Government Arts and Science College (Women), Sathankulam – 628704

Class	: II B.Sc. (Computer Science)
Semester	: IV
Subject	: Programming with PHP &MySQL
Topic	: Unit I
Faculty	: Mrs. A. Angeline Nancy Sophia M.Sc., M.Phil., Department of Computer Science

INTRODUCTION TO PHP

- It is a widely used general-purpose scripting languages suitable for web and can be embedded into HTML
- It allow web developers to write dynamically generated web pages quickly
- It is executed on the server. It also works independently of the user's web browser

Why do we prefer PHP?

- Simple – easy to learn and powerful to use
- Uses clear, simple syntax which makes it easy to read and understand
- It is available free of cost on the Internet for a variety of platforms include UNIX, Microsoft windows and Mac OS and for most web servers
- It is an interpreted language. So we can perform incremental, iterative development and testing without compile-test-debug cycle each time we change code.
- Has a variety of data types
- Powerful object-oriented engine
- Has extensive library of in-built functions
- Support most current web technologies and protocols

History of PHP

- First version of PHP was PHP/FI was developed by Rasmus Lerdorf in mid 1995. Support basic functions
- PHP/FI 1.0 , PHP/FI 2.0
- In 1997 PHP 3.0 was developed by Andi Gutmans and Zeev Suraski.
  - It included wider range of databases including MySQL and Oracle.
  - Encouraged independent developers creating in their own language
- PHP 4.0 released in 2003, to deliver better performance, greater reliability and scalability , support for web servers and host new language features and better OOP support
- PHP 5.0 includes exception handling, improved MySQL support, better memory manager

FEATURES

Simplicity

- Uses consistent and logical syntax with clear written manual, even novices find it easy to learn
- Adherence to KISS (Keep It Simple Stupid)

- Access C libraries

Portability

- A program can work on different platforms. PHP scripts written on one platform works on any other platform

Speed

- Faster than other scripting languages. Use of object handles that reduces memory consumption and help applications run faster

Open source

- Source code is freely available on web and developers can install and use it without paying licensing fees.
- Ensures faster bug fixes and quicker integration of new technologies into the core language

Extensible

- Can easily add support for new technologies to the language through modular extensions
- Dynamically create image, PDF (Portable Document Format) and SWF (Shock Wave Flash)files
- Connect to IMAP and POP3 servers
- Handle electronic payments
- Execute Perl, Java and COM code
- Online repository of free PHP classes called PEAR (PHP Extension and Application Repository) which provides reusable and bug-free PHP components

XML and Database Support

- Enables new features in the MySQL RDBMS and also supports DB@, PostgresSQL, Oracle,mSQL, MS-SQL, Informix, Sybase and SQLite

EMBEDDING PHP IN HTML

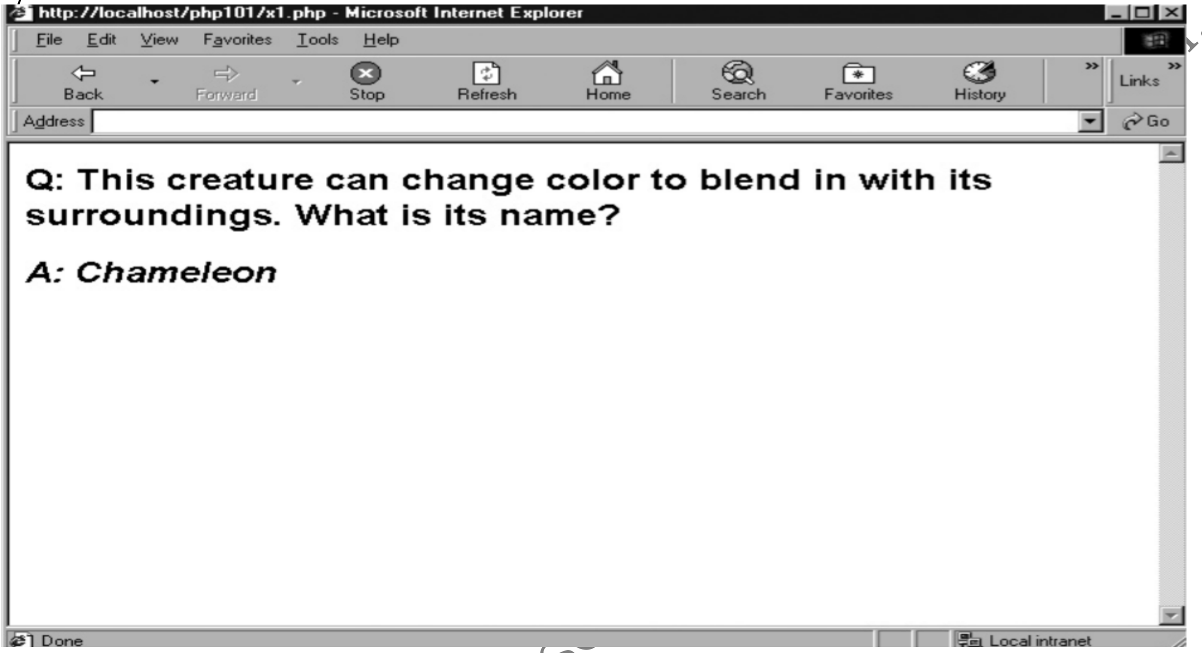
- PHP embed commands in regular HTML pages
- These commands are enclosed within special start and end tags which are read by PHP interpreter

```
<?php
… PHP code…
?>
```

