

# Introduction to PHP

---

SERVER SIDE SCRIPTING LANGUAGE

# Server side scripting with PHP

---

- Server side scripting runs on the server computer
- It is used to retrieve and generated dynamic content of HTML
- Commonly used server side scripting are: PHP, Ruby, ASP, ASP.NET, JSP, Python etc.
- ASP/ASP.NET :
  - Active Server Pages developed by Microsoft to make advanced web pages
- JSP/Servlets:
  - Java Server Pages which include JSP tags mixed with HTML. Most popular language for higher level applications
- ColdFusion:
  - Runs on top of JSP/servlet engine. Users can download a free developer edition but is limited to one IP address
- PHP: PHP is easy to learn, which is similar to C programming language

# Features of PHP

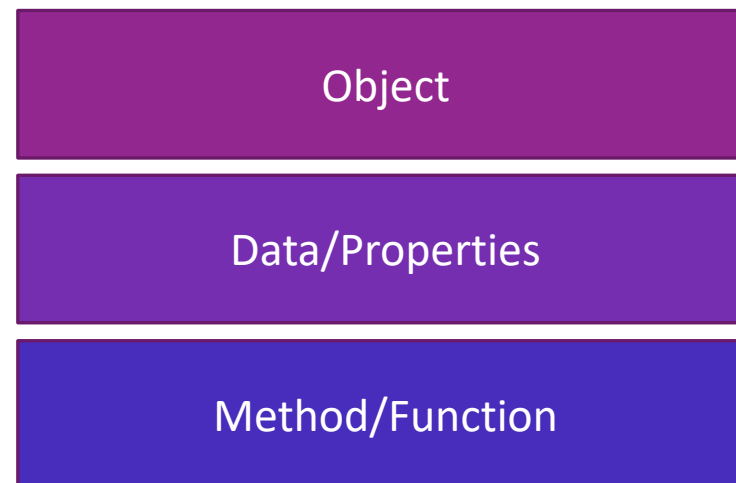
---

- PHP is an acronym for "PHP: Hypertext Preprocessor"
- PHP is a widely used, open source scripting language
- PHP scripts are executed on the server
- PHP is free to download and use
- PHP code is hidden from end user making it more secure
- PHP is easy to learn and write
- PHP files have file extension of ".php"
- PHP generates HTML page which is sent to clients
- PHP runs on different platforms(Windows, Linux, Unix etc)

# Object Oriented Programming in PHP

---

- OOP is a programming model that is based on the concept of classes and objects
- Procedural language focuses on writing procedures or functions that perform operations on data
- OOP programming focuses on the creation of objects which can contain both data and functions together



# Advantages of OOP

---

- It provides modular structure for programs
- Easier to understand because of structured concept of OOP
- Easier to maintain, modify and debug
- Provides high degree of reusability
- Reduces the development time with less line of code (LOC)
- Hide program complexity
- Develop complex and reusable programs
- Secure program can be developed
- Provides: Inheritance, Object, Class, Polymorphism, Encapsulation, abstraction etc.

# OOP features

---

- Class:

- It is programmer defined data type which includes local data and function.
- It is template for making many instances of the same class of object

- Object:

- And individual instance of the data structure defined by a class.
- We can make many objects from a class

- Inheritance:

- Reusing feature of existing functions and properties from parent to child class is called inheritance
- Child class will inherit all or few functions and variable of parent class

# OOP features

---

- Polymorphism:
  - It is concept where same function can be used for different purpose
  - "Poly" stands for "many" and "morph" stands for "forms"
- Overloading:
  - It is a type of polymorphism in which some or all of operators have different implementations depending on the types of their arguments
  - Functions can also be overloaded with different implementations
- Data abstraction:
  - Any representation of data in which the implementation details are hidden
- Encapsulation:
  - It refers to a concept where all data and member functions are bound together to form an object
  - It hides unnecessary details in a program

