PHP—code with dary

Fundamentals:

1. Introduction
2. Variables and datatypes
3. Operators
4. Control structures
5. Functions {Arguments/params, Return value, Include and require}
6. Built-in functions {math, string, array, date, time, random}

A built in function is a function that is built into an application and can be accessed by end-users.

1. How to work with superglobals {POST, GET, SESSION, COOKIE, FILES}
2. Introduction to databases
3. Using a database to create a login/register script

{ connecting to the database, prepared statements, different types of arrays, password hasing}

1. Assessments

d/f b/w echo and print?

session-8: Line Breaks in php: **nl2br()**

echo nl2br("My name is shiva \r\n and kumar");

The nl2br() function **inserts HTML line breaks (<br> or <br />) in front of each newline (\n) in a string**.

Another \_way:

//Line breaks

echo "My name is shiva and I'm in indian <br />";

echo "I live in Hyderabad <br />";

echo "I live in Hyderabad" . "<br />";

$region = "Indian";

//traditional way

echo "My name is shiva and I'm in .$region. <br />";

//advanced way

echo "My name is shiva and I'm in {$region} <br />";

Session-9: Variables and Datatypes

$name = "Shiva";

//traditional\_way

echo "My name is " . $name . "<br />";

echo $name . " is 24 years old" . "<br />";

//advanced\_way

echo "My name is {$name}" . "<br />";

echo "{$name} is 24 years old" . "<br />";

Session-10:Variables in Echo and Print

//Echo

#Single quotes - variable named are echoed as it is

#Doublt quotes - Variables names are replaced by the vairable values

Session – 11: Datatypes

* Integer
* Float
* String
* Boolean
* Array
* Object
* Resource
* Null

Session-12 Arrays in PHP

Array creation in PHP

Syntax:

**$variable = array(values);**

1. **var\_dump($cars);**

result:

Honda

C:\wamp64\www\php\_tutorial\_2023\Arrays.php:12:

**array** *(size=4)*

0 => string 'BMW' *(length=3)*

1 => string 'Adui' *(length=4)*

2 => string 'Honda' *(length=5)*

1. => string 'Ferarri' *(length=7)*
2. **print\_r($cars);**

result:

Honda Array ( [0] => BMW [1] => Adui [2] => Honda [3] => Ferarri )

Built-in-functions of Array in PHP:

1. array\_merge() -which merges two arrays into one.

eg: $cars = array\_merge($cars, $car2)

var\_dump($cars)

C:\wamp64\www\php\_tutorial\_2023\Arrays.php:16:

**array** *(size=7)*

0 => string 'BMW' *(length=3)*

1 => string 'Adui' *(length=4)*

2 => string 'Honda' *(length=5)*

3 => string 'Ferarri' *(length=7)*

4 => string 'volvo' *(length=5)*

5 => string 'Chevy' *(length=5)*

6 => string 'Volkswagen' *(length=10)*

print\_r($cars)

Array ( [0] => BMW [1] => Adui [2] => Honda [3] => Ferarri [4] => volvo [5] => Chevy [6] => Volkswagen

Session-13: Associative Arrays in PHP

//Associative arrays

$cars = array("audi" => 55.500, "Ferarri" => 45.900, "Honda" => 60.750);

foreach ($cars as $key => $value) {

echo "My " . $key . " has " . $value . " mileage <br />";

}

Session-14: Multidimensional Array

<?php

//Mutidimensional Array

//Expensive cars

//Audi, Mercedes, BMW

//Inexpensive cars

//volvo, ford, Toyota

$cars = array(

"Expensive" => array("Audi", "Mercedes", "BMW"),

"Inexpensive" => array("Volvo", "Ford", "Toyota")

);

echo $cars["Expensive"][0];

echo "<br />";

echo $cars["Inexpensive"][1];

?>

Session -15

Exercises on Datatypes:

Exe-1:

