PHP echo and print Statements

We frequently use the echo statement to display the output. There are two basic ways to get the output in PHP:

- echo
- print

echo and print are language constructs, and they never behave like a function. Therefore, there is no requirement for parentheses. However, both the statements can be used with or without parentheses. We can use these statements to output variables or strings.

Difference between echo and print

echo

- echo is a statement, which is used to display the output.
- echo can be used with or without parentheses.
- echo does not return any value.
- We can pass multiple strings separated by comma (,) in echo.
- echo is faster than print statement.

print

- print is also a statement, used as an alternative to echo at many times to display the output.
- print can be used with or without parentheses.
- print always returns an integer value, which is 1.
- Using print, we cannot pass multiple arguments.
- print is slower than echo statement.

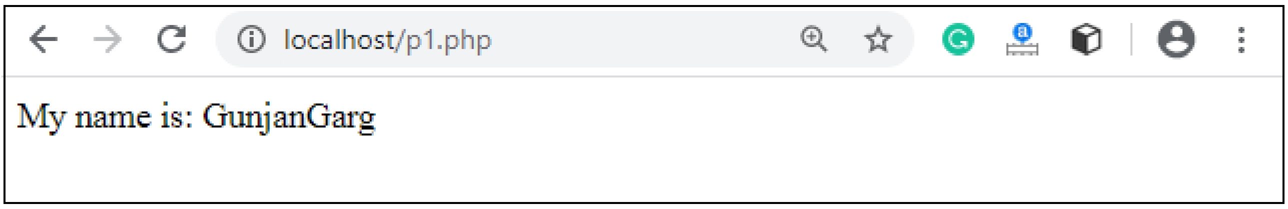
You can see the difference between echo and print statements with the help of the following programs.

For Example (Check multiple arguments)

You can pass multiple arguments separated by a comma (,) in echo. It will not generate any syntax error.

```php
<?php
    $fname = "Gunjan";
    $lname = "Garg";
    echo "My name is: ".$fname,$lname;
?>
```

Output:

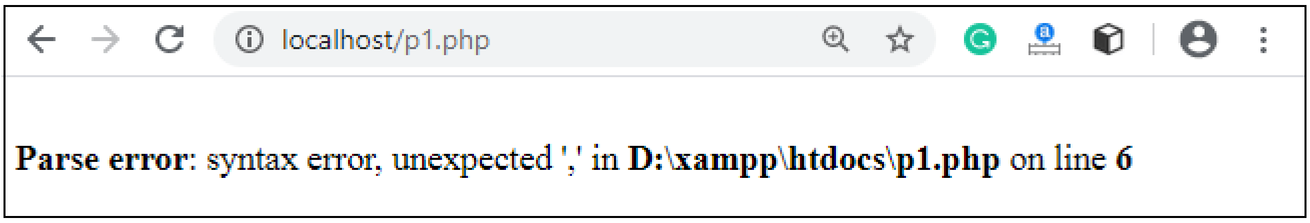


```
My name is: GunjanGarg
```

It will generate a **syntax error** because of multiple arguments in a print statement.

```php
<?php
    $fname = "Gunjan";
    $lname = "Garg";
    print "My name is: ".$fname,$lname;
?>
```

Output:



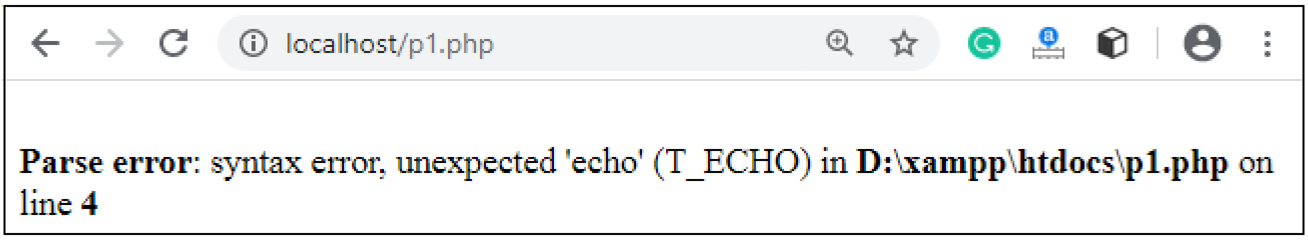
```
Parse error: syntax error, unexpected ',' in D:\xampp\htdocs\p1.php on line 6
```