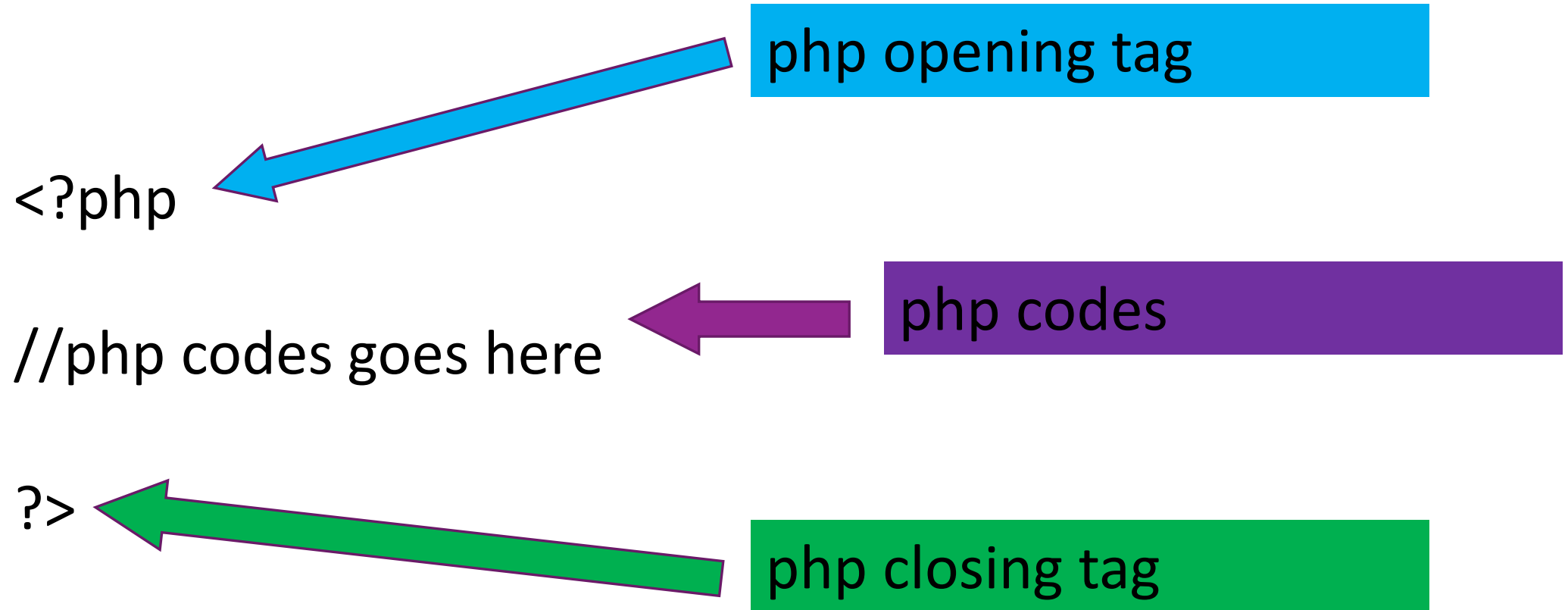
# Creating PHP file

---

- Create new file in any text editor (notepad, sublime text)
- Type HTML codes along with PHP codes
- Save the file with some name and .php extension (e.g. home.php)
- PHP code or script can be placed anywhere in the document
- Save the file in server folder (in local computer the path is C:\xampp\htdocs )
- Open browser and type address  
`https://localhost/home.php`

# Basic PHP syntax

---



# First PHP Program

---

```
<html>
```

```
....
```

```
<body>
```

```
<h1>Our first PHP program</h1>
```

```
<?php
```

```
    echo "hello world";
```