Built-in function

**Number\_format()**

Session -16: Arithmetic operator

Session -17: Assignment operator

Assignment operators

+= --- Add and assign

-= --- Subtract and assign

\*= --- Multiply and assign

/= --- Divide and assign

.= --- Concatenate and assign

Session-18: Comparison operators

Comparison operators

<, >, ==, !=, <>, ===(identical), <=, >=, (spaceship operator -1, 0, 1 { <=> (symbol)})

Spaceship --if the value on the left is less than the value on the right, the system will return -1 ( 1 <=> 2 )

-- if the value on the left is equal to the value on the right, the system will return 0 (2 <=> 2)

-- if the value on the left is greater than the value on the right, the system will return 1 (4 <=> 2)

Session-19: Increment and Decrement operators

Increment and Decrement

++$x --- Pre-Increment

$x++ --- Post-Increment

--$x --- Pre-Decrement

$x-- --- Post-Decrement

NOTE:

Inc/Dec only effects Numbers and strings

Inc/Dec will not effect Arrays, objects, Resources, boolean

NOTE: we cannot Decrement NULL but we can Increment it

Session-20: Logical operators

// NOTE:

Logical operators

And --- Both X and Y are true

&& --- Both X and Y are true

Or --- Either X or Y are true

|| --- Either X or Y are true

Xor --- Either X or Y are true, not both

! --- True if X is not true

1 = True , 0 = False

$x = 0;

$y = 10;

if ($x == $y && 1 == 1) {

echo "True!";

}

else {

echo "False!";

}

if ($x == $y xor 1 == 5) {

echo "True!";

}

else {

echo "False!";

}

if (!$x == $y && 1 == 1) {

echo "True!";

}

else {

echo "False!";

}

Session- 21: Control Structures

NOTE: Conditional statements

If statement

Else statement

Else if statement

Switch statement

Session- 22: Switch statement

Session- 23: Control statements – Loops

NOTE:

While Loop

Do-while Loop

For Loop

foreach Loop

Session – 25: For loop

<?php

for (initialization; condition; increment/decrement) {

//Code

}

?>

Session – 26: Foreach

<?php

// NOTE: ForEach Loop - iterate over array variables

syntax

foreach (array\_Expression as $key) {

}

array

$names = array("john", "michael", "rahul", "srinivas");

foreach ($names as $name) {

echo "My name is: " .$name . "<br />";

}

Associative array

$person = array("Name" => "shiva", "Age" => 30, "Gender" => "male");

foreach ($person as $key => $value) {

echo $key . ": " . $value;

echo "<br />";

}

?>

Session – 27: exercise on control structures

Exe-1:

<?php

$value1 = 10;

$value2 = 5;

if ($value1 > $value2)

{

$Highest = $value1;

$lowest = $value2;

echo "Highest value is: " . $value1 . "<br />";

echo "Lowest value is: " . $value2;

}

?>

Exercise done.

Session -28: PHP Functions

NOTE: Functions

A function is a block of code written in a program to perform some specific task

two major types of Functions

Built-in-functions

User-defined Functions

why should you use Functions

1. Reuseability

2. Easy for error detection

3. Easily maintained

NOTE:

Camel case

Every word after the first one is a capital

Eg: myFunction();

Lower case

All lowercase , underscore inbetween

Eg: my\_function();

Pascal case

Every word is capitalized

Eg: MyFunction();

NOTE: camelcase and lowercase is preferrable to functions and pascal case is most preferrable to classes

NOTE: we have to call this function outside the function block

function myFunction() {

echo "Hello!";

// NOTE: we have to call this function outside the function block

}

myFunction();

Session – 30: function with arguments/Parameters

Note: passbyreference – we will maniplate the value.

