LAB – 1

Setting Environment for ServerSide Scripting

Available In:

https://www.highapproach.com/bim4th

https://www.asheshneupane.com.np/bim4th

LAB – 2

Write a basic PHP program and print "Hello World".

Source Code:

```
<?php
// This is a basic PHP program
echo "Hello World";
?>
```

Output:

```
Hello World
```


Write a PHP Source Code to print the sum of two numbers.

Source Code:

```
<?php
$a = 10;
$b = 20;
$sum = $a + $b;
echo "The sum is: " . $sum;
?>
```

Output:

```
The sum is: 30
```


Write a PHP script that demonstrates the use of comments and data types.

Source Code:

```
<?php
// This is a single-line comment
/* This is a multi-line comment
   showing how to use comments in PHP */
// String
$name = "Ashesh";
// Integer
$age = 20;
// Float
$marks = 75.5;
// Boolean
$passed = true;
// Array
$subjects = ["Basic Mathematics", "Foundation of Information Technology", "C Programming"];
// Displaying the values
echo "Name: $name <br>";
echo "Age: $age <br>";
echo "Marks: $marks <br>";
echo "Passed: $passed <br>";
echo "Subjects: $subjects[0], $subjects[1], $subjects[2]";
?>
```

Output:

```
Name: Ashesh
Age: 20
Marks: 75.5
Passed: 1
Subjects: Basic Mathematics, Foundation of Information Technology, C Programming
```


Write a PHP script to define a constant and show the difference between a constant and a variable.

Source Code:

```
<?php
// Defining a constant
define("COLLEGE", "Shanker Dev Campus");
// Defining a variable
$program = "BIM";
// Displaying the constant
echo "College (Constant): ".COLLEGE."<br>";
// Displaying the variable
echo "Program (Variable): ".$program."<br>";
?>
```

Output:

```
College (Constant): Shanker Dev Campus
Program (Variable): BIM
```


Write a PHP script that demonstrates different types of operators.

Source Code:

```
<?php
// Arithmetic Operators
$a = 10;
$b = 5;
echo "Arithmetic Operators:<br>";
echo "$a + $b = " . ($a + $b) . "<br>";
echo "$a - $b = " . ($a - $b) . "<br>";
echo "$a * $b = " . ($a * $b) . "<br>";
echo "$a / $b = " . ($a / $b) . "<br>";
echo "$a % $b = " . ($a % $b) . "<br><br>";
// Assignment Operators
$x = 10;
echo "Assignment Operators:<br>";
$x += 5;
echo "x += 5: $x<br>";
$x -= 3;
echo "x -= 3: $x<br><br>";
// Comparison Operators
$m = 20;
$n = 15;
echo "Comparison Operators:<br>";
echo "$m == $n: " . ($m == $n ? "True" : "False") . "<br>";
echo "$m > $n: " . ($m > $n ? "True" : "False") . "<br><br>";
// Logical Operators
$val1 = true;
$val2 = false;
echo "Logical Operators:<br>";
echo "val1 AND val2: " . (($val1 && $val2) ? "True" : "False") . "<br>";
echo "val1 OR val2: " . (($val1 || $val2) ? "True" : "False") . "<br>";
```

```
echo "NOT val1: " . ((!$val1) ? "True" : "False") . "<br><br>";

// Increment/Decrement Operators

$num = 5;

echo "Increment/Decrement Operators:<br>";

echo "Original value: $num<br>";

echo "Post-increment: " . $num++ . "<br>";

echo "After post-increment: $num<br>";

echo "Pre-increment: " . ++$num . "<br>";

?>
```

Output:

```
Arithmetic Operators:
10 + 5 = 15
10 - 5 = 5
10 * 5 = 50
10 / 5 = 2
10 % 5 = 0

Assignment Operators:
x += 5: 15
x -= 3: 12

Comparison Operators:
20 == 15: False
20 > 15: True

Logical Operators:
val1 AND val2: False
val1 OR val2: True
NOT val1: False

Increment/Decrement Operators:
Original value: 5
Post-increment: 5
After post-increment: 6
Pre-increment: 7
```


Write a PHP script to demonstrate the use of string literals in PHP.

Source Code:

```
<?php
// Using single-quoted string literal
$name = 'Ashesh';
echo 'Hello, my name is $name <br>'; // variables are not parsed
// Using double-quoted string literal
echo "Hello, my name is $name <br>"; // variables are parsed
// Escaping characters
echo 'It\'s a sunny day! <br>';
echo "Nishant said, \"I like eating momo!\" <br>";
?>
```

Output:

```
Hello, my name is $name
Hello, my name is Ashesh
It's a sunny day!
Nishant said, "I like eating momo!"
```


Write a PHP script for variable interpolation.

Source Code:

```
<?php
$name = "Ashesh";
$course = "BIM";
// Variable Interpolation using double quotes
echo "Hello, my name is $name and I study $course.<br>";
// Without interpolation (single quotes)
echo 'Hello, my name is $name and I study $course.<br>';
?>
```

Output:

```
Hello, my name is Ashesh and I study BIM.
Hello, my name is $name and I study $course.
```


Write a program in PHP for levels of error.

Source Code:

```
<?php
// Show all types of errors
error_reporting(E_ALL);
ini_set("display_errors", 1);
// Notice Error: Using a variable without defining it
echo $name; // $name is not defined
// Warning Error: Including a file that doesn't exist
include("nishantdon.php");
// Fatal Error:
testFunction(); // This function doesn't exist
?>
```

[Output is not necessary for this program]

Output for educational purpose:

Notice: Undefined variable: name in /home/D8vJ42/prog.php on line 6 Warning: include(nishantdon.php): failed to open stream: No such file or directory in /home/D8vJ42/prog.php on line 8 Warning: include(): Failed opening 'nishantdon.php' for inclusion (include_path='.:/usr/share/php') in /home/D8vJ42/prog.php on line 8 Fatal error: Uncaught Error: Call to undefined function testFunction() in /home/D8vJ42/prog.php:10 Stack trace: #0 {main} thrown in /home/D8vJ42/prog.php on line 10

Write a PHP script to illustrate control statements (conditional and iterative statements).

Source Code:

```
<?php
// Conditional Statement: if-else
$marks = 70;
echo "Marks: $marks<br>";
if ($marks >= 40) {
    echo "Result: Pass<br>";
} else {
    echo "Result: Fail<br>";
}
// Iterative Statement: for loop
echo "Counting from 1 to 5 using for loop:<br>";
for ($i = 1; $i <= 5; $i++) {
    echo $i . "<br>";
}
?>
```

Output:

```
Marks: 70
Result: Pass
Counting from 1 to 5 using for loop:
1
2
3
4
5
```


Define a PHP function that takes two arguments and returns their sum. Demonstrate how to call this function with sample arguments.

