**PHP NOTES**

<https://www.youtube.com/watch?v=OK_JCtrrv-c&ab_channel=freeCodeCamp.org>

**What is PHP**

* Php is basically a server-side scripting language embedded in a HTML.
* Php stands for Hypertext Pre-processor.
* Php allows web developers to create dynamic content and interact with databases.
* It is known for its simplicity, speed and flexibility.
* On cmd type **echo %path%** to see the paths that are been made global in the environment variables.
* On cmd type **php -v** to see the version.
* When we want to use php then we have to run php on web server.
* It is a programming language that runs on a web-server.
* We use php on web server to interact with the HTML pages.
* Web server comes with php.
* Command prompt is used to interact with the computer with the text command.
* To create a web-server for php we have to write the following command on cmd:

**php -S localhost:4000** (S is in capital)

* The above line is used to run a webserver on a local machine in port 4000.
* Don’t exit the cmd window when you start webserver or else the server will get close.
* PHP is very tightly coupled with HTML language.
* We basically write our php code inside a HTML layout.
* PHP tag:

**<?php**

**----------------**

**----------------**

**?>**

* Anything that is been written inside this php tag is considered as a PHP code.
* To print Hello world in the screen we use the following syntax:
  + **echo(“Hello World”);**
  + echo is a print statement in php.
  + We can also print <h1>Koushik’s Website</h1> using echo command.
* PHP codes are always executed in order.
* **Variables:** It is a container that contains the pieces of information in our program.
  + **$** symbol is used to create a variable in php.
  + E.g. **$name = “Koushik Sadhu”;**
  + To embed variable data in a echo statement we just need to write **$** symbol along with that variable name.
  + E.g. **echo(“My name is *$name*”);**

**DATA-TYPES IN PHP:**

* $phrase = "PHP is a server side scripting language."; **// String Type.**
* $number = 30; **// Integer Type.**
* $cgpa = 9.52; **// Float Type.**
* $isMale = true; **// Boolean Type.**
* **Null** in PHP specifies no value.

**STRINGS IN PHP:**

* $str = “This is a String.”; // storing inside a string variable.
* echo $str; //to print

**STRING FUNCTIONS:**

**$str = “Koushik”;**

* strtolower($str) – koushik
* strtoupper($str) – KOUSHIK
* strlen($str) – 7
* To fetch the character from a particular index of a string: **$str[1]**
* **“Sadhu”[0]** – S // Fetching the first character by putting index position in a square bracket**.**
* strsubstr($str, 1, 4); -- oush // start from index 1 and go up to 4 ahead. 4 is basically the length

**NUMBERS IN PHP:**

* echo "20 + 5 = ",20+5;
* echo "20 - 5 = ",20-5;
* echo "20 \* 5 = ",20\*5;
* echo "20 / 5 = ",20/5;
* echo "20 % 5 = ",20%5;
* echo "20 + 5 \* 4 = ",20+5\*4;
* echo "(20 + 5) \* 4 = ",(20+5)\*4;
* echo "2 ^ 4 = ",pow(2, 4);
* echo "root(16) = ",sqrt(16);
* $num = 10;
* echo "num = ",$num;
* $num++;
* echo "num++ = ",$num;
* $num--;
* echo "num-- = ",$num;
* ++$num;
* echo "++num = ",$num;
* --$num;
* echo "--num = ",$num;
* $num += 20;
* echo "num += 20 = ",$num;
* $num -= 20;
* echo "num -= 20= ",$num;

**MATH FUNCTIONS:**

* echo "abs(-100) = ",abs(-100),"<br>";
* echo "pow(2, 10) = ",pow(2, 10),"<br>";
* echo "sqrt(25) = ",sqrt(25),"<br>";
* echo "max(100, -10) = ",max(100, -10),"<br>";
* echo "min(100, -10) = ",min(100, -10),"<br>";
* echo "ceil(20.000009) = ",ceil(20.436584),"<br>";
* echo "floor(20.999999) = ",floor(20.436584),"<br>";

**USER INPUTS:**

<form action="userInputs.php" method="GET">

<label>Name: </label>

<input type="text" name="username">

<br>

<label>Age: </label>

<input type="number" name="age">

<br>

<input type="submit">

</form>

<br>

<?php

echo **$\_GET["username"]**," form is submitted successfully.<br>”;

echo “Age of ",$\_GET["username"]," is: ",**$\_GET["age"]**,".";

?>

**GET AND POST URL PARAMETERS METHOD:**

The GET and POST methods are two commonly used HTTP methods for sending data to a server. While both methods can be used to send URL parameters, they differ in how the parameters are transmitted and the implications for data handling.

