MODULE – Advance PHP

Q-1) What Is Object Oriented Programming?

Ans-1) PHP is a server-side scripting language, mainly used for web development but also used as a general-purpose programming language. Object-Oriented Programming (PHP OOP), is a type of programming language principle added to php5, that helps in building complex, reusable web applications.

Q-2) What Are Properties Of Object Oriented Systems?

Ans-2) In object-oriented programming (OOP), systems are designed using objects that interact with one another. PHP, being an object-oriented language, supports the following fundamental properties of OOP:

* Encapsulation
* Definition: Encapsulation is the mechanism of hiding the internal state of an object and requiring all interaction to be performed through an object's methods.
* Use private, protected, and public visibility keywords to control access to class members (properties and methods).
* Inheritance:
* Definition: Inheritance is a way to form new classes using classes that have already been defined. This promotes code reuse.
* Use the extends keyword to create a subclass.
* Polymorphism:
* Definition: Polymorphism allows objects of different classes to be treated as objects of a common superclass. It also allows a method to do different things based on the object it is acting upon.
* Achieved through method overriding and interfaces.
* Abstraction:
* Definition: Abstraction is the concept of hiding the complex implementation details and showing only the essential features of the object.
* Use abstract classes and methods.

Q-3) What Is Difference Between Class And Interface?

Ans-3) Key differences between class and interface:

* Constructors can be included in a class but not an inheritance.
* A method body can exist in a class, but it cannot exist in an interface.
* Classes do not support multiple inheritance, but it is supported by inheritance.
* To declare a class, you can use the keyword, class. However, in order to declare an interface, a keyword called interface is used.
* A class’s members do not have to be made public. It can be public, private, or protected. Members of an interface, on the other hand, are always public.

Q-4) What Is Overloading?

Ans-4) Overloading in PHP provides means to dynamically create properties and methods. These dynamic entities are processed via magic methods one can establish in a class for various action types.

The overloading methods are invoked when interacting with properties or methods that have not been declared or are not visible in the current scope. The rest of this section will use the terms inaccessible properties and inaccessible methods to refer to this combination of declaration and visibility. All overloading methods must be defined as public.

Q-5) What Is T\_PAAMAYIM\_NEKUDOTAYIM (Scope Resolution Operator (::) with Example.

Ans-5) The scope resolution operator also known as Paamayim Nekudotayim or more commonly known as the double colon is a token that allows access to static, constant, and overridden properties or methods of a class.

It is used to refer to blocks or codes in context to classes, objects, etc. An identifier is used with the scope resolution operator. The most common example of the application of the scope resolution operator in PHP is to access the properties and methods of the class.

Example: This type of definition is used while defining constants within a class.

<? php

class democlass {

const PI = 3.14;

}

echo democlass::PI;

?>

Output: 3.14

Q-6) What are the differences between abstract classes and interfaces?

Ans-6) Difference between Abstract class and Interfaces.

Abstract class:

* Abstract class comes under partial abstraction.
* Abstract classes can maintain abstract methods and non abstract methods.
* In abstract classes, we can create the variables.
* In abstract classes, we can use any access specifier.
* By using 'extends' keyword we can access the abstract class features from derived class.

Multiple inheritance is not possible.

Interface:

* Interface comes under fully abstraction.
* Interfaces can maintain only abstract methods.
* In interfaces, we can't create the variables.
* In interface, we can use only public access specifier.
* By using 'implement' keyword we can get interface from derived class.
* By using interfaces multiple inheritance is possible.

Q-7) Define Constructor and Destructor?

Ans-7) Constructors are the blueprints for object creation providing values for member functions and member variables. Once the object is initialized, the constructor is automatically called. Destructors are for destroying objects and automatically called at the end of execution.

Syntax:

* \_\_construct():

function \_\_construct()

{

// initialize the object and its properties by assigning

//values

}

* \_\_destruct():

function \_\_destruct()

{

// destroying the object or clean up resources here

}

* The constructor is defined in the public section of the Class. Even the values to properties of the class are set by Constructors.

Q-8) How to Load Classes in PHP?

Ans-8) PHP load classes are used for declaring its object etc. in object oriented applications. PHP parser loads it automatically, if it is registered with spl\_autoload\_register() function. PHP parser gets the least chance to load class/interface before emitting an error.

Syntax:

spl\_autoload\_register(function ($class\_name) {

include $class\_name . '.php';

});

Q-9) How to Call Parent Constructor?

Ans-9) We will face two cases while calling the parent constructor method in child class.

* Case1
* We can't run directly the parent class constructor in child class if the child class defines a constructor. In order to run a parent constructor, a call to parent::\_\_construct() within the child constructor is required.

Example:-

<?php

class grandpa{

public function \_\_construct(){

echo "I am in Tutorials Point".;

}

}

class papa extends grandpa{

public function \_\_construct(){

parent::\_\_construct();

echo "I am not in Tutorials Point";

}

}

$obj = new papa();

?>

Output: I am in Tutorials Point

I am not in Tutorials Point

Explanation:-

In the above example, we have used parent::\_\_construct() to call the parent class constructor.

* Case2
* If the child does not define a constructor then it may be inherited from the parent class just like a normal class method(if it was not declared as private).

Example

<? php

class grandpa{

public function \_\_construct(){

echo "I am in Tutorials point";

}

}

class papa extends grandpa{

}

$obj = new papa();

?>

Output: I am in Tutorials point

Explanation:-

Here the parent class is called implicitly because in child class we have not declared any constructor function in child class.

