Chapter-Two
Server Side Scripting with Database Connectivity

Introduction to server side scripting language

- ☞ Server side scripting is a method of designing websites so that the process or user request is run on the originating server.
- ☞ It is a technique used in web development which involves employing scripts on a web server which produce a response customized for each user's request to the website.
- ☞ Server side scripting provide an interface to the user and are used to limit access to proprietary data and help keep control of the script source code. Server side web development uses specific tools and frameworks to build dynamic programs that generate dynamic web sites. This scripting is often used to provide a customized interface for the user.
- ☞ It is designed such that user requests are run on the primary server. It provides an interface to users to make requests, give instructions to the server to process. Server side scripting is also responsible for user authentication and data security.
- ☞ Server side technology are a set of tools and programing languages that are used to design, build and maintain server side operations of a web application.
- ☞ Many languages may be used to create these server side script is as follows:
    - ❖ ASP
    - ❖ ASP.NET
    - ❖ Java
    - ❖ JavaScript(using SSJS (Server Side JavaScript) e.g. node.js)
    - ❖ Perl
    - ❖ PHP
    - ❖ Python
    - ❖ Ruby

Advantages and Disadvantages of Server Side Scripting

Advantages of Server Side Scripting:

i. Server-side scripting prevents increasing of the load as it does not require plugins or browser scripting technology (such as JavaScript). Overloading leads to problems like slow loading, high CPU usage and even freezing.

ii. It executes scripts on server, it reduces the load on user's computer.

iii. It can be used to create database web application.
iv. It is use to generate dynamic website easily in which site administrator can alter the content at any time.
v. It does not depend upon the version of browser.
vi. Website owners can create their own applications and make use of CMS (Content Management System) to easily create and update content on the web without coding as the Server-side scripting languages like PHP can be configured to run CMS applications like wordPress and Joomla.
vii. Loading time of the web pages is often reduced with Server-side scripting which helps to improve your site's Google ranking.

Disadvantages of Server Side Scripting

I. Server-side scripting is slow at times like: It the user disconnects from the internet or the web hosting is down, then the script may take a long time to execute.
II. The scripting software has to be installed on the server.
III. New security concerns are associated with the dynamic scripts as in some cases hackers exploit the code flows to gain access to servers.
IV. Server side scripts are difficult to debug.
V. A database is required to store the dynamic data that needs a regular backing up and has to be kept secure.
VI. Websites using large applications and with heavy traffic have to make use of more powerful hosting methods like dedicated servers or cloud hosting to cope with demand.

Introduction to PHP (Hypertext Preprocessor)

- PHP is a widely-used open source general purpose server side scripting language that is especially suited for web development and can be embedded into HTML code.
- PHP is typically used as a server side language (as opposed to a language like JavaScript that's generally executed on the client-side).
- It is also used to develop dynamic websites or web applications. PHP scripts can only be interpreted on a server that has PHP installed. The client computers accessing the PHP scripts require a web browser only.
- PHP file contains PHP tags and ends with the extension ".php".
- PHP is also known as server-side language. That's because the PHP doesn't get executed on your computer, but on the computer you requested the page from. The result are then handed over to you and displayed in your browser.
- PHP is compatible with all major operating systems including Linux, windows and MAC OS, It is also compatible with most popular web servers such as Apache and LiteSpeed. PHP can be used to create dynamic content such as

PDFs, images and is compatible with the most popular types of database tools. Such as MySQL.

- ☞ PHP is used to power some of the most popular software such as blogging tools like WordPress and Joomla as well as Ecommerce shopping carts like Open Cart and Prestashop and Bulletin Board tools like PHPBB and Vbulletin.
- ☞ PHP can also be used to add extra functionality to your website by integrating applications built for third party websites such as Facebook or by creating PHP based mailing lists or you can even create WordPress plugins for commercial use.

Features of PHP

- ☞ Some important features of PHP are as follows:

1. Simple Programming language
   - ☞ It is very simple and easy to use, compare to other scripting language it is very simple and easy, this is widely used all over the world.
2. Interpreted language
   - ☞ It is an interpreted language, i.e. there is no need for compilation.
3. Faster Programming language
   - ☞ It is faster than other scripting language e.g. asp and jsp.
4. It is an open source programming language
   - ☞ PHP is an interpreted scripting language (actually precompiled opcodes), free, open source and distributed under a license authorizing the modification and redistribution.
5. It is platform independent language
   - ☞ PHP code will be run on every platform, Linux, UNIX, Mac Operating System and windows.
6. Case Sensitive language
   - ☞ PHP is case sensitive scripting language at time of variable declaration. In PHP, all keywords (e.g. if, else, while, echo etc.), classes, functions and user-defined functions are NOT case-sensitive.
7. Error Reporting
   - ☞ PHP have some predefined error reporting constants to generate a warning or error notice.
8. Loosely Typed language
   - ☞ PHP supports variable usage without declaring its data type. It will be taken at the time of the execution based on the type of data it has on its value.

Basic Component for PHP Web Development

- Some basic components of PHP web development are as follows:

I. Web Server

- File servers, database servers, mail servers and web servers use different kinds of server software. Each of these application can access file stored on a physical server and use them for various purposes.
- A web server is server software or hardware dedicated to running server software that can satisfy client requests on the World Wide Web.
- The job of the web server is to serve websites on the Internet. To achieve that goal, it acts as a middleman between the server and client machines. It pulls contents from the server on each user request and delivers it to the web.
- The biggest challenge of a web server is to serve many different web users at the same time each of whom is requesting different pages.
- Web servers process files written in different programming languages such as PHP, Python, Java and so on.

II. Apache

- The Apache HTTP server, colloquially called Apache, is free and open-source cross-platform web server software, released under the terms of Apache license 2.0.
- Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation. It allows website owners to serve content on the web- hence the "web server."
- It is one of the oldest and most reliable web servers which was released in 1995.
- When someone wants to visit a website, they enter a domain name into the address bar of their browser. Then, the web server delivers the requested files by acting as a virtual delivers the requested files by acting as a virtual delivery man.

III. MySQL

- MySQL is an open-source relational database management system that allows you to manage relational databases. Its name is a combination of "My", the name of co-founders Michael Widenius's daughter and "SQL", the abbreviation for Structured Query Language.
- It is open source software backed by Oracle. MySQL can run on various platforms UNIX, Linux, windows etc. you can install it on a server or even in a desktop. Besides MySQL is reliable and fast. MySQL is an essential component of the AMP stack, which includes the Apache, MySQL and PHP.

- ☞ The MySQL server provides a database management system with querying and connectivity capability, as well as the ability to have excellent data structure and integration with many different platforms. It can hand le large databases reliably and quickly in high-demanding production environments.
- ☞ The MySQL server also provides rich function such as its connectivity, speed and security that makes it suitable for accessing databases.

IV. PHP

- ☞ PHP is quite a simple language with roots in C and Perl, yet it looks more like Java. It is also very flexible, but there are few rules that you need to learn about its syntax and structure.
- ☞ PHP can be embedded in HTML, and it's well suited for web development and the creation of dynamic web pages for web application. It's considered a friendly large language with abilities to easily connect with MySQL, Oracle, and other database.
- ☞ PHP scripts can be used on most of the well-known operating systems like Linux, Unix, Solaries, Microsoft windows, MAC OS and many others. It also supports most web servers including Apache and IIS. Using PHP affords web developers the freedom to choose their operating system and web server.

The use of PHP are as follows:

i. It is used to make a dynamic website.
ii. It is used to interacting with web server.
iii. It is used to collaborate with any back-end / database server for example MySQL.
iv. To communicate with the local record arrangement of the OS.
v. It can be used to Encrypt Data.
vi. Access Cookies variable and set treats.
vii. It is utilized to send and get E-Mails.
viii. Utilizing PHP you can include, erase and alter components inside your database through PHP. Access treats factors and set treats.

Basic PHP Syntax

- ☞ PHP code is start with <?php and end with ?>
- ☞ Every PHP statements end with a semicolon (;)
- ☞ PHP code save with ".php" extension
- ☞ PHP contains some HTML tag and PHP code.
- ☞ You can place PHP code anywhere in your document.

Syntax for PHP is:

```
<?php
//statements or blocks

?>
```

Note:
PHP can support multiple start and end tags. Each start tag must be closed by its own end tags.

Comments in PHP

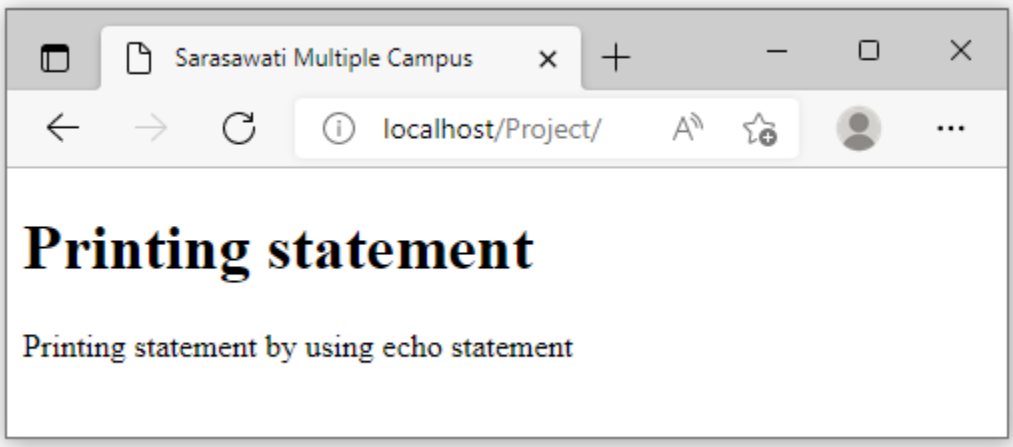
- There are two ways in which you can add comments to your PHP code.
  1. **Single line comment**
- The two backslash (//) or the hash (#) is used to post a single line comment in PHP.
  2. **Multiline comment**
- The multiline comment start with /* and end with */. Everything between the /* and */ is treated as a comment by the PHP engine.

Printing Statement in PHP

- There are two different function we can use to print or display contents.
- To print the statement in PHP we use ***print*** and ***echo*** statement.
- In order to output one or more strings, we can use echo statement. Anything that can be displayed to the browser using echo statement, such as string, number, variables, values, the results of expressions etc.
- Unlike some other language constructors echo does not behave like a function, so it cannot always be used in the context of a function. The end of echo statement is identified by the semi-colon (;).

Example:
Example program to demonstrate echo statement.

```
<BODY>
<h1>Printing statement</h1>

<?php
echo “Printing statement by using echo statement”;
?>
</BODY>
```



PHP Short - Tag

- For less motivated typists, an even shorter delimiter syntax is available, known as short tags. However, to use this feature, you need to enable PHP's short_open_tag directive into php.ini file.
  An example follows:
  <?print "This is another PHP Syntax.";?>

  When short-tags syntax is enabled and you want to quickly escape to and from PHP to output a bit of dynamic text, you can omit these statements using an output variation known as short circuit syntax:
  <?="This is another PHP Syntax.";?>
  This is functionally equivalent to both of the following varations:
  <?echo "This is another PHP Syntax.";?>
  <?php echo "This is another PHP Syntax.";?>

PHP Variables

- Variable is an identifier that hold the data of your program manipulates while it runs, such as user information that you have loaded from a database or entries that have been typed into an HTML form.
- In PHP, variables are denoted by a (dollar sign) $ followed by the variable's name.
- To assign a value to a variable, use an equals sign (=). This is known as the assignment operator.

Rules for declaring variable in PHP

- A variable start with a dollar sign, followed by the name of the variable.
- A variable name must start with a letter or the underscore character.
- A variable name can't start with a number.

- ❖ A variable name can only contain alpha-numeric characters and underscores (A-Z, 0-9, and _).
- ❖ Variable name are case-sensitive (eg. $txt and$TXT both are two different)

Variable names may only include:

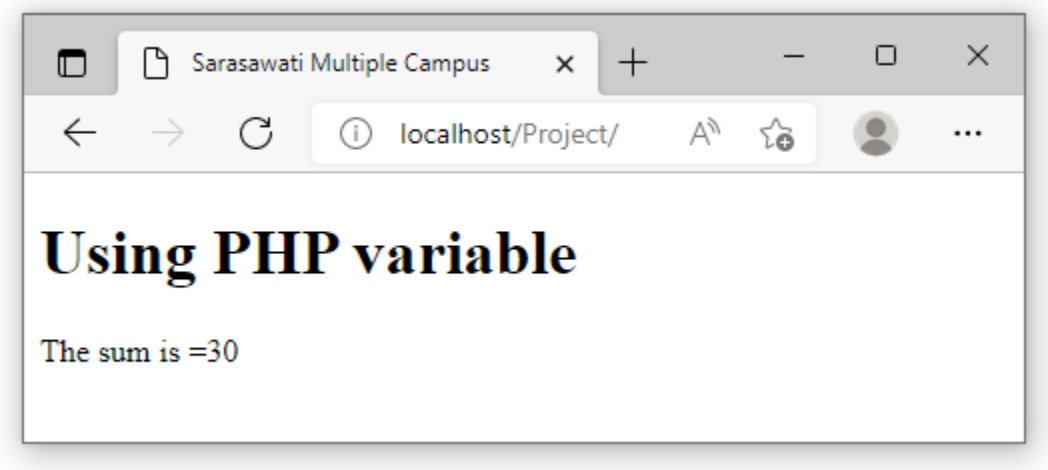
- ❖ Uppercase or lowercase Basic Latin letters (A-Z and a-z)
- ❖ Digits are (0-9)
- ❖ Underscore character (_)
- ❖ Any non-Basic Latin character (such as c), if you are using a character ending such as UTF-8 for your program file.
- ❖ Additionally, the first character of a variable name is not allowed to be a digit
- ☞ Some example of variables are as follows:
  $book, $mybook, $My_FIRST_BOOK, $_m_y_b_o_o_k, $myBook2myFriend etc.

