**1.What is the php static method ? with an example .**

In PHP, a static method is a method that belongs to the class itself, rather than an object (instance) of the class. You can call a static method without creating an object of the class. It uses the static keyword.

✅ Syntax

class ClassName {

public static function methodName() {

// code

}

}

To call it:

ClassName::methodName();

**2.What is  Inheritance?**

Inheritance is an Object-Oriented Programming (OOP) concept where a child class (also called a subclass) can inherit properties and methods from a parent class (also called a superclass).

This allows code reuse and makes programs easier to manage and extend.

✅ Syntax

class ParentClass {

// properties and methods

}

class ChildClass extends ParentClass {

// inherits everything from ParentClass

// can add or override methods

}

**3.What is the use of session\_start() and session\_destroy() functions in PHP?**

PHP sessions are used to store data across multiple pages (like user login status).

The two main functions involved in session handling are:

🔹 **session\_start()**

Purpose:

Starts a new session or resumes an existing one.

Must be called before any output (HTML or echo) is sent to the browser.

<?php

session\_start();

$\_SESSION['username'] = "JohnDoe";

?>

🔹 **session\_destroy()**

Purpose:

Ends the current session.

Destroys all session data on the server.

<?php

session\_start(); // Required before destroying

session\_destroy();

?>

✅ What it does:

Deletes the session file from the server.

But note: $\_SESSION variables will still be accessible until the page ends unless they are manually unset.

**4.What is the difference between the include() and require() functions?**

| **Feature/Aspect** | | **include()** | | **require()** |
| --- | --- | --- | --- | --- |
| **Definition** | | Includes and evaluates a specified file. | | Same as include(), but with stricter error handling. |
| **Error on Missing File** | | Issues a **warning** (E\_WARNING) if the file is not found. | | Issues a **fatal error** (E\_COMPILE\_ERROR) if the file is not found. |
| **Script Execution** | | Continues executing the script even if the file is missing. | | Stops script execution immediately if the file is missing. |
| **Use Case** | | Best for **optional** files (e.g., header, sidebar, banner). | | Best for **essential** files (e.g., database connection, authentication). |
| **Return Value** | | Returns 1 if successful, false on failure (file not found). | | Same behavior. |
| **Performance** | Same as require(); both are interpreted at runtime. | | Same as include(); no significant performance difference. | | |
| **Error Type** | | Non-fatal error → script continues. | | Fatal error → script terminates. |
| **Inside Conditional Block** | | Can be used inside if, loop, function, etc. | | Same, but use with care if file is critical. |
| **Code Execution Flow** | | Useful when the included file may not always be necessary. | | Ensures that required files are always included for the code to run. |
| **Best Practice** | | Use when **missing file should not break** the application. | | Use when **missing file should break** the application for safety. |

**5. Explain the differences between abstract classes and interfaces.**

| **Abstract Class** | **Interface** |
| --- | --- |
| A class that **cannot be instantiated** and may contain abstract and concrete methods. | A completely abstract type that defines only method signatures (no implementation). |
| Can have properties | Can not have properties |
| Can have **both** abstract (no body) and **concrete** (with body) methods. | All methods are **public** and **abstract by default**. No body allowed. |
| Can have properties with any visibility (public, protected, private). | Can only have public constants (no regular properties). |
| Can have a constructor. | Cannot define a constructor. |
| Can use all visibility levels (public, protected, private). | All methods are implicitly public. |
| Used when classes share **some common behavior** and require **code reuse**. | Used to define a **contract** or **capability** that other classes must implement. |

**6.Differentiate between GET and POST methods in PHP.**

| **Feature** | **GET Method** | **POST Method** |
| --- | --- | --- |
| **Data Visibility** | Data is visible in the **URL** (e.g., ?name=John) | Data is **not visible** in the URL (sent in HTTP body) |
| **Security** | Less secure (data can be bookmarked, cached, sniffed) | More secure for sensitive data like passwords, forms |
| **Use Case** | When retrieving or **displaying data** (search, filter) | When **sending or updating data** (login, form submission) |
| **Caching** | Can be cached by browser | Not cached by default |
| **Bookmarkable** | Yes, because data is in URL | No, data is not stored in URL |

**7. What are magic methods in PHP?**

**Magic methods** in PHP are **special methods** that start with **double underscores (\_\_)**.  
They are **automatically triggered** by specific actions/events — like creating an object, accessing properties, or calling inaccessible methods.

They help implement powerful object-oriented features like **custom property access**, **object-to-string conversion**, **serialization**, etc.

