**Creating (Declaring) PHP Variables**

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total\_volume).

Rules for PHP variables:

* A variable starts with the $ sign, followed by the name of the variable
* A variable name must start with a letter or the underscore character
* A variable name cannot start with a number
* A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_ )
* Variable names are case-sensitive ($age and $AGE are two different variables)
* Php variable name are case sensitive

### Example

<?php  
$x = 5;  
$y = 4;  
echo $x + $y;  
?>

PHP has three different variable scopes:

* local
* global
* static

A variable declared **outside** a function has a GLOBAL SCOPE and can only be accessed outside a function.

### Example

<?php  
$x = 5; // global scope  
function myTest() {  
  // using x inside this function will generate an error  
  echo "<p>Variable x inside function is: $x</p>";  
}   
myTest();  
  
echo "<p>Variable x outside function is: $x</p>";  
?>

A variable declared **within** a function has a LOCAL SCOPE and can only be accessed within that function:

### Example

<?php  
function myTest() {  
  $x = 5; // local scope  
  echo "<p>Variable x inside function is: $x</p>";  
}   
myTest();  
// using x outside the function will generate an error

echo "<p>Variable x outside function is: $x</p>";  
?>

**PHP echo and print Statements**

echo and print are more or less the same. They are both used to output data to the screen.

The differences are small: echo has no return value while print has a return value of 1 so it can be used in expressions. echo can take multiple parameters (although such usage is rare) while print can take one argument. echo is marginally faster than print.

**PHP Data Types**

Variables can store data of different types, and different data types can do different things.

PHP supports the following data types:

* String
* Integer
* Float (floating point numbers - also called double)
* Boolean
* Array
* Object
* NULL
* Resource

## PHP String

A string is a sequence of characters, like "Hello world!".

A string can be any text inside quotes. You can use single or double quotes:

### Example

<?php   
$x = "Hello world!";  
$y = 'Hello world!';  
  
echo $x;  
echo "<br>";   
echo $y;  
?>

## PHP Integer

An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.

Rules for integers:

* An integer must have at least one digit
* An integer must not have a decimal point
* An integer can be either positive or negative
* Integers can be specified in: decimal (base 10), hexadecimal (base 16), octal (base 8), or binary (base 2) notation

In the following example $x is an integer. The PHP var\_dump() function returns the data type and value.

### Example

<?php   
$x = 5985;  
var\_dump($x);  
?>

## PHP Float

A float (floating point number) is a number with a decimal point or a number in exponential form.

In the following example $x is a float. The PHP var\_dump() function returns the data type and value:

### Example

<?php   
$x = 10.365;  
var\_dump($x);  
?>

**PHP Boolean**

A Boolean represents two possible states: TRUE or FALSE.

$x = true;  
$y = false;

## PHP Array

An array is a special variable which can hold more than one value at a time.

### Example

<?php   
$cars = array(0=>"Volvo",1=>"BMW",2=>"Toyota");

print\_r ($cars);

?>

**PHP Object**

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

A class is a template for objects, and an object is an instance of a class.

**PHP NULL Value**

Null is a special data type which can have only one value: NULL.

A variable of data type NULL is a variable that has no value assigned to it.

If a variable is created without a value, it is automatically assigned a value of NULL.

**PHP String Functions:-**

1. The PHP **strlen()** function returns the length of a string.

### Example

<?php  
echo strlen("Hello world!"); // outputs 12  
?>

1. The PHP **str\_word\_count()** function counts the number of words in a string.

### Example

<?php  
echo str\_word\_count("Hello world!"); // outputs 2  
?>

1. The PHP **strrev()** function reverses a string.

### Example

<?php  
echo strrev("Hello world!"); // outputs !dlrow olleH  
?>

1. The PHP **str\_replace()** function replaces some characters with some other characters in a string.

### Example

<?php  
echo str\_replace("world", "Dolly", "Hello world!"); // outputs Hello Dolly!  
?>