Source Code:

```
<?php
// Function definition
function addNumbers($a, $b) {
    return $a + $b;
}
// Calling the function with sample arguments
$sum1 = addNumbers(5, 10);
$sum2 = addNumbers(20, 30);
// Displaying the results
echo "Sum of 5 and 10 is: $sum1 <br>";
echo "Sum of 20 and 30 is: $sum2 <br>";
?>
```

Output:

```
Sum of 5 and 10 is: 15
Sum of 20 and 30 is: 50
```


Write a PHP script to illustrate templating functions in detail.

Source Code:

```
<?php
// Header Template Function
function showHeader() {
    echo "<h2>Welcome to My Website</h2>";
    echo "<hr>";
}
// Footer Template Function
function showFooter() {
    echo "<hr>";
    echo "<p>&copy; 2025 Ashesh Neupane</p>";
}
// Content Function
function showContent($pageTitle, $message) {
    echo "<h3>$pageTitle</h3>";
    echo "<p>$message</p>";
}
// Using the template functions
showHeader();
showContent("ASHESH NEUPANE", "Welcome to ASHESHNEUPANE.COM.NP");
showFooter();
?>
```

Output:

Welcome to My Website

ASHESH NEUPANE

Welcome to ASHESHNEUPANE.COM.NP

© 2025 Ashesh Neupane

LAB – 3

Write a PHP code to display debugger function.

Source Code:

```
<?php
// Sample variables
$name = "Ashesh";
$age = 21;
$marks = [90, 85, 80];
// Using var_dump() – shows data type and value
echo "Using var_dump():<br>";
var_dump($name);
echo "<br>";
var_dump($age);
echo "<br><br>";
// Using print_r() – shows array structure
echo "Using print_r():<br>";
print_r($marks);
?>
```

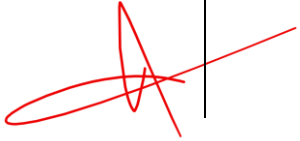
Output:

```
Using var_dump():
string(6) "Ashesh"
int(21)

Using print_r():
Array ( [0] => 90 [1] => 85 [2] => 80 )
```


Write a program in PHP to create a function called calculate which calculates the area, perimeter, and volume of a rectangle. Make assumptions for the parameters. When the function is called as calculate(10, 20, 'area'), it should return an output like "The area of a rectangle is: 200", and so on for the rest.

Source Code:

```
<?php
function calculate($length, $breadth, $type)
{
    if ($type == 'area')
    {
        $area = $length * $breadth;
        echo "The area of a rectangle is: $area";
    }
    else if ($type == 'perimeter')
    {
        $perimeter = 2 * ($length + $breadth);
        echo "The perimeter of a rectangle is: $perimeter";
    }
    else if ($type == 'volume')
    {
        $height = 10; // assumed height
        $volume = $length * $breadth * $height;
        echo "The volume of a rectangle is: $volume";
    }
    else
    {
        echo "Invalid type requested.";
    }
}
// Example function calls
calculate(10, 20, 'area');
echo "<br>";
```



```
calculate(10, 20, 'perimeter');

echo "<br>";

calculate(10, 20, 'volume');

?>
```

Output:

```
The area of a rectangle is: 200
The perimeter of a rectangle is: 60
The volume of a rectangle is: 2000
```


Write a function that accepts string as an argument and convert it to uppercase leaving the first letter as small.

Source Code:

```
<?php
function customUppercase($text)
{
    $first = strtolower(substr($text, 0, 1)); // Make first letter lowercase
    $rest = strtoupper(substr($text, 1));     // Make rest of the string uppercase
    return $first . $rest;
}
echo customUppercase("ashesh");
?>
```

Output:

```
aSHESH
```


Write a PHP code to illustrate default arguments in functions.

Source Code:

```
<?php
function greet($name = "Guest")
{
    echo "Namaste, $name!<br>";
}
greet("Ashesh");
greet();
?>
```

Output:

```
Namaste, Ashesh!
Namaste, Guest!
```


Write a PHP script for following string functions:

a. strlen() b. strrev() c. strpos() d. strtolower() e. strtoupper() f. ucwords() g. ucfirst() h. lcfirst() i. substr() j. str_replace() k. strncasecmp() l. stripslashes() m. htmlspecialchars() n. htmlentities() o. trim() p. strip_tags()

Source Code:

```
<?php
$text = "Ashesh";
$html = "<b>Ashesh</b> & <i>Coder</i>";
// a
echo "a. " . strlen($text) . "<br>";
// b
echo "b. " . strrev($text) . "<br>";
// c
echo "c. " . strpos($text, "sh") . "<br>";
// d
echo "d. " . strtolower($text) . "<br>";
// e
echo "e. " . strtoupper($text) . "<br>";
// f
echo "f. " . ucwords($text) . "<br>";
// g
echo "g. " . ucfirst($text) . "<br>";
// h
echo "h. " . lcfirst("ASHESH") . "<br>";
// i
echo "i. " . substr($text, 0, 3) . "<br>";
// j
echo "j. " . str_replace("sh", "**", $text) . "<br>";
// k
echo "k. " . strncasecmp("Ash", "ash", 3) . "<br>";
// l
```

```
echo "l. " . stripslashes("This is \\Ashesh\\") . "<br>";
// m
echo "m. " . htmlspecialchars("<Ashesh & Code>") . "<br>";
// n
echo "n. " . htmlentities("<Ashesh & Code>") . "<br>";
// o
echo "o. '" . trim("  Ashesh  ") . "'<br>";
// p
echo "p. " . strip_tags($html) . "<br>";
?>
```

Output:

```
a. 6
b. hsehsA
c. 1
d. ashesh
e. ASHESH
f. Ashesh
g. Ashesh
h. aSHESH
i. Ash
j. A**e**
k. 0
l. This is Ashesh
m. <Ashesh & Code>
n. <Ashesh & Code>
o. 'Ashesh'
p. Ashesh & Coder
```


LAB – 4

Create a numeric array called $months that contains the months of the year.

Source Code:

```
<?php
$months = array(
    "January", "February", "March", "April",
    "May", "June", "July", "August",
    "September", "October", "November", "December"
);
// Display using for loop
for ($i = 0; $i < count($months); $i++)
{
    echo $months[$i] . "<br>";
}
?>
```

Output:

```
January
February
March
April
May
June
July
August
September
October
November
December
```


Use array( ) to create an associative array of months and the number of days in each month and display the $months array.

Source Code:

```
<?php
// Create associative array of months and their days
$months = array(
    "January" => 31,
    "February" => 28,
    "March" => 31,
    "April" => 30,
    "May" => 31,
    "June" => 30,
    "July" => 31,
    "August" => 31,
    "September" => 30,
    "October" => 31,
    "November" => 30,
    "December" => 31
);
// Display the array
foreach ($months as $month => $days)
{
    echo "$month has $days days.<br>";
}
?>
```


Output:

```
January has 31 days.
February has 28 days.
March has 31 days.
April has 30 days.
May has 31 days.
June has 30 days.
July has 31 days.
August has 31 days.
September has 30 days.
October has 31 days.
November has 30 days.
December has 31 days.
```


Write a program in PHP to create a function that takes an array as a parameter and returns the average value of that array.

