UNIT 1.1- BUILDING BLOCKS OF PHP

1.WRITE ABOUT PHP?

★ PHP stands for HYPERTEXT PRE PROCESSOR, introduced by RASMUS LERDORF in the year 1994.
★ It includes various versions like php 3,4,5,7 and latest version of php is php8.
★ It is an open source, interpreted and object oriented scripting language.
★ Php is a server side scripting language that is embedded in html. Php files contain text, html , JavaScript code and php code.
★ Php code is executed on the server and the result is returned to the browser as plain html. The extension of php file is .php
★ It is used to manage dynamic content, data bases, build ecommerce websites.

WHAT CAN PHP DO:

- ✓ It is used for creating dynamic content and dynamic web pages.
- ✓ It is used for encrypt the data, add, delete and modify the data in data bases, control the user access.
- ✓ Access cookies variables and set cookies.
- ✓ Using php, you can restrict to access some pages of your web site.
- ✓ Php can collect form data.

CHARACTERISTICS OF PHP:

- ♦ Simplicity
- ♦ Efficiency
- ♦ Flexibility, familiarity and security.

2. HOW TO INSTALL PHP?

★ Install php and mysql in your computer. Download php official resource https://www.php.net
★ Click on the windows downloads button.
★ Click on safe version, zip button and download it.
★ Now check for the zip file in downloads in your system and extract it.
★ Now copy the extracted folder in your windows drive in the program files folder and click continue for permission.
★ Edit environment system variables and requirements.
★ Choose the path and click new button, paste the address and click ok button.
★ Now your php is installed on your computer.

3.EXPLAIN ABOUT FEATURES OF PHP?

Php stands for hypertext pre processor. It is an interpreted, open source and server side scripting language.
It is used to create dynamic web pages, development of web applications.

FEATURES OF PHP:

- ✓ **SERVER SIDE SCRIPTING:** PHP code runs on the server and generates dynamic content.
- ✓ **DYNAMIC CONTENT GENRATION:** php generates dynamic web pages based on user input and data base queries.
- ✓ **DATABASE INTEGRATION:** php handles large amount of data and wide range of data bases
- ✓ it supports MYSQL, SQLITE.
- ✓ **LOOSE TYPING:** php is a dynamically typed language, which means that variable types are determined at run time.
- ✓ **OBJECT ORIENTED PROGRAMMING:** php supports oop concepts such as class, object, inheritance, polymorphism etc.
- ✓ **EXTENSIVE LIBRARIES:** php contains vast collection of libraries and extensions for various tasks including file handling, networking and encryption.
- ✓ **CROSS PLATFORAM COMPATIBILITY:** php runs on various operating systems including windows, Mac os & Linux.
- ✓ **OPEN SOURCE:** php is an open source language, which means free to use, modify and distribute.

- ✓ **ERROR REPORTING:** php has pre defined error reporting constants to generate an error notice or warning at run time.
- ✓ **WEB SERVER SUPPORT:** php supports all local servers like apache, Netscape etc.
- ✓ **SECURITY:** php is a secure language to develop the website. It consists of multiple layers of security to prevent threads and malicious attacks.
- ✓ **PERFORMANCE:** php works faster, high processing speed and better performance.

4.WHAT ARE THE BUILDING BLOCKS OF PHP?

The basic building blocks of php,

Variables   Data types   Operators   Expressions   Constants

VARIABLES:

★ A variable is simply a container that holds a certain value. A variable consists of 2 parts i.e.; variables name and variables value.

★ Php is loosely typed language, so we do not need to declare the data types of the variables. It automatically analyzes the values and makes conversions to its correct data type.

Rules:

- ✓ **A** variable must start with a dollar sign $, followed by variable name.
- ✓ It does not contain spaces.
- ✓ A variable name must start with a letter or underscore(A-Z,0-9,_)
- ✓ A php variable, does not start with a number.
- ✓ Assign the value for php variable by using "=" operator. It is case sensitive.

