Mr. Vidyashankara
Assistant Professor & Head
Department of Computer Science
JSS College of Arts, Commerce and Science (Autonomous)
Ooty Road, Mysuru – 570025.

PHP Building Blocks - III

PHP DECISION MAKING, LOOPING STATEMENTS

ARRAYS

STRINGS

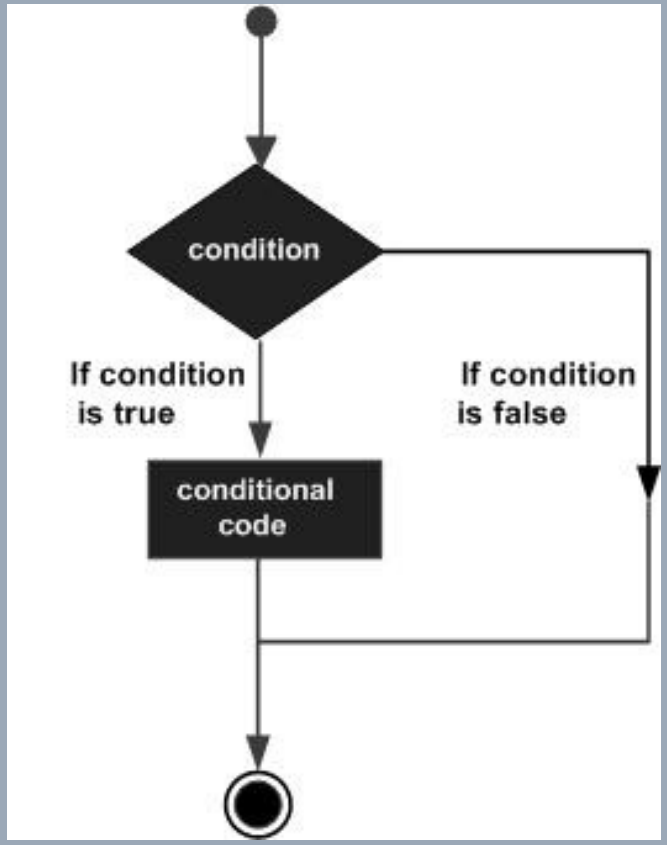
• PHP - Decision Making

The if, elseif ...else and switch statements are used to take decision based on the different condition.
You can use conditional statements in your code to make your decisions. PHP supports following three decision making statements –

if...else statement *– use this statement if you want to execute a set of code when a condition is true and another if the condition is not true*

elseif statement *– is used with the if...else statement to execute a set of code if one of the several condition is true*

switch statement *– is used if you want to select one of many blocks of code to be executed, use the Switch statement. The switch statement is used to avoid long blocks of if..elseif..else code.*



The If...Else Statement

If you want to execute some code if a condition is true and another code if a condition is false, use the if....else statement.

Syntax

```
if (condition)
     code to be executed if condition is true;
else
     code to be executed if condition is false;
```

The ElseIf Statement

If you want to execute some code if one of the several conditions are true use the elseif statement

Syntax

```
if (condition)
    code to be executed if condition is true;
elseif (condition)
    code to be executed if condition is true;
else
    code to be executed if condition is false;
```

The Switch Statement

If you want to select one of many blocks of code to be executed, use the Switch statement.
The switch statement is used to avoid long blocks of if..elseif..else code.

Syntax

```
switch (expression)
{
 case label1: code to be executed if expression =
label1;
 break;
 case label2: code to be executed if expression =
label2;
 break;
 default: code to be executed if expression is
different from both    label1 and label2;
}
```

PHP - Loop Types

Loops in PHP are used to execute the same block of code a specified number of times. PHP supports following four loop types.

for *– loops through a block of code a specified number of times.*

while *– loops through a block of code if and as long as a specified condition is true.*

