Functions in PHP

Session 11

Objectives

- *Explain functions in PHP*
- *Describe the built-in functions in PHP*
- *Explain the process of creating a user-defined function*
- *Explain the process of passing arguments to a function*
- *Explain the process of returning values from a function*
- *Explain the use of recursive functions*

Introduction

- Functions
  - Are named section of a program
  - Are used to perform a specific task
  - Split program into modules
  - Are used to enable the developer to reuse the same piece of code
  - Can be easily modified in a program instead of going through entire code to make changes

Functions

- Statements are grouped into a single unit to perform a specific task
- Enhance the logical flow in a program by dividing complicated code sequences into smaller modules
- Enable to write a piece of code and assign a name to it
- Include parameters that are:
  - Variables
  - Specified within the parenthesis after the name of a function
  - Used to add more functionality
- Executed or invoked anywhere in the program using the assigned name

Built-in PHP Functions

- Provides different built-in functions to be included in the PHP script
- Built-in functions are grouped into following categories:
  - Mathematical functions
  - String functions
  - Date and time functions
  - Error handling functions
  - Database functions
  - Array functions
  - Mail functions

Mathematical Functions

- Operate on numerical data

Table lists and describes some of the mathematical functions in PHP:

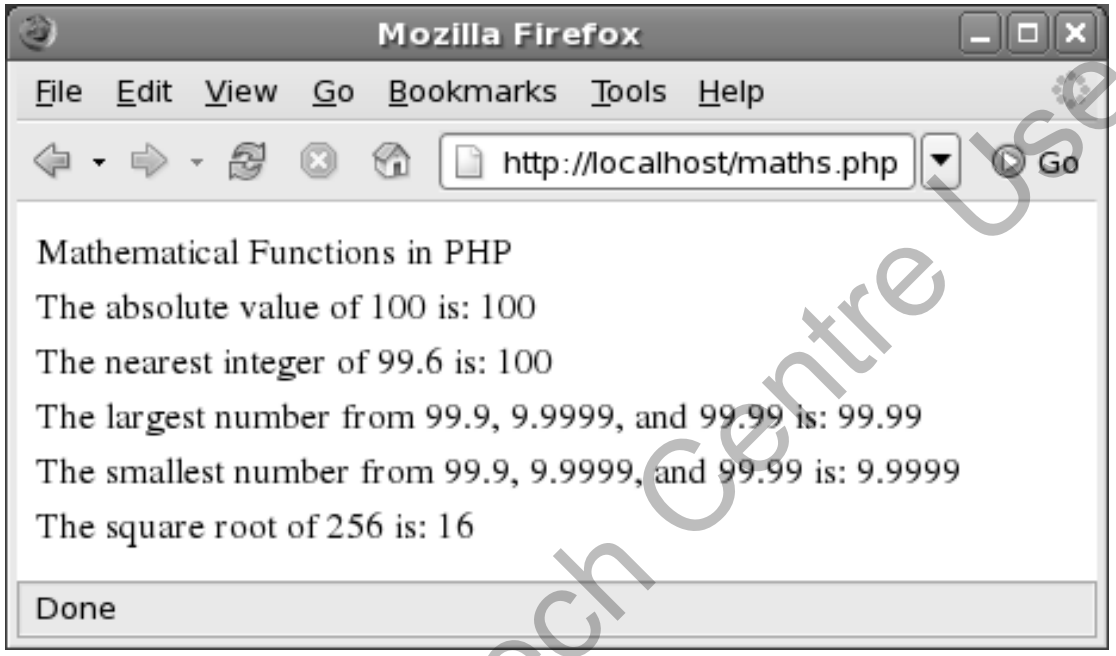
Function Name	Syntax	Description
abs	abs(arg)	Returns the absolute value of the argument
max	max(arg1,arg2,....)	Returns the largest value from the specified arguments. It also allows comparing multiple arrays
min	min(arg1,arg2,....)	Returns the smallest value from the specified arguments. It also allows comparing multiple arrays
sqrt	sqrt(arg)	Returns the square root of the argument
pow	pow(base, exp)	Returns the value of the base raised to the power of the exponential
round	round(number)	Returns the nearest integer of the specified number
rand()	rand(min, max)	Returns a random integer
ceil()	ceil(x)	Returns the value of a number rounded upwards to the nearest integer
floor()	floor(x)	Returns the value of a number rounded downwards to the nearest integer

- **maths.php** - Use of mathematical functions

Snippet

```
<?php
echo "Mathematical Functions in PHP";
echo "<br>";
echo "The absolute value of 100 is: ";
echo abs(100);
echo "<br>";
echo "The nearest integer of 99.6 is: ";
echo round(99.6);
echo "<br>";
echo "The largest number from 99.9, 9.9999, and
99.99 is: ";
echo min(99.9,9.9999,99.99);
echo "<br>";
echo "The square root of 256 is: ";
echo sqrt(256);
echo "<br>";
?>
```

