Unit-IV Functions in PHP

Functions:

- A Function in PHP is a reusable piece or block of code that performs a specific action.
- It takes input from the user in the form of parameters, performs certain actions, and gives the output.
- Functions can either return values when called or can simply perform an operation without returning any value.

Why use Functions?

- **Better code organization –** PHP functions allow us to group blocks of related code that perform a specific task together.
- **Reusability –** once defined, a function can be called by a number of scripts in our PHP files. This saves us time of reinventing the wheel when we want to perform some routine tasks such as connecting to the database
- **Easy maintenance**- updates to the system only need to be made in one place.

Type of the Functions?

- Functions are of two types: (1) Built-in functions (2) User Defined Functions

Built-In Functions of PHP

- Built-in functions are the functions that are provided by PHP to make programming more convenient.
- Many built-in functions are present in PHP which can be used in our code. These functions are very helpful in achieving programming goals.
- The built-in functions in PHP are very simple and easy to use. They make PHP powerful and useful.
- PHP is very rich in terms of Built-in functions.

Array Function

Function Name	Description
sizeof($arr)	Returns the size of the array
is_array($arr)	It returns true if the variable is an array and returns false otherwise.
in_array($var, $arr)	To check whether a certain value is present in the array or not. It returns true if the variable is in array and returns false otherwise.
array_merge($arr1, $arr2)	Combine two different arrays into a single array

array_values($arr)	Retrieve only value from array
array_keys($arr)	Retrieve only key from array
array_pop($arr)	Removes the last element of the array
array_push($arr, $val)	Add a new element at the end of the array.
sort($arr)	Sorts the array elements in ascending order
array_flip($arr)	Interchange the keys and the values
array_reverse($arr)	Reverse the order of elements

String Function

Function Name	Description
strlen($str)	Returns the length of the string
str_word_count($str)	Returns the number of words in the string
strrev($str)	Used to reverse a string
strpos($str, $text)	Used to find the position of any text/word in a given string. string also assign index value to the characters stored in it, starting from zero.
str_replace($replacethis, $replacewith, $str)	Used to replace a part of the string with some text.
ucwords($str)	converts first letter of every word in the string to uppercase.
strtoupper($str)	To convert string to uppercase
strtolower($str)	To convert string to lowercase
str_repeat($str, $nooftime)	repeat a string a given number of times
strcmp($str1, $str2)	Used to compare two strings. The comparison is done alphabetically. If the first string is greater than second string, the result will be greater than 0, if the first string is equal to the second string, the result will be equal to 0 and if the second string is greater than the first string, then the result will be less than 0.
substr($str, $start, $length)	Find the substring from given string. Used to take out a part of the string(substring), starting from a particular position, of a particular length.
trim($str)	Used to remove extra whitespaces from beginning and the end of a string.

PDF Function

- FPDF is a PHP class which allows generating PDF files with PHP code. It is free to use.
- FPDF stands for Free PDF. It means that any kind of modification can be done in PDF files.
- The latest version requires at least PHP 5.1 and is compatible with PHP 7 and PHP 8.

