Module II

PHP(Hyper Text PreProcessor)

History

- Started as a Perl hack in 1994 by Rasmus Lerdorf.
- PHP development began in 1994 as a personal project of Rasmus Lerdorf, who had created a series of Perl scripts which he referred to as his "Personal Home Page Tools" for the maintenance of his personal web page.
- By 1997 up to PHP 3.0 with a new parser engine by Zeev Suraski and Andi Gutmans
- Version 5.2.4 is current version, rewritten by Zend to include a number of features, such as an object model
- Current is version 5
- php is one of the premier examples of what an open source project can be

PHP Introduction

- PHP is a recursive acronym for

  "PHP: Hypertext Preprocessor".
- It is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML.
- > PHP is a server-side scripting language. PHP scripts are executed on the server
- > PHP supports many databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.)
- > PHP is open source software
- > PHP is free to download and use

Client-side Environment

The client-side environment used to run scripts is usually a browser. The processing takes place on the end users computer. The source code is transferred from the web server to the users computer over the internet and run directly in the browse.

Server-side Environment

The **server-side environment** that runs a scripting language is a web server. A user's request is fulfilled by running a script directly on the web server to generate dynamic HTML pages. This

HTML is then sent to the client browser. It is usually used to provide interactive web sites that interface to databases or other data stores on the server.

Client Side vs. Server Side

When a client (your computer) makes a request for a web page that information is processed by the web server. If the request is a server side script (e.g. Perl or PHP) before the information is returned to the client the script is executed on the server and the results of the script is returned to the client.

Once the client recieves the returned information from the server if it contains a client side script (e.g. JavaScript) your computer browser executes that script before displaying the web page.

PHP Introduction

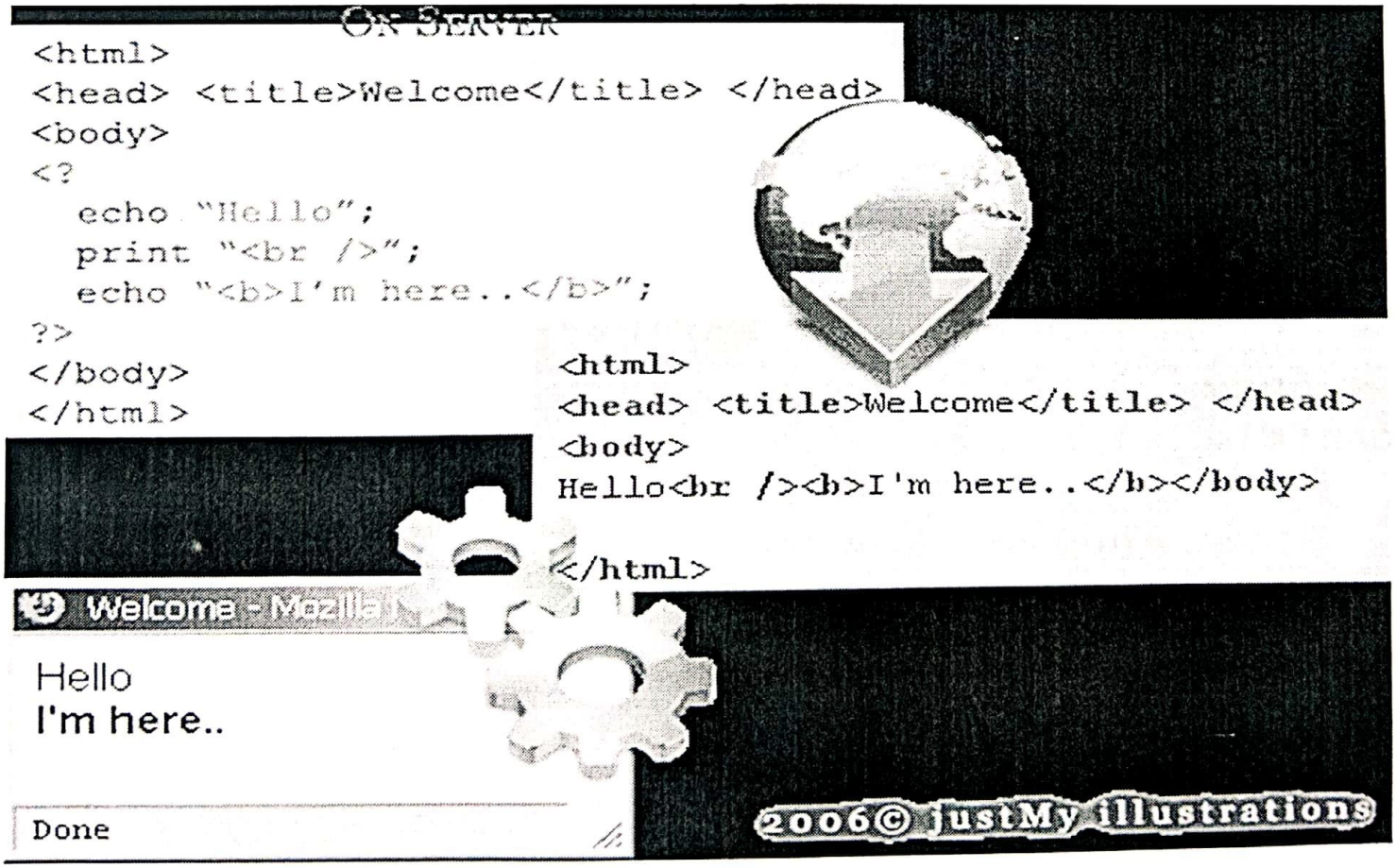
- PHP runs on different platforms (Windows, Linux, Unix, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP is FREE to download from the official PHP resource: www.php.net
- P is easy to learn and runs efficiently on the server side
- PHThe PHP code is enclosed in special start and end processing instructions **<?php** and **?>** that allow you to jump into and out of "PHP mode."

