PHP FINAL EXAM **~SAURAV CHAUDHARY**

1. Explain PHP Data Types (5 Marks)

PHP supports the following data types, which are used to declare variables:

1. **String**: A sequence of characters, e.g., "Hello, World!".

   php
   Copy code
   $string = "Hello, PHP!";

2. **Integer**: Non-decimal numbers, e.g., 42, -10.

   php
   Copy code
   $int = 25;

3. **Float (Double)**: Numbers with decimal points or in exponential form, e.g., 3.14, 1.2e3.

   php
   Copy code
   $float = 3.14;

4. **Boolean**: Represents two possible states: true or false.

   php
   Copy code
   $bool = true;

5. **Array**: A collection of values, indexed or associative.

   php
   Copy code
   $array = ["Apple", "Banana", "Cherry"];

6. **Object**: Used to store data and methods for a specific class.

   php
   Copy code
   class Car {

```
    public $color;
    function __construct($color) {
        $this->color = $color;
    }
}
$car = new Car("Red");
```

7. **NULL**: A variable with no value.

   ```
   php
   Copy code
   $var = NULL;
   ```

8. **Resource**: A special data type for holding references to external resources like database connections.

---

2. Explain Features of PHP (5 Marks)

PHP is a popular server-side scripting language with the following key features:

1. **Open Source**: PHP is free to use and has a large, active community for support and development.
2. **Platform Independent**: PHP scripts can run on any operating system, such as Windows, Linux, or macOS.
3. **Ease of Use**: PHP syntax is simple and easy to learn, making it ideal for beginners.
4. **Dynamic and Flexible**: PHP can handle dynamic content generation, making it suitable for creating interactive web applications.
5. **Integration**: PHP can easily integrate with various databases (e.g., MySQL, PostgreSQL) and other technologies like HTML, CSS, and JavaScript.
6. **Fast Execution**: PHP is lightweight and performs well, especially when embedded in HTML.
7. **Support for Web Protocols**: PHP supports HTTP, FTP, and various other protocols for web communication.
8. **Extensibility**: PHP allows developers to extend its functionality using libraries and frameworks like Laravel and CodeIgniter.

3. Explain PHP Array with an example. 4. Explain PHP Inbuilt functions with an example.

:

3. Explain PHP Array with an Example (5 Marks)

An **array** in PHP is a data structure that allows storing multiple values in a single variable. Arrays in PHP are of three types:

1. **Indexed Array**: Stores values with numeric keys.

   ```
   php
   Copy code
   $fruits = ["Apple", "Banana", "Cherry"];
   echo $fruits[0]; // Output: Apple
   ```

2. **Associative Array**: Uses named keys instead of numeric indices.

   ```
   php
   Copy code
   $person = ["Name" => "John", "Age" => 30, "City" => "New York"];
   echo $person["Name"]; // Output: John
   ```

3. **Multidimensional Array**: Contains arrays within arrays.

   ```
   php
   Copy code
   $students = [
      ["Name" => "Alice", "Marks" => 85],
      ["Name" => "Bob", "Marks" => 78]
   ];
   echo $students[1]["Name"]; // Output: Bob
   ```

Arrays in PHP are versatile and commonly used for storing and managing data collections.

---

4. Explain PHP Inbuilt Functions with an Example (5 Marks)

PHP provides a wide range of **inbuilt functions** for different tasks. Here are examples:

1. **String Function**: strlen() returns the length of a string.

   php
   Copy code
   $text = "Hello, PHP!";
   echo strlen($text); // Output: 11

2. **Array Function**: array_push() adds elements to the end of an array.

   php
   Copy code
   $fruits = ["Apple", "Banana"];
   array_push($fruits, "Cherry", "Date");
   print_r($fruits);
   // Output: Array ([0] => Apple [1] => Banana [2] => Cherry [3] => Date
)

3. **Math Function**: sqrt() calculates the square root of a number.

   php
   Copy code
   echo sqrt(16); // Output: 4

4. **Date/Time Function**: date() formats the current date.

   php
   Copy code
   echo date("Y-m-d"); // Output: 2024-12-09 (current date)

5. **File Handling Function**: file_get_contents() reads the content of a file.

   php
   Copy code
   $content = file_get_contents("example.txt");
   echo $content; // Output: Content of the file

These built-in functions simplify coding and improve efficiency in PHP development.

5. Explain PHP Superglobal arrays. 6. Explain PHP Array and String functions with examples

5. Explain PHP Superglobal Arrays (5 Marks)

PHP **Superglobals** are predefined arrays accessible from anywhere in the script, regardless of scope. These arrays store important information like user input, server details, and more.

Common Superglobals:

1. **$_GET**: Stores data sent via URL parameters.

   php
   Copy code
   // URL: example.com?name=John
   echo $_GET['name']; // Output: John

2. **$_POST**: Stores data sent via HTTP POST method.

   php
   Copy code
   // Assuming a form submission with a "name" field.
   echo $_POST['name'];

3. **$_REQUEST**: Combines data from $_GET, $_POST, and $_COOKIE.

   php
   Copy code
   echo $_REQUEST['name'];

4. **$_SESSION**: Stores session variables.

   php
   Copy code
   session_start();
   $_SESSION['user'] = "John";
   echo $_SESSION['user']; // Output: John

5. **$_COOKIE**: Stores data from cookies.

   php
   Copy code
   echo $_COOKIE['user'];

6. **$_SERVER**: Provides server and execution environment information.

php
Copy code
echo $_SERVER['PHP_SELF']; // Output: /current/script/path.php

7. **$_FILES**: Stores uploaded file data.

   php
   Copy code
   echo $_FILES['upload']['name'];

8. **$_ENV**: Stores environment variables.

   php
   Copy code
   echo $_ENV['PATH'];

Superglobals are essential for handling form data, file uploads, sessions, and server communication.

