of attributes than HTML so we can give far better look to the HTML page in comparison of HTML attributes.
5. **Multiple Device Compatibility:** Style sheets allow content to be optimized for more than one type of device. By using the same HTML document, different versions of a website can be presented for handheld devices such as PDAs and smart phones etc.
6. **Global Web Standards:** Now HTML attributes are being deprecated and it is being recommended to use CSS. So it's a good idea to start using CSS in all the HTML pages to make them compatible to future browsers.
7. **Control on Web Pages:** CSS is easy to learn and understand but it provides powerful control over the presentation of an HTML document.
8. **Browser Support:** CSS is supported by all the browsers and search engines.

### 1.2.1 Basic Concept of CSS

- Cascading Style Sheets (CSS) is a language used for describing the presentation of a web document written in a markup language such as HTML.
- CSS is used for formatting content on web pages and it is used to enhance the presentation of the web pages.
- CSS handles the look and feel part of a web page. The CSS intended to simplify the process of making web pages presentable.
- Using CSS, we can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, layout designs, and variations in display for different devices (desktop, laptop, smart phones etc.) and screen sizes as well as a variety of other effects.
- The goal of CSS is to be able to define the styles of the various page elements, separate from the content of the page.
- A CSS comprises of style rules that are interpreted by the browser and then applied to the corresponding elements in the web document.
- A style rule is made of three parts as shown in Fig. 1.7 and explained below:
  1. **Selector:** A selector is an HTML tag at which style will be applied. This could be any tag like `<h1>` or `<table>` etc.
  2. **Property:** A property is a type of attribute of HTML tag. Put simply, all the HTML attributes are converted into CSS properties. They could be color or border etc.
  3. **Value:** Values are assigned to properties. For example color property can have value either red or `#F1F1F1` etc.

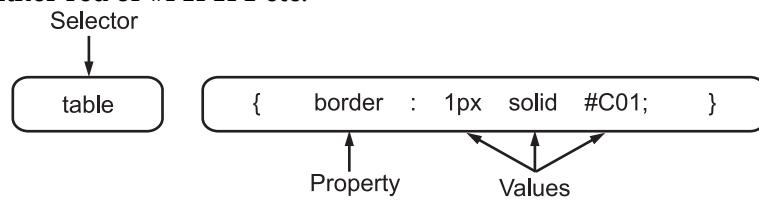


Fig. 1.7

**Syntax of CSS:** `selector { property: value }`

**Example:** We can define a table border as follows:

```
table{ border:1px solid #C00; }
```

Here, table is a selector and border is a property and given value 1px solid #C00 is the value of that property. We can define selectors in various simple ways based on our comfort.

- CSS selectors are used to "find" (or select) the HTML elements we want to style. Let us see these selectors one by one.

1. **Type Selectors:** A type selector sometimes referred to as an element selector selects HTML elements based on the element name such as `<p>`, `<h1>`, `<div>` and so on. The example to give a color to all level 1 headings like this:

```
h1 {
    color: #36CFFF;
}
```

2. **Universal Selectors:** Rather than selecting elements of a specific type, the universal selector quite simply matches the name of any element type. The universal selector (\*) selects all HTML elements on the page.

```
* {  
    color: #000000;  
}
```

This rule renders the content of every element in our document in black.

3. **Descendant Selectors:** The descendant selector matches all elements that are descendants of a specified element. Suppose we want to apply a style rule to a particular element only when it lies inside a particular element. As given in the following example, style rule will apply to `<em>` element only when it lies inside `<ul>` tag.

```
ul em {  
    color: #000000;  
}
```

4. **Class Selectors:** We can define style rules based on the class attribute of the elements. All the elements having that class will be formatted according to the defined rule. To select elements with a specific class, write a period (.) character, followed by the name of the class.

```
.black {  
    color: #000000;  
}
```

Above rule renders the content in black for every element with class attribute set to black in our document. We can make it a bit more particular. For example:

```
h1.black {  
    color: #000000;  
}
```

Above rule renders the content in black for only `<h1>` elements with class attribute set to black.

5. **ID Selectors:** We can define style rules based on the id attribute of the elements. All the elements having that id will be formatted according to the defined rule. To select an element with a specific id, write a hash (#) character, followed by the id of the element.

```
#black {  
    color: #000000;  
}
```

---

This rule renders the content in black for every element with id attribute set to black in our document. You can make it a bit more particular. For example:

```
h1#black {
    color: #000000;
}
```

This rule renders the content in black for only `<h1>` elements with id attribute set to black.

The true power of id selectors is when they are used as the foundation for descendant selectors, For example:

```
#black h2 {
    color: #000000;
}
```

In above example all level 2 headings will be displayed in black color only when those headings will lie within tags having id attribute set to black.

6. **Child Selectors:** The child selector selects all elements that are the children of a specified element. Consider the following example:

```
body > p {
    color: #000000;
}
```

This rule will render all the paragraphs in black if they are direct child of `<body>` element. Other paragraphs put inside other elements like `<div>` or `<td>` etc. would not have any effect of this rule.

7. **Attribute Selectors:** We can also apply styles to HTML elements with particular attributes. The attribute selector is used to select elements with a specified attribute or attribute value. The style rule below will match all input elements that have a type attribute with a value of text:

```
input[type="text"]{
    color: #000000;
}
```

The advantage to this method is that the `<input type="submit" />` element is unaffected, and the color applied only to the desired text fields.

There are following rules applied to attribute selector.

- o **p[lang]:** Selects all paragraph elements with a lang attribute.
- o **p[lang="fr"]:** Selects all paragraph elements whose lang attribute has a value of exactly "fr".
- o **p[lang~="fr"]:** Selects all paragraph elements whose lang attribute contains the word "fr".
- o **p[lang|= "en"]:** Selects all paragraph elements whose lang attribute contains values that are exactly "en", or begin with "en-".

**Multiple Style Rules:** We may need to define multiple style rules for a single element. We can define these rules to combine multiple properties and corresponding values into a single block as defined in the following example:

```
h1
{
    color: #36C;
    font-weight: normal;
    letter-spacing: .4em;
    margin-bottom: 1em;
    text-transform: lowercase;
}
```

Here, all the property and value pairs are separated by a semi colon (;). We can keep them in a single line or multiple lines.

For better readability we keep them into separate lines. For a while do not bother about the properties mentioned in the above block.

8. **Grouping Selectors:** The grouping selector selects all the HTML elements with the same style definitions. We can apply a style to many selectors if we like. Just separate the selectors with a comma as given in the following example:

```
h1, h2, h3 {
    color: #36C;
    font-weight: normal;
    letter-spacing: .4em;
    margin-bottom: 1em;
    text-transform: lowercase;
}
```

This define style rule will be applicable to h1, h2 and h3 element as well. The order of the list is irrelevant. All the elements in the selector will have the corresponding declarations applied to them.

We can combine various class selectors together as shown below:

```
#content, #footer, #supplement {
    position: absolute;
    left: 510px;
    width: 200px;
}
```

## 1.2.2 Three Ways to Use CSS

---

- When a browser reads a style sheet, it will format the HTML document according to the information in the style sheet.
- CSS can be inserted in three ways namely, Inline CSS, Internal CSS and External CSS.

### 1. Inline CSS:

- An inline CSS is used to apply a unique style to a single HTML element. In HTML document we can add inline CSS by using the style attribute inside HTML elements.
- An inline CSS we can use the style attribute of an HTML element to define style rules. These rules will be applied to that element only..
- The **syntax** for inline style sheet is given below:

```
<tag_name style="property1: value;  
                  property2: value;">  
</tag_name>
```

---

#### Example for inline CSS:

```
<!DOCTYPE html>  
<html>  
  <body>  
    <p style="background: pink; color: blue;">  
      Text color is green with blue background  
    </p>  
  </body>  
</html>
```

#### Output:

Text color is green with blue background

---

### 2. Embedded (Internal) CSS:

- An internal CSS is used to define a style for a single HTML page.
- An internal CSS is defined in the `<head>` section of an HTML page, within a `<style>` element followed by type attribute.
- The **syntax** for internal style sheet is given below:

```
<head>  
  <style type="text/css">  
    selector { property: value; }  
  </style>  
</head>
```

---

**Example for Internal/Embedded Stylesheet:**

```
<!DOCTYPE html>
<html>
  <head>
    <style type="text/css">
      p {
        color: blue;
      }
      h1 {
        color: red;
        text-align: center;
      }
      body {
        background-color: grey;
      }
    </style>
  </head>
  <body>
    <h1>Internal CSS Example</h1>
    <p> Now thats how CSS is inserted internally </p>
  </body>
</html>
```

---

- In the above example, we inserted CSS code in the head section, using style tag. This tells the browser that the code that follows will be style sheet code.
  - **Paragraph** text will be blue in color.
  - **heading (h1)** will be center aligned and text color will be red.
  - **body** background color will be grey.

**Output:****3. External CSS:**

- An external style sheet is ideal when the style is applied to many pages. With an external style sheet, we change the look of an entire web site by changing one file.
-

- External CSS contains only CSS code and is saved with a ".css" file extension. This CSS file is referred from HTML file using the <link> tag.
  - To use an external style sheet, add a <link> tag in the <head> section of each HTML page.
- 

**CSS Code:**

```
p {
    text-align: center;
    color: white;
}
body {
    background-color: green;
}
```

- Save the above file as sample.css
- 

**HTML Code:**

```
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" type="text/css" href="sample.css" />
    </head>
    <body>
        <p> This is an example for inserting CSS externally. </p>
    </body>
</html>
```

**Output:**

This is an example for inserting CSS externally.

**1.2.3 CSS Box Model**

- In CSS, the term “box model” is used when talking about design and layout. In CSS box model all HTML elements can be considered as box and each box has a specific purpose.
  - The CSS box model is essentially a box that wraps around every HTML element.
  - The Fig. 1.8 shows CSS box model. The CSS box model actually relies on a series of four boxes namely, content, padding, border and margin.
    - **Content** is the innermost box is where the text and images appear. Content box is the area where the content is displayed, which can be sized using properties like width and height.
-

- **Padding** is the second inner most box, sits outside of the content area. The padding is the space between the content area box and the border box of the HTML element. The size of padding box can be controlled using padding and related properties.
- **Border** is the second outer most box sits inside the margin, and surrounds the padding and content of the HTML element. The border box wraps the content and any padding. Its size and style can be controlled using border and related properties.
- **Margin** is the outermost box that consists of transparent space outside of the border of an element. The margin box wrapping the content, padding and border as whitespace between this box and other elements. Its size can be controlled using margin and related properties.

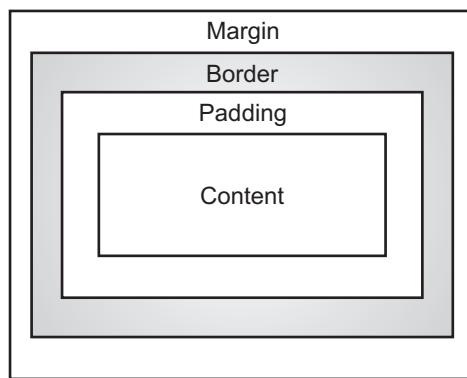


Fig. 1.8: CSS Box Model

#### 1.2.4 CSS Properties

- We style HTML elements via CSS properties. CSS properties are the styles used on specified selectors.
- CSS properties can be organized into CSS rules. A CSS rule groups a set of CSS properties together, and applies all properties to the HTML elements matched by the CSS rule.
- There are many CSS properties we can specify for different HTML elements. In short, a CSS property styles an aspect of an HTML element.
- A CSS property consists of a property name and a property value. The property name comes first, then a colon, and then the value such as `property_name : property_value`.
- If we specify more than one CSS property, the each name - value pair is separated by a semicolon as given below:

```

property_name1 : property_value1;
property_name2 : property_value2;
:
:
:
property_nameN : property_valueN;

```

**CSS Text Properties:** Defines how to manipulate text using CSS properties.

1. The **color** property is used to set the color of a text.
2. The **direction** property is used to set the text direction.
3. The **letter-spacing** property is used to add or subtract space between the letters that make up a word.
4. The **word-spacing** property is used to add or subtract space between the words of a sentence.
5. The **text-indent** property is used to indent the text of a paragraph.
6. The **text-align** property is used to align the text of a document.
7. The **text-decoration** property is used to underline, overline, and strikethrough text.
8. The **text-transform** property is used to capitalize text or convert text to uppercase or lowercase letters.
9. The **white-space** property is used to control the flow and formatting of text.
10. The **text-shadow** property is used to set the text shadow around a text.

**CSS Font Properties:** Defines how to set fonts of a content, available in an HTML element.

1. The **font-family** property is used to change the face of a font.
2. The **font-style** property is used to make a font italic or oblique.
3. The **font-variant** property is used to create a small-caps effect.
4. The **font-weight** property is used to increase or decrease how bold or light a font appears.
5. The **font-size** property is used to increase or decrease the size of a font.
6. The **font** property is used as shorthand to specify a number of other font properties.

**CSS Background Properties:** Defines how to set backgrounds of various HTML elements.

1. The **background-color** property is used to set the background color of an element.
  2. The **background-image** property is used to set the background image of an element.
  3. The **background-repeat** property is used to control the repetition of an image in the background.
  4. The **background-position** property is used to control the position of an image in the background.
  5. The **background-attachment** property is used to control the scrolling of an image in the background.
  6. The **background** property is used as a shorthand to specify a number of other background properties.
-

**CSS Border Properties:** The border properties allow you to specify how the border of the box representing an element should look.

1. The **border-color** specifies the color of a border.
2. The **border-style** specifies whether a border should be solid, dashed line, double line, or one of the other possible values.
3. The **border-width** specifies the width of a border.

**CSS Margin Properties:** Defines the space around an HTML element.

1. The **margin** specifies a shorthand property for setting the margin properties in one declaration.
2. The **margin-bottom** specifies the bottom margin of an element.
3. The **margin-top** specifies the top margin of an element.
4. The **margin-left** specifies the left margin of an element.
5. The **margin-right** specifies the right margin of an element.

**CSS Link Properties:** Defines how to set different properties of a hyper link using CSS.

1. The **:link** signifies unvisited hyperlinks.
2. The **:visited** signifies visited hyperlinks.
3. The **:hover** signifies an element that currently has the user's mouse pointer hovering over it.
4. The **:active** signifies an element on which the user is currently clicking.

**CSS List Properties:** Defines how to control list type, position, style, etc., using CSS.

1. The **list-style-type** allows us to control the shape or appearance of the marker.
2. The **list-style-position** specifies whether a long point that wraps to a second line should align with the first line or start underneath the start of the marker.
3. The **list-style-image** specifies an image for the marker rather than a bullet point or number.
4. The **list-style** serves as shorthand for the preceding properties.
5. The **marker-offset** specifies the distance between a marker and the text in the list.

**CSS Table Properties:** Defines how to set different properties of an HTML table using CSS.

1. The **border-collapse** specifies whether the browser should control the appearance of the adjacent borders that touch each other or whether each cell should maintain its style.
2. The **border-spacing** specifies the width that should appear between table cells.
3. The **caption-side** captions are presented in the `<caption>` element. By default, these are rendered above the table in the document. We use the caption-side property to control the placement of the table caption.
4. The **empty-cells** specifies whether the border should be shown if a cell is empty.
5. The **table-layout** allows browsers to speed up layout of a table by using the first width properties it comes across for the rest of a column rather than having to load the whole table before rendering it.

## 1.2.5 CSS Navigation Bar

---

- A navigation bar is a user interface element within a webpage that contains links to other sections or web pages of the website.
- A navigation bar is one of the main components of a website, because it is the first part that the user sees when he/she enter a website and it links to the other main parts of the website.
- A navigation bar comes under GUI that helps the visitors in accessing information. It is the UI element on a webpage that includes links for the other sections of the website.
- A navigation bar is mostly displayed on the top of the page in the form of a horizontal list of links.
- Using following steps, we can easily create the Navigation bar:

**Step 1:** Firstly, we have to type the Html code in any text editor or open the existing Html file in the text editor in which we want to make a Navigation Bar.

**Step 2:** Now, we have to define the <nav> tag in the <body> tag where we want to make the bar.

---

### Example for navigation bar:

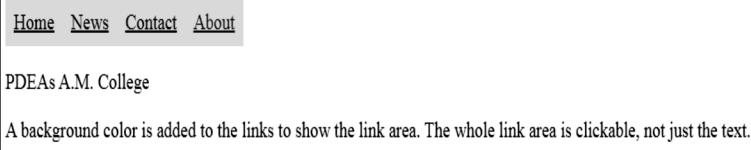
```
<!DOCTYPE html>
<html>
  <head>
    <style>
      ul {
        list-style-type: none;
        margin: 0;
        padding: 0;
        overflow: hidden;
      }

      li {
        float: left;
      }
      li a {
        display: block;
        padding: 8px;
        background-color: #dddddd;
      }
    </style>
  </head>
  <body>
    <ul>
```

```

<li><a href="d:\Ty\home.html">Home</a></li>
<li><a href="d:\Ty\news.html">News</a></li>
<li><a href="d:\Ty\contact.html">Contact</a></li>
<li><a href=""d:\Ty\About.html">About</a></li>
</ul>
<p> PDEAs A.M. College </p>
<p>A background color is added to the links to show the link area. The whole link area is clickable, not just the text.</p>
</body>
</html>

```

**Output:**

- In above example the, list-style-type: none; - Removes the bullets. A navigation bar does not need list markers. Set margin: 0; and padding: 0; to remove browser default settings.

## **1.3 INTRODUCTION TO WEB SERVER AND WEB BROWSER**

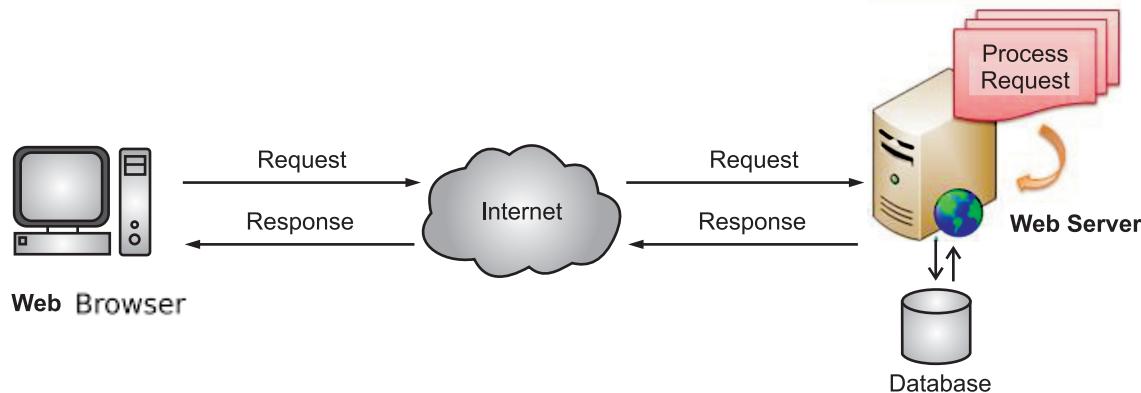
- The World Wide Web (WWW), which is also known as a Web, is a collection of websites or web pages stored in web servers and connected to the computers through the internet.
- These websites contain text pages, digital images, audios, videos, etc. Users can access the content of these sites from any part of the world over the internet using their devices such as computers, laptops, smart phones etc.
- Internet (or the Web) is a massive distributed client/server information system. Web servers and Web browsers are the basic components of the WWW or the Web.
- Web browser is an application program that displays a web document while Web server is a program or a computer used to store, process and serve the Web pages.
- The Web browser (clients) requests for the web document then the Web server accepts and response to the request made by a Web browser for a web document.

### **1.3.1 Web Server**

**[April 16, 19, Oct. 16, 18]**

- The web is a collection of files known as web pages. These web pages (or web site) reside on a computer known as a web server.
- A web server interacts with the client through the web browser and processes requests, sent by the client.
- The primary function of a web server is to store, process and deliver web pages to clients.

- The function of a typical Web server is shown in Fig. 1.9. The communication between client and server takes place using the HyperText Transfer Protocol (HTTP).
- To view a website, the browser sends a request to the server. On receiving the request, the server sends (response) the appropriate web page to the client's machine.
- The client's machine (browser) receives the web page and displays onto the user's computer screen.



**Fig. 1.9: Function of a Typical Web Server**

- Some leading web servers available today are listed below:
  - Apache HTTP Server** is the most popular web server in the world developed by the Apache Software Foundation. Apache web server is open source software and can be installed on almost all operating systems including Linux, UNIX, Windows, FreeBSD, Mac OS X and more.
  - Internet Information Services (IIS)** is a high performance Web Server from Microsoft. This web server runs on Windows NT/2000 and 2003 platforms.
  - Sun Java System Web Server** from Sun Microsystems is suited for medium and large web sites. It runs on Windows, Linux and UNIX platforms.
  - Lighttpd Web Server** pronounced lighty and it is a free and open source secure web server. Lighttpd can run on Windows, Mac OS X, Linux and Solaris operating systems.
  - Jigsaw Web Server** is open source and free and can run on various platforms like Linux, UNIX, Windows, Mac OS etc. Jigsaw has been written in Java and can run CGI scripts and PHP programs.

### 1.3.2 Web Browser

[April 16, 17, 18, Oct. 16, 18]

- To view Web pages, we need a Web browser which is a software program that requests, downloads and displays Web pages stored on a Web server.
- Web browser used to access the Internet and the WWW. It is basically used to access and view the web pages of the various websites available on the Internet.
- A Web browser (a browser) is application software for retrieving, presenting, and traversing information resources on the Internet and World Wide Web (WWW).

- A browser can be defined as, "a software used to locate, retrieve and display content on the Internet and World Wide Web, including web pages, images, audio, video and other files".
- Browsers are of two types as explained below:
  1. **Graphical Browsers:** These browsers allow retrieval of text, images, audio, and video. Navigation is accomplished by pointing and clicking with a mouse on highlighted words and graphics. Both Netscape Navigator and Internet Explorer are graphical browsers.
  2. **Text Browsers:** These browsers provide access to the web in text-only mode. Navigation is accomplished by highlighting emphasized words on the screen with the arrow up and down keys, and then pressing the Enter key to follow the link. Lynx is an example of text-based browser.
- Web browser is application software that allows us to view and explore information on the web. Common and popular Web Browsers are explained below:
  1. **Opera (O):** Opera is a web browser developed by Opera Software. The latest version is available for Windows, macOS, and Linux operating systems. Opera is a fast and secure browser.
  2. **Internet Explorer (IE):** Internet Explorer (IE) is a web browser from Microsoft. IE was released as the default browser with Windows 95 (1995). Internet Explorer connects your computer to the Web through an Internet connection. The browser lets you view, print, and search for information on the Web.
  3. **Mozilla Firefox (F):** Firefox is a free and open source web browser developed for Microsoft Windows, Mac OS X, and Linux (with its mobile versions available for Android, and Firefox OS) co-ordinated by Mozilla Corporation and Mozilla Foundation.
  4. **Safari (S):** Safari is web browser that was produced and developed by Apple Inc. which functions on a Mac OS X, iOS, and Windows operating system.
  5. **Google Chrome (G):** Google Chrome is a freeware browser developed by Google using the WebKit layout engine.

#### Differences between Web Browser and Web Server:

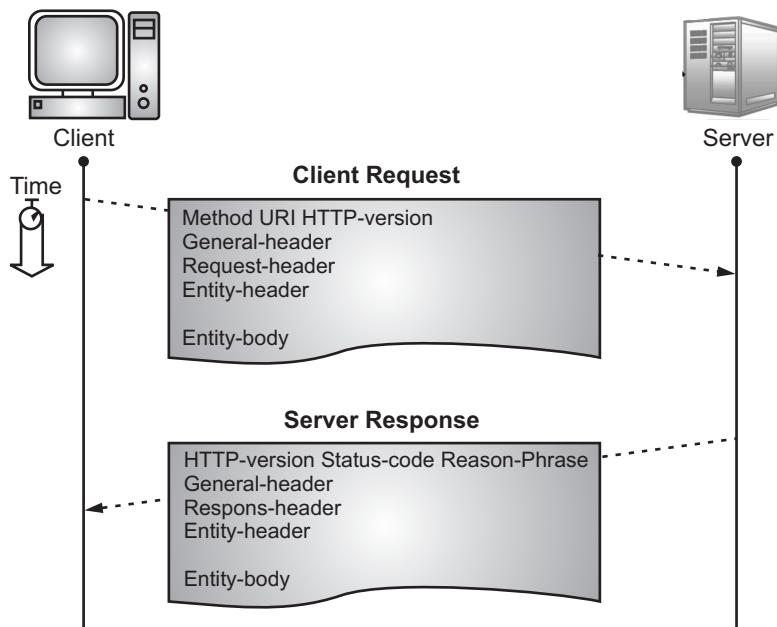
Sr. No.	Web Browser	Web Server
1.	Web browser is software which is used to browse and display web pages available over the internet.	Web server is a software which provides the documents when requested by web browsers.
2.	A web browser sends request to server for web based documents.	Web server sees and approves those requests made by web browsers and sends the document in response.

*contd. ...*

3.	Web browser is installed on user's (client) machine.	Web server can be installed anywhere at remote location computer or local location computer but it need to be on a network.
4.	Examples include Mozilla Firefox, Google Chrome, Internet Explorer, Opera, Netscape Navigator etc.	Examples include Apache HTTP Server, Internet Information Services (IIS), Lighttpd etc.
5.	Web browser sends an HTTP request and gets a HTTP response.	Web server receives HTTP request and sends a HTTP response.
6.	Web browsers stores data in cookies for websites.	Web servers provide an area to store and organize the pages of the website.

## 1.4 HTTP BASICS

- HTTP stands for HyperText Transfer Protocol. A protocol is a set of rules that govern the way two or more computers communicate with one another.
- HTTP is used to transfer data across the Web. HTTP is connectionless and stateless application layer protocol.
- HTTP is the foundation for data communication for the World Wide Web or Web (i.e. internet) since 1990. HTTP is at the heart of the Web and described in [RFC 1945] and [RFC 2616].



**Fig. 1.10: HTTP Transaction between the Client and Server**

- HTTP specifies how clients' request data will be constructed and sent to the server, and how the servers respond to these requests.

- The HTTP protocol is based on the client-server based architecture where Web browsers act as clients and the Web server acts as a server.
- The Client (also called as HTTP client) and server (also called as HTTP server) communicate by sending messages.
- The client sends a request message (also called as HTTP Request) to the server. The server, in turn, returns a response message (also called as HTTP Response).
- Fig. 1.10 shows the HTTP transaction (request and response) between the client and server.

### HTTP Messages:

- HTTP message is used to show how data is exchanged between the client and the server. It is based on client-server architecture.
- An HTTP client is a program that establishes a connection to a server to send one or more HTTP request messages.
- An HTTP server is a program that accepts connections to serve HTTP requests by sending an HTTP response messages.
- HTTP messages are simple, line-oriented sequences of characters. There are two types of HTTP messages i.e. request message and response message.
- The HTTP messages sent from web clients to web servers are called request messages while the messages from servers to clients are called response messages.
- The formats of the request and response messages are shown in Fig. 1.11.

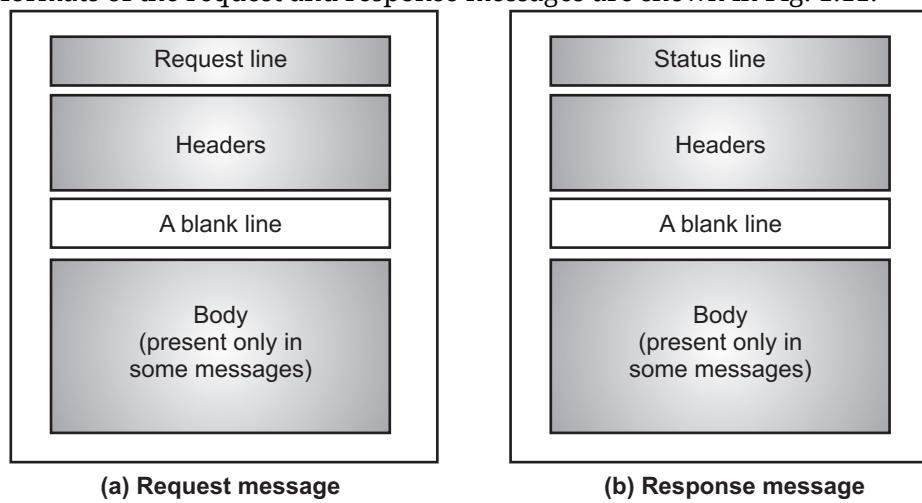


Fig. 1.11

#### 1. HTTP Request Message:

[Oct. 16]

HTTP Requests are messages which are sent by the client or user to initiate an action on the server. HTTP Request messages consist of following three parts:

**Request Line:** The first line of the header is called the request line. The request line looks like this:

```
GET/index.html HTTP/1.1
```

This line specifies the HTTP method followed by address of the document requested and the version of the HTTP.

**Request Headers:** The HTTP Request Header contains optional header information sent by the client like MIME type the browser accepts, information about the web browser etc.

**For example:** Accept: image/gif, image/jpeg, text/\*, \*/\*  
Accept-Language: en-us  
Host: www.example.com  
User-Agent: Mozilla (x11; I; Linux 2.0.32 i586)

After header the HTTP request contains a **blank line** which indicates the end of the header section.

**Request Body:** In case of GET method the HTTP request body is empty, and with the POST method it contains additional data. Message body is optional in HTTP request.

2. **HTTP Response Message:** The web server receives the request from client, processes it and sends a response back to the client browser through Response message.

**Status Line:** The first line is called status line which looks like this:

HTTP/1.1 200 OK

This line specifies the protocol version, a status code, and a description of that code. In this case, the status code is “200”, meaning that the request was successful (hence the description “OK”).

**Response Headers:** The response header gives client additional information about the response like information about the web server software, MIME type of the data included in the response etc.

For example: Date: Sat, 26 Jan 2002 20:25:12 GMT

Server: Apache 1.3.22 (Unix) mod\_perl/1.26 PHP/4.1.0  
Content-Type: text/html  
Content-Length: 141

After header the HTTP request contains a blank line which indicates the end of the header section.

**Response Body:** If the response was successful, the HTTP response body contains the HTML code, ready for the browser interpretation. If unsuccessful, a failure code is sent.

---

**Example for HTTP Request Message and HTTP Response Message:** HTTP request to fetch hello.htm page from the web server running on niraliprakashan.com. The HTML code given below:

```
<html>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>
```

**Client Request:**

```
GET /hello.htm HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
Host: www.niraliprakashan.com
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: Keep-Alive
```

**Server Response:**

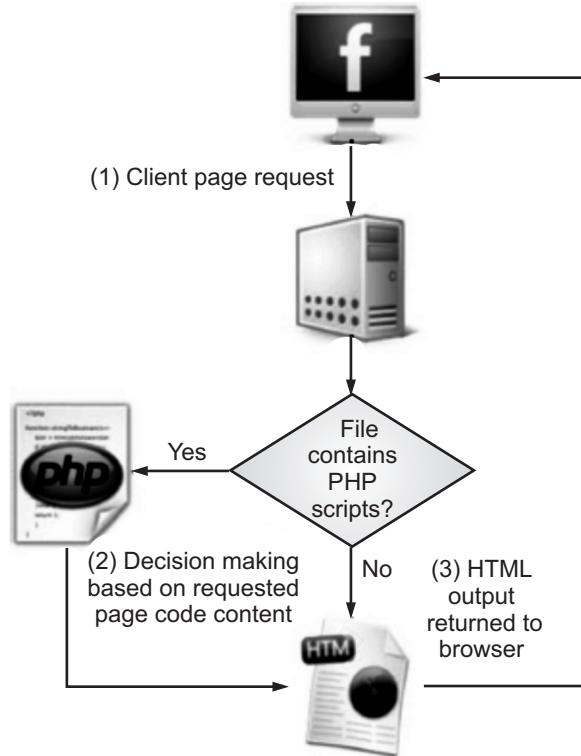
```
HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
Content-Length: 88
Content-Type: text/html
Connection: Closed
```

---

**1.5 PHP BASICS****[April 19]**

- PHP is a server side scripting language which is used to create dynamic and interactive web pages.
- A script is a set of programming instructions that is interpreted at runtime. A scripting language is a language that interprets scripts at runtime.
- Server side scripts are interpreted on the server while client side scripts are interpreted by the client.
- PHP is a server side script that is interpreted on the server while JavaScript is an example of a client side script that is interpreted by the client browser.
- PHP is a recursive acronym for "PHP: Hypertext Preprocessor". PHP is an open-source, interpreted, and object-oriented scripting language that can be executed at the server-side.
- PHP files can contain text, HTML, CSS, JavaScript, and PHP code. PHP codes are executed on the server, and the result is returned to the browser as plain HTML. PHP files have extension ".php".
- PHP 7 is a major release of PHP programming language in 2015 and is touted to be a revolution in the way web applications can be developed and delivered for mobile to enterprises and the cloud.
- The latest version of PHP is PHP 8 was released on November 26, 2020. Facebook, Flickr and Yahoo are the examples of PHP applications.
- PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.

- PHP is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
- The basic architecture of a PHP web application and how the server handles the requests is shown in Fig. 1.12.



**Fig. 1.12: Basic Architecture of PHP Web App**

- We use PHP because of following reasons:
  1. PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
  2. PHP supports many databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
  3. PHP is free and open source. Download it from the official PHP resource [www.php.net](http://www.php.net) without any charge.
  4. PHP is easy to learn and runs efficiently on the server side.
  5. PHP is compatible with almost all servers used today (Apache, IIS, etc.)

#### Advantages of PHP:

[Oct. 17]

1. **Speed:** It is relative fast since it uses much system resource.
2. **Easy to use:** It uses C like syntax, so for those who are familiar with C, it's very easy for them to pick up and to create a website.
3. **Stable:** Since it is maintained by many developers, so when bugs are found, it can be quickly fixed.

4. **Platform independent:** Can be run on many platforms, including Windows, Linux and Mac, it's easy for users to find hosting service providers.
5. **Open source:** It is developed and maintained by a large group of PHP developers, this will help in creating a support community, abundant extension library.
6. **Built-in database connection modules:** We can connect to database easily using PHP, since many websites are data/content driven, so we will use database frequently, this will largely reduce the development time of web apps.
7. **Powerful library support:** You can easily find functional modules you need such as PDF, Graph etc.

**Disadvantages of PHP:**

1. **Security:** Since it is open sourced, so all people/user can see the source code, if there are bugs in the source code, it can be used by people to explore the weakness of PHP.
2. **Weak type:** Implicit conversion may surprise unwary programmers and lead to unexpected bugs. For example, the strings "1000" and "1e3" compare equal because they are implicitly cast to floating point numbers.
3. **Not suitable for large applications:** Hard to maintain since it is not very modular.

### 1.5.1 Uses of PHP

---

- Common uses of PHP are listed below:
  1. PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
  2. Using PHP we can handle forms, i.e. gather data from files, save data to a file.
  3. PHP is used for creating dynamic website.
  4. Using PHP, we can restrict users to access some pages of the website.
  5. PHP is used to interact with web server (Apache, IIS etc.).
  6. PHP is used for connecting web application with Database like MySQL, DB2, PostgreSQL etc. We add, delete, and modify elements within the database through PHP.
  7. PHP can be used for encrypt data.
  8. PHP is usually used to output HTML code to the browser.
  9. Using PHP we can access cookies variables and set cookies.

### 1.5.2 Features of PHP

---

**[April 17]**

- Features of PHP are listed below:
  1. **Performance:** PHP script is executed much faster than other scripting languages such as JSP and ASP.
  2. **Database Connectivity:** PHP supports all the leading databases such as MySQL, SQLite, PostgreSQL, DB2, etc.

3. **Existing Libraries Support:** Lot of functions for common web development task e.g. Sending E-mail, XML parsing etc.
4. **Portability:** PHP support on all major operating system like Windows, Mac OS, Linux etc. A PHP application developed in one operating system can be easily executed in other operating system also.
5. **Simplicity:** In PHP there is no need to include libraries, special compilation directives, or anything of the sort.
6. **Efficiency:** PHP is efficient in a multiuser environment such as the WWW. PHP also support object-oriented programming with similar syntax and feature as C++ and Java.
7. **Flexibility:** Because PHP is an embedded language, it is extremely flexible towards meeting the needs of the developer. PHP code can be easily embedded within HTML tags and script.
8. **Familiarity:** In PHP many of the language's constructs are borrowed from C and Perl. PHP has easily understandable syntax. Programmers are comfortable coding with it.
9. **Security:** PHP is a secure scripting language to develop the website. It consists of multiple layers of security to prevent threads and malicious attacks.
10. **Open Source:** PHP source code and software are freely available on the web. Anyone can develop all the versions of PHP according to his/her requirement without paying any cost.

### 1.5.3 Syntax of PHP

---

- PHP script is executed on the server, and the plain HTML result is sent back to the browser.
- A PHP script can be placed anywhere in the document. A PHP programming script starts with <? php and ends with ?>. Every PHP command ends with a semicolon (;).

```
<?php
    // PHP code goes here with PHP commands...;
?>
```
- The default file extension for PHP files is ".php". A PHP file normally contains HTML tags and some PHP scripting code.
- Below, we have an example of a simple PHP file, with a PHP script that uses a built-in PHP function "echo" to output the text "Hello World!" on a web page.

---

#### Example for first PHP program:

```
<!DOCTYPE html>
<html>
    <body>
        <h1>My first PHP page</h1>
```

```

<?php
    echo "Hello PHP";
?
</body>
</html>

```

**Output:**

- PHP code can be included or embedded into webpage in following ways:
  1. XML style <?php ..... ?>
  2. SGML style <? ..... ?>
  3. ASP style <% ..... %>
  4. Script style <script language = "PHP"> PHP Code ..... </script>
- On Linux platform we directly execute PHP program. On Windows Environment we can use XAMPP web server to run PHP program. XAMPP is an open-source, cross-platform web server.
- XAMPP stands for Cross-Platform (X), Apache (A), MariaDB (M), PHP (P) and Perl (P). XAMPP is a software distribution which provides the Apache web server, MySQL database (actually MariaDB), PHP and Perl (as command-line executables and Apache modules) all in one package.

## **1.6 LEXICAL STRUCTURE**

- The lexical structure of a programming language is the set of basic rules that governs how you write programs in that language.
- A PHP program consists of tokens. Token is a smallest unit of a program such as numbers, strings, variables, constants, operators, delimiters, and keywords etc.

### **1.6.1 Case Sensitivity**

- PHP is a case sensitive language. Case sensitivity defines whether uppercase and lowercase letters are treated as distinct (case-sensitive) or equivalent (case-insensitive).
- The names of user defined classes and functions as well as built in constructs and keywords are case insensitive. But variables on the other hand are case sensitive.  
i.e. echo("hello")  
 ECHO("hello")  
 ECHO("hello")
- All above three statements are same. But variable names \$name, \$NaME and \$NAME are three different variables.

