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

As the name suggests, Object-Oriented Programming or OOPs refers to languages that use objects in programming. Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism, etc in programming. The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.

OOPs Concepts:

1. Class
2. Objects
3. Data Abstraction
4. Encapsulation
5. Inheritance
6. Polymorphism
7. Dynamic Binding
8. Message Passing
9. **What Are Properties Of Object Oriented Systems?**

An object oriented system revolves around a Class and objects. A class is used to describe characteristics of any entity of the real world. An object is a pattern of the class. An actual object created at runtime is called as an instance. A class, apart from characteristics has some functions to perform called as methods. For.e.g A class named “Food” has attributes like ‘price’, ‘quantity’. “Food” class has methods like Serve\_food(), bill\_food().  
  
Object: Objects in Object Oriented Systems interact through messages.  
  
Inheritance: The main class or the root class is called as a Base Class. Any class which is expected to have ALL properties of the base class along with its own is called as a Derived class. The process of deriving such a class is Derived class. For the “Food” class, a Derived class can be “Class Chinesefood”.  
  
Abstraction: Abstraction is creating models or classes of some broad concept. Abstraction can be achieved through Inheritance or even Composition.  
  
Encapsulation: Encapsulation is a collection of functions of a class and object. The “Food” class is an encapsulated form. It is achieved by specifying which class can use which members (private, public, protected) of an object.  
  
Polymorphism: Polymorphism means existing in different forms. Inheritance is an example of Polymorphism. A base class exists in different forms as derived classes. Operator overloading is an example of Polymorphism in which an operator can be applied in different situations.

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

|  |  |
| --- | --- |
| 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. **What Is Overloading in php ?**

Function overloading is the ability to create multiple functions of the same name with different implementations. Function overloading in PHP? 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. For example, find the area of certain shapes where radius are given then it should return area of circle if height and width are given then it should give area of rectangle and others. Like other OOP languages function overloading can not be done by native approach. In PHP function overloading is done with the help of magic function \_\_call(). This function takes function name and arguments.

**Property and Rules of overloading in PHP:**

1. All overloading methods must be defined as Public.
2. After creating the object for a class, we can access a set of entities that are properties or methods not defined within the scope of the class.
3. Such entities are said to be overloaded properties or methods, and the process is called as overloading.
4. For working with these overloaded properties or functions, PHP magic methods are used.
5. Most of the magic methods will be triggered in object context except \_\_callStatic() method which is used in a static context.
6. **What Is T\_PAAMAYIM\_NEKUDOTAYIM (Scope Resolution Operator (::) with Example**

T\_PAAMAYIM\_NEKUDOTAYIM is the scope resolution operator.    In most languages it is the double-colon (::).

Most people that are looking for this odd-looking word are probably running into an issue with a PHP program.   For as often as it comes up, there is very little information on what is causing the problem.

With PHP it typically means the application has a syntax error.    The PHP interpreter cannot figure out how to compile your code properly and crashes with the Unexpected T\_PAAMAYIM\_NEKUDOTAYIM error message.   Often it is due to a class name or static method reference that PHP cannot handle.  Either the class name is entered incorrectly OR your PHP version is so old it cannot process variable names before the double-colon operator.

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

|  |  |  |
| --- | --- | --- |
| S.No. | Abstract Class | Interface |
| 1. | An abstract class can contain both abstract and non-abstract methods. | Interface contains only abstract methods. |
| 2. | An abstract class can have all four; static, non-static and final, non-final variables. | Only final and static variables are used. |
| 3. | To declare abstract class abstract keywords are used. | The interface can be declared with the interface keyword. |
| 4. | It supports multiple inheritance. | It does not support multiple inheritance. |
| 5. | The keyword ‘extend’ is used to extend an abstract class | The keyword implement is used to implement the interface. |
| 6. | It has class members like private and protected, etc. | It has class members public by default. |

1. **Define Constructor and Destructor?**

Constructors start with two underscores and generally look like normal PHP functions. Sometimes these constructors are called as magic functions starting with two underscores and with some extra functionality than normal methods. After creating an object of some class that includes constructor, the content of constructor will be automatically executed.  
Note: If the PHP Class has a constructor, then at the time of object creation, the constructor of the class is called. The constructors have no Return Type, so they do not return anything not even void.  
Advantages of using Constructors: 

Constructors provides the ability to pass parameters which are helpful in automatic initialization of the member variables during creation time .

