WEB DEVLOPMENT(3151606)

UNIT-4:PHP

PREPARED BY: PROF. MAYUR PRAJAPATI

What is PHP ?

- PHP stands for **P**HP: **H**ypertext **P**reprocessor
- PHP is a server-side scripting language
- PHP scripts are executed on the server
- PHP supports many databases (MySQL, Oracle, SQL, Generic ODBC, etc.)
- PHP is an open source software
- PHP is free to download and use

What is a PHP File ?

- PHP files can contain text, HTML tags and scripts
- PHP files are returned to the browser as plain HTML
- PHP files have a file extension of ".php", ".php3", or ".phtml"

What is MySQL ?

- MySQL is a Database Server
- MySQL is ideal for both small and large applications
- MySQL supports standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use

Why PHP ?

- 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

  (Version 7.4.3)
- PHP is easy to learn and runs efficiently on the server side.

PHP advantages

- PHP can generate dynamic page content
- PHP can create, open, read, write, delete and close files on the server
- PHP can collect form data
- PHP can send and receive cookies
- PHP can add, delete, modify data in your database
- PHP can be used to control user-access
- PHP can encrypt data

Install LAMP

- **LAMP Server** is a collection of open source software used to create a web server.
- The collection consists of:

1. **Linux** – the operating system
2. **Apache server** – the server
3. **MySQL** – the database system
4. **PHP** – the programming language

Install LAMP

- First refresh your package index...

  sudo apt-get update

- ... and then install the LAMP stack:

  sudo apt-get install lamp-server^

- create file through command prompt using **gedit.**
- save your php file in **.var/www/html.**
- sudo chmod -R 777 /var/www

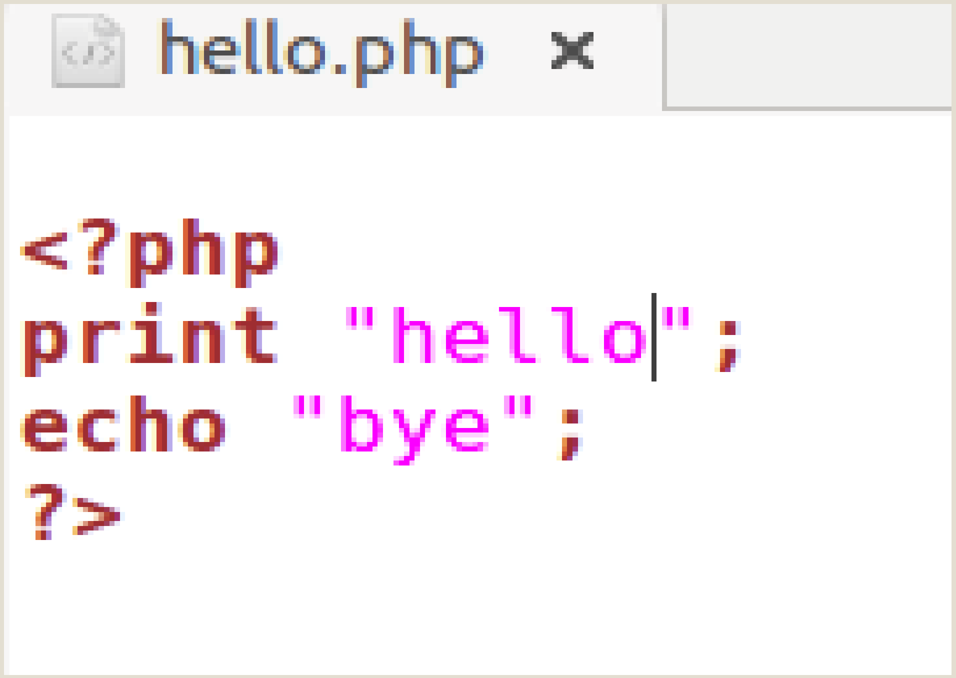
WAMP

- **WAMP Server** is a software stack for Microsoft windows created by Romain Bourdon.

1. **Windows** – the operating system
2. **Apache server** – the server
3. **MySQL** – the database system
4. **PHP** – the programming language

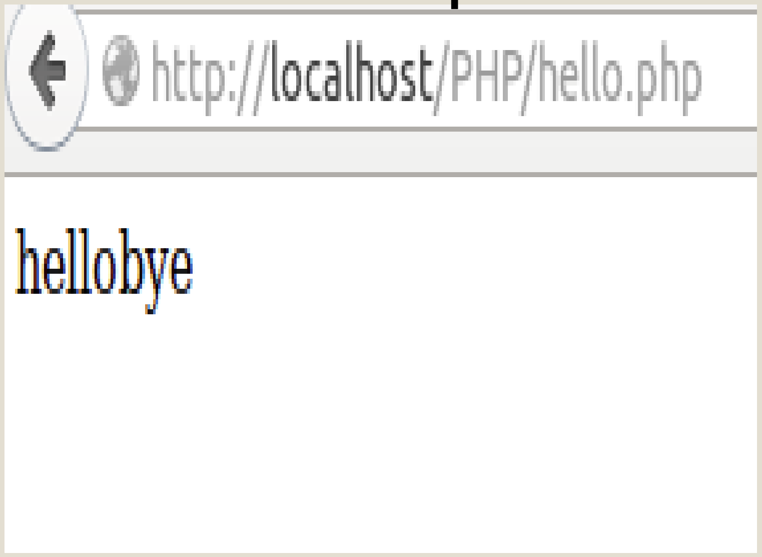
How to write simple PHP script ?

- PHP code must embedded with <? php ?> tag.
- Using **print** and **echo** statement the O/P can be displayed on the screen.
- Example : **hello.php**



```
<?php
print "hello";
echo "bye";
?>
```

Output



```
hellobye
```

Comments in PHP

- Single line comment specified by //or #.
- Multiline comment specified by /* and */.

Example:

```
 <?php
// This is a single-line comment
#  This is also a single-line comment
/* This is a multiple-lines
     comment block
*/
 ?>
```

Case sensitivity in PHP

- PHP Statement must be terminated by semicolon (;).
- In PHP, all keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are NOT case-sensitive.
- All variable names are case-sensitive.

Some Reserved Words For PHP

- All these are special words, associated with some meaning.

foreach	continue	function	or	break
false	do	global	require	true
extends	else	list	return	var
and	elseif	not	this	default
new	for	include	switch	static
while	virtual	xor		

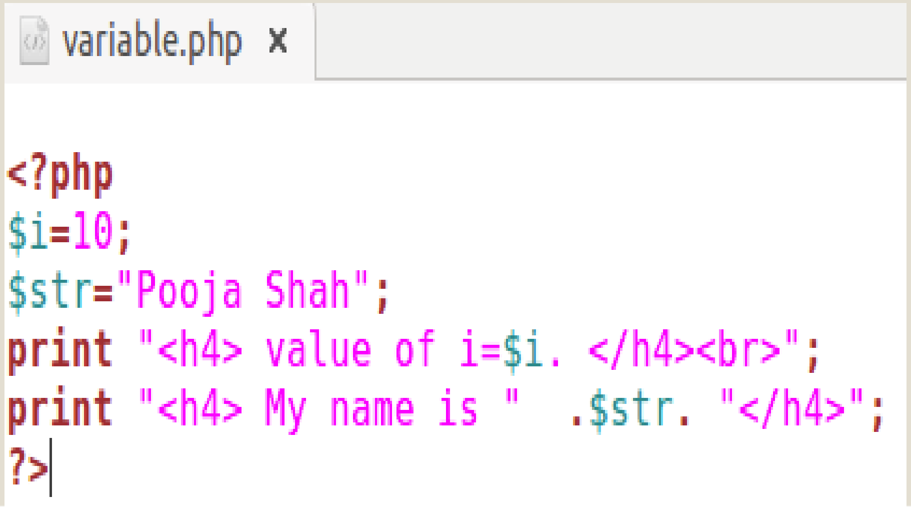
Variables in PHP

- PHP is dynamically typed language. i.e. PHP has no type declaration.
- **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)

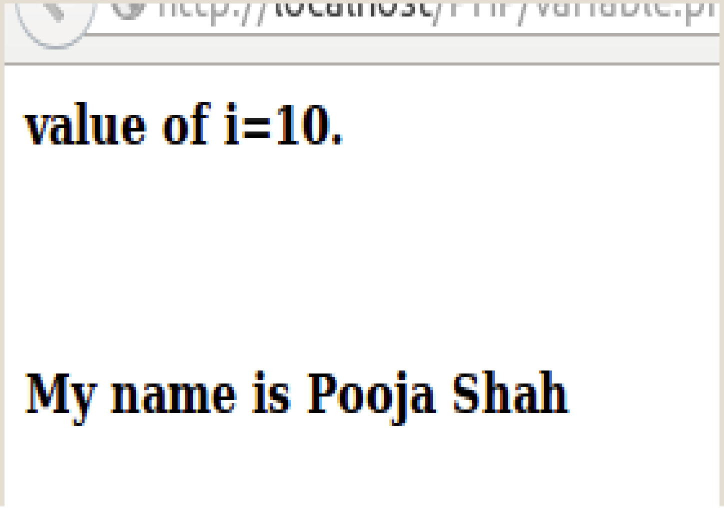
Variables in PHP

- The value can be assigned to variable by
- **$variable_name=value;**
- If value is not assigned to variable then by default the value is NULL.
- These variables are called **unbound variables**, if they are used in expression then NULL value is automatically converted to 0.



variable.php ×

```php
<?php
$i=10;
$str="Pooja Shah";
print "<h4> value of i=$i. </h4><br>";
print "<h4> My name is "  .$str. "</h4>";
?>
```

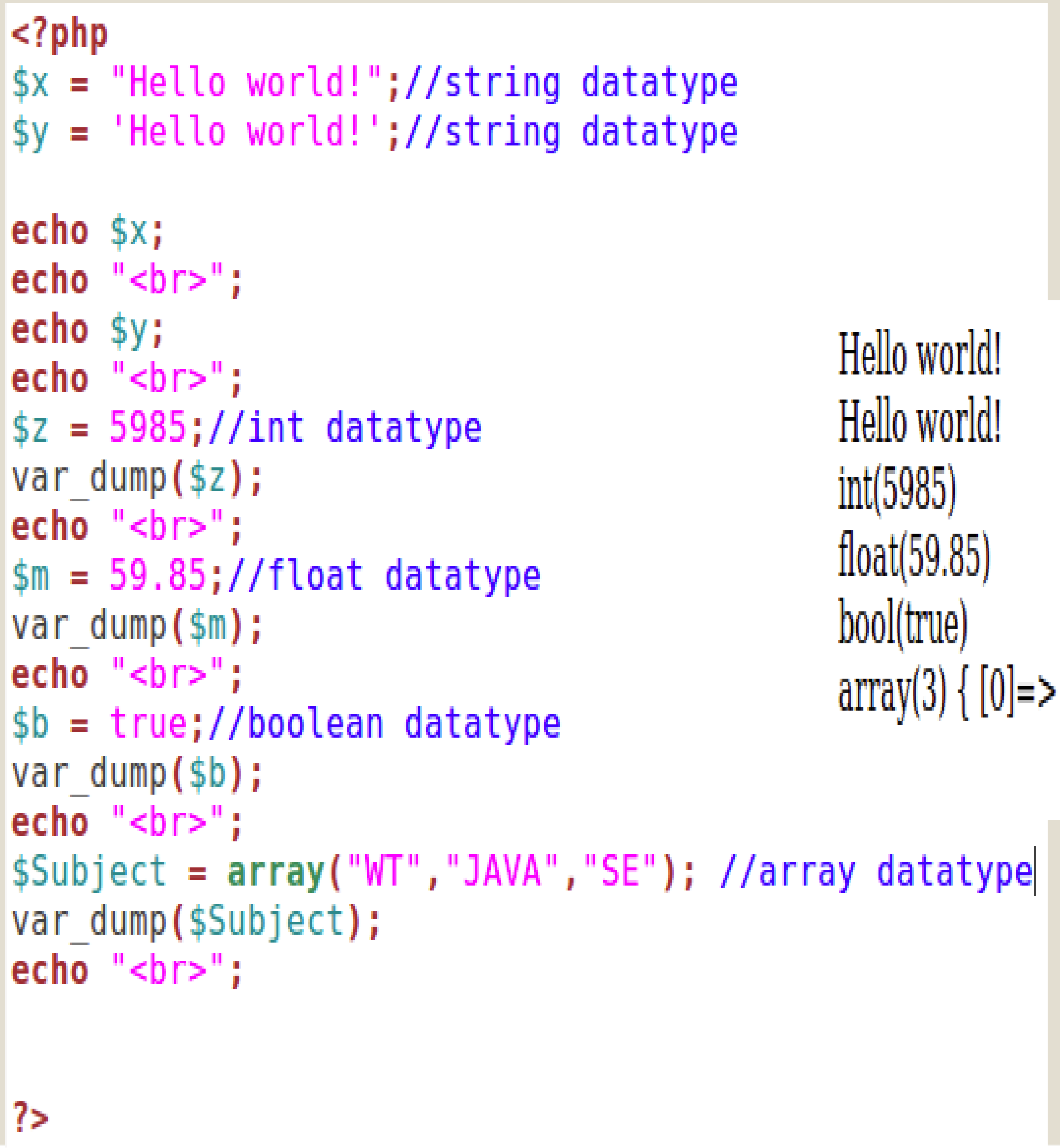


```
value of i=10.

My name is Pooja Shah
```