**✅ Common Magic Methods in PHP**

| **Magic Method** | **Triggered When...** |
| --- | --- |
| \_\_construct() | An object is created (constructor) |
| \_\_destruct() | An object is destroyed (destructor) |
| \_\_get($name) | Accessing an **undefined or inaccessible property** |
| \_\_set($name, $value) | Setting an **undefined or inaccessible property** |
| \_\_isset($name) | Calling isset() or empty() on inaccessible properties |
| \_\_unset($name) | Calling unset() on inaccessible properties |
| \_\_call($name, $arguments) | Calling an **undefined or inaccessible method** |
| \_\_callStatic($name, $arguments) | Calling an undefined **static method** |
| \_\_toString() | Object is treated as a **string** (e.g., echo $object) |
| \_\_invoke() | Object is called as a **function** (e.g., $obj()) |
| \_\_clone() | Object is cloned with clone keyword |
| \_\_debugInfo() | Called when var\_dump() is used on an object |

**8.What is the difference between “echo” and “print” in PHP?**

| **Feature/Aspect** | **echo** | **print** |
| --- | --- | --- |
| **Return Value** | **No return value** | Returns 1 (can be used in expressions) |
| **Usage in Expressions** | ❌ Cannot be used in expressions | ✅ Can be used (returns a value) |
| **Multiple Parameters** | ✅ Yes (e.g., echo "a", "b";) | ❌ No — only one argument allowed |
| **Common Use** | Preferred when **just displaying** output | Preferred when used **in expressions** |

9. How many types of an array are there in PHP?

| **Type** | **Description** | **Example** |
| --- | --- | --- |
| **Indexed Array** | Arrays with **numeric indexes**, starting from 0 by default. | php $arr = ["apple", "banana", "cherry"]; |
| **Associative Array** | Arrays with **named keys** (strings) instead of numeric indexes. | php $arr = ["name" => "John", "age" => 30]; |
| **Multidimensional Array** | Arrays containing **one or more arrays** as elements (arrays inside arrays). | php $arr = [ ["a", "b"], ["c", "d"] ]; |

**10.** What is the meaning of a final class and a final method?

**A final class is a class that cannot be extended or inherited. Any methods, instance variables, and constants inside the course cannot be modified or used in another category.**

**Final Class**

* A class declared with the final keyword **cannot be extended** (no subclass can inherit from it).
* It **prevents inheritance**, ensuring the class's implementation stays unchanged.

**Final Method**

* A method declared as final **cannot be overridden** by any child class.
* The class itself can be extended, but the specific method stays exactly as defined.

11. List the main types of errors in PHP and explain their differences.

**There are three main error types in PHP:**

* **Notices. These are non-critical errors that can occur during the script execution. These are not visible to users. Accessing an undefined variable is an example of a 'Notice'.**
* **Warnings. These are more critical than Notices, but just like them, Warnings don't interrupt the script execution. However, these are visible to the user by default. Example: include() a file that doesn't exist.**
* **Fatal. This is the most critical error type which, when occurs, immediately terminates the script execution. Accessing a property of a non-existent object or require() a non-existent file is an example of this error type**

**1.Differentiate between variables and constants in PHP**

Few difference between variables and constants in PHP are given below:

| **Variables** | **Constants** |
| --- | --- |
| The value of a variable can be changed during the execution. | The constant value can’t be changed during script execution. |
| Variables require compulsory usage of the $ sign at the start. | No dollar sign ($) is required before using a constant. |
| It is possible to define a variable by simple assignment. | Constants can’t be defined by simple assignments. They are defined using the define() function. |
| The default scope is the current access scope. | Constants can be accessed throughout without any scoping rules. |

**2. What is a session in PHP?**

A session in PHP is a way to store information to be used across multiple pages of an entire website. The information is not stored on the user’s computer, unlike cookies. In a temporary directory on the server, a file will be created by the session where registered session variables and their values are stored. This information will be available to all pages on the site during that visit.

When you work with an application, you open it, do some modifications, and then you close it. This is much like a Session. The computer knows who you are. It knows when the application is started and ended by you.

But on the internet, the webserver does not know who you are or what you do, because the HTTP address doesn’t maintain a state. This problem is solved using session variables by storing user information to be used across multiple pages (e.g. username, favorite color, etc).

By default, session variables will last until the user closes the browser.

So Session variables hold single user information and are available to all pages in one application.

**4. Explain the difference between $message and $$message.**

The main difference between the $message and $$message is given below:

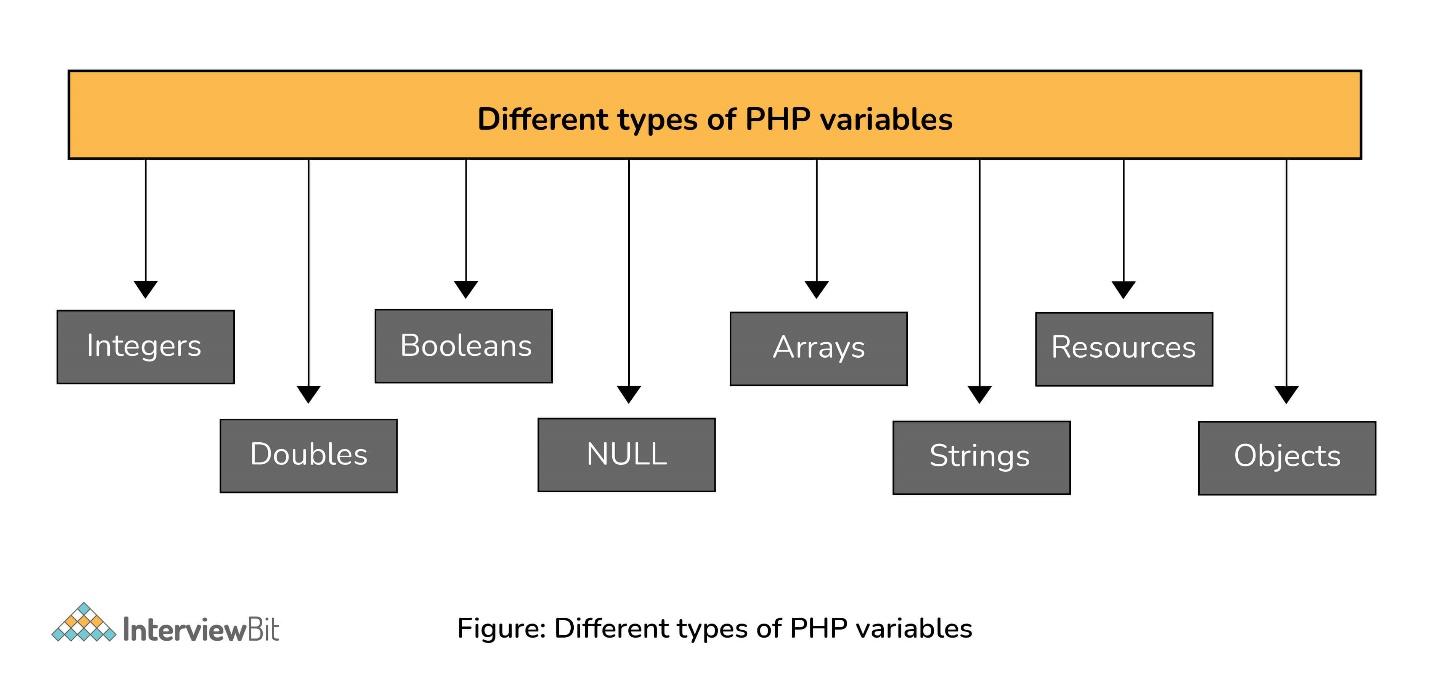
| **$message** | **$$message** |
| --- | --- |
| $message is a regular variable. | $$message is a reference variable. |
| It has a fixed name and stores a fixed value. | It stores data about the variable. |
| Data stored in $message is fixed. | The value of the $$message can change dynamically as the value of the variable changes. |

**5. Is PHP a case-sensitive language?**

PHP can be considered as a partial case-sensitive language. The variable names are completely case-sensitive but function names are not. Also, user-defined functions are not case-sensitive but the rest of the language is case-sensitive.

For example, user-defined functions in PHP can be defined in lowercase but later referred to in uppercase, and it would still function normally.

**6. What are the different types of variables present in PHP?**

Types of PHP Variables

There are 8 primary data types in PHP which are used to construct the variables. They are:

* Integers: Integers are whole numbers without a floating-point. Ex: 1253.
* Doubles: Doubles are floating-point numbers. Ex: 7.876
* Booleans: It represents two logical states- true or false.
* NULL: NULL is a special type that only has one value, NULL. When no value is assigned to a variable, it can be assigned with NULL.
* Arrays: Array is a named and ordered collection of similar type of data. Ex: $colors = array("red", "yellow", "blue");
* Strings: Strings are a sequence of characters. Ex: “Hello InterviewBit!”
* Resources: Resources are special variables that consist of references to resources external to PHP(such as database connections).
* Objects: An instance of classes containing data and functions. Ex: $mango = new Fruit();