Displays the following output:



String Functions

- Operate on character type of data
- Table lists some of the string functions in PHP:

Function Name	Syntax	Description
chr	chr(ascii)	Returns the character equivalent to the specified ASCII code
bin2hex	bin2hex(string)	Converts a string of ASCII characters to hexadecimal values
strtolower	strtolower(string)	Converts the specified string to lower case
strlen	strlen(string)	Returns the length of the string specified as an argument
strcmp	strcmp(string1,string2)	Compares two strings. Returns zero if string1 is equal to string2. It returns less than zero, if string1 is less than string2. Otherwise, it returns greater than zero when string1 is greater than string2
strtoupper	strtoupper(string)	Converts the specified string to upper case
strrev	strrev(string)	Returns the reverse of the string
stristr()	stristr(string,search)	Finds the first occurrence of a string inside another string (case-insensitive)
strrchr()	strrchr(string,char)	Finds the last occurrence of a string inside another string

String Functions

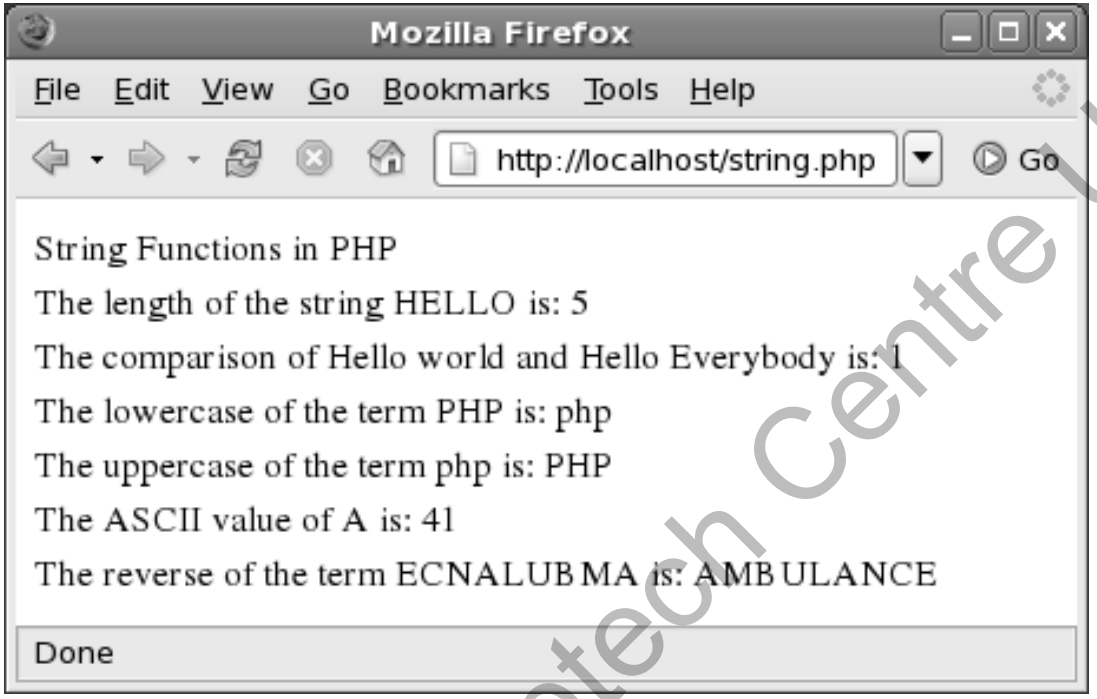
Function Name	Syntax	Description
strrpos()	strrpos(string,find,start)	Finds the position of the last occurrence of a string inside another string (case-sensitive)
strncmp()	strncmp(string1,string2,length)	String comparison of the first n characters

- **`string.php`** - Use of string functions

Snippet

```
echo strtoupper("php");

echo "<br>";

echo "The ASCII value of A is: ";

echo bin2hex("A");

echo "<br>";

echo "The reverse of the term ECNALUBMA is: ";

echo strrev("ECNALUBMA");

?>
```

Displays the following output:



- Enables to calculate the date and time on the system
- Table lists and describes some of the date and time functions :

Function Name	Syntax	Description
checkdate	checkdate(month,day,year)	Returns the value as 1 if the specified date is valid. A valid date contains: ◆ Month between 1 and 12 ◆ Day within the range of days for the specified month ◆ Year between 1 and 32767
getdate	getdate(timestamp)	Returns an array containing date and time information. The information is returned for a Unix timestamp
time	time()	Returns the current time measured in the number of seconds