Data Types in PHP

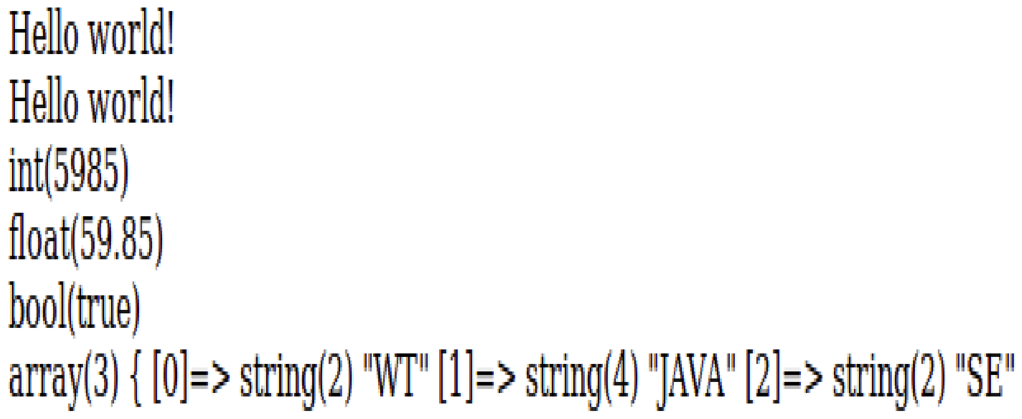
- **Integers**
- **Doubles**
- **Booleans**
- **NULL** − is a special type that only has one value: NULL.
- **Strings**
- **Arrays** − are named and indexed collections of other values.
- **Objects** − are instances of programmer-defined classes, which can package up both other kinds of values and functions that are specific to the class.
- **Resources** − are special variables that hold references to resources external to PHP (such as database connections).

datatype.php



```php
<?php
$x = "Hello world!";//string datatype
$y = 'Hello world!';//string datatype

echo $x;
echo "<br>";
echo $y;
echo "<br>";
$z = 5985;//int datatype
var_dump($z);
echo "<br>";
$m = 59.85;//float datatype
var_dump($m);
echo "<br>";
$b = true;//boolean datatype
var_dump($b);
echo "<br>";
$Subject = array("WT","JAVA","SE"); //array datatype
var_dump($Subject);
echo "<br>";



?>
```

output

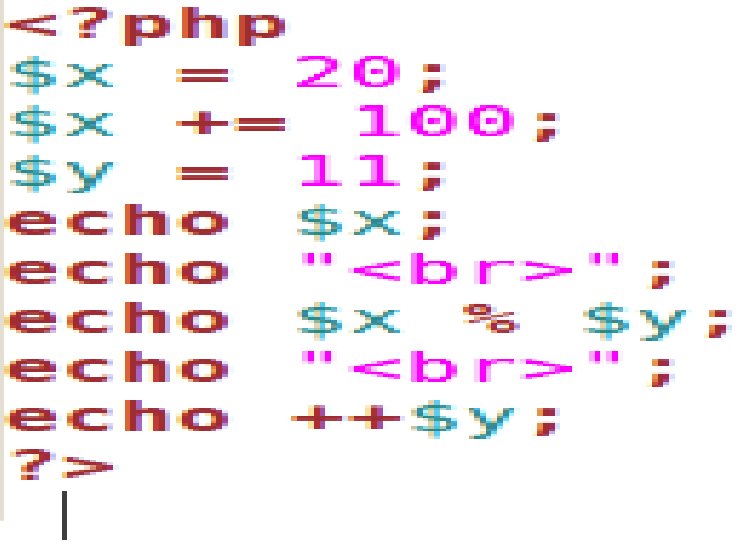


```
Hello world!
Hello world!
int(5985)
float(59.85)
bool(true)
array(3) { [0]=> string(2) "WT" [1]=> string(4) "JAVA" [2]=> string(2) "SE"
```

Operators

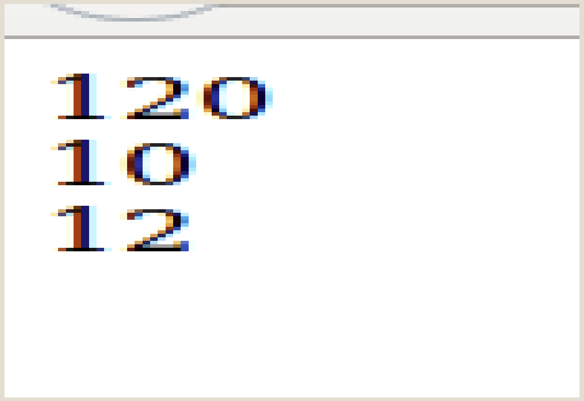
- Arithmetic : (+,-,/,*,%).
- Relational : (<,>,<=,>=,==)..
- Logical : (!= ,&&,||).
- Assignment : (=).
- Increment : (++).
- Decrement : (--).

operator.php



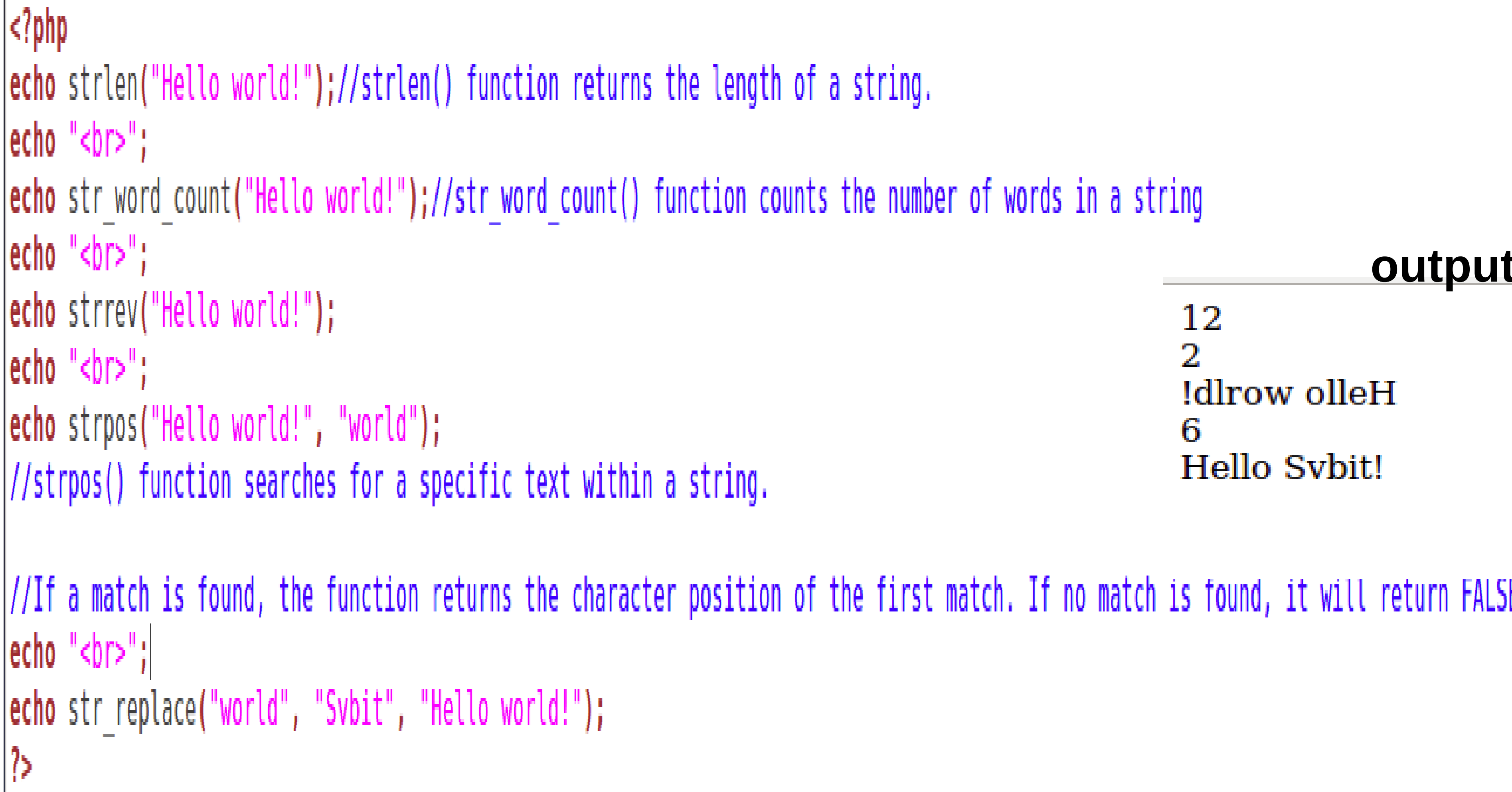
```
<?php
$x  =  20;
$x  +=  100;
$y  =  11;
echo  $x;
echo  "<br>";
echo  $x  %  $y;
echo  "<br>";
echo  ++$y;
?>
```

output



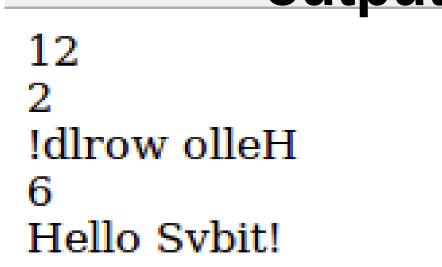
```
120
10
12
```

Various String Operations



```
<?php
echo strlen("Hello world!");//strlen() function returns the length of a string.
echo "<br>";
echo str_word_count("Hello world!");//str_word_count() function counts the number of words in a string
echo "<br>";
echo strrev("Hello world!");
echo "<br>";
echo strpos("Hello world!", "world");
//strpos() function searches for a specific text within a string.

//If a match is found, the function returns the character position of the first match. If no match is found, it will return FALS
echo "<br>";
echo str_replace("world", "Svbit", "Hello world!");
?>
```

output



```
12
2
!dlrow olleH
6
Hello Svbit!
```

2nd Example

2) Concatenating the two string:

Ans : <? Php

```
                    $str1 = “Hello”;

                    $str2 = “Friends”;

                    print $str1 “ “ $str2 ;

               ?>
```

- O/P : Hello Friends.

3rd Example

3) Comparing Two string:

```
Ans : <? Php
                $str1 = “Hello”;
                $str2 = “Friends”;
                if($strcmp($str1,$str2)==0)
                     print “Both r equal”;
                else
                     print “Both r not equal”;
            ?>
```

- O/P : Both r not equal.

4th Example

4) Reverse string:

Ans : <? Php

```
        $str1 = “Hello”;

        $str2 = “Friends”;

        print $strrev($str1);           ?>
```

- O/P : olleH.

Decision Making

- If..else..
- If.elseif..else
- switch

Cont..

- It supports if-else, while statement and for loops.

1)if-else statement will be-

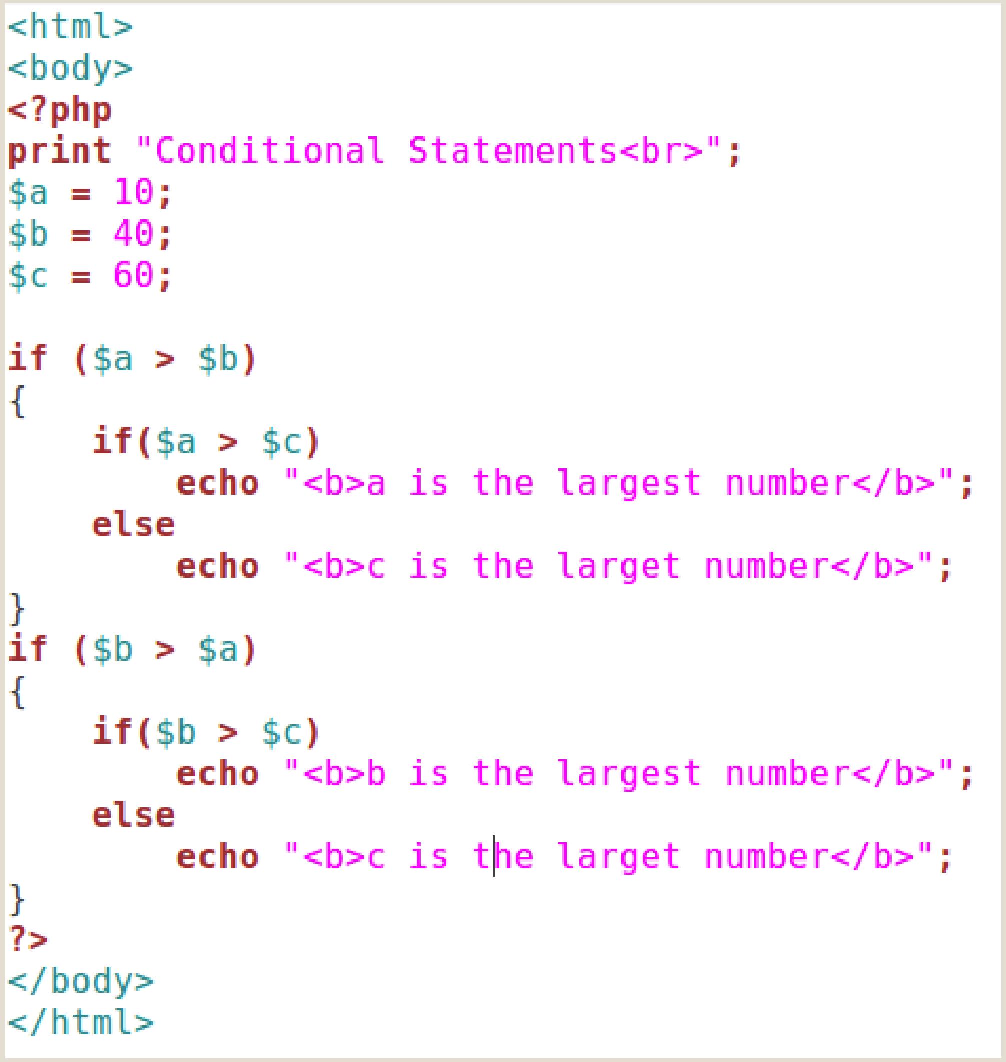
```
If ($a>$b)
    $msg= “a is greater”;
else
    $msg = “b is greater”;
```