Example:

Example program to demonstrate PHP variables.

```
<?php
                $a=10;
                $b=20;
                $c=$a+$b;
                echo "The sum is =".$c;
?>
```

Out Put:



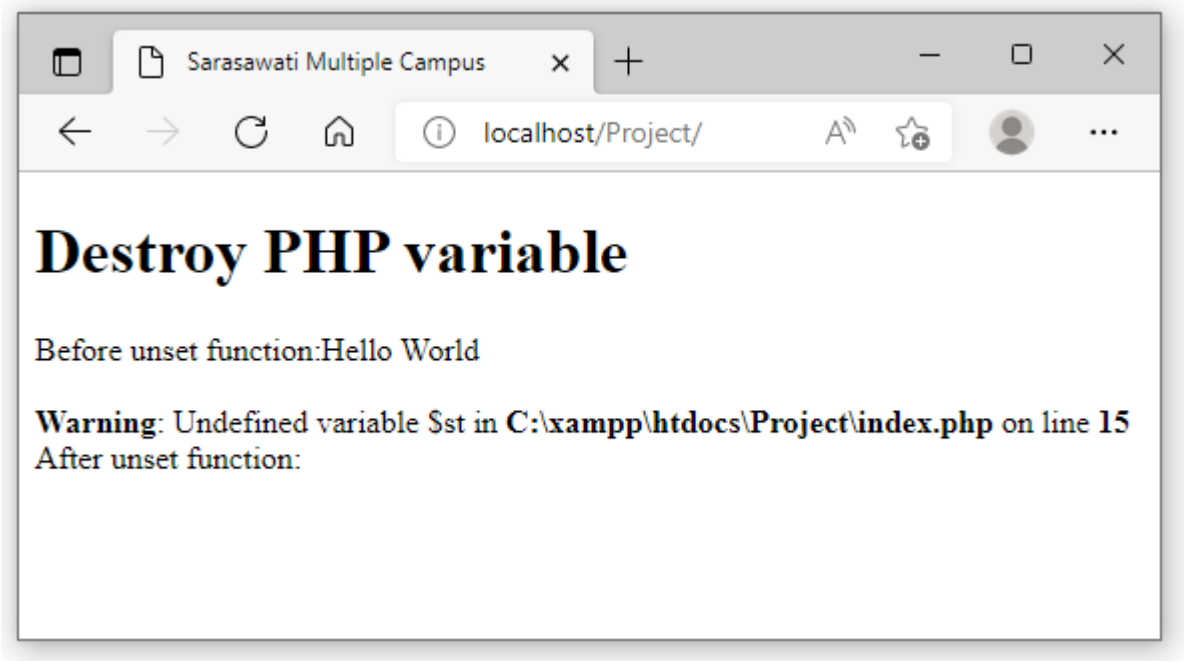
Destroying PHP Variables

- To destroy a variable, pass the variable to PHP's **unset()** function. If **unset()** is called inside a user defined function, it unsets the local variables.
- If a user wants to unset the global variable inside the function, then he/she has to use $GLOBALS array to do so. The **unset()** function has no return value. After using unset function our program throw notice error "Undefined variable".

Example:

```
<body>
<h1>Destroy PHP variable</h1>
<?php
        $st="Hello World";
        echo "Before unset function:".$st;
        echo "</br>";
        unset($st);
        echo "After unset function:".$st;
?>

</body>
```

Out Put:



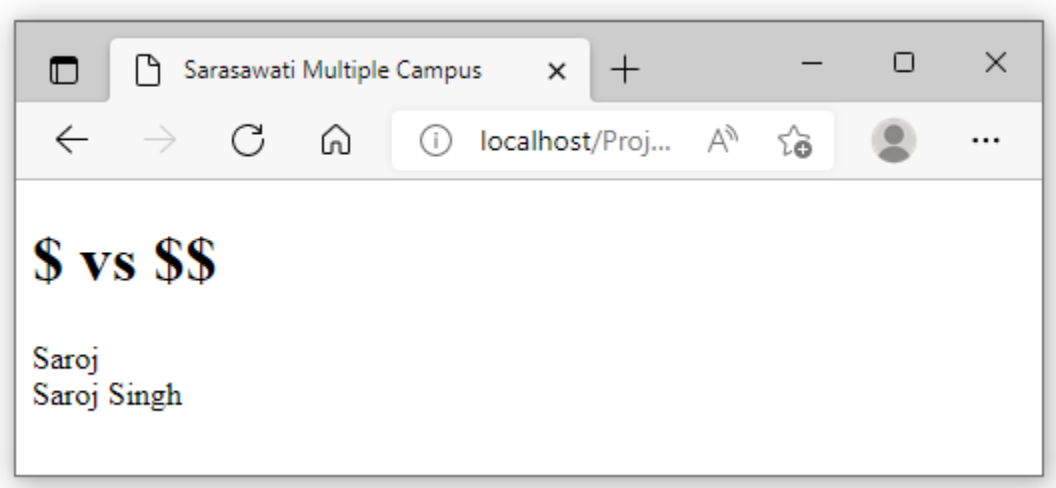
Difference Between $var and $$var in PHP

- PHP $$var uses the value of the variable whose name is the value of $var. It means $$var is known as reference variable whereas $var is normal variable. It allows you to have a "variable's variable" – the program can create the variable name the same way it can create any other storing.

Example:

```
<body>
<h1>$ vs $$</h1>
<?php
                $name="Saroj";
                $Saroj="Saroj Singh";
                echo $name."</br>";
                echo $$name . "</br>";
?>

</body>
</html>
```

Out Put:



Super Global Variable in PHP

- PHP super global variable is used to access global variables from anywhere in the PHP script. PHP super global variables is accessible inside the same page that defines it, as well as outside the page.
- While local variable's scope is within the page that defines it. But there is no guarantee that each web server recognizes every super global.

- The PHP super global variables are:

  - $GLOBALS
  - $_SERVER
  - $_REQUEST
  - $_POST
  - $_GET
  - $_FILES
  - $_ENV
  - $_COOKIE
  - $_SESSION

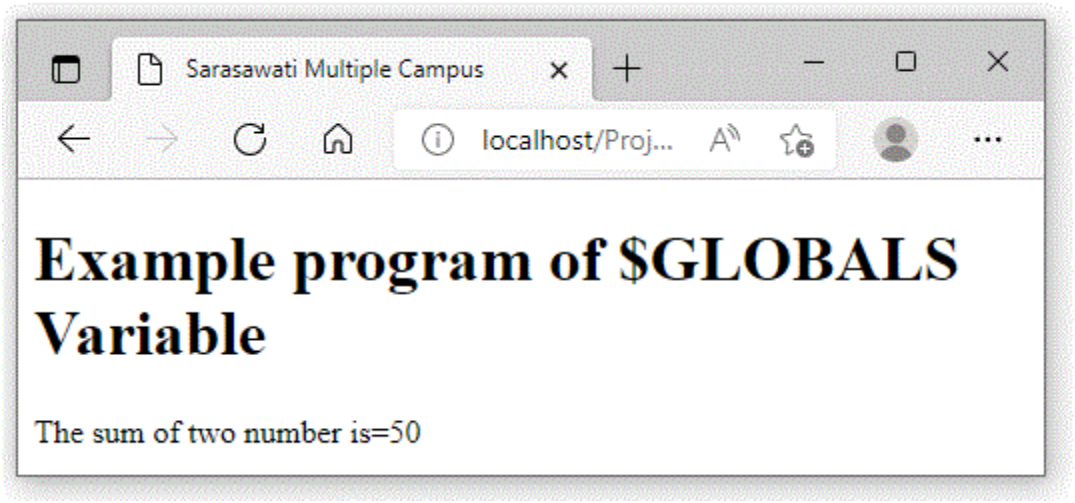
I. $GLOBALS Variables

- $GLOBAL is a PHP super global variable which can be used instead of 'global' keyword to access variables from global scope, i.e. the variable which can be accessed from anywhere in a PHP script even within functions or methods.
- PHP stores all global variables in an array called $ GLOBALS [inxex]. The index holds the name of the variable.

Example:

```
<body>
<h1>Example program of $GLOBALS Variable</h1>
<?php
  $x =25;
  $y = 25;
  function add()
  {
    $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
  }
  add();
  echo "The sum of two number is=".$z;
?>

</body>
```

Out Put:



II. $_SERVER Variable

- $_SERVER Variable is one of the PHP global variables. It is referred to as super global variable, which contains information about server and execution environments. These are pre-defined variables so they are always accessible from any class, function or file.
- $_SERVER is a super global variable containing information such as headers, paths and script locations.
- The entries here are recognized by web servers, but there is no guarantee that each web server recognizes every super global. These three PHP $_SERVER arrays all behave in similar way they return information about the file in use. When exposed to different scenarios, in some cases they behave differently. These example may help you decide which is best for what you need.
- List of $_SERVER arrays is available at the PHP website.

Element/Code	Description
$_SERVER['PHP_SELF']	Return the filename of the currently executing script.
$_SERVER['GATEWAY_INTERFACE']	Returns the version of the Common gateway Interface (CGI) the server is using.
$_SERVER['SERVER_ADDR']	Return the IP address of the host server.
$_SERVER['SERVER_NAME']	Return the name of the host server.
$_SERVER['SERVER_SOFTWARE']	Return the server identification string (Such as Apache 1.6.27)

$_SERVER['SERVER_PROTOCOL']	Return the name and revision of the information protocol (such as HTTP/1.1)
$_SERVER['REQUEST_METHOD']	Returns the request method used to access the page (such as POST).
$_SERVER['REQUEST_TIME']	Returns the timestamp of the start of the request (Such as 1377687496)
$_SERVER['QUERY_STRING']	Returns the query string if the page is accessed via a query string.
$_SERVER['HTTP_ACCEPT']	Return the accept header from the current request.
$_SERVER['HTTP_ACCEPT_CHARSET']	Return the accept charset header from the current request (such as utf-8, ISO-8859-1
$_SERVER['HTTP_HOST']	Return the Host header from the current request.
$_SERVER['HTTP_REFERER']	Returns the complete URL of the current page (not reliable because not all user-agents support it)
$_SERVER['HTTPS']	Is the script queried through a secure HTTP protocol.
$_SERVER['REMOTE_ADDR']	Returns the IP address from where the user is viewing the current page.
$_SERVER['REMOTE_HOST']	Return the host name from where the user is viewing the current page.
$_SERVER['REMOTE_PORT']	Returns the port being used on the user's machine to communicate with the web server.
$_SERVER['SCRIPT_FILENAME']	Return the absolute path name of the currently executing script.
$_SERVER['SERVER_ADMIN']	Returns the value given to the SERVER_ADMIN directive in the web server configuration file (if your script runs on a virtual host, it will be the value defined for that virtual host) (such as user@domain.com)
$_SERVER['SERVER_PORT']	Returns the port on the server machine being used by the web server for communication (Such as 80)

$_SERVER[‘SERVER_SIGNATURE’]	Returns the server version and virtual host name which are added to server-generated pages.
$_SERVER[‘PATH_TRANSLATED’]	Returns the file system based path to the current script.
$_SERVER[‘SCRIPT_NAME’]	Returns the path of the current script
$_SERVER[‘SCRIPT_URL’]	Returns the URL of the current page.

III. $_REQUEST Variable

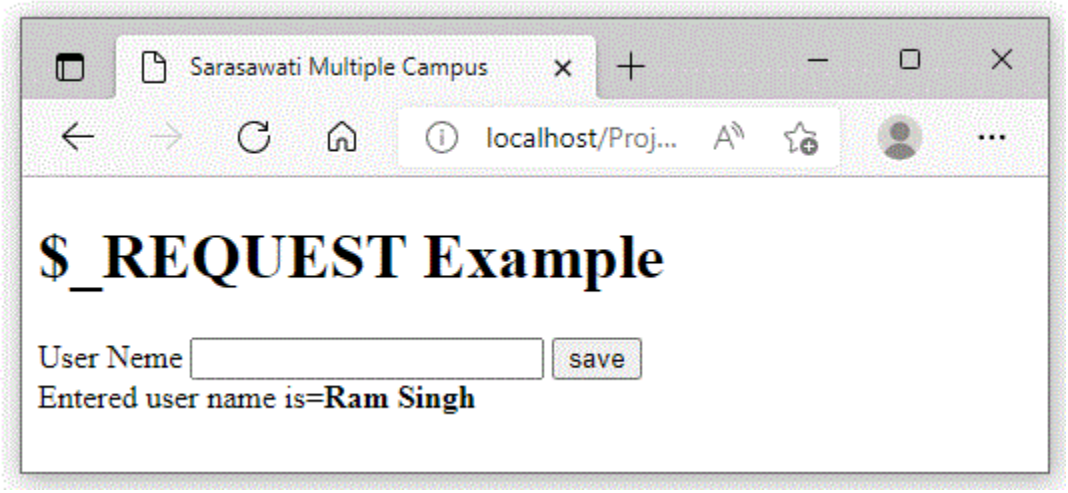
- $_REQUEST can be used to collect data with both post and get method.
- It contains values passed by post, get and cookies. As get is easy to hack so safer mechanism would be to use post when send data from one html/ php file to another. Then you need to use $_POST to get the data.

Example:

```
<body>
<h1>$_REQUEST Example</h1>
<form method="post" action="<?php echo $_SERVER['PHP_SELF'];
?>">
User Neme <input type="text" name="username" />
<input type="submit" value="save" />
</form>
<?php
if($_SERVER["REQUEST_METHOD"]=="POST")
{
  //Collect value of input field
  $username = $_REQUEST['username'];
  if(empty($username))
  {
    echo "User name is empty";
  }
  else
  {
```

```
        echo "Entered user name is="."<b>".$username."</b>";
    }
}
?>

</body>
```

Out Put:



IV. $_POST Variable

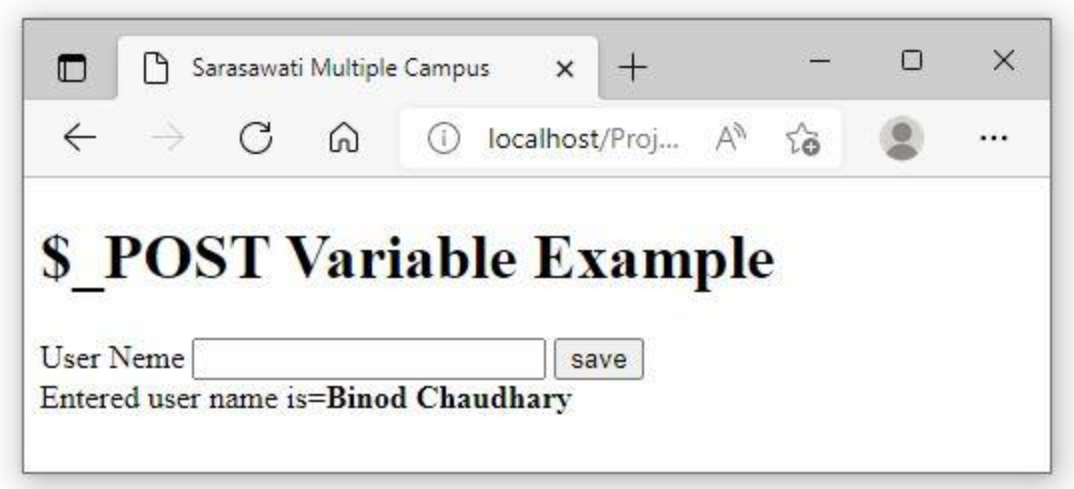
- $_POST is a super global variable which is widely used to pass variables. This super global variable is widely used to handle form data.

Example:

Example program to demonstrate $_POST variable

```
<body>
<h1>$_POST Variable Example</h1>
<form method="post" action="<?php echo $_SERVER['PHP_SELF'];
?>">
User Neme <input type="text" name="username" />
<input type="submit" value="save" />
</form>
<?php
if($_SERVER["REQUEST_METHOD"]=="POST")
{
  //Collect value of input field
  $username = $_POST['username'];
  if(empty($username))
  {
```

```
    echo "User name is empty";
  }
  else
  {
    echo "Entered user name is="."<b>".$username."</b>";
  }
}
?>

</body>
```



V. $_GET Variable

- ☞ $_GET is a super global variable which can be used to do the same job done by POST. But besides, $_GET can do some other wonderful jobs as far as passing data is concerned.
- ☞ It is used to collect value from a form (HTML Script) sent with method='get'.
- ☞ Information sent from a form with the method='get' is visible to everyone (it display on the browser URL bar).