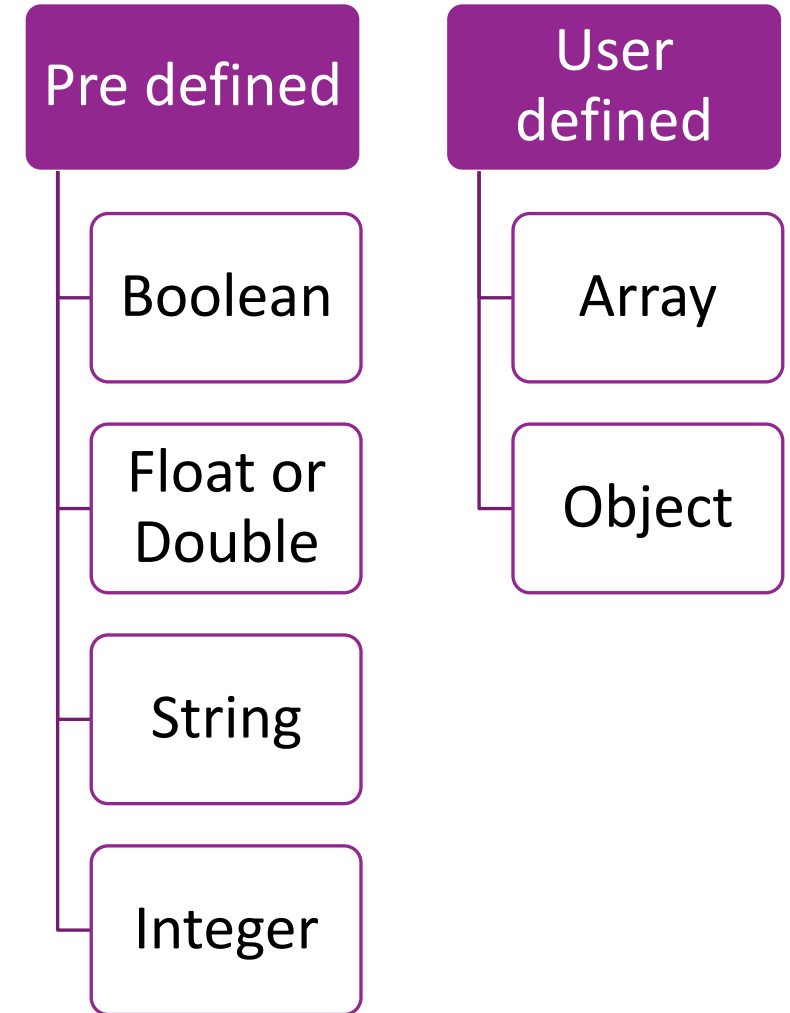
```
?>
```

```
</body>
```

```
</html>
```

# PHP Data Types

- Data types defines the type of data a variable can store
- It is not compulsory to define the data types during declaration of variable
- PHP automatically converts the variable to the correct data type depending on its value



# Data Types

---

- Boolean

- Boolean has only two possible values either TRUE(1) or FALSE(0)
- Boolean are used in conditional testing statements

- Integers

- Integer is a numeric data type without decimal (i.e. 1,5,35,-7)
- Integers can be either positive or negative
- It can be decimal, hexadecimal or octal
- Default is decimal

- Float or Double

- Floating point numbers are also known as real numbers or fractional numbers
- e.g. 3.5, 2.9, -10.6

# Data Types

---

## ■ String

- Strings can be enclosed in either single or double quote with different behavior at read time
- Singly quoted string: Singly quoted strings read in and store their characters literally
- Doubly quoted string: Variable names (String with \$) are replaced with string representation of their values

# Variables

---

- Variables are containers for storing information
- PHP variables are case sensitive
- PHP is loosely typed language i.e. it automatically converts the variable to correct data type depending on its value. No need to define explicitly
- Variable starts with \$ sign followed by variable name

```
<?php
```

```
    $txt = "Hello world";
```

```
    $x = 5;
```

```
    $y = 10.5;
```

```
?>
```

