**BASIC CONCEPT OF PROGRAMMING**

**How to create folder dynamically via date using cmd?**

**Concept 1** : A folder name can't contain, illegal character like **\, /, :, \*, "", ?, |, <>**

If while running the command, **mkdir %date%** if you get an error, incorrect syntax error.

Make sure you date format does not contain illegal character **\, /, :, \*, "", ?, |, <>**

**BASIC STORY**

PHP is a web technology, and PHP is a server side scripting language.

**Ques : Why PHP is not executed on browser?**

https://www.facebook.com/

file:///c:/Users/Abhay%20Singh/Desktop/App-2021/PHP/home.html

1). According to client server model,

2). Anything which is loaded by client (browser) will be called as resource.

https://www.facebook.com/profile-picture.png (Resource)

**Common terms used in Web Technology :**

1). Human (End User)

2). God (Server)

3). Prayer (Request)

4). Blessings (Response)

**Ques : How end user makes a request?**

**Ans :** URL (Uniform Resource Locater)

**Note :** End-user will use browser as an interface/platform to make a request by using URL.

**Ques : How to see request and response?**

**Ans :** Inspect your browser,

Go to the response tab.

**Some Important Status Code :**

**100** continue

**102** processing

**200** ok

**202** accepted

**204** no content

**205** reset content

**300** multiple choice

**302** found

**306** unused

**400** bad request

**402** payment request

**403** forbidden

**404** not found

**405** method not allowed

**406** not acceptable

**408** request timeout

**423** locked

**425** too early

**429** too many requests

**500** internal server request

**503** service unavailable

**507** insufficient storage

**Ques : How many types of request are there?**

1). Synchronous Request

2). Asynchronous Request

**1). Synchronous Request :**

A request in which one event depends on other request to happen such type of request is called synchronous request. These requests are blocking in nature.

**Example :**

JS --> window.alert()

PHP --> Form submission

**2). Asynchronous Request :**

A request in which one event doesn't depends on other request to happen such type of request is called asynchronous request. These requests are non-blocking in nature.

**Example :**

1). Node.js --> window.alert()

R1 --> Waiting

R2 --> Processing --> Succeed

R3 --> Waiting

R4 --> Processing --> Succeed

2). PHP

R1 --> Waiting --> Succeed

R2 --> Waiting --> Processing --> Succeed

R3 --> Waiting --> Processing --> Succeed

R4 --> Waiting --> Processing

**INTRODUCTION TO LARAGON**

**Getting started with Laragon :**

**1).** While opening laragon screen you will called as laragon control panel.

**Menu :** All the importants,

**Settings, logs, errors, initialization, configuration related to apache, MySQL, PHP**, will be listed here.

**Important Files :**

.**ini** - initialization file

It will contain all the important services, assets, resources, permissions, utilities required by any program during startup.

**Examples :**

php.ini

laragon.ini

mysql.ini

etc.

**.log - error file**

It will contains all the run-time errors and exceptions occured during execution of the program.

It stores the information with alert status.

1). Success

2). Info

3). Warning

4). Danger (Critical)

**.cnf or .conf - configuration file**

It contains the configurations which are required during execution of the program at runtime execution.

**2). www :** This menu will show all the projects you will create inside **C:/laragon/www.**

**1).** **Server Document Root** : It contains the system path **C:/laragon/www**

Document root of wamp : **C:/wamp/www**

Document root of xampp : **C:/xampp/htdocs**

**2). Explore Menu** : It is used to open the existing projects in explorer window.

**3). Share** : It is used to share the resources with your team mates connected to network.

**4). Project Launcher** : It will launch every project in URL.

**3). Quick App** : In PHP, every project is known as an app.

These apps can be classified **[community-name] App**

**For example :**

1). wordpress community

wordpress app

2). laravel community

laravel app

3). drupal community

drupal app

4). codeIgnitor community

codeIgnitor App or CI App

Using quick app we can automatically configure and install the application.

Make sure you have internet connection.

https://github.com/awnishmmg/Zwavexampp-list-plugin/blob/master/Xampp-Directory-Listing.zip?raw=true

**Ques : What is index.php file?**

**Ans :** index.php file is the by default file of apache which will be directly loaded in the URL.

**Ques : Which server is use by PHP?**

**Ans :** Apache Server

**Ques : Apache server is installed in which server?**

**Ans :** Laragon Server

**Ques : What is my current file name?**

**Ans :** p1.php

**Ques : Where this file is located?**

**Ans :** inside **c:/laragon/www** inside **laragon server.**

**Ques :** Where index.php file is located?

**Ans :** inside c:/laragon/www

| index.php

**Ques :** From where you can access this index.php?

**Ans :** From client

**Ques :** How to we can make a request?

**Ans :** URL (Uniform Resource Locator)

URI (Uniform Resource Identifier)

**DIFFERENCE B/W HTTP AND HTTPS**

**HTTP HTTPS**

**URL**  http:// https://

**Security** Unsecure Enhanced Security

**Port**  PORT 80 PORT 443

**OSI Layer**  Application Layer Transport Layer

**TLS Certificate** No Yes

**Domain Validation**  Not required Domain validation[+legal validation]

**Encryption**  No Yes

**TLS :** Transport Layer Security

**OSI** : Open System Interconnection

**What Is the OSI Model?**

The Open Systems Interconnection (OSI) model describes seven layers that computer systems use to communicate over a network. It was the first standard model for network communications, adopted by all major computer and telecommunication companies in the early 1980s

The modern Internet is not based on OSI, but on the simpler TCP/IP model. However, the OSI 7-layer model is still widely used, as it helps visualize and communicate how networks operate, and helps isolate and troubleshoot networking problems.

OSI was introduced in 1983 by representatives of the major computer and telecom companies, and was adopted by ISO as an international standard in 1984.

**OSI Model Explained: The OSI 7 Layers**

**7). Application Layer**

The application layer is used by end-user software such as web browsers and email clients. It provides protocols that allow software to send and receive information and present meaningful data to users. A few examples of application layer protocols are the Hypertext Transfer Protocol (HTTP), File Transfer Protocol (FTP), Post Office Protocol (POP), Simple Mail Transfer Protocol (SMTP), and Domain Name System (DNS).

**6). Presentation Layer**

The presentation layer prepares data for the application layer. It defines how two devices should encode, encrypt, and compress data so it is received correctly on the other end. The presentation layer takes any data transmitted by the application layer and prepares it for transmission over the session layer.

**5). Session Layer**

The session layer creates communication channels, called sessions, between devices. It is responsible for opening sessions, ensuring they remain open and functional while data is being transferred, and closing them when communication ends. The session layer can also set checkpoints during a data transfer—if the session is interrupted, devices can resume data transfer from the last checkpoint.