For Example (Check Return Value)

echo statement does not return any value. It will generate an error if you try to display its return value.

```php
<?php
    $lang = "PHP";
    $ret = echo $lang." is a web development language.";
    echo "</br>";
    echo "Value return by print statement: ".$ret;
?>
```

Output:

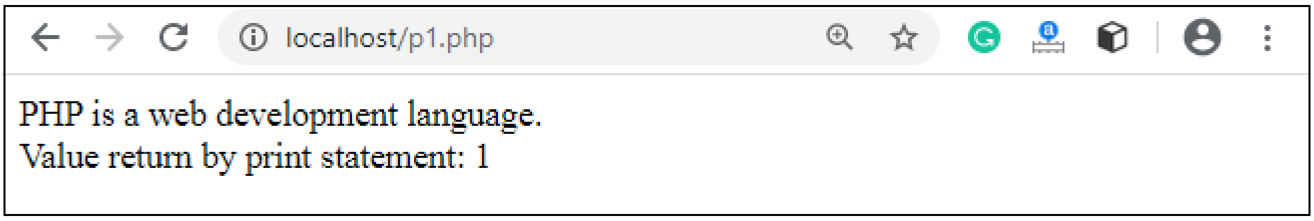


```
Parse error: syntax error, unexpected 'echo' (T_ECHO) in D:\xampp\htdocs\p1.php on
line 4
```

As we already discussed that print returns a value, which is always 1.

```php
<?php
    $lang = "PHP";
    $ret = print $lang." is a web development language.";
    print "</br>";
print "Value return by print statement: ".$ret;
?>
```

Output:



```
PHP is a web development language.
Value return by print statement: 1
```

PHP Variables

In PHP, a variable is declared using a **$ sign** followed by the variable name. Here, some important points to know about variables:

- As PHP is a loosely typed language, so we do not need to declare the data types of the variables. It automatically analyzes the values and makes conversions to its correct datatype.
- After declaring a variable, it can be reused throughout the code.
- Assignment Operator (=) is used to assign the value to a variable.

Syntax of declaring a variable in PHP is given below:

```
$variablename=value;
```

Rules for declaring PHP variable:

- A variable must start with a dollar ($) sign, followed by the variable name.
- It can only contain alpha-numeric character and underscore (A-z, 0-9, _).
- A variable name must start with a letter or underscore (_) character.
- A PHP variable name cannot contain spaces.
- One thing to be kept in mind that the variable name cannot start with a number or special symbols.
- PHP variables are case-sensitive, so $name and $NAME both are treated as different variable.

PHP Variable: Declaring string, integer, and float

Let's see the example to store string, integer, and float values in PHP variables.

File: variable1.php

```
<?php
$str="hello string";
$x=200;
$y=44.6;
echo "string is: $str <br/>";
echo "integer is: $x <br/>";
```

```
7.  echo "float is: $y <br/>";
8.  ?>
```

Output:

```
string is: hello string

integer is: 200

float is: 44.6
```

PHP Variable: Sum of two variables

File: variable2.php

```
1.  <?php
2.  $x=5;
3.  $y=6;
4.  $z=$x+$y;
5.  echo $z;
6.  ?>
```

Output:

```
11
```

PHP Variable: case sensitive

In PHP, variable names are case sensitive. So variable name "color" is different from Color, COLOR, COLor etc.

File: variable3.php

```
1.  <?php
2.  $color="red";
3.  echo "My car is " . $color . "<br>";
4.  echo "My house is " . $COLOR . "<br>";
5.  echo "My boat is " . $coLOR . "<br>";
6.  ?>
```

Output:

```
My car is red

Notice: Undefined variable: COLOR in C:\wamp\www\variable.php on line 4

My house is

Notice: Undefined variable: coLOR in C:\wamp\www\variable.php on line 5

My boat is
```

PHP Variable: Rules

PHP variables must start with letter or underscore only.

