PHP - Introduction

PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages.

## What is PHP?

1. PHP is an acronym for "PHP: Hypertext Preprocessor"
2. PHP is a widely-used, open source scripting language
3. PHP scripts are executed on the server
4. PHP is free to download and use

## What is a PHP File?

1. PHP files can contain text, HTML, CSS, JavaScript, and PHP code
2. PHP code is executed on the server, and the result is returned to the browser as plain HTML
3. PHP files have extension ".php"

## What Can PHP Do?

1. PHP can generate dynamic page content
2. PHP can create, open, read, write, delete, and close files on the server
3. PHP can collect form data
4. PHP can send and receive cookies
5. PHP can add, delete, modify data in your database
6. PHP can be used to control user-access
7. PHP can encrypt data

With PHP you are not limited to output HTML. You can output images, PDF files, and even Flash movies. You can also output any text, such as XHTML and XML.

## Why PHP?

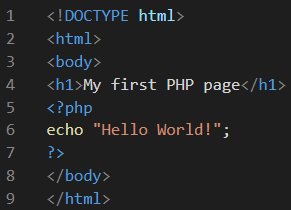
1. PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
2. PHP is compatible with almost all servers used today (Apache, IIS, etc.)
3. PHP supports a wide range of databases
4. PHP is easy to learn and runs efficiently on the server side

# PHP Syntax

A PHP script is executed on the server, and the plain HTML result is sent back to the browser.

1. A PHP script can be placed anywhere in the document.
2. A PHP script starts with <?php and ends with ?>
3. The default file extension for PHP files is ".php".

Example:



Note:

1. PHP statements end with a semicolon (;).
2. In PHP, keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are not case-sensitive.

In PHP, we can use either double slash (//) or hash (#) for single commented lines and /\* and \*/ for the multiple commented lines.

# PHP Variables

In PHP, a variable starts with the $ sign, followed by the name of the variable.

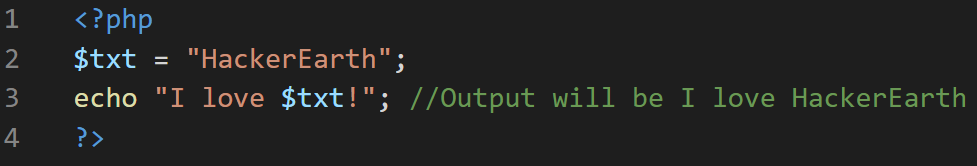
A variable can have a short name (like x and y) or a more descriptive name (age, carname, total\_volume).

Rules for PHP variables:

1. A variable starts with the $ sign, followed by the name of the variable
2. A variable name must start with a letter or the underscore character
3. A variable name cannot start with a number
4. A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_ )
5. Variable names are case-sensitive ($age and $AGE are two different variables)

The PHP echo statement is often used to output data to the screen.

Example:



## PHP is a Loosely Typed Language

1. PHP automatically associates a data type to the variable, depending on its value. Since the data types are not set in a strict sense, you can do things like adding a string to an integer without causing an error.
2. In PHP 7, type declarations were added. This gives an option to specify the data type expected when declaring a function, and by enabling the strict requirement, it will throw a "Fatal Error" on a type mismatch.

## PHP Variables Scope

In PHP, variables can be declared anywhere in the script.

The scope of a variable is the part of the script where the variable can be referenced/used.

PHP has three different variable scopes:

1. local
2. global
3. static

## Global and Local Scope

1. A variable declared outside a function has a GLOBAL SCOPE and can only be accessed outside a function
2. A variable declared within a function has a LOCAL SCOPE and can only be accessed within that function.
3. The global keyword is used to access a global variable from within a function.

## PHP Data Types

Variables can store data of different types, and different data types can do different things.

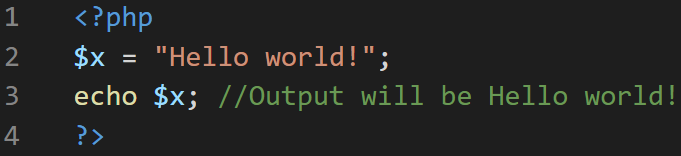
PHP supports the following data types:

1. String
2. Integer
3. Float (floating point numbers - also called double)
4. Boolean
5. Array
6. Object
7. NULL
8. Resource

## PHP String

A string is a sequence of characters, like "Hello world!".

A string can be any text inside quotes. You can use single or double quotes



## PHP Integer

An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.

Rules for integers:

* An integer must have at least one digit
* An integer must not have a decimal point
* An integer can be either positive or negative
* Integers can be specified in: decimal (base 10), hexadecimal (base 16), octal (base 8), or binary (base 2) notation

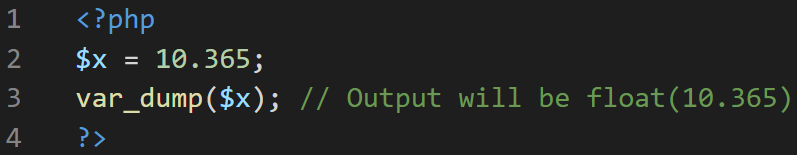
Example:



## PHP Float

A float (floating point number) is a number with a decimal point or a number in exponential form.

Example:



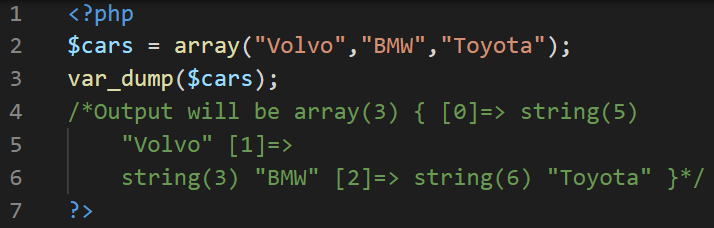
## PHP Boolean

A Boolean represents two possible states: TRUE or FALSE.

## PHP Array

An array stores multiple values in one single variable.

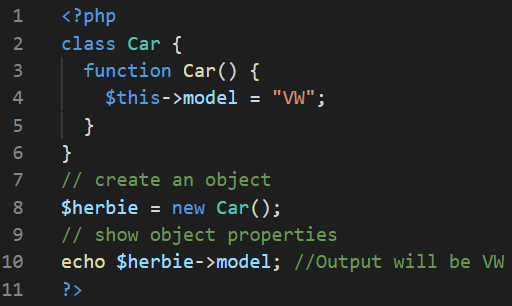
Example:



## PHP Object

An object is a data type which stores data and information on how to process that data. In PHP, an object must be explicitly declared. First we must declare a class of object.

Example:

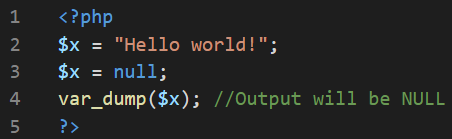


## PHP NULL Value

Null is a special data type which can have only one value: NULL.

A variable of data type NULL is a variable that has no value assigned to it.

Example:



# PHP String methods

1. The PHP strlen() function returns the length of a string.
2. The PHP str\_word\_count() function counts the number of words in a string.
3. The PHP strrev() function reverses a string.
4. The PHP strpos() function searches for a specific text within a string. If a match is found, the function returns the character position of the first match. If no match is found, it will return FALSE.
5. The PHP str\_replace() function replaces some characters with some other characters in a string.

# PHP Math

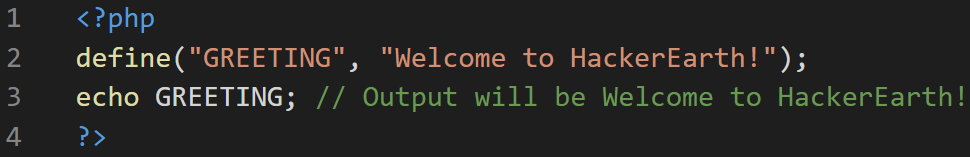
PHP has a set of math functions that allows you to perform mathematical tasks on numbers.

1. The pi() function returns the value of PI
2. The min() and max() functions can be used to find the lowest or highest value in a list of arguments
3. The abs() function returns the absolute (positive) value of a number
4. The sqrt() function returns the square root of a number
5. The round() function rounds a floating-point number to its nearest integer
6. The rand() function generates a random number

# PHP Constants

1. A constant is an identifier (name) for a simple value. The value cannot be changed during the script.
2. A valid constant name starts with a letter or underscore (no $ sign before the constant name).
3. Note: Unlike variables, constants are automatically global across the entire script.
4. The define() function is used to create the constant values.

Example:



## PHP Operators

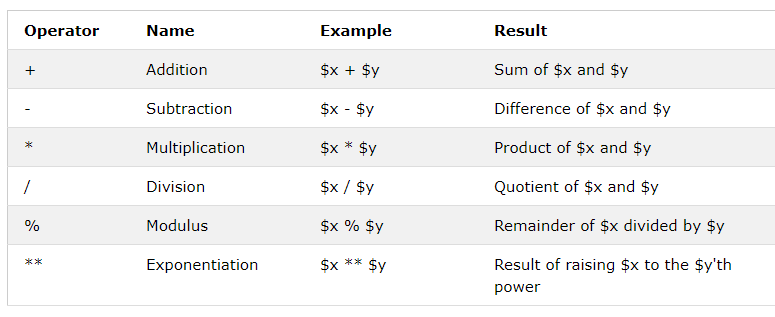
Operators are used to perform operations on variables and values.

