Unit 3: Working with Functions

1.1 Unit Structure

- Learning Objectives
- Introduction
- PHP Function
- User Defined function
- Built in function.
- Math/Numeric function
- String function
- Date function
- File Inclusion function
- File I/O operation function

1.2 Learning Objectives

After studying this unit student should be able to:

- Understand various user defined and built in functions.
- Understand mathematical and string function.
- Perform I/O operation on file.

1.3 PHP Functions

- A function is a reusable piece or block of code that performs some specific tasks.
- Functions can either return values when called or can simply perform an operation without returning any value.

We can relate functions in programs to employees in an office in real life for a better understanding of how functions work. Suppose the boss wants his employee to calculate the annual budget. The employee will take information about the statistics, calculate the budget, and show the result to his boss. Functions work in a similar manner. They take information as parameters, executes a block of statements or perform operations on these parameters and returns the result.

1.3.1 Types of functions:

- ***Built-in functions:*** PHP provides us with a huge collection of built-in library functions. These functions are already coded and stored in the form of functions. To use those we just need to call them as per our requirement like, var_dump, fopen(), print_r(), gettype() and so on.
- ***User Defined Functions:*** Apart from the built-in functions, PHP allows us to create our own customized functions called the user-defined functions. Using this we can create our own packages of code and use it wherever necessary by simply calling it.

1.3.2 Why should we use functions?

- ***Reusability***: If we have a common code that we would like to use at various parts of a program, we can simply contain it within a function and call it whenever required. This reduces the time and effort of repetition of a single code. This can be done both within a program and also by importing the PHP file, containing the function, in some other program.
- ***Easier error detection***: Since our code is divided into functions, we can easily detect in which function the error could lie and fix them fast and easily.
- ***Easily maintained***: As we have used functions in our program, so if anything, or any line of code needs to be changed, we can easily change it inside the function and the change will be reflected everywhere, where the function is called. Hence, easy to maintain.

1.3.3 Factors to take into consideration while creating a function

While creating a user defined function we need to keep few things in mind:

1. Any name ending with an open and closed parenthesis is a function.
2. A function name always begins with the keyword function.
3. To call a function we just need to write its name followed by the parenthesis.
4. A function name cannot start with a number. It can start with an alphabet or underscore.
5. A function name is not case-sensitive.

In fact, you hardly need to create your own PHP function because there are already more than thousands of built-in library functions created for different area and you just need to call them according to your requirement.

1.4 Built in Functions

Built in functions are functions that exist in PHP installation package. These built in functions are what make PHP a very efficient and productive scripting language.

PHP is very rich in terms of Buil-in functions. The built-in functions can be classified into many categories. Here is the list of some important function categories in PHP.

- Array Functions
- Calendar Functions
- Class/Object Functions
- Character Functions
- Date & Time Functions
- Directory Functions
- Error Handling Functions
- File System Functions
- MySQL Functions
- Network Functions
- ODBC Functions
- String Functions
- SimpleXML Functions
- XML Parsing Functions

There are various other function categories which are not covered here. You can see all the functions by clicking on the following link:

https://www.php.net/manual/en/language.functions.php

1.5 User-Defined Function

1.5.1 About user Defined Functions:

- A function is a block of statements that can be used repeatedly in a program.
- A function will not execute immediately when a page loads.
- A function will be executed by a call to the function.

1.5.2 Why use User Defined Functions?

General purpose of user-defined function:

- You have routine tasks in your application such as adding data to the database.
- Performing validation checks on the data.
- Authenticating users in the system etc.

These activities will be spread across a number of pages. Creating a function that all these pages can be called is one of the features that make PHP a powerful scripting language. Before we create our first user defined function, let's discuss the rules that we must follow when creating user defining functions.

1.5.3 Rules for creating user defining function:

- Function names **must start with a letter or an underscore** but **NOT a number**.
- The function name **must be unique across the entire project** development.
- The **function name must not contain spaces**.
- Functions can optionally accept parametric and non-parametric function and return values too.

1.5.4 Syntax of User Defined function:

A user-defined function declaration starts with the word "function".

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

Note: Give the function a name that reflects what the function does. Function names are NOT case-sensitive.

1.5.5 How to create User Defined function

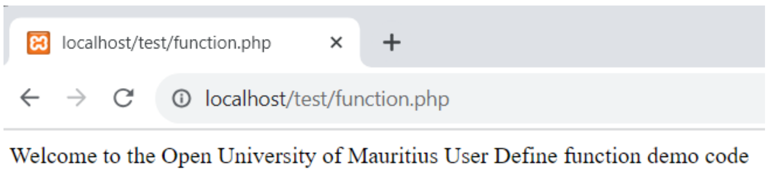
It's very easy to create your own PHP function. Suppose that you want to create a PHP function which will simply write a simple message on your browser. The following example creates a function called PrintWelcome() and then calls it just after creating it.

Note that while creating a function, its name should start with keyword function and all the PHP code should be put inside {Business Logic} braces as shown in the following example below:

Example of User Define function:

```
<?php
function PrintWelcome () {
echo "Welcome to the Open University of Mauritius User Define function demo code";
}
PrintWelcome (); // call the function
?>
```

In the above example, we created a function named "PrintWelcome()". The opening curlybrace ({) indicates the beginning of the function code and the closing curly brace (}) indicates the end of the function. The function outputs "Welcome to the Open University of Mauritius User Define function demo code". To call the function, just write its name:



```
Welcome to the Open University of Mauritius User Define function demo code
```

Figure 1: PHP User Define Function Example Output

1.5.6 Types of User-Defined function:

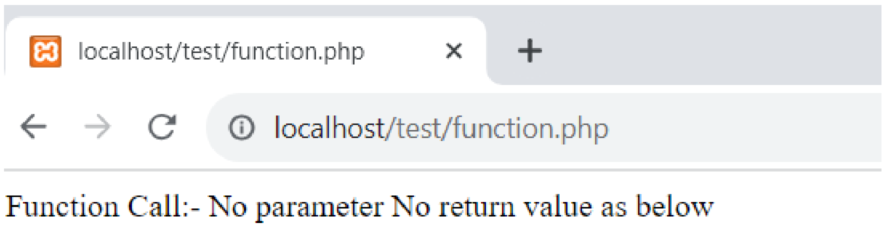
- No parameter, No return value.
- No parameter, Get return value.
- Pass parameter, No return value.
- Pass parameter, Get return value.