**GET method:** $\_GET[“variable\_name”]

* Parameters in URL: In the GET method, parameters are appended to the URL as a query string. For example, http://example.com/page?param1=value1&param2=value2.
* Visibility: Since the parameters are part of the URL, they are visible in the browser's address bar and can be bookmarked or shared easily.
* Data length limitation: There is a limitation on the length of the URL, so the amount of data that can be sent is limited. Different browsers and servers have different limitations, but it is typically around a few thousand characters.
* Caching: GET requests can be cached by browsers and other intermediaries, which means that the URL parameters might be stored in browser history or caching servers.
* Idempotent: GET requests are considered idempotent, meaning that making the same request multiple times should have the same result.

**POST method:** $\_POST[“variable\_name”]

* Parameters in request body: In the POST method, parameters are sent in the body of the HTTP request, rather than as part of the URL.
* Visibility: The parameters are not visible in the URL or the browser's address bar, which can provide some level of privacy.
* Data length flexibility: There is no inherent limitation on the data length in a POST request, allowing larger amounts of data to be sent compared to the GET method.
* Caching: POST requests are not typically cached, as they are considered non-idempotent, meaning that making the same request multiple times may have different results.
* Security: POST requests are considered more secure for transmitting sensitive data, as the parameters are not visible in the URL and are encrypted when using HTTPS.

In summary, the main differences between GET and POST methods for URL parameters lie in the visibility of parameters, data length limitations, caching behavior, and security implications. GET is typically used for retrieving data, while POST is often used for submitting data to a server.

**POST is more secure than GET.**

**ARRAYS:**  An array is a container or a structure where we can store multiple pieces of information.

Array in PHP can hold the element of same or different data type. So, in PHP we can store element with more than one data type inside an array.

**For e.g.,** $arr = array(“Koushik”, 27, false, “Shuvam”, 10.20, “Pranay”, “Anmol”, “Adarsh”);

echo $arr; ---- Array

echo $arr[0] ---- Koushik

echo $arr[count($arr)-1]; ---- Adarsh

**Modifying element in an array:**

$arr[3] = “Soubhik”;

Echo $arr[3]; ---- Soubhik

**NOTE:** If anything is having false value then nothing will be echoed. Boolean false value does not show up in PHP.

In PHP, array works dynamically i.e., its size is not fixed we can add any element to it anytime.

**For e.g.,** $arr[count($arr)] = “Last\_Element”; // Here, we are adding element in the last position.

**To print the total number of element present inside the array:** $len = count($arr);

**FOREACH LOOP:** “foreach” loop is basically used to iterate over each element of an array.

**For e.g.,** foreach(*$arr* as *$i*){

echo *$i*,” “;

}

---- All the element that are present in an array will get printed except FALSE value and in place of TRUE, 1 will be printed.

**CHECKBOXES:**

****

**ASSOCIATIVE ARRAYS:**

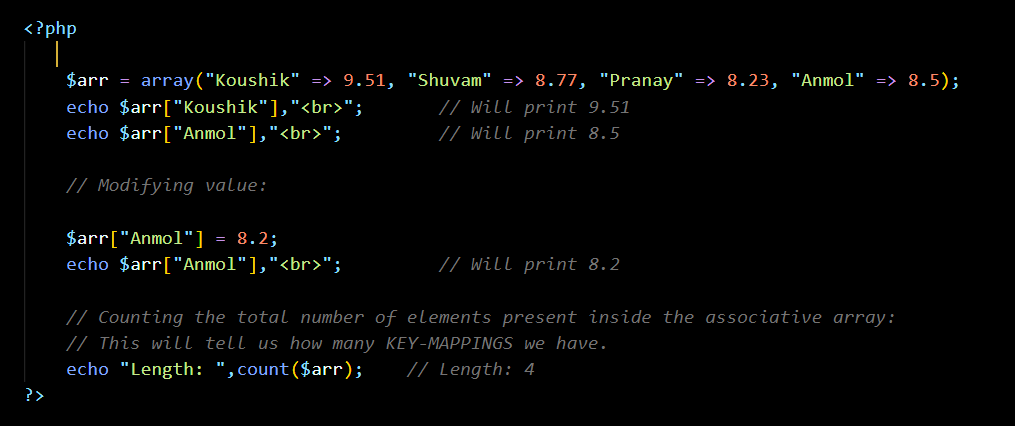
An associative array is a type of array in which we not only store data value but also can store key-value pairs. So, unlike a normal array I could just store like numbers or text or combination of both in an associative array.

