Web-based scripting using PHP

Chapter-2

What is PHP

- PHP stands for Hypertext Preprocessor.
- PHP is an interpreted language, i.e., there is no need for compilation.
- PHP is a server-side scripting language.
- PHP is faster than other scripting languages, for example, ASP and JSP.

Example

```
<html>
  <body>
  <?php
  echo "Welcome to PHP world";
  ?>
  </body>
  </html>
```

Comments in PHP

- A comment in PHP code is a line that is not read/executed as part of the program. Its only purpose is to be read by someone who is looking at the code.

Example of comment

```
< html>
<body>
<?php
// This is a single-line comment
# This is also a single-line comment
/*
This is a multiple-lines comment block
that spans over multiple
lines
*/
// You can also use comments to leave out parts of a code line
$x = 5 /* + 15 */ + 5;
echo $x;
?>
</body>
</html>
```

PHP Case Sensitivity

- In PHP, No keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are case-sensitive.

```
<html>
<body>
<?php
ECHO "Hello World!<br>";
echo "Hello World!<br>";
EcHo "Hello World!<br>";
?>
</body>
</html>
```

Output:

Hello World!
Hello World!
Hello World!

However; all variable names are case-sensitive.

```
<html>
<body>
<?php
$color = "red";
echo "My car is  $color <br>";
echo "My house is  $COLOR <br>";
echo "My boat is $coLOR <br>";
?>
</body>
</html>
```

Output:

My car is red
My house is
My boat is

PHP Variables

- Variables are "containers" for storing information. A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).

Rules for PHP variables:

- A variable starts with the $ sign, followed by the name of the variable
- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive ($age and $AGE are two different variables)

Add two number

```
<html>
<body>
<?php
$a =5 ;
$b=6;
$c=$a+$b;
echo "The sum of  $a  and  $b is  $c”;
?>
</body>
</html>
```

Output:

The sum of 5 and 6 is 11

PHP Variables Scope

- In PHP, variables can be declared anywhere in the script.
- The scope of a variable is the part of the script where the variable can be used.
- **PHP has three different variable scopes:**
- local
- global
- static

Global and Local Scope

- A variable declared **outside** a function has a GLOBAL SCOPE and can only be accessed outside a function:

```
<html>
<body>
<?php
$x = 5; // global scope
function myTest() {
echo "<p>Variable x inside function is: $x</p>";
}
myTest();
echo "<p>Variable x outside function is: $x</p>";
?>
</body>
</html>
```

Output:

Variable x inside function is:

Variable x outside function is: 5

- A variable declared **within** a function has a LOCAL SCOPE and can only be accessed within that function:

```
<html>
<body>
<?php
function myTest() {
    $x = 5; // local scope
    echo "<p>Variable x inside function is: $x</p>";
}
myTest();
// using x outside the function will generate an error
echo "<p>Variable x outside function is: $x</p>";
?>
</body>
</html>
```

Output:

Variable x inside function is: 5

Variable x outside function is:

PHP The static Keyword

- Normally, when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job.To do this, use the static keyword when you first declare the variable:

```
<html>
<body>
<?php
function myTest() {
    static $x = 0;
    echo $x;
    $x++;
}
myTest();
echo "<br>";
myTest();
echo "<br>";
myTest();
?>
</body>
</html>
```

Output:
0
1
2

PHP The global Keyword

- The global keyword is used to access a global variable from within a function.
- To do this, use the global keyword before the variables (inside the function):

```
<html>
<body>
<?php
$x = 5;
$y = 10;
function myTest() {
    global $x, $y;
    $y = $x + $y;
}
myTest();  // run function
echo $y; // output the new value for variable $y
?>
</body>
</html>
```

Output:
15

PHP Conditional Statements

- Conditional statements are used to perform different actions based on different conditions.