# Rules for PHP variable Naming

---

- A variable must start with the \$ (dollar) sign, followed by variable name
- Variable name must start with a letter or the underscore (\_) character
- Variable name cannot start with number
- Variable name can only have alpha numeric characters and underscore (a-z,A-Z,0-9 and \_)
- Variable names are case sensitive (\$age and \$AGE are not same)

# PHP Output Commands

---

- In PHP, Echo and Print statements are used to output data to screen or window
- Echo has no return value while print has return value of 1 (true) so it can be used in expressions
- Echo can take multiple parameters while print can take one argument
- Echo is marginally faster than print

```
<?php
```

```
    echo "hello world";
```

```
    print "Good morning";
```

```
?>
```

# Operators

---

- Operators are used to perform operation on variables and values
  - Similar to JavaScript there are Unary, Binary and Tertiary operators
1. Arithmetic operators (+, -, \*, /, %)
  2. Assignment operators (=, +=, -=, \*=, /=)
  3. Comparison operators (<, >, <=, >=, !=, ==)
  4. Increment/Decrement operators (++ , --)
  5. Logical operators (&&, ||, !)
  6. String operators (., .=)
  7. Array operators (+, ==, ===, !=, <>, !==)
  8. Conditional operators (? :)

# Arithmetic operator

- Arithmetic operators are used with numeric values to perform common arithmetic operations such as addition, subtraction, multiplication division

Operator	Name	Example	Result
+	Addition	$\$x + \$y$	adds value of $\$x$ & $\$y$
-	Subtraction	$\$x - \$y$	subtracts value of $\$x$ and $\$y$
*	Multiplication	$\$x * \$y$	multiplies $\$x$ with $\$y$
/	Division	$\$x / \$y$	divides $\$x$ by $\$y$
%	Modulus	$\$x \% \$y$	gives remainder of $\$x / \$y$
**	Exponentiation	$\$x ** \$y$	raising $\$x$ to $\$y$ power

# Arithmetic operator example

---

.....

```
<body>
```

```
    <?php
```

```
        $x = 10;
```

```
        $y = 6;
```

```
        $sum = $x + $y;
```

```
        echo "Sum is ".$sum;
```

```
    ?>
```

```
</body>
```

# Assignment operator

- Assignment operators are used to copy or store value of right side of equal to sign (=) to the left side variable

Operator	Name	Example	Result
=	Simple assignment	$\$x = \$y$	stores value of $\$y$ to $\$x$
+=	Addition assignment	$\$x += \$y$	adds $\$x$ & $\$y$ and stores to $\$x$
-=	Subtraction assignment	$\$x -= \$y$	subtracts $\$x$ & $\$y$ and stores to $\$x$
*=	Multiplication assignment	$\$x *= \$y$	multiplies $\$x$ & $\$y$ and stores to $\$x$
/=	Division assignment	$\$x /= \$y$	divides $\$x$ by $\$y$ and stores to $\$x$
%=	Modulus assignment	$\$x \% = \$y$	divides $\$x$ by $\$y$ and stores the remainder to $\$x$

# Assignment operator example

---

.....

```
<body>
```

```
    <?php
```

```
        $x = 10;
```

```
        $y = 6;
```

```
        $x += $y; // $x = $x + $y;
```

```
        echo "Sum is ".$x;
```

```
    ?>
```

```
</body>
```

# Relational / comparison operator

---

- Relational or comparison operators are used to compare two or more values. It returns either TRUE or FALSE value

Operator	Name	Example	Returns TRUE if
==	Equal to	\$x == \$y	\$x and \$y are equal
!= / <>	Not equal to	\$x != \$y	\$x and \$y are not equal
===	Identical	\$x === \$y	\$x and \$y are equal and of same type
!==	Not Identical	\$x !== \$y	\$x and \$y are not equal and not of same type
>	Greater than	\$x > \$y	\$x is greater than \$y
<	Less than	\$x < \$y	\$x is less than \$y
>=	Greater than or equal to	\$x >= \$y	\$x is greater than or equal to \$y
<=	Less than or equal to	\$x <= \$y	\$x is less than or equal to \$y

# Relational /Comparison operator example

---

.....