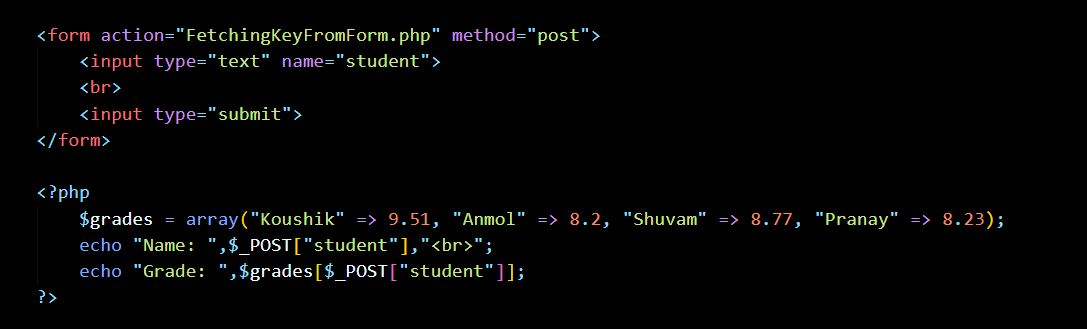
* In Associative Array, **=>** (Equals to and greater sign) is used to store KEY-VALUE pairs.

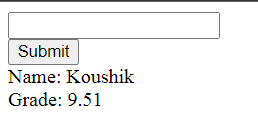
**Example:** $arr = array(“Koushik” => 9.52, “Shuvam” => 8.77, “Pranay” => “8.23”);

* The KEY must be unique, No duplicate key should be present in the associative array.
* The VALUE may or may not be unique but the KEY should always be unique.
* If the KEY is not unique and if that non-unique key is being fetched than the last key of that name, value will be fetched.

**NOTE:** In Associative array, we access elements using key whereas In Normal array, we access element using index number.



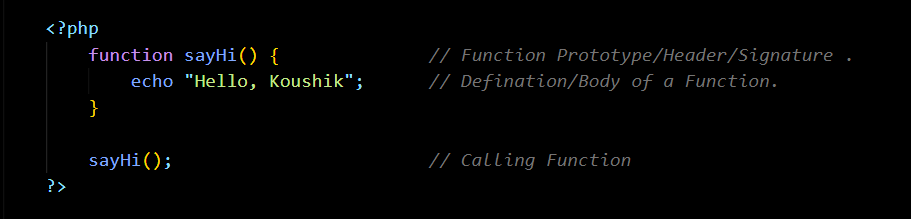




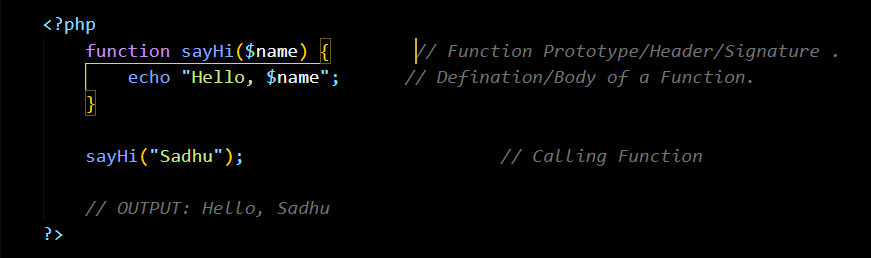
**OUTPUT:**

**FUNCTIONS:** A function is a piece/block of code that perform a specific task, when it is called.

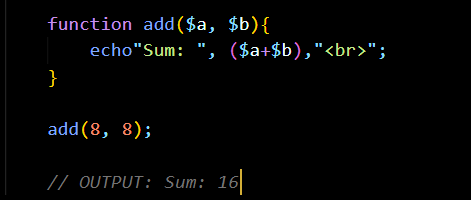
The advantage of using function is that, we can write the code for one time and execute this code multiple time by simply calling that function.