  In PHP we have the following conditional statements:
- if statement - executes some code if one condition is true
- if...else statement - executes some code if a condition is true and another code if that condition is false
- if...elseif...else statement - executes different codes for more than two conditions
- switch statement - selects one of many blocks of code to be executed

The if Statement

- The if statement executes some code if one condition is true.
- **Syntax**
- if (*condition*) {
  code to be executed if condition is true;
  }

Example of if statement

```
<html>
<body>
<?php
$t = 10;
if ($t < "20") {
    echo "Have a good day!";
}
?>
</body>
</html>
```

Output:

Have a good day!

The if...else Statement

- The if...else statement executes some code if a condition is true and another code if that condition is false.
- **Syntax**

```
if (condition) {
    code to be executed if condition is true;
}
else {
    code to be executed if condition is false;
}
```

Example of if...else Statement

```
<html>
<body>
<?php
$t = 22;
if ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
?>
</body>
</html>
```

Output:

Have a good day!

The if...elseif...else Statement

- The if...elseif...else statement executes different codes for more than two conditions.
- **Syntax**
- if (*condition*) {
  code to be executed if this condition is true;
  } elseif (*condition*) {
  code to be executed if first condition is false and this condition is true;
  } else {
  code to be executed if all conditions are false;
  }

Example of if...elseif...else Statement:

```
<html>
<body>
<?php
$t = 15;
if ($t < "10") {
    echo "Have a good morning!";
} elseif ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
?>
</body>
</html>
```

Output:

Have a good day!

The PHP switch Statement

- switch statement is used to **select one of many blocks of code to be executed**.
- **Syntax**
- switch (*n*) {

```
    case label1:
        code to be executed if n=label1;
        break;
    case label2:
        code to be executed if n=label2;
        break;
    case label3:
        code to be executed if n=label3;
        break;
    ...
    default:
        code to be executed if n is different from all labels;
}
```

Example of PHP switch Statement

```
<html>
<body>
<?php
$favcolor = "red";
switch ($favcolor) {
    case "red":
        echo "Your favorite color is red!";
        break;
    case "blue":
        echo "Your favorite color is blue!";
        break;
    case "green":
        echo "Your favorite color is green!";
        break;
    default:
        echo "Your favorite color is neither red, blue, nor green!";
}
?>
</body>
</html>
```

Output:

Your favorite color is red!

PHP Loops

- It is a control statement which executes block of code repeatedly until the condition is true.

  In PHP, we have the following looping statements:

- while loop
- do…while loop
- for loop
- Foreach loop

while Loop

- The while loop executes a block of code as long as the specified condition is true.
- **Syntax**
- while (*condition is true*) {
  code to be executed;
  }

Example of while Loop

```
<html>
<body>
<?php
$x = 1;
while($x <= 5) {
  echo "The number is: $x <br
  $x++;
}
?>
</body>
</html>
```

Output:
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5

do...while Loop

- The do…while loop will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true.
- **Syntax**
- do {
  code to be executed;
  } while (*condition is true*);

Example of do...while Loop

```
<html>
<body>
<?php
$x = 1;
do {
    echo "The number is: $x <br>";
    $x++;
} while ($x <= 5);
?>
</body>
</html>
```

Output:
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5

for Loop

- for loop execute a block of code a specified number of times.
- for loop is used when you know in advance how many times the script should run.
- **Syntax**
- for (*init counter; test counter; increment counter*) {
  code to be executed;
  }

Example of **for Loop**

```
<html>
<body>
<?php
for ($x = 0; $x <= 5; $x++) {
  echo "The number is: $x <
}
?>
</body>
</html>
```

Output:
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5

foreach Loop

- The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.
- **Syntax**
- foreach ($*array* as $*value*) {
  code to be executed;
  }

Example of foreach loop

```
<html>
<body>
<?php
$colors = array("red", "green", "blue", "yellow");
foreach ($colors as $value) {
  echo "$value <br>";
}
?>
</body>
</html>
```

Output
red
green
blue
yellow

User Defined Functions

- A function is a block of statements that can be used repeatedly in a program.
- A function will not execute immediately when a page loads. A function will be executed by a call to the function.
- Function names are NOT case-sensitive.
- **Syntax**
- function *functionName*() {
  code to be executed;
  }

Example of User Defined Functions

```
<html>
<body>
<?php
function writeMsg() {
    echo "Hello Nabraj!";
}
writeMsg();
?>
</body>
</html>
```

Output:

Hello Nabraj!