PHP Function No parameter, No return value.

The function can display value as per business logical code.

Example:

```
<?php
function NPNR() {
    echo "Function Call:- No parameter No return value as below ";
}
NPNR(); // call the function
?>
```



```
Function Call:- No parameter No return value as below
```

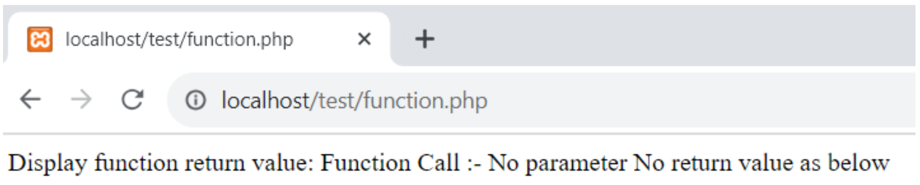
Figure 2 User Define Function NPNR output.

PHP Function No parameter Get return value.

The function will return value after executing the business logical code using **return** keyword.

Example:

```
<?php
function NPGR() {
    return ("Function Call :- No parameter No return value as below");
}
$return_value=NPGR(); // call the function
echo "Display function return value: ".$return_value;
?>
```



```
Display function return value: Function Call :- No parameter No return value as below
```

Figure 3 PHP User Define Function NPGR output.

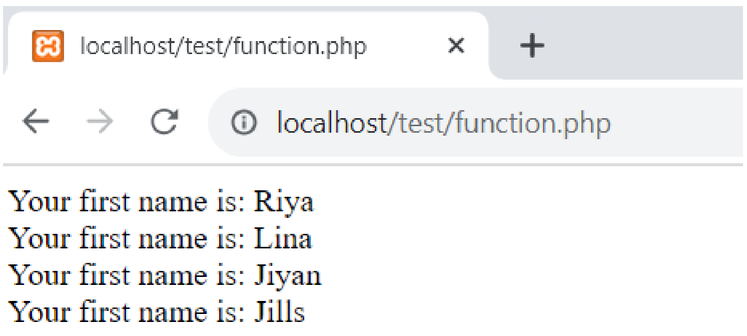
PHP Function Arguments/Parameter (Pass parameter, No return value)

- Information can be passed to functions through arguments.
- An argument is just like a variable.
- Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

The following example has a function with one argument ($FirstName). When the DisplayName() function is called, we also pass along a name (e.g. Riya), and the name is used inside the function, which outputs several different first names, but an equal last name:

Example:

```
<?php
function DisplayName ($FirstName) {
        echo "Your first name is: $FirstName<br>";
}
DisplayName("Riya");
DisplayName("Lina");
DisplayName("Jiyan");
DisplayName("Jills");
?>
```



```
Your first name is: Riya
Your first name is: Lina
Your first name is: Jiyan
Your first name is: Jills
```

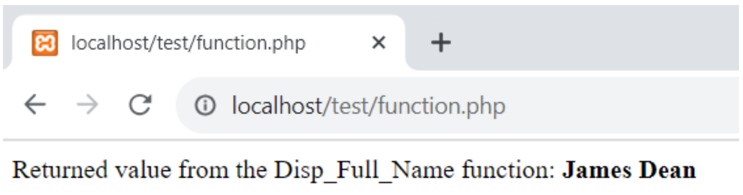
Figure 4: PHP User Define Function with argument.

PHP Function Pass parameter Get Return value:

A function can return a value using the return statement in conjunction with a value or object. Return stops the execution of the function and sends the value back to the calling code.

The following example takes two string parameters and concatenates them together and then returns their full name to the calling program. Note that return keyword is used to return a value from a function.

```
<?php
function Disp_Full_Name($First_Name, $Last_Name) {
        $Full_Name = $First_Name." ".$Last_Name;
        return $Full_Name;
}
$Ret_Full_Name = Disp_Full_Name("James", "Dean");
echo    "Returned value from the Disp_Full_Name function: <b>$Ret_Full_Name</b>";
?>
```



```
Returned value from the Disp_Full_Name function: James Dean
```

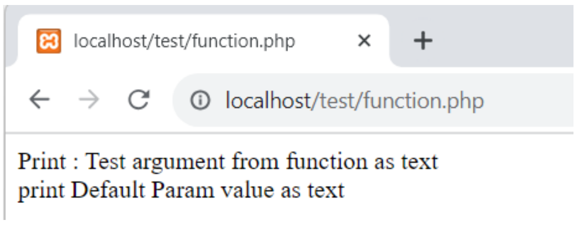
Figure 5: PHP User Define Function with argument and return value

1.5.7 Setting Default Values for Function Parameters

You can set a parameter to have a default value if the function's caller doesn't pass it.

The following function prints "print Default Param value as text" in case no value is passed to this function.

```
<?php
function printMe($param = "print Default Param value as text") {
	echo $param."<br>";
}
printMe("Print : Test argument from function as text");
printMe();
?>
```



```
Print : Test argument from function as text
print Default Param value as text
```

Figure 6: PHP User Define Function Default Parameter output

1.6 BUILT IN FUNCTION

- Are functions that exist in PHP installation package. PHP comes standard with many functions and constructs.
- These functions are what make PHP a very efficient and productive scripting language.
- Can be classified into many categories.