Example:

Example program to demonstrate 'get' variable

```
<body>
<h1>$_GET Variable Example</h1>
<form method="get" action="<?php echo $_SERVER['PHP_SELF'];
?>">
User Neme <input type="text" name="username" />
<input type="submit" value="save" />
</form>
```

```
<?php
if($_SERVER["REQUEST_METHOD"]=="GET")
{
  //Collect value of input field
  $username =$_GET['username'];
  if(empty($username))
  {
    echo "User name is empty";
  }
  else
  {
    echo "Entered user name is=".".<b>".$username."</b>";
  }
}
?>

</body>
```

Out Put:

VI. $_FILES

- $_FILES is a super global variable which can be used to upload files. It is a two dimensional associative global array of items which are being uploaded by via HTTP POST method and holds the attributes of files.

ATTRIBUTE	DESCRIPTION
name	Name of the file which is uploading

size	Size of the file
type	Type of the file (like .pdf, .zip, .jpeg…etc)
tmp_name	A temporary address where the file is located before processing the upload request.
error	Types of error occurred when the file is uploading.

VII. $_ENV (Environment Variable)

- ☞ $_ENV is used to return the environment variables from the web server. It's a super global variable i.e. an array of environment variables. That environment may be bash (or some other shell running the php-cli interpreter) or Apache (or some other server related environment). I have only used them with Apache, so my knowledge is a bit limited.
  For example in Magneto you would define ENV variable in .htaccess(or some other web server configuration file) and they would be imported in the global namespace of PHP as part of the super global variable mentioned.

VIII. $_COOKIE

- ☞ Cookie are small text files loaded from a server to a client computer storing some information regarding the client computer, so that when the same page from the server is visited by the user, necessary information can be collected from the cookie itself, decreasing the latency to open the page.

IX. $_SESSION

- ☞ Sessions are wonderful ways to pass variables. All you need to do is start a session by session_start(); Then all the variables you store within a $_SESSION, you can access it from anywhere on the server.

PHP's Supported Data Types

- ☞ A data type is a generic name assigned to any data sharing a common set of characteristics.
- ☞ It defines the type of data a variable can store.
- ☞ PHP allows eight different types of data types.
- ☞ Common data type include Boolean, integer, float, string and array.
- ☞ The first three called simple data types and the last three are called compound data type.

I. String
II. Integer

III. Float
IV. Boolean
V. NULL
VI. Array
VII. Object
VIII. Resource

i. String Constant

- ☞ String constant is a collection of character enclosed within single or double quote.
- ☞ A string literal can be specified in four different ways:
  - ❖ Single quoted
  - ❖ Double quoted
  - ❖ heredoc syntax
  - ❖ nowdoc syntax

Single quoted

- ☞ To specify a literal single quote, escape it with a backslash(\). To specify a literal backslash, double it(\\). All other instances of backslash will be treated as a literal backslash: this means that the other escape sequences you might be used to, such as \r or \n, will be output literally as specified rather than having any special meaning.

  Example:

```
'Pashupati Multiple Campus'
'PMC'
'scripting-language'
'*9subway\n'
'123$%^789'
'I\'ll be back';
'Variables do not $exand$either';
```

Double Quoted

- ☞ If the string is enclosed in double-quotes ("), PHP will interpret the following escape sequences for special characters.

Sequence	Meaning
\n	It is used for linefeed
\r	It is used for carriage return
\t	It is used for horizontal tab
\v	It is used for vertical tab
\e	It is used for escape
\f	It is used for form feed
\\	It is used for backslash

\$	It is used for dollar sign
\"	It is used for double quote
\[0-7]{1,3}	The sequence of characters matching the regular expression is a character in octal notation, which silently overflows to fit in a byte.
\x[0-9A-Fa-f]{1,2}	The sequence of characters matching the regular expression is a character in hexadecimal notation.
U{[0-9A-Fa-f]+}	The sequence of characters matching the regular expression is a Unicode code point, which will be output to the string as that codepoint's UTF-8 representation (added in PHP 7.00)

Heredoc

- A third way to delimit strings is the heredoc syntax <<<. After this operator, an identifier is provided, then a newline.
- The string itself follows, and then the same identifier again to close the quotation.

Nowdoc

- Nowdocs are to single-quoted strings what heredocs are to double-quoted strings.
- A nowdoc is specified similarly to a heredoc, but no parsing is done inside a nowdoc.

Integer

- An integer is a whole number. It is not fractional valued number.
- PHP supports integer values represented in base 10 (decimal), base 8 (octal), base 16 (hexadecimal) number system.
- To use octal notation, precede the number with a 0 (zero). To use hexadecimal notation precede the number with 0x. To use binary notation precede the number with 0b.
- As of PHP 7.4.0, integer literals may contain underscores(_) between digits, for better readability of literals.

  Examples:

  ```
  $a = 145; //decimal number
  $a=0123; //octal number
  $a=0x1A; //hexadecimal number
  $a=0b1111//binary number
  $a=1_234_567;//decimal number
  ```

Float

- Floating point is a fractional valued number. It may be double, real number.

Example:

```
$a=1.234;
```

```
$a=1.2e3;
$a=7E-10;
$a=1_234.567;
```

Boolean:

- ☞ A Boolean expresses a truth value. It can be either TRUE or FALSE. To, specify a Boolean literal, use the constants TRUE or FALSE. Both are case sensitive.

Example:

```
<?php
$gender= TRUE;
?>
```

NULL

- ☞ The NULL value represents a variable with no value.
- ☞ It is only possible value of type null.
- ☞ A variable is considered to be null if:
  a. It has been assigned the constant NULL.
  b. It has not been set to any value yet.
  c. It has been unset()

Note: there is only one value of type null, and that is the case-insensitive constant NULL.

Example:

```
<?php

        $var = NULL;
?>
```

Compound Data Types

Compound data types allow for multiple items of the same type to be aggregated under a single representative entity.

- ☞ Compound data types are two types:
  a. Array
  b. Object

Array

- ☞ An array is a collection of similar data elements. It is also referred to as indexed collection of data values.

- Each member of the array index references a corresponding value and can be a simple numerical reference to the value's position in the series or it could have some direct correlation to the value.

  Example:

```
$province[0]="Province-1";
$province[1]="Province-2";
$province[2]="Province-3";
$province[3]="Province-3";
$province[4]="Province-4";
$province[5]="Province-6";
$province[6]="Province-7";
```

Object:

- Object is an another compound data type supported by PHP.
- The object is central concept of object oriented programming concept.
- Unlike other data types contained in the PHP, an object must be explicitly declared.
- This declaration of an object's characteristics and behavior takes place within the class.

Example:

```
class Student
{
    private $name;
    function setName($name)
    {
            $this->name=$name;
    }

}
$ram= new  Student;
```

Here,

The ram object's name attribute can then be set by making use of the method

```
setName(){
$ram->setName("Ramesh  Shrestha");
    }
```

Resource

- Unlike other data types, it is acting as a reference or an identifier to access resource data. For example, when we attempt to access the database or a file resource, we can have a resource identifier. This identifier will be used to hook the resource to access data.

- The following code shows how to get the database resource identifier by requesting the MySQL connection. In this code, the database information is specified for the mysqli_connect() and it returns MySQL connection object as a resource identifier.
  $conn=mysqli_connect(localhost,"root","","db_scripting_language");

- While working with files, we need to get the file resource identifier. With the reference to this identifier, we can perform the write, append and more file manipulation operations. The following PHP code is used to create a file resource object reference.
  $fp=fopen("index.php",'r');
- After finishing all the operations with the resource data, the connection or the hook we created with the external resources will be cleared. The following PHP functions are used to clear the resource object reference created for the database and the file resource.
  mysqli_close($conn);
  fclose($fp);

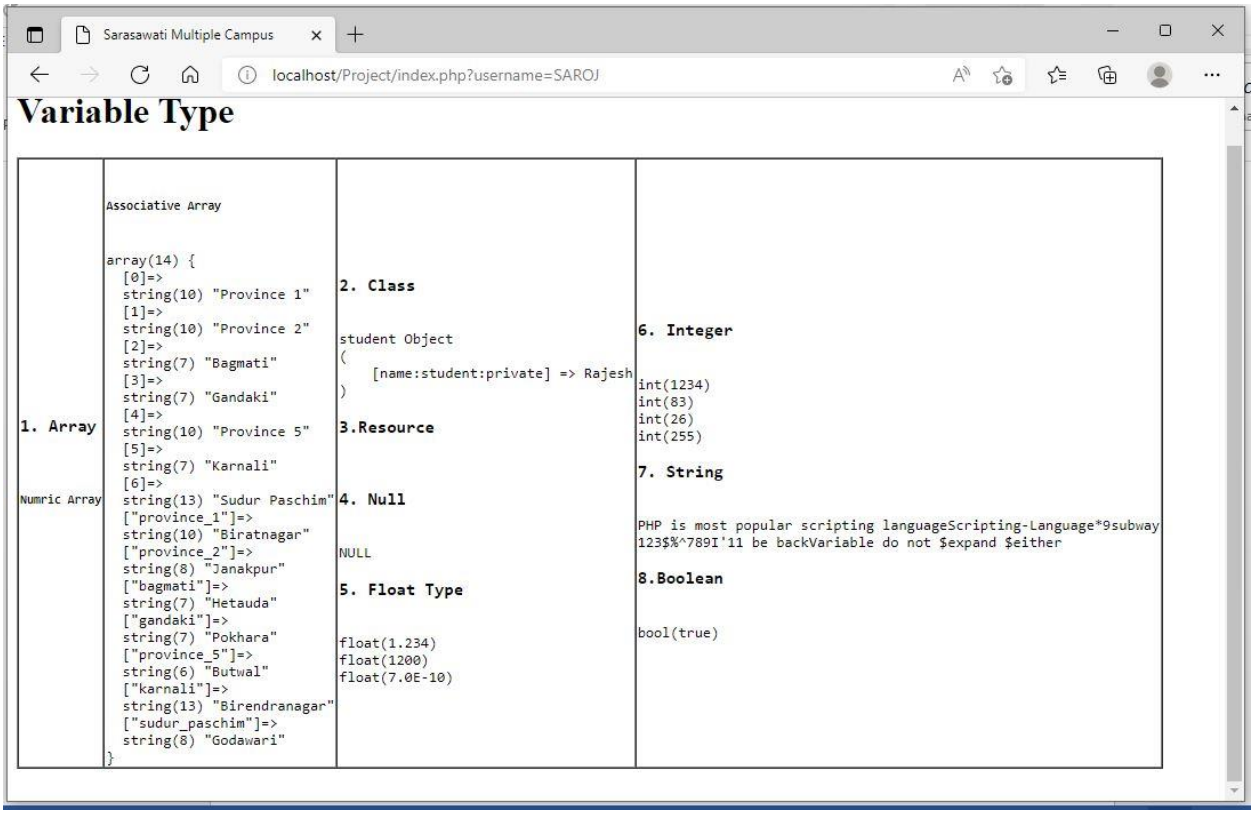
  Example:

```
<body>
<h1>Variable Type</h1>
<pre>
<table border="1" cellspacing="0">
<tr>
<td>
<h3>1. Array</h3>
<h5>Numric Array</h5>
<?php
$province[0] = "Province 1";
$province[1] = "Province 2";
$province[2] = "Bagmati";
$province[3] = "Gandaki";
$province[4] = "Province 5";
$province[5] = "Karnali";
$province[6] = "Sudur Paschim";
?>
</td>
<td>
<h5>Associative Array</h5>
<?php
$province["province_1"] = "Biratnagar";
```

```
$province["province_2"] = "Janakpur";
$province["bagmati"] = "Hetauda";
$province["gandaki"] = "Pokhara";
$province["province_5"] = "Butwal";
$province["karnali"] = "Birendranagar";
$province["sudur_paschim"] = "Godawari";
var_dump($province);
?>
</td>
<td>
<h3>2. Class</h3>
<?php
class student{
        private $name;
        function setName($name)
        {
                $this ->name =$name;
        }
}
$ram = new student;
$ram->setName('Rajesh');
print_r($ram);
?>
<h3>3.Resource</h3>
<?php
/*$fp = fopen("superglobal.php",'r');
print_r($fp);*/
?>
<h3>4. Null</h3>
<?php
$var =NULL;
var_dump($var);
?>
<h3>5. Float Type</h3>
<?php
$a =1.234;
$b =1.2e3;
$c=7E-10;
var_dump($a);
var_dump($b);
var_dump($c);
```

```
?>
</td>
<td>
<h3>6. Integer</h3>
<?php
$a = 1234;
$b= 0123;
$c = 0x1A;
$d = 0b11111111;
var_dump($a);
var_dump($b);
var_dump($c);
var_dump($d);
?>
<h3>7. String</h3>
<?php
echo 'PHP is most popular scripting language';
echo 'Scripting-Language';
echo "*9subway\n";
echo '123$%^789';
echo 'I\'11 be back';
echo 'Variable do not $expand $either';
?>
<h3>8.Boolean</h3>
<?php
$martialStatus=true;
var_dump($martialStatus);

?>
</td>
</tr>
</table>
</pre>
</body>
```

Out Put:



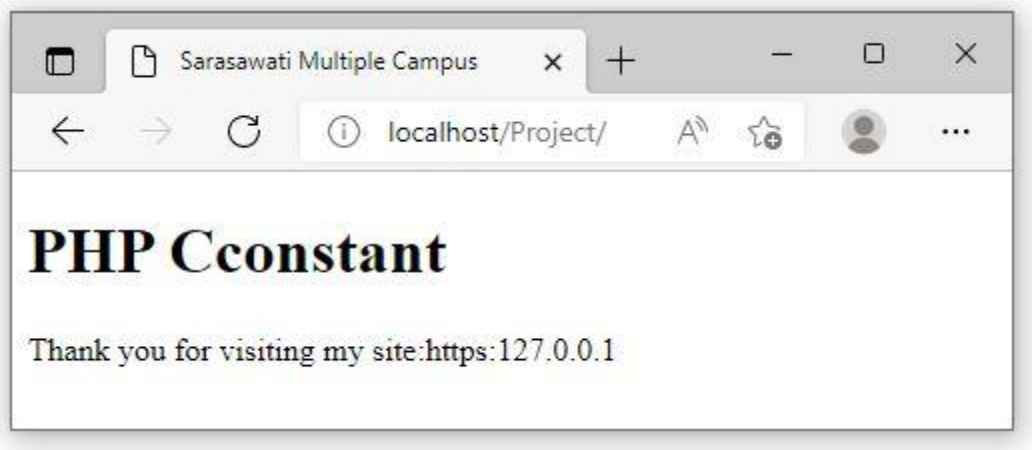
PHP Constant

- A constant is a value which does not change at runtime.
- Name of constants must follow the same rules as variable names, which means a valid constant name must starts with a letter or underscore, followed by any number of letters, numbers or underscores with one exception: the $ prefix is not required for constant names.