Source Code:

```
<?php
function findAverage($numbers)
{
    $sum = array_sum($numbers);
    $count = count($numbers);
    if ($count == 0)
    {
        return 0;
    }
    return $sum / $count;
}
$data = array(10, 20, 30, 40, 50);
$average = findAverage($data);
echo "The average is: " . $average;
?>
```

Output:

```
The average is: 30
```


Write a PHP program to print the following List in array and display as follows:

a. Bachelor in Information Management

b. Bachelor in Business Administration

c. Bachelor in Business Studies

d. Bachelor in Business Management

Source Code:

```
<?php
$courses = array(
    "Bachelor in Information Management",
    "Bachelor in Business Administration",
    "Bachelor in Business Studies",
    "Bachelor in Business Management"
);
echo "<ol type='a'>";
for ($i = 0; $i < count($courses); $i++)
{
    echo "<li>" . $courses[$i] . "</li>";
}
echo "</ol>";
?>
```

Output:

a. Bachelor in Information Management
b. Bachelor in Business Administration
c. Bachelor in Business Studies
d. Bachelor in Business Management

Write a PHP program to create an array containing information of four persons (name, age, and gender) as given below:

- ***Ashesh Neupane, 20, Male***
- ***Nishant Lamichhane, 21, Male***
- ***Bishow Raj Saru Magar, 19, Male***
- ***Sanyam Kothari, 19, Male***

Display this information in a tabular format with columns S.No., Name, Age, and Gender.

Source Code:

```
<?php
$people = array(
    array("name" => "Ashesh Neupane", "age" => 20, "gender" => "Male"),
    array("name" => "Nishant Lamichhane", "age" => 21, "gender" => "Male"),
    array("name" => "Bishow Raj Saru Magar", "age" => 19, "gender" => "Male"),
    array("name" => "Sanyam Kothari", "age" => 19, "gender" => "Male")
);
echo "<table border='1' cellpadding='5' cellspacing='0' style='border-collapse: collapse; width: 60%;'>";
echo "<tr>
        <th>S.No.</th>
        <th>Name</th>
        <th>Age</th>
        <th>Gender</th>
      </tr>";
for ($i = 0; $i < count($people); $i++) {
    echo "<tr>";
    echo "<td>" . ($i + 1) . "</td>";
    echo "<td>" . $people[$i]['name'] . "</td>";
    echo "<td>" . $people[$i]['age'] . "</td>";
    echo "<td>" . $people[$i]['gender'] . "</td>";
    echo "</tr>";
}
```

```
echo "</table>";
?>
```

Output:

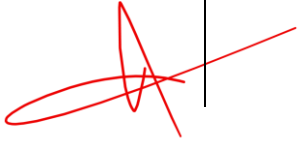
S.No.	Name	Age	Gender
1	Ashesh Neupane	20	Male
2	Nishant Lamichhane	21	Male
3	Bishow Raj Saru Magar	19	Male
4	Sanyam Kothari	19	Male

LAB – 5

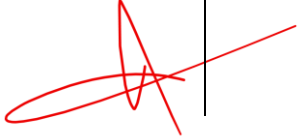
Write a PHP program using object-oriented programming to define a class called Book with attributes such as title, author, ISBN, and status (which can be either "available" or "borrowed"). The class should also have a method to display the details of the book. Additionally, define another class called Member with attributes such as name, membership ID, and a list of borrowed books. This class should include a method to display the member's details and another method to add or remove books from the borrowed book list.

Source Code:

```
<?php
class Book
{
    public $title;
    public $author;
    public $isbn;
    public $status;
    function __construct($title, $author, $isbn, $status = "Available")
    {
        $this->title = $title;
        $this->author = $author;
        $this->isbn = $isbn;
        $this->status = $status;
    }
    function displayDetails()
    {
        echo "Title: $this->title<br>";
        echo "Author: $this->author<br>";
        echo "ISBN: $this->isbn<br>";
        echo "Status: $this->status<br><br>";
    }
}
class Member
```



```
{
    public $name;
    public $membershipID;
    public $borrowedBooks = [];
    function __construct($name, $membershipID)
    {
        $this->name = $name;
        $this->membershipID = $membershipID;
    }
    function displayMember()
    {
        echo "Member Name: $this->name<br>";
        echo "Membership ID: $this->membershipID<br>";
        echo "Borrowed Books: ";
        if (empty($this->borrowedBooks))
        {
            echo "None<br><br>";
        }
        else
        {
            foreach ($this->borrowedBooks as $book)
            {
                echo $book->title . "<br>";
            }
            echo "<br>";
        }
    }
    function borrowBook($book)
    {
        if ($book->status == "Available")
        {
            $this->borrowedBooks[] = $book;
```

```
        $book->status = "Borrowed";
        echo "$book->title has been borrowed.<br><br>";
    }
    else
    {
        echo "$book->title is already borrowed.<br><br>";
    }
}
function returnBook($book)
{
    $found = false;
    $newList = [];
    foreach ($this->borrowedBooks as $b)
    {
        if ($b->title == $book->title)
        {
            $found = true; // Book found
        } else {
            $newList[] = $b; // Keep other books
        }
    }
    if ($found)
    {
        $this->borrowedBooks = $newList;
        $book->status = "Available";
        echo "$book->title has been returned.<br><br>";
    }
    else
    {
        echo "Error: $book->title was not borrowed by $this->name.<br><br>";
    }
}
```

```
}
// Create books
$book1 = new Book("Financial Accounting", "Bhoj Raj Ojha", "1234567890");
$book2 = new Book("Business Statistics Using Excel", "Keshar Singh Khati", "0987654321", "Borrowed");
// Display book details
$book1->displayDetails();
$book2->displayDetails();
// Create a member
$member1 = new Member("Ashesh Neupane", "01/080");
// Member tries to borrow both books
$member1->borrowBook($book1);
$member1->borrowBook($book2);
// Display updated member and book info
$member1->displayMember();
$book1->displayDetails();
$book2->displayDetails();
// Attempt to return book2 (which was never borrowed by member1)
$member1->returnBook($book2);
// Return book1 correctly
$member1->returnBook($book1);
// Display again
$member1->displayMember();
$book1->displayDetails();
$book2->displayDetails();
?>
```

Output:

```
Title: Financial Accounting
Author: Bhoj Raj Ojha
ISBN: 1234567890
Status: Available

Title: Business Statistics Using Excel
Author: Keshar Singh Khati
ISBN: 0987654321
Status: Borrowed
```

Financial Accounting has been borrowed.

