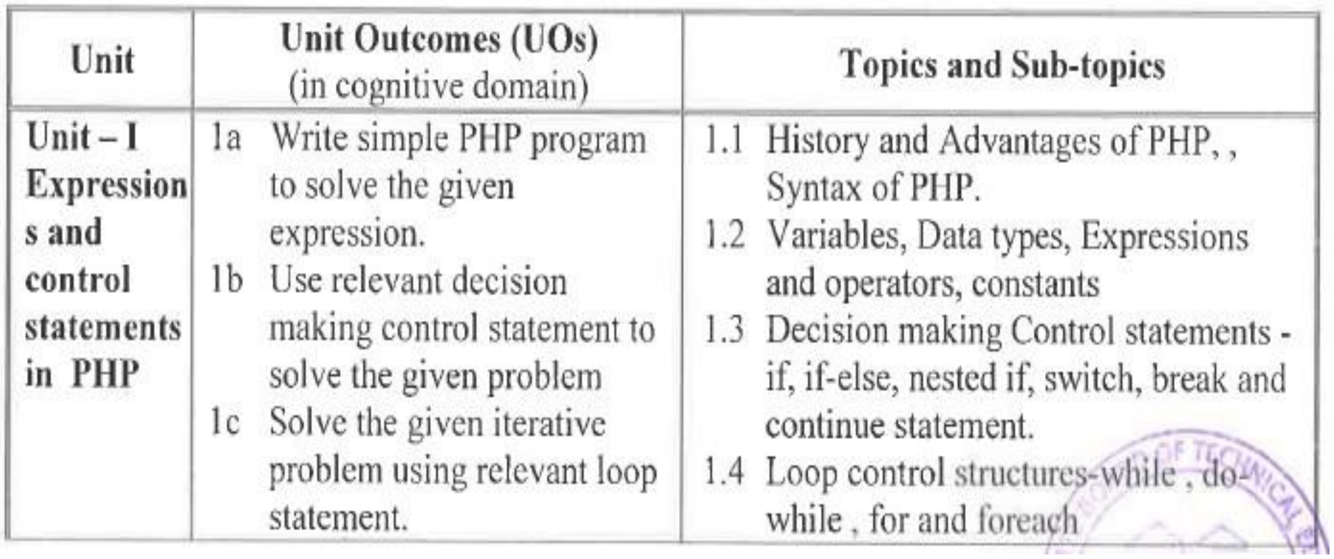


Unit	Unit Outcomes (UOs) (in cognitive domain)	Topics and Sub-topics
Unit – I Expressions and control statements in PHP	1a Write simple PHP program to solve the given expression. 1b Use relevant decision making control statement to solve the given problem 1c Solve the given iterative problem using relevant loop statement.	1.1 History and Advantages of PHP, , Syntax of PHP. 1.2 Variables, Data types, Expressions and operators, constants 1.3 Decision making Control statements - if, if-else, nested if, switch, break and continue statement. 1.4 Loop control structures-while , do-while , for and foreach

1.1 History and Advantages of PHP, Syntax of PHP.

PHP started out as a small open source project that evolved as more and more people found out how useful it was. Rasmus Lerdorf unleashed the first version of PHP way back in 1994.

- PHP is a recursive acronym for "PHP: Hypertext Preprocessor".
- PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
- It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
- PHP is pleasingly zippy in its execution, especially when compiled as an Apache module on the Unix side. The MySQL server, once started, executes even very complex queries with huge result sets in record-setting time.
- PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.
- PHP is forgiving: PHP language tries to be as forgiving as possible.
- PHP Syntax is C-Like.

Advantages of PHP :

1. Most important advantage of PHP is that it’s open source and freed from cost. It is often downloaded anywhere and readily available to use for web applications.
2. It is platform independent. PHP based applications can run on any OS like UNIX, Linux and windows, etc.
3. Applications can easily be loaded which are based on PHP and connected to a database. it’s mainly used due to its faster rate of loading over slow internet and speed than other programming languages.
4. It has less learning curve, because it is straightforward and straightforward to use. If a private knows C programming can easily work on PHP.

5. It is more stable from a few years with assistance of providing continuous support to various versions.
6. It helps in reusing an equivalent code and no got to write lengthy code and sophisticated structure for the event of web applications.
7. It helps in managing code easily.
8. It has powerful library support to use various function modules for data representation.
9. PHP's built-in database connection modules help in connecting databases easily and reduce trouble and time for development of web applications and content based sites.
10. Popularity of PHP gave rise to various communities of developers, a fraction of which may be potential candidates for hire.

Basic PHP Syntax

A PHP script can be placed anywhere in the document. A PHP script starts with <?php and ends with ?>:

```
<?php

// PHP code goes here

?>
```

The default file extension for PHP files is ".php".A PHP file normally contains HTML tags, and some PHP scripting code.

