INTRODUCTION TO PHP

Course Name: PHP Programming

Course code: 21PCA105

UNIT: I

Prepared By : Dr.A.DEVI

* PHP - Introduction :-

→ PHP was developed by "Rasmus Lerdorf" in 1994.

↳ In 1995, he developed package called personal Home page tools, which became the first publicly distributed version of PHP.

↳ originally, PHP was an acronym for "Hypertext preprocessor"

What is PHP :-

↳ PHP is an open source, interpreted and object-oriented Server-side scripting language. It is used to develop web applications.

↳ PHP is an interpreted language, i.e there is no need for compilation.

↳ As a server-side scripting language, PHP is naturally used for form handling and database access.

↳ Database access has been a prime focus of PHP development; as a result, it has driver support for 15 different database systems.

↳ PHP is a server-side, XHTML-embedded scripting language, as such, it is an alternative to CGI (Common Gateway Interface) ASP (Active Server pages) and JSP (Java Server pages).

→ The PHP processor has two modes of operations

       ↳ Copy mode

       ↳ Interpret mode

→ The PHP processor, takes a PHP document file as input and produces an XHTML document file.

↳ When, the PHP processor finds XHTML code in the input file, it simply copies it to the output file.

↳ When, it encounters PHP script in the input file, it interprets it and sends output of the script to the output file.
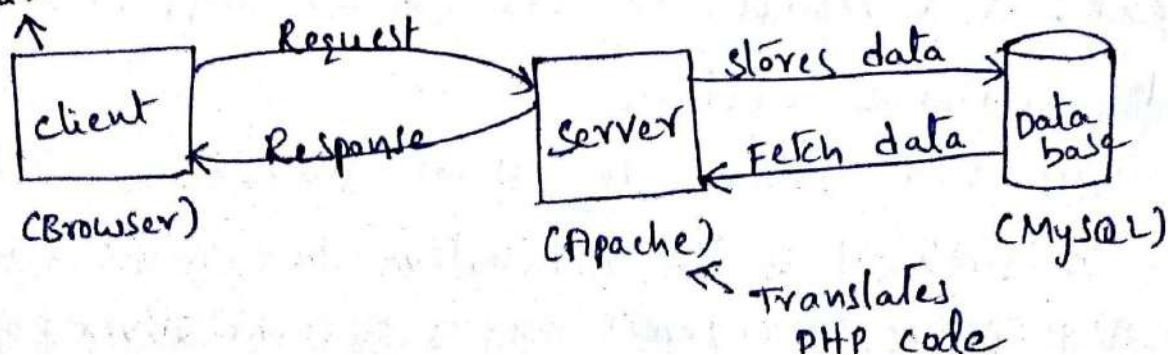
↳ The new file (output file) is sent to the requesting browser

Note:- PHP is usually purely interpreted programming language

→ PHP is can be used in popular websites like, Facebook, yahoo, wikipedia and word processor.

* <u>client - server model (Request - Response cycle)</u>

understands
simple stuff (html, JS, CSS)

CBrowser)   CApache)   CMySQL)

Translates
PHP code

→ In this, Apache can be used as server, MySQL as database and PHP as scripting language.

Installing PHP :-

⤷ To install PHP, We need to install AMP (Apache, MySQL, PHP) software pack. It is available for all operating systems. And it is an open source (free of cost).

⤷ WAMP for Windows

⤷ LAMP for Linux

⤷ MAMP for Mac

⤷ SAMP for Solaris

⤷ XAMPP (cross, Apache, MySQL, PHP, perl) for cross platforms.

→ To check, the WAMP stack is installed or not, just open browser and type "Localhost" at address bar as url, then we will get WAMP homepage (i.e installed successfully).

* PHP Example :-

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

⇒ To create PHP file, better use text editors (notepad, notepad++
- - -).

→ All PHP code write between php tag.

    syntax of php tag is

opening tag ← ——— <? php

      // code here

      ?> ——→ closing tag

→ In php, each statement ends with ; (semicolon).

* <u>echo statement</u> :-

    ↳ echo statement is used to print something on screen. Just like print function in 'c' prog language.

      <u>Ex</u>:- echo " Hello... Welcome!".

      o/p: Hello ... Welcome!

      <u>Ex</u>: echo " Hello " . 10 * 3;

      o/p' Hello 30

<u>Note</u>: In PHP, .(dot) symbol is used to concatenate two strings.

↳ to create PHP file, We need to, know basic Knowledge of html (Hypertext markup language). to generate static content where php script generates dynamic content.

    HTML tags + php script = php file.

Example :-

"First.php"

```
<html>
  <head>
    <title> First PHP Example </title>
  </head>
  <body>
  <?php
    echo "This is my first php example";
  ?>
  </body>
</html>
```

o/p:-

| @ First PHP Example — ☐ X |
| This is my first php example |

* Comments in PHP :-

Ⅼ➤ Generally, comments are used to describe (or) hide any line of code so that developer can understand the code easily.

Ⅼ➤ PHP supports Single line and Multi-line comments.

Single line ↓ //

Multi-line ↓ /* ... ... */

→ Single line comments   //

     Ex: // single line comment

→ Multi-line comments

     Ex: /* This is
         a multi-line
         comments */

* Declaring variables in PHP :-

    ↳ A variable in PHP is a name of memory location that holds data.

    (or)

    ↳ A variable is a temporary storage that is used to store data temporarily.

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

        Syntax: $variablename = value;

Rules for naming variable :-

    ↳ In php, variables must starts with letter (or) under - score only.