1.6.1 Math/Numeric Function

- Numeric functions are functions that return numeric results.
- Numeric PHP function can be used to format numbers, return constants, perform mathematical computations etc.

Examples of some common PHP numeric/math functions

Function	**Description**	**Example**	**Output**

is_number	Accepts an argument and returns true if its numeric and false if it's not	**Example 1**
		`<?php`
		`if(is_numeric("Welcome"))`
		`{`
		`echo "true";`
		`}`
		`else`
		`{`
		`echo "false";`
		`}`
		`?>`
			False

		Example 2
		`<?php`
		`if(is_numeric (123))`
		`{`
		`echo "true";`
		`}`
		`else`
		`{`
		`echo "false";`
		`}`
		`?>`
			true

number_format	Used to formats a numeric value using digit separators and decimal points	<?php echo number_format(2509663); ?>	2,509,663
rand	Used to generate a random number.	<?php echo rand(); ?>	Random number
round	Round off a number with decimal points to the nearest whole number.	<?php echo round(3.6555); ?>	4
sqrt	Returns the square root of a number	<?php echo sqrt(100); ?>	10
cos	Returns the cosine	<?php echo cos(45); ?>	0.52532199
sin	Returns the sine	<?php echo sin(45); ?>	0.85090352
tan	Returns the tangent	<?php echo tan(45); ?>	1.61977519
pi	Constant that returns the value of PI	<?php echo pi(); ?>	3.14159265

1.6.2 String Function

What is a string?

- A string is a ***collection of characters.***
- String is one of the data types supported by PHP.
- The string variables can contain alphanumeric characters.
- You declare variable and assign string characters to it.
- You can directly use them with echo statement.

Defining PHP strings variable with value:

The simplest way to create a string is to use **single quotes**.

For example:

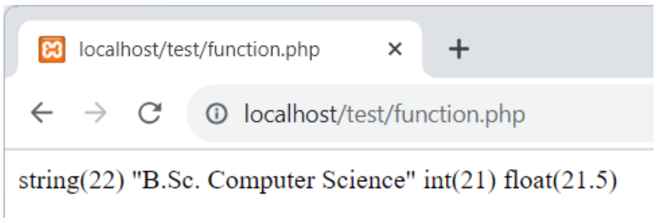
```
 <?php
$str = ‘B.Sc. Computer Science’;
?>
```

You can also create a string by using **double quotes**:

```
<?php
$str= “B.Sc. Computer Science”;
?>
```

Checking data type of a variable:

```
<?php
var_dump("B.Sc. Computer Science");
var_dump(21);
var_dump(21.5);
?>
```



```
string(22) "B.Sc. Computer Science" int(21) float(21.5)
```

Figure 7: Output of PHP_String_Datatype script

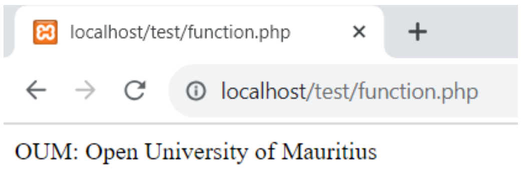
Create Strings: Double Quote V/S Single Quote:

The double quotes are used to create relatively complex strings compared to single quotes. Variable names can be used inside double quotes and their values will be displayed.

Let’s look at an example.

```
<?php
$Uni_Name= 'OUM';
echo "$Uni_Name: Open University of Mauritius ";
?>
```

In the above example, we created a simple string variable “$Uni_Name” with the value of ‘OUM’. The variable name is then used in the string created using double quotes and its value is interpolated at run time.



```
OUM: Open University of Mauritius
```

Figure 8: PHP_String_DoubleQuotes output

In addition to variable interpolations, the double quote string can also escape more special characters such as “\n for a linefeed, \$ dollar for the dollar sign” etc.

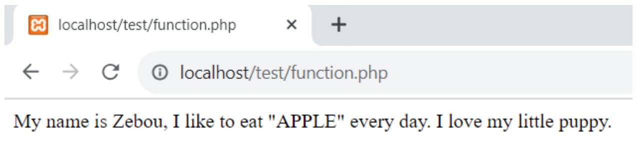
PHP Heredoc

- The heredoc methodology is used to create complex strings as compared to double quotes.
- The heredoc supports all the features of double quotes and allows creating string values with more than one line without PHP string concatenation.
- Using double quotes to create strings that have multiple lines generates an error.
- You can also use double quotes inside without escaping them.
- The example below illustrates how the Heredoc method is used to create string values.

```
<?php
$baby_name = "Zebou";
echo <<<EOT
        My name is $baby_name,
        I like to eat "APPLE" every day.
        I love my little puppy.
EOT;
?>
```

<<<EOT is the string delimiter. EOT is the acronym for end of text. It should be defined at the beginning of the string and at the end.

Note: you can use anything you like in place of EOT



```
My name is Zebou, I like to eat "APPLE" every day. I love my little puppy.
```

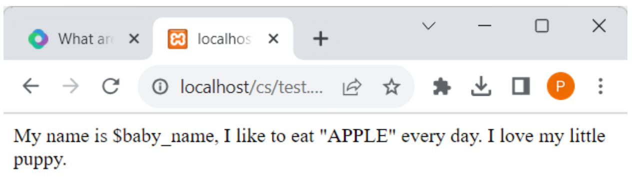
Figure 9: Output of PHP_Heredoc script

PHP Nowdoc

- The Nowdoc string creation method is similar to the heredoc method but works like the way single quotes work. That is, it does not print any variable values as compared to double quotes.
- No parsing takes place inside the Nowdoc.
- Nowdoc is ideal when working with raw data that does not need to be parsed.
- To denote a Nowdoc, we just need to ***use single quotes around the opening identifier***.
- The code below shows the Nowdoc implementation.

```
<?php
$baby_name = "Zebou";
echo <<<'EOT'
My name is $baby_name,
I like to eat "APPLE" every day.
I love my little puppy.
EOT;
?>
```

The output from the above code is as follows:



```
My name is $baby_name, I like to eat "APPLE" every day. I love my little
puppy.
```

Figure: Output of Nowdoc

Note: See the single quotes around the opening identifier 'EOT' in the code. It is used to denote the Nowdoc command. In the output, you will see that the variable name "$baby_name" has not been parsed in the Nowdoc command.