The Constructors can have as many parameters as required and they can be defined with the default arguments.

They encourage re-usability avoiding re-initializing whenever instance of the class is created .

You can start session in constructor method so that you don’t have to start in all the functions everytime.

They can call class member methods and functions.

They can call other Constructors even from Parent class.

Destructor: Destructor is also a special member function which is exactly the reverse of constructor method and is called when an instance of the class is deleted from the memory. Destructors (\_\_destruct ( void): void) are methods which are called when there is no reference to any object of the class or goes out of scope or about to release explicitly.   
They don’t have any types or return value. It is just called before de-allocating memory for an object or during the finish of execution of PHP scripts or as soon as the execution control leaves the block.   
Global objects are destroyed when the full script or code terminates. Cleaning up of resources before memory release or closing of files takes place in the destructor method, whenever they are no longer needed in the code. The automatic destruction of class objects is handled by PHP Garbage Collector.

~ ClassName()

{

}

**Comparison between \_\_constructors and \_\_destructors:** 

| **Constructors** | **Destructors** |
| --- | --- |
| Accepts one or more arguments. | No arguments are passed. Its void. |
| function name is \_construct(). | function name is \_destruct() |
| It has same name as the class. | It has same name as the class with prefix ~tilda. |
| Constructor is involved automatically when the object is created. | Destructor is involved automatically when the object is destroyed. |
| Used to initialize the instance of a class. | Used to de-initialize objects already existing to free up memory for new accommodation. |
| Used to initialize data members of class. | Used to make the object perform some task before it is destroyed. |
| Constructors can be overloaded. | Destructors cannot be overloaded. |
| It is called each time a class is instantiated or object is created. | It is called automatically at the time of object deletion . |
| Allocates memory. | It deallocates memory. |
| Multiple constructors can exist in a class. | Only one Destructor can exist in a class. |

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

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';

});

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

using parent::

/\* Settings \*/

class Settings{

function \_\_CONSTRUCT(){

echo "Settings Construct";

}

}

/\* PageManager \*/

class PageManager extends Settings{

function \_\_CONSTRUCT(){

parent::\_\_CONSTRUCT();

echo "PageManager Construct";