Function Arguments

- Information can be passed to functions through arguments. An argument is just like a variable.
- Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

Function Arguments

```
<html>
<body>
<?php
function familyName($fname) {
    echo "$fname<br>";
}
familyName("Nabraj");
familyName("Ramesh");
familyName("Pawan");
familyName("Nabin");
familyName("Khim");
?>
</body>
</html>
```

Output:
Nabraj
Ramesh
Pawan
Nabin
Khim

Functions - Returning values

- **To let a function return a value, use the return statement:**

```
<html>
<body>
<?php
function sum(int $x, int $y) {
    $z = $x + $y;
    return $z;
}
echo sum(5,10)."<br>";
echo sum(7,13);
?>
</body>
</html>
```

Output:

15

20

Array

- An array is a special variable, which stores related data items that share a common name.
- In PHP, the array() function is used to create an array: array();

```
<html>
<body>
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like $cars[0], $cars[1] and $cars[2]" ;
?>
</body>
</html>
```

Output:

I like Volvo, BMW and Toyota

Types of arrays:

- **Indexed arrays** - Arrays with a numeric index
- **Associative arrays** - Arrays with named keys
- **Multidimensional arrays** - Arrays containing one or more arrays

PHP Indexed Arrays

There are two ways to create indexed arrays:

- The index can be assigned automatically (index always starts at 0), like this:

```
$cars = array("Volvo", "BMW", "Toyota");
```

OR the index can be assigned manually:

```
$cars[0] = "Volvo";
$cars[1] = "BMW";
$cars[2] = "Toyota";
```

Example of Indexed Array:

```
<html>
<body>
<?php
$cars = array("Volvo", "BMW", "Toyota");
$arrlength = count($cars);
for($x = 0; $x < $arrlength; $x++) {
    echo $cars[$x];
    echo "<br>";
}
?>
</body>
</html>
```

Output:
Volvo
BMW
Toyota

PHP Associative Arrays

- Associative arrays are arrays that use named keys that you assign to them.

  There are two ways to create an associative array:

- $age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

  or:

- $age['Peter'] = "35";
  $age['Ben'] = "37";
  $age['Joe'] = "43";

Example of **PHP Associative Arrays**

```
<html>
<body>
<?php
$age = array("Peter"=>"35", "Ben"=>"37",
  "Joe"=>"43");
foreach($age as $x => $value) {
    echo "Key=" . $x . ", Value=" . $value;
    echo "<br>";
}
?>
</body>
</html>
```

Output
Key=Peter, Value=35
Key=Ben, Value=37
Key=Joe, Value=43

Multidimensional Arrays

- A multidimensional array is an array containing one or more arrays.
- Sometimes you want to store values with more than one key. This can be stored in multidimensional arrays.
- For a two-dimensional array you need two indices to select an element

Example of **Multidimensional Arrays**

```
<html>
<body>
<?php
$cars = array
  (
  array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
  );
for ($row = 0; $row < 4; $row++) {
  for ($col = 0; $col < 3; $col++) {
    echo $cars[$row][$col]."  ";
  }
  echo "<br>";
}
?>
</body>
</html>
```