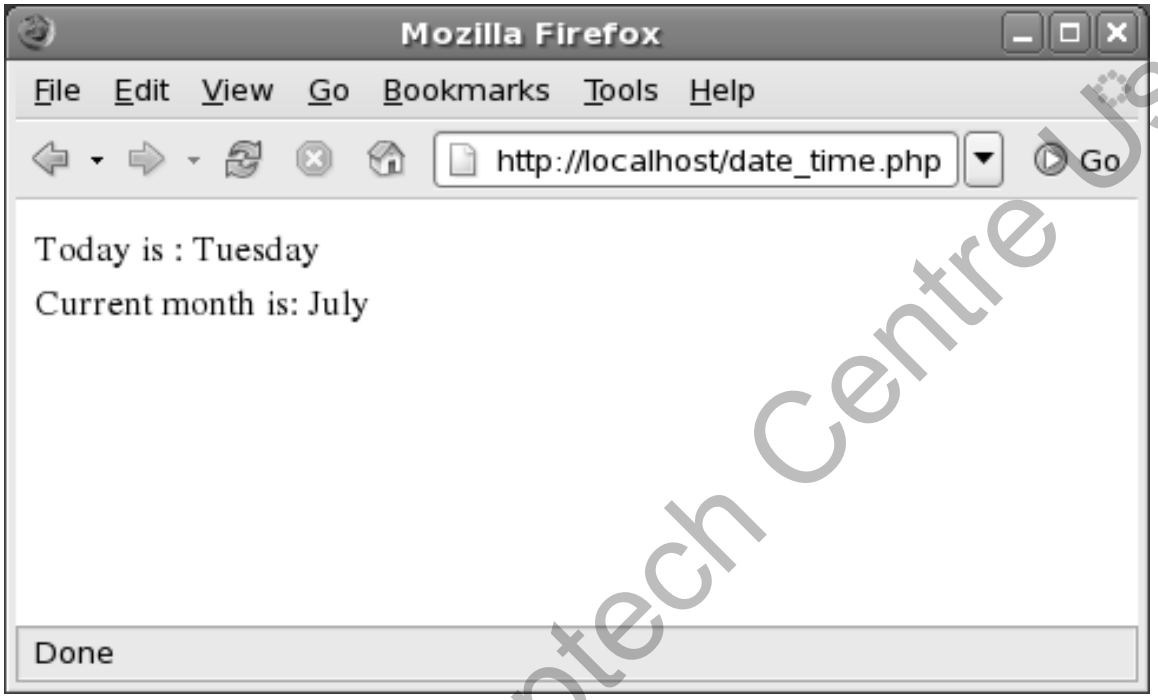
Function Name	Syntax	Description
Date	date(format, timestamp)	Returns a string depending on the timestamp or the current local time, if the timestamp is not specified Some of the formats that can be used are as follows: d – day of the month D – textual representation of the day j – day of the month without leading zeros (1 to 31) m – numeric representation of the month M – textual representation of the month y – a two digit representation of the year Y – a four digit representation of the year a – Lowercase am or pm h – 12 hour format of an hour H – 24 hour format of an hour i – minutes with leading zero s – seconds with leading zeros

- **`date_time.php`** - Use of common date and time functions

Snippet

```
<?php

date_default_timezone_set('Asia/Calcutta');

echo "Today is : " .date("l");

$Today_Date=getdate();

$current_month=$Today_Date['month'];

echo "<br>";

echo "Current month is: ";

echo $current_month;

?>
```

Displays the following output:



The "1" is the argument string, which corresponds to the textual representation of the day of the week and will return the day.

Error Handling Functions

- Defines the error handling rules and modify the way the errors are handled
- Table lists the error handling functions:

Function Name	General Form	Description
trigger_error	trigger_error(error_msg [,error_type])	Generates an error message, that is, it defines an error message at a specified condition as specified by the user User-defined function such as set_error_handler() inbuilt error handler can be used with it
set_error_handler	set_error_handler(error_handler)	Handles errors during runtime by using a user-defined function
error_reporting	error_reporting(Constant)	Specifies which PHP errors are reported. PHP provides many levels of errors. You can use this function to set a level during the run-time of the script
strtotime	strtotime(string time [,now])	Parses an English textual date or time into a Unix timestamp (the number of seconds since January 1 1970 00:00:00 GMT)

Table lists some of the error constants:

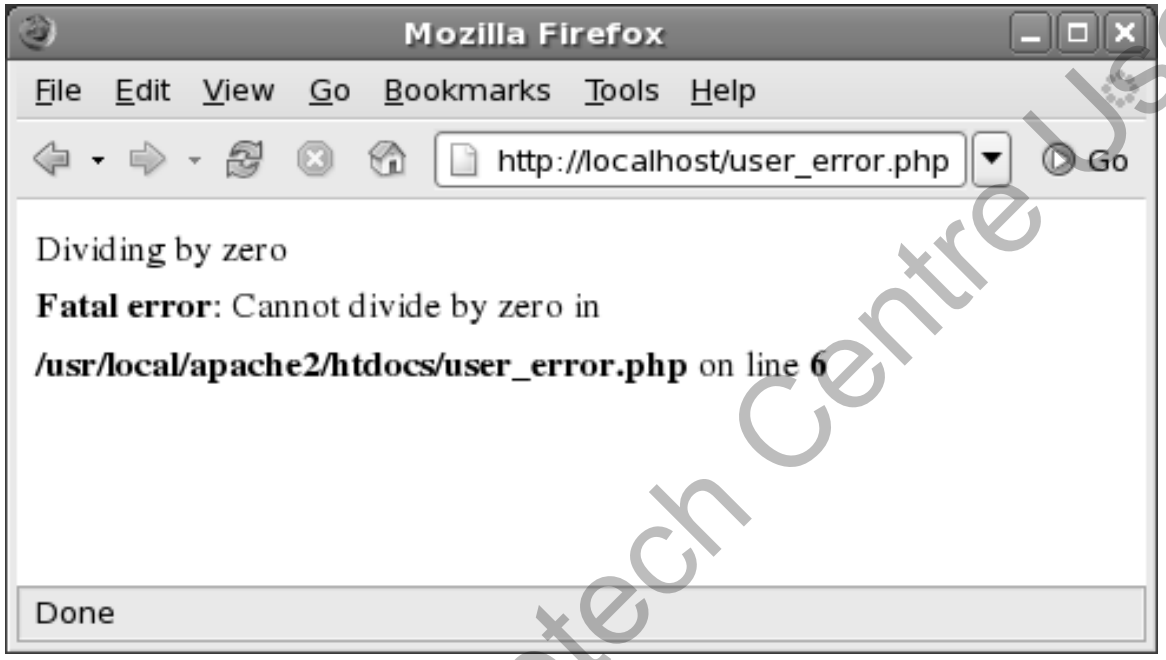
Constant Name	Value	Description
E_ERROR	1	Displays errors which are not recoverable
E_WARNING	2	Displays non-fatal run-time errors, however, this does not halt the execution of the script
E_PARSE	4	Displays the errors generated by the parser during the compilation
E_NOTICE	8	Displays run time error notices
E_COMPILE_ERROR	64	Displays compile time errors
E_USER_ERROR	256	Displays user generated error message
E_USER_WARNING	512	Displays user generated warning message
E_CORE_ERROR	16	Displays Fatal errors during PHP start up
E_CORE_WARNING	32	Displays Non-fatal errors during PHP start up
E_COMPILE_WARNING	128	Displays errors generated by the Zend Scripting Engine
E_ALL	8191	Displays all errors and warnings that are supported

- **user_error.php -** Use of error handling functions

Snippet

```
<?php

$num1=0;

if($num1==0)

{

echo "Dividing by zero";

trigger_error("Cannot divide by zero", E_USER_ERROR);

}

else

{

$B=100/$num1;

}

?>
```

Displays the following output:



The value of **$num1** variable is tested.

User-defined Functions

- Function can be defined or created
- Function definition contains the code to be executed
- Defining a function is as follows:

Syntax

```
function function_name (arg1, arg2, arg3, …)
{
statement list
}
```

- The `return expr` statement within the body of the function is used to return a value from a function
- Snippet accepts an argument, `$x`, and returns its square

Snipppet

```
function square ($x)
{
return $x*$x;
}
```

- To execute the function, invoke the function as follows:

Syntax

```
fun_name();
```

where,

fun_name – specifies the name of the function

User-defined Functions

- A PHP code can be included inside a function, other functions, and class definitions
- Rules to define function names are similar to the rules for defining labels in PHP
- Functions are not required to be defined before referencing
- When defining a function in a conditional manner, the script must first process the function definition before invoking the function