PHP variable can't be start with numbers and special symbols.

File: variablevalid.php

```php
<?php
$a="hello";//letter (valid)
$_b="hello";//underscore (valid)

echo "$a <br/> $_b";
?>
```

Output:

```
hello
hello
```

File: variableinvalid.php

```php
<?php
$4c="hello";//number (invalid)
$*d="hello";//special symbol (invalid)

echo "$4c <br/> $*d";
?>
```

Output:

```
Parse error: syntax error, unexpected '4' (T_LNUMBER), expecting variable (T_VARIABLE)
 or '$' in C:\wamp\www\variableinvalid.php on line 2
```

PHP: Loosely typed language

PHP is a loosely typed language, it means PHP automatically converts the variable to its correct data type.

PHP Variable Scope

The scope of a variable is defined as its range in the program under which it can be accessed. In other words, "The scope of a variable is the portion of the program within which it is defined and can be accessed."

PHP has three types of variable scopes:

1. Local variable
2. Global variable
3. Static variable

Local variable

The variables that are declared within a function are called local variables for that function. These local variables have their scope only in that particular function in which they are declared. This means that these variables cannot be accessed outside the function, as they have local scope.

A variable declaration outside the function with the same name is completely different from the variable declared inside the function. Let's understand the local variables with the help of an example:

File: local_variable1.php

```php
<?php
    function local_var()
    {
        $num = 45;  //local variable
        echo "Local variable declared inside the function is: ". $num;
    }
    local_var();
?>
```

Output:

```
Local variable declared inside the function is: 45
```

File: local_variable2.php

```php
<?php
    function mytest()
    {
        $lang = "PHP";
        echo "Web development language: " .$lang;
    }
    mytest();
    //using $lang (local variable) outside the function will generate an error
    echo $lang;
?>
```

Output:

```
Web development language: PHP
Notice: Undefined variable: lang in D:\xampp\htdocs\program\p3.php on line 28
```

Global variable

The global variables are the variables that are declared outside the function. These variables can be accessed anywhere in the program. To access the global variable within a function, use the GLOBAL keyword before the variable. However, these variables can be directly accessed or used outside the function without any keyword. Therefore, there is no need to use any keyword to access a global variable outside the function. Let's understand the global variables with the help of an example:

Example:

File: global_variable1.php

```php
<?php
   $name = "Sanaya Sharma";       //Global Variable
   function global_var()
   {
      global $name;
      echo "Variable inside the function: ". $name;
      echo "</br>";
   }
   global_var();
   echo "Variable outside the function: ". $name;
?>
```

Output:

```
Variable inside the function: Sanaya Sharma
Variable outside the function: Sanaya Sharma
```

Note: Without using the global keyword, if you try to access a global variable inside the function, it will generate an error *that the variable is undefined.*

Example:

File: global_variable2.php

```php
<?php
   $name = "Sanaya Sharma";       //global variable
   function global_var()
   {
      echo "Variable inside the function: ". $name;
      echo "</br>";
   }
   global_var();
?>
```

Output:

```
Notice: Undefined variable: name in D:\xampp\htdocs\program\p3.php on line 6
```

Variable inside the function:

Using $GLOBALS instead of global

Another way to use the global variable inside the function is predefined $GLOBALS array.

Example:

File: global_variable3.php

```php
<?php
    $num1 = 5;      //global variable
    $num2 = 13;     //global variable
    function global_var()
    {
            $sum = $GLOBALS['num1'] + $GLOBALS['num2'];
            echo "Sum of global variables is: " .$sum;
    }
    global_var();
?>
```

Output:

```
Sum of global variables is: 18
```

If two variables, local and global, have the same name, then the local variable has higher priority than the global variable inside the function.

Example:

File: global_variable2.php

```php
<?php
    $x = 5;
    function mytest()
    {
        $x = 7;
        echo "value of x: " .$x;
    }
    mytest();
?>
```

Output:

```
Value of x: 7
```

Note: local variable has higher priority than the global variable.