PHP string functions that are used to manipulate string values.

We are now going to look at some of the commonly used string functions in PHP:

Function	Description	Example	Output
strtolower	Used to convert all string characters to lower case letters	<?php echo strtolower('OUM'); **?>**	oum
strtoupper	Used to convert all string characters to upper case letters	<?php echo strtoupper('Mauritius'); **?>**	MAURITIUS
strlen	The string length function is used to count the number of characters in a string. Spaces in between characters are also counted.	<?php echo strlen('Open University of Mauritius'); ?>	28
explode	Used to convert strings into an array variable	<?php $settings = explode(';', "host=localhost; db=oum; uid=root; pwd=demo"); print_r($settings); ?>	Array ([0] => host=localhost [1] => db=oum [2] => uid=root [3] => pwd=demo)
substr	Used to return part of the string. It accepts three (3) basic parameters. 1. The string to be shortened. 2. The position of the starting point. 3. The number of characters to be returned.	<?php $my_var = 'This is a really long sentence that I wish to cut short'; Echo substr($my_var,0,12).'...'; ?>	This is a re...
str_replace	Used to locate and	<?php	that mobile in

	replace specified string values in a given string. The function accepts three arguments: 1. The text to be replaced. 2. The replacement texts. 3. The text that is analyzed.	echo str_replace ('the', 'that', 'the mobile in the shop is very expensive'); ?>	that shop is very expensive
strpos	Used to locate and return the position of a character(s) within a string. This function accepts two arguments: 1. The String 2. The character(s)	<?php echo strpos('PHP Programing','Pro'); ?>	4
sha1	Used to calculate the SHA-1 hash of a string value	<?php echo sha1('password'); ?>	5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8
md5	Used to calculate the md5 hash of a string value	<?php echo md5('password'); ?>	5f4dcc3b5aa765d61d8327deb882cf99
str_word_count	Used to count the number of words in a string.	<?php echo str_word_count ('This is a really long sentence that I want to count the number of words'); ?>	15
ucfirst	Make the first character of a string value upper case	<?php echo ucfirst('manchester'); ?>	Manchester
lcfirst	Make the first	<?php	mAN

	character of a string value lower case	echo lcfirst('MAN'); ?>	

Some function list with description (For reference only – Not Examinable):

Function	Description
addcslashes()	Returns a string with backslashes in front of the specified characters
addslashes()	Returns a string with backslashes in front of predefined characters
bin2hex()	Converts a string of ASCII characters to hexadecimal values
chop()	Removes whitespace or other characters from the right end of a string
chr()	Returns a character from a specified ASCII value
chunk_split()	Splits a string into a series of smaller parts
convert_cyr_string()	Converts a string from one Cyrillic character-set to another
convert_uudecode()	Decodes a uuencoded string
convert_uuencode()	Encodes a string using the uuencode algorithm
count_chars()	Returns information about characters used in a string
crc32()	Calculates a 32-bit CRC for a string
crypt()	One-way string hashing
echo()	Outputs one or more strings
explode()	Breaks a string into an array
fprintf()	Writes a formatted string to a specified output stream
get_html_translation_table()	Returns the translation table used by htmlspecialchars() and htmlentities()
hebrev()	Converts Hebrew text to visual text
hebrevc()	Converts Hebrew text to visual text and new lines (\n) into
hex2bin()	Converts a string of hexadecimal values to ASCII characters
html_entity_decode()	Converts HTML entities to characters
htmlentities()	Converts characters to HTML entities
htmlspecialchars_decode()	Converts some predefined HTML entities to characters
htmlspecialchars()	Converts some predefined characters to HTML entities
implode()	Returns a string from the elements of an array
join()	Alias of implode()
lcfirst()	Converts the first character of a string to lowercase
levenshtein()	Returns the Levenshtein distance between two strings
localeconv()	Returns locale numeric and monetary formatting information
ltrim()	Removes whitespace or other characters from the left side of a string
md5()	Calculates the MD5 hash of a string
md5_file()	Calculates the MD5 hash of a file
metaphone()	Calculates the metaphone key of a string
money_format()	Returns a string formatted as a currency string
nl_langinfo()	Returns specific local information
nl2br()	Inserts HTML line breaks in front of each newline in a string
number_format()	Formats a number with grouped thousands
ord()	Returns the ASCII value of the first character of a string
parse_str()	Parses a query string into variables
print()	Outputs one or more strings
printf()	Outputs a formatted string
quoted_printable_decode()	Converts a quoted-printable string to an 8-bit string
quoted_printable_encode()	Converts an 8-bit string to a quoted printable string