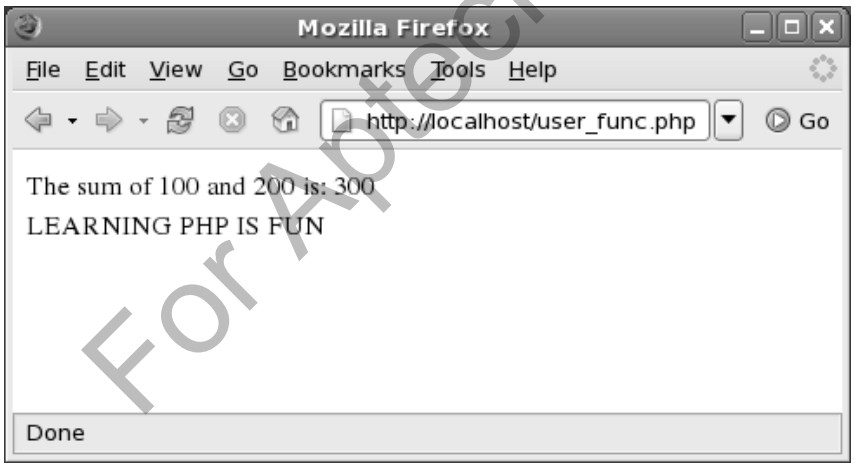
- **user_func.php** - Use of user defined functions

Snippet

```
<?php
//A function to calculate the sum of two variables
function addition()
{
      $A=100;
      $B=200;
      $C=$A+$B;
echo "The sum of 100 and 200 is: $C";
}
addition();
```

```
echo "<br>";

// A function to display the text

function Display()

{

      echo "LEARNING PHP IS FUN";

}

Display();

?>
```

Displays the following output:



Passing Arguments to Functions

- PHP supports passing of arguments to a function
- The three different ways of passing arguments to a function are as follows:
  - Passing arguments by value
  - Passing arguments by reference
  - Setting default values for arguments
- The function definition determines the method of passing arguments to the function

- A function with arguments is as follows:

Syntax

```
function fun_name(arg1,arg2,...)
{
Code to be executed
}
```

Where,

arg - specifies the argument that is passed to the function

Passing Arguments by Value

- When an argument is passed by value it must:
  - Prefix with the dollar (`$`) sign
  - Be any valid expression which are evaluted and assigned to the corresponding variable

- **arg_value.php** - Passing arguments by value

Snippet

```
<?php

//Creating a function to calculate the square of a
     number

function Square($A)

{

     //The argument is passed by Value in the
     function definition

     //using the $ sign.

     //Calculating the square of the number

     $A=$A*$A;

     //Displaying the square of the number
```

```
     echo $A;
}

//Assigning a value to the variable
$A=5;

//Displaying the text
echo "The square of $A is: ";

//Calling the function
Square($A);
// Creating a function to subtract one variable from another
function subtraction($A,$B)
{
      //Calculating the difference
     $C=$A-$B;
     //Displaying the text
     echo "<br>The difference of $A and $B is: $C";
}
//Calling the function and assigning values to the argument
subtraction(90,45);
?>
```

Passing Arguments by Value

Displays the following output:

The `subtraction()` function subtracts the variable **`$B`** from **`$A`** and stores the resultant value in **`$C`**.

Passing Argument by Reference

- When an argument is passed by reference it must:
  - Be a variable
  - Prefix the arguments with the ampersand (&) sign to indicate that the value is passed by reference

- **arg_ref.php -** Passing values to the function by reference

Snippet

```
<php

//Defining a function and passing value to the
      arguments by reference

function Square(&$A)

{

      //Calculating the square of the number and
      storing it in a variable

      $A=$A*$A;

      //Displaying the result

      echo $A;

}

//Assigning value outside the function

$A=5;

//Displaying text
```

Passing Argument by Reference

- **arg_ref.php -** Passing values to the function by reference

Snippet

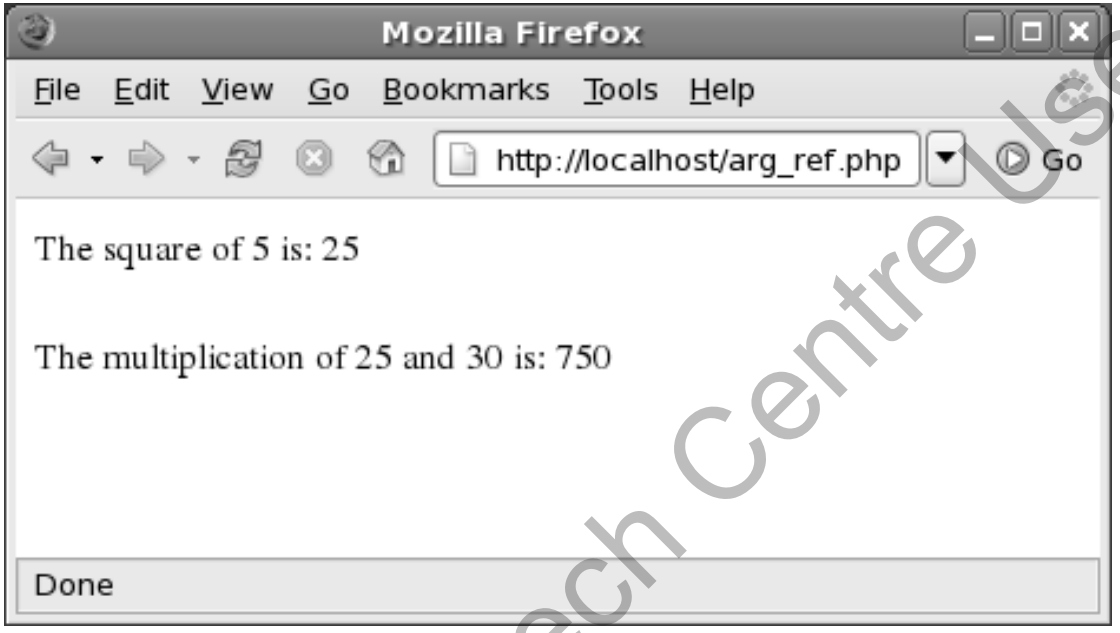
```
echo "The square of $A is: ";

//Executing the function by passing value to argument
by reference

Square($A);

//Defining a function and passing value to the
arguments by reference

function multiplication(&$A,&$B)

{

      //Calculating the multiplication of two numbers
        storing it in a variable
```