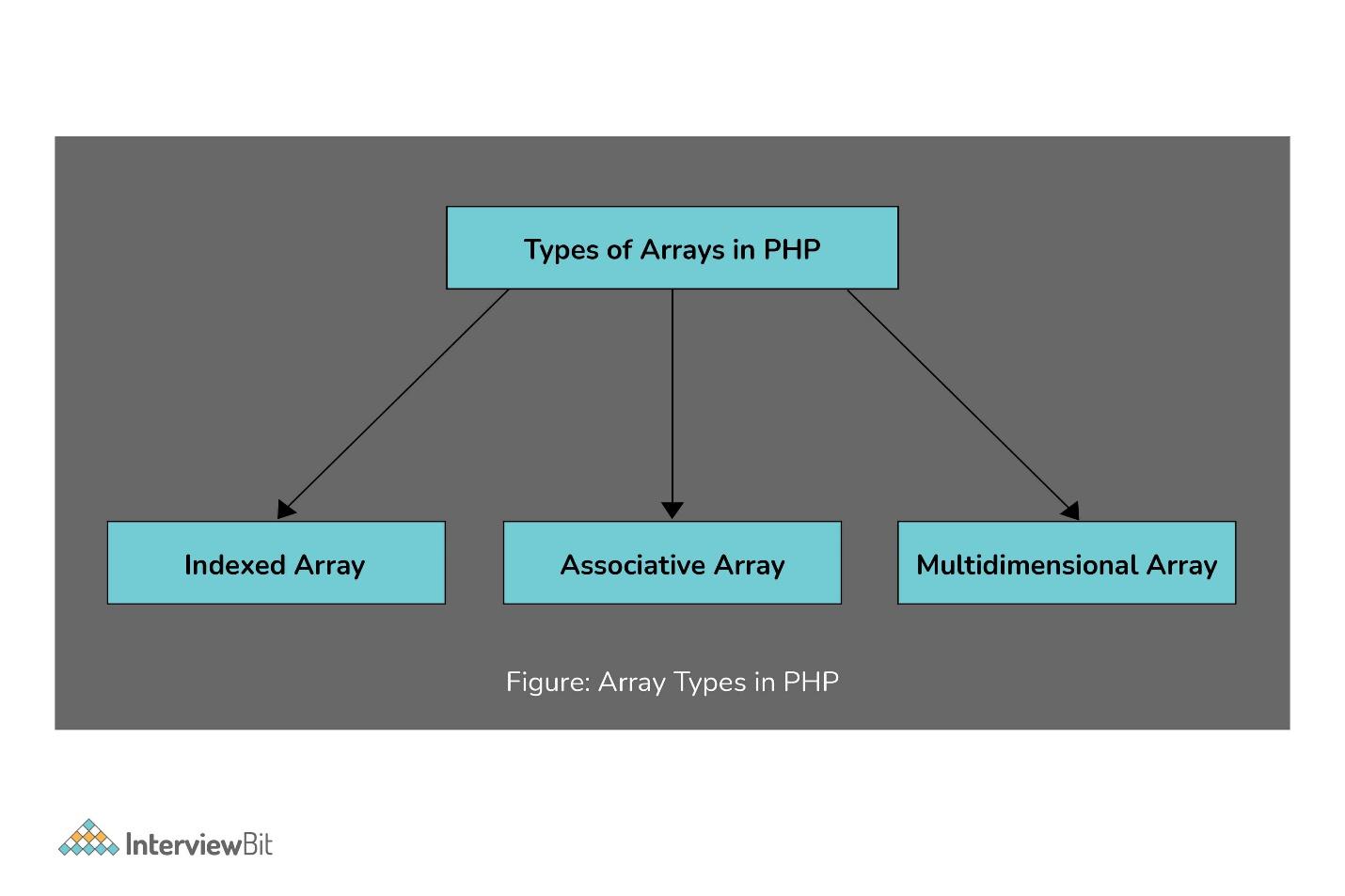
**7. What are the rules for naming a PHP variable?**

The following two rules are needed to be followed while naming a PHP variable:

* A variable must start with a dollar symbol, followed by the variable name. For example: $price=100; where price is a variable name.
* Variable names must begin with a letter or underscore.
* A variable name can consist of letters, numbers, or underscores. But you cannot use characters like + , – , % , & etc.
* A PHP variable name cannot contain spaces.
* PHP variables are case-sensitive. So $NAME and $name both are treated as different variables.

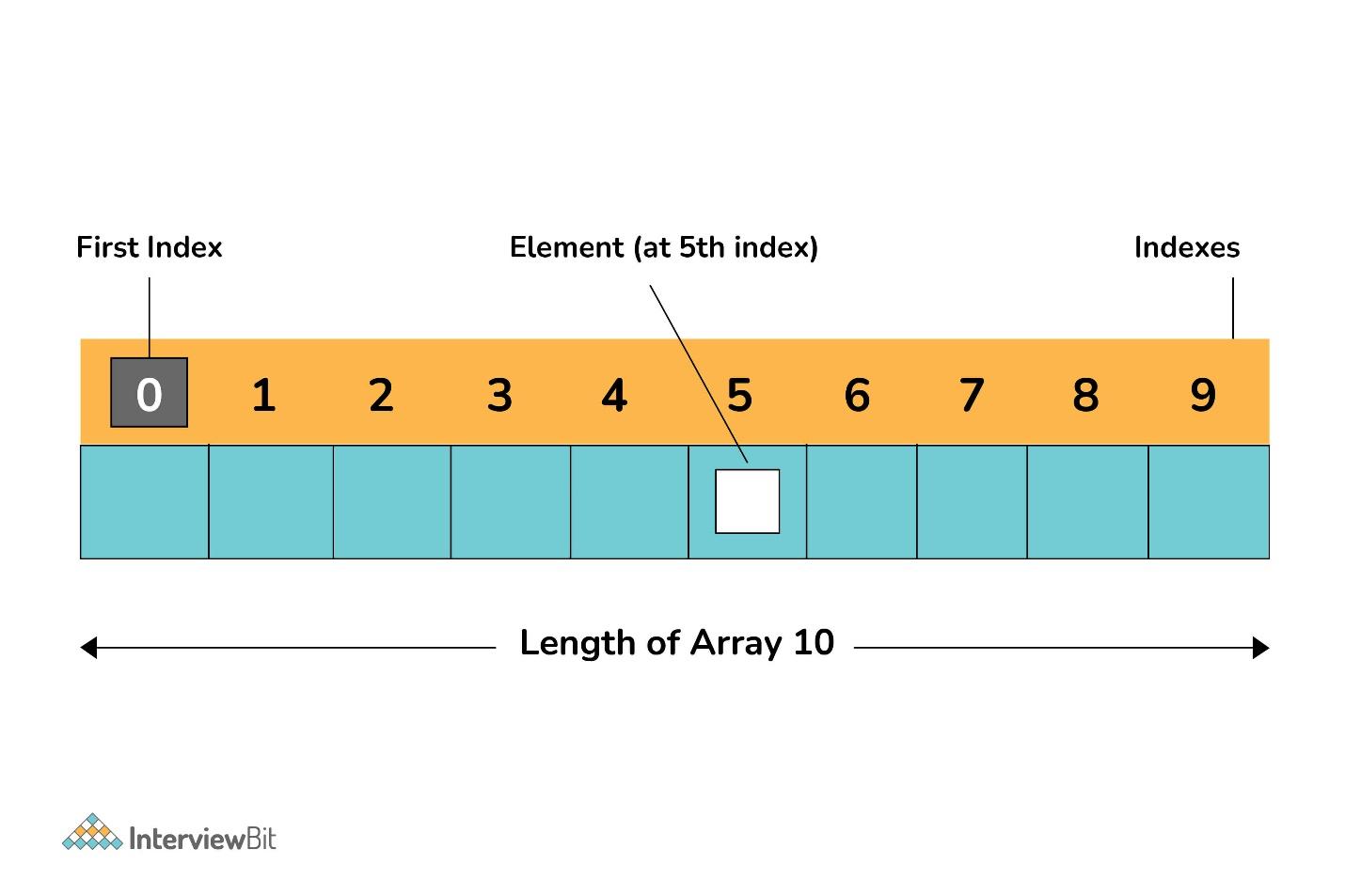
**8. What are the different types of Array in PHP?**