---

6. Explain PHP Array and String Functions with Examples (5 Marks)

PHP Array Functions:

1. **array_merge()**: Merges two or more arrays.

   php
   Copy code
   $arr1 = [1, 2];
   $arr2 = [3, 4];
   $result = array_merge($arr1, $arr2);
   print_r($result); // Output: [1, 2, 3, 4]

2. **in_array()**: Checks if a value exists in an array.

   php
   Copy code
   $colors = ["Red", "Green", "Blue"];
   echo in_array("Green", $colors) ? "Found" : "Not Found"; // Output: Found

3. **array_keys()**: Retrieves all keys from an array.

   php

Copy code
$person = ["Name" => "Alice", "Age" => 25];
print_r(array_keys($person)); // Output: [Name, Age]

PHP String Functions:

1. **str_replace()**: Replaces a substring with another.

   php
   Copy code
   $text = "Hello, World!";
   echo str_replace("World", "PHP", $text); // Output: Hello, PHP!

2. **strpos()**: Finds the position of the first occurrence of a substring.

   php
   Copy code
   $text = "Welcome to PHP";
   echo strpos($text, "PHP"); // Output: 11

3. **explode()**: Splits a string into an array based on a delimiter.

   php
   Copy code
   $csv = "Red,Green,Blue";
   print_r(explode(",", $csv)); // Output: [Red, Green, Blue]

Both array and string functions provide powerful tools for manipulating data and are widely used in PHP development.

7. Explain the difference between PHP include and require with examples. 8. Explain PHP file reading functions.

:

7. Explain the Difference Between PHP include and require with Examples (5 Marks)

The include and require statements in PHP are used to insert the content of one PHP file into another. The key difference lies in their error handling.

include

- If the file to be included is not found, PHP generates a **warning** and continues executing the script.
- Use when the included file is optional or non-critical.

Example:

php
Copy code
include "optional_file.php";
echo "This code runs even if the file is missing.";

require

- If the file to be included is not found, PHP generates a **fatal error** and stops executing the script.
- Use when the included file is critical to the application.

Example:

php
Copy code
require "important_file.php";
echo "This code will NOT run if the file is missing.";

Key Difference:

- include is used for non-essential files.
- require is used for mandatory files.

---

8. Explain PHP File Reading Functions (5 Marks)

PHP provides several functions to read files. Here are the most commonly used ones:

1. fopen() and fread():

- Opens a file and reads a specific number of bytes.
- Requires closing the file using fclose().

Example:

php
Copy code
```
$file = fopen("example.txt", "r");
$content = fread($file, filesize("example.txt"));
fclose($file);
echo $content;
```

---

2. file_get_contents():

- Reads the entire file into a string in one step.
- Simpler and more efficient for small files.

Example:

php
Copy code
```
$content = file_get_contents("example.txt");
echo $content;
```

---

3. file():

- Reads the file into an array, with each line as an array element.

Example:

php
Copy code
```
$lines = file("example.txt");
foreach ($lines as $line) {
    echo $line . "<br>";
}
```

---

4. fgets():

- Reads one line at a time from an open file.

Example:

php
Copy code

```
$file = fopen("example.txt", "r");
while (($line = fgets($file)) !== false) {
    echo $line . "<br>";
}
fclose($file);
```

---

5. readfile():

- Reads and outputs the content of a file directly.

Example:

php
Copy code
readfile("example.txt");

These functions allow developers to handle file reading efficiently, depending on the use case and file size.

9. Explain PHP File Writing Functions (5 Marks)

PHP provides functions to write content into files. Below are some commonly used file writing functions:

1. **fopen() and fwrite()**:
    - Use fopen() to open a file in write (w) or append (a) mode.
    - Use fwrite() to write data into the file.

    Example:

    php
    Copy code
    $file = fopen("example.txt", "w");
    fwrite($file, "Hello, this is written to the file.");
    fclose($file); // Always close the file after writing

2. **file_put_contents()**:
    - A simpler way to write directly into a file.
    - It overwrites the content by default.

    Example:

    php

Copy code
file_put_contents("example.txt", "This is another way to write.");

3. **Appending Data**:
   - Use a mode in fopen() or
     pass FILE_APPEND in file_put_contents() to add data without overwriting.

   Example with file_put_contents():

   php
   Copy code
   file_put_contents("example.txt", " Adding more content.",
   FILE_APPEND);

These functions are useful for creating or updating files in PHP.

---

10. Explain PHP Session and Cookie Functions with Examples (5 Marks)

Session Functions:

Sessions store temporary data on the server for each user.

1. **Start a Session**:
   - Use session_start() to initialize a session.
   - Data is stored in the $_SESSION array.