Business Statistics Using Excel is already borrowed.

Member Name: Ashesh Neupane
Membership ID: 01/080
Borrowed Books: Financial Accounting

Title: Financial Accounting
Author: Bhoj Raj Ojha
ISBN: 1234567890
Status: Borrowed

Title: Business Statistics Using Excel
Author: Keshar Singh Khati
ISBN: 0987654321
Status: Borrowed

Error: Business Statistics Using Excel was not borrowed by Ashesh Neupane.

Financial Accounting has been returned.

Member Name: Ashesh Neupane
Membership ID: 01/080
Borrowed Books: None

Title: Financial Accounting
Author: Bhoj Raj Ojha
ISBN: 1234567890
Status: Available

Title: Business Statistics Using Excel
Author: Keshar Singh Khati
ISBN: 0987654321
Status: Borrowed

Write a PHP program using object-oriented programming to define a class called Pet with attributes such as name, species, age, and price. Include a constructor to initialize these attributes when a Pet object is created. Also define a destructor to display a message when a pet is removed from the system. Then, create multiple Pet objects and demonstrate adding them to a pet store's inventory. Display the details of all pets available in the store.

Source Code:

```
<?php
class Pet {
    public $name;
    public $species;
    public $age;
    public $price;
    // Constructor to initialize attributes
    function __construct($name, $species, $age, $price) {
        $this->name = $name;
        $this->species = $species;
        $this->age = $age;
        $this->price = $price;
        echo "Pet added: $this->name ($this->species)<br>";
    }
    // Destructor to display message when object is destroyed
    function __destruct() {
        echo "Pet removed from system: $this->name ($this->species)<br>";
    }
    // Method to display pet details
    function display() {
        echo "Name: $this->name<br>";
        echo "Species: $this->species<br>";
        echo "Age: $this->age years<br>";
        echo "Price: Rs. $this->price<br><br>";
    }
```

```
}
// Pet store inventory
$inventory = [];
// Create and add pets to the inventory
$inventory[] = new Pet("Buddy", "Dog", 2, 5000);
$inventory[] = new Pet("Mittens", "Cat", 1, 3000);
$inventory[] = new Pet("Goldie", "Fish", 1, 500);
// Display pet store inventory
echo "<h3>Pet Store Inventory:</h3>";
foreach ($inventory as $pet) {
    $pet->display();
}
?>
```

Output:

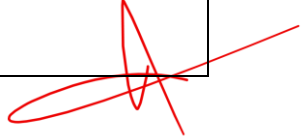
```
Pet added: Buddy (Dog)
Pet added: Mittens (Cat)
Pet added: Goldie (Fish)

Pet Store Inventory:

Name: Buddy
Species: Dog
Age: 2 years
Price: Rs. 5000

Name: Mittens
Species: Cat
Age: 1 years
Price: Rs. 3000

Name: Goldie
Species: Fish
Age: 1 years
Price: Rs. 500

Pet removed from system: Buddy (Dog)
Pet removed from system: Mittens (Cat)
Pet removed from system: Goldie (Fish)
```

Write a PHP program using object-oriented programming to create a base class called Person that has protected attributes name and email. Include methods to get and set these attributes. From this base class, derive two classes: Student and Staff. The Student class should include additional attributes such as studentID and course, while the Staff class should include employeeID and department. Use appropriate access modifiers to protect sensitive information and ensure proper encapsulation. Demonstrate creating objects of both derived classes and display their details.

Source Code:

```
<?php
class Person {
    protected $name;
    protected $email;
    public function setName($name) {
        $this->name = $name;
    }
    public function getName() {
        return $this->name;
    }
    public function setEmail($email) {
        $this->email = $email;
    }
    public function getEmail() {
        return $this->email;
    }
}
class Student extends Person {
    private $studentID;
    private $course;
    public function __construct($name, $email, $studentID, $course) {
        $this->setName($name);
        $this->setEmail($email);
```

```
        $this->studentID = $studentID;
        $this->course = $course;
    }
    public function displayStudent() {
        echo "Student Name: " . $this->getName() . "<br>";
        echo "Email: " . $this->getEmail() . "<br>";
        echo "Student ID: " . $this->studentID . "<br>";
        echo "Course: " . $this->course . "<br><br>";
    }
}
class Staff extends Person {
    private $employeeID;
    private $department;
    public function __construct($name, $email, $employeeID, $department) {
        $this->setName($name);
        $this->setEmail($email);
        $this->employeeID = $employeeID;
        $this->department = $department;
    }
    public function displayStaff() {
        echo "Staff Name: " . $this->getName() . "<br>";
        echo "Email: " . $this->getEmail() . "<br>";
        echo "Employee ID: " . $this->employeeID . "<br>";
        echo "Department: " . $this->department . "<br><br>";
    }
}
$student1 = new Student("Bishow Raj Saru Magar", "bishowmagar16299@gmail.com", "16299", "BIM");
$student1->displayStudent();
$staff1 = new Staff("Ashesh Neupane", "ashesh@highapproach.com", "COA02", "Office of the Administrator,
HighApproach");
$staff1->displayStaff();
?>
```



Output:

Student Name: Bishow Raj Saru Magar
Email: bishowmagar16299@gmail.com
Student ID: 16299
Course: BIM

Staff Name: Ashesh Neupane
Email: ashesh@highapproach.com
Employee ID: COA02
Department: Office of the Administrator, HighApproach

Write a PHP program using object-oriented programming to define an interface IVehicle that includes methods such as start(), stop(), and fuelStatus(). Create an abstract class Vehicle that implements this interface and introduces a constant named FUEL_CAPACITY. Then, derive two concrete classes Car and Truck from Vehicle, where each class provides specific implementations for all the abstract methods and may also include additional features or behaviors unique to that particular vehicle type. Finally, create objects of Car and Truck and demonstrate calling their methods.