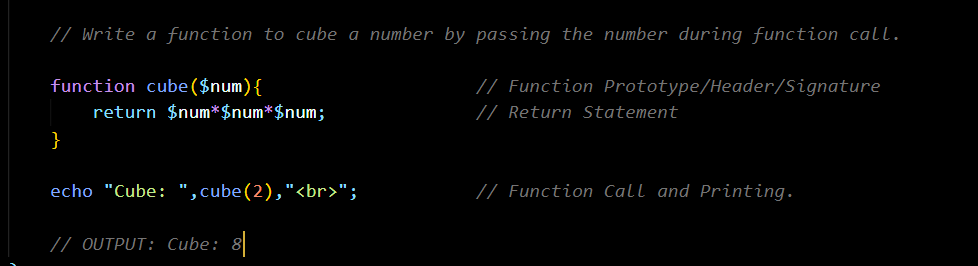
**Passing Parameters in a Function:**

****

**Addition of two numbers using function call and passing parameters during function call:**



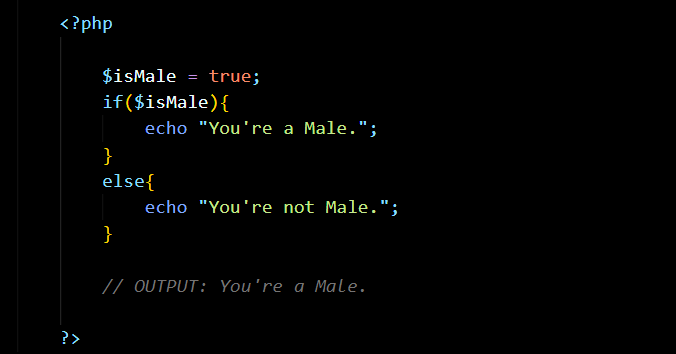
**Return Statement:** Giving an information back from a function, when it is been called.