**4). Transport Layer**

The transport layer takes data transferred in the session layer and breaks it into “segments” on the transmitting end. It is responsible for reassembling the segments on the receiving end, turning it back into data that can be used by the session layer. The transport layer carries out flow control, sending data at a rate that matches the connection speed of the receiving device, and error control, checking if data was received incorrectly and if not, requesting it again.

**3). Network Layer**

The network layer has two main functions. One is breaking up segments into network packets, and reassembling the packets on the receiving end. The other is routing packets by discovering the best path across a physical network. The network layer uses network addresses (typically Internet Protocol addresses) to route packets to a destination node.

**2). Data Link Layer**

The data link layer establishes and terminates a connection between two physically-connected nodes on a network. It breaks up packets into frames and sends them from source to destination. This layer is composed of two parts—Logical Link Control (LLC), which identifies network protocols, performs error checking and synchronizes frames, and Media Access Control (MAC) which uses MAC addresses to connect devices and define permissions to transmit and receive data.

**1). Physical Layer**

The physical layer is responsible for the physical cable or wireless connection between network nodes. It defines the connector, the electrical cable or wireless technology connecting the devices, and is responsible for transmission of the raw data, which is simply a series of 0s and 1s, while taking care of bit rate control.

**An URL have following parts :**

URL is a kind of web address to some resource located at any server.

An URL can be two types :

**1). Protocol Based**

**2). Non-Protocol Based (Uses IP Address)**

**URL Syntax**

[protocol-name]://<sub-domain>/<domain>/<host>:<port>/<resource-url>

**URI Segment :** In any URL which parts are separated by '/' is called as URI segment.

**Explanation :** After port segment it's basically path to resource.

After protocol segment and before endpoint is known as domain name.

Every domain name may have a sub-domain or not but will have a port, and every domain will have an IP address too. Since anyone can't learn or remember IP address so we provide a alias name this alias name is called as domain name and this domain name has to be registered (mapped) with the IP and using DNS (Domain Name Server).

**Ques :** Identify protocol, port, URI segment, domain host, domain name, sub domain, resource URL, extension, mime type, IP address, loopback address, local address, foreign address and resource.

URL : **http://dev.localhost/myproject/admin/p1.php**

**Ans :**

**Protocol :** http://

**Port :** 80 (Unsecured)

**URI segment :** dev.localhost/myproject/admin/p1.php

**Domain host :** localhost

**Domain name :** unavailable

**Sub domain :** dev

**Resource URL :** myproject/admin/p1.php

**Extension :** .php

**MIME type** : .php file

**IP address :** 127.0.0.1

**Loopback address :** 127.0.0.1

**Foreign address :** [::1]

**Resource :** p1.php

**Local address :** DESKTOP-E0AP5IL:0

**Ques : What is MIME type?**

**Ans :** When a file present on any server then we call file extension as MIME type.

**Ques :** How types of files exist in browser?

**Ans :** Two (Renderable and Non-renderable).

**1). MIME type supported :** Occupie Memory

**(i) Renderable :** The file which follows SGML standard or WHATWG standard.

SGML : HTML4

WHATWG : HTML5

file:///c:/Users/Deeproshan%20Kumar/Desktop/test/index.html

Protocol : File

**(ii) Non-renderable :** The file which don't follow SGML or WHATWG standard will be loaded as plain text (as raw file) in that case we -need to add handlers to handle such type of file.

**file:///c:/Users/Deeproshan%20Kumar/Desktop/test/index.html**

**Protocol :** File

**2). MIME type not supported** : Doesn't occupy memory direct download.

**GETTING STARTED WITH PHP**

Goto - C:/laragon/www

**Step 1** : Create folder

**mkdir** app-2021

**Step 2** : Goto inside app-2021 folder

**cd** app-2021

**Step 3** : How to create PHP file by using CMD?

**start notepad p1.php**

**Step 4** : How to run php file?

**php <filename.php>**

There are two possible cases.

**Case 1** : You will get an output.

**Case 2** : PHP not recognized as internal or external command.

You have written the code of PHP----> Execute PHP interpreter

PHP interpreter ----> laragon

1). Set the environment variable

2). Use the laragon inbuilt terminal

**How many ways we can run PHP programs :**

**1). Using Laragon Terminal**

**Launch** laragon terminal

**Goto** file

**Type** : PHP filename.php

**Output** : Terminal ----> PHP CLI mode

**2). Using CMD/git bash...**

**Goto** file

**Type** : PHP filename.php

**Output** : Terminal ----> PHP CLI mode

**3). Using PHP Inbuilt Server**

**CLI** server

**Goto** file

**Type** : **php -S localhost:<port-no>** (Where, -S is command line argument)

**Goto** browser

Make a request

**Localhost**:<port-no>

**Type** : URL : filename.php

**URL** : localhost : <port-no>/file-name.php

**Output** : Web ----> PHP CLI web mode

**Note** : In this case you don't need to enable laragon server.

In version **5.6.2** PHP supports inbuilt server.

Press **Ctrl+c** to stop running server.

**4). Laragon (Server) Web Interface**

**Laragon Server** : ON

**PHP** : Apache (ON)

**MySQL** : ON

Make a request using browser for **http://localhost:<apache-port>**

OR

Quick Open (By using CMD)

**Window+R**

Type **cmd** and hit enter.

**How many ways exists to execute PHP code?**

**1). PHP CLI mode,**

(i) Laragon terminal

(ii) Git bash

(iii) CMD/Powershell

(iv) PHP interactive mode ---> php REPL php (>)

**How to launch PHP REPL mode?**

cmd:>

**php -a** (Where, -a is command line argument)

php > Without using opening and closing of PHP tags but ; is mandatory.

**WAP a program in php interactive mode.**

php > $x="Deeproshan";

php > echo $x;

Deeproshan

When we supply these arguments, the program starts executing in the same, given environment.

**How to see list of all arguments?**

\*\*\***cmd : > php -help**\*\*\*

**php -a** : Run as interactive shell

**php -n** : No configuration (ini) files will be used

**php -i** : PHP information

**php -r** : Execute the code as readline

**php -v** : display the current version of PHP

**php --version** : display the current version of PHP

**php -s** : Run the inbuilt CLI server of PHP <@v php 5.6.2>

**php -h** : Run the helps

**php -m** : Loads the important modules initialized by php.ini

**WAP in PHP to run PHP code as readline.**

We will use **php -r** command to run the code in readline.

cmd : >**php -r "echo 'My name is Deeproshan';"**

My name is Deeproshan

**2). PHP Web Mode**

(i) PHP CLI Server **(php -S <url> : <any-port-no>**

(ii) Laragon Web Interface

**We have only two types of program in any programming language :**

1). With input

2). Without input

**How to create our first program with input in web interface?**

1). Create a HTML form

2). Add any number of fields

3). Add a submit button

4). Add form tag with action, **method="GET"** or **method="POST"**

5). Add the server side page URL which is going to handle the request at Server-side.