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

       Example:
          $a = 'hello';
          $_b = "Madhu";   } valid

          $4c = "hello";   } invalid
          $*d = 'Madhu";

→ In php, variable names are case-sensitive, so variable name "A" is different from "a". i.e $A ≠ $a.   **EX:** $A = "Madhu";

                                        $a = "kiran";
          where both ($A, $a) are different.

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

> Ex:- $str = "hello";   // string type.
>
> $x = 200;   // integer type.
>
> $y = 14.6;   // float type.

here, no need to declare (or) specify any data type.

Example :-

"vardemo.php"

```
<html>
  <head>
    <title> Variables Demo </title>
  </head>
  <body>
    <?php
        $str = " Madhu";
        $x = 112;
        $y = 13.42;
        echo " String is : $str <br/>";
        echo " Int is : $x <br/>";
        echo " float is : $y <br/>";
    ?>
  </body>
</html>
```

o/p :-

```
@ Variables Demo                    - □ X
 String is : Madhu
 Int is : 112
 float is : 13.42
```
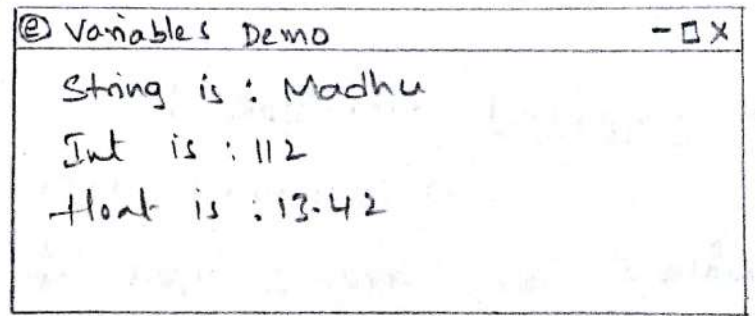
* <u>Datatypes in PHP</u> :-

       ↳ Generally, the datatypes are used to hold different Types of data (or) values.

<div align="center">(or)</div>

       ↳ Datatype specifies what type of data (or) value will be stored in variable.

       → PHP supports 8 primitive data types that can be categorized into 3 types

1. Scalar data types
2. compound data types
3. special data types

⇒ <u>Scalar data types</u> :

       ↳ scalar data type contains only a single value.

there are 4 scalar data types.

     → boolean   <u>EX</u>: $status = TRUE;

     → integer   <u>EX</u>: $n = 15;

     → float   <u>EX</u>: $f = 16.61;

     → string   <u>EX</u>: $str = "Madhu";

⇒ <u>compound data types</u> :

       ↳ compound data type contains more than one value. There are 2 types of compound data types

→ array　　　　　EX:-　$n = array (1,2,3,4,5);

→ object　　　　EX:　$s = new student();

⇒ _Special data types_ :-

　　　↳ php supports 2 special data types

　　　　　→ resource　　EX: Any file (or) database

　　　　　→ Null　　　　EX: NULL (or) null (or) $age= null;

---

*_Operators in PHP_ :-

　　　↳ An operator is a symbol i.e used to perform operations

on operands.

　　　↳ PHP supports following operators,

those are

• _Arithmetic operators_ :-

　　+　　Addition

　　−　　subtraction

　　*　　Multiplication

　　/　　Division

　　%　　Modulus

　　**　　exponentiation

• _Assignment operators_ :-

　　=　　Assignment

　　+=　　Assignment after Addition

　　−=　　Assignment after Subtraction

　　*=　　Assignment after Multiplicati

　　/=　　"　　　　　"　　Division,

　　%=　　"　　　　　"　　Modulus.

• _Increment / Decrement operators_ :-

　　++　　Increment

　　−−　　Decrement

- **Comparison (or) Relational operators :-**

==	Equal	===	Identical (if variables are of the same type & both variables have same value)
<> (or) !=	Not equal	!==	Not Identical
>	Greater than	>=	Greater than (or) equal to
<	Less than	<=	Less than or equal to

- **Logical operators :-**

and	And	&&	And
or	Or	||	Or
xor	Xor	!	Not

- **String operators :-**
  - .   Concatenation
  - .=   Concatenation assignment

- **Array operators :-**

+	Union	===	Identity
==	Equality	!==	Non-Identity
<> (or) !=	InEquality		

  ↳ We can also Categorize operators on behalf of operands.

They can be categorized in 3 forms

→ Unary operators : Works on single operands, such as
++, -- etc.

→ Binary operators : Works on two operands, such as +, -,
*, /, etc.

→ Ternary Operators : Works on three operands, such as
condition? value1 : value2

Example :-

"OperatorDemo.php"

```php
<html>
  <head>
    <title> operators Demo </title>
  </head>
  <body>
    <?php
        $a = 42;
        $b = 20;
        $c = $a + $b;
        echo " Addition is : $c <br/>";
        $c = $a - $b;
        echo " Subtraction is : $c <br/>";

        $c = $a++;
        echo " Increment is : $c <br/>";
        $c = $a--;
        echo " Decrement is : $c <br/>";

        $c += $a;
        echo " Addition assignment is : $c <br/>";

        $c -= $a;
        echo " Subtraction assignment is : $c <br/>";
    ?>
  </body>
</html>
```

o/p:-

```
┌──────────────────────────────────────────────┐
│ ① operators Demo                    - □ ×     │
├──────────────────────────────────────────────┤
│   Addition is : 62                            │
│   Subtraction is : 22                         │
│   Increment is : 42                           │
│   decrement is : 43                           │
│   Addition assignment is : 85                 │
│   Subtraction assignment is : 43              │
│                                               │
│                                               │
└──────────────────────────────────────────────┘
```

* ### control structures in php :-

> control structures (or) statements

> Conditional statements
> Loop statements
> Jump statements

• ### conditional statements :-

→ conditional statements are used to perform differe

actions based on different conditions.

