Architecting Web Applications using PHP

Session 7

Working with Functions in PHP

Session Overview

In this session, learners would be able to:

- Describe PHP built-in functions
- Define PHP user-defined functions and explain how to create a user-defined function in PHP
- Describe PHP function arguments
- Identify the purpose of PHP default argument values
- Elaborate on return values in PHP functions
- Describe return type declarations
- Outline how to pass arguments by reference
- Identify the use of named arguments
- Define dynamic function calls - `date()` and `time()`

Functions in PHP

Reusable code

- PHP functions are reusable and can be invoked many times within the same program.

Minimal code

- Using functions gives the flexibility to write code once and reuse the same whenever required. This reduces the amount of written code and also reduces time taken to write code.

Clarity of code

- PHP functions segregate the programming logic. This allows the user to comprehend application flow as code is split into different functions.

PHP Built-in Functions

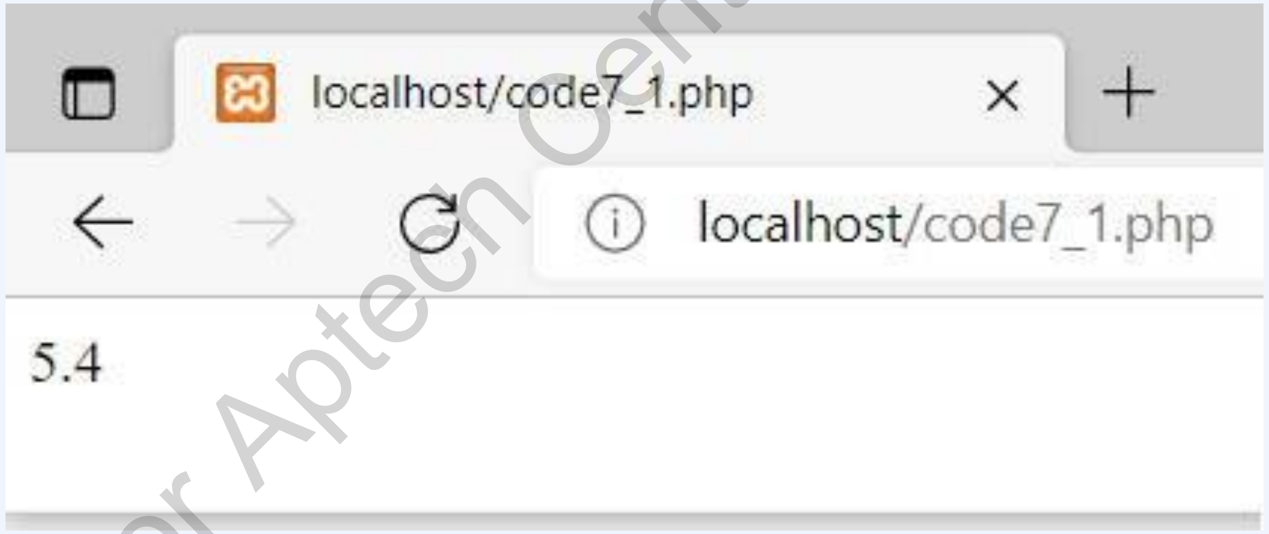
Some of the important PHP functions which will be covered in the session are as follows:

- `abs()`
- `gettype()`
- `var_dump()`

`abs()` Function

Code Snippet:

```
<?php
echo abs(-5.4);
?>
```



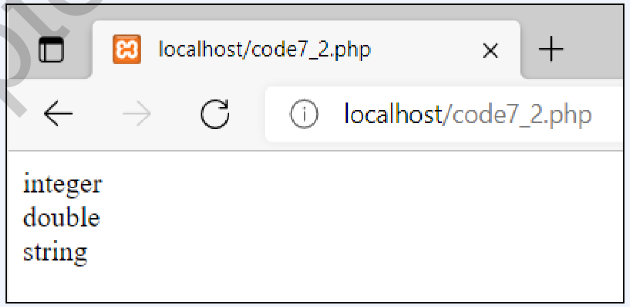
```
5.4
```

Figure: Output for Code Snippet

gettype() Function

Code Snippet:

```
<?php
// PHP program to illustrate gettype() function
$var1 = 3; // integer value
$var2 = 5.6; // double value
$var3 = "Abc3462"; // string value
echo gettype($var1) . "<br>";
echo gettype($var2). "<br>";
echo gettype($var3). "<br>";
?>
```



