1. The basic syntax is

<?php

/\*

This is multi line comment

\*/

?>

It usually follows syntax from C++.

1. 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 \_ )
2. 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)

1. <?php

echo("Subin Oh \n Pizza \n Soccer \n Colorado");

?>

1. <?php

$age= “25”

echo $age

?>

<?php

$PI= “3.1415”

Echo $PI

?>

1. A PHP function provides code that a PHP script can call to perform a task, such as Count(), file\_get\_contents(), and header(). The PHP language supports both procedural and object-oriented programming paradigms.
2. A constant is an identifier (name) for a simple value. As the name suggests, that value cannot change during the execution of the script (except for magic constants, which aren't actually constants). Constants are case-sensitive. By convention, constant identifiers are always uppercase. A constant is an identifier (name) for a simple value. As the name suggests, that value cannot change during the execution of the script (except for magic constants, which aren't actually constants). Constants are case-sensitive. By convention, constant identifiers are always uppercase.

<?php

//Valid constant names

define("FOO", "something");

define("FOO2", "something else");

define("FOO\_BAR", "something more");

//Invalid constant names

define("2FOO", "something");

// This is valid, but should be avoided:

// PHP may one day provide a magical constant

// that will break your script

define("\_\_FOO\_\_", "something");

?>

1. The reason why echo and print statements are important is that they are programmed with PHP, but eventually output to HTML. It's about outputting content in HTML so that we can see the results in our web browser, and echo and print statements play an easy but important role.

the difference between

echo and print is that the echo and print are not actual functions, but language structures, but print can be used as functions.

factor (input) value

concept of function

concept of return (output) value

function

print has one input and has a return value. i always return a number of 1. print is not a function, but it can be used like a function.

Echo outputs one or more strings. It's about multiple input values. I don't give anything back to the result. Of course, it outputs in HTML, but it doesn't have a return value.

int print (string $arg) - outputs

one string - int means the return value is integer, string refers to the string, $ is the variable, string means string.

void echo (string $arg 1 [, string $...] ) - outputs one or more strings - void says there is no return value, string refers to string, $ means variable, string means string.

Let's start with an example with The PHP echo Statement

bar.

echo "PHP is Fun!";

echo "Hello world!";

echo "I´m about to learn PHP!";

echo "This ", "string ", "was ", "made ", "with multiple

echo can be a string. multiple strings are possible.

the following example outputs a variable in a string.

in addition, variables are computed and outputted.

it's not very difficult.

$txt1 = "Learn PHP";

$txt2 = "subin.com";

$x = 5;

$y = 4;

echo $txt1";

echo "Study PHP at $txt2";

echo $x + $y;

You can use () parentheses in The PHP print Statementprint

, or you can use them without ()parentheses.

print("PHP is Fun!");

print "Hello world!";

print "I´m about to learn PHP!";

let's use variables with printf. it's no different from echo here.

$txt1 = "Learn PHP";

$txt2 = "subin.com";

$x = 5;

$y = 4;

print "$txt1";

print "Study PHP at $txt2";

print $x + $y;

let's take a little hard example to see the difference between echo and print.

let's start with the three-term operator. the ternary operator executes the true content if a variable is true, and false, if it is false.

(what variable) ? true: false content

if $some\_var is true, print ́true ́ is executed and outputs true.

if $some\_var is false, print ́false ́ is executed and printed with false.

($some\_var) ? print ´true´ : print ´false´; no problem

the above print works. this is because it can be written like a function and used as an factor value.

below, echo doesn't work like a function, so the following code is invalid: this is because it cannot be used as an input value.

($some\_var) ? echo ´true´ : echo ´false´; error

so in order to use it as echo, you need to change the code as below.

echo $some\_var ? ´true´: ´false´; no problem

this code is forcibly created to explain the difference between print and echo. they're both similar, the bottom line is that it is better to use echo. because in terms of speed, the echo is slightly faster than print.

1. In PHP, variables can be declared anywhere in the script. variable scope refers to the range within a script that refers to or can use a specific variable.

PHP classifies the types of variables according to the effective range of variables as follows.

1. local variable

2. global variable

local variables

variables declared inside a function can only be accessed from inside the function. in addition, variables declared inside the function are removed from memory when the function's call ends. this variable declared inside the function is called a local variable.

Example)

function varFunc() {

$var = 10; Declare

as a local variable echo "The value of the local variable var called from inside the function is {$var}.<br>";

}

varFunc();

echo "The value of the local variable var called outside the function is {$var}.";

in the example above, we want to refer to the local variable var inside the function outside of the function. however, since the call to the function has ended, all local variables declared inside the function have already been removed from memory. therefore, if you try to reference a local variable outside of a function, you will not get any value.

global variable

variables declared outside of a function can only be accessed directly outside the function. if you want to approach variables declared outside of a function inside a function, you should use the global keyword together. variables declared outside of the function are called global variables.

Example)

$var = 10; Declare

as global variable function varFunc() {

echo "The value of the global variable var called from inside the function is {$var}.<br>";

global $var; Specify the global variable to use within the

function echo "The value of the global variable var called from inside the function is {$var}.<br>";

}

varFunc();

echo "The value of the global variable var called outside the function is {$var}.";

in the example above, the first echo function called does not output any value.

however, the echo function, which is called again after specifying the global variable to use as the global keyword, outputs the value of the correct global variable.

PHP STORES ALL GLOBAL VARIABLES IN A $GLOBALS ARRAY.

if you use the name of a variable as an index in this array, you can access the value of that global variable. this array can also be accessed from inside the function, which allows you to change the value of the global variable directly.

Example)

$var = 10; Declare

as global variable function varFunc() {

echo "The value of the global variable var called from inside the function is {$var}.<br>";

echo "The value of the global variable var called from inside the function is {$GLOBALS ['var']}.<br>";

var

Func();

echo "The value of the global variable var called outside the function is {$var}.";

The example above uses an array of $GLOBALS to access global variables instead of using global keywords inside functions.