→ PHP supports following conditional statements

> ### if statement :- executes some code if one condition

is true.   syntax: if (condition)
                        {
                            statements (or) code
                        }

↳ **if-else statement :-** executes some code if condition is true and another code if that condition is false.

Syntax:
```
if (condition)
{
    code (or) statements
}
else
{
    code (or) statements
}
```

↳ **if-elseif-else statement :-** executes different codes for more than two conditions.

Syntax:-
```
if (condition)
{
    code
}
elseif (condition)
{
    code
}
elseif (condition)
{
    code
}
    .
    .
else
{
    code
}
```

Example:-

"ConditionalDemo.php"

```php
<html>
  <head>
    <title> conditional Demo </title>
  </head>
  <body>
  <?php
      $x = 15;
      $y = 25;
      if ($x > $y)
      {
          echo " $x is greater than $y";
      }
      elseif ($x < $y)
      {
          echo " $x is less than $y";
      }
      else
      {
          echo " $x is equal to $y";
      }
  ?>
  </body>
</html>
```

o/p:-

© Conditional Demo — ☐ ✕
15 is less than 25

↳ <u>switch statement</u> :- used to execute one block of statements from multiple conditions.

<div align="center">

Syntax: switch (expression)
{
    case value1 : code
             break;

    case value2 : code
             break;

    - - -

    default : code
}
</div>

<u>Example</u> :-   "<u>switchDemo.php</u>"

```php
<html>
  <head>
    <title> SwitchDemo </title>
  </head>
  <body>
  <?php
    $x = 15;
    $y = 10;
    $op = ' * ';
    switch ($op)
    {
    case '+' : echo $x + $y;
              break;
    case '-' : echo $x - $y;
              break;
```
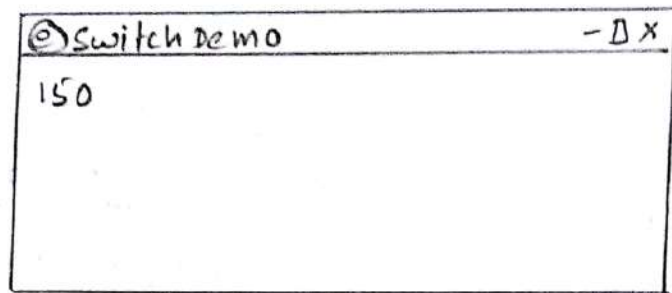
```
case '*': echo $x * $y;
            break;
case '/': echo $x / $y;
            break;
case '%.': echo $x % $y;
            break;
default: echo "Invalid operator!";
?>
</body>
</html>
```

o/p:-

```
┌─────────────────────────────────────┐
│ ☺ Switch Demo              - □ X    │
├─────────────────────────────────────┤
│ 150                                 │
│                                     │
│                                     │
│                                     │
│                                     │
└─────────────────────────────────────┘
```

• **Loop Statements :-**

→ Loop statements can be used to execute set of code for the specified number of times.

→ PHP supports following loop statements

⤷ **While Loop :-** executes a block of code as long as the specified condition is true.

Syntax:- while (condition)
         {
             code (or) statements
         }

Note:- While should be used if number of iteration is not known.

Example :- "while Demo.php"
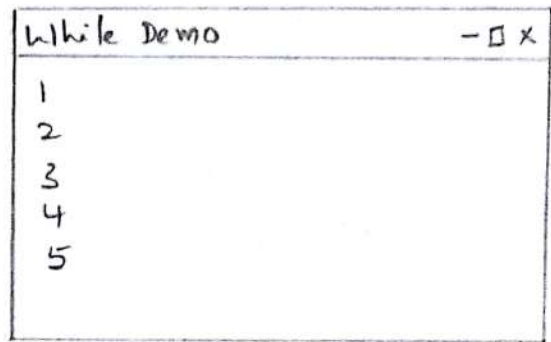
```
<hlml>
  <head>
    <title> While Demo </title>
  </head>
  <body>
    <?php
      $n=1;
      while ($n<=5)
      {
        echo "$n <br/>";
        $n++;
      }
    ?>
  </body>
</html>
```

o/p:-

While Demo   — □ ×
1
2
3
4
5

↳ Do..While Loop :- It will always execute the block of code once, it will then check condition, and repeat the loop while the Specifiled condition is True.
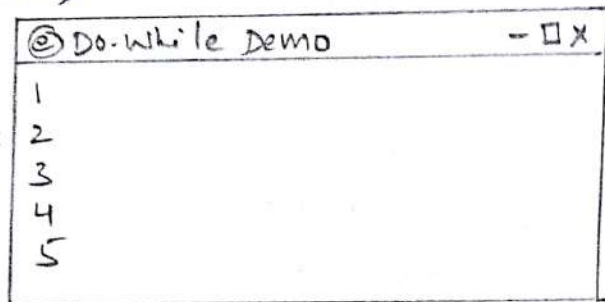
Syntax:-
```
do
{
  code (or) statements
} while (condition);
```

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

Example:- "DowhileDemo.php"

```php
<html>
  <head>
    <title> Do-while Demo </title>
  </head>
  <body>
    <?php
      $n=1;
      do
      {
        echo "$n <br/>";
        $n++;
      } while ($n<=5);
    ?>
  </body>
</html>
```

```
⊚ Do-while Demo                    - □ X
  1
  2
  3
  4
  5
```

↳ FOR LOOP :- Executes a block of code for the specified number of times.