Syntax: $var_name=value;
E.g.: $a=5;

Program:

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

DATA TYPES:

★ Data types are used to store different types of data into variables. Php provides 3 categories of data types. They are

- ➢ Scalar data type
- ➢ Compound data type
- ➢ Special data type

Scalar data type:

The scalar data types are

1) Integer:

★ It is one of the data type. It is used to store whole numbers without decimal numbers(+ve, -ve)

★ An integer must have at least one digit.

★ It supports decimal, hexa decimal, octal, binary number systems(range: -2147483648 to 21474 8647)

★ Syntax & eg:

```
<?php
$x=598;
Var_dump($x);
?>                                output: int(598)
```

2) Float:

★ It stores decimal values. It is also called as doubles.

```
Syntax&eg:
              $x=10.365;
              Var_dump($x);                    output: float(10.365)
```

3) Boolean:

★ It is used in conditional statements. It represents 2 types of values true and false.

```
Syntax&eg:
           $isActive=true;
           Var_dump($isActive);                    output: bool(true)
```

4) String:

★ String is a sequence of characters and encloses with a single quote or double quotes.

```
Syntax&eg:
            $x= “hello”;
            $y= ‘hello world!’;
            Var_dump($x);
          Var_dump($y);
                                                        Output: string(5) “hello”
                                                                String(12) ‘hello world!’
```

Compound data type:

The compound data types are

1) **Array:**

★ It is a compound data type. It is used to store multiple values of same data type in a single variable.

★ An array is a variable that can hold more than one value at a time. It is useful to aggregate a series of related items together. An array is formally defined as an indexed collection of data values.

```
Eg:  <?php
      $bikes = array ("Royal Enfield", "Yamaha", "KTM");
      echo "Array Element1: $bikes[0] </br>";
      echo "Array Element2: $bikes[1] </br>";
      echo "Array Element3: $bikes[2] </br>";
     ?>                                          Output:      royal Enfield
                                                              Yamaha
                                                              KTM
```

2) **Object:**

★ An object is a data type that not only allows storing data but also information on, how to process that data.

★ An object is a specific instance of a class which serves as templates for objects. Objects are created based on this template via the new keyword.

★ Every object has properties and methods corresponding to those of its parent class.

Example:

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

<u>SPECIAL DATA TYPES:</u>

Special data types encompass those types serving some sort of specific purpose, which makes it impossible to group them in any other type category.

1) Resource:

PHP is often used to interact with external data sources such as databases, files &network streams. Typically this interaction takes place through "handles?

EX: such functions include fopen(), mysqli_connect()

2) Null:

The special NULL value is used to represent empty variables in PHP. A variable of type NULL is a variable without any data.

<u>OPERATORS:</u>

An operator is a symbol that specifies a particular action in an expression. The following are the different classes of operators in PHP:

Arithmetic Operators:

The arithmetic operators perform various mathematical operations and will probably used frequently in many of your php programs.

S. No	Operator	Label	Example	Outcome
1	+	Addition	$a + $b	Sum of $a and $b
2	-	Subtraction	$a - $b	Difference of $a and $b
3	*	Multiplication	$a * $b	Product of $a and $b
4	/	Division	$a / $b	Quotient of $a and $b
5	%	Modulus	$a % $b	Remainder od $a and $b

Assignment Operators:

The assignment operators assign a data value to a variable. The simplest form of assignment operator just assigns some value

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

String Operators:
PHP's string operators provide a convenient way in which to concatenate strings together.

S.No	Operator	Label	Example	Outcome
1	.	Concatenation	$a="abc"."def"	$a equals "abcdef"
2	.=	Concatenation Assignment	$a .= "ghi"	$a equals its current value concatenated with "ghi"

Increment & Decrement Operators: Providing shortened means by which you can add 1 to or subtract 1 from the current value of a variable.

S. No	Operator	Label	Example	Outcome
1	++	Increment	$a++, ++$a	Increment $a by 1
2	--	Decrement	$a--, --$a	Decrement $a by 1

Logical Operators: Logical operators make it possible to direct the flow of a program, and are used frequently with control structures, such as the if conditional and the while and for loops.