6). Fill the form and hit on submit button.

7). You will get a page not found 404 error. Because resource doesn't exist.

8). Create a server-side page, which can handle, client request and process data and give the response.

9). Use the super global variable of PHP to accept the request using **$\_REQUEST** to handle the form fields.

10). Since, **$\_REQUEST** is a Array, it has a key as name field attribute, in HTML form and use echo statement to display the output and after words comment it.

11). Note that, add **id** and **name** attribute on HTML forms to handle each form controls on server side.

**Concept** : This concept is valid for all language JSP, Python, asp.net and PHP also.

ID ---> For JavaScript or client side.

Name ---> For server side

**Client Side :** Front-end language (HTML, CSS, CSS3, Less, SASS, Vanilla JS, jQuery,Bootstrap, Tailwind CSS, SVG, Images, Fonts, Icons, Widgets, Layouts, Other framework of JS (React JS, Angular JS, Backbone JS, Ember JS, NextJS, Redux, Vue Js...).

**Server Side :** Back-end Language including database (C++, GO lang, Rust, C, PHP, Python, Ruby, R, Node.js, ExpressJs, Java(JSP, Servlet), ASP.net, Scala, Perl,..)

**Database :**

**SQL Based Database : (Structured Database)**

Oracle

MySQL

SQLite

MariaDB

MS SQL

**No-SQL Database : (Unstructured Database)**

MongoDB

Dynamo DB

CouchDB

Candendra

Redis

PostgreSQL

Firebase

S3 (Simple Storage Service) Bucket (AWS)

**USING CLIENT SIDE VALIDATION ON PHP**

**What is validation and types of validation?**

1). Validating or restricting the user form entering the invalid data in the form field is called as validation.

**Most confusing question :**

**Ques : What is difference b/w validation, authentication, authorization?**

Ans : Validation will prevents server from providing invalid data or uploading illegal files format in the database on server by the stupid user.

Note : In the case of validation process there is no involvement of database there can be exist only logic.

For example :

Enter the name : 7271053944 (invalid)

Upload Pic : Video.mp4 (invalid)

Both side validation (Client side and Server side) should be done, because there may be chance that user may have blocked his javascript or disabled the javascript.

But if we check the format of email using regex (Regular Expression) then it's validation.

**Ques : Making an email id unique for each registration is example?**

Ans : Authentication (Because user email id will be checked from databse).

**Authentication :** Cross checking the identity of existing user from database is called as authentication.

Such type of feature is used in case of Login, Email verification, Unique email, Forget password, Google login, Facebook login, Linkedin Login, Github login, OTP verification etc.

**Ques : Captcha is which type of validation?**

**Ans :** Completely Automated Public Turing Test to Tell Computers and Humans Apart.

Captcha is always server side because it involves session storage which is a process at server side only. And data of session is always encrypted.

**Authorization :** Authorization is process of after authentication. It refers to permission given to authenticated user, once the user is valid then it may have few permission or may have any permission. This concept is authorization.

For example : Register ---> Logged In ---> Authenticated ---> Logged In (But unable to see any contents) ---> Means you are not authorized person.

**Types of Authorization :**

1). No Authorization (User)

2). Fully Authorization (Super Admin)

3). Partially Authorization (Admin)

Where, User, Super Admin and Admin are roles. Roles and permission is kind of authorizations.

**Example :**

Trial Version | Premium Version

10 Features | 20 Feautures

Non-authorized Authorized

**Example :** School ERP (Enterprise Resource Planning) :

1). Teacher

| Add Attendance

| View Attendance

2). Student

| View Attendance

Here, Student can't add attendance because student is not authorized for add attendance option.

**WORKING WITH PHP CLI**

1). Hello World (Without Input)

2). Addition of two numbers (With Input)

In order to take input on console window or CLI mode we need a scanner.php file where we define input() function.

**Scanner.php file**

<?php

// Scanner Function

function input($msg=' '){

if($msg==' '){

exit('Message is required');

}

echo $msg;

$input=fgets(STDIN,1024);

return trim($input);

}

?>

**Input() function (Inside scanner.php) :**

In order to understand this how input() function works following concepts are required.

1). Use of echo statement

**Ques : Do PHP supports inbuilt console input function?**

**Ans :** Yes, readline() is pre-defined function used to take input.

**Ques :** WAP in PHP to perform addition of two numbers using readline() method in CLI mode.

**Ans :**

<?php

$num1=readline('Enter the first number : ');

$num2=readline('Enter the second number : ');

$sum=$num1+$num2;

echo $sum;

?>

**Output :**

Enter the first number : 30

Enter the second number : 50

The sum is : 80

**Use of echo Statement :**

We can echo statement for printing the string value, or any variable value or any constant literal.

php> echo 'Hello World';

Hello World

php> echo "Hello World";

Hello World

php> echo 10; // Constant literal

10

php> $x=10;

10

php> echo true; // Constant literal

1

php> echo false // Constant literal

php> echo [10,20,30]

Array to string conversion error.

**How to print array with echo :**

php> echo [10,20,30];

Notice : Array to string conversion in php shell code on line 1

Array

php> echo join([10,20,30]);

102030

php> echo implode('',[10,20,30]);

102030

php> echo null;

php> echo json\_encode(null);

null

php> echo (null==false);

1

php> echo (null===false);

Boolean false

php> echo (0=='')

1

php> echo (0==null);

1

php> echo (0==false)

1

**Note 1 ::** null==0==false==''==true==1

**Note 2 ::** echo will evaluate the expression as a string, but in the case of === we will see difference.

Lower conversion (int) + Upper conversion (string)

php> echo json\_encode('');

""

php> echo json\_encode(false);

false

php> echo json\_encode(EOL);

"\r\n"

php> echo getType(); // Data type check

**Ques 1 :** $x=(false=='');

**Ans :** 1

**Ques 2 :** $x=(false==='');

**Ans :**

**2). Driver protocol [protocol]://**

PHP supports list of driver protocols.

These are also called as wrapper, or scheme in the PHP.

file:// — Accessing local filesystem

http:// — Accessing HTTP(s) URLs

ftp:// — Accessing FTP(s) URLs

php:// — Accessing various I/O streams

zlib:// — Compression Streams

data:// — Data (RFC 2397)

glob:// — Find pathnames matching pattern

phar:// — PHP Archive

ssh2:// — Secure Shell 2

rar:// — RAR

ogg:// — Audio streams

expect:// — Process Interaction Streams

php:// - It can handle any kind of input and output from any interface it can be terminal, it can be UI based, or any third party API.