Simple program

```
<html>
```

```
<head><basefont face="Arial"></head>
<body>
<h2>Q: This creature can change color to blend in with its surroundings. What is its name?</h2>
<?
echo '<h2><i>A : Chameleon</i></h2>';
?>
</body>
</html>
```


http://localhost/php101/x1.php - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Links
Address Go
Done
Local intranet

Q: This creature can change color to blend in with its surroundings. What is its name?

A: Chameleon

Comments

```
<?php
// this is a single-line comment
# so is this
/* and this is a multiline comment */
?>
```

VARIABLES

- Variables are building blocks of any programming language.
- Used to store both numeric and nonnumeric data.
- The contents of variable can be altered during program execution
- Can be compared and manipulated using operators
- Variable types:
  - Booleans
  - Integers
  - Floating point numbers
  - Strings
  - Arrays
  - Objects resources NULLS

- PHP can automatically determine variable types by its context
- Every variable has a name preceded by a dollar ($) symbol
- It must begin with a letter or underscore character.
- Example: $popeye, $one_day, $INCOME are valid names

```
<html>
<head> <basefont face="Arial"></head>
<body>
<h2>Q: This creature has tusks made of ivory.
    What is its name? </h2>
<?php
//define variable
$answer = 'Elephant';
//print output
echo  "<h2><i>$answer</i></h2>";
?>
</body>
</html>
```

echo() function is used to print data

Assigning and using variables

- Use assignment operator (=) to assign a variable
- Value may be a variable, an expression

```
<?php
$age = $dob + 15
?>
```

To use a variable we can also call the variable by name and PHP will substitute its value at run time

```
<?php
$today = "Aug 10 2020";
echo "Today is $today";
?>
```

Data Types

Data Type	**Description**	**Example**
Boolean	Specifies true or false	$auth = true;
Integer	Whole number	$age = 99;
Floating point	Fractional number specified either in decimal or scientific notation	$temperature = 56.89;
String	Sequence of characters enclosed either in double quotes(" ") or single quotes(' ')	$name = "Harry";

Detecting the data type of variable

gettype() function accepts a variable or value as argument and find out what type of variable it is.

```
<?php
$auth = true;
$age = 27;
$name = 'John';
$temp = 98.6;
echo gettype($name);
echo gettype($auth);
echo gettype($age);
echo gettype($temp);
?>
```

Output

```
String
Boolean
Integer
double
```

Functions to Check if a variable or value belongs to a specific type

Function	**What it does**
is_bool()	Checks if a variable or value is boolean
is_string()	Checks if a variable or value is a string
is_numeric()	Checks if a variable or value is a numeric string
is_float()	Checks if a variable or value is a floating point number
is_int()	Checks if a variable or value is an integer
is_null()	Checks if a variable or value is NULL
is_array()	Checks if a variable or value is an array
is_object()	Checks if a variable or value is an object

String values

- String values enclosed in double quotes are automatically parsed for variable names;
- If variable names are found, they are automatically replaced with variable value
- If string contains quotes, carriage returns or backslashes we can use with backslash.

```
<?php
$identity = 'James Bond';
$car = 'BMW';
$sentence = "$identity drives a $car";
$sentence = '$identity drives a $car';
?>
```

Output

James Bond drives a car

$identity drives a $car

Example: $statement = 'It's hot outside'; // error due to mismatched quotes

$statement = 'It\'s hot outside';

NULL Values

- PHP 4.0 introduced new data type for empty variables called NULL
- It means variable has no value.