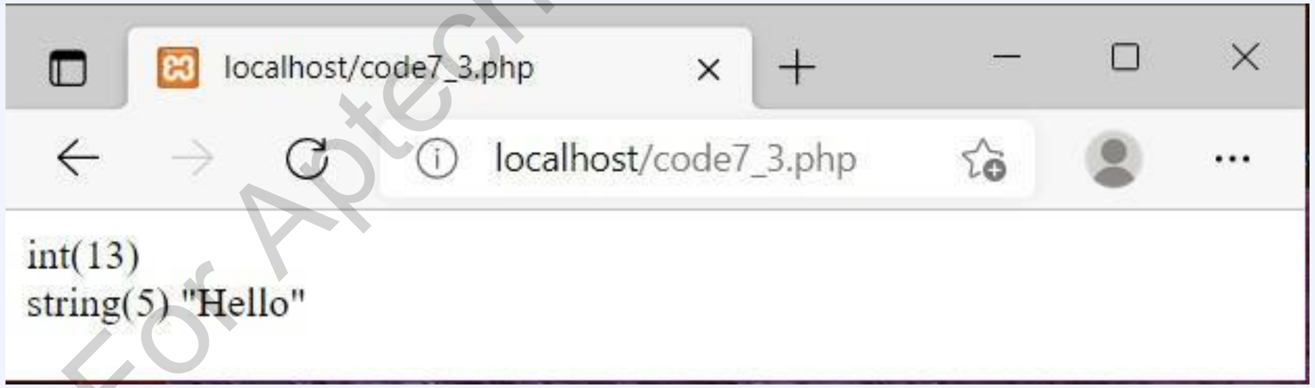
```
integer
double
string
```

Figure: Output for Code Snippet

var_dump() Function

Code Snippet:

```
<?php
$var1=13;
$var2="Hello";
var_dump($var1);
echo "<br>";
var_dump($var2);
?>
```



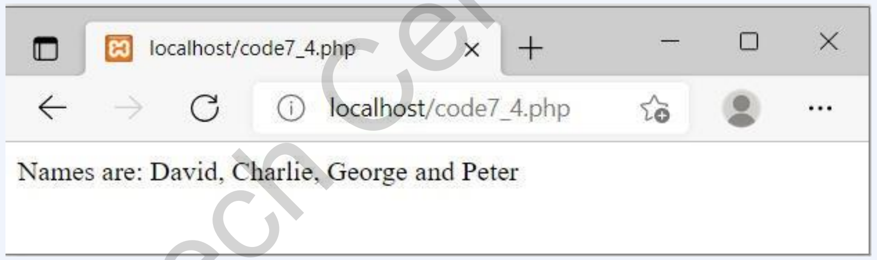
```
int(13)
string(5) "Hello"
```

Figure: Output for Code Snippet

PHP Array Functions

Code Snippet:

```
<?php
$names=array("David","Charlie","George","Peter");
echo "Names are: $names[0], $names[1], $names[2] and $names[3]";
?>
```



```
Names are: David, Charlie, George and Peter
```

Figure: Output for Code Snippet

There are three types of arrays that can be created in PHP:

Indexed Arrays
Associative Arrays
Multidimensional Arrays

Keys and Values

Function	Description
`array_change_key_case(array $array, int $case)`	This function switches all the keys in an array to lowercase or uppercase.
`array_chunk(array $array, int $length, bool $preserve_keys)`	This function breaks an array into smaller chunks of arrays.
`array_column(array $array, int|string|null $column_key, int|string|null $index_key)`	This function displays values from the column mentioned.
`array_combine(array $keys, array $values)`	This function generates an array utilizing elements from keys array and values array.
`array_count_values(array $array)`	This function counts all the values of an array.
`array_diff(array $array, array ...$arrays)`	This function displays differences after comparing the arrays. It compares only values.
`array_diff_assoc(array $keys, array $values)`	This function also displays differences, but compares both the keys and values.
`array_diff_key(array $array, array ...$arrays)`	This function also displays differences, but compares only the keys.
`array_fill(int $start_index, int $count, mixed $value))`	This function helps in inserting values to an array.
`array_filter(array $array, ?callable $callback, int $mode))`	This function helps in filtering the array values utilizing a callback function.
`array_flip(array $array)`	This function interchanges all the keys with their corresponding values in an array
`array_intersect(array $array, array ...$arrays)`	This function compares the values in arrays and displays the matches.
`array_map(?callable $callback, array $array, array ...$arrays)`	This function transfers each value of an array to a user-made function, which in turn displays new values.
`array_key_exists(string|int $key, array $array)`	This function checks the array for the mentioned key.
`array_keys(array $array)`	This function displays all the keys of an array.
`array_merge(array ...$arrays)`	This function combines one or more arrays into a single array.

Table: Array Functions

PHP `string` Functions [1-3]

PHP `string` functions help the user to perform different operations on strings. They are as follows:

- `strrev()` function
- `strtolower()` function
- `str_repeat()` function

PHP `string` Functions [2-3]

`strrev()` function

- This function accepts a string as the argument and displays the original string in reverse order.
- For example, `echo strrev("Hello, World!");`. This prints `!dlroW ,olleH`.