There are 3 main types of arrays that are used in PHP:

Types of Arrays in PHP

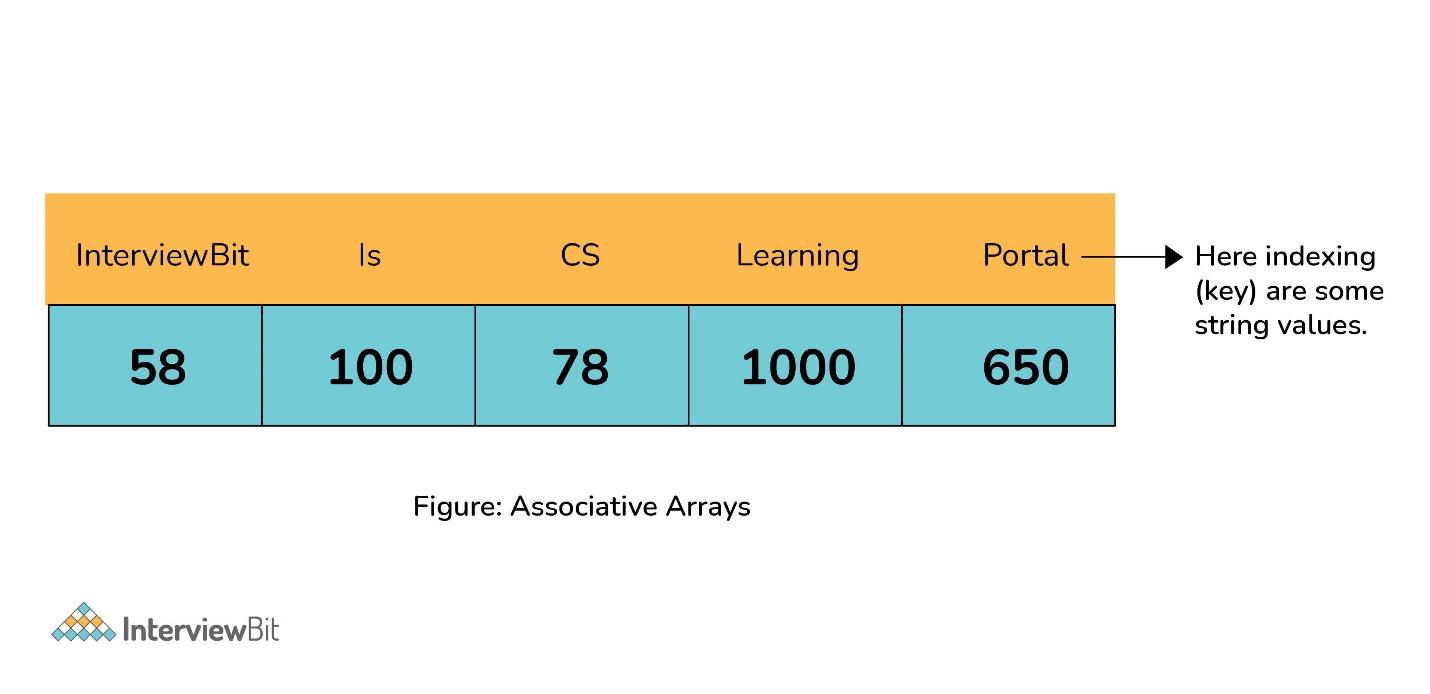
**Indexed Array**

An array with a numeric key is known as the indexed array. Values are stored and accessed in linear order.