S.No	Operator	Label	Example	Outcome
1	&& AND	Logical AND	$a && $b $a AND $b	True if both $a and $b are true
2	| | OR	Logical OR	$a | | $b $a OR $b	True if either $a or $b is true
3	! NOT	Logical NOT	!$a NOT $a	True if $a is not true
4	XOR	Exclusive OR	$a XOR $b	True if only $a or only $b is true

Equality Operators: Equality operators are used to compare two values, testing for equivalence.

S.No	Operator	Label	Example	Outcome
1	= =	Is equal to	$a == $b	True if $a and $b are equivalent
2	! =	Is not equal to	$a != $b	True if $a is not equal to $b
3	= = =	Is identical to	$a ===$b	True if $a and $b are equivalent and $a & $b are of same type

Comparison Operators: Comparison operators, like logical operators, provide a method by which to direct program flow through examination of the comparative values of two or more variables.

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

<u>EXPRESSIONS:</u>
Almost everything in a PHP script is an expression. Anything that has a value is an expression. In a typical assignment statement ($x=100), a literal value, a function or operands processed by operators is an expression, anything that appears to the right of assignment operator (=)

Syntax

$x=100; //100 is an expression

$a=$b+$c; //b+$c is an expression

$c=add($a,$b); //add($a,$b) is an expression

$val=sqrt(100); //sqrt(100) is an expression

$var=$x!=$y; //$x!=$y is an expression

Expression with Ternary conditional operator

Ternary operator has three operands. First one is a logical expression. If it is TRU, second operand expression is evaluated otherwise third one is evaluated

```
<?php
$marks=60;
$result= $marks<50 ? "fail" : "pass";
echo $result;
?>                                   OUTPUT: pass
```

CONSTANTS:

★ A constant is an identifier (name) for a simple value. A constant value cannot change during the execution of the script.

★ Constants are useful for storing data that doesn't change while the script is running Constants are defined using define() function, which accepts two arguments: the name of the constant, and its value.

★ PHP constants can be defined by 2 ways:

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

★ Constants are similar to the variable except once they defined, they can never be undefined or changed.

★ They remain constant across the entire program. PHP constants follow the same PHP variable rules. For example, it can be started with a letter or underscore only.

PHP constant: define()

Use the define() function to create a constant. It defines constant at runtime.
syntax of define() function in PHP.

define(name, value, case- insensitive)

name: It specifies the constant name.

Value :It specifies the constant value.

case-insensitive: Specifies whether a constant is case-insensitive. Default value is false. It means it is case sensitive by default.

Example: **<?php define("MESSAGE", "Hello PHP");**
Echo MESSAGE; ?>   output: Hello PHP

PHP constant: const keyword

★ PHP introduced a keyword const to create a constant. The const keyword defines constants at compile time. It is a language construct, not a function.

★ The constant defined using const keyword are case-sensitive.

Example: **<?php**
Const MESSAGE="Hello STUDENTS";

```
Echo MESSAGE;
?>                    Output: Hello STUDENTS
```

1.2 FLOW CONTROL FUNCTIONS IN PHP,

5.WRITE ABOUT FLOW CONTROL FUNCTIONS IN PHP?

The Control Structures (or) Statements used to alter the execution process of a program. These statements are mainly categorized into following:

- Conditional Statements
- Loop Statements
- Jump Statements

Conditional Statements:

Conditional Statements performs different computations or actions depending on conditions. In PHP, the following are conditional statements

- ✓ **If** statement
- ✓ **if -else** statement
- ✓ **if-elseif-else** statement
- ✓ **switch** statement

☞**if statement**

The if statement is used to test a specific condition. If the condition is true, a block of code(if- block) will be executed.

Syntax:

```
if (condition)
{
        Statements;
}
```

```
$age=18;
    If($ag==18){
     Echo " major";
          }
```

☞**if-else statement**

The if-else statement provides an else block combined with the if statement which is executed in the false case of the condition.

