echo has no return value while print has a return value of 1.

echo can take multiple parameters (although such usage is rare) while print can take one argument.

 echo is marginally faster than print.

The both (echo and print) statement can be used with or without parentheses: echo or echo().print or print().

‘\n’ for new line

**Print:**

<?php  
$txt1 = "Learn PHP";  
$txt2 = "W3Schools.com";  
$x = 5;  
$y = 4;  
  
echo "<h2>" . $txt1 . "</h2>";  
echo "Study PHP at " . $txt2 . "<br>";  
echo $x + $y;  
?>

**Echo:**

<?php  
$txt1 = "Learn PHP";  
$txt2 = "W3Schools.com";  
$x = 5;  
$y = 4;  
  
print "<h2>" . $txt1 . "</h2>";  
print "Study PHP at " . $txt2 . "<br>";  
print $x + $y;  
?>

**Learn PHP**

Study PHP at W3Schools.com  
9

**$this** is a pseudo-variable which is a reference to the current object.

**$this** is mostly used in object oriented code.

$this variable is used to call non-static method, if you are trying to call static method then it will throw the error that means $this variable is not available inside the static method.

PHP reference:

$b=2;

$a=$b;

$a=3;

print $a;

print $b;

// output is 32

$b=2;

$a=&$b; // note the & operator

$a=3;

print $a;

print $b;

// output is 33

The **sprintf()** function writes a formatted string to a variable.

sprintf() work with echo.

$txt = sprintf("There are %u million bicycles in %s.",$number,$str);

#### Note:**The sprintf() works with echo. The formatted string returns by the sprintf() function is printed by echo on the browser whereas printf() directly put the output on browser.**

**Gettype()-**return type of variable

The gettype() function is an inbuilt function in PHP which is used to get the type of a variable. It is used to check the type of existing variable.

$var1 = true; // boolean value

$var2 = 3; // integer value

$var3 = 5.6; // double value // for historical reasons “double” is returned in case of float)

$var4 = "Abc3462"; // string value

$var5 = array(1, 2, 3); // array value

$var6 = new stdClass; // object value

$var7 = NULL; // null value

$var8 = tmpfile(); // resource value

echo gettype($var1)."\n";

echo gettype($var2)."\n";

echo gettype($var3)."\n";

echo gettype($var4)."\n";

echo gettype($var5)."\n";

echo gettype($var6)."\n";

echo gettype($var7)."\n";

echo gettype($var8)."\n";

**output**:

boolean

integer

double

string

array

object

NULL

resource

In PHP, **objects passed by value not reference.**

**session\_unset()** function frees all session variables.

The boolean function **isset** determines if a variable is set and is not NULL.

It is possible to use the document.form.submit() function to submit the form. For example: <input type=button value="SUBMIT" onClick="document.form.submit()">

PHP supports **only single inheritance**;

The following values are considered to be empty:

These values are considered to be as an empty value:

* **“” (an empty string)**
* **0 (0 as an integer)**
* **0.0 (0 as a float)**
* **“0” (0 as a string)**
* **NULL**
* **FALSE**
* **array() (an empty array)**

$var1 = 0;

$var2 = 0.0;

$var3 = "0";

$var4 = NULL;

$var5 = false;

$var6 = array();

$var7 = "";

// for value 0 as integer

empty($var1) ? print\_r("True\n") : print\_r("False\n");

// for value 0.0 as float

empty($var2) ? print\_r("True\n") : print\_r("False\n");

// for value 0 as string

empty($var3) ? print\_r("True\n") : print\_r("False\n");

// for value Null

empty($var4) ? print\_r("True\n") : print\_r("False\n");

// for value false

empty($var5) ? print\_r("True\n") : print\_r("False\n");

// for array

empty($var6) ? print\_r("True\n") : print\_r("False\n");

// for empty string

empty($var7) ? print\_r("True\n") : print\_r("False\n");

// for not declare $var8

empty($var8) ? print\_r("True\n") : print\_r("False\n");

?> //all return true

PHP has three different variable scopes:

* local
* global
* static

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

<?php

$x = 5; // global scope

function myTest()

{// using x inside this function will generate an error

echo "$x";

}

myTest();

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

?>

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

3.The global keyword is used to access a global variable from within a function.

To do this, use the global keyword before the variables (inside the function):

<?php

$x = 5;

$y = 10;

function myTest() {

global $x, $y;

$y = $x + $y;

} myTest(); // run function

echo $y; // output the new value for variable $y

?> // outputs 15

4. PHP also stores all global variables in an array called $GLOBALS[*index*]. The *index* holds the name of the variable.

<?php  
$x = 5;  
$y = 10;  
function myTest() {  
    $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];  
}  
myTest();  
echo $y; // outputs 15  
?>

5. when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted.

To do this, use the static keyword when you first declare the variable:

<?php  
function myTest() {  
    static $x = 0;  
    echo $x;  
    $x++;  
}  
myTest();  
myTest();  
myTest();  
?> // outputs 0 1 2

MCQ:

Link1: <https://www.examveda.com/php/practice-mcq-question-on-basic-php/?page=11>

Link2: <https://www.examveda.com/mcq-question-on-php/>

**Is PHP a case sensitive language ?**  
No, PHP is a partially case sensitive language. It means the variable names are **case-sensitive** and the function names are **not case sensitive** i.e. user-defined functions are not case sensitive.