quotemeta()	Quotes meta characters
rtrim()	Removes whitespace or other characters from the right side of a string
setlocale()	Sets locale information
sha1()	Calculates the SHA-1 hash of a string
sha1_file()	Calculates the SHA-1 hash of a file
similar_text()	Calculates the similarity between two strings
soundex()	Calculates the soundex key of a string
sprintf()	Writes a formatted string to a variable
sscanf()	Parses input from a string according to a format
str_getcsv()	Parses a CSV string into an array
str_ireplace()	Replaces some characters in a string (case- insensitive)
str_pad()	Pads a string to a new length
str_repeat()	Repeats a string a specified number of times
str_replace()	Replaces some characters in a string (case-sensitive)
str_rot13()	Performs the ROT13 encoding on a string
str_shuffle()	Randomly shuffles all characters in a string
str_split()	Splits a string into an array
str_word_count()	Count the number of words in a string
strcasecmp()	Compares two strings (case-insensitive)
strchr()	Finds the first occurrence of a string inside another string (alias of strstr())
strcmp()	Compares two strings (case-sensitive)
strcoll()	Compares two strings (locale based string comparison)
strcspn()	Returns the number of characters found in a string before any part of some specified characters are found
strip_tags()	Strips HTML and PHP tags from a string
stripcslashes()	Unquotes a string quoted with addcslashes()
stripslashes()	Unquotes a string quoted with addslashes()
stripos()	Returns the position of the first occurrence of a string inside another string (case-insensitive)
stristr()	Finds the first occurrence of a string inside another string (case-insensitive)
strlen()	Returns the length of a string
strnatcasecmp()	Compares two strings using a "natural order" algorithm (case-insensitive)
strnatcmp()	Compares two strings using a "natural order" algorithm (case-sensitive)
strncasecmp()	String comparison of the first n characters (case- insensitive)
strncmp()	String comparison of the first n characters (case- sensitive)
strpbrk()	Searches a string for any of a set of characters
strpos()	Returns the position of the first occurrence of a string inside another string (case-sensitive)
strrchr()	Finds the last occurrence of a string inside another string
strrev()	Reverses a string
strripos()	Finds the position of the last occurrence of a string inside another string (case-insensitive)
strrpos()	Finds the position of the last occurrence of a string inside another string (case-sensitive)
strspn()	Returns the number of characters found in a string that contains only characters from a specified charlist
strstr()	Finds the first occurrence of a string inside another string (case-sensitive)
strtok()	Splits a string into smaller strings
strtolower()	Converts a string to lowercase letters
strtoupper()	Converts a string to uppercase letters

strtr()	Translates certain characters in a string
substr()	Returns a part of a string
substr_compare()	Compares two strings from a specified start position (binary safe and optionally case-sensitive)
substr_count()	Counts the number of times a substring occurs in a string
substr_replace()	Replaces a part of a string with another string
trim()	Removes whitespace or other characters from both sides of a string
ucfirst()	Converts the first character of a string to uppercase
ucwords()	Converts the first character of each word in a string to uppercase
vfprintf()	Writes a formatted string to a specified output stream
vprintf()	Outputs a formatted string
vsprintf()	Writes a formatted string to a variable
wordwrap()	Wraps a string to a given number of characters

1.6.3 Date Time Function

What is PHP Date Function?

PHP date function is an in-built function that simplifies working with date data types. The PHP date function is used to format a date or time into a human readable format. It can be used to display the date of articles, news, blogs and updates published. It can also record the last updated date and time or timestamp of a data in a database.

PHP Date Syntax & Example

```
<?php
date (format, [timestamp]);
?>
```

The syntax and parameters of the Date Function:

- "date(…….)" is the function that returns the current time on the server.
- "format" is the general format which we want our output to be, for example:
  - "Y-m-d" for PHP date format YYYY-MM-DD.
  - "Y" to display the current year.
- "[timestamp]" is optional. If no timestamp has been provided, PHP will get the use the PHP current date time on the server.

For example, the date function that displays the current year.

```
<?php
echo date("Y");
?>
```

The output is 2023.

What is a TimeStamp?

A timestamp is a numeric value in seconds between the current time and value as from 1st January, 1970 00:00:00 Greenwich Mean Time (GMT).

- The value returned by the time function depends on the default time zone.
- The default time zone is set in the php.ini file.
- It can also be set programmatically using date_default_timezone_set function.

The code below displays the current time stamp.

```
<?php
echo time();
?>
```

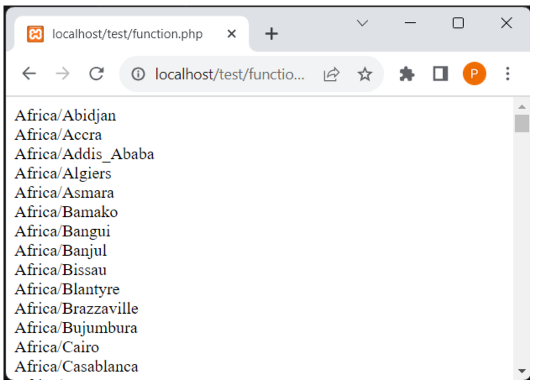
The output is 1698049933.

Note: the value of the timestamp is not constant. It changes every second.

Getting a list of available time zone identifiers:

Before looking at how to set the default time zone programmatically, let's look at how to get a list of supported time zones.

```
<?php
$timezone_identifiers = DateTimeZone::listIdentifiers();
foreach($timezone_identifiers as $key => $list){
	echo $list . "<br>";
}
?>
```



```
Africa/Abidjan
Africa/Accra
Africa/Addis_Ababa
Africa/Algiers
Africa/Asmara
Africa/Bamako
Africa/Bangui
Africa/Banjul
Africa/Bissau
Africa/Blantyre
Africa/Brazzaville
Africa/Bujumbura
Africa/Cairo
Africa/Casablanca
```

Figure 10: output of PHP_TimeZone world time zone list.

Scroll down and find Mauritius time zone as (Indian/Mauritius)

- "$timezone_identifiers = DateTimeZone::listIdentifiers();" calls the listIdentifiers static method of the DateandTime Zone built in class.
- The listIdentifiers method returns a list of constants that are assigned to the variable $timezone_identifiers.
- "foreach{…}" iterates through the numeric array and prints the values.

PHP – How to set Time zone Programmatically?