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <title>Example</title>
    </head>
    <body>

        <?php
            echo "Hi, I'm a PHP script!";
        ?>

    </body>
</html>
```

- PHP code is executed on the server, generating HTML which is then sent to the client. The client would receive the results of running that script, but would not know what the underlying code was.



PHP Getting Started

- On windows, you can download and install WAMP. With one installation and you get an Apache webserver, database server and php.

  http://www.wampserver.com

- On mac, you can download and install MAMP.

  http://www.mamp.info/en/index.html

WAMP is full form of software stack named **Windows**, **Apache**, **MYSQL**, **PHP**. It is variation of **LAMP** for Windows. It is used for testing web pages without publishing them live on the internet. In short it is used for **off-line preview of web pages** so developer can easily get idea to how their webpage looks on real web browser.

Wamp, or wampserver, is a software that allows you to use your computer as a local server, and "host" websites on it. It is a very useful tool to check how your website would work on a dedicated server, especially for back-end aspects, like PHP and MySQL

PHP Hello World

```
<html>
 <head>
  <title>PHP Test</title>
 </head>
 <body>
 <?php echo '<p>Hello World</p>'; ?>
 </body>
</html>
```

- It renders as HTML that looks like this:

```
<html>
 <head>
  <title>PHP Test</title>
 </head>
 <body>
 <p>Hello World</p>
 </body>
</html>
```

- This program is extremely simple and you really did not need to use PHP to create a page like this. All it does is display: Hello World using the PHP **echo()** statement.

PHP Echo

PHP echo is a language construct not a function, so you don't need to use parenthesis with it. But if you want to use more than one parameters, it is required to use parenthesis.

The syntax of PHP echo is given below:

void echo (string $arg1 [, string $...])

<?php

echo "Hello by PHP echo";

?>

Output

Hello by PHP echo

<?php

$msg="Hello JavaTpoint PHP";

echo "Message is: $msg";

?>

Output:

Message is: Hello JavaTpoint PHP

