**Module1-**

**Introduction to php programming language**

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.

## **Common uses of PHP**

* PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
* PHP can handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user.
* You add, delete, modify elements within your database through PHP.
* Access cookies variables and set cookies.
* Using PHP, you can restrict users to access some pages of your website.
* It can encrypt data.

## **Characteristics of PHP**

Five important characteristics make PHP's practical nature possible −

* Simplicity
* Efficiency
* Security
* Flexibility
* Familiarity

## **"Hello World" Script in PHP**

To get a feel for PHP, first start with simple PHP scripts. Since "Hello, World!" is an essential example, first we will create a friendly little "Hello, World!" script.

As mentioned earlier, PHP is embedded in HTML. That means that in amongst your normal HTML (or XHTML if you're cutting-edge) you'll have PHP statements like this −

[Live Demo](http://tpcg.io/cJf0To)

<html>

<head>

<title>Hello World</title>

</head>

<body>

<?php echo "Hello, World!";?>

</body>

</html>

It will produce following result −

Hello, World!

If you examine the HTML output of the above example, you'll notice that the PHP code is not present in the file sent from the server to your Web browser. All of the PHP present in the Web page is processed and stripped from the page; the only thing returned to the client from the Web server is pure HTML output.

All PHP code must be included inside one of the three special markup tags ATE are recognised by the PHP Parser.

<?php PHP code goes here ?>

<? PHP code goes here ?>

<scriptlanguage="php"> PHP code goes here </script>

A most common tag is the <?php...?> and we will also use the same tag in our tutorial.

From the next chapter we will start with PHP Environment Setup on your machine and then we will dig out almost all concepts related to PHP to make you comfortable with the PHP language.

**AdvantagesofPHP**

#### . Open Source

PHP is open-source and free of cost, which helps developers to install it quickly and readily available for use. There are a lot of [PHP frameworks](https://www.educba.com/top-php-frameworks/) and developer can choose any of the frameworks to work. All the features and tools will be provided to the developer for that framework easily. As it is open-source, it makes the system ready with PHP in quick time and makes the web development faster with the help of providing the tools and other features easily.

#### 2. Platform Independent

PHP is mainly supported by all the operating systems like Windows, Unix, Linux etc. The PHP based developed web applications can be easily run on any platform. It can be integrated with other programming language and database easily and there is no requirement of re-development. It helps in saving a lot of effort and cost.

#### 3. Simple and Easy

This advantage of PHP is simple and easy to learn and code. It is mainly organized code and clean, which helps the new developers also. The command functions of PHP can easily learn and understood. The one who knows any programming language can easily work on PHP. It is simple to learn, as its learning curve is not large. The syntax is simple and flexible to use.

#### 4. Database

PHP is easily connected with the database and make the connection securely with databases. It has a built-in module that is used to connect to the database easily. There are many web applications, which require strong programming language with a good database management system. PHP and its [database connection](https://www.educba.com/php-database-connection/) solve the purpose for development of web applications. It reduces the time to connect to a database management system as well. Multiple databases can be integrated with PHP.

 Popular Course in this category

**[PHP Training (5 Courses, 3 Project)](https://www.educba.com/software-development/courses/php-course/?btnz=edu-blg-inline-banner1" \t "_blank)**[5 Online Courses | 3 Hands-on Project | 28+ Hours | Verifiable Certificate of Completion | Lifetime Access   
4.5 (5,507 ratings)](https://www.educba.com/software-development/courses/php-course/?btnz=edu-blg-inline-banner1" \t "_blank)

[Course Price  
₹4999](https://www.educba.com/software-development/courses/php-course/?btnz=edu-blg-inline-banner1" \t "_blank)~~[₹27999](https://www.educba.com/software-development/courses/php-course/?btnz=edu-blg-inline-banner1" \t "_blank)~~**[View Course](https://www.educba.com/software-development/courses/php-course/?btnz=edu-blg-inline-banner1" \t "_blank)**

Related Courses

[Java Servlet Training (6 Courses, 12 Projects)](https://www.educba.com/software-development/courses/java-servlet-training/?btnz=edu-blg-inline-banner1)[All in One Software Development Bundle (600+ Courses, 50+ projects)](https://www.educba.com/software-development/courses/software-development-course/?btnz=edu-blg-inline-banner1)

#### 5. Fast

PHP is known as the fastest Programming language as compared to another. PHP applications can be easily loaded over the slow Internet and data speed. Other applications take a lot of time to connect the database and fetch the data after executing certain queries to the database. PHP does not face this problem and it loads the website very easily and fast. The fast speed of PHP provides the developer with an edge to develop the web applications in PHP programming language.

#### 6. Maintenance

PHP framework is mainly used to make the web application development easier and maintain the code automatically. The model view controller architecture in PHP framework helps the code to be easily maintained and used. The MVC architecture helps the separation of a file for different module separately.

#### 7. Support

This advantage of PHP has great online support and community, which helps the new developers to help in writing the code and developing the web applications. The documentation provided at the official site helps in using the different features of PHP and its framework. The latest updates are released timely by the PHP to make it better for the developer to develop the web-based applications.

#### 8. Testing

PHP based web applications can be easily tested. PHP unit uses to perform the unit testing quickly and easily. It also helps the programmers to [write test](https://www.educba.com/how-to-write-test-case/) cases and perform the testing smoothly. For PHP based web applications, the developers do not need to write the additional code. PHP frameworks help in automating the different tasks.

