1. Single-Quoted vs. Double-Quoted Strings in PHP

Single Quoted (`'`).	**Double Quoted (`"`).**
Single quoted string literals returns the string as it is: e.g $x = "John"; echo 'Hello $x'; The output is: ***Hello $x***	Used to process special characters e.g $x = "John"; echo "Hello $x"; **The output is: Hello John**
Escape sequences are limited (e.g., `\\` and `\'`).	Supports a wider range of escape sequences (e.g., `\n`, `\t`, `\$`).
We use single-quoted strings when we don't need variable interpolation or special escape sequences.	We use double-quoted strings when we need an interpolation or special characters.

2. Describe the principles of Object-Oriented Programming (OOP) in PHP. How do you define a class and create objects in PHP? Provide an example of a class and its instantiation.

OOP -> Object oriented programming involves creating objects that contain both data and functions as opposed to procedural programming that deals with writing functions or procedures that perform actions on the data.

It involves the use of classes and objects;

Class: A class is a user-defined data type. It consists of data members and member functions, which can be accessed and used by creating an instance of that class. It represents the set of properties or methods that are common to all objects of one type. A class is like a blueprint for an object.

Object: It is a basic unit of Object-Oriented Programming and represents the real-life entities. An Object is an instance of a Class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated. An object has an identity, state, and behavior. Each object contains data and code to manipulate the data. Objects can interact without having to know details of each other's data or code, it is sufficient to know the type of message accepted and type of response returned by the objects.

Principles of OOP:

Encapsulation: Encapsulation is defined as the wrapping up of data under a single unit. It is the mechanism that binds together code and the data it manipulates. In Encapsulation, the variables or data of a class are hidden from any other class and can be accessed only through any member function of their class in which they are declared. As in encapsulation, the data in a class is hidden from other classes, so it is also known as data-hiding.

Inheritance: Inheritance is an important pillar of OOP (Object-Oriented Programming). The capability of a class to derive properties and characteristics from another class is called Inheritance. When we write a class, we inherit properties from other classes. So when we create a class, we do not need to write all the properties and functions again and again, as these can be inherited from another class that possesses it. Inheritance allows the user to reuse the code whenever possible and reduce its redundancy.

Polymorphism: The word polymorphism means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form. For example, a person at the same time can have different characteristics. Like a man at the same time is a father, a husband, an employee. So the same person possess different behavior in different situations.

Abstraction: Data abstraction is one of the most essential and important features of object-oriented programming. Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation. Consider a real-life example of a man driving a car. The man only knows that pressing the accelerators will increase the speed of the car or applying brakes will stop the car, but he does not know about how on pressing the accelerator the speed is increasing, he does not know about the inner mechanism of the car or the implementation of the accelerator, brakes, etc. in the car. This is what abstraction is.

Defining a Class and Creating Objects:

https://github.com/JosephKithome/DPOpaytest/blob/master/index.php

```php
<?php
class Candidate {
  // Properties
  public $first_name;
  public $last_name;
  public $email;
  public $phone;

  // Getter Methods
  function get_first_name() {
    return $this->first_name;
  }
  function get_last_name() {
    return $this->last_name;
  }
  function get_email() {
    return $this->email;
  }
  function get_phone() {
    return $this->phone;
  }
```

```php
// Setter Methods
  function set_first_name($first_name) {
    $this->first_name = $first_name;
  }
  function set_last_name($last_name) {
    $this->last_name = $last_name;
  }
  function set_email($email) {
    $this->email = $email;
  }
  function set_phone($phone) {
    $this->phone = $phone;
  }
}

$c = new Candidate();
$c->set_first_name('Joseph');
$c->set_last_name('Kithome');
$c ->set_email('josephkithome.jmk@gmail.com');
$c->set_phone('254717064174');

echo $c->get_first_name();
echo "<br>";
echo $c->get_last_name();
echo "<br>";
echo $c->get_email();
echo "<br>";
echo $c->get_phone();
echo "<br>";

// Output
// Joseph
// Kithome
// josephkithome.jmk@gmail.com
// 254717064174
```

3. Explain the purpose of exception handling in PHP. How do you catch and handle exceptions in your code? Provide an example of how you would use try-catch blocks.

Purpose:

Exception handling allows you to manage errors gracefully, ensuring the program can handle unexpected conditions without crashing.