Source Code:

```
<?php
interface IVehicle {
    public function start();
    public function stop();
    public function fuelStatus();
}
abstract class Vehicle implements IVehicle {
    const FUEL_CAPACITY = 100; // in liters
    protected $fuelLevel;
    public function __construct($fuelLevel) {
        $this->fuelLevel = $fuelLevel;
    }
    public function fuelStatus() {
        echo "Fuel level: {$this->fuelLevel}L out of " . self::FUEL_CAPACITY . "L<br>";
    }
    abstract public function start();
    abstract public function stop();
}
class Car extends Vehicle {
    public function start() {
        echo "Car engine started.<br>";
    }
    public function stop() {
```

```
        echo "Car engine stopped.<br>";
    }
    public function playMusic() {
        echo "Playing music in the car.<br>";
    }
}
class Truck extends Vehicle {
    public function start() {
        echo "Truck engine roared to life.<br>";
    }
    public function stop() {
        echo "Truck engine shut down.<br>";
    }
    public function loadCargo($weight) {
        echo "Loaded $weight kg of cargo into the truck.<br>";
    }
}
$car = new Car(60);
$car->start();
$car->fuelStatus();
$car->playMusic();
$car->stop();
$truck = new Truck(80);
$truck->start();
$truck->fuelStatus();
$truck->loadCargo(1500);
$truck->stop();
?>
```


Output:

```
Car engine started.
Fuel level: 60L out of 100L
Playing music in the car.
Car engine stopped.
Truck engine roared to life.
Fuel level: 80L out of 100L
Loaded 1500 kg of cargo into the truck.
Truck engine shut down.
```


Write a PHP program using object-oriented programming to create a class called Calculator that includes methods for basic arithmetic operations: add, subtract, multiply, and divide. Implement exception handling in the divide method to catch division by zero errors and display a user-friendly error message. Demonstrate the use of these methods by performing sample calculations.

Source Code:

```
<?php
class Calculator
{
    public function add($a, $b)
    {
        return $a + $b;
    }
    public function subtract($a, $b)
    {
        return $a - $b;
    }
    public function multiply($a, $b)
    {
        return $a * $b;
    }
    public function divide($a, $b)
    {
        try
        {
          if ($b == 0)
            {
              throw new Exception("Error: Division by zero is not allowed.");
            }
            return $a / $b;
        }
        catch (Exception $e)
```

```
        {
            return $e->getMessage();
        }
    }
}
$calc = new Calculator();
echo "Addition (10 + 5): " . $calc->add(10, 5) . "<br>";
echo "Subtraction (10 - 5): " . $calc->subtract(10, 5) . "<br>";
echo "Multiplication (10 * 5): " . $calc->multiply(10, 5) . "<br>";
echo "Division (10 / 5): " . $calc->divide(10, 5) . "<br>";
echo "Division (10 / 0): " . $calc->divide(10, 0) . "<br>"; // triggers exception
?>
```

Output:

```
Addition (10 + 5): 15
Subtraction (10 - 5): 5
Multiplication (10 * 5): 50
Division (10 / 5): 2
Division (10 / 0): Error: Division by zero is not allowed.
```


LAB – 6

Write a program that creates a file and writes contents to it and display it.

Source Code:

```
<?php
// Step 1: Create or open a file
$file = fopen("asheshneupane.txt", "w");
// Step 2: Write content to the file
fwrite($file, "Hello, I am Ashesh Neupane and I would like to welcome you to ASHESHNEUPANE.COM.NP");
// Step 3: Close the file after writing
fclose($file);
// Step 4: Open the file again to read content
$file = fopen("asheshneupane.txt", "r");
// Step 5: Read the content and display it
$content = fread($file, filesize("asheshneupane.txt"));
echo "File Content: <br>";
echo $content;
// Step 6: Close the file
fclose($file);
?>
```

Output:

```
File Content:
Hello, I am Ashesh Neupane and I would like to welcome you to ASHESHNEUPANE.COM.NP
```


Write a program that creates a file name "test.txt" and append a content of 20 characters to it.

Source Code:

```
<?php
// Step 1: Open the file in append mode ('a')
$file = fopen("test.txt", "a");
// Step 2: Define a 20-character string
$content = "Hello from Ashesh12!"; // Exactly 20 characters
// Step 3: Write the content to the file
fwrite($file, $content . "\n");
// Step 4: Close the file
fclose($file);
echo "Content appended to test.txt successfully.";
?>
```

Output:

```
Content appended to test.txt successfully.
```


Write a program that writes and reads a ".json" file.

Source Code:

```
<?php
// Step 1: Create an array of data
$data = array(
    "name" => "Ashesh Neupane",
    "email" => "ashesh@highapproach.com",
    "website" => "asheshneupane.com.np"
);
// Step 2: Convert the array to JSON format
$jsonData = json_encode($data, JSON_PRETTY_PRINT);
// Step 3: Write JSON data to a file
file_put_contents("data.json", $jsonData);
// Step 4: Read the content from the JSON file
$readData = file_get_contents("data.json");
// Step 5: Convert JSON back to PHP array
$decodedData = json_decode($readData, true);
// Step 6: Display the data
echo "Data from JSON file:<br>";
echo "Name: " . $decodedData["name"] . "<br>";
echo "Email: " . $decodedData["email"] . "<br>";
echo "Website: " . $decodedData["website"];
?>
```

Output:

```
Data from JSON file:
Name: Ashesh Neupane
Email: ashesh@highapproach.com
Website: asheshneupane.com.np
```

Write a program that reads only a single line of data or text from the file.

Source Code:

```
<?php
// Step 1: Open the file in read mode
$file = fopen("asheshneupane.txt", "r");
// Step 2: Read a single line from the file
$line = fgets($file);
// Step 3: Display the line
echo "First line from file: " . $line;
// Step 4: Close the file
fclose($file);
?>
```

Output:

```
First line from file: Hello, I am Ashesh Neupane
```


Create a PHP script to allow users to upload an image file with size and extension restrictions while preserving a unique filename.

Source Code:

```
<?php
if (isset($_FILES['image'])) {
    $file = $_FILES['image'];
    // Allowed file extensions
    $allowed = ['jpg', 'jpeg', 'png', 'gif'];
    $maxSize = 2 * 1024 * 1024; // 2MB
    $filename = $file['name'];
    $filesize = $file['size'];
    $tmpname = $file['tmp_name'];
    // Get file extension
    $ext = strtolower(pathinfo($filename, PATHINFO_EXTENSION));
    // Check extension
    if (!in_array($ext, $allowed)) {
        echo "Only JPG, JPEG, PNG, and GIF files are allowed.";
        exit;
    }
    // Check size
    if ($filesize > $maxSize) {
        echo "File size exceeds 2MB.";
        exit;
    }
    // Create a unique name and upload
    $newName = uniqid("img_", true) . "." . $ext;
    $uploadDir = "uploads/";
    // Make sure uploads folder exists
    if (!is_dir($uploadDir)) {
        mkdir($uploadDir, 0777, true);
    }
```

```
    if (move_uploaded_file($tmpname, $uploadDir . $newName)) {
        echo "File uploaded successfully!<br>";
        echo "<img src='$uploadDir$newName' width='200'>";
    } else {
        echo "Error uploading file.";
    }
} else {
    echo "No file uploaded.";
}
?>
```


Write a program to append an array data to a json file.