Syntax:

```
 if (condition)
  {
          Statements;
  }
  else
  {
          Statements;
  }
```

```
$age=18;
If($age<18){
Echo " minor";
}
Else{
Echo "major";
}
```

☞**if-else if-else statement**

The else if statement enables us to check multiple conditions and execute the specific block of Statements depending up on the true condition among them.

Syntax:

```
if (condition1)
  {
          Statements;
```

```
            }
  else if (condition2)
  {
            Statements;
  }
  else if (condition3)
  {
            Statements;
  }
  .
  .
  else
  {
            Statements;
  }
```

```
$age=18;
If($age<18){
Echo " minor";
}
Else if ($age==18){
Echo " just turn into an adult"
}
Else
{
Echo "major";
}
```

☞**switch statement**

The switch statement enables us to execute a block of code from multiple conditions depending upon the expression.

Syntax:

```
switch (expression)
  {
  case 1: statements; break;
case 2: statements; break;
case 3: statements; break;
  .
  default: statements;
  }
```

```
<html>
  <head>
  <title>SwitchDemo</title>
  </head>
  <body>
  <?php
        $x=15;
        $y=10;
        $op= '*';
        Switch($op){
        Case '+': $z=Sx+Sy; echo "addition is:$z"; break;
        Case '-': $z=Sx-Sy; echo "subtraction is:$z"; break;
        Case '*': $z=Sx*Sy; echo "multiplication  is:$z"; break;
        Case '/': $z=Sx/Sy; echo "division  is:$z"; break;
        Default: echo "invalid operator"; }
        ?></bodv></html>
```

Loop Statements:

Sometimes we may need to alter the flow of the program. If the execution of a specific code may need to be repeated several numbers of times then we can go for loop statements.

In PHP, the following are loop statements

- **While** loop
- **do-while** loop
- **for** loop

☞**while loop statement**

The while loop we can execute a set of statements as long as a condition is true. The while loop is mostly used in the case where the number of iterations is not known in advance.

Syntax:

```
while (condition)
{
        Statements;
}
```

```
<html>
<head>
<title>WhileDemo</title>
</head>
<body>
<h1>While Demo</h1>
<?php
$n=1; while($n<=5){
Echo $n<br>;
$n++;
}
?>
</body> </html>
```

do-while loop statement

The do-while loop will always execute a set of statements at least once and then execute a set of statements as long as a condition is true.

Syntax:

```
do
{
   Statements;
} while (condition);
```

```
<html>
  <head>
  <title>Do-While Demo</title>
  </head>
  <body>
  <h1>Do-While Demo</h1>
  <?php
  $n=1; do{
  Echo "$n<br>";
  $n++;
  }while($n<=5);?>
  </body>
  </html>
```

For loop statement

The for loop we can execute a set of statements specified number of times. The for loop is mostly used in the case where the number of iterations is known in advance..

Syntax:

```
for(initialization; condition; increment/decrement)
{
   Statements;
}
```

Eg:

```
<html>
<head>
<title>ForDemo</title>
</head>
<body>
<h1>For Demo</h1>
<?php
for($i=1;$i<=5;$i++)
{
        echo "$i <br/>";
}
?>
</body>
</html>
```

Jump Statements:

Jump statements in PHP are used to alter the flow of a loop like you want to skip a part of a loop or terminate a loop.
In PHP, the following are jump statements