PHP provides a number of miscellaneous I/O streams that allow access to PHP's own input and output streams, the standard input, output and error file descriptors, in-memory and disk-backed temporary file streams, and filters that can manipulate other file resources as they are read from and written to.

The meaning of STDIN : php://input (Standard Input)

The meaning of STDOUT : php://output (5.6.2) or php://memory<(5.6.-) (Standard Output)

The meaning of STDERR : Standard Error

**php://stdin, php://stdout and php://stderr**

php://stdin, php://stdout and php://stderr allow direct access to the corresponding input or output stream of the PHP process. The stream references a duplicate file descriptor, so if you open php://stdin and later close it, you close only your copy of the descriptor-the actual stream referenced by STDIN is unaffected. It is recommended that you simply use the constants STDIN, STDOUT and STDERR instead of manually opening streams using these wrappers.

php://stdin is read-only, whereas php://stdout and php://stderr are write-only.

**php://input**

php://input is a read-only stream that allows you to read raw data from the request body. php://input is not available with enctype="multipart/form-data".

**php://output**

php://output is a write-only stream that allows you to write to the output buffer mechanism in the same way as print and echo.

**Note :** These are used for raw input only (JSON input) from client request.

**What is fgets()?**

fgets() allows you to read the data from any input resource. It internally uses, file pointer. By default it can read 1024 characters (doesn't recognized EOL (End of line) at a time from input buffer.

1). fgets() : Terminal read

fgets(STDIN,1024);

2). fgets() : Read file

The fgets() can read any numbers of characters from any file, or resource before EOF.

**Case 1** : fgets accepts two arguments :

1. resource handler this can be a wrapper, scheme, file pointer.

2. buffer\_size : no. of memory blocks reserved for, input characters stream.

It works like a link list

where,

0 : Address

1 : Base Data : Header Data : Header Node

2 : Other Nodes

if buffer\_size not supplied : default 1024

if buffer\_size == 0 : false

if buffer\_size == 1 : false

if buffer\_size > 1 : Character bit reserved in memory block headed (pointed by) by base node.

**3). Concept of user-defined function :**

Syntax to define user-defined function ---

function <function-name>($args1, $args2, $args3,...){

// Code to be executed

}

function wish(){

echo "Hii ! Good Morning...";

}

**WAP in PHP to add two number using user-defined function.**

<?php

$num1=readline("Enter the first number : ");

$num2=readline("Enter the second number : ");

echo add($num1,$num2);

function add($num1,$num2){

return $num1+$num2;

}

?>

**Output :**

Enter the first number : 20

Enter the second number : 10

30

<?php

$num1=readline("Enter the first number : ");

$num2=readline("Enter the second number : ");

#echo add($num1,$num2);

function add($num1=0,$num2=0){

return $num1+$num2;

}

printf("The result with 0 argument : %d\n",add());

printf("The result with 1 argument : %d\n",add($num1));

printf("The result with 1 argument : %d\n",add($num2));

printf("The result with 0 argument : %d\n",add($num1,$num2));

?>

**Output :**

Enter the first number : 10

Enter the second number : 20

The result with 0 argument : 0

The result with 1 argument : 10

The result with 1 argument : 20

The result with 0 argument : 30

**4). Return value of function :**

Using return keyword inside function the function should return any value we have already seen in above add() function.

**5). Default argument of function :**

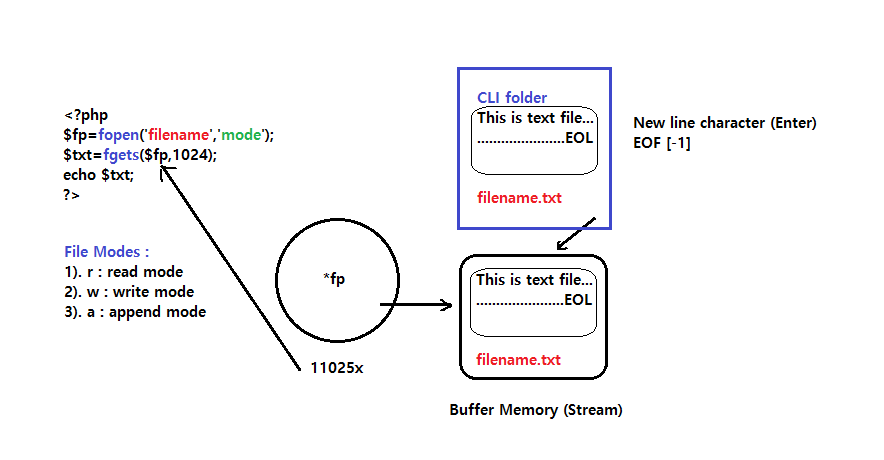
If we want to make argument optional (can pass or don’t pass) in that case define a default value to the argument in function declaration and definition.

function add($a, $b){

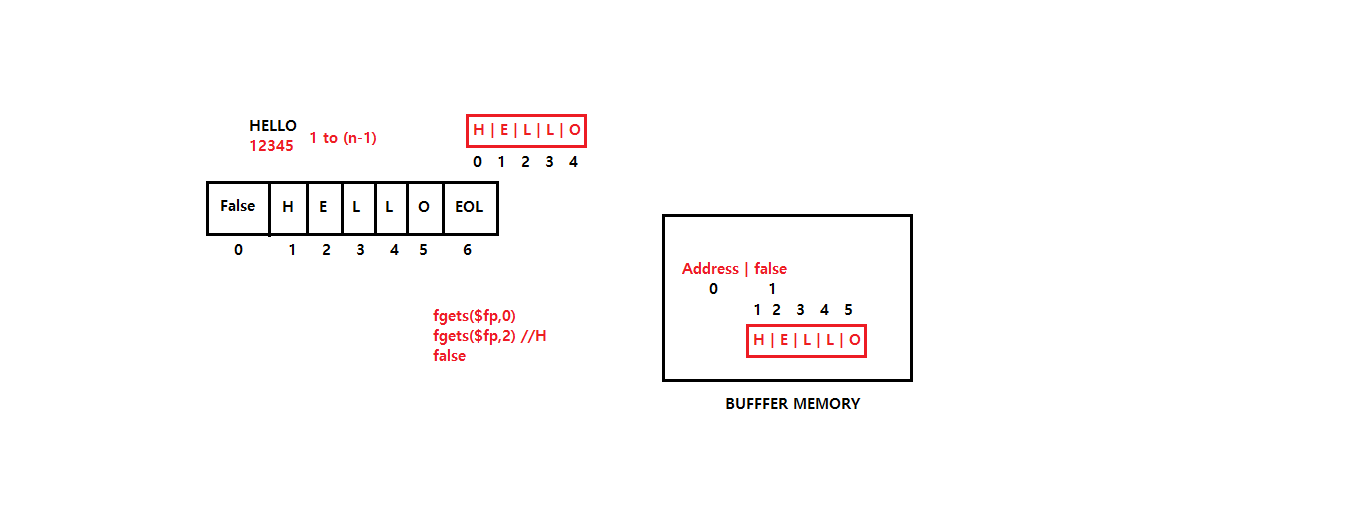
}

**$a :** **Mandatory Argument :** The arguments which are compulsory.

**$b :** **Optional Argument :** The optional are arguments have a default value which makes them optional such that if we pass the value $argument takes the assigned value. If not it is going to use the default value. This makes these argument (parameters, inputs) as optional argument.