   Example:

   php
   Copy code
   session_start();
   $_SESSION['username'] = "John";
   echo $_SESSION['username']; // Output: John

2. **Destroy a Session**:
   - Use session_destroy() to end a session.

   Example:

   php
   Copy code
   session_start();

session_destroy();

Cookie Functions:

Cookies store small data on the user's browser.

1. **Set a Cookie**:
   - Use setcookie() to create a cookie.

   Example:

   php
   Copy code
   setcookie("user", "John", time() + (86400 * 7)); // Cookie valid for 7 days

2. **Access a Cookie**:
   - Use $_COOKIE to retrieve cookie data.

   Example:

   php
   Copy code
   echo $_COOKIE['user']; // Output: John (if the cookie is set)

3. **Delete a Cookie**:
   - Set the cookie's expiration time in the past.

   Example:

   php
   Copy code
   setcookie("user", "", time() - 3600); // Deletes the cookie

Key Difference: Sessions are stored on the server, while cookies are stored in the user's browser. Both are used for maintaining user data.

11. Explain PHP Abstract Class and Method (5 Marks)

Abstract Class

An **abstract class** in PHP is a class that cannot be instantiated on its own. It is meant to serve as a base class for other classes. An abstract class can have **abstract methods** (methods without implementation) and **concrete methods**(methods with implementation).

Syntax for Abstract Class

php
Copy code

```
abstract class AbstractClassName {
    // Abstract method (must be implemented in the child class)
    abstract public function abstractMethod();

    // Concrete method (can be used directly by child class)
    public function concreteMethod() {
        echo "This is a concrete method.";
    }
}
```

Abstract Method

An **abstract method** is defined in the abstract class without implementation. Any class inheriting the abstract class must implement the abstract methods.

Syntax for Abstract Method:

php
Copy code

```
abstract public function methodName();
```

Example of Abstract Class and Method

php
Copy code

```php
abstract class Animal {
    // Abstract method
    abstract public function makeSound();

    // Concrete method
    public function sleep() {
        echo "Sleeping...";
    }
}

// Child class inheriting Animal
class Dog extends Animal {
    public function makeSound() {
        echo "Woof!";
    }
}

class Cat extends Animal {
    public function makeSound() {
        echo "Meow!";
    }
}

// Using the classes
$dog = new Dog();
$dog->makeSound(); // Output: Woof!
$dog->sleep(); // Output: Sleeping...

$cat = new Cat();
$cat->makeSound(); // Output: Meow!
$cat->sleep(); // Output: Sleeping...
```

Key Points

1. **Abstract Class**: Cannot be instantiated directly.
2. **Abstract Method**: Must be implemented in child classes.
3. **Concrete Methods**: Can be directly used by child classes without needing overrides.

---

12. Explain PHP Inheritance (5 Marks)

Inheritance

Inheritance is an **object-oriented programming concept** that allows one class (child class) to inherit properties and methods from another class (parent class). This promotes **reusability** and creates a hierarchical relationship between classes.

Types of Inheritance in PHP

1. **Single Inheritance**: A child class inherits from a single parent class.
2. **Multiple Inheritance**: PHP doesn't support multiple inheritance directly, but traits can simulate this behavior.
3. **Multilevel Inheritance**: A class inherits from a child class, which itself inherits from another parent class.
4. **Hierarchical Inheritance**: Multiple child classes inherit from the same parent class.

Example of Single Inheritance

```
php
Copy code
// Parent class
class Vehicle {
    public $brand;

    public function __construct($brand) {
        $this->brand = $brand;
    }

    public function displayBrand() {
        echo "Vehicle brand: " . $this->brand;
    }
}

// Child class inheriting Vehicle
class Car extends Vehicle {
```

```
    public $model;

    public function __construct($brand, $model) {
        parent::__construct($brand);
        $this->model = $model;
    }

    public function displayDetails() {
        echo "Brand: " . $this->brand . ", Model: " . $this->model;
    }
}

// Create an instance of Car
$myCar = new Car("Toyota", "Corolla");
$myCar->displayBrand(); // Output: Vehicle brand: Toyota
echo "<br>";
$myCar->displayDetails(); // Output: Brand: Toyota, Model: Corolla
```

Explanation of Example

1. **Parent Class** (Vehicle) has a common property $brand and a method displayBrand.
2. **Child Class** (Car) inherits from the Vehicle class.
    - It uses parent::__construct($brand) to access and initialize the parent class's properties.
3. The child class has additional properties ($model) and methods (displayDetails).
4. You can call both inherited and new methods from the child class instance.

Key Points of Inheritance

1. **Child classes can access public and protected properties and methods from their parent classes.**
2. **The parent keyword** is used to call methods or constructors from the parent class.
3. It promotes **code reusability**, reducing code duplication.
4. PHP supports **single inheritance** natively but can use traits for multiple inheritance-like functionality.

13. What is Laravel? 14. How does Laravel implement the MVC design?

:

13. What is Laravel? (5 Marks)

Laravel is a **PHP framework** used for building web applications. It follows the **Model-View-Controller (MVC)**design pattern and offers tools to simplify development.