## 1.6.2 Statements and Semicolons

- A statement is a collection of PHP code that does something. PHP uses semicolons (;) to separate simple statements.
- A compound statement uses curly braces to mark start and end of the statement and in this case semicolon is not required after closing brace.
- For example:

```
<?php
    echo "hello";
    $a=1;
    $name="PHP";
    $b=$a*$b;
    echo $b
?>
```

- Semicolon is not required for last statement of the PHP code i.e. before closing PHP tag. It's good practice to include optional semicolons, as they make it easier to add code later.

## 1.6.3 White Space and Line Breaks

- White space in PHP is used to separate tokens in PHP source file. It is used to improve readability of the source code.
- In PHP, there is no restriction on usage of white spaces. We can use this facility to make the code more readable.
- **For example:** names(\$a, \$b, \$c);
- We can write same statement as follows:

```
names($a,
      $b,
      $c,
);
```

- To create a line break in PHP, all we have to do is put the code '<br>' in the echo statement.

## 1.6.4 Comments

- Comments give information to people who read the code, but they are not interpreted by PHP engine.
- Comments in PHP programming language is used to describe code and make simple to understand other programmer/users and it is ignore by compiler or interpreter.
- PHP supports 'C', 'C++' and Unix shell-style (Perl style) comments.

**For example:**

```
<?php  
    echo 'This is a test'; // This is a one-line C++ style comment  
    # Shell-style comment  
    /* This is a multi line comment.  
       This is a C style comment.  
       Yet another line of comment */  
    echo 'This is yet another test';  
    echo 'One Final Test'; # This is a one-line shell-style comment  
?>
```

- When PHP encounters two slash characters (//) within the code, everything from the slashes to the end of the line or the end of the section of code, whichever comes first, is considered a comment. This method of commenting is derived from C++. The result is the same as the shell comment style.
- PHP supports block comments, whose syntax comes from the C programming language.
- When PHP encounters a slash followed by an asterisk (\*), everything after that until it encounters an asterisk followed by a slash (\*) is considered a comment.
- When PHP encounters a hash mark (#) within the code, everything from the hash mark to the end of the line or the end of the section of PHP code (whichever comes first) is considered a shell-style comment.
- The shell-style comment is found in Unix shell scripting languages and is useful for annotating single lines of code or making short notes.

## 1.6.5 Literals

---

- A literal is a data value in PHP that appears directly in a program.
- **For example:**

```
"Hello"  
100  
true  
null
```

## 1.6.6 Identifiers

---

- An identifier is simply a name. In PHP, identifiers are used to name variables, functions, constants, and classes.
- The first character of an identifier must be an ASCII letter (a-z, A-Z), the underscore character (\_) or any character between ASCII Ox7F and ASCII OxFF. After the initial character, combinations of these previous characters and the digits 0-9 are valid.

- Variable Names:** Variables in PHP starts with a dollar sign (\$). The variable name is case sensitive.

**Valid Variable Names:**

```
$bill
$bill_amt
$GrossSal
$_underscore
```

**Invalid Variable Names:**

```
$not valid
$7X
```

Following valid variables are all different:

```
$Net_sal
$net_sal
$net_sal
$NET_SAL
```

- Function Names:** Function names are not case sensitive. And so following function names refer to the same function:

**For example:** howdy(), Howdy(), HOWDY(), HOWdy()

- Class Names:** Class names follow the standard rules for PHP identifiers and are not case sensitive.

**For example,**

```
person or Person is same
Account or Account is same
```

## 1.6.7 Constants

[April 18]

- A constant is a name or an identifier for a simple value. A constant value cannot change during the execution of the PHP script.
- A constant is case-sensitive by default. By convention, constant identifiers are always uppercase. Only single values i.e. scalars can be constants, like Boolean, integer, double, string etc.
- In PHP we use the define() function to create a constant. It defines constant at run time.

**Syntax:** define("constant\_Name", value)

For example,

```
define('PI', 3.14);
echo PI;
```

- In PHP the const keyword also defines constants at compile time. It is a language construct not a function. It is always case sensitive.

For example,

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

## 1.6.8 Keywords

[Oct. 17]

- Words reserved by the language for its core functionality are called as keyword or reserved word.
- In PHP, keywords are case insensitive. A keyword is a word reserved by the language for its core functionality - we cannot give a variable, function, class, or constant the same name as a keyword.
- Some examples of keywords in PHP are given below:

break	exception	new
case	while	or
catch	endwhile	public
class	exit( )	static
do	function	switch
for	if	throw
endfor	else	try
extends	elseif	var

## 1.7 LANGUAGE BASICS

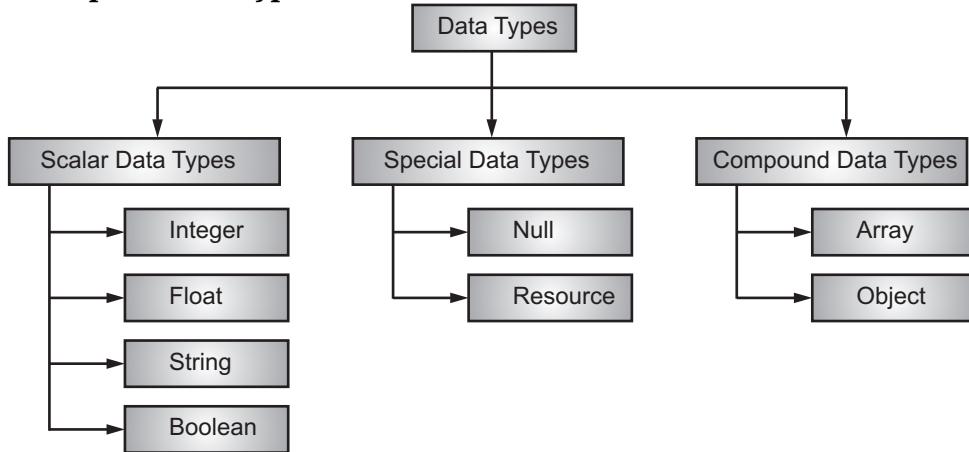
- PHP is an open source, server side scripting language. PHP is one of the most popular scripting languages of the last couple of years for developing web application (Web Apps).
- A Web application is a computer software or program that performs some specific tasks at its client by using a Web browser.
- Web applications are usually based on the client-server architecture where the client input/request data while the server stores and responds with result.
- PHP is an interpreted language, i.e. there is no need for compilation. In this section we will basic language concepts of PHP.

### 1.7.1 Data Types

[Oct. 17, April 18]

- A data type is defined as "a set of values and the allowable operations on those values". The data type determines the operations that we can perform on it.
- PHP data types are used to hold different types of data or values. PHP data types define the type of data a variable can store.

- PHP supports eight primitive types as shown in Fig. 1.13 and given below:
  - **Four scalar data types:** integer, float (floating-point number, aka double), string, and Boolean.
  - **Two compound data types:** array and object.
  - **Two special data types:** resource and NULL.



**Fig. 1.13: Data Types in PHP**

- Let us see above data types in detail:

### 1. Integers:

- The integer data type is used to specify a numeric value without a fractional component.
- The range of integers in PHP is equivalent to the range of the long data type in C. On 32-bit platforms, integer values can range from  $-2,147,483,648$  to  $+2,147,483,647$ .
- Integers can be written in decimal, octal or hexadecimal. If it is decimal, it is just the number. If it is octal, then number should precede with (zero) 0. If it is hexadecimal then precede with OX.
- PHP does not support unsigned integers. Integer size can be determined using the constant PHP\_INT\_SIZE.
- In PHP, `is_int()` or `is_integer()` are used to test whether a value is an integer.

```

<?php
if(is_int($a))
{
    echo"Number is an integer";
}
? >
  
```

### 2. Floating Point Numbers:

- Floating point numbers are real numbers, representing numeric values with decimal digits.
- Usually, this allows numbers between  $1.7E - 308$  and  $1.7E + 308$  with 15 digits of accuracy.

- PHP recognizes floating point numbers written in two different formats:
  - (i) **Common Format:**  
3.14, 0.25
  - (ii) **Scientific Format:**  
17.0E-3 // 17.0\*10-3, or 0.017
- Use the `is_float()` function (or its `is_real()` alias) to test whether a value is a floating point number:

```
if (is_float($x))
{
    // $x is a floating-point number
}
```

### 3. Strings:

- A string is a sequence of characters where a character is the same as a byte. PHP only supports a 256-character set.
- String literals are delimited by either single ('...') or double quotes ("...").  
**For example:** 'Hello PHP' and "Hello PHP" is same.
- Variables are expanded within double quotes not within single quote.

```
$a="Good";
echo "$a, morning \n";
echo '$a, morning';
```

#### Output:

```
Good, morning
$a, morning
```

- Use `is_string()` function to test whether the value is string or not.

```
if(is_string($x))
{
    // $x is a string
}
```

### 4. Boolean:

- Boolean value can be either TRUE value or FALSE value. Both are case-insensitive.

#### For example:

```
<?php
$x = True; // assign the value TRUE to $x
?>
```

- In PHP, `is_bool()` function is used to test whether value is Boolean or not.

```
$x = True;
if(is_bool($x))
{
    // $x is boolean;
}
```

- In PHP, the following values are false:
  - The keyword false.
  - The integer 0.
  - The floating-point value 0.0.
  - The empty string ("") and the string "0".
  - An array with zero elements.
  - An object with no values or functions.
  - The NULL value.

### 5. Arrays:

- An array stores group of values under single variable name. In PHP array is a collection of the different type of values.
- The values can be identified by position or some identifying name called as associative.

```
$a = array('A', 'B', 'C');
$b = array('First' => 'A',
           'Second' => 'B'
           'Third' => 'C');
```

\$a is indexed array and elements are recognized by index starting with 0.

i.e.    \$a[0] = "A";
 \$a[1] = "B";
 \$a[2] = "C";

\$b is associative array in which key and value both are given.

i.e.    \$b['First'] = "A";
 \$b['Second'] = "B";
 \$b['Third'] = "C";

foreach loop is most common in arrays.

i.e.    foreach (\$a1 as \$value1)
{
 echo "Hello, \$a1 \n";
}

### 6. Objects:

- PHP supports Object Oriented Programming (OOP). Objects are the special instances of the classes that are created by user.
- Object is a term used in association with classes. A class is a definition of a structure that contains properties (variables) and methods (functions).
- Classes are defined with the 'class' keyword. Once a class is defined, any number of objects can be made from it with the 'new' keyword.
- Objects properties and methods can be accessed with the → construct.

**For example:**

```
class Person
{
    public $name = ' ';
    function name ($newname)
    {
        $this → name = $newname;
    }
}
$p = new Person();
$p->name("Amar");
echo "Hello $p->name";
```

**Output:**

```
Hello Amar
```

- Use `is_object()` to test whether a value is an object.

```
if(is_object($x))
{
    // $x is an object
}
```

## 7. Resource:

[April 16]

- The resource is a special PHP data type that refers to external resource (e.g. file, image etc.) which is not part of the PHP native language.
- It is basically used for dealing with the outside world. Resources are created and used by special functions.
- For example, database connection function returns a resource which is used to identify that connection when we call the query and close functions.
- Their main benefit is that they're garbage collected when no longer in use. When the last reference to a resource value goes away, the extension that created the resource is called to free any memory, close any connection, etc. for that resource.

For example,

```
$result = database_connect( );
database_query ($result);
$result="something"; // connection is closed.
```

- Use the `is_resource()` function to test whether a value is a resource:

```
if (is_resource($x))
{
    // $x is a resource
}
```

**8. NULL:**

- This data type is having only one value and denoted by the keyword NULL. The NULL value represents a variable that has no value.
- A variable is considered to be null if:
  - (i) it has been assigned the constant NULL.
  - (ii) it has not been set to any value yet.
  - (iii) it has been unset().
- The is\_null() function is used to test whether a value is NULL.

## **1.7.2 Variables**

---

- PHP variables are nothing but a named storage locations in the memory. A variable is a named container in a PHP script in which a data value can be stored.
- The stored value can be referenced using the variable's name and changed (varied) as the script proceeds/executes.
- Variables in PHP are identifiers prefixed with a dollar sign (\$). For example:

```
$name  
$Age  
$_debugging  
$MAXIMUM_IMPACT
```

- A variable may hold any type of value. There is no compile-time or runtime type checking on variables. You can store any type of value in the same variable.

For example,

```
$a = "Hello";  
$a = 12;  
$a = array(10, 20, 30);
```

**Variable Declaration:**

- Variables are "containers" for storing information. In PHP, a variable is declared using a \$ sign followed by the name of the variable.

**Syntax:** \$variable\_name;

**For example:** \$sum;

**Rules for Variable Declaration:**

1. A PHP variable must start with (\$) dollar sign.
2. A variable name must start with an alphabet letter or the underscore (\_).
3. A variable name cannot start with a number like \$10RollNo.
4. PHP variable can be of any length.
5. Variable names are case-sensitive, (\$RollNo and \$rollNo are two different variables).
6. PHP variable does not contain spaces.

### Defining Variables:

- A variable stores a value of any type such as string, number, array, object or resource. In PHP, defining variables means assigning values to the variables.
- Assigning a value to a variable in PHP is accomplished with the assignment operator (=), with the variable on the left-hand side and the value to be evaluated on the right.
- **Syntax** to define a variable in PHP is `$variable_name=value;`

**For example:** `$EmpId=10;`

```
$StudentName="Vedant";
```

- A variable whose value has not been assign or set behaves like the NULL value. Following example shows example of variables in 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 world!
Integer is: 5
Float is: 10.5
```

### 1.7.2.1 Variable Variables

[Oct. 16]

- PHP allows us to use dynamic variable names called as variable variables. Variable variables are simply variables whose names are dynamically created by another variable's value.
- In variable variables the value of existing variable is used as a name of new variable. For example,

```
$a = 'hello'; //hello is value of variable $a
$$a = 'PHP'; //$(a) is equals to $(hello)
echo $$a; //$$a is PHP i.e. #a is new variable with value 'PHP'
```

### 1.7.2.2 Variable References

- Variable reference is an alias (duplicate name) of existing variable.
- In PHP we can create reference to some variable. Consider the following examples,

```
$a = 5;
$b = &$a;
```

- In above example, \$b is the reference variable or \$b is an alias for the variable \$a. \$b is now another name for the value that is stored in \$a.
- Now, same value can be used by both the names.

```
echo $b; // Output: 5
$b = $b + 2;
echo $a; //
```

**Output:**

7 (\$b is alias of \$q i.e. \$q and \$b are same variable).

- We can unset the variable by using unset function but the reference is still set.

```
unset($a);
echo $b; // Output: 5
```

**1.7.2.3 Variable Scope**

[Oct. 16, 18]

- Scope of variable is an area or part of program in which it is accessible. The scope of a variable is the context within which it is defined.
- Scope can be defined as, "the range of availability of a variable has to the program in which it is declared".
- There are four types of variable scope in PHP i.e., local, global, static and function parameter.

**1. Local Scope:**

- A variable which is declared inside the function is called as local variable.
- Local variable has access or life only within that function. Local variable cannot be accessed from outside the function.

**Example for local scope:**

```
<?
    $a = 4; // global scope, $a is a global variable
    function assigna ()
    {
        $a = 0; // local variable $a
        print "a inside function is $a.";
    }
    assigna();
    print "a outside of function is $a. ";
?>
```

**Output:**

a inside function is 0.  
a outside of function is 4.

- Only functions can provide local scope. Unlike in other languages, in PHP we can't create a variable whose scope is a loop, conditional branch or other type of block.

## 2. Global Scope:

- Variables declared outside function are by default global variables and can accessed from any part of program.
- A global variable can be accessed in any part of the program except inside the functions. Inside the function, these variables are accessed using the 'global' keyword.

### Example for global scope:

```
<?php
    $x = 15;
    $y = 20;
    function addit()
    {
        GLOBAL    $x; // global variable;
        $y = 10; // global variable;
        $x++;
        $y++;
        printf "x = $x & y = $y";
    } addit();
?>
```

### Output:

---

x = 16 and y = 11

---

## 3 Static Variable:

- We know that, when function ends then all the variables declared inside the function are destroyed. Static variables are those variables which can hold value when function called again.
- Static variables are used only with the functions. These variables retain their values between different calls to a function.
- We can declare a variable to be static simply by placing the keyword STATIC in front of the variable name.
- The initialization of STATIC variable is done only for the first call to function and not after that.

### Example for static scope or variable:

```
<?php
    function keep_track()
    {
        STATIC $count = 0;
        $count++;
        print $count;
        print "<br>";
    }
```

---

```

        keep_track();
        keep_track();
        keep_track();

    ?>

```

**Output:**

```

1
2
3

```

---

**4. Function Parameter:**

- Function parameters are local, i.e. they are available only inside the function.
  - Function parameters are declared after the function name and inside parentheses. They are declared much like a typical variable.
- 

**Example for function parameter:**

```

<?php
    // multiply a value by and return it to the caller
    function multiply ($value)
    {
        $value = $value * 4;
        return $value;
    }
    $retval = multiply (10);
    Print "Return value is $retval\n";
?

```

**Output:**

```
Return value is 40
```

---

**1.7.3 Expressions and Operators**

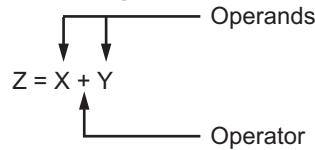
---

- An operating system is defined as, "a organized collection of programs that controls
  - An expression in PHP can be evaluated to produce a value. A value a number, a string of text, or a Boolean.
  - The simplest expressions are literal values and variables. A literal value evaluates to itself, while a variable evaluates to the value stored in the variable. More complex expressions can be formed using simple expressions and operators.
  - Almost everything in a PHP script is an expression. Anything that has a value is an expression. For example, `$x = 5` is an expression, because it evaluates to the value 5.
  - An expression is a combination of values, variables, operators, and functions that results in a value.
  - Operators in PHP are used to perform operations on variables and values. Operators are the symbols that tell PHP what operations to perform.
-

### 1.7.3.1 Operators

[April 16]

- An operator is a symbol that manipulates one or more values usually producing a new value. Operators in PHP are used to performing operations on variables and values.
- Operators are special symbols that perform some specific operations using the operands and operator as shown in Fig. 1.14.



**Fig. 1.14: Concept of operator and operand**

- In Fig. 1.14 operators indicate the operation to be carried out on operands, while operands are the values going to be operated.
- Categories of operators in PHP are as follows:
  1. **Unary Operators** operate on single operand.
  2. **Binary Operators** which take two operands and produces output/result value.
  3. **Conditional Operator (Ternary Operator)** which takes three operands and evaluates either the second or third expression, depending on the evaluation of the first expression.
- Let see various operators in PHP in detail:

#### 1. Arithmetic Operators:

- The PHP arithmetic operators are used to perform common arithmetical operations, such as addition, subtraction, multiplication etc.
- The arithmetic operators require numeric values and non-numeric values are converted automatically to numeric values.
- There are following arithmetic operators supported by PHP language:

Example	Name	Result
<code>-\$a</code>	Negation	Opposite of \$a.
<code>\$a + \$b</code>	Addition	Sum of \$a and \$b.
<code>\$a - \$b</code>	Subtraction	Difference of \$a and \$b.
<code>\$a * \$b</code>	Multiplication	Product of \$a and \$b.
<code>\$a / \$b</code>	Division	Quotient of \$a and \$b..
<code>\$a % \$b</code>	Modulus	Remainder of \$a divided by \$b.
<code>\$a ** \$b</code>	Exponentiation	Result of raising \$a to the \$b'th power. Introduced in PHP 5.6.

- The division operator ("/") returns a float value unless the two operands are integers (or strings that get converted to integers).

### String Concatenation Operator:

- The concatenation operator returns the concatenation of its right and left operands. Operands are first converted to strings, if necessary.

For example:

```
<?php
    $a = "Hello ";
    $b = $a . "World!"; // now $b contains "Hello World!"
    echo $b; // Display Hello World!
?>
```

### 2. Autoincrement and Autodecrement Operators:

- PHP supports C-style pre- and post-increment and decrement operators.
- The increment/decrement operators only affect numbers and strings. The increment and decrement operators are used to increase and decrease the value of a variable.

Example	Name	Effect
<code>++\$a</code>	Pre-increment	Increments \$a by one, then returns \$a.
<code>\$a++</code>	Post-increment	Returns \$a, then increments \$a by one.
<code>--\$a</code>	Pre-decrement	Decrements \$a by one, then returns \$a.
<code>\$a--</code>	Post-decrement	Returns \$a, then decrements \$a by one.

- These operators can be applied to strings as well as numbers. Incrementing an alphabetic character turns it into the next letter in the alphabet.

For example:

```
Incrementing "a" gives "b"
Incrementing "z" gives "aa"
Incrementing "spaz" gives "spba"
Incrementing "K9" gives "L0"
```

### 3. Comparison Operators:

- PHP provides comparison operators to compare two values such as number or string.
- A comparison operator returns a Boolean value, either True or False. If the comparison is truthful, the comparison operator returns True, otherwise it returns False.
- There are following comparison operators supported by PHP language:

Example	Name	Result
<code>\$a == \$b</code>	Equal	True if \$a is equal to \$b after type juggling.
<code>\$a === \$b</code>	Identical	True if \$a is equal to \$b, and they are of the same type.

*contd. ...*

<code>\$a != \$b or \$a &lt;&gt; \$b</code>	Not equal	True if \$a is not equal to \$b after type juggling.
<code>\$a !== \$b</code>	Not identical	True if \$a is not equal to \$b, or they are not of the same type.
<code>\$a &lt; \$b</code>	Less than	True if \$a is strictly less than \$b.
<code>\$a &gt; \$b</code>	Greater than	True if \$a is strictly greater than \$b.
<code>\$a &lt;= \$b</code>	Less than or equal to	True if \$a is less than or equal to \$b.
<code>\$a &gt;= \$b</code>	Greater than or equal to	True if \$a is greater than or equal to \$b.

#### 4. Logical Operators:

- Logical operators in PHP first convert their operands to Boolean values (True or False) and then perform the respective comparison.
- There are following logical operators supported by PHP language:

Example	Name	Result
<code>\$a &amp;and \$b , \$a and \$b</code>	and	TRUE if both \$a and \$b are TRUE.
<code>\$a    \$b , \$a or \$b</code>	or	TRUE if either \$a or \$b is TRUE.
<code>\$a xor \$b</code>	xor	TRUE if either \$a or \$b is TRUE, but not both.
<code>! \$a</code>	not	TRUE if \$a is not TRUE.

#### 5. Assignment Operators:

- The PHP assignment operators are used with numeric values to write a value to a variable.
- The basic assignment operator (=) assigns a value to a variable. The left-hand operand is always a variable. The right-hand operand can be any expression - any simple literal, variable, or complex expression. The right-hand operand's value is stored in the variable named by the left-hand operand.
- In addition to the basic assignment operator, there are "combined operators" (or short hand assignment operator) for all of the binary arithmetic, array union and string operators that allow us to use a value in an expression and then set its value to the result of that expression.

**For example:**

```
<?php
    $a = 3;
    $a += 5; // sets $a to 8, as if we had said: $a = $a + 5;
?>
```

- These assignment operators are Plus-equals (`+=`), Minus>equals (`-=`), Divide>equals (`/=`), Multiply>equals (`*=`), Modulus>equals (`%=`), Bitwise-XOR>equals (`^=`), Bitwise-AND>equals (`&=`), Bitwise-OR>equals (`|=`) and Concatenate>equals (`.=`).

## 6. Bitwise Operators:

- Bitwise operators perform operations on the binary representation of the operands. These operators allow the evaluation and manipulation of specific bits within the integer.
- The following illustrates bitwise operators in PHP:

Example	Name	Result
<code>\$a and \$b</code>	and	If both bits are 1, the corresponding bit in the result is 1; otherwise, the corresponding bit is 0.
<code>\$a   \$b</code>	or (inclusive or)	If both bits are 0, the resulting bit is 0; otherwise, the resulting bit is 1.
<code>\$a ^ \$b</code>	xor (exclusive or)	If either of the bits in the pair, but not both, is 1, the resulting bit is 1; otherwise, the resulting bit is 0.
<code>~ \$a</code>	not	changes 1s to 0s and 0s to 1s in the binary representations of the operands
<code>\$a &lt;&lt; \$b</code>	shift left	Shift the bits of \$a \$b steps to the left (each step means "multiply by two")
<code>\$a &gt;&gt; \$b</code>	shift right	Shift the bits of \$a \$b steps to the right (each step means "divide by two")

## 7. Casting Operators:

- Casting operator changes type of its operand by force. PHP casting operators are (int), (float), (string), (bool), (array), and (object).

For example:

```
$a = "5"; // type of $a is string
$b = (int) $a; // type of $b is integer
```

## 8. Miscellaneous Operators:

(i) **Error suppression (@):** This operator is also called as error control operator, works only on expression. When the operator is used with expression then any error messages that might be generated by that expression will be ignored.

(ii) **Execution ('...'):** PHP supports one execution operator: backticks (''). Note that these are not single-quotes! PHP will attempt to execute the contents of the backticks as a shell command

```
$output = `ls -l`; // shell command for long list of files.
echo $output; // Displays long list of files from present directory
```

**(iii) Conditional (?:) Operator:** It is also called as ternary operator which first evaluates an expression for a true or false value and then executes one of the two given statements depending upon the result of the evaluation.

**Syntax:** expr1? expr2: expr 3

Meaning is, if expr1 is true, execute expr2 otherwise expr3.

---

**For example:**

```
<?
    $a = 10;
    $b = 15;
    $max = $a > $b? $a: $b;
    $min = $a < $b? $a: $b;
    echo "max = $max";
    echo "min = $min";
?>
```

**Output:**

```
max = 15
min = 10
```

---

### 1.7.3.2 Operator Precedence and Associativity

- The precedence and associativity of operators are significant characteristics of a programming language.
- Operator precedence is a characteristic of operators that determines the order in which they evaluate the operands surrounding them.
- The associativity characteristic of an operator specifies how operations of the same precedence are evaluated as they are executed.
- Associativity can be performed in two directions, left-to-right or right-to-left. Left-to-right associativity means that the various operations making up the expression are evaluated from left to right.
- Operator Precedence is the order in which operators in an expression are evaluated. For example, in the expression  $1 + 5 * 3$ , the answer is 16 and not 18 because the multiplication ("\*") operator has a higher precedence than the addition ("+") operator. Parentheses may be used to force precedence, if necessary. For instance:  $(1 + 5) * 3$  evaluates to 18.
- When operators have equal precedence their associativity decides how the operators are grouped. For example "-" is left-associative, so  $1 - 2 - 3$  is grouped as  $(1 - 2) - 3$  and evaluates to - 4. "=" on the other hand is right-associative, so  $$a = $b = $c$  is grouped as  $$a = ($b = $c)$ .

- Operators of equal precedence that are non-associative cannot be used next to each other, for example `1 < 2 > 1` is illegal in PHP. The expression `1 <= 1 == 1` on the other hand is legal, because the `==` operator has lesser precedence than the `<=` operator.
- The following table shows the operators with the highest precedence appear at the top of the table; those with the lowest appear at the bottom. Within an expression, higher precedence operators will be evaluated first.

Category	Operator	Associativity
Unary	<code>! ++ --</code>	Right to left
Multiplicative	<code>* / %</code>	Left to right
Additive	<code>+ -</code>	Left to right
Relational	<code>&lt; &lt;= &gt; &gt;=</code>	Left to right
Equality	<code>== !=</code>	Left to right
Logical AND	<code>&amp;and</code>	Left to right
Logical OR	<code>  </code>	Left to right
Conditional	<code>?:</code>	Right to left
Assignment	<code>= += -= *= /= %=</code>	Right to left

#### 1.7.4 Implicit Casting

- The conversion of a value from one type to another is called casting. Implicit casting is called as type juggling in PHP.
- The rules for the type juggling done by arithmetic operators are shown in following table:

Type of First Operand	Type of Second Operand	Conversion Performed
Integer	Floating point	The integer is converted to a floating-point number
Integer	String	The string is converted to a number; if the value after conversion is a floating-point number, the integer is converted to a floating-point number
Floating point	String	The string is converted to a floating-point number

- Some other operators have different expectations of their operands, and thus have different rules. For example, the string concatenation operator converts both operands to strings before concatenating them:

3. `2.74 // gives the string 32.74`

- We can use a string anywhere PHP expects a number. The string is presumed to start with an integer or floating-point number.
- If no number is found at the start of the string, the numeric value of that string is 0. If the string contains a period (.) or uppercase or lowercase e, evaluating it numerically produces a floating-point number.

For example:

```
"9 Lives" - 1;           // 8 (int)
"3.14 Pies" * 2;        // 6.28 (float)
"9 Lives." - 1;          // 8 (float)
"1E3 Points of Light" + 1; // 1001 (float)
```

## 1.7.5 Control Flow Statements

---

- The control statements are used to control the flow of execution of the program. This execution order depends on the supplied data values and the conditional logic.
- PHP supports a number of traditional programming constructs for controlling the flow of execution of a program.
- Conditional statements allow a program to execute different piece of code depending on the condition.
- In PHP the if else and switch statements are used to take decision based on the different condition.

### 1. if Statement:

- The if statement allows us to make decision based on one or more conditions and execute a piece of code conditionally.
- The if statement in PHP contain statements that are only executed when the condition is satisfied means the condition is True. The if statement cannot do anything if the condition is False.

- The **syntax** of the if statement:

```
if(expression)
    Statement...
```

- The alternative **syntax** for if statement:

```
if (condition)
{
    // put the PHP code here;
}
```

- If the expression evaluates to true, the code block inside the if statement is executed.

For example:

```
<?php
$a="Amar";
$b="Akbar";
```

```

if ($a==$b)
    echo "Both string are equal";
    echo "if block not executed";
?>

```

**Output:**

```
if block not executed
```

**2. if else Statement:**

- The if else statement allows us to execute one block of code if the specified condition is evaluated to True and another block of code, if it is, evaluated to False.
- The **syntax** of the if else statement:

```

if(expression)
    Statement
else
    Statement
• The alternative syntax for if else statement:
if (condition)
{
    // execute if condition evaluates to True...
}
else
{
    // execute if condition evaluates to False...
}
```

- We can enhance the if statement by adding else statement to run an alternative code block if the condition evaluates to false.
- 

**For example 1:**

```
<?php
$var1="php";
$var2="java";
if ($var1 == $var2)
    echo "Both the strings are equal";
else
    echo "Both the strings are not equal";
?>
```

**Output:**

```
Both the strings are not equal
```

---

**For example 2:**

```
<?php
    $marks=80;
    if ($marks >= 60 )
    {
        echo "You are passed".<br>;
        echo "Your marks= ".$marks."<br>";
        echo "First division";
    }
    else
    {
        echo "Failed";
    }
?>
```

**Output:**

```
You are passed
Your marks= 80
First division
```

---

- If we have more than one Boolean condition to test, we can use the if elseif statement. If we want to execute some code if one of the several conditions are True use the if elseif else statement.
- The **syntax** if elseif else statement is given below:

```
if(condition)
    // code to be executed if condition is True;
elseif (condition)
    // code to be executed if condition is True;
else
    // code to be executed if condition is False;
```

- The alternative **syntax** if elseif else statement:

```
if(condition)
{
    // code to be executed if condition is True;
}
elseif(condition)
{
    // code to be executed if condition is True;
}
else
{
    // code to be executed if condition is True;
}
```

---

**For example:**

```
<? php
    $x = 20;
    if($x > 0)
    {
        echo "$x is greater than zero"
    }
    else if($x == 0)
    {
        echo "$x is zero";
    }
    else
    {
        echo "$x is less than zero";
    }
?>
```

**Output:**

20 is greater than zero

---

- PHP also provide an alternative for conditional and looping control structure. In place of opening brace colon (:) is used and to close the block endif keyword is used (in this case).

**Syntax:**

```
if(condition);
    //code block to be executed if condition is true
    //...
else
    // code block to be executed if condition is false
    // ...
endif;
```

---

**For example:**

```
<?php
    $a=7
    if ($a == 5):
        echo "a equals 5";
        echo "...";
    elseif ($a == 6):
        echo "a equals 6";
        echo "!!!!";
```

---

```
    else:  
        echo "$a is neither 5 nor 6";  
    endif;  
?>
```

**Output:**

```
a is neither 5 nor 6
```

**Nested if Statement:**

- When one if statement contains another if statement then such type of structure is known as nested if. Nested if structure also helps in multi-way decision making where one condition depends on other.
- The nested if statement has the following **form/syntax**:

```
if(condition 1)  
{  
    if(condition 2) /* nested if */  
    {  
        Statement(s);  
    }  
    else  
    {  
        Statement(s);  
    }  
}  
else  
{  
    Statement(s);  
}
```

---

**Example for nested if statement:**

```
<!DOCTYPE html>  
<html>  
    <body>  
        <?php  
            $a=10;  
            $b=20;  
            if($a==10)  
            {  
                if($b==20)
```

```

        {
            echo "The addition is: ".($a+$b);
        }
    }

?>
</body>
</html>
```

**Output:**

The addition is: 30

**3. switch Statement:**

[April 19]

- Consider the case where value of a single variable may determine one of a number of different choices (e.g., the variable holds the username and we want to do something different for each user). The switch statement is designed for this situation.
- Switch statement is used to compare the value with multiple cases or values. All statements in a matching case are executed up to the first break keyword.
- If none match and ‘default’ is given, all statements following the default keyword are executed up to the first break keyword.
- The **syntax** for switch statement:

```
switch(variable)
{
    case value1:
        // code block 1
        break;
    case value2:
        // code block 2
        break;
    default:
        // default code block
}
```

- The alternative **syntax** for switch statement is:

```
switch(variable):
    case value1:
        // code block 1
        break;
    case value2:
        // code block 2
        break;
    default:
        // default code block
endswitch;
```

**For example:**

```
<?php  
    $var=date("D");  
    switch($var)  
    {  
        case "Sun":  
            echo "It is Sunday.>";  
            break;  
        case "Mon":  
            echo "It is Monday.>";  
            break;  
        case "Tue":  
            echo "It is Tuesday.>";  
            break;  
        case "Wed":  
            echo "It is Wednesday.>";  
            break;  
        case "Thu":  
            echo "It is Thursday.>";  
            break;  
        case "Fri":  
            echo "It is Friday.>";  
            break;  
        case "Sat":  
            echo "It is Saturday.>";  
            break;  
        default:  
            echo "Something wrong";  
    ?>
```

**Output:**

---

It is monday

---

## 1.7.6 Loops

---

- Loop in PHP is used to execute a statement or a block of statements, multiple times until and unless a specific condition is met.
  - Loops in PHP help the user to save both time and effort of writing the same code multiple times.
-

- In short, loops in PHP are used to execute the same block of code a specified number of times. PHP supports the following four loop types:
  1. while loop loops through a block of code if and as long as a specified condition is true.
  2. do while loop loops through a block of code once and then repeats the loop as long as special condition is true.
  3. for loop loops through a block of code a specified number of times.
  4. for each loop loops through a block of code for each element in an array.
- Let us see above loops in detail.

**1. while Loop:****[April 19]**

- The while loop is the simplest loop in PHP. He while loop will execute block of code until certain condition is met.
- The while loop statement checks the expression at the beginning of each iteration. If the expression evaluates to true, the code block inside the curly braces is executed. If the expression evaluates to false, the loop exits.

**Syntax:**

```
while(expression)
{
    //code block to be executed
}
```

- The following program displays number from 1 to 10.

```
<?php
    $i=1;
    while($i<=10)
    {
        echo $i . " ";
        $i++;
    }
?>
```

**Output:**

1 2 3 4 5 6 7 8 9 10

- The alternative **syntax** for while statement is:

```
while(expression):
    // Statement;
    ...;
endwhile;
```

**For example:**

```
<?php
    $i=1;
    while($i<=10):
        echo $i . " ";
        $i++;
    endwhile;
?>
```

**Output:**

1 2 3 4 5 6 7 8 9 10

**2. do while Loop:**

- A do while loop in PHP works same as while, except that the expression is evaluated at the end of each iteration.
- The **syntax** for do while loop:

```
do{
    //code block to be executed
}while(expression);
• Use a do while loop to ensure that the loop body is executed at least once - it then will
repeat the loop as long as a condition is true.
• The following program displays number from 1 to 10 using do while loop:
<?php
    $i=1;
    do {
        echo $i . " ";
        $i++;
    } while($i<=10);
?>
```

**3. for Loop:**

- The for loop is used when we know how many times we want to execute a statement or a block of statements.

**Syntax:**

```
for(init_expr; condition_expr;increment_expr)
{
    //code block to be executed
}
```

- At the beginning, the counter initialization occurs only once. Each time through the loop, the expression condition is tested.
- If it is true, the body of the loop is executed; if it is false, the loop ends. The expression increment/decrement is evaluated after the loop body runs.

- The following program displays number from 1 to 10 using for loop.

```
<?php  
    for($i=1; $i<=6; $i++)  
    {  
        echo $i . " ";  
    }  
?>
```

**Output:**

1 2 3 4 5 6

- The alternative **syntax** for 'for' loop :

```
for(initialization; condition; increment):  
    Statement;  
    ...;  
endfor;
```

**For example:**

```
<?php  
    for ($i=1; $i<=5; $i++):  
        echo $i . " ";  
    endfor;  
?>
```

**Output:**

1 2 3 4 5

**4. foreach Loop:**

- The foreach construct provides an easy way to iterate over arrays. The foreach loop works only on arrays and objects, and will issue an error when we try to use it on a variable with a different data type or an uninitialized variable.
- There are two syntax:

```
foreach (array_expression as $value)  
    Statement
```

**OR**

```
foreach (array_expression as $key => $value)  
    Statement
```

- The first form loops over the array given by array\_expression. On each iteration, the value of the current element is assigned to \$value and the internal array pointer is advanced by one (so on the next iteration, you'll be looking at the next element).
- The second form will additionally assign the current element's key to the \$key variable on each iteration.

**For example:**

```
<?php
    $arr = array(1, 2, 3, 4);
    foreach ($arr as $value)
    {
        echo $value . " ";
    }
?>
```

**Output:**

1 2 3 4

- The alternative **syntax** for foreach loop:

```
foreach (array_expression as $value):
    // ...
endforeach;
```

- Just like other programming languages, PHP also have two statements break and continue to control the loop. These statements are known as jumping statement or flow of transfer in the program.

### 1. **break Statement:**

**[April 17]**

- The PHP break keyword is used to terminate the execution of a loop prematurely.
- As the break statement encountered inside a loop, it immediately terminated the loop statement and transferred the control to the next statement, followed by a loop to resume the execution.

**Syntax:**

```
for(...)
{
    // Statements;
    if(condition)
    {
        break;
    }
    // Statements; /* Never executed after executing break statement*/
}
```

- If the condition is true, the break statement will be executed and the loop will be break. It will skip all the remaining statement of the loop.
- In the following code, \$i never reach a value of 6, because the loop is stopped once it reaches 5:

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

**Output:**

1 2 3 4 5

- Optionally, we can put a number after the break keyword, indicating how many levels of loop structures to break out of. In this way, a statement buried deep in nested loops can break out of the outermost loop. For example:

```
<?php
    $i = 1;
    while ($i <= 10)
    {
        $j = 1;
        while ($j <= 10)
        {
            if ($j == 5)
                break 2; // breaks out of two while loops
            $j++;
        }
        $i++;
    }
    echo $i;
    echo "<br>";
    echo $j;
?>
```

**Output:**

1  
5

**2. continue Statement:****[April 17]**

- The PHP continue keyword is used to skip the current iteration of a loop but it does not terminate the loop.
- As the continue statement is encountered in a loop, it skips the current iteration of the loop and transfers the control to the beginning of the loop for further execution of the loop.

**Syntax:**

```
for(initialisation;condition;increment/decrement)
{
    ...
    if (True Condition) //Continues Loop with the Next Value
    continue;
}
```

- The following program displays only odd numbers from the array:

```
<?php
    $a = array(1,2,3,4,5);
    foreach($a as $value)
    {
        if($value%2 == 0)
        continue;
        echo $value . " ";
    }
?>
```

**Output:**

1 3 5

- Like break statement, you can continue through an optional number of levels of loop structure.