`strtolower()` function

- This function helps users to change an argument string into all lowercase letters.
- For example, echo `strtolower("HeLLo");`. This prints `hello`.

`str_repeat()` function

- The function takes two arguments - first is a string and the second is a number. It outputs the argument string iterating it by the count mentioned in the second argument.
- For example, `echo str_repeat("hi", 10);`. This prints `hihihihihihihihihihi`.

PHP string Functions [3-3]

Function	Description
sprintf()	This function writes a formatted string to a variable.
sscanf()	This function parses input from a string as per a format.
strcasecmp()	This function performs a comparison of the two given strings and it is case-insensitive.
strchr()	This function determines the first instance of a string within another string.
strcmp()	This function performs a comparison of the two given strings and it is case-sensitive.
strcoll()	This function performs a comparison of the two given strings and it is case-insensitive.
stripos()	This function displays the position of the first instance of a string within another string. It is case-insensitive.
stristr()	This function displays the first instance of a string within another string. It is case-insensitive.
strlen()	This function displays the length of a string.
strpbrk()	This function looks for a given set of characters in a string.
strpos()	This function displays the position of the first instance of a string within another string. It is case-sensitive.
strrev()	This function displays a given string in the reverse order.
strripos()	This function displays the position of the last instance of a string within another string. It is case-insensitive.
strrpos()	This function displays the position of the last instance of a string within another string. It is case-sensitive.
strspn()	This function displays the number of characters present in a string having only characters from a particular charlist.
substr_count()	This function displays the total number of times a substring appears in a string.
wordwrap()	This function wraps a string to a specified number of characters.
vsprintf()	This function writes a formatted string to a variable.

Table: String Functions

PHP `stream` Functions [1-2]

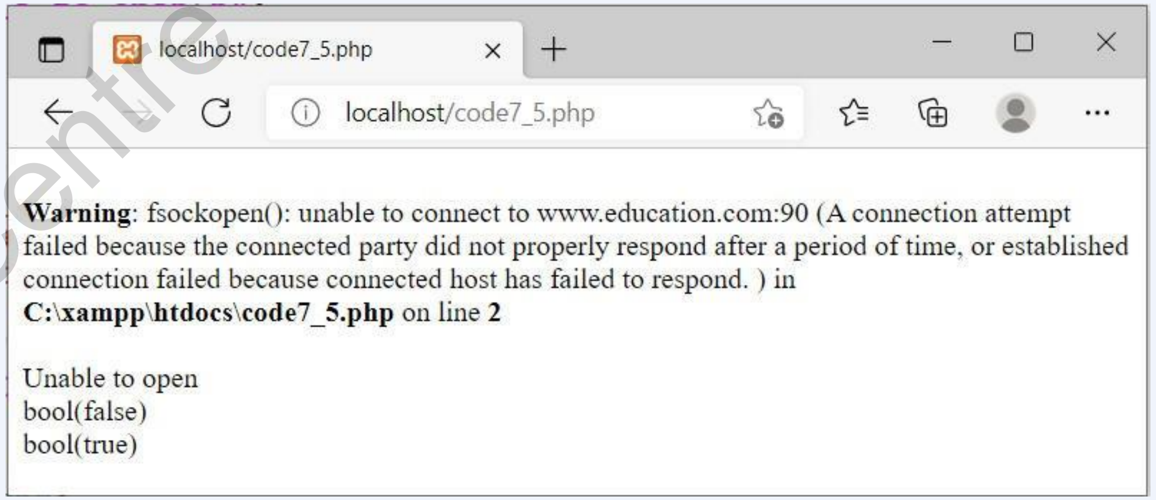
Function	Description
`stream_bucket_append()`	This function joins the bucket to the brigade.
`stream_bucket_make_writeable()`	This function displays a bucket object from the brigade to further work on.
`stream_socket_server()`	This function generates an Internet or Unix domain server socket.
`stream_supports_lock()`	This function informs if the stream has locking or not.
`stream_wrapper_register()`	This function records a URL wrapper set up as a PHP class.
`stream_socket_shutdown()`	This function stops a full-duplex connection.
`stream_wrapper_restore()`	This function helps to restore a built-in wrapper that was previously not registered.
`stream_wrapper_unregister()`	This function helps to unregister a URL wrapper.
`stream_copy_to_stream()`	This function copies information from one stream to another stream.
`stream_is_local()`	This function inspects whether a stream is a local stream or not.