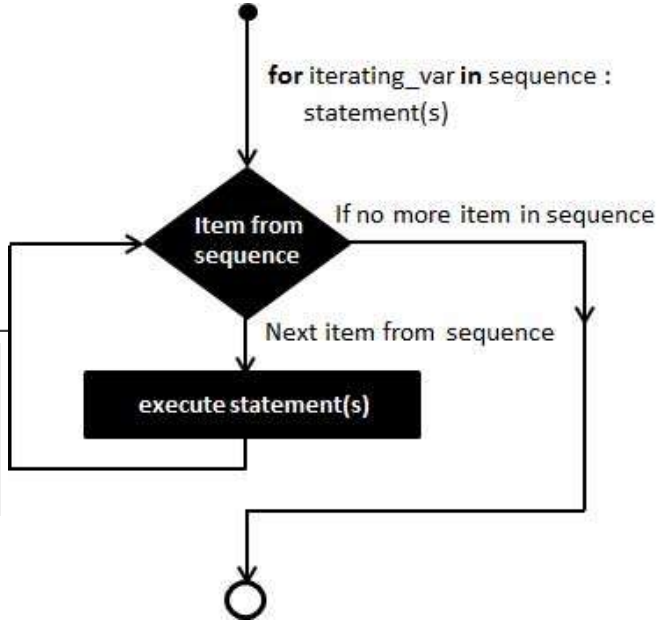
do...while *– loops through a block of code once, and then repeats the loop as long as a special condition is true.*

foreach *– loops through a block of code for each element in an array.*

We will discuss about continue and break keywords used to control the loops execution.

The for loop statement

The for statement is used when you know how many times you want to execute a statement or a block of statements.
The initializer is used to set the start value for the counter of the number of loop iterations. A variable may be declared here for this purpose and it is traditional to name it $i.



Syntax

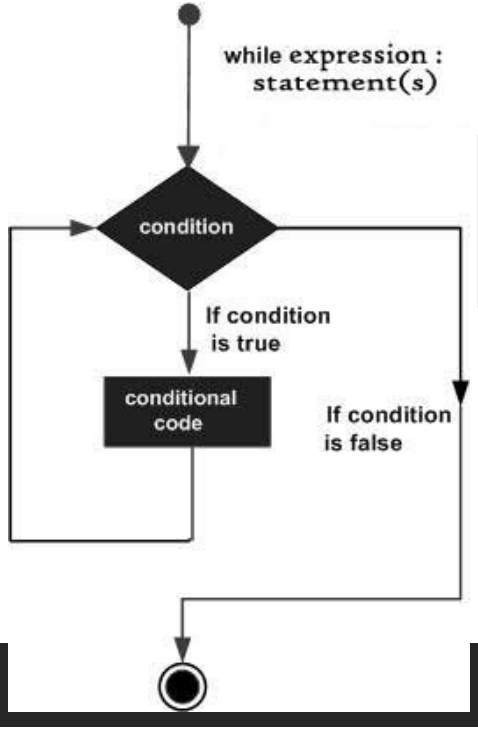
```
for (initialization; condition; increment)
{
 code to be executed;
}
```

The while loop statement

The while statement will execute a block of code if and as long as a test expression is true.
If the test expression is true then the code block will be executed. After the code has executed the test expression will again be evaluated and the loop will continue until the test expression is found to be false.

Syntax

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



The do…while loop statement

The do...while statement will execute a block of code at least once - it then will repeat the loop as long as a condition is true.

Syntax

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

The foreach loop statement

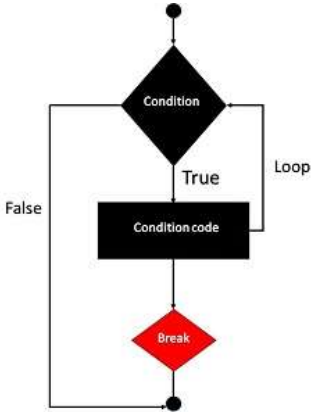
The foreach statement is used to loop through arrays. For each pass the value of the current array element is assigned to $value and the array pointer is moved by one and in the next pass next element will be processed..

Syntax

```
foreach (array as value)
{
  code to be executed;
}
```

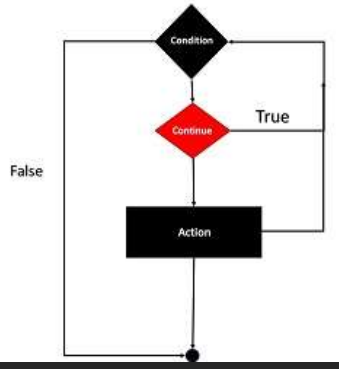
The break statement

The PHP **break** keyword is used to terminate the execution of a loop prematurely.
The **break** statement is situated inside the statement block. It gives you full control and whenever you want to exit from the loop you can come out. After coming out of a loop immediate statement to the loop will be executed.



