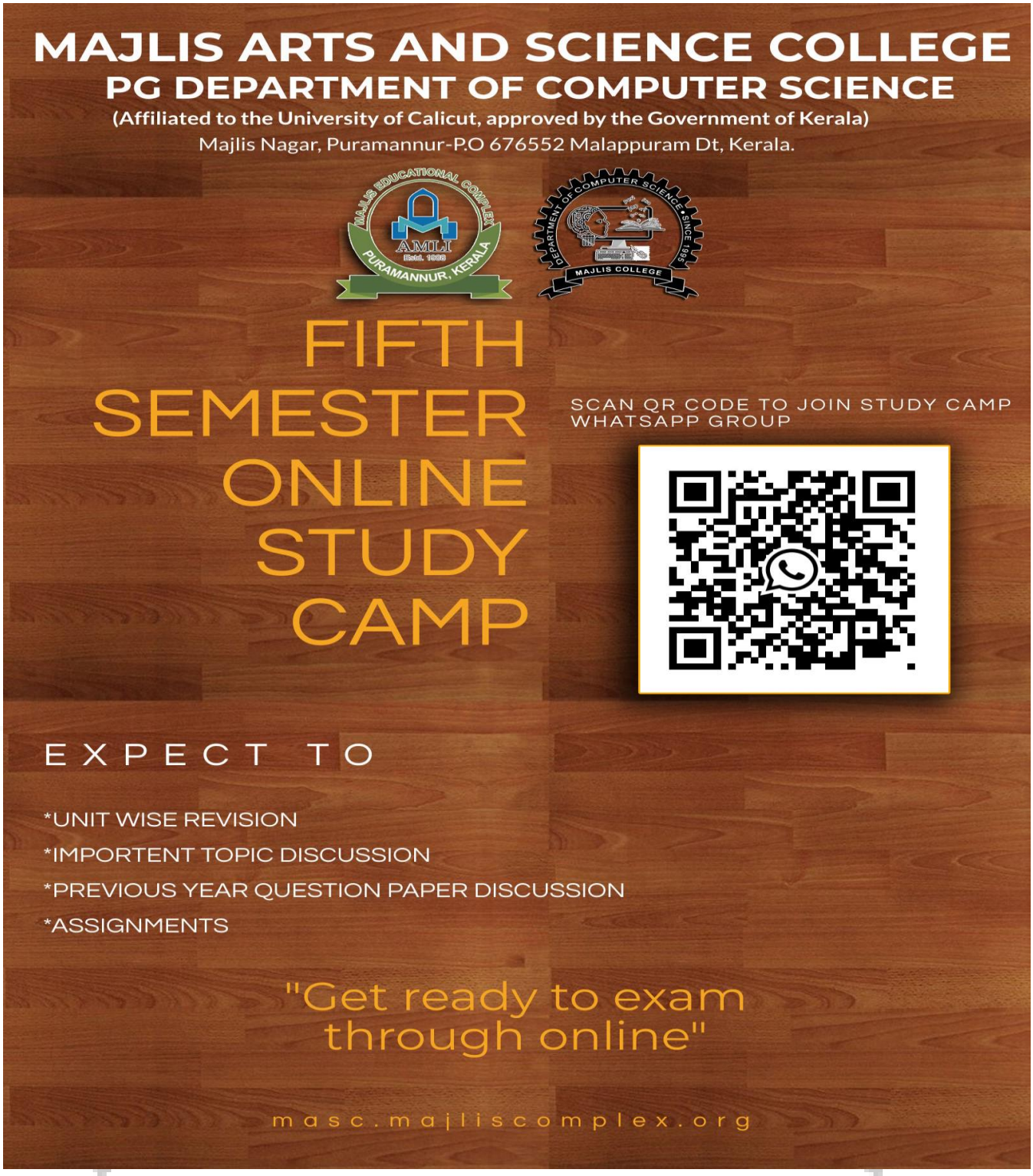


WEB PROGRAMMING USING PHP

MODULE 3

1. **What is PHP?**

   PHP is an acronym for "PHP: Hypertext Preprocessor". PHP is an open-source, interpreted, and object-oriented scripting language that can be executed at the server-side. PHP is well suited for web development. Therefore, it is used to develop web applications

2. **Write php Syntax**

   PHP source code is embedded in an HTML-based document, and is identified by special delimiting tags,

   <?php *content* ?>

3. **Why PHP is called scripting language?**

   The language which is used interpreter as a translator is called scripting language. In other words interpreted languages are called scripting languages. PHP is an interpreted language that is why PHP is called scripting language.