1.2 Variable, Data type, expression and operators, Constants

1. Variable: -In PHP, a variable is declared using a $ sign followed by the variable name. Here, some important points to know about variables:

- As PHP is a loosely typed language, so we do not need to declare the data types of the variables. It automatically analyzes the values and makes conversions to its correct data type.
- After declaring a variable, it can be reused throughout the code.
- Assignment Operator (=) is used to assign the value to a variable.

Syntax of declaring a variable in PHP is given below:

1. $variablename=value;

Rules for declaring PHP variable:

- A variable must start with a dollar ($) sign, followed by the variable name.
- It can only contain alpha-numeric characters and underscore (A-z, 0-9, _).
- A variable name must start with a letter or underscore (_) character.
- A PHP variable name cannot contain spaces.
- One thing to be kept in mind that the variable name cannot start with a number or special symbols.
- PHP variables are case-sensitive, so $name and $NAME both are treated as different variables.

PHP Variable: Declaring string, integer, and float

Let's see the example to store string, integer, and float values in PHP variables.

File: variable1.php

```php
<?php
$str="hello string";
$x=200;
$y=44.6;
echo "string is: $str <br/>";
echo "integer is: $x <br/>";
echo "float is: $y <br/>";
?>
```

Output:

```
string is: hello string


integer is: 200


float is: 44.6
```

PHP Data Types

PHP data types are used to hold different types of data or values. PHP supports 8 primitive data types that can be categorized further in 3 types:

1. Scalar Types (predefined)
2. Compound Types (user-defined)

3. Special Types

PHP Data Types: Scalar Types

It holds only a single value. There are 4 scalar data types in PHP.

1. boolean
2. integer
3. float
4. string

PHP Data Types: Compound Types

It can hold multiple values. There are 2 compound data types in PHP.

1. array
2. object

PHP Data Types: Special Types

There are 2 special data types in PHP.

1. resource
2. NULL

PHP Boolean

Booleans are the simplest data type works like switch. It holds only two values: **TRUE (1)** or **FALSE (0)**. It is often used with conditional statements. If the condition is correct, it returns TRUE otherwise FALSE.

Example:

```
<?php
  if (TRUE)
      echo "This condition is TRUE.";
  if (FALSE)
```

```
echo "This condition is FALSE.";
?>
```

Output:

```
This condition is TRUE.
```

PHP Integer

Integer means numeric data with a negative or positive sign. It holds only whole numbers, i.e., numbers without fractional part or decimal points.

Rules for integer:

- An integer can be either positive or negative.
- An integer must not contain decimal point.
- Integer can be decimal (base 10), octal (base 8), or hexadecimal (base 16).
- The range of an integer must be lie between 2,147,483,648 and 2,147,483,647 i.e., -2^31 to 2^31.

Example:

```
<?php
  $dec1 = 34;
  $oct1 = 0243;
  $hexa1 = 0x45;
  echo "Decimal number: " .$dec1. "</br>";
  echo "Octal number: " .$oct1. "</br>";
echo "HexaDecimal number: " .$hexa1. "</br>";
?>
```

Output:

```
Decimal number: 34

Octal number: 163

Hexadecimal number: 69
```

PHP Float

A floating-point number is a number with a decimal point. Unlike integer, it can hold numbers with a fractional or decimal point, including a negative or positive sign.

Example:

```
<?php
    $n1 = 19.34;
    $n2 = 54.472;
    $sum = $n1 + $n2;
    echo "Addition of floating numbers: " .$sum;
?>
```

Output:

```
Addition of floating numbers: 73.812
```

PHP String

A string is a non-numeric data type. It holds letters or any alphabets, numbers, and even special characters.

String values must be enclosed either within **single quotes** or in **double quotes**. But both are treated differently. To clarify this, see the example below:

Example:

```
<?php
    $company = "Javatpoint";
    //both single and double quote statements will treat different
    echo "Hello $company";
    echo "</br>";
    echo 'Hello $company';
```

```
?>
```

Output:

```
Hello Javatpoint

Hello $company
```

PHP Array

An array is a compound data type. It can store multiple values of same data type in a single variable.

Example:

```
<?php
  $bikes = array ("Royal Enfield", "Yamaha", "KTM");
  var_dump($bikes);  //the var_dump() function returns the datatype and values
  echo "</br>";
  echo "Array Element1: $bikes[0] </br>";
  echo "Array Element2: $bikes[1] </br>";
echo "Array Element3: $bikes[2] </br>";
?>
```

Output:

```
array(3) { [0]=> string(13) "Royal Enfield" [1]=> string(6) "Yamaha" [2]=> string(3) "KTM" }

Array Element1: Royal Enfield

Array Element2: Yamaha

Array Element3: KTM
```

You will learn more about array in later chapters of this tutorial.