The date_default_timezone_set function allows you to set the default time zone from a PHPscript. The set time zone will then be used by all date PHP function scripts. It has the following syntax.

```
<?php
date_default_timezone_set( string $timezone_identifier );
?>
```

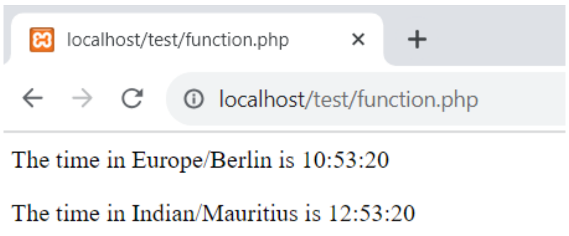
- “date_default_timezone_set()” is the function that sets the default time zone.
- “string $timezone_identifier” is the time zone identifier.

The script below displays the time according to the default time zone set in php.ini. It then changes the default time zone to Indian/Mauritius and displays the time again.

```
<?php

echo "The time in " .date_default_timezone_get() . " is " . date("H:i:s");
date_default_timezone_set("Indian/Mauritius");
echo "<p>The time in " .date_default_timezone_get() . " is " . date("H:i:s");

?>
```



```
The time in Europe/Berlin is 10:53:20

The time in Indian/Mauritius is 12:53:20
```

Figure 11: PHP_SetTimeZone output

PHP Mktime Function:

The mktime function returns the timestamp in a Unix format.

```
<?php
mktime(hour, minute, second, month, day, year, is_dst);
?>
```

Brief about mktime parameters:

- "mktime(…)" is the make PHP timestamp function
- "hour" is optional, it is the number of hours.
- "minute" is optional, it is the number of minutes.
- "second" is optional, it is the number of seconds.
- "month" is optional, it is the number of the months.
- "day" is optional, it is the number of the days.
- "year" is optional, it is the number of the years.
- "is_dst" is optional, it is used to determine the day saving time (DST). 1 is for DST, 0 if it is not and -1 if it is unknown.

Parameter	Description	Example
Time Parameters		
"r"	Returns the full date and time	<?php echo date("r"); ?>

"a","A"	Returns whether the current time is a.m. or p.m., A.M or P.M respectively	<?php
		echo date("a");
		echo date("A");
		?>

"g","G"	Returns the hour without leading zeroes [1 to 12], [0 to 23] respectively	<?php
		echo date("g");
		echo date("G");
		?>

"h","H"	Returns the hour with leading zeros [01 to 12],[00 to 23] respectively	<?php
		echo date("h");
		echo date("H");
		?>

"i","s"	Returns the minutes/seconds with leading zeroes [00	<?php

	to 59]	echo date("i");
		echo date("s");
		?>

	Day parameters	
Parameter	Description	Example
“d”	Returns the day of the month with leading zeroes [01 to 31]	<?php echo date("d"); ?>
“j”	Returns the day of the month without leading zeroes [1 to 31]	<?php echo date("j"); ?>
“D”	Returns the first 3 letters of the day name [Sun to Sat]	<?php echo date("D"); ?>
“l”	Returns day name of the week [Sunday to Saturday]	<?php echo date("l"); ?>
“w”	Returns day of the week without leading zeroes [0 to 6] Sunday is represent by zero (0) through to Saturday represented by six (6)	<?php echo date("w"); ?>
“z”	Returns the day of the year without leading spaces [0 through to 365]	<?php echo date("z"); ?>
	Month Parameters	
“m”	Returns the month number with leading zeroes [01 to 12]	<?php echo date("m"); ?>
“n”	Returns the month number without leading zeroes [01 to 12]	<?php echo date("n"); ?>
“M”	Returns the first 3 letters of the month name [Jan to Dec]	<?php echo date("M"); ?>
“F”	Returns the month name [January to December]	<?php echo date("F");

		?>
“t”	Returns the number of days in a month [28 to 31]	<?php echo date("t"); ?>
Year Parameters		
“L”	Returns 1 if it’s a leap year and 0 if it is not a leap year	<?php echo date("L"); ?>
“Y”	Returns four digit year format	<?php echo date("Y"); ?>
“y”	Returns two (2) digits year format (00 to 99)	<?php echo date("y"); ?>

In the date function we learned,

- The date function is used to format the timestamp into a human desired format.
- The timestamp is the number of seconds between the current time and 1st January 1970, 00:00:00 GMT. It is also known as the UNIX timestamp.
- All date functions use the default time zone set in the php.ini file.
- The default time zone can also be set programmatically using PHP scripts.

1.6.4 File Inclusion Function

There are two PHP functions which can be used to include one PHP file into another PHP file.

1. The include() Function
2. The require() Function

- The include (or require) statement takes all the text/code/markup that exists in the specified file and copies it into the file that uses the include statement.
- Including files is very useful when you want to include the same PHP, HTML, or text on multiple pages of a website.

This is a strong point of PHP which helps in creating functions, headers, footers, or elements that can be reused on multiple pages. This will help developers to make it easy to change the layout of the complete website with minimal effort. If there is any change required, then instead of changing thousands of files, just change the included file.

It is possible to insert the content of one PHP file into another PHP file (before the server executes it), with the include or require statement.

Difference between include and require function:

The include and require statements are identical, except upon failure:

- require will produce a fatal error (E_COMPILE_ERROR) and stop the script.
- include will only produce a warning (E_WARNING) and the script will continue.

So, if you want the execution to go on and show users the output, even if the included file is missing, use the include statement.

Otherwise, in case of Framework, CMS, or a complex PHP application coding, always use the require statement to include a key file to the flow of execution. This will help avoid compromising your application's security and integrity, just in case one key file is accidentally missing.

Including files saves a lot of work. This means that you can create a standard header, footer, or menu file for all your web pages. Then, when the header needs to be updated, you can only update the header include file.

Syntax:

include 'filename';

or

require 'filename';

The include() Function:

The include() function takes all the text in a specified file and copies it into the file that uses the include function. If there is any problem in loading a file, then the include() function generates a warning but the script will continue execution.