Example:

```
<?php
//Defining constant
Define("SITE_URL",https://127.0.0.1);
//Using constant
echo'Thank you for visiting my site:' .SITE_URL;

?>
```

Out Put:



Converting Data Type using Type Casting

- Converting data values from one data type to another is known as **type casting**.
- A variable can be evaluated once as a different type by casting it to another. This is accomplished by placing the intended type in front of the variable to be cast.

Cast Operator => Conversion

- (array) =>Array
- (bool) or (Boolean)=>Boolean
- (int) or (integer)=>Integer
- (object) =>Object
- (real) or (double) or (float) => Float
- (string) => String

1. **Cast an integer as a double:**
   $weight=(double) 13;
   Var_dump($weight);
2. **Cast double to an integer**
   Type casting a double to an integer will result in the integer value being rounded down, regardless of the decimal value.
   $weight=(int)14.8;
   Var_dump($weight)
3. **Cast string data type to integer**
   $name="Saroj singh";
   echo (int)$name;
4. **Simple datatype to Array type**
   You can also cast a datatype to be a number of an array. The value being cast simply becomes the first element of the array.
   $age=25;
   $ageList=(array)$age;

```
echo $ageList[0]; // Out put=25
```

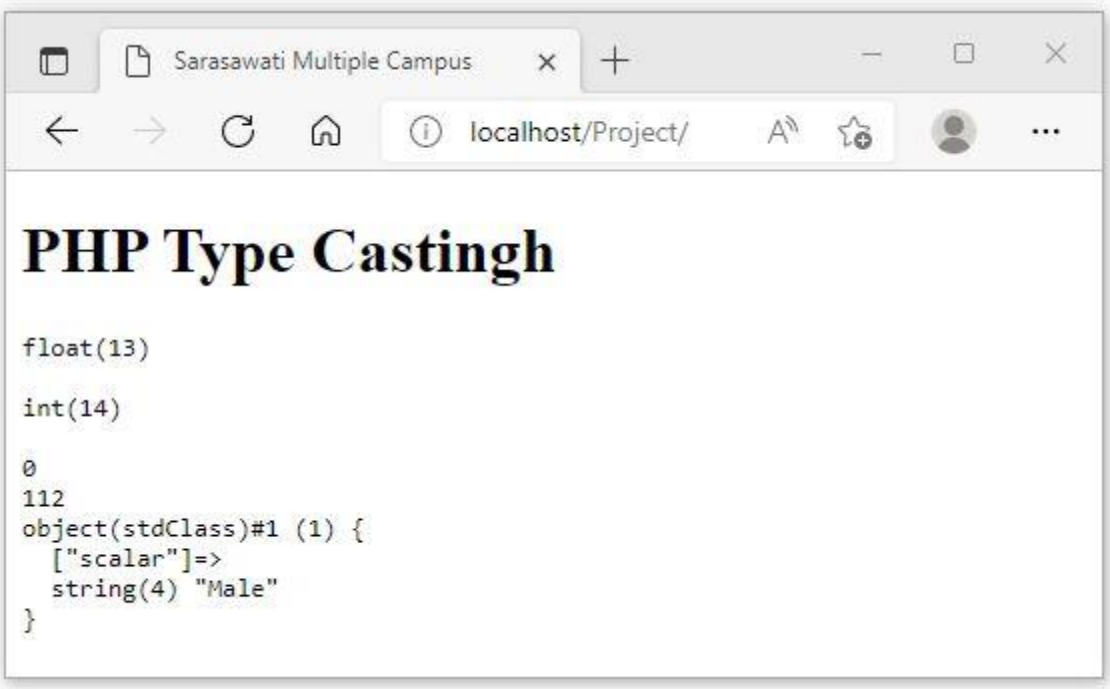
5. **Other type to object type**
   Any data type can be cast as an object. The result is that the variable becomes an attribute of the object, the attribute having the name scalar:

```
$gender=”male”;
$obj=(object)$gender;
Var_dump($obj);
echo $obj->scalar;
```

Example:

```
<body>
<h1>PHP Type Castingh</h1>
<pre>
<?php
$weight = (double) 13;
var_dump($weight);
echo '<br>';
$weight=(int) 14.8;
var_dump($weight);
echo '<br>';
$sentence="This is my test string";
echo (int) $sentence;
echo '<br>';
$age = 112;
$ageList = (array)$age;
echo $ageList[0];
echo '<br>';
$gender = "Male";
$obj=(object) $gender;
var_dump($obj);
?>
</pre>

</body>
```

Out Put:



PHP Operators

- Operator is a symbol which performs some kinds of task.
- PHP is automatic type conversion will convert types based on the type of operator placed between the two operands.

Types of Operator

1. Arithmetic Operator
2. Comparison Operator
3. Bitwise Operator
4. Logical Operator
5. String Operator
6. Increment / Decrement Operator
7. Array Operator
8. Type Operator
9. Execution Operator
10. Error Control Operator
11. Assignment Operator

String Operator:

Operator	Description	Example
.	Concatenation	$a=”Ram”.”singh”; It prints the value “Ram singh”

.=	Concatenation-assignment	$a.=”test”; $a equals its current value concatenated with “test”

Control Structure

- ☞ Control statement is used to regulate the flow of program.
- ☞ There are four types of control structure used in PHP.

1. Sequential Control Statement
2. Selection Control Statement
3. Case Control Statement
4. Loop / Iteration Control Statement

1. Sequential Control Statement

- ☞ Sequential control structure is a logic which flow of program one after another in a sequence.
- ☞ It involves executing all the codes in the order in which they have been written.

2. Selection Control Statement

- ☞ Selection control statement is used to make decision in a program. It works on the basis of two binary values i.e. Zero or One.
- ☞ PHP supports three decision making statements.
  - i. If statement
  - ii. If-else statement
  - iii. Compound if-else statement (if-else if else) statement

If statement

- ☞ When the given condition is true it print the statement otherwise it skips the statement.

Syntax:

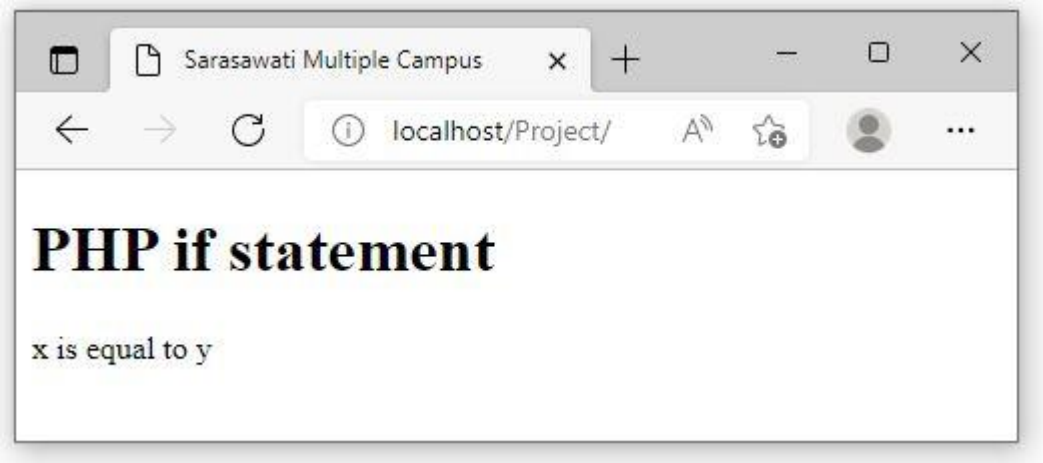
```
if(condition)
{
        //code to be executed
}
```

Example:

```
<?php
$x=10;
$y=10;
if($x==$y)
{
        echo "x is equal to y";
```

```
}
?>
```

Out put:



if-else Statement

- ☞ When the given condition is true then true part will be executed otherwise false part will be executed.

  Syntax:

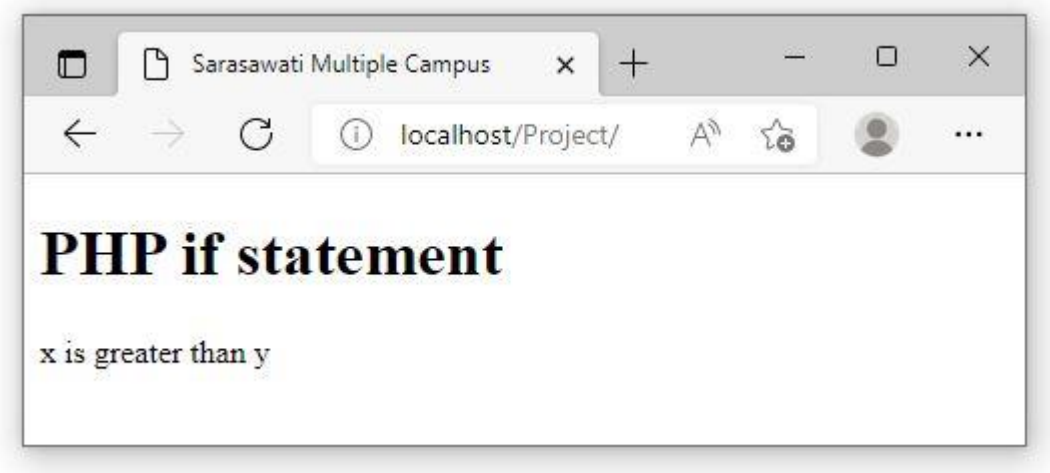
```
if(condition)
{
        //code to be executed if true;
}
else
{
        //code to be executed if false
}
```

Example:

```
<body>
<h1>PHP if-else statement</h1>
<?php
$x= 10;$y= 5;
if($x>$y)
{
  echo "x is greater than y";
}
else
{
  echo "x is less than y";
```

```
}
?>


</body>
```

Out Put:



Compound if-else statement

- ☞ Compound if-else statement print only one statement at time when more than one condition is given.

Syntax:

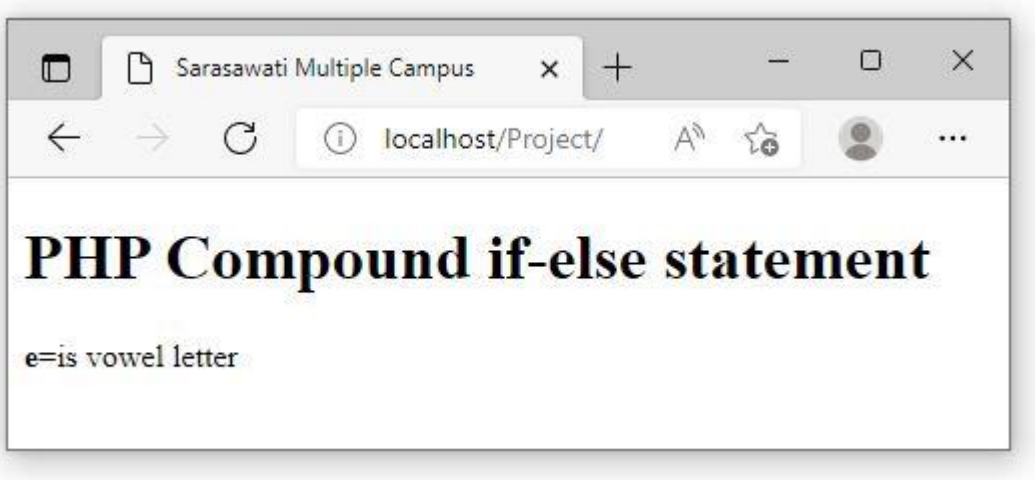
```
if(condition-1)
{
  //code to be executed if true
}
else if(condition-2)
{
  //code to be executed if true
}
else if(condition-3)
{
  //code to be executed if true
}
else
{
  //code to be executed if false
}
```

Example:
PHP program to check given character is vowel or consonant

```
<body>
<h1>PHP Compound if-else statement</h1>
<?php
        $ch="e";
if($ch=="a")
{
        echo $ch." is vowel letter";
}
else if($ch=="e")
{
        echo "<B>". $ch."</b>". "=is vowel letter";
}
else if($ch=="i")
{
        echo $ch. "is vowel letter";
}
else if($ch=="0")
{
        echo $ch. "is vowel letter";
}
else if($ch=="u")
{
        echo $ch. "is vowel letter";
}
else
```

```
{
        echo $ch. "is consonant letter";


}
?>
</body>
```

Out Put:



Switch Statement/case control statement

- statement is alternative to compound if – else statement but only difference is that compound if-else statement print the statement by checking its condition whereas case control statement print the statement by matching its label.

Syntax:

```
switch(expression)
{
        case lebel-1:
                        //code to be executed if lebel-1 match
                        Break;
        case lebel-2:
                        //code to be executed if lebel-2 match
                        Break;
        case lebel-3:
                        //code to be executed if lebel-3 match
                        Break;
```

```
        case lebel-4:
                        //code to be executed if lebel-1 match
                        Break;

        ……..
        ……..
        case lebel-n:
                        //code to be executed if lebel-1 match
                        Break;
        default:
                        //code to be executed if case does not match

}
```

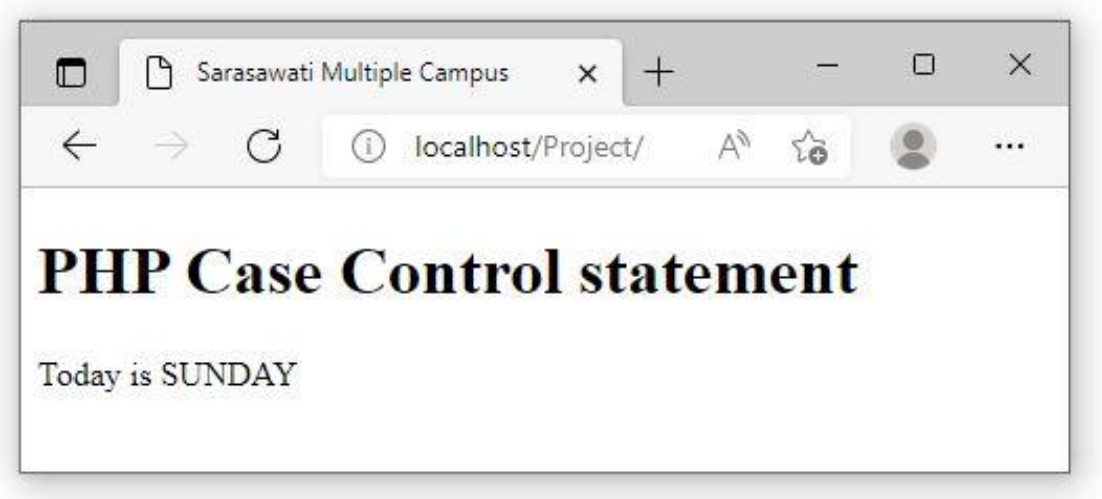
Example

PHP program to demonstrate case control statement.

```
<body>
<h1>PHP Case Control statement</h1>
<?php
$day=1;
switch($day)
{
        case 1:
                echo "Today is SUNDAY";
                        break;
        case 2:
                echo "Today is MONDAY";
                        break;

        case 3:
                echo "Today is TUESDAY";
```

```
                    break;

        case 4:
            echo "Today is WEDNESDAY";
                    break;

        case 5:
            echo "Today is THURSDAY";
                    break;

case 6:
            echo "Today is FRIDAY";
                    break;
        case 7:
            echo "Tody is SATURDAY";
                    break;

}
?>
</body>
```

Out Put:



Loop Statement in PHP

- ☞ When one statement requires number of time then it is said to be a loop.
- ☞ Loops in PHP are used to execute the same block of code a specified number of times. PHP supports following four loop tyes:

I. for loop
II. while loop
III. do-while loop
IV. foreach loop

Loop control variable:

- ☞ A variable which controls the flow of loop is known as loop control variable.
- ☞ Generally there are three loop control variable.
  a. Initialization
  b. Condition
  c. Updation(Increment/decrement)

I. for loop:

- ☞ For loop is used when you know how many time a statement or a block of statements.