#### 9. Security

PHP frameworks built-in feature and tools make it easier to protect the web applications from the outer attacks and security threats. The security threats can be like [SQL injection](https://www.educba.com/cheat-sheet-sql/), data tampering, and forgery etc. To protect from these security threats, developers used PHP frameworks for developing web applications.

#### 10. Stable

PHP is also stable as compared to other programming languages. It has been in existence for a long time. The developers have worked on PHP to make it easy for the programmers to work on developing the PHP web-based applications. They have fixed the issues and bugs over the period of time for the different version of PHP and make it very stable.

**The server side architecture Decomposed**

**History of Php**

PHP was conceived sometime in the fall of 1994 by [RasmusLerdorf](mailto:rasmus@php.net" \t "_top). Early non-released versions were used on his home page to keep track of who was looking at his online resume. The first version used by others was available sometime in early 1995 and was known as the Personal Home Page Tools. It consisted of a very simplistic parser engine that only understood a few special macros and a number of utilities that were in common use on home pages back then. A guestbook, a counter and some other stuff. The parser was rewritten in mid-1995 and named PHP/FI Version 2. The FI came from another package Rasmus had written which interpreted html form data. He combined the Personal Home Page tools scripts with the Form Interpreter and added mSQL support and PHP/FI was born. PHP/FI grew at an amazing pace and people started contributing code to it.

It is difficult to give any hard statistics, but it is estimated that by late 1996 PHP/FI was in use on at least 15,000 web sites around the world. By mid-1997 this number had grown to over 50,000. Mid-1997 also saw a change in the development of PHP. It changed from being Rasmus' own pet project that a handful of people had contributed to, to being a much more organized team effort. The parser was rewritten from scratch by ZeevSuraski and Andi Gutmans and this new parser formed the basis for PHP Version 3. A lot of the utility code from PHP/FI was ported over to PHP3 and a lot of it was completely rewritten.

Today (end-1999) either PHP/FI or PHP3 ships with a number of commercial products such as C2's StrongHold web server and RedHat Linux. A conservative estimate based on an extrapolation from numbers provided by [NetCraft](http://www.netcraft.com/" \t "_top) (see also [Netcraft Web Server Survey](http://www.netcraft.com/survey/" \t "_top)) would be that PHP is in use on over 1,000,000 sites around the world. To put that in perspective, that is more sites than run Netscape's flagship Enterprise server on the Internet.

Also as of this writing, work is underway on the next generation of PHP, which will utilize the powerful [Zend](http://www.zend.com/) scripting engine to deliver higher performance, and will also support running under webservers other than Apache as a native server module.

# **OBJECT-ORIENTED PROGRAMMING IN PHP**

PHP is a server-side scripting language, mainly used for web development but also used as a general-purpose programming language. Object-Oriented Programming (PHP OOP), is a type of programming language principle added to php5, that helps in building complex, reusable web applications.

The Object Oriented concepts in PHP are:

* Class − This is a programmer-defined data type, which includes local functions as well as local data. You can think of a class as a template for making many instances of the same kind (or class) of object.
* Object − An individual instance of the data structure defined by a class. You define a class once and then make many objects that belong to it. Objects are also known as instance.
* Inheritance − When a class is defined by inheriting existing function of a parent class then it is called inheritance. Here child class will inherit all or few member functions and variables of a parent class.
* Polymorphism − This is an object oriented concept where same function can be used for different purposes. For example function name will remain same but it make take different number of arguments and can do different task.
* Overloading − a type of polymorphism in which some or all of operators have different implementations depending on the types of their arguments. Similarly functions can also be overloaded with different implementation.
* Data Abstraction − Any representation of data in which the implementation details are hidden (abstracted). \* Encapsulation − refers to a concept where we encapsulate all the data and member functions together to form an object.
* Constructor − refers to a special type of function which will be called automatically whenever there is an object formation from a class.
* Destructor − refers to a special type of function which will be called automatically whenever an object is deleted or goes out of scope.

**Class & Object:**

Class is a programmer-defined data type, which includes local methods and local variables. It is a collection of objects. Object has properties and behaviour.

First we have to define a php class, where classname should be same as filename.

Example for simple class:

<?php

class Books{

public function name(){

echo "Drupal book";

}

public function price(){

echo "900 Rs/-";

}

}

//To create php object we have to use a new operator. Here php object is the object of the Books Class.

$obj = new Books();

$obj->name();

$obj->price();

?>

Output:

Drupal book

900 Rs/-

**Creating Objects in PHP**

When class is created, we can create any number of objects to that class. The object is created with the help of new keyword.

**Calling Member Function**

When the object is created we can access the variables and method function of the class with the help of operator ‘->, accessing the method is done to get the information of that method. Also look into how we can access object properties via variables

<?php

class Mobile {

/\* Member variables \*/

var $price;

var $title;

/\* Member functions \*/

function setPrice($par){

$this->price = $par;

}

function getPrice(){

echo $this->price ."

";

}

function setName($par){

$this->title = $par;

}

function getName(){

echo $this->title ."

";

}

}

$Samsung = new Mobile();

$Xiaomi = new Mobile();

$Iphone = new Mobile();

$Samsung->setName( "SamsungS8 );

$Iphone->setName( "Iphone7s" );

$Xiaomi->setName( "MI4" );

$Samsung->setPrice( 90000 );

$Iphone->setPrice( 65000 );

$Xiaomi->setPrice( 15000 );

Now you call another member functions to get the values set by in above example

$Samsung->getName();

$Iphone->getName();

$Xiaomi->getName();

$Samsung->getPrice();

$Iphone->getPrice();

$Xiaomi->getPrice();

?>

Output

Samsung S8

Iphone S7

MI4

90000

65000

15000

**Inheritance**

When the properties and the methods of the parent class are accessed by the child class, we call the concept has inheritance. The child class can inherit the parent method and give own method implementation, this property is called overridden method. When the same method of the parent class is inherited we call as inherited method. Now let us see types of inheritance supported in Object Oriented Programming and corresponding Php inheritance examples.