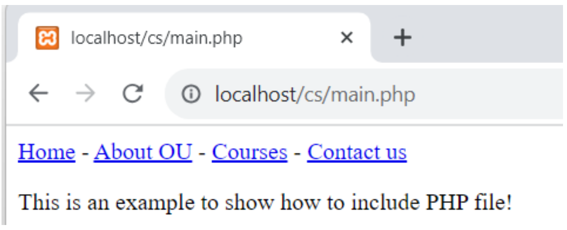
Assume you want to create a common menu for your website. Then create a file OUM_menu.php with the following content.

```
<!--// save script as OUM_Menu.php -->
<a href="index.php">Home</a> -
<a href="About_us.php">About BAOU</a> -
<a href="courses.php">Courses</a> -
<a href="contact.php">Contact us</a><br />
```

Now create as many pages as you like and include this file to create header. For example, now your PHP_Include.php file can have the following content.

```
<html><?php // Script save as php_include.php ?>
 <body>
<?php include("OU_Menu.php"); ?>
<p>This is an example to show how to include PHP file!</p>
</body>
</html>
```

It will produce the following result:



```
Home - About OU - Courses - Contact us

This is an example to show how to include PHP file!
```

Figure12 PHP_include script output

The require() Function

The require() function takes all the text in a specified file and copies it into the file that uses the include function. If there is any problem in loading a file then the require() function generates a fatal error and halts the execution of the script.

So there is no difference between require() and include() except they handle error conditions. It is recommended to use the require() function instead of include(), because scripts should not continue executing if files are missing or misnamed.

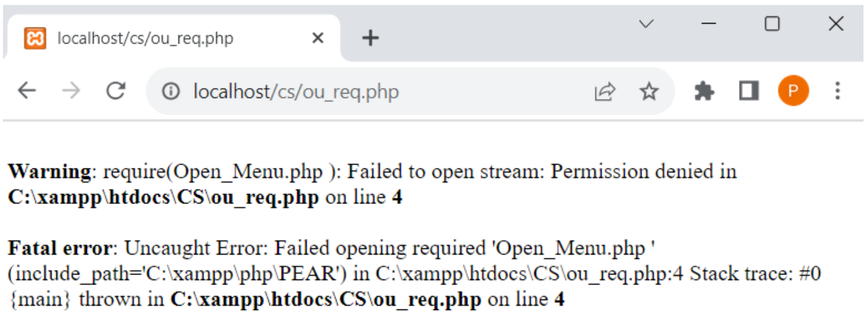
You can try using the above example with require() function and it will generate same result.

Example where file does not exist.

Now let's try the same example with require() function.

```
<html>
<?php // Script save as PHP_Require.php ?>
<body>
<?php require("Open_Menu.php "); ?>
<p>This is an example to show how to include wrong PHP file!</p>
</body>
</html>
```

The Output is as follows:



```
Warning: require(Open_Menu.php ): Failed to open stream: Permission denied in
C:\xampp\htdocs\CS\ou_req.php on line 4

Fatal error: Uncaught Error: Failed opening required 'Open_Menu.php '
(include_path='C:\xampp\php\PEAR') in C:\xampp\htdocs\CS\ou_req.php:4 Stack trace: #0
{main} thrown in C:\xampp\htdocs\CS\ou_req.php on line 4
```

Figure13 Required file missing.

1.6.5 File I/O Operation Function

What is a File?

A file is simply a resource for storing information on a computer. Files are usually used to store information such as configuration settings of a program, simple data such as contact names against the phone numbers, images, pictures, photos, etc.

Basic File functions:

- PHP File Formats Support
- PHP files Functions
- PHP File_exists Function
- PHP Fopen Function
- PHP Fwrite Function
- PHP Fclose Function
- PHP Fgets Function
- PHP Copy Function
- Deleting a file
- PHP File_get_contents Function

Files provide a permanent cost-effective data storage solution for simple data compared to databases that require other software and skills to manage DBMS systems.

PHP files Functions

- PHP provides a convenient way of working with files via its rich collection of built-in functions.
- Operating systems such as Windows and MAC OS are not case sensitive while Linux or Unix operating systems are case sensitive.
- Adopting a naming conversion such as lower-case letters only for file naming is a good practice that ensures maximum cross platform compatibility.

Let's now look at some of the most commonly used PHP file functions.

PHP File_exists Function.

This function is used to determine whether a file exists or not. This function comes handy when we want to know if a file exists or not before processing it. You can also use this function when creating a new file and you want to ensure that the file does not already exist on the server.

The file_exist function has the following syntax.

```
<?php
file_exists($filename);
?>
```

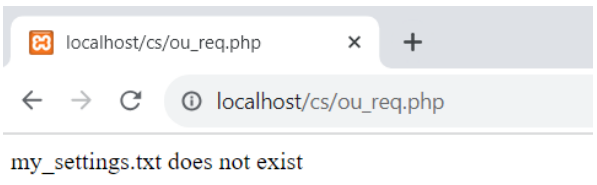
Explanation:

"file_exists" is the PHP function that returns true if the file exists and false if it does not exist.

"$filename" is the path and name of the file to be checked.

The code below uses file_exists function to determine if the file my_settings.txt exists.

```
<?php
if (file_exists('my_settings.txt')){
echo 'file found!';
}
else {
echo ' my_settings.txt does not exist';
}
?>
```



```
my_settings.txt does not exist
```

Figure 14: Output of file check

PHP fopen Function

The fopen function is used to open files. It has the following syntax:

```
<?php
        fopen($file_name,$mode,$use_include_path,$context);
?>
```

Explanation:

- "fopen" is the PHP open file function.
- "$file_name" is the name of the file to be opened.
- "$mode" is the mode in which the file should be opened, the table below shows the modes:

Mode	Description
▪ **r**	▪ Read file from beginning. ▪ Returns false if the file doesn't exist. ▪ Read only.
▪ **r+**	▪ Read file from beginning. ▪ Returns false if the file doesn't exist. ▪ Read and write.
▪ **w**	▪ Write to file at beginning. ▪ Truncate file to zero length. ▪ If the file doesn't exist attempt to create it. ▪ Write only.
▪ **w+**	▪ Write to file at beginning, truncate file to zero length. ▪ If the file doesn't exist attempt to create it. ▪ Read and Write.
▪ **a**	▪ Append to file at end. ▪ If the file doesn't exist attempt to create it. ▪ Write only.
▪ **a+**	▪ PHP append to file at end. ▪ If the file doesn't exist attempt to create it. ▪ Read and write.