Indexed Array

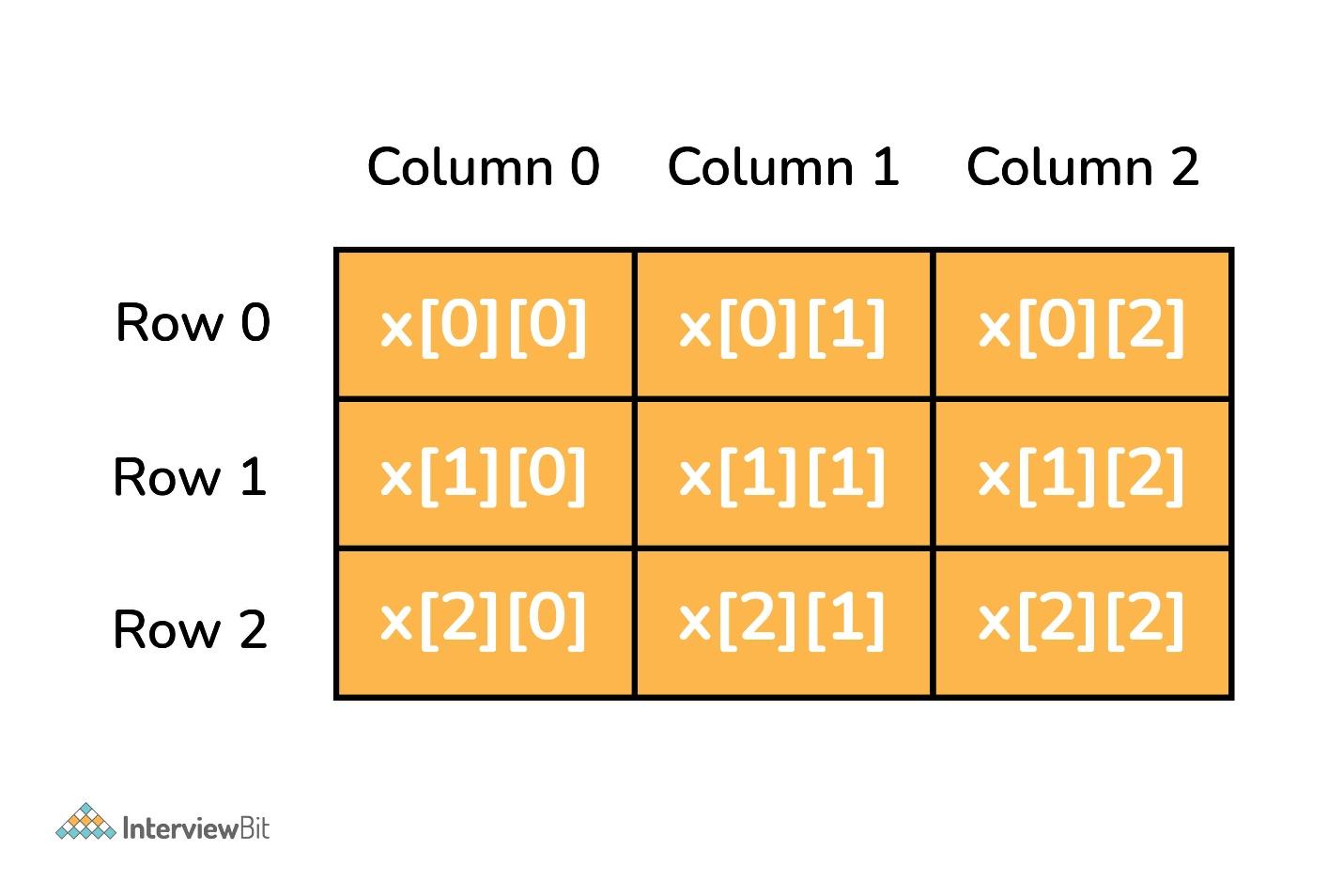
**Associative Array**

An array with strings for indexing elements is known as the associative array. Element values are stored in association with key values rather than in strict linear index order.

Associative Array

**Multidimensional Array**

An array containing one or more arrays within itself is known as a multidimensional array. The values are accessed using multiple indices.

Multidimensional Array

**9. Explain the main types of errors.**

**The 3 main types of errors in PHP are:**

* **Notices: Notices are non-critical errors that can occur during the execution of the script. These are not visible to users. Example: Accessing an undefined variable.**
* **Warnings: These are more critical than notices. Warnings don’t interrupt the script execution. By default, these are visible to the user. Example: include() a file that doesn’t exist.**
* **Fatal: This is the most critical error type which, when occurs, immediately terminates the execution of the script. Example: Accessing a property of a non-existent object or require() a non-existent file.**

**10. What is the difference between the include() and require() functions?**

**include() function**

This function is used to copy all the contents of a file called within the function, text wise into a file from which it is called.

When the file is included cannot be found, it will only produce a warning (E\_WARNING) and the script will continue the execution.

**require() function:**

The require() function performs same as the include() function. It also takes the file that is required and copies the whole code into the file from where the require() function is called.

When the file is included cannot be found, it will produce a fatal error (E\_COMPILE\_ERROR) and terminates the script.