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

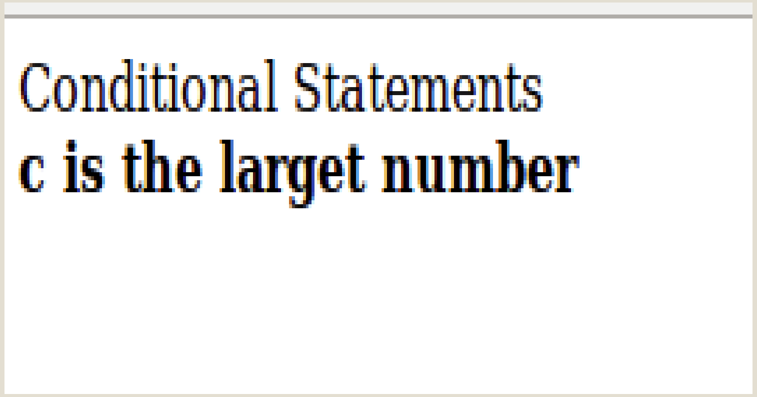
Conditional Statements Example

ifelse.php



```
<html>
<body>
<?php
print "Conditional Statements<br>";
$a = 10;
$b = 40;
$c = 60;

if ($a > $b)
{
    if($a > $c)
        echo "<b>a is the largest number</b>";
    else
        echo "<b>c is the larget number</b>";
}
if ($b > $a)
{
    if($b > $c)
        echo "<b>b is the largest number</b>";
    else
        echo "<b>c is the larget number</b>";
}
?>
</body>
</html>
```

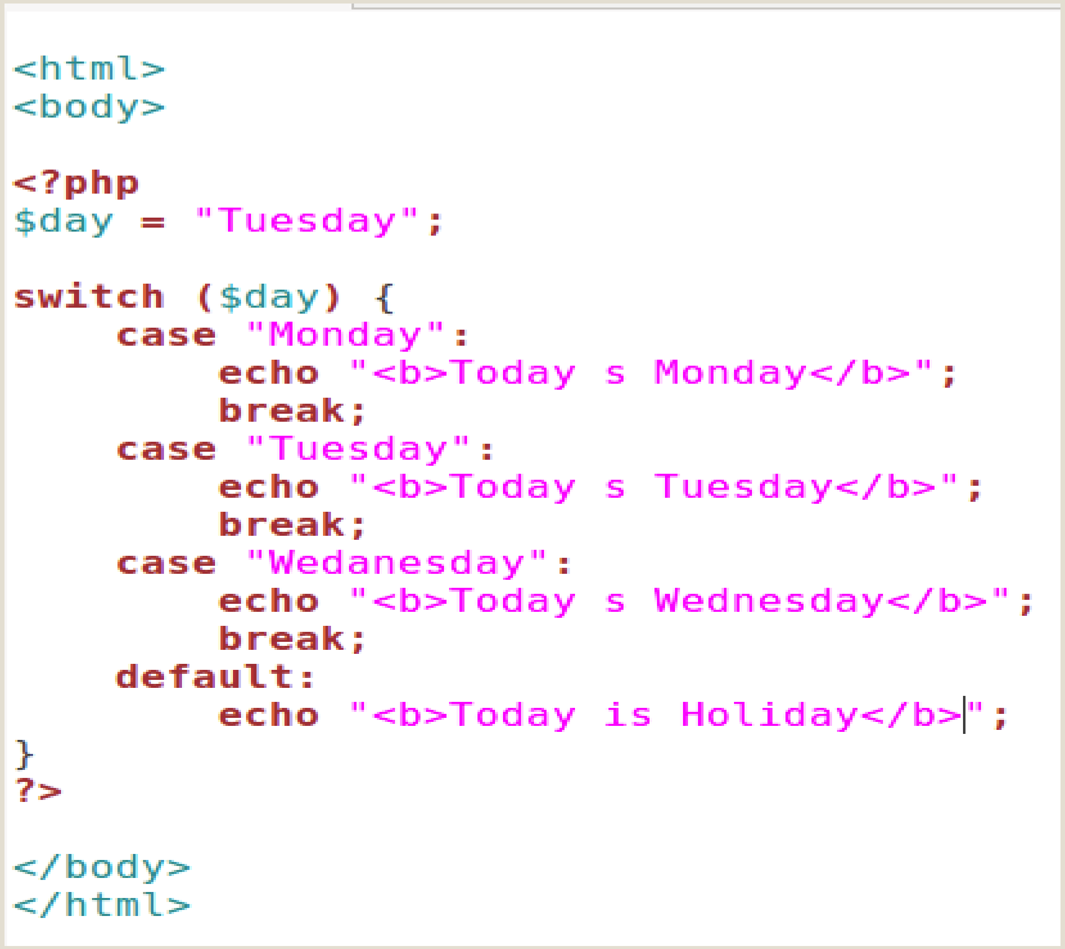
output



```
Conditional Statements
c is the larget number
```

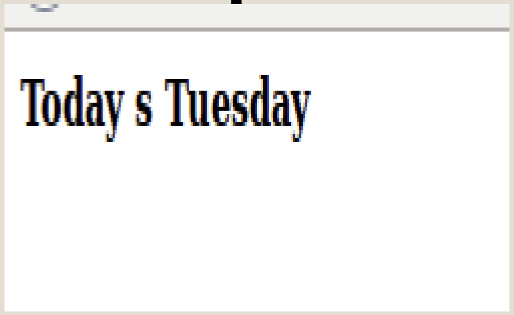
Conditional Statements Example

switch.php



```
<html>
<body>

<?php
$day = "Tuesday";

switch ($day) {
    case "Monday":
        echo "<b>Today s Monday</b>";
        break;
    case "Tuesday":
        echo "<b>Today s Tuesday</b>";
        break;
    case "Wedanesday":
        echo "<b>Today s Wednesday</b>";
        break;
    default:
        echo "<b>Today is Holiday</b>";
}
?>

</body>
</html>
```

output



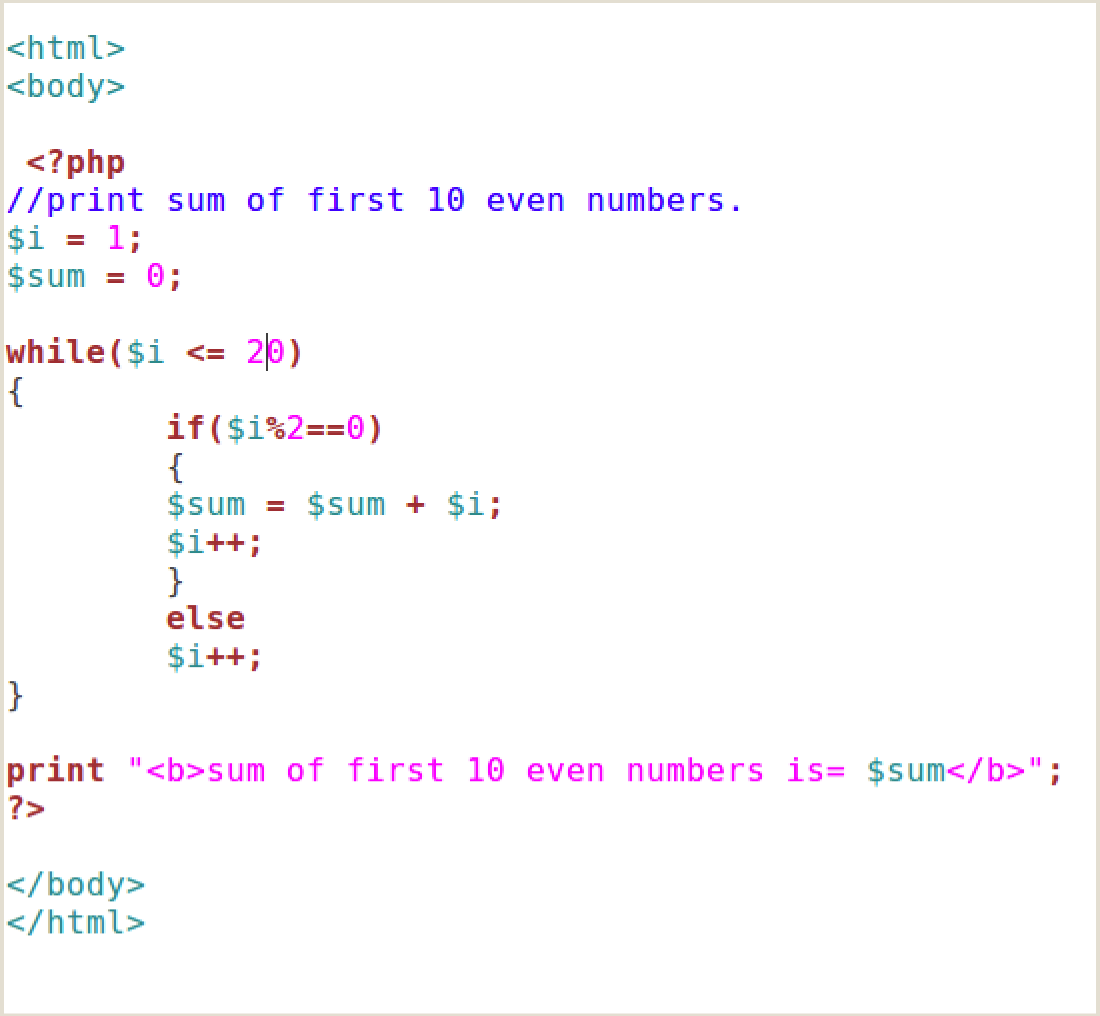
```
Today s Tuesday
```

Looping

- **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

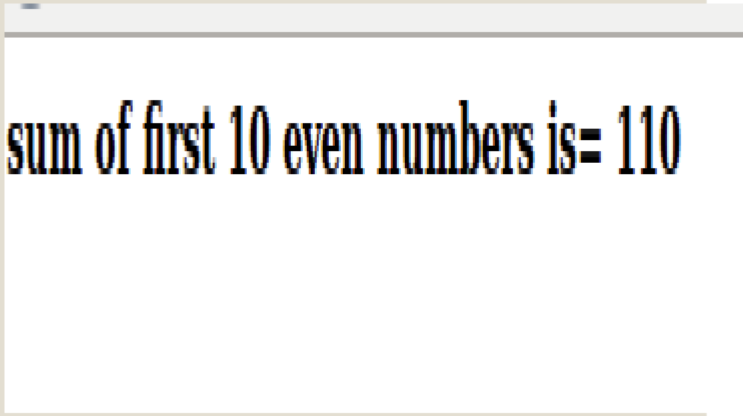
Looping Statements Example

while.php



```
<html>
<body>

 <?php
//print sum of first 10 even numbers.
$i = 1;
$sum = 0;

while($i <= 20)
{
        if($i%2==0)
        {
        $sum = $sum + $i;
        $i++;
        }
        else
        $i++;
}

print "<b>sum of first 10 even numbers is= $sum</b>";
?>

</body>
</html>
```