Source Code:

```
<?php
// Step 1: Define new data to append
$newData = array(
    "name" => "Ashesh Neupane",
    "email" => "ashesh@highapproach.com",
    "website" => "asheshneupane.com.np"
);
// Step 2: Read existing data from the file (if it exists)
$filename = "data.json";
if (file_exists($filename)) {
    $json = file_get_contents($filename);
    $data = json_decode($json, true);
} else {
    $data = array(); // empty array if file doesn't exist
}
// Step 3: Append new data
$data[] = $newData;
// Step 4: Save back to JSON file
file_put_contents($filename, json_encode($data, JSON_PRETTY_PRINT));
// Step 5: Show success message
echo "Data has been appended to data.json successfully.";
?>
```

Output:

```
Data has been appended to data.json successfully.
```


Create an array that contains name, address, age and gender. Then write the array to the file name "user-info.csv".

Source Code:

```
<?php
// Step 1: Create an array with personal info
$userInfo = array("Ashesh Neupane", "Kathmandu", "21", "Male");
// Step 2: Open the CSV file for writing
$file = fopen("user-info.csv", "w");
// Step 3: Write the array to the file as a CSV row
fputcsv($file, $userInfo);
// Step 4: Close the file
fclose($file);
echo "Data has been written to user-info.csv successfully.";
?>
```

Output:

```
Data has been written to user-info.csv successfully.
```


Generate a program that creates a folder or directory "practice" and then write a dummy text inside the file "dummy.txt".

Source Code:

```
<?php
// Step 1: Define the folder name
$folderName = "practice";
// Step 2: Check if the folder exists, if not, create it
if (!file_exists($folderName)) {
    mkdir($folderName);
    echo "Folder 'practice' created successfully.<br>";
} else {
    echo "Folder 'practice' already exists.<br>";
}
// Step 3: Create and open the file inside the folder
$filePath = $folderName . "/dummy.txt";
$file = fopen($filePath, "w");
// Step 4: Write dummy content to the file
fwrite($file, "This is some dummy text written by Ashesh Neupane.");
// Step 5: Close the file
fclose($file);
echo "File 'dummy.txt' has been created and written inside the 'practice' folder.";
?>
```

Output:

```
Folder 'practice' created successfully.
File 'dummy.txt' has been created and written inside the 'practice' folder.
```


Write a program to copy the content of odd lines from source file to destination file.

Source Code:

```
<?php
// Step 1: Define file names
$sourceFile = "source.txt";
$destinationFile = "destination.txt";
// Step 2: Check if the source file exists
if (!file_exists($sourceFile)) {
    die("Source file does not exist.");
}
// Step 3: Read all lines from the source file
$lines = file($sourceFile); // returns an array of lines
// Step 4: Open destination file in write mode
$dest = fopen($destinationFile, "w");
// Step 5: Loop through each line and copy only odd lines
foreach ($lines as $index => $line) {
    // Line numbers start from 0, so we check for even index
    if ($index % 2 == 0) {
        fwrite($dest, $line);
    }
}
// Step 6: Close the file
fclose($dest);
echo "Odd lines copied to '$destinationFile' successfully.";
?>
```

Output:

```
Odd lines copied to 'destination.txt' successfully.
```

LAB – 7

Write a PHP program which contains only an input and accepts only your first name. It should check for the correct name format and should not accept characters or spaces.

Source Code:

```
<!DOCTYPE html>
<html>
<head>
   <title>First Name Input</title>
</head>
<body>
   <form method="post">
      Enter First Name: <input type="text" name="fname">
      <input type="submit" value="Submit">
   </form>
   <?php
   if($_SERVER["REQUEST_METHOD"] == "POST"){
      $fname = $_POST["fname"];
      // Allow only alphabets (no spaces, no numbers, no symbols)
      if(preg_match("/^[A-Za-z]+$/", $fname)){
         echo "Valid First Name: " . $fname;
      } else {
         echo "Invalid input! Only alphabets allowed without spaces.";
      }
   }
   ?>
</body>
</html>
```


Write a PHP program which validates Nepali mobile phone number.

Source Code:

```
<!DOCTYPE html>
<html>
<head>
   <title>Nepali Mobile Number Validation</title>
</head>
<body>
   <form method="post">
      Enter Mobile Number: <input type="text" name="mobile">
      <input type="submit" value="Submit">
   </form>
   <?php
   if ($_SERVER["REQUEST_METHOD"] == "POST") {
      $mobile = $_POST["mobile"];
      // Must start with 97 or 98 and be exactly 10 digits
      if (preg_match("/^(97|98)[0-9]{8}$/", $mobile)) {
         echo "Valid Nepali Mobile Number: " . $mobile;
      } else {
         echo "Invalid Number! Must be 10 digits starting with 97 or 98.";
      }
   }
   ?>
</body>
</html> .
```

Write a PHP program to create a form that contains two textboxes which accepts numbers and a submit button. When a submit button is clicked, sum of two numbers will be displayed.

Source Code:

```
<!DOCTYPE html>
<html>
<head>
   <title>Sum of Two Numbers</title>
</head>
<body>
   <form method="post">
      Enter First Number: <input type="text" name="num1"><br><br>
      Enter Second Number: <input type="text" name="num2"><br><br>
      <input type="submit" value="Calculate Sum">
   </form>
   <?php
   if ($_SERVER["REQUEST_METHOD"] == "POST") {
      $num1 = $_POST["num1"];
      $num2 = $_POST["num2"];
      $sum = $num1 + $num2;
      echo "The Sum of $num1 and $num2 is: " . $sum;
   }
   ?>
</body>
</html>
```


Write a PHP program which validates checkboxes. It should contain 5 boxes; when a submit button is clicked in a specific or multiple checkbox it should display their names. (Note: empty fields are not allowed).

Source Code:

```
<!DOCTYPE html>
<html>
<head>
   <title>Checkbox Validation</title>
</head>
<body>
   <form method="post">
      <h3>Select Your Favorite Subjects:</h3>
      <input type="checkbox" name="subject[]" value="Math"> Math <br>
      <input type="checkbox" name="subject[]" value="Science"> Science <br>
      <input type="checkbox" name="subject[]" value="English"> English <br>
      <input type="checkbox" name="subject[]" value="Computer"> Computer <br>
      <input type="checkbox" name="subject[]" value="Economics"> Economics <br><br>
      <input type="submit" value="Submit">
   </form>
   <?php
   if ($_SERVER["REQUEST_METHOD"] == "POST") {
      if (!empty($_POST["subject"])) {
         echo "You selected:<br>";
         foreach ($_POST["subject"] as $sub) {
            echo $sub . "<br>";
         }
      } else {
         echo "<span style='color:red;'>Please select at least one option!</span>";
      }
   }
   ?>
</body>
```

```
</html>
```

Write a PHP program which validates select boxes. It should contain 5 options; when a submit button is clicked in a specific or multiple select boxes it should display their names. (Note: empty fields are not allowed).