Key Features of Laravel:

1. **Easy Routing**: Laravel makes it simple to define and manage routes.
2. **Blade Template Engine**: Provides a lightweight and powerful templating system for views.
3. **Eloquent ORM**: Simplifies database interaction using models.
4. **Built-in Authentication**: Easy-to-implement user login and registration features.
5. **Migration System**: Helps in managing database schema without writing raw SQL.
6. **Artisan CLI**: A command-line tool to perform tasks like creating controllers, models, etc.

Laravel is popular because it provides a clean and efficient structure for creating modern web applications.

---

14. How Does Laravel Implement the MVC Design? (5 Marks)

Laravel uses the **Model-View-Controller (MVC)** pattern to separate an application into three interconnected components:

1. **Model**:
   - Represents the data and business logic.
   - Interacts with the database using Laravel's Eloquent ORM. **Example**:

   php
   Copy code
   class User extends Model {

```
    protected $table = 'users';
}
```

2. **View**:
    - Handles the presentation of data.
    - Uses Blade, Laravel's template engine, to create dynamic views. **Example** (Blade file: welcome.blade.php):

```
html
Copy code
<h1>Hello, {{ $name }}</h1>
```

3. **Controller**:
    - Connects the Model and View.
    - Processes user requests and returns the appropriate response. **Example**:

```
php
Copy code
class UserController extends Controller {
    public function show($id) {
        $user = User::find($id);
        return view('user.profile', ['user' => $user]);
    }
}
```

How it Works:

1. The **user sends a request** (e.g., URL access).
2. The **controller** processes the request and interacts with the **model** for data.
3. The **view** displays the data to the user.

This structure ensures organized code, making it easier to maintain and scale applications.

15. What are the pros and cons of using Laravel? 16. Explain Laravel application structure.

:

15. What Are the Pros and Cons of Using Laravel? (5 Marks)

Pros of Laravel:

1. **MVC Architecture**: Provides clear separation of concerns, making the application organized and maintainable.
2. **Built-in Features**: Includes authentication, routing, and caching, saving development time.
3. **Eloquent ORM**: Simplifies database interaction with an easy-to-use query builder.
4. **Blade Template Engine**: Allows dynamic views with minimal overhead.
5. **Community Support**: Has extensive documentation and a large community for help and packages.

Cons of Laravel:

1. **Performance**: Not as fast as some lightweight frameworks for high-performance applications.
2. **Learning Curve**: Beginners may find it challenging due to its many features and concepts.
3. **Heavy Framework**: May be overkill for simple or small-scale projects.
4. **Frequent Updates**: Regular updates can require adjustments to existing code.

---

16. Explain Laravel Application Structure (5 Marks)

Laravel's application structure is organized into a set of directories. Below are the main directories and their purposes:

1. **app/**:
    - Contains the core application logic.
    - Subdirectories include:
        - Models/: Stores Eloquent models for database interactions.
        - Http/Controllers/: Contains controllers for handling requests.
        - Http/Middleware/: Stores middleware for request filtering.
2. **routes/**:
    - Defines application routes.
    - Common files:
        - web.php: Routes for web applications.
        - api.php: Routes for APIs.

3. **resources/**:
    - Holds views, language files, and frontend assets.
        - views/: Contains Blade templates.
        - lang/: Stores language files for localization.
4. **database/**:
    - Contains database migrations, seeders, and factories.
        - migrations/: Defines schema changes.
        - seeders/: Populates database with initial data.
5. **public/**:
    - The entry point for the application.
    - Stores public assets like images, JavaScript, and CSS files.
6. **config/**:
    - Contains configuration files for the application (e.g., database, mail).
7. **storage/**:
    - Stores logs, cache, and compiled Blade templates.
8. **bootstrap/**:
    - Initializes the framework, including loading configuration and starting the application.

Summary:

Laravel's structure ensures clean organization, making it easier to develop, manage, and scale web applications.

17. Write a PHP script to insert data. 18. Write the advantages of PDO prepared statements.

:

17. Write a PHP Script to Insert Data (5 Marks)

Below is an example PHP script that inserts data into a MySQL database using **MySQLi**:

php
Copy code
<?php

```
// Database connection
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "my_database";

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Data to insert
$name = "John Doe";
$email = "john@example.com";

// SQL query to insert data
$sql = "INSERT INTO users (name, email) VALUES ('$name', '$email')";

if ($conn->query($sql) === TRUE) {
    echo "New record inserted successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}

// Close connection
$conn->close();
?>
```

Notes:

- Replace my_database and users with your database and table name.
- Use prepared statements to prevent SQL injection in real-world applications.

---

18. Write the Advantages of PDO Prepared Statements (5 Marks)

PDO (PHP Data Objects) prepared statements offer several advantages:

1. **Prevents SQL Injection**:

    - Automatically escapes user input, reducing the risk of malicious SQL queries.
2. **Reusability**:
    - The same prepared statement can be executed multiple times with different data, improving performance.
3. **Supports Multiple Databases**:
    - Works with various databases (MySQL, PostgreSQL, SQLite, etc.) without changing much code.
4. **Type Safety**:
    - Automatically handles data types (e.g., strings, integers) when binding values.
5. **Improved Performance**:
    - Parsing and planning the SQL query occurs once, saving processing time for repeated executions.