PHP object

Objects are the instances of user-defined classes that can store both values and functions. They must be explicitly declared.

Example:

```
<?php
    class bike {
        function model() {
            $model_name = "Royal Enfield";
            echo "Bike Model: " .$model_name;
          }
    }
    $obj = new bike();
    $obj -> model();
?>
```

Output:

```
Bike Model: Royal Enfield
```

This is an advanced topic of PHP, which we will discuss later in detail.

PHP Resource

Resources are not the exact data type in PHP. Basically, these are used to store some function calls or references to external PHP resources. **For example** - a database call. It is an external resource.

This is an advanced topic of PHP, so we will discuss it later in detail with examples.

PHP Null

Null is a special data type that has only one value: **NULL**. There is a convention of writing it in capital letters as it is case sensitive.

The special type of data type NULL defined a variable with no value.

Example:

1. <?php
2. $nl = NULL;
3. echo $nl; //it will not give any output4.

?>

Output:

Type of operators.

- Arithmetic Operators
- Comparison Operators
- Logical (or Relational) Operators
- Assignment Operators
- Conditional (or ternary) Operators

Lets have a look on all operators one by one.

Arithmetic Operators

There are following arithmetic operators supported by PHP language –

Assume variable A holds 10 and variable B holds 20 then –

Show Examples

Operator	Description	Example
+	Adds two operands	A + B will give 30
-	Subtracts second operand from the first	A - B will give -10
*	Multiply both operands	A * B will give 200
/	Divide numerator by de-numerator	B / A will give 2

%	Modulus Operator and remainder of after an integer division	B % A will give 0
++	Increment operator, increases integer value by one	A++ will give 11
--	Decrement operator, decreases integer value by one	A-- will give 9

Comparison Operators

There are following comparison operators supported by PHP language

Assume variable A holds 10 and variable B holds 20 then –

Show Examples

Operator	Description	Example
==	Checks if the value of two operands are equal or not, if yes then condition becomes true.	(A == B) is not true.
!=	Checks if the value of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

Logical Operators

There are following logical operators supported by PHP language

Assume variable A holds 10 and variable B holds 20 then –

Show Examples

Operator	Description	Example
and	Called Logical AND operator. If both the operands are true then condition becomes true.	(A and B) is true.
or	Called Logical OR Operator. If any of the two operands are non zero then condition becomes true.	(A or B) is true.
&&	Called Logical AND operator. If both the operands are non zero then condition becomes true.	(A && B) is true.
||	Called Logical OR Operator. If any of the two operands are non zero then condition becomes true.	(A || B) is true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is false.

Assignment Operators

There are following assignment operators supported by PHP language –

Show Examples

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	C = A + B will assign value of A + B into C
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand	C += A is equivalent to C = C + A

-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand	C %= A is equivalent to C = C % A

Conditional Operator

There is one more operator called conditional operator. This first evaluates an expression for a true or false value and then execute one of the two given statements depending upon the result of the evaluation. The conditional operator has this syntax −

Show Examples

Operator	Description	Example
? :	Conditional Expression	If Condition is true ? Then value X : Otherwise value Y

Operators Categories

All the operators we have discussed above can be categorised into following categories −

- Unary prefix operators, which precede a single operand.
- Binary operators, which take two operands and perform a variety of arithmetic and logical operations.
- The conditional operator (a ternary operator), which takes three operands and evaluates either the second or third expression, depending on the evaluation of the first expression.
- Assignment operators, which assign a value to a variable.

Precedence of PHP Operators

Operator precedence determines the grouping of terms in an expression. This affects how an expression is evaluated. Certain operators have higher precedence than others; for example, the multiplication operator has higher precedence than the addition operator −

For example x = 7 + 3 * 2; Here x is assigned 13, not 20 because operator * has higher precedence than + so it first get multiplied with 3*2 and then adds into 7.

Here operators with the highest precedence appear at the top of the table, those with the lowest appear at the bottom. Within an expression, higher precedence operators will be evaluated first.

Category	Operator	Associativity
Unary	! ++ --	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Logical AND	&&	Left to right
Logical OR	||	Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %=	Right to left

PHP Constants

PHP constants are name or identifier that can't be changed during the execution of the script except for magic constants, which are not really constants. PHP constants can be defined by 2 ways:

1. Using define() function
2. Using const keyword

Constants are similar to the variable except once they are defined, they can never be undefined or changed. They remain constant across the entire program. PHP constants follow the same PHP variable rules. For example, it can be started with a letter or underscore only.

Conventionally, PHP constants should be defined in uppercase letters.

Note: Unlike variables, constants are automatically global throughout the script.