Types Of Inheritance

* Single Level Inheritance
* Multilevel Inheritance

**Single Level Inheritance:**

In Single Level Inheritance the Parent class methods will be extended by the child class. All the methods can be inherited.

Example for Single Level Inheritance

<?php

class A {

public function printItem($string) {

echo 'Hi : ' . $string . "\n;

}

public function printPHP() {

echo 'I am from valuebound' . PHP\_EOL;

}

}

class B extends A {

public function printItem($string) {

echo 'Hi: ' . $string . PHP\_EOL;

}

public function printPHP() {

echo "I am from ABC";

}

}

$a = new A();

$b = new B();

$a->printItem('Raju');

$a->printPHP();

$b->printItem('savan');

$b->printPHP();

?>

Output

Hi : Pavan

I am from valuebound

Hi: savan

I am from ABC

**MultiLevel Inheritance :**

In MultiLevel Inheritance, the parent class method will be inherited by child class and again subclass will inherit the child class method.

<?php

class A {

public function myage() {

return "age is 80\n";

}

}

class B extends A {

public function mysonage() {

return "age is 50\n";

}

}

class C extends B {

public function mygrandsonage() {

return "age is 20\n";

}

public function myHistory() {

echo "Class A " .$this->myage();

echo "Class B ".$this->mysonage();

echo "Class C " . $this->mygrandsonage();

}

}

$obj = new C();

$obj->myHistory();

?>

Output

Class A age is 80

Class B age is 50

Class C age is 20

1

​

2

<?php

3

/\*PHP program to build a class and store the data of the mobile phones with screen size,ram,company and processor along with Constructor and function to estimate cost of phone based on conditions

4

samsung:7000(pr)

5

moto:5999

6

lenovo:4000

7

nokia:6100

8

est cost = pr\*(scr size/4.0)+pr\*(ram/1.0)+pr\*(proc/1.0)+0.05\*pr

9

​

10

Input

11

enter screen size:5

12

enter ram:3

13

enter company name:moto

14

enter processor:3

15

​

16

Output

17

price:43792

18

\*/

19

**class**Mobile {

20

**var**$screen\_size;

21

**var**$ram;

22

**var**$company;

23

**var**$processor;

24

**var**$ss;

25

**var**$r;

26

**var**$pr;

27

**var**$proc;

28

//initializes the value of screensize,ram,company and processor

29

**function**Mobile($screen\_size,$ram,$processor,$company)

30

{

31

 $this->screen\_size=$screen\_size;

32

 $this->ram=$ram;

33

 $this->processor=$processor;

34

 $this->company=$company;

35

}

36

//calculates the price of the mobile according to the features

37

**function**set\_price()

38

{

39

 $s=$this->screen\_size;

40

 $r=$this->ram;

41

 $proc=$this->processor;

42

 $pr1=$this->company;

43

 //converts the company name to lower case

44

 $pr=strtolower($pr1);

45

 //this block gets executed if the user enters the valid company i.e available in our catalog

46

**if**($pr=="samsung"||$pr=="moto"||$pr=="lenovo"||$pr=="nokia" )

47

{

48

**if**($pr=="samsung")

49

  {

50

    $cost=7000\*($s/4.0)+7000\*($r/1.0)+7000\*($proc/1.0)+0.05\*7000;

51

  }

52

**elseif**($pr=="moto")

53

  {

54

    $cost=5999\*($s/4.0)+5999\*($r/1.0)+5999\*($proc/1.0)+0.05\*5999;

55

  }

56

**elseif**($pr=="lenovo")

57

  {

58

    $cost=4000\*($s/4.0)+4000\*($r/1.0)+4000\*($proc/1.0)+0.05\*4000;

59

  }

60

**elseif**($pr=="nokia")

61

  {

62

    $cost=6100\*($s/4.0)+6100\*($r/1.0)+6100\*($proc/1.0)+0.05\*6100;

63

  }

64

  //displays the cost

65

**echo**"price:".(int)$cost."\n";

66

}

67

**else**

68

**echo**"specified company not available in our catalog\n";

69

}

70

}

71

**echo**"enter screen size:";

72

$ss=trim(fgets(STDIN, 1024));

73

**echo**"enter ram:";

74

$rr=trim(fgets(STDIN, 1024));

75

**echo**"enter company name:";

76

$cc=trim(fgets(STDIN, 1024));

77

**echo**"enter processor:";

78

$pp=trim(fgets(STDIN, 1024));

79

//calls the mobile function

80

$mobile1=**new**Mobile($ss,$rr,$pp,$cc);

81

//calls the function set\_price

82

$mobile1->set\_price();

83

​

84

?>

85

​

[Try](https://www.greycampus.com/codelabs/php/php-lab/oops-example)

1

<?php

2

/\*PHP program to use the mobile phone class as parent and build the accessory class for a shop for biling purpose.