Output:
Volvo 22 18
BMW 15 13
Saab 5 2
Land Rover 17 15

MySQL Database

- With PHP, you can connect to and manipulate databases.
- MySQL is the most popular database system used with PHP.
- The data in a MySQL database are stored in tables. A table is a collection of related data, and it consists of columns and rows.
- MySQL is a database system used on the web
- MySQL is a database system that runs on a server
- MySQL is ideal for both small and large applications MySQL is very fast, reliable, and easy to use
- MySQL uses standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use
- MySQL is developed, distributed, and supported by Oracle Corporation

Connecting to MySQL Server

- Before we can access data in the MySQL database, we need to be able to connect to the server:
- The process to connect MySQL server is given below:

```
 <?php
$user = 'root';
$pass = '';
// Create connection
$conn = mysqli_connect('localhost', $user, $pass);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
else
{
echo "Connected successfully";
}
?>
```

Create a MySQL Database

- The CREATE DATABASE statement is used to create a database in MySQL.
- The following examples create a database named “nawaraj":

```
<?php
$user = 'root';
$pass = '';
// Create connection
$conn = mysqli_connect('localhost', $user, $pass);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
else
{
echo "Connected successfully";
}
// Create database
$sql = "CREATE DATABASE nabraj";
if (mysqli_query($conn, $sql)) {
    echo "Database created successfully";
} else {
    echo "Error creating database: " . mysqli_error($conn) }
?>
```

Create MySQL Tables

- A database table has its own unique name and consists of columns and rows.
- The CREATE TABLE statement is used to create a table in MySQL.
- We will create a table named "MyGuests", with five columns: "id", "firstname", "lastname", "email" and "reg_date":
- CREATE TABLE MyGuests (
  id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  firstname VARCHAR(30) NOT NULL,
  lastname VARCHAR(30) NOT NULL,
  email VARCHAR(50),
  reg_date TIMESTAMP
)

Insert Data Into MySQL

- After a database and a table have been created, we can start adding data in them.
- The INSERT INTO statement is used to add new records to a MySQL table:
- INSERT INTO table_name (column1, column2, column3,...)
  VALUES (value1, value2, value3,...)

  Here are some syntax rules to follow:
- The SQL query must be quoted in PHP
- String values inside the SQL query must be quoted
- Numeric values must not be quoted
- The word NULL must not be quoted

Example to insert single record into a table

```
 <?php
$user = 'root';
$pass = '';
$dbname = "nabraj";
// Create connection
$conn = mysqli_connect('localhost', $user, $pass, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
else
{
echo "Connected successfully";
}
$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";
if (mysqli_query($conn, $sql)) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}
?>
```

Example to insert multiple record into a table

```
 <?php
$user = 'root';
$pass = '';
$dbname = "nabraj";
// Create connection
$conn = mysqli_connect('localhost', $user, $pass, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
else
{
echo "Connected successfully";
}
$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Nabraj', 'Koirala', 'koiralanabraj@gmail.com'),
('Apil', 'Koirala', 'apil@gmail.com'),
('Ram', 'Pandey', 'rampandey@gmail.com')";
if (mysqli_query($conn, $sql)) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}
?>
```

Select Data From a MySQL Database

- The SELECT statement is used to select data from one or more tables:

  SELECT column_name(s) FROM table_name

- or we can use the * character to select ALL columns from a table:

  SELECT * FROM table_name

Example of Select Data From a MySQL Database

```
<?php
$user = 'root';
$pass = '';
$dbname = "nabraj";
// Create connection
$conn = mysqli_connect('localhost', $user, $pass, $dbname);
//$sql = "SELECT id, firstname, lastname FROM MyGuests";
$sql = "SELECT * FROM  MyGuests";
$result = $conn->query($sql);
if ($result->num_rows > 0) {
    // output data of each row
    while($row = $result->fetch_assoc()) {
  echo "<br> id: ". $row["id"]. " - Name: ". $row["firstname"]. " " . $row["lastname"] ."
  Email:".$row["email"]. " Registration Date" .$row["reg_date"]. "<br>";
    }
} else {
    echo "0 results";
}
?>
</body>
</html>
```