PHP Constants
PHP constants are names or identifiers that can't be changed during the execution of the script except for magic constants, which are not really constants. PHP constants can be defined by 2 ways:

Using define() function
Using const keyword
Constants are similar to the variable except once they are defined, they can never be undefined or changed. They remain constant across the entire program. PHP constants follow the same PHP variable rules. For example, it can be started with a letter or underscore only.

Conventionally, PHP constants should be defined in uppercase letters.

Note: Unlike variables, constants are automatically global throughout the script.
PHP constant: define()
Use the define() function to create a constant. It defines constant at runtime. Let's see the syntax of define() function in PHP.

define(name, value, case-insensitive)
name: It specifies the constant name.
value: It specifies the constant value.
case-insensitive: Specifies whether a constant is case-insensitive. Default value is false. It means it is case sensitive by default.
Let's see the example to define PHP constant using define().

File: constant1.php

```
<?php
define("MESSAGE","Hello JavaTpoint PHP");
echo MESSAGE;
?>
```
Output:

```
Hello JavaTpoint PHP
```
Create a constant with case-insensitive name:

File: constant2.php

```
<?php
define("MESSAGE","Hello JavaTpoint PHP",true);//not case sensitive
echo MESSAGE, "</br>";
echo message;
?>
```
Output:

```
Hello JavaTpoint PHP
Hello JavaTpoint PHP
```
File: constant3.php

```
<?php
define("MESSAGE","Hello JavaTpoint PHP",false);//case sensitive
echo MESSAGE;
echo message;
?>
```
Output:

```
Hello JavaTpoint PHP
Notice: Use of undefined constant message - assumed 'message'
in C:\wamp\www\vconstant3.php on line 4
message
```
PHP constant: const keyword

PHP introduced a keyword const to create a constant. The const keyword defines constants at compile time. It is a language construct, not a function. The constant defined using const keyword are case-sensitive.

File: constant4.php

```
<?php
const MESSAGE="Hello const by JavaTpoint PHP";
echo MESSAGE;
?>
```
Output:

Hello const by JavaTpoint PHP
Constant() function
There is another way to print the value of constants using constant() function instead of using the echo statement.

Syntax

The syntax for the following constant function:

```
constant (name)
```

File: constant5.php

```
<?php
   define("MSG", "JavaTpoint");
   echo MSG, "</br>";
   echo constant("MSG");
   //both are similar
?>
```

Output:

```
JavaTpoint
JavaTpoint
```

1.3 PHP provides us with four conditional statements:

- if statement
- if…else statement
- if…elseif…else statement
- switch statement

Let us now look at each one of these in details:

if Statement: This statement allows us to set a condition. On being TRUE, the following block of code enclosed within the if clause will be executed.
Syntax :

```
if (condition){
   // if TRUE then execute this code
}
```

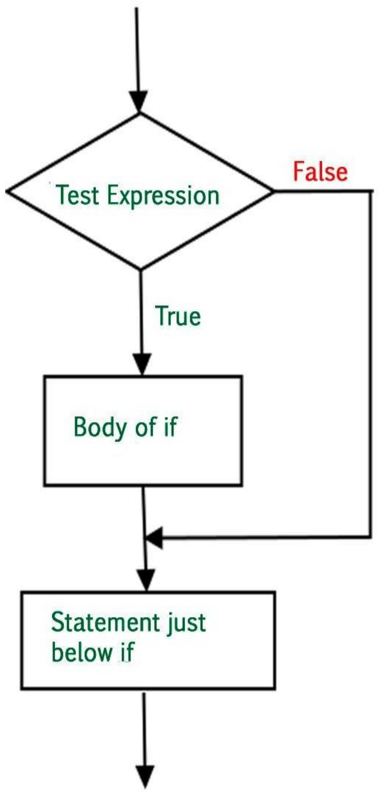
1. Example:

```
<?php
$x = 12;

if ($x > 0) {
   echo "The number is positive";
}
?>
```

Output:

```
The number is positive
```

2. Flowchart:



if…else Statement: We understood that if a condition will hold i.e., TRUE, then the block of code within if will be executed. But what if the condition is not TRUE and we want to perform an action? This is where else comes into play. If a condition is TRUE then if block gets executed, otherwise else block gets executed.

Syntax:

```
if (condition) {
   // if TRUE then execute this code
}
else{
   // if FALSE then execute this code
}
```

3.