Table: stream Functions

PHP `stream` Functions [2-2]

Code Snippet:

```
<?php
$fp = fsockopen("www.education.com", 80);
if (!$fp) {
    echo "Unable to open\n";
}
else {
    fwrite($fp, "GET / HTTP/1.0\r\n\r\n");
    stream_set_timeout($fp, 2);
    $res = fread($fp, 2000);
    $info = stream_get_meta_data($fp);
    fclose($fp);
    if ($info['timed_out']) {
        echo 'Connection timed out!';
    }
else {
        echo $res;
    }
}
var_dump(stream_is_local("http://education.com"));
echo "<br>";
var_dump(stream_is_local("/etc"));
?>
```



```
Warning: fsockopen(): unable to connect to www.education.com:90 (A connection attempt
failed because the connected party did not properly respond after a period of time, or established
connection failed because connected host has failed to respond. ) in
C:\xampp\htdocs\code7_5.php on line 2

Unable to open
bool(false)
bool(true)
```

Figure: Output for Code Snippet

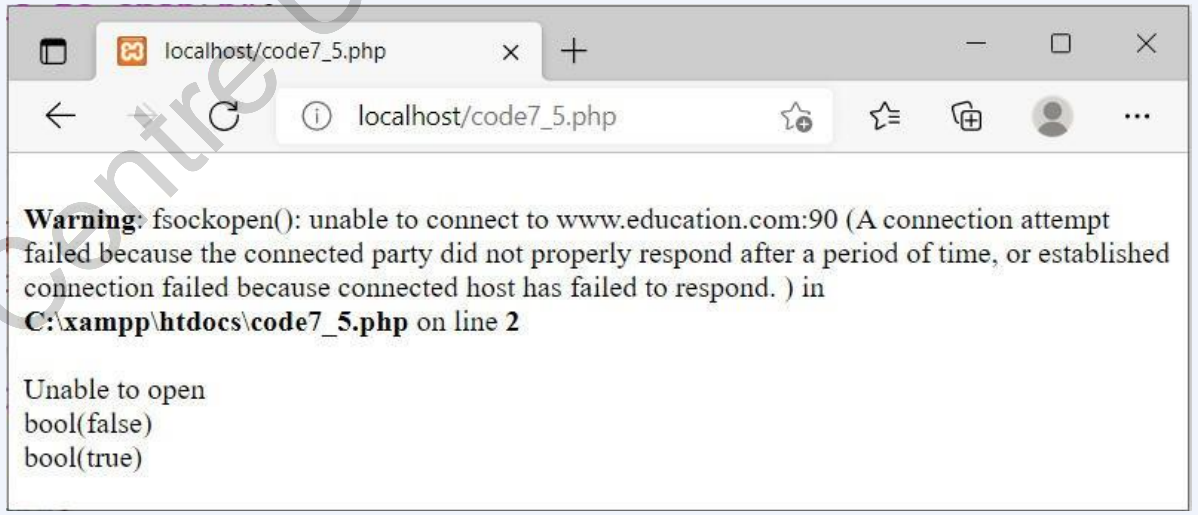
PHP User-defined Functions [1-2]

- While PHP has numerous built-in functions, it also gives users flexibility to create their own functions.
- Unlike built-in functions which are readily available, user-defined functions are created by users.

PHP User-defined Functions [2-2]

Code Snippet:

```
<?php
function even_number()
{
   for( $i=0; $i<=10; $i++ )
{
    if( $i%2 == 0 ){
        echo "<br>", $i;
    }
}
}
even_number();
?>
```



```
Warning: fsockopen(): unable to connect to www.education.com:90 (A connection attempt
failed because the connected party did not properly respond after a period of time, or established
connection failed because connected host has failed to respond. ) in
C:\xampp\htdocs\code7_5.php on line 2

Unable to open
bool(false)
bool(true)
```

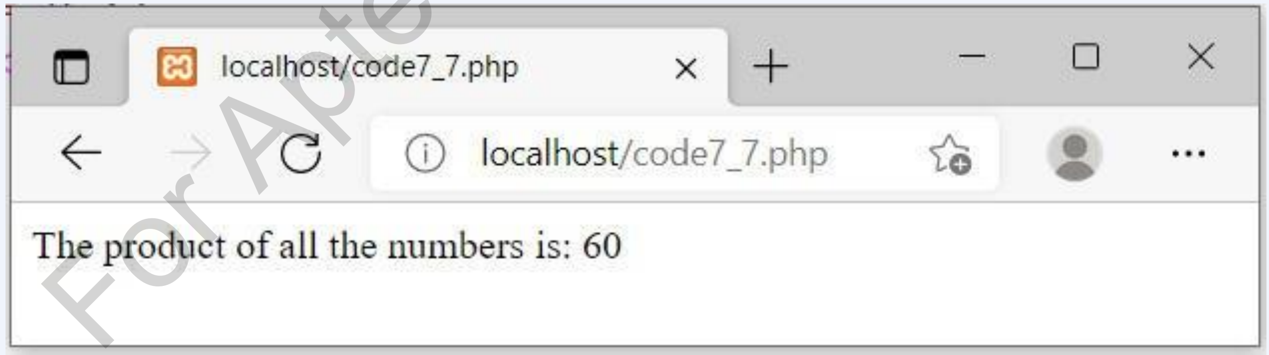
Figure: Output for Code Snippet

PHP Function Arguments and Parameters

Code Snippet:

```
<?php
function numbers($num1, $num2, $num3)
{
    $product = $num1 * $num2 * $num3;
    echo "The product of all the numbers is: $product";
}

// Calling the function
// Passing three arguments
numbers(4, 3, 5);
?>
```