}

}

1. **Is Parent Constructor Called Implicitly When Create An ObjectOf Class?**

[No, parent constructors are not called implicitly when creating an object of a class](https://www.bing.com/ck/a?!&&p=5fda5af975af3c42JmltdHM9MTY5NTc3MjgwMCZpZ3VpZD0yMjM1MzA1ZC1iYmQ3LTY0NTEtMDQ3Mi0yMzMxYmE0YzY1ZTUmaW5zaWQ9NTcxNw&ptn=3&hsh=3&fclid=2235305d-bbd7-6451-0472-2331ba4c65e5&psq=Are+Parent+Constructor+Called+Implicitly+When+Create+An+ObjectOf+Class%3f&u=a1aHR0cHM6Ly93d3cud2lzZG9tam9icy5jb20vZS11bml2ZXJzaXR5L29iamVjdC1vcmllbnRlZC1wcm9ncmFtbWluZy1pbi1waHAtaW50ZXJ2aWV3LXF1ZXN0aW9ucy5odG1s&ntb=1)[1](https://www.bing.com/ck/a?!&&p=ebf56d62cf845909JmltdHM9MTY5NTc3MjgwMCZpZ3VpZD0yMjM1MzA1ZC1iYmQ3LTY0NTEtMDQ3Mi0yMzMxYmE0YzY1ZTUmaW5zaWQ9NTcxOA&ptn=3&hsh=3&fclid=2235305d-bbd7-6451-0472-2331ba4c65e5&psq=Are+Parent+Constructor+Called+Implicitly+When+Create+An+ObjectOf+Class%3f&u=a1aHR0cHM6Ly93d3cud2lzZG9tam9icy5jb20vZS11bml2ZXJzaXR5L29iamVjdC1vcmllbnRlZC1wcm9ncmFtbWluZy1pbi1waHAtaW50ZXJ2aWV3LXF1ZXN0aW9ucy5odG1s&ntb=1)[2](https://www.bing.com/ck/a?!&&p=c9d6abd0f89b8740JmltdHM9MTY5NTc3MjgwMCZpZ3VpZD0yMjM1MzA1ZC1iYmQ3LTY0NTEtMDQ3Mi0yMzMxYmE0YzY1ZTUmaW5zaWQ9NTcxOQ&ptn=3&hsh=3&fclid=2235305d-bbd7-6451-0472-2331ba4c65e5&psq=Are+Parent+Constructor+Called+Implicitly+When+Create+An+ObjectOf+Class%3f&u=a1aHR0cHM6Ly9zdGFja292ZXJmbG93LmNvbS9xdWVzdGlvbnMvMTMxMTQ2MDIvYXJlLXBhcmVudC1jb25zdHJ1Y3RvcnMtY2FsbGVkLWlmLWEtY2hpbGQtY2xhc3MtZG9lcy1ub3QtZGVmaW5lLWEtY29uc3RydWN0b3I&ntb=1)[3](https://www.bing.com/ck/a?!&&p=cba4c64c1f117db6JmltdHM9MTY5NTc3MjgwMCZpZ3VpZD0yMjM1MzA1ZC1iYmQ3LTY0NTEtMDQ3Mi0yMzMxYmE0YzY1ZTUmaW5zaWQ9NTcyMA&ptn=3&hsh=3&fclid=2235305d-bbd7-6451-0472-2331ba4c65e5&psq=Are+Parent+Constructor+Called+Implicitly+When+Create+An+ObjectOf+Class%3f&u=a1aHR0cHM6Ly93d3cucGhwLm5ldC9tYW51YWwvZW4vbGFuZ3VhZ2Uub29wNS5kZWNvbi5waHA&ntb=1)[4](https://www.bing.com/ck/a?!