Static variable

It is a feature of PHP to delete the variable, once it completes its execution and memory is freed. Sometimes we need to store a variable even after completion of function execution. Therefore, another important feature of variable scoping is static variable. We use the static keyword before the variable to define a variable, and this variable is called as **static variable**.

Static variables exist only in a local function, but it does not free its memory after the program execution leaves the scope. Understand it with the help of an example:

Example:

File: static_variable.php

```php
<?php
    function static_var()
    {
        static $num1 = 3;      //static variable
        $num2 = 6;          //Non-static variable
        //increment in non-static variable
        $num1++;
        //increment in static variable
        $num2++;
        echo "Static: " .$num1 ."</br>";
        echo "Non-static: " .$num2 ."</br>";
    }

//first function call
    static_var();

    //second function call
    static_var();
?>
```

Output:

```
Static: 4
Non-static: 7
Static: 5
Non-static: 7
```

You have to notice that $num1 regularly increments after each function call, whereas $num2 does not. This is why because $num1 is not a static variable, so it freed its memory after the execution of each function call.

PHP $ and $$ Variables

The **$var** (single dollar) is a normal variable with the name var that stores any value like string, integer, float, etc.

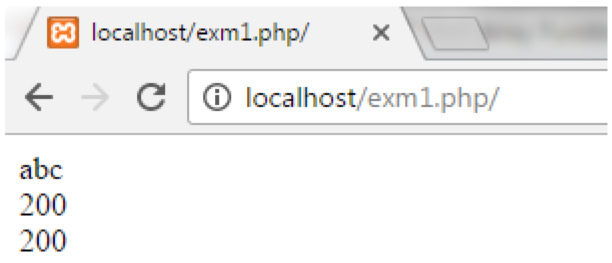
The **$$var** (double dollar) is a reference variable that stores the value of the $variable inside it.

To understand the difference better, let's see some examples.

Example 1

```
<?php
$x = "abc";
$$x = 200;
echo $x."<br/>";
echo $$x."<br/>";
echo $abc;
?>
```

Output:



```
abc
200
200
```

In the above example, we have assigned a value to the variable **x** as **abc**. Value of reference variable **$$x** is assigned as **200**.

Now we have printed the values **$x, $$x** and **$abc**.

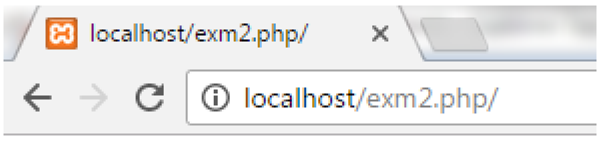
Example2

```
<?php
 $x="U.P";
$$x="Lucknow";
echo $x. "<br>";
echo $$x. "<br>";
```

6. echo "Capital of $x is " . $$x;

7. ?>

Output:




```
U.P
Lucknow
Capital of U.P is Lucknow
```

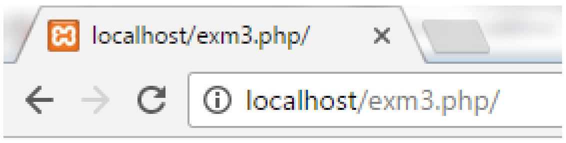
In the above example, we have assigned a value to the variable **x** as **U.P**. Value of reference variable **$$x** is assigned as **Lucknow.**

Now we have printed the values **$x, $$x** and a string.

Example3

1. <?php
2. $name="Cat";
3. ${$name}="Dog";
4. ${${$name}}="Monkey";
5. echo $name. "<br>";
6. echo ${$name}. "<br>";
7. echo $Cat. "<br>";
8. echo ${${$name}}. "<br>";
9. echo $Dog. "<br>";
10. ?>

Output:



```
Cat
Dog
Dog
Monkey
Monkey
```

In the above example, we have assigned a value to the variable name **Cat**. Value of reference variable **${$name}** is assigned as **Dog** and **${${$name}}** as **Monkey**.

Now we have printed the values as **$name, ${$name}, $Cat, ${${$name}}** and **$Dog.**