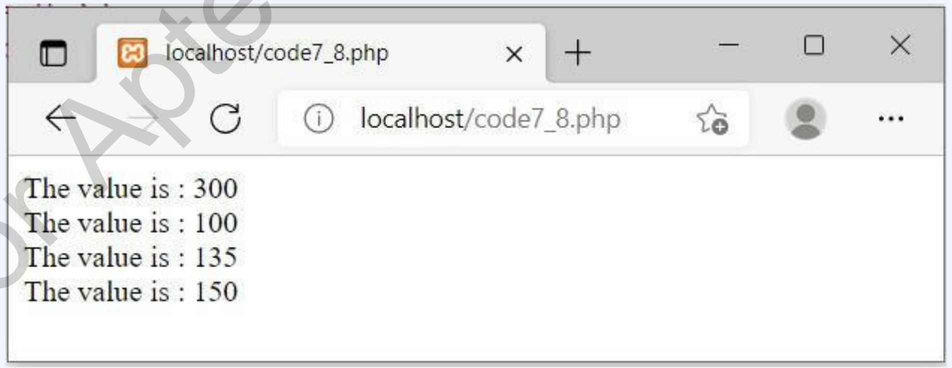
```
The product of all the numbers is: 60
```

Figure: Output for Code Snippet

PHP Default Argument Value

Code Snippet:

```
<?php declare(strict_types=1); // strict requirement ?>
<body>
<?php
function value(int $Val_default = 100) {
  echo "The value is : $Val_default <br>";
}
value(300);
value();
value(135);
value(150);
?>
```



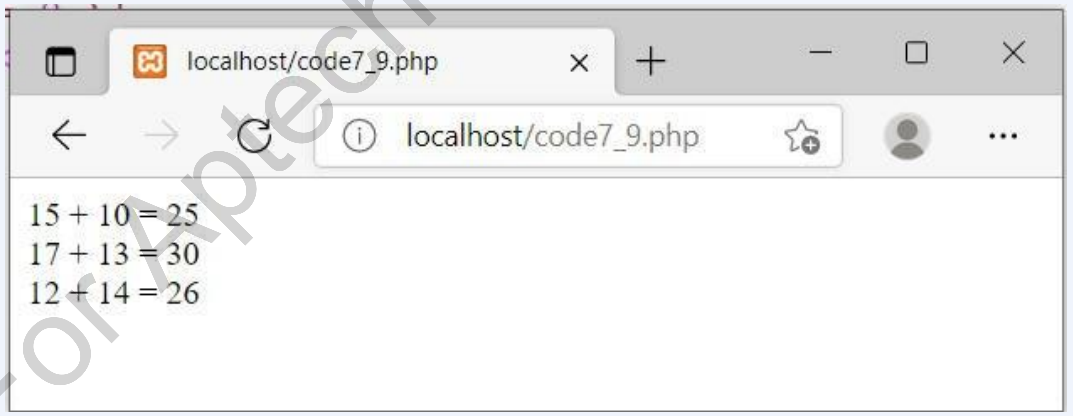
```
The value is : 300
The value is : 100
The value is : 135
The value is : 150
```

Figure: Output for Code Snippet

PHP Functions - Returning Values

Code Snippet:

```
<?php
function Add_Numbers(int $a, int $b) {
  $c = $a + $b;
  return $c;
}
echo "15 + 10 = " . Add_Numbers(15,10) . "<br>";
echo "17 + 13 = " . Add_Numbers(17,13) . "<br>";
echo "12 + 14 = " . Add_Numbers(12,14);
?>
```