4. **Differentiate between Client side scripting and server side scripting**

   The client-side Scripting language usually runs on Client browser. It is completely browser dependent. The client-side scripting is performed by a browser. Source code is visible to user.It runs on user's computer.It does not provide security for data. HTML, CSS and javascript are example for Client side scripting

   Web servers are used to execute server side scripting. They are basically used to create dynamic pages. It can also access the file system residing at web server.It provides more security for data. Server-side scripting helps in connecting to the databases that are already present in the web server. PHP, Python, Java, Ruby etc are example of server side scripting.

5. **What are the different data types used in PHP**

   PHP Data types specify the different types of data that are supported in PHP language. There are total 8 data types supported in PHP, which are categorized into 3 main types. They are:

   1. **Scalar Types**: boolean, integer, float and string.
   2. **Compound Types**: array and object.
   3. **Special Types**: resource and NULL

Integers − An Integer data type is used to store any non-decimal numeric value. An integer value can be negative or positive, but it cannot have a decimal.

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

Booleans − A boolean data type can have two possible values, either **True** or **False**.

NULL − NULL data type is a special data type which means **nothing**. If you create any variable and do not assign any value to it, it will automatically have NULL stored in it. Also, we can use NULL value to empty any variable.

Strings − String data type in PHP is a sequence of characters enclosed within quotes. You can use single or double quotes. A string is a non-numeric data type. It holds letters or any alphabets, numbers, and even special characters.

Arrays – Arrays are named and indexed collections of other values.

Objects – Objects are instances of programmer-defined classes.

Resources – Resources are special variables that hold references to resources external to PHP.

6. **What is constant? How a constant is define in PHP?**

Constants are variables whose value cannot be changed. In other words, a constant value cannot change during the execution of the script. A constant name starts with a letter or underscore, followed by any number of letters, numbers, or underscores.

In PHP, there are two ways to define a constant:

1. Using the define() method.
2. Using the const keyword.

While naming a constant, **we don't have to use** $ symbol with the constant's name.

1. **Using define()**

Below is the syntax for using the define() function to create a constant.

define(name, value, case-insensitive)

1. name: Name of the constant
2. value: Value of the constant
3. case-insensitive: Specifies whether the constant name is case sensitive or not. It's default value is false, which means, by default, the constant name is case sensitive.

e.g

define(pi,3.14)

2. **Using the const Keyword**

We can also define constants in PHP using the const keyword.

e.g

const pi=3.14

7. **Explain the different features of PHP**

- **Performance:** PHP script is executed much faster than those scripts which are written in other languages such as JSP and ASP.
- **Open Source:** PHP source code and software are freely available on the web. All its components are free to download and use.
- **Embedded:** PHP code can be easily embedded within HTML tags and script.
- **Platform Independent:**PHP is available for WINDOWS, MAC, LINUX & UNIX operating system. A PHP application developed in one OS can be easily executed in other

OS also.

- **Database Support:**PHP supports all the leading databases such as MySQL, SQLite, ODBC, etc.
- **Loosely Typed Language**:PHP allows us to use a variable without declaring its datatype.
- **Web servers Support:**PHP is compatible with almost all local servers used today like Apache, Netscape, Microsoft IIS, etc.

8. **What is a variable in PHP? What are the rules for declaring PHP variable?**

   In PHP, a variable is declared using a **$ sign** followed by the variable name. As PHP is a loosely typed language, so we do not need to declare the data types of the variables.