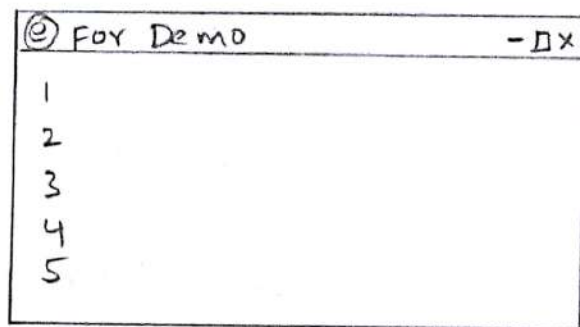Syntax:- for (initialization; condition; increment/decrement)
{
    code (or) statements
}

Note:- FOR loop should be used if number of iteration is known otherwise use while loop.

Example :-   "ForDemo.php"

```php
<html>
  <head>
    <title> For Demo </title>
  </head>
  <body>
    <?php
    for ($n=1; $n<=5; $n++)
    {
      echo " $n <br/>";
    }
    ?>
  </body>
</html>
```

```
⊙ For Demo                          - □ ×
  1
  2
  3
  4
  5
```

- **Jump statements :-**

→ Jump statements are used to alter the normal control flow of loop statements.

→ PHP supports following jump statements

↳ **break:-** When a break statement is encountered inside a loop, the loop is terminated and program control resumes at the next statement following the loop.
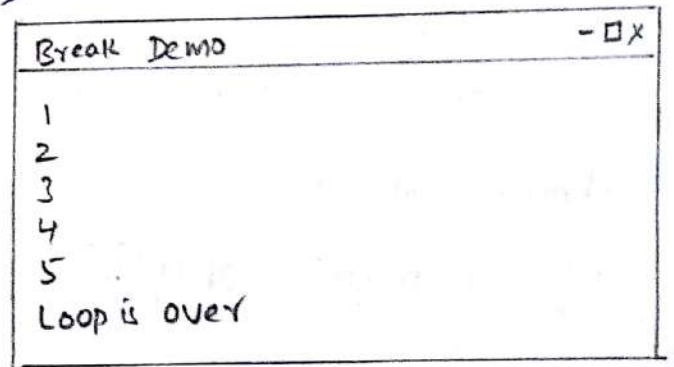
      Syntax:- break;

→ In simple words, the break is used to breaks (stops) a loop execution.

Example :-

"BreakDemo.php"

```
<html>
  <head>
    <title> Break Demo </title>
  </head>
  <body>
    <?php
      for( $n=1; $n<=10; $n++)
      {
        if ( $n == 6)
        {
          break; //terminates loop if $n is 6
        }
        echo "$n <br/>";
      }
      echo "Loop is over";
    ?>
  </body>
</html>
```

```
Break Demo                                    - □ X

1
2
3
4
5
Loop is over
```

↳ **continue :-** When a continue statement is encountered inside the loop, remaining statements are skipped and loop proceeds with the next iteration.

Syntax :- continue;
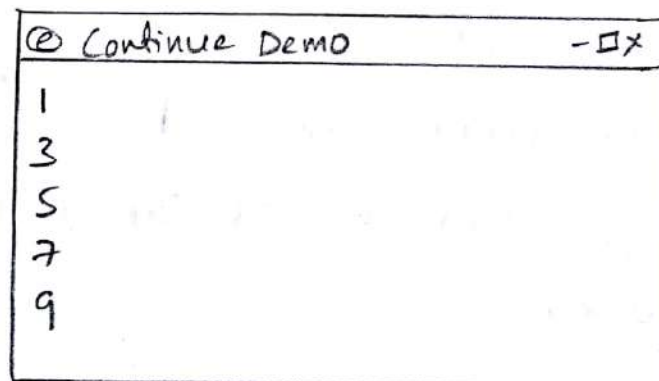
→ In simple words, The continue is used to skip the

particular iteration and jumps to the next iteration of a particular loop.

Example :-

"continueDemo.php"

```php
<html>
  <head>
    <title> Continue Demo </title>
  </head>
  <body>
    <?php
      for($n=1; $n<=10; $n++)
      {
        if(i%2 == 0)
        {
          continue; //skip next stmt if $n is even.
        }
        echo "$n <br/>";
      }
    ?>
  </body>
</html>
```

```
@ Continue Demo                      - □ ✗
1
3
5
7
9
```

* Expressions:-

    ↳ Expressions are the most important building blocks of PHP. In PHP, almost anything you write is an expression.

    ↳ The simplest way to define an expression is " anything that has a value".

    ↳ In other words, an expression is madeup of with variables and operators, that evalutes to a single value.

        Ex:- $a = 15;

                        $b = 20;

                        $c = $a + $b;

* Arrays:-

    ↳ An array is a special variable, which can hold more than one value at a time.

                (or)

    ↳ An array can hold many values under single name, and we can access the values by referring to an index number.

    → In php, The array() function is used to create an array.   array();

→ There are 3 Types of arrays
  ↳ Indexed Array
  ↳ Associative Array
  ↳ Multidimensional Array.