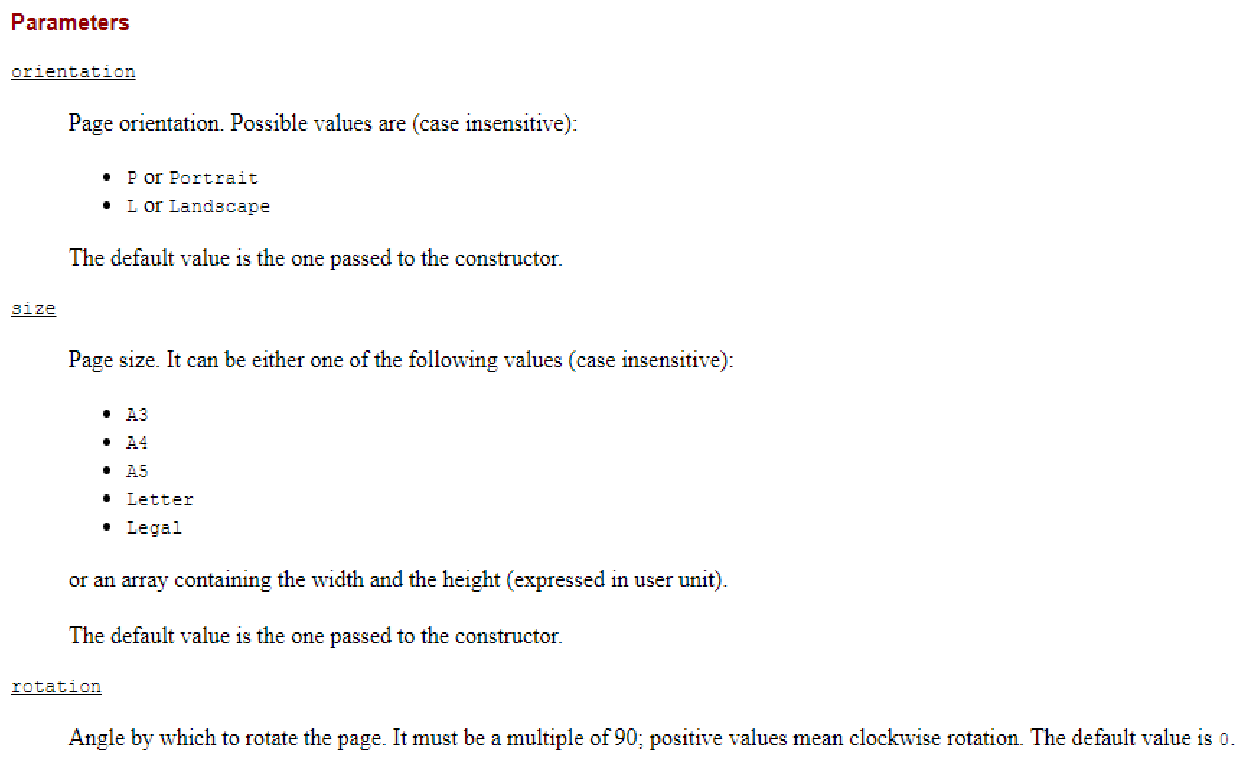
Note: Download the latest version of this Class from http://www.fpdf.org/en/download.php

- FPDF has other benefits: high level functions. Here is a list of its main features:
    - Choice of measure unit, page format and margins
    - Page header and footer management
    - Automatic page break
    - Automatic line break and text justification
    - Image support (JPEG, PNG and GIF)
    - Colors
    - Links
    - TrueType, Type1 and encoding support
    - Page compression

AddPage()

- Adds a new page to the document.

Syntax: AddPage([string orientation [, mixed size [, int rotation]]])



Parameters

`orientation`

Page orientation. Possible values are (case insensitive):

- `P` or `Portrait`
- `L` or `Landscape`

The default value is the one passed to the constructor.

`size`

Page size. It can be either one of the following values (case insensitive):

- `A3`
- `A4`
- `A5`
- `Letter`
- `Legal`

or an array containing the width and the height (expressed in user unit).

The default value is the one passed to the constructor.