3

​

4

Input

5

enter screen size:5

6

enter ram:3

7

enter company name:moto

8

enter processor:3

9

do u want screenguard enter 1 if yes enter 0 if no:1

10

do u want pouch enter 1 if yes enter 0 if no:0

11

do u want headset enter 1 if yes enter 0 if no:1

12

​

13

* Output

14

total cost of your moto with accesories:44692

15

\*?>

16

**class**Mobile {

17

**var**$screen\_size;

18

**var**$ram;

19

**var**$company;

20

**var**$processor;

21

**var**$ss;

22

**var**$r;

23

**var**$pr;

24

**var**$proc;

25

//initializes the value of screensize,ram,company and processor

26

**function**Mobile($screen\_size,$ram,$company,$processor)

27

{

28

 $this->screen\_size=$screen\_size;

29

 $this->ram=$ram;

30

 $this->processor=$processor;

31

 $this->company=$company;

32

}

33

//calculates the price of the mobile according to the features

34

**function**set\_price()

35

{

36

  $s=$this->screen\_size;

37

  $r=$this->ram;

38

  $pr1=$this->company;

39

  $proc=$this->processor;

40

  //converts the company name to lower case

41

  $pr=strtolower($pr1);

42

  //this block gets executed if the user enters the valid company i.e available in our catalog

43

**if**($pr=="samsung"||$pr=="moto"||$pr=="lenovo"||$pr=="nokia" )

44

  {

45

**if**($pr=="samsung")

46

    {

47

     $cost=7000\*($s/4.0)+7000\*($r/1.0)+7000\*($proc/1.0)+0.05\*7000;

48

    }

49

**elseif**($pr=="moto")

50

    {

51

     $cost=5999\*($s/4.0)+5999\*($r/1.0)+5999\*($proc/1.0)+0.05\*5999;

52

    }

53

**elseif**($pr=="lenovo")

54

    {

55

     $cost=4000\*($s/4.0)+4000\*($r/1.0)+4000\*($proc/1.0)+0.05\*4000;

56

    }

57

**elseif**($pr=="nokia")

58

    {

59

     $cost=6100\*($s/4.0)+6100\*($r/1.0)+6100\*($proc/1.0)+0.05\*6100;

60

    }

61

**return** (int)$cost;

62

  }

63

**else**

64

**echo**"enter a valid company\n";

65

  }

66

}

67

​

68

//extends the parent class

69

**class**Bill**extends**Mobile

70

{

71

**var**$screen\_guard;

72

**var**$pouch;

73

**var**$head\_set;

74

**function**Bill($screen\_size,$ram,$company,$processor,$screen\_guard,$pouch,$head\_set)

75

{

76

 //inherits the values of the parent class

77

**parent**::\_\_construct($screen\_size,$ram,$company,$processor);

78

 $this->screen\_guard=$screen\_guard;

79

 $this->pouch=$pouch;

80

 $this->head\_set=$head\_set;

81

}

82

**function**set\_bill()

83

{

84

$sg=$this->screen\_guard;

85

 $p=$this->pouch ;

86

 $hs=$this->head\_set;

87

 //calls the setprice function in parent class

88

 $cost1=$this->set\_price();

89

 //calculates the price of the requirements

90

 $cost2=$sg\*200+$p\*300+$hs\*700;

91

**echo**"total cost of your ". $this->company. "with accesories:".(int)($cost1+$cost2)."\n";

92

}

93

}

94

//enters all the details of requires features

95

**echo**"enter screen size:";

96

$ss=trim(fgets(STDIN, 1024));

97

**echo**"enter ram:";

98

$rr=trim(fgets(STDIN, 1024));

99

**echo**"enter company name:";

100

$cc=trim(fgets(STDIN, 1024));

101

**echo**"enter processor:";

102

$pp=trim(fgets(STDIN, 1024));

103

**echo**"do u want screenguard enter 1 if yes enter 0 if no:";

104

$sg=trim(fgets(STDIN, 1024));

105

**echo**"do u want pouch enter 1 if yes enter 0 if no:";

106

$po=trim(fgets(STDIN, 1024));

107

**echo**"do u want headset enter 1 if yes enter 0 if no:";

108

$hs=trim(fgets(STDIN, 1024));

109

//calls the sub class constructor

110

$mobile1=**new**Bill($ss,$rr,$cc,$pp,$sg,$po,$hs);

111

//calls the bill compute function

112

$mobile1->set\_bill();

113

?>

114

​

[Try](https://www.greycampus.com/codelabs/php/php-lab/oops-example-1)

1

<?php

2

/\*PHP program to use Function Overriding in a class

3

​

4

Output

5

B's hello

6

A's good bye

7

\*/

8

**class**A

9

{

10

#constructor of A

11

**function**A()

12

{

13

 $x=1;

14

}

15

#m1 function of parent

16

**function**m1($Ab)

17

{

18

**echo**"A's ".$Ab."\n";

19

}

20

}

21

**class**B**extends**A

22

{

23

#constructor of B

24

**function**B()

25

{

26

  $y=1;

27

}

28

#m1 function of child

29

# print("m1 from B")

30

**function**m1($Ab)

31

{

32

**echo**"B's ".$Ab."\n";

33

}

34

}

35

$c=**new**B();

36

$d=**new**A();

37

#c.m1() prints hello from B

38

$c->m1("hello");

39

$d->m1("good bye");

40

​

41

?>

42

**Benefits in running php as a server side script**

1. Server-side scripting prevents increasing of the load as it **does not require plugins or browser scripting technology** (such as Javascript). Overloading leads to [**problems like slow loading**](http://www.apachebooster.com/#features), high CPU usage, and even freezing.
2. It is used to create pages dynamically on the fly. Based on the user interaction, new pages can instantly be created.
3. Server-side scripting is necessary to run dynamic pages on browsers that do not fully support [Javascript](https://www.javascript.com/" \t "_blank).
4. Server-side scripting does not depend on **[browser](http://apachebooster.com/kb/basics-of-web-browser-web-server-big-data-and-hadoop/)**processing as all the processing is performed on the server side.
5. Website owners can create their own applications and make use of CMS (content management system) to easily create and update content on the web without coding as the Server-side scripting languages like PHP can be configured to run [**CMS applications, like WordPress and Joomla**](http://apachebooster.com/kb/compare-top-cms-sites-wordpress-vs-drupal-vs-joomla/).
6. As the scripting is done on the server, it is not sent back to the browser, which prevents it from being cloned, copied, or scrutinized for [**hacking vulnerabilities**](http://www.ndimensionz.com/vulnerability-assessment-penetration-testing.php).
7. Loading time of the web pages is often reduced with Server-side scripting which helps to improve your site’s Google ranking.
8. An increased security is ensured for user privacy and is the preferred choice for e-commerce, membership and social media sites.