Source Code:

```
<!DOCTYPE html>
<html>
<head>
   <title>Select Box Validation</title>
</head>
<body>
   <form method="post">
      <h3>Select Your Favorite Subjects:</h3>
      <select name="subject[]" multiple size="5">
         <option value="Math">Math</option>
         <option value="Science">Science</option>
         <option value="English">English</option>
         <option value="Computer">Computer</option>
         <option value="Economics">Economics</option>
      </select><br><br>
      <input type="submit" value="Submit">
   </form>
   <?php
   if ($_SERVER["REQUEST_METHOD"] == "POST") {
      if (!empty($_POST["subject"])) {
         echo "You selected:<br>";
         foreach ($_POST["subject"] as $sub) {
            echo $sub . "<br>";
         }
      } else {
         echo "<span style='color:red;'>Please select at least one option!</span>";
```

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

Write a PHP program that validates a name and email field in a form. Name and email should be in a strict format. Also, both the fields should be validated against spaces, special characters, unwanted html tags and unwanted html backslashes(\).

Source Code:

```
<!DOCTYPE html>
<html>
<head>
  <title>Name and Email Validation</title>
</head>
<body>
  <form method="post">
    Name: <input type="text" name="name"><br><br>
    Email: <input type="text" name="email"><br><br>
    <input type="submit" value="Submit">
  </form>
  <?php
  if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Function to clean input
    function clean_input($data) {
      $data = trim($data);            // Remove extra spaces
      $data = stripslashes($data);    // Remove backslashes \
      $data = strip_tags($data);      // Remove HTML tags
      $data = htmlspecialchars($data); // Convert special characters
      return $data;
    }
```

```
    $name = clean_input($_POST["name"]);
    $email = clean_input($_POST["email"]);
    // Validate name (only alphabets, no digits or special chars, no spaces)
    if (!preg_match("/^[A-Za-z]+$/", $name)) {
        echo "<p style='color:red;'>Invalid Name! Only alphabets allowed without spaces.</p>";
    }
    // Validate email (standard email format)
    elseif (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
        echo "<p style='color:red;'>Invalid Email Format!</p>";
    }
    else {
        echo "<p style='color:green;'>Valid Input:</p>";
        echo "Name: " . $name . "<br>";
        echo "Email: " . $email;
    }
}
?>
</body>
</html>
```


LAB – 8

Make following web page with two text boxes (Name and Age) and a button (OK). As button is clicked values given in textboxes should be inserted in database. (For database make your own assumptions)

Source Code:

```
<!DOCTYPE html>
<html>
<head>
  <title>Input Form</title>
</head>
<body>
  <h2>Input Form</h2>
  <form method="post">
    Name: <input type="text" name="name" required><br><br>
    Age: <input type="number" name="age" required><br><br>
    <input type="submit" name="submit" value="OK">
  </form>
  <?php
  if (isset($_POST['submit'])) {
    // Database connection
    $servername = "localhost";
    $username = "root";   // default username in XAMPP/WAMP
    $password = "";       // default password is empty
    $dbname = "student_db"; // assumed database
    // Create connection
    $conn = mysqli_connect($servername, $username, $password, $dbname);
    // Check connection
    if (!$conn) {
      die("Connection failed: " . mysqli_connect_error());
    }
```

```
    // Get values from form
    $name = $_POST['name'];
    $age = $_POST['age'];
    // Insert into database
    $sql = "INSERT INTO students (name, age) VALUES ('$name', '$age')";
    if (mysqli_query($conn, $sql)) {
        echo "<p style='color:green;'>Record inserted successfully!</p>";
    } else {
        echo "Error: " . mysqli_error($conn);
    }
    mysqli_close($conn);
}
?>
</body>
</html>
```


Create a login system, making your own assumptions about database and authorize the login credentials. (Hint: Only the user in database should be allowed to login)

Source Code:

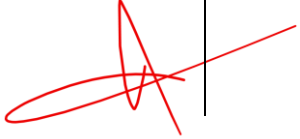
```
<!DOCTYPE html>
<html>
<head>
   <title>Login System</title>
</head>
<body>
   <h2>Login Form</h2>
   <form method="post">
      Username: <input type="text" name="username" required><br><br>
      Password: <input type="password" name="password" required><br><br>
      <input type="submit" name="login" value="Login">
   </form>
   <?php
   if (isset($_POST['login'])) {
      // Database connection
      $servername = "localhost";
      $username = "root";    // default
      $password = "";        // default
      $dbname = "user_db";   // assumed database
      $conn = mysqli_connect($servername, $username, $password, $dbname);
      if (!$conn) {
         die("Connection failed: " . mysqli_connect_error());
      }
      // Get form data
      $user = $_POST['username'];
      $pass = $_POST['password'];
      // Check credentials
      $sql = "SELECT * FROM users WHERE username='$user' AND password='$pass'";
```

```
        $result = mysqli_query($conn, $sql);
        if (mysqli_num_rows($result) == 1) {
            echo "<p style='color:green;'>Login successful! Welcome, $user</p>";
        } else {
            echo "<p style='color:red;'>Invalid username or password!</p>";
        }
        mysqli_close($conn);
    }
    ?>
</body>
</html>
```


Write a PHP program to create a table in a predefined database. Table name should be 'dummy' and the table should contain (id, name, email, address and contact_number) as column fields. Make your own assumptions for each column data types.

Source Code:

```
<?php
// Database connection details
$servername = "localhost";
$username = "root";   // default username
$password = "";       // default password
$dbname = "test_db";  // predefined database
// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
// SQL query to create table
$sql = "CREATE TABLE dummy (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL,
    address VARCHAR(150),
    contact_number VARCHAR(15)
)";
// Execute query
if (mysqli_query($conn, $sql)) {
    echo "Table 'dummy' created successfully!";
} else {
    echo "Error creating table: " . mysqli_error($conn);
}
```



```
// Close connection
mysqli_close($conn);
?>
```


Write a PHP program that reads CSV file which contains (itemId, itemDesc, rate, qty) and insert them into a database. Assumption can be made if required.

Source Code:

```
<?php
// Database connection
$servername = "localhost";
$username = "root";   // default
$password = "";       // default
$dbname = "inventory_db";
// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
// Open CSV file
$filename = "items.csv";
if (($handle = fopen($filename, "r")) !== FALSE) {
    while (($data = fgetcsv($handle, 1000, ",")) !== FALSE) {
        // Reading each column
        $itemId = $data[0];
        $itemDesc = $data[1];
        $rate = $data[2];
        $qty = $data[3];
        // Insert into database
        $sql = "INSERT INTO items (itemId, itemDesc, rate, qty)
                VALUES ('$itemId', '$itemDesc', '$rate', '$qty')";
        if (mysqli_query($conn, $sql)) {
            echo "Inserted: $itemDesc<br>";
        } else {
            echo "Error: " . mysqli_error($conn) . "<br>";
        }
```

```
    }
    fclose($handle);
} else {
    echo "Error opening file.";
}
mysqli_close($conn);
?>
```


Assume that you have already created a database named "php_bim" and inside it, there is a table called "details". The table contains the following columns: "id", "name", and "address". The data in the table looks like this:

- ***Record 1: id = 1, name = Ram Nepali, address = Kathmandu***
- ***Record 2: id = 2, name = Shyam Nepali, address = Biratnagar***
- ***Record 3: id = 3, name = Sita Nepali, address = Hetauda***

Now, write a PHP program to delete the record with id = 2 (i.e., record number 2) from this table.