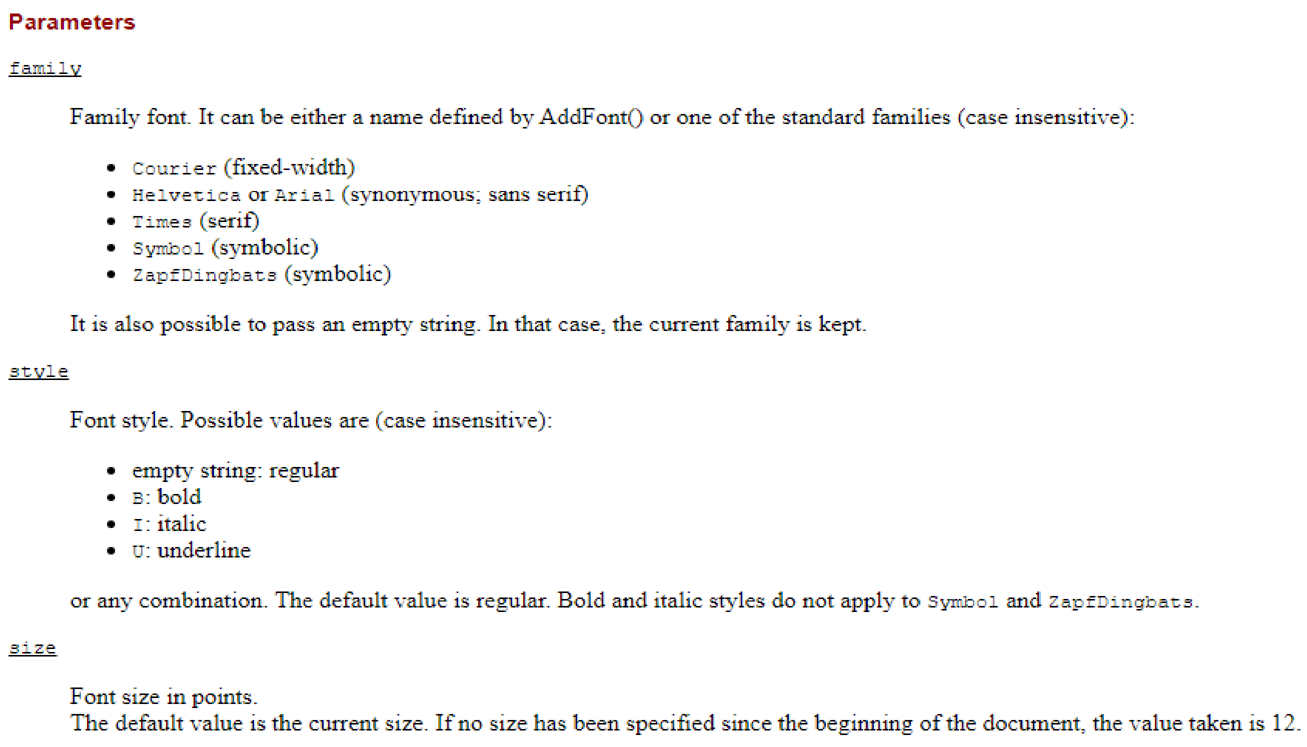
`rotation`

Angle by which to rotate the page. It must be a multiple of 90; positive values mean clockwise rotation. The default value is `0`.

SetFont()

- Sets the font used to print character strings.
- It is mandatory to call this method at least once before printing text or the resulting document would not be valid.

Syntax: SetFont(string family [, string style [, float size]])



Parameters

family

Family font. It can be either a name defined by AddFont() or one of the standard families (case insensitive):

- `Courier` (fixed-width)
- `Helvetica` or `Arial` (synonymous; sans serif)
- `Times` (serif)
- `Symbol` (symbolic)
- `ZapfDingbats` (symbolic)

It is also possible to pass an empty string. In that case, the current family is kept.

style

Font style. Possible values are (case insensitive):

- empty string: regular
- `B`: bold
- `I`: italic
- `U`: underline

or any combination. The default value is regular. Bold and italic styles do not apply to `Symbol` and `ZapfDingbats`.

size

Font size in points.
The default value is the current size. If no size has been specified since the beginning of the document, the value taken is 12.

SetFontSize()

- Defines the size of the current font.

Syntax: SetFontSize(float size) where size is the font size.

Cell()

- Prints a cell (rectangular area) with optional borders, background color and character string. The upper-left corner of the cell corresponds to the current position.
- The text can be aligned or centered.
- After the call, the current position moves to the right or to the next line. It is possible to put a link on the text.
- If automatic page breaking is enabled and the cell goes beyond the limit, a page break is done before outputting.

Syntax: Cell(float w [, float h [, string txt [, mixed border [, int ln [, string align [, boolean fill [, mixed link]]]]]]])

Parameters

`w`

Cell width. If `0`, the cell extends up to the right margin.

`h`

Cell height. Default value: `0`.

`txt`

String to print. Default value: empty string.