Example:

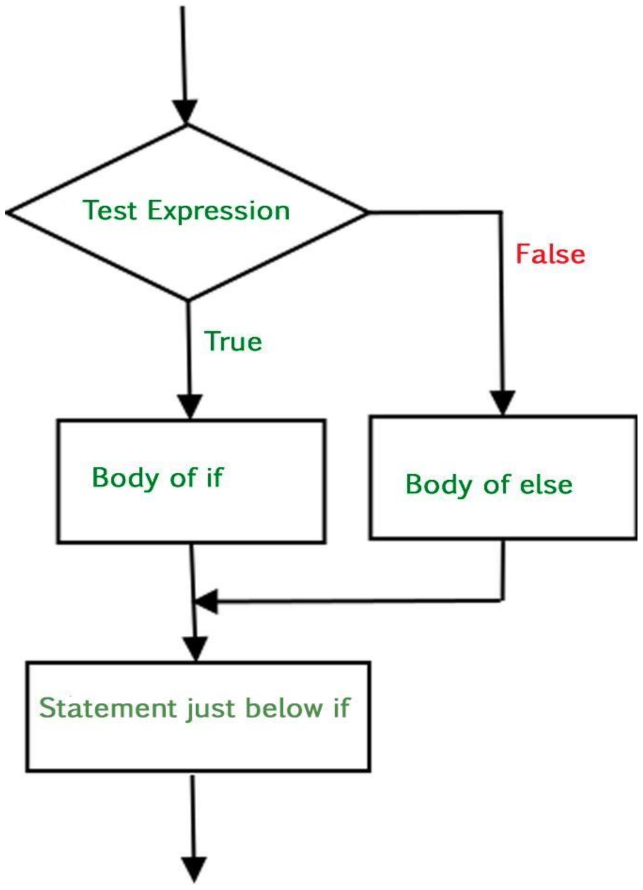
```
<?php
$x = -12;

if ($x > 0) {
   echo "The number is positive";
}

else{
   echo "The number is negative";
}
?>
```

Output:

```
The number is negative
```

4.

Flowchart:



if…elseif…else Statement: This allows us to use multiple if…else statments. We use this when there are multiple conditions of TRUE cases.

Syntax:

```
if (condition) {
   // if TRUE then execute this code
}
elseif {
   // if TRUE then execute this code
}
elseif {
   // if TRUE then execute this code
}
else {
   // if FALSE then execute this code
}
```

5.

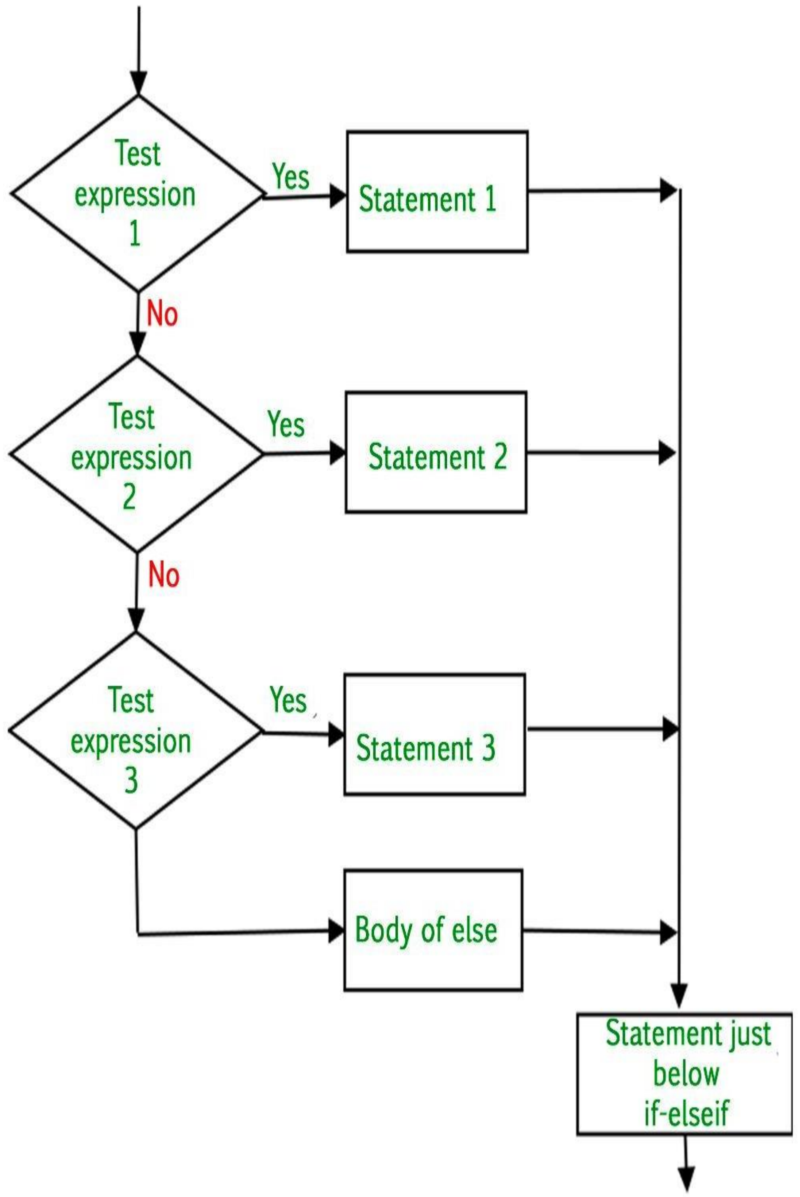
Example:

```
<?php
$x = "August";

if ($x == "January") {
    echo "Happy Republic Day";
}

elseif ($x == "August") {
    echo "Happy Independence Day!!!";
}

else{
    echo "Nothing to show";
}
?>
```