PHP divides the operators in the following groups:

1. Arithmetic operators
2. Assignment operators
3. Comparison operators
4. Increment/Decrement operators
5. Logical operators
6. String operators
7. Array operators
8. Conditional assignment operators

## PHP Arithmetic Operators

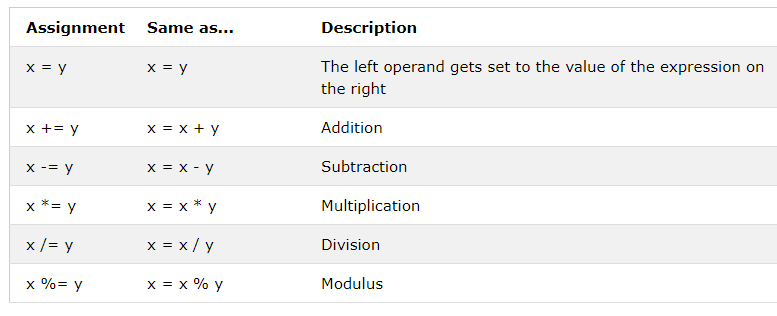
The PHP arithmetic operators are used with numeric values to perform common arithmetical operations, such as addition, subtraction, multiplication etc.



## PHP Assignment Operators

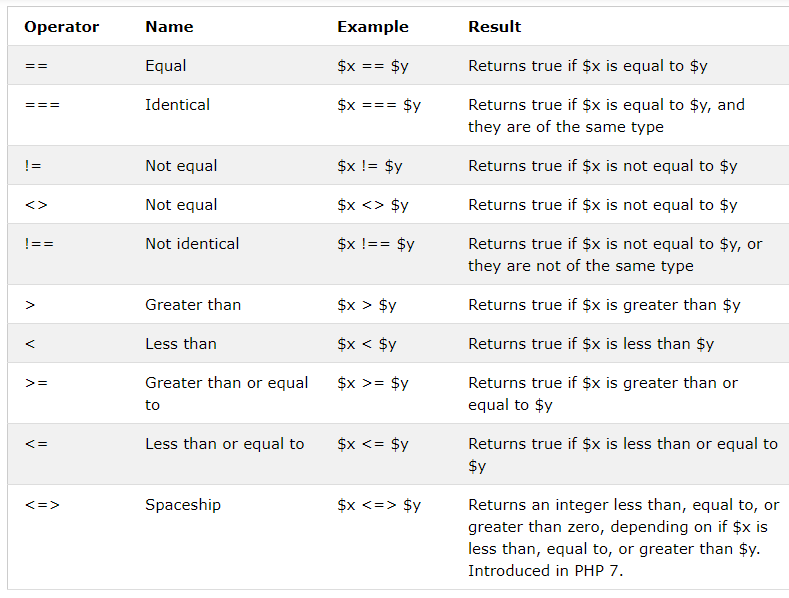
The PHP assignment operators are used with numeric values to write a value to a variable.

The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the assignment expression on the right.



## PHP Comparison Operators

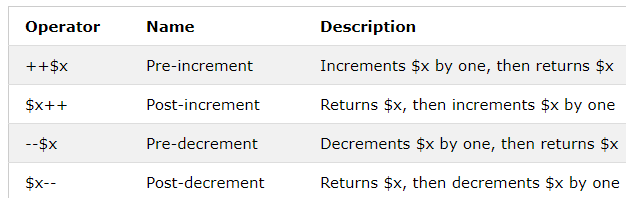
The PHP comparison operators are used to compare two values (number or string):



## PHP Increment / Decrement Operators

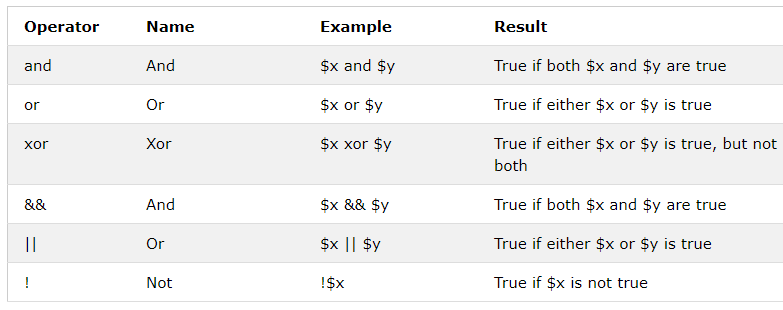
The PHP increment operators are used to increment a variable's value.

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



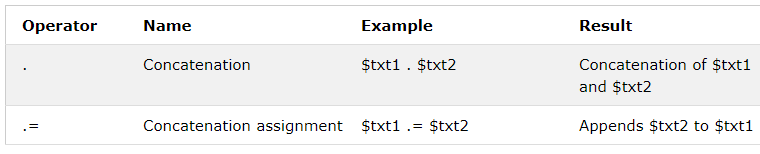
## PHP Logical Operators

The PHP logical operators are used to combine conditional statements.



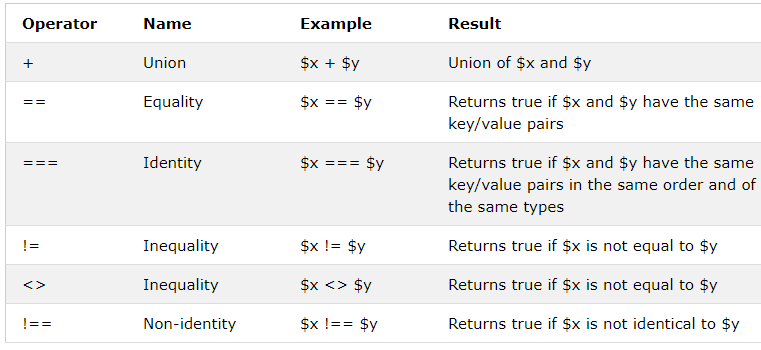
## PHP String Operators

## PHP has two operators that are specially designed for strings.



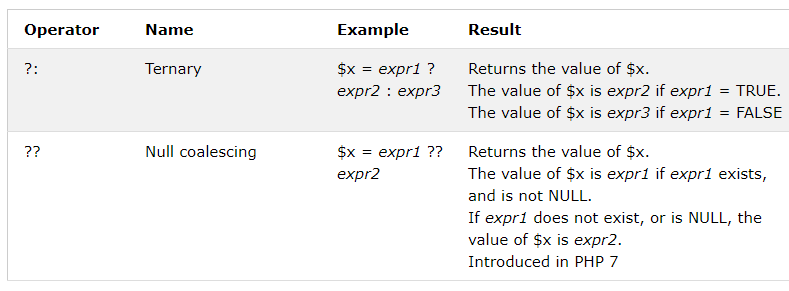
## PHP Array Operators

The PHP array operators are used to compare arrays.



## PHP Conditional Assignment Operators

The PHP conditional assignment operators are used to set a value depending on conditions:



## PHP Conditional Statements

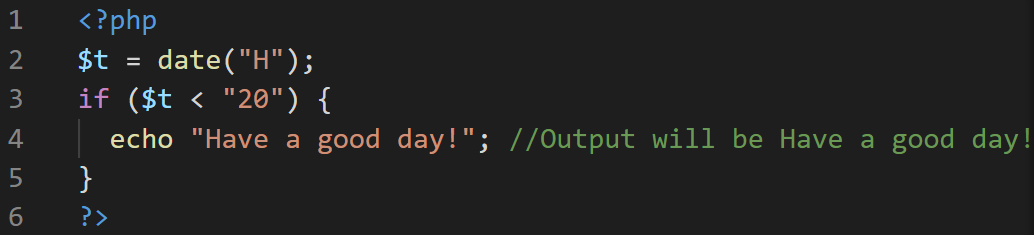
Very often when you write code, you want to perform different actions for different conditions. You can use conditional statements in your code to do this.

In PHP we have the following conditional statements:

1. if statement - executes some code if one condition is true
2. if...else statement - executes some code if a condition is true and another code if that condition is false
3. if...elseif...else statement - executes different codes for more than two conditions
4. switch statement - selects one of many blocks of code to be executed

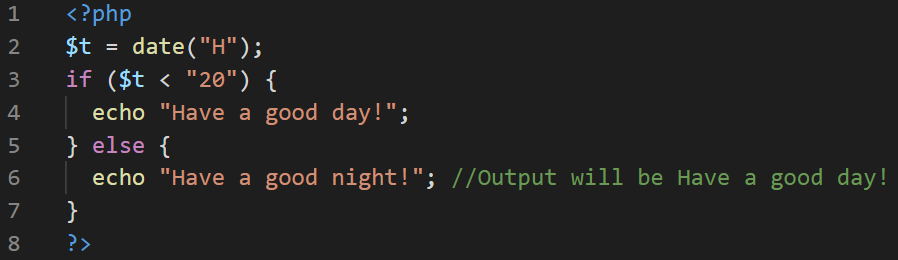
## PHP - The if Statement

The if statement executes some code if one condition is true.



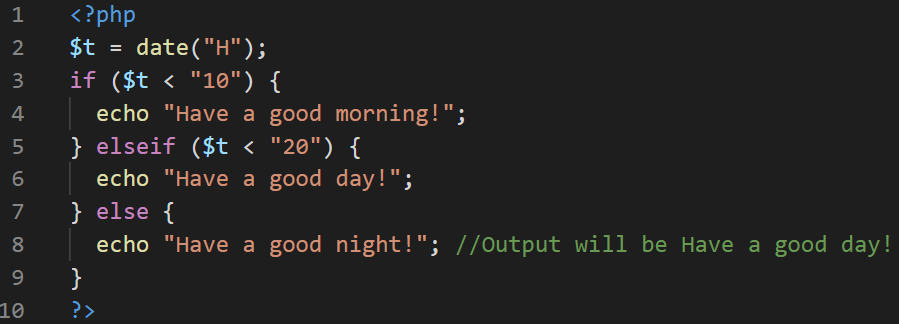
## PHP - The if...else Statement

The if...else statement executes some code if a condition is true and another code if that condition is false.