- **Break** statement
- **Continue** statement

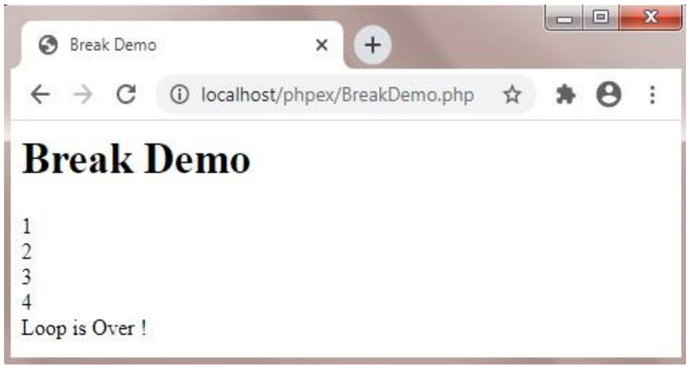
☞**break statement**

- ★ The break statement breaks the loops one by one ,i.e. ,in the case of nested loops, it breaks the inner loop first and then proceeds to outer loops. The break is commonly used in the cases where we need to break the loop for a given condition.

Syntax: break;

Eg:

```
<html>
<head>
<title>Break Demo</title>
</head>
<body>
<h1>Break Demo</h1>
<?php
for($i=1;$i<=10;$i++)
{
        if($i==5)
        {
break;
        }
        echo "$i <br/>";
}
echo "Loop is Over !";
?>
</body></html>
```



```
Break Demo

1
2
3
4
Loop is Over !
```

Continue statement

- ★ The continue statement in php is used to bring the program control to the beginning of the loop. i.e. when a continue statement is encountered inside the loop, remaining statements are skipped and loop proceeds with the next iteration.
- ★ The continue statement skips the remaining lines of code inside the loop and start with the next iteration.

It is mainly used for a particular condition inside the loops that we can skip some specific code for a particular condition.

Syntax: continue;

Eg:

```
<html>
<head>
<title>Continue Demo</title>
</head>
<body>
<h1>Continue Demo</h1>
<?php
for($i=1;$i<=10;$i++)
{
        if($i%2==0)
        {
continue;
          }
          echo "$i <br/>";
  }
  echo "Loop is Over !";
  ?>
  </body>
  </html>
```

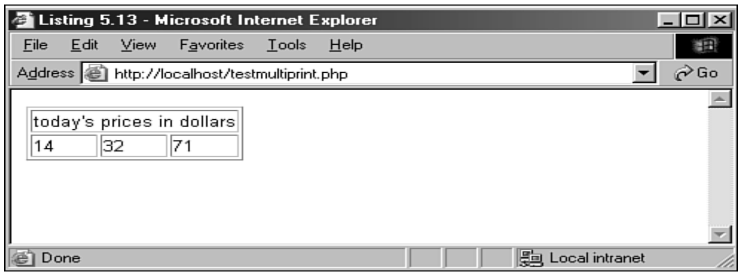
6.EXPLAIN ABOUT CODE BLOCKS & BROWSER OUTPUT IN PHP?

Imagine a script that outputs a table of values only when a variable is set to the Boolean value true. Shows a simplified HTML table constructed with the code block of an if statement.

```
<html>

<head>

<title>Listing 5.13</title>

</head>

<body>

<?php
$display_prices=true; 8: if ($display_prices) {
print "<table border=\"1\">";

print "<tr><td colspan=\"3\">"; print"today'spricesindollars"; print "</td></tr>";
print "<tr><td>14</td><td>32</td><td>71</td></tr>";
    print"</table>";

    }
```

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

If $ display_prices is set to true inline7, the table is printed. For the sake of readability, we split the output into multiple print()statements, and once again escape any quotation marks.
Put these lines into a text file called testmultiprint.php, and place this file in your Web server document root . When you access this script through your Web browser, it should look like


Listing 5.13 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost/testmultiprint.php
Go
Done
Local intranet

```
today's prices in dollars
14    32    71
```

1.3 WORKING WITH FUNCTIONS

7. EXPLAIN ABOUT FUNCTIONS IN PHP?

functions

PHP function is a piece of code that can be reused many times. It can take input as argument list and return value. There are thousands of built-in functions in PHP. In PHP, we can define Conditional function, Function within Function and Recursive function also.