Syntax:

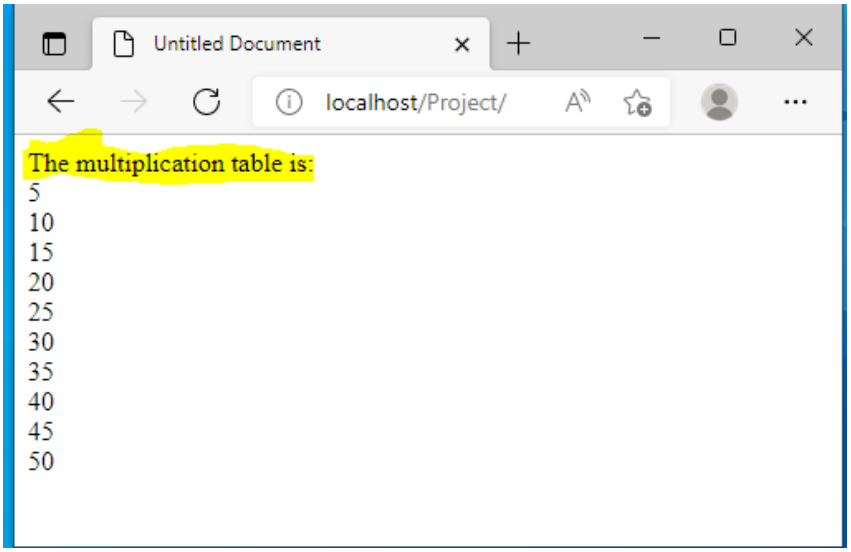
```
for(initialization; condition; updation)
{
    //code to be executed
}
```

Example:

PHP program to demonstrate for loop.

```
<?php
$num=5;
echo "The multiplication table is:<br>";

for($i=1;$i<=10;$i++)
{
        $mul=$num * $i;
        echo $mul."<br>";

}
?>
```

Out Put:



ii. PHP while loop

- ☞ While loop print the statement after checking it's condition.
- ☞ It is mainly used to read records returned from a database query.

Syntax:

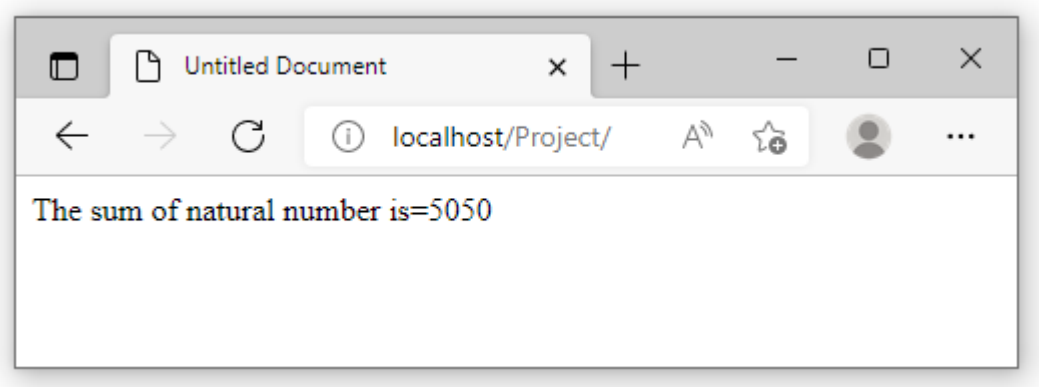
```
while(condition)
{
        //block of code to be executed.
}
```

Example:
PHP program to demonstrate while loop.

```
<?php
```

```
$i=1;
$sum=0;


while($i<=100)
{
        $sum=$sum +$i;
        $i=$i+1;

}
echo "The sum of natural number is=".$sum;
?>
```

Out Put:



iii. do-while loop

- ☞ do-while loop print the statement at least one before checking its condition.

Syntax:

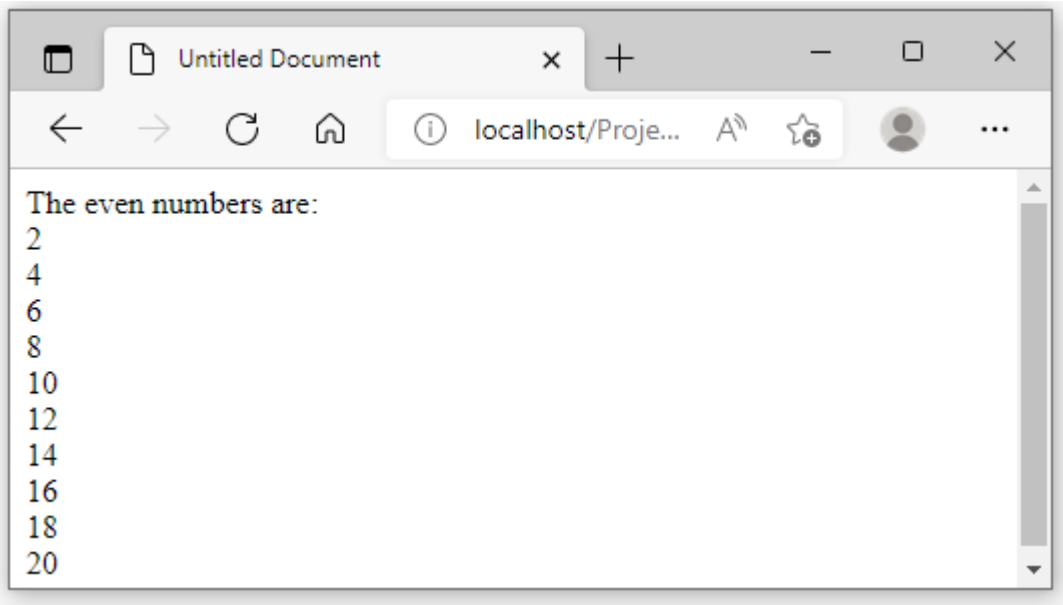
```
Do
{
                        //code to be executed
}while(condition);
```

Example:
PHP program to display even number between 1 to 20.

```
<?php
$i=2;
echo "The even numbers are:<br>";
while($i<=20)
{
                        echo $i."<br>";
                        $i=$i+2;
```

```
}
?>
```

Out Put:



iv. for-each Loop

- ☞ The PHP for each loop is used to iterate through array values. For each pass the value of the current array element is assigned to **$key**=> $value and the array pointer is moved by one and in the next pass next element will be processed.

Syntax:

```
foreach($array as $key => value)
{
        //code to be executed
}
```

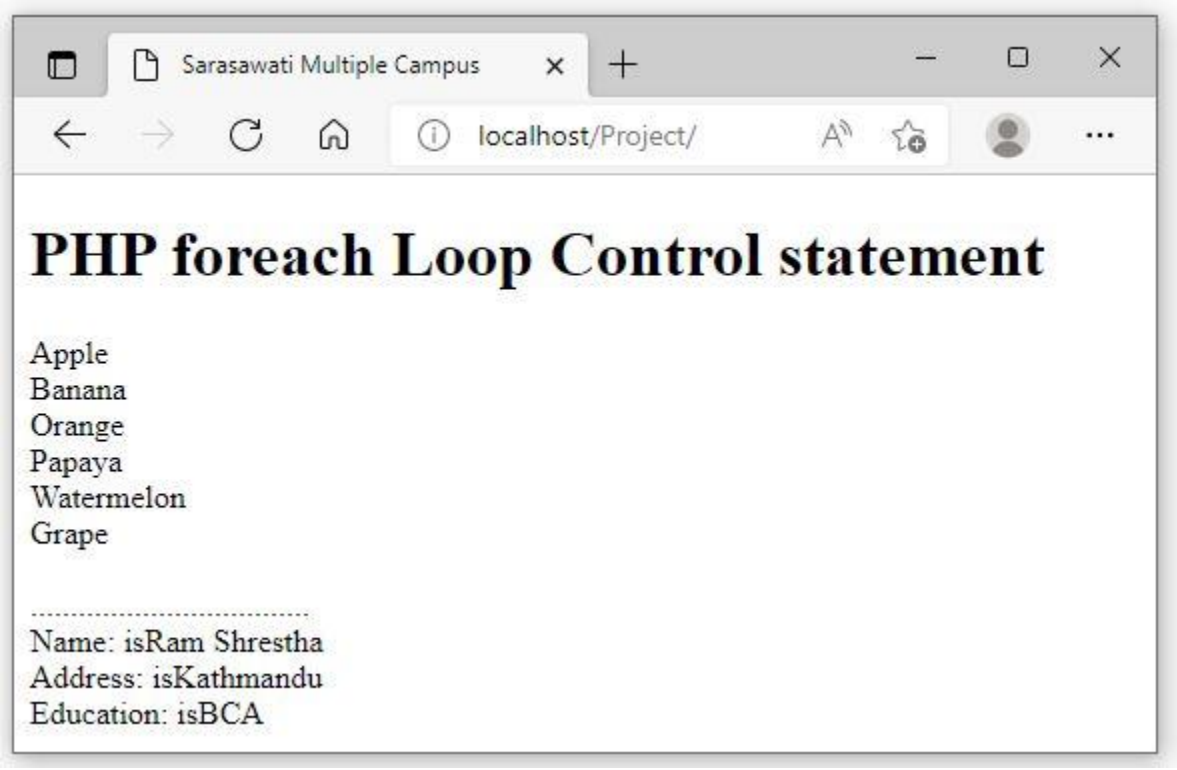
Where,
$array is the array variable to be looped through
$key is the temporary variable that holds the current array item key.
$value is the temporary variable that holds the current array item key.

Example:

```
<body>
<h1>PHP foreach Loop Control statement</h1>
<?php
```

```
$fruits =array("Apple","Banana","Orange","Papaya","Watermelon","Grape");
foreach($fruits as $fruit)
{
                                        echo $fruit."<br>";
}

?>
<br>
<?php
echo ".....<br>";
?>
<?php
                                        $student =array("name:&amp;nbsp;"=>"Ram Shrestha",
"Address:&amp;nbsp;" =>"Kathmandu", "Education:&amp;nbsp;" => "BCA");
Foreach($student as $key =>$value)
{
                                        echo ucfirst($key). "is" . $value . "<br>";
}
?>

</body>
```

Out Put:



ARRAY

- ☞ An array is a collection of similar data element, which have same name and same type.
- ☞ It is a variable that stores more than one piece of related data in a single variable.

Types of Array

I. Numeric index array
II. Associative index array
III. Multidimensional array

I. Numeric index array

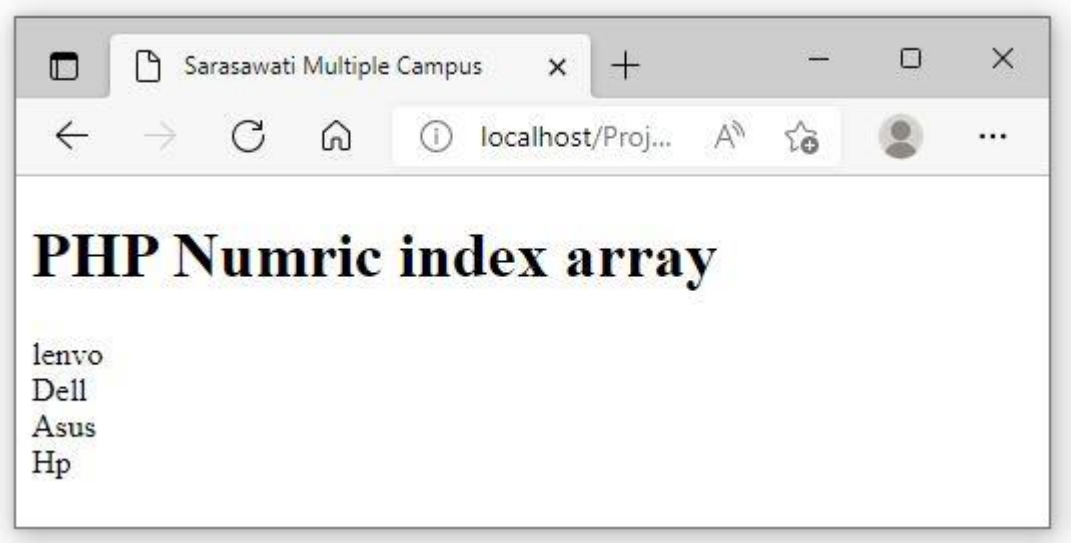
- ☞ Numeric arrays use number as access keys. An access key is a reference to a memory slot in an array variable.
- ☞ The access key is used whenever we want to read or assign a new value an array element.
- ☞ Arrays with sequential numeric index, such as 0,1,2 etc.
- ☞ The different way of creating numeric index array as follows:

```
$array_name[0] = value1;
$array_name[1] = value2;
$array_name[2] = value3;
$array_name[3] = value4;
$array_name[4] = value5;

OR
$array_name = array(value-1, value-2, value-3, value-4, value-5);

OR
$array_name =[value1, value2,value3, value-4, value-5];
```

Example:

```
<?php
    $laptops= array("lenvo","Dell","Asus","Hp");
    for($i= 0; $i<count($laptops); $i++)
    {
        echo $laptops[$i];
        echo "<br>";
    }
?>
```

Out Put:



II. Associative index array

- ☞ This is the most frequently used type of PHP arrays whose elements are defined in key /value pair.
- ☞ Associative array differ from numeric array in the sense that associative arrays use descriptive names for id keys.
- ☞ Associative array are also very useful when retrieving data from the database.
- ☞ The different ways of creating numeric index array as:

Syntax:
$array_name["key1"] = value-1;
$array_name["key2"] = value-2;
$array_name["key3"] = value-3;
$array_name["key4"] = value-4;

OR;
$array_name = array("key1"=>value1, "key2" =>value2, "key3" =>value3, "key4"=> value4);

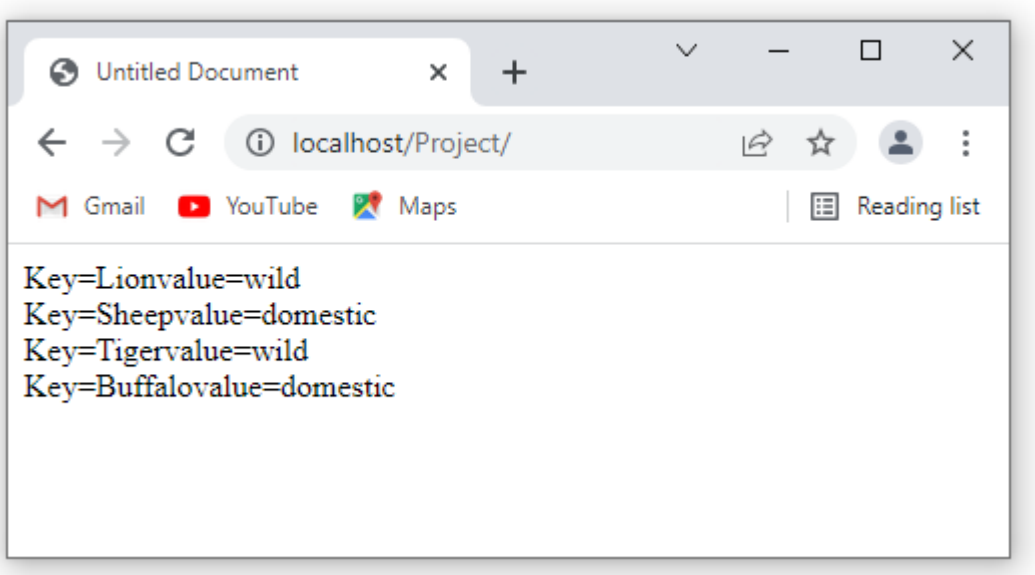
OR
$array_name =["key1"=>value1, "key2" =>value2, "key3" =>value3, "key4"=> value4];

Example:

```
<?php
        $animals = array(“Lion” =>”wild”, “sheep”=> “domestic”, “Tiger” =>”wild”, “Buffalo” =>”domestic”);
foreach($animals as $key => $value)
echo “key=”.$key,”value=”.$value;
```

```
echo "</br>";
}
?>
```

Out Put:



III. Multi-dimensional Array

☞ A multi-dimensional array is an array which stores another array at each index rather than storing a single value. It is an array of arrays. These are arrays that contain other nested arrays. The advantage of multidimensional array is that they allow us to group related data together. There is no limit in the level of dimensions.