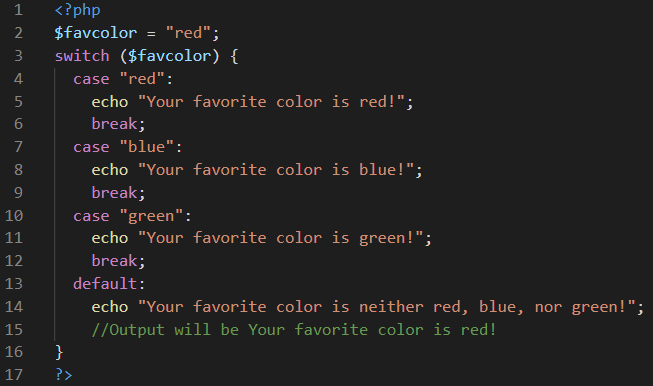
## PHP - The if...elseif...else Statement

The if...elseif...else statement executes different codes for more than two conditions.



## The PHP switch Statement

Use the switch statement to select one of many blocks of code to be executed.



## PHP Loops

Often when you write code, you want the same block of code to run over and over again a certain number of times. So, instead of adding several almost equal code-lines in a script, we can use loops.

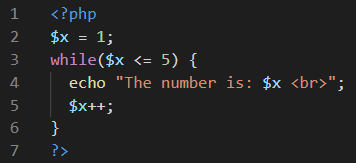
Loops are used to execute the same block of code again and again, as long as a certain condition is true.

In PHP, we have the following loop types:

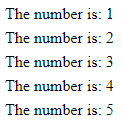
* while - loops through a block of code as long as the specified condition is true
* do...while - loops through a block of code once, and then repeats the loop as long as the specified condition is true
* for - loops through a block of code a specified number of times
* foreach - loops through a block of code for each element in an array

## The PHP while Loop

The while loop executes a block of code as long as the specified condition is true.



The output of the above code is as follows:

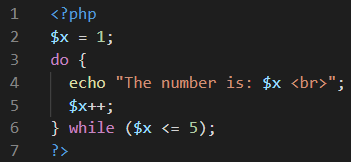


### Example Explained

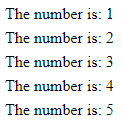
1. $x = 1; - Initialize the loop counter ($x), and set the start value to 1
2. $x <= 5 - Continue the loop as long as $x is less than or equal to 5
3. $x++; - Increase the loop counter value by 1 for each iteration

## The PHP do...while Loop

The do...while loop will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true.

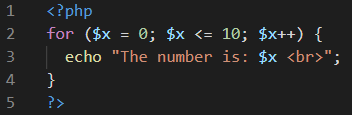


The output of the above code is as follows:

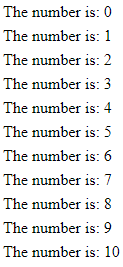


## The PHP for Loop

The for loop is used when you know in advance how many times the script should run.



The output of the above code is as follows:

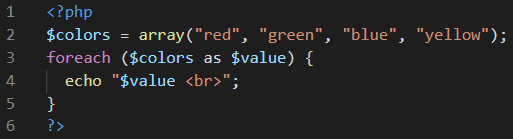


### Example Explained

* $x = 0; - Initialize the loop counter ($x), and set the start value to 0
* $x <= 10; - Continue the loop as long as $x is less than or equal to 10
* $x++ - Increase the loop counter value by 1 for each iteration

## The PHP foreach Loop

The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.



The output of the above code is as follows:



## PHP Break

You have already seen the break statement used in an earlier chapter of this tutorial. It was used to "jump out" of a switch statement.

The break statement can also be used to jump out of a loop.

## PHP Continue

The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

# PHP Functions

## PHP User Defined Functions

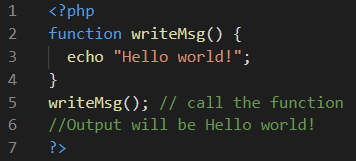
Besides the built-in PHP functions, it is possible to create your own functions.

1. A function is a block of statements that can be used repeatedly in a program.
2. A function will not execute automatically when a page loads.
3. A function will be executed by a call to the function.

## Create a User Defined Function in PHP

A user-defined function declaration starts with the word function

Example:

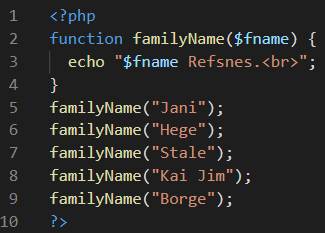


## PHP Function Arguments

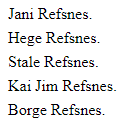
Information can be passed to functions through arguments. An argument is just like a variable.

Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

The following example has a function with one argument ($fname). When the familyName() function is called, we also pass along a name (e.g. Jani), and the name is used inside the function, which outputs several different first names, but an equal last name:

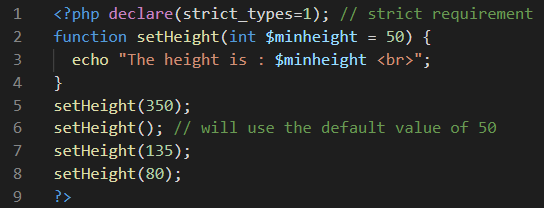


The output for the above code is as follows:

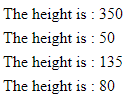


## PHP Default Argument Value

The following example shows how to use a default parameter. If we call the function setHeight() without arguments it takes the default value as argument:

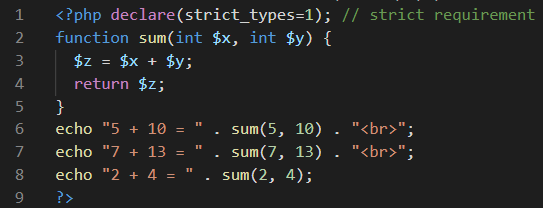


The output of the above code will be as follows:



## PHP Functions - Returning values

To let a function return a value, use the return statement:



The output of the above code is as follows:

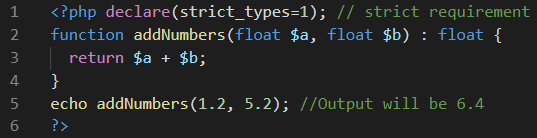


## PHP Return Type Declarations

PHP 7 also supports Type Declarations for the return statement. Like with the type declaration for function arguments, by enabling the strict requirement, it will throw a "Fatal Error" on a type mismatch.

To declare a type for the function return, add a colon ( : ) and the type right before the opening curly ( { )bracket when declaring the function.

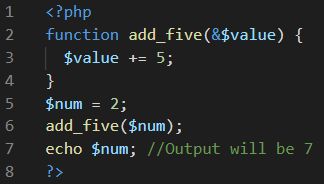
In the following example we specify the return type for the function:



## Passing Arguments by Reference

In PHP, arguments are usually passed by value, which means that a copy of the value is used in the function and the variable that was passed into the function cannot be changed.

When a function argument is passed by reference, changes to the argument also change the variable that was passed in. To turn a function argument into a reference, the & operator is used:



# PHP Arrays

An array is a special variable, which can hold more than one value at a time.

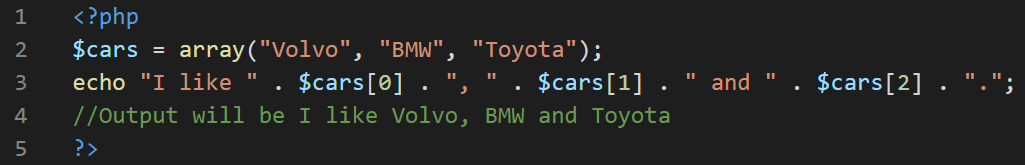
In PHP, the array() function is used to create an array.

In PHP, there are three types of arrays:

1. Indexed arrays - Arrays with a numeric index
2. Associative arrays - Arrays with named keys
3. Multidimensional arrays - Arrays containing one or more arrays

## PHP Indexed Arrays

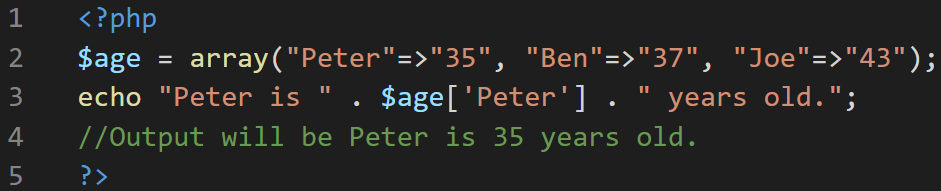
The following example creates an indexed array named $cars, assigns three elements to it, and then prints a text containing the array values:



## PHP Associative Arrays

Associative arrays are arrays that use named keys that you assign to them.

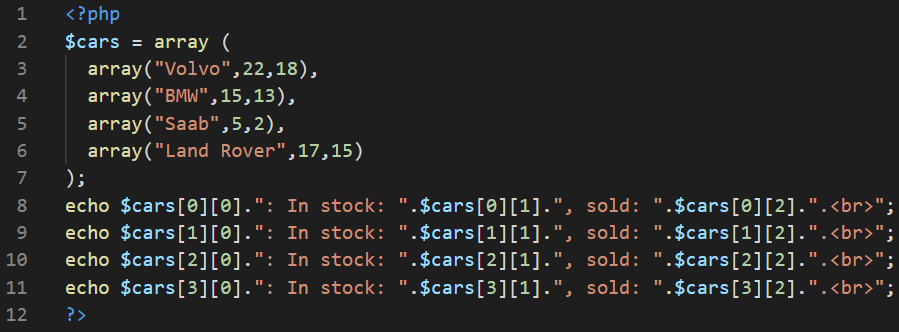
Example:



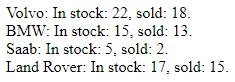
## PHP - Multidimensional Arrays

A multidimensional array is an array containing one or more arrays.

Example:



The output of the above code is as follows:



## PHP - Sort Functions For Arrays

1. sort() - sort arrays in ascending order
2. rsort() - sort arrays in descending order
3. asort() - sort associative arrays in ascending order, according to the value
4. ksort() - sort associative arrays in ascending order, according to the key
5. arsort() - sort associative arrays in descending order, according to the value
6. krsort() - sort associative arrays in descending order, according to the key.