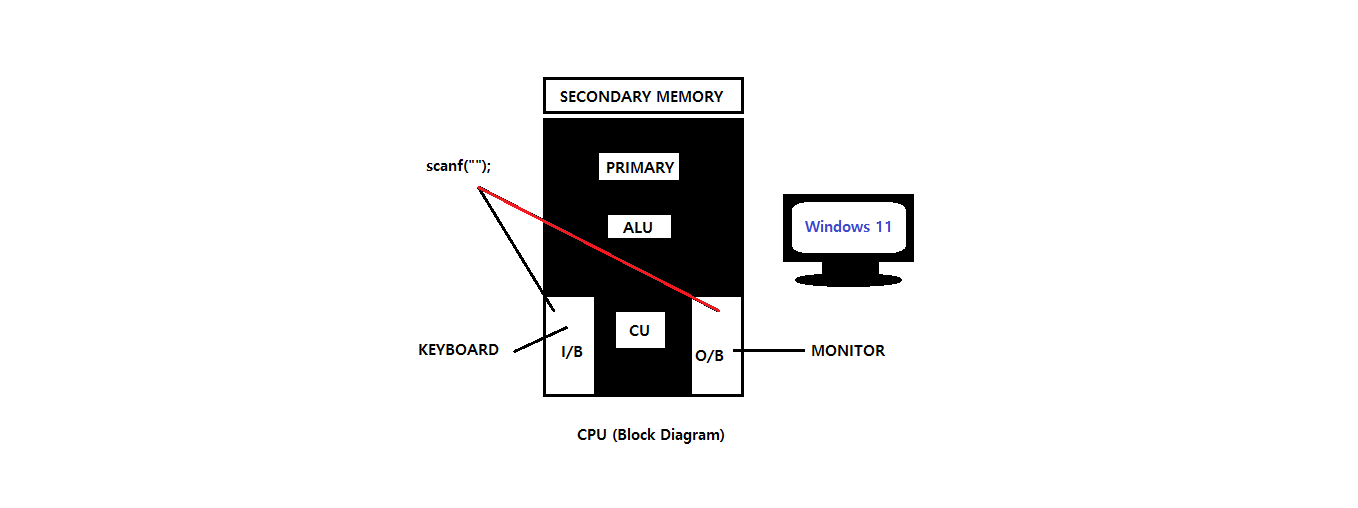
**6). File-pointer and file handling :** File pointer is a special types of variable that store reference to memory allocation of any file. $fp=fopen('filename.txt','mode') where $fp acts as a file pointer which is pointing to the reference address of anyfile.txt. $fp is also known as resource handler.

**File handling :** Performing I/O operation on the file, to store the data permanently in the file system of OS is called file handling.

**7). I/O buffer or I/O streams :** Whenever we are going to take input from standard input devices. It is electrically converted into digital signals and stored in buffer by BIOS which can be used by kernel (Core of OS) to internally use this input and output values.

**I/B : Input Buffer** - Stores all binary form of input taken from input device which are digitally converted from an analog signal in form of current.

**O/B : Output Buffer** - Stores all binary form of output and calculations calculated by ALU and sent to primary memory which is stored at output buffer which can be transferred to, any output device.

**8). Use of trim() function** - It truncates the extra spaces and new line characters from a input string.

<?php

$name=" Deeproshan Kumar ";

echo $name;

echo PHP\_EOL;

printf("The length of the name is : %d",strlen($name));

$clean\_name=trim($name);

printf("The length of the name is : %d",strlen($clean\_name));

?>

**VARIABLE ($) & REFERENCE TO VARIABLE ($$) OR VARIABLE TO VARIABLE REFERENCE IN PHP**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A variable is a kind of identifier used for storing the constant literals. A variable is also a reference to memory.

**Identifier :** The name given to anything is called as identifier.

**Example :** It can be variable name, array name, function name, file name, class name, ... etc. User defined naming convention is known as identifier.

**Ques 1 :**

class Test{

private $a;

private $b;

public function dance($msg){

return $msg;

}

}

**How many identifiers are there?**

5 Identifiers

**Syntax :** $<variable-name>=<any-value>;

$x; // Valid

echo $x; // Error : Undefined Index $x;

**Rules to make an identifier :**

1). total // valid

2). total\_ // valid

3). total123 // valid

4). 123total //invalid

5). \_123total //valid

6). \_total // valid

7). \_123 // valid

8). total\_123 // valid

9). total 123 // invalid

10). is\_passed // valid

11). is passed // invalid

12). \_Total // valid

13). TOTAL // valid

14). TOTAL\_ // valid

**KEYPOINTS**

(i) alphanumeric\_(underscore) is valid, but numericalpha\_(underscore) is invalid.

(ii) Allowed characters : a-z, A-Z, \_, 0-9[1,n]

(iii) Special characters can't be used instead of $ and $$ but it's only at 0th index for variable naming convention.

(iv) In PHP variable names are case sensitive, means $name and $Name are two different different variables.

A variable can hold any type of value depending on the data-type.

**For Example :**

1). A variable can hold numerical values, like int, float.

2). A variable can hold a Boolean value like true and false.

3). A variable can hold a collection of value like array.

4). A variable can hold a value like empty or null value.

5). A variable can hold Resource like mysqli resource object.

6). A variable can hold a string value B/W ‘’ and “”, heredoc, newdoc strings.

7). A variable can hold an object or instances.

8). A variable can hold context of super globals (also called as super global variable).

**Example 1 :**

<?php

// Making of numerical variable

<?php

// Making of numerical variable

$x=10; // **declaration+initilization**

echo $x; // echo is **predefined language construct**

echo PHP\_EOL; // **predefined constant**

echo getType($x); // **predefined function**

?>

**Output :**

10

integer

**Example 2 :**

<?php

// WAP in PHP to show floating point numbers.

$x=011; // Octal number system

echo $x;

echo PHP\_EOL;

#$a=08; // Invalid numeric literal (Conversion to octal range not possible)

#echo $a;

?>

**Output :**

9

**Example 3 :**

<?php

// wap in php to boolean data types stored in variables.