```
while(condition1)
{
    while (condition2)
    {
        if (condition3)
        continue 2; // continues through two levels, i.e. to condition1
        // ...
    }
    // ...
}
```

**3. exit and return Statements:**

- The exit statement ends execution of the script as soon as it is reached. The return statement returns from a function or from the script.
- The exit statement takes an optional value. If this is a number, it's the exit status of the process. If it's a string, the value is printed before the process terminates. The exit( ) construct is an alias for die( ).

**For example:** \$a = fopen("a.txt", "r") OR die("Can not open");

- If file a.txt does not exist, and we try to open the file in read mode then the above statement executes the right hand part of the 'or' operator. Hence the die construct first displays the message and terminates the script.

**4. include and require:**

- PHP include construct allow you to create modular and reusable code. PHP provides four construct that help you to organize the script files into modules and reuse them in another script file.

- Those constructs are include(), require(), include\_once() and require\_once().
- PHP include construct allows us to include code from a script file inside another script file.
- A common use of include is to separate page-specific content from general site design. Common elements such as headers and footers go in separate HTML files and each page then looks like:

```
<? include 'header.html';?>
    content
<? include 'footer.html';?>
```

- PHP provides the require() function that works the same as the include() function with a slight difference.
- The difference between the include() and require() is that if the file to be included cannot be found include() raises a PHP warning while require() raises a fatal error and terminates the script.
- If a program uses 'include' or 'require' to include the same file twice, the file is loaded and the code is run or the HTML is printed twice. This can result in errors about the redefinition of functions or multiple copies of headers or HTML being sent.
- To prevent these errors from occurring, use the include\_once() and require\_once() constructs. They behave the same as include and require the first time a file is loaded, but quietly ignore subsequent attempts to load the same file.

[April 17]

# PRACTICE QUESTIONS

## **Q.I Multiple Choice Questions:**



23. The PHP syntax is most similar to:

- (a) VBScript
- (b) JavaScript
- (c) Perl and C
- (d) All of the above

24. The script that executes at the browser side is called as,

- (a) Client side scripting
- (b) Server side scripting
- (c) Browsing side scripting
- (d) None of the above

25. Find the output of following code:

```
<?php  
    $x = 8;  
    $y = 8.0;  
    if ($x === $y)  
        echo "Same";  
    else  
        echo "Different";  
?>
```

- (a) 8 === 8
- (b) Same
- (c) Different
- (d) 1

26. Find the output of following code:

```
<?php  
    for($num = 1; $num <= 10; $num += 2)  
    {  
        echo "$num "; }  
?>
```

- (a) 1 3 5 7
- (b) 1 2 3 4 5
- (c) 9 7 5 3 1
- (d) error

27. Which of the conditional statements is/are supported by PHP?

- (a) if statements
- (b) if-else statements
- (c) if-elseif statements
- (d) switch statement

28. Which of the following is the correct syntax of PHP?

- (a) <?php>
- (b) ?php ?
- (c) <php >
- (d) <?php?>

29. Which one is not a data type in PHP?

- (a) Resources
- (b) Object
- (c) Null
- (d) void

30. The while loop is an \_\_\_\_\_ control loop.

- (a) exit
- (b) exist
- (c) easy
- (d) entry

31. Which is not Rules for Variable Declaration:

- (a) A PHP variable must start with (\$) dollar sign.
- (b) A variable name must start with an alphabet letter or the underscore (\_).
- (c) A variable name can start with a number.
- (d) PHP variable can be of any length.

32. What is the output of following code?

```
$a = 5;  
$b = &$a;  
unset($a);  
$b = $b + 2;  
echo $b;
```

- (a) 5 (b) 7
- (c) 9 (d) 2

33. Which of following operator statement is not correct?

- (a) \$a == \$b (b) \$a <> \$b
- (c) \$x=\$a > \$b? \$a: \$b; (d) \$a=+\$b;

34. Website is a collection of related web pages that may contain,

- (a) text (b) exist images
- (c) audio and video (d) All of mentioned

35. HTML documents are simply a,

- (a) text file (b) exist
- (c) easy (d) entry

36. Which is an following empty tag.

- (a) <b> (b) <br>
- (c) <em> (d) <sub>

37. Which tag defines a division or a section in an HTML document?

- (a) <sup> (b) <hr>
- (c) <div> (d) <span>

38. Which identifies location of a web page on the Internet?

- (a) WWW (b) Browser
- (c) Web server (d) URL

39. One list inside another list is known as,

- (a) ordered list (b) nested list
- (c) unordered list (d) sorted list

40. The basic syntax of PHP is,

- (a) <?php...?> (b) <script language = "PHP">...</script>
  - (c) <?...?> (d) All of mentioned
-



52. HTTP is,
- (a) connectionless
  - (b) media independent
  - (c) stateless
  - (d) All of mentioned
53. An HTTP client sends an HTTP request to a server in the form of which message.  
After receiving and interpreting a request message, a server responds with an HTTP response message
- (a) request
  - (b) response
  - (c) Both (a) and (b)
  - (d) None of mentioned
54. Which statement is used when we want to execute a set of code when a condition is true and another if the condition is false.
- (a) if
  - (b) if...else
  - (c) switch
  - (d) All of mentioned

### Answers

1. (c)	2. (a)	3. (c)	4. (d)	5. (b)	6. (b)	7. (d)	8. (a)	9. (b)	10. (d)
11. (a)	12. (d)	13. (a)	14. (c)	15. (d)	16. (a)	17. (d)	18. (c)	19. (c)	20. (b)
21. (b)	22. (b)	23. (c)	24. (a)	25. (c)	26. (a)	27. (d)	28. (d)	29. (d)	30. (d)
31. (c)	32. (b)	33. (d)	34. (d)	35. (a)	36. (b)	37. (c)	38. (d)	39. (b)	40. (d)
41. (a)	42. (b)	43. (a)	44. (d)	45. (d)	46. (a)	47. (d)	48. (c)	49. (b)	50. (d)
51. (c)	52. (d)	53. (a)	54. (b)						

### Q.II Fill in the Blanks:

1. \_\_\_\_\_ feature allows a web developer to render graphics on the fly.
2. The \_\_\_\_\_ defines a hyperlink in which the href attribute specifies the URL the page.
3. \_\_\_\_\_ attribute specifies the address of the document's application cache manifest and the value must be a valid URL.
4. An ordered list starts with the \_\_\_\_\_ tag.
5. Variables declared outside function are by default \_\_\_\_\_ variables and can accessed from any part of program.
6. To create a line break in PHP, we have to put \_\_\_\_\_ code in the echo statement.
7. PHP uses \_\_\_\_\_ to separate simple statements.
8. The \_\_\_\_\_ structure of a programming language is the set of basic rules that governs how we write programs in that language.
9. \_\_\_\_\_ in PHP is used to separate tokens in PHP source file.
10. PHP allows us to use dynamic variable names called as \_\_\_\_\_variables.

11. PHP code is executed on the \_\_\_\_.
  12. CSS is the language we use to style an \_\_\_\_ document.
  13. The HTML5 \_\_\_\_ refers to the semantic tags that provide meaning to an HTML page.
  14. PHP is a widely-used, \_\_\_\_ source scripting language
  15. PHP files have extension \_\_\_\_.
  16. The HTTP Web \_\_\_\_ (client) initiates an HTTP request and after a request is made, the client waits for the response. The Web server (server) processes the request and sends a response back after which client.
  17. HTTP is based on the \_\_\_\_ architecture model.
  18. The style definitions are normally saved in external \_\_\_\_ files.
  19. \_\_\_\_ can be defined as the range of availability/visibility a variable has to the program in which it is declared.
  20. To use an external style sheet, add a \_\_\_\_ attribute to in the <head> section of the HTML page.
  21. A \_\_\_\_ is the portion of a program that exists only for the human reader and stripped out before displaying the programs result.
  22. The \_\_\_\_ specifies a container for navigation links.
  23. An HTTP client is a program (Web browser) that establishes a connection to a server for the purpose of sending one or more HTTP \_\_\_\_ messages.
  24. The basic \_\_\_\_ operator in PHP is "=".
  25. All variables in PHP are denoted with a leading \_\_\_\_.
  26. The \_\_\_\_ element identifies a self-contained piece of content.
  27. \_\_\_\_ is a stylesheet language that describes the presentation of an HTML (or XML) document.
  28. The PHP \_\_\_\_ operators are used to increment a variable's value.
  29. \_\_\_\_ is foundation for data communication for the World Wide Web (i.e. internet) since 1990.
  30. Loops in PHP are used to execute the same block of code again and again, as long as a certain condition is \_\_\_\_.
  31. In PHP a \_\_\_\_ value cannot change during the execution of the script.
  32. The \_\_\_\_ element is used to identify content that is related to the primary content of the webpage, but does not constitute the primary content of the page.
  33. Internal styling is defined in the <head> section of an HTML page, within a \_\_\_\_ element.
  34. The CSS \_\_\_\_ property defines the text color to be used for the HTML element.
  35. Operators in PHP are used to perform \_\_\_\_ on variables and values.
  36. \_\_\_\_ give names to functions, variables, and classes etc. in PHP.
-

37. The CSS font-family property defines the \_\_\_\_\_ to be used for the HTML element.  
 38. In CSS, the term \_\_\_\_\_ is used when talking about design and layout.

### Answers

1. Canvas	2. anchor <a> tag	3. Manifest	4. <ol>	5. global
6.  	7. semicolons (;)	8. lexical	9. White space	10. variable
11. server	12. HTML	13. semantics	14. open	15. .php
16. browser	17. client-server	18. .css	19. Scope	20. link
21. comment	22. <nav>	23. request	24. assignment	25. \$
26. <article>	27. CSS	28. increment	29. HTTP	30. true
31. constant	32. <aside>	33. <style>	34. color	35. operations
36. Identifiers	37. font	38. box model		

### Q.III State True or False:

1. HTML is a platform-dependent language.
2. The singular tag is also known as a stand-alone tag or empty tag.
3. The vlink attribute specifies the color of visited links in a document.
4. The <sub> tag defines smaller text.
5. In ordered list<ol>tag, each list item starts with the <li> tag.
6. PHP is a server side scripting language which is used to create dynamic and interactive web pages.
7. PHP script is executed much slower than other scripting languages such as JSP and ASP.
8. A PHP program consists of tokens.
9. A constant value cannot change during the execution of the script.
10. The bool() function is used to test whether value is Boolean or not.
11. A constant is case-sensitive by default.
12. In PHP the const keyword also defines constants at compile time.
13. Web applications are usually based on the client-server architecture.
14. PHP support unsigned integers.
15. Objects are the special instances of the classes that are created by user.
16. A PHP script starts with <? php and ends with ?>.
17. PHP is a server side scripting language that is embedded in HTML.
18. PHP is a not case sensitive language.
19. The <article> element represents an independent article in a web page.
20. Semantic elements mean elements with a meaning. Semantic elements have a simple and clear meaning for both, the browser and the developer.

21. CSS is used to control the style of a web document in a simple and easy way.
22. An external style sheet is used to define the style for many pages.
23. The Web server requests for the web document then the Web browser (clients) accepts and response to the request made by a Web browser for a web document.
24. The <section> element defines a section in a document. A section is a thematic grouping of content, typically with a heading.
25. An HTTP "server" is a program ( Web server) that accepts connections in order to serve HTTP requests by sending HTTP response messages.
26. By default, a constant is case-sensitive.
27. In PHP, a variable starts with the \$ sign, followed by the name of the variable.
28. You can put your CSS rules into an HTML document using the <style> element. This tag is placed inside the <head>...</head> tags. "
29. The <section> element specifies independent, self-contained content.
30. The CSS box model is essentially a box that wraps around every HTML element.
31. PHP is a server side scripting language.

### Answers

1. (F)	2. (T)	3. (T)	4. (F)	5. (T)	6. (T)	7. (F)	8. (T)	9. (T)	10. (F)
11. (T)	12. (T)	13. (T)	14. (F)	15. (T)	16. (T)	17. (T)	18. (F)	19. (T)	20. (T)
21. (T)	22. (T)	23. (F)	24. (T)	25. (T)	26. (T)	27. (T)	28. (T)	29. (F)	30. (T)
31. (T)									

### Q.IV Answer the following Questions:

#### (A) Short Answer Questions:

1. What is HTML?
2. What is PHP?
3. Define Web browser.
4. Define Web server.
5. Give any two features of HTML?
6. What is the use of PHP?
7. What is CSS?
8. Give the purpose of HTML?
9. Which HTML tag is used to add video in page?
10. What is HTTP?
11. HTTP uses client-server architecture. Justify true or false.
12. Which semantic elements used by HTML5 in web page.
13. Explain the role of web browser and web server.
14. Give the ways to add CSS in
15. How to add PHP in HTML page?

16. Compare HTML and PHP any two points.
17. Define variables in PHP.
18. What is literal in PHP?
19. List uses of PHP.

**(B) Long Answer Questions:**

1. Define the following terms:
  - (i) Web page,
  - (ii) Web site,
  - (iii) Web browser, and
  - (iv) Web Server.
2. Why HTML is called as language for Web? Explain in detail.
3. With the help of diagram describe HTML elements.
4. How to add style to the HTML document? Explain with example.
5. What is meant by link? Explain with example.
6. What is image? Enlist various image file formats.
7. What is table? How to create in HTML? Explain with example.
8. What is list in HTML? How to nested list? Explain with example.
9. Write the HTML codes for the following:

I have :

- One car
- One Motorcycle
- One Bicycle

I also have:

- One Bookshelf
- One Computer
  - Laptop
  - Notebook
  - Tab
- One CD-Player

10. What is a frame? How to create it? Explain with example.
  11. What is a frameset?
  12. Explain different attributes that can be used with frame elements.
  13. What is a form? How to create a form? Explain with example.
  14. What are the semantic elements used by HTML5? Explain with example.
  15. How web browser and web server communicate with each other? Explain diagrammatically.
  16. With the help of diagram describe working of HTTP.
  17. Give message formats for HTTP request and response messages.
-

18. Write code for following form:

<b>Nirali Prakashan</b>	
Name of employee :	<input type="text"/>
Address :	<input type="text"/>
Phone No. :	<input type="text"/>
<input type="button" value="Submit"/>	

19. What is CSS? Give its syntax. Explain CSS ID in detail.
20. What are the types of CSS? Explain embedded style sheet with example.
21. Explain web server in detail. What are the types of web servers?
22. How web browser works? Enlist types of web browsers.
23. How to embed video, audio in HTML? Explain with example.
24. With the help of diagram describe working of PHP.
25. Define variable. How to create it? Explain with example.
26. Write note on variable scope in PHP.
27. Write the difference between break and continue statement of PHP with example.
28. Write PHP program to calculate area of circle.
29. Explain any two control statements in PHP with syntax and example.
30. Explain lexical structure of PHP in detail.
31. Explain constants in PHP with example.
32. What are the decision statements used by PHP? Explain two of them with example.
33. Describe syntax of PHP with example.
34. What is operator? Which operators used by PHP? Explain with example.
35. Describe data types in PHP in detail.
36. What is layout? How to create layouts in HTML? Explain with example.
37. With the help of diagram describe CSS box model.

## UNIVERSITY QUESTIONS AND ANSWERS

**April 2016**

1. Give two examples of web browsers. [1 M]
- Ans.** Refer to Section 1.3.2.
2. Which operator is used to compare data as well as data type? [1 M]
- Ans.** Refer to Section 1.7.3.1.
3. Write role of web server. [2 M]
- Ans.** Refer to Section 1.3.1.
4. Write purpose of resource data type with suitable example. [1 M]
- Ans.** Refer to Section 1.7.1, Point (7).

**October 2016**

1. Discuss the scope of a variable in PHP with an example.

[5 M]

**Ans.** Refer to Section 1.7.2.3.

2. What is the role of web browser and web server?

[3 M]

**Ans.** Refer to Section 1.3.1 and 1.3.2.

3. Explain the concept of variable variables with an example.

[2 M]

**Ans.** Refer to Section 1.7.2.1.

---

**April 2017**

1. State the features of PHP.

[1 M]

**Ans.** Refer to Section 1.5.2.

2. Write the difference between break and continue statement of PHP with an example.

[5 M]

**Ans.** Refer to Pages 1.112 and 1.114.

3. Give two examples of web browsers.

[1 M]

**Ans.** Refer to Section 1.3.2.

---

**October 2017**

1. State the advantages of PHP.

[1 M]

**Ans.** Refer to Section 1.5.

2. “Keywords are case sensitive in PHP” Justify True or False.

[1 M]

**Ans.** Refer to Section 1.6.8.

3. What is data type? List different data types in PHP. Explain any two.

[5 M]

**Ans.** Refer to Section 1.7.1.

---

**April 2018**

1. How do you define constant PI with value 3.142 in PHP?

[1 M]

**Ans.** Refer to Section 1.6.7.

2. List any four web browsers name.

[1 M]

**Ans.** Refer to Section 1.3.2.

3. What is data type? List different Data types in PHP? Explain any two.

[5 M]

**Ans.** Refer to Section 1.7.1.

---

**October 2018**

1. What is the role of web server and web browser?

[1 M]

**Ans.** Refer to Sections 1.3.1 and 1.3.2.

2. Explain scope of a variable in PHP with suitable example.

[5 M]

**Ans.** Refer to Section 1.7.2.3.

---

**April 2019**

1. What is web server?

**[1 M]**

**Ans.** Refer to Section 1.3.1.

2. Explain the execution of PHP script using diagram.

**[5 M]**

**Ans.** Refer to Section 1.5.

3. Explain the following statements with syntax and example:

(i) while statement (ii) switch statement.

**[4 M]**

**Ans.** Refer to Sections 1.7.5, Point (3) and 1.7.6, Point (1).

■ ■ ■

# Function and String

## Objectives...

- To understand Basic Concepts of Function and String in PHP
- To learn Function Declaration, Types, Definition, Arguments etc.
- To learn Basic Concepts of Strings like Declaration, String Functions, etc.

### 2.0 INTRODUCTION

- A function in PHP is a block of statements that can be used repeatedly in a program. A string in PHP is a collection of characters.
- A function in PHP is a reusable block of code that performs a specific action or task. It takes input from the user in the form of parameters, performs certain actions, and gives the output.
- Functions can either return values when called or can simply perform an operation without returning any value.

### 2.1 OVERVIEW OF FUNCTION

- A function is a named block of code that is defined to perform a specific task, possibly acting upon a set of values given to it, or parameters and may or may not return a value.
- A function can be called from another part of the program. The code inside a function is not executed until the function is called.
- There are two types of functions i.e. built-in function and user defined functions.
  1. **Built-in Functions** are also called as pre-defined functions that perform a wide range of operations. PHP is very rich in terms of Built-in functions.
  2. **User Defined Functions** are defined by user as per their need or requirements.
- Functions are useful because they contribute to rapid, reliable, error-reducing coding. A function can be called many times in a page but it is compiled only once.
- As a function is written only once and removes the need to write the codes for the same task every time it requires, it reduces number of bugs and saves time for programming.
- As functions separate codes that perform a specific task, it improves readability.

**Advantage of PHP Functions:**

1. **Code Reusability:** PHP functions are defined only once and can be invoked many times, like in other programming languages.
2. **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.
3. **Easy to Understand:** PHP functions separate the programming logic. So it is easier to understand the flow of the program because the program logic is divided in the form of functions.
4. **Better Code Organization:** PHP functions allow us to group blocks of related code that perform a specific task together.

**2.1.1 Defining a Function**

---

- A function is a self-contained block of code that performs a specific task. To define a function In PHP following **syntax** is used:
 

```
function [&] function_name([parameter[,...]])  
{  
    //Code to be executed...  
}
```
- A function definition starts with the keyword ‘function’ followed by the name of function with parentheses (). The [&] is optional and if it is used then the function returns reference.
- A function name can be any string that begins with a alphabet letter or underscore (\_). All the PHP code in function should be put inside curly brackets (braces { }) containing the code to be executed each time that function gets called.
- Function names are case-insensitive. It means count(), Count() and COUNT() refer to the same function. By convention, PHP built-in functions are in lowercase.

**Example for function:**

```
<?php  
/* Defining a PHP Function */  
function writeMsg()  
{  
    echo "Hello PHP";  
}  
/* Calling a PHP Function */  
writeMsg();  
?>
```

**Output:**

Hello PHP

---

## 2.1.2 Calling a Function

- Calling a function is simple – just write the name of the function followed by a pair of parentheses.
  - We can call a function in PHP by using the following **syntax**:
- ```
$some_value=function_name([parameter, ...]);
```
- After calling a function, the code/statements inside it are executed. After the function execution is completed, the program control returns to the point where the function was called.
  - Fig. 2.1 shows how caller and receiver functions work together.

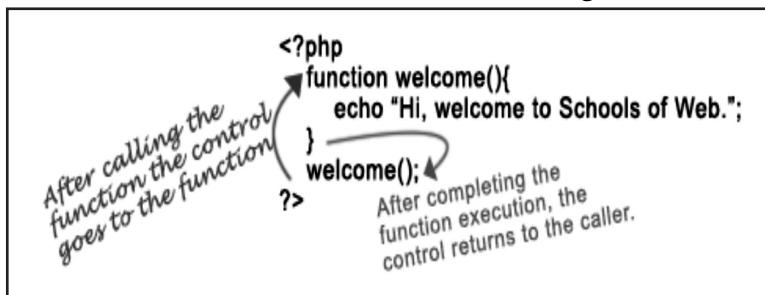


Fig. 2.1: How Caller and Receiver Functions Work Together

**For example:**

```
<?php
function welcome()
{
    echo "Hi, welcome to Schools of Web.";
}
welcome(); // Function welcome() is called here.
?>
```

**Output:**

Hi, welcome to Schools of Web.

- Explanation of above Program:** The function welcome() is defined. In this stage nothing happens. When the function is called, the interpreter finds it and enters into it, executes the statement(s) inside it. Here it prints the sentence “Hi, welcome to Schools of Web.”.

## 2.1.3 Function Parameters

[Oct. 18, April 19]

- Information can be passed to functions through arguments. Arguments (or parameters) are specified after the function name, inside the parentheses () .
- If a function accepts more than one parameter, each parameter has to be separated by a comma (,). The arguments are evaluated from left to right.

**For example:**

```
function strcat($left, $right)
{
    $combined_string = $left . $right;
    echo $combined_string;
}
```

- PHP supports passing arguments by value (the default), passing by reference, and default argument values. Variable-length argument lists are also supported.
  - By default, function arguments are passed by value (so that if the value of the argument within the function is changed, it does not get changed outside of the function).
  - To allow a function to modify its arguments, they must be passed by reference. When a function argument is passed by reference, changes to the argument also change the variable that was passed in.
  - We can pass an argument by reference by adding an ampersand (&) to the variable name in either the function call or the function definition.
- 

**For example:**

```
<?php
    function doubler(&$value)
    {
        $value = $value * 2;
    }
    $a = 3;
    doubler($a);
    echo $a; // Outputs 6
?>
```

---

## 2.1.4 Returning Values

---

- A function can return a value using the return statement. Any type may be returned, including arrays and objects.
- When PHP encounters the return statement, it terminates the function immediately and returns the value back.

**For example:**

```
<?php
    function square($num)
    {
        return $num * $num;
    }
    echo square(4); // outputs 16
?>
```

- A function cannot return multiple values, but similar results can be obtained by returning an array.

```
<?php
    function day_name()
    {
        $day1 = "Monday";
        $day2 = "Tuesday";
        $day3 = "Wednesday";
        return array($day1, $day2, $day3);
    }
    $days = day_name();
    echo $days[0] . " " . $days[1] . " " . $days[2];
?>
```

**Output:**

Monday Tuesday Wednesday

- Function day\_name() creates an array that has three elements in it and returns to its caller.
- When the function is called, the function returns the array to the caller assigns it in the \$days variable. The next line prints the values of the array elements.

## 2.1.5 Function Body

- The code inside the curly braces {} is a function body. We can put PHP code, HTML code or mixed PHP and HTML code inside the body.
- For example, the following function displays the header of a web page:

```
function display_header($header)
{
    echo "<h1>" . $header . "</h1>";
}
```

## 2.2 DEFAULT PARAMETERS

- We can specify a default argument value in function. While calling PHP function, if we do not specify any argument, it will take the default argument.
- PHP function allows us to pass default values to its parameters. To specify a default value for a parameter, we assign the parameter a scalar value in the function declaration.

**Example:**

```
<?php
    function makecoffee($type = "cappuccino")
    {
        return "Making a cup of $type.\n";
    }
    echo makecoffee();
    echo makecoffee(null);
    echo makecoffee("espresso");
?>
```

**Output:**

```
Making a cup of cappuccino.
Making a cup of .
Making a cup of espresso.
```

- The default value must be a constant expression, not (for example) a variable, a class member or a function calls.
- Note that when using default arguments, any defaults should be on the right side of any non-default arguments; otherwise, things will not work as expected.

## 2.3 VARIABLE PARAMETERS

- A function may require a variable number of arguments. A function may require a variable number of arguments instead of having fixed number of arguments.
- PHP has support for variable-length argument lists in user-defined functions. This is implemented using the "..." token in PHP 5.6 and later versions and using the func\_num\_args(), func\_get\_arg(), and func\_get\_args() functions in earlier versions.
- To declare a function with a variable number of arguments, keep parameter block completely empty.

```
function xyz()
{
    // some code ...
}
```

- In PHP following are the three functions are available to retrieve the parameters passed to it:
  1. **func\_get\_args()** returns an array containing all parameters passed to the function.

[April 18]

2. `func_num_args()` returns the number of parameters provided to the function.

[Oct. 17]

3. `func_get_args()` returns specific arguments from the parameters.

**Example for displaying the sum of all the parameter:**

```
<?php
    function sum()
    {
        $count = 0;
        for($i = 0; $i < func_num_args( ); $i++)
        {
            $count += func_get_arg($i);
        }
        return $count;
    }
    echo sum(1, 2, 3, 4);
    echo sum(1, 4);
    echo sum();
    echo sum(10, 20, 30);
?>
```

**Output:**

```
10
5
0
60
```

## 2.4 MISSING PARAMETERS

[April 17, Oct. 17]

- We can pass any number of arguments to the function. If we do not pass any value to the parameter, the parameter remains unset and a warning is displayed for each of them.

**Example:**

```
<?php
    function xyz($a, $b, $c)
    {
        if (isset ($a))
            echo "a is set <br>";
```

```

if (isset ($b))
    echo "$b is set <br>";
if (isset ($c))
    echo "c is set <br>";
}
echo "with 3 arguments <br>";
xyz(1, 2, 3);
echo "with 2 arguments<br>";
xyz(1, 2);
echo "with 1 argument<br>";
xyz(1);

?>

```

**Output:**

```

with 3 arguments
1 is set
2 is set
3 is set
with 2 arguments
1 is set
2 is set
Warning: Missing argument 3 for xyz()
with 1 argument.
1 is set.
Warning: Missing argument 2 for xyz()
Warning: Missing argument 3 for xyz()

```

- In above program:
  1. To check if parameter is missing or not, we can use built in function `isset()`.
  2. The `isset()` functions returns true if the value is set to the argument passed to it, otherwise it returns result as false.

**[April 16]****2.5 VARIABLE FUNCTION AND ANONYMOUS FUNCTION**

- If name of a variable has parentheses (with or without parameters in it) in front of it, PHP parser tries to find a function whose name corresponds to value of the variable and executes it. Such a function is called variable function.
- The variable function feature of PHP is useful in implementing callbacks, function tables etc.
- Anonymous function is a function without any user defined name. Such a function is also called closure or lambda function.

## 2.5.1 Variable Function

[April 17, Oct. 17]

- PHP supports the concept of variable functions. This means that we can call function based on value of a variable.
- If a variable name has round parentheses appended to it, PHP will look for a function with the same name as whatever the variable evaluates to, and will attempt to execute it.
- The variable function does not work with echo(), print(), unset(), isset() language constructs.
- PHP allows us to store a name of a function in a string variable and use the variable to call the function. Variable functions are useful in cases that we want to execute functions dynamically at run-time.
- The following example demonstrates how to use variable functions:

```
<?php
    function find_max($a,$b)
    {
        return $a > $b? $a: $b;
    }
    function find_min($a,$b)
    {
        return $a < $b? $a: $b;
    }
    $f = 'find_max';
    echo $f(10,20); // Outputs 20
    $f = 'find_min';
    echo $f(10,20); // Outputs 10
?>
```

- First, we defined two functions namely, find\_min() and find\_max(). Then, we called those function based on the value of string variable \$f.
- Notice that we need to put parentheses after the variable in order to call the function specified by the value of the variable.

## 2.5.2 Anonymous Function

[Oct. 17, 18, April 18]

- In PHP we can define a function that has no specified name called as an anonymous function or a closure.
- An anonymous function also called as lambda function. We often use anonymous functions as values of callback parameters.
- The anonymous function is created using the create\_function(). It takes following two parameters:
  - First parameter is the parameter list for anonymous functions
  - Second parameter contains the actual code of the function. The function name is randomly generated and returned.

- The **syntax** for anonymous function (or lambda function) using `create_function()`:

```
$f_name = create_function(args_string, code_string);
```

**For example:**

```
<?php
    $add = create_function('$a,$b', 'return($a+$b);');
    $r = $add(2,3);
    echo $r;
?>
```

**Output:**

5

## 2.6 TYPES OF STRINGS IN PHP

[April 16, 17, 18, Oct. 16, 18]

- PHP string is a sequence of characters and used to store and manipulate text.
- A “string” of text can be stored in a variable in much the same way as a numeric value but the assignment must surround the string with quote marks (“...” or ‘...’) to denote its beginning and end like "Hello world!" or 'Hello world!'.
- Strings are very useful to store names, passwords, address, and credit-card number and so on. For that reason, PHP has large number of functions for working with strings.
- There are three different ways to write string in PHP as explained below:

### 1. Single-Quoted String:

- In this method, string is enclosed with single quotation mark ('...'). It is the easiest way to specify string in PHP.

**For Example:** 'Hello PHP' 'Computer Basics' 'Pune' etc.

- The limitation of single-quoted string is that variables are not interpolated.

**For example:**

```
<?php
    $name = 'Amar;
    $str = 'Hello $name'; // Single-quoted string
    echo $str;
?>
```

**Output:**

Hello \$name

- In the above output the value of variable \$name is not printed.

### 2. Double-Quoted String:

- It is the most commonly used quote to express string. In this method, characters are enclosed with double quotation marks (“...”).
- PHP interpreter interprets variables and special characters inside double quotes.

- In the following example, the above example is re-written using double quotes.

**For example:**

```
<?php
    $name = 'PHP';
    $str = "Hello $name"; // Double-quoted string
    echo $str;
?>
```

**Output:**

```
Hello PHP
```

---

### 3. Here Documents:

- Single-quoted and double-quoted strings allow string in single line. To write multiline string into a program heredoc is used.

**Rules of using heredoc Syntax:**

- Heredoc syntax begins with three less than signs (<<<) followed by identifier (a user defined name). The identifier can be any combination of letters, numbers, underscores but the first character must be a letter or an underscore.
- The string begins in the next line and goes as long as it requires.
- After the string, the same identifier that is defined after the (<<<) signs in the first line should be placed at the end. Nothing can be added in this last line except a semicolon after the identifier and it is optional.
- Space before and after <<< is essential.

**Syntax:**

```
$var_name = <<< identifier
// String statement
// String statements
    identifier;
```

---

**For example:**

```
<?php
    $str = <<< EOD
    PHP is suited for web development and can be embedded into HTML.\n
    PHP is server-side scripting language
    EOD;
    echo $str;
?>
```

**Output:**

```
PHP is suited for web development and can be embedded into HTML. PHP is
server-side scripting language.
```

### 4. Nowdoc syntax:

- If heredoc syntax works similar to double quote, then nowdoc syntax works similar to single quote.
- Like single quote, no variable inside the nowdoc is interpreted.

**Rules of using nowdoc Syntax:**

- (i) Nowdoc supports all the syntax rules of heredoc except the starting name.
- (ii) It must be enclosed by single quotes.

**For example:**

```
<?php
    $a = 5;
    $str = <<<'EOD'
        PHP is suited for web development and
        can be embedded into HTML $a.\n
        PHP is server-side scripting language.
    EOD;
    echo $str;
?>
```

**Output:**

PHP is suited for web development and can be embedded into HTML \$a.  
\nPHP is server-side scripting language.

- Single and double quotes in heredoc are passed through.

```
<?php
    $a = <<< END
        "It's not going to happen!"
    END;
    echo $a;
?>
```

**Output:**

"It's not going to happen!"

The new line before the trailing terminator is removed.

**Variable Interpolation:**

**[Oct. 16, 17, April 19]**

- Interpolation is the process of replacing variable names in the string with the values of those variables.
- Single quoted strings do not interpolate variables. When variable is included inside double quoted string or heredoc string, the variable is interpreted (or parsed).
- There are following two ways to interpolate variables into strings:

**1. Simple Syntax:**

- In this way, the variable name is simply put inside the double-quoted string or heredoc.
- 

**For example:**

```
<?php
    $a = 'Hello';
    $b = 'Nirali Prakashan';
    echo "$a $b";
?>
```

**Output:**

Hello Nirali Prakashan

---

## 2. Complex Syntax:

- In this way, variable is wrapped with { and }.

**For example:**

```
<?php
    $a=1;
    echo "You are the {$a}st employee";
?>
```

**Output:**

You are the 1st employee.

- Without { } the PHP parser consider \$ast as a variable and displays Noticed undefined variable: \$ast on line 3.

## Character Escaping:

- Escape sequences are used for escaping a character during the string parsing. In PHP, an escape sequence starts with a backslash \ followed by the character which may be an alphanumeric or a special character.
- A single-quoted string only uses the escape sequences for a single quote ('...'). All the escape sequences like \r or \n, will be output as specified instead of having any special meaning.
- Single quotes are used in the special case is that if we to display a literal single-quote, escape it with a backslash () and if we want to display a backslash, we can escape it with another backslash (\\\).
- Escape sequences also apply to double-quoted strings. Escape sequences for double-quoted strings are given below:

| Escape Sequence | Character Represented                                    |
|-----------------|----------------------------------------------------------|
| \"              | double quotes                                            |
| \n              | newline                                                  |
| \r              | carriage return                                          |
| \t              | tab                                                      |
| \\\             | Backslash                                                |
| \\$             | Dollar sign                                              |
| \{              | left brace                                               |
| \}              | right brace                                              |
| \[              | left square bracket                                      |
| \]              | right square bracket                                     |
| \o              | through \777 ASCII character represented by octal value. |
| \xo             | through \XFF ASCII character represented by hex value.   |

|     |                                                          |
|-----|----------------------------------------------------------|
| \\" | double quotes                                            |
| \n  | newline                                                  |
| \r  | carriage return                                          |
| \t  | tab                                                      |
| \\\ | Backslash                                                |
| \\$ | Dollar sign                                              |
| \{  | left brace                                               |
| \}  | right brace                                              |
| \[  | left square bracket                                      |
| \]  | right square bracket                                     |
| \o  | through \777 ASCII character represented by octal value. |
| \xo | through \XFF ASCII character represented by hex value.   |

## 2.7 PRINTING FUNCTIONS

- In PHP there are various methods to print data. There are following ways to send output to the browser:
  - `echo` construct: Print many values at once.
  - `print()` : Prints only one value.
  - `printf()` : Builds a formatted string.
  - `print_r()` : Prints the contents of arrays, objects, and other things, in a more-or-less human-readable form. **[Oct. 17]**
  - `var_dump()` : Prints the content of arrays, objects and other things in more human readable form alongwith size, type, length of elements useful for debugging. **[Oct. 17]**
- Let us see above functions in detail.

### 1. echo Construct:

**[April 18]**

- The echo is not a PHP function it a construct. The echo construct puts string into HTML of a PHP-generated page then it send to the browser.
- The echo construct looks like a function but it is a language constraints. This means that we can omit the parentheses, so the following are equivalent:

```
echo "Printy";
echo("Printy"); // also valid
```

- We can specify multiple items to print by separating them with commas:

```
echo "First", "second", "third";
```

#### Output:

```
First Second Third
```

- It is a parse error to use parentheses when trying to echo multiple values:

```
// this is a parse error
echo("Hello", "world"); //invalid
```

- Because echo is not a true function, we can't use it as part of a larger expression:

```
// parse error
if (echo("test")) { //invalid
    echo("it worked!");
}
```

- Such errors are easily remedied, though, by using the `print()` or `printf()` functions.

### 2. print() Function:

**[April 18]**

- The `print()` function sends one value (its argument) to the browser. It returns true if the string was successfully displayed and false otherwise.

**For example:**

```
if(! print("Hello, world"))
{
    die("you're not listening to me!");
}
```

**Output:**

Hello, world

### 3. printf() Function:

- The printf() function outputs a formatted string. The first argument to printf() is the format string. The remaining arguments are the values to be substituted in.

#### Format Modifiers:

- Each substitution marker in the template consists of a % sign, followed by modifiers and ends with a type specifier.
- Following sequence of modifiers is necessary.
  - A padding specifier denoting the character to use to pad the result to the appropriate string size. Padding with spaces is the default.
  - A sign minus (-) forces the string to be left justified. Default is right justified.
  - The minimum length of the element. If the result is less than this number of characters, then padded with some value.
  - For floating point numbers, it gives or dictates how many decimal digits will be displayed.

#### Type Specifiers:

- The type specifier tells printf() what type of data is being substituted. There are different types as shown below:

| Specifier | Meaning                                                                      |
|-----------|------------------------------------------------------------------------------|
| B         | The argument is integer and is displayed as a binary number.                 |
| C         | The argument is integer and is displayed as a character with that value.     |
| d or I    | The argument is integer and is displayed as a decimal number.                |
| e, E or f | Argument is double, displayed as a floating point number.                    |
| g or G    | Argument is double with precision, displayed as floating point number.       |
| O         | Argument is integer and displayed as a floating point number.                |
| S         | Argument is string and displayed as such.                                    |
| U         | Argument is an unsigned integer and is displayed as decimal.                 |
| x         | Argument is integer, displayed in hexadecimal with lowercase letters         |
| X         | Argument is an integer and displayed as a hexadecimal with uppercase letter. |

Some examples:

```
printf('%.2f', 27.452);
```

**Output:** 27.45

```
printf('The hex value of %d is %x', 214, 214);
```

**Output:** The hex value of 214 is d6

```
printf('Bond. James Bond. %03d.', 7);
```

**Output:** Bond. James Bond. 007.