<body>

<?php

\$x = 10;

\$y = 6;

if(\$x==\$y){

echo "Both are equal";

}else{

echo "Not equal";

}

?>

</body>

# Increment/Decrement operator

---

- They are used to increment or decrement value of variable by 1

Operator	Name	Example	Returns TRUE if
++\$x	Pre-increment	++\$x	Increment \$x by 1 and returns \$x
\$x++	Post-increment	\$x++	Returns \$x and increments \$x by 1
--\$x	Pre-decrement	--\$x	Decrements \$x by 1 and returns \$x
\$x--	Post-decrement	\$x--	Returns \$x and decrements \$x by 1

# Relational /Comparison operator example

---

.....

```
<body>
```

```
    <?php
```

```
        $x = 10;
```

```
        echo $x++;
```

```
    ?>
```

```
</body>
```

# Logical operators

---

- Logical operators are used to combine conditional statements

Operator	Name	Example	Returns TRUE if
&&	AND	\$x && \$y	True if both \$x and \$y are true
	OR	\$x    \$y	True if either \$x or \$y or both are true
!	NOT	!\$x	True if \$x is not true
XOR	XOR	\$x xor \$y	True if either \$x or \$y is true but not both

# Relational /Comparison operator example

---

.....

```
<body>
```

```
    <?php
```

```
        $x = 10;
```

```
        $y = 8;
```

```
        if($x < $y || $x > $y){
```

```
            echo "Not Equal";
```

```
        }
```

```
    ?>
```

```
</body>
```

# String operators

---

- String operators are specially designed for strings

Operator	Name	Example	Returns TRUE if
.	Concatenation	\$txt1.\$txt2	Concatenation of text \$txt1 and \$txt2
.=	Concatenation and assignment	\$txt1 .= \$txt2	Concatenation of text \$txt1 and \$txt2 and assigns to \$txt1

# String operator example

---

.....

```
<body>
```

```
    <?php
```

```
        $txt1 = "Hello";
```

```
        $txt2 = "Good Morning";
```

```
        $txt1 .= $txt2;
```

```
        echo $txt1;
```

```
    ?>
```

```
</body>
```

# Array operators

---

- Array operators are used to compare arrays

Operator	Name	Example	Returns TRUE if
+	Union	$\$x + \$y$	Union of $\$x$ and $\$y$
==	Equality	$\$x == \$y$	Returns true if $\$x$ and $\$y$ have same key/value
===	Identity	$\$x === \$y$	Returns true if $\$x$ and $\$y$ have same key/value pair in same order and of same type
!= or <>	Inequality	$\$x != \$y$	Returns true if $\$x$ is not equal to $\$y$
!==	Non identical	$\$x !== \$y$	Returns true if $\$x$ is not identical to $\$y$

# Array operator example

---

.....

```
<body>
```

```
    <?php
```

```
        $num1 = array(1,2,3);
```

```
        $num2 = array(2,3,4);
```

```
        if($num1 == $num2){
```

```
            echo "array are equal";
```

```
        }
```

```
    ?>
```

```
</body>
```

# Conditional operators

---

- Conditional operators are used to return a value depending on a condition
- Syntax:
  - Result = condition ? expression1 : expression2

Operator	Name	Example	Returns TRUE if
? :	Conditional	<code>\$x &gt; \$y ? "X" : "Y"</code>	Returns expression1 if condition is true else expression2 is returned.