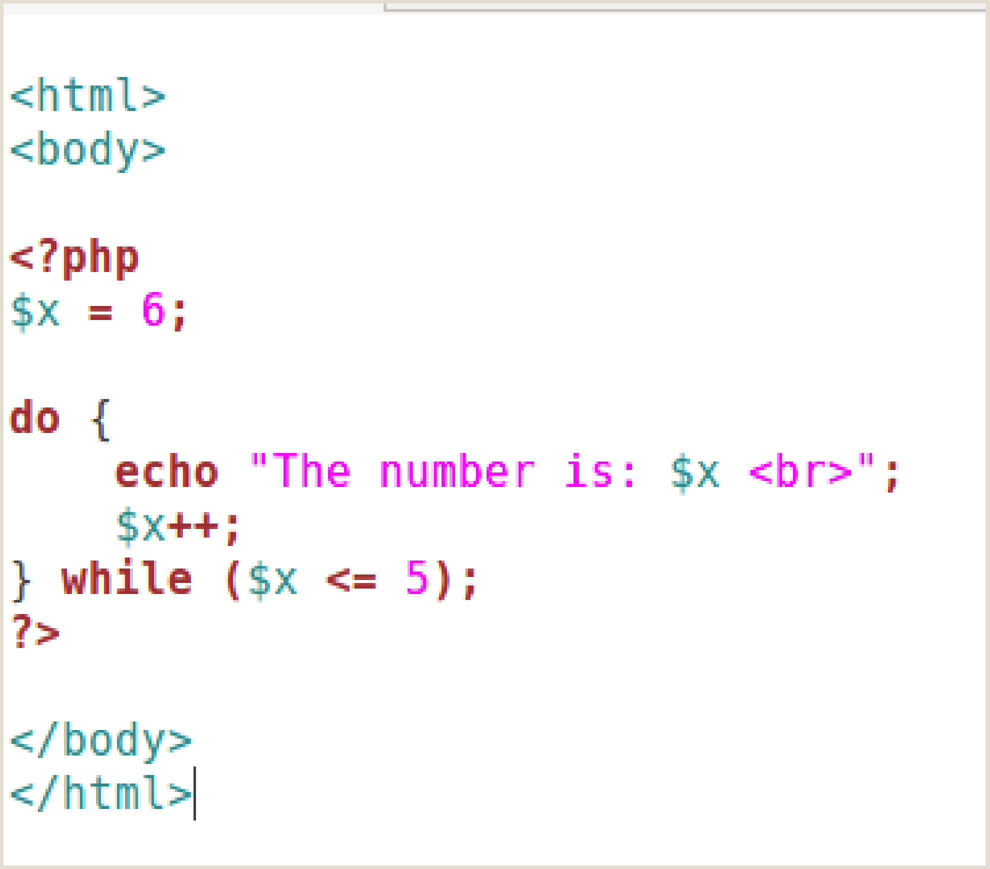
output



```
sum of first 10 even numbers is= 110
```

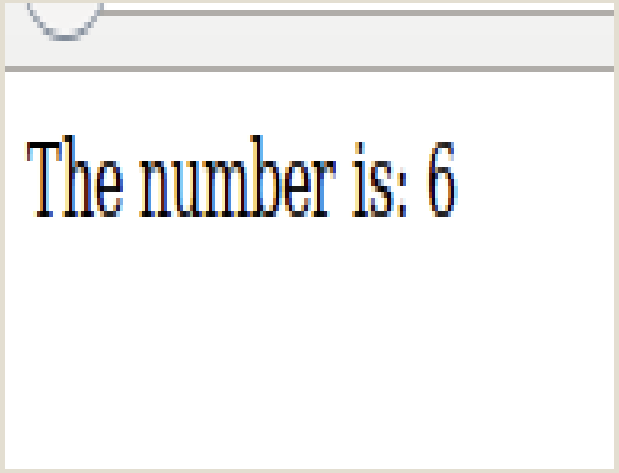
Looping Statements Example

dowhile.php



```
<html>
<body>

<?php
$x = 6;

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

</body>
</html>
```

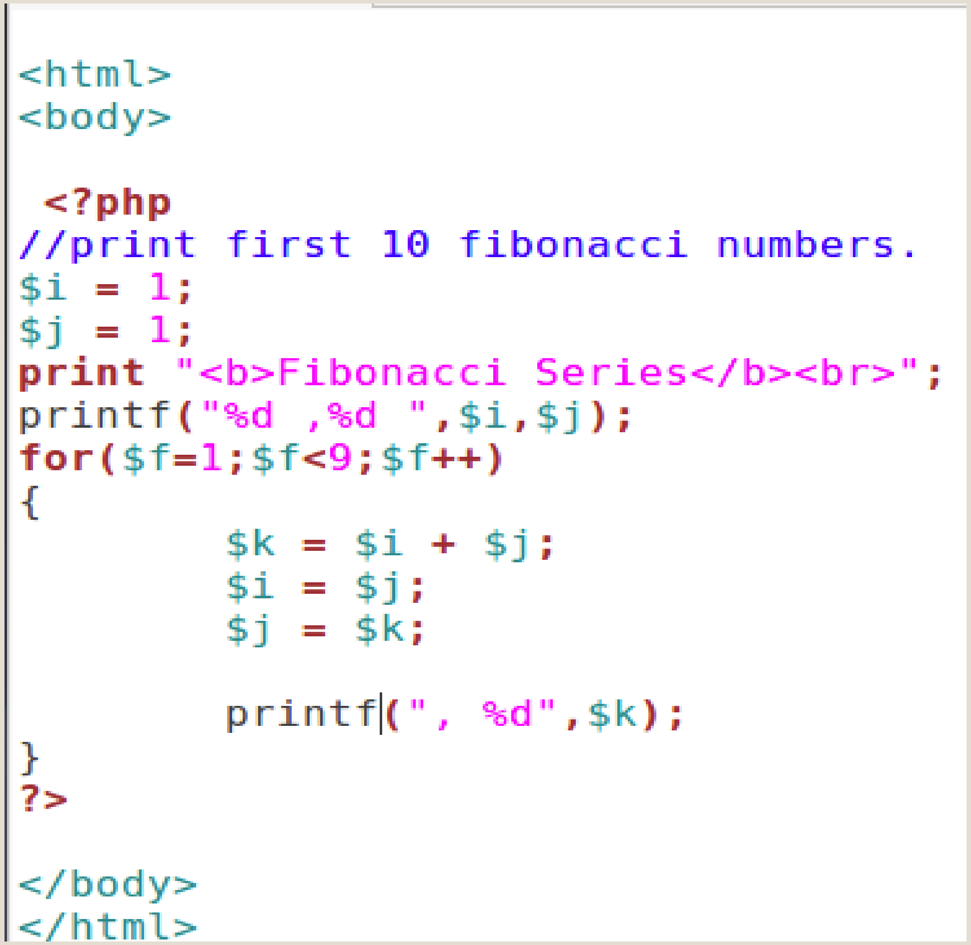
output



```
The number is: 6
```

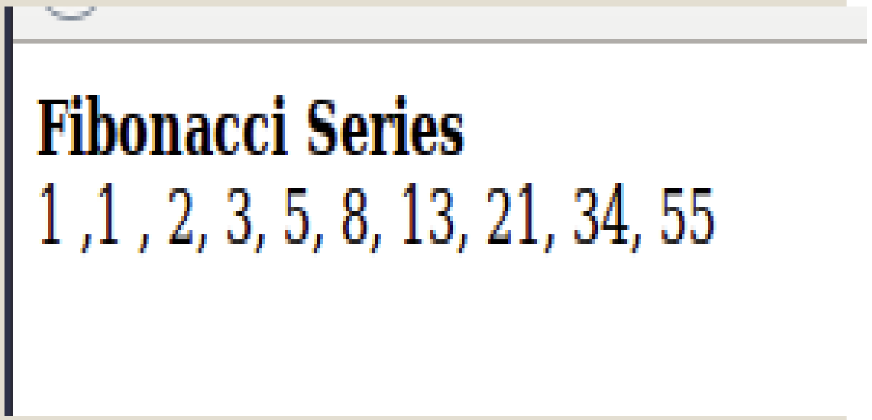
Looping Statements Example

forfibo.php



```
<html>
<body>

 <?php
//print first 10 fibonacci numbers.
$i = 1;
$j = 1;
print "<b>Fibonacci Series</b><br>";
printf("%d ,%d ",$i,$j);
for($f=1;$f<9;$f++)
{
        $k = $i + $j;
        $i = $j;
        $j = $k;

        printf(", %d",$k);
}
?>

</body>
</html>
```

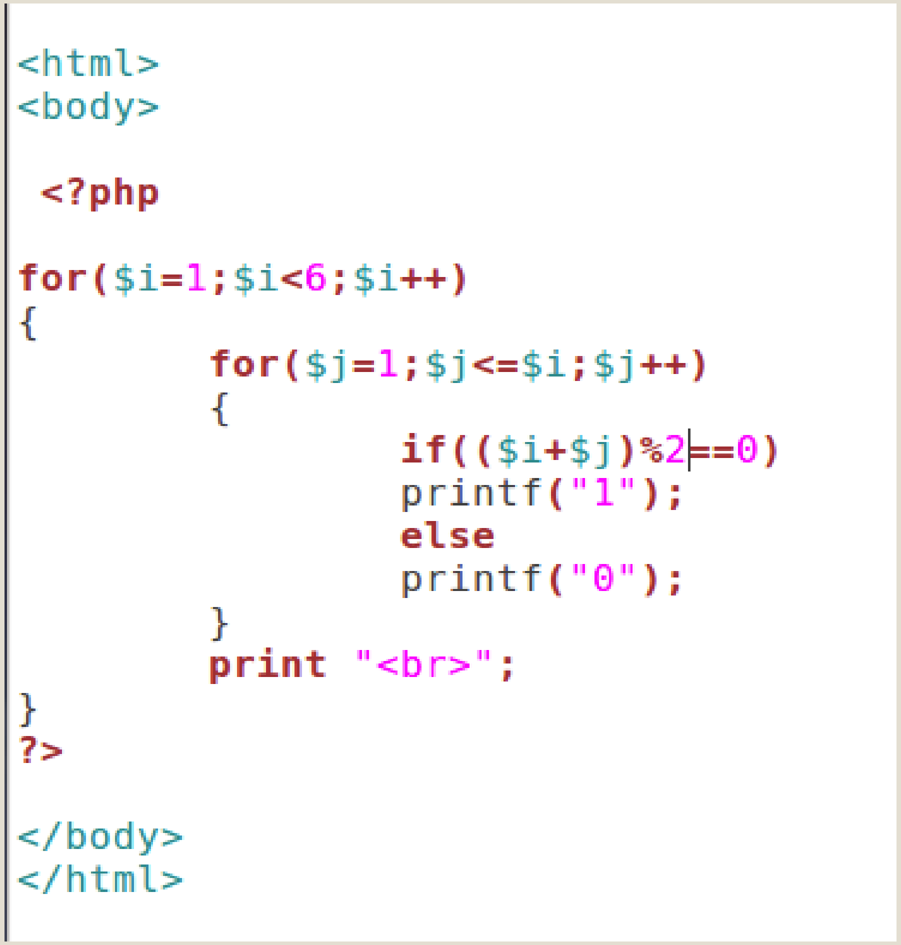
output



```
Fibonacci Series
1 ,1 , 2, 3, 5, 8, 13, 21, 34, 55
```

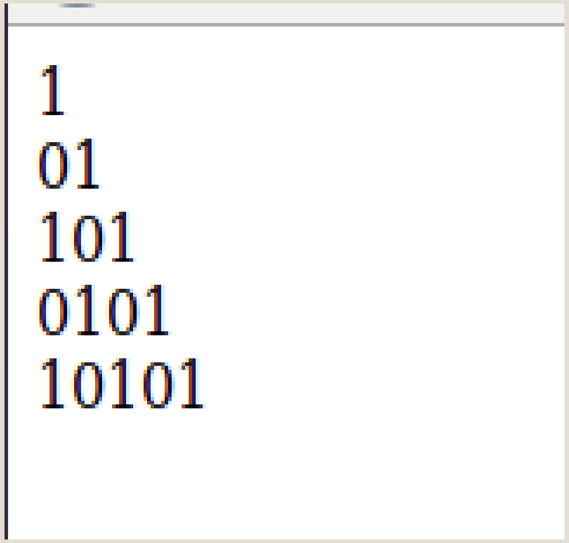
Looping Statements Example

forpattern.php



```
<html>
<body>

 <?php

for($i=1;$i<6;$i++)
{
        for($j=1;$j<=$i;$j++)
        {
                if(($i+$j)%2==0)
                printf("1");
                else
                printf("0");
        }
        print "<br>";
}
?>

</body>
</html>
```