Output:

```
Happy Independence Day!!!
```

6.
Flowchart:



7. switch Statement: The “switch” performs in various cases i.e., it has various cases to which it matches the condition and appropriately executes a particular case block. It first evaluates an expression and then compares with the values of each case. If a case matches then the same case is executed. To use switch, we need to get familiar with two different keywords namely, break and default.
    1. The break statement is used to stop the automatic control flow into the next cases and exit from the switch case.
    2. The default statement contains the code that would execute if none of the cases match.

Syntax:

```
switch(n) {
   case statement1:
      code to be executed if n==statement1;
      break;
   case statement2:
      code to be executed if n==statement2;
      break;
   case statement3:
      code to be executed if n==statement3;
      break;
   case statement4:
      code to be executed if n==statement4;
      break;
   ......
   default:
      code to be executed if n != any case;
```

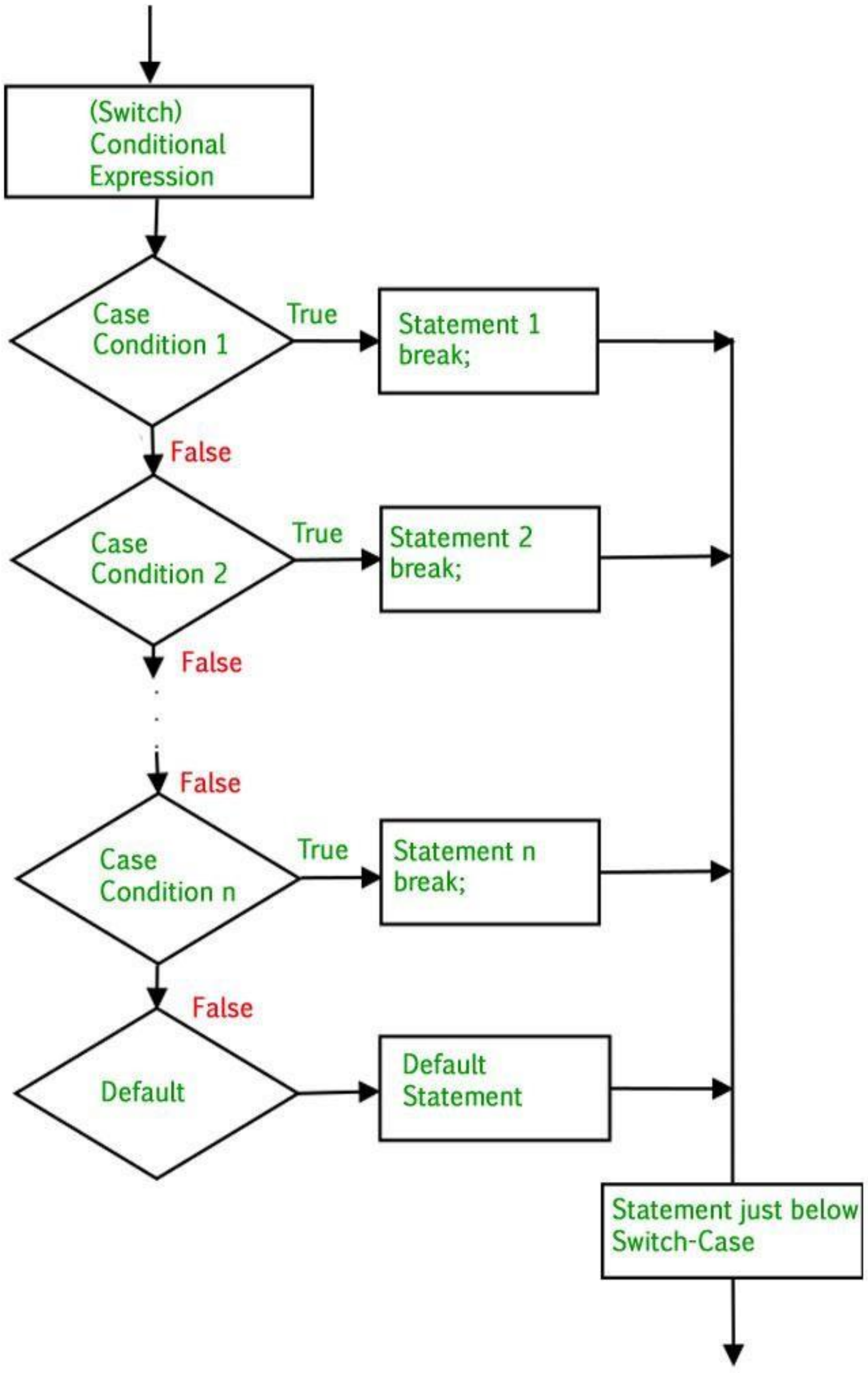
8.