Session- 32: global and local scopes

Session- 33: constants

NOTE: constants

a constant is a name or identifier for a fixed value

Two ways

1. Function - define()

2. Keyword - CONST

constants are always in uppercase

Name has the same rules as variables

for Constants , we won't use $ sign

// NOTE: 1. Define()

define(ARGUMENT1, ARGUMENT2);

// ARGUMENT1 = CONSTANTS\_NAME

// ARGUMENT2 = the value that you want to assign to that constant

d/f b/w define() and const in constants.

Session – 34 How to include documents

<?php

Include "C:\wamp64\www\php\_tutorial\_2023\Functions\includes\head.php";

include "C:\wamp64\www\php\_tutorial\_2023\Functions\includes\header.php";

?>

Session – 35 How to require documents

d/f b/w include and require?

NOTE:

Include -- if path is wrong - we only get a warning

Require -- if path is wrong - the script will stop

Include and require by using these keywords we can do inclusion of files/docs.

Include\_once and require\_once does the same thing!.

Session-36: Exercise on functions

Session – 37: built-in functions

NOTE: Built in functions

String functions

Math functions

Date functions

Array functions

Random number functions

Session -41: date and time functions on PHP

NOTE: Date & Time

DateTime

'd' = Day

'j' = Day without zeros

'D' = Day of week (3 letters)

'l' = Full day of the week

'm' = Month as a number with zeros

'n' = Month as a number without zeros

'M' = Month (3 letters)

'F' = Full Month

'y' = Two-digit year

'Y' = Full Year

echo date('y-m-d'); //Result : 23-03-04

echo date('Y-M-D'); //Result :2023-Mar-Sat

echo date('Y-m-d'); //Result : 2023-03-04

NOTE: Times

'g' = Hours in 12-hour format without zeros

'h' = Hours in 12-hour format with zeros

'G' = Hours in 24=hour format without zeros

'H' = Hours in 24-hour format with zeros

'a' = am/pm in lowercase

'A' = am/pm in uppercase

'i' = minutes without leading zeros

's' = seconds without leading zeros

echo date('Y-m-d h:i:s'); //standard dateTime format Result: 2023-03-04 11:00:44

Session- 42 : Random numbers

Rend()

Getrandmax()

// NOTE: Random numbers

echo rand(1, 10); //rand(min\_value, max\_value)

$min = 1;

$max = 10;

echo rand($min, $max); //rand(min\_value, max\_value)

Session – 43: Superglobals

Session – 46: Sessions

d/f b/w cookies and sessions?

NOTE: Session Superglobals

Session

Good to use for sensitive information

Session exists as long as the browser is open

session\_start();

// storing information

$\_SESSION["Name"] = "Shiva";

$\_SESSION["Age"] = 24;

echo "Hello " . $\_SESSION["Name"];

echo "<br />";

session\_destroy();

// whenever we close the browser session get destroyed !

echo "Hello " . $\_SESSION["Name"];

Session – 47: Cookies

Session-48 : Files

<?php

NOTE: files superglobals

$\_FILES

How to upload files

1. Upload it to the root

2. Directly to the database

enctype="multipart/form-data

Specifies how the form data should be encoded

echo $name = $\_FILES['file']['name'] . "<br />";

echo $type = $\_FILES['file']['type'] . "<br />";

echo $tmp\_location = $\_FILES['file']['tmp\_name'] . "<br />";

echo $error = $\_FILES['file']['error'] . "<br />";

?>

<form class="" action="FILES.php" method="post" enctype="multipart/form-data">

Upload: <input type="file" name="file" value=""><br />

<button type="submit" name="button">Submit</button>

</form>

Session: 49: Files continuation..