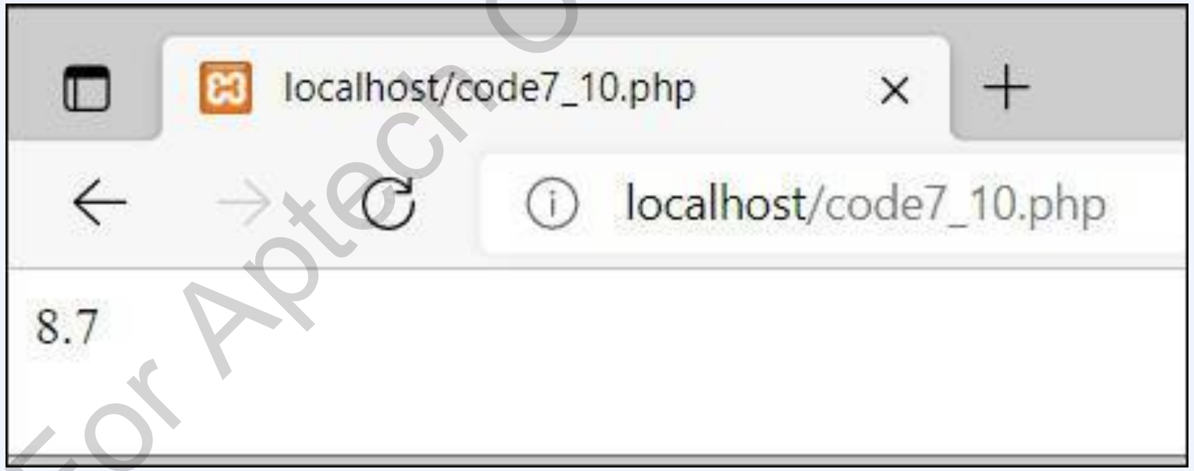
```
15 + 10 = 25
17 + 13 = 30
12 + 14 = 26
```

Figure: Output for Code Snippet

PHP Return Type Declarations [1-2]

Code Snippet:

```
<?php declare(strict_types=1); // strict requirement
function AddNumbers(float $n1, float $n2) : float {
  return $n1 + $n2;
}
echo AddNumbers(1.5, 7.2);
?>
```



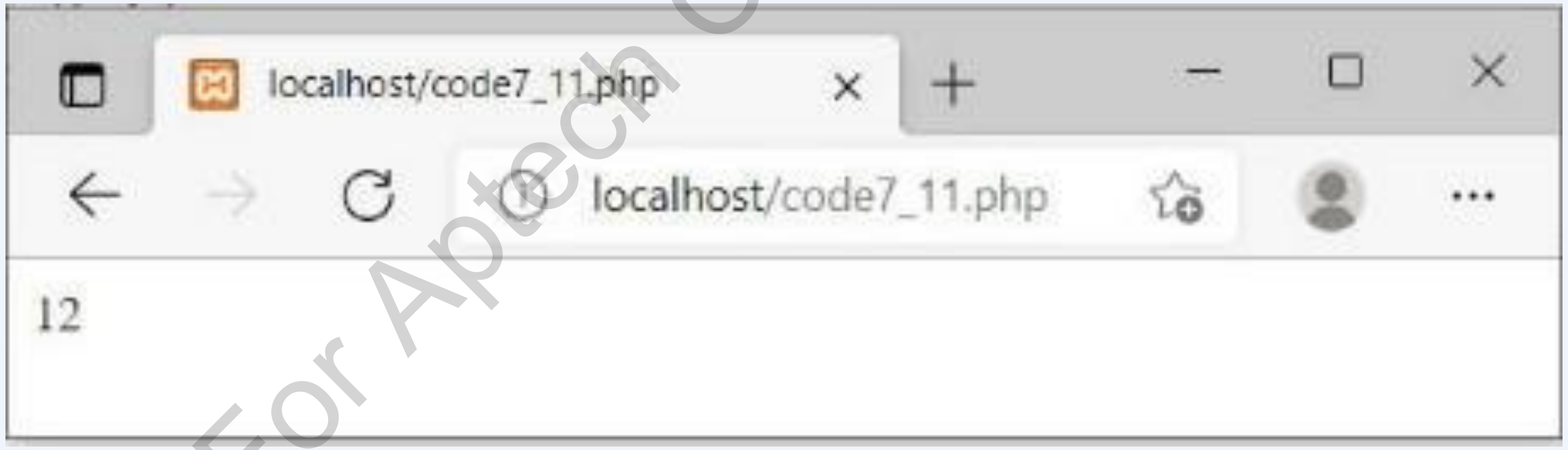
```
8.7
```

Figure: Output for Code Snippet

PHP Return Type Declarations [2-2]

Code Snippet:

```
<?php declare(strict_types=1); // strict requirement
function SumNumbers(float $x1, float $x2) : int {
  return (int)($x1 + $x2);
}
echo SumNumbers(5.2, 7.2);
?>
```