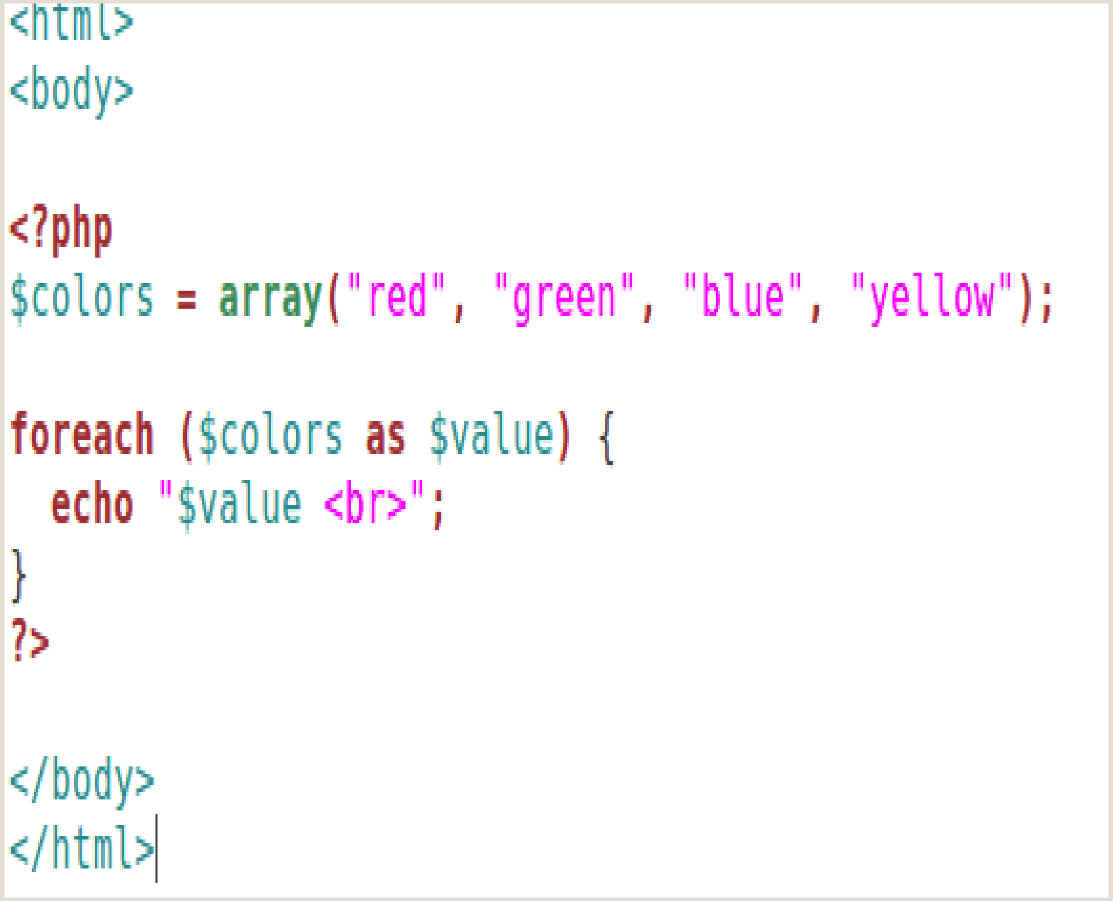
output



```
1
01
101
0101
10101
```

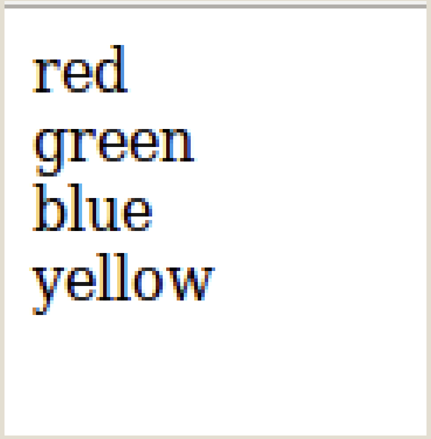
Looping Statements Example

foreach.php



```
<html>
<body>

<?php
$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $value) {
  echo "$value <br>";
}
?>


</body>
</html>
```

output



```
red
green
blue
yellow
```

Arrays

- It is collection of similar type of elements.
- PHP Array Types

1. Indexed Array
2. Associative Array
3. Multidimensional Array

Arrays

PHP Indexed Array

- PHP index is represented by number which starts from 0. We can store number, string and object in the PHP array. All PHP array elements are assigned to an index number by default.
- There are two ways to define indexed array:
  - 1st way: **$season=array("summer","winter","spring","autumn");**
  - 2nd way:

$season[0]="summer";

$season[1]="winter";

$season[2]="spring";

$season[3]="autumn";

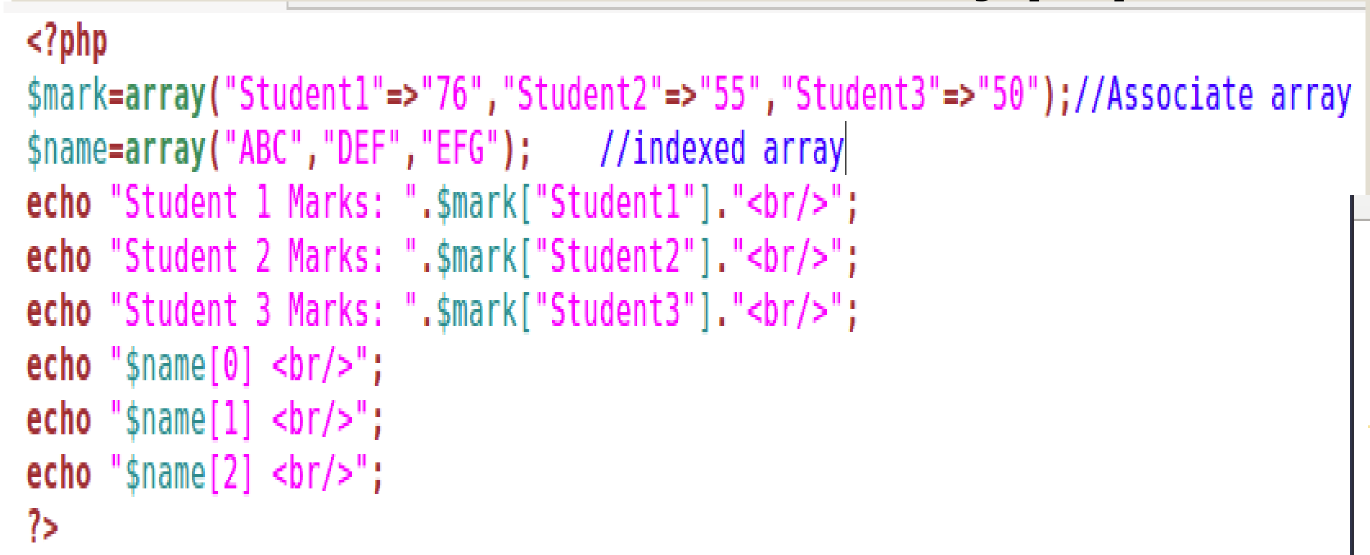
Arrays

PHP Associative Array

- PHP allows you to associate name/label with each array elements in PHP using => symbol.
- Ex:

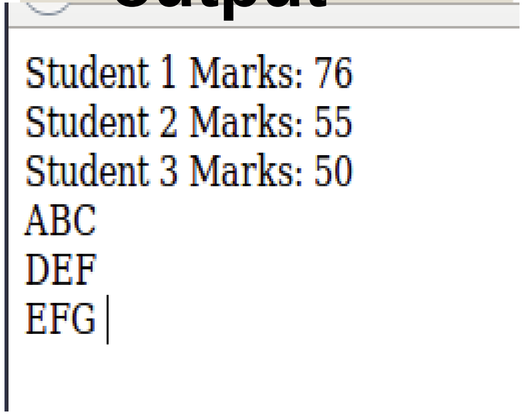
  $salary=array("A"=>"1","B"=>"2","C"=>"3");

Assoindexarray.php



```
<?php
$mark=array("Student1"=>"76","Student2"=>"55","Student3"=>"50");//Associate array
$name=array("ABC","DEF","EFG");    //indexed array
echo "Student 1 Marks: ".$mark["Student1"]."<br/>";
echo "Student 2 Marks: ".$mark["Student2"]."<br/>";
echo "Student 3 Marks: ".$mark["Student3"]."<br/>";
echo "$name[0] <br/>";
echo "$name[1] <br/>";
echo "$name[2] <br/>";
?>
```

output



```
Student 1 Marks: 76
Student 2 Marks: 55
Student 3 Marks: 50
ABC
DEF
EFG
```

Arrays

PHP Multidimensional Array

```
<?php
$emp = array
  (
  array(1,"A",4000),
  array(2,"B",5000),
  array(3,"C",3000)
  );
      for ($row = 0; $row < 3; $row++) {
  for ($col = 0; $col < 3; $col++) {
    echo $emp[$row][$col]."  ";
  }
  echo "<br/>";
}      ?>
```

```
1  A  4000
2  B  5000
3  C  3000
```

Functions

- Function can be placed anywhere inside the PHP script.
- And it executed only after a call to it.
- Syntax of function is

```
function function_name()
{
    Function body
}
```

➢ Three types of functions :

1. Functions having no parameter and no return value.

Ex: <? Php

```
        function sum()
        {
            $a = 10;
            $b = 20;
            $c = $a + $b;
       print $c;
        }
      print “The sum is”;
        sum();
   ?>
```

2. Functions having parameters passed to it but no return value.

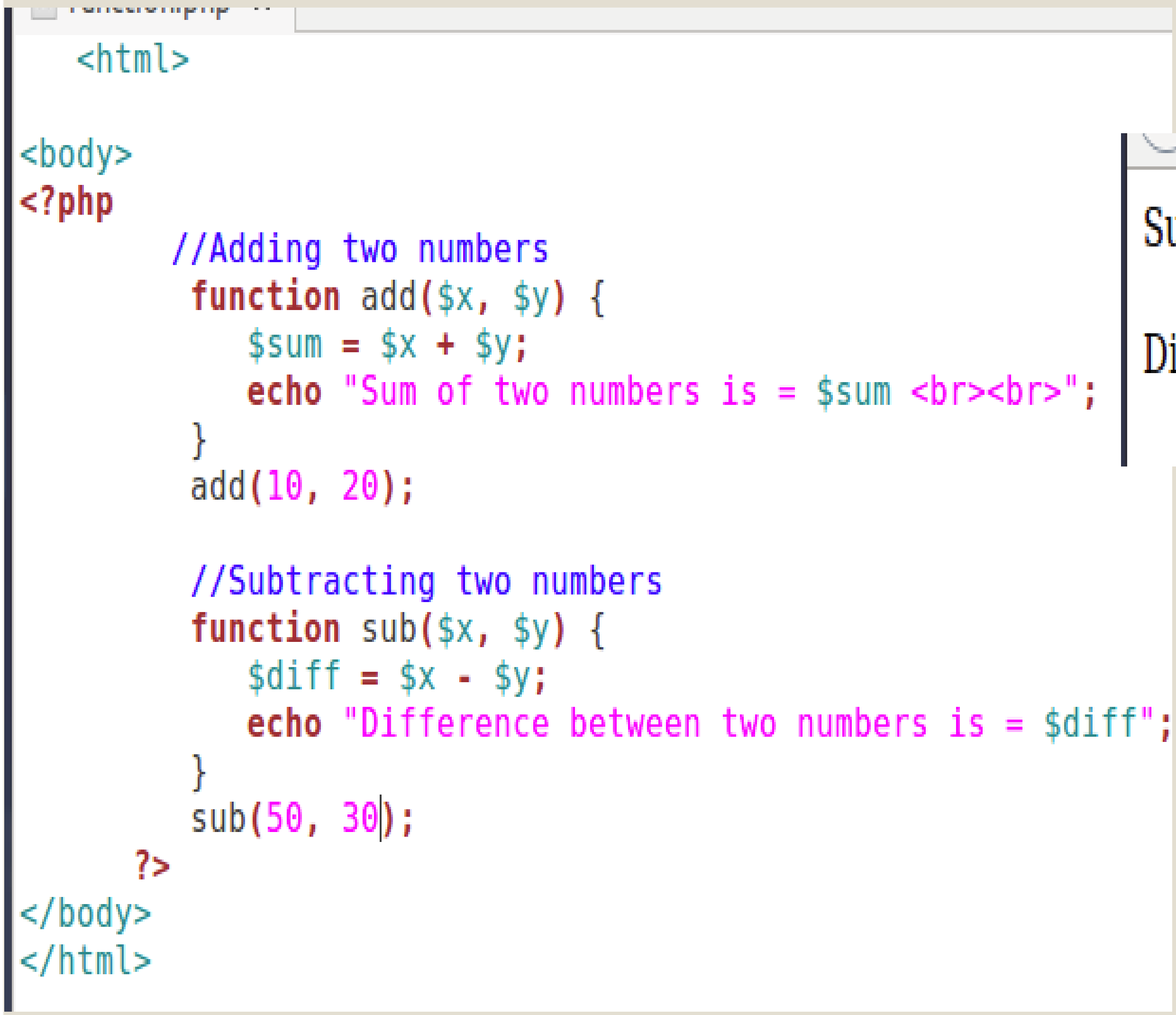
Ex: <? Php

```
        function sum ($a, $b)
        {
           $c = $a + $b;
          print $c;
        }
     print “The sum is”;
     $x=10;
        $y=20;
        sum ($x, $y);
    ?>
```

3. Functions having parameters passed to it and return some value.

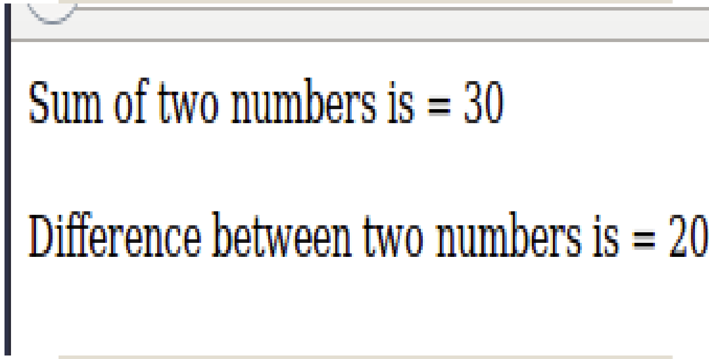
```
Ex: <? Php
          function sum ($a, $b)
          {
             $c = $a + $b;
         return $c;
          }
       print “The sum is”;
       $x=10;
          $y=20;
       $z=sum ($x, $y);
          print $z;
     ?>
```

Function.php



```
   <html>

<body>
<?php
        //Adding two numbers
         function add($x, $y) {
            $sum = $x + $y;
            echo "Sum of two numbers is = $sum <br><br>";
         }
         add(10, 20);

         //Subtracting two numbers
         function sub($x, $y) {
            $diff = $x - $y;
            echo "Difference between two numbers is = $diff";
         }
         sub(50, 30);
      ?>
</body>
</html>
```