   Rules for declaring PHP variable:

   - A variable must start with a dollar ($) sign, followed by the variable name.
   - It can only contain alpha-numeric character and underscore (A-z, 0-9, _).
   - A variable name must start with a letter or underscore (_) character.
   - A PHP variable name cannot contain spaces.
   - PHP variables are case-sensitive

9. **Difference between identical operator and equality operator**

   equality (== double equals) operator

   It compares only value of variable, not data types. it accepts two inputs to compare and return true if the values are same and return false if values are not same.

   e.g

```php
<?php
   $a = 123;
   $b = '123';
   if ($a == $b) {
      echo 'Values are same';
   }
   else {
      echo 'Values are not same';
   }
?>
```

   Identical Operator === operator

   Identical operator = = = allows for stricter comparison between variables. Identical operator returns true if both operands contain the same value and are of the same type.

```php
<?php
   $a = 123;
   $b = '123';
```

```
    if ($a === $b) {
        echo 'Values and types are same';
    }
    else {
        echo 'Values and types are not same';
    }
?>
```

10. Difference between Echo and Print in PHP:

Echo	Print
echo does not return any value.	print always returns an integer value, which is 1.
We can pass multiple strings separated by comma (,) in echo.	Using print, we cannot pass multiple arguments.
echo is faster than print statement.	print is slower than echo statement.

11. Explain comments used in PHP

Comments on source code will be useful for denoting the details the code logic. For an example, if we define a function in our program, we should add comment lines to state about the parameters passed to the function, what the function is doing and what it returns. These comment lines will be useful to understand the code easily. Adding comments on source code is the best programming practice.

PHP supports two types of commenting styles; those are the single-line comment and the multi-line comment.

- **Single line comment**

The (//) or hash(#) characters are used to add single line comments. Single line comment can also be referred as the inline comment.

- **Multi-line comment style**

We have to use /* and */ delimiters to add multi-line comments. By supporting multi-line statement in a comment line, we can add descriptive comments on our source code

12. Explain Different operators used in PHP

Operators are symbols that tell the PHP processor to perform certain actions.

PHP language supports following type of operators.

- Arithmetic Operators
- Comparison Operators

- Logical (or Relational) Operators
- Assignment Operators
- Conditional (or ternary) Operators
- String Operator
- Assignment Operators

- **Arithmetic Operators**

The arithmetic operators are used to perform common arithmetical operations, such as addition, subtraction, multiplication etc.

Operator	Type	Description
+	Addition	Calculates the sum of two operands
-	Subtraction	Calculates the difference between two operands
*	Multiplication	Multiplies two operands
/	Division	Divides two operands
%	Modulus	Returns the remainder from dividing the first operand by the second

Comparison Operators

The comparison operators are used to compare two values in a Boolean fashion.

Operator	Type	Description
==	Equal to	Returns *true* if first operand equals second
!=	Not equal to	Returns *true* if first operand is not equal to second
<>	Not equal to	Returns *true* if first operand is not equal to second
===	Identical to	Returns *true* if first operand equals second in both value and type
!==	Not identical to	Returns *true* if first operand is not identical to second in both value and type
<	Less than	Returns *true* if the value of the first operand is less than the second.
>	Greater than	Returns *true* if the value of the first operand is greater than the second
<=	Less than or equal to	Returns *true* if the value of the first operand is less than, or equal to, the second

>=	Greater than or equal to	Returns *true* if the value of the first operand is greater than, or equal to, the second

Logical Operators

The logical operators are typically used to combine conditional statements.

Operator	Type	Description
&&	AND	Performs a logical "AND" operation.
||	OR	Performs a logical "OR" operation.
Xor	XOR	Performs a logical "XOR" (exclusive OR) operation.

Increment and Decrement Operators

The increment/decrement operators are used to increment/decrement a variable's value.

++$a	Pre Increment,	Increments $a by one, then returns $a
$a++	Post Increment	Returns $a, then increments $a by one
--$b	Pre Decrement	Decrements $a by one, then returns $a
$b--	Post Decrement	Returns $a, then decrements $a by one

Conditional operator

The ternary operator is a conditional operator that decreases the length of code while performing comparisons . This method is an alternative for using if-else and nested if-else statements.

Syntax:

e.g (Condition) ? (Statement1) : (Statement2);

Condition: It is the expression to be evaluated which returns a boolean value.

Statement 1: it is the statement to be executed if the condition results in a true state.

Statement 2: It is the statement to be executed if the condition results in a false state.

e.g

```
<?php
$marks=40;
print ($marks>=40) ? "pass" : "Fail";
?>
```

String Operator

Concatenation	. (a dot)	It is used to concatenate (join together) two

		strings.
Concatenation Assignment	.= (dot equal to)	It is used to append one string to another.

Assignment operators

The assignment operators are used to assign values to variables.