⇒ **Indexed Array :-**
  → In php, index is represented by number.
which is starts from 0.
  → In this array, all elements are assigned
to an index number by default.
  → There are two ways to define indexed array:
  1st - Way :
      $marks = array (60, 72, 66);

  2nd - Way :
      $marks [0] = 60;
      $marks [1] = 72;
      $marks [2] = 66;

Example :-

              "IArray.php"
  <html>
    <head>
      <title> Indexed Array </title>
    </head>

```
<body>
<?php
                                                         $marks[0] = 60;
                                                         $marks[1] = 72;
    $marks = array(60, 72, 66, 75);       (or)    $marks[2] = 66;
                                                         $marks[3] = 75;
    echo "Marks are : $marks[0], $marks[1], $marks[
                     and $marks[3]";
?>
<lbody>
<lhtml>
```

O/p:-

@ Indexed Array   — ☐ X

Marks are : 60, 72, 66 and 75

→ Associative Array :-

    → The associative arrays are very similar to Indexed
arrays in term of functionality but they are different in
terms of their index.

    → Associative array will have their index as string
So that we can establish a strong association between
key and value.

    → In php, we can associate name with each
array element using '⇒' symbol.

    → There are two ways to define associative
array.

1st way :

```php
$marks = array("Madhu" => 60, "Kiran" => 72,
                "Giri" => 66, "Kalam" => 75);
```

2nd way :

```php
$marks ["Madhu"] = 60;
$marks ["Kiran"] = 72;
$marks ["Giri"] = 66;
$marks ["Kalam"] = 75;
```

Example:-

"AArray.php"

```php
<html>
  <head>
    <title> Associative Array </title>
  </head>
  <body>
    <?php
      $marks = array("Madhu" => 60, "Kiran" => 72,
                      "Giri" => 66, "Kalam" => 75);

      echo "Marks of Madhu :". $marks ["Madhu"]."<br/>";
      echo "Marks of Kiran :". $marks ["Kiran"]."<br/>";
      echo "Marks of Giri :". $marks ["Giri"]."<br/>";
      echo "Marks of Kalam :". $marks ["Kalam"]."<br/>";
    ?>
  </body>
</html>
```

o/p:-

```
@ Associative  Array                        - □ X

  Marks  of  Modhu :  60

  Marks  of  kiran :  72

  Marks  of  Giri  :  66

  Marks  of  kalam  : 75
```

⇒ Multi dimensional Array :-

→ In php, Multidimensional array is also known as array of arrays. It allows you to store tabular data in an array.

→ Multi dimensional array can be represented in The form of Matrix which represented by rows and columns.

Def ⁿ:

$ students = array (

array (501, "Hari", 72),

array (502, "Madhu", 62),

array (503, "Naveen", 82)

);

→ A Multidimensional array is an array containing one or more arrays. php understands multidimensional arrays that are two, three, four, five (or) more levels deep.

Example :-

```php
<html>
    <head>
        <title> Multi-Dimensional Array </title>
    </head>
    <body>
    <?php
        $students = array(
                        array(501, "Hari", 72),
                        array(521, "Madhu", 65);
                        array(536, "Naveen", 82)));

        for($i=0; $i<3; $i++)
        {
        for($j=0; $j<3; $j++)
        {
            echo $students[$i][$j]."   ";
        }
            echo "<br/>";
        }
    ?>
    </body>
</html>
```

"MDArray.php"

o/p:-

```
© Multi-Dimensional Array          — ☐ ✗

501   Hari   72

521   Madhu  65

536   Naveen 82
```

* Strings :-

↳ A string is a sequence of characters i.e used to store and manipulate text.

↳ There are 2 ways to specify string in PHP.

→ Single quotes Ex:- $str = 'Hello warld';

→ Double quotes Ex:- $str = "Hello warld";

→ Where in single quoted string, we can store multi-line text, special characters and escape sequences.

→ Where in Double quoted string, we can't able use special characters directly.

Example :- $str = ' php stands for "Hypertext preprocessor"';

echo $str; o/p: php stands for "Hypertext pre processor"

$str = " php stands for "Hypertext preprocessor"";

echo -$str; o/p: parse error, syntax error.

⇒ String functions in php :-

↳ PHP provides various string functions to access and manipulate strings.

↳ A list of important string functions are

1. **strtolower():-**

   → It returns string in lowercase letter.

   Syntax:- strtolower (string $str)

   Example:
   ```php
   <?php
       $str = "My name is MADHU";

       $str = strtolower ($str);
       echo $str;      o|p:- my name is madhu
   ?>
   ```

2. **strtoupper ():-**

   → It returns string in upper case letter.

   Syntax:- strtoupper (string $str)

   Example:
   ```php
   <?php
       $str = "madhu";
       $str = strtoupper ($str);
       echo $str;      o|p:- MADHU
   ?>
   ```

3. **ucwords():-**

   → It returns string converting first character of

   each word into uppercase

   Syntax:- ucwords (string $str)

   Example:-
   ```php
   <?php
       $str = "my name is madhu";

       $str = ucwords ($str);
       echo $str;      o|p:- My Name Is Madhu
   ```

4. strlen():-
→ It returns length of the string.

Syntax:- strlen(string $str)

Example:-
```php
<?php
    $str = "Madhu T";
    $len = strlen($str);
    echo $len;    o/p:- 7
?>
```

5. strrev():-
→ It returns reversed string.

Syntax:- strrev(string $str)

Example:-
```php
<?php
    $str = "Madhu T";
    $str = strrev($str);
    echo $str;
?>        o/p:- T uhdam
```

6. str_word_count():-
→ It counts the number of words in a string.

Syntax:- str_word_count(string $str)

Example:-
```php
<?php
    $str = "Hello World!";
    $wc = str_word_count($str);
    echo $wc        o/p:- 2
?>
```

7. strpos() :-

→ It searches for a specific text within a string.
If a match is found, the function returns the character position of the first match. If no match is found, it will return False.