The continue statement

The PHP **continue** keyword is used to halt the current iteration of a loop but it does not terminate the loop.
Just like the **break** statement the **continue** statement is situated inside the statement block containing the code that the loop executes, preceded by a conditional test. For the pass encountering **continue** statement, rest of the loop code is skipped and next pass starts.



Arrays

An array is a data structure that stores one or more similar type of values in a single Variable Name. For example if you want to store 100 numbers then instead of defining 100 variables its easy to define an array of 100 length.

There are three different kind of arrays and each array value is accessed using an ID c which is called array index.

•**Numeric array** − An array with a numeric index. Values are stored and accessed in linear fashion.

•**Associative array** − An array with strings as index. This stores element values in association with key values rather than in a strict linear index order.

•**Multidimensional array** − An array containing one or more arrays and values are accessed using multiple indices

NOTE − Built-in array functions is given in function reference PHP Array Functions

Numeric Array

These arrays can store numbers, strings and any object but their index will be represented by numbers. By default array index starts from zero.

Example

Following is the example showing how to create and access numeric arrays.

Here we have used **array()** function to create array. This function is explained in function reference.

```
<html>
      <body>
            <?php
            /* First method to create array. */
            $numbers = array( 1, 2, 3, 4, 5);
            foreach( $numbers as $value )
            {
              echo "Value is $value <br />";
            } /* Second method to create array. */
            $numbers[0] = "one";
            $numbers[1] = "two";
            $numbers[2] = "three";
            $numbers[3] = "four";
            $numbers[4] = "five";
            foreach( $numbers as $value )
            {
              echo "Value is $value <br />";
            }
            ?>
       </body>
</html>
```

Associative Arrays

The associative arrays are very similar to numeric arrays in term of functionality but they are different in terms of their index. Associative array will have their index as string so that you can establish a strong association between key and values.

To store the salaries of employees in an array, a numerically indexed array would not be the best choice. Instead, we could use the employees names as the keys in our associative array, and the value would be their respective salary.

NOTE − Don't keep associative array inside double quote while printing otherwise it would not return any value.

```
<html>
      <body>
           <?php /* First method to associate create array. */
                $salaries = array("mohammad" => 2000, "qadir" => 1000, "zara" => 500);
                echo "Salary of mohammad is ". $salaries['mohammad'] . "<br />";
                echo "Salary of qadir is ". $salaries['qadir']. "<br />";
                echo "Salary of zara is ". $salaries['zara']. "<br />";
                /* Second method to create array. */
                $salaries['mohammad'] = "high";
                $salaries['qadir'] = "medium";
                $salaries['zara'] = "low";
                echo "Salary of mohammad is ". $salaries['mohammad'] . "<br />";
                echo "Salary of qadir is ". $salaries['qadir']. "<br />";
                 echo "Salary of zara is ". $salaries['zara']. "<br />";
            ?>
      </body>
</html>
```

Multidimensional Arrays

A multi-dimensional array each element in the main array can also be an array. And each element in the sub-array can be an array, and so on. Values in the multi-dimensional array are accessed using multiple index.

Example

In this example we create a two dimensional array to store marks of three students in three subjects −

This example is an associative array, you can create numeric array in the same fashion.

```
<html>
      <body>
            <?php
                  $marks = array( "mohammad" => array ("physics" => 35,"maths" => 30, "chemistry" => 39 ),
                                  "qadir" => array ( "physics" => 30, "maths" => 32, "chemistry" => 29 ),
                                  "zara" => array ( "physics" => 31, "maths" => 22, "chemistry" => 39 ) );
                    /* Accessing multi-dimensional array values */
                  echo "Marks for mohammad in physics : " ;
                  echo $marks['mohammad']['physics'] . "<br />";
                  echo "Marks for qadir in maths : ";
                  echo $marks['qadir']['maths'] . "<br />";
                  echo "Marks for zara in chemistry : " ;
                  echo $marks['zara']['chemistry'] . "<br />";
            ?>
       </body>
 </html>
```

Strings

They are sequences of characters, like "PHP supports string operations“
Singly quoted strings are treated almost literally, whereas doubly quoted strings replace variables with their values as well as specially interpreting certain character sequences.