Advantages of PHP Functions

- ✓ **Code Reusability**: PHP functions are defined only once and can be invoked many times, like inother programming languages.
- ✓ **Less Code**: It saves a lot of code because you don't need to write the logic many times. By the use of function, you can write the logic only once and reuse it.
- ✓ **Easy to understand**: PHP functions separate the programming logic. So it is easier to understand the flow of the application because every logic is divided in the form of functions.

PHP User-defined Functions

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

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

PHP Functions Example

File:function1.php

Eg:

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

Output: Hello PHP Function";

PHP Function Arguments

We can pass the information in PHP function through arguments which is separated by comma. PHP supports **Call by Value** (default),**Call by Reference**, **Default argument values** and **Variable-length argument list**.

File:functionarg.php

```
<?php
function sayHello($name,$age){
echo "Hello $name, you are $age years old<br/>";
}
sayHello("Ramu",27);
sayHello("Venu",29);
?>
Output:
```

```
Hello Ramu, you are 27yearsold
Hello Venu,you are 29 yearsold
```

PHP Function: Returning Value *File:functiondefaultarg.php*

```
<?php
functioncube($n){
return $n*$n*$n;
}
    echo"Cubeof3is:".cube(3);
    ?>
```

Output: Cubeof3is:27

8.**Explain about Call By Value And Call By Reference In PHP?**

Call By Value

In this method, only values of actual parameters are passing to the function. So there are two addresses stored in memory. Making changes in the passing parameter does not affect the actual parameter.

Example

```
<?php
function square($num){
        $s=$num*$num;
        return $s;
}
$a = 5;
echo $a."<br>"; //Output: 5
echo $b; //Output: 25
?>
```

Output: 5

Call by Reference

In this method, the address of actual parameters is passing to the function. So any change made by function affects actual parameter value.

Example

```
<?php
function cube(&$num){
```

```
$c=$num*$num*$num;
return $c;
}
$a = 5;
$b = cube($a);
echo “$a;<br>”;
echo $b; ?>
output:
 125
 125
```

9. Differences between echo and print?

echo	**Print**
Echo is a statement, which is used to display the output.	Print is also a statement ,used as an alternative to echo at many times to display the output.
Echo can be used with or without parentheses.	Print can be used with or without parentheses.
Echo does not return any value.	Print always returns an integer value, which is 1.
We can pass multiple strings separated by comma(,)in echo.	Using print, we can not pass multiple arguments.
Echo is faster than print statement.	Print is slower than echo statement.
<?php $fname = "krishna"; $lname =”chandrika"; echo"My nameis: ".$fname,$lname; **?>** **Output: Krishna chandrika**	**<?php** $fname = "krishna"; $lname =”chandrika"; print"My nameis: ".$fname; print $lname; **?>** Output: Krishna chandrika

10. DEFINE VARIABLE SCOPE? WHAT ARE THE DIFFERENT SCOPES OF VARIABLES IN PHP?

Variable Scopes:

The scope of a variable is defined as its extent in the program within which it can be accessed, i.e., the scope of a variable is the portion of the program within which it is visible or can be accessed. Depending on the scopes, PHP has three variable scopes.

Local variables:

The variables declared within a function are called local variables to that function and have their scope only in that particular function. In simple words, it can not be accessed outside that function. Any declaration of a variable outside the function with the same name as that of the one within the function is a completely different variable. For now, consider a function as a block of statements.

```
<?php

function sum($n1,$n2) {

$s=0; //local variable
$s=$n1+$n2; return $s;
}

$a=50;
$b=30;
echo "sum of $a and $b is sum($a,$b)";

?>
```

Output:

```
Sumof50and30is80
```

Global variables:

The variables declared outside a function are called global variables. These variables can be accessed directly outside a function. To get access within a function, we need to use the “global” keyword before the variable to refer to the global variable.

Eg:

```
<?php
  $s=0;        //global variable
function mul($n1,$n2) {

global $s; //global variable access using global keyword
$s=$n1*$n2; return $s;
}

$a=50;
$b=30;
echo "mul of $a and $b is mul($a,$b)";

?>
```

Output:

```
mul of 50 and 30 is 150
```