Example:

php
Copy code

```
<?php
$pdo = new PDO("mysql:host=localhost;dbname=my_database", "root", "");

// Prepared statement
$stmt = $pdo->prepare("INSERT INTO users (name, email) VALUES (:name,
:email)");

// Bind parameters
$stmt->bindParam(':name', $name);
$stmt->bindParam(':email', $email);

// Insert data
$name = "Jane Doe";
$email = "jane@example.com";
$stmt->execute();

echo "Data inserted successfully";
?>
```

Prepared statements ensure secure and efficient database operations.

19. Define the following functions: var_dump, isset, die, gettype, print_r, echo, and phpinfo. 20. Write a short note on PDO error modes used with the setAttribute method.

:

19. Define the Following Functions (5 Marks)

1. **var_dump()**:
   - Displays structured information (type and value) about a variable, including its data type and length.
   - Useful for debugging purposes.
   - **Example**:

     php
     Copy code
     $name = "John";
     var_dump($name);
     // Output: string(4) "John"

---

2. **isset()**:
   - Checks if a variable is set and is not null.
   - Returns true if the variable exists; otherwise, returns false.
   - **Example**:

     php
     Copy code
     $name = "John";
     if (isset($name)) {
     echo "Variable is set";
     }

---

3. **die()**:
   - Terminates the script execution at the point it's called.
   - Often used for error handling or debugging.
   - **Example**:

     php
     Copy code
     if (!$db_connection) {

```
    die("Connection failed");
}
```

---

4. **gettype()**:
   - Returns the data type of a variable as a string.
   - **Example**:

     ```
     php
     Copy code
     $age = 25;
     echo gettype($age); // Output: integer
     ```

---

5. **print_r()**:
   - Prints human-readable information about a variable, especially arrays or objects.
   - Useful for debugging.
   - **Example**:

     ```
     php
     Copy code
     $array = [1, 2, 3];
     print_r($array);
     // Output:
     // Array
     // (
     //     [0] => 1
     //     [1] => 2
     //     [2] => 3
     // )
     ```

---

6. **echo**:
   - Outputs text or variable values to the browser.
   - **Example**:

     ```
     php
     Copy code
     $name = "John";
     echo $name; // Output: John
     ```

---

7. **phpinfo()**:

- Displays information about the current PHP configuration, loaded modules, version, environment, and other details.
- **Example**:

  php
  Copy code
  phpinfo();

---

20. Write a Short Note on PDO Error Modes Used with the setAttribute Method (5 Marks)

The **PDO (PHP Data Objects)** class uses error modes to handle database errors in different ways. These modes are set using the setAttribute method.

PDO Error Modes:

1. **PDO::ERRMODE_SILENT**:
   - Default mode.
   - Errors are not displayed to the user.
   - You need to check for errors manually using $pdo->errorInfo().
   - Example:

     php
     Copy code
     $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_SILENT);

---

2. **PDO::ERRMODE_WARNING**:
   - Generates warnings for errors.
   - Useful during debugging because warnings provide error details.
   - Example:

     php
     Copy code
     $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING);

---

3. **PDO::ERRMODE_EXCEPTION**:
   - Throws exceptions for database errors.
   - Recommended for production because you can use try-catch blocks to handle exceptions.

- Example:

```
php
Copy code
try {
    $pdo->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
    $pdo->query("SELECT * FROM users");
} catch (PDOException $e) {
    echo "Error: " . $e->getMessage();
}
```

Summary:

- **PDO::ERRMODE_SILENT**: No error message unless explicitly checked.
- **PDO::ERRMODE_WARNING**: Generates warnings for easier debugging.
- **PDO::ERRMODE_EXCEPTION**: Uses exceptions for robust error handling.

Setting the appropriate error mode using setAttribute ensures proper error handling depending on the environment (development or production).

21. Describe the various methods of fetching data from a MySQL database in PHP. 22. Describe the features of PDO.

:

21. Describe the Various Methods of Fetching Data from a MySQL Database in PHP (5 Marks)

There are several methods in PHP to fetch data from a MySQL database after executing a query. Below are the most common ones:

---

1. Using mysqli_fetch_assoc()

- Fetches the result row as an associative array.
- The keys of the array correspond to column names.

Example:

```
php
Copy code
$conn = new mysqli("localhost", "root", "", "my_database");
$result = $conn->query("SELECT * FROM users");

while ($row = mysqli_fetch_assoc($result)) {
   echo $row['name'] . "<br>";
}
```

---

2. Using mysqli_fetch_array()

- Fetches the result row as both an associative array and a numeric array.

Example:

```
php
Copy code
while ($row = mysqli_fetch_array($result)) {
   echo $row[0] . " - " . $row['name'] . "<br>";
}
```

---

3. Using mysqli_fetch_object()

- Fetches the result as an object where column names are treated as properties.

Example:

```
php
Copy code
while ($row = mysqli_fetch_object($result)) {
   echo $row->name . "<br>";
}
```

---

4. Using PDO::FETCH_ASSOC with PDO

- Fetches data as an associative array using PDO.

Example:

```
php
Copy code
$pdo = new PDO("mysql:host=localhost;dbname=my_database", "root", "");
$stmt = $pdo->prepare("SELECT * FROM users");
$stmt->execute();

while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
   echo $row['name'] . "<br>";
}
```

---

5. Using PDO::FETCH_OBJ with PDO

- Fetches data as an object where column names are object properties.