- **arg_ref.php -** Passing values to the function by reference

Snippet

```
    $C=$A*$B;

    //Displaying text

    echo "<br><br>The multiplication of
$A and $B is: $C";

}

//Assigning value outside the function


$A=25;

$B=30;

//Executing the function by passing
value to argument by reference

multiplication($A,$B);

?>
```

Displays the following output:



The `multiplication()` function calculates the product of two variable **$A** and **$B** and stores the value in **$C**.

Setting Default Values

- PHP enables to assign default values to an argument in a function
- The default values enable the developer to initialize the function parameters when the function is invoked without any value being passed
- The default value assigned can be any one of the following:
  - Constant
  - Scalar
  - Array with scalar values or constant

Setting Default Values

- The use of default parameters

Snippet

```
function increment(&$num, $increment = 1)

{

$num += $increment;

}

$num = 4;

increment($num);


increment($num, 3);
```

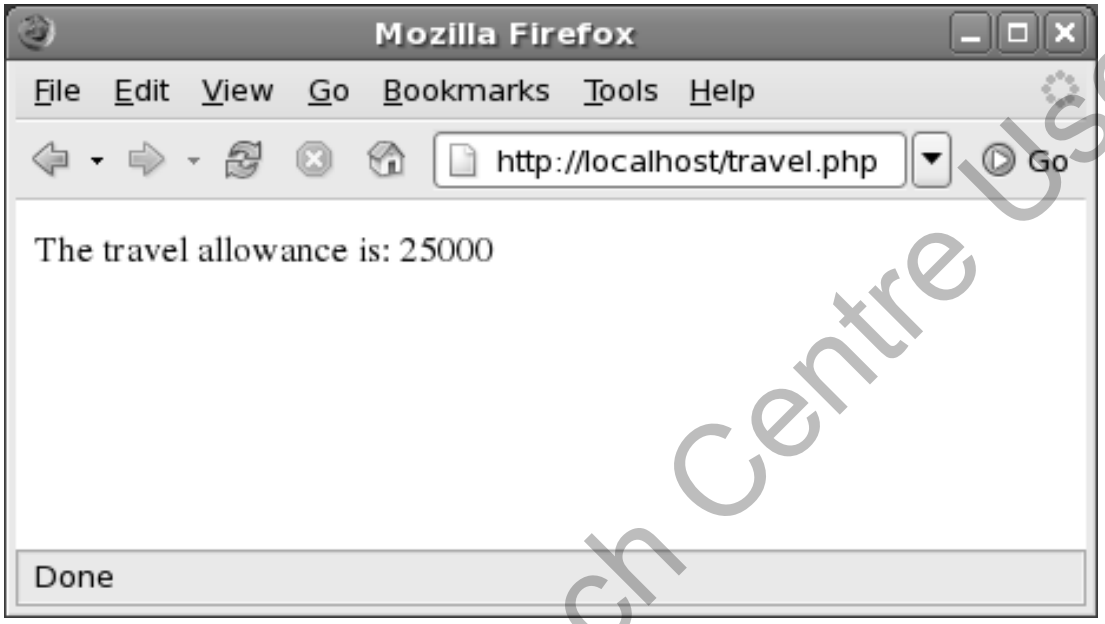
Setting Default Values

- **travel.php** - Assigning default values for an argument

Snippet

```
<?php
//Creating a function and assigning default value to
the argument
function T_ALLOWANCE($BASIC_SAL=100000)
{
//Calculating the travel allowance and storing it in
a variable
$T_ALLOWANCE=0.25*$BASIC_SAL;
//Displaying text
echo "The travel allowance is: $T_ALLOWANCE";
}
//Executing the function
T_ALLOWANCE();
?>
```

Displays the following output:



The `T_ALLOWANCE()` function calculates and displays the traveling allowance.