- "$use_include_path" is optional, default is false, if set to true, the function searches in the include path too.
- "$context" is optional, can be used to specify the context support.

PHP fwrite Function.

The PHP fwrite function is used to write to an open file. It has the following syntax:

```
<?php
fwrite($handle, $string, $length);
?>
```

Explanation:

- "fwrite" is the PHP function for writing to files.
- "$handle" is the file pointer resource.
- "$string" is the data to be written in the file.
- "$length" is optional, can be used to specify the maximum file length.

PHP fclose Function

PHP fclose Function is used to close a file which is already open.

```
<?php
fclose($handle);
?>
```

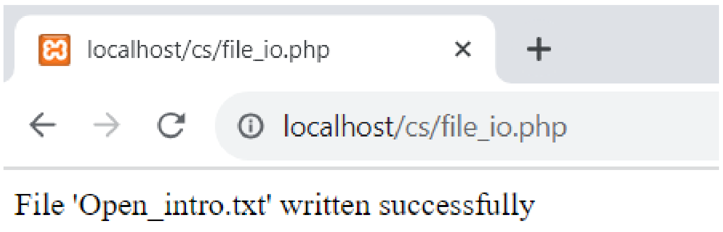
Explanation:

- "fclose" is the PHP function for closing an open file
- "$handle" is the file pointer resource.

Let's now look at an example that creates Open_intro.txt. We will use the fopen, fwrite, fclose functions.

The code below implements the above example.

```
<?php
        $fh = fopen("Open_intro.txt", 'w') or die("Failed to create file");
        $text = "Open University File – Testing file creation";
        fwrite($fh, $text) or die("Could not write to file");
        fclose($fh);
        echo "File 'Open_intro.txt' written successfully";
?>
```



```
File 'Open_intro.txt' written successfully
```

Figure 15: Output confirming that file has been created successfully.

Note: if your disk is full or you do not have permission to write files, you will get an error message. Kindly refresh the same URL.

PHP fgets Function.

PHP fgets Function is used to read PHP files line by line. It has the following basic syntax:

```
<?php
fgets($handle)
?>
```

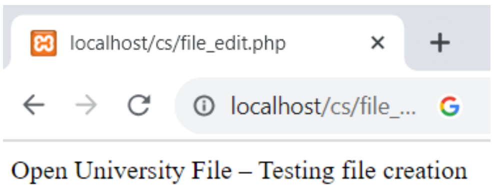
Explanation:

- “$fgets” is the PHP function for reading file lines.
- “$handle” is the file pointer resource.

Let’s now look at an example that reads Open_intro.txt file using the fopen and fgets functions.

```
<?php
$fh = fopen("Open_intro.txt", 'r') or die("File does not exist or you lack permission to open it");
$line = fgets($fh);
echo $line;
fclose($fh);
?>
```

The output is as follows:



```
Open University File – Testing file creation
```

Figure 16: Reading the Open_intro.txt file.

- "fopen" function returns the pointer to the file specified in the file path.
- "die()" function is called if an error occurs. It displays a message and exits execution of the script in case of errors.

PHP Copy Function

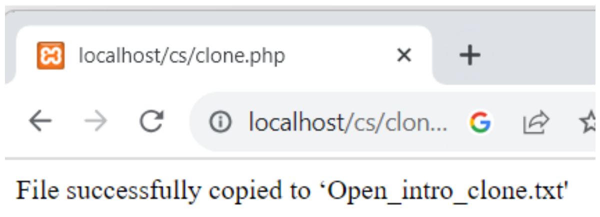
The PHP copy function is used to copy files. It has the following basic syntax.

```
<?php
copy($file, $copied_file);
?>
```

- "$file" specifies the file path and name of the file to be copied.
- "copied_file" specified the path and name of the copied file. The code below illustrates the implementation.

```
<?php
copy('Open_intro.txt', 'Open_intro_clone.txt') or die("Could not copy file");
echo "File successfully copied to 'Open_intro_clone.txt'";
?>
```

Output:



```
File successfully copied to 'Open_intro_clone.txt'
```

Figure 17: Cloning the Open_intro.txt file.

PHP unlink() function - to delete a file

The unlink function is used to delete the file. The code below illustrates the implementation.

```
<?php
if (!unlink('Open_intro _clone.txt')){
echo "Could not delete file";
}
Else {
echo "File 'Open_intro _clone.txt' successfully deleted";
}
?>
```

PHP file_get_contents Function

PHP file_get_contents function is used to read the entire file contents.

The difference between file_get_contents and fgets is that file_get_contents returns the file data as a string while fgets reads the file line by line.

The code below illustrates the implementation.

```
<?php
echo "<pre>"; // Enables display of line feeds
echo file_get_contents("Open_intro.txt");
echo "</pre>"; // Terminates pre tag
?>
```

Conclusion of File I/O function:

- A file is a resource for storing data
- PHP has a rich collection of built in functions that simplify working with files.
- Common file functions include fopen, fclose, file_get_contents

The table below shows a summary of the functions covered with respect to file.

Function	Description
file_exists	Used to determine if a file exists or not
fopen	Used to open a file. Returns a pointer to the opened file
fwrite	Used to write to files
fclose	Used to open closed files
fgets	Used to read a file line by line
copy	Used to copy an existing file
unlink	Used to delete an existing file
file_get_contents	Used to return the contents of a file as a string