### **Disadvantages of server side scripting**

* Server-side scripting is slow at times like: 1.) If the user disconnects from the internet, or 2.) The web hosting is down, Then the script may take a long time to execute.
* The scripting software has to be installed on the server.
* New security concerns are associated with the dynamic scripts as in some cases hackers exploit the code flaws to gain access to servers.
* A [**database**](http://www.ndimensionz.com/kb/importance-of-database-security-and-requirements/) is required to store the dynamic data that needs a regular backing up and has to be kept secure.
* Here the **pages have to be refreshed often** in order to show the dynamic content. Ajax, an innovative method of exchanging data with a server is used by the developers which resolve the problem.
* Websites using large applications and with heavy traffic have to make use of more powerful hosting methods like dedicated servers or cloud hosting to cope with demand.

|  |  |
| --- | --- |
| **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 single value. There are 4 scalar data types in PHP.   1. [boolean](https://www.javatpoint.com/php-data-types#boolean) 2. [integer](https://www.javatpoint.com/php-data-types#integer) 3. [float](https://www.javatpoint.com/php-data-types#float) 4. [string](https://www.javatpoint.com/php-data-types#string)  **PHP Data Types: Compound Types** It can hold multiple values. There are 2 compound data types in PHP.   1. [array](https://www.javatpoint.com/php-data-types#array) 2. [object](https://www.javatpoint.com/php-data-types#object)  **PHP Data Types: Special Types** There are 2 special data types in PHP.   1. [resource](https://www.javatpoint.com/php-data-types#resource) 2. [NULL](https://www.javatpoint.com/php-data-types#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:**   1. <?php 2. **if** (TRUE) 3. echo "This condition is TRUE."; 4. **if** (FALSE) 5. echo "This condition is FALSE."; 6. ?>   **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:**   1. <?php 2. $dec1 = 34; 3. $oct1 = 0243; 4. $hexa1 = 0x45; 5. echo "Decimal number: " .$dec1. "</br>"; 6. echo "Octal number: " .$oct1. "</br>"; 7. echo "HexaDecimal number: " .$hexa1. "</br>"; 8. ?>   **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:**   1. <?php 2. $n1 = 19.34; 3. $n2 = 54.472; 4. $sum = $n1 + $n2; 5. echo "Addition of floating numbers: " .$sum; 6. ?>   **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:**   1. <?php 2. $company = "Javatpoint"; 3. //both single and double quote statements will treat different 4. echo "Hello $company"; 5. echo "</br>"; 6. echo 'Hello $company'; 7. ?>   **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:**   1. <?php 2. $bikes = **array** ("Royal Enfield", "Yamaha", "KTM"); 3. var\_dump($bikes);   //the var\_dump() function returns the datatype and values 4. echo "</br>"; 5. echo "Array Element1: $bikes[0] </br>"; 6. echo "Array Element2: $bikes[1] </br>"; 7. echo "Array Element3: $bikes[2] </br>"; 8. ?>   **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:**   1. <?php 2. **class** bike { 3. **function** model() { 4. $model\_name = "Royal Enfield"; 5. echo "Bike Model: " .$model\_name; 6. } 7. } 8. $obj = **new** bike(); 9. $obj -> model(); 10. ?>   **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 output 4. ?>   **Output:**   |  | | --- | | **PHP Variables** <="" p="">  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 datatype. * 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:   * $variablename=value;   Rules for declaring PHP variable:   * A variable must start with a dollar ($) sign, followed by the variable name. * It can only contain alpha-numeric 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. * 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 variable.  **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*   1. **<?php** 2. $str="hello string"; 3. $x=200; 4. $y=44.6; 5. echo "string is: $str **<br/>**"; 6. echo "integer is: $x **<br/>**"; 7. echo "float is: $y **<br/>**"; 8. **?>**   Output:  string is: hello string  integer is: 200  float is: 44.6 **PHP Variable: Sum of two variables** *File: variable2.php*   * **<?php** * $x=5; * $y=6; * $z=$x+$y; * echo $z; * **?>**   Output:  11 **PHP Variable: case sensitive** In PHP, variable names are case sensitive. So variable name "color" is different from Color, COLOR, COLor etc.  *File: variable3.php*   * **<?php** * $color="red"; * echo "My car is " . $color . "**<br>**"; * echo "My house is " . $COLOR . "**<br>**"; * echo "My boat is " . $coLOR . "**<br>**"; * **?>**   Output:  My car is red  Notice: Undefined variable: COLOR in C:\wamp\www\variable.php on line 4  My house is  Notice: Undefined variable: coLOR in C:\wamp\www\variable.php on line 5  My boat is **PHP Variable: Rules** PHP variables must start with letter or underscore only.  PHP variable can't be start with numbers and special symbols.  *File: variablevalid.php*   * **<?php** * $a="hello";//letter (valid) * $\_b="hello";//underscore (valid) * echo "$a **<br/>** $\_b"; * **?>**   Output:  hello  hello  *File: variableinvalid.php*   * **<?php** * $4c="hello";//number (invalid) * $\*d="hello";//special symbol (invalid) * echo "$4c **<br/>** $\*d"; * **?>**   Output:  Parse error: syntax error, unexpected '4' (T\_LNUMBER), expecting variable (T\_VARIABLE)  or '$' in C:\wamp\www\variableinvalid.php on line 2 **PHP: Loosely typed language** PHP is a loosely typed language, it means PHP automatically converts the variable to its correct data type.  [← Prev](https://www.javatpoint.com/php-echo-and-print-statements)[Next →](https://www.javatpoint.com/php-variable-scope) |   Youtube For Videos Join Our Youtube Channel: [Join Now](https://bit.ly/2FOeX6S) |