Following are valid examples of string

```
$string_1 = "This is a string in double quotes";
$string_2 = "This is a somewhat longer, singly quoted string";
$string_39 = "This string has thirty-nine characters
$string_0 = ""; // a string with zero characters
```

```
<?php
      $variable = "name";
      $literally = 'My $variable will not print!\\n';
      print($literally); print "<br />";
      $literally = "My $variable will print!\\n";
      print($literally);
?>
```

Output:

```
My $variable will not print!\n
My name will print!\n
```

NOTE − Built-in string functions is given in function reference PHP String Functions

Strings

There are no artificial limits on string length - within the bounds of available memory, you ought to be able to make arbitrarily long strings.

Strings that are delimited by double quotes (as in "this") are preprocessed in both the following two ways by PHP −

•Certain character sequences beginning with backslash (\) are replaced with special characters
•Variable names (starting with $) are replaced with string representations of their values.

The escape-sequence replacements are −

•**\n** is replaced by the newline character
•**\r** is replaced by the carriage-return character
•**\t** is replaced by the tab character
•**\$** is replaced by the dollar sign itself ($)
•**\"** is replaced by a single double-quote (")
•**** is replaced by a single backslash (\)

String Concatenation Operator

To concatenate two string variables together, use the dot (.) operator −

```php
<?php
     $string1="Hello World";
     $string2="1234";
     echo $string1 . " " . $string2;
?>
```

Output:

```
Hello World 1234
```

Strings

There are no artificial limits on string length - within the bounds of available memory, you ought to be able to make arbitrarily long strings.
Strings that are delimited by double quotes (as in "this") are preprocessed in both the following two ways by PHP −

•Certain character sequences beginning with backslash (\) are replaced with special characters
•Variable names (starting with $) are replaced with string representations of their values.
The escape-sequence replacements are −
•**\n** is replaced by the newline character
•**\r** is replaced by the carriage-return character
•**\t** is replaced by the tab character
•**\$** is replaced by the dollar sign itself ($)
•**\"** is replaced by a single double-quote (")
•\\ is replaced by a single backslash (\)

String Concatenation Operator

To concatenate two string variables together, use the dot (.) operator −
If we look at the code above you see that we used the concatenation operator two times. This is because we had to insert a third string. Between the two string variables we added a string with a single character, an empty space, to separate the two variables.

```
<?php
     $string1="Hello World";
     $string2="1234";
     echo $string1 . " " . $string2;
?>
```

Output:

```
Hello World 1234
```

```
<?php?>
```

Strings Function

Using the strlen() function

The strlen() function is used to find the length of a string.
Let's find the length of our string "Hello world!"

The length of a string is often used in loops or other functions, when it is important to know when the string ends. (i.e. in a loop, we would want to stop the loop after the last character in the string)

```
<?php
     echo strlen("Hello world!");
?>
Output:
12
```

Using the strpos() function

The strpos() function is used to search for a string or character within a string.
If a match is found in the string, this function will return the position of the first match. If no match is found, it will return FALSE.

Let's see if we can find the string "world" in our string As you see the position of the string "world" in our string is position 6. The reason that it is 6, and not 7, is that the first position in the string is 0, and not 1.

```
<?php
     echo strpos("Hello world!","world");
?>
Output:
6
```

PHP Array Functions - I

array() Create an array
array_change_key_case() Returns an array with all keys in lowercase or uppercase
array_chunk() Splits an array into chunks of arrays
array_column() Return the values from a single column in the input array
array_combine() Creates an array by using one array for keys and another for its values
array_count_values() Returns an array with the number of occurrences for each value
array_diff() Compares array values, and returns the differences
array_diff_assoc() Compares array keys and values, and returns the differences
array_diff_key() Compares array keys, and returns the differences
array_diff_uassoc() Compares array keys and values, with an additional user-made function check, and returns the differences
array_diff_ukey() Compares array keys, with an additional user-made function check, and returns the differences
array_fill() Fills an array with values
array_fill_keys() Fill an array with values, specifying keys
array_filter() Filters elements of an array using a user-made function
array_flip() Exchanges all keys with their associated values in an array
array_intersect() Compares array values, and returns the matches
array_intersect_assoc() Compares array keys and values, and returns the matches
array_intersect_key() Compares array keys, and returns the matches
array_intersect_uassoc() Compares array keys and values, with an additional user-made function check, and returns the matches
array_intersect_ukey() Compares array keys, with an additional user-made function check, and returns the matches
array_key_exists() Checks if the specified key exists in the array
array_keys() Returns all the keys of an array
array_map() Sends each value of an array to a user-made function, which returns new values
array_merge() Merges one or more arrays into one array
array_merge_recursive() Merges one or more arrays into one array
array_multisort() Sorts multiple or multi-dimensional arrays