Operator	**Description**
=	Assign
+=	Add and assign
-=	Subtract and assign
*=	Multiply and assign
/=	Divide and assign quotient
%=	Divide and assign modulus

13. What is conditional statement? Explain different conditional statements used in PHP

A conditional statement is also known as a decision statement. A decision statement is used for executing commands based on some situation or conditions. There are four types of conditional statements in PHP, which are as follows.

- The **if** statement
- The **if...else** statement
- The **if...elseif....else** statement
- Switch …. Case Statement

14. Explain if statements in PHP with example

A conditional statement is also known as a decision statement. A decision statement is used for executing commands based on some situation or conditions. There are four types of conditional statements in PHP, which are as follows.

- The **if** statement
- The **if...else** statement
- The **if...elseif....else** statement
- Switch …. Case Statement

The if Statement

The *if* statement is used to execute a block of code only if the specified condition evaluates to true. This is the simplest PHP's conditional statements and can be written like:

Syntax

if (*condition*) {
code to be executed if condition is true;
}

Flow chart

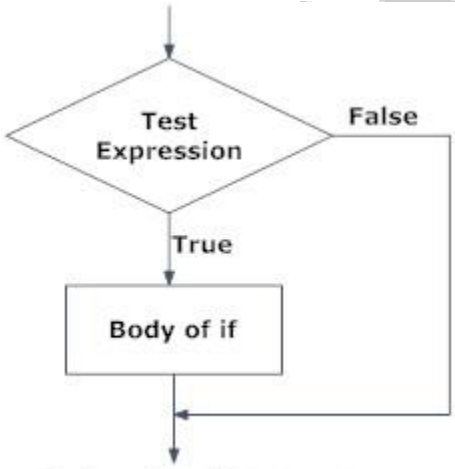


Fig: Operation of if statement

e.g

```
<?php
$a=10
if ($a%2==0)
{
echo “Even Number”;
}
?>
```

The if...else Statement

The *if...else* statement allows you to execute one block of code if the specified condition is evaluates to true and another block of code if it is evaluates to false.

Syntax:

```
if (condition)
{
block_of_code_1
}
else
{
block_of_code_2
}
```

block_of_code_1: This would execute if the given condition is true
block_of_code_2: This would execute if the given condition is false

e.g

```
<?php
$a=5
if ($a%2==0)
{
echo “Even Number”;
}
else
{
echo “Odd Number”;
?>
```

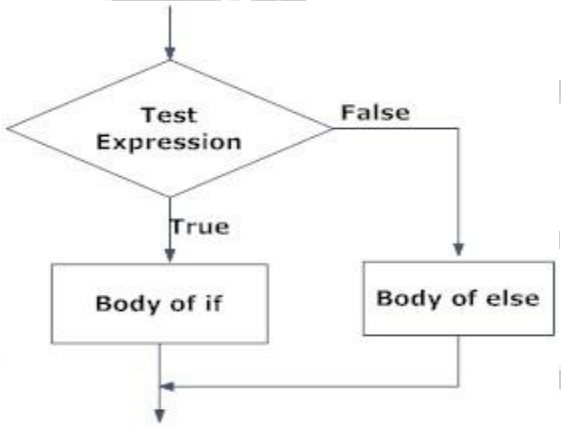


Fig: Operation of if...else statement

The if...elseif....else Statement

We can check multiple conditions using this statement.