Syntax :- strpos ($str, $text)

Example :-
```
<?php
    echo strpos ("Hello world!", "World");
?>
```
O/p: 6.

8. str_replace() :-

→ It replaces some characters with some other characters in a string.

Syntax :- str_replace ($oText, $Ntext, $str)

Example :-
```
<?php
    echo str_replace ("world", "Madhu",
                        "Hello world!");
?>
```
O/p :- Hello Madhu!

9. substr() :-

→ It returns a sub part of a string.

Syntax :- substr ($string, $start, $length)

Example :-
```
<?php
    echo substr ("Hello world", 6);
    echo substr ("Hello world", 1, 4);
?>
```
O/p :- world
        ello

Example :-

"String Fun Demo. php"

```php
<html>
    <head>
        <title> String - Functions </title>
    </head>
    <body>
    <?php
        $str = "My name is MADHU";
        echo strtolower ($str); echo "<br/>";
        echo str to upper ($str); echo "<br/>";
        echo ucwords ($str); echo "<br/>";
        echo strlen ($str); echo "<br/>";
        echo strrev ($str); echo "<br/>";
        echo str_word_count ($str); echo "<br/>";
        echo strpos ($str, "MADHU"); echo "<br/>";
        echo str_replace ("MADHU", "Kiran", $str); echo "<br/>";
        echo substr ($str, 1, 4);
    ?>
    </body>
</html>
```

o/p

```
© String - Functions                    - □ ×
My name is madhu
MY NAME IS MADHU
My Name Is MADHU
16
UHDAM SI eman yM
4
11
My name is kiran
y na
```

* **Functions :-**

→ A function is a piece of code that is used to perform a particular task.

→ PHP supports both built-in and user-defined functions.

→ The main advantage of functions is that code-resusability. (write once invoke Multiple).

→ PHP supports thousands of built-in functions.

→ And, php allows the user to define own functions, by using "function" keyword.

syntax :-
```
function functionname()
{
    //code
}
```

Note :- Function name must be start with letter and under score only.

Example :-   "SimpleFun.php"

```
<html>
<head>
    <title> A simple function </title>
</head>
<body>
<?php
    function msg() { //defining function
    } echo "welcome functions";
```

```
        msg();  // calling function.
    ?>
    </body>
    </html>
```


A Simple Function — □X
Welcome functions
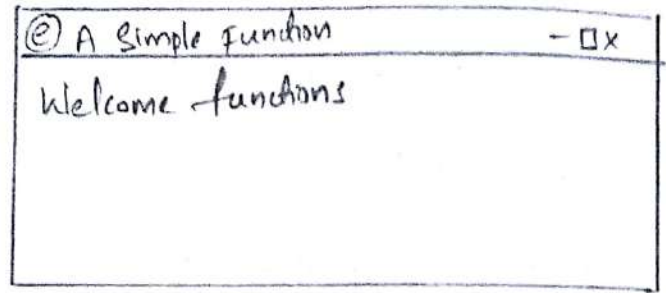
→ parameterized functions are functions with parameters. you can pass any number of parameters inside a function.

→ We can pass the information in function through arguments which are separated by comma.

→ These passed parameters (or) arguments acts as variables inside your function.

Example :-   "paramfun.php"

```
    <html>
    <head>
        <title> parameterized functions </title>
    </head>
    <body>
    <?php
        function add ( $x, $y) // defining function
        {
            $sum = $x + $y;
            echo " sum of two numbers is : $sum <br/>";
        }
        add (467, 123); // calling function
```

```php
function sub($a, $y) // defining function
{
    $diff = $a - $y;
    echo "diffence of two numbers is : $diff";
}
sub(467, 123); // calling function
?>
</body>
</html>
```

o/p:-

→ PHP allows you to call function by value and reference.

* <u>call by value</u> :

In case of call by value, actual value is not modified if it is modified inside the function.

"CallbyvalueFun.php"

Example:-
```php
<html>
<head><title> Call-By-Value </title></head>
<body>
<?php
function increment($i)
{
    $i++;
}
$i = 10;
increment($i);
echo $i;
</body>
</html> ?>
```

o/p:

Scanned by CamScanner

* **call by reference :**

→ In case of call by reference, actual value is modified, if it is modified inside the function.

→ In such case, you need to use & (ampersand) symbol with formal arguments.

→ The & represents reference of the variable.

Example:—

"callbyreference Fun.php"

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

o/p:-

```
ⓔ Call-By - Reference       - □X
This is Call By Reference
```

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

Example :-
```html
<html>
  <head>
    <title> Default -argument Function </title>
  </head>
  <body>
    <?php
    function msg ( $name ="Madhu")
    {
      echo " Hello $name <br/>";
    }
    msg("Kiran");
    msg();
    msg("Srinu");
    ?>
  </body>
</html>
```

O/P:

```
Default-argument Functin        -☐x
Hello Kiran
Hello Madhu
Hello Srinu
```

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

O/P: Addition is : 20
     Addition is : 30
     Addition is : 60

* Recursive Function :-

→ PHP also supports recursive function. In such case, we call current function within function. It is also known as recursion.