## PHP Global Variables - Superglobals

Some predefined variables in PHP are "superglobals", which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.

The PHP superglobal variables are:

1. $GLOBALS
2. $\_SERVER
3. $\_REQUEST
4. $\_POST
5. $\_GET
6. $\_FILES
7. $\_ENV
8. $\_COOKIE
9. $\_SESSION

## PHP $GLOBALS

$GLOBALS is a PHP super global variable which is used to access global variables from anywhere in the PHP script (also from within functions or methods).

## PHP $\_SERVER

$\_SERVER is a PHP super global variable which holds information about headers, paths, and script locations.

## PHP $\_REQUEST

PHP $\_REQUEST is a PHP super global variable which is used to collect data after submitting an HTML form.

## PHP $\_POST

PHP $\_POST is a PHP super global variable which is used to collect form data after submitting an HTML form with method="post". $\_POST is also widely used to pass variables.

## PHP $\_GET

PHP $\_GET is a PHP super global variable which is used to collect form data after submitting an HTML form with method="get".

# PHP Regular Expressions

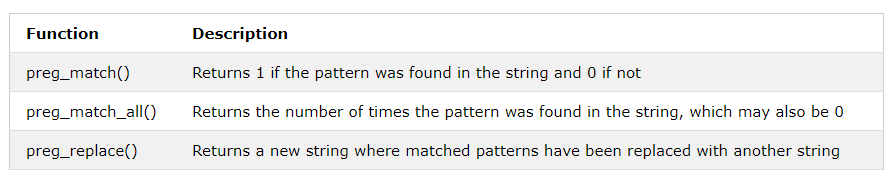
A regular expression is a sequence of characters that forms a search pattern. When you search for data in a text, you can use this search pattern to describe what you are searching for.

A regular expression can be a single character, or a more complicated pattern.

Regular expressions can be used to perform all types of text search and text replace operations.

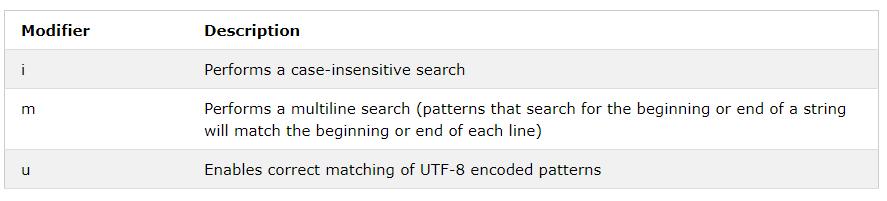
## Regular Expression Functions

PHP provides a variety of functions that allow you to use regular expressions. The preg\_match(), preg\_match\_all() and preg\_replace() functions are some of the most commonly used ones:



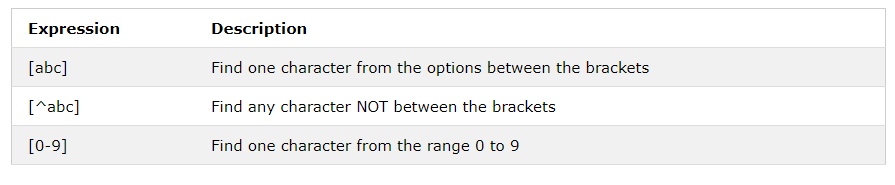
## Regular Expression Modifiers

Modifiers can change how a search is performed.



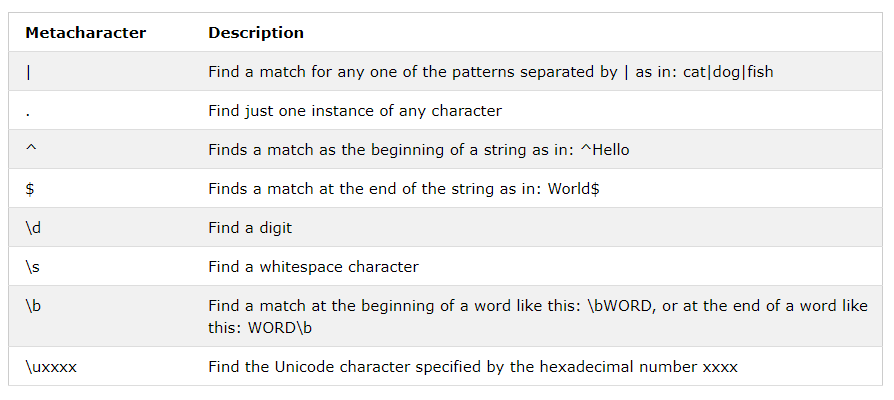
## Regular Expression Patterns

Brackets are used to find a range of characters:



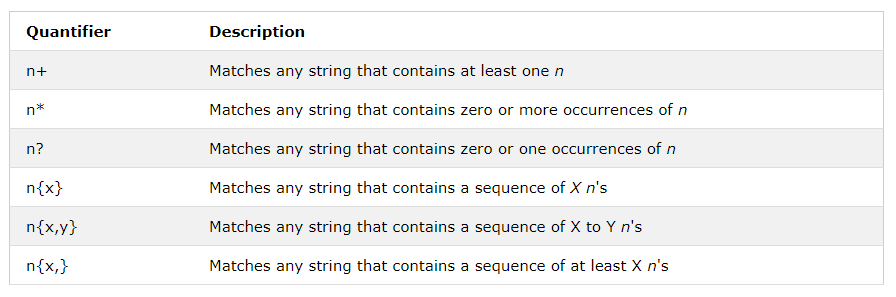
## Metacharacters

Metacharacters are characters with a special meaning:



## Quantifiers

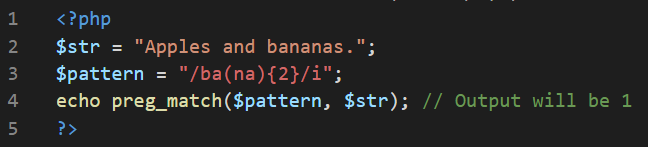
Quantifiers define quantities:



## Grouping

You can use parentheses ( ) to apply quantifiers to entire patterns. They also can be used to select parts of the pattern to be used as a match.

Example:



# 

# PHP Forms

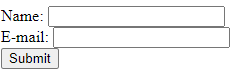
# PHP Form Handling

The PHP superglobals $\_GET and $\_POST are used to collect form-data.

Consider the following HTML code on forms:



The output of the above code is as follows:



When the user fills out the form above and clicks the submit button, the form data is sent for processing to a PHP file named "welcome.php". The form data is sent with the HTTP POST method.

## PHP Form Validation

## The Form Element

The HTML code of the form looks like this:



## PHP Form Security

The $\_SERVER["PHP\_SELF"] variable can be used by hackers!

If PHP\_SELF is used in your page then a user can enter a slash (/) and then some Cross Site Scripting (XSS) commands to execute.

## How To Avoid $\_SERVER["PHP\_SELF"] Exploits?

$\_SERVER["PHP\_SELF"] exploits can be avoided by using the htmlspecialchars() function.

The form code should look like this:



## Validate Form Data With PHP

The first thing we will do is to pass all variables through PHP's htmlspecialchars() function.

When we use the htmlspecialchars() function; then if a user tries to submit the following in a text field:

<script>location.href('http://www.hacked.com')</script>

- this would not be executed, because it would be saved as HTML escaped code, like this:

&lt;script&gt;location.href('http://www.hacked.com')&lt;/script&gt;

The code is now safe to be displayed on a page or inside an e-mail.

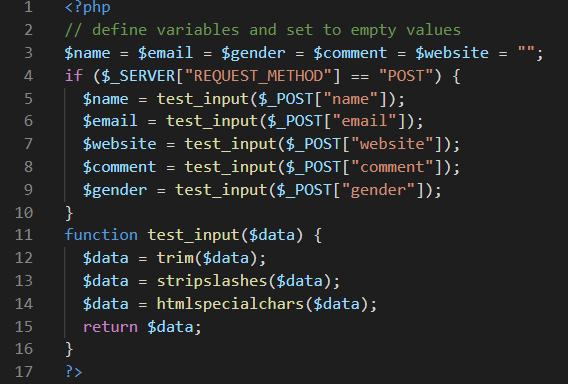
We will also do two more things when the user submits the form:

1. Strip unnecessary characters (extra space, tab, newline) from the user input data (with the PHP trim() function)
2. Remove backslashes (\) from the user input data (with the PHP stripslashes() function)

The next step is to create a function that will do all the checking for us (which is much more convenient than writing the same code over and over again).

We will name the function test\_input().

Now, we can check each $\_POST variable with the test\_input() function, and the script looks like this:



# PHP Date and Time

The PHP date() function is used to format a date and/or a time.

The PHP date() function formats a timestamp to a more readable date and time.

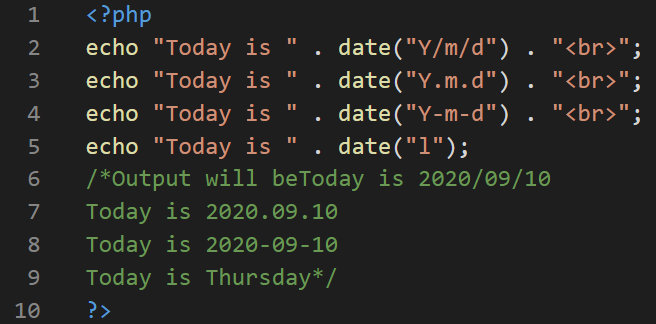
## Get a Date

The required *format* parameter of the date() function specifies how to format the date (or time).