`border`

Indicates if borders must be drawn around the cell. The value can be either a number:

- `0`: no border
- `1`: frame

or a string containing some or all of the following characters (in any order):

- `L`: left
- `T`: top
- `R`: right
- `B`: bottom

Default value: `0`.

`ln`

Indicates where the current position should go after the call. Possible values are:

- `0`: to the right
- `1`: to the beginning of the next line
- `2`: below

Putting `1` is equivalent to putting `0` and calling Ln() just after. Default value: `0`.

`align`

Allows to center or align the text. Possible values are:

- `L` or empty string: left align (default value)
- `C`: center
- `R`: right align

`fill`

Indicates if the cell background must be painted (`true`) or transparent (`false`). Default value: `false`.

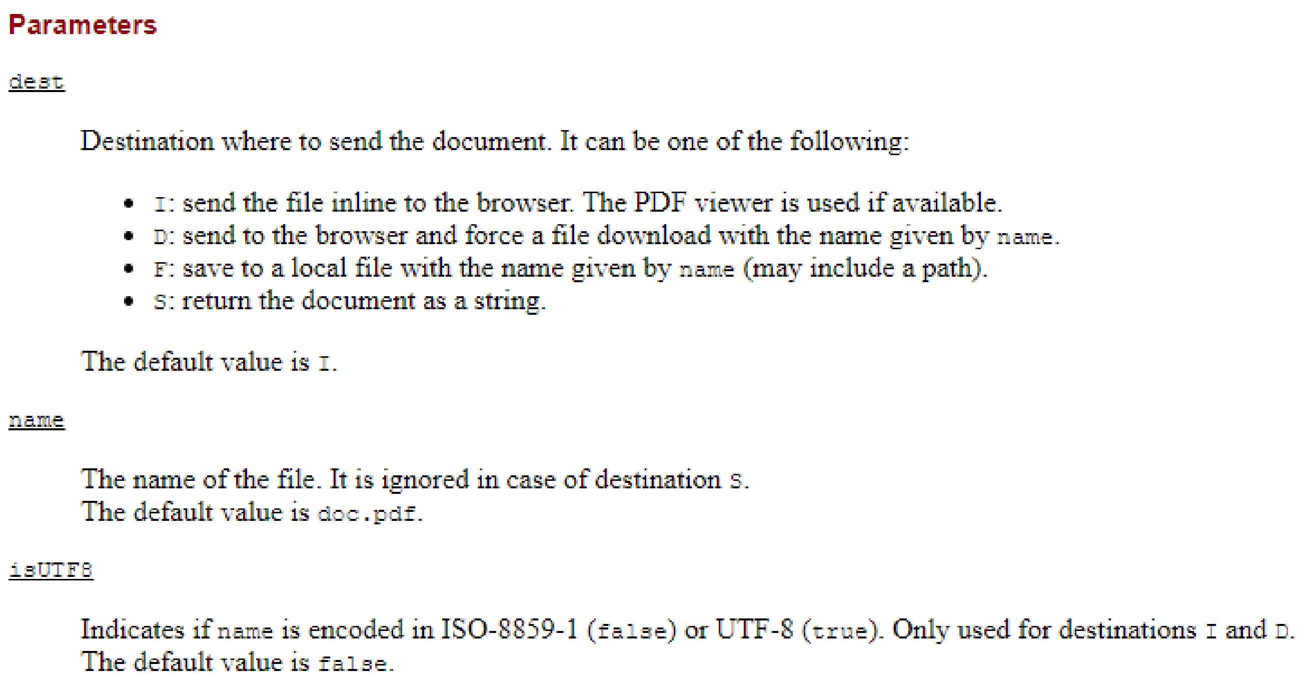
`link`

URL or identifier returned by AddLink().

Output()

- Send the document to a given destination: browser, file or string. In the case of a browser, the PDF viewer may be used or a download may be forced.

Syntax: Output([string dest [, string name [, boolean isUTF8]]])



Parameters

`dest`

Destination where to send the document. It can be one of the following:

- `I`: send the file inline to the browser. The PDF viewer is used if available.
- `D`: send to the browser and force a file download with the name given by `name`.
- `F`: save to a local file with the name given by `name` (may include a path).
- `S`: return the document as a string.