"Recursive Fun. php"

Example :-

```html
<html>
<head>
<title> Recursive Function </title>
</head>
<body>
<?php
function factorial ($n)
{
    if ($n < 0)
        return -1;
    if ($n == 0)
        return 1;
    return ($n * factorial ($n-1));
}
echo factorial (5);
?>
</body>
</html>
```

o/e

© Recursive Function	- □ X
120	
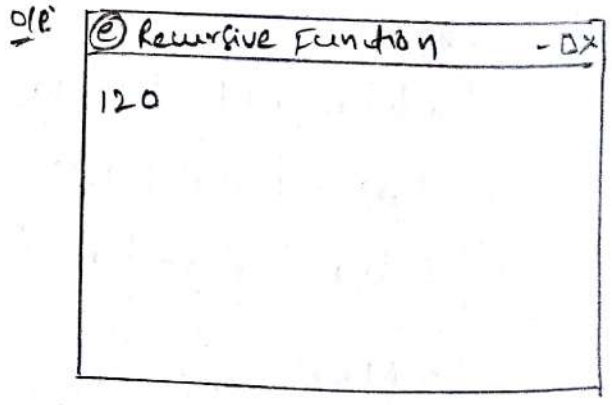
* **Reading data from web form controls :-**

→ To read data from a form, we need to use superglobal variables.

→ A superglobal variable is a built in php variable that is available in any scope.

→ PHP supports two superglobal variables, those are

    ↳ $_GET - contains list of all field names and values sent by a form using the get method.

    ↳ $_POST - contains list of all field names and values sent by a form using the post method.

⇒ **Get Form :-**

→ Get request is the default form request. The data passed through get request is visible on the URL browser so it is not secured.

→ We can able to send limited amount of data through get request.

Example :-

                              "GetForm.html"

```
<html>
 <head>
   <title> Get Form </title>
 </head>
   <body>
```

```
<form action = "Getform.php" method = "get">
Name : <input type = "text" name = "name" /> <br/> <br/>
AGe : <input type = "text" name = "age" /> <br/> <br/>
<input type = "submit" value = "click me" />
<lform >
</body>
</html>
```
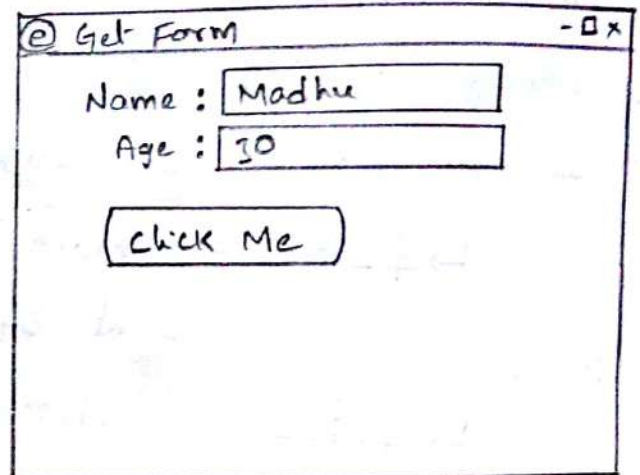
"GetForm.php"

```
<html>
<head>
<title> Welcome </title>
</head>
<body>
<?php

$name = $_GET ["name"];

$age = $_GET ["age"];

echo " welcome, $name your age is : $age";

?>
</body>
</html>
```
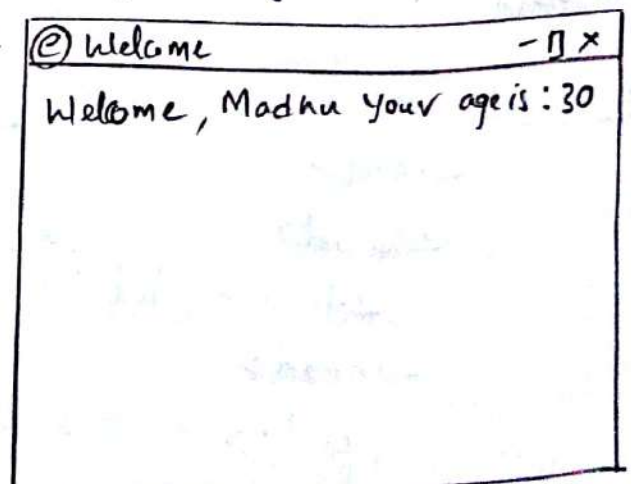
o/p

```
┌─────────────────────────────────────┐
│ @ Get Form                    - □ x  │
│                                      │
│    Name : │Madhu            │        │
│    Age :  │30               │        │
│                                      │
│      ( click Me )                    │
│                                      │
│                                      │
└─────────────────────────────────────┘
```

o/p

```
┌─────────────────────────────────────┐
│ @ Welcome                     - □ x  │
│                                      │
│  Welcome, Madhu Your age is : 30     │
│                                      │
│                                      │
│                                      │
│                                      │
└─────────────────────────────────────┘
```

⟹ post Form :-

→ post request is widely used to submit form that have large amount of data such as file upload, login form, registration form etc.

→ The data passed through post request is not visible on the URL browser so it is secured.