```
$month = 2; $day = 15; $year = 2002;
```

```
printf('%02d/%02d/%04d', $month, $day, $year);
```

**Output:** 02/15/2002

```
printf('%.2f%% Complete', 2.1);
```

**Output:** 2.10% Complete

```
printf('You\'ve spent $%.2f so far', 4.1);
```

**Output:** You've spent \$ 4.10 so far

#### 4. print\_r() Function:

- The print\_r() is useful for debugging purpose. It prints the contents of array, objects and other things in more or less human-readable form.

**For example:**

```
$a = array('name' => 'Amar', 'age' => 35);
print_r($a);
```

**Output:**

```
Array
(
    [name] => Amar
    [age] => 35
)
```

#### 5. var\_dump() Function:

- The var\_dump() function is used to display structured information (type and value) about one or more variables.

**For example:**

```
<?php
    var_dump(14) . "<br>"; // <br> is used for line change
    var_dump("Hello") . "<br>";
    var_dump(25, "ty1");
?>
```

**Output:**

```
Int(14)
String(5) "Hello"
Int(25)string(3) "ty1"
```

---

- Boolean values and NULL are not meaningfully displayed by print\_r().

**For example:**

```
print_r(true); // Outputs 1
print_r(false); // Nothing is displayed
print_r(null); // Nothing is displayed
```

- For this reason, var\_dump() is preferable to print\_r() for debugging. The var\_dump() function displays any PHP value in more human-readable format.

**For example:**

```
var_dump(true);
Output: bool(true)

var_dump(false);
Output: bool(false);

var_dump(null);
Output: bool(null);

var_dump(array('name' => Amar, 'age' => 35));
```

**Output:**

```
array(2)
{
    ["name"]=>string(4) "Amar"
    ["age"]=>int(35)
}
```

- In recursive structures, print\_r() loops infinitely while var\_dump() cuts off after visiting the same element three times.

**Accessing Individual Characters:**

- We can access the individual character of a string using [].

**For example:**

```
<?php
$str = "Hello";
echo $str[0]; // H
?>
```

---

- The **strlen()** function returns the number of characters in a string.

```
<?php
$str = "Hello";
for ($i=0; $i < strlen($str); $i++)
{
    printf("The %dth character is %s\n", $i, $str[$i]);
}
?>
```

#### Output:

The 0<sup>th</sup> character is H The 1<sup>st</sup> character is e. The 2<sup>th</sup> character is l.  
The 3<sup>rd</sup> character is l. The 4<sup>th</sup> character is o

#### Cleaning String:

- Two common problems with raw data often we get from user or files are the presence of extra white spaces and incorrect capitalization (uppercase versus lowercase).
- To recover from these problems, the data needs to be cleaned up before use.

#### Removing Whitespace:

- We can remove leading or trailing whitespace with the trim(), ltrim() and rtrim() functions explained below:

- trim()**: Returns the string with whitespaces removed from the beginning and from the end. [April 19]

**Syntax:** trim(string [, charlist ]);

- ltrim()**: It removes white spaces only from the start. Here 'l' indicates 'left'.

**Syntax:** ltrim(string [, charlist ]);

- rtrim()**: It removes whitespaces only from the end. Here r indicates 'right'.

**Syntax:** rtrim(string [, charlist ]);

- Apart from the whitespaces following default characters are removed.

| Character | Meaning         |
|-----------|-----------------|
| “ ”       | space           |
| “\t”      | tab             |
| “\n”      | newline         |
| “\r”      | carriage return |
| “\0”      | null byte       |
| “\XOB”    | vertical tab    |

- The charlist is optional string parameter which specifies the characters apart from default characters to be trimmed (removed) from the string.

#### For example:

```
$title = " Programming PHP \n";
$str_1 = ltrim($title); // $str_1 is "Programming PHP \n"
$str_2 = rtrim($title); // $str_2 is " Programming PHP"
$str_3 = trim($title); // $str_3 is "Programming PHP"
```

**Changing Case:**

- PHP has several functions for changing the case of strings. Each function takes a string as an argument and returns a copy of that string, appropriately changed.
- PHP has following functions for changing the case of strings:
  1. **strtolower()** and **strtoupper()** operate on entire string.
  2. **ucfirst()** operates only on first character of the string.
  3. **ucwords()** operates on first character of each word of the string.

**[April 19]****For example:**

```
$S1 = strtolower ("Hello World"); // $S1 = "hello world"
$S2 = strtoupper ("Hello World"); // $S2 = "Hello WORLD"
$S3 = ucfirst ("hello world"); // $S3 = "Hello world"
$S4 = ucwords ("hello world"); // $S4 = "Hello World"
```

**2.8 ENCODING AND ESCAPING**

- PHP programs often interact with HTML pages, web addresses (URLs) and databases. There are functions to help us to work with those types of data.
- HTML web page addresses, and database commands are all strings, but they each require different characters to be escaped in different ways.
- For instance, a space in a web address must be written as %20, while a literal less-than sign (<) in an HTML document must be written as &lt;.
- PHP has a number of built-in functions to convert to and from these encodings.

**1. HTML:**

- Special characters in HTML are represented by entities such as & and &lt;. There are two functions to convert special characters in a string into their entities.

**(i) htmlentities() Function:**

- The htmlentities() function converts all applicable characters to HTML entities.

**Syntax:** string htmlentities(string str[, int quote\_style][, String charset])**For example:**

```
<?php
    $str = htmlentities("Hello World <p>");
    echo $str;
?>
```

**Output in Terminal:**

```
Hello World &lt;p&gt;
```

**Output in Browser:**

```
Hello World <p>
```

- Second optional parameter contains the following value:
 

|              |   |                                      |
|--------------|---|--------------------------------------|
| ENT_COMPAT   | : | Default. converts only double quotes |
| ENT_QUOTES   | : | Converts double and single quotes    |
| ENT_NOQUOTES | : | Does not convert any quotes          |

**For example:**

```
$input = <<< end
    "Programming PHP" by Nirali
end;
echo htmlentities($input);
// "Programming PHP" by Nirali
echo htmlentities($input, ENT_QUOTES);
// "Programming PHP" by O'Neill
echo htmlentities($input, ENT_NOQUOTES);
// "Programming PHP" by Nirali
```

- Charset is optional. It is a string that specifies which character-set to use. The default is UTF-8.

**(ii) htmlspecialchars() Function:**

- The htmlspecialchars() function converts special characters to HTML entities.
- Certain characters have special significance in HTML, and should be represented by HTML entities if they are to preserve their meanings. This function returns a string with these conversions made.
- If we require all input substrings that have associated named entities to be translated, use htmlentities() instead.
- The translations performed are:
  - & (ampersand) becomes &amp;
  - " (double quote) becomes &quot;
  - ' (single quote) becomes &#039;
  - < (less than) becomes &lt;
  - > (greater than) becomes &gt;
- If we have an application that displays data that a user has entered in a form, we need to run that data through htmlspecialchars( ) before displaying or saving it.
- For example, if user enters a string like "angle < 30", the browser will think the special characters are HTML, and we will have a garbled page.

**Syntax:**

```
string htmlspecialchars(string str[, int quote_style][, String charset])
```

- The syntax and meaning of quote\_style and charset are same as htmlentities().

**For example:**

```
<?php
$new = htmlspecialchars("<a href='test'>Test</a>", ENT_QUOTES);
echo $new; // <a href="test">Test</a>;
?>
```

**(iii) Removing HTML Tags:**

- The strip\_tags() function removes HTML tags from a string.
- Syntax:** strip\_tags(string, allow);
- First parameter is input string, second parameter is optional which specifies allowable tags will not be removed.

**Example:**

```
$input = "Programming <b><i>PHP</i></b>";
echo $input;
// Programming PHP displays the string PHP in bold and italic
echo strip_tags($input);
// Programming PHP both the tags i.e. <b> and <i> removed
echo strip_tags($input, "<b>");
// Programming PHP
allow <b> tag to be used, all other tag will be removed
```

**2. SQL:**

- In database queries we use single quotes or double quotes with some string literals. For example, name or some varchar type of data must be quoted with single quote.
- When we use such type of SQL queries as a string, that will be evaluated by PHP, so it is required to escape single quote, double quote, NULL bytes and backslashes by using backslash present in the query.
- In PHP addslashes() function adds the slashes and stripslashes() function removes them.

**For example:**

```
<?php
$input = "SELECT * FROM customer WHERE name = 'Amar';";
$output = addslashes($input);
echo $output. "<br>";
echo stripslashes($output);
?>
```

**Output:**

```
SELECT * FROM customer WHERE name = \\'Amar\\';
SELECT * FROM customer WHERE name = 'Amar';
```

**2.9 COMPARING STRINGS**

- PHP has different operators and functions for comparing strings to each other.

**2.9.1 Exact Comparisons**

- When comparing values in PHP for equality we can use either the == operator or the === operator.
- The == (Equal) operator just checks to see if the left and right values are equal. But, the === (identical) operator (note the extra “=” ) actually checks to see if the left and right values are equal, and also checks to see if they are of the same variable type (like whether they are both booleans, ints, etc.).

- The == operator casts non-string operands to strings,

```
$a=5;
$b="5";
if($a==$b)      // Output is true
```

- The === operator does not cast, and returns false in above case.

**For example:**

```
$a=5;
$b="5";
$c=5;
if(($a==$b)
{  // returns false
echo .....
}
if($a==$c) {  // returns true
echo...
}
```

- The comparison operators (<, <=, >, >=) also work on strings. Here these operator considers alphabetical sequence for result.

**For example:**

```
<?php
$a="a";
$b="b";
if($a<$b)
{
    print" $a comes before $b";
}
else
    print" $b comes before $a";
?>
```

**Output:**

---

a comes before b

- The functions strcmp() and strcasecmp() are also used for string comparison.

**For example:**

```
$result=strcmp ($S1, $S2);
```

Here, function returns a number less than 0 if \$S1 sorts before \$S2, greater than 0 if \$S2 sorts before \$S1 or 0 if they are same.

- Whereas strcasecmp() converts strings to lowercase before comparing them. It means it is case insensitive.

## 2.9.2 Approximate Equality

- An approximation is anything that is similar but not exactly equal to something else. PHP provides several functions which tests that whether two strings are approximately equal. The functions are soundex(), metaphone(), similar\_text(), levenshtein().
- The soundex() and metaphone() functions compares the pronunciations of both strings. If pronunciation is same then strings are same otherwise not. **[Oct. 17]**
- The metaphone() function is generally more accurate. The soundex() function calculates the soundex key of a string.

**Syntax:** soundex(string)

**For example:**

```
$a="Fred";
$b="Phred";
if(soundex($a)==soundex($b))
{
    print"soundex: sounds same";
}
else
{
    print "soundex: doesn't sound same";
}
```

**Output:**

soundex: doesn't sound same.

- The metaphone() function calculates the metaphone key of a string.

**Syntax:** metaphone(string,length)

**For example:**

```
$a="Fred";
$b="Phred";
if(metaphone($a)==metaphone($b))
{
    print"metaphone: sounds same";
}
else
{
    print"metaphone: doesn't sounds same";
}
```

**Output:**

metaphone: sounds same

- The similar\_text() function calculates the similarity between two strings in percent. It returns the number of characters that its two string arguments have in common.

[April 18]

**Syntax:** int similar\_text (string \$first, string \$second [, float &\$percent])

- First and second parameter contains first and second string. Third optional parameter specifies a variable name for storing the similarity in percent.
- The similar\_text() function returns the number of characters that its two string arguments have in common. Hence, third parameter stores the value in percentage.

**For example:**

```
<?php
    $s1="There";
    $s2="Their";
    $c=similar_text($s1, $s2, $p);
    printf("%d characters in common. Strings are %f percent same", $c, $p);
?>
```

**Output:**

4 characters in common. Strings are 80.000000 percent same

- Then levenshtein() function gives how many characters you must add, substitute, or remove to make them the same.
- The levenshtein() function returns the Levenshtein distance between two strings.
- The Levenshtein distance is the number of characters we have to replace, insert or delete to transform string1 into string2.

**Syntax:** int levenshtein(string \$str1, string \$str2)

**For example:**

```
$similarity = levenshtein("cat", "cot");
// $similarity is 1
```

## 2.10 MANIPULATING AND SEARCHING STRINGS

[April 18]

- In PHP, there are many functions for modification and searching of string. Some common functions are explained in this section.

### 1. substr() Function:

[April 17, Oct. 17]

- The substr() function returns a part (substring) of a string.
- Syntax:** string substr (string \$string, int \$start [, int \$length])
- This function returns the portion of string specified by the start and length parameters.
- Here 2<sup>nd</sup> argument 'start' means position in string at which to begin copying. The length argument is the number of character to copy (default is upto end of string).

**For example:**

```
<?php
    echo substr('abcdef', 1); // bcd
    echo substr('abcdef', 1, 3); // bcd
    echo substr('abcdef', 0, 4); // abcd
    echo substr('abcdef', 0, 8); // abcdef
    echo substr('abcdef', -1, 1); // f
?>
```

## 2. substr\_replace() Function:

- This function replaces text within a portion of a string.

**Syntax:** `string substr_replace (string $original, string $replacement,  
int $start [, int $length])`

- The function replaces the part of ‘original’ indicated by the ‘start’ (0 means the start of the string) and ‘length’ values with the string ‘replacement’.
- If no fourth argument is given, substr\_replace( ) removes the text from ‘start’ to the end of the string.

**For example:**

```
<?php
    $greeting = "good morning Nirali";
    $farewell = substr_replace($greeting, "bye", 5, 7);
    echo $farewell;
?>
```

**Output:**

good bye Nirali

## 3. substr\_count() Function:

- This function count the number of substring occurrences.

**Syntax:**

```
int substr_count(string $big_str, string $small_str  
[, int $offset = 0 [int $length]])
```

- Returns the number of times the small\_str substring occurs in the big\_str string. Please note that small\_str is case sensitive. ‘offset’ is from where to start counting.

**For example:**

```
<?php
    $n = 'This is a test';
    echo substr_count($n, 'is'); // 2
?>
```

#### 4. substr\_compare() Function:

- Comparison of two strings from an offset, up to length characters.

##### Syntax:

```
int substr_compare(string $main_str, string $str, int $offset
                  [, int $length[, bool $case_insensitivity = false ]])
```

- The substr\_compare() compares 'main\_str' from position 'offset' with 'str' upto 'length' characters.
- Returns < 0 if 'main\_str' from position 'offset' is less than 'str', > 0 if it is greater than 'str', and 0 if they are equal.

##### For example:

```
<?php
    echo substr_compare("abcde", "bc", 1, 2); // 0
    echo substr_compare("abcde", "de", -2, 2); // 0
    echo substr_compare("abcde", "bcg", 1, 2); // 0
    echo substr_compare("abcde", "BC", 1, 2, true); // 0
    echo substr_compare("abcde", "bc", 1, 3); // 1
    echo substr_compare("abcde", "cd", 1, 2); // -1
    echo substr_compare("abcde", "abc", 5, 1); // warning
?
>
```

#### 5. strrev() Function:

[April 18]

- This function takes a string and returns a reversed copy of it.

---

**Syntax:** string strrev(string \$string)

---

##### For example:

```
<?php
    echo strrev("Hello World!");
?
>
```

##### Output:

---

!dlrow olleH

---

#### 6. str\_repeat() Function:

- This function takes a string and count and after it, repeats the string that many times.

**Syntax:** string str\_repeat(string \$input , int \$multiplier)

---

Returns 'input' repeated 'multiplier' times.

---

**For example:**

```
<?php
    $string = "Hello ";
    $x = 5;
    $result = str_repeat($string, $x);
    echo $result;
?>
```

**Output:**

---

```
Hello Hello Hello Hello Hello
```

---

## 7. str\_pad() Function:

[April 19]

- The str\_pad() function pads a string to a new length.

**Syntax:**

```
string str_pad (string $input , int $pad_length
                [, string $pad_string = " " [, int $pad_type = STR_PAD_RIGHT]])
```

- This function returns the ‘input’ string padded on the left, the right, or both sides to the specified padding length.
- If the optional argument ‘pad\_string’ is not supplied, the ‘input’ is padded with spaces, otherwise it is padded with characters from ‘pad\_string’ up to the limit.
- Optional argument ‘pad\_type’ can be STR\_PAD\_RIGHT, STR\_PAD\_LEFT, or STR\_PAD\_BOTH. If ‘pad\_type’ is not specified it is assumed to be STR\_PAD\_RIGHT.

**For example:**

```
<?php
    $input = "PHP";
    echo str_pad($input, 10); // "PHP "
    echo str_pad($input, 10, "=", STR_PAD_LEFT); // "=====PHP"
    echo str_pad($input, 10, "=", STR_PAD_BOTH); // "====PHP===="
    echo str_pad($input, 10, "="); // "PHP====="
?>
```

## 2.10.1 Decomposing a String

[April 16]

- PHP provides several functions to break a string into smaller components. These functions are explode(), implode(), strtok() and sscanf().

### 1. explode() Function:

[April 19]

- It breaks a string into smaller parts and stores it in an array.

**Syntax:** array explode (string \$delimiter, string \$str [, int \$limit])

- Returns an array of strings, each of which is a substring of ‘str’ formed by splitting it on boundaries formed by the string ‘delimiter’.
- If ‘limit’ is set and positive, the returned array will contain a maximum of ‘limit’ elements with the last element containing the rest of ‘str’.
- If the ‘limit’ parameter is negative, all components except the last –‘limit’ are returned.
- If the ‘limit’ parameter is zero, then this is treated as 1.

**For example:**

```
<?php
    $str = 'one|two|three|four';
    $arr = explode('|', $str);
    print_r($arr);
    echo "<br>";
    $arr = explode('|', $str, 2);
    print_r($arr);
    echo "<br>";
    $arr = explode('|', $str, -1);
    print_r($arr);
?>
```

**Output:**

```
Array ( [0] => one [1] => two [2] => three [3] => four )
Array ( [0] => one [1] => two|three|four )
Array ( [0] => one [1] => two [2] => three )
```

- Since, in the second case the limit is two, hence the string will be divided into two parts. In the third case since the limit is negative so except the component ‘four’, remaining components will be returned.

## 2. **implode()** Function:

[April 19]

- It creates a string from an array of smaller strings.
- **Syntax:** string implode (string \$glue, array \$pieces)
- This function Join array elements with a ‘glue’ string.
- Returns a string containing a string representation of all the array elements in the same order, with the glue string between each element.

**For example:**

```
<?php
    $array = array('lastname', 'email', 'phone');
    $comma_separated = implode(", ", $array);
    echo $comma_separated;
?>
```

**Output:**

```
lastname,email,phone
```

- The join() is an alias of implode() function.

### 3. strtok() Function:

- The strtok() function splits a string into smaller strings (tokens). It gives us new token each time.
- Syntax:** string strtok (string \$str, string \$separator)
- The strtok() function splits a string 'str' into smaller strings (tokens), with each token being delimited by any character from 'separator'.
  - That is, if we have a string like "This is an example string" we could tokenize this string into its individual words by using the space character as the separator.
  - Note that only the first call to strtok() uses the string argument. Every subsequent call to strtok() only needs the separator to use, as it keeps track of where it is in the current string.
- 

**For example:**

```
<?php
    $string = "lastname,email,phone";
    $token = strtok($string, ",");
        while ($token !== false)
    {
        echo "$token <br>";
        $token = strtok(",");
    }
?>
```

**Output:**

```
lastname
email
phone
```

---

- In the above program, the character ‘,’ we use as the separator, because we want to break the string by ‘.’. Inside while loop we display each token of the string. When there are no more tokens to be return this function returns false.

### 4. sscanf() Function:

[April 16]

- This function in PHP decomposes a string.
- The sscanf() function parses a string into variables based on the format string.

**Syntax:** sscanf(string,format,arg1,arg2,arg++)

---

**For example:**

```
<?php
    $str = "If you divide 4 by 2 you'll get 2";
    $format = sscanf($str,"%s %s %s %d %s %d %s %s %c");
        print_r($format);
?>
```

**Output:**

```
Array ( [0] => If [1] => you [2] => divide [3] => 4 [4]
      => by [5] => 2 [6] => you'll [7] => get [8] => 2 )
```

---

## 2.10.2 Searching Functions

- PHP has many functions to work with strings. The most commonly used functions for searching are explained in this section.

### 1. strpos() Function:

- This function finds the position of the first occurrence of a substring in a string.
- Syntax:** mixed strpos(string \$str, mixed \$find[, int \$offset = 0])
- Find the numeric position of the first occurrence of ‘find’ in the ‘str’ string. Mixed means any PHP data type.
- If ‘find’ is not a string, it is converted to an integer and applied as the ordinal value of a character.
- ‘offset’ specifies that, search will start this number of characters counted from the beginning of the string.
- The function returns the position of where the ‘find’ exists relative to the beginning of the ‘str’ string (independent of offset). Also note that string positions start at 0 and not 1.
- Returns False if the ‘find’ was not found.

**For example:**

```
<?php
    echo strpos("I am a PHP programmer", "am");
    echo "<br>";
    echo strpos("I am a PHP programmer", "am", 5);
?>
```

**Output:**

```
2
16
```

- In the above program the position (index) of character ‘T’ is 0, hence it displays position of ‘am’ is 2. In the second case, searching starts from 5<sup>th</sup> character, hence ‘am’ is at position 16.

### 2. strrpos() Function:

[Oct. 11]

- The syntax of this function is same as strpos() function. This function finds the last occurrence of substring in the main string.

**For example:**

```
<?php
    echo strrpos("I am a PHP programmer", "am");
    echo "<br>";
    echo strrpos("I am a PHP programmer", "am", 5);
?>
```

**Output:**

```
16
16
```

**3. strstr() Function:**

- This function finds the first occurrence of a small string and returns from that small string onwards.

**Syntax:** string strstr (string \$str, mixed \$find)

- Returns part of 'str' string the matching point and including the first occurrence of 'find' word to the end of 'str'.

**For example:**

```
<?php
    $str="w3resource.com";
    $newstring = strstr($str,".");
    echo $newstring;
?>
```

**Output:**

.com

---

- Few more searching functions are:
  1. **stristr()** : Case-insensitive version of strstr() function.
  2. **strchr()** : Alias of strstr().
  3.  **strrchr()** : Find last occurrence of a character in a string.

**Decomposing URLs:**

- The **parse\_url()** function returns an array of components of a URL:

**Syntax:** \$array = parse\_url(url);

---

**For example:**

```
<?php
    $bits = parse_url('http://me:secret@example.com:8080/
                        php/test?user=Amar#res');
    print_r($bits);
?>
```

**Output:**

```
Array ( [scheme] => http
        [host] => example.com
        [port] => 8080
        [user] => me
        [pass] => secret
        [path] => /php/test
        [query] => user=Amar
        [fragment] => res
    )
```

---

## 2.11 REGULAR EXPRESSIONS

[Oct. 16]

- Regular expression is string that represents a pattern. PHP provides two different types of regular expressions:
  1. POSIX Regular Expression and
  2. Perl compatible Regular Expression.
- POSIX regular expressions are less powerful, and sometimes slower, than the Perl-compatible functions, but can be easier to read.
- There are following three uses of the regular expressions:
  1. **matching:** To match the given pattern.
  2. **substituting:** Substitution of new text for the matching text.
  3. **splitting:** Splitting a string into an array of smaller chunks.
- Regular expression functions accept two parameters. First is pattern and second is string to be matched with pattern.

**For example:**

```
ereg('students', 'Hello students');
      // returns true
```

- Some special characters are also used to perform the matching like ^ (start of string), \$ (end of string), . (matches any single character).
- Regular expressions are case-sensitive by default. If we want case-insensitive then use eregi().
  1. **For matching :** The ereg() and eregi() functions are useful.
  2. **For replacing :** The ereg\_replace() is used. This function takes the pattern and replacement string and a string in which to search and returns replace string.
  3. **For splitting :** The split() is use to divide a string into smaller chunks, which are returned as an array.
- Let us see some functions in detail.

### 1. ereg() Function:

[Oct. 17]

- The ereg() function is used for regular expression match.

**Syntax:** int ereg (string \$pattern , string \$input\_str [, array &\$regs])

- The first parameter ‘pattern’ is a regular expression. The function searches an ‘input\_str’ for matches to the regular expression given in ‘pattern’ in a case-sensitive way.
- If matches are found for parenthesized substrings of ‘pattern’ and the function is called with the third argument ‘regs’, the matches will be stored in the elements of the array ‘regs’.

- \$regs[1] will contain the substring which starts at the first left parenthesis; \$regs[2] will contain the substring starting at the second, and so on. The \$regs[0] will contain a copy of the complete string matched.
- Returns the length of the matched string if a match for ‘pattern’ was found in input\_str or false if no matches were found or an error occurred.
- If the optional parameter ‘regs’ was not passed or the length of the matched string is 0, this function returns 1.
- The following example takes a date in ISO format (YYYY-MM-DD) and prints it in DD.MM.YYYY format:

**For example:**

```
<?php
    $date = "2015-04-12";
    if (ereg ("([0-9]{4})-([0-9]{1,2})-([0-9]{1,2})", $date, $regs))
    {
        echo "$regs[3].$regs[2].$regs[1]";
    }
    else
    {
        echo "Invalid date format: $date";
    }
?>
```

**Output:**

12.04.2015

- To specify a set of acceptable characters in your pattern, we can either build a character class yourself or use a predefined one.
- In PHP we can create our own character class by enclosing the acceptable characters in square brackets.

**For example:**

```
ereg('C[aeiou]t', 'cut') // returns true
ereg('c[aeiou]t', 'crt') // returns false
ereg('[0-9]%', 'we are 25% complete'); // returns true
```

- In character classes ‘\’ is used for negation.

**For example:**

```
ereg('c[^aeiou]t', 'cut') // returns false
```

- In regular expressions a caret (^) at the beginning of a regular expression indicates that it must match the beginning of the string.

**For example:**

```
ereg('^PHP', 'I am a PHP programmer')      // returns false
ereg('^PHP', 'PHP programmer')              // returns true
```

- Similarly, a dollar sign (\$) at the end of a regular expression means that it must match the end of the string

**For example:**

```
ereg('PHP$', 'Programming in PHP')          // returns true
```

- We can define a range of characters with a hyphen (-)

**For example:**

```
ereg('c[a-z]t', 'cat');      // returns true
```

'|' (pipe) character is used to specify alternatives in a regular expression.

```
ereg('A|B', 'A')           // returns true
```

- To specify a repeating pattern, we can use quantifiers. Following table lists qualifiers:

| Quantifier | Meaning                         |
|------------|---------------------------------|
| ?          | 0 or 1                          |
| *          | 0 or more                       |
| +          | 1 or more                       |
| {n}        | exactly n times                 |
| {n, m}     | atleast n, no more than m times |
| {n, }      | atleast n times.                |

**For example:**

```
ereg('ca+t', 'caaat'); // true
```

```
ereg('ca*t', 'ct');    // true
```

```
ereg('ca+t', 'ct');    // false
```

**2. ereg\_replace() Function:**

- The ereg\_replace() function takes a 'pattern', a 'replacement' string, and a 'string' in which to search.
- If 'pattern' found in the string 'string' then it is replaced by the string 'replacement'. This function returns the modified string.

**Syntax:**


---

```
string ereg_replace(string $pattern, string $replacement, string $string)
```

---

**For example:**

```
<?php
    $num = "7";
    $str = ereg_replace("seven", $num, "There are seven days in a week");
    echo $str;
?>
```

**Output:**


---

```
There are 7 days in a week
```

---

- The eregi\_replace() function is a case-insensitive form of ereg\_replace().

**3. split() Function:****[April 19]**

- The split() function uses a regular expression to divide a string into smaller chunks, which are returned as an array.

**Syntax:** \$chunks = split(pattern, string [, limit ]);

- The pattern matches the text that separates the chunks.

**For example:**

```
<?php
    $expression = '3*5+i/6-12';
    $terms = split('/[+*-]', $expression);
    print_r($terms);
?>
```

**Output:**


---

```
Array ([0] => 3 [1] => 5 [2] => i [3] => 6 [4] => 12)
```

---

- If 'limit' is set, the returned array will contain a maximum of 'limit' elements with the last element containing the whole rest of string.

**For example:**

```
<?php
    $expression = '3*5+i/6-12';
    $terms = split('/[+*-]', $expression, 3);
    print_r($terms);
?>
```

**Output:**


---

```
Array ( [0] => 3 [1] => 5 [2] => i/6-12 )
```

---

- The ereg(), ereg\_replace(), and split() function has been deprecated as of PHP 5.3.0. Relying on this feature is highly discouraged.

**Perl-Compatible Regular Expressions:****[April 17]**

- POSIX regular expressions are designed for use with only textual data. To do matches against arbitrary binary data, we will need to use Perl-compatible regular expressions.
- The functions described above i.e. ereg(), ereg\_replace(), and split() use the POSIX regular expression for matching, replacing and splitting.
- The corresponding Perl-Compatible Regular expression functions are preg\_match(), preg\_replace(), and preg\_split().
- Perl-style regular expressions use the Perl syntax for patterns, which means that each pattern must be enclosed in a pair of delimiters.
- Traditionally, the slash (/) character is used; for example, /pattern/. The other delimiters are #...#, (), {}, [], and <>.

**For example:**

```
preg_match('#PHP#', 'I am a PHP programmer');      // returns true
echo preg_replace('/,/ ', '+', 'a,b,c');           // Displays a+b+c
print_r(preg_split('/-/ ', '12-04-2015'));

```

**Output:**

```
Array ( [0] => 12 [1] => 04 [2] => 2015 )
```

## PRACTICE QUESTIONS

### Q.I Multiple Choice Questions:

1. The [:alpha:] can also be specified as,
 

|                 |              |
|-----------------|--------------|
| (a) [A-Za-z0-9] | (b) [A-za-z] |
| (c) [A-z]       | (d) [a-z]    |
2. A function in PHP which starts with double underscore is known as,
 

|                      |                           |
|----------------------|---------------------------|
| (a) Magic Function   | (b) Inbuilt Function      |
| (c) Default function | (d) User Defined Function |
3. A function name always begins with the keyword,
 

|              |                       |
|--------------|-----------------------|
| (a) fun      | (b) def               |
| (c) function | (d) None of mentioned |
4. A variable \$str is set to "HELLO WORLD", which of the following script returns it title case,
 

|                         |                                     |
|-------------------------|-------------------------------------|
| (a) echo ucwords(\$str) | (b) echo ucwords(strtolower(\$str)) |
| (c) echo ucfirst(\$str) | (d) echo ucfirst(strtolower(\$str)) |
5. How many types of functions are available in PHP?
 

|       |       |
|-------|-------|
| (a) 5 | (b) 4 |
| (c) 3 | (d) 2 |
6. POSIX stands for,
 

|                                                    |
|----------------------------------------------------|
| (a) Portable Operating System Interface for Unix   |
| (b) Portable Operating System Interface for Linux  |
| (c) Portative Operating System Interface for Unix  |
| (d) Portative Operating System Interface for Linux |
7. The strstr [egg] function selects a substring by its,
 

|                     |             |
|---------------------|-------------|
| (a) Numerical value | (b) Content |
| (c) Position        | (d) zero    |
8. Which function compares the two strings s1 and s2, ignoring the case of the characters?
 

|                  |                   |
|------------------|-------------------|
| (a) strtolower() | (b) toLowerCase() |
| (c) strcasecmp() | (d) lc()          |
9. Which function can be used to compare two strings using a case-insensitive binary algorithm?
 

|                  |              |
|------------------|--------------|
| (a) strcmp()     | (b) strcmp() |
| (c) strcasecmp() | (d) stri()   |



22. Which following is not advantage of PHP functions:
- (a) Code reusability
  - (b) Easy to understand
  - (c) Better code organization
  - (d) Maximum code
23. Which of the following is not a rule for heredoc?
- (a) heredoc syntax begins with three less than signs (<<<) followed by identifier
  - (b) Space before and after <<< is essential.
  - (c) After the string, the same identifier that is defined after the (<<<) signs in the first line should be placed at the beginning
  - (d) The string begins in the next line and goes as long as it requires.
24. What is the output?
- ```
<?php
$a=1;
echo "You are the {$a}st employee";
?>
```
- (a) You are the 1st employee
  - (b) You are the \$ast employee
  - (c) You are the {\$a}st employee
  - (d) You are the {1}st employee
25. Which is not printing function in PHP?
- (a) print()
  - (b) print\_f()
  - (c) var\_dump()
  - (d) echo()
26. What will be the output?
- ```
<?php
$a=5;
$b="5";
$c='5';
if($a==$b)
    echo" a&b equal";
else if($b==$c) {
    echo "b&c equal";
}
else if ($a==$c)
    echo "a&c equal";
else
    echo "Not Equals";
?>
```
- (a) a&b equal
  - (b) b&c equal
  - (c) a&c equal
  - (d) Not equals
27. Which expressions are nothing more than a sequence or pattern of characters itself?
- (a) Regular
  - (b) Constant
  - (c) Literal
  - (d) None of mentioned

28. Following which functions used for variable parameters:

  - (a) func\_get\_args()
  - (b) func\_num\_args()
  - (c) func\_get\_arg()
  - (d) All of mentioned

29. Following which printing functions used by PHP:

  - (a) echo construct
  - (b) print()
  - (c) printf()
  - (d) All of mentioned

## Answers

|         |         |         |         |         |         |         |         |         |         |
|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| 1. (b)  | 2. (a)  | 3. (c)  | 4. (b)  | 5. (d)  | 6. (a)  | 7. (b)  | 8. (c)  | 9. (c)  | 10. (a) |
| 11. (d) | 12. (c) | 13. (d) | 14. (a) | 15. (a) | 16. (c) | 17. (d) | 18. (b) | 19. (c) | 20. (b) |
| 21. (d) | 22. (d) | 23. (c) | 24. (a) | 25. (b) | 26. (b) | 27. (a) | 28. (d) | 29. (d) |         |

## **Q.II Fill in the Blanks:**

1. A \_\_\_\_\_ is a self-contained block of code that performs a specific task.
  2. By default, function arguments are \_\_\_\_\_.
  3. A function can return a value using the \_\_\_\_\_ statement
  4. \_\_\_\_\_ functions returns true if the value is set to the argument passed to it, otherwise it returns result as false.
  5. The anonymous function is created using the \_\_\_\_\_.
  6. \_\_\_\_\_ syntax begins with three less than signs (<<<) followed by identifier
  7. The code inside the \_\_\_\_\_ braces is a function body
  8. \_\_\_\_\_ function is a function without any user defined name.
  9. \_\_\_\_\_ supports all the syntax rules of heredoc except the starting name.
  10. The \_\_\_\_\_ function converts all applicable characters to HTML entities.
  11. The \_\_\_\_\_ function is used to find the length of a string.
  12. A function is a block of statements that can be used \_\_\_\_\_ repeatedly in a program.
  13. A function will be executed by a \_\_\_\_\_ call to the function.
  14. When a function argument is passed by \_\_\_\_\_ (use & operator), changes to the argument also change the variable that was passed in.
  15. The \_\_\_\_\_ function reverses a string.
  16. We can set a parameter to have a \_\_\_\_\_ value if the function's caller does not pass it.
  17. In \_\_\_\_\_ variable functions we can call function based on value of a variable.
  18. An anonymous function (no specified name) also called as \_\_\_\_\_ function.
  19. If we want a variable in the global scope to be accessible from within a function, we can use the \_\_\_\_\_ keyword.
  20. A \_\_\_\_\_ expression is a sequence of characters that forms a search pattern.
  21. The \_\_\_\_\_ function returns a part (substring) of a string.
  22. A function is a \_\_\_\_\_ block of code that performs a specific task.

### Answers

|              |                    |              |               |                      |
|--------------|--------------------|--------------|---------------|----------------------|
| 1. function  | 2. passed by value | 3. return    | 4. isset()    | 5. create_function() |
| 6. heredoc   | 7. curly { }       | 8. Anonymous | 9. Nowdoc     | 10. htmlentities()   |
| 11. strlen() | 12. repeatedly     | 13. call     | 14. reference | 15. strrev()         |
| 16. default  | 17. variables      | 18. lambda   | 19. global    | 20. regular          |
| 21. substr() | 22. self-contained |              |               |                      |

**Q.III State True or False:**

1. Function names are case-insensitive.
2. A function can return a value using the return statement, may be by returned, including file and objects.
3. PHP does not support for variable-length argument lists in user-defined functions.
4. The func\_get\_arg() returns specific arguments from the parameters.
5. Anonymous function is a function with any user defined name.
6. An anonymous function also called as lambda function.
7. Interpolation is the process of replacing variable names in the string with the values of those variables.
8. The var\_dump() function is used to display information about functions.
9. The ucfirst() operates only on first word of the string.
10. The split() function uses a regular expression to divide a string into smaller chunks.
11. The code inside a function is not executed until the function is called.
12. After the function execution is completed, the program control returns to the point where the function was called.
13. The variable functions mean that we can call function based on value of a variable.
14. The trim() function returns the string with whitespaces removed from the beginning only.
15. To create a function in PHP its name should start with keyword function and all the PHP code should be put inside { and } braces.
16. A function can return a value using the return statement.
17. In PHP, arguments are usually passed by value, which means that a copy of the value is used in the function and the variable that was passed into the function cannot be changed.
18. The PHP strpos() function searches for a specific text within a string.
19. Information can be passed to functions through arguments.
20. We can create an anonymous function using create\_function().
21. To check if parameter is missing or not, we can use PHP built in isset() function.
22. We put multiline strings into the program with a heredoc. The <<< identifier tells the PHP parser that we are writing a heredoc.
23. As with variable variables, we can call a function based on the value of a variable.

24. PHP functions can return only a single value with the return keyword.
25. In PHP, regular expressions are strings composed of delimiters, a pattern and optional modifiers.
26. To specify a default parameter, assign the parameter value in the function declaration.
27. The structure of a POSIX regular expression is not dissimilar to that of a typical arithmetic expression: various elements (operators) are combined to form more complex expressions.
28. As with variable variables, we can call a function based on the value of a variable.
29. A regular expression is a string that represents a pattern.

### Answers

|         |         |         |         |         |         |         |         |         |         |
|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| 1. (T)  | 2. (F)  | 3. (F)  | 4. (T)  | 5. (F)  | 6. (T)  | 7. (T)  | 8. (F)  | 9. (F)  | 10. (T) |
| 11. (T) | 12. (T) | 13. (T) | 14. (F) | 15. (T) | 16. (T) | 17. (T) | 18. (T) | 19. (T) | 20. (T) |
| 21. (T) | 22. (T) | 23. (T) | 24. (T) | 25. (T) | 26. (T) | 27. (T) | 28. (T) | 29. (T) |         |

### Q.IV Answer the following Questions:

#### (A) Short Answer Questions:

1. What is function?
2. What is string?
3. Which keyword is used for creating a function?
4. How to call a function?
5. How to create string?
6. Which function is used for missing parameters?
7. Give function used for variable parameters.
8. How to assign default argument in PHP?
9. Give purpose of return statement.
10. Define variable function?
11. What is anonymous function?
12. List types of string.
13. Define regular expression.
14. List printing functions for string.
15. Which function is used for comparing two strings?

#### (B) Long Answer Questions:

1. Define function? How to declare and call it? Explain with example.
2. How to declare a string? Explain with example.
3. With the help of example explain anonymous function.
4. What is default parameter? Describe in detail with example.
5. Write short note on:
  - (i) Variable parameters (ii) Missing parameters (iii) Variable function.
6. With the help of example explain printing functions.

7. What is meant by encoding and escaping? List functions for encoding and escaping. Also explain with example.
8. Explain comparing two strings with example.
9. How to manipulate and search a string? Explain with example.
10. Define regular expression. How to create it?
11. Write a program using function to calculate factorial of function.
12. With the help of example explain types of strings.

## UNIVERSITY QUESTIONS AND ANSWERS

**April 2016**

1. Give function to check if a variable has value or not. **[1 M]**
- Ans.** Refer to Section 2.4.
2. Write a PHP program to accept a string and print string divided into equal number of characters. (Consider spaces and enter character in calculation).What is function of bootstrap loader? **[5 M]**
- Ans.** Refer to Section 2.6.
3. Write any two functions of decompose string with suitable example. **[2 M]**
- Ans.** Refer to Section 2.10.1, Point (4).

**October 2016**

1. What is an interpolation? **[1 M]**
- Ans.** Refer to Page 2.12.
2. Differentiate between single quoted string and double quoted string. **[1 M]**
- Ans.** Refer to Section 2.6.
3. What is an introspection? Explain any four introspective functions provided by PHP. **[5 M]**
- Ans.** Refer to Page 2.12.
4. What is regular expression? Also write a PHP script using regular expression to check whether a small string appears at the end of a large string. **[4 M]**
- Ans.** Refer to Section 2.11.
3. State True or False. "The fun\_num\_args() returns an array of all parameters provided to the function". **[1 M]**
- Ans.** Refer to Section 2.3, Point (2).

**April 2017**

1. "A function can have variable number of arguments." State true or false. **[1 M]**
- Ans.** Refer to Section 2.5.1.
2. How to find out the position of the first occurrence of a substring in a string? **[1 M]**
- Ans.** Refer to Section 2.10, Point (1).
3. Write a PHP script to accept three strings str1, str2, str3 from user. Search str2 in str1 and replace all occurrences of str2 by str3. Also display total number of occurrences. **[5 M]**
- Ans.** Refer to Section 2.6.

4. Explain the concept of missing parameters to a function with suitable example. **[4 M]**

**Ans.** Refer to Section 2.4.

**October 2017**

1. Explain the following functions with syntax and example:  
(i) func\_num\_arg() (ii) var\_dump() (iii) print\_r()  
(iv) soundex() (v) strrpos(). **[5 M]**

**Ans.** Refer to Sections 2.3, Point (2), 2.7, Points (4) and (5), 2.9.2 and 2.10.2, Point (2).

2. Explain variable function concept in PHP. **[2 M]**

**Ans.** Refer to Section 2.5.1.

3. Write a short note on: Introspection. **[4 M]**

**Ans.** Refer to Page 2.12.

4. How to find out the position of the first occurrence of a substring in a string? **[5 M]**

**Ans.** Refer to Section 2.10, Point (1).

5. Explain the concept of missing parameters to a function with suitable example. **[5 M]**

**Ans.** Refer to Section 2.4.

6. Write an anonymous function to maximum of two numbers. **[2 M]**

**Ans.** Refer to Section 2.5.2.

7. Explain ereg() function. List and explain (any three) with example special character used in regular function. **[5 M]**

**Ans.** Refer to Section 2.11, Point (1).

**April 2018**

1. What is the difference between echo() and print() functions? **[1 M]**

**Ans.** Refer to Section 2.7, Points (1) and (2).

2. What is the difference between single and double quoted string? **[1 M]**

**Ans.** Refer to Section 2.6.

3. Explain anonymous function concept in PHP. **[2 M]**

**Ans.** Refer to Section 2.5.2.

4. Explain the following functions with syntax and example:

- (i) func\_get\_arg() (ii) var\_dump() (iii) strrev()  
(iv) similar\_text() (v) str\_replace().

**[5 M]**

**Ans.** Refer to Sections 2.3, Point (1), 2.7, Point (4), 2.10 (5), 2.9.2 and 2.10.

**October 2018**

1. Differentiate between single-quoted string and double-quoted string. **[1 M]**

**Ans.** Refer to Section 2.6.

2. Write the output of the following PHP script:

[1 M]