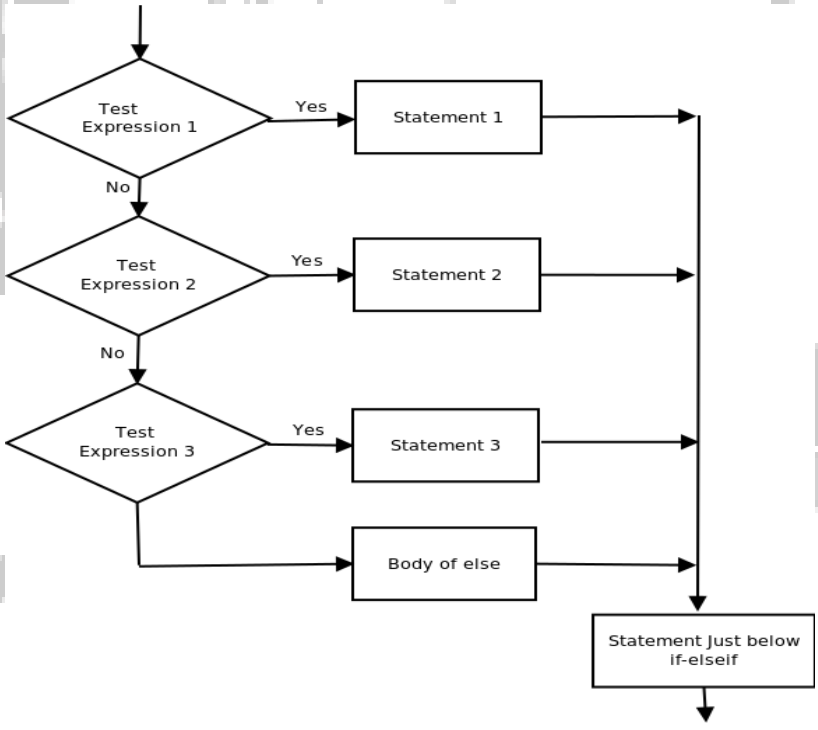
Syntax:

```
if (condition)
{
code to be executed if this condition is true;
}
elseif (condition)
{
code to be executed if this condition is true;
}
else
{
code to be executed if all conditions are false;
}
```

Example

```
<?php
$x = 20;
if($x > 0)
{
 echo "$x is Positive Number";
}
elseif($x == 0)
{
 echo "$x is zero";
}
else
{
 echo "$x is Negative Number";
}
```

Flow Chart



15. Explain switch statements with example

The switch statement has multiple choices. In the switch statement we use the **case**, **default** and **break** keywords. The **case** keyword is used to specify each choice. The **default** keyword is used to

execute a statement when no case matches any switch expression. The **break** keyword is used to terminate the switch case.

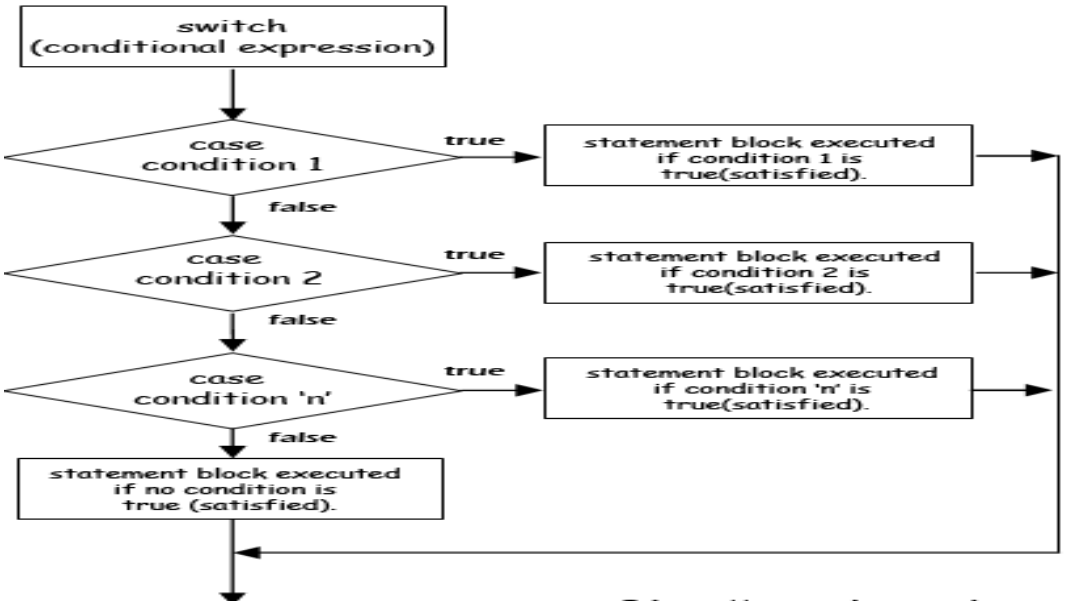
Syntax

```
switch(variable){
case value1:
 // code block 1
break;
 case value2:
 // code block 2
break;
 default:
 // default code block
break;
}
```

Let's examine the switch statement syntax in more detail:

- First, you put a variable or expression that you want to test within parentheses after the switch keyword.
- Second, inside the curly braces are multiple case constructs where you put the values you want to compare with the variable or expression. In case the value of the variable or expression matches the value in a case construct, the code block in the corresponding case construct will execute. If the value of the variable or expression does not match any value, the code block in the default construct will execute.
- Third, the break statement is used in each case or default construct to exit the entire switch statement.