&&p=eb2069e37b264214JmltdHM9MTY5NTc3MjgwMCZpZ3VpZD0yMjM1MzA1ZC1iYmQ3LTY0NTEtMDQ3Mi0yMzMxYmE0YzY1ZTUmaW5zaWQ9NTcyMQ&ptn=3&hsh=3&fclid=2235305d-bbd7-6451-0472-2331ba4c65e5&psq=Are+Parent+Constructor+Called+Implicitly+When+Create+An+ObjectOf+Class%3f&u=a1aHR0cHM6Ly93d3cuZGV2YXNraW5nLmNvbS9pc3N1ZS9hcmUtYWJzdHJhY3QtY2xhc3MtY29uc3RydWN0b3JzLW5vdC1pbXBsaWNpdGx5LWNhbGxlZC13aGVuLWEtZGVyaXZlZC1jbGFzcy1pcy1pbnN0YW50aWF0ZWQ&ntb=1). [If the child class defines a constructor, the parent constructor is not called implicitly](https://www.bing.com/ck/a?!&&p=2600b70076afad15JmltdHM9MTY5NTc3MjgwMCZpZ3VpZD0yMjM1MzA1ZC1iYmQ3LTY0NTEtMDQ3Mi0yMzMxYmE0YzY1ZTUmaW5zaWQ9NTcyMg&ptn=3&hsh=3&fclid=2235305d-bbd7-6451-0472-2331ba4c65e5&psq=Are+Parent+Constructor+Called+Implicitly+When+Create+An+ObjectOf+Class%3f&u=a1aHR0cHM6Ly93d3cud2lzZG9tam9icy5jb20vZS11bml2ZXJzaXR5L29iamVjdC1vcmllbnRlZC1wcm9ncmFtbWluZy1pbi1waHAtaW50ZXJ2aWV3LXF1ZXN0aW9ucy5odG1s&ntb=1)[1](https://www.bing.com/ck/a?!&&p=e5603ffb6415dce2JmltdHM9MTY5NTc3MjgwMCZpZ3VpZD0yMjM1MzA1ZC1iYmQ3LTY0NTEtMDQ3Mi0yMzMxYmE0YzY1ZTUmaW5zaWQ9NTcyMw&ptn=3&hsh=3&fclid=2235305d-bbd7-6451-0472-2331ba4c65e5&psq=Are+Parent+Constructor+Called+Implicitly+When+Create+An+ObjectOf+Class%3f&u=a1aHR0cHM6Ly93d3cud2lzZG9tam9icy5jb20vZS11bml2ZXJzaXR5L29iamVjdC1vcmllbnRlZC1wcm9ncmFtbWluZy1pbi1waHAtaW50ZXJ2aWV3LXF1ZXN0aW9ucy5odG1s&ntb=1)[3](https://www.bing.com/ck/a?!&&p=bdff8531d89968ceJmltdHM9MTY5NTc3MjgwMCZpZ3VpZD0yMjM1MzA1ZC1iYmQ3LTY0NTEtMDQ3Mi0yMzMxYmE0YzY1ZTUmaW5zaWQ9NTcyNA&ptn=3&hsh=3&fclid=2235305d-bbd7-6451-0472-2331ba4c65e5&psq=Are+Parent+Constructor+Called+Implicitly+When+Create+An+ObjectOf+Class%3f&u=a1aHR0cHM6Ly93d3cucGhwLm5ldC9tYW51YWwvZW4vbGFuZ3VhZ2Uub29wNS5kZWNvbi5waHA&ntb=1)[4](https://www.bing.com/ck/a?!&&p=1817c9221bd6ecd7JmltdHM9MTY5NTc3MjgwMCZpZ3VpZD0yMjM1MzA1ZC1iYmQ3LTY0NTEtMDQ3Mi0yMzMxYmE0YzY1ZTUmaW5zaWQ9NTcyNQ&ptn=3&hsh=3&fclid=2235305d-bbd7-6451-0472-2331ba4c65e5&psq=Are+Parent+Constructor+Called+Implicitly+When+Create+An+ObjectOf+Class%3f&u=a1aHR0cHM6Ly93d3cuZGV2YXNraW5nLmNvbS9pc3N1ZS9hcmUtYWJzdHJhY3QtY2xhc3MtY29uc3RydWN0b3JzLW5vdC1pbXBsaWNpdGx5LWNhbGxlZC13aGVuLWEtZGVyaXZlZC1jbGFzcy1pcy1pbnN0YW50aWF0ZWQ&ntb=1). [If the child class does not define a constructor, the parent constructor is called implicitly](https://www.bing.com/ck/a?!