Here are some characters that are commonly used for dates:

1. d - Represents the day of the month (01 to 31)
2. m - Represents a month (01 to 12)
3. Y - Represents a year (in four digits)
4. l (lowercase 'L') - Represents the day of the week

Other characters, like"/", ".", or "-" can also be inserted between the characters to add additional formatting.

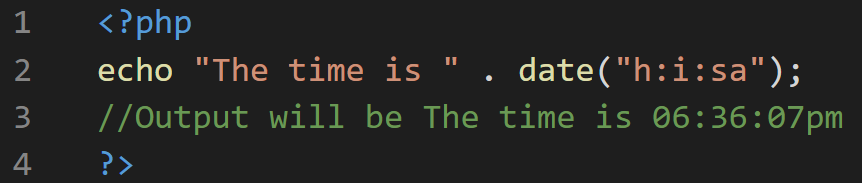


## Get a Time

Here are some characters that are commonly used for times:

1. H - 24-hour format of an hour (00 to 23)
2. h - 12-hour format of an hour with leading zeros (01 to 12)
3. i - Minutes with leading zeros (00 to 59)
4. s - Seconds with leading zeros (00 to 59)
5. a - Lowercase Ante meridiem and Post meridiem (am or pm)

The example below outputs the current time in the specified format:



## Get Your Time Zone

If the time you got back from the code is not correct, it's probably because your server is in another country or set up for a different timezone.

So, if you need the time to be correct according to a specific location, you can set the timezone you want to use.

## Create a Date With mktime()

The optional *timestamp* parameter in the date() function specifies a timestamp. If omitted, the current date and time will be used (as in the examples above).

The PHP mktime() function returns the Unix timestamp for a date. The Unix timestamp contains the number of seconds between the Unix Epoch (January 1 1970 00:00:00 GMT) and the time specified.

## Create a Date From a String With strtotime()

The PHP strtotime() function is used to convert a human readable date string into a Unix timestamp (the number of seconds since January 1 1970 00:00:00 GMT).

## PHP include and require Statements

It is possible to insert the content of one PHP file into another PHP file (before the server executes it), with the include or require statement.

The include and require statements are identical, except upon failure:

1. require will produce a fatal error (E\_COMPILE\_ERROR) and stop the script
2. include will only produce a warning (E\_WARNING) and the script will continue

So, if you want the execution to go on and show users the output, even if the include file is missing, use the include statement. Otherwise, in case of FrameWork, CMS, or a complex PHP application coding, always use the require statement to include a key file to the flow of execution. This will help avoid compromising your application's security and integrity, just in-case one key file is accidentally missing.

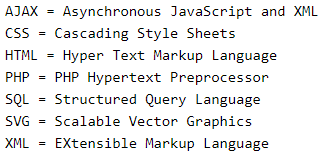
Including files saves a lot of work. This means that you can create a standard header, footer, or menu file for all your web pages. Then, when the header needs to be updated, you can only update the header include file.

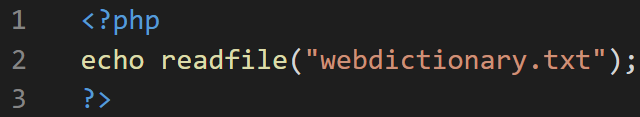
# PHP File Handling

## PHP readfile() Function

The readfile() function reads a file and writes it to the output buffer.

Assume we have a text file called "webdictionary.txt", stored on the server, that looks like this:





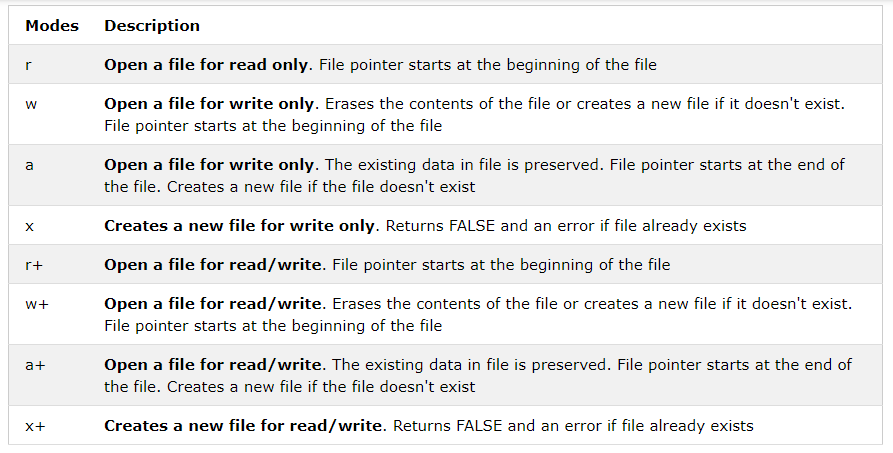
The output of the above code is as follows:



## PHP Open File - fopen()

A better method to open files is with the fopen() function. This function gives you more options than the readfile() function.

The file may be opened in one of the following modes:



## PHP Read File - fread()

The fread() function reads from an open file.

The first parameter of fread() contains the name of the file to read from and the second parameter specifies the maximum number of bytes to read.

## PHP Close File - fclose()

The fclose() function is used to close an open file.

## PHP Read Single Line - fgets()

The fgets() function is used to read a single line from a file.

## PHP Check End-Of-File - feof()

The feof() function checks if the "end-of-file" (EOF) has been reached.

The feof() function is useful for looping through data of unknown length.

## PHP Read Single Character - fgetc()

The fgetc() function is used to read a single character from a file.

## PHP Create File - fopen()

The fopen() function is also used to create a file. Maybe a little confusing, but in PHP, a file is created using the same function used to open files.

If you use fopen() on a file that does not exist, it will create it, given that the file is opened for writing (w) or appending (a).

## PHP File Permissions

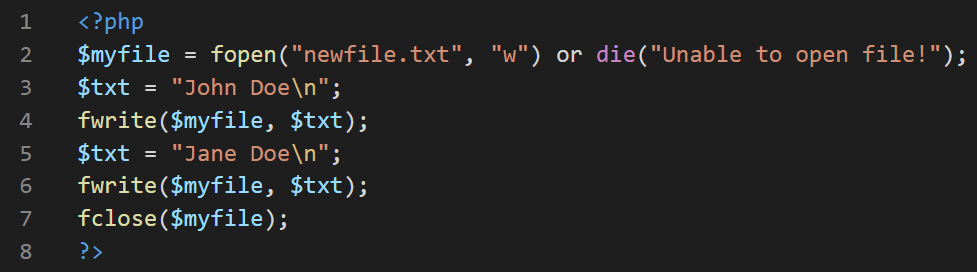
If you are having errors when trying to get this code to run, check that you have granted your PHP file access to write information to the hard drive.

## PHP Write to File - fwrite()

The fwrite() function is used to write to a file.

The first parameter of fwrite() contains the name of the file to write to and the second parameter is the string to be written.

The example below writes a couple of names into a new file called "newfile.txt":



Notice that we wrote to the file "newfile.txt" twice. Each time we wrote to the file we sent the string $txt that first contained "John Doe" and second contained "Jane Doe". After we finished writing, we closed the file using the fclose() function.

If we open the "newfile.txt" file it would look like this:



# PHP File Upload

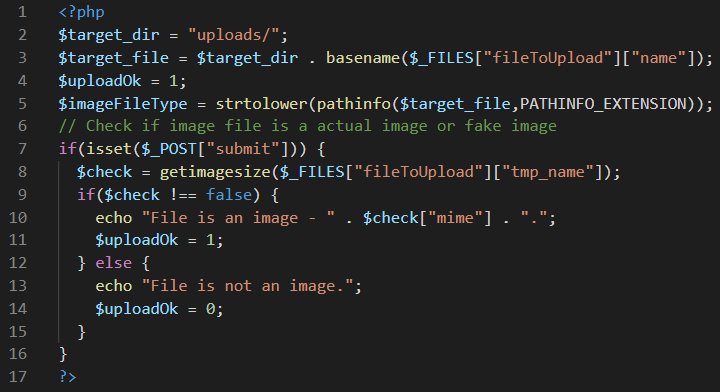
## Configure The "php.ini" File

First, ensure that PHP is configured to allow file uploads.

In your "php.ini" file, search for the file\_uploads directive, and set it to On.

## Create The Upload File PHP Script

The "upload.php" file contains the code for uploading a file:



PHP script explained:

1. $target\_dir = "uploads/" - specifies the directory where the file is going to be placed
2. $target\_file specifies the path of the file to be uploaded
3. $uploadOk=1 is not used yet (will be used later)
4. $imageFileType holds the file extension of the file (in lower case)
5. Next, check if the image file is an actual image or a fake image

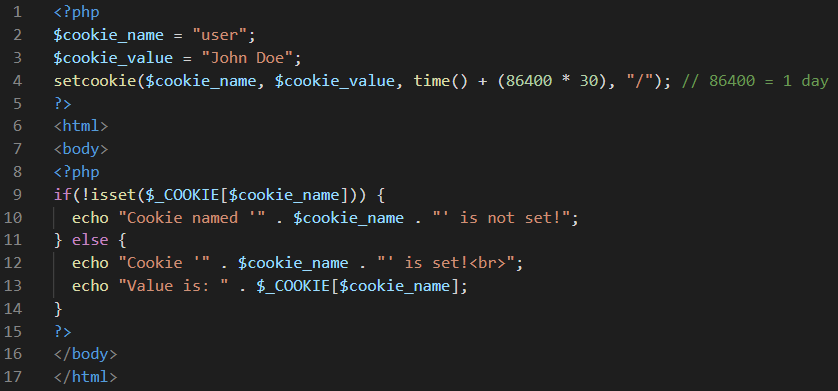
# PHP Cookies

## PHP Create/Retrieve a Cookie

A cookie is created with the setcookie() function.

The following example creates a cookie named "user" with the value "John Doe". The cookie will expire after 30 days (86400 \* 30). The "/" means that the cookie is available in entire website (otherwise, select the directory you prefer).