Q-10) Are Parent Constructor Called Implicitly When Create An ObjectOf Class?

Ans-10) Parent constructors are not called implicitly if the child class defines a constructor.

Q-11) What Happen, If Constructor Is Defined As Private Or Protected?

Ans-11) You cannot create an instance of class A from outside the class using that particular constructor which is declared private. Any member of a class declared as private can only be accessed from within the class only.

A constructor is defined private in certain design patterns like Singleton where the idea is to have a single instance created for a class during the entire life span of the program.

Q-12)What are PHP Magic Methods/Functions? List them Write program for Static Keyword in PHP?

Ans-12) Magic methods in PHP are special functions prefixed with \_\_ (double underscore) that empower developers to dynamically handle events and operations in classes. They modify how class properties, method calls, and other operations work, making classes more adaptable to specific needs.

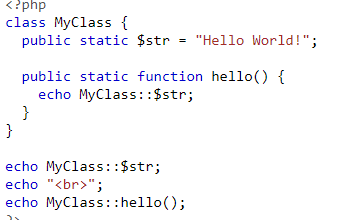
List of Magic Methods in PHP

Here is a list of commonly used magic methods in PHP:

* \_\_construct(): This magic method is the constructor of a class and is automatically called when an object is created. It is used to initialize object properties and perform any necessary setup.
* \_\_destruct(): The \_\_destruct() magic method is called automatically when an object is no longer referenced or when the script finishes execution. It is used to perform cleanup tasks or release resources held by the object.
* \_\_get(): This magic method is invoked when accessing inaccessible or non-existent properties of an object. It allows you to define custom logic for retrieving the value of a property.
* \_\_set(): The \_\_set() magic method is triggered when assigning a value to an inaccessible or non-existent property of an object. It allows you to define custom logic for setting the value of a property.
* \_\_isset(): This magic method is called when using the isset() function to check if an inaccessible or non-existent property of an object is set. It allows you to define custom logic for checking the existence of a property.
* \_\_unset(): The \_\_unset() magic method is invoked when using the unset() function to unset an inaccessible or non-existent property of an object. It allows you to define custom logic for removing a property.
* \_\_call(): This magic method is triggered when invoking inaccessible or non-existent methods of an object. It allows you to define custom logic for handling method calls.
* \_\_to String(): The \_\_to String() magic method is called when an object is treated as a string, such as when using the echo or print functions. It allows you to define a string representation of the object.

The static keyword is used to declare properties and methods of a class as static. Static properties and methods can be used without creating an instance of the class.

Example:-



Output: 

Q-13) Create multiple Traits and use it in to a single class?

Ans-13) In PHP, it is possible to use multiple traits in a single class using the use keyword followed by the names of the traits separated by commas. This allows a class to reuse code from multiple traits, providing more flexibility and reducing code duplication.

Here is an example of using multiple traits in a single class:

trait TraitA {

// trait A code here

}

trait TraitB {

// trait B code here

}

class MyClass {

use TraitA, TraitB;

// class code here

}

Explanation : In this example, the MyClass class uses both TraitA and TraitB. This allows the class to reuse code from both traits and avoid duplicating code. Run the above code in your editor for a better and clear explanation.

If there are methods with the same name in multiple traits, the class must provide an implementation of the method to avoid a fatal error. Here is an example of using multiple traits with conflicting method names:

<?php

trait TraitX {

public function conflictMethod() {

echo "TraitX conflictMethod\n";

}

}

trait TraitY {

public function conflictMethod() {

echo "TraitY conflictMethod\n";

}

}

class AnotherClass {

use TraitX, TraitY {

TraitX::conflictMethod insteadof TraitY;

TraitY::conflictMethod as anotherConflictMethod;

}

}

$anotherObject = new AnotherClass();

$anotherObject->conflictMethod(); // Output: TraitX conflictMethod

$anotherObject->anotherConflictMethod(); // Output: TraitY conflictMethod

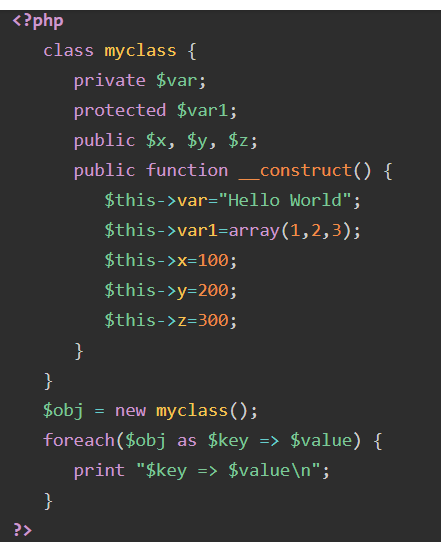
?>

Q-14) Write PHP Script of Object Iteration?

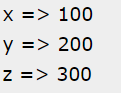
Ans-14) A foreach loop may be employed to iterate through all the publicly visible members of an object of a PHP class.

PHP also defines Iterator interface which can be used for the purpose.

Example:



Output:



Q-15) Use of The $this keyword

Ans-15) The $this keyword lets us use our class methods and properties. This means that we can, for example, call a method or a property within the current class itself.

Example:

class Dog {

public $name;

public $age;

public function intro(){

echo 'This is '.$this -> name . 'and it is ' .$this -> age . 'old.';

}

}