PHP Array Functions - II

array_pad() Inserts a specified number of items, with a specified value, to an array
array_pop() Deletes the last element of an array
array_product() Calculates the product of the values in an array
array_push() Inserts one or more elements to the end of an array
array_rand() Returns one or more random keys from an array
array_reduce() Returns an array as a string, using a user-defined function
array_reverse() Returns an array in the reverse order
array_search() Searches an array for a given value and returns the key
array_shift() Removes the first element from an array, and returns the value of the removed element
array_slice() Returns selected parts of an array
array_splice() Removes and replaces specified elements of an array
array_sum() Returns the sum of the values in an array
array_udiff() Compares array values in a user-made function and returns an array
array_udiff_assoc() Compares array keys, and compares array values in a user-made function, and returns an array
array_udiff_uassoc() Compares array keys and array values in user-made functions, and returns an array
array_uintersect() Compares array values in a user-made function and returns an array
array_uintersect_assoc() Compares array keys, and compares array values in a user-made function, and returns an array
array_uintersect_uassoc() Compares array keys and array values in user-made functions, and returns an array
array_unique() Removes duplicate values from an array
array_unshift() Adds one or more elements to the beginning of an array
array_values() Returns all the values of an array
array_walk() Applies a user function to every member of an array
array_walk_recursive() Applies a user function recursively to every member of an array
arsort() Sorts an array in reverse order and maintain index association
asort() Sorts an array and maintain index association

PHP Array Functions - III

compact() Create array containing variables and their values
count() Counts elements in an array, or properties in an object
current() Returns the current element in an array
each() Returns the current key and value pair from an array
end() Sets the internal pointer of an array to its last element
extract() Imports variables into the current symbol table from an array
in_array() Checks if a specified value exists in an array
key() Fetches a key from an array
krsort() Sorts an array by key in reverse order
ksort() Sorts an array by key
list() Assigns variables as if they were an array
natcasesort() Sorts an array using a case insensitive "natural order" algorithm
natsort() Sorts an array using a "natural order" algorithm
next() Advance the internal array pointer of an array
pos() Alias of current()
prev() Rewinds the internal array pointer
range() Creates an array containing a range of elements
reset() Sets the internal pointer of an array to its first element
rsort() Sorts an array in reverse order
shuffle() Shuffles an array
sizeof() Alias of count()
sort() Sorts an array
uasort() Sorts an array with a user-defined function and maintain index association
uksort() Sorts an array by keys using a user-defined function
usort() Sorts an array by values using a user-defined function

PHP Array Constants -

CASE_LOWER Used with array_change_key_case() to convert array keys to lower case
CASE_UPPER Used with array_change_key_case() to convert array keys to upper case
SORT_ASC Used with array_multisort() to sort in ascending order
SORT_DESC Used with array_multisort() to sort in descending order
SORT_REGULAR Used to compare items normally
SORT_NUMERIC Used to compare items numerically
SORT_STRING Used to compare items as strings
SORT_LOCALE_STRING Used to compare items as strings, based on the current locale
COUNT_NORMAL
COUNT_RECURSIVE
EXTR_OVERWRITE
EXTR_SKIP
EXTR_PREFIX_SAME
EXTR_PREFIX_ALL
EXTR_PREFIX_INVALID
EXTR_PREFIX_IF_EXISTS
EXTR_IF_EXISTS
EXTR_REFS

PHP String Functions - I

addcslashes - It returns the string with blackslashes
addslashes - It returns the string with blackslashes in front of predefined characters
bin2hex - It is used to convert primary data to hexadecimal representation
chop - It is used to removes whitespace
chr - It returns the specific characters
chunk split - It is used to split a string into chunks.
convert cyr string - It is used to convert from one Cyrillic character set to another
convert uudecode - It is used to decode a encoded string
count chars - It is used to returns the information about character used in a string
Crc32 - It is used to calculates 32-bit CRC
crypt - It is used to hashing the string
Echo - It is give the output as one or more string
explode - It is used to split a string by string
fprintf - It is used to write a formatted string to a stream
get html translation table - It returns the translation table used by htmlspecialchars() and htmlentities()
hebrev - It is used to convert logical Hebrew text to visual text
Hebrevc - It is used to convert logical Hebrew text to visual text with newline conversion
hex2bin - It is used to convert a string of hexadecimal to ASCII character
html entity decode - It is used to convert HTML entities to their application characters
htmlentities - It is used to convert all applicable characters to Html entities
html special chars decode - It is used to convert convert special HTML entities back to characters.
htmlspecialchars - It is used to convert special characters to HTML entities
implode - It is used to Join array elements with a string.
join - It is alias of implode(), it returns string from the elements of an array
lcfirst - It is used to make a string's first character should be lowercase.

