1. **What Is Object-Oriented Programming (OOP)**

**OOP** is a programming style that uses objects to represent real-world entities. These objects have properties (attributes) and behaviors (methods).

Oops is about creating class and object . Class serves as a template and multiple object can be created using a class.

1. **What Are Properties of Object-Oriented Systems?**

**Encapsulation**: Hiding internal details and showing only the necessary parts.

**Inheritance**: Creating new classes from existing ones.

**Polymorphism**: Objects can take on many forms.

**Abstraction**: Simplifying complex systems by modeling classes appropriate to the problem.

1. **What Is the Difference Between Class and Interface?**

**Class**: A blueprint for creating objects, containing properties and methods.

**Interface**: A contract that defines what methods a class should implement, but doesn't define how they work.

Classes are templates for creating object

if car is a class then maruti suzuki alto and maruti swift are two objects.

1. **What Is Overloading?**

**Overloading** is when multiple methods have the same name but different parameters within the same class. In PHP, method overloading is achieved through the \_\_call() magic method.

1. **What Is T\_PAAMAYIM\_NEKUDOTAYIM (Scope Resolution Operator (::))?**

It's a double colon ::, used to access static, constant, and overridden properties or methods of a class.

**Example**:

class Exm {

public static $value = 10;

public static function showValue() {

echo self::$value;

}

}

Exm::showValue();

1. **What Are the Differences Between Abstract Classes and Interfaces?**

**Abstract Classes**: Can have both defined and undefined (abstract) methods, and can contain properties.

**Interfaces**: Only declare methods; cannot have any defined methods or properties.

1. **Define Constructor and Destructor?**

**Constructor**: A special method called when an object is created, used to initialize the object.

**Destructor**: A special method called when an object is destroyed, used to clean up resources.

1. **How to Load Classes in PHP?**

**Using require, include, require\_once, or include\_once**: Load classes manually.

**Using spl\_autoload\_register()**: Automatically load classes when they are instantiated.

1. **How to Call Parent Constructor?**

Use parent::\_\_construct(); within the child class's constructor to call the parent's constructor.

1. **Are Parent Constructors Called Implicitly When Creating an Object of a Class?**

No, the parent constructor is not called automatically in PHP. You must explicitly call it using parent::\_\_construct(); in the child class constructor.

1. **What Happens If a Constructor Is Defined as Private or Protected?**

**Private Constructor**: Cannot be accessed or instantiated outside the class.

**Protected Constructor**: Can only be accessed within the class and its subclasses.

1. **What Are PHP Magic Methods/Functions? List Them.**

Magic methods in PHP are special methods with \_\_ prefix that are triggered automatically.

**Examples**: \_\_construct(), \_\_destruct(), \_\_call(), \_\_callStatic(), \_\_get(), \_\_set(), \_\_isset(), \_\_unset(), \_\_sleep(), \_\_wakeup(), \_\_toString(), \_\_invoke(), \_\_clone(), \_\_debugInfo().

1. **Write a Program for Static Keyword in PHP**

class StaticExample {

public static $count = 0;

public static function increment() {

self::$count++;

}

}

StaticExample::increment();

echo StaticExample::$count; // Outputs 1

1. **Create Multiple Traits and Use Them in a Single Class**

trait TraitOne {

public function methodOne() {

echo "Method from TraitOne";

}

}

trait TraitTwo {

public function methodTwo() {

echo "Method from TraitTwo";

}

}

class MyClass {

use TraitOne, TraitTwo;

}

$obj = new MyClass();

$obj->methodOne(); // Outputs "Method from TraitOne"

$obj->methodTwo(); // Outputs "Method from TraitTwo"

1. **Write PHP Script of Object Iteration**

class MyClass implements Iterator {

private $items = [];

private $position = 0;

public function \_\_construct($items) {

$this->items = $items;

}

public function current() {

return $this->items[$this->position];

}

public function key() {

return $this->position;

}

public function next() {

++$this->position;

}

public function rewind() {

$this->position = 0;

}

public function valid() {

return isset($this->items[$this->position]);

}

}

$obj = new MyClass([1, 2, 3, 4]);

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

echo "$key => $value\n";

}

1. **Use of the $this Keyword**

The $this keyword refers to the current object instance inside a class.

**Example**: class MyClass {

public $value = 10;

public function showValue() {

echo $this->value;

}

}

$obj = new MyClass();

$obj->showValue(); // Outputs 10