&&p=7dc064bd398da32aJmltdHM9MTY5NTc3MjgwMCZpZ3VpZD0yMjM1MzA1ZC1iYmQ3LTY0NTEtMDQ3Mi0yMzMxYmE0YzY1ZTUmaW5zaWQ9NTcyNg&ptn=3&hsh=3&fclid=2235305d-bbd7-6451-0472-2331ba4c65e5&psq=Are+Parent+Constructor+Called+Implicitly+When+Create+An+ObjectOf+Class%3f&u=a1aHR0cHM6Ly93d3cud2lzZG9tam9icy5jb20vZS11bml2ZXJzaXR5L29iamVjdC1vcmllbnRlZC1wcm9ncmFtbWluZy1pbi1waHAtaW50ZXJ2aWV3LXF1ZXN0aW9ucy5odG1s&ntb=1)[1](https://www.bing.com/ck/a?!&&p=20f02fe432c94411JmltdHM9MTY5NTc3MjgwMCZpZ3VpZD0yMjM1MzA1ZC1iYmQ3LTY0NTEtMDQ3Mi0yMzMxYmE0YzY1ZTUmaW5zaWQ9NTcyNw&ptn=3&hsh=3&fclid=2235305d-bbd7-6451-0472-2331ba4c65e5&psq=Are+Parent+Constructor+Called+Implicitly+When+Create+An+ObjectOf+Class%3f&u=a1aHR0cHM6Ly93d3cud2lzZG9tam9icy5jb20vZS11bml2ZXJzaXR5L29iamVjdC1vcmllbnRlZC1wcm9ncmFtbWluZy1pbi1waHAtaW50ZXJ2aWV3LXF1ZXN0aW9ucy5odG1s&ntb=1). [If a child overrides the constructor, this will be used when creating new objects and parent's constructor is not called implicitly](https://www.bing.com/ck/a?!&&p=93af806b68fde192JmltdHM9MTY5NTc3MjgwMCZpZ3VpZD0yMjM1MzA1ZC1iYmQ3LTY0NTEtMDQ3Mi0yMzMxYmE0YzY1ZTUmaW5zaWQ9NTcyOA&ptn=3&hsh=3&fclid=2235305d-bbd7-6451-0472-2331ba4c65e5&psq=Are+Parent+Constructor+Called+Implicitly+When+Create+An+ObjectOf+Class%3f&u=a1aHR0cHM6Ly9zdGFja292ZXJmbG93LmNvbS9xdWVzdGlvbnMvMTMxMTQ2MDIvYXJlLXBhcmVudC1jb25zdHJ1Y3RvcnMtY2FsbGVkLWlmLWEtY2hpbGQtY2xhc3MtZG9lcy1ub3QtZGVmaW5lLWEtY29uc3RydWN0b3I&ntb=1)[2](https://www.bing.com/ck/a?!&&p=394fcf42ee6dc43aJmltdHM9MTY5NTc3MjgwMCZpZ3VpZD0yMjM1MzA1ZC1iYmQ3LTY0NTEtMDQ3Mi0yMzMxYmE0YzY1ZTUmaW5zaWQ9NTcyOQ&ptn=3&hsh=3&fclid=2235305d-bbd7-6451-0472-2331ba4c65e5&psq=Are+Parent+Constructor+Called+Implicitly+When+Create+An+ObjectOf+Class%3f&u=a1aHR0cHM6Ly9zdGFja292ZXJmbG93LmNvbS9xdWVzdGlvbnMvMTMxMTQ2MDIvYXJlLXBhcmVudC1jb25zdHJ1Y3RvcnMtY2FsbGVkLWlmLWEtY2hpbGQtY2xhc3MtZG9lcy1ub3QtZGVmaW5lLWEtY29uc3RydWN0b3I&ntb=1).