Example:

```
php
Copy code
while ($row = $stmt->fetch(PDO::FETCH_OBJ)) {
   echo $row->name . "<br>";
}
```

---

6. Using PDO::FETCH_NUM with PDO

- Fetches data as a numeric array where columns are indexed numerically.

Example:

```
php
Copy code
while ($row = $stmt->fetch(PDO::FETCH_NUM)) {
   echo $row[0] . "<br>";
}
```

Summary of Fetch Methods:

Method	Type	Example Data Structure
mysqli_fetch_assoc()	Associative Array	$row['column_name']
mysqli_fetch_array()	Associative + Numeric Array	$row[0], $row['column_name']
mysqli_fetch_object()	Object	$row->column_name
PDO::FETCH_ASSOC	Associative Array	$row['column_name']
PDO::FETCH_OBJ	Object	$row->column_name
PDO::FETCH_NUM	Numeric Array	$row[0]

These methods allow flexibility depending on how you want to access the fetched data.

---

22. Describe the Features of PDO (5 Marks)

PDO (PHP Data Objects) is a database access layer in PHP that provides a unified interface for working with various databases. It simplifies database operations by using object-oriented programming.

Features of PDO:

1. **Database Support**:
   - PDO works with multiple database systems like MySQL, PostgreSQL, SQLite, Oracle, and more.
2. **Prepared Statements**:
   - Helps prevent SQL injection by allowing safe data binding.
   - **Example**:

     php
     Copy code
     $stmt = $pdo->prepare("SELECT * FROM users WHERE id = :id");
     $stmt->bindParam(':id', $id);
     $stmt->execute();

3. **Error Handling**:
   - PDO supports error handling using exceptions with PDO::ERRMODE_EXCEPTION.
   - Errors can be caught and managed using try-catch.
4. **Transactions**:
   - PDO supports database transactions to ensure multiple queries are executed safely.

- **Example**:

  php
  Copy code
  $pdo->beginTransaction();
  $pdo->query("INSERT INTO users (name) VALUES ('John')");
  $pdo->query("INSERT INTO users (email) VALUES ('john@example.com')");
  $pdo->commit();

5. **Fetching Modes**:
   - PDO allows different modes for fetching data:
     - PDO::FETCH_ASSOC: Fetch as associative arrays.
     - PDO::FETCH_OBJ: Fetch as objects.
     - PDO::FETCH_NUM: Fetch as numeric arrays.
6. **Portability**:
   - Code written with PDO is database-agnostic, meaning you can switch databases without rewriting most of the code.
7. **Secure Data Access**:
   - PDO's prepared statements and bound parameters ensure secure communication with the database.

Summary:

PDO provides a secure, unified, and object-oriented way to interact with databases. It has built-in support for multiple databases, prevents SQL injection through prepared statements, supports transactions, and offers flexible data fetching methods.

23. Write a PHP script to perform database CRUD operations. 24. Explain the usage of $_POST, $_GET, and $_REQUESTwith examples.

:

23. Write a PHP Script to Perform Database CRUD Operations (5 Marks)

Below is a PHP script that performs **CRUD (Create, Read, Update, Delete)** operations using **MySQLi**:

Database Table Structure

Assume we have a database my_database and a table users with the following structure:

sql
Copy code
```
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL
);
```

CRUD Operations Script

php
Copy code
```
<?php
// Database connection
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "my_database";

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Create Operation
if (isset($_POST['create'])) {
    $name = $_POST['name'];
    $email = $_POST['email'];
    $conn->query("INSERT INTO users (name, email) VALUES ('$name',
'$email')");
    echo "User created successfully<br>";
}
```

```
// Read Operation
echo "<h3>All Users:</h3>";
$result = $conn->query("SELECT * FROM users");
while ($row = $result->fetch_assoc()) {
    echo "ID: " . $row['id'] . " | Name: " . $row['name'] . " | Email: " .
$row['email'] . "<br>";
}

// Update Operation
if (isset($_POST['update'])) {
    $id = $_POST['id'];
    $name = $_POST['name'];
    $email = $_POST['email'];
    $conn->query("UPDATE users SET name='$name', email='$email' WHERE
id=$id");
    echo "User updated successfully<br>";
}

// Delete Operation
if (isset($_POST['delete'])) {
    $id = $_POST['id'];
    $conn->query("DELETE FROM users WHERE id=$id");
    echo "User deleted successfully<br>";
}

// Close connection
$conn->close();
?>

<!-- Create Form -->
<h3>Create User</h3>
<form method="POST">
    Name: <input type="text" name="name" required>
    Email: <input type="email" name="email" required>
    <input type="submit" name="create" value="Create">
</form>

<!-- Update Form -->
<h3>Update User</h3>
<form method="POST">
    ID: <input type="number" name="id" required>
    Name: <input type="text" name="name" required>
    Email: <input type="email" name="email" required>
```

```
    <input type="submit" name="update" value="Update">
</form>

<!-- Delete Form -->
<h3>Delete User</h3>
<form method="POST">
    ID: <input type="number" name="id" required>
    <input type="submit" name="delete" value="Delete">
</form>
```

Explanation

1. **Create**: Adds a new user to the database table users.
2. **Read**: Retrieves and displays all users from the database.
3. **Update**: Updates a user's name and email based on their id.
4. **Delete**: Deletes a user from the database based on their id.