## Arithmetic Operators

There are following arithmetic operators supported by PHP language −

Assume variable A holds 10 and variable B holds 20 then

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| + | Adds two operands | A + B will give 30 |
| - | Subtracts second operand from the first | A - B will give -10 |
| \* | Multiply both operands | A \* B will give 200 |
| / | Divide numerator by de-numerator | B / A will give 2 |
| % | Modulus Operator and remainder of after an integer division | B % A will give 0 |
| ++ | Increment operator, increases integer value by one | A++ will give 11 |
| -- | Decrement operator, decreases integer value by one | A-- will give 9 |
| Comparison Operators There are following comparison operators supported by PHP language  Assume variable A holds 10 and variable B holds 20 then:   |  |  |  | | --- | --- | --- | | **Operator** | **Description** | **Example** | | == | Checks if the value of two operand are equal or not, if yes then condition becomes true. | (A == B) is not true. | | != | Checks if the value of two operands are equal or not, if values are not equal then condition becomes true. | (A != B) is true. | | > | Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true. | (A > B) is not true. | | < | Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true. | (A < B) is true. | | >= | Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true. | (A >= B) is not true. | | <= | Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true | (A <= B) is true. | |  |  |

**PHP Conditional Statements**

Very often when you write code, you want to perform different actions for different conditions. You can use conditional statements in your code to do this.

In PHP we have the following conditional statements:

* if statement - executes some code if one condition is true.
* if...else statement - executes some code if a condition is true and another code if that condition is false
* if...elseif...else statement - executes different codes for more than two conditions
* switch statement - selects one of many blocks of code to be executed

**PHP Loops**

Often when you write code, you want the same block of code to run over and over again a certain number of times. So, instead of adding several almost equal code-lines in a script, we can use loops.

Loops are used to execute the same block of code again and again, as long as a certain condition is true.

In PHP, we have the following loop types:

* while - loops through a block of code as long as the specified condition is true
* do...while - loops through a block of code once, and then repeats the loop as long as the specified condition is true. In a do...while loop the condition is tested AFTER executing the statements within the loop. This means that the do...while loop will execute its statements at least once, even if the condition is false.
* for - loops through a block of code a specified number of times
* foreach - loops through a block of code for each element in an array

## The PHP for Loop

The for loop is used when you know in advance how many times the script should run.

### Syntax

for (*init counter; test counter; increment counter*) {  
  *code to be executed for each iteration;*  
}

Parameters:

* *init counter*: Initialize the loop counter value
* *test counter*: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
* *increment counter*: Increases the loop counter value

# PHP foreach Loop

The foreach loop - Loops through a block of code for each element in an array. The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.

### Syntax

foreach ($*array* as$*value*) {  
  *code to be executed;*  
}

For every loop iteration, the value of the current array element is assigned to $value and the array pointer is moved by one, until it reaches the last array element.

## PHP - Functions

PHP has over 1000 built-in functions that can be called directly, from within a script, to perform a specific task.

* A function is a block of statements that can be used repeatedly in a program.
* A function will not execute automatically when a page loads.
* A function will be executed by a call to the function.

### Syntax

function *functionName*() {  
*code to be executed*;  
}

**Example:-**

<?php

function name() {

echo “apple is best fruit”;

}

name();

?>

# PHP Arrays

An array stores multiple values in one single variable. In PHP, the array() function is used to create an array: array();

In PHP, there are three types of arrays:

* **Indexed arrays** - Arrays with a numeric index
* **Associative arrays** - Arrays with named keys
* **Multidimensional arrays** - Arrays containing one or more arrays

**PHP Indexed Arrays**

There are two ways to create indexed arrays: The index can be assigned automatically (index always starts at 0), like this

$cars = array("Volvo", "BMW", "Toyota"); or the index can be assigned manually:

$cars[0] = "Volvo";  
$cars[1] = "BMW";  
$cars[2] = "Toyota";

**Example of indexed array**

<?php

    $name = array("sam"=>"30", "ricky"=>"26", "ellary"=>"25");

    echo "Ellary is " . $name['ellary'] . " Years Old.";

?>

Output: - Ellary is 25 Years Old.

**PHP Associative Arrays**

Associative arrays are arrays that use named keys that you assign to them. There are two ways to create an associative array:

$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43"); or:

$age['Peter'] = "35";  
$age['Ben'] = "37";  
$age['Joe'] = "43";

**Example of associative array**

<?php

$name = array("Rocky"=>"26" , "hudson"=>"30" , "Sam"=>"20");

foreach($name as $x => $y) {

echo " Name = " . $x .   ", Age = $y " . "<br>";

}

?>

Output:-

Name = Rocky, Age = 26   
Name = hudson, Age = 30   
Name = Sam, Age = 20

**PHP - Multidimensional Arrays**

A multidimensional array is an array containing one or more arrays.