A constant is a name or an identifier for a simple value. A constant value cannot change during the execution of the script. By default, a constant is case-sensitive. By convention, constant identifiers are always uppercase. A constant name starts with a letter or underscore, followed by any number of letters, numbers, or underscores. If you have defined a constant, it can never be changed or undefined.

To define a constant you have to use define() function and to retrieve the value of a constant, you have to simply specifying its name. Unlike with variables, you do not need to have a constant with a $. You can also use the function constant() to read a constant's value if you wish to obtain the constant's name dynamically.

## **constant() function**

As indicated by the name, this function will return the value of the constant.

This is useful when you want to retrieve value of a constant, but you do not know its name, i.e. It is stored in a variable or returned by a function.

## **constant() example**

<?php

define("MINSIZE",50);

echo MINSIZE;

echo constant("MINSIZE");// same thing as the previous line

?>

Only scalar data (boolean, integer, float and string) can be contained in constants.

## **Differences between constants and variables are**

* There is no need to write a dollar sign ($) before a constant, where as in Variable one has to write a dollar sign.
* Constants cannot be defined by simple assignment, they may only be defined using the define() function.
* Constants may be defined and accessed anywhere without regard to variable scoping rules.
* Once the Constants have been set, may not be redefined or undefined.

## **Valid and invalid constant names**

// Valid constant names

define("ONE", "first thing");

define("TWO2", "second thing");

define("THREE\_3", "third thing");

define("\_\_THREE\_\_", "third value");

// Invalid constant names

define("2TWO", "second thing");

## **PHP Magic constants**

PHP provides a large number of predefined constants to any script which it runs.

There are five magical constants that change depending on where they are used. For example, the value of \_\_LINE\_\_ depends on the line that it's used on in your script. These special constants are case-insensitive and are as follows −

A few "magical" PHP constants are given below −

|  |  |
| --- | --- |
| **Sr.No** | **Name & Description** |
| 1 | **\_\_LINE\_\_**  The current line number of the file. |
| 2 | **\_\_FILE\_\_**  The full path and filename of the file. If used inside an include,the name of the included file is returned. Since PHP 4.0.2, **\_\_FILE\_\_**always contains an absolute path whereas in older versions it contained relative path under some circumstances. |
| 3 | **\_\_FUNCTION\_\_**  The function name. (Added in PHP 4.3.0) As of PHP 5 this constant returns the function name as it was declared (case-sensitive). In PHP 4 its value is always lowercased. |
| 4 | **\_\_CLASS\_\_**  The class name. (Added in PHP 4.3.0) As of PHP 5 this constant returns the class name as it was declared (case-sensitive). In PHP 4 its value is always lowercased. |
| 5 | **\_\_METHOD\_\_**  The class method name. (Added in PHP 5.0.0) The method name is returned as it was declared (case-sensitive). |

**What is Operator?** Simple answer can be given using expression *4 + 5 is equal to 9*. Here 4 and 5 are called operands and + is called operator. PHP language supports following 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](https://www.tutorialspoint.com/php/php_arithmatic_operators_examples.htm)

|  |  |  |
| --- | --- | --- |
| **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](https://www.tutorialspoint.com/php/php_comparison_operators_examples.htm)

|  |  |  |
| --- | --- | --- |
| **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](https://www.tutorialspoint.com/php/php_logical_operators_examples.htm)

|  |  |  |
| --- | --- | --- |
| **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](https://www.tutorialspoint.com/php/php_assignment_operators_examples.htm)

|  |  |  |
| --- | --- | --- |
| **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](https://www.tutorialspoint.com/php/php_conditional_operator_examples.htm)

|  |  |  |
| --- | --- | --- |
| **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 |

An array is a data structure that stores one or more similar type of values in a single value. 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](https://www.tutorialspoint.com/php/php_array_functions.htm)

## **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.

[Live Demo](http://tpcg.io/X30tPl)

<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>

This will produce the following result −

Value is 1

Value is 2

Value is 3

Value is 4

Value is 5

Value is one

Value is two

Value is three

Value is four

Value is five

## **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.

### **Example**

[Live Demo](http://tpcg.io/7qCRbI)

<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>

This will produce the following result −

Salary of mohammad is 2000

Salary of qadir is 1000

Salary of zara is 500

Salary of mohammad is high

Salary of qadir is medium

Salary of zara is low

## **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.

[Live Demo](http://tpcg.io/7Itd7K)

<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>

This will produce the following result −

Marks for mohammad in physics : 35

Marks for qadir in maths : 32

Marks for zara in chemistry : 39

# **PHP If Else**

PHP if else statement is used to test condition. There are various ways to use if statement in PHP.