Example:

```php
<?php
$n = "February";

switch($n) {
   case "January":
      echo "Its January";
      break;
   case "February":
      echo "Its February";
      break;
   case "March":
      echo "Its March";
      break;
   case "April":
      echo "Its April";
      break;
   case "May":
      echo "Its May";
      break;
   case "June":
      echo "Its June";
      break;
   case "July":
      echo "Its July";
      break;
   case "August":
      echo "Its August";
      break;
   case "September":
      echo "Its September";
      break;
   case "October":
      echo "Its October";
      break;
   case "November":
      echo "Its November";
      break;
   case "December":
      echo "Its December";
      break;
   default:
      echo "Doesn't exist";
}
?>
```

Output:
Its February

9. Flowchart:



Ternary Operators

In addition to all this conditional statements, PHP provides a shorthand way of writing if…else, called Ternary Operators. The statement uses a question mark (?) and a colon (:) and takes three operands: a condition to check, a result for TRUE and a result for FALSE.

Syntax:

(condition) ? if TRUE execute this : otherwise execute this;

Example:

```
<?php
$x = -12;

if ($x > 0) {
   echo "The number is positive \n";
}
else {
   echo "The number is negative \n";
}

// This whole lot can be written in a
// single line using ternary operator
echo ($x > 0) ? 'The number is positive' :
               'The number is negative';
?>
```

Output:

```
The number is negative
The number is negative
```

1.4 Loop Structure in Php

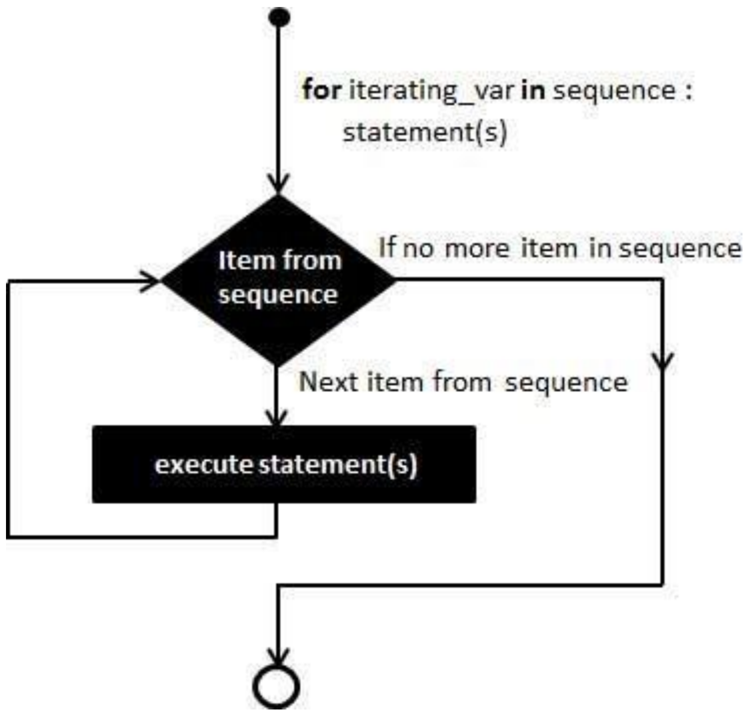
Loops in PHP are used to execute the same block of code a specified number of times. PHP supports the following four loop types.

- 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.



Syntax

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

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.

Example

The following example makes five iterations and changes the assigned value of two variables on each pass of the loop –

Live Demo

```
<html>
  <body>

    <?php
      $a = 0;
      $b = 0;

      for( $i = 0; $i<5; $i++ ) {
        $a += 10;
        $b += 5;
      }

      echo ("At the end of the loop a = $a and b = $b" );
```

```
   ?>

  </body>
</html>
```