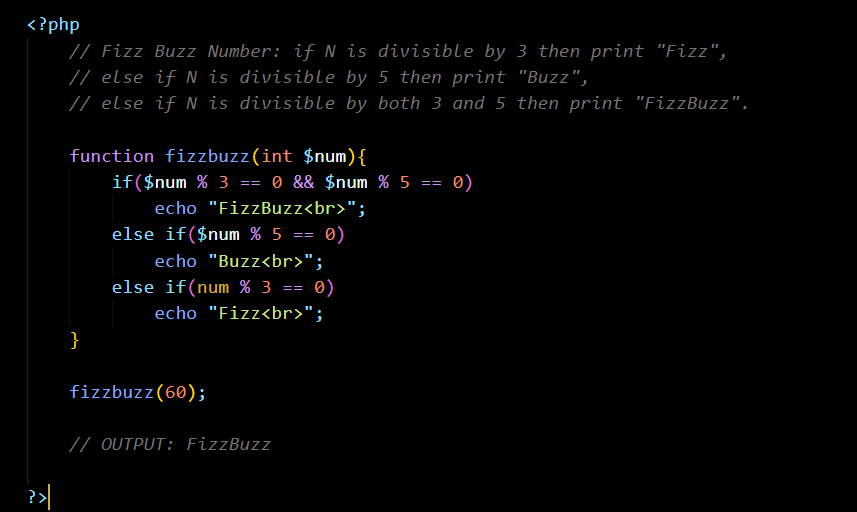


**IF-ELSE Statements:** If-else statements are basically used in our programs to make decisions.

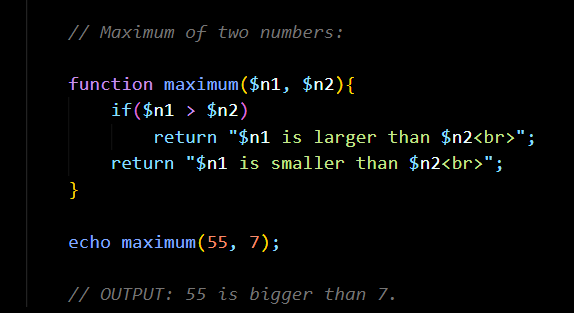
**&&: AND OPERATOR**

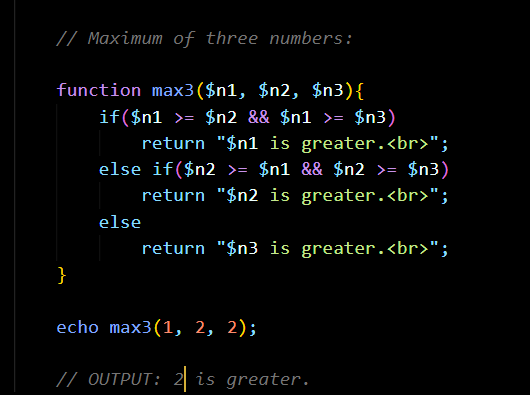
**|| : OR OPERATOR**

****

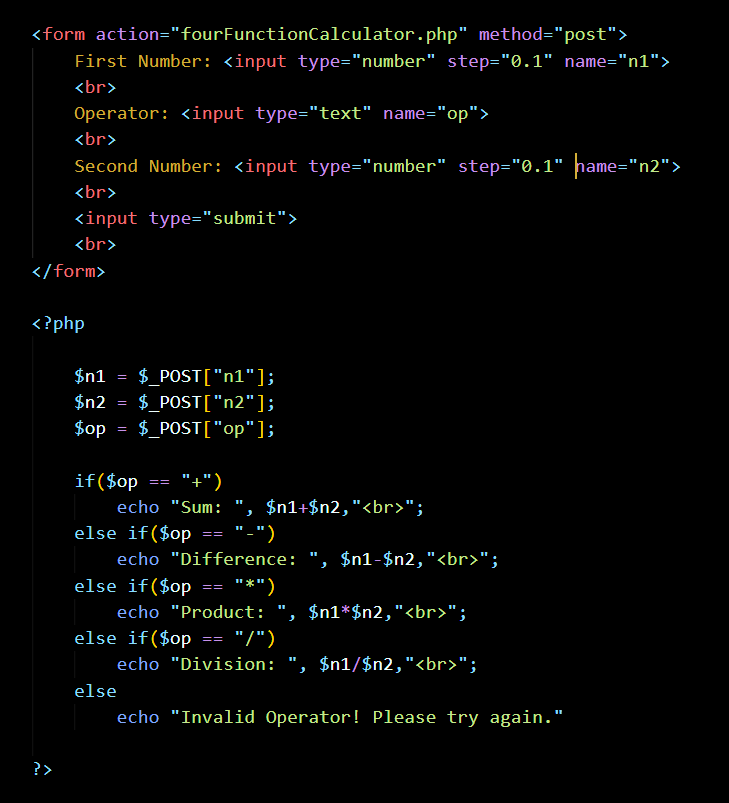


**COMPARISONS IN IF-STATEMENTS:**

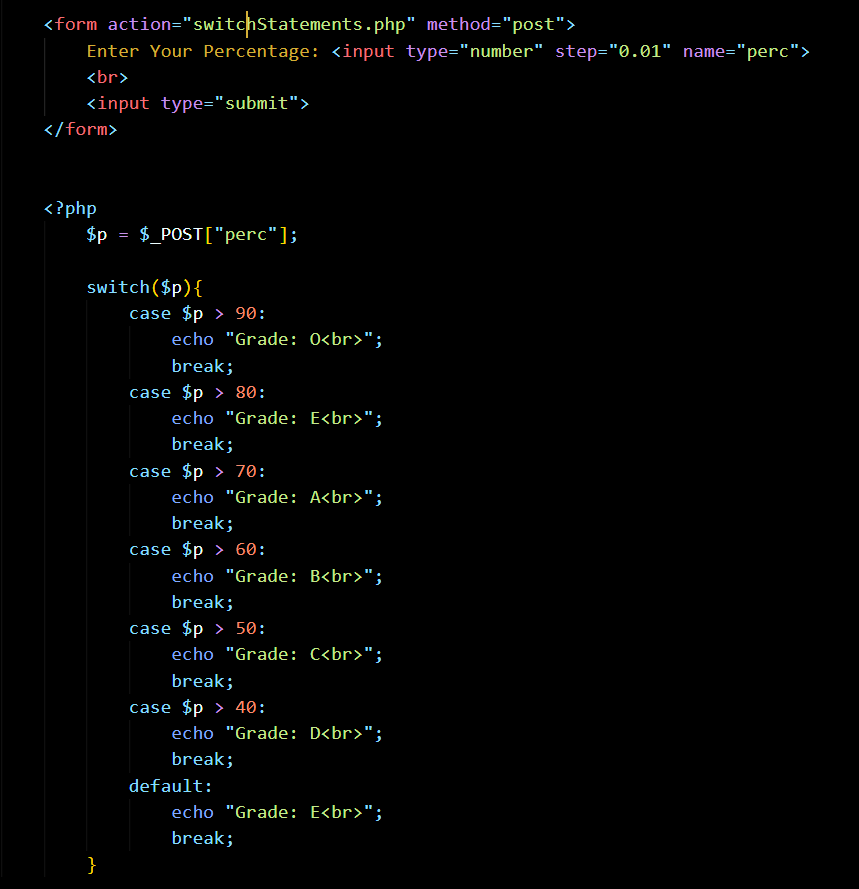




**CALCULTOR:**

****

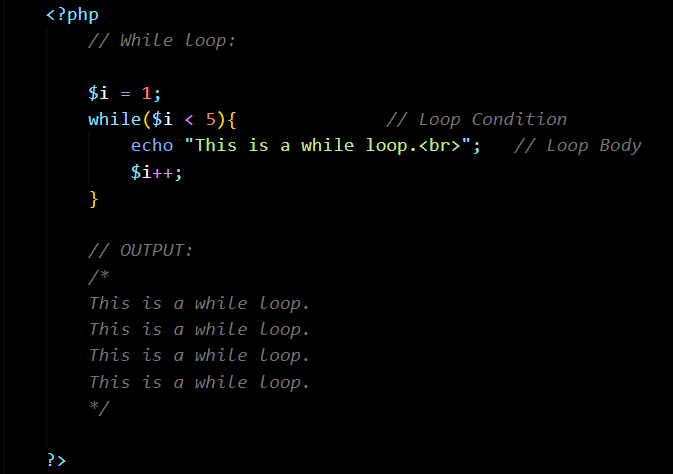
**SWITCH-CASE STATEMENTS:** It is a special type of if-else statements where we can compare one value with a bunch of multiple values.



**WHILE-LOOP:** While loop is basically a programming structure which is basically used to loop over a specified block of code, while a certain condition is TRUE.

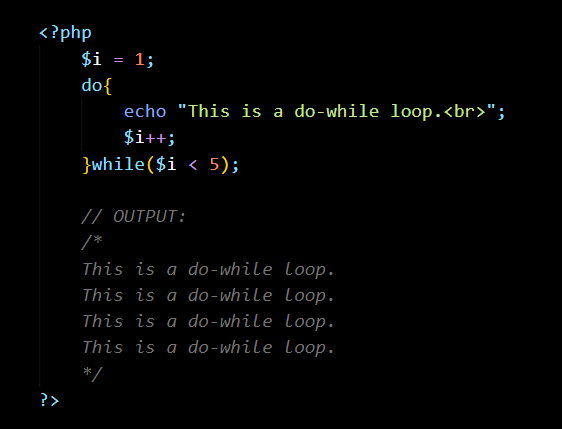
Loop allows us to keep repeating something as long as the certain condition is true.

In this condition is checked first and then loop body is being executed.

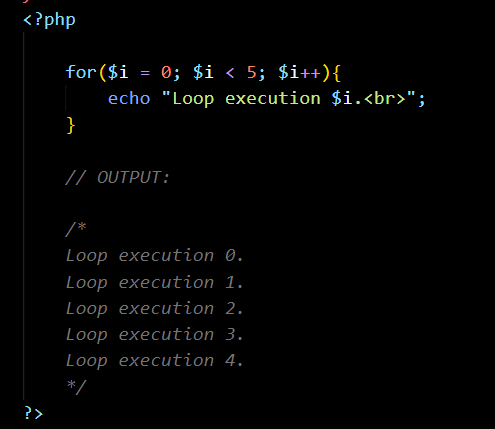


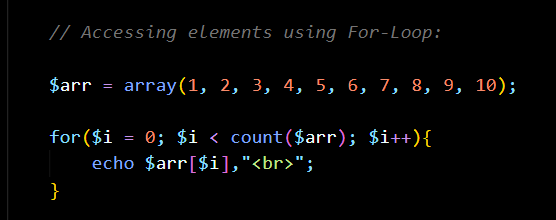
Infinite loop: It is a loop in which condition of the loop will never be FALSE. It is always TRUE.

**DO-WHILE LOOP:** It is a type of loop in which loop body is executed first and then body is checked. In do-while loop, a loop will execute at least once.



**FOR-LOOP:** It is a special types of loop which is used in conjunction with a variable. It keeps track of the specific variables as you go through (which is also called Iterating variable). This loop is basically used when we know exactly how many times the loop will execute.





**COMMENTS:** Comments are a piece of code inside a PHP file which is not rendered by the PHP. It is used to add some extra information or reminders /notes for ourselves in the code which is not considered by PHP.

* Single line comments: // This is a single line comment.
* Multi Line Comments: /\*

This is a multiline

comment.

\*/

**INCLUDE HTML IN PHP:** Include statement is used to include any other PHP file or HTML file into our current PHP file and use that code.

03:35