We then retrieve the value of the cookie "user" (using the global variable $\_COOKIE). We also use the isset() function to find out if the cookie is set:



Cookie named 'user' is not set! will be displayed on executing this code

## Modify a Cookie Value

To modify a cookie, just set (again) the cookie using the setcookie() function

## Delete a Cookie

To delete a cookie, use the setcookie() function with an expiration date in the past.

# PHP Sessions

When you work with an application, you open it, do some changes, and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.

Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc). By default, session variables last until the user closes the browser.

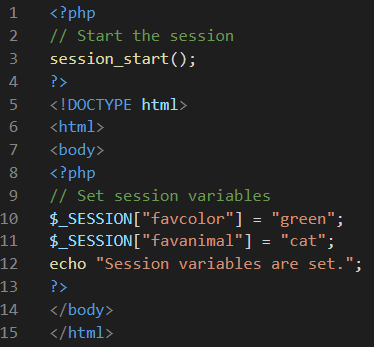
So; Session variables hold information about one single user, and are available to all pages in one application.

## Start a PHP Session

A session is started with the session\_start() function.

Session variables are set with the PHP global variable: $\_SESSION.

Now, let's create a new page called "demo\_session1.php". In this page, we start a new PHP session and set some session variables:

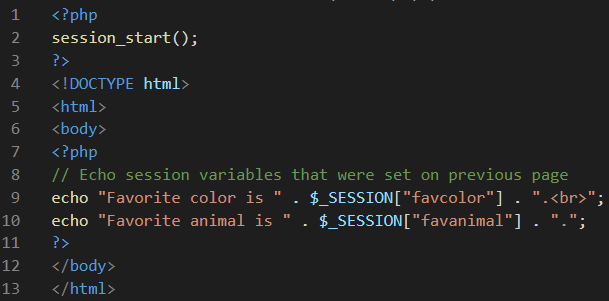


## Get PHP Session Variable Values

Next, we create another page called "demo\_session2.php". From this page, we will access the session information we set on the first page ("demo\_session1.php").

Notice that session variables are not passed individually to each new page, instead they are retrieved from the session we open at the beginning of each page (session\_start()).

Also notice that all session variable values are stored in the global $\_SESSION variable.



## Destroy a PHP Session

To remove all global session variables and destroy the session, use session\_unset() and session\_destroy()

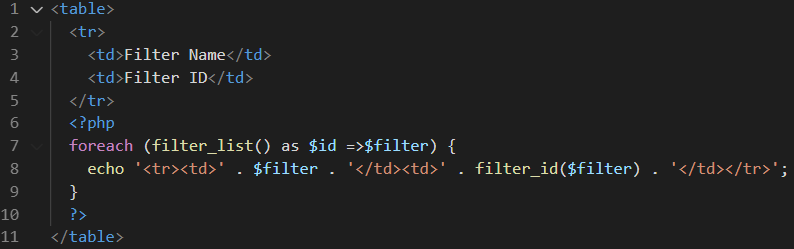
# PHP Filters

PHP filters are used to validate and sanitize external input.

The PHP filter extension has many of the functions needed for checking user input, and is designed to make data validation easier and quicker.

The filter\_list() function can be used to list what the PHP filter extension offers.

Example:



Why Use Filters?

Many web applications receive external input. External input/data can be:

1. User input from a form
2. Cookies
3. Web services data
4. Server variables
5. Database query results

## PHP filter\_var() Function

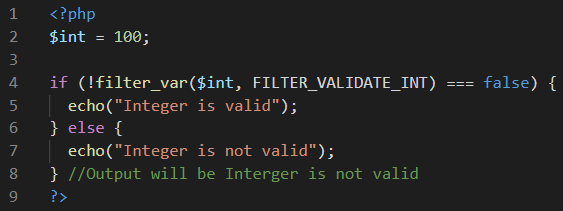
The filter\_var() function both validate and sanitize data.

The filter\_var() function filters a single variable with a specified filter. It takes two pieces of data:

1. The variable you want to check
2. The type of check to use

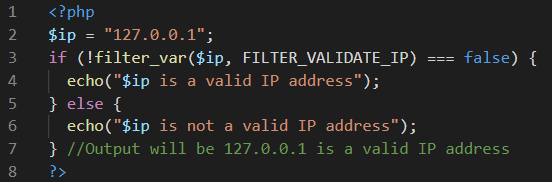
## Validate an Integer

The following example uses the filter\_var() function to check if the variable $int is an integer. If $int is an integer, the output of the code below will be: "Integer is valid". If $int is not an integer, the output will be: "Integer is not valid":



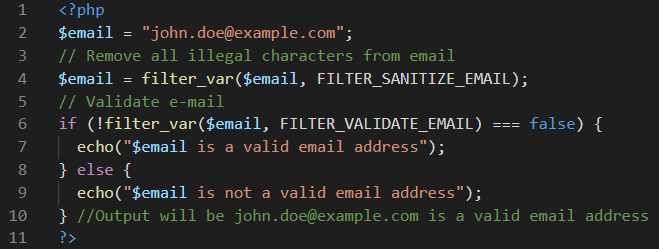
## Validate an IP Address

The following example uses the filter\_var() function to check if the variable $ip is a valid IP address:



## Sanitize and Validate an Email Address

The following example uses the filter\_var() function to first remove all illegal characters from the $email variable, then check if it is a valid email address:



# PHP Callback Functions

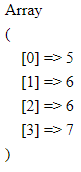
A callback function (often referred to as just "callback") is a function which is passed as an argument into another function.

Any existing function can be used as a callback function. To use a function as a callback function, pass a string containing the name of the function as the argument of another function.

Example:

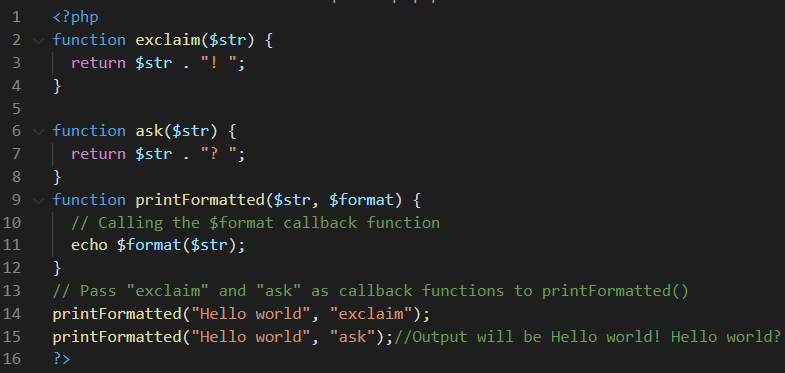


The output for the above code is as follows:



Callbacks in User Defined Functions

User-defined functions and methods can also take callback functions as arguments. To use callback functions inside a user-defined function or method, call it by adding parentheses to the variable and pass arguments as with normal functions:



## PHP and JSON

PHP has some built-in functions to handle JSON.

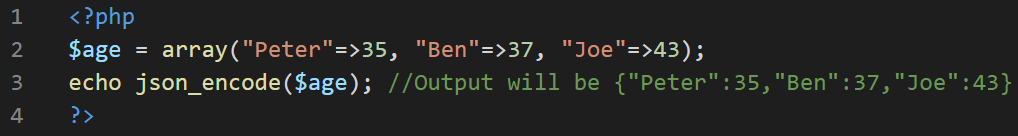
First, we will look at the following two functions:

1. json\_encode()
2. json\_decode()

## PHP - json\_encode()

The json\_encode() function is used to encode a value to JSON format.

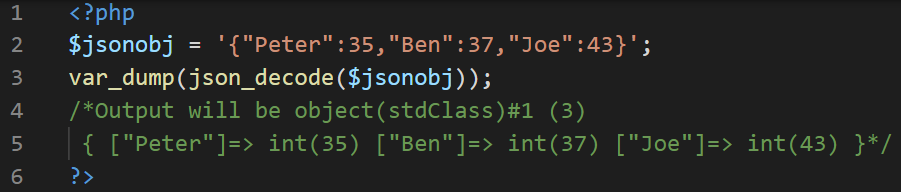
Example:



## PHP - json\_decode()

The json\_decode() function is used to decode a JSON object into a PHP object or an associative array.

Example:



# PHP Exceptions

## What is an Exception?

1. An exception is an object that describes an error or unexpected behaviour of a PHP script.
2. Exceptions are thrown by many PHP functions and classes.
3. User defined functions and classes can also throw exceptions.
4. Exceptions are a good way to stop a function when it comes across data that it cannot use.

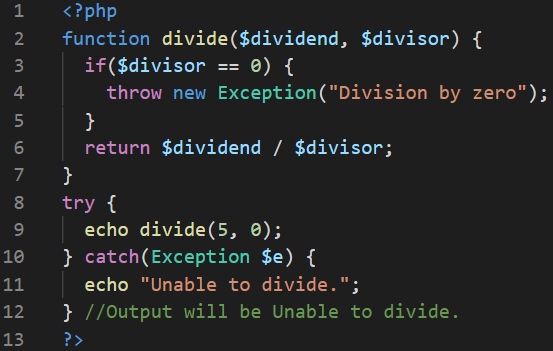
## Throwing an Exception

1. The throw statement allows a user defined function or method to throw an exception. When an exception is thrown, the code following it will not be executed.
2. If an exception is not caught, a fatal error will occur with an "Uncaught Exception" message.

## The try...catch Statement

We use the try...catch statement to catch exceptions and continue the process.