$x=true;

echo $x;

echo PHP\_EOL;

echo getType($x); // boolean

echo PHP\_EOL;

var\_dump($x); // bool(true)

echo PHP\_EOL;

$z=TRUE;

echo $z;

echo PHP\_EOL;

echo getType($z);

echo PHP\_EOL;

var\_dump($z);

echo PHP\_EOL;

$y=TrUe;

echo $y;

echo PHP\_EOL;

echo getType($y);

echo PHP\_EOL;

var\_dump($y);

echo PHP\_EOL;

echo 'On comparing all the three values :';

echo PHP\_EOL;

echo (true==TRUE);

echo PHP\_EOL;

echo (True==TRUE);

echo PHP\_EOL;

echo (1==true);

echo PHP\_EOL;

var\_dump (1===true);

?>

**Output :**

1

boolean

bool(true)

1

boolean

bool(true)

1

boolean

bool(true)

On comparing all the three values

1

1

1

**Note : :** What is difference between json\_encode and json\_decode?

**json\_encode** : php native data type to json native representation. The data representation is completely raw.

**json\_decode** : json representation converts to php native data type. Raw data-type to php data type.

<?php

// wap in php to show boolean false stored in variable.

$x=false;

echo $x;

echo PHP\_EOL;

echo getType($x); // boolean

echo PHP\_EOL;

var\_dump($x); // bool(false)

echo json\_encode($x); // raw format

echo PHP\_EOL;

echo 'On comparing the values : ';

echo PHP\_EOL;

echo (false==False);

echo PHP\_EOL;

echo (false==FALSE);

echo PHP\_EOL;

var\_dump (0==FALSE); // Type is optional only content match

echo PHP\_EOL;

var\_dump (0==false); // var\_dump (0==(int)false); // Implicit conversion // PHP dynamic

echo PHP\_EOL;

var\_dump (0===false); Type + Content will be matched together

?>

**Output :**

boolean

bool(false)

false

On comparing the values :

1

1

bool(true)

bool(true)

bool(false)

**Example :**

<?php

// WAP in PHP to show how a variable can hold a variable value.

$arr=array(10,20,30,'red','green');

echo $arr; // Array to string conversion error

echo PHP\_EOL;

print\_r($arr); // r-resource

echo PHP\_EOL;

var\_dump($arr)

?>

**Output :**

Notice: Array to string conversion in C:\laragon\www\app-2021\variables\p6.php on line 4

Array

Array

(

[0] => 10

[1] => 20

[2] => 30

[3] => red

[4] => green

)

array(5) {

[0]=>

int(10)

[1]=>

int(20)

[2]=>

int(30)

[3]=>

string(3) "red"

[4]=>

string(5) "green"

}

**Note ::** We can use print\_r() or var\_dump() to print output of any array.

**print\_r() :** Prints the array or any resource in readable format it only show the data inside it.

**var\_dump() :** It will dump the entire data about the variable hence var\_dump() shows the data or data, called meta data.

**var\_export() :** Alias of var\_dump() same as var\_dump() but it shows the representation of the data in more readable not metadata.

**Example :**

<?php

// wap in php to assign the array to variable.

$color[0]='red';

$color[1]='green';

$color[2]='orange';

$color[3]='black';

echo getType($color);

echo PHP\_EOL;

print\_r($color);

$color1[]='RED';

$color1[]='GREEN';

$color1[]='ORANGE';

$color1[]='BLACK';

$color1[]='WHITE';

echo PHP\_EOL;

print\_r($color1);

?>

**Output :**

array

Array

(

[0] => red

[1] => green

[2] => orange

[3] => black

)

Array

(

[0] => RED

[1] => GREEN

[2] => ORANGE

[3] => BLACK

[4] => WHITE

)

<?php

// WAP in PHP to show nullable and empty variable.

$x='';

var\_dump($x);

echo PHP\_EOL;

var\_dump($x=='');

echo PHP\_EOL;

var\_dump($x=="");

echo PHP\_EOL;

var\_dump($x==='');

echo PHP\_EOL;

var\_dump($x==="");

echo PHP\_EOL;

var\_dump(empty($x));

echo PHP\_EOL;

var\_dump(empty(''));

echo PHP\_EOL;

var\_dump(empty(""));

$y="";

var\_dump($y);

echo PHP\_EOL;

var\_dump($x==$y);

echo PHP\_EOL;

var\_dump($x===$y);

?>

**Output :**

string(0) ""

bool(true)

bool(true)

bool(true)

bool(true)

bool(true)

bool(true)

bool(true)

string(0) ""

bool(true)

bool(true)

**Note :** Length of empty=0 and data type = string, ASCII value=0.

**EMPTY V/S SPACE**

**Example :**

<?php

echo 'Be aware b/w empty and <space>';

echo PHP\_EOL;

printf("The value from empty:",'');

echo PHP\_EOL;

printf("The value from empty space : ",' ');

echo PHP\_EOL;

var\_dump(''==' ');

var\_dump(''===' ');

echo PHP\_EOL;

echo '';

echo PHP\_EOL;

echo getType('');

echo PHP\_EOL;

echo strlen('');

echo PHP\_EOL;

echo ' ';

echo PHP\_EOL;

echo getType(' ');

echo PHP\_EOL;

echo ord(' '); // Prints ASCII

echo PHP\_EOL;

echo ord('a'); // Prints ASCII

?>

**Output :**

Be aware b/w empty and <space>

The value from empty:

The value from empty space :

bool(false)

bool(false)

string

0

string

32

97

**Example :**

<?php

// WAP in PHP to show nullable data type.

$x=null; // Doesn't exist

echo getType($x);

echo PHP\_EOL;

var\_dump(is\_null($x));

var\_dump(empty($x));

var\_dump($x==null);

var\_dump($x===null);

var\_dump(''==null); // Implicit typecast have done.

var\_dump(''===null);

echo 'Very very important';

echo PHP\_EOL;

$y='';

var\_dump($y);

var\_dump(isset($y));

var\_dump(isset($w));

var\_dump(empty($x));

var\_dump(isset($x));

var\_dump($w);

var\_dump($x);

?>

**Output :**

NULL

bool(true)

bool(true)

bool(true)

bool(true)

bool(true)

bool(false)

Very very important

string(0) ""

bool(true)

bool(false)

bool(true)

bool(false)

Notice: Undefined variable: w in C:\laragon\www\app-2021\variables\p10.php on line 20

NULL

NULL

**Ques : What is the major difference b/w null and undefined variable in PHP?**

**Ans :** $x=’a’; isset($x); ---> true

echo $x; prints ---> a

If you try these things with a undefined variable then,

$y;

echo $y; // undefined variable

Since you are printing this with echo statement you will see nothing.

var\_dump($y); // Some error, undefined variable $y but data type is NULL.

Isset($); false ---> valid ---> $y doesn’t exist at all ---> $y is given memory.