Try-Catch Example:

https://github.com/JosephKithome/DPOpaytest/blob/master/exception.php

```php
<?php
class Candidate {
  // Properties
  public $first_name;
  public $last_name;
  public $email;
  public $phone;

  // Getter Methods
  function get_first_name() {
```

```php
    return $this->first_name;
  }
  function get_last_name() {
    return $this->last_name;
  }
  function get_email () {
    return $this->email;
  }
  function get_phone () {
    return $this->phone;
  }

  // Setter Methods
  function set_first_name ($first_name) {
    $this->first_name = $first_name;
  }
  function set_last_name ($last_name) {
    $this->last_name = $last_name;
  }
  function set_email ($email) {
    // Basic email validation
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
      throw new Exception('Invalid email format');
    }
    $this->email = $email;
  }
  function set_phone($phone) {
    // Basic phone validation
    if (!preg_match('/^[0-9]{10,15}$/', $phone)) {
      throw new Exception('Invalid phone number format');
    }
    $this->phone = $phone;
  }
}

try {
  $c = new Candidate();
  $c->get_first_name('Joseph');
  $c->get_last_name('Kithome');
  $c->set_email('josephkithome.jmkgmail.com');
  $c->set_phone('254717064174');

  echo $c->get_first_name();
  echo "<br>";
  echo $c->get_last_name();
  echo "<br>";
  echo $c->get_email();
  echo "<br>";
  echo $c->get_phone();
  echo "<br>";

  // Output
  // Joseph
  // Kithome
  // josephkithome.jmk@gmail.com
  // 254717064174

} catch (Exception $e) {
  echo 'Caught exception: ',  $e->getMessage(), "<br>";
}
?>
```

4. Discuss different methods for connecting to a database in PHP. Describe the differences between MySQLi and PDO. Provide an example of how to perform a basic database query using one of these methods.

MySQLi vs. PDO:

MySQLi: MySQL Improved extension, supports procedural and object-oriented programming and only works with MySQL databases.

PDO: PHP Data Objects, supports multiple databases (MySQL, PostgreSQL, SQLite, etc.), only object-oriented and if you have to switch a database you only need to update the connection string.

Example using PDO:

https://github.com/JosephKithome/DPOpaytest/blob/master/Pdo.php

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "macmobile";

try {
  $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
  // set the PDO error mode to exception
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  echo "Connected successfully ";
  echo "Fetching data.... <br>";
  $sql = "SELECT ch.call_type, ch.customer_group, COUNT(*) as total
            FROM call_history as ch
            GROUP BY ch.call_type, ch.customer_group";

    $stmt = $conn->query($sql);
    $result = $stmt->fetchAll(PDO::FETCH_ASSOC);

    header('Content-Type: application/json');
    echo json_encode($result);
} catch(PDOException $e) {
  echo "Connection failed: " . $e->getMessage();
}
?>
```

5. How would you protect a PHP application from common security vulnerabilities such as SQL injection and cross-site scripting (XSS)? Provide code examples or best practices for mitigating these threats.

-Use prepared statements with parameterized queries to prevent SQL Injection.

https://github.com/JosephKithome/DPOpaytest/blob/master/Preparedstmt.php

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "macmobile";

try {
  $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
  // set the PDO error mode to exception
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  echo "Connected successfully ";
  echo "Fetching data.... <br>";

  $sql = "SELECT ch.call_type, ch.customer_group, COUNT(*) as total
            FROM call_history as ch
            GROUP BY ch.call_type, ch.customer_group";

  $stmt = $conn->prepare($sql); // prepare the SQL statement
  $stmt->execute(); // execute the prepared statement

  $result = $stmt->fetchAll(PDO::FETCH_ASSOC);

  header('Content-Type: application/json');
  echo json_encode($result);
} catch(PDOException $e) {
  echo "Connection failed: " . $e->getMessage();
}
?>
```

- Sanitize user inputs and escape outputs to prevent Cross-Site Scripting (XSS).

https://github.com/JosephKithome/DPOpaytest/blob/master/index.php

```php
// Setter Methods
  function set_first_name($first_name) {

    echo htmlspecialchars($first_name(), ENT_QUOTES, 'UTF-8');

    $this->first_name = $first_name;
  }
```

6. Compare and contrast the major cloud service providers (e.g., AWS, Azure, Google Cloud). Describe the advantages and use cases for each. If you were to deploy a PHP application, which cloud provider would you choose, and why?

AWS	AZURE	GCP
Advantages: Extensive services, global reach, strong community and support. **Use Cases:** Scalable web applications, big data analytics, serverless computing.	**Advantages:** Strong integration with Microsoft products, hybrid cloud capabilities. **Use Cases:** Enterprise applications, IoT solutions, AI and machine learning.	**Advantages:** Strong in data analytics and machine learning, competitive pricing. **Use Cases:** Data processing, containerized applications, AI-driven projects.

Choice for PHP Application Deployment:

I would choose AWS Due to its extensive services, global infrastructure, and mature ecosystem.