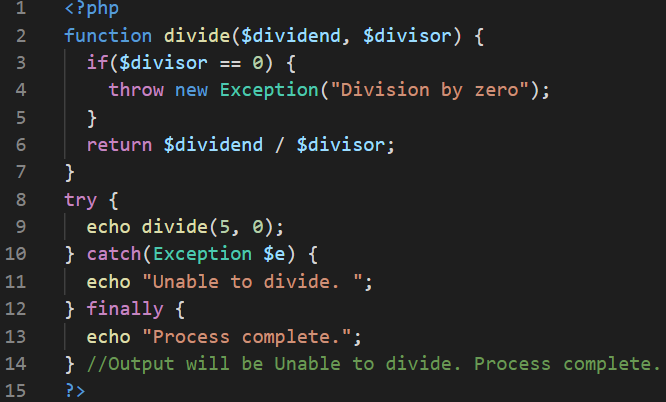
Example:



## The try...catch...finally Statement

The try...catch...finally statement can be used to catch exceptions. Code in the finally block will always run regardless of whether an exception was caught. If finally is present, the catch block is optional.

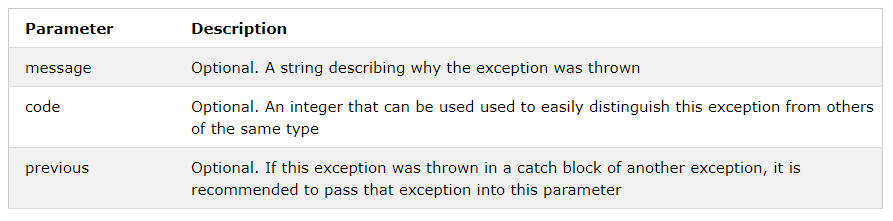
Example:



## The Exception Object

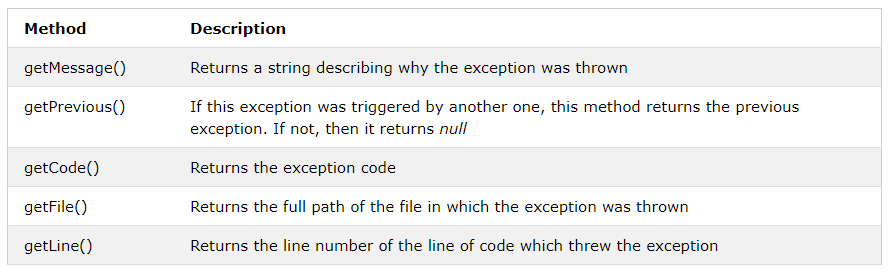
The Exception Object contains information about the error or unexpected behaviour that the function encountered.

The following is the list of parameter values with their corresponding descriptions:



### Methods

When catching an exception, the following table shows some of the methods that can be used to get information about the exception:



# PHP - OOP

## PHP What is OOP?

OOP stands for Object-Oriented Programming.

Procedural programming is about writing procedures or functions that perform operations on the data, while object-oriented programming is about creating objects that contain both data and functions.

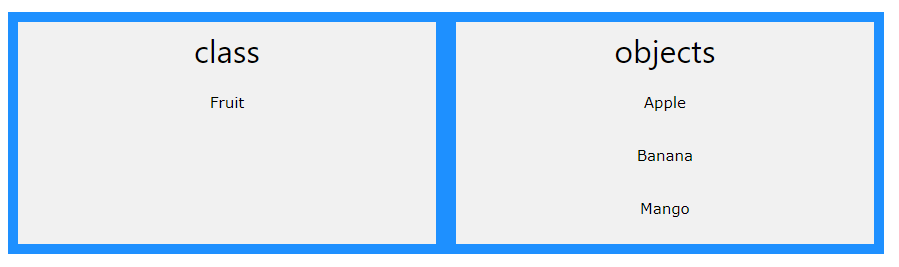
Object-oriented programming has several advantages over procedural programming:

1. OOP is faster and easier to execute
2. OOP provides a clear structure for the programs
3. OOP helps to keep the PHP code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug
4. OOP makes it possible to create full reusable applications with less code and shorter development time

## PHP - What are Classes and Objects?

Classes and objects are the two main aspects of object-oriented programming.

Look at the following illustration to see the difference between class and objects:



So, a class is a template for objects, and an object is an instance of a class.

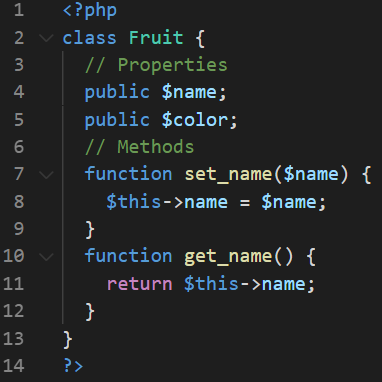
When the individual objects are created, they inherit all the properties and behaviors from the class, but each object will have different values for the properties.

# PHP OOP - Classes and Objects

## Define a Class

A class is defined by using the class keyword, followed by the name of the class and a pair of curly braces ({}). All its properties and methods go inside the braces.

Example:

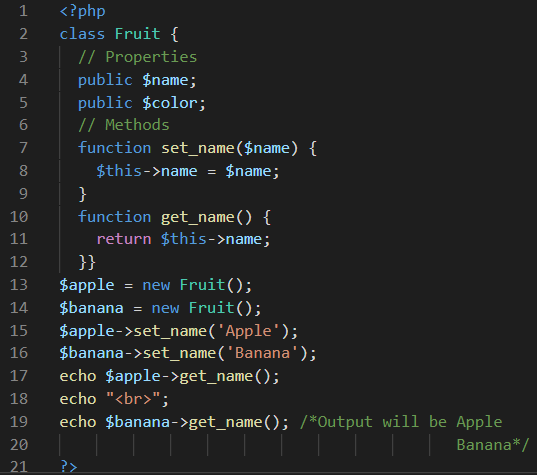


## Define Objects

Classes are nothing without objects! We can create multiple objects from a class. Each object has all the properties and methods defined in the class, but they will have different property values.

Objects of a class is created using the new keyword.

In the example below, $apple and $banana are instances of the class Fruit:



## PHP - instanceof

You can use the instanceof keyword to check if an object belongs to a specific class

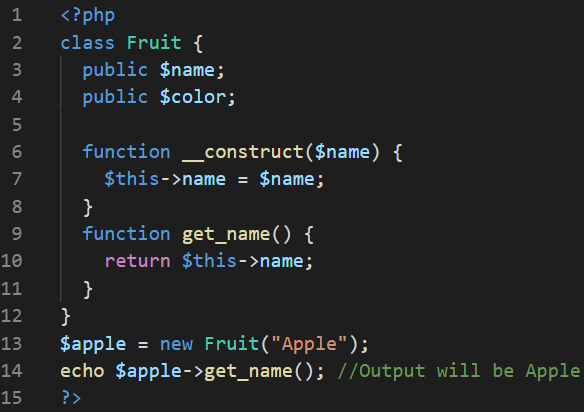
## PHP - The \_\_construct Function

A constructor allows you to initialize an object's properties upon creation of the object.

If you create a \_\_construct() function, PHP will automatically call this function when you create an object from a class.

Notice that the construct function starts with two underscores (\_\_)!

We see in the example below, that using a constructor saves us from calling the set\_name() method which reduces the amount of code:



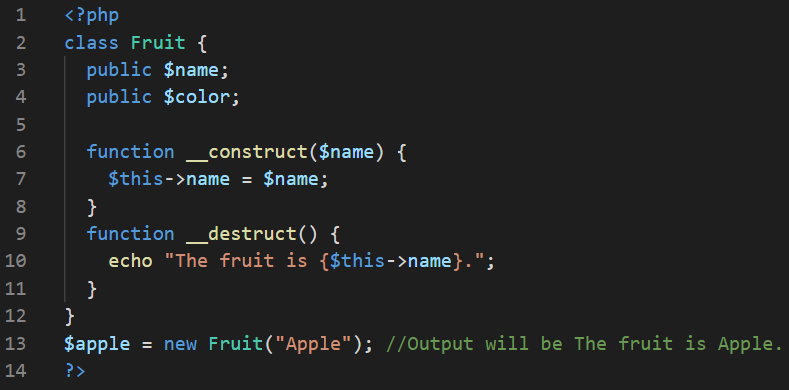
## PHP - The \_\_destruct Function

A destructor is called when the object is destructed or the script is stopped or exited.

If you create a \_\_destruct() function, PHP will automatically call this function at the end of the script.

Notice that the destruct function starts with two underscores (\_\_)!

The example below has a \_\_construct() function that is automatically called when you create an object from a class, and a \_\_destruct() function that is automatically called at the end of the script:



## PHP - Access Modifiers

Properties and methods can have access modifiers which control where they can be accessed.

There are three access modifiers:

* public - the property or method can be accessed from everywhere. This is default
* protected - the property or method can be accessed within the class and by classes derived from that class
* private - the property or method can ONLY be accessed within the class

In the following example we have added three different access modifiers to the three properties. Here, if you try to set the name property it will work fine (because the name property is public). However, if you try to set the color or weight property it will result in a fatal error (because the color and weight property are protected and private):



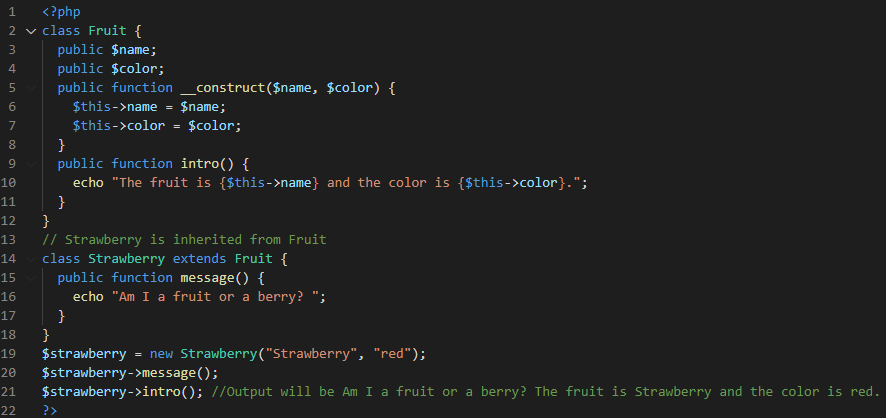
## PHP - What is Inheritance?