Note

- This script uses basic forms (POST) for performing each CRUD operation.
- Always use **prepared statements** to prevent SQL injection in production scripts.

---

24. Explain the Usage of $_POST, $_GET, and $_REQUEST with Examples (5 Marks)

---

1. $_POST

- **Definition**: $_POST is a superglobal array used to collect form data sent via the HTTP POST method.
- It is secure because data is sent in the request body, not visible in the URL.

Example:

php

Copy code
```
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $name = $_POST['name'];
    echo "Name submitted: " . $name;
}
?>

<form method="POST">
    Name: <input type="text" name="name">
    <input type="submit" value="Submit">
</form>
```

---

2. $_GET

- **Definition**: $_GET is a superglobal array used to collect query string data sent via the HTTP GET method.
- Data is visible in the URL.

Example:

php
Copy code
```
if (isset($_GET['name'])) {
    $name = $_GET['name'];
    echo "Name received from URL: " . $name;
}
?>

<a href="index.php?name=John">Send Name via GET</a>
```

---

3. $_REQUEST

- **Definition**: $_REQUEST is a superglobal array that combines $_GET, $_POST, and $_COOKIE data.
- It checks data from any of these methods, depending on how the request was sent.

Example:

php
Copy code
```
if (isset($_REQUEST['name'])) {
```

```
    echo "Name from REQUEST: " . $_REQUEST['name'];
}
?>

<form method="POST">
    Name: <input type="text" name="name">
    <input type="submit" value="Submit">
</form>
<a href="index.php?name=Jane">Send Name via GET</a>
```

Key Differences

Superglobal	Method	Scope	Visibility
$_POST	POST method	Data sent via POST request	Not visible in URL
$_GET	GET method	Data sent via URL query string	Visible in URL
$_REQUEST	GET, POST, Cookies	Combines $_GET, $_POST, $_COOKIE	Depends on request type

---

When to Use Which?

1. **Use $_POST** when dealing with sensitive data like login credentials.
2. **Use $_GET** when data is meant to be shared in the URL, like search parameters.
3. **Use $_REQUEST** when you want to access data regardless of the request method but avoid overusing it unless necessary.

25. Explain conditional statement structures in PHP with suitable examples. 26. Explain PHP's various types of loops with examples.

:

---

25. Explain Conditional Statement Structures in PHP with Suitable Examples (5 Marks)

In PHP, **conditional statements** allow you to execute different blocks of code depending on whether a condition is true or false. The most commonly used conditional statement structures are:

---

1. if Statement

The if statement is used to execute a block of code if a condition is true.

Syntax:

```
php
Copy code
if (condition) {
    // code to execute if condition is true
}
```

Example:

```
php
Copy code
$age = 20;
if ($age >= 18) {
    echo "You are eligible to vote.";
}
```

---

2. if-else Statement

The if-else statement allows you to execute one block of code if a condition is true and another block if the condition is false.

Syntax:

```
php
Copy code
if (condition) {
    // code if true
} else {
    // code if false
```

}

Example:

php
Copy code
```
$age = 16;
if ($age >= 18) {
    echo "You are eligible to vote.";
} else {
    echo "You are not eligible to vote.";
}
```

---

3. if-else if-else Statement

The if-else if-else structure allows for multiple conditions to be checked in sequence.

Syntax:

php
Copy code
```
if (condition1) {
    // code if condition1 is true
} elseif (condition2) {
    // code if condition2 is true
} else {
    // code if none of the conditions is true
}
```

Example:

php
Copy code
```
$grade = 85;

if ($grade >= 90) {
    echo "You got an A.";
} elseif ($grade >= 75) {
    echo "You got a B.";
} elseif ($grade >= 50) {
    echo "You got a C.";
} else {
```

```
    echo "You failed.";
}
```

---

4. switch Statement

The switch statement is used when multiple conditions need to be compared against a single variable.

Syntax:

```
php
Copy code
switch (variable) {
    case value1:
        // code block for value1
        break;
    case value2:
        // code block for value2
        break;
    default:
        // code if no case matches
}
```

Example:

```
php
Copy code
$day = "Monday";

switch ($day) {
    case "Monday":
        echo "Today is Monday.";
        break;
    case "Tuesday":
        echo "Today is Tuesday.";
        break;
    default:
        echo "It's another day.";
}
```

Summary of Conditional Statements

Statement	Description	Example
if	Executes if the condition is true.	if ($x > 10) { echo "x is greater"; }
if-else	Executes one block of code if true, another if false.	if ($x > 10) { echo "x is large"; } else { echo "x is small"; }
if-else if-else	Tests multiple conditions in sequence.	if ($x > 20) { echo "high"; } elseif ($x > 10) { echo "medium"; } else { echo "low"; }
switch	Tests a variable against multiple values.	switch($day) { case 'Monday': echo 'Today is Monday'; break; }

---

26. Explain PHP's Various Types of Loops with Examples (5 Marks)

Loops are used in PHP to execute a block of code repeatedly based on a condition. PHP supports several types of loops:

---

1. for Loop

The for loop is used when the number of iterations is known in advance.