The default value is `I`.

`name`

The name of the file. It is ignored in case of destination `S`.
The default value is `doc.pdf`.

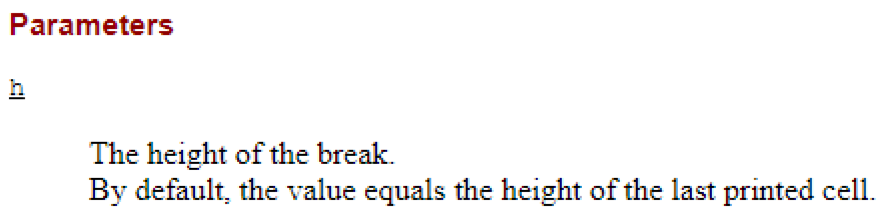
`isUTF8`

Indicates if `name` is encoded in ISO-8859-1 (`false`) or UTF-8 (`true`). Only used for destinations `I` and `D`.
The default value is `false`.

Ln()

- Performs a line break.

Syntax: Ln([float h])



Parameters

`h`

The height of the break.
By default, the value equals the height of the last printed cell.

SetFillColor()

- Defines the color used for all filling operations (filled rectangles and cell backgrounds).
- It can be expressed in RGB components or gray scale.
- The method can be called before the first page is created and the value is retained from page to page.

Syntax: SetFillColor(int r [, int g, int b])

Parameters

r

If g and b are given, red component; if not, indicates the gray level. Value between 0 and 255.

g

Green component (between 0 and 255).

b

Blue component (between 0 and 255).

SetTextColor()

- Defines the color used for text.
- It can be expressed in RGB components or gray scale.
- The method can be called before the first page is created and the value is retained from page to page.

Syntax: SetTextColor(int r [, int g, int b])

Parameters

r

If g and b are given, red component; if not, indicates the gray level. Value between 0 and 255.