**11. What are cookies? How to create cookies in PHP?**

A cookie is a small record that the server installs on the client’s computer. They store data about a user on the browser. It is used to identify a user and is embedded on the user’s computer when they request a particular page. Each time a similar PC asks for a page with a program, it will send the cookie as well.

After verifying the user’s identity in encrypted form, cookies maintain the session id generated at the back end. It must reside in the browser of the machine. You can store only string values not object because you cannot access any object across the website or web apps.

By default, cookies are URL particular. For example, Gmail cookies are not supported by Yahoo and vice versa. Cookies are temporary and transitory by default. Per site 20 cookies can be created in a single website or web app. 50 bytes is the initial size of the cookie and 4096 bytes is the maximum size of the cookie.

**In PHP, we can create cookies using the setcookie() function:**

setcookie(name, value, expire, path, domain, secure, httponly);

Here name is mandatory and the remaining parameters are optional.

**12. What is the use of session\_start() and session\_destroy() functions in PHP?**

The session\_start() function is used to start a new session. Also, it can resume an existing session if it is stopped. In this particular case, the return will be the current session if resumed.

Syntax:

session\_start();

The session\_destroy() function is used to destroy all of the session variables as given below:

<?php

session\_start();

session\_destroy();

?>

**15. Differentiate between GET and POST**

The difference between GET and POST are given below:

| **GET** | **POST** |
| --- | --- |
| GET method is used for requesting data from a specified resource. | POST is used for sending the data to the server as a package in a separate communication with the processing script. |
| Data is sent in the form of URL parameters which are strings of name-value pairs separated by ampersands(&) | Data sent through the POST method will not be seen in the URL |
| GET method cannot be used for sending binary data like images or word documents | The POST method can be used to send ASCII as well as binary data like images and word documents |
| This method must not be used if you have any sensitive information like a password to be sent to the server. | Sensitive information can be sent using this method. |
| It can be used for submitting the form where the user can bookmark the result. | Submissions by form with POST cannot be bookmarked. |
| You can use this method only for data that is not secure. | Data sent through this method is secure. |
| GET method is not safer since parameters may be stored in web server logs or browser history. | POST method is safer than GET because the parameters are not stored in web server logs or browser history. |

**16. What is the difference between “echo” and “print” in PHP?**

**The main difference between echo and print in PHP are given below:**

|  |  |
| --- | --- |
| **echo** | **print** |
| **echo can output one or more strings.** | **print can only output one string and it always returns 1.** |
| **echo is faster than print because it does not return any value.** | **print is slower compared to echo.** |
| **If you want to pass more than one parameter to echo, a parenthesis should be used.** | **Use of parenthesis is not required with the argument list.** |

**=======OOP PHP=========**

Four Principles of OOP

The four pillars of object-oriented programming are:

* **Inheritance:** child classes inherit data and behaviors from the parent class
* **Encapsulation:** containing information in an object, exposing only selected information
* **Abstraction:** only exposing high-level public methods for accessing an object
* **Polymorphism:** many methods can do the same task

Classes and objects are the two main aspects of object-oriented programming.

Look at the following illustration to see the difference between class and objects:

* **class**
* Fruit
* **objects**
* Apple
* Banana
* Mango

##A class is defined by using the class keyword, followed by the name of the class and a pair of curly braces ({}). All its properties and methods go inside the braces:

Syntax

<?php  
class Fruit {  
  // code goes here...  
}  
?>

Define Objects

Classes are nothing without objects! We can create multiple objects from a class. Each object has all the properties and methods defined in the class, but they will have different property values.

Objects of a class are created using the new keyword.

**PHP - The \_\_construct Function**

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.

Notice that the construct function starts with two underscores (\_\_)!

**PHP - The \_\_destruct Function**

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.

Notice that the destruct function starts with two underscores (\_\_)!

**PHP - Access Modifiers**

Properties and methods can have access modifiers which control where they can be accessed.

There are three access modifiers:

* public - the property or method can be accessed from everywhere. This is default
* protected - the property or method can be accessed within the class and by classes derived from that class
* private - the property or method can ONLY be accessed within the class

**PHP - What is Inheritance?**

Inheritance in OOP = When a class derives from another class.

The child class will inherit all the public and protected properties and methods from the parent class. In addition, it can have its own properties and methods.

An inherited class is defined by using the extends keyword.

**PHP - Class Constants**

Class constants can be useful if you need to define some constant data within a class.

A class constant is declared inside a class with the const keyword.

A constant cannot be changed once it is declared.

Class constants are case-sensitive. However, it is recommended to name the constants in all uppercase letters.

We can access a constant from outside the class by using the class name followed by the scope resolution operator (::) followed by the constant name, like here

<?php  
class Goodbye {  
  const LEAVING\_MESSAGE = "Thank you for visiting W3Schools.com!";  
}  
  
echo Goodbye::LEAVING\_MESSAGE;  
?>

Or, we can access a constant from inside the class by using the self keyword followed by the scope resolution operator (::) followed by the constant name, like here:

Example

<?php  
class Goodbye {  
  const LEAVING\_MESSAGE = "Thank you for visiting W3Schools.com!";  
  public function byebye() {  
    echo self::LEAVING\_MESSAGE;  
  }  
}  
  
$goodbye = new Goodbye();  
$goodbye->byebye();  
?>

**PHP - What are Abstract Classes and Methods?**

Abstract classes and methods are when the parent class has a named method, but need its child class(es) to fill out the tasks.