7. Explain the concept of Infrastructure as Code and its importance in cloud infrastructure management. Provide an example of how you would define infrastructure components using a tool like Terraform or AWS CloudFormation.

Importance:

Infrastructure as code (IaC) is the ability to provision and support your computing infrastructure using code instead of manual processes and settings.

Example using Terraform

https://github.com/JosephKithome/DPOpaytest/blob/master/terraform/main.tf

```
terraform {
  required_providers {
    aws ={
        source = "hashicorp/aws"
        version = "3.5.0"
    }
```

```
  }
}

provider "aws" {
  region = "us-east-1"
}
resource "aws_instance" "DPO-test" {
    ami = "ami-011899242bb"
    instance_type = "t2.micro"
    subnet_id = "${aws_subnet.public-subnet-1.id}"
    tags = {
        Name = "DPO-test"
    }
    count = 1



}

resource "aws_budgets_budget" "like-and-subscribe" {
    name = "Monthly budget"
    budget_type = "COST"
    time_unit = "MONTHLY"
    limit_amount = "0.5"
    limit_unit = "USD"
    time_period_start = "2024-06-07"
    time_period_end = "2024-06-30"

}
```

8. Write a PHP function that takes an array of integers and returns the sum of all even numbers in the array.

https://github.com/JosephKithome/DPOpaytest/blob/master/SumEvenNumbers.php

```
<?php
class Calculator {

  public array $arr;

  public function sumEvenNumbers($arr) {
    try {
      $sum = 0;
      foreach ($arr as $value) {
        if ($value % 2 == 0) {
          $sum += $value;
        }
      }
      return $sum;
    } catch (Exception $e) {
      echo "An error occurred: " . $e->getMessage();
      return null;
    }
  }
}

$c = new Calculator();
$result = $c->sumEvenNumbers([1, 2, 3, 4, 5, 6, 7, 8, 9]);
```

```php
if ($result !== null) {
  echo "Sum of even numbers: " . $result;
  echo "<br>";
}
?>
```

9. Create a PHP script that reads a text file, counts the number of words in the file, and displays the result. Ensure that your code handles file open and read errors gracefully.

https://github.com/JosephKithome/DPOpaytest/blob/master/countWordsInFile.php

```php
<?php
class WordCounter {

  public function countWordsInFile($filePath) {
    try {
      if (!file_exists($filePath)) {
        throw new Exception("File not found.");
      }
      $content = file_get_contents($filePath);
      if ($content === false) {
        throw new Exception("Could not read file.");
      }
      $wordCount = str_word_count($content);
      return $wordCount;
    } catch (Exception $e) {
      return "Error: " . $e->getMessage();
    }
  }
}

$counter = new WordCounter();
echo $counter->countWordsInFile('words.txt');
?>
```

10.Using PHP, make a GET request to a sample REST API (e.g., JSONPlaceholder) to retrieve a list of users. Parse the JSON response and display the user's name and email address.

https://github.com/JosephKithome/DPOpaytest/blob/master/displayUsers.php

11.Describe how you would design an auto-scaling setup in AWS to handle a PHP application with fluctuating traffic. What services and features would you use, and provide a high-level architecture diagram if possible.

Prerequisite:

AWS Account: Have an active AWS account that allows you to access and manage AWS services.

Have a basic understanding of Amazon EC2 (Elastic Compute Cloud) instances, including how to launch and manage them.

Have knowledge about AWS Availability Zones, which are distinct physical locations within an AWS Region designed to provide high availability and fault tolerance.

Steps:

Step 1: Sign into the AWS console

Step 2: Search and select EC2

Step 3: Create an Auto Scaling Group

Step 5: Create a Launch Template

Step 6: Customize the Launch Template

- Select an AMI image for your EC2 instance and choose an instance type. Create a security group to control inbound and outbound traffic.

Step 7: Return to Auto Scaling Group Creation

Go back to the previous tab and click the refresh button. Select the launch template you just created. Then click next.

8. Write a PHP function that takes an array of integers and returns the sum of all even numbers in the array.

https://github.com/JosephKithome/DPOpaytest/blob/master/SumEvenNumbers.php

9. Create a PHP script that reads a text file, counts the number of words in the file, and displays the result. Ensure that your code handles file open and read errors gracefully.

https://github.com/JosephKithome/DPOpaytest/blob/master/countWordsInFile.php

10. Using PHP, make a GET request to a sample REST API (e.g., JSONPlaceholder) to retrieve a list of users. Parse the JSON response and display the user's name and email address.

https://github.com/JosephKithome/DPOpaytest/blob/master/displayUsers.php

11. Describe how you would design an auto-scaling setup in AWS to handle a PHP application with fluctuating traffic. What services and features would you use, and provide a high-level architecture diagram if possible.