Syntax:

```
$array = array(
                array(value-1, value-2,……..,value-n),
                array(value-1, value-2,……..,value-n),
                array(value-1, value-2,……..,value-n)

                );

$array = [
                Array[value-1, value-2,……..,value-n],
                Array[value-1, value-2,……..,value-n],
                Array[value-1, value-2,……..,value-n]

                ];
```

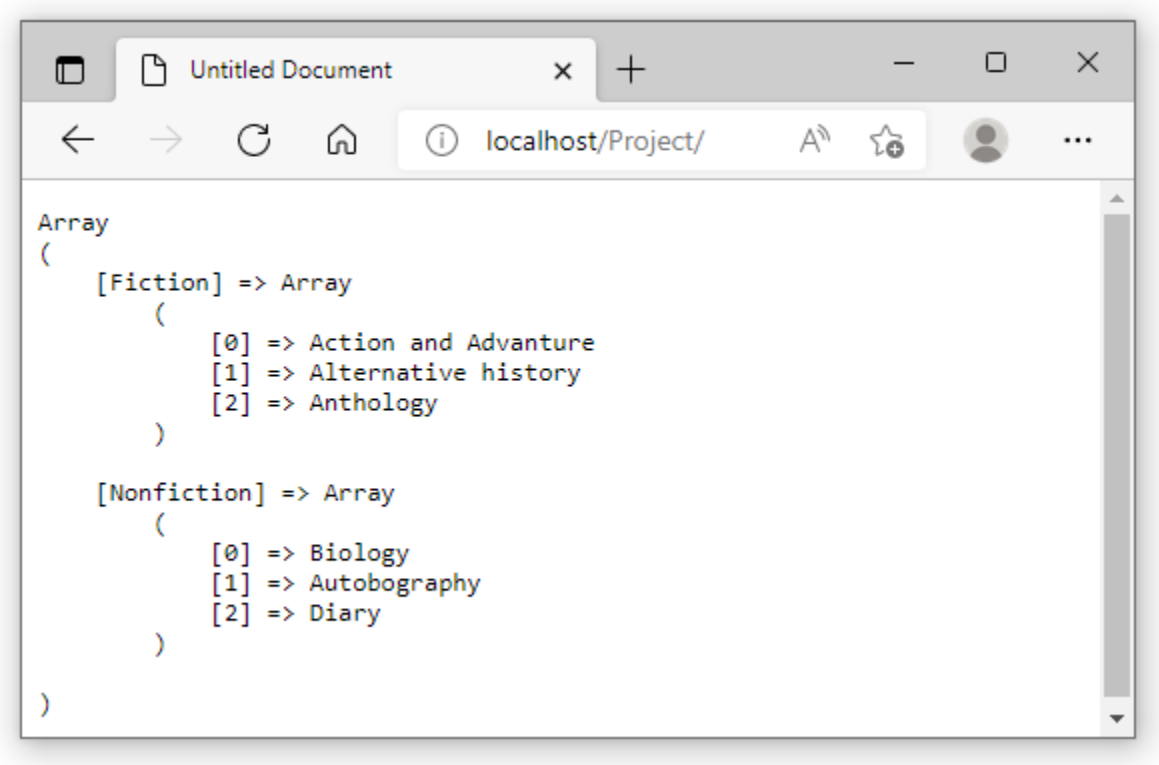
```
$array =array(
                    Key => array(
                        Key => value-1;
                        Key => value-2;
                        ……..,
                        Key => value-n
                    ),

                    Key => array(
                        Key => value-1;
                        Key => value-2;
                        ……..,
                        Key => value-n),

);

$array =array[
                    Key => [
                        Key => value-1;
                        Key => value-2;
                        ……..,
                        Key => value-n
                        ],
                    Key => [
                        Key => value-1;
                        Key => value-2;
                        ……..,
                        Key => value-n],


];
```

Example:

```
<?php
    $movies=array(
    "Fiction" =>array("Action and Advanture","Alternative history","Anthology"),
    "Nonfiction" =>array("Biology","Autobography","Diary")
    );
    echo '<pre>';
```

```
        print_r($movies);
        echo '</pre>';

?>
```

Out Put:



FUNCTIONS

- Function is a set of instruction which performs some kinds of task.
- In other word function is nothing but a 'block of statement' Which generally performs a specific task and can be used repeatedly in our program.
- Function can be called with their name any where any time in a program.
- The advantage of using function is that they are reusable.
- Function can be defined by user or by PHP program.
- A function doesn't execute when it's defined, it executed when it is called.
- The built-in functions are part of the PHP language. Examples are: print-r(), var_dump(), phpinfo(), isset() or empty(). The user defined functions are created by application programmers as per their needs. They are created with the **function** keyword.

User Defined Functions

- A function defined to solve specific requirement is known as user defined function.
- The naming rules to define user defined functions are:
  - Function name must start with a letter or an underscore but not a number.
  - The function name must be unique.
  - The function name must not contain spaces.
  - It is considered a good practice to use descriptive function names.
  - Functions can optionally accept parameters and return values too.

Syntax:

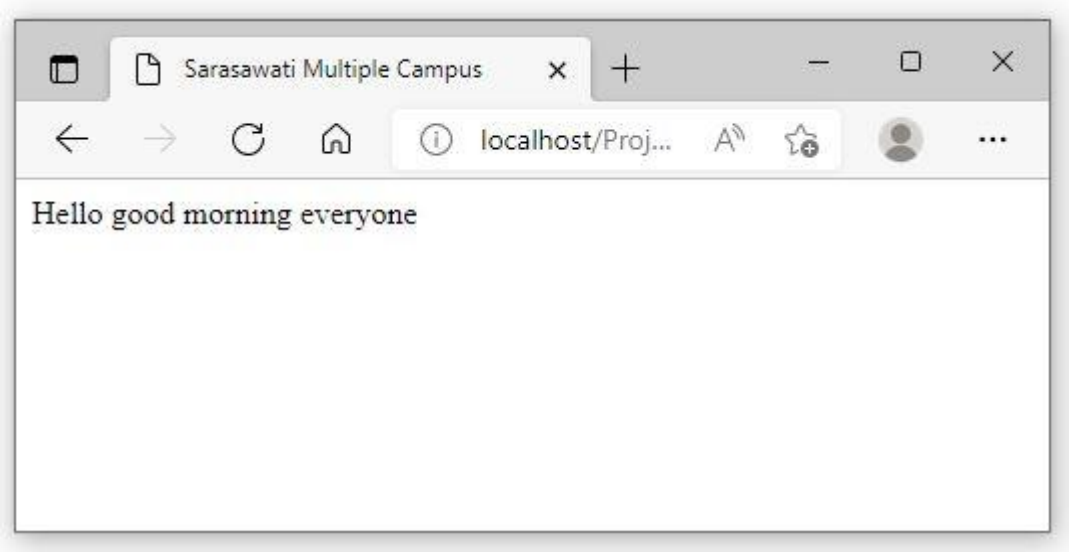
```
<?php
function functionName()
{
        //code to be executed
}
?>
```

Example:

Example program to demonstrate function as

```
<body>
<?php
//Defining a function
function sayHello()
{
        echo "Hello good morning everyone";
}
//Calling Function
sayHello();
?>
</body>
```

Out Put:



Function with Parameter

- We can specify parameter when we define a function to accept input values at run time.
- The parameter work like a place holder variables within a function. They are replaced at run time by the values (Known as argument) provided to the function at the time of invocation.

Syntax:

```
<?php
function function_name($parameter-1,$parameter-2, $parameter-3,…….)
{
        //code to be executed
}
?>
```

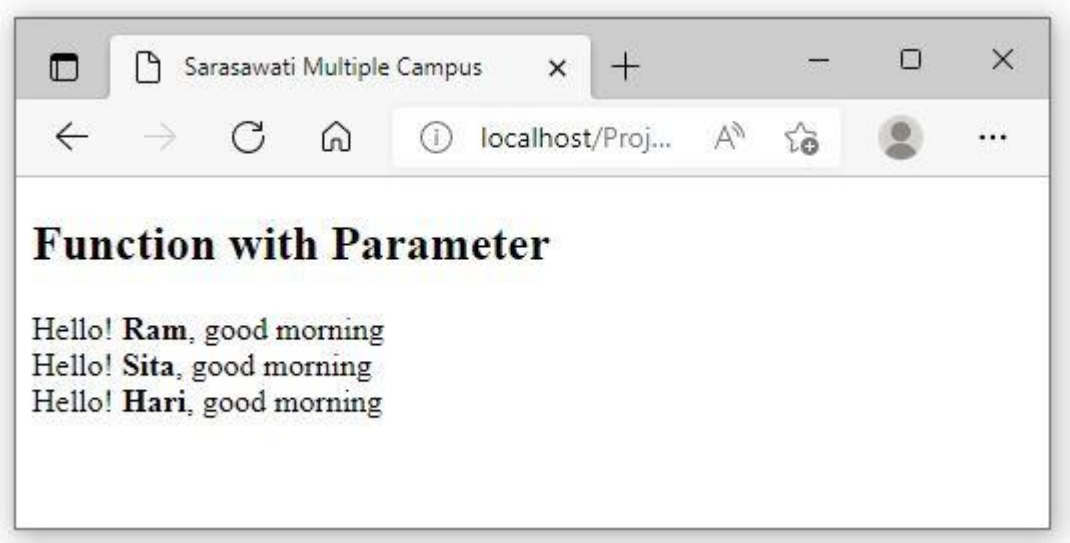
Example:

Example program to demonstrate function with parameter

```
<body>
<h2>Function with Parameter</h2>
<?php
//Defining a function
function sayHello($user)
{
```

```
        echo "Hello!<b> $user</b>, good morning<br>";
}
//Calling Function
sayHello("Ram");
sayHello("Sita");
sayHello("Hari");
?>


</body>
```

Out Put:



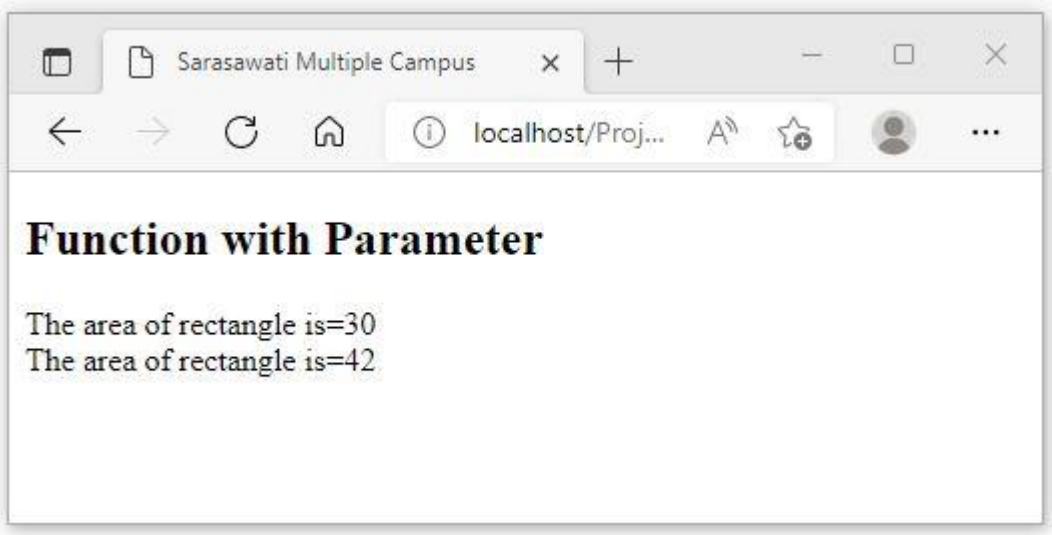
Example-2

Program to find the area of rectangle box?

```
<body>
<h2>Function with Parameter</h2>
<?php
//Defining a function
function Area($l,$b)
{
        $A=$l*$b;

        echo "The area of rectangle is=".$A."</br>";
}
//Calling Function
Area(5,6);
Area(7,6);
```

```
?>


</body>
```

Out Put:



Function with Optional Parameter and Default Value

- ☞ We can also create function with optional parameters for that, just insert parameter name, followed by an equal sign (=), followed by a default value.

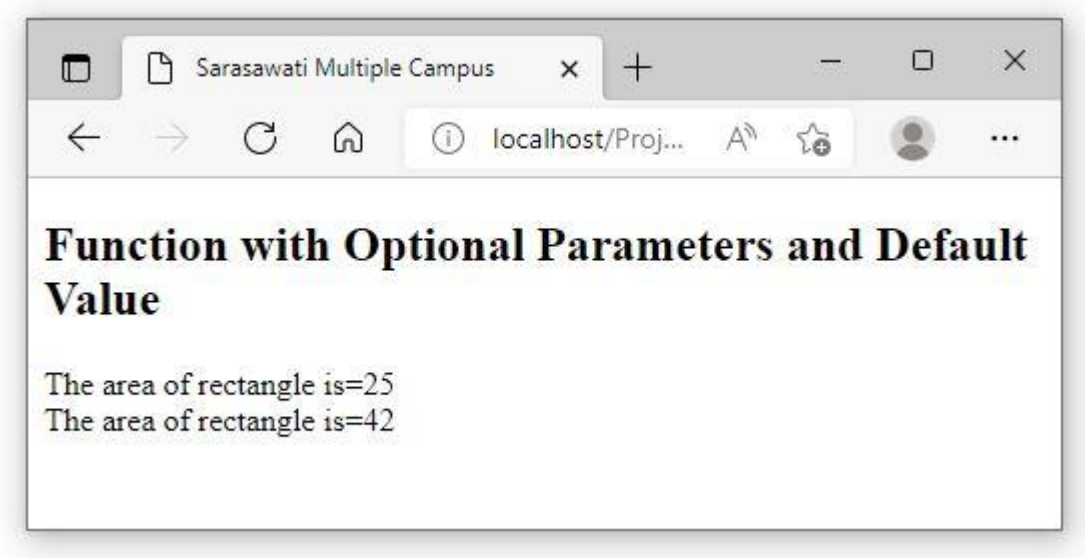
Example:

Example program to demonstrate function with optional parameter and default value.

```
<body>
<h2>Function with Optional Parameters and Default Value</h2>
<?php
//Defining a function
function Area($l,$b=5)
{
        $A=$l*$b;

        echo "The area of rectangle is=".$A."</br>";
}
//Calling Function
Area(5);
Area(7,6);
?>
```

```
</body>
</html>
```

Out Put:



Returning value from a function

- ☞ A function can return a value back to the script that called the function using the return statement. The value may be of any type, including arrays and objects.