```
1 <?php
2 echo "This is echo demo with double quotes</BR>";
3 echo ("This is echo demo with double quote in parenthesis</BR>");
4 echo 'This is echo demo in single quote';
5 ?>
6
```

```
1 <?php
2 $var_echo = "echo demo";
3 echo "This is $var_echo with double quotes</BR>";
4 echo ("This is $var_echo with double quote in parenthesis</BR>");
5 echo 'This is $var_echo in single quote';
6 ?>
7
```

```
<?php
$txt1 = "Learn PHP";
$txt2 = "W3Schools.com";
$x = 5;
$y = 4;

echo "<h2>$txt1</h2>";
echo "Study PHP at $txt2<br>";
echo $x + $y;
?>
```

Output

Study PHP at W3Schools.com
9

PHP Print

Like PHP echo, PHP print is a language construct, so you don't need to use parenthesis with the argument list. Unlike echo, it always returns 1.

The syntax of PHP print is given below:

int print(string $arg)

```
<?php

print "Hello by PHP print ";

print ("Hello by PHP print()");

?>
```

Output:

Hello by PHP print Hello by PHP print()

```
<?php

print "Hello by PHP print

this is multi line

text printed by

PHP print statement

";
```

?>

Output:

Hello by PHP print this is multi line text printed by PHP print statement

while(print('still in the loop'))

<?php

$msg="Hello print() in PHP";

print "Message is: $msg";

?>

Output:

Message is: Hello print() in PHP

PHP Comments

- In PHP, we use // to make a single-line comment or /* and */ to make a large comment block.

```
<html>
<body>

<?php
//This is a comment

/*
This is
a comment
block
*/
?>

</body>
</html>
```

PHP Variables

- > Variables are used for storing values, like text strings, numbers or arrays.
- > When a variable is declared, it can be used over and over again in your script.
- > All variables in PHP start with a $ sign symbol.
- > The correct way of declaring a variable in PHP:

```
$var_name = value;
```

```php
<?php
$txt="Hello World!";
$x=16;
?>
```

- > In PHP, a variable does not need to be declared before adding a value to it.
- > In the example above, you see that you do not have to tell PHP which data type the variable is.
- > PHP automatically converts the variable to the correct data type, depending on its value.
- > A variable name must start with a letter or an underscore "_" -- not a number
- > A variable name can only contain alpha-numeric characters, underscores (a-z, A-Z, 0-9, and _)
- > A variable name should not contain spaces. If a variable name is more than one word, it should be separated with an underscore ($my_string) or with capitalization ($myString)

PHP Concatenation

- > The concatenation operator (.) is used to put two string values together.
- > To concatenate two string variables together, use the concatenation operator:

```php
<?php
$txt1="Hello World!";
$txt2="What a nice day!";
echo $txt1 . " " . $txt2;
?>
```

- The output of the code on the last slide will be:

```
Hello World! What a nice day!
```

- If we look at the code you see that we used the concatenation operator two times. This is because we had to insert a third string (a space character), to separate the two strings.

PHP Operators

- Operators are used to operate on values. There are four classifications of operators:
- Arithmetic
- Assignment
- Comparison
- Logical

Arithmetic Operators

Operator	Description	Example	Result
+	Addition	x=2 x+2	4
-	Subtraction	x=2 5-x	3
*	Multiplication	x=4 x*5	20
/	Division	15/5 5/2	3 2.5
%	Modulus (division remainder)	5%2 10%8 10%2	1 2 0
++	Increment	x=5 x++	x=6
--	Decrement	x=5 x--	x=4

Assignment Operators

Operator	Example	Is The Same As
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
.=	x.=y	x=x.y
%=	x%=y	x=x%y

Comparison Operators

Operator	Description	Example
==	is equal to	5==8 returns false
!=	is not equal	5!=8 returns true
<>	is not equal	5<>8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than or equal to	5>=8 returns false
<=	is less than or equal to	5<=8 returns true

Logical Operators

Operator	Description	Example
&&	and	x=6 y=3 (x < 10 && y > 1) returns true
||	or	x=6 y=3 (x==5 || y==5) returns false
!	not	x=6 y=3 !(x==y) returns true

PHP Constants

PHP constants are name or identifier that can't be changed during the execution of the script. PHP constants can be defined by 2 ways:

1. Using define() function
2. Using const keyword

PHP constants follow the same PHP variable rules. For example, it can be started with letter or underscore only.

Conventionally, PHP constants should be defined in uppercase letters.

PHP constant: define()

define(name, value, case-insensitive)

name: specifies the constant name

value: specifies the constant value

case-insensitive: Default value is false. It means it is case sensitive by default.