Delete Data From MySQL

- The DELETE statement is used to delete records from a table:
- DELETE FROM table_name
  WHERE some_column = some_value

Example of Delete Data From MySQL

```
 <?php
$user = 'root';
$pass = '';
$dbname = "nabraj";
// Create connection
$conn = mysqli_connect('localhost', $user, $pass, $dbname);
// sql to delete a record
$sql = "DELETE FROM MyGuests WHERE id=3";
if (mysqli_query($conn, $sql)) {
    echo "Record deleted successfully";
} else {
    echo "Error deleting record: " . mysqli_error($conn);
}
?>
```

Update Data in MySQL

- The UPDATE statement is used to update existing records in a table:
- UPDATE table_name
  SET column1=value, column2=value2,...
  WHERE some_column=some_value

Example of Update Data in MySQL

```
 <?php
$user = 'root';
$pass = '';
$dbname = "nabraj";
// Create connection
$conn = mysqli_connect('localhost', $user, $pass, $dbname);
// sql to update a record
$sql = "UPDATE MyGuests SET lastname='Bashyal' WHERE
  id=6";
if (mysqli_query($conn, $sql)) {
    echo "Record updated successfully";
} else {
    echo "Error updating record: " . mysqli_error($conn);
}
?>
```

File Handling

- File handling is an important part of any web application. You often need to open and process a file for different tasks.

File Open/Read/Close

PHP Open File - fopen():

- A better method to open files is with the fopen() function.
- The first parameter of fopen() contains the name of the file to be opened and the second parameter specifies in which mode the file should be opened.

```
<html>
<body>
<?php
$myfile = fopen("d:\project.txt", "r") or die("Unable to open file!");
echo fread($myfile,filesize("d:\project.txt"));
fclose($myfile);
?>
</body>
</html>
```

The file may be opened in one of the following

Modes	Description
r	**Open a file for read only**. File pointer starts at the beginning of the file
w	**Open a file for write only**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
a	**Open a file for write only**. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
x	**Creates a new file for write only**. Returns FALSE and an error if file already exists
r+	**Open a file for read/write**. File pointer starts at the beginning of the file
w+	**Open a file for read/write**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
a+	**Open a file for read/write**. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
x+	**Creates a new file for read/write**. Returns FALSE and an error if file already exists

PHP Read File - fread()

- The fread() function reads from an open file.
- The first parameter of fread() contains the name of the file to read from and the second parameter specifies the maximum number of bytes to read.
- The following PHP code reads the “project.txt" file to the end:
- fread($myfile,filesize(“project.txt"));

PHP Close File - fclose()

- The fclose() function is used to close an open file.
- It's a good programming practice to close all files after you have finished with them.
- The fclose() requires the name of the file (or a variable that holds the filename) we want to close:

PHP Read Single Line - fgets()

- The fgets() function is used to read a single line from a file.
- The example below outputs the first line of the “project.txt" file:
- **Example**

```
<html>
<body>
<?php
$myfile = fopen("d:\project.txt", "r") or die("Unable to open file!");
echo fgets($myfile);
fclose($myfile);
?>
</body>
</html>
```

PHP Check End-Of-File - feof()

- The feof() function checks if the "end-of-file" (EOF) has been reached.
- The feof() function is useful for looping through data of unknown length.
- The example below reads the “project.txt" file line by line, until end-of-file is reached:

Example of End-Of-File - feof()

```
<html>
<body>
<?php
$myfile = fopen("d:\project.txt", "r") or die("Unable to open
   file!");
// Output one line until end-of-file
while(!feof($myfile)) {
  echo fgets($myfile) . "<br>";
}
fclose($myfile);
?>
</body>
</html>
```

PHP Read Single Character - fgetc()