# Conditional operator example

---

.....

```
<body>
```

```
    <?php
```

```
        $x = 10;
```

```
        $y = 9;
```

```
        echo $x>$y ? "X is greater" : "Y is greater" ;
```

```
    ?>
```

```
</body>
```

# POST method

---

- POST method is super global array variable that is used to get values submitted through HTTP POST method
- The array variable can be accessed from any script in the program i.e. it has global scope
- It is secure method to pass variables from one page to another
- The data or values are encrypted or hidden
- Syntax:

```
<?php
    $_POST['variable_name'];
?>
```

# GET method

---

- GET method is super global array variable that is used to get values submitted through HTTP GET method
- The array variable can be accessed from any script in the program i.e. it has global scope
- It is insecure method of passing variable from one page to another
- Data & variables are sent along with URL. Only limited number of data can be sent. Visible in browser's URL bar
- Syntax:

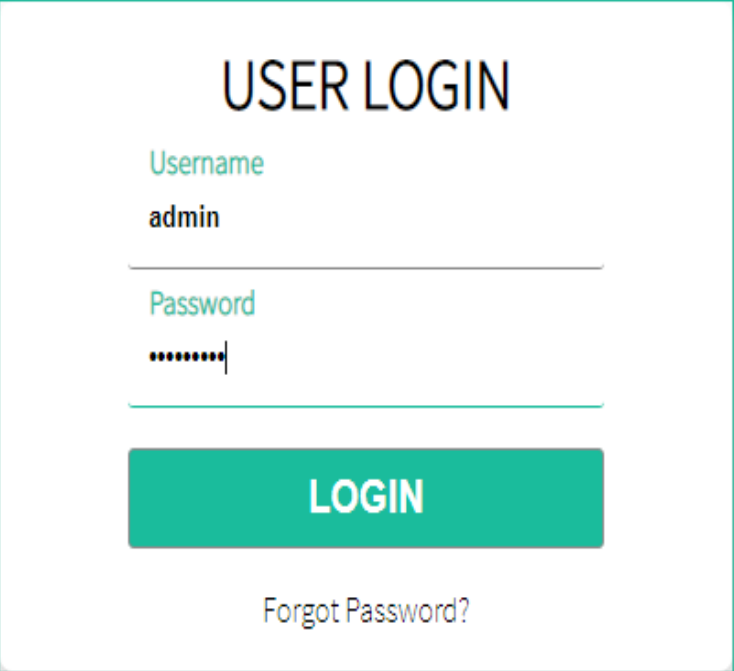
```
<?php
```

```
    $_GET['variable_name'];
```

```
?>
```

# Login page (login.html)

```
<html>
<head>
    <title>Login page</title>
</head>
<body>
    <h1> User Login </h1>
    <form action="home.php" method="GET|POST" >
        Username: <input type="text" name="uname"/>
        Password: <input type="text" name="upass"/>
        <input type="submit" value="login"/>
    </body>
</html>
```



USER LOGIN

Username  
admin

Password  
.....

LOGIN

[Forgot Password?](#)

# home.php (Using POST Method)

---

.....

```
<body>
```

```
    <?php
```

```
        echo $_POST['uname'];
```

```
        echo $_POST['upass'];
```

```
        if($_POST['uname'] == "admin" || $_POST['upass'] == "12345"){
```

```
            echo "Logged in Successful";
```

```
        }
```

```
    ?>
```

```
</body>
```

# home.php (Using GET Method)

---

.....

```
<body>
```

```
    <?php
```

```
        echo $_GET['uname'];
```

```
        echo $_GET['upass'];
```

```
        if($_GET['uname'] == "admin" || $_GET['upass'] == "12345"){
```

```
            echo "Logged in Successful";
```

```
        }
```

```
    ?>
```

```
</body>
```

# POST Method Vs GET Method

---

## GET METHOD

1. Only limited amount of data can be sent because data is sent in header or URL
2. GET request is not secured because data is exposed in URL bar
3. GET request can be bookmarked
4. GET is essentially used for fetching the information
5. It can be cached
6. GET method is the default method if not specified in the form

## POST METHOD

1. Large amount of data can be sent because data is sent in body
2. POST request is secured because data is not exposed in URL bar
3. POST request cannot be bookmarked
4. The purpose of POST method is to update the data
5. It cannot be cached
6. POST method must be specified in the form. It is not default method