Flow chart



16. Explain different looping statements used in PHP with example

Looping Statement

A Loop is an Iterative Control Structure that involves executing the same number of code a number of times until a certain condition is met.

- **The While Loop**

The While statement executes a block of code if and as long as a specified condition evaluates to true. If the condition becomes false, the statements within the loop stop executing and control passes to the statement following the loop. The While loop syntax is as follows:

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

Example

```
<?php
$i = 1;
while($i <= 10)
{
$i++;
echo "The number is " . $i . "<br>";
}
?>
```

- **Do...While Loop**

The Do...While statements are similar to While statements, except that the condition is tested at the end of each iteration, rather than at the beginning. This means that the Do...While loop is guaranteed to run at least once. The Do...While loop syntax is as follows:

```
do
{
code to be exected;
}
while (condition);
<?php
$i = 1;
do
{
$i++;
echo "The number is " . $i . "<br>";
}
while($i <= 10);
?>
```

- **for Loop**

The for loop repeats a block of code as long as a certain condition is met. It is typically used to execute a block of code for certain number of times. For this reason, the For loop is known as a definite loop. The for loop syntax is as follows:

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

The for statement takes three expressions inside its parentheses, separated by semi-colons. When the for loop executes, the following occurs:

1. The initializing expression is executed. This expression usually initializes one or more loop counters.
2. The condition expression is evaluated. If the value of condition is true, the loop statements execute. If the value of condition is false, the For loop terminates.
3. The update expression increment executes.

Example

```
<?php
for($i=1; $i<=3; $i++)
{
echo "The number is " . $i . "<br>";
}
?>
```

- **foreach Loop**

The foreach loop is a variation of the for loop and allows you to iterate over elements in an array. There are two different versions of the foreach loop. The foreach loop syntaxes are as follows:

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

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

Example

```
<?php
$colors = array("Red", "Green", "Blue");
foreach($colors as $value)
{
echo $value . "<br>";
}
?>
```

17. Differentiate between Break and Continue Statement in PHP

A break statement can be used to terminate or to come out from the loop. It can be used in switch statement to break and come out from the switch statement after each case expression. Whenever, break statement is encounter within the program then it will break the current loop or block. A break statement is normally used with if statement.

Example

```
?php
        for( $i = 1; $i <= 10 ; $i++ )
  {
      if ($i > 5)
         break;    // terminate loop
      echo "$i"."</br>" ;
  }
?>
```

A continue statement can be used into the loop when we want to skip some statement to be executed and continue the execution of above statement based on some specific condition. Continue is also used with if statement. When compiler encounters continue, statements after continue are skipped and control transfers to the statement above continue.

```
<?php
    for ( $i = 51 ; $i <= 100 ; $i++ )
  {
     if($i % 2==0 )
          continue ;
          echo " $i "."</br>" ;
  }
```

18. Explain different output statements used in PHP

In PHP there are two basic ways to get output: **echo** and **print**.

- **print()**

The *print()* is used to create PHP print statement to print given data to the browser. It accepts single data and prints it on the browser. It is a PHP language construct and not a function. So, we can use 'print' without parenthesis for creating a print statement.

Display Strings of Text

```
<?php
print "Apple";
echo ("Apple");
?>
```

Display HTML Code

```
<?php
print"<h1>This is a simple heading.</h1>";
?>
```

Display Variables

```
<?php
$num=234;
Print  $num;
?>
```

- **echo()**

The echo statement can display anything that can be displayed to the browser, such as string, numbers, variables values, the results of expressions etc.

Since echo is a language construct not actually a function, you can use it without parentheses The *echo()* will accept multiple data separated by commas. While sending multiple values to the *echo()* statement, we have to use parenthesis to enclose the values. If we use single data in an echo() statement, we can ignore parenthesis.

e.g

Display Strings of Text

```
<?php
echo "Apple";
echo ("Apple","Orange","Grapes");
?>
```

Display HTML Code

```
<?php
echo"<h1>This is a simple heading.</h1>";
?>
```

Display Variables

```
<?php
$num=234;
echo $num;
?>
```