Returning Values from Functions

- The `return` statement in a function returns the value from the function
- The value returned can be an array or an object
- The `return` keyword causes the function to stop execution and pass the control to the line from which it was invoked
- The reference operator is required to be used while declaring a function as well as when assigning the returned value to the variable

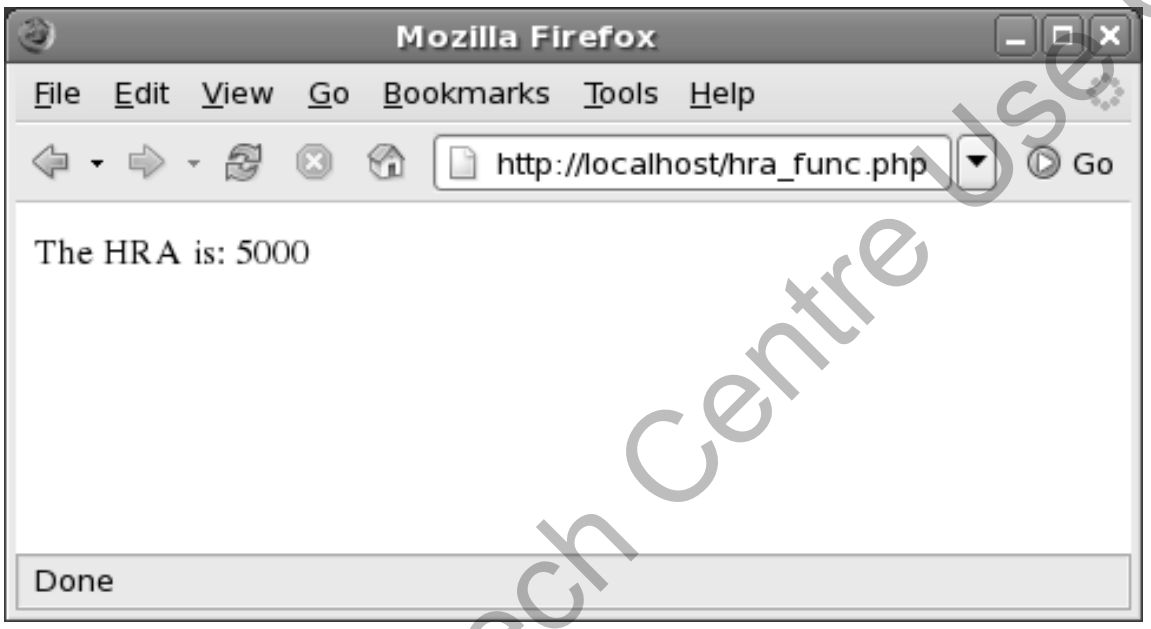
- **hra_func.php -** The use of return keyword to calculate the house rent allowance

Snippet

```
<?php

//Creating a function

function HRA($Basic_Sal)

{

//Calculating the HRA and storing it in a variable

$HRA=0.25*$Basic_Sal;

//Returning the value stored in the variable

return $HRA;

}
```

```
//Storing the output of the function in a variable after setting a
//default value

$B=HRA(20000);

//Displaying the text

echo "The HRA is: ";

//Displaying the output

echo $B;

?>
```

Displays the following output



The `HRA()` function returns the house rent allowance, calculated at the rate of 25% of basic salary.

Nesting of Functions

- One function can be dependent on another function in the program
- The execution of one function inside another function is called nesting of functions

- **nested.php** - Use of nested functions

Snippet

```
<?php
//Assigning value to variable
$Basic_Sal=75000;
//Creating a function and passing value by reference
function HRA($Basic_Sal)
{
//Calculating the HRA
$HRA=3/10 * $Basic_Sal;
//Displaying Text
echo "Your HRA is: ";
//Displaying the computed HRA
echo $HRA;
echo "<br>";
}
//Creating a function and passing value by reference
function TA($Basic_Sal)
{
```

Nesting of Functions

```
//Calculating the TA

$TA=1/4*$Basic_Sal;

//Displaying Text

echo "Your TA is: ";

//Displaying the computed TA

echo $TA;

echo "<br>";

}

//Creating a function and passing value by reference

function TAX($Basic_Sal)

{

//Calculating the tax

$TAX=1/10*$Basic_Sal;

//Displaying Text

echo "Your TAX is: ";
```

Nesting of Functions

```
//Displaying the computed tax
echo $TAX;
echo "<br>";
}
//Creating a function and passing value by reference

function Net_Salary($Basic_Sal)
{
//Storing tax, HRA and TA in variables
$A=3/10 * $Basic_Sal;
$B=1/4*$Basic_Sal;
$C=1/10*$Basic_Sal;
//Calculating the Net Salary
```