$z=null; ---> // $z ---> Memory allocated

echo $z; // No error ---> No output will be generated.

Var\_dump($z); // NULL with no error

isset($z); // False here $z is allocated into memory without any value.

unset($y); // Deletes from memory

echo $y;

unset($z); // Deletes from memory

echo $z;

var\_dump($y); // Undefined variable ---> NULL

var\_dump($z); // No error ---> NULL

unset($y);

unset($z);

var\_dump($y); // Undefined variable ---> NULL

var\_dump($z); // Undefined variable ---> NULL

**Example :**

<?php

// WAP in PHP to show the difference b/w null and undefined.

$x='a'; // Memory assigned

echo $x;

echo PHP\_EOL;

var\_dump($x);

var\_dump(isset($x));

echo PHP\_EOL;

printf("The memory x=%d", memory\_get\_usage($x));

unset($x);

printf("The memory x=%d", memory\_get\_usage($x));

echo PHP\_EOL;

$y;

echo PHP\_EOL;

var\_dump($y);

var\_dump(isset($y));

echo PHP\_EOL;

printf("The memory y=%d", memory\_get\_usage($y));

unset($y);

printf("The memory y=%d",memory\_get\_usage($y));

$z=null; // Memory

echo $z;

echo PHP\_EOL;

var\_dump($z);

var\_dump(isset($z));

echo PHP\_EOl;

printf("The memory z=%d",memory\_get\_usage($z));

unset($z);

printf("The memory z=%d",memory\_get\_usage($z));

?>

**Output :**

a

string(1) "a"

bool(true)

The memory x=2097152

Notice: Undefined variable: x in C:\laragon\www\app-2021\variables\p11.php on line 11

The memory x=396272

Notice: Undefined variable: y in C:\laragon\www\app-2021\variables\p11.php on line 15

NULL

bool(false)

Notice: Undefined variable: y in C:\laragon\www\app-2021\variables\p11.php on line 18

The memory y=396272

Notice: Undefined variable: y in C:\laragon\www\app-2021\variables\p11.php on line 20

The memory y=396272

NULL

bool(false)

Warning: Use of undefined constant PHP\_EOl - assumed 'PHP\_EOl' (this will throw an Error in a future version of PHP) in C:\laragon\www\app-2021\variables\p11.php on line 26

PHP\_EOlThe memory z=396272

Notice: Undefined variable: z in C:\laragon\www\app-2021\variables\p11.php on line 29

The memory z=396272

**Resources :**

<?php

$conn=mysqli\_connect(‘localhost:3306’, ‘root’, ‘’, ‘’);

var\_dump($conn); #Object ---> Resource

?>

**Object :**

<?php

// WAP in PHP to show object using class.

class Student{

private $name=’Deeproshan’;

private $class=’BTech’;

private $rollno=’1001’;

}

$std= new Student();

var\_dump($std); // User-defined ---> Type(Resource)

?>

**Output :**

object(Student)#1 (3) {

["name":"Student":private]=>

string(16) "’Deeproshan’"

["class":"Student":private]=>

string(11) "’BTech’"

["rollno":"Student":private]=>

string(10) "’1001’"

}

**SUPER GLOBAL VARIABLES**

These are special with global context and scope these predefined variable by PHP and it has been reserved for some special purpose.

Super global variable are of two types :

1). Mingled Super Global Variable

2). Non-mangled Super Global Variable

**Super global variable in general terms :**

$\_GLOBALS (Non mangled)

$\_SERVER (Mangled)

$\_GET (Mangled)

$\_POST (Mangled)

$\_FILES (Mangled)

$\_COOKIE (Mangled)

$\_SESSION (Mangled)

$\_REQUEST (Mangled)

$\_ENV (Mangled)

**Other Super Global Variables :**

$http\_response\_header : Non-mangled

$argc : Non-mangled (Argument Count)

$argv : Non-mangled (Argument Variable)

$php\_errormsg : Deprecated

**Note ::** Non mangled : Which starts with $ and \_ as prefix this naming convention is called mangling.

Same thing is also valid in Python : \_\_name\_\_, \_\_main\_\_\_, \_\_init(self)\_\_

In Javascript : \_\_proto\_\_

1). Mangled Super Global Variable are the variable whose naming convention is such that, they are declared using $ but have \_at prefix.

For eg : Normal variable $name, $class, $\_SERVER, $\_REQUEST, $\_GET, $\_POST

2). Non-mangled Super Global Variable are those variable whose naming convention are like normal variables and use no special prefixing like \_ or \_\_ the can be treated as normal variable in case naming convention only.

For eg : $name, $class, $marks,, $GLOBALS, $argc, $http\_response\_header, $php\_errormsg

**What is the difference between normal variable and super global variable?**

1). Super Global Variables are pre-defined but normal variables are user-defined.

2). Super Global Variable have global scope and normal variable have local scope.

3). Normal variable don’t have any concept of name mangling but super global variable have.

**$\_GLOBALS** : BOTH SAPI (CGI+CLI)

**$\_SERVER** : BOTH SAPI (CGI+CLI) But Environment Dependent.

**$\_GET** : SAPI (CGI)

**$\_POST** : SAPI (CGI)

**$\_FILES** : SAPI (CGI)

**$\_COOKIE** : SAPI (CGI)

**$\_SESSION** : SAPI (CGI)

**$\_REQUEST** : SAPI (CGI)

**$\_ENV** : SAPI (CGI+CLI)

Other Super Global Variable :

**$http\_response\_header** : SAPI (CGI)

**$argc** : CLI

**$argv** : CLI

**$php\_errormsg** : BOTH (CGI+CLI)

**What is SAPI?**

SAPI stands for server/system Application Programming Interface.

It a type of Service, Which recognizes, the environment in which execution of the program takes place.

This service is, handled by Web-Server.

**php\_sapi\_name()** – Returns the type of interface between web server and PHP.

**WAP in PHP to show different sapi name at different environment.**

**Ans :** Consider sapi.php

<?php

echo php\_sapi\_name();

?>

1). php sapi.php // cli

2). php –S localhost:8000 // cli-server

3). Apache (laragon) 🡪 <http://localhost:786/> //apache2handler

**Ques 1 :** What is sapi of $name=’Deep’?

**Ans :** Both (CLI & CGI)

**Ques 2 :** What is correct value for $argc when you run it at url :

<http://localhost:786/p1.php>

**Ans :** 0

**COMMAND LINE ARGUMENTS IN PHP**

In any programming language there are two types on input environment.

**1). RUN TIME INPUT** : Inputs and arguments supplied during program execution is called as run time argument.

For example : scanf(), getch(), getche(), gets(), getchar()…

You can take input from these function, or you can supply the input, in these only when program is running in primary memory.