Example:

```
<?php
echo gettype($me); // returns NULL because it is not intialized
$me = 'David'
echo gettype($me); // returns string
?>
```

Operators

PHP has 15 operators including operators for assignment, arithmetic, string, comparison and logical operations

Operator	**What it does**
=	Assignment
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus

.	String concatenation
==	Equal to
===	Equal to and of same type
!==	Not equal to or not of same type
<> aka !=	Not equal to
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
&&	Logical AND
||	Logical OR
!	Logical NOT
++	Addition by 1
--	Subtraction by 1

Using Arithmetic Operators

To perform mathematical operations on variables, we use standard arithmetic operators.

```
<?php
//define variables
$num1 = 101;
$num2 = 5;
$sum = $num1 + $num2;
$diff = $num1 - $num2;
$product = $num1 * $num2;
$quotient = $num1 / $num2;
$remainder = $num1 % $num2;
?>
```

Using String Operators

To add two strings together, we use the string concatenation operator, represented by a period (.).

```
<?php
$username = 'john';
$domain = 'example.com';
```

```
$email = $username . '@' . $domain;
?>
```

We can concatenate and assign simultaneously as follows:

```
<?php
$str = 'the';
$str .= 'n';
//$str now contains "then"
?>
```

Using Comparison Operator

To test whether two variables are different, we use any one of PHP's comparison operators.

```
<?php
$mean = 29;
$median = 40;
$mode = 29;
$result = ($mean < $median);   //returns true.
 $result = ($mean > $median);   //returns false.
$result = ($mean <= $mode);   //returns false.
$result = ($mean >= $mode);   //returns true.
$result = ($mean == $mode);   //returns true.
$result = ($mean != $mode);   //returns false.
$result = ($mean <> $mode);   //returns false.
?>
```

The comparison test is always a Boolean value.

The === Operator

PHP 4.0 has === operator which test both for equality and type.

```
<?php
$str = '14';
$int = 14;
$result = ($str == $int);        // returns true
$result = ($str === $int);             // returns false
?>
```

Using Logical Operators

To link together related conditions in a simple manner PHP use four logical operators: logical AND, logical OR, logical XOR and logical NOT.

```
<?php
$user = 'joe';
$pass = 'trym3';
$savecookie = 1;
$status = 1;
$result = (($user == 'joe') && ($pass == 'trym3'));          // returns true.
$result = (($status < 1) || ($savecookie == 0));        // returns false.
$result = !($savecookie ==1);                              // returns false.
$result = (($status == ) xor ($savecookie == 1));       // returns false
?>
```

Using the Auto-Increment and Auto-Decrement Operators

The auto-increment operator is designed to automatically increment the value of the variable it is attached to by 1. It is represented by a double addition symbol (++).

```
<?php
$total = 10;
$total++;           // $total is now 11.
?>
```

The auto-decrement operator is designed to automatically decrement the value of the variable it is attached to by 1. It is represented by a double subtraction symbol (--).

```
<?php
$total = 10;
$total--;           // $total is now 9.
?>
```

Operator Precedence

- !   ++   - -
- *   /   %
- +   -   .
- <   <=   >   >=
- ==   !=   ===   !==
- &&

- ||
- ? :

CONDITIONAL STATEMENTS

A conditional statement enables us to test whether a specific condition is true or false and to perform different actions on the basis of the test result.

Using the if ( ) statement

The simplest form of conditional statement is the if ( ) statement.

```
<?php
if (conditional test)
{
        Do this;
}
?>
```

Example

```
<?php
if ($temp >=100)
{
        echo 'Very hot!';
}
?>
```

If the statement evaluates to true, all PHP code within the curly braces is executed, if not the code within the curly braces is skipped and lines following the if ( ) construct are executed.