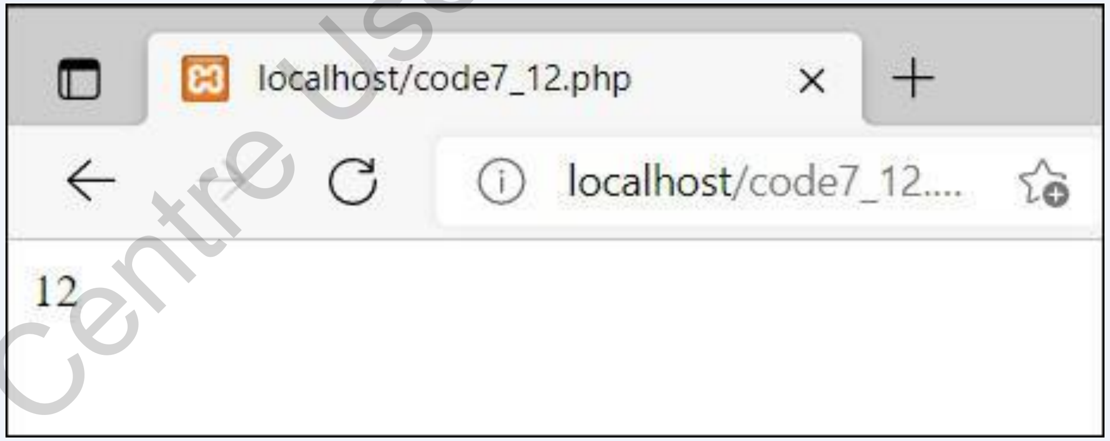
```
12
```

Figure: Output for Code Snippet

Passing Arguments by Reference

Code Snippet:

```
<?php
function add(&$value1) {
  $value1 += 10;
}
$num1 = 2;
add($num1);
echo $num1;
?>
```



```
12
```

Figure: Output for Code Snippet

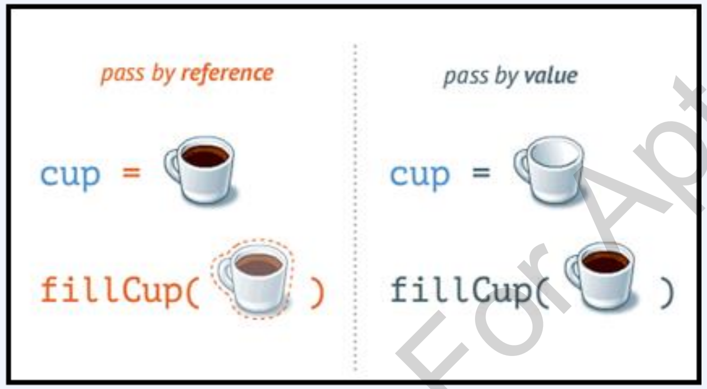
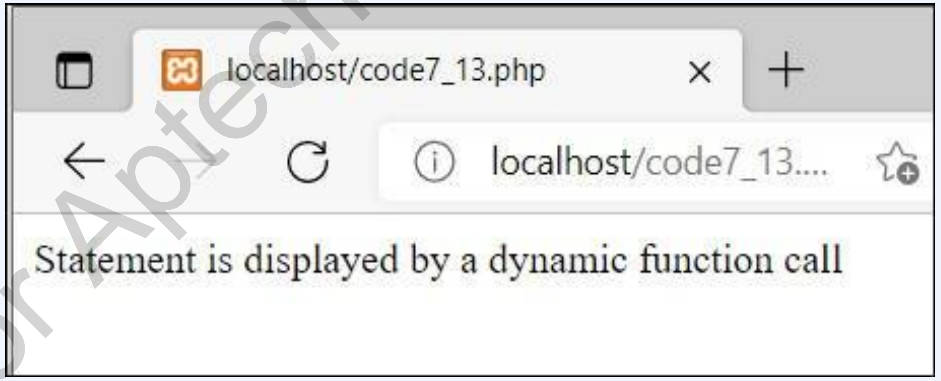


Figure: Pass By Reference and Pass By Value

Dynamic Function Calls

Code Snippet:

```
<?php
function Test()
{
echo "Statement is displayed by a dynamic function call<br />";
}
$function_holder = "Test";
$function_holder();
?>
```



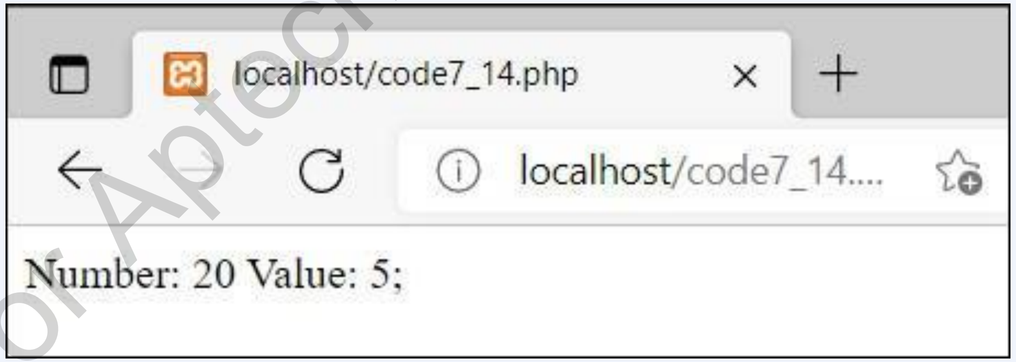
```
Statement is displayed by a dynamic function call
```