* [if](https://www.javatpoint.com/php-if-else#if)
* [if-else](https://www.javatpoint.com/php-if-else#if-else)
* [if-else-if](https://www.javatpoint.com/php-if-else#if-else-if)
* [nested if](https://www.javatpoint.com/php-if-else#nested-if)

## **PHP If Statement**

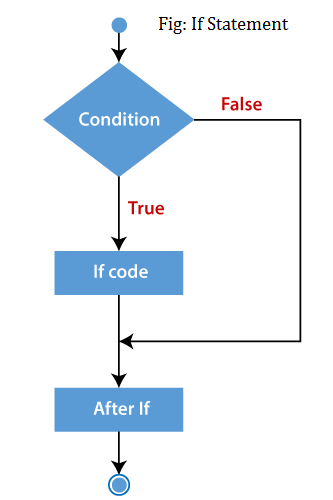
PHP if statement allows conditional execution of code. It is executed if condition is true.

If statement is used to executes the block of code exist inside the if statement only if the specified condition is true.

**Syntax**

1. **if**(condition){
2. //code to be executed
3. }

**Flowchart**



**Example**

1. <?php
2. $num=12;
3. **if**($num<100){
4. echo "$num is less than 100";
5. }
6. ?>

**Output:**

12 is less than 100

## **PHP If-else Statement**

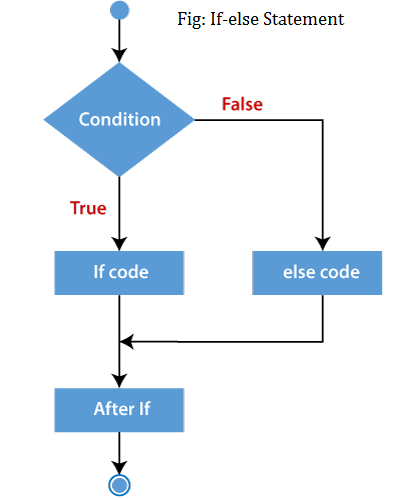
PHP if-else statement is executed whether condition is true or false.

If-else statement is slightly different from if statement. It executes one block of code if the specified condition is **true** and another block of code if the condition is **false**.

**Syntax**

1. **if**(condition){
2. //code to be executed if true
3. }**else**{
4. //code to be executed if false
5. }

**Flowchart**



**Example**

1. <?php
2. $num=12;
3. **if**($num%2==0){
4. echo "$num is even number";
5. }**else**{
6. echo "$num is odd number";
7. }
8. ?>

**Output:**

12 is even number

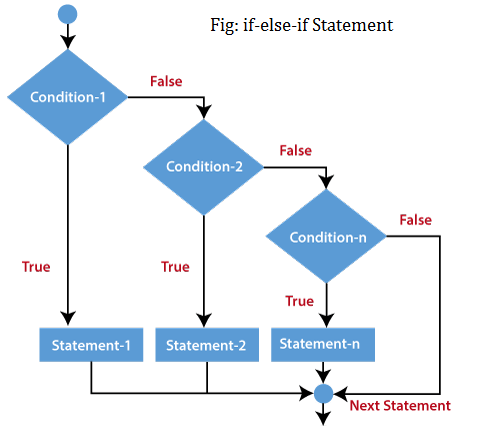
## **PHP If-else-if Statement**

The PHP if-else-if is a special statement used to combine multiple if?.else statements. So, we can check multiple conditions using this statement.

**Syntax**

1. **if** (condition1){
2. //code to be executed if condition1 is true
3. } **elseif** (condition2){
4. //code to be executed if condition2 is true
5. } **elseif** (condition3){
6. //code to be executed if condition3 is true
7. ....
8. }  **else**{
9. //code to be executed if all given conditions are false
10. }

**Flowchart**



**Example**

1. <?php
2. $marks=69;
3. **if** ($marks<33){
4. echo "fail";
5. }
6. **else** **if** ($marks>=34 && $marks<50) {
7. echo "D grade";
8. }
9. **else** **if** ($marks>=50 && $marks<65) {
10. echo "C grade";
11. }
12. **else** **if** ($marks>=65 && $marks<80) {
13. echo "B grade";
14. }
15. **else** **if** ($marks>=80 && $marks<90) {
16. echo "A grade";
17. }
18. **else** **if** ($marks>=90 && $marks<100) {
19. echo "A+ grade";
20. }
21. **else** {
22. echo "Invalid input";
23. }
24. ?>

**Output:**

B Grade

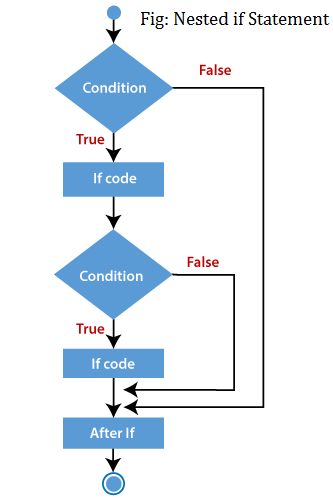
## **PHP nested if Statement**

The nested if statement contains the if block inside another if block. The inner if statement executes only when specified condition in outer if statement is **true**.