```
<?php
define("MESSAGE","Hello  PHP");
echo MESSAGE;
?>
```

Output

Hello PHP

```
<?php
define("MESSAGE","Hello PHP",true);//not case sensitive
echo MESSAGE;
echo message;
?>
```

Output

Hello PHP

PHP constant: const keyword

The const keyword defines constants at compile time. It is a language construct not a function.

It is bit faster than define().

It is always case sensitive.

```
<?php
const MESSAGE="Hello  PHP";
echo MESSAGE;
?>
```

Output

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:

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

Output

Hello world!
Hello world!

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

```
<?php
$x = 5985;
var_dump($x);
?>
```

Output

int(5985)

The PHP var_dump() function returns the data type and value:

PHP Float

A float (floating point number) is a number with a decimal point .

```
<?php
$x = 10.365;
var_dump($x);
?>
```

Output

float(10.365)

PHP Boolean

A Boolean represents two possible states: TRUE or FALSE.

```
$x = true;
$y = false;
```

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.

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

Variables can also be emptied by setting the value to NULL:

```
<?php
$x = "Hello world!";
$x = null;
var_dump($x);
?>
```

Output

NULL

PHP Resource

The special resource type is not an actual data type. It is the storing of a reference to functions and resources external to PHP.

A common example of using the resource data type is a database call.

PHP Conditional Statements

- **if** statement - use this statement to execute some code only if a specified condition is true
- > **if...else** statement - use this statement to execute some code if a condition is true and another code if the condition is false
- > **if...elseif....else** statement - use this statement to select one of several blocks of code to be executed
- > **switch** statement - use this statement to select one of many blocks of code to be executed

Conditional Statements

PHP If Else

PHP if else statement is used to test condition. There are various ways to use if statement in PHP.

if

if-else

if-else-if

nested if

PHP If Statement

PHP if statement is executed if condition is true.

Syntax

```
if(condition){
//code to be executed
```

```
}
<?php
$num=12;
if($num<100){
echo "$num is less than 100";
}
?>
```

Output

12 is less than 100

PHP If-else Statement

PHP if-else statement is executed whether condition is true or false.

Syntax

```
if(condition){
//code to be executed if true
}else{
//code to be executed if false
}
```

```
<?php
$num=12;
if($num%2==0){
echo "$num is even number";
}else{
echo "$num is odd number";
}
?>
```

Output:

12 is even number

d - The day of the month (from 01 to 31)

D - A textual representation of a day (three letters)

j - The day of the month without leading zeros (1 to 31)

l (lowercase 'L') - A full textual representation of a day

w - A numeric representation of the day (0 for Sunday, 6 for Saturday)

w - A numeric representation of the day (0 for Sunday, 6 for Saturday)

Y - A four digit representation of a year

y - A two digit representation of a year

PHP Conditional Statements

- Use the **if....else** statement to execute some code if a condition is true and another code if a condition is false.