# PHP Program 1

---

1. Write a PHP code to enter your name and display it

```
<body>
```

```
<form method="POST">
```

```
Enter your name: <input type="text" name="fullname"/>
```

```
<input type="submit" value="enter"/>
```

```
</form>
```

```
<?php
```

```
    $name = $_POST['fullname'];
```

```
    echo "Your name is : ".$name;
```

```
?>
```

# PHP Program 2

---

1. Write a php code to display the factorial of a number given by user

```
<form method="POST">
```

```
Enter number: <input type="text" name="number"/>
```

```
<input type="submit" value="factorial"/>
```

```
</form>
```

```
<?php
```

```
    $n = $_POST['number'];
```

```
    $fact = 1;
```

```
    for($i=1; $i<=$n; $i++){
```

```
        $fact = $fact*$i;
```

```
    }
```

```
    echo "The factorial of $n = $fact";
```

```
?>
```

# PHP Program 3

---

1. Write a php code to display all even numbers upto 50

```
<?php
    for($i=2; $i<=50; $i=$i+2){
        echo "$i ";
    }
?>
```

# PHP Program 4

---

1. Write a php code to display the following patterns

```
<?php
```

```
    for($i=1 $i<=4; $i++){  
        for($j=1; $j<=i; $j++){  
            echo $j;  
        }  
        echo "<br/>";  
    }
```

```
?>
```

```
1  
1 2  
1 2 3  
1 2 3 4
```

# PHP Program 5

---

1. Write a PHP code to display multiplication table of number

```
<?php
```

```
    for($i=1; $i<=10; $i++){
```

```
        echo "<tr>";
```

```
        for($j=1; $j<=10; $j++){
```

```
            echo "<td>$i * $j = ".$i*$j."</td>";
```

```
        }
```

```
        echo "<tr/>";
```

```
    }
```

```
?>
```

# PHP Program 6

---

## 1. Write a PHP code to display simple interest

```
<?php
```

```
    $p = $_POST['p'];
```

```
    $t = $_POST['t'];
```

```
    $r = $_POST['r'];
```

```
    $interest = ($p * $t * $r)/100;
```

```
    echo "Simple interest is = $interest";
```

```
?>
```

# SQL Queries

---

- Structured Query Language (SQL) is the database language
- It performs operations on stored data such as inserting, updating, deleting, modifying tables, views etc
- To perform SQL queries we must have databases like: MySQL, FoxPro, PostGre, SQL Server, Oracle etc.
- SQL performs
  - Create new database and tables
  - Retrieve data from database
  - Insert Records in database
  - Update records in database
  - Delete records from database

# SQL Queries

---

## ■ MySQL Create Database

- To create new database we use CREATE DATABASE statement
- Syntax:
  - CREATE DATABASE database\_name;
  - Example: CREATE DATABASE dbClass;

## ■ MySQL Create Table

- To create new table within database we use CREATE TABLE statement
- Syntax:
  - CREATE TABLE table\_name (column\_name column\_type);
  - Example: CREATE TABLE tb\_students(  
    rollNo integer(10),  
    name varchar(20),  
    address varchar(20));

# SQL Queries

---

## ■ MySQL Insert

- To insert string data types, it is required to keep all the values into double or single quotes
- Syntax:
- `INSERT INTO table_name (column1, column2, ....) VALUES (value1, value2, ....);`
- Example: `INSERT INTO tb_students (rollNo, name, address) VALUES (1, "RAM", "Bhaktapur");`

## ■ MySQL Select

- Select statement is used to fetch data from database or table
- We can use one or more tables separated by comma to fetch data
- We can use WHERE clause to filter out data
- We can fetch single or more fields/columns with single SELECT command
- We can limit number of returns using LIMIT attribute
- `SELECT * FROM tbl_std WHERE age < 20 AND address="Kathmandu"`