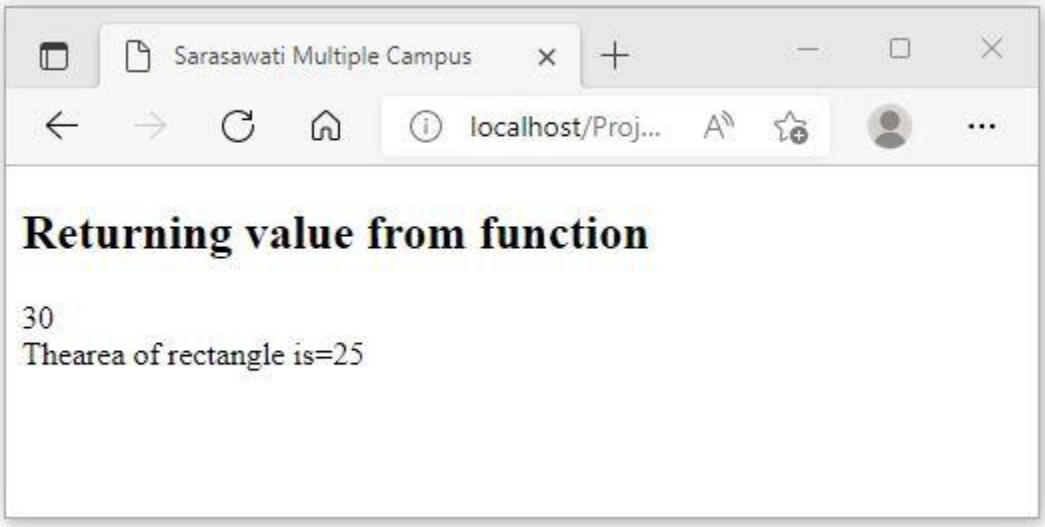
Example:

Example program to demonstrate function with return value.

```
<body>
<h2>Returning value from function</h2>
<?php
//Defining a function
function Area($l,$b)
{
                                        $A=$l*$b;
                                        return $A;

}
//Printing return value

echo Area(5,6);
echo "</br>";
$A1=Area(5,5);
```

```
echo "Thearea of rectangle is=".$A1;
?>
</body>
```

Out Put:



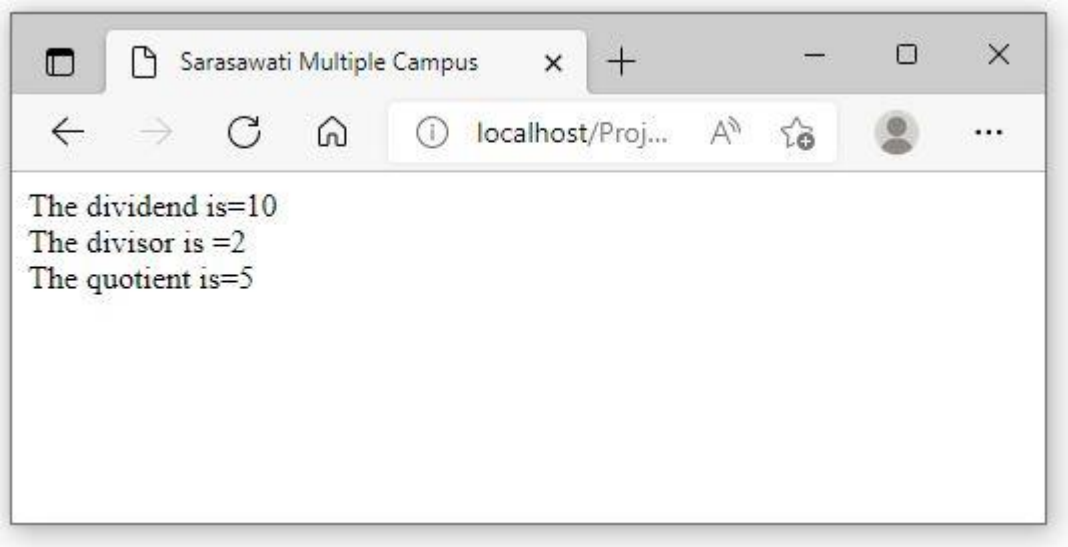
Note:

A function cannot return multiple values, However, we can obtain similar results by returning an array.

```
<body>
<?php
//Defining a function
function divideNumbers($dividend, $divisor)
{
        $quotient = $dividend / $divisor;
        $array = array($dividend, $divisor, $quotient);
        return $array;
}
//Assign variables as if they were an array
list($dividend, $divisor,$quotient)= divideNumbers(10,2);
echo "The dividend is=".$dividend;
```

```
echo "<br>";
echo "The divisor is =".$divisor;
echo "<br>";
echo "The quotient is=". $quotient;
?>
</body>
```

Out Put:



PHP function Overloading

- ☞ Function Overloading means function has same name but different signature.
- ☞ Function overloading is a feature that permits making creating several methods with a similar name that works differently from one another in the type of the input parameters it accepts as arguments.

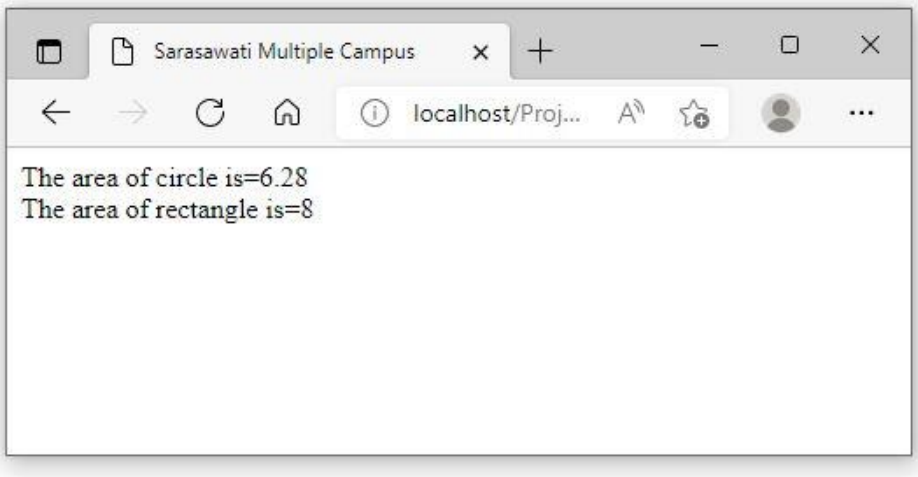
Example:

Example program to demonstrate function overloading.

```
<body>
<?php
class shape {

  // __call is magic function which accepts
  // function name and arguments
  function __call($name_of_function, $arguments) {
```

```
        // It will match the function name
        if($name_of_function == 'area') {

            switch (count($arguments)) {

                // If there is only one argument
                // area of circle
                case 1:
                    return 3.14 * $arguments[0];

                // IF two arguments then area is rectangle;
                case 2:
                    return $arguments[0]*$arguments[1];
            }
        }
    }
}

// Declaring a shape type object
$s = new Shape;

// Function call
echo("The area of circle is=".$s->area(2));
echo "<br>";

// calling area method for rectangle
echo ("The area of rectangle is=".$s->area(4, 2));
?>
</body>
```

Out Put:



Extracting function argument using built-in PHP functions

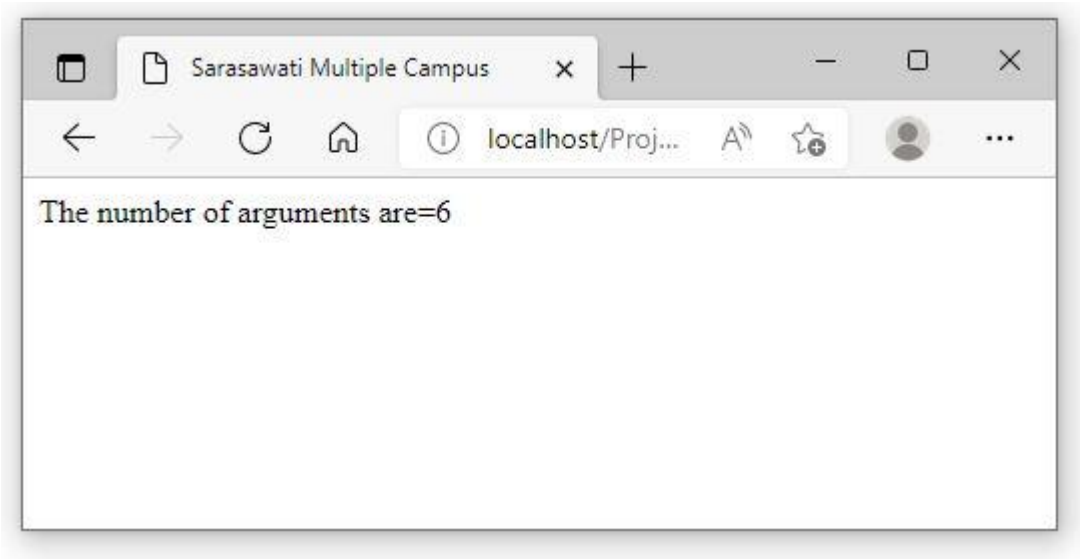
I. func_num_args

☞ This function is used to returns the numbers of arguments passed to the function.

Example:

```
<body>
<?php
function getNumberOfArguments()
{
	$numargs = func_num_args();
	echo "The number of arguments are=".$numargs;
}
getNumberOfArguments(10,20,30,40,50,60);
?>
</body>
```

Out Put:



II. func_get_arg

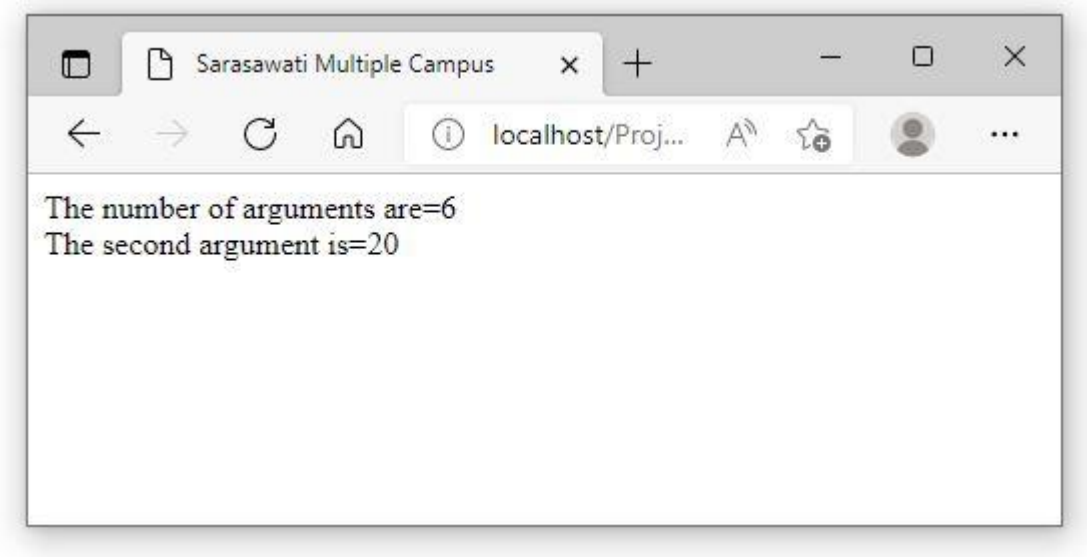
☞ This function is used to get the specified argument from a user defined function's argument list. Start with 0.

Example:

```
<body>
<?php
function getSpecifiedParameter()
{
	$numargs = func_num_args();
```

```
        echo "The number of arguments are=".$numargs."<br>";
         if($numargs >=2)
           {
               echo "The second argument is=".func_get_arg(1)."<br>";
            }
}
getSpecifiedParameter(10,20,30,40,50,60);
?>
</body>
```

Out Put:



t

III. func_get_args

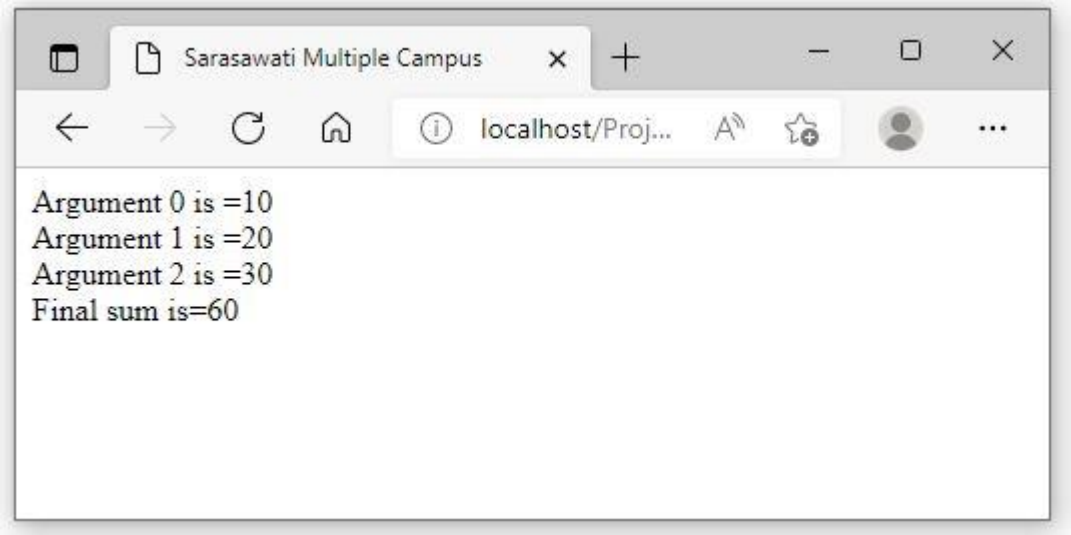
- ☞ This function is used to return an array in which each element is a copy of the corresponding number of the current user-defined function's argument list.
- ☞ It will generate warning if called from outside of a function definition. This function cannot be used directly as a function parameter. Instead, its result may be assigned to a variable, which cannot be passed to a function.
- ☞ This function may be used in conjunction with func_get_arg() and func_num_args() to all user defined function to accept variable length argument list.