Syntax:

```
php
Copy code
for (initialization; condition; increment) {
   // code to execute
}
```

Example:

```
php
Copy code
for ($i = 1; $i <= 5; $i++) {
   echo "Number: $i <br>";
}
```

---

2. while Loop

The while loop runs as long as the given condition is true.

Syntax:

php
Copy code
```
while (condition) {
    // code to execute
}
```

Example:

php
Copy code
```
$i = 1;
while ($i <= 5) {
    echo "Number: $i <br>";
    $i++;
}
```

---

3. do-while Loop

The do-while loop is similar to the while loop but executes the loop **at least once**, even if the condition is false on the first check.

Syntax:

php
Copy code
```
do {
    // code to execute
} while (condition);
```

Example:

php
Copy code
```
$i = 1;
do {
    echo "Number: $i <br>";
    $i++;
} while ($i <= 5);
```

---

4. foreach Loop

The foreach loop is specifically used for iterating over arrays.

Syntax:

```
php
Copy code
foreach ($array as $value) {
   // code to execute
}
```

Example:

```
php
Copy code
$colors = ["Red", "Blue", "Green", "Yellow"];
foreach ($colors as $color) {
   echo "Color: $color <br>";
}
```

Summary of Loops

Loop	Description	Example
for	Runs a block of code a set number of times.	for ($i = 0; $i < 5; $i++) { echo $i; }
while	Runs a block of code as long as a condition is true.	while ($i < 5) { echo $i; $i++; }
do-while	Executes a block of code at least once, then checks condition.	do { echo $i; $i++; } while ($i < 5);
foreach	Used to loop through elements of an array.	foreach ($array as $value) { echo $value; }

These looping structures make it easier to perform repetitive tasks efficiently in PHP programming.

27. Explain PHP user-defined functions. 28. How
do include, include_once, require, and require_once work, and how do they differ from each other?

---

27. Explain PHP User-Defined Functions (5 Marks)

In PHP, **user-defined functions** are custom functions that you create to perform specific tasks. These functions allow you to reuse code, simplify programming logic, and make your code easier to maintain.

User-defined functions can take input parameters and return data as needed.

Syntax for Creating a User-Defined Function

```
php
Copy code
function functionName($parameter1, $parameter2) {
    // Code to execute
    return $result; // Optional: Return a value
}
```

Example of a User-Defined Function

Here's an example of a user-defined function that adds two numbers:

```
php
Copy code
// User-defined function
function addNumbers($num1, $num2) {
    return $num1 + $num2;
}

// Calling the function
$result = addNumbers(5, 10);
echo "The result is: " . $result;
```

Types of User-Defined Functions

1. **Simple Functions**: Functions without parameters.

   php
   Copy code
   function sayHello() {
   echo "Hello, World!";
   }
   sayHello();

---

2. **Functions with Parameters**: Functions that accept input values.

   php
   Copy code
   function greet($name) {
   echo "Hello, " . $name . "!";
   }
   greet("Alice");

---

3. **Functions with Default Parameters**: Parameters with default values.

   php
   Copy code
   function greet($name = "Guest") {
   echo "Hello, " . $name . "!";
   }
   greet(); // Uses default
   greet("John"); // Uses provided value

---

4. **Functions Returning Values**: Functions can return data.

   php
   Copy code
   function square($num) {
   return $num * $num;
   }
   $result = square(4);
   echo "The square is: " . $result;

Benefits of User-Defined Functions

1. **Reusability**: Write the logic once and call it multiple times.
2. **Readability**: Makes the code modular and easier to read.
3. **Maintainability**: Changes to logic only need to be made in one location.
4. **Scalability**: Simplifies debugging and testing by isolating logic.

---

28. How do include, include_once, require, and require_once Work, and How Do They Differ? (5 Marks)

These four PHP statements are used to include external files into your script. They allow you to reuse code across multiple PHP files.

---

1. include

- Includes and evaluates a specified file.
- If the file is not found, it generates a **warning**, but the script continues execution.

Example:

php
Copy code
include 'header.php';
echo "Main content here.";

---

2. include_once

- Includes and evaluates the file, **only if it hasn't already been included**.
- Avoids including the file multiple times.

Example:

php
Copy code
include_once 'header.php';
include_once 'header.php'; // Will only include once

---

3. require

- Includes and evaluates a specified file.
- If the file is not found, it generates a **fatal error**, and the script stops execution.

Example:

php
Copy code
```
require 'config.php';
echo "Connected to database.";
```

---

4. require_once

- Includes and evaluates the file, **only if it hasn't already been included**.
- If the file is not found, it generates a **fatal error**, and the script stops execution.

Example:

php
Copy code
```
require_once 'config.php';
require_once 'config.php'; // Will only include once
```

Key Differences

Function	Behavior if the File is Missing	Will Stop Script?	Prevents Multiple Includes?
include	Generates a **warning**, but script continues.	No	No
include_once	Generates a **warning**, but script continues.	No	Yes
require	Generates a **fatal error** and stops the script.	Yes	No
require_once	Generates a **fatal error** and stops the script.	Yes	Yes

---

When to Use Which?

1. Use **require** when the file is essential for the script to run. If it's missing, you don't want the script to continue.
2. Use **include** for optional files that can be absent without affecting core logic.
3. Use **require_once** or **include_once** when you want to ensure the file is included only once to avoid conflicts or repeated execution.