An abstract class is a class that contains at least one abstract method. An abstract method is a method that is declared, but not implemented in the code.

Syntax

<?php  
abstract class ParentClass {  
  abstract public function someMethod1();  
  abstract public function someMethod2($name, $color);  
  abstract public function someMethod3() : string;  
}  
?>

**PHP - What are Interfaces?**

Interfaces allow you to specify what methods a class should implement.

Interfaces make it easy to use a variety of different classes in the same way. When one or more classes use the same interface, it is referred to as "polymorphism".

Interfaces are declared with the interface keyword:

Syntax

<?php  
interface InterfaceName {  
  public function someMethod1();  
  public function someMethod2($name, $color);  
  public function someMethod3() : string;  
}  
?>

**PHP - What are Traits?**

PHP only supports single inheritance: a child class can inherit only from one single parent.

So, what if a class needs to inherit multiple behaviors? OOP traits solve this problem.

Traits are used to declare methods that can be used in multiple classes. Traits can have methods and abstract methods that can be used in multiple classes, and the methods can have any access modifier (public, private, or protected).

Traits are declared with the trait keyword:

Syntax

<?php  
trait TraitName {  
  // some code...  
}  
?>

**PHP - Static Methods**

Static methods can be called directly - without creating an instance of the class first.

Static methods are declared with the static keyword:

Syntax

<?php  
class *ClassName* {  
  public static function *staticMethod*() {  
    echo "Hello World!";  
  }  
}  
?>

To access a static method use the class name, double colon (::), and the method name:

Syntax

*ClassName*::*staticMethod*();

Example

<?php  
class greeting {  
  public static function welcome() {  
    echo "Hello World!";  
  }  
}  
  
// Call static method  
greeting::welcome();  
?>

**PHP - Static Properties**

Static properties can be called directly - without creating an instance of a class.

Static properties are declared with the static keyword:

Syntax

<?php  
class ClassName {  
  public static $staticProp = "W3Schools";  
}  
?>

To access a static property use the class name, double colon (::), and the property name:

Syntax

ClassName::$staticProp;

Let's look at an example:

Example

<?php  
class pi {  
  public static $value = 3.14159;  
}  
  
// Get static property  
echo pi::$value;  
?>

**PHP Namespaces**

Namespaces are qualifiers that solve two different problems:

1. They allow for better organization by grouping classes that work together to perform a task
2. They allow the same name to be used for more than one class

For example, you may have a set of classes which describe an HTML table, such as Table, Row and Cell while also having another set of classes to describe furniture, such as Table, Chair and Bed. Namespaces can be used to organize the classes into two different groups while also preventing the two classes Table and Table from being mixed up.

**Declaring a Namespace**

Namespaces are declared at the beginning of a file using the namespace keyword:

Syntax

Declare a namespace called Html:

<?php  
namespace Html;  
?>

**Note:** A namespace declaration must be the first thing in the PHP file. The following code would be invalid:

<?php  
echo "Hello World!";  
namespace Html;  
...  
?>

**PHP - What is an Iterable?**

An iterable is any value which can be looped through with a foreach() loop.

The iterable pseudo-type was introduced in PHP 7.1, and it can be used as a data type for function arguments and function return values.

**PHP - Using Iterables**

The iterable keyword can be used as a data type of a function argument or as the return type of a function:

Example

Use an iterable function argument:

<?php  
function printIterable(iterable $myIterable) {  
  foreach($myIterable as $item) {  
    echo $item;  
  }  
}  
  
$arr = ["a", "b", "c"];  
printIterable($arr);  
?>

**Explain the differences between abstract classes and interfaces.**

|  |  |
| --- | --- |
| Interfaces | Abstract classes |
| Interfaces cannot have properties. | Abstract classes can have properties. |
| All interfaces must be public. | Abstract classes can be public or protected. |
| In the interface, we can only use public access specifiers. | In abstract classes, we can use all access specifiers. |

**What are the differences between a session and a cookie?**

|  |  |
| --- | --- |
| Cookie | Session |
| It is stored on the user’s web browser. | It is stored server-side, that is, on the web server. |
| It lasts for a longer time, even after the user closes their browser. | It lasts for a shorter period of time, usually only until the user closes their browser. |
| It can only hold a small amount of data, up to 4KB. | It can hold a large amount of data, usually 64mb or 128mb. |

**34. What are magic methods in PHP?**

Magic methods in PHP are special built-in methods that start with a double underscore ” \_ “. Few examples of magic methods are:

* \_\_construct() – The constructor method that is called automatically when an object is created
* \_\_destruct() – The destructor method that is invoked when the object is destroyed
* \_\_get() – Invoked when trying to read a non-accessible property
* \_\_set() – Automatically called when trying to assign a value to a non-accessible property