```
<?php
    $str = 'abc, pqr, lmn, xyz';
    $a = explode(' ', ' ', $str, 3);
    print_r($a);
?>
```

**Ans.** Array ( [0] =>abc, pqr, lmn, xyz )

3. Consider a string \$str = "PUNE UNIVERSITY"; write a PHP script to display above string in the format given below using built-in function:

- (i) pune university
- (ii) PUNE UNIVERSITY
- (iii) Pune university
- (iv) Pune University
- (v) Pune University

[5 M]

**Ans.** Refer to Section 2.10.

4. What is an anonymous function? How is it different from normal function? [4 M]

**Ans.** Refer to Section 2.5.2.

5. How to pass parameters to a function by reference? Explain with example. Also write its advantage. [4 M]

**Ans.** Refer to Section 2.1.3.

### April 2019

1. Write the purpose of explode function.

[1 M]

**Ans.** Refer to Section 2.10.1, Point (1).

2. Define an Introspection.

[1 M]

**Ans.** Refer to Page 2.12.

3. Write the output of the following script.

[1 M]

```
<?php
    $a = 'PHP';
    $b = '$a interpolation';
    echo $b;
?>
```

**Ans.** \$a interpolation

4. Explain the following functions with example:

- (i) ucwords()
- (ii) trim()
- (iii) str-pad()
- (iv) ucfirst()
- (v) chunk-split()

[5 M]

**Ans.** Refer to Page Point (3), Page Point (1), Section 2.10, Point (7), Page Point (3).

5. Explain different types of arguments passing to functions with example.

[4 M]

**Ans.** Refer to Section 2.1.3.

6. Explain implode() with suitable example.

[2 M]

**Ans.** Refer to Section 2.10.1, Point (2).

■ ■ ■

# Arrays

## Objectives...

- To understand Basic Concepts of Arrays
- To study Types of Arrays in PHP
- To learn Traversing Arrays and Extracting Data from Arrays

### 3.0 INTRODUCTION

[April 16, 18]

- An array is a collection of data values, organized as an ordered collection of key-value pairs.
- PHP arrays stores multiple values of different data type in a single variable at a time.
- Commonly used terms or element in array are explained below:
  1. **Element:** Element of an array is the items it contains. An array can contain one or more elements.
  2. **Value:** Each element contains one value.
  3. **Key or index:** Key or index of an array is a unique number or a string that is associated with each value of an element.
  4. **Length:** The length of an array is the number of elements it contains.

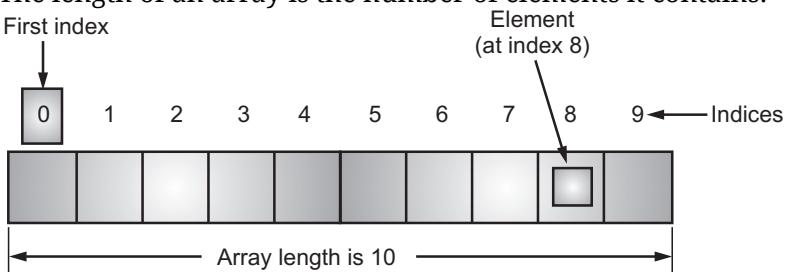


Fig. 3.1: Element of Array

### Creating an Array:

- In PHP, an array can be created using the array() language construct. It takes any number of comma-separated key => value pairs as arguments.

#### Syntax:

```
$array_name = array
{
    key1 => value1,
    key2 => value2,
    key3 => value3,
    ...
};
```

- The key can either be an integer or a string. The value can be of any type.

**For example:**

```
<?php(
    $month = array
        0 => "January";
        1 => "February";
);
?>
```

## 3.1 INDEXED VS ASSOCIATIVE ARRAYS

[April 18]

- There are two types of arrays in PHP namely, Indexed array (with a numeric index) and Associative array (with named keys).

### 3.1.1 Indexed Array

[April 18]

- The keys of an indexed array are integers beginning at 0. Indexed arrays are used when identification of array elements are by their position.

**For example:**

```
$city = array( 0 => "Pune", 1 => "Mumbai", 2 => "Delhi");
echo $city[1]; // Mumbai
```

- Indexed array can also be created without keys. In this case the key will be started from 0.

**For example:**

```
$city = array("Pune", "Mumbai", "Delhi");
echo $city[3]; // Chennai
```

### 3.1.2 Associative Array

[April 16, 18, 19, Oct. 17]

- An associative array has strings as keys. Associative array will have their index as string so that we can establish a strong association between key and values.

**For example:**

```
$marks = array("Maths" => 36, "Physics" => 28, "Chemistry" => 30);
echo $marks['Physics']; // 28
$v = array("a" => "one", "b" => "two", "c" => "three");
echo $v['a']; // one
```

- PHP internally stores all arrays as associative arrays, so the only difference between associative and indexed arrays is what the keys happen to be.
- In both cases, the keys are unique i.e., we can't have two elements with the same key, regardless of whether the key is a string or an integer.

## 3.2 IDENTIFYING ELEMENTS OF AN ARRAY

- We can access specific values from an array using array variable's name followed by element's keys (i.e. index) within square brackets.  
`$A['one']`  
`$A[1]`
- The key can be either a string or an integer. String values that are equivalent to integer numbers (without leading zeros) are treated as integers i.e. `A[1]` and `A['1']` reference the same element. But `A['01']` references a different element.
- Negative numbers are valid keys. No need to quote single word strings i.e. `$A['one']` and `$A[one]` both are same.
- But for better PHP style, always use quotes, because quote less keys are indistinguishable from constants.

**For example:**

```
define ('index', 5);
echo $array [index];
```

- Echo statement will consider `array[5]` instead of `array['index']`. If we interpolate the array variable, quotes ('...' or "...") must not be used.

**For example:**

```
echo "Hello $person['name']"; // Syntax Error
echo "Hello $person["name"]"; // Syntax Error
echo "Hello $person[name]"; // ok
```

### Modifying Values of an Array:

- To change an existing value of an element, we need to specify the new value in the array mentioning the key of that element in [] brackets.

**Syntax:**

```
$array_name[existing_key] = "new_value";
```

**For example:**

```
<?php
$month = array(
    0 => "January",
    1 => "February",
    2 => "March",
)
array[2] = "December";
?>
```

- To remove a key/value pair, call the unset() function on it.

```
<?php
    $arr = array(5 => 1, 12 => 2);
    unset($arr[5]); // This removes the element from the array by key
    unset($arr);    // This deletes the whole array
?>
```

### 3.3 STORING DATA IN ARRAYS

- Storing a value in an array will create the array if it didn't already exist. Use simple assignment to initialize an array in the program.

**For indexed array:**

```
$a[0]=10;
$a[1]=20;
$a[2]=30;
```

**For associative array:**

```
$a['one']=1;
$a['two']=2;
$a['three']=3;
```

- Another method to initialize an array is use the array() construct, which builds an array from its arguments.

**For example:**

```
$a=array(1, 2, 3);
```

- This builds an indexed array and index values (starting from 0) are created automatically.

**For example:**

```
$a=array('one' => 1, 'two' => 2, 'three' => 3);
```

- This structure creates an associative array.

**For example:**

```
$a=array();
```

Here, it constructs empty array.

- It is possible to specify the key only for some elements and leave it out for others:

**For example:**

```
$v = array("a", "b", 6 => "c", "d", "e");
```

Here,  $v[0] = "a"$ ,  $v[1] = "b"$ ,  
 $v[6] = "c"$ ,  $v[7] = "d"$ ,  
 $v[8] = "e"$

**For example:**

```
$a = array('one' => 1, 2, 3);
then   a['one']=1
       a[0]=2
       a[1]=3
```

### 3.3.1 Adding Values to the End of Array

[April 17]

- To insert more values at the end of existing array, use [ ] syntax.

**For example:** Indexed array:

```
$A=array(1, 2, 3);
$A[ ]=4; // $A[3]=4
```

**For example:** Associative array:

```
$A=array('one' => 1, 'two' => 2, 'three' => 3);
$A[ ]=4; // $A[0]=4
```

### 3.3.2 Assigning a Range of Values

- The range() function creates an array of consecutive integer or character between two values we pass to it as a parameter.

**Syntax:** array range(mixed \$start, mixed \$end [, number \$step = 1])

- Returns an array of elements from start to end, inclusive. Step is an optional argument which indicates the difference between each consecutive element of an array.

```
$number = range(2, 5); // $number = array(2, 3, 4, 5);
$letter = range('a', 'd'); // $letter = array( ('a', 'b', 'c', 'd')
$a = range(5, 2); // $a = array( (5, 4, 3, 2)
```

- Only the first character of the string argument is used to build the range.

```
range('aaa', 'zzz') // same as range ('a', 'z')
```

**For example:**

```
<?php
    $number = range(0,50,10);
    $character=range('a','i',2)
    print_r ($number);
    print_r ($characters);
?>
```

**Output:**

```
Array ( [0] => 0 [1] => 10 [2] => 20 [3] => 30 [4] => 40 [5] => 50 )
```

### 3.3.3 Getting the Size of an Array

[April 17, 19]

- The count() functions are used to return the number of elements in the array.

**Syntax:** count(\$array, \$mode);

**For example:**

```
$a = array(1, 2, 3, 4);
$size = count($a); // size is 5
```

- The sizeof() function is an alias of the count() function.

**Syntax:** sizeof(\$array, \$mode);

**For example:**

```
$a = array(100 => 'one', 200 => 'two', 300 => 'three');
$size = sizeof($a); // size is 3
```

### 3.3.4 Padding an Array

- For padding an array array\_pad() function is used. This function pads array to the specified length with a value.

**Syntax:** array array\_pad (array \$array, int \$size, mixed \$value)

Where, array\_pad() returns a copy of the 'array' padded to size specified by 'size' with value 'value'. If 'size' is positive then the array is padded on the right, if it's negative then it is padded on the left.

```
$a = array(1, 2, 3);
$pad = array_pad($a, 5, 0);
// $pad is now array (1, 2, 3, 0, 0)
$pad = array_pad($a, -5, 0);
// $pad is now array (0, 0, 1, 2, 3)
```

- If we pad an associative array, existing keys will be preserved. New elements will have numeric keys starting with 0.

**For example:**

```
<?php
$arr = array("a"=>"Horse", "b"=>"Cat", "c"=>"Dog");
print_r(array_pad($arr, 7, "COW"));
?>
```

**Output:**

```
Array
{
    [a] => Horse
    [b] => Cat
    [c] => Dog
    [0] => COW
    [1] => COW
    [2] => COW
    [3] => COW
}
```

### 3.4 MULTIDIMENSIONAL ARRAY

[April 16]

- PHP also support multi-dimensional array. In a multi-dimensional array each element in the main array can also be an array. And each element in the sub-array can be an array, and so on.
- A two-dimensional array is an array of arrays. A three-dimensional array is an array of arrays of arrays.
- The values in the multi-dimensional array are accessed using multiple index.

```
$row0=array(1, 2, 3);
$row1=array(4, 5, 6);
$row2=array(7, 8, 9);
$multi=array($row0, $row1, $row2);
$val=$multi[2][0]; // 2nd row, 0th column and $val=7
```

**For example:**

```
<?php
$marks = array(
    "Amar" => array (
        "physics" => 35,
        "maths" => 30,
        "chemistry" => 39
    ),
    "Kiran" => array (
        "physics" => 30,
        "maths" => 32,
        "chemistry" => 29
    ),
    "Deepa" => array (
        "physics" => 31,
        "maths" => 22,
        "chemistry" => 39
    )
);
/* Accessing multi-dimensional array values */
echo "Marks for Amar in physics : " ;
echo $marks['Amar']['physics'] . "<br />";
echo "Marks for Kiran in maths : " ;
echo $marks['Kiran']['maths'] . "<br />";
echo "Marks for Deepa in chemistry : " ;
echo $marks['Deepa']['chemistry'] . "<br />";
?>
```

**Output:**

```
Marks for Amar in physics: 35
Marks for Kiran in maths: 32
Marks for Deepa in chemistry: 39
```

**3.5 EXTRACTING MULTIPLE VALUES**

- To copy an array's values into variables use the list() construct or function.

**Syntax:** list(\$var1, \$var2, ...)=\$array;

First value of an array is copied into var1, second value into var2 and so on.

**For example:**

```
$a=array(1, 2, 3);
list($x, $y, $z)= $a; // $x=1, $y=2, $z=3
```

- If array has more values than listed variables, then extra values are ignored.

**For example:**

```
$a=array(1, 2, 3);
list($x, $y)= $a; // $x=1, $y=2
```

- If list has more variables than in array the extra values are set to NULL.

**For example:**

```
$a=array(1, 2, 3);
list($x, $y, $z, $m)= $a; // $x=1, $y=2, $z=3; $m=NULL
```

- Two or more consecutive commas in the list() skip the values in array.

**For example:**

```
$A=(1, 2, 3, 4, 5)
list($x,, $y,, $z)= $A; // $x=1, $y=3, $z=5
<?php
$my_array = array("mango", "apple", "orange");
list($a, $b, $c) = $my_array;
echo "I have several fruits, $a, $b and $c.";
?>
```

**Output:**

```
I have several fruits, mango, apple and orange
```

- The list() function only works on numerical arrays and assumes the numerical indices start at 0.

**3.5.1 Slicing an Array**

[April 18]

- To extract a slice of the array, use the array\_slice() function.

**Syntax:** array array\_slice(array \$array, int \$offset

```
[, int $length = NULL [, bool $preserve_keys = false ]])
```

- First parameter ‘array’ is the input array. If offset is non-negative, the slicing will start at this point. If offset is negative, the slicing will start that far from the end of the ‘array’.

**For example:**

```
$input = array("a", "b", "c", "d", "e");
$output = array_slice($input, 2); // returns "c", "d", and "e"
```

- If the parameter ‘length’ is given and is positive, then the slicing will have up to that many elements in it. If the array is shorter than the ‘length’, then only the available array elements will be present.
- If ‘length’ is given and is negative then the sequence will stop that many elements from the end of the array. If it is omitted, then the sequence will have everything from ‘offset’ up until the end of the ‘array’.

```
$input = array("a", "b", "c", "d", "e");
$output = array_slice($input, 0, 3); // returns "a", "b", and "c"
$output = array_slice($input, -2, 1); // returns "d"
```

- The `array_slice()` will reorder and reset the numeric array indices by default. You can change this behavior by setting ‘`preserve_keys`’ to True.

**For example:**

```
<?php
$input = array("a", "b", "c", "d", "e");
// note the differences in the array keys
print_r(array_slice($input, 2, 2));
print_r(array_slice($input, 2, 2, true));
?>
```

**Output:**

```
Array{
    [0] => c
    [1] => d
}
Array
{
    [2] => c
    [3] => d
}
```

### 3.5.2 Splitting an Array into Chunks

- To divide an array into smaller, evenly sized arrays, use the `array_chunk()` function.

**Syntax:**

```
array array_chunk (array $input, int $size [, bool $preserve_keys]);
```

- This function returns a multidimensional numerically indexed array, starting with zero, with each dimension containing size elements.
- The parameter\_keys is optional. If not set it takes the default value FALSE which will reindeer the chunk numerically.

**For example:**

```
<?php
    $input_array = array('a', 'b', 'c', 'd', 'e');
    print_r(array_chunk($input_array, 2));
?>
```

**Output:**

```
Array
{
    [0] => Array
    {
        [0] => a
        [1] => b
    }
    [1] => Array
    {
        [0] => c
        [1] => d
    }
    [2] => Array
    {
        [0] => e
    }
}
```

- When the parameter preserve\_keys is set to TRUE, keys will be preserved.

**For example:**

```
<?php
    $input_array = array('a', 'b', 'c', 'd', 'e');
    print_r(array_chunk($input_array, 2, true));
?>
```

**Output:**

```
Array
{
    [0] => Array
    {
        [0] => a
        [1] => b
    }
}
```

---

```
[1] => Array
{
    [2] => c
    [3] => d
}
[2] => Array
{
    [4] => e
}
}
```

---

**3.5.3 Keys and Values****[Oct. 18]**

- The array\_keys() function returns an array consisting of only the keys in the array.

**Syntax:**

```
array array_keys (array $input [, mixed $search_value [, bool
                                $strict = false]]);
```

- First parameter is an array containing keys to return. Second and third parameters are optional.
- 

**For example:**

```
<?php
$array = array(0 => 100, "color" => "red");
print_r(array_keys($array));
$array = array("color" => array("blue", "red", "green"),
              "size"  => array("small", "medium", "large"));
print_r(array_keys($array));
?>
```

**Output:**

```
Array
{
    [0] => 0
    [1] => color
}
Array
{
    [0] => color
    [1] => size
}
```

---

- In the above program only keys are returned by the array\_keys function from both the arrays.
  - If search\_value is specified then that value will be searched and keys of that value will be returned.
- 

**For example:**

```
<?php
    $array = array("blue", "red", "green", "blue", "blue");
    print_r(array_keys($array, "blue"));
?>
```

**Output:**

```
Array
{
    [0] => 0
    [1] => 3
    [2] => 4
}
```

---

- The parameter ‘strict’ determines if strict comparison (==) should be used during the search.

### 3.5.4 Checking whether an Element Exists

---

- The array\_key\_exists() function is used to see if an element exists in the array.
  - Syntax:** bool array\_key\_exists (mixed \$key, array \$array)
  - This function returns TRUE if the given key is set in the array otherwise False.
- 

**For example:**

```
<?php
    $a=array('one' => 1, 'two' => 2);
    if(array_key_exists('one', $a))
    {
        echo "Key 'one' exists in the array";
    }
    else
    {
        echo "Key 'one' does not exist in the array";
    }
?>
```

**Output:**

```
Key 'one' exists in the array
```

---

### 3.5.5 Removing and Inserting Elements in an Array

[Oct. 17]

- The array\_splice() function can remove or insert elements in an array.
- The function removes selected elements from an array and replaces it with new elements. The function also returns an array with the removed elements.

**Syntax:**

```
array array_splice (array &$input, int $offset [, int $length
   [, mixed $replacement = array()]])
```

- This function removes the elements designated by offset and length from the input array, and replaces them with the elements of the replacement array, if supplied. It returns an array containing the extracted elements.

**For example:**

```
$subjects = array('physics', 'chem', 'math', 'bio', 'cs', 'drama');
$removed = array_splice($subjects, 2);
// $removed is array('math', 'bio', 'cs', 'drama')
// $subjects is array('physics', 'chem')
```

Here 'length' is not specified and offset is 2 hence the function removes everything from position 2 to the end of the array subjects. The function returns the removed elements. After removing the array subjects will be reduced.

- The example below removes only 3 elements i.e. 'math', 'bio', 'cs':
 

```
$subjects = array('physics', 'chem', 'math', 'bio', 'cs', 'drama');
$removed = array_splice($subjects, 2, 3);
// $removed is array('math', 'bio', 'cs')
// $subjects is array('physics', 'chem', 'drama')
```
- To insert new elements use the fourth argument
 

```
$subjects = array('physics', 'chem', 'math', 'bio', 'cs', 'drama');
$new = array('law', 'business');
$removed = array_splice($subjects, 2, 3, $new);
// $removed is array('math', 'bio', 'cs')
// $subjects is array('physics', 'chem', 'law', 'business', 'drama')
```
- The size of the replacement array doesn't have to be the same as the number of elements you delete. The array grows or shrinks as needed.
- The array\_splice() also works on associative arrays.

```
<?php
    a1=array("0"=>"red","1"=>"green");
    $a2=array("a"=>"purple","b"=>"orange");
    array_splice($a1,1,0,$a2);
    print_r($a1);
?>
```

**Output:**

```
Array ([0] => red [1] => purple [2] => orange [3] => green)
```

- Since 3<sup>rd</sup> parameter is 0 so nothing will be removed, and array \$a2 will get inserted from index 1. Here, new keys will be assigned starting with 0.

**3.6****CONVERTING BETWEEN ARRAYS AND VARIABLES**

- PHP provides two functions extract() and compact(), that convert between arrays and variables.

**[April 17]****1. extract() Function:****[Oct. 17]**

- The extract() function automatically creates local variables from an array. The indexes of the array elements are the variable names:

**Syntax:** int extract(array &\$array)

- This function is used to import variables from an array into the current symbol table. It takes an associative 'array' and treats keys as variable names and values as variable values.
- For each key/value pair it will create a variable in the current symbol table, subject to extract type and prefix parameters.
- The function returns the number of variables successfully imported into the symbol table.

**For example:**

```
<?php
    $n = array("a"=>10, "b"=>35, "c"=>12);
    extract($n);
    echo $a. " " . $b. " " . $c;
?>
```

**Output:**

```
10 35 12
```

**2. compact() Function:**

- The compact() function creates an array from variables and their values.

**Syntax:** array compact(mixed \$varname1 [, mixed \$... ])

- This function is the complement of extract( ). It takes variable names as parameters and creates an associative array whose keys are the variable names and whose values are the variable's values.

- Returns the output array with all the variables added to it.

**For example:**

```
<?php
    $n1 = 10;
    $n2 = 20;
    $n3 = 30;
    $arr = array("n1", "n2", "n3");
    $output = compact($arr);
    print_r($output);
?>
```

**Output:**

```
Array ([n1] => 10 [n2] => 20 [n3] => 30)
```

## 3.7 TRAVERSING ARRAYS

- Traversing an array means to visit each and every element of array using a looping structure and iterator functions.
- There are several ways to traverse arrays in PHP. Some of them are given below:

**Using foreach Construct:**

- PHP provides a very special kind of looping statement called foreach that allows you to loop over arrays. The foreach statement only works with arrays and objects.

**For example:**

```
$a = array('aaa', 'bbb', 'ccc');
foreach($a as $value)
{
    echo "$value\n";
}
```

**Output:**

```
aaa
bbb
ccc
```

- In the above foreach loop, the loop will be executed once for each element of \$a, i.e. 3 times. And for each iteration \$value will store the current element.

**For example:**

```
$a=array('one' => 1, 'two' => 2, 'three' => 3);
foreach ($a as $k => $v)
{
    echo "$k is $v \n";
}
```

**Output:**

```
one is 1
two is 2
three is 3
```

- In this case the key for each element is stored in \$k and the corresponding value is stored in \$v.
- The foreach operates on copy of array, not on array itself.

**Using a for Loop:**

- The for loop operates on the array itself, not on a copy of the array.

**For example:**

```
$a = array(1, 2, 3, 4);
for($i=0; $i<count($a); $i++)
{
    echo $a[$i] . "<br>";
}
```

**Output:**

```
1
2
3
4
```

**Using Iterator Functions:**

- Every PHP array keeps track of the current element. The pointer to the current element is known as the iterator. PHP has functions to set, move, reset this iterator.
- The iterator functions are:
  1. **current()**: Returns the currently pointed element.
  2. **reset()**: Moves the iterator to the first element in the array and returns it.
  3. **next()**: Moves the iterator to the next element in the array and returns it.
  4. **prev()**: Moves the iterator to the previous element in the array and returns it.
  5. **end()**: Moves the iterator to the last element in the array and returns it.
  6. **each()**: Returns the key and value of the current element as an array and moves the iterator to the next element in the array.
  7. **key()**: Returns the key of the current element.

**For example:**

```
<?php
$transport = array('foot', 'bike', 'car', 'plane');

$mode = current($transport); // $mode = 'foot';
$mode = next($transport); // $mode = 'bike';
$mode = current($transport); // $mode = 'bike';
$mode = prev($transport); // $mode = 'foot';
$mode = end($transport); // $mode = 'plane';
$mode = current($transport); // $mode = 'plane';

?>
```

- After executing each() the iterator moves to the next element.

```
<?php
    $a = array(10, 20, 30, 40, 50);
    reset($a);
    while (list($key, $val) = each($a))
    {
        echo "$key => $val <br>";
    }
?>
```

**Output:**

```
0 => 10
1 => 20
2 => 30
3 => 40
4 => 50
```

**Calling a Function for Each Array Element:**

- The array\_walk() function apply a user defined function to each element of an array.
- Syntax:** bool array\_walk (array &\$arr, callable \$function\_name)
- This function takes two parameters, first is the input array and second is the user defined function name.
  - The user defined function takes two arguments, first contains arr's value and second contains arr's key.
  - Returns True on success or False on failure.

**For example:**

```
<?php
    function print_row($v, $k)
    {
        echo "$v $k <br>";
    }
    $a = array(10, 20, 30, 40);
    array_walk($a, 'print_row');
?>
```

**Output:**

```
10 0
20 1
30 2
40 3
```

- In the above program the ‘print\_row’ function will be called 4 times i.e. for each elements of the array \$a. The ‘print\_row’ function then displays the values along with the keys.

### Reducing an Array:

- The array\_reduce() function apply a user defined function to each element of an array, so as to reduce the array to a single value.

**Syntax:** `mixed array_reduce(array $array, callable $callback [, mixed $initial = NULL])`

- The function takes two arguments: the running total, and the current value being processed. It should return the new running total.

### For example:

```
<?php
    function add($sum, $value)
    {
        $sum += $value;
        return $sum;
    }
    $n = array(2, 3, 5, 7);
    $total = array_reduce($n, 'add');
    echo $total;
?>
```

### Output:

17

- The function ‘add’ will be called for each element, i.e. 4 times. The function then finds the sum and returns it.
- If the optional initial is available, it will be used at the beginning of the process.

```
$total = array_reduce($n, 'add', 10);
echo $total; // 27 (i.e. 10 + 17)
```

### Searching for Values:

- The in\_array() function searches if a value exists in an array or not.

#### Syntax:

```
bool in_array(mixed $to_find, array $input [, bool $strict = FALSE])
```

- The in\_array() function returns true or false, depending on whether the element ‘to\_find’ is in the array ‘input’ or not.

**For example:**

```
<?php
    $os = array("Mac", "NT", "Irix", "Linux");
    if (in_array("Irix", $os))
    {
        echo "Got Irix";
    }
    if(in_array("mac", $os))
    {
        echo "Got mac";
    }
?>
```

- The second condition is false because `in_array()` is case-sensitive, so the program above will display:  
**Got Irix**
- If the third parameter **strict** is set to TRUE then the `in_array()` function will also check the types of the \$value.

**For example:**

```
<?php
    $a = array(2, 3, "4", "5");
    if(in_array('3', $a, true))
    {
        echo "'3' found with strict check\n";
    }
    if(in_array('4', $a, true))
    {
        echo "'4' found with strict check\n";
    }
?>
```

**Output:**

**'4' found with strict check**

- In the above program the type of '3' which we are searching is string but in the array 'a', 3 is integer. Hence first condition is false.

**array\_search() Function:**

- The `array_search()` function search an array for a value and returns the key.

**Syntax:**

```
mixed array_search (mixed $ to_find, array $input [, bool $strict = FALSE])
```

- The `in_array()` function returns the key of the element ‘to\_find’ if it is found in the array ‘input’, otherwise returns FALSE.

**For example:**

```
<?php
    $a=array("a"=>"5", "b"=>5, "c"=>"5");
    echo array_search(5,$a,true);
?>
```

**Output:**

```
b
```

## 3.8 SORTING

[April 16]

- The functions provided by PHP to sort an array are shown in following table:

|                       |                    |                                                                |
|-----------------------|--------------------|----------------------------------------------------------------|
| <code>sort()</code>   | Ascending          | sort indexed array by values reassign indexes starting with 0. |
| <code>rsort()</code>  | Descending         |                                                                |
| <code>usort()</code>  | user_defined order |                                                                |
| <code>asort()</code>  | ascending          |                                                                |
| <code>arsort()</code> | descending         | sort array by values.                                          |
| <code>uasort()</code> | user_defined order |                                                                |
| <code>ksort()</code>  | ascending          |                                                                |
| <code>krsort()</code> | descending         |                                                                |
| <code>uksort()</code> | user_defined order | sort array by keys.                                            |

- Let us see above functions in detail.

### 1. `sort()` Function:

- This function sorts an indexed array in ascending order. This function assigns new keys for the elements in array.

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

**For example:**

```
<?php
    $fruits = array("lemon", "orange", "banana", "apple");
    sort($fruits);
    print_r($fruits);
?>
```

**Output:**

```
Array ( [0] => apple [1] => banana [2] => lemon [3] => orange )
```

- The optional second parameter sort\_flags may be used to modify the sorting behavior using these values:
  - (i) **SORT\_REGULAR**: Compare items normally (don't change types)
  - (ii) **SORT\_NUMERIC**: Compare items numerically
  - (iii) **SORT\_STRING**: Compare items as strings
  - (iv) **SORT\_NATURAL**: Compare items as strings using "natural ordering" like natsort()

### 2. **rsort() Function:**

- The syntax of rsort() function is same but it sorts an indexed array in descending order.

### 3. **usort() Function:**

- The usort() function sorts an array using a user-defined comparison function.

**Syntax:** bool usort(array &\$array, callable \$value\_compare\_func)

- The 'value\_compare\_func' is a user defined function where, the first argument is considered to be less than, equal to, or greater than the second, then the function return an integer less than, equal to, or greater than zero respectively.

**For example:**

```
int callback (mixed $a, mixed $b)
<?php
    function my_sort($a,$b)
    {
        if ($a==$b) return 0;
        return ($a<$b)?-1:1;
    }
    $a=array(4,2,8,6);
    usort($a,"my_sort");
    print_r($a); // Array ( [0] => 2 [1] => 4 [2] => 6 [3] => 8 )
?>
```

### 4. **asort() Function:**

**[Oct. 16, 17, 18]**

- This function mainly used to sort associative array.
- The function maintains their key/value association after sorting the array elements.

**For example:**

```
<?php
    $fruits = array("d" => "lemon", "a" => "orange", "b" => "banana",
                    "c" => "apple");
    asort($fruits);
    print_r($fruits);
?>
```

**Output::**

```
Array ( [c] => apple [b] => banana [d] => lemon [a] => orange )
```

**5. ksort() Function:****[April 18]**

- This function sorts an array by key, maintaining key to data correlations. This is useful mainly for associative arrays.

**For example:**

```
<?php
$fruits = array("d" => "lemon", "a" => "orange", "b" => "banana",
                "c" => "apple");

ksort($fruits);
print_r($fruits);
?>
```

**Output:**

```
Array ( [a] => orange [b] => banana [c] => apple [d] => lemon )
```

**3.8.1 Natural Order Sorting**

- PHP's built-in sort functions correctly sort strings and numbers, but they don't correctly sort strings that contain numbers.
- For example:** a1.php, a10.php, a5.php are three files. The normal order sorting function will sort in this order, a1.php a10.php a5.php.
- For correct sort, we use natsort() and natcasesort() functions.
- The natsort() function sorts an array by using a "natural order" algorithm. The values keep their original keys.

**Syntax:** bool natsort(array &\$input)**For example:**

```
<?php
$array1 = $array2 = array("img12.png", "img10.png", "img2.png",
                           "img1.png");
echo "Standard sorting<br>";
sort($array1);
print_r($array1);
echo "<br>Natural order sorting<br>";
natsort($array2);
print_r($array2);
?>
```

**Output:**

```
Standard sorting
Array ( [0]=>img1.png [1]=>img10.png [2]=>img12.png [3]=>img2.png)
Natural order sorting
Array ( [3]=>img1.png [2]=>img2.png [1]=>img10.png [0]=>img12.png)
```

### 3.8.2 Sorting Multiple Arrays at Once

- The array\_multisort() function sorts multiple arrays. The function takes a series of arrays and sorting orders (SORT\_ASC, SORT\_DESC) as a parameter and sort several arrays at once.

**Syntax:** array\_multisort(array1 [, array2, ... ]);

**For example:**

```
<?php
    $ar1 = array(10, 100, 100, 0);
    $ar2 = array(1, 3, 2, 4);
    array_multisort($ar1, $ar2);
    var_dump($ar1);
    var_dump($ar2);
?>
```

**Output:**

```
array(4)
{
    [0]=> int(0)
    [1]=> int(10)
    [2]=> int(100)
    [3]=> int(100)
}
array(4)
{
    [0]=> int(4)
    [1]=> int(1)
    [2]=> int(2)
    [3]=> int(3)
}
```

- In this example, after sorting, the first array will contain 0, 10, 100, 100. The second array will contain 4, 1, 2, 3.
- The entries in the second array corresponding to the identical entries in the first array (100 and 100) were sorted as well.

### 3.8.3 Reversing Arrays

- We can reverse array elements. The PHP array\_reverse() function to reverse the order of the array elements. It returns an array with items in reverse order.

#### 1. array\_reverse() Function:

- The array\_reverse() function returns an array with elements in reverse order.

**Syntax:**

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

- If ‘preserve\_keys’ is set to True then numeric keys are preserved.

**For example:**

```
<?php
    $input = array('a', 'b', 'c', 'd');
    $reversed = array_reverse($input);
    $preserved = array_reverse($input, true);
    print_r($input); // Array ( [0] => a [1] => b [2] => c [3] => d )
    print_r($reversed); // Array ( [0] => d [1] => c [2] => b [3] => a )
    print_r($preserved); // Array ( [3] => d [2] => c [1] => b [0] => a )
?
>
```

## 2. array\_flip() Function:

[Oct. 18]

- The array\_flip() function flips/exchanges all keys with their associated values in an array.

**Syntax:** array array\_flip(array \$input)

**For example:**