**Syntax**

1. **if** (condition) {
2. //code to be executed if condition is true
3. **if** (condition) {
4. //code to be executed if condition is true
5. }
6. }

**Flowchart**



**Example**

1. <?php
2. $age = 23;
3. $nationality = "Indian";
4. //applying conditions on nationality and age
5. **if** ($nationality == "Indian")
6. {
7. **if** ($age >= 18) {
8. echo "Eligible to give vote";
9. }
10. **else** {
11. echo "Not eligible to give vote";
12. }
13. }
14. ?>

**Output:**

Eligible to give vote

**PHP Switch Example**

1. <?php
2. $a = 34; $b = 56; $c = 45;
3. **if** ($a < $b) {
4. **if** ($a < $c) {
5. echo "$a is smaller than $b and $c";
6. }
7. }
8. ?>

**Output:**

34 is smaller than 56 and 45

# PHP Loops

In this tutorial you will learn how to repeat a series of actions using loops in PHP.

## Different Types of Loops in PHP

Loops are used to execute the same block of code again and again, as long as a certain condition is met. The basic idea behind a loop is to automate the repetitive tasks within a program to save the time and effort. PHP supports four different types of loops.

* **while** — loops through a block of code as long as the condition specified evaluates to true.
* **do…while** — the block of code executed once and then condition is evaluated. If the condition is true the statement is repeated as long as the specified condition is true.
* **for** — loops through a block of code until the counter reaches a specified number.
* **foreach** — loops through a block of code for each element in an array.

You will also learn how to loop through the values of array using [foreach()](https://www.tutorialrepublic.com/php-tutorial/php-loops.php#foreach-loop) loop at the end of this chapter. The foreach() loop work specifically with arrays.

## PHP while Loop

The while statement will loops through a block of code as long as the condition specified in the while statement evaluate to true.

while(condition){  
    // Code to be executed  
}

The example below define a loop that starts with $i=1. The loop will continue to run as long as $i is less than or equal to 3. The $i will increase by 1 each time the loop runs:

#### Example

[**Run this code »**](https://www.tutorialrepublic.com/codelab.php?topic=php&file=while-loop)

<?php

$i=1;

while($i<=3){

$i++;

echo"The number is ".$i."<br>";

}

?>

## PHP do…while Loop

The do-while loop is a variant of while loop, which evaluates the condition at the end of each loop iteration. With a do-while loop the block of code executed once, and then the condition is evaluated, if the condition is true, the statement is repeated as long as the specified condition evaluated to is true.

do{  
    // Code to be executed  
}  
while(condition);

The following example define a loop that starts with $i=1. It will then increase $i with 1, and print the output. Then the condition is evaluated, and the loop will continue to run as long as $i is less than, or equal to 3.

#### Example

[**Run this code »**](https://www.tutorialrepublic.com/codelab.php?topic=php&file=do-while-loop)

<?php

$i=1;

do{

$i++;

echo"The number is ".$i."<br>";

}

while($i<=3);

?>

## Difference Between while and do…while Loop

The while loop differs from the do-whileloop in one important way — with a while loop, the condition to be evaluated is tested at the beginning of each loop iteration, so if the conditional expression evaluates to false, the loop will never be executed.

With a do-while loop, on the other hand, the loop will always be executed once, even if the conditional expression is false, because the condition is evaluated at the end of the loop iteration rather than the beginning.

## PHP 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(initialization; condition; increment){  
    // Code to be executed  
}

The parameters of for loop have following meanings:

* initialization — it is used to initialize the counter variables, and evaluated once unconditionally before the first execution of the body of the loop.
* condition — in the beginning of each iteration, condition is evaluated. If it evaluates to true, the loop continues and the nested statements are executed. If it evaluates to false, the execution of the loop ends.
* increment — it updates the loop counter with a new value. It is evaluate at the end of each iteration.

The example below defines a loop that starts with $i=1. The loop will continued until $i is less than, or equal to 3. The variable $i will increase by 1 each time the loop runs:

#### Example

[**Run this code »**](https://www.tutorialrepublic.com/codelab.php?topic=php&file=for-loop)

<?php

for($i=1;$i<=3;$i++){

echo"The number is ".$i."<br>";

}

?>

## PHP foreach Loop

The foreach loop is used to iterate over arrays.

foreach($array as $value){  
    // Code to be executed  
}

The following example demonstrates a loop that will print the values of the given array:

#### Example

[**Run this code »**](https://www.tutorialrepublic.com/codelab.php?topic=php&file=foreach-loop)

<?php

$colors=array("Red","Green","Blue");

// Loop through colors array

foreach($colorsas$value){

echo$value."<br>";

}

?>

There is one more syntax of foreach loop, which is extension of the first.

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

#### Example

[**Run this code »**](https://www.tutorialrepublic.com/codelab.php?topic=php&file=foreach-loop-extended)

<?php

$superhero=array(

"name"=>"Peter Parker",

"email"=>"peterparker@mail.com",

"age"=>18

);

// Loop through superhero array

foreach($superheroas$key=>$value){

echo$key." : ".$value."<br>";

}

?>

# **Special loop keywords**

There are two special keywords you can use with loops, and they are "break" and "continue". We already used break previously when we looked at case switching - it is used there to exit a switch/case block, and it has the same effect with loops. When used inside loops in order to manipulate the loop behaviour: break causes PHP to exit the loop and carry on immediately after it, and continue causes PHP to skip the rest of the current loop iteration and go onto the next.

For example:

<?php

for ($i = 1; $i<10; $i = $i + 1) {

if ($i == 3) continue;

if ($i == 7) break;

print"Number $i\n";

}

?>

That is a modified version of our original for loop script. This time the output is this:

Number 1  
Number 2  
Number 4  
Number 5  
Number 6

Note that number 3 is missed out entirely, and the script exits after number 6. The reason for this is because of the two if statements - if the current number is 3, "continue" is used to skip the rest of that iteration and go on to number 4. Also, if the number is 7, "break" is used to exit the loop altogether.