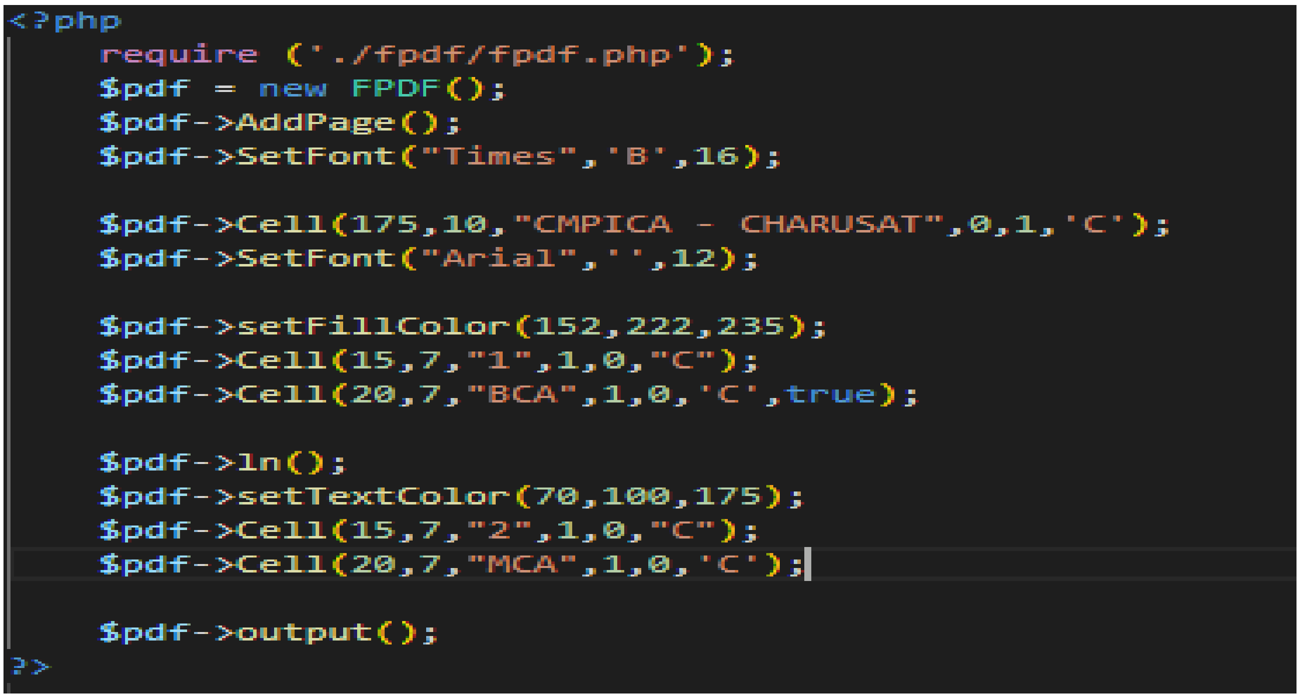
g

Green component (between 0 and 255).

b

Blue component (between 0 and 255).

Example:



```
<?php
    require ('./fpdf/fpdf.php');
    $pdf = new FPDF();
    $pdf->AddPage();
    $pdf->SetFont("Times",'B',16);

    $pdf->Cell(175,10,"CMPICA - CHARUSAT",0,1,'C');
    $pdf->SetFont("Arial",'',12);

    $pdf->setFillColor(152,222,235);
    $pdf->Cell(15,7,"1",1,0,"C");
    $pdf->Cell(20,7,"BCA",1,0,'C',true);

    $pdf->ln();
    $pdf->setTextColor(70,100,175);
    $pdf->Cell(15,7,"2",1,0,"C");
    $pdf->Cell(20,7,"MCA",1,0,'C');

    $pdf->output();
?>
```

File Handling Function

Function Syntax	Use	Parameter	Returns
bool **file_exists** (string $filename);	Checks whether a file or directory exists	Path of file or directory	TRUE if the file or directory specified by filename exists FALSE otherwise
bool **is_file** (string $filename);	Tells whether the filename is a regular file	Path to the file	TRUE if the filename exists and is a regular file. FALSE otherwise
bool **is_dir** (string $filename);	Tells whether the filename is a directory	Path to the directory	TRUE if the filename exists and is a directory FALSE otherwise
bool **is_readable** (string $filename)	Tells whether a file exists and is readable	Path to the file	TRUE if the file or directory specified by filename exists and is readable FALSE otherwise
bool **is_writable** (string $filename)	Tells whether the filename is writable	The filename being checked	Returns TRUE if the filename exists and is writable.
bool **is_executable** (string $filename)	Tells whether the filename is executable	Path to the file	TRUE if the filename exists and is executable, or FALSE on error
int **filesize** (string $filename)	Gets the size for the given file.	Path to the file	Size of the file in bytes or FALSE and generates an error of level E_WARNING in case of an error
int **fileatime** (string $filename)	Gets the last access time of the given file.	Path to the file	Time (as a Unix timestamp) the file was last accessed, or FALSE on failure.
int **filemtime** (string $filename)	Gets file modification time	Path to the file	Time (as a Unix timestamp) the file was last modified, or FALSE on failure.
int **filectime** (string $filename)	Gets last change time of file	Path to the file	Time (as a Unix timestamp) the file was last changed, or FALSE on failure.
bool touch (string $filename)	Create an empty file	Path to the file	TRUE on success, FALSE on fail
bool unlink (string $filename)	Remove file	Path to the file	TRUE on success, FALSE on fail

----

File Handling Function (Reading and Writing)

Function	Syntax	Description
fopen()	fopen(filename,mode)	The fopen() function is used to open file for reading and writing purpose base on mode. The first parameter is filename and second parameter is mode of file.
fread()	fread(filename,filesize)	The fread() function reads from an open file. The first parameter contains the name of the file to read from and the second parameter specifies the maximum number of bytes to read.
fclose()	fclose(filename)	The fclose() function is used to close an open file. The fclose() requires the name of the file (or a variable that holds the filename) we want to close:
fgets()	fgets(filename)	The fgets() function is used to read a single line from a file. The fgets() requires the name of the file (or a variable that holds the filename) from which we want to read data.
feof()	feof(filename)	The feof() function checks if the "end-of-file" (EOF) has been reached. The feof() function is useful for looping through data of unknown length.
fgetc()	fgetc(filename)	The fgetc() function is used to read a single character from a file.
fwrite()	fwrite(filename,data)	The fwrite() function is used to write to a file. The first parameter of fwrite() contains the name of the file to write to and the second parameter is the string to be written.
fseek()	fseek(filename, offset, whence)	This function moves the file pointer from its current position to a new position, forward or backward, specified by the number of bytes.
file_get_contents()	file_get_contents(filename)	Reading whole file into a string.
file_put_contents()	file_put_contents(filename,data)	Writing whole data inti a file.