Nesting of Functions

```
$Net_Sal=75000+$A+$B-$C;

//Displaying Text

echo "Your Net Salary is:";

//Displaying the Net Salary

echo $Net_Sal;

}

//Calling the functions

HRA($Basic_Sal);

echo "<br>";

TA($Basic_Sal);

echo "<br>";

TAX($Basic_Sal);

echo "<br>";
```

Nesting of Functions

```
Net_Salary($Basic_Sal);

echo "<br>";

?>
```

Displays the following output:

The `Net_Salary()` function calls three functions, `HRA()`, `TA()`, and `TAX()`.

Recursion

- The execution of a function within another function is called nested functions
- When a function executes itself repeatedly it is known as recursion
- When a function calls itself, the same block of code is executed

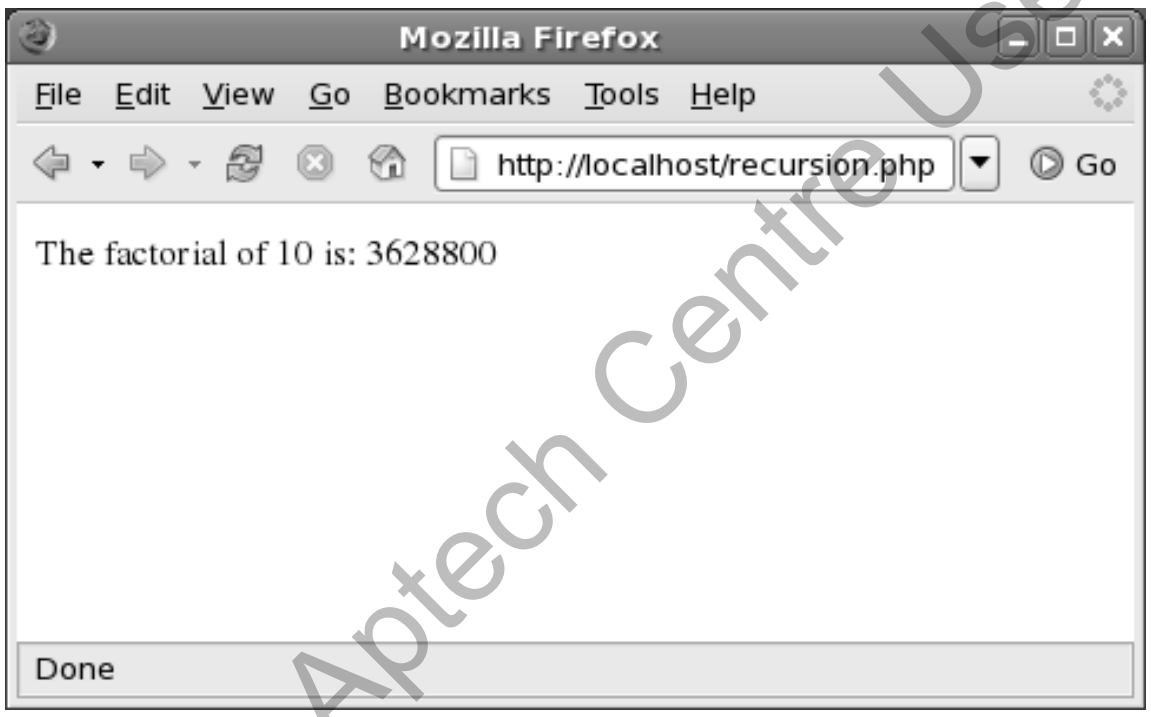
- **recursion.php** - Use of recursive functions in programs to calculate the factorial

Snippet

```
<?php

//Assigning Value to a variable

$A=10;

//Creating a function to calculate the factorial

function factorial($A)

{

//Calculating the factorial

if($A<=1)

{
```

```
return 1;
}
else
{
return $A * factorial($A-1);
}
}
//Displaying Text
echo "The factorial of $A is: ";
//Assigning the result to a variable
$B = factorial($A);
//Displaying the result
echo $B;
?>
```

Displays the following output:



The `factorial()` function returns 1 if the input is 0 or 1.

Summary

- Functions can be used to implement execution of repetitive lines of code
- The built-in functions in PHP are mathematical, string, date and time, error handling, database, array and mail functions
- Mathematical functions operate on numerical data
- String functions operate on character type of data
- Date and time functions are used to display the system date and time
- Error handling functions are used to define the error handling rules
- Arguments can be passed to a function by value and reference. Default values can also be passed to a function
- Recursive functions have the ability to call themselves repeatedly