```
<?php
    $a1=array("a"=>"red", "b"=>"green", "c"=>"blue");
    $result=array_flip($a1);
    print_r($result);
?
>
```

**Output:**

Array ( [red] => a [green] => b [blue] => c )

## 3. shuffle() Function:

[Oct. 16]

- The shuffle() function is used to traverse the elements in an array in a random order. After shuffling the function assigns new keys to the elements

**Syntax:** shuffle(\$array);

**For example:**

```
<?php
    $input = array('a', 'b', 'c', 'd');
    shuffle($input);
    print_r($input);
?
>
```

**Output:**

Array ( [0] => c [1] => d [2] => b [3] => a )

## 3.9 ACTION OR ACTING ON ENTIRE ARRAY

- PHP has several useful functions for modifying or applying an operation to all elements of an array.

- We can merge arrays, find the difference, calculate the total, and more, all using built-in functions.

### 1. array\_sum() Function:

- The array\_sum() function returns the sum of all the values in the array.

**Syntax:** `array_sum($array);`

---

**For example:**

```
<?php
    $a = array(2, 4, 6, 8);
    echo "sum(a) = " . array_sum($a) . "<br>";
    $b = array("a" => 1.2, "b" => 2.3, "c" => 3.4);
    echo "sum(b) = " . array_sum($b) . "<br>";
?>
```

**Output:**

```
sum(a) = 20
sum(b) = 6.9
```

---

### 2. array\_merge() Function:

- Merges the elements of one or more arrays together so that the values of one are appended to the end of the previous one.

**Syntax:**

```
array array_merge(array $array1 [, array $array2 [, array $array3...]])
```

- After merging the numeric keys are renumbered.
- 

**For example:**

```
<?php
    $a1=array("red","green");
    $a2=array("blue","yellow");
    print_r(array_merge($a1,$a2));
?>
```

**Output:**

```
Array ( [0] => red [1] => green [2] => blue [3] => yellow )
```

---

- If the input arrays have the same string keys, then the later value for that key will overwrite the previous one.
  - If however, the arrays contain numeric keys, the later value will not overwrite the original value, but will be appended.
- 

**For example:**

```
<?php
$array1 = array("color" => "red", 2, 4);
$array2 = array("a", "b", "color" => "green", "shape" => "trapezoid",4);
$result = array_merge($array1, $array2);
print_r($result);
?>
```

---

**Output:**

```
Array
{
    [color] => green
    [0] => 2
    [1] => 4
    [2] => a
    [3] => b
    [shape] => trapezoid
    [4] => 4
}
```

---

**3. array\_diff() Function:**

- The array\_diff() function identifies values from one array that are not present in others.

**Syntax:**


---

```
array array_diff ( array $array1, array $array2 [, array $array3 ...] );
```

**For example:**

```
<?php
    $a = array(1, 2, 3, 4);
    $b = array(2, 4);
    $result=array_diff($a,$b);
    print_r($result);
?>
```

**Output:**


---

```
Array ( [0] => 1 [2] => 3 )
```

- From array \$a the values 1 and 3 is not present in the array \$b.

**4. array\_filter() Function:**

**[April 18, Oct. 18]**

- The array\_filter() function filters the values of an array using a callback function.
- This function passes each value of the input array to the callback function. If the callback function returns true, the current value from input is returned into the result array. Array keys are preserved.

**Syntax:**


---

```
array array_filter ( array $input [, callback $callback] );
```

**For example:**

```
<?php
    function is_odd($var)
    {
        return ($var % 2);
    }
```

```
$a = array(6, 7, 8, 9, 10, 11, 12);
$odd = array_filter($a, "is_odd");
    echo "Odd Numbers:<br>";
print_r($odd);
?>
```

## Output:

## Odd Numbers:

```
Array ( [1] => 7 [3] => 9 [5] => 11 )
```

- In the above program array\_filter function will call ‘is\_odd’ function. Each element will be passed one by one.
  - For odd value the function will return 1 i.e. true. For true the array\_filter function returns the current value which will be stored in the array \$odd.

# PRACTICE QUESTIONS

## **Q.I Multiple Choice Questions:**

1. Which function removes the first element from an array, and returns the value of the removed element?  
(a) array\_unshift()  
(b) array\_pop()  
(c) array\_shift()  
(d) array\_push()
  2. Which function removes duplicate values from an array?  
(a) array\_del()  
(b) array\_duplicate()  
(c) array\_remove()  
(d) array\_unique()
  3. Which of the following are not the iterator functions?  
(a) reset()  
(b) each()  
(c) key()  
(d) walk()
  4. What is output of following PHP code:  

```
$a = array(2, 3, "4", "5");
if (in_array('4', $a, true))
{
    echo "'4' found \n";
}
else
    echo "'4' Not found \n";
```

  
(a) '4' found  
(b) '4' Not found  
(c) Warning  
(d) Both 2 and 3
  5. Which function will merge the values together, forming a new array with the preexisting key as its name?  
(a) array\_merge()  
(b) array\_combine()  
(c) array\_merge\_recursive()  
(d) array\_merge\_all()



17. Which function will pass each element of an array to the user-defined function?  
(a) array\_walk()  
(b) walk()  
(c) array\_walk\_recursive()  
(d) array\_work()
  18. Which function produces a new array consisting of a submitted set of keys and corresponding values?  
(a) array\_merge()  
(b) array\_combine()  
(c) array\_merge\_recursive()  
(d) array\_merge\_all()
  19. Which function returns a key-preserved array consisting only of those values present in the first array that are also present in each of the other input arrays?  
(a) array\_common()  
(b) array\_intersect()  
(c) array\_intersect\_key()  
(d) array\_intersect\_ukey()
  20. Which function will return keys located in an array that is located in any of the other provided arrays?  
(a) array\_common()  
(b) array\_intersect()  
(c) array\_intersect\_key()  
(d) array\_intersect\_ukey()
  21. Which function removes all duplicate values found in an array, returning an array consisting of solely unique values?  
(a) array\_duplicate()  
(b) array\_unique()  
(c) array\_remove\_duplicate()  
(d) array\_unique\_values()
  22. The function \_\_\_\_\_ will return a random number of keys found in an array.  
(a) rand()  
(b) array\_random()  
(c) array\_rand()  
(d) array\_rand\_key()
  23. The function \_\_\_\_\_ allows us to compare the keys of multiple arrays with the comparison algorithm determined by a user-defined function.  
(a) diff()  
(b) array\_diff()  
(c) array\_diff\_key()  
(d) array\_diff\_ukey()
  24. Which function returns an array consisting of associative key/value pairs?  
(a) array\_count()  
(b) array\_count\_keys()  
(c) count()  
(d) array\_count\_values()
  25. Which array function returns a section of an array based on a starting and ending offset value?  
(a) array\_slice()  
(b) array\_flip()  
(c) array\_splice()  
(d) array\_reverse()
  26. Which function removes all elements of an array found within a specified range, returning those removed elements in the form of an array?  
(a) array\_slice()  
(b) array\_remove()  
(c) array\_remove\_all()  
(d) array\_splice()
-

26. What is the output of the following php code?

```
<?php
    $alphabet=array("A", "B", "C");
    echo(next($alphabet));
?>
```

- (a) A
- (b) B
- (c) C
- (d) Error

27. What is the output of the following PHP code?

```
<?php
    $alphabet=array("A" => array("php"=>"php 7.0", "date" => "3 December
    2019"), "B" => array("python" => "python 3.8.2", "date" => "24
    December 2019"));
    echo $alphabet ["A"]["date"];
?>
```

- (a) php 7.0
- (b) 3 December 2019
- (c) python 3.8.2
- (d) 24 December 2019

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

```
<?php
    $arr=array("lets", "find", "course", ".Com");
    echo (array_search (".com", $arr) );
```

- ?>
- (a) 0
- (b) Garbage value
- (c) 3
- (d) No Output

29. Which function returns an array consisting of associative key/value pairs?

- (a) count()
- (b) array\_count()
- (c) array\_count\_values()
- (d) count\_values()

30. Which in-built function will add a value to the end of an array?

- (a) array\_unshift()
- (b) into\_array()
- (c) inend\_array()
- (d) array\_push()

31. Which of the following are correct ways of creating an array?

- |                                    |                                         |
|------------------------------------|-----------------------------------------|
| (i) arr[0] = "letsfindcourse";     | (ii) \$arr[] = array("letsfindcourse"); |
| (iii) \$arr[0] = "letsfindcourse"; | (iv) \$arr = array("letsfindcourse");   |
| (a) (ii), (iii) and (iv)           | (b) (ii) and (iii)                      |
| (c) Only (i)                       | (d) (iii) and (iv)                      |

32. The keys of an indexed array are integers, beginning at,

- (a) -1
- (b) 2
- (c) 0
- (d) 3

33. Which of the following function is used to set the array pointer to the value of last key?

- (a) last()  
(b) end()  
(c) next()  
(d) final()

34. What is the output of following PHP:

```
<?php
    function add($sum, $value)
    {
        $sum += $value;
        return $sum;
    }
$n = array(-2, 3, 5, 7);
$total = array_reduce($n, 'add');
echo $total;
?>
```

- (a) 15  
(b) 13  
(c) 17  
(d) 18

### Answers

1. (c)	2. (d)	3. (a)	4. (a)	5. (c)	6. (a)	7. (a)	8. (c)	9. (b)	10. (b)
11. (a)	12. (a)	13. (c)	14. (b)	15. (d)	16. (b)	17. (a)	18. (b)	19. (b)	20. (c)
21. (b)	22. (c)	23. (d)	24. (d)	25. (a)	26. (d)	27. (b)	28. (b)	29. (d)	30. (d)
31. (d)	32. (c)	33. (b)	34. (b)						

### Q.II Fill in the Blanks:

- PHP arrays stores multiple values of different data type in a \_\_\_\_\_ at a time.
- There are two types of arrays in PHP namely \_\_\_\_\_ and \_\_\_\_\_ arrays.
- To remove a key/value pair, call the \_\_\_\_\_ function on it.
- The \_\_\_\_\_ function creates an array of consecutive integer or character between two values we pass to it as a parameter.
- To divide an array into smaller, evenly sized arrays, use the \_\_\_\_\_ function.
- The \_\_\_\_\_ function is used to traverse the elements in an array in a random order.
- In PHP, an array can be created using the\_\_\_\_\_ language construct
- An \_\_\_\_\_ array has strings as keys .
- The \_\_\_\_\_ functions are used to return the number of elements in the array.
- The array\_key\_exists() function is used to see if an element \_\_\_\_\_ in the array.
- The \_\_\_\_\_ function returns the sum of all the values in the array.
- We can access specific values from an array using the array variable's name, followed by the element's key (index) within \_\_\_\_\_ brackets.

13. In numeric array the default array index starts from \_\_\_\_.
14. To create an array initialized to the \_\_\_\_ value, use array\_pad() function.
15. To copy all of an array's values into variables, use the \_\_\_\_ construct.
16. The array\_splice() function can \_\_\_\_ or \_\_\_\_ elements in an array.
17. PHP provides two functions namely, extract() and compact(), that \_\_\_\_ convert between arrays and variables.
18. There are several ways to traverse arrays in PHP, the most common way to loop over elements of an array is to use the \_\_\_\_ construct.
19. Every PHP array keeps track of the current element you're working with; the pointer to the current element is known as the \_\_\_\_ iterator.
20. The array\_multisort( ) function sorts multiple indexed arrays at \_\_\_\_.
21. The array\_reverse() function reverses the \_\_\_\_ order of elements in an array.
22. The array\_merge( ) function intelligently \_\_\_\_ two or more arrays.

### Answers

1. single variable	2. Indexed and Associative	3. unset()	4. range()	5. array_chunk()
6. shuffle()	7. array()	8. Associative	9. count()	10. exists
11. array_sum()	12. square	13. 0 (zero)	14. same	15. list()
16. remove, insert	17. convert	18. foreach	19. iterator	20. once
21. internal	22. merges			

### Q.III State True or False:

1. PHP arrays stores multiple values of different data type in a single variable at a time.
2. An associative array has strings as keys.
3. To insert more values at the end of existing array, use () syntax.
4. The range() function creates an array of random integer or character between two values we pass to it as a parameter.
5. The count() functions are used to return the number of elements in the array.
6. To copy an array's keys into variables use the list() construct or function.
7. The array\_key\_exists() function is used to see if an element exists in the array.
8. The array\_splice() cannot works on associative arrays.
9. The extract()function automatically creates local variables from an array.
10. The array\_flip() function flips/exchanges all keys with another array.
11. Key or index of an array is a unique number or a string that is associated with each value of an element.
12. To change an existing value of an element, we need to specify the new value in the array mentioning the key of that element in [] brackets.
13. In an indexed array, index values (starting from 1) are created automatically.

14. In a multi-dimensional array each element in the main array can also be an array.
15. The compact() function automatically creates local variables from an array.
16. The array() function is used to create an array.
17. Values in the multi-dimensional array are accessed using single index.
18. To traverse the elements in an array in a random order, use the shuffle().

### Answers

1. (T)	2. (T)	3. (F)	4. (F)	5. (T)	6. (F)	7. (T)	8. (F)	9. (T)	10. (F)
11. (T)	12. (T)	13. (F)	14. (T)	15. (F)	16. (T)	17. (F)	18. (T)		

### Q.IV Answer the following Questions:

#### (A) Short Answer Questions:

1. Define array.
2. What is the purpose of array in PHP?
3. Lists types of arrays.
4. What is indexed array?
5. Which function is used for creating an array?
6. Define multidimensional array.
7. Give the use of associative array.
8. Compare indexed and associative array (any two points).
9. Give the purpose of range().
10. List function for sorting an array.

#### (B) Long Answer Questions:

1. What is array? How to create it? Explain with example.
2. With the help of example describe how to sorting arrays.
3. Explain keys and values in arrays with diagram.
4. How to convert arrays into variables? Explain in detail.
5. Write a note on: Traversing array.
6. Describe multidimensional array with example.
7. What is slicing and splitting? Describe with example.
8. Find output

```
<?php
$marks=array (10, 20);
$padded=array_pad ($marks, -6, 0);
print_r ($padded);
?>
```

9. Find the output:

```
<?php
$a = array ('apple', 'banana', 'coconut');
$b = array_pad ($a, 5, '');
print_r ($b);
?>
```

10. How to extract multiple values in arrays? Explain with example.  
 11. Describe how to identify elements of an array in detail.

## UNIVERSITY QUESTIONS AND ANSWERS

### April 2016

1. What is an associative array? Give an example. [1 M]

**Ans.** Refer to Section 3.2.

2. What is array? Write a short note on multidimensional array with examples. [5 M]

**Ans.** Refer to Sections 3.0 and 3.4.

3. Write a PHP program to sort array on marks, (Array contains names and marks). [5 M]

**Ans.** Refer to Section 3.8.

### October 2016

1. Write the purpose of asort() function. [1 M]

**Ans.** Refer to Section 3.8, Point (4).

2. How to convert an object to array? [1 M]

**Ans.** Refer to Section 3.6.

### April 2017

1. The count() function returns the number of elements in the array. [1 M]

**Ans.** Refer to Section 3.3.3.

2. Define array\_unique(). [1 M]

**Ans.** The array\_unique() function creates and returns an array containing each element in the given array. If any values are duplicated, the later values are ignored. Keys from the original array are preserved.

**Syntax:** array array\_unique(array)

**Example:**

```
<?php
$a=array("a"=>"red","b"=>"green","c"=>"red");
print_r(array_unique($a));
?>
```

**Output:**

```
array ( [a] => red [b] => green )
```

3. Explain the PHP functions used to convert array into variables and vice versa. [4 M]

**Ans.** Refer to Section 3.6.

**October 2017**

1. How to check if given variable is array or not? [1 M]

**Ans.** The `is_array()` function checks whether a variable is an array or not. This function returns true if the variable is an array, otherwise it returns false.

**Syntax:** `is_array(variable);`

2. How to insert and remove the last element of an array. [1 M]

**Ans.** Refer to Section 3.3.1.

3. What is an associative array? Explain with suitable example how it is different from indexed array. Explain `foreach` function(). [5 M]

**Ans.** Refer to Section 3.1.2.

4. Explain the following functions with respect to array:

(i) `extract()` (ii) `shuffle()` (iii) `array_splice()` (iv) `arsort()`.

[4 M]

**Ans.** Refer to Sections 3.6, Point (1), 3.8.3, Point (3), 3.5.5 and 3.8, Point (4).

**April 2018**

1. State the purpose of `array_filter()` function. [1 M]

**Ans.** Refer to Section 3.9, Point (4).

2. What is array? Explain different types of Array with an example. [5 M]

**Ans.** Refer to Sections 3.0 and 3.1.

3. Explain the following functions with respect to Array:

(i) `array_slice()` (ii) `krsort()`.

[2 M]

**Ans.** Refer to Sections 3.5.1 and 3.8, Point (5).

**October 2018**

1. Which function is used to remove duplicate elements from an array? [1 M]

**Ans.** The `array_unique()` function removes duplicate values from an array. If two or more array values are the same, the first appearance will be kept and the other will be removed.

2. How to get array of keys and array of values from an associative array? Illustrate with suitable example using built-in functions. [5 M]

**Ans.** Refer to Section 3.5.3.

3. Write the output of the following PHP script:

```
<?php
$age=array("Peter_35", "Ben_37", "Joe_43");
arsort ($age);
print_r ($age);
?>
```

**Ans.** Array ( [0] => Peter\_35 [2] => Joe\_43 [1] => Ben\_37 )

4. Explain the following functions with respect to Array:

- (i) arrsy\_flip() (ii) array\_filter().

[2 M]

**Ans.** Refer to Sections 3.8.3, Point (2) and 3.9, Point (4).

---

**April 2019**

1. How will you find number of elements in array?

[1 M]

**Ans.** Refer to Section 3.3.3.

2. Write a PHP program to accept associative array of five expenses (electricity bill, phone bill, petrol bill property tax, college fees) and their respective amount of two persons. Print the total expenditure of each person.

[5 M]

**Ans.** Refer to Section 3.1.2.

---

■ ■ ■

# Files and Database Handling

## Objectives...

- To understand Basic Concepts of File and Directories
- To learn Basics of Files like Open, Close, Random Access etc.
- To understand Basic Concepts of Database
- To learn PEAR DB Basics
- To study Relational Databases and SQL

### 4.0 INTRODUCTION

- PHP is a server side programming language. PHP allows us to work with files and directories stored on the Web server.
- In computer a file is a resource for storing information, which is available to a computer program and is usually based on some kind of durable storage.
- File is a collection of data or information that has a name called the filename. Directory is an organizational unit, which is used to organize files into a hierarchical structure.
- File handling is an important part of PHP. PHP has several functions for creating, reading, uploading and editing files. PHP can write or erase files and directories on the server.
- A database is a collection of information that is organized so that it can easily be accessed, managed and updated. With PHP, we can connect to and manipulate databases.
- A database is a collection of related data. A Database Management System (DBMS) is system software for creating and managing databases.
- The DBMS provides users and programmers with a systematic way to create, retrieve, update and manage data.
- Nowadays, we use Relational Database Management Systems (RDBMS) to store and manage huge volume or amount of data.
- PHP works MySQL, PostgreSQL, and Oracle databases are the backbone of most modern dynamic web sites.

**4.1****WORKING WITH FILES AND DIRECTORIES**

- PHP provides a set of functions that allow us to work with directories effectively. The input and output in a web application usually flow between browser, server and database, but there are many circumstances in which files are involved too.
- Files are useful for retrieving remote web pages for local processing, storing data without a database, and saving information that other programs need access to.
- PHP provides a set of file access functions for the purposes like opening and closing the file, reading from and writing to file, renaming and deleting a file, splitting name and path from file.
- PHP provides a number of functions that can be used to perform tasks such as identifying and changing the current directory, creating new directories, deleting existing directories and listing the contents of a directory.
- A file system stores a lot of additional information about files aside from their actual contents. This information includes such particulars as the file's size, what directory it's in, and access permissions for the file.

**4.1.1 Opening and Closing Files**

- There are typically three steps involved in working with file:
  1. Open the file by associating a file handle with it.
  2. Read from or write to the file.
  3. Close the file.
- Let us see various functions used by PHP for file handling.

**1. fopen() Function:**

- Before doing anything else with a file, we need to open it. To open a file or URL in PHP, we use the fopen() function.
- In simple words, the PHP fopen() function is used to open a file. The fopen() function opens the file and returns a file handle.
- A file handle is simply a file pointer resource that associated with the file. Through the file handle, we can manipulate the content of the file.

**Syntax:** `$file_handle = fopen(filename, mode)`

- The fopen() functions takes three arguments; filename, mode and the optional use\_include\_path.
  - The first parameter is the name of the file that we want to open.
  - The second parameter specifies the mode that indicates how the file is being used e.g. open file for reading, writing or appending.

- Files modes can be specified as one of the following six options:

Mode	Purpose
r	Opens the file for reading only. If file with specified name does not exists then fails to open file. Places the file pointer at the beginning of the file.
r+	Opens the file for reading and writing. If file with specified name does not exist then creates new file with same new and opens it. Places the file pointer at the beginning of the file.
w	Opens the file for writing only. Places the file pointer at the beginning of the file. If file does not exist then it creates a new file. If file exist, all the contents are deleted.
w+	Opens the file for reading and writing only. Places the file pointer at the beginning of the file. If file does not exist then it creates a new file. If file exist, all the contents are deleted i.e. truncate the contents of file.
a	Opens the file for writing only. Places the file pointer at the end of the file. If file does not exist then it creates a new file.
a+	Opens the file for reading and writing only. Places the file pointer at the end of the file. If file does not exist then it creates a new file.

- If the third optional parameter is set to 1, the function searches for the file specified by filename first in own directory, and then in the directories defined by the variable include\_path (set in php.ini file) like /home/apache.
- The function returns false if some error occurs while opening a file.

**For example:**

```
$fh=fopen("filename", "r");
if(!$fh) die ("cannot open the file");
```

Alternatively, we can write it like:

```
$fh = fopen("filename", "r") die ("cannot open");
```

We can even specify a file on a remote host, opening it with a HTTP URL or via FTP as,

```
if(!($fh=fopen(http://www.example.com/index.html), "r")))
die("cannot open the file");
```

**2. fclose() Function:**

- The file needs to be closed, once we finished working on that file. We can do this using fclose(), having single argument as file handle.

**Syntax: fclose(\$file)**

- This function returns True on success or false on failure.

---

```
fclose($fp);
```

- Although PHP should close all open files automatically when our script terminates.

```
<?php
    $file = fopen("test.txt", "r");
    //some code to be executed
    fclose($file);
?>
```

## 4.1.2 Getting Information about File

[Oct. 18]

- The information about files like size, when they were modified, owner of file etc. can be known.
- The stat() function gives information about file by providing the filename as an argument and it returns an indexed array that contains file statistics and information within each spot in the array.
- There are 13 array spots which are given below:

Index Number	Associative Name	Information Contained
0	Dev	Device number
1	Ino	Inode number
2	Mode	Inode protection mode
3	nlink	Number of links
4	uid	User Id of owner
5	gid	Group ID of owner
6	rdev	Device type, if inode device
7	size	Size in bytes
8	atime	Time of last access
9	mtime	Time of last modification
10	ctime	Time of last change
11	blksize	Block size of file system IO
12	blocks	Number of blocks allocated

- Syntax of stat() function is: stat(filename)

For example:

```
<?php
    $stat_info = stat('test.txt');
    echo 'Access time: ' . $stat_info['atime'];
    echo '<br />Modification time: ' . $stat_info['mtime'];
    echo '<br />Device number: ' . $stat_info['dev'];
?>
```

**Output:**

```
Access time: 1141633430
Modification time: 1141298003
Device number: 0
```

- The following example retrieves the size of file my.txt if exists.

```
$my_file_stats = stat("my.txt");
$my_file_size = $my_file_stats [7];
```

### 4.1.3 Reading and Writing to Files

[April 17, 19]

- We can use fread() and fwrite() function to read and write the data in file.

#### 1. fwrite() Function:

[Oct. 18, April 19]

- The fwrite() function writes the contents of string to a file.
- The function will stop at the end of the file or when it reaches the specified length, whichever comes first.
- This function returns the number of bytes written or false on failure.

**Syntax:** fwrite(file, string, length)

- It has two arguments: a file handle (\$fp) and a string (string of text to the file) returning the number of bytes written (or -1 on error).

**For example:** fwrite(\$fp, "abcdefghijkl", 4);

- Writes the first 4 bytes of "abcdefghijkl" (that is, "abcd") to the file referenced by \$fp.

**For example:**

```
<?php
    $file = fopen("pragati.txt", "w");
    echo fwrite($file, "Hello World. Testing!");
    fclose($file);
?>
```

**Output:**

21

- In above example 21 is the total number of characters (bytes) written in the file 'pragati.txt'.

#### 2. fread() Function:

[April 19]

- The PHP fread() function is used to read the content of the file.
- The function will stop at the end of the file or when it reaches the specified length, whichever comes first.
- The fread() function returns the read string or false on failure.

**Syntax:** fread(file, length)

- It takes two arguments namely, a file handle fp and an integer length.

```
$fp=fopen("data.txt", "r")
```

and then say:

```
$data=fread($fp, 10);
```

- The fread() function will read the first 10 bytes from data.txt and assign them to \$data as a string.

#### **4.1.4 Splitting the Name and Path from a File**

---

- In order to get a file name out of a file path, we use the basename() function. It accepts a complete file path as a parameter and returns the file name.

**Syntax:** `string basename(string $path [, string $suffix]);`

---

**For example:**

```
<?php
    $path = "/home/httpd/html/contact.php";
    $file = basename($path);
    echo "File name is $file\n";
    $file = basename($path, ".php");
    echo "File name is $file\n";
?
?
```

**Output:**

```
File name is contact.php
File name is contact
```

---

**File Extension:**

**[April 13]**

- To get file extension only, we use the function pathinfo().

**Syntax:** `pathinfo(string path,[int options])`

---

**For example:**

```
<?php
    $path_file = "/www/Nirali/abc.php";
    echo pathinfo ($pathfile, PATHINFO_DIRNAME);
    echo pathinfo ($pathfile, PATHINFO_BASENAME);
    echo pathinfo ($pathfile, PATHINFO_EXTENSION);
    echo pathinfo ($pathfile, PATHINFO_FILENAME);
?
?
```

**Output:**

```
/www/niraliabc.php
php
abc
```

---

- if option is used it returns a specified element from the following:
  - PATHINFO\_DIRNAME: Returns dirname with path.
  - PATHINFO\_BASENAME: Returns filename with path.
  - PATHINFO\_EXTENSION: Returns extension of file.
  - PATHINFO\_FILENAME: Returns filename without extension.

### 4.1.5 Renaming and Deleting File

- The rename() function in PHP is an inbuilt function which is used to rename a file or directory. It makes an attempt to change an old name of a file directory with a new name specified by the user.
- Deleting a file in PHP means completely erase a file from a directory so that the file is no longer exist. PHP has an unlink() function that allows us to delete a file.

#### 1. Renaming File:

- The rename() function is used to rename a file or directory.
- **Syntax:** `bool rename(oldname, newname)`
- This function also allows us to move a file to a different directory. For example, to rename the `nirali.txt` file to `niralipublication.txt`, we use the following code:

**For example:**

```
<?php
$fn = './nirali.txt';
$newfn = './niralipublication.txt';
if(rename($fn,$newfn))
{
    echo sprintf("%s was renamed to %s",$fn,$newfn);
}
else
{
    echo 'An error occurred during renaming the file';
}
?>
```

**Output:**

`nirali.txt was renamed to niralipublication.txt`

#### 2. Deleting File:

**[Oct. 18, April 19]**

- The unlink() function takes a single string argument referring to the name of a file we want to delete.
- This function returns True on success, or False on failure.

**Syntax:** `bool unlink(filename);`

**For example:**

```
<?php
    $fn = './backup/test.bak';
    if(unlink($fn))
    {
        echo sprintf("The file %s deleted successfully",$fn);
    }
    else
    {
        echo sprintf("An error occurred deleting the file %s",$fn);
    }
```

#### 4.1.6 Copying Files

- To copy a file, we use the `copy()` function. First, we need to determine which file to copy by passing the file path to the first parameter of the `copy()` function. Second, we need to specify the file path to copy the file to.
- The `copy()` function returns true if the file was copied successfully, otherwise it returns false.

**Syntax:** `bool copy(string $source, string $dest);`

**For example:**

```
<?php
    echo copy("source.txt","nirali.txt");
?>
```

**Output:**

1

#### 4.1.7 Reading and Writing Characters in Files

- PHP provides set of functions for reading and writing characters in files i.e. `fgetc()`, `feof()`, `fgets()`, `fgetcsv()`, and `fputs()`.

##### 1. **fgetc() Function:**

[April 19]

- The `fgetc()` function reads a single character from a file. Returns FALSE on end of file.

**Syntax:** `string fgetc(resource $handle);`

**For example:**

```
<?php
    $file = fopen("nirali2.txt","r");
    while (!feof($file))
    {
        echo fgetc($file);
    }
    fclose($file);
?>
```

**Output:**

Hello, this is a nirali file.

- The feof() function returns true on reaching the end of a specified file (or if an error occurs) and returns false otherwise. So, when the file pointer reaches at end of file the while loop terminates.
- The fgetc() is the same as the fread():

```
$one_char = fgetc($fp) it is same as
$one_char = fread($fp, 1)
```

## 2. fgets() Function:

[Oct. 17, April 19]

- This function is used to read sets of characters.
- Syntax:** string fgets(resource \$handle [, int \$length]);
- This function takes two arguments, fp and length and returns a string of maximum length (length-1) in bytes, as read from the file pointed to by fp.
- It stops reading for any one of the three reasons:
  - (i) The specified number of bytes has been read.
  - (ii) A new line is encountered.
  - (iii) The end of the file is reached.
- The difference between fgets() and fread() is that fgets() stops reading when it reaches end-of-line and reads upto length-1 bytes, while fread() reads past end-of-line and reads upto length bytes.

### For example:

```
<?php
    $file = fopen("nirali.txt","r");
    echo fgets($file);
    fclose($file);
?>
```

### Output:

Hello, this is a nirali file

## 3. fgetcsv() Function:

- The fgetcsv() function read the data in files with the assumption that it is properly formatted in csv (comma separated-value) and puts data into an array.

### Syntax:

```
array fgetcsv (resource $handle [, int $length [, string $delimiter
                                [, string $enclosure [, string $escape]]]]);
```

- The fgetcsv() function must be given a valid file handle, a numerical value higher than the length of each line (including the end-of-line characters) and we can optionally specify data delimiters (the default is comma) and data enclosures (the default is double-quotes).

- The following code snippet shows how we might retrieve a line of data values from a file in csv format:

**For example:**

```
<?php
    $file = fopen("contacts.csv","r");
    while(! feof($file))
    {
        print_r(fgetcsv($file));
    }
    fclose($file);
?>
```

**The CSV file (contacts.csv) contains:**

```
Amar Salunkhe, Refsnes, Stavanger, Norway
Hege, Refsnes, Stavanger, Norway
```

**Output:**

```
Array
{
[0] => Amar Salunkhe
[1] => Refsnes
[2] => Stavanger
[3] => Norway
}
Array
{
[0] => Hege
[1] => Refsnes
[2] => Stavanger
[3] => Norway
}
```

#### 4. fputs() Function:

[April 18]

- The fputs() function is an alias of the fwrite() function.
- The fputs() function writes to an open file.

**Syntax:** fputs(file, string, length)

**For example:**

```
<?php
    $file=fopen("nirali.txt","w");
    echo fputs($file,"Hello World.Testing!");
    fclose($file);
?>
```

**Output:**

20

## 4.1.8 Directories

[April 16]

- PHP provides a set of handy functions that allow us to work with directories in the same way as files.
- We manipulate a directory through a directory handle that is a special variable pointing to a directory.
- The directory functions allow us to retrieve information about directories and their contents.
- The opendir() function opens up a directory handle to be used in subsequent closedir(), readdir(), and rewinddir() calls.
- The closedir() function closes a directory handle. The readdir() function reads the specified directory and returns the file names contained by that directory.

## 4.2 READING ENTIRE FILES

[April 16, Oct. 17, 18]

### 1. file() Function:

- The file() reads a file into an array. Each array element contains a line from the file, with newline still attached.

**Syntax:** file(path, include\_path, context)

**For example:**

```
$ary=file ("/home/BCS/f.txt");
foreach ($ary as $val)
    echo $val;
```

- The file() returns the entire contents of f.txt (in the directory /home/BCS) as an array, using the newline character to delimit elements.
- The newline character remains attached at the end of each line stored in the array. It automatically opens, reads, and once it's done, closes the file.
- We can do the same using a mixture of feof() and fgets() functions.
- It can also fetch files on a remote host:

```
$file_lines=file("http://www.bgcollege.com/index.html");
foreach ($file_lines as $line)
    echo $line;
```

**For example:**

```
<?php
    print_r(file("nirali.txt"));
?>
```

**Output:**

```
Array
(
    [0] => Hello World. Testing testing!
    [1] => Another day, another line.
    [2] => If the array picks up this line,
    [3] => then is it a pickup line?
)
```

---

**2. fpassthru() Function:**

- The fpassthru() function prints the entire file to the web browser from the current position to EOF.
- This function returns the number of characters passed or False on failure.

**Syntax:** fpassthru(file)**For example:**

```
<?php
    $counter_file = "./count.dat";
    if(!$fp=fopen($counter_file, "r"))
        die("cannot open $counter_file");
    echo "File is opened";
    fpassthru($fp);
?>
```

- The file is closed when the function finishes reading, so there is no need to call fclose().

**3. readfile() Function:****[April 18]**

- The readfile() function enables us to print the contents of a file without even having to call fopen().
- It takes a filename as its single argument, reads the whole file and then writes it to standard output, returning the number of bytes read.

**Syntax:** readfile(filename, include\_path, context)**For example:**

```
<?php
    echo readfile("nirali.txt");
?>
```

**Output:**

```
There are two lines in this file.
This is the last line.
```

## 4.3 RANDOM ACCESS TO FILE DATA

- We can move the file position indicator around in the file without having to close and reopen the file. There are two function to do this i.e. fseek() and ftell().

### 1. fseek() Function:

[April 18]

- This function moves the file pointer from its current position to a new position, forward or backward, specified by the number of bytes.
  - The fseek() function returns 0 on success, or -1 on failure.
- Syntax:** fseek(\$file,offset,whence)
- The fseek() will move the file position indicator associated with fp to a position determined by offset. If offset is negative file pointer is set in backward order.

```
fseek( $fp, 5)
$one_char=fgetc($fp);
```

- The call to fgetc() will returns the contents of the sixth byte.
- The third optional argument which can have value like:
  - SEEK\_SET : The beginning of the file + offset
  - SEEK\_CUR : Current position + offset (default)
  - SEEK\_END : End of the file + offset

#### For example:

```
<?php
$file = fopen("nirali.txt","r");
// read first line
fgets($file);
// move back to beginning of file
fseek($file,0);
?>
```

### 2. ftell() Function:

- The ftell() function takes a file handle and returns the current offset (in bytes) of the corresponding file position indicator or false on failure.

**Syntax:** ftell(file)

#### For example:

```
<?php
$file = fopen("nirali.txt","r");
// print current position
echo ftell($file);
// change current position
fseek($file,"20");
```

```
// print current position again
echo "<br />" . ftell($file);
fclose($file);
?>
```

**Output:**

```
0
20
```

**3. rewind() Function:**

- The rewind() function "rewinds" the position of the file pointer to the beginning of the file.
- The rewind() function returns TRUE on success, or FALSE on failure.

**Syntax:** `rewind(file)`**For example:**

```
<?php
    $file = fopen("nirali.txt","r");
    //Change position of file pointer
    fseek($file,"20");
    //Set file pointer to 0 ...
    rewind($file);
    fclose($file);
?>
```

**4.4 GETTING INFORMATION ON FILES**

- PHP provides some functions using that we can access useful file information.

**1. file\_exists() Function:**

- This function returns true if the file or directory specified by the parameter ‘filename’ exists, false otherwise.

**Syntax:** `bool file_exists(string $filename);`**For example:**

```
<?php
    $file_x = "my_file.php";
    if (file_exists($file_x))
    {
        echo $file_x . " exists";
    }
```

```

        else
        {
            echo $file_x . " does not exist";
        }
    ?>

```

- We can use file\_exists() to check whether the file exists.

## 2. file\_size() Function:

- To get the size of the file, we use the file\_size() function.
- The filesize() function returns the size of a given file in bytes, or false in case an error occurred.

**Syntax:** filesize(filename)

- The following example shows us how to get the size of the test.txt file that resides in the same directory as the script file.