PHP supports multidimensional arrays that are two, three, four, five, or more levels deep. However, arrays more than three levels deep are hard to manage for most people.

**The dimension of an array indicates the number of indices you need to select an element.**

* For a two-dimensional array you need two indices to select an element
* For a three-dimensional array you need three indices to select an element

**Example of multidimensional array**

<?php

    $name = array(

        array("Sam","20","Usa"),

        array("Ellary","25","Boston"),

    );

    echo $name[0][0] ." is ".$name[0][1] ." live in ". $name[0][2] ."<br>";

    echo $name[1][0] ." is ". $name[1][1] . " live in ". $name[1][2] . "<br>";

?>

Output:-

Sam is 20 live in Usa  
Ellary is 25 live in Boston

**PHP - Sort Functions For Arrays**

In this chapter, we will go through the following PHP array sort functions:

* sort() - sort arrays in ascending order
* rsort() - sort arrays in descending order
* asort() - sort associative arrays in ascending order, according to the value
* ksort() - sort associative arrays in ascending order, according to the key
* arsort() - sort associative arrays in descending order, according to the value
* krsort() - sort associative arrays in descending order, according to the key

Example of sorting in array:-

<?php

    $number = array("3","9","90","57","1","20","40","22");

    sort($number);

    $countnum = count($number);

    for($y = 0; $y < $countnum; $y++) {

        echo $number[$y]."&nbsp";

    }

?>

Output :- 1 3 9 20 22 40 57 90

**What is OOPS ?**

oops stands for object oriented programming structure

oops used to create a dynamic projects

oops used to create a secured web application or website

oops is used to create a application i.e support MVC architectures

**OOPS Features :-**

1. **Inheritance**
2. **Interface**
3. **Abstract**
4. **Encapsulation**
5. **Polymorphism**
6. **Class**
7. **Object**

**CLASS:- A class is a member and its member function. Class is an instance of object.**

<?php

class name

{

    public function test()

    {

        echo "Hello, What is Your Name?";

    }

}

$obj=new name;

$obj->test();

?>

// Output - Hello, What is Your Name?

**OBJECT:- An object is instances of class. Object will be called through new keyword.**

<?php

/\*

Syntax of Object:-

class myclass

{

   ...

}

$obj=new myclass;

\*/

class test

{

    public function demo()

    {

        $details="Laravel is framework of PHP";

        echo "$details";

    }

}

$obj=new test;

$obj->demo();

?>

// Output - Laravel is framework of PHP

**PSEUDO VARIABLE:- Pseudo variable is used to call direct inside of method or globally access any variables in the method.**

**It is used with $this variable.**

<?php

class name

{

    public $surname="My name is Kishan & surname is Patel";

    public function test()

    {

        echo $this->surname;

    }

}

$obj=new name;

$obj->test();

?>

// Output - My name is Kishan & surname is Patel

?>

**INHERITANCE:- Inheritance is used to access one class property with another class it is called inheritance.**

**There are 3 types of inheritance:**

1. **Single Inheritance**

**Here in single inheritance one class property only access to its one child.**

<?php

    class fruit

    {

        public function item()

        {

            $name = "Fuits are the best diet option.";

            echo $name."<br>";

        }

    }

    class fru\_type extends fruit

    {

        public function item1()

        {

            $fruit = "Mango is the king of fruit.";

            echo "$fruit";

        }

    }

    $obj= new fru\_type;

    $obj->item();

    $obj->item1();

?>

// Output - Fuits are the best diet option.  
 Mango is the king of fruit.

1. **Multilevel Inheritance**

**Here in multilevel inheritance if a=>b=>c=>d if this chain occurs then it is multilevel inheritance.**

<?php

    class fruit

    {

        public function display()

        {

            $name = "Fuits are the best option in breakfast.";

            echo "$name"."<br>";

        }

    }

    class fru\_name extends fruit

    {

        public function display1()

        {

            $frname = "one should eat banana in breakfast good for health";

            echo "$frname"."<br>";

        }

    }

    class fru\_type extends fru\_name

    {

        public function display2()

        {

            $frtype = "Banana gives us carbohydrates";

            echo "$frtype"."<br>";

        }

    }

    class fru\_price extends fru\_type

    {

        public function display3()

        {

            $frprice = "Rate of Banana is 20 for 1 Kg.";

            echo "$frprice";

        }

    }

$obj=new fru\_price;

$obj->display();

$obj->display1();

$obj->display2();

$obj->display3();

?>

// output - Fuits are the best option in breakfast.  
 one should eat banana in breakfast good for health  
 Banana gives us carbohydrates  
 Rate of Banana is 20 for 1 Kg.