Output



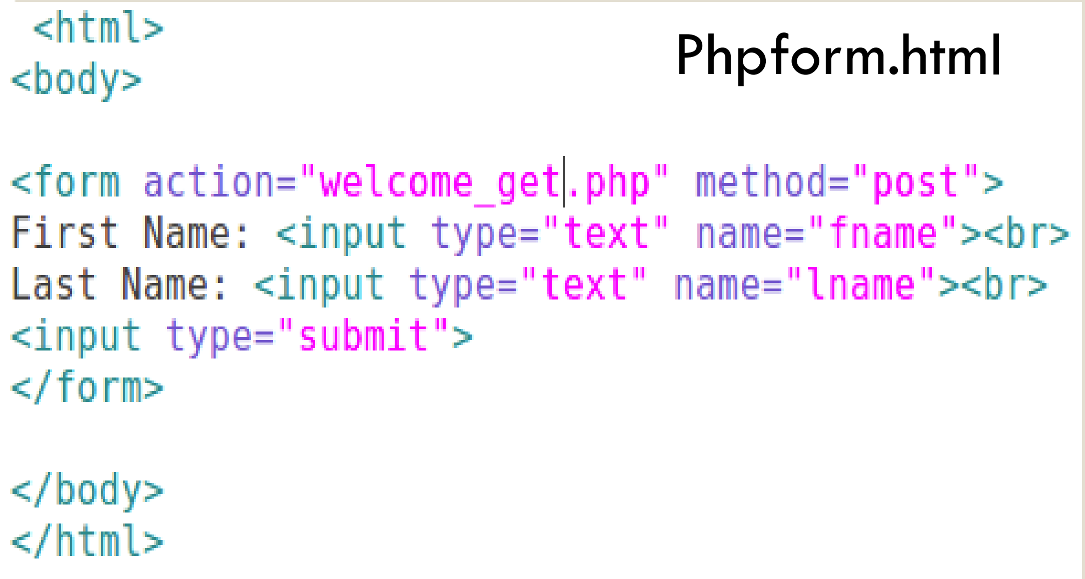
```
Sum of two numbers is = 30

Difference between two numbers is = 20
```

Forms

- Form gives GUI the user.
- The value of the form elements can be obtained by the PHP script using $_GET and $_POST variables.
- If we collect the values using GET method then $_GET variable is used in the PHP script.
- Note: if the GET method is used then all variable names and values will be displayed in the URL as a **Query String**.
- But if we use POST method then these values will not be displayed in the URL.
- Both GET and POST create an array (e.g. array(key => value, key2 => value2, key3 => value3, ...)). This array holds key/value pairs, where keys are the names of the form controls and values are the input data from the user.

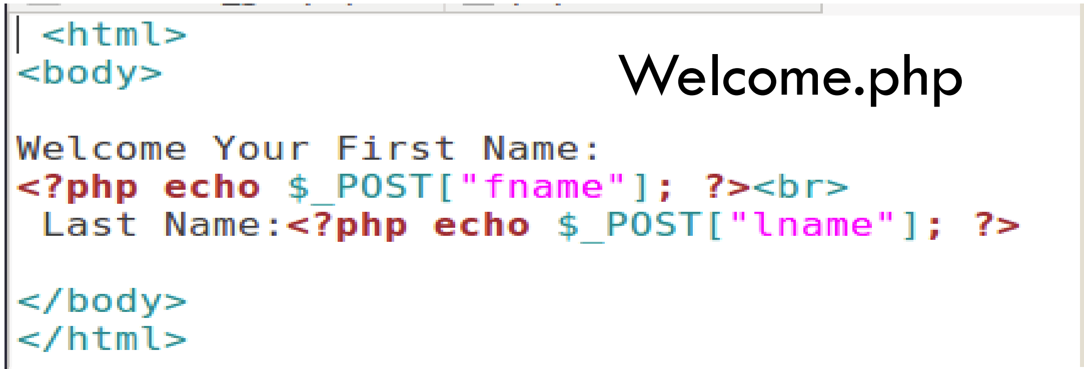
Form Example

Phpform.html



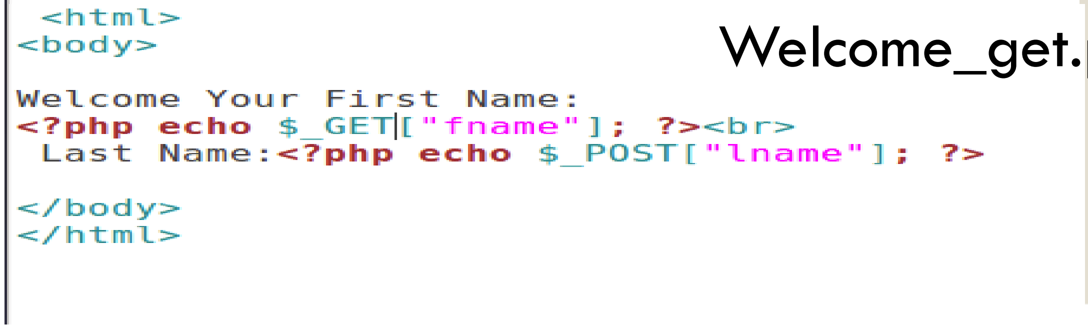
```
 <html>
<body>

<form action="welcome_get.php" method="post">
First Name: <input type="text" name="fname"><br>
Last Name: <input type="text" name="lname"><br>
<input type="submit">
</form>

</body>
</html>
```

Welcome.php



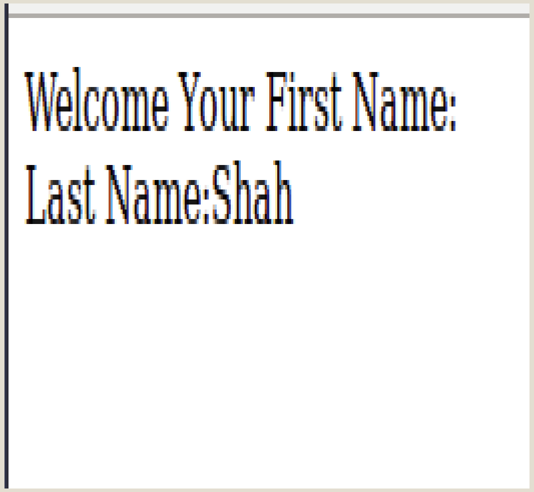
```
 <html>
<body>

Welcome Your First Name:
<?php echo $_POST["fname"]; ?><br>
 Last Name:<?php echo $_POST["lname"]; ?>

</body>
</html>
```

Welcome_get.php



```
 <html>
<body>

Welcome Your First Name:
<?php echo $_GET["fname"]; ?><br>
 Last Name:<?php echo $_POST["lname"]; ?>

</body>
</html>
```

output



```
Welcome Your First Name:
Last Name:Shah
```

Get Vs. Post

- **When to use GET?**
- Information sent from a form with the GET method is visible to everyone (all variable names and values are displayed in the URL). GET also has limits on the amount of information to send. The limitation is about 2000 characters. However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.
- GET may be used for sending non-sensitive data.
- Note: GET should NEVER be used for sending passwords or other sensitive information!

Get Vs. Post

- **When to use POST?**
- Information sent from a form with the POST method is invisible to others (all names/values are embedded within the body of the HTTP request) and has no limits on the amount of information to send.
- Moreover POST supports advanced functionality such as support for multi-part binary input while uploading files to server.
- However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

- Developers prefer POST for sending form data.

Form Validation

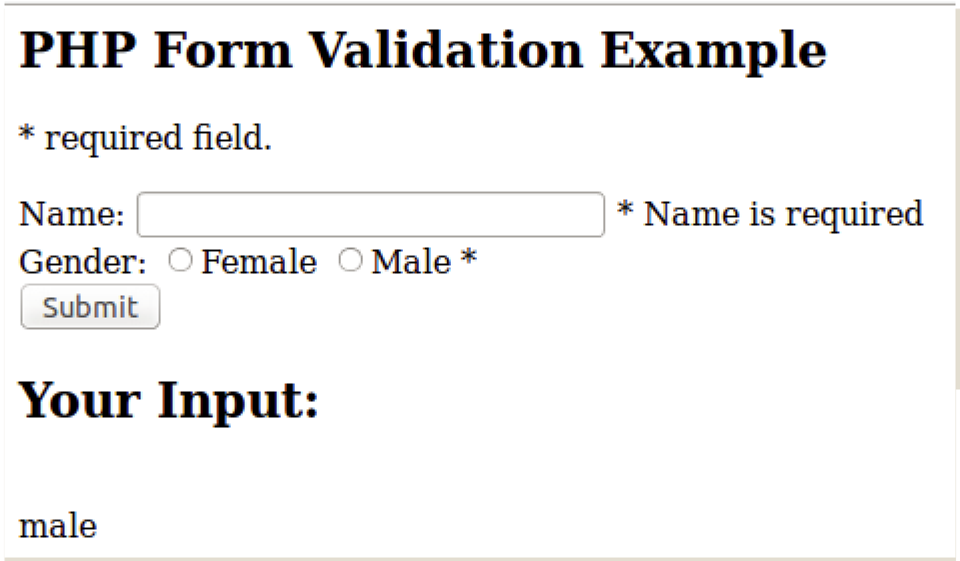
```
<html>
<body>
<?php
// define variables and set to empty values
$nameErr = $genderErr = "";
$name = $gender = "";
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = test_input($_POST["name"]);
  }
    if (empty($_POST["gender"])) {
    $genderErr = "Gender is required";
  } else {
    $gender = test_input($_POST["gender"]);
  }}
function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>
```

Form Validation

```
<h2>PHP Form Validation Example</h2>
<p><span class="error">* required field.</span></p>
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
```

/*$_SERVER["PHP_SELF"] sends the submitted form data to the page itself, instead of jumping to a different page. This way, the user will get error messages on the same page as the form.*/

```
  Name: <input type="text" name="name">
  <span class="error">* <?php echo $nameErr;?></span>
  <br>
  Gender:
  <input type="radio" name="gender" value="female">Female
  <input type="radio" name="gender" value="male">Male
  <span class="error">* <?php echo $genderErr;?></span>
  <br>
  <input type="submit" name="submit" value="Submit">
</form>
<?php
echo "<h2>Your Input:</h2>";
echo $name;
echo "<br>";
echo $gender;
?></body></html>
```



PHP Form Validation Example

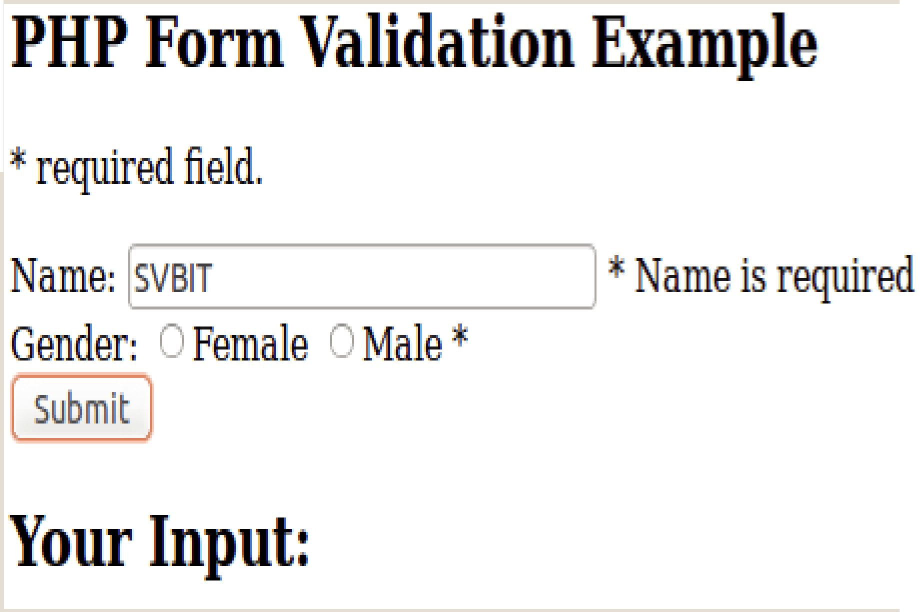
* required field.

Name: * Name is required

Gender: ○ Female ○ Male *

Your Input:

male



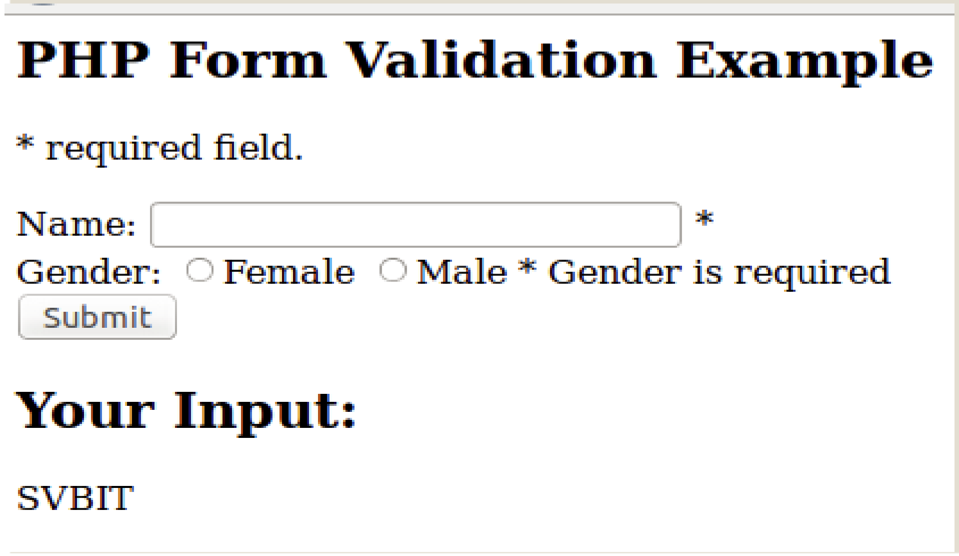
PHP Form Validation Example

* required field.