Example :-   "post Form. html"

```html
<html>
 <head>
  <title> post form < Hitle>
 </head>
  <body>
  <form action= "postForm. php' method ="post'>
   <table >
   <tr> <td> User Name : </td>
        <td> < input type ="text." name = "uname" /> </td>
   </tr>
   <tr> <td> Password : </td>
        <td> <input type ='password" name = "pwd" /> </td>
   </tr>
   <tr> <td> </td>
        <td> <input type =" submit' value =" login"/> </td>
   </tr>
   </table>
  </form>
 </body>
</html>
```

"Post Form.php"

```html
<html>
  <head>
    <title> Welcome </title>
  </head>
  <body>
    <?php
      $name = $_POST["uname"];
      $pwd = $_POST["pwd"];

      echo "Welcome : $name, your password is: $pwd";
    ?>
  </body>
</html>
```

o/p:

postForm. html

```
┌─────────────────────────────────────┐
│ ⓔ Post Form              - □ X │
├─────────────────────────────────────┤
│                                     │
│ userName : [ Madhu52ht        ]     │
│                                     │
│ password : [ ● ● ● ● ● ●      ]     │
│                                     │
│        ( Login )                    │
│                                     │
│                                     │
│                                     │
│                                     │
│                                     │
└─────────────────────────────────────┘
```

post Form. php

```
┌─────────────────────────────────────┐
│ ⓔ Welcome               - □ X │
├─────────────────────────────────────┤
│ Welcome : madhu52ht, your password is│
│                          :123456     │
│                                     │
│                                     │
│                                     │
│                                     │
│                                     │
│                                     │
│                                     │
│                                     │
└─────────────────────────────────────┘
```

<u>Note:-</u> post method uses http protocol, to send data. This provide

Secured way to send the data.

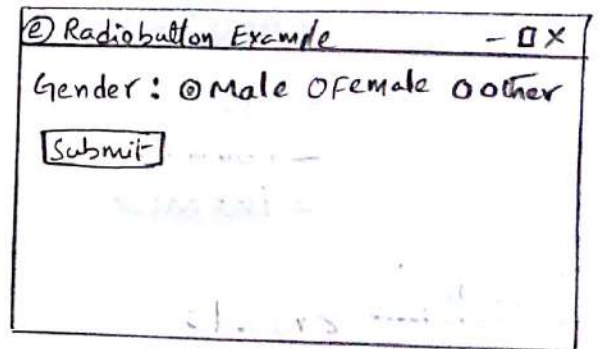→ Don't use Get method if form has password or other sensitive
   information sent to the server.

- Reading data from Radio button :-

Example :-   "Radiogen.html"

```html
<html>
<head>
  <title> Radiobutton Example </title>
</head>
<body>
<form action ="radiogen.php' method ="post'>

<b> Gender : </b>

<input type ="radio" name ="gen" value ="Male"> Male </input>
<input type ="radio" name ="gen" value ="FeMale"> Female </input>
<input type ="radio" name ="gen" value ="Other"> Other </input>

<br/> <br/>
<input type ="Submit" value ="Submit"/>
```
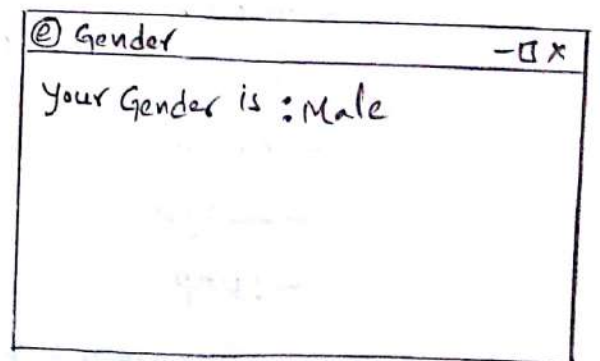
o/p:

```
┌──────────────────────────────────────────┐
│ ⓔ Radiobutton Example          - □ X │
├──────────────────────────────────────────┤
│ Gender: ⦿ Male OFemale Oother            │
│ [Submit]                                  │
│                                           │
│                                           │
│                                           │
└──────────────────────────────────────────┘
```

```html
</form>
</body>
</html>
```

"radiogen.php"

```php
<html>
<head>
  <title> Gender </title>
</head>
<body>
<?php
   $gen = $_POST['gen'];

   echo "your Gender is : $gen";
?>
</body>
</html>
```

```
┌──────────────────────────────────────────┐
│ ⓔ Gender                        - □ X │
├──────────────────────────────────────────┤
│ Your Gender is : Male                     │
│                                           │
│                                           │
│                                           │
│                                           │
└──────────────────────────────────────────┘
```

• Reading data from select list :-

Example:

```
<html>
   <head>                                    " ListDemo.html"
   <title> List Example <title>
   </head>
   <body>
   <form  action =" list.php"  method ="post">
   <b> Branch :</b>  
   <select  name = "branch">
   <option value ="CSE"> CSE </option>
   <option value ="ECE"> ECE </option>
   <option value = "EEE"> EEE </option>
   </select>
        <input type="Submit" value =" Submit"/>
   </form>
   </body>
   </html>
```
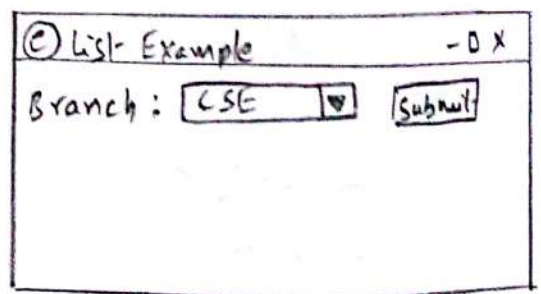
o/p:

```
+-----------------------------------------+
| © List Example              - □ X       |
| Branch : [CSE    ] [▼]  [Submit]        |
|                                         |
|                                         |
+-----------------------------------------+
```

"list.php"

```
<html>
   <head>
   <title> List </title>
   </head>
   <body>
   <?php
      $branch = $_POST ['branch'];
      echo "your branch is : $branch";
   ?>
   </body>
   </html>
```

```
+-----------------------------------------+
| © List                      - □ X       |
| your Branch is : ECE                     |
|                                         |
|                                         |
+-----------------------------------------+
```