Source Code:

```
<?php
// Database connection details
$servername = "localhost";
$username   = "root";
$password   = "";
$dbname     = "php_bim";
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
// SQL query to delete record with id = 2
$sql = "DELETE FROM details WHERE id = 2";
if ($conn->query($sql) === TRUE) {
    echo "Record with id = 2 deleted successfully.";
} else {
    echo "Error deleting record: " . $conn->error;
}
// Close connection
$conn->close();
?>
```

Assume that you have already created a database named "php_bim" and inside it, there is a table called "details". The table contains the following columns: "id", "name", and "address". The data in the table looks like this:

- ***Record 1: id = 1, name = Ram Nepali, address = Kathmandu***
- ***Record 2: id = 2, name = Shyam Nepali, address = Biratnagar***
- ***Record 3: id = 3, name = Sita Nepali, address = Hetauda***

Now, write a PHP program to update the record with id = 1 (i.e., record number 1) from this table.

Source Code:

```
<?php
// Database connection details
$servername = "localhost";
$username   = "root";
$password   = "";
$dbname     = "php_bim";
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
// SQL query to update record with id = 1
$sql = "UPDATE details SET name = 'Ramesh Nepali', address = 'Pokhara' WHERE id = 1";
if ($conn->query($sql) === TRUE) {
    echo "Record with id = 1 updated successfully.";
} else {
    echo "Error updating record: " . $conn->error;
}
// Close connection
$conn->close();
?>
```

LAB – 9

Make a web page that uses a SESSION to keep track of how many times a user has viewed the page. The first time a particular user looks at the page, it should print something like "Number of views: 1." The second time the user looks at the page, it should print "Number of views: 2," and so on.

Source Code:

```
<?php
// Start the session
session_start();
// Check if 'views' session variable is set
if (isset($_SESSION['views'])) {
    $_SESSION['views'] = $_SESSION['views'] + 1;
} else {
    $_SESSION['views'] = 1; // First time user visits
}
?>
<!DOCTYPE html>
<html>
<head>
    <title>Page View Counter</title>
</head>
<body>
    <h2>Number of views: <?php echo $_SESSION['views']; ?></h2>
</body>
</html>
```


Write a Program to develop a full-fledged login and logout app using session and cookies.

Source Code:

```
<?php
session_start();
// Hardcoded username & password (can be from DB)
$valid_username = "admin";
$valid_password = "12345";
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = $_POST['username'];
    $password = $_POST['password'];
    if ($username === $valid_username && $password === $valid_password) {
        // Store in session
        $_SESSION['username'] = $username;
        // Set a cookie for 1 hour
        setcookie("username", $username, time() + 3600, "/");
        header("Location: dashboard.php");
        exit();
    } else {
        $error = "Invalid username or password!";
    }
}
?>
<!DOCTYPE html>
<html>
<head>
    <title>Login</title>
</head>
<body>
    <h2>Login Page</h2>
    <?php if (isset($error)) echo "<p style='color:red;'>$error</p>"; ?>
```

```
    <form method="POST" action="">
        Username: <input type="text" name="username" required><br><br>
        Password: <input type="password" name="password" required><br><br>
        <button type="submit">Login</button>
    </form>
</body>
</html>
```


Create a login form with appropriate credentials and the session should be expired after 5 hours, from the first time he/she has logged in.

Source Code:

login.php

```
<?php
session_start();
$valid_username = "admin";
$valid_password = "12345";
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = $_POST['username'];
    $password = $_POST['password'];
    if ($username === $valid_username && $password === $valid_password) {
        $_SESSION['username'] = $username;
        $_SESSION['start_time'] = time();
        $_SESSION['expire_time'] = $_SESSION['start_time'] + (5 * 60 * 60); // 5 hours = 18000 sec
        header("Location: dashboard.php");
        exit();
    } else {
        $error = "Invalid Username or Password!";
    }
}
?>
<!DOCTYPE html>
<html>
<head>
    <title>Login</title>
</head>
<body>
    <h2>Login Page</h2>
    <?php if (isset($error)) echo "<p style='color:red;'>$error</p>"; ?>
```

```
    <form method="POST" action="">
        <label>Username:</label>
        <input type="text" name="username" required><br><br>
        <label>Password:</label>
        <input type="password" name="password" required><br><br>
        <button type="submit">Login</button>
    </form>
</body>
</html>
```

dashboard.php

```
<?php
session_start();
// Check if session exists
if (!isset($_SESSION['username'])) {
    header("Location: login.php");
    exit();
}
// Check if session expired
if (time() > $_SESSION['expire_time']) {
    session_unset();
    session_destroy();
    echo " ⚠️ Session expired. Please <a href='login.php'>login again</a>.";
    exit();
}
?>
<!DOCTYPE html>
<html>
<head>
    <title>Dashboard</title>
</head>
<body>
    <h2>Welcome, <?php echo $_SESSION['username']; ?>!</h2>
```

```
    <p>You are logged in. Your session will expire after 5 hours.</p>
    <a href="logout.php">Logout</a>
</body>
</html>
```

logout.php

```
<?php
session_start();
session_unset();
session_destroy();
header("Location: login.php");
exit();
?>
```


Write a PHP code that illustrates "Basic HTTP Authentication".

Source Code:

```
<?php
// Set valid username and password
$valid_username = "admin";
$valid_password = "12345";
// Check if authentication headers are set
if (!isset($_SERVER['PHP_AUTH_USER']) || !isset($_SERVER['PHP_AUTH_PW'])) {
    // Send authentication headers
    header('WWW-Authenticate: Basic realm="Restricted Area"');
    header('HTTP/1.0 401 Unauthorized');
    echo "You must enter a valid username and password to access this page.";
    exit;
} else {
    // Verify credentials
    if ($_SERVER['PHP_AUTH_USER'] === $valid_username && $_SERVER['PHP_AUTH_PW'] === $valid_password) {
        echo "<h2>Welcome, " . htmlspecialchars($_SERVER['PHP_AUTH_USER']) . "!</h2>";
        echo "<p>You have successfully logged in using Basic HTTP Authentication.</p>";
    } else {
        // Wrong credentials → ask again
        header('WWW-Authenticate: Basic realm="Restricted Area"');
        header('HTTP/1.0 401 Unauthorized');
        echo "Invalid username or password!";
        exit;
    }
}
?>
```