File opening modes

mode	**Purpose**	**File Pointer**	**Other**
r	Reading only	Beginning of file	NA
r+	Reading and writing		
w	Writing only	Beginning of file	Attempts to create, if file doesn't exists Content will be truncated, if file exists
w+	Reading and writing		
a	Writing only	End of file	Attempts to create, if file doesn't exists
a+	Reading and writing		
x	Create and open for writing only	Beginning of file	Attempts to create, if file doesn't exists Generate an error of level E_WARNING, if file exists
x+	Create and open for reading and writing		

CSV File Handling Function

User Define Functions of PHP

- You can define your own functions using the **function statement**

Syntax

```
function function_name($argument1, $argument2 . . .) {
//function code here
}
```

- The name of the function follows the function statement and precedes a set of parentheses.
- If your function requires arguments, you must place comma separated variable names (function parameters) within the parentheses.
- These variables will be filled by the values passed to your function.
- Even if your function doesn't require arguments, you must nevertheless supply the parentheses.

Example

```
<?php
        function message($msg) // here $msg is formal parameter
        {
                echo "<br>" . $msg;
        }
        message("hello"); // here “Hello” actual parameter
        message ("how are you");
?>
```

Note: The naming rules for functions are similar to the naming rules for variables except one that is, unlike variable name function name doesn't start with $ sign.

RETURNING VALUES FROM USER-DEFINED FUNCTIONS

- A function can return a value using the return statement.
- The return statement stops the execution of the function and sends the value back to the calling code.

Example

```
<?php
        function add($n1,$n2){
                $ans = $n1 + $n2;
                return $ans;
        }
```

```
          echo "<br>Answer = ".add(10,20);
          echo "<br>Answer = ".add(2.50,3);
          echo "<br>Answer = ".add("3","2hello");
?>
```

- The return statement can return a value or nothing at all. How we arrive at a value passed by return can vary. The value can be hard-coded like return 4;
- It can be the result of an expression like return $a/$b;
- It can be the value returned by yet another function call like return another_function($an_argument);

Default Value for Arguments

- PHP gives you a nifty feature to help build flexible functions.
- Until now, we've said that some functions require one or more arguments.
- By making some arguments optional, you can render your functions a little less autocratic.

Example

```
<?php
          function default_args($a, $b=0, $c=0) {
                    echo "<br>", $a + $b + $c;
          }
          default_args(10, 20, 30);       // 60
          default_args(10, 20);           // 30
          default_args(10);               // 10
?>
```

Note: You must define any arguments with default value to the right of any arguments without default value. For Example

```
function make ($name = "anil", $age)        // incorrect
function make ($age, $name = "anil")        // correct
```

Passing Variable Reference

- When you pass arguments to functions, they are stored as copies in parameter variables.
- Any changes made to these variables in the body of the function are local to that function and are not reflected beyond it.

Example: Pass by Value

```
<?php
          function addFive($num) {
                    $num += 5;
          }
```

```
        $orignum = 10;
        addFive($orignum);
        echo $orignum;
?>
```

- We can change this behavior by creating a reference to our original variable.
- You can think of a reference as a signpost that points to a variable.
- In working with the reference, you are manipulating the value to which it points.
- You can pass an argument by reference by adding an ampersand to the argument name in the function definition.

Example: Pass by Reference

```
<?php
        function addFive(&$num) {
            $num += 5;
        }
        $orignum = 10;
        addFive($orignum);
        echo $orignum;
?>
```