- The fgetc() function is used to read a single character from a file.
- The example below reads the “project.txt" file character by character, until end-of-file is reached:

```
<html>
<body>
<?php
$myfile = fopen("d:\project.txt", "r") or die("Unable to open file!");
// Output one character until end-of-file
while(!feof($myfile)) {
  echo fgetc($myfile);
}
fclose($myfile);
?>
</body>
</html>
```

File Create/Write

PHP Create File - fopen()

- The fopen() function is also used to create a file. Maybe a little confusing, but in PHP, a file is created using the same function used to open files.
- If you use fopen() on a file that does not exist, it will create it, given that the file is opened for writing (w) or appending (a).
- The example below creates a new file called "testfile.txt". The file will be created in the same directory where the PHP code resides:
- **Example**
- $myfile = fopen(“d:\testfile.txt", "w")

PHP Write to File - fwrite()

- The fwrite() function is used to write to a file.
- The first parameter of fwrite() contains the name of the file to write to and the second parameter is the string to be written.
- The example below writes a couple of names into a new file called "testfile.txt":
- **Example**

```
<?php
$myfile = fopen("d:\ testfile.txt", "w") or die("Unable to open file!");
$txt = "Nabraj Koirala\n";
fwrite($myfile, $txt);
$txt = "Apil Koirala\n";
fwrite($myfile, $txt);
fclose($myfile);
?>
```

File Upload

- With PHP, it is easy to upload files to the server.
- **Example:**
- Save this file upload.html

```
<html>
<form action = "receive.php" method="POST"
  enctype = "multipart/form-data">
<p> <input type = "file" name = "myfile"> </p>
<p> <input type = "submit" name = "upload" value =
  "send"></p>
</form>
</body>
</html>
```

- **Save this file receive.php**

```
<?php
  $name = $_FILES['myfile']['name'];
  $tmp_name=$_FILES['myfile']['tmp_name'];
  if(move_uploaded_file($tmp_name,$name))
  {
  echo "file uploaded";
  }
  else
  {
  echo "file not uploaded";
  }
  ?>
```

Cookie

- A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

  Syntax

- setcookie(*name, value, expire, path, domain, secure, httponly*);
- Only the *name* parameter is required. All other parameters are optional.

Example to Create/Retrieve a Cookie:

```
<html>
<?php
$cookie_name = "user";
$cookie_value = "Nabraj Koirala";
setcookie($cookie_name, $cookie_value, time() + 10); // 86400 = 1 day
?>
<html>
<body>
<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>
<p><strong>Note:</strong> You might have to reload the page to see the value of the cookie.</p>
</body>
</html>
```

Delete a Cookie

- To delete a cookie, use the setcookie() function with an expiration date in the past:

```
<html>
<?php
// set the expiration date to 10 second ago
setcookie("user", "", time() - 10);
?>
<html>
<body>
<?php
echo "Cookie 'user' is deleted.";
?>
</body>
</html>
```

PHP Sessions

- A session is a way to store information (in variables) to be used across multiple pages.
- Unlike a cookie, the information is not stored on the users computer.
- When you work with an application, you open it, do some changes, and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.
- Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc). By default, session variables last until the user closes the browser.

Start a PHP Session:

- A session is started with the session_start() function.
- Session variables are set with the PHP global variable: $_SESSION.

```
 <?php
// Start the session
session_start();
?>
<html>
<body>
<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>
</body>
</html>
```

Get PHP Session Variable Values:

```
 <?php
session_start();
?>
<html>
<body>
<?php
// Echo session variables that were set on previous page
echo "Favorite color is " . $_SESSION["favcolor"] .
   ".<br>";
echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
?>
</body>
</html>
```

Destroy a PHP Session

- To remove all global session variables and destroy the session, use session_unset() and session_destroy():
- **Example**

```
<?php
session_start();
?>
<html>
<body>
<?php
// remove all session variables
session_unset();
// destroy the session
session_destroy();
echo "All session variables are now removed, and the session is destroyed."
?>
</body>
</html>
```