Prerequisite:

AWS Account: Have an active AWS account that allows you to access and manage AWS services.

Have a basic understanding of Amazon EC2 (Elastic Compute Cloud) instances, including how to launch and manage them.

Have knowledge about AWS Availability Zones, which are distinct physical locations within an AWS Region designed to provide high availability and fault tolerance.

Steps:

Step 1: Sign into the AWS console

Step 2: Search and select EC2

Step 3: Create an Auto Scaling Group

Step 5: Create a Launch Template

Step 6: Customize the Launch Template

- Select an AMI image for your EC2 instance and choose an instance type. Create a security group to control inbound and outbound traffic.

Step 7: Return to Auto Scaling Group Creation

- Go back to the previous tab and click the refresh button. Select the launch template you just created. Then click next.

Step 8: Configure Instance Launch Options

- Choose the VPC in which you want to launch your resources. Select the desired availability zones.

Step 9: Configure Advanced Options

- In this step, we will attach a new load balancer that distributes traffic across your EC2 instances. Provide a name for the load balancer and select the "Internet-facing" option to receive traffic from the internet. Refer to the image below for visual guidance.

Step 10: Configure Group Size and Scaling Policies

- In this step, you will specify the number of instances you want in your Auto Scaling Group. Set the desired capacity, which represents the number of instances you want to maintain at any given time. Define the minimum and maximum capacity, ensuring your Auto Scaling Group operates within your desired bounds.

Step 11: Add Notifications:

Step 12: Add Tags:

In this step, we include tags to our Auto Scaling Group for better organization and management.

Step 13: Review and Confirm:

Before proceeding, please take a moment to review the options you have selected in the previous steps and ensure their accuracy. Once you have confirmed everything, click the "Create Auto Scaling Group" button to finalize the process.

Step 14: Testing the Setup

Navigate back to the EC2 dashboard.

Step 15: Testing the Load Balancer

Access the Load Balancer section from the left-hand side menu. Copy the Load Balancer DNS and test it in your browser

Advanced questions:

12. Write a PHP script that performs asynchronous processing using a message queue system like RabbitMQ or Redis. The script should receive a task (e.g., an email sending request) and process it in the background without blocking the main application. Demonstrate how you would set up the message queue and create a worker script to handle the tasks.

1. Set up RabbitMQ

Ensure RabbitMQ is installed and running. You can download RabbitMQ from the official site and follow the installation instructions for your operating system.

2. Install Composer

3. Navigate to your project directory and install php-amqplib using Composer

- Run *composer require php-amqplib/php-amqplib*

4. Create a Producer Script;

https://github.com/JosephKithome/DPOpaytest/blob/master/advanced/producer.php

5. Create a consumer(Worker) script

https://github.com/JosephKithome/DPOpaytest/blob/master/advanced/consumer.php

6. Run the following scripts to test:

- php producer.php
- php consumer.php

13.Write a PHP script that serializes a large data structure (e.g., an array or object), compresses it, saves it to a file, and then unserializes and decompresses the data from the file. You can use standard PHP functions for serialization and a compression library like zlib to achieve this.

https://github.com/JosephKithome/DPOpaytest/blob/master/compressor.php

DataProcessor Class:

The class DataProcessor is created to handle data processing and retrieval.

The constructor accepts the file path where the compressed data will be saved.

processData Method:

- ✓ Creates a large data structure.
- ✓ Serializes and compresses the data.
- ✓ Saves the compressed data to a file.
- ✓ Includes error handling with try-catch blocks.

retrieveData Method:

- ✓ Reads the compressed data from the file.
- ✓ Decompresses and unserializes the data.
- ✓ Prints a sample of the data to verify.
- ✓ Includes error handling with try-catch blocks.

Usage Example:

An instance of DataProcessor is created with the specified file path.

processData and retrieveData methods are called to demonstrate the complete process.

14. Write a PHP script that integrates with a REST API protected by OAuth 2.0 authentication. Implement the OAuth 2.0 authorization code flow to obtain an access token and use that token to make authenticated requests to the API. Provide a code example that demonstrates the complete authentication and data retrieval process.
https://github.com/JosephKithome/DPOpaytest/tree/master/oauth

15. Develop a PHP application that connects to an MS SQL Server database, retrieves data from multiple tables, performs a complex SQL query to join and aggregate data, and then returns the results as JSON. Demonstrate proper error handling and security measures in your code.

The whole implementation of the query can be found here;
https://github.com/JosephKithome/DPOpaytest/tree/master/advancedSQL