Name: SVBIT * Name is required

Gender: ○ Female ○ Male *

Your Input:



PHP Form Validation Example

* required field.

Name: *

Gender: ○ Female ○ Male * Gender is required

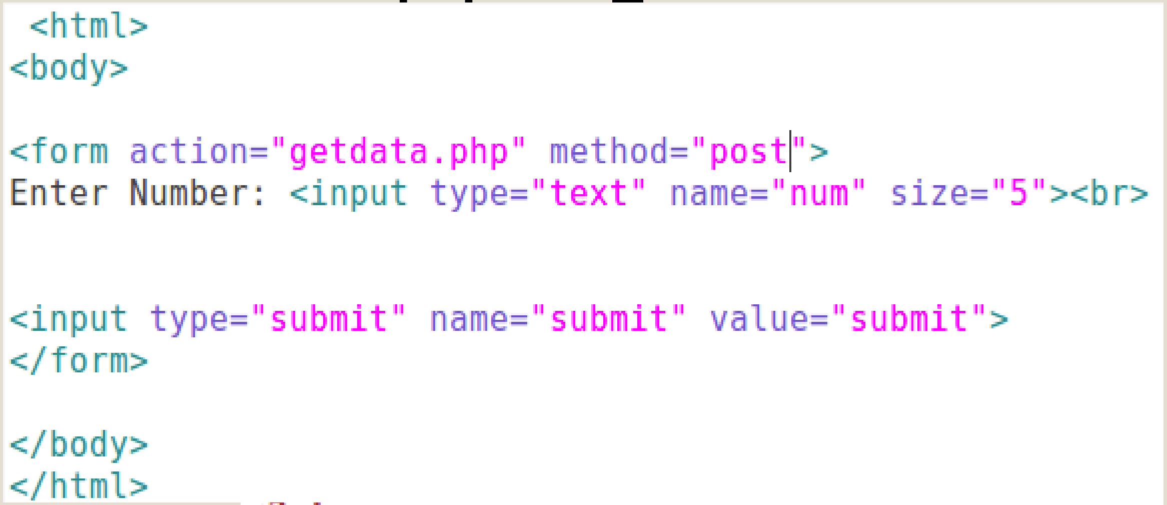
Your Input:

SVBIT

Form Example

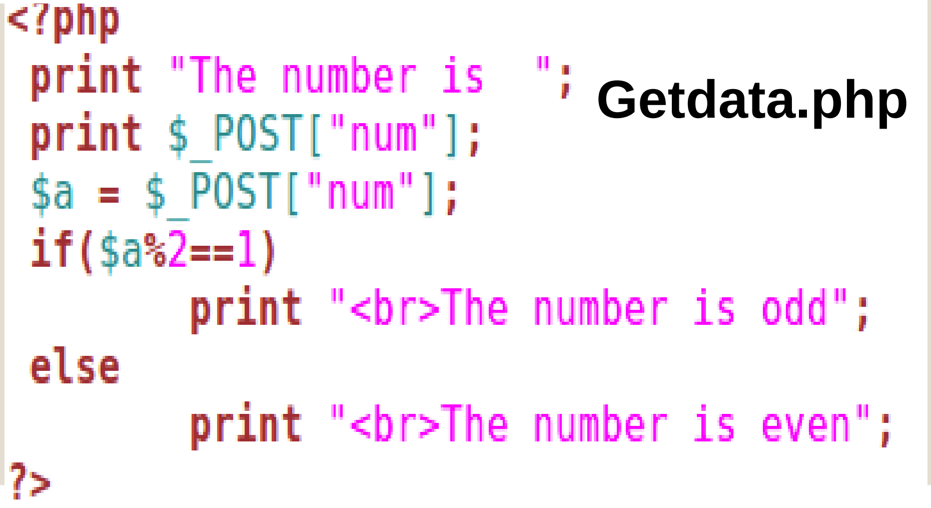
- ➢ Create HTML form to enter number. Write PHP code to display message if number is even or odd.

phpform_odd.html



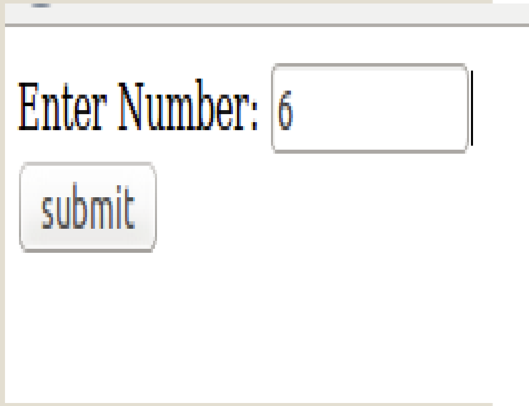
```
 <html>
<body>

<form action="getdata.php" method="post">
Enter Number: <input type="text" name="num" size="5"><br>


<input type="submit" name="submit" value="submit">
</form>

</body>
</html>
```

Getdata.php

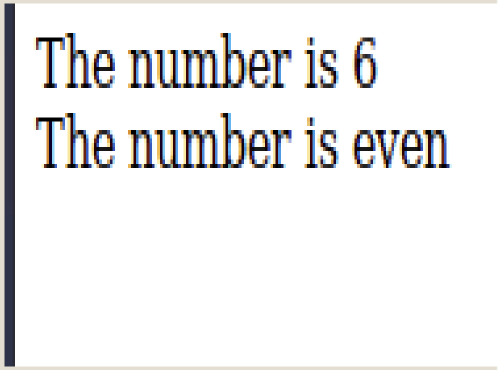


```
<?php
 print "The number is  ";
 print $_POST["num"];
 $a = $_POST["num"];
 if($a%2==1)
        print "<br>The number is odd";
 else
        print "<br>The number is even";
?>
```

output



Enter Number: 6



The number is 6
The number is even

PHP file

- PHP File System allows us to create file, read file line by line, read file character by character, write file, append file, delete file and close file.

- The PHP fopen() function is used to open a file.

- The PHP fclose() function is used to close an open file pointer.

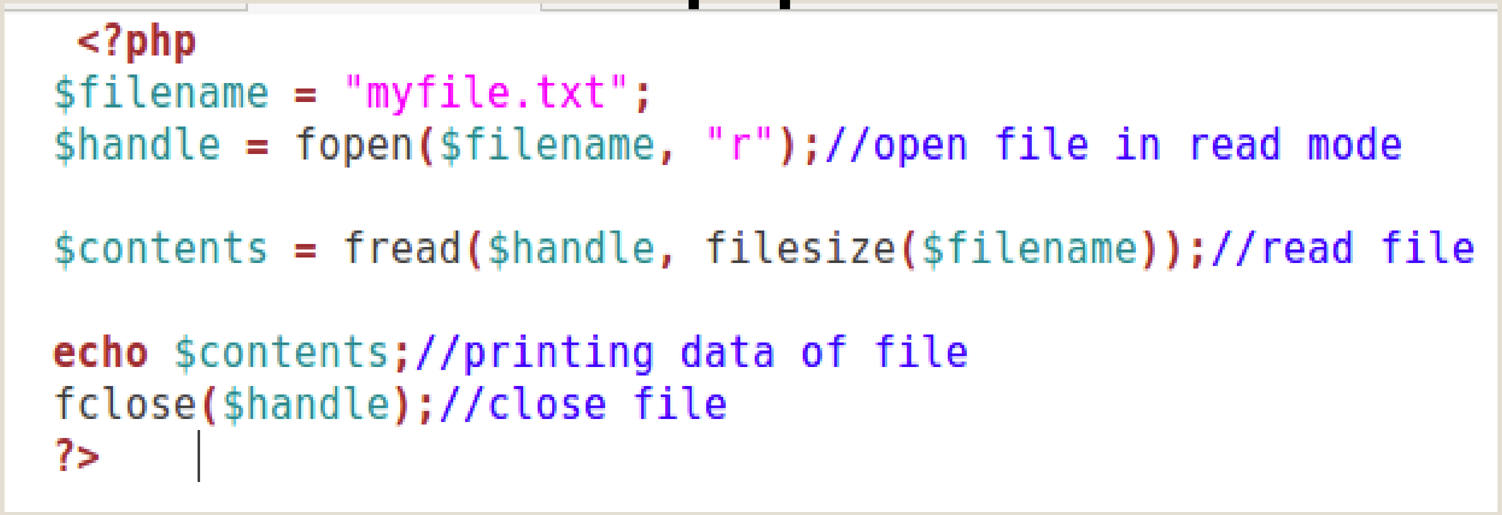
- The PHP fread() function is used to read the content of the file. It accepts two arguments: resource and file size.

- The PHP fwrite() function is used to write content of the string into ffile.

- The PHP unlink() function is used to delete file.

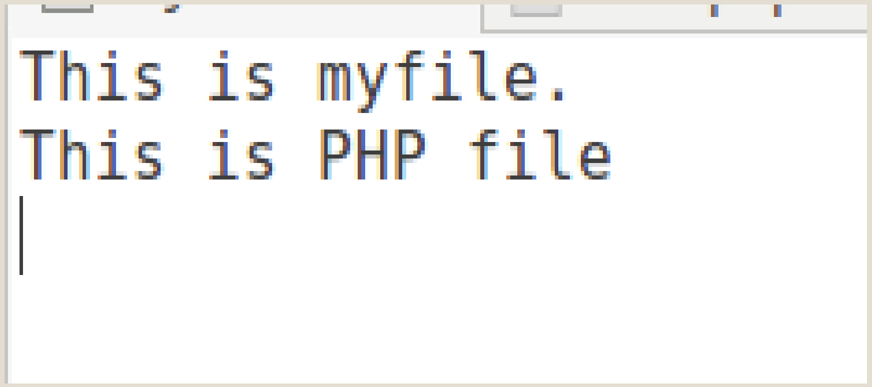
PHP file contents reading example

file1.php



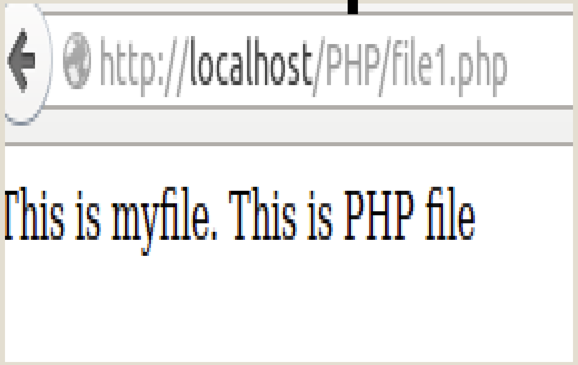
```
 <?php
$filename = "myfile.txt";
$handle = fopen($filename, "r");//open file in read mode

$contents = fread($handle, filesize($filename));//read file

echo $contents;//printing data of file
fclose($handle);//close file
?>
```

myfile.txt



```
This is myfile.
This is PHP file
```

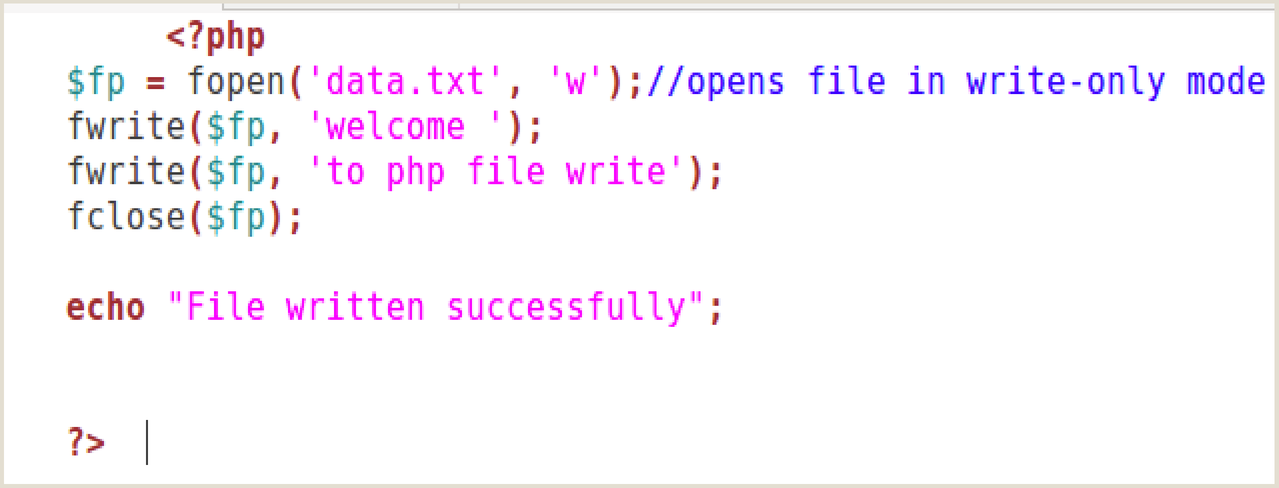
output



```
This is myfile. This is PHP file
```

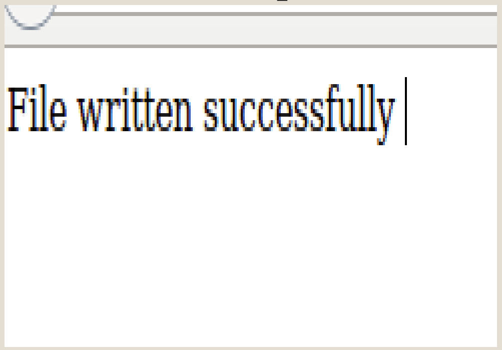
PHP Writing to File example

file2.php



```
     <?php
$fp = fopen('data.txt', 'w');//opens file in write-only mode
fwrite($fp, 'welcome ');
fwrite($fp, 'to php file write');
fclose($fp);

echo "File written successfully";


?>
```

output



```
File written successfully
```

EXCEPTION HANDLING:

➢PHP 5 has an exception model similar to that of other programming languages. Exceptions are important and provides a better control over error handling.