# SQL Queries

---

## ■ MySQL Where clause

- Where clause is used along with SELECT, UPDATE,DELETE commands to filter out data and to specify selection criteria
- `SELECT field1, field2, .... FROM table_name1, table_name2,... WHERE condition1 AND/OR condition2 ...`
- `SELECT fullname, address FROM student_table WHERE class=12`

## ■ MySQL Delete

- Delete command is used to delete record or data from database/table
- If WHERE clause is not specified properly then all other records will also be deleted
- `DELETE FROM table_name WHERE conditions`
- `DELETE FROM student_table WHERE marks<40`

# SQL Queries

---

## ■ MySQL Update

- Update command is used to modify or change existing data or record in database
- Syntax
- `UPDATE table_name SET field1 = value1, field2 = value2 ..... WHERE conditions`
- `UPDATE student_table SET address="Bhaktapur" WHERE rollNo = '3'`

# SQL Commands Practice

1. Create database 'SCHOOL\_MANAGEMENT'
2. Create table 'student\_info' which has ROLLNO, FULLNAME, ADDRESS, CLASS, MARKS, STATUS
3. Write SQL command to add the any one records in 'student\_info'

RollNO	Fullname	Address	Class	Marks	Status
1	Laxman Shrestha	Pokhara	12	56	Active
2	Gita Sharma	Kathmandu	11	45	Active
3	Sita Baraili	Pokhara	11	87	Inactive
4	Dipendra Shahi	Kailali	12	65	Active
5	Hari Dhakal	Hetauda	11	23	Inactive

4. Display all records from table
5. Display fullname and address of students from Pokhara
6. List all active students who have secured more than 40
7. Change address to Kathmandu of Rollno 4
8. Delete the records who are inactive from class 11

# Database connection

---

## ■ **mysqli\_Connect()**

- This function opens a new connection to the MySQL server. MySQLi is extension to MySQL database
- It allows access to new functionalities found in MySQL system
- Syntax: `mysqli_connect(host, username, password, dbname, port, socket);`

Parameter	Description
Host	Specifies a host name or an IP Address
Username	Specifies MySQL username
Password	Specifies MySQL password
dbname	Specifies the default database to be used
Port	Optional: specifies port number
Socket	Optional: specifies socket

# Example

---

## ■ Login.html

```
<!DOCTYPE html>
<html>
<head>
  <title>User login</title>
</head>
<body>

  <form method="POST" action="process.php">
    <h1>User Login</h1>
    Username: <input type="text" name="uname"><br><br>
    Password: <input type="password" name="upass"><br><br>
    <input type="submit" name="btnSubmit" value="Login">
  </form>

</body>
</html>
```

# Example

---

## ■ process.php

```
<?php
$username = $_POST['uname'];
$userpass = $_POST['upass'];

$conn = new mysqli("localhost", "root", "", "dbuser");
if($conn){
    $sql = "SELECT * FROM user_info WHERE username = '". $username. "' AND password = '". $userpass. "'";

    $result = mysqli_query($conn,$sql);
    if(mysqli_num_rows($result)>0){
        echo "login Successful";

    }else{
        echo "Login Failed. No such Username and Password in database";
    }
}
else{
    echo "not connected";
}
?>
```

# Example

---

## ■ view.php

```
<!DOCTYPE html>
<html>
<head>
    <title>Diplay data from database</title>
</head>
<body>
<?php
$conn = new mysqli("localhost","root","","dbuser");
if($conn){
    $sql = "SELECT * FROM user_info";
    $result = mysqli_query($conn, $sql);
    .....
}
```

# Example

---

## ■ view.php

.....

```
        if(mysqli_num_rows($result)>0){
            while($row = mysql_fetch_array($result)){
                echo $row['uid'];
                echo $row['username'];
                echo $row['password'];
            }
        }
    }else{
        die("Connection failed");
    }
?>
</body>
</html>
```

# End of Unit