PHP String Functions - II

levenshtein - It is used calculate Levenshtein distance between two strings
localeconv - It is used to get numeric formatting information
ltrim - It used to strip whitespace or other characters from the beginning of a string
md5_file - It is used to calculates the md5 hash of a given file
md5 - It is used to calculates the md5 hash of a string
metaphone - It is used to calculates the metaphone key of a string
money format - It is used to formats a number as a currency string
nl langinfo - It has contained information about language and locale
nl2br - It Inserts HTML line breaks before all newlines in a string
number format - It is used to formats a number with grouped thousands
ord - It returns ASCII Value of character
parse str - It is used to parses the string into variables
print - It returns output a string
printf - It returns output a formatted string
quoted printable decode - It is used to convert the quoted printable string to 8 bit string
quoted printable decode - It is used to convert the quoted printable string to 8 bit string
quoted printable encode - It is used to convert 8 bit string to the quoted printable string
quotemeta - It is used to quote meta characters
rtrim - It is used to remove the white spaces from end of the string
setlocale - It is used to set locale information
sha1 file - It is used to calculate the sha1 hash of a file
sha1 - It is used to calculate the sha1 hash of a string
similar text - It is used to calculate the similarity between two strings
soundex - It is used to calculate the soundex key of string
sprintf - It is used to a formatted string

PHP String Functions - III

str getcsv - It is used to parse parse a CSV string into array
str ireplace - It is used to replace the characters with some other characters
str pad - It is used to pads a string to a new length.
str repeat - It is used to repeat a string
str replace - It is used to replace the string with another string
str rot13 - It is used to perform the rot13 transform on a string
str shuffle - It is used to randomly shuffles a string
str split - It is used to convert a string to an array
str word count - It returns information about words used in a string
strcasecmp - It is used to compare two strings
strchr - It is used to searches for the first occurrence of a string inside another
strcmp - It is used to compare two strings
strcoll - It is used to compare two strings based on locale
strcspn - It returns number of characters find in a string before any part of the specified characters are found.
strip tags - It is used to string HTML and PHP tags from a string.
stripcslashes - It is used to removes backslashes
stripos - It is used to find the position of first occurrence of a string inside the another string
stripslashes - It is used to un-quoted a quoted string
stristr - It is used searches for the first occurrence of a string inside another string.
strlen - It is used get string length.
strnatcasecmp - It is used compare two strings with a natural algorithm.
strnatcmp - It is used compare two strings with a natural order algorithm.
strncasecmp - It is used compare two strings.
strncmp - It is used compare first n character.
strpbrk - It is used search's a string for a specific character.

PHP String Functions - IV

strpos - It is used to find the position of first occurrence of a string inside another string.
strrchr - It is used to find the last occurrence of a character in a string.
strrev - It is used to reverse a string.
strripos - It is used find the position of the last occurrence of a string inside the another string
strspn - It returns number of characters found in the string from the charlist parameter.
strstr - It is used to find the first occurrence of a string
strtok - It is a tokenize string
strtolower - It used to make a string lower case
strtoupper - It used to make a string upper case
strtr - It used to translates character or replace substring
substr_compare - It used to compare two string format with a specific start position
substr_count - It used to count the number of sub strings
substr_replace - It used to replace the part of string with another string
substr - It used to return a part of a string
trim - It used to remove the whitespaces and other characters
ucfirst - It used to convert the first character of a string to upper case
ucwords - It used to convert the first character of a string to upper case in each string
vfprintf - It used to convert formatted string to specific output
vprintf - It used to convert string to formatted string
vsprintf - It returns the formatted string
wordwrap - It used to convert the long words to be broken and its arrange to the next line

Mr. Vidyashankara
Assistant Professor & Head
Department of Computer Science
JSS College of Arts, Commerce and Science
(Autonomous)
Ooty Road, Mysuru – 570025.