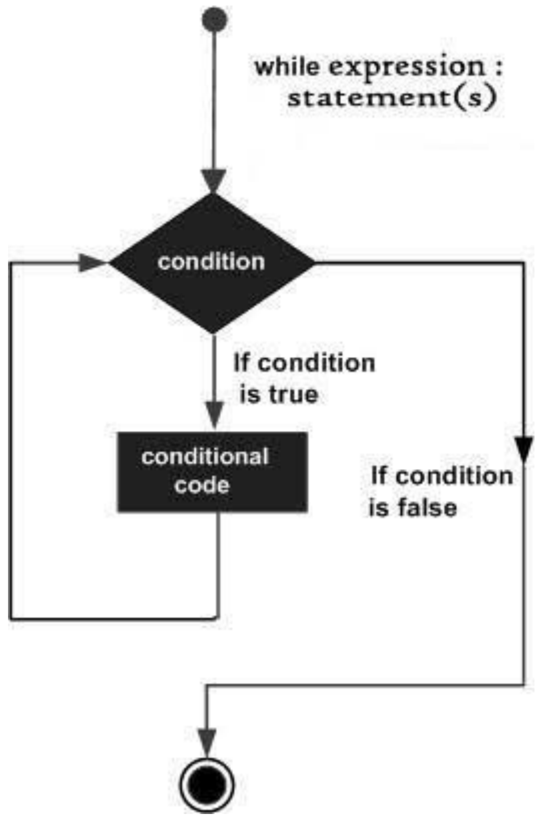
This will produce the following result –

```
At the end of the loop a = 50 and b = 25
```

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;
}
```

Example

This example decrements a variable value on each iteration of the loop and the counter increments until it reaches 10 when the evaluation is false and the loop ends.

Live Demo

```
<html>
  <body>

    <?php
      $i = 0;
      $num = 50;

      while( $i < 10) {
        $num--;
        $i++;
      }

      echo ("Loop stopped at i = $i and num = $num" );
    ?>

  </body>
</html>
```

This will produce the following result –

```
Loop stopped at i = 10 and num = 40
```

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);
```

Example

The following example will increment the value of i at least once, and it will continue incrementing the variable i as long as it has a value of less than 10 –

Live Demo

```
<html>
  <body>

    <?php
```

```
      $i = 0;
      $num = 0;

      do {
        $i++;
      }

      while( $i < 10 );
      echo ("Loop stopped at i = $i" );
    ?>

  </body>
</html>
```

This will produce the following result –

```
Loop stopped at i = 10
```

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;
}
```

Example

Try out following example to list out the values of an array.

Live Demo

```
<html>
  <body>

    <?php
      $array = array( 1, 2, 3, 4, 5);

      foreach( $array as $value ) {
        echo "Value is $value <br />";
      }
    ?>

  </body>
</html>
```

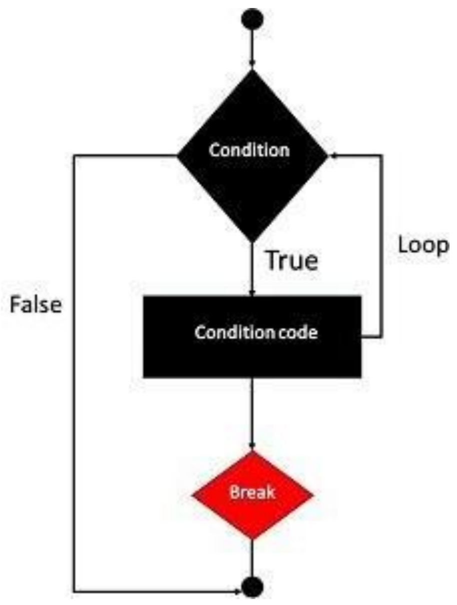
This will produce the following result –

```
Value is 1
Value is 2
Value is 3
Value is 4
Value is 5
```

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.



Example

In the following example condition test becomes true when the counter value reaches 3 and loop terminates.

Live Demo

```
<html>
  <body>

    <?php
      $i = 0;

      while( $i < 10) {
        $i++;
        if( $i == 3 )break;
      }
      echo ("Loop stopped at i = $i" );
    ?>

  </body>
</html>
```

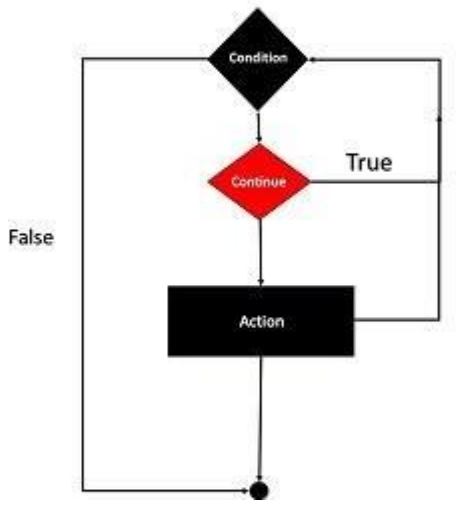
This will produce the following result −

```
Loop stopped at i = 3
```

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.



Example

In the following example loop prints the value of array but for which condition becomes true it just skip the code and next value is printed.

Live Demo

```
<html>
  <body>

    <?php
      $array = array( 1, 2, 3, 4, 5);

      foreach( $array as $value ) {
        if( $value == 3 )continue;
        echo "Value is $value <br />";
      }
      ?>

   </body>
</html>
```

This will produce the following result −

```
Value is 1
Value is 2
Value is 4
Value is 5
```