➢Lets explain there new keyword related to exceptions.

1. **Try** − A function using an exception should be in a "try" block. If the exception does not trigger, the code will continue as normal. However if the exception triggers, an exception is "thrown".
2. **Throw** − This is how you trigger an exception. Each "throw" must have at least one "catch".
3. **Catch** − A "catch" block retrieves an exception and creates an object containing the exception information.

CONT..

- When an exception is thrown, code following the statement will not be executed, and PHP will attempt to find the first matching catch block. If an exception is not caught, a PHP Fatal Error will be issued with an "Uncaught Exception ...

1. An exception can be thrown, and caught ("catched") within PHP. Code may be surrounded in a try block.
2. Each try must have at least one corresponding catch block. Multiple catch blocks can be used to catch different classes of exceptions.
3. Exceptions can be thrown (or re-thrown) within a catch block.

➢**Example**

Following is the piece of code, copy and paste this code into a file and verify the result.

```
<?php
   try {
      $error = 'Always throw this error';
      throw new Exception($error);

      // Code following an exception is not executed.
      echo 'Never executed';
   }catch (Exception $e) {
      echo 'Caught exception: ',  $e->getMessage(), "\n";
   }

   // Continue execution
   echo 'Hello World';
?>
```

➢In the above example $e->getMessage function is used to get error message.

There are following functions which can be used from **Exception** class.

1. **getMessage()** − message of exception

2. **getCode()** − code of exception

3. **getFile()** − source filename

4. **getLine()** − source line

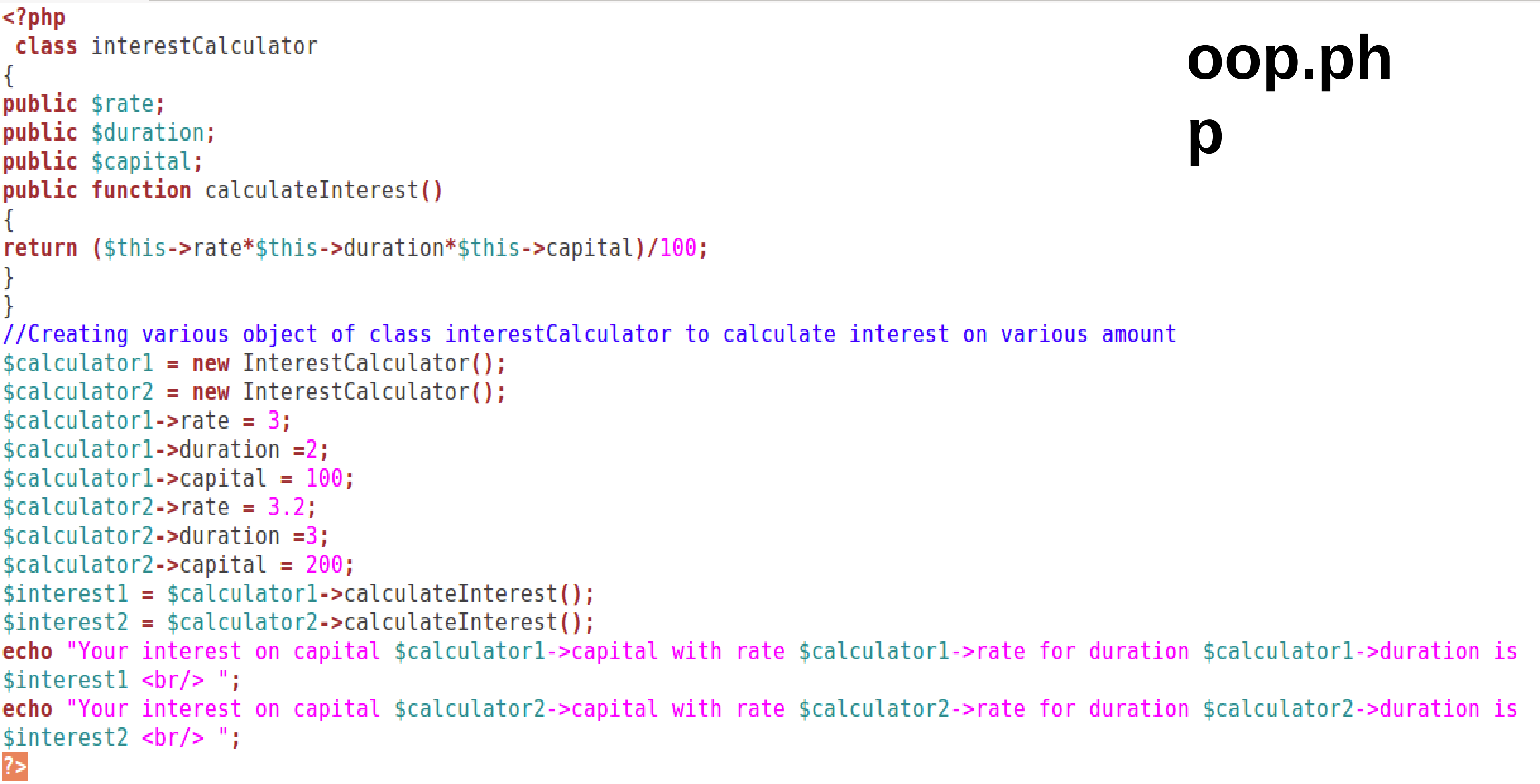
5. **getTrace()** − n array of the backtrace()

6. **getTraceAsString()** − formated string of trace

OOP with PHP

- **Encapsulation**
- **Message passing**
- **Inheritance**
- **Polymorphism**

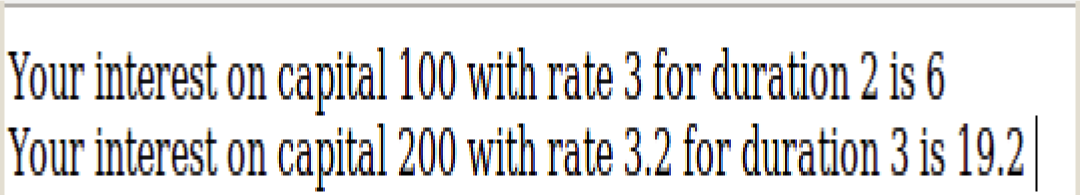
Example using class and object



```php
<?php
 class interestCalculator
{
public $rate;
public $duration;
public $capital;
public function calculateInterest()
{
return ($this->rate*$this->duration*$this->capital)/100;
}
}
//Creating various object of class interestCalculator to calculate interest on various amount
$calculator1 = new InterestCalculator();
$calculator2 = new InterestCalculator();
$calculator1->rate = 3;
$calculator1->duration =2;
$calculator1->capital = 100;
$calculator2->rate = 3.2;
$calculator2->duration =3;
$calculator2->capital = 200;
$interest1 = $calculator1->calculateInterest();
$interest2 = $calculator2->calculateInterest();
echo "Your interest on capital $calculator1->capital with rate $calculator1->rate for duration $calculator1->duration is
$interest1 <br/> ";
echo "Your interest on capital $calculator2->capital with rate $calculator2->rate for duration $calculator2->duration is
$interest2 <br/> ";
?>
```

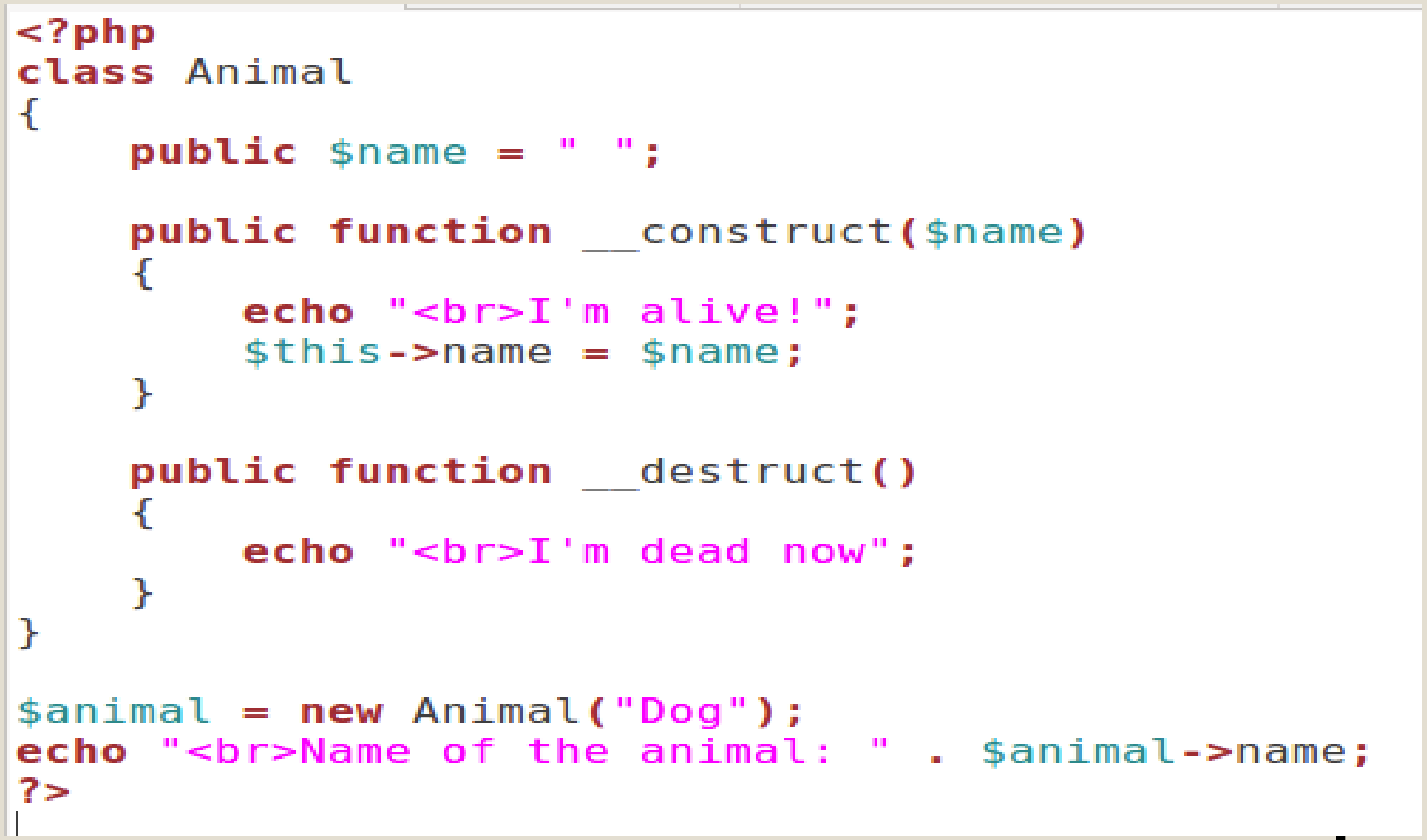
oop.php

output



```
Your interest on capital 100 with rate 3 for duration 2 is 6
Your interest on capital 200 with rate 3.2 for duration 3 is 19.2
```

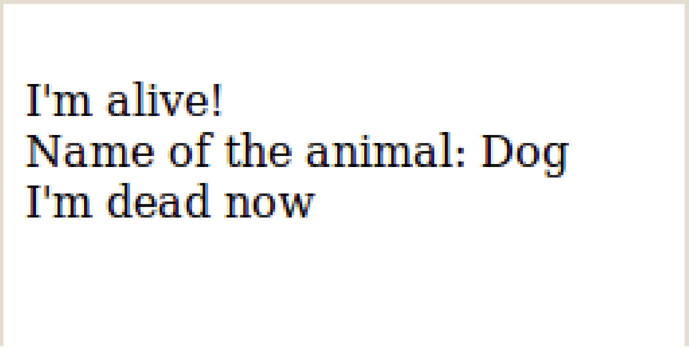
Example using constructor and destructor



```php
<?php
class Animal
{
    public $name = " ";

    public function __construct($name)
    {
        echo "<br>I'm alive!";
        $this->name = $name;
    }

    public function __destruct()
    {
        echo "<br>I'm dead now";
    }
}

$animal = new Animal("Dog");
echo "<br>Name of the animal: " . $animal->name;
?>
```

condes.php

output



```
I'm alive!
Name of the animal: Dog
I'm dead now
```

THANK YOU……