```
<html>
<body>

<?php
$d=date("D");
if ($d=="Fri")
  echo "Have a nice weekend!";
else
  echo "Have a nice day!";
?>

</body>
</html>
```

PHP Conditional Statements

- If more than one line should be executed if a condition is true/false, the lines should be enclosed within curly braces { }

```
<html>
<body>

<?php
$d=date("D");
if ($d=="Fri")
  {
  echo "Hello!<br />";
  echo "Have a nice weekend!";
  echo "See you on Monday!";
  }
?>

</body>
</html>
```

- The following example will output "Have a nice weekend!" if the current day is Friday, and "Have a nice Sunday!" if the current day is Sunday. Otherwise it will output "Have a nice day!":

```
<html>
<body>

<?php
$d=date("D");
if ($d=="Fri")
  echo "Have a nice weekend!";
elseif ($d=="Sun")
  echo "Have a nice Sunday!";
else
  echo "Have a nice day!";
?>

</body>
</html>
```

PHP Switch

PHP switch statement is used to execute one statement from multiple conditions. It works like PHP if-else-if statement.

Syntax

switch(expression){

case value1:

//code to be executed

break;

case value2:

//code to be executed

break;

.....

default:

code to be executed **if** all cases are not matched;

}

<?php

$num=20;

switch($num){

case 10:

```
echo("number is equals to 10");
break;
case 20:
echo("number is equal to 20");
break;
case 30:
echo("number is equal to 30");
break;
default:
echo("number is not equal to 10, 20 or 30");
}
?>
```

Output:

number is equal to 20

PHP Conditional Statements

- For switches, first we have a single expression n (most often a variable), that is evaluated once.
- The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed.
- Use break to prevent the code from running into the next case automatically. The default statement is used if no match is found.

PHP Iteration Statements(Looping)

In PHP, we have the following looping statements:

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

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

Iteration Statements

PHP For Loop

PHP for loop can be used to traverse set of code for the specified number of times.

It should be used if number of iteration is known otherwise use while loop.

Syntax

```
for(initialization; condition; increment/decrement){
//code to be executed
}
```

```
<?php
for($n=1;$n<=10;$n++){
echo "$n<br/>";
}
?>
```

Output:

1

2

3

4

5

6

7

8

9

10

The PHP while Loop

The while loop executes a block of code as long as the specified condition is true.

Syntax

```
    code to be executed;
}
```

```
<?php
$x = 1;

while($x <= 5) {
    echo "The number is: $x <br>";
    $x++;
}
?>
```

Output

```
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
```

PHP Loops

- > Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal lines in a script we can use loops to perform a task like this.
- > In PHP, we have the following looping statements:
- > **while** - loops through a block of code while a specified condition is true

- > **do...while** - loops through a block of code once, and then repeats the loop as long as a specified condition is true
- > **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

PHP Loops – While

- The while loop executes a block of code while a condition is true. The example below defines a loop that starts with
- i=1. The loop will
- continue to run as
- long as i is less
- than, or equal to 5.
- i will increase by 1
- each time the loop
- runs:

```
<html>
<body>

<?php
$i=1;
while($i<=5)
  {
  echo "The number is " . $i . "<br />";
  $i--;
  }
?>

</body>
</html>
```

Output:

```
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
```

PHP Loops – Do ... While

- The do...while statement will always execute the block of code once, it will then check the condition, and repeat the loop while the condition is true.
- The next example defines a loop that starts with i=1. It will then increment i with 1, and write some output. Then the condition is checked, and the loop will continue to run as long as i is less than, or equal to 5:

PHP Loops – Do ... While

```
<html>
<body>

<?php
$i=1;
do
  {
  $i++;
  echo "The number is " . $i . "<br />";
  }
while ($i<=5);
?>

</body>
</html>
```

Output:

```
The number is 2
The number is 3
The number is 4
The number is 5
The number is 6
```

PHP Loops – For

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

Syntax

```
for (init; condition; increment)
  {
  code to be executed;
  }
```

- Parameters:
- > **init**: Mostly used to set a counter (but can be any code to be executed once at the beginning of the loop)
- > **condition**: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
- > **increment**: Mostly used to increment a counter (but can be any code to be executed at the end of the loop)
- The example below defines a loop that starts with i=1. The loop will continue to run as long as i is less than, or equal to 5. i will increase by 1 each time the loop runs:

```
<html>
<body>

<?php
for (Si=1; Si<=5; Si++)
  {
  echo "The number is " . Si . "<br />";
  }
?>

</body>
</html>
```

Output:

```
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
```

PHP Loops – Foreach

```
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) - so on the next loop iteration, you'll be looking at the next array value.
- The following example demonstrates a loop that will print the values of the given array:

```
<html>
<body>

<?php
$x=array("one","two","three");
foreach ($x as $value)
   {
   echo $value . "<br />";
   }
?>

</body>
</html>
```

- Winner of the most impressive slide award

Output:

```
one
two
three
```

PHP Functions

- > We will now explore how to create your own functions.
- > To keep the script from being executed when the page loads, you can put it into a function.
- > A function will be executed by a call to the function.
- > You may call a function from anywhere within a page.
- A function will be executed by a call to the function.

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

- Give the function a name that reflects what the function does
- > The function name can start with a letter or underscore (not a number)

```
<?php
function sayHello(){
echo "Hello PHP Function";
}
sayHello();//calling function
?>
```

Output:

Hello PHP Function

PHP Function Arguments

We can pass the information in PHP function through arguments which is separated by comma.

PHP supports **Call by Value** (default), **Call by Reference**, **Default argument values** and **Variable-length argument list.**

```
<?php
function sayHello($name){
echo "Hello $name<br/>";
}
sayHello("Sonoo");
sayHello("Vimal");
sayHello("John");
?>
```

Output:

```
Hello Sonoo
Hello Vimal
Hello John
```

```
<?php
function sayHello($name,$age){
echo "Hello $name, you are $age years old<br/>";
}
sayHello("Sonoo",27);
sayHello("Vimal",29);
sayHello("John",23);
?>
```

Output

```
Hello Sonoo, you are 27 years old
Hello Vimal, you are 29 years old
Hello John, you are 23 years old
```

PHP Call By Reference

Value passed to the function doesn't modify the actual value by default (call by value). But we can do so by passing value as a reference.

By default, value passed to the function is call by value. To pass value as a reference, you need to use ampersand (&) symbol before the argument name

```
<?php
function adder(&$str2)
{
  $str2 .= 'Call By Reference';
}
$str = 'Hello ';
adder($str);
echo $str;
?>
```

Output

Hello Call By Reference

PHP Function: Default Argument Value

We can specify a default argument value in function. While calling PHP function if you don't specify any argument, it will take the default argument.

```
<?php
function sayHello($name="Sonoo"){
echo "Hello $name<br/>";
}
sayHello("Rajesh");
sayHello();//passing no value
sayHello("John");
?>
```

Output

Hello Rajesh

Hello Sonoo

Hello John

PHP Function: Returning Value

```
<?php
function cube($n){
return $n*$n*$n;
}
echo "Cube of 3 is: ".cube(3);
?>
```

Output

Cube of 3 is: 27

PHP Variable Length Argument Function

PHP supports variable length argument function. It means you can pass 0, 1 or n number of arguments in function. To do so, you need to use 3 ellipses (dots) before the argument name.

The 3 dot concept is implemented for variable length argument since PHP 5.6.

```
<?php
function add(...$numbers) {
  $sum = 0;
  foreach ($numbers as $n) {
    $sum += $n;
  }
  return $sum;
}

echo add(1, 2, 3, 4);
?>
```

Output

10

```
    display($number+1);
  }
}

display(1);
?>
```

Output

```
1
2
3
4
5
```

PHP Recursive Function

PHP also supports recursive function call like C/C++. In such case, we call current function within function. It is also known as recursion.

```
<?php
function display($number) {
   if($number<=5){
    echo "$number <br/>";
    display($number+1);
   }
}

display(1);
?>
```

Output

```
1
2
3
4
5
```