Built-in function: (new)

Explode()

Implode()

End()

in\_array()

uniqid();

Session -50: create and write files

<https://www.php.net/manual/en/function.fopen.php>

1. fopen(‘path’, ‘access\_type’);
2. fwrite(file\_name, ‘txt’);
3. fclose(file\_name);

Session – 52: How to create a working contact form

Session – 54: creating a mysql database

Session- 55: creating a table

Session -56 : insert data using mysql

Session – 57: updating data using mysql

Session – 59: Deleting data using mysql

Session – 60: How to order data using mysql

Reference:

<https://www.youtube.com/watch?v=35ZOdzb3wzc&list=PLFHz2csJcgk_fFEWydZJLiXpc9nB1qfpi&index=15>

<https://onedrive.live.com/?authkey=%21AEeZRSYfXguFM%2DY&id=CF73B766D78740D7%215153&cid=CF73B766D78740D7>

<https://www.php.net/manual/en/functions.internal.php>

website reference for practise purpose

<https://tutorials.supunkavinda.blog/php/conditionals#:~:text=Nested%20If%20Statements,are%20called%20Nested%20If%20Statements>.

Part -2:

Session -1: OOPs in php

1.procedural PHP

\* Adding new behaviour

\* scaling the behaviour

\*simple checks

With OOPs,

1. Resuse code
2. Companies use it
3. Group work

MVC:

PDO: PHP Data Objects

Session -2: classes, Properties, Methods

//// NOTE: Define a class

class User {

// Properties

public $name;

// Methods

public function sayHello() {

return $this->name . " Hello users";

}

}

// Instantiate user object from User class

$user = new User();

// '->' is called Object Operator

//echo $user->name;

//var\_dump($user->$name);

$user->name = "shiva";

echo "<br />";

echo $user->sayHello();

Session -3: Access Modifiers

Public -> anywhere in the code

Protected -> within in a class and its sub class

Private -> within a class itself

NOTE: Access Modifiers = Public, Private, Protected

class User {

// Properties

public $name;

}

$user = new User();

$user->name = "Dary";

echo $user->name;

Session -4: Class inheritance

NOTE: Class Inheritance

class User {

// Properties

public $name;

public $email;

public function welcomeMsg() {

return "Have a good working day!";

}

}

class Admin extends User {

public $level;

}

$user = new User();

$user->name = "shiva";

$user->email = "shiva@example.com";

echo $user->name . " " . $user->email;

echo "<br />";

$admin = new Admin();

$admin->name = "John";

$admin->email = "john@yees.com";

$admin->level = "superadmin";

echo $admin->name . " " . $admin->email . " " . $admin->level;

echo "<br />";

echo $admin->welcomeMsg();

Session -5: Constructor and deconstructor

// NOTE: Constructor and Deconstructor

class User {

// Properties

public $name;

public $age;

public function \_\_construct($name, $age) {

// echo "This is my constructor";

$this->name = $name;

$this->age = $age;

// echo "My name is: " . $this->name;

// echo "<br />";

// echo "I am " . $this->age . " years old";

}

// NOTE: Deconstructor

public function \_\_destruct() {

echo "Destructor works!";

}

public function myAge() {

return "My age is: " . $this->age;

}

}

$user = new User("Shiva", 24);

echo $user->myAge();

Deconstructor will be executed at the end of the code!

Session – 6 Getter and Setters in PHP

// NOTE: Getters and Setters

class User {

private $name;

private $age;

public function \_\_construct($name, $age) {

$this->name = $name;

$this->age = $age;

}

/\*\*

\* @return $name

\*/

public function getName() {

return $this->name;

}

/\*

\* @param $name

\*/

public function setName($name) {

$this->name = $name;

}

/\*\*

\* @return $age

\*/

public function getAge() {

return $this->age;

}

/\*

\* @param $age

\*/

public function setAge($age) {

$this->age = $age;

}

}

$user = new User("Shiva", 24);

echo $user->getName();

echo "<br />";

echo $user->getAge();

Note: Magic Method

PHP built-in method:

property\_exists()

­­­­­­­­­­­­­­­­­­­­­­­­­­­­

// NOTE: Getters and Setters

class User {

private $name;

private $age;

public function \_\_construct($name, $age) {

$this->name = $name;

$this->age = $age;