Inheritance in OOP = When a class derives from another class.

The child class will inherit all the public and protected properties and methods from the parent class. In addition, it can have its own properties and methods.

An inherited class is defined by using the extends keyword.

Let's look at an example:



### Example Explained

The Strawberry class is inherited from the Fruit class.

This means that the Strawberry class can use the public $name and $color properties as well as the public \_\_construct() and intro() methods from the Fruit class because of inheritance.

The Strawberry class also has its own method: message().

## PHP - Overriding Inherited Methods

Inherited methods can be overridden by redefining the methods (use the same name) in the child class.

## PHP - The final Keyword

The final keyword can be used to prevent class inheritance or to prevent method overriding.

## PHP - Class Constants

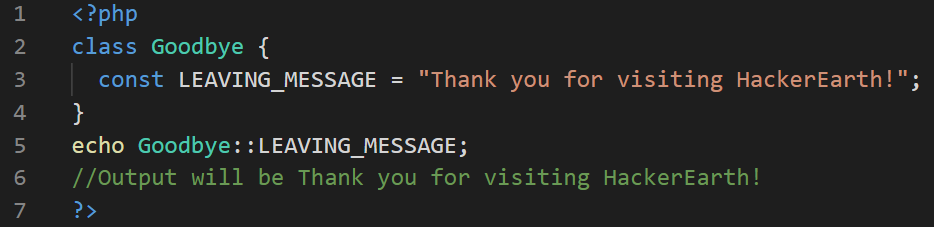
Constants cannot be changed once it is declared.

Class constants can be useful if you need to define some constant data within a class.

A class constant is declared inside a class with the const keyword.

Class constants are case-sensitive. However, it is recommended to name the constants in all uppercase letters.

We can access a constant from outside the class by using the class name followed by the scope resolution operator (::) followed by the constant name, like here:



## PHP - What are Abstract Classes and Methods?

Abstract classes and methods are when the parent class has a named method, but need its child class(es) to fill out the tasks.

An abstract class is a class that contains at least one abstract method. An abstract method is a method that is declared, but not implemented in the code.

An abstract class or method is defined with the abstract keyword.

When inheriting from an abstract class, the child class method must be defined with the same name, and the same or a less restricted access modifier. So, if the abstract method is defined as protected, the child class method must be defined as either protected or public, but not private. Also, the type and number of required arguments must be the same. However, the child classes may have optional arguments in addition.

So, when a child class is inherited from an abstract class, we have the following rules:

1. The child class method must be defined with the same name and it redeclares the parent abstract method
2. The child class method must be defined with the same or a less restricted access modifier
3. The number of required arguments must be the same. However, the child class may have optional arguments in addition

## PHP - What are Interfaces?

Interfaces allow you to specify what methods a class should implement.

Interfaces make it easy to use a variety of different classes in the same way. When one or more classes use the same interface, it is referred to as "polymorphism".

Interfaces are declared with the interface keyword.

## PHP - Interfaces vs. Abstract Classes

Interface are similar to abstract classes. The difference between interfaces and abstract classes are:

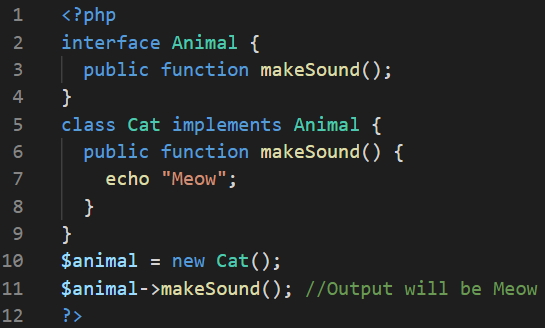
* Interfaces cannot have properties, while abstract classes can
* All interface methods must be public while abstract class methods may also be private or protected
* All methods in an interface are abstract, so they cannot be implemented in code and the abstract keyword is not necessary
* Classes can implement an interface while inheriting from another class at the same time

## PHP - Using Interfaces

To implement an interface, a class must use the implements keyword.

A class that implements an interface must implement all of the interface's methods.

Example:



## PHP - What are Traits?

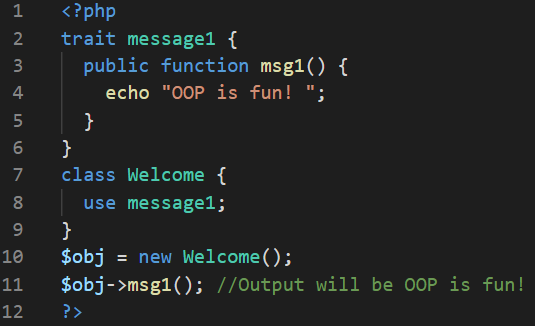
PHP only supports single inheritance: a child class can inherit only from one single parent.

So, what if a class needs to inherit multiple behaviors? OOP traits solve this problem.

Traits are used to declare methods that can be used in multiple classes. Traits can have methods and abstract methods that can be used in multiple classes, and the methods can have any access modifier (public, private, or protected).

Traits are declared with the trait keyword

Example:



### Example Explained

Here, we declare one trait: message1. Then, we create a class: Welcome. The class uses the trait, and all the methods in the trait will be available in the class.

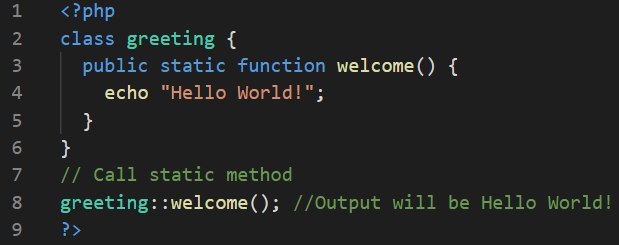
If other classes need to use the msg1() function, simply use the message1 trait in those classes. This reduces code duplication, because there is no need to redeclare the same method over and over again.

## PHP - Static Methods

Static methods can be called directly - without creating an instance of a class.

Static methods are declared with the static keyword. To access a static method use the class name, double colon (::), and the method name

Example:



### Example Explained

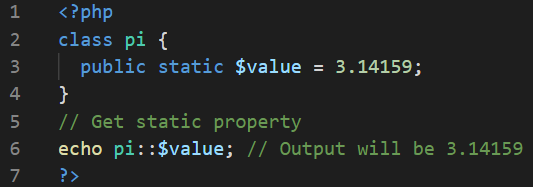
Here, we declare a static method: welcome(). Then, we call the static method by using the class name, double colon (::), and the method name (without creating a class first).

## PHP - Static Properties

Static properties can be called directly - without creating an instance of a class.

Static properties are declared with the static keyword. To access a static property use the class name, double colon (::), and the property name

Example:



### Example Explained

Here, we declare a static property: $value. Then, we echo the value of the static property by using the class name, double colon (::), and the property name (without creating a class first).

## PHP Namespaces

Namespaces are qualifiers that solve two different problems:

1. They allow for better organization by grouping classes that work together to perform a task
2. They allow the same name to be used for more than one class

For example, you may have a set of classes which describe an HTML table, such as Table, Row and Cell while also having another set of classes to describe furniture, such as Table, Chair and Bed. Namespaces can be used to organize the classes into two different groups while also preventing the two classes Table and Table from being mixed up.

## Declaring a Namespace

Namespaces are declared at the beginning of a file using the namespace keyword.

## Using Namespaces

Any code that follows a namespace declaration is operating inside the namespace, so classes that belong to the namespace can be instantiated without any qualifiers. To access classes from outside a namespace, the class needs to have the namespace attached to it.

## Namespace Alias

It can be useful to give a namespace or class an alias to make it easier to write. This is done with the use keyword

## PHP - What is an Iterable?

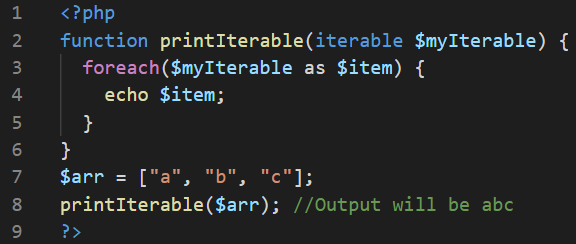
An iterable is any value which can be looped through with a foreach() loop.

The iterable pseudo-type was introduced in PHP 7.1, and it can be used as a data type for function arguments and function return values.

## PHP - Using Iterables

The iterable keyword can be used as a data type of a function argument or as the return type of a function.

Example:



## PHP - Creating Iterables

Arrays

All arrays are iterables, so any array can be used as an argument of a function that requires an iterable.

Iterators

Any object that implements the Iterator interface can be used as an argument of a function that requires an iterable.

An iterator contains a list of items and provides methods to loop through them. It keeps a pointer to one of the elements in the list. Each item in the list should have a key which can be used to find the item.

An iterator must have these methods:

* current() - Returns the element that the pointer is currently pointing to. It can be any data type
* key() Returns the key associated with the current element in the list. It can only be an integer, float, boolean or string
* next() Moves the pointer to the next element in the list
* rewind() Moves the pointer to the first element in the list
* valid() If the internal pointer is not pointing to any element (for example, if next() was called at the end of the list), this should return false. It returns true in any other case

-----------------------Reference credits: w3schools,Tutorispoint, [https://github.com/hegdenaveen1/Become-A-Full-Stack-Web-Developer#start-here](https://www.google.com/url?q=https://github.com/hegdenaveen1/Become-A-Full-Stack-Web-Developer%23start-here&sa=D&source=hangouts&ust=1599916002593000&usg=AFQjCNFqAEgym0xUC_MkOX8mYy5AtiOv1g) -------------------