Using if-else ( ) statement

if-else ( ) construct is used to define an alternate block of code that gets executed when the conditional expression in the if ( ) statement evaluates as false.

```
<?php
if (conditional test)
{
        Do this;
}
else
{
        do this;
}
?>
```

Example

```
<?php
if ($temp >=100)
{
        echo 'Very hot!';
}
else
{
        echo 'within tolerable limits';
}
?>
```

Using if-elseif-else ( ) statement

This construct consists of listing a number of possible results, one after another and specifying the action to be taken for each.

```
<?php
if (conditional test #1)
{
        do this;
}
elseif (conditional test #2)
{
        do this;
}
…
elseif (conditional test #n)
{
        do this;
}
else
{
        do this;
}
?>
```

Example

```
<?php
if ($country = = 'UK')
{
        $capital = 'London';
}
elseif ($country = = 'US')
{
        $capital = 'Washington';
}
elseif ($country = = 'FR')
{
        $capital = 'Paris';
}
else
{
        $capital = 'unknown';
}
?>
```

The if-elseif-else ( ) assigns a different value to the $capital depending on the country code. If one of the if ( ) is found to be true, PHP executes the corresponding code and skip the remaining statements in the block and jump to the lines following the if-elseif-else ( ) block.

Using Switch ( ) Statement

A switch ( ) statement evaluates a condition, an appropriate case( ) block is executed. If no matches can be found, a default block is executed.

```
<?php
switch (condition variable)
{
        case possible result #1:
                do this;
        case possible result #2:
                do this;
        …
        case possible result #n:
                do this;
        case default:
                do this;
}
?>
```

Example

```
<?php
switch ($country)
{
        case ‘UK’:
                $capital = ‘London’;
                break;
        case ‘US’:
                $capital = ‘Washington’;
                break;
        case ‘FR’:
                $capital = ‘Paris’;
                break;
        default:
                $capital = ‘unknown’;
                break;
}
?>
```

LOOPING STATEMENTS

A loop is a control structure that enables to repeat the same set of statements or commands over and over again on a number of repetitions or on the fulfillment of a certain conditions.

Using the while ( ) loop

The first and simplest loop is the while ( ) loop. This loop executes as long as the conditional expression specified evaluates to true. When the condition becomes false, loop is exited and the statement following the loop will be executed.

Syntax

```
<?php
while (condition is true)
{
        do this;
}
?>
```

Example

```
<?php
$num = 11;
$upperlimit = 10;
$lowerlimit = 1;
while( $lowerlimit <= $upperlimit)
{
        echo $num x $lowerlimit = " . ($num * $lowerlimit);
        $lowerlimit++;
}
?>
```

This script uses a while( ) loop to count forwards from 1 until the values of $lowerlimit and $upperlimit are equal.

Using the do ( ) loop

In this construct the statements within the loop are executed first and the condition to be tested is checked after.

Syntax

```
<?php
do
{
        do this;
} while (condition is true)
?>
```

Example

```
<?php
$num = 11;
$upperlimit = 10;
$lowerlimit = 12;
```

```
do
{
        echo $num x $lowerlimit = " . ($num * $lowerlimit);
        $lowerlimit++;
} while( $lowerlimit <= $upperlimit)

?>
```

Using the for ( ) loop

This loop is used to execute a certain set of statements a fixed number of times.

Syntax

```
<?php
for ( initialize counter; conditional test; update counter)
{
        do this;
}
?>
```

This loop uses a counter that is initialized to a numeric value and keeps track of number of times the loop is executed. Before each execution of the loop, a conditional statement is tested. If it is true, the loop will execute once more and the counter is incremented by1. If it is false, the loop will be broken and the lines following the loop will be executed instead.

```
<?php
for ($x = 2; $x <= 100; $x++)
{
        echo "$x "
}
?>
```

Break and Continue Statements

Break

The break keyword is used to exit a loop when it encounters an unexpected situation.

```
<?php
for ($x = -10; $x <=10; $x++)
{
        if ( $x == 0) { break; }
        echo '100 / ' . $x . ' = ' . (100/$x);
}
?>
```

When dividing one number by another one it is advisable to check the divisor and use the break statement to exit the loop as soon as it becomes equal to zero.

Continue

The continue keyword is used to skip a particular iteration of the loop and move to next iteration immediately. This statement can be used to make execution of the code within the loop block dependent on particular circumstances.

```
<?php
for ( $x = 10; $x <= 100; $x++)
{
        if (($x % 12) == 0)
        {
                echo "$x ";
        }
        else
        {
                continue;
        }
}
?>
```