## MODULE – 4(Advance PHP)

**OOPS ::**

1. **What Is Object Oriented Programming ?**

**Ans:-**

OOP stands for Object-Oriented Programming.

Procedural programming is about writing procedures or functions that perform operations on the data, while object-oriented programming is about creating objects that contain both data and functions.

1. **What Are Properties Of Object Oriented Systems ?**

**Ans:-**

#### **Encapsulation.**

#### **Abstraction.**

#### **Inheritance.**

#### **Polymorphism.**

1. **What Is Difference Between Class And Interface ?**

**Ans:-**

|  |  |
| --- | --- |
| Class | Interface |
| * The keyword used to create a class is “class” | * The keyword used to create an interface is “interface” |
| * A class can be instantiated i.e, objects of a class can be created. | * An Interface cannot be instantiated i.e, objects cannot be created. |
| * Classes does not support multiple inheritance. | * Interface supports multiple inheritance. |
| * It can be inherit another class. | * It cannot inherit a class. |
| * It can be inherited by another class using the keyword ‘extends’. | * It can be inherited by a class by using the keyword ‘implements’ and it can be inherited by an interface using the keyword ‘extends’. |
| * It cannot contain abstract methods. | * It contains abstract methods only. |

1. **What Is Overloading?**

**Ans:-**

Function overloading in PHP is used to dynamically create properties and methods. These dynamic entities are processed by magic methods which can be used in a class for various action types. Function overloading contains same function name and that function performs different task according to number of arguments

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

**Ans:-**

The Scope Resolution Operator (also called Paamayim Nekudotayim) or in simpler terms, the double colon, is a token that allows access to static, constant, and overridden properties or methods of a class.

Example :-

<?php  
class MyClass {  
    const CONST\_VALUE = 'A constant value';  
}  
  
$classname = 'MyClass';  
echo $classname::CONST\_VALUE;  
  
echo MyClass::CONST\_VALUE;  
?>

1. **What are the differences between abstract classes and interfaces?**

**Ans:-**

|  |  |
| --- | --- |
| Interface Class | Abstract Class |
| Interface class supports multiple inheritance feature | Abstract class does not support multiple inheritances. |
| This does not contain a data member. | Abstract class does contain a data member. |
| The interface does not allow containers. | The abstract class supports containers. |
| An interface class only contains incomplete members which refer to the signature of the member. | Abstract class contains both incomplete(i.e. abstract) and complete members. |
| Since everything is assumed to be public, an interface class does not have access modifiers by default. | An abstract class can contain access modifiers within subs, functions, and properties. |
| Any member of an interface cannot be static. | Only a complete member of the abstract class can be static. |

1. **Define Constructor and Destructor ?**

**Ans:-**

**Constructor:-**

A constructor allows you to initialize an object's properties upon creation of the object.

If you create a \_\_construct() function, PHP will automatically call this function when you create an object from a class.

<?php  
class Fruit {  
  public $name;  
  public $color;  
  function \_\_construct($name) {  
    $this->name = $name;  
  }  
  function get\_name() {  
    return $this->name;  
  }  
}  
  
$apple = new Fruit("Apple");  
echo $apple->get\_name();  
?>

**Destructor:-**

A destructor is called when the object is destructed or the script is stopped or exited.

If you create a \_\_destruct() function, PHP will automatically call this function at the end of the script.

<?php  
 class Fruit {  
  public $name;  
  public $color;  
  
  function \_\_construct($name) {  
    $this->name = $name;  
  }  
  function \_\_destruct() {  
    echo "The fruit is {$this->name}.";  
  }  
}  
  
$apple = new Fruit("Apple");  
?>