Figure: Output for Code Snippet

PHP Named Arguments

Code Snippet:

```
<?php
function named_arguments($number = 11, $value1 = 5){
echo "Number: "; $number;
echo " ";
echo "Value: "; $value1;
}
named_arguments (value1: 5, number: 20); //Named arguments in // different order
?>;
```



```
Number: 20 Value: 5;
```

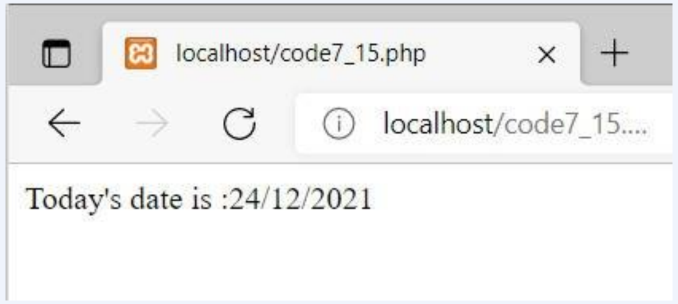
Figure: Output for Code Snippet

date() Function

d	m	Y	l
• This character indicates the day of the month in two digits, for example, 01 to 31.	• This character indicates the month from 01 to 12.	• This character indicates the year in four digits.	• This character, which is the lowercase of the letter L, indicates the day of the week.

Code Snippet:

```
<?php
echo "Today's date is :";
$today1 = date("d/m/Y");
echo $today1;
?>
```



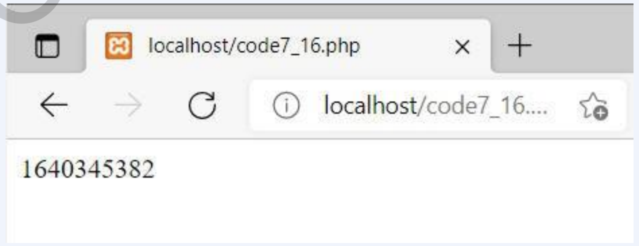
```
Today's date is :24/12/2021
```

Figure: Output for Code Snippet

time() Function

Code Snippet:

```
<?php
$currentTimeinSeconds = time();
echo $currentTimeinSeconds;
?>
```



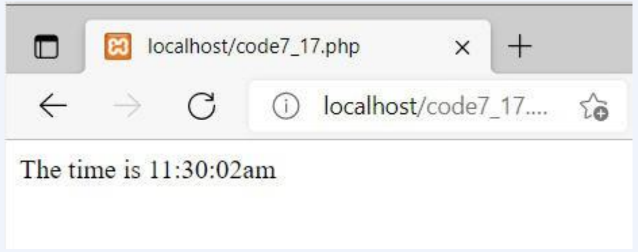
```
1640345382
```

Figure: Output for Code Snippet

H	h	i	s	A
• This character indicates a 24-hour format of an hour from 00 to 23.	• This character indicates a 12-hour format of an hour from 01 to 12.	• This character indicates the minutes from 00 to 59.	• This character indicates the seconds from 00 to 59.	• This character indicates Ante Meridiem (AM) and Post Meridiem (PM).

Code Snippet:

```
<?php
echo "The time is " . date("h:i:sa");
?>
```



```
The time is 11:30:02am
```

Figure: Output for Code Snippet

Summary

- PHP has over 1000 built-in functions that help users to complete common tasks. This makes it easy for the users to execute and rerun common tasks in a program.
- Some of the advantages of using PHP functions are reusable code, minimal code, and clarity of code.
- PHP gives users the flexibility to create their own functions. User-defined functions can be created according to the task user wants to perform.
- An argument is similar to a variable and is used to pass input data into functions.
- Default argument value will be utilized if no value is entered for the argument when the function is executed.
- If a function argument is entered by reference, any modification done to the argument will automatically modify the variable that was entered.
- An important aspect of PHP functions is that they can also return results during a later stage of the program.
- PHP allows users to allocate function names as strings to variables and later utilize these variables in the same way as the function name.
- Named arguments is a new feature introduced in PHP 8, which permits users to pass arguments to a function considering only the parameter names and not the parameter positions.
- While the PHP date() function displays the current date and/or time of the server, the time() function displays the current time in terms of seconds.