}

// NOTE: Magic Method - \_\_GET Method

public function \_\_get($property) {

if (property\_exists($this, $property)) {

return $this->$property;

}

}

// // NOTE: Magic Method - \_\_SET Method

public function \_\_set($property, $value) {

if (property\_exists($this, $property)) {

$this->$property = $value;

}

return $this;

}

}

$user = new User("Shiva", 24);

$user->\_\_set('name', 'Rahul');

echo $user->\_\_get('name');

Session-7: static properties and methods

// NOTE: Static Properties and Methods

class User {

//static property

public static $nextId = 0;

public $myId;

public static function printId($id) {

return $id;

}

}

/\*

$user = new User();

$user->myId = 10;

echo $user->myId;

\*/

// NOTE: class :: propertes - means we are directly entering into class to access static property and static methods

// // NOTE: keyword self

echo User::$nextId;

echo "<br />";

echo User::printId(5);

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

// NOTE: Static Properties and Methods

class User {

//static property

private static $nextId = 0;

private $myId;

public static function printId() {

// NOTE: keyword self

return "User id is: " . self::$nextId;

}

}

/\*

$user = new User();

$user->myId = 10;

echo $user->myId;

\*/

// NOTE: class :: propertes - means we are directly entering into class to access static property and static methods

//

//echo User::$nextId;

//echo "<br />";

echo User::printId();

Session -8: Abstract classes and methods

// NOTE: Abstarct classes and Methods

abstract class Manager {

abstract protected function showProject($project);

}

class Employee extends Manager {

public function showProject($project = "Sony") {

return "I'm working in " . $project;

}

public function startProject($project) {

return $this->showProject($project);

}

}

$employee = new Employee();

echo $employee->showProject();

Session -9: toString() Method in OOP PHP

// NOTE: toString() Method in OOP php

class User {

//properties

private $name;

private $age;

// Constructor

public function \_\_construct($name, $age) {

$this->name = $name;

$this->age = $age;

}

// Magic Method - toString() -- which implictly converts objects into strings

public function \_\_toString() {

return "Name: " . $this->name . "<br /> Age: " . $this->age;

}

// NOTE: Getters and Setters for name and age

public function getName() {

return $this->name;

}

public function setName($name) {

$this->name = $name;

}

public function getAge() {

return $this->age;

}

public function setAge($age) {

$this->age = $age;

}

}

$user = new User("Shiva", 23);

// echo $user->getName();

// echo "<br />";

//// echo $user->getAge();

//print\_r($user); //Result User Object ( [name:User:private] => Shiva [age:User:private] => 23 )

echo $user; //Result Name: Shiva Age: 23

//because of magic method converts objests to string implicitly

Session – 10: Final Method and class

// NOTE: Final Method and class

// If we declare class as final we cannot extend or override that class

// and method as Final, we cannot override it.

// class User {

// public $name ="John";

// public $age;

//

// final public function printName($name) {

// return "My name is " . $name;

// }

// }

//

// class Manager extends User {

// public $name = "David";

//

// // final public function printName($name) {

// // return "Hi mr. " . $name;

// // }

// }

//

// $manager = new Manager();

// echo $manager->printName("Shiva"); //result: Fatal error: Cannot override final method User::printName() in C:\wamp64\www\php\_tutorial\_2023\OOPS\index.php on line 280

// even if we create instance of User , the result won't change

Session -11: PDO

What is PDO.

1. PHP Data Objects
2. Lightweight, consistent interface
3. Provides Data-access abstraction

Two ways to access Database:

PDO vs MySQLi

Benefits

Session – 12: MVC

Software design pattern

* Stands for model view controller
* Makes creating huge applications easy
* Programming language framework

Model:

* “in touch” with the database
* Fetch, insert,update, and delete data
* Communicates with controller

View:

* The UI
* It’s your html,css,js,xml
* Shows data

Controller:

* Processes the data
* Request from view, update in model