Example:

```
<body>
<?php
function sumAllParam()
{
        $numargs = func_num_args();
       $arg_list = func_get_args();
```

```
        $sum = 0;
        for($i =0; $i <$numargs ; $i++)
            {
                $sum = $sum + $arg_list[$i];
               echo "Argument $i is =".$arg_list[$i]."<br>";
            }
    echo "Final sum is=".$sum;
}
    sumAllParam(10,20,30);
?>
</body>
```

Out Put:



PHP variable and function scope

- Scope can be defined as the range of availability a variable has to the program in which it is declared.
- The scope of a variable in PHP is the context in which the variable was created and in which it can be accessed.
- Generally, there are four types of scope in PHP variable.

i. Local variables
ii. Functions parameters
iii. Global variables
iv. Static variables

i. Local variables

- A variable which is declared within function is called local variable.

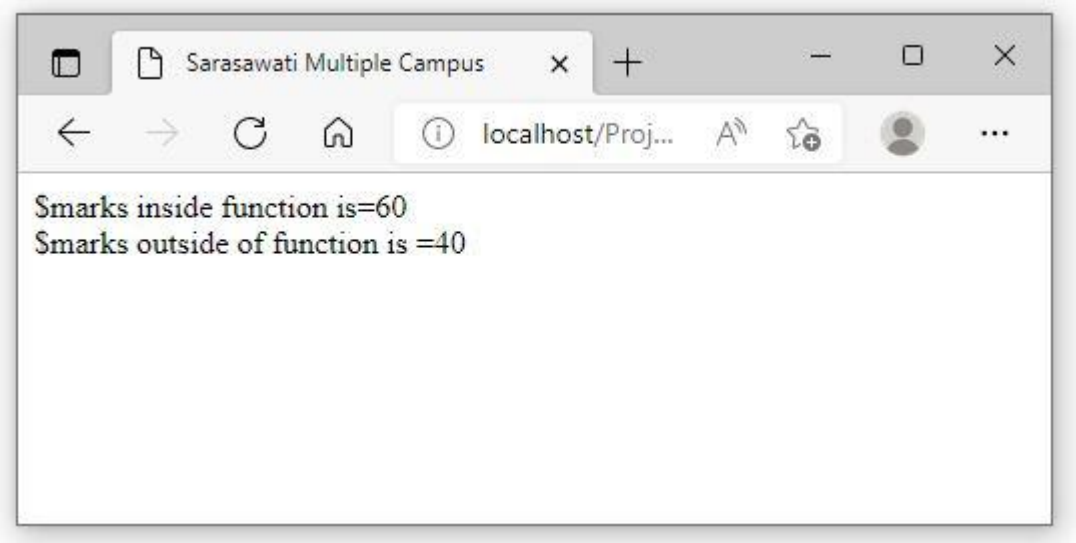
- ☞ Any assignment outside of that function will be considered to be a entirely different variable from the one contained in the function.

Example:

Example program to demonstrate local variable as:

```
<body>
<?php
$marks =40;
function assignMarks()
{
        $marks=60;
        print "\$marks inside function is=".$marks;
        echo "<br>";

}
assignMarks();
print "\$marks outside of function is =".$marks;
?>
</body>
```

Out Put:



ii. Functions parameters

- ☞ Functions parameters are declared after the function name and inside the parentheses.

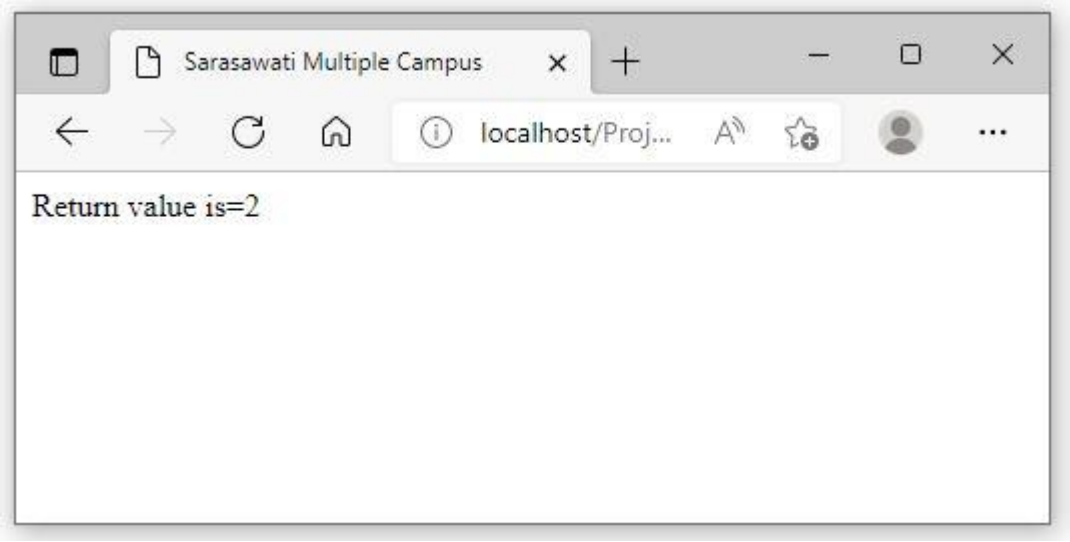
Example:

Example program to demonstrate the functions parameters.

```
<body>
<?php
//find reminder a value by divided by 3 and return it to the caller
```

```
function findReminder($value)
{
                $value=$value%3;
                return $value;
}
$retval = findReminder(11);
print "Return value is=".$retval;
?>
</body>
```

Out Put:



iii. Global variables

- ☞ A global variable is a variable which can accessed in any part of the program.
- ☞ However, a global variable must be explicitly declared to be global in the function in which it is to be modified.
- ☞ By using the keyword GLOBAL in front of the variable that should be recognized as global.

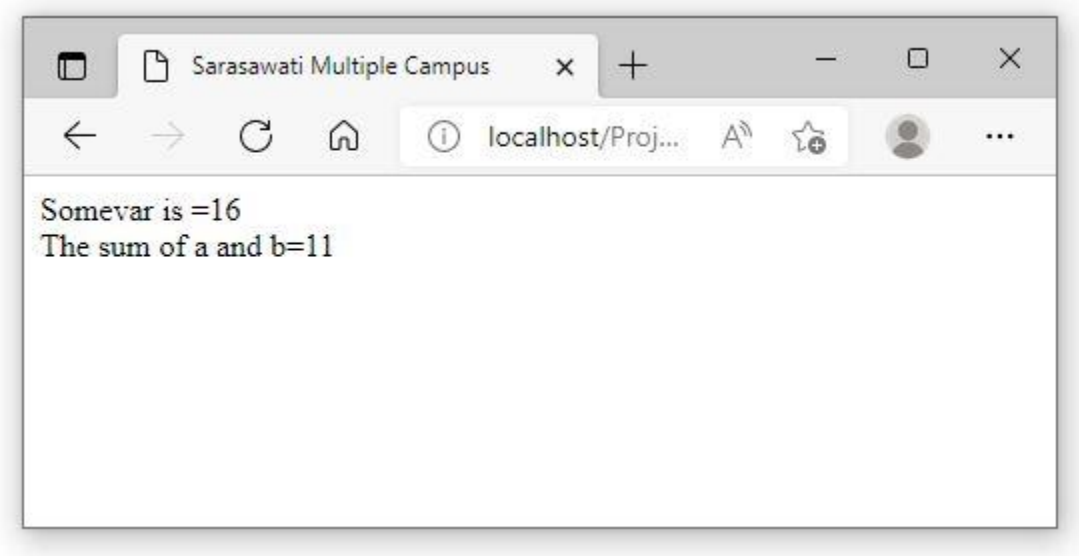
Example:

Example program to demonstrate Global variable.

```
<body>
<?php
$first_num =15;
function add()
{
                global $first_num;
                $first_num ++;
```

```
                print "Somevar is =$first_num";
                echo "<br>";
}
add();
$a=5;
$b=6;
function Sum()
{
                global $a,$b;
                $b=$a + $b;
}
Sum();
echo $b;
?>
</body>
```

Out Put:



iv. Static Variable

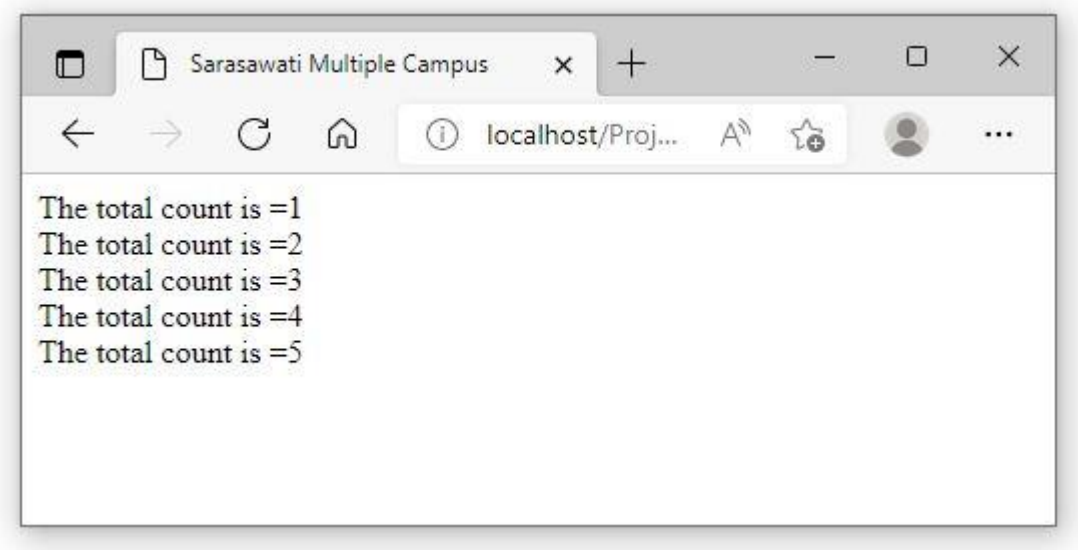
- ☞ In contrast to the declared as function parameters, which are destroyed on the function's exit, a static variable will not lose its value when the function exits and will still hold that value should the function be called again.
- ☞ We can declare a variable to be static simply using keyword STATIC in front of the variable name.

Example:

Example program to demonstrate static variable.

```
<body>
<?php
function keep_track()
{
                    STATIC $count =0;
                    $count ++;
                    print "The total count is =".$count;
                    print "<br>";
}
keep_track();
keep_track();
keep_track();
keep_track();
keep_track();
?>
</body>
```

Out Put:



Some PHP Array Functions

1. **Merge multiple array into single one**
   - ☞ The array_merge() function is used to merge list of arrays.
   - ☞ It can take more parameters and return back into single one.

   Syntax:

   array_merge(array $array1[,array $array2[,array $array3.....]]);

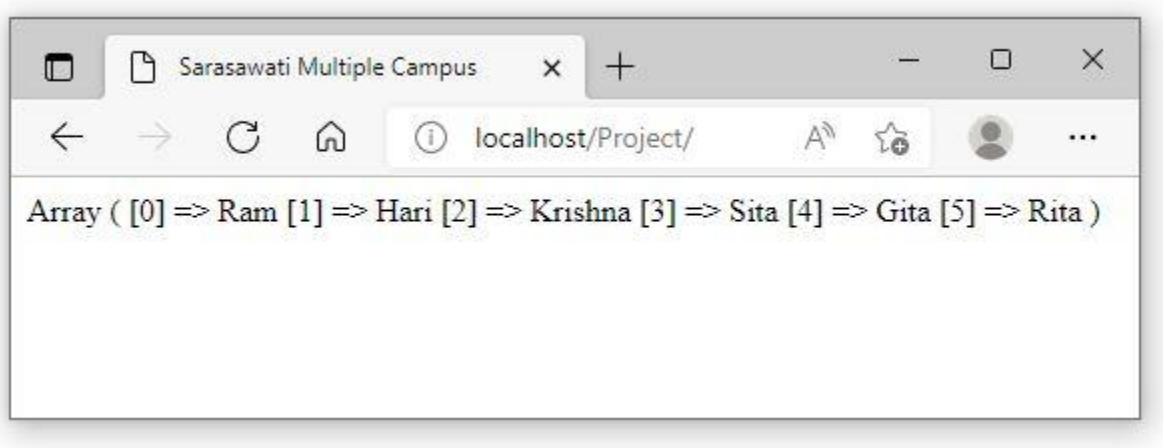
Example:

Example program to demonstrate Multiple arrays into single one.

```
<body>
<?php
$boys = array('Ram','Hari','Krishna');
$girls = array('Sita','Gita','Rita');
//now merge array of boys and girls into single array and call then student of single class
$students = array_merge($boys,$girls);
print_r($students);
?>
</body>
```

Out Put:

The same



2. **Merge array recursively**
   - ☞ It is used to merge the elements of one or more arrays together so that the value of one are appended to the end of previous one.
   - ☞ It the input arrays have the same string keys, then the values for these keys are merged together into an array and this is done recursively, so that if one of the values is an array itself, the function will merge it with a corresponding entry in another array too.

   Syntax:

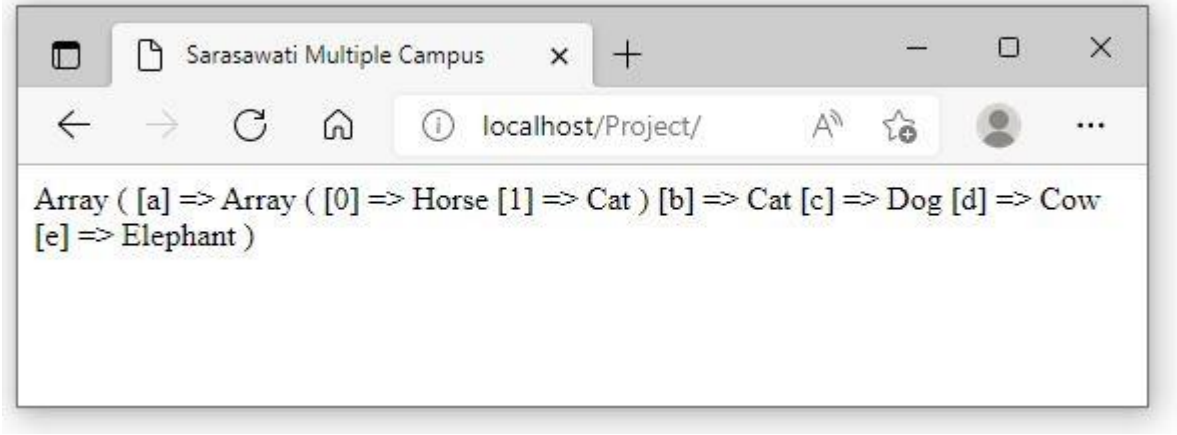
   Array_merge_recursive(array $array1[,array $array2……])

Example:

Example program to demonstrate Merge array recursively

```
<body>
<?php
$array1 =array("a"=>"Horse","b"=>"Cat","c"=>"Dog");
$array2 =array("d"=>"Cow","a"=>"Cat","e"=>"Elephant");
print_r(array_merge_recursive($array1,$array2));
?>
</body>
```

Out Put:



3.