?>

1. **Multiple Inheritance**

**Multiple inheritance is not supported in php & java it is only supported in c++ and python.**

<?php

//Note : multiple inheritance is not supported in php and java only c++ and python will support multiple inheritance

    class fruit

    {

        public function display()

        {

            $name = "Fuits are the best option in breakfast.";

            echo "$name"."<br>";

        }

    }

    class fru\_name extends fruit

    {

        public function display1()

        {

            $frname = "one should eat banana in breakfast good for health";

            echo "$frname"."<br>";

        }

    }

    class fru\_type extends fru\_name,fruit

    {

        public function display2()

        {

            $frtype = "Banana gives us carbohydrates";

            echo "$frtype"."<br>";

        }

    }

    $obj= new fru\_type;

    $obj->display();

    $obj->display1();

    $obj->display2();

?>

**INTERFACE:-** 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:

To implement an interface, a class must use the implements keyword.

<?php

    interface A

    {

        public function display();

    }

    interface B

    {

        public function display1();

    }

    interface C

    {

        public function display2();

    }

    class D implements A,B,C

    {

        public function display()

        {

            echo"Fuits are the best option in breakfast.";

            echo "<br>";

        }

        public function display1()

        {

            echo"One should eat banana in breakfast good for health.";

            echo "<br>";

        }

        public function display2()

        {

            echo"Banana gives us carbohydrates.";

        }

    }

    $obj= new D;

    $obj->display();

    $obj->display1();

    $obj->display2();

?>

// Output - Fuits are the best option in breakfast.  
 One should eat banana in breakfast good for health.  
 Banana gives us carbohydrates.

?>

**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:

<?php

    trait A

    {

        public function test()

        {

            $name = "Fuits are the best option in breakfast.";

            echo "$name"."<br>";

        }

    }

    trait B

    {

        public function test1()

        {

            $frtype = "Banana gives us carbohydrates";

            echo "$frtype"."<br>";

        }

    }

    class C

    {

        use A,B;

    }

$obj= new C;

$obj->test();

$obj->test1()

?>

// Output:- Fuits are the best option in breakfast.  
 Banana gives us carbohydrates

?>

**POLYMORPHISM:-** Polymorphism in OOPs is a concept that allows you to create classes with different functionalities in a single interface.

Polymorphism in php does not support method overloading.

1. **Method Overloading (compiletime)**

<?php

class A

{

    public function number($num1,$num2)

    {

        $num=$num1+$num2;

        echo $num;

    }

}

class B extends A

{

    public function number($num1,$num2,$num3)

    {

        $num=$num1\*$num2\*$num3;

        echo $num;

    }

}

$obj=new B;

$obj->number(2,7,9);

?>

1. **Method Overriding (runtime)**

<?php

    class A

    {

        public function test($number1,$number2)

        {

          $num = $number1\*$number2;

          echo $num;

        }

    }

    class B extends A

    {

        public function test($number1,$number2)

        {

            $num = $number1+$number2;

            echo $num;

        }

    }

$obj= new B;

$obj->test(100,400);

?>

//Output:- 500

**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.

1. **User Defined Constructor**

<?php

    class item

    {

        public function item()

        {

           $name="My name is patel";

           echo $name;

        }

    }

$boj=new item;

// User Defined Constructor does not support the php version 8, it gives the output in verion 7 or earlier.

// so that's why we do not use the user defined constructor.

?>

1. **Default Constructor**

<?php

class A

{

    public function \_\_construct()

    {

        $name="Hello how are you?";

        echo $name;

    }

}

$boj=new A;

//Output:- Hello how are you?

?>

1. **Parameterized Constructor**

<?php

class item

{

    public function \_\_construct($num1,$num2)

    {

        $number=$num1\*$num2;

        echo "Multiplication of two number is : ".$number;

    }

}

$obj=new item(70,50);

// Output:- Multiplication of two number is : 3500

?>