**What is the main difference between PHP 4 and PHP 5 ?**  
PHP 5 contains many additional OOP (object-oriented programming) features.

[**What are Traits in PHP ?**](https://www.geeksforgeeks.org/multiple-inheritance-in-php/)  
The trait is a type of class which enables **multiple inheritance**.

**Contant in PHP**

A valid constant name starts with a letter or underscore (**no $ sign** before the constant name). **Note:** Unlike variables, constants are automatically **global across the entire script(can be use in function);**

To create a constant, use the define() function.

Syntax: define(*name*, *value*, *case-insensitive*)

*case-insensitive*: Specifies whether the constant name should be case-insensitive. **Default is false**

<?php  
define("GREETING", "Welcome to W3Schools.com!");  
echo GREETING;  
?>output: Welcome to W3Schools.com!

<?php  
define("GREETING", "Welcome to W3Schools.com!", true);  
echo greeting;  
?>output: Welcome to W3Schools.com!

**Data types** in PHP:boolen,string,float,**array object,null,integer**

Array Function in php: read Pdf page No:265

Array:

Create an Array in PHP

In PHP, the array() function is used to create an array:

array();

Example:

<?php  
$cars = array("Volvo", "BMW", "Toyota");  
echo count($cars);  
?>

## PHP Indexed Arrays

<?php  
$cars = array("Volvo", "BMW", "Toyota");  
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";  
?>

## PHP Associative Arrays

<?php  
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");  
echo "Peter is " . $age['Peter'] . " years old.";  
?>

## PHP - Multidimensional Arrays

**Implode** function return a string from element of an array

**Explode** function breaks the string into array

**Fatal error** stop the execution of script.

File uploading

<form action="upload.php" method="post" **enctype="multipart/form-data**">

**multipart/form-data:** No characters are encoded. This value is required when you are using forms that have a file upload control

## The PHP Filter Extension

Validating data = Determine if the data is in proper form.

Sanitizing data = Remove any illegal character from the data.

## PHP filter\_var() Function

The **filter\_var()** function both **validate and sanitize** data.

The **filter\_var()** function filters a single variable with a specified filter. It takes two pieces of data:

* The variable you want to check
* The type of check to use

## Sanitize a String

The following example uses the filter\_var() function to remove all HTML tags from a string:

### **Example**

<?php  
$str = "<h1>Hello World!</h1>";  
$newstr = filter\_var($str, FILTER\_SANITIZE\_STRING);  
echo $newstr;  
?>

## Validate an Integer

<?php  
$int = 100;  
  
if (!filter\_var($int, FILTER\_VALIDATE\_INT) === false) {  
    echo("Integer is valid");  
} else {  
    echo("Integer is not valid");  
}  
?>

## Validate an IP Address

FILTER\_VALIDATE\_IP

## Validate an Email Address

FILTER\_VALIDATE\_EMAIL

**Types of error**

Basically there are four types of errors in PHP, which are as follows:

* **Parse Error** (Syntax Error):missing something(like”,$.. and stop execution
* **Fatal Error:(** access the undefined functions and stop executipn)
* **Warning Error:(**missing file and not stop execution)
* **Notice Error**:( access the undefined variable and not stop execution)

# **Traits vs. Interfaces in PHP**

**Traits:multiple inheritance**

Traits are a mechanism for code reuse in single inheritance languages such as PHP. Write the same code again, to avoid this use the traits. The traits are used when multiple classes share the same functionality.

<?php  
class Base {  
    public function sayHello() {  
        echo 'Hello ';  
    }  
}  
  
trait SayWorld {  
    public function sayHello() {  
        parent::sayHello();  
        echo 'World!';  
    }  
}  
  
class MyHelloWorld extends Base {  
    use SayWorld;  
}  
  
$o = new MyHelloWorld();  
$o->sayHello();  
?>

Output: Hello World!

Example:2

<?php  
trait HelloWorld {  
    public function sayHello() {  
        echo 'Hello World!';  
    }  
}  
  
class TheWorldIsNotEnough {  
    use HelloWorld;  
    public function sayHello() {  
        echo 'Hello Universe!';  
    }  
}  
  
$o = new TheWorldIsNotEnough();  
$o->sayHello();  
?>

Output: Hello Universe

Example:3

<?php  
trait Hello {  
    public function sayHello() {  
        echo 'Hello ';  
    }  
}  
  
trait World {  
    public function sayWorld() {  
        echo 'World';  
    }  
}  
  
class MyHelloWorld {  
    use Hello, World;  
    public function sayExclamationMark() {  
        echo '!';  
    }  
}  
  
$o = new MyHelloWorld();  
$o->sayHello();  
$o->sayWorld();  
$o->sayExclamationMark();  
?>

The above example will output:

Hello World!

**Interface:**

<?php

// PHP program to demonstrate working

// of interface.

interface MyInterface{

    public function examplemethod1();

    public function examplemethod2();

}

class MyClass implements MyInterface{

    public function examplemethod1(){

        echo "ExampleMethod1 Called" . "\n";

    }

    public function examplemethod2(){

        echo "ExampleMethod2 Called". "\n";

    }

}

$ob = new MyClass;

$ob->examplemethod1();

$ob->examplemethod2();

?>