1. **What Happen, If Constructor Is Defined As Private Or Protected?**

The constructor may be made private or protected to prevent it from being called externally. If so, only a static method will be able to instantiate the class. Because they are in the same class definition they have access to private methods, even if not of the same object instance. The private constructor is optional and may or may not make sense depending on the use case.

The three public static methods then demonstrate different ways of instantiating the object.

* fromBasicData() takes the exact parameters that are needed, then creates the object by calling the constructor and returning the result.
* fromJson() accepts a JSON string and does some pre-processing on it itself to convert it into the format desired by the constructor. It then returns the new object.
* fromXml() accepts an XML string, preprocesses it, and then creates a bare object. The constructor is still called, but as all of the parameters are optional the method skips them. It then assigns values to the object properties directly before returning the result.
* In all three cases, the static keyword is translated into the name of the class the code is in.

1. **What are PHP Magic Methods/Functions? List them Write program for Static Keyword in PHP?**

In [PHP](https://www.edureka.co/blog/php-tutorial-for-beginners/), special functions can be defined in such a way that they can be called automatically and does not require any function call to execute the code inside these functions. This feature is available in a special method known as magic methods. In this article, we will discuss the top Magic Methods in PHP.

[**What are Magic Methods?**](https://www.edureka.co/blog/magic-methods-in-php#what)

[List of Magic Methods in PHP](https://www.edureka.co/blog/magic-methods-in-php#list)

**What are Magic Methods in PHP?**

Methods that begin with 2 underscores(\_\_) are generally called Magic methods in PHP. These methods names are limited to some list of [PHP](https://www.php.net/manual/en/intro-whatis.php) supported keywords that are reserved. So any function should not be defined with the name of PHP magic methods.

Usually, these functions should be defined by the user and there is no need to call them explicitly.

**List of Magic Methods in PHP**

* \_\_construct()
* \_\_destruct()
* \_\_call($fun, $arg)
* \_\_callStatic($fun, $arg)
* \_\_get($property)
* \_\_set($property, $value)
* \_\_isset($content)
* \_\_unset($content)
* \_\_sleep()
* \_\_wakeup()
* \_\_toString()
* \_\_invoke()
* \_\_set\_state($array)
* \_\_clone()
* \_\_debugInfo(

1. **Create multiple Traits and use it in to a single class?**

traits are a mechanism for code reuse in single inheritance languages such as PHP. A Trait is intended to reduce some limitations of single inheritance by enabling a developer to reuse sets of methods freely in several independent classes living in different class hierarchies. The semantics of the combination of Traits and classes is defined in a way which reduces complexity, and avoids the typical problems associated with multiple inheritance and Mixins.

A Trait is similar to a class, but only intended to group functionality in a fine-grained and consistent way. It is not possible to instantiate a Trait on its own. It is an addition to traditional inheritance and enables horizontal composition of behavior; that is, the application of class members without requiring inheritance.

Example #1 Trait example

<?php  
trait ezcReflectionReturnInfo {  
function getReturnType() { /\*1\*/ }  
function getReturnDescription() { /\*2\*/ }  
}  
  
class ezcReflectionMethod extends ReflectionMethod {  
use ezcReflectionReturnInfo;  
/\* ... \*/  
}  
  
class ezcReflectionFunction extends ReflectionFunction {  
use ezcReflectionReturnInfo;  
/\* ... \*/  
}  
?>

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

In PHP, object iteration refers to the process of iterating over the properties of an object using a loop. This can be achieved using the “foreach” loop, which allows you to loop through the properties of an object and perform operations on each one.

To iterate over an object, you need to define a variable to represent each property of the object, and then use the foreach loop to loop through the object properties. The syntax for this is as follows:

foreach ($object as $property => $value) { // Perform operations on $value }

In this code, “$object” represents the object you want to iterate over, “$property” represents the name of each property in the object, and “$value” represents the value of each property. You can perform operations on the “$value” variable inside the loop to manipulate the object properties as needed.

It’s important to note that not all properties of an object can be iterated over. Private and protected properties cannot be accessed outside of the object’s class definition, so they will not be included in the iteration. Additionally, some built-in PHP classes may not be iterable.

1. **What is the Use of $this keyword**

**$this** is a reserved keyword in PHP that refers to the calling object. It is usually the object to which the method belongs, but possibly another object if the method is called statically from the context of a secondary object. This keyword is only applicable to internal methods.

**$this – a pseudo-variable:** Unlike other reserved keywords used in the context of class like the static, parent, etc does not need to be declared with the dollar sign (‘$’). This is because in PHP *$this* is treated as a pseudo-variable.  
In PHP, this is declared like a variable declaration (with the ‘$’ sign) even though it is a reserved keyword. More specifically, *$this* is a special read-only variable that is not declared anywhere in the code and which represents a value that changes depending on the context of program execution.