```

<?php
$fn = './nirali.txt';
echo filesize($fn);
?>

```

### 4.4.1 Time Related Properties

[Oct. 18]

- In this section we will study time related properties for files in PHP.

## 1. fileatime() Function:

[Oct. 17]

- The function fileatime() returns the last access time for a file in a UNIX timestamp format.

**Syntax:** fileatime(filename)

**For example:**

```

<?php
echo fileatime("nirali.txt");
echo "<br />";
echo "Last access: ".date("F d Y H:i:s.",fileatime("test.txt"));
?>

```

**Output:**

```

1140684501
Last access: February 23 2006 09:48:21.

```

## 2. filectime() Function:

[Oct. 18]

- The filectime() function returns the time at which the file was last changed as a Unix timestamp.
- A file is considered changed if it is created or written or when its permission has been changed.

**Syntax:** filectime(filename)

**For example:**

```
<?php
    echo filectime("nirali.txt");
    echo "<br />";
    echo "Last change: ".date("F d Y H:i:s.",filectime("test.txt"));
?
>
```

**Output:**

```
1138609592
Last change: January 30 2006 09:26:32.
```

### 3. **filemtime() Function:**

[April 16, 18, Oct. 16]

- The filemtime() returns the time at which the file was last modified as a Unix time stamp.
- A file is considered modified if it is created or has its contents changed.

**Syntax:** `filemtime(filename)`

**For example:**

```
<?php
    echo filemtime("nirali.txt");
    echo "<br />";
    echo "Last modified: ".date("F d Y H:i:s.",filemtime("test.txt"));
?
>
```

**Output:**

```
1139919766
Last modified: February 14 2006 13:22:46.
```

## 4.5 OWNERSHIP AND PERMISSIONS

[April 17]

- We can get information on file ownership and permissions. All files are associated with a specific user and a specific group of users and assigned flags that determine who has permission to read, write or execute their contents.
- There are three permissions associated with the files which are read, write or execute. Each of these three permissions (read, write, execute) can be granted to (or withheld from):
  1. **File Owner:** By default, the user whose account was used to create the file.
  2. **A Group of Users:** By default, the group to which the owner belongs.
  3. **All Users:** Everyone with an account on the system.

- Let us see various functions used in PHP for file ownership and permissions:

### 1. posix\_getpwuid() Function:

[April 17]

- When we want to get information of a user by his ID number, we can use the posix\_getpwuid() function, which returns an associative array with the following references:

Name	Explanation
name	The username.
passwd	The encrypted user password.
uid	ID number of user.
gid	Group id of user.
gecos	A comma separated list containing the user's details.
dir	Absolute path to the home directory of the user.
shell	Absolute path to the user's default shell.

- The **Syntax** for posix\_getpwuid() function: `array posix_getpwuid(int $uid)`  
Returns an array of information about the user referenced by the given user ID.

**For example:**

```
<?php
    $userinfo = posix_getpwuid(10000);
    print_r($userinfo);
?>
```

**Output:**

```
Array
{
    [name]    => Amar
    [passwd]  => x
    [uid]     => 10000
    [gid]     => 42
    [gecos]   => "Amar,,,,"
    [dir]     => "/home/Amar"
    [shell]   => "/bin/bash"
}
```

## 2. posix\_getgrgid() Function:

- The posix\_getgrgid() function return info about a group by group id.  
**Syntax:** array posix\_getgrgid(int \$gid)
- The function, posix\_getgrgid( ), returns, an associative array on a group identified by a group id. It contains following elements of the group structure.

Name	Explanation
name	The name of the group.
gid	The ID number of the group.
members	The number of members belonging to the group.

**For example:**

```
<?php
    $groupid = posix_getegid();
    $groupinfo = posix_getgrgid($groupid);
    print_r($groupinfo);
?>
```

**Output:**

```
Array
{
    [name] => Pragati
    [passwd] => x
    [members] => Array
    {
        [0] => Amar
        [1] => Akbar
    }
    [gid] => 42
}
```

### 4.5.1 File Permissions

- File permissions specify what we can do with a particular file in the system such as reading, writing or executing the file.
- Notice that PHP automatically grants appropriate permissions behind the scenes. For example, if we create a new file for writing, PHP automatically grants read and write permissions to us.
- In addition, PHP also provides some useful functions for checking and changing the file permissions.

### 1. PHP checking File Permissions:

- PHP gives us three handy functions that allow us to check file permissions:
  - (i) `is_readable()` returns true if we are allowed to read the file, otherwise returns false.
  - (ii) `is_writable()` returns true if we are allowed to write the file, otherwise returns false.
  - (iii) `is_executable()` returns true if we are allowed to execute the file, otherwise returns false.

**For example:**

```
<?php
$fn = './nirali.text';
$msg = is_readable($fn) ? $msg = 'File is readable'
: $msg = 'File is not readable';
echo $msg . '<br/>';
$msg = is_writable($fn) ? $msg = 'File is writable'
: $msg = 'File is not writable';
echo $msg . '<br/>';
$msg = is_executable($fn) ? $msg = 'File is executable'
: $msg = 'File is not executable';
echo $msg . '<br/>';
?>
```

- Besides those functions, PHP also provides the `fileperms()` function that returns an integer, which represents the permissions that are set on a particular file.

**Syntax:** `fileperms($filename)`

**For example:**

```
<?php
echo fileperms("nirali.txt");
?>
```

**Output:**

33206

### 2. Changing File Permissions:

- To change the file permissions, or mode, we use `chmod()` function.
- First, we need to pass the name of the file that we want to set permission. Second, we need to specify the desired permission.

- The chmod() function returns true if the permission was set successfully otherwise it returns false.

**Syntax:** chmod(&file, &mode)

- A file permission is represented by an octal number that contains three digits:
  - The first digit specifies what the owner of the file can do with file.
  - The second digit specifies what the owner group of the file can do with the file.
  - The third digit specifies what everyone can do with the file.

**For example:**

```
<?php
    chmod("/somedir/somefile", 755); // decimal; probably incorrect
    chmod("/somedir/somefile", "u+rwx,go+rx"); // string; incorrect
    chmod("/somedir/somefile", 0755); // octal; correct value of mode
?>
```

- We can use following three functions to get information from our PHP scripts.

### 1. fileowner() Function:

- This function returns the user ID (owner) of the specified file on success or FALSE on failure.

**Syntax:** fileowner(\$filename)

**For example:**

```
<?php
    echo fileowner("nirali.txt");
?>
```

### 2. filegroup() Function:

- The filegroup() function returns the group ID of the specified file, or FALSE on failure.

**Syntax:** filegroup(\$filename)

**For example:**

```
<?php
    echo filegroup("nirali.txt");
?>
```

### 3. filetype() Function:

- The filetype() function returns the file type of a specified file or directory.
- This function returns the one of the seven possible values on success or False on failure.
- Possible return values:
  - fifo
  - char
  - dir

- o block
- o link
- o file
- o unknown

**Syntax:** filetype(\$filename):

**For example:**

```
<?php
    echo filetype("images");
?>
```

**Output:**

Dir

## 4.6

## USING PHP TO ACCESS A DATABASE

[April 16, 17, 18, 19, Oct. 16, 18]

- There are two ways to access databases from PHP.
  1. One is to use a database-specific extension;
  2. The other is to use the database-independent PEAR DB library.
- The database-specific extension is a set of functions which are used to work with the specific database using PHP.

**For example:** The MySQL extension's function names, parameters, error handling, and so on are completely different from those of the other database extensions. If we want to move the database from MySQL to PostgreSQL, it will involve significant changes to the code.

- The PEAR DB, on the other hand, does not use the database-specific functions.
- In PEAR DB library the set of functions are generalized. So moving between database systems can be as simple as changing one line of the program. [Oct. 17]
- Code that uses the PEAR DB is also a little slower than code that uses a database-specific extension.
- PEAR DB is portable and easy to use as compare to database-specific extension. The speed and feature costs are more using the database-specific extensions.
- We can write PHP program using PEAR DB or without PEAR DB to access databases. First we see how to access database without PEAR.

### 4.6.1 Database Specific Extension

- PostgreSQL database is an open source product and featured many of the object-relational concepts. PostgreSQL is a database server and can be worked with PHP.

- Steps taken by PHP to open the database and execute different queries on it.

**Step 1:** Create a connection to PostgreSQL database.

**Step 2:** Create a database (can also be done without PHP- in the PostgreSQL terminal).

**Step 3:** Create a table (can also be done without PHP- in the PostgreSQL terminal).

**Step 4:** Execute the query.

**Step 5:** Close the connection.

#### 4.6.1.1 Opening Database Connection

[April 19]

- The pg\_connect() function is used to open a PostgreSQL connection.
- Syntax:** resource pg\_connect(string \$connection\_string)
- The parameter connection\_string can contain one or more parameter setting separated by white space.
  - The parameter keywords are: host, hostaddr, port, dbname (defaults to value of user), user, password, connect\_timeout, options, and service.
  - If connection established successfully the function returns PostgreSQL connection resource, or False on failure.

**For example:**

```
$dbconn = pg_connect("dbname=test");
    // connect to a database named "test"
    // Default values are taken for remaining parameters
$dbconn2 = pg_connect("host=localhost port=5432 dbname=test");
    // connect to a database named "test" on "localhost" at port "5432"
$dbconn3 = pg_connect("host=localhost port=5432 dbname=test
                        user=ty1 passwo d=ty123");
    // connect to a database named "test" on the host "localhost"
                                with a username and password
```

#### 4.6.1.2 Closing Database Connection

[April 16]

- The pg\_close() function closes a PostgreSQL connection.

**Syntax:** bool pg\_close([resource \$connection])

- The connections are automatically closed at the end of the script.

**For example:**

```
<?php
    $dbconn = pg_connect("host=localhost port=5432 dbname=test")
    or die("Could not connect");
    echo "Connected successfully";
    pg_close($dbconn);
?>
```

### 4.6.1.3 Executing Query

- The pg\_query() function executes a query on the specified database connection.  
**Syntax:** resource pg\_query([resource \$connection], string \$query)
- First parameter is optional. When connection is not present, the default connection is used. The second parameter is the SQL query to be executed.
- The function returns a query result resource on success or false on failure.

### 4.6.1.4 pg\_fetch\_row

[Oct. 16]

- In case of SELECT query the pg\_query() function returns result set. To fetches one row of data from the result set, pg\_fetch\_row() function is used.  
**Syntax:** array pg\_fetch\_row ( resource \$result[, int \$row] )
- The first parameter \$result is the result set returned by the pg\_query() function.
- The second parameter \$row is the row number in result to fetch. Rows are numbered from 0 upwards. If omitted or NULL, the next row is fetched.
- The function return an array, indexed from 0 upwards, with each value represented as a string.

**For example:**

```
<?php
    $conn = pg_pconnect("dbname=test");
    if(!$conn)
    {
        echo "An error occurred.";
        exit;
    }
    $result = pg_query($conn, "SELECT name, email FROM student");
    if(!$result)
    {
        echo "An error occurred.";
        exit;
    }
    while($row = pg_fetch_row($result))
    {
        echo "Name: $row[0]  E-mail: $row[1]";
        echo "<br>";
    }
?>
```

### **4.6.1.5 pg\_fetch\_array()**

---

- The pg\_fetch\_array() function fetches a row as an array.

**Syntax:**

```
array pg_fetch_array(resource $result [, int $row
                      [, int $result_type = PGSQL_BOTH]])
```

- This function is same as pg\_fetch\_row(). But, it stores data in the numeric indices as well as associative indices. It stores both indices by default.
- An optional parameter \$result\_type specifies how the returned array is indexed. It takes the following values: PGSQL\_ASSOC, PGSQL\_NUM and PGSQL\_BOTH.
- Using PGSQL\_NUM, pg\_fetch\_array() will return an array with numerical indices, using PGSQL\_ASSOC it will return only associative indices while PGSQL\_BOTH, the default, will return both numerical and associative indices.

**For example:**

```
<?php
$conn = pg_pconnect("dbname=test") or die("An error occurred.");
$result = pg_query($conn, "SELECT name, email FROM student") or
die("An error occurred.");
$arr = pg_fetch_array($result, 0, PGSQL_NUM);
echo $arr[0] . " <- Row 1 Name<br>";
echo $arr[1] . " <- Row 1 E-mail<br>";
/* to pass a result_type. Successive calls to pg_fetch_array
will return the next row. */
$arr = pg_fetch_array($result, NULL, PGSQL_ASSOC);
echo $arr["name"] . " <- Row 2 Name<br>";
echo $arr["email"] . " <- Row 2 E-mail<br>";
?>
```

### **4.6.1.6 pg\_fetch\_assoc()**

---

- The pg\_fetch\_assoc() returns an associative array that corresponds to the fetched row (records).
- The pg\_fetch\_assoc() function is equivalent to pg\_fetch\_array() with PGSQL\_ASSOC as the optional third parameter. It only returns an associative array.

**Syntax:** array pg\_fetch\_assoc(resource \$result [, int \$row])

**For example:**

```
<?php
    $conn = pg_pconnect("dbname=test") or die("An error occurred.");
    $result = pg_query($conn, "SELECT roll_no, name, email FROM student") or
    die("An error occurred.");
    while ($row = pg_fetch_assoc($result))
    {
        echo $row['roll_no'];
        echo $row['name'];
        echo $row['email'];
    }
?>
```

#### **4.6.1.7 pg\_fetch\_object()**

---

- The pg\_fetch\_object() function fetches a row as an object.
- Syntax:** object pg\_fetch\_object(resource \$result [, int \$row])
- Both the parameters are same as pg\_fetch\_row().
  - The pg\_fetch\_object() returns an object with properties that correspond to the fetched row's field names.
  - It can optionally instantiate an object of a specific class and pass parameters to that class's constructor.

**For example:**

```
<?php
    $db_conn = pg_pconnect("dbname=test") or die("An error occurred.");
    $result = pg_query($db_conn, "SELECT name, email FROM student") or
    die("An error occurred.");
    while($data = pg_fetch_object($result))
    {
        echo $data->name . " ";
        echo $data->email . "<br>";
    }
    pg_free_result($result);
    pg_close($db_conn);
?>
```

---

### 4.6.1.8 pg\_fetch\_result()

- The pg\_fetch\_result() function returns the value of a particular row and column in a result set.

**Syntax:**

```
string pg_fetch_result(resource $result, int $row, mixed $column)
string pg_fetch_result(resource $result, mixed $column)
```

- The parameters \$result and \$row are same as pg\_fetch\_row(). The parameter \$column is a string representing the name of the column to fetch, otherwise an int representing the column number to fetch. Columns are numbered from 0 upwards.

**For example:**

```
<?php
$db = pg_connect("dbname=users user=me") || die();
$res = pg_query($db, "SELECT * FROM student");
$val = pg_fetch_result($res, 1, 0);
echo "Roll no of the second student is: ". $val;
?>
```

**Output:**

---

Roll no of the second student is: 102

---

### 4.6.1.9 pg\_num\_fields()

- The pg\_num\_fields() function returns the number of columns in a result set.

**Syntax:** int pg\_num\_fields(resource \$result)

**For example:**

```
<?php
$result = pg_query($conn, "SELECT * FROM student");
$num = pg_num_fields($result);
echo $num . "No of columns : $num";
?>
```

**Output:**

---

No of columns : 5

---

### 4.6.1.10 pg\_num\_rows()

- The pg\_num\_rows() function returns the number of rows in a result set.

**Syntax:** int pg\_num\_rows(resource \$result)

**For example:**

```
<?php
    $result = pg_query($conn, "SELECT 1");
    $rows = pg_num_rows($result);
    echo $rows . " row(s) returned.\n";
?>
```

**Output:**

---

1 row(s) returned.

---

#### 4.6.1.11 pg\_result\_error()

- The pg\_result\_error() function returns any error message associated with result set.  
**Syntax:** string pg\_result\_error(resource \$result)
- This function returns empty string if there is no error. If there is an error associated with the result parameter, returns false.

**For example:**

```
<?php
    $dbconn = pg_connect("dbname=test") or die("Could not connect");
    if(!pg_connection_busy($dbconn))
    {
        pg_send_query($dbconn, "select * from doesnotexist;");
    }
    $res1 = pg_get_result($dbconn);
    echo pg_result_error($res1);
?>
```

### 4.7 RELATIONAL DATABASES AND SQL

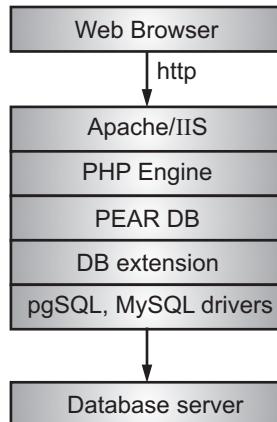
- A Relational Database Management System (RDBMS) is a server that manages data for us.
- The data is structured into tables, where each table has some number of columns, each of which has a name and a type. Tables are grouped together into databases.
- PHP communicates with relational databases such as PostgreSQL, MySQL and Oracle using the Structured Query Language (SQL). We can use SQL to create, modify, and query relational databases.

### 4.8 PEAR DB BASICS

[Oct. 16]

- PEAR stands for PHP Extension and Application Repository. PEAR is an extensive online collection of free PHP modules and classes for many different functions.
- PEAR DB acts as an abstraction layer to help us interface with our database. In the PEAR DB codes the functions are generic.

- The PEAR DB is a framework and distribution system of PHP code components.
- PEAR DB is a database abstraction layer for multiple database platforms as shown in Fig. 4.1.



**Fig. 4.1: PEAR DB System**

- PEAR DB library comes with PHP which is used to connect with the database, issue queries, check for errors etc.
- The library is object-oriented, where we use class methods (DB:: connect(), DB::iserror()) and object methods (\$db->query(), \$q->fetchInfo() ) etc.

**For example:**

```

<?php
    require_once('DB.php');
    $db = DB::connect("pgsql://typhp@localhost/test");
    if(DB::iserror($db))
        die($db->getMessage());
    $sql = "SELECT * FROM student";
    $q = $db -> query($sql);
    if(DB::iserror($q))
        die($db->getMessage());
    while($q->fetchInfo($row))
        echo $row[0]. " : " . $row[1] . "<br>";
    ?>
  
```

- In above program we connect PHP with the Postgresql database, which is specified in the string (DSN) passed as a parameter to the connect() method.
- To move from one database to another database system, just change the database type. The remaining statements are generalized statements which are common for all the databases.

### 1. Data Source Names:

- A Data Source Name (DSN) is a string that specifies the database location, type of databases, the username and password to use when connecting to the database, and more.
- The components of a DSN are assembled into a URL-like string:  
`type(dbsyntax)://username:password@protocol+hostspec/database`
- The only mandatory field is type, which specifies the type of database to use.
- The following table shows the name of databases and their type (dbsyntax).

Name	Database
Mysql	MySQL
Pgsql	PostgreSQL
Ibase	InterBase
Mssql	Microsoft SQL Server
oci8	Oracle 7/8/8i
Odbc	ODBC
Sybase	SyBase

- In DSN the 'protocol' is the communication protocol to use. The two common values are "tcp" and "unix", corresponding to Internet and Unix domain sockets.

**For example:**

`Mysql://typhp@localhost/test`

- Here, pgsql is the database type to which PHP communicates, typhp is the username, localhost is the hostspec, and test is the database.

### 2. Connecting:

[April 16, 17]

- Database connection is created using the connect() method. The DSN is passed as a parameter to this method. This returns a database object. This object is used for tasks such as issuing queries and quoting parameters:

```
$db = DB::connect(DSN [, options ]);
```

- The options value can either be Boolean, indicating whether or not the connection is to be persistent, or an array of options settings. The options values are given in Table.

Option	Controls
persistent	Connection persists between accesses.
optimize	What to optimize for.
debug	Display debugging information.

- By default, the connection is not persistent and no debugging information is displayed. Permitted values for optimize are 'performance' and 'portability'.

- The default is 'performance'. Here's how to enable debugging and optimize for portability:

```
$db = DB::connect($dsn, array('debug' => 1, 'optimize' => 'portability'));
```

### 3. Error Checking:

[April 17]

- The following method is used to check the error:

```
DB::isError()
```

- The DB::isError() method returns true if an error occurred while working with the database object or executing queries.
- If some error occurs and we want to display the error message, then we call the getMessage() method. We can call getMessage() on any PEAR DB object.

**For example:**

```
if(DB::iserror($db))
    die($db->getMessage());
```

Here, \$db is the database connection object. So we check whether any error occurs while connecting with the database. If some error occurs the isError() method returns true, hence, die() method will be called, which terminates the program after displaying the error message.

### 4. Issuing a Query:

- The query() method is used to execute query. This method sends SQL query to the database.
- The method is called using a database object:

```
$result = $db->query($sql);
```

- In case of INSERT, UPDATE, DELETE query the method returns the DB\_OK constant to indicate success. In case of SELECT query the query() method returns an object that we can use to access the results.
- We can check whether the query is executed successfully or not as follows:

```
$q = $db->query($sql);
if(DB::isError($q))
{
    die($q->getMessage());
}
```

### 5. Fetching Results from a Query:

- Consider the following statement:

```
$result = $db->query($sql);
```

- If \$sql is a SELECT query, in that case the query() method returns a result set. The result set is nothing but the table.

- To fetch single row (record) from result set PEAR DB provides two methods. One returns an array corresponding to the next row, and the other stores the row array into a variable passed as a parameter.

## 6. Returning the Row:

- The fetchRow() method returns an array of the next row of results:

```
$row = $result->fetchRow([ mode ]);
```

- This returns an array of data or NULL if there is no more data. If an error occurs this method returns DB\_ERROR. The mode parameter controls the format of the array returned.

- Commonly the fetchRow() method is used to process a result, one row at a time, as follows:

```
while ($row = $result->fetchRow())  
{  
    if(DB::isError($row))  
    {  
        die($row->getMessage());  
    }  
    // do something with the row  
}
```

## 7. Storing the Row:

- The fetchInto() method also gets the next row, but stores it into the array variable passed as a parameter:

```
$success = $result->fetchInto(array, [mode]);
```

- Like, fetchRow( ), fetchInto( ) returns NULL if there is no more data, or DB\_ERROR if an error occurs.

- To process all results the fetchInto( ) is used as follows:

```
while($success = $result->fetchInto($row))  
{  
    if(DB::isError($success))  
    {  
        die($success->getMessage());  
    }  
    // do something with the row  
}
```

- By default, the variable \$row contains a one dimensional indexed arrays, where the positions in the array correspond to the order of the columns in the returned result. For example:

```

$sql = "SELECT * FROM student";
$result = $db->query($sql);
while($row = $result->fetchRow())
    echo $row[0]. " : " . $row[1] . "<br>";

```

- We can pass a mode parameter to fetchRow( ) or fetchInto( ) to control the format of the row array. The default behaviour, shown previously, is specified with DB\_FETCHMODE\_ORDERED.
- The fetch mode DB\_FETCHMODE\_ASSOC creates an array whose keys are the column names and whose values are the values from those columns:

```

$sql = "SELECT * FROM student";
$result = $db->query($sql);
while($row = $result->fetchRow(DB_FETCHMODE_ASSOC))
    echo $row['roll_no'] . " : " . $row['name'] . "<br>";

```

- In this case, to access data, we use column name instead of indices, so \$row['roll\_no'] gives roll number of that row.
- The DB\_FETCHMODE\_OBJECT mode turns the row into an object, with a property for each column in the result row.
- To access data in the object, use the \$object->property notation:

```

$sql = "SELECT * FROM student";
$result = $db->query($sql);
while($row = $result->fetchRow(DB_FETCHMODE_OBJECT))
    echo $row->roll_no . " : " . $row->name . "<br>";

```

## 8. Finishing the result:

- A query result object typically holds all the rows returned by the query.
- This may consume a lot of memory. To return the memory consumed by the result of a query to the operating system, use the free( ) method:

```
$result->free();
```

- This is not strictly necessary, as free() is automatically called on all queries when the PHP script ends.

## 9. Disconnecting:

- To force PHP to disconnect from the database, use the disconnect() method on the database object:

```
$db->disconnect();
```

- This is not strictly necessary, however, as all database connections are disconnected when the PHP script ends.

## 4.9 ADVANCED DATABASE TECHNIQUES

- PEAR DB provides several shortcut functions for fetching result rows, as well as a unique row ID system and separate prepare/execute steps that can improve the performance of repeated queries.

### 1. Placeholders:

[April 19]

- While writing the query like INSERT or UPDATE, we can put a placeholder '?' in place of specific value.
- Then in the query() method, we can add a second parameter consisting of the array of values to insert into the SQL:

```
$result = $db->query(SQL, values);
```

- For example, this code inserts three entries into the student table:

```
$students = array(array(11, 'Akash'), array(12, 'Mohit'),
array(13, 'Kunal'));
foreach ($students as $st)
{
    $db->query('INSERT INTO student (roll_no, name) VALUES (?,?)', $st);
}
```

- There are three characters that we can use as placeholder values in an SQL query:
  - ? A string or number, which will be quoted if necessary (recommended).
  - | A string or number, which will never be quoted.
  - & A filename, the contents of which will be included in the statement (for example, for storing an image file in a BLOB field).

### 2. Prepare/Execute:

[April 16]

- When issuing the same query repeatedly, it can be more efficient to compile the query once and then execute it multiple times, using the prepare(), execute() and executeMultiple() methods.

**(i) prepare() Method:** The first step is to call prepare() method on the query. The prepare() method is used to create a generic statement.

```
$compiled = $db->prepare(SQL);
```

This method returns a compiled query object.

**(ii) execute() Method:** After preparing the statement you can execute the query.

```
$response = $db->execute(compiled, values);
```

The execute() method fills the value in place of any placeholders in the query and sends it to the RDBMS. The method returns either a query response object, or DB\_ERROR if an error occurred. For example, we could insert multiple values into the student table like this:

```

$students = array(array(11, 'Akash'), array(12, 'Mohit'),
                  array(13, 'Kunal'));
$compiled = $db->prepare('INSERT INTO student (roll_no, name)
                           VALUES (?,?)');
foreach ($students as $st)
{
    $db->execute($compiled, $st);
}

```

**(iii) executeMultiple() Method:** The executeMultiple() method is used to avoid the foreach loop in the above example. The syntax is same as execute():

```
$responses = $db->executeMultiple($compiled, $values);
```

A better way to write the above example:

```

$students = array(array(11, 'Akash'), array(12, 'Mohit'),
                  array(13, 'Kunal'));
$compiled = $db->prepare('INSERT INTO student (roll_no, name) VALUES
                           (?,?)');
$db->executeMultiple($compiled, $st);

```

### 3. Shortcuts:

- PEAR DB provides a number of methods that perform a query and fetch the results in one step: getOne(), getRow(), getCol(), getAssoc(), and getAll(). All of these methods permit placeholders.

**(i) getOne() Method:** The getOne() method runs a query and returns the first column of the first row:

```
$value = $db->getOne(SQL [, values ]);
```

**For example:**

```
$m = $db->getOne("SELECT avg(marks) FROM student");
echo "The average marks of student is $m";
```

**(ii) getRow() Method:** The getRow() method runs a query and returns the first row:

```
$row = $db->getRow(SQL [, values ]);
```

This is useful if we want to return only one row. For example:

```
$row = $db->getRow("SELECT * FROM student");
print_r($row); // prints the first row only
```

**(iii) getCol() Method:** The getCol() method runs a query and returns the data from a single column:

---

```
$col = $db->getCol(SQL [, column [, values ]]);
```

---

The column parameter can be either a number (0, the default, is the first column), or the column name. For example, this fetches the names of all the student:

```
$n = $db->getAll("SELECT name FROM student");
foreach ($n as $v)
{
    echo "$v\n";
}
```

**(iv) getAll() Method:** The getAll() method runs a query and returns all the data as an array:

```
$all = $db->getAll(SQL [, values [, fetchmode ]]);
```

For example, the following code displays the roll\_no and name of all the students.

```
$table = $db->getAll("SELECT roll_no, name FROM student");
foreach ($table as $row)
{
    echo $row[0] . " : " . $row[1] . "\n";
}
```

All the get\*() methods return DB\_ERROR when an error occurs.

#### 4. Details about a Query Response:

- Four PEAR DB methods are used to get information on a query result object i.e. numRows(), numCols(), affectedRows() and tableInfo().

**(i) numRows() and numCols() Methods:** The numRows() and numCols() methods tell you the number of rows and columns returned from a SELECT query:

```
$howmany = $response->numRows();
$howmany = $response->numCols();
```

**(ii) affectedRows() Methods:** The affectedRows() method tells you the number of rows affected by an INSERT, DELETE, or UPDATE operation:

```
$howmany = $response->affectedRows();
```

**(iii) tableInfo() Methods:** The tableInfo() method returns detailed information on the type and flags of columns returned from a SELECT operation:

```
$info = $response->tableInfo();
```

The following code displays the student table information:

```
$sql = "SELECT * FROM student";
$result = $db->query($sql);
$info = $result->tableInfo();
foreach ($info as $k => $v)
{
    echo $k . " : " . $v . "<br>";
}
```

**5. Sequences:**

- Sequence is a database object that generates numbers in sequential order.
- Some of the RDBMS supports the feature to assign unique row IDs. PEAR DB sequences are an alternative to database-specific ID assignment (for instance, MySQL's AUTO\_INCREMENT).
- The nextID() method returns the next ID for the given sequence:

```
$id = $db->nextID(sequence);
```

- A sequence is really a table in the database that keeps track of the last-assigned ID.
- We can explicitly create and destroy sequences with the createSequence() and dropSequence() methods:

```
$res = $db->createSequence(sequence);
$res = $db->dropSequence(sequence);
```

- The result will be the result object from the create or drop query, or DB\_ERROR if an error occurred.

**6. Metadata:**

- The getListOf() method is used to query the database to get information on available databases, users, views, and functions:

```
$data = $db->getListOf(what);
```

- The 'what' parameter is a string which may contain "databases", "users", "views", and "functions".
- For example, this stores a list of available databases in \$dbs:

```
$dbs = $db->getListOf("databases");
```

**7. Transactions:**

- Some RDBMSs support transactions, in which a series of database changes can be committed (all applied at once) or rolled back (discarded, with the changes not applied to the database).
- For example, when a bank handles a money transfer, the withdrawal from one account and deposit into another must happen together - neither should happen without the other, and there should be no time between the two actions.
- PEAR DB offers the commit() and rollback() methods to help with transactions:

```
$res = $db->commit();
$res = $db->rollback();
```

- If we call commit() or rollback() on a database that doesn't support transactions, the methods return DB\_ERROR.

**Sample Application (Mini Project):**

- This section shows us what we can do with a database driven website. Here, we create few web pages that manage student information.

- In the first step we create a table using PostgreSQL which stores the student information as follows:

```
student(id, name, roll_no, address, email, class)
```

- This application performs the following task:
  - Insert student information,
  - View student records,
  - Update student records, and
  - Delete student records.

- First create a student table as follows:

```
CREATE TABLE student{
    id int(11) NOT NULL auto_increment,
    name varchar(20) NOT NULL,
    roll_no varchar(6) NOT NULL,
    address varchar(50) NOT NULL,
    email varchar(20) NOT NULL,
    class varchar(20) NOT NULL,
    PRIMARY KEY(id)
};
```

### Connecting to Database:

- Create PHP code to connect to an existing database. Call it “conn.inc.php”. This page will be included in the other pages for PostgreSQL connection.

```
<?php
$host= "host=localhost";
$port = "port=5432";
$dbname = "dbname=testdb";
$credentials = "user=root password=pass123";
$conn = pg_connect("$host $port $dbname $credentials") or
die("Could not connect");
?>
```

### Insert Student Information:

- Here we create a page to insert student record. Call it “insert\_student.php”.
- If users do not fill out required fields, or insert already inserted record, the page will notify them.
- This page also contains a link to view list of records already inserted.

```
<?php
include "conn.inc.php";
?>
```

```

<html>
<head>
    <title>Untitled Document</title>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body>
    <h1>Insert Student Information</h1>
    <?php
        if(isset($_POST['submit']) && $_POST['submit'] == "Insert")
        {
            if($_POST['name'] != "" &&
                $_POST['roll_no'] != "" &&
                $_POST['address'] != "" &&
                $_POST['email'] != "" &&
                $_POST['class'] != "")
            {
                $query = "SELECT name, email FROM student WHERE
                    name='".$POST['name']."' and email='".$POST['email']."' ";
                $result = pg_query($conn, $query) or
                    die(pg_last_error($conn));
                if(pg_num_rows($result))
                {
                    ?>
                    <p>
                        <font color="#FF0000"><b>The Student,
                            <?php echo $_POST['name']; ?>, is already in the database, please
                            insert another.</b></font>
                        <form action="insert_student.php" method="post">
                            Name: <input type="text" name="name" value="<?php echo
                                $_POST['name']; ?>"><br>
                            Roll No: <input type="text" name="roll_no" value="<?php echo
                                $_POST['roll_no']; ?>"><br>
                            Address: <input type="text" name="address" value="<?php echo
                                $_POST['address']; ?>"><br>
                }
            }
        }
    ?>
    <p>
        <font color="#FF0000"><b>The Student,
            <?php echo $_POST['name']; ?>, is already in the database, please
            insert another.</b></font>
        <form action="insert_student.php" method="post">
            Name: <input type="text" name="name" value="<?php echo
                $_POST['name']; ?>"><br>
            Roll No: <input type="text" name="roll_no" value="<?php echo
                $_POST['roll_no']; ?>"><br>
            Address: <input type="text" name="address" value="<?php echo
                $_POST['address']; ?>"><br>

```

```

Email: <input type="text" name="email" value="<?php echo
$_POST['email']; ?>"><br>
Class: <input type="text" name="class" value="<?php echo
$_POST['class']; ?>"><br>
<br><br>
<input type="submit" name="submit" value="Insert"> &nbsp;
<input type="reset" value="Clear">
</form>
</p>
<?php
}
else
{
    $query = "INSERT INTO student(name, roll_no, address, email,
class)".
"VALUES ('". $_POST['name'] . "', '" .
$_POST['roll_no'] . "', '" . $_POST['address'] .
"', '" . $_POST['email'] . "', '" . $_POST['class'] . "');");
$result = pg_query($conn, $query) or
die(pg_last_error($conn));
?>
<p>
Record inserted successfully.<br>
<a href="view_student.php">View Records</a>
<?php
}
}
else
{
?>
<p>
<font color="#FF0000"><b>The Name, Roll No, Address, Email, Class
fields are required</b></font>
<form action="insert_student.php" method="post">

```

```
Name: <input type="text" name="name" value="<?php echo  
$_POST['name']; ?>"><br>  
Roll No: <input type="text" name="roll_no" value="<?php echo  
$_POST['roll_no']; ?>"><br>  
Address: <input type="text" name="address" value="<?php echo  
$_POST['address']; ?>"><br>  
Email: <input type="text" name="email" value="<?php echo  
$_POST['email']; ?>"><br>  
Class: <input type="text" name="class" value="<?php echo  
$_POST['class']; ?>"><br>  
<br><br>  
<input type="submit" name="submit" value="Insert"> &nbsp;  
<input type="reset" value="Clear">  
</form>  
</p>  
      <a href="view_student.php">View Records</a>  
<?php  
    }  
}  
else  
{  
?  
<p>  
The Username, Password, Email, First Name, Last Name fields are  
required!  
<form action="insert_student.php" method="post">  
  Name: <input type="text" name="name"><br>  
  Roll No: <input type="text" name="roll_no"><br>  
  Address: <input type="text" name="address"><br>  
  Email: <input type="text" name="email"><br>  
  Class: <input type="text" name="class"><br>  
<br><br>  
<input type="submit" name="submit" value="Insert"> &nbsp;  
<input type="reset" value="Clear">  
</form>
```

---

```

</p>
      <a href="view_student.php">View Records</a>
<?php
}
?>
</body>
</html>

```

**View Student Records:**

- Create a page “view\_student.php” to display all the records along with the links to update and delete the record as follows:

```

<?php
    include "conn.inc.php";
    $query = "SELECT * FROM student;";
    $result = pg_query($conn, $query) or die(pg_last_error($conn));
?>
<html>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body>
<h1>Student Information</h1>
    <table border = "1" cellspacing = "0">
        <tr><th>Name</th><th>Roll No</th><th>Address</th><th>Email</th><th>Class</th><th></th><th></th></tr>
        <?php
            while($row = pg_fetch_array($result, NULL, PGSQL_ASSOC))
            {
?>
                <tr>
                    <td><?php echo $row['name']; ?></td>
                    <td><?php echo $row['roll_no']; ?></td>
                    <td><?php echo $row['address']; ?></td>

```

```

<td><?php echo $row['email']; ?></td>
<td><?php echo $row['class']; ?></td>
<td><a href="update_show.php?sid=<?php
                                echo $row['id'];?>">Update</a></td>
<td><a href="delete_student.php?sid=<?php
                                echo $row['id'];?>">Delete</a></td>
</tr>
<?php
}
?>
</table>
<p>
<a href="insert_student.php">Click Here</a> to insert more students.
</p>

```

### Update Student Records:

- Create pages “update\_show.php” and “update\_student.php” to update student information, with the following code:

#### **update\_show.php**

```

<?php
    include "conn.inc.php";
    $sid = $_GET['sid'];
?>
<html>
<body>
<h1>Update Student Information</h1>
Here you can update Student record.<br><br>
<p>
<?php
    $query = "SELECT name, roll_no, address, email, class FROM
              student WHERE id = " . $sid;
    $result = pg_query($conn, $query) or die(pg_last_error($conn));
    $row = pg_fetch_array($result, NULL, PGSQL_ASSOC);
?>

```

```

<p>
<form action="update_student.php" method="post">
    Name: <?php echo $row['name']; ?><br>
    Roll No: <input type="text" name="roll_no" value="<?php echo
                $row['roll_no']; ?>"><br>
    Address: <input type="text" name="address" value="<?php echo
                $row['address']; ?>"><br>
    Email: <input type="text" name="email" value="<?php echo
                $row['email']; ?>"><br>
    Class: <input type="text" name="class" value="<?php echo
                $row['class']; ?>"><br>
    <input type="hidden" name="sid" value="<?php echo $sid; ?>">
    <br><br>
    <input type="submit" name="submit" value="Update" > &nbsp;
    <input type="button" value="Cancel" onClick="history.go(-1);">
</form>
</p>
</body>
</html>
update_student.php
<?php
    include "conn.inc.php";
    $sid = $_POST['sid'];
?>
<html>
<body>
<p>
<?php
    $query_update = "UPDATE student SET " .
        "roll_no = '" . $_POST['roll_no'] . "', " .
        "address = '" . $_POST['address'] . "', " .
        "email = '" . $_POST['email'] . "', " .
    "class = '" . $_POST['class'] . "' WHERE id = " . $sid;
    $result_update = pg_query($conn, $query_update)
                    or die(pg_last_error($conn));
    echo "The record has been updated!";
?>
</p>
<a href="view_student.php">Click Here</a> to return to view records.
</body>
</html>

```

## Delete Student Records:

- Create a page “delete\_student.php” to allow users to delete student information, using the following code:

```
<?php
    include "conn.inc.php";
    $sid = $_GET['sid'];
?>
<html>
<body>
<p>
<?php
    $query_delete = "DELETE FROM student " .
                    " WHERE id = " . $sid;
    $result_delete = pg_query($conn, $query_delete)
                    or die(pg_last_error($conn));
    echo "The record has been deleted!";
?>
</p>
<a href="view_student.php">Click Here</a> to return to view records.
</body>
</html>
```

# PRACTICE QUESTIONS

## **Q.I Multiple Choice Questions:**





## Answers

1. (a)	2. (c)	3. (d)	4. (b)	5. (c)	6. (b)	7. (d)	8. (d)	9. (b)	10. (b)
11. (d)	12. (d)	13. (b)	14. (b)	15. (a)	16. (b)	17. (c)	18. (a)	19. (d)	20. (a)
21. (d)	22. (a)	23. (c)	24. (a)	25. (b)	26. (d)	27. (d)			

## **Q.II Fill in the Blanks:**

1. The PHP \_\_\_\_\_ function is used to open a file.
  2. \_\_\_\_\_ mode is used to Opens the file for reading and writing only and Places the file pointer at the end of the file.
  3. The \_\_\_\_\_ function gives information about file by providing the filename as an argument.
  4. The \_\_\_\_\_ function writes the contents of string to a file.
  5. The \_\_\_\_\_ function returns the group ID of the specified file.
  6. The \_\_\_\_\_ function is used to open a PostgreSQL connection.
  7. The \_\_\_\_\_ function executes a query on the specified database connection.
  8. The \_\_\_\_\_ method returns true in PEAR DB,if an error occurred while working with the database object or executing queries.
  9. In PEAR DB the\_\_\_\_\_ method returns an array of the next row of results.
  10. The \_\_\_\_\_ function checks if the "end-of-file (eof)" has been reached for an open file.
  11. The set of SQL commands, used to create and modify the database structures that hold the data, is known as \_\_\_\_\_.
  12. \_\_\_\_\_ database systems such as MySQL, PostgreSQL, and Oracle are the backbone of most modern dynamic web sites.
  13. PHP communicates with relational databases such as MySQL and Oracle using the \_\_\_\_\_.
  14. We can access database such as Microsoft SQL Server, Microsoft Access, Sybase from \_\_\_\_\_ DB on Linux and Unix.
  15. The fclose() function is used to \_\_\_\_\_ an open file.
  16. \_\_\_\_\_ is a collection of related data and data is a collection of facts and figures that can be processed to produce information.
  17. The software which is used to manage database is called as \_\_\_\_\_.

18. The Structured Query Language (SQL) is used to \_\_\_\_\_ relational databases.
19. \_\_\_\_\_ is a powerful, open source object-relational database system.
20. The pg\_connect() function is used to \_\_\_\_\_ a PostgreSQL connection.
21. The \_\_\_\_\_ method returns the first row of data returned by an SQL query.
22. PEAR DB offers the commit() and rollback() methods to help with \_\_\_\_\_.
23. To change the file permissions, or mode, we use \_\_\_\_\_ function.
24. To \_\_\_\_\_ information of a user by his ID number, we can use the posix\_getpwuid() function.
25. The is\_readable() returns true if we are allowed to \_\_\_\_\_ the file, otherwise returns false.

### Answers

1. fopen()	2. a+	3. stat()	4. fwrite()	5. filegroup()
6. pg_connect()	7. pg_query()	8. DB::isError()	9. fetchRow()	10. feof()
11. DDL	12. Relational	13. SQL	14. PEAR	15. close
16. Database	17. DBMS	18. manipulate	19. PostgreSQL	20. open
21. getRow()	22. transactions	23. chmod()	24. get	25. read

### Q.III State True or False:

1. The fileopen() function opens the file and returns a file handle.
2. The stat() function gives information about file by providing the filename as an argument.
3. The fwrite() function will stop at the end of the file or when it reaches the specified length, whichever comes first.
4. The rename() function in PHP is an inbuilt function which is used to rename a file or directory.
5. PHP has an unlink() function that allows us to delete a file.
6. The fcopy() function returns true if the file was copied successfully.
7. The fgets() function reads a single line from a file.
8. The pg\_connect() function is used to open a PostgreSQL connection.
9. The pg\_query() function executes a query on the specified database connection.
10. The pg\_fetch\_assoc() function is equivalent to pg\_fetch\_array() with PGSQL\_ASSOC as the optional third parameter.
11. Close the file with fclose() function.
12. Once a file is opened using fopen() function it can be read with a function called fread().
13. The DML, is used to retrieve and modify data in an existing database.
14. A RDBMS is a database that is based on the relational model as introduced by E. F. Codd.

15. The getCol( ) method returns a single column from the data returned by an SQL query.
16. The numRows( ) and numCols( ) methods tell you the number of rows and columns returned from a SELECT query.
17. The nextID( ) method returns the next ID for the given sequence.
18. A data source name (DSN) is a string that specifies where the database is located, what kind of database it is, the username and password to use when connecting to the database, and more.
19. A new file can be written or text can be appended to an existing file using the PHP fread().
20. A file must be opened before we can read from it or write to it.
21. The fgetc() function in PHP is an inbuilt function which is used to return a single character from an open file.
22. The chmod() function in PHP is used to make a copy of a specified file. It makes a copy of the source file to the destination file.
23. A Data Source Name (DSN) is a string that specifies where the database is located, what kind of database it is, the username and password to use when connecting to the database, and more.

### Answers

1. (F)	2. (T)	3. (T)	4. (T)	5. (T)	6. (F)	7. (F)	8. (T)	9. (T)	10. (T)
11. (T)	12. (T)	13. (T)	14. (F)	15. (T)	16. (T)	17. (T)	18. (T)	19. (F)	20. (T)
21. (T)	22. (F)	23. (T)							

### Q.IV Answer the following Questions:

#### (A) Short Answer Questions:

1. What is file?
2. Define database?
3. What is directory?
4. Define DBMS.
5. What is RDBMS?
6. Which functions are used for random access to file data?
7. What is the purpose of chmod()?
8. In PHP which function is used to getting information about a file.
9. List functions for reading and writing characters in file.
10. How to read entire file in PHP?
11. Which function is used to delete files in PHP?
12. What is the purpose of pg\_query?
13. In PHP which function is used to create a connection to the database.
14. Give the use of feof().
15. 'PostgreSQL is open source relational database system'. Comment this statement.

**(B) Long Answer Questions:**

1. Define file? How to create it? Explain with example.
2. What is PEAR DB? Explain in detail.
3. Explain the purpose and syntax of the following functions:
  - (i) pg\_connect()
  - (ii) pg\_close()
  - (iii) pg\_query()
  - (iv) stat().
4. How to splitting name and path from file? Explain with example.
5. With the help of example describe how to read and write a file.
6. Write short note on: Reading entire file.
7. With the help of example explain pg\_fetch\_row () .
8. Explain advantages and disadvantages of using PEAR DB to access database.
9. What is DSN? Explain its purpose.
10. How the SQL statement is executed in PEAR DB?
11. What is sequence and metadata? Define them.
12. What is transaction? Explain the example.
13. What is file ownership and permissions? How to give file ownership and permissions in PHP? Explain with example.
14. Write short note on: Relational databases and SQL.
15. Describe advanced database techniques in detail.

**UNIVERSITY QUESTIONS AND ANSWERS****April 2016**

1. What is purpose of file()? [1 M]
- Ans.** Refer to Section 4.2, Point (1).
2. Write connect function is PEAR DB for PostgreSQL. [1 M]
- Ans.** Refer to Section 4.6.
3. Write a short note on prepare and execute functions in databases. [3 M]
- Ans.** Refer to Section 4.9, Point (2).

**October 2016**

1. What is PEAR? [1 M]
- Ans.** Refer to Section 4.6.
2. Consider tables emp(eno, ename, designation) and project (pno, pname, budget). Assume empdb with emp and project tables already exists. Write a PHP script using ProgreSQL functions to retrieve employee details for a given project. [5 M]
- Ans.** Refer to Sample Example.
3. Write a PHP script to accept a directory name and a file name from user. Check whether a given file exists in a directory or not. If a file exists in a given directory then display its inode number and size. [5 M]
- Ans.** Refer to Section Sample Example.

4. What are different methods to retrieve data from resultset using PEAR DB functions? [4 M]

**Ans.** Refer to Section 4.6.

**April 2017**

1. State the PEAR DB function to get system error message, if the database connection fails. [1 M]

**Ans.** Refer to Section 4.6.

2. How to get the user ID of the owner of the specified file? [1 M]

**Ans.** Refer to Section 4.5.

3. Assume database empdb is already exists. Write a PHP script using PostgreSQL to increment salary of the employees by 10%. Consider table emp(no, ename, salary). [5 M]

**Ans.** Refer to Sample Example.

4. Write a PHP script to display date and time of a file when it was last accessed. [5 M]

**Ans.** Refer to Section 4.4.1.

5. Explain the functions used for reading and writing characters in files. [4 M]

**Ans.** Refer to Section 4.1.3.

**October 2017**

1. What are the different class methods and object methods available in PEAR DB library. Explain. [5 M]

**Ans.** Refer to Section 4.6.

2. Consider the following relational database:

Movie(Movie\_no, Movie\_name, Year)

Actor(Actor\_no, Actor\_name, Movie\_no)

Write a PHP script which accept Movie\_name and display actors acted in same movie. [5 M]

**Ans.** Refer to Section 4.6.

3. Explain the following functions with example:

(i) fgets() (ii) flock() (iii) file() (iv) fileatime().

[4 M]

**Ans.** Refer to Sections 4.1.7, Point (2), The flock() function locks and releases a file, 4.2, Point (1), 4.4.1, Point (1).

**April 2018**

1. What is PEAR DB Library? [1 M]

**Ans.** Refer to 4.6.

2. List and explain (any three) the functions of PEAR. [5 M]

**Ans.** Refer to Section 4.6.

3. Write a PHP script to accept filename from the user and print total number of words. [5 M]

**Ans.** Refer to Section 4.1.

4. Explain the following functions with example:

(i) fputs() (ii) fseek() (iii) readfile() (iv) fileatime().

[4 M]

**Ans.** Refer to Sections 4.1.7, Point (4), 4.3, Point (1), 4.2, Point (3), 4.4.1, Point (3).

**October 2018**

1. State the advantage of using PEAR DB functions.

**[1 M]**

**Ans.** Refer to Section 4.6.

2. Consider a table student (rno, name, class). Assume database stud already exists. Write a PHP script to accept a student roll no. and display details of that student using PEAR DB functions.

**[5 M]**

**Ans.** Refer to Section 4.6.

3. Explain the following functions with example:

(i) filectime()    (ii) file()    (iii) stat()    (iv) unlink()    (v) fwrite().

**[5 M]**

**Ans.** Refer to Sections 4.4.1, Point (2), 4.2, Point (1), 4.1.2, 4.1.5, Point (2) and 4.1.3, Point (1).

4. Write a PHP script to accept a file name from user and display last access date and time of a file.

**[4 M]**

**Ans.** Refer to Section 4.4.1.

**April 2019**

1. How to delete file in PHP?

**[1 M]**

**Ans.** Refer to Section 4.1.5, Point (2).

2. What are the different placeholders used in SQL query?

**[1 M]**

**Ans.** Refer to Section 4.9, Point (1).

3. Write a PHP script to read a file DNA. TXT where file contains character A, T, C G and space. Count occurrences of each character and write it to the DNACOUNT. TXT file.

**[5 M]**

**Ans.** Refer to Section 4.1.3.

4. Write steps to create connection with PostgreSQL database and display the data.

**[5 M]**

**Ans.** Refer to Section 4.6.1.1.

5. Consider the following relational database:

Movie(Movie\_no, Movie\_name, Year)

Actor(Actor\_no, Actor\_name, Movie\_no)

Write a PHP script which accept Movie\_name and display actors acted in same movie.

**[5 M]**

**Ans.** Refer to Section 4.6.

6. Explain the following functions with example:

(i) fread()    (ii) fwrite()    (iii) fgetc()    (iv) fgets().

**[4 M]**

**Ans.** Refer to Sections 4.1.3, Point (2) and (1), 4.1.7, Points (1) and (2).

■ ■ ■

# Handling Email with PHP

## Objectives...

- To Understand Basic Concepts of Email
- To Study Protocols of Email like SMTP, POP3 etc.
- To Learn How to Send a Email through in PHP

### 5.0 INTRODUCTION

- By using PHP scripts, emails can be sent directly. Email stands for electronic mail, e-mail or email.
- A method of exchanging messages instantly from one system to another with the help of the internet is called an Email.
- The e-mail is a message that may contain text, files, images, audio, video or other attachments sent through a network (such as Internet) to a specified individual or group of individuals.
- PHP mail() is the built in PHP function that is used to send emails from PHP scripts. For sending emails PHP uses SMTP, IMAP and POP3 protocols.

### 5.1 EMAIL BACKGROUND

[Oct. 16, April 17, 18, 19]

- Email or electronic mail is the fastest method of transferring of messages (called emails or email messages) over computer networks like the Internet.
- PHP uses mail() function for creating and sending email messages. In PHP, it very easy to send email from within the PHP scripts.
- Originally email was a text-only communication medium. Then it was extended to carry multimedia content attachment called Multipurpose Internet Mail Extensions (MIME). MIME also provides for sending email with attachments.
- Email system is based on client-server architecture. The email server is a machine or group of machines which accepts, forwards, delivers and stores messages. Sender and receiver of the message need not be online at the same time.

- To send an email message in PHP, we use the mail() function.
  - On Unix servers such as Linux and Mac OS X, PHP uses the operating system's built-in Mail Transfer Agent (MTA) to send the email. (Common MTAs include sendmail, postfix etc.).
  - On non - Unix servers such as Windows, PHP talks directly to an SMTP mail server (either on the server or on another machine).
- Fig. 5.1 shows working of email system using SMTP (Simple Mail Transfer Protocol). Which contains following components:
  1. **Mail User Agent (MUA):** The MUA is the application the sender(who composes mail) uses to compose and read email, such as Eudora, Outlook, Thunderbird, etc.
  2. **Mail Delivery Agent (MDA):** A mail delivery agent or message delivery agent (MDA) is a computer software component that is responsible for the delivery of e-mail messages to a local recipient's mailbox.
  3. **Mail Transfer Agent (MTA):** A message transfer agent or mail transfer agent (MTA) or mail relay is software that transfers electronic mail messages from one computer to another, using client-server architecture.
  4. **Domain Name System (DNS):** A DNS translates the domain name to IP addresses.

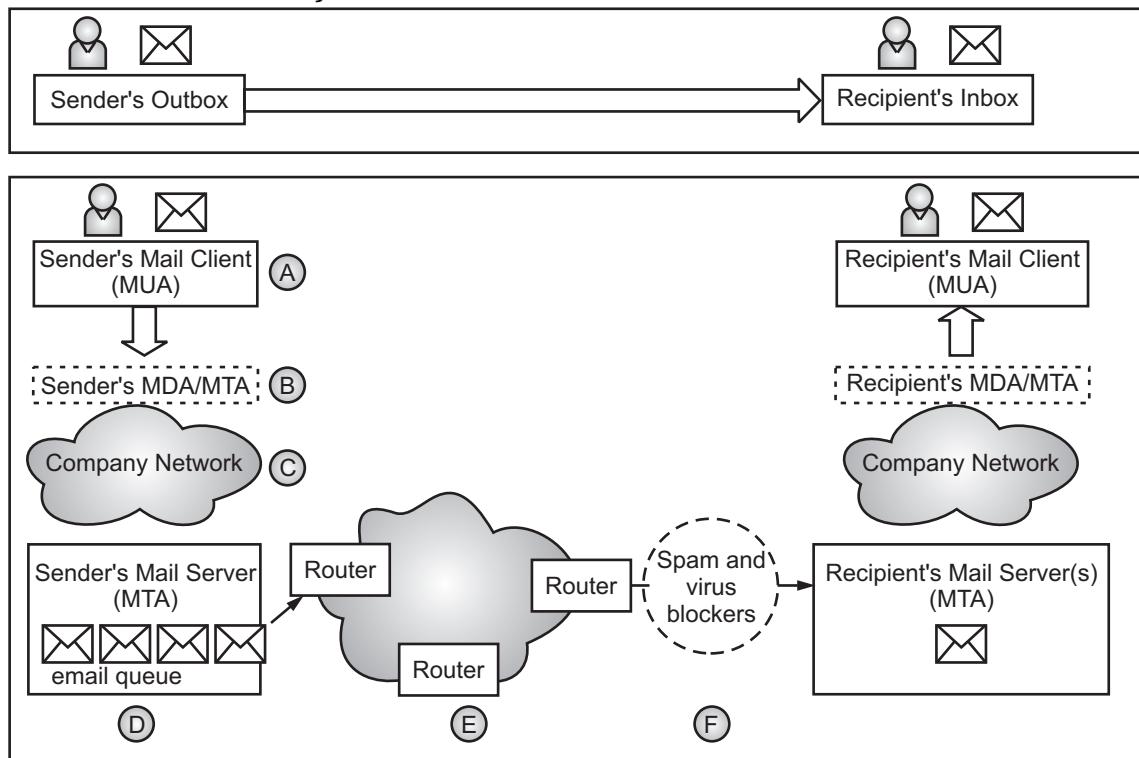


Fig. 5.1: Working of Mail System

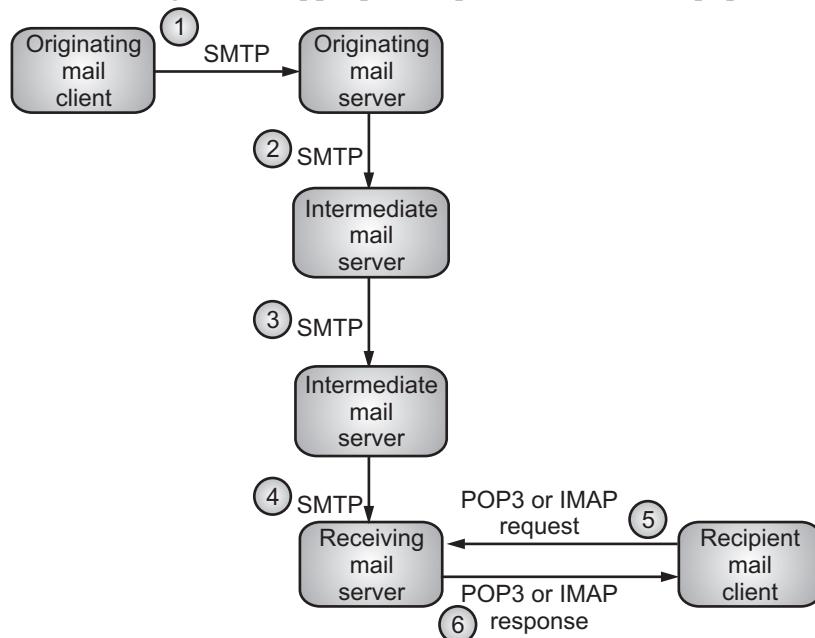
## 5.2 INTERNET MAIL PROTOCOLS

[April 16, 17, Oct. 16]

- A protocol is a set of rules that governs the communications between computers on a network.
- E-mail protocols are set of rules that help the client to properly transmit the information to or from the mail server.
- The most common email protocol for exchanging messages includes SMTP, POP, and IMAP. These are protocols are based on RFCs (Requests For Comment).

### 1. SMTP:

- To send email, SMTP (Simple Mail Transfer Protocol) is used. SMTP is reliable, connection oriented and text based protocol.
- SMTP is the standard protocol for sending emails over the Internet. Port 25 is the default SMTP non-encrypted port.
- The communication model shown in Fig. 5.2 the client uses a SMTP server.
- The server then forwards the message to the appropriate receiving SMTP server. Then it will place the message in the appropriate spot i.e. mailbox or popbox.



**Fig. 5.2: Client used a SMTP Server**

- That gets the mail from one server to the next, there are couple of protocols that are used in this case – POP3 and IMAP.
- 2. POP3 (Post Office Protocol 3):** [Oct. 18]
- POP3 protocol provides a simple, standardized way for users to access mailboxes and download messages to their computers.

- When using the POP protocol all the email messages will be downloaded from the mail server to the local computer. We can choose to leave copies of the emails on the server as well.
- POP3 generally used to support a single client and allows only one mailbox to be created on server.
- POP3 is the most recent version of a standard protocol for receiving e-mail. POP3 is a client-server protocol in which e-mail is received and held for client by client Internet server.

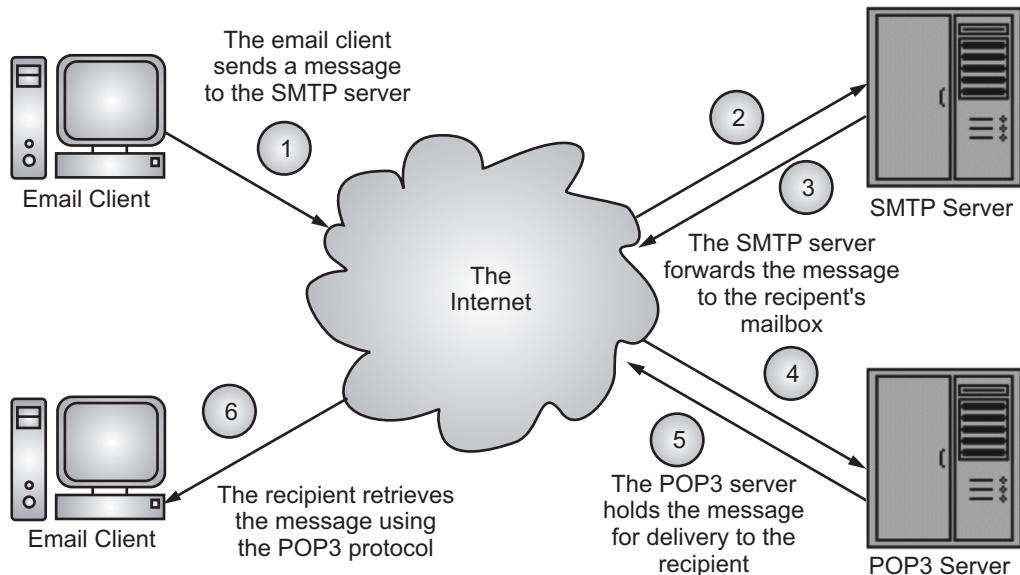
### 3. IMAP (Internet Message Access Protocol):

[April 16, 19, Oct. 17]

- IMAP is a standard protocol for accessing email from the local server. IMAP used to receive the emails from the mail server and retrieving the emails.
- IMAP allows the client program to manipulate the email message on the server without downloading them on the local computer.
- In IMAP the e-mail is hold and maintained by the remote server. IMAP enables us to take any action such as downloading, delete the mail without reading the mail.
- IMAP enables us to create, manipulate and delete remote message folders called mail boxes. IMAP also enables the users to search the e-mails.
- IMAP allows concurrent access to multiple mailboxes on multiple mail servers. The IMAP is a used by email clients to retrieve email messages from a mail server over the Internet.

### How Combination of These Protocols Works?

- E-mail is delivered over a TCP connection to port 25 of the destination mail server using SMTP. An e-mail daemon, whose job it is to monitor this port, transfers incoming messages from them into the appropriate mailbox.
- If a message cannot be delivered (usually because the named mailbox does not exist on the server) an error report is generated and sent to the incoming message's point of origin.
- The protocol used by a recipient to retrieve mail from their mailbox on the server is POP3, which allows the user to log in and retrieve messages.
- By default, messages that have been downloaded to a recipient's computer are deleted from the server.
- The IMAP is a more sophisticated mail protocol that stores all incoming and outgoing mail on the server so that mail clients with mailboxes on the server can access their e-mail from anywhere.
- Mail is not downloaded to the user's PC, and is only deleted from the client's mailbox if the client specifies that it is to be deleted.

**Fig. 5.3: Internet**

- SMTP and POP3 are the protocols most commonly employed in an exchange of email messages.

#### 4. HTTP:

[April 18, Oct. 18]

- HTTP is the foundation of the modern web. HTTP is at the heart of the Web. It is described in [RFC 1945] and [RFC 2616].
- HTTP is a connectionless, stateless protocol which provides a standardized way for computers to communicate with each other.
- HTTP defines mechanism for communication between browser or client and the web server or server by sending messages.
- HTTP is called as request and response protocol because the communication between browser and web server takes place in request and response form.
- In HTTP the client or browser (also called as HTTP client) and web server (also called as HTTP server).

### **5.3 STRUCTURE OF AN EMAIL MESSAGE**

- Email allows us to send the message in electronic mode over the Internet. Each user of email is assigned a unique name for his/her email account known as E-mail address. Different users can send and receive messages according to the e-mail address.
- Email is generally of the form `username@domainname`. For example, `webmaster@yahoo.com` is an e-mail address where `webmaster` is `username` and `tutorialspoint.com` is `domain name`.
- The `username` and the `domain name` are separated by `@` (at) symbol.

- The structure of Email message consists of Headers followed by Body of the message and may also include separate files as Attachment.
- RFC 2822 provides definition for the composition of email messages. It defines a message as being composed of ASCII characters divided into lines of characters with each line ending with CRLF (\r\n).
- The header fields are lines of characters separated by a special. The Header field contains field name, a colon, and the field body.
- The fields order is not important. The only required field is from which contains the origination date and originator address.
- Email Header play an important role in the identification of sender & receiver of the email and other additional information related to the email message.
- The Header fields are:
  1. **From:** This is required field. It contains the address of the sender.
  2. **Trace:** Includes resent-date, resent-from, resent-to etc. and is used when a message is resent.
  3. **Sender:** Contains a single address from which mail is being sent.
  4. **Reply-to:** Contains an optional reply-to address.
  5. **To:** Contains comma-separated list of addresses to which the message is sent.
  6. **Cc:** Carbon copy (Cc) comma-separated list of addresses to which copies of the message are sent.
  7. **Bcc:** Blind carbon copy (Bcc)comma-separated list of addresses to which copies of the message are sent, while preventing other recipients from seeing or knowing that any other Bcc recipient received the message.
  8. **Message-id:** Optional, but every message should have one. It contain unique message ID.
  9. **In-reply-to:** Optional, one or more message identifier.
  10. **References:** Optional, one or more message Id.
  11. **Subject:** Optional, contains a short string identifying the subject of the message.
  12. **Comments:** Optional, comment about the body of the message.
  13. **Keywords:** Optional, keywords, comma-separated that the user might find important.
  14. **Optional:** Optional, its content is unspecified.
- Email body is the field that is commonly used by the email users to communicate. Email attachments are the computer files which contain the data that are not included in main body of the email file.
- Attachments are usually used to simplify the sharing of large amount textual and non textual data across the internet.

## 5.4 SENDING EMAIL WITH PHP

[April 16, 18, 19, Oct. 16]

- To send email using PHP we must configure the php.ini file with the details of how the system sends email.
- Open the file php.ini and go to the section entitled [mail function].
- To set the SMTP setting:

```
smtp = smtp.my.server.net
```

- Set the sendmail\_from setting to reflect your email address:

```
sendmail_from = abc@example.com
```

- Linux users simply need to let PHP know the location of their sendmail application. The path and any desired switches should be specified to the sendmail\_path directive.
- The configuration for Linux should look something like this:

```
smtp =
sendmail_from =
; for unix only
sendmail_path = /usr/sbin/sendmail -t -i
```

### 5.4.1 Using the mail() Function

[Oct. 18, April 19]

- PHP uses mail() function to send an email.
- The mail() function requires three mandatory arguments that specify the recipient's email address, the subject of the message and the actual message. Additionally there are other two optional parameters.
- **Syntax:**

```
bool mail(string $to, string $subject, string $message
         [, string $additional_headers [, string $additional_parameters]])
```

- Here is the description for each parameters of above syntax:

Parameter	Description
to	Required. Specifies the receiver / receivers of the email.
subject	Required. Specifies the subject of the email. This parameter cannot contain any newline characters.
message	Required. Defines the message to be sent. Each line should be separated with a LF (\n). Lines should not exceed 70 characters.
headers	Optional. Specifies additional headers, like From, Cc, and Bcc. The additional headers should be separated with a CRLF (\r\n).
parameters	Optional. Specifies an additional parameter that can be used to pass additional flags as command line options to the program configured to be used when sending mail, as defined by the sendmail_path configuration setting.

**Return Values:**

- Returns True if the mail was successfully accepted for delivery, False otherwise.
- Following example will send an email message to xyz@somedomain.com.

```
<?php
    $to = "xyz@somedomain.com";
    $subject = "This is subject";
    $message = "This is simple text message.";
    $header = "From:abc@somedomain.com \r\n";
    if(mail ($to,$subject,$message,$header))
    {
        echo "Message sent successfully...";
    }
    else
    {
        echo "Message could not be sent...";
    }
?>
```

**Sending HTML Email:**

- When we send a text message using PHP then all the content will be treated as simple text.
- Even if we will include HTML tags in a text message, it will be displayed as simple text and HTML tags will not be formatted according to HTML syntax. But PHP provides option to send an HTML message as actual HTML message.
- While sending an email message you can specify a Mime version, content type and character set to send an HTML email.
- Following example will send an HTML email message to xyz@somedomain.com copying it to afgh@somedomain.com.
- We can code this program in such a way that it should receive all content from the user and then it should send an email.

```
<?php
    $to = "xyz@example.com";
    $subject = "This is subject";
    $message = "<b>This is HTML message.</b>";
    $message .= "<h1>This is headline.</h1>";
    $header = "From:abc@ example.com \r\n";
    $header .= "Cc:afgh@ example.com \r\n";
    $header .= "MIME-Version: 1.0\r\n";
    $header .= "Content-type: text/html\r\n";
```

```
if(mail ($to,$subject,$message,$header))
{
    echo "Message sent successfully...";
}
else
{
    echo "Message could not be sent...";
}
?>
```

---

**Example:** Sending HTML message to multiple recipients.

```
<?php
// multiple recipients
$to = 'aidan@example.com' . ', ' . 'wez@example.com';
// subject
$subject = 'Birthday Reminders for August';
// message
$message = '
<html>
<head>
    <title>Birthday Reminders for August</title>
</head>
<body>
    <p>Here are the birthdays upcoming in August!</p>
    <table>
        <tr>
            <th>Person</th><th>Day</th><th>Month</th><th>Year</th>
        </tr>
        <tr>
            <td>Joe</td><td>3rd</td><td>August</td><td>1970</td>
        </tr>
        <tr>
            <td>Sally</td><td>17th</td><td>August</td><td>1973</td>
        </tr>
    </table>
</body>
</html>
';
```

---

```

// To send HTML mail, the Content-type header must be set
$headers = 'MIME-Version: 1.0' . "\r\n";
$headers .= 'Content-type: text/html; charset=iso-8859-1' . "\r\n";
// Additional headers
$headers .= 'To: Mary <mary@example.com>, Bianca <bianca@example.com>' .
"\r\n";
$headers .= 'From: Birthday Reminder <birthday@example.com>' . "\r\n";
$headers .= 'Cc: birthdayarchive@example.com' . "\r\n";
$headers .= 'Bcc: birthdaycheck@example.com' . "\r\n";
// Mail it
if(mail ($to,$subject,$message,$header))
{
    echo "Message sent successfully...";
}
else
{
    echo "Message could not be sent...";
}
?>

```

## 5.5 VALIDATION OF EMAIL\_ID

[April 18]

- Validation is the process in which we accept data from the user and check it with some predefined standard.
- The email address must follow the RFC 2822 standard.

### 1. Validate Email Address using Filters:

```

<?php
$email = "xyz@example.com";
if(filter_var($email , FILTER_VALIDATE_EMAIL))
    echo "Email is valid";
else
    echo "Email is not valid";
?>

```

#### Output:

Email is valid

### 2. Validate Email using Regular Expression Functions:

- Before checking the email address is valid or not, remove any unnecessary characters from the email address to prevent any malicious attacks by using the htmlspecialchars(), stripslashes() and strip\_tags() functions.

- After that use the regular expression functions ereg() to verify that the email address is in proper format.

```
<?php
    $email_id = "xyz@gmail.com";
    //parse unnecessary characters to prevent exploits
    $email=htmlspecialchars(stripslashes(strip_tags($email_id)));
    //checks to make sure the email address is in a valid format
    if (ereg("^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9]+\(\.[a-z0-9]+\)*
        (\.[a-z]{2,3})$", $email))
    {
        echo "Email is valid";
    }
}
?>
```

**Example:** A PHP script to validate given email ID. Design necessary screen layouts.

```
<html>
<head>
    <title>Enter E-mail Data</title>
</head>
<body>
    <form action="test.php" method="post">
        <table>
            <tr><td>To:</td>
            <td><input type="text" name="to" size="50"></td>
            </tr>
            <tr>
                <td>From:</td>
                <td><input type="text" name="from" size="50"></td>
            </tr>
            <tr>
                <td>Subject:</td><td><input type="text" name="subject" size="50"></td>
            </tr>
            <tr>
                <td valign="top">Message:</td>
                <td>
                    <textarea cols="60" rows="10" name="message">
                        Enter your message here</textarea>
                </td>
            </tr>
            <tr><td></td>
            <td>
```

```
<input type = "Submit" value = "Send" name = "Submit">
<input type="Reset" value="Reset">
</td>
</tr>
</table>
</form>
</body>
</html>
<?php
    $to = $_POST["to"];
    $from = $_POST["from"];
    $subject = $_POST["subject"];
    $message = $_POST["message"];
    if(isset($_POST["Submit"]))
    {
        if (ereg("^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9]+(\.[a-z0-9]+)*
                (\.[a-z]{2,3})$", $to))
        {
            $headers = "From " . $from . "\r\n";
            $mailsent = mail($to, $subject, $message, $headers);
            if ($mailsent)
            {
                echo "Congrats! The following message has been sent: <br><br>";
                echo "<b>To:</b> $to<br>";
                echo "<b>From:</b> $from<br>";
                echo "<b>Subject:</b> $subject<br>";
                echo "<b>Message:</b><br>";
                echo $message;
            }
            else
            {
                echo "There was an error...";
            }
        }
        else
        {
            echo "Email address is invalid";
        }
    }
?>
```

**Output:**

To:	<input type="text" value="xyz@example.com"/>
From:	<input type="text" value="abc@yahoo.com"/>
Subject:	<input type="text" value="Hello Friends"/>
Message:	<input type="text" value="...we will be there soon!"/>
<input type="button" value="Send"/> <input type="button" value="Reset"/>	

**PRACTICE QUESTIONS****Q.I Multiple Choice Questions:**

1. Which of the following is not a component of SMTP?
 

(a) MUA	(b) MDA
(c) MTA	(d) MPA
2. Which of the option is correct in Simple Mail Transfer Protocol (SMTP)?
 

(a) reliable	(b) connection oriented
(c) text based protocol	(d) All of mentioned
3. POP3 generally used to support a single client and allows \_\_\_\_\_ to be created on server.
 

(a) one mailbox	(b) two mailboxes
(c) three mailboxes	(d) multiple mailboxes
4. IMAP allows concurrent access to \_\_\_\_\_ on multiple mail servers.
 

(a) one mailbox	(b) two mailboxes
(c) three mailboxes	(d) multiple mailboxes
5. Which PHP function is used to create a 32 digit hexadecimal number to create unique number?
 

(a) SHA12()	(b) md5
(c) md100	(d) RFC()
6. Which method of transferring of messages over computer networks like the Internet?
 

(a) email	(b) sms
(c) mms	(d) Chat (RCS)
7. Email system is based on \_\_\_\_\_ architecture.
 

(a) peer-to-peer	(b) client-server
(c) server-internet	(d) None of mentioned

8. Which function in PHP is used to send an email message?
  - (a) sendmail()
  - (b) mailsend()
  - (c) mail()
  - (d) All of mentioned
9. Which two most commonly used Internet mail protocols for retrieving emails?
  - (a) IMAP
  - (b) POP3
  - (c) Both (a) and (b)
  - (d) None of mentioned
10. Which protocol is at the heart of the Web?
  - (a) IMAP
  - (b) POP3
  - (c) HTTP
  - (d) IMAP
11. The structure of email consists of,
  - (a) Header
  - (b) Message body
  - (c) may also attachment
  - (d) All of mentioned

### Answers

1. (d)`	2. (d)	3. (a)	4. (d)	5. (b)	6. (a)	7. (b)	8. (c)	9. (c)	10. (c)
11. (d)									

### Q.II Fill in the Blanks:

1. \_\_\_\_\_ is a connectionless, stateless protocol which provides a standardized way for computers to communicate with each other.
2. PHP use \_\_\_\_\_ function to send an email.
3. \_\_\_\_\_ is a standard protocol for accessing e-mail from the local server.
4. \_\_\_\_\_ protocol provides a simple, standardized way for users to access mailboxes and download messages to their computers.
5. The email address must follow the \_\_\_\_\_ standard.
6. SMTP is the standard protocol for \_\_\_\_\_ emails across the Internet.
7. \_\_\_\_\_ allows you to download email messages on your local computer and read them even when you are offline.
8. IMAP allows the client program to manipulate the e-mail message on the server without downloading them on the \_\_\_\_\_ computer.

### Answers

1. HTTP	2. mail()	3. IMAP	4. POP	5. RFC 2822
6. sending	7. POP3	8. local		

### Q.III State True or False:

1. For sending emails PHP uses SMTP, IMAP and POP3 protocols.
2. Email system is based on client-server architecture.
3. To send an email message in PHP, we use the mail() function.
4. MDA is a computer software component that is responsible for the accepting of e-mail messages from remote recipient's mailbox.
5. IMAP used to receive the emails from theClient.
6. Sending email not requires validation and verification of email address.

7. POP3 is a standard mail protocol used to receive emails from a remote server to a local email client.
8. The email address must follow the RFC 2826 standard.

**Answers**

1. (T)	2. (T)	3. (T)	4. (F)	5. (F)	6. (F)	7. (T)	8. (F)		
--------	--------	--------	--------	--------	--------	--------	--------	--	--

**Q.IV Answer the following Questions:****(A) Short Answer Questions:**

1. What is email?
2. What is SMTP?
3. What is POP3?
4. What is HTTP?
5. What is MIME?
6. What is the function of MTA?
7. Which protocols are used to retrieve the mail from server?

**(B) Long Answer Questions:**

1. What is protocol? Which protocol is used for Internet email?
2. With the help of diagram describe working of mail system.
3. Compare IMAP and SMTP.
4. How the SMTP, POP3 and IMAP work? Explain diagrammatically.
5. Explain structure of email message.
6. Describe mail() in detail.
7. Explain header fields in email message.
8. Explain in brief how to send an email.
9. Write a PHP function to validate the email.
10. Write a complete PHP script to accept email from user and check it is valid or not.
11. ‘SMTP protocol is used to send email’. Justify True/False.
12. Explain the structure of an email message.
13. Which are the internet protocols used for mail handling? Explain any one in brief.
14. Explain how message can be send on email server?
15. Write a PHP script to accept email address and validate it. Also print domain name of the email and result of validation.

**UNIVERSITY QUESTIONS AND ANSWERS****April 2016**

1. Write the protocols used to retrieve email from server. [1 M]
- Ans. Refer to Section 5.2.
2. When IMAP4 protocol is used in email handling? [1 M]
- Ans. Refer to Section 5.2, Point (3).

**October 2016**

1. List protocols used in email. [1 M]
- Ans. Refer to Section 5.2.

2. How to send an email in PHP?

[2 M]

**Ans.** Refer to Section 5.4.

**April 2017**

1. Which protocol is used to send an email?

[1 M]

**Ans.** Refer to Section 5.2.

2. How to send an email in PHP? Explain with suitable example.

[5 M]

**Ans.** Refer to Section 5.1.

**October 2017**

1. Discuss advantages and disadvantages of IMAP protocol.

[5 M]

**Ans.** Refer to Section 5.2, Point (3).

**April 2018**

1. List the different features of HTTP.

[1 M]

**Ans.** Refer to Section 5.2, Point (4).

2. Explain how to send email with PHP.

[4 M]

**Ans.** Refer to Section 5.4.

3. Discuss how email id validation and verification is done in PHP.

[4 M]

**Ans.** Refer to Section 5.5.

**October 2018**

1. “HTTP is stateless protocol.” Comment.

[1 M]

**Ans.** Refer to Section 5.2, Point (4).

2. “POP3 Protocol is used to receive the email.” True/false justify.

[1 M]

**Ans.** Refer to Section 5.2, Point (2).

3. Write syntax of mail() function.

[1 M]

**Ans.** Refer to Section 5.4.1.

4. Explain the features of HTTP protocol.

[1 M]

**Ans.** Refer to Section 5.2, Point (4).

**April 2019**

1. Which are the two parts while message are sent using SMTP?

[1 M]

**Ans.** Refer to Section 5.1.

2. Explain how to send email through PHP with example.

[5 M]

**Ans.** Refer to Section 5.4.

3. When IMAP4 protocol is used in email handling? Explain its features.

[4 M]

**Ans.** Refer to Section 5.2, Point (3).

■ ■ ■