**2). COMPILE TIME INPUT OR ARGUMENTS :** The arguments supplied at compile time of the program is called command line arguments.

In PHP we have two super global variable for this :

1). $argc : Argument Count

2). $argv : Argument Vector / Variable

These will work under only sapi mode=’CLI’.

**Example :**

<?php

print\_r($argv);

echo PHP\_EOL;

echo $argc;

echo PHP\_EOL;

echo getType($argc);

echo PHP\_EOL;

?>

**Output :**

Array

(

[0] => p2.php

)

1

integer

**The Properties of Command Line Argument :**

1). The Command Line Arguments are that arguments which are supplied at compile time.

2). $argv : Argument vector is responsible for accepting the arguments at compile time.

3). $argc : Holds the no. of counts or no. of variable arguments supplied.

4). The type of $argc : Integer

5). The default size of $argc is : 1

6). The size of $argc = (n+1) where n is the no. of arguments.

7). $argc and $argv both work only SAPI == CLI mode.

8). $argv and $argc are super global variable according to PHP standards. In, the super global variable they also belongs to category of non-mangled super global variable.

9). $argv is a homogeneous array.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Note ::** In general in any programming language, array are collection of constant literals.

It has following classification:

**1). ON THE BASIS OF DATA**

(i) Same Type : Homogeneous Data

(ii) Different Type : Heterogeneous

(iii) Mixed Type : Heterogeneous

**2). ON THE BASIS OF DIMENSION**

(i) 1D Array (Single-dimensional Array)

(ii) 2D Array (Multi-dimensional Array)

(iii) 3D Array (Multi-dimensional Array)

(iv) ND Array (Multi-dimensional Array)

**3). ON THE BASIS OF STRUCTURE/ NATURE.**

(i) Indexed Array

(ii) Mapped Array (Key and Value Pair)

**Note ::** There can be n of names, for the mapped array.

For example,

Python ---> Dictionary

Java ---> HashMap

C# ---> HashArray

JavaScript ---> ArrayOfObject

PHP ---> Associative Array

C++ ---> VectorArray

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**$argv :** $argv will be always homogeneous.

**WAP in PHP to check an array is homogeneous or heterogeneous.**

<?php

// WAP in PHP to check an array is homogeneous or heterogeneous.

function is\_homo($array=[]){

$key\_type=getType($array[0]);

for($i=0; $i<count($array); $i++){

// echo getType(array[$i]);

// echo PHP\_EOL;

if($key\_type===getType($array[$i])){

}else{

return false;

}

}

return true;

}

var\_dump(is\_homo($argv));

?>

**Output :**

bool (true)

bool (true)

10). $argv has homogeneous type as string Array, i.e. is why it is also, string vector.

11). The $argv supports SSV only (Space Separated Values).

12). The data type of each element either float, int, Boolean, array everything is converted into string only.

13). If you supplied space then we need to enclosed that in “” otherwise it will consider all space separated values as different arguments.

14). The index o is always file name.

15). Even though PHP is a dynamic type language so in command line arguments it is stricted to string type because 0th element index is a filename which is a string value, due to which the entire elements of different data type are implicitly converted into string data type.

**// WAP in PHP to find sum of all the arguments supplied.**

<?php

// WAP in PHP to find sum of all the arguments supplied.

print\_r($argv);

$sum=0;

$avg=0;

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

$sum=$sum+(int)$argv[$i];

}

$avg=$sum/($argc-1);

printf("The sum = %d\n",$sum);

printf("The average = %d\n",$avg);

?>

**Output :**

Array

(

[0] => p3.php

[1] => 10

[2] => 20

[3] => 30

)

The sum = 60

The average = 20

**ANONYMOUS SCRIPT/ FILE :** A file without extension are called as Anonymous file, and such type of file recognized as FILE or <FILENAME Type>.

**Why this concept is required?**

CodeIgnitor 4 : spark ---> php file

Laravel : artisan ---> php file

Zwave : zwave ---> php file

This is used so that we can hide our real code without allowing Operating System, to identify the file type.

**WAP in PHP to make an anonymous script?**

<?php

echo "This is Deeproshan's Terminal";

echo PHP\_EOL;

echo 'The Deeproshan Framework : Version 1.x';

echo PHP\_EOL;

$commands=[

'--h' => 'This is help command',

'--v' => '1.x',

'--a' => 'for Addition',

'--s' => 'for Subtraction',

'--m' => 'for Multiplication',

'--d' => 'for Division',

'--r' => 'for Remainder'

];

echo PHP\_EOL;

foreach($commands as $key => $value){

printf("%s\t\t: \t\t%s\n",$key,$value);

}

if(isset($argv[1])){

$command=$argv[1];

echo "\e[H\e[J";

switch($command){

case '--v':

case '--version':

$output = '1.x';

break;

case '--help':

case '--h':

$output='This is help command.';

break;

case '--a':

$x=readline('Enter the number 1 : ');

$y=readline('Enter the number 2 : ');

$output=$x+$y;

break;

case '--s':

$x=readline('Enter the number 1 : ');

$y=readline('Enter the number 2 : ');

$output=$x-$y;

break;

case '--m':

$x=readline('Enter the number 1 : ');

$y=readline('Enter the number 2 : ');

$output=$x\*$y;

break;

case '--d':

$x=readline('Enter the number 1 : ');

$y=readline('Enter the number 2 : ');

$output=$x/$y;

break;

case '--r':

$x=readline('Enter the number 1 : ');

$y=readline('Enter the number 2 : ');

$output=$x%$y;

break;

default :

$output='Invalid Command';

break;

}

}

echo $output;

?>

**VARIABLE TO VARIABLE REFERENCE IN PHP**

Creating the variable from the reference of existing variable it’s denoted by $$.

This concept can be found in CodeIgnitor, Laravel, YII frameworks where we want to pass the data from controller to view.

**Example :**

<?php

// WAP in PHP to show var-to-var-reference.

$name='Deep';

echo $name;

echo PHP\_EOL;

$$name=500;

printf("The value of Deep=%s",$Deep);

?>

**Output :**

Deep

The value of Deep=500

**Example :**

<?php

// WAP in PHP to show var-to-var referene.

$x='10';

echo $x;

echo PHP\_EOL;

$y='Ravi';

echo $y;

echo PHP\_EOL;

$Ravi=50;

echo $Ravi;

echo PHP\_EOL;

$w='\_100';

$$w='King';

printf("The value of w=%s\n",$w);

printf("The value of King=%d\n",$$w);

printf("The value of \_100=%s\n",$\_100);

$exp='x';

echo $exp;

echo PHP\_EOL;

echo getType($exp);

echo PHP\_EOL;

echo $$exp;

echo PHP\_EOL;

$student['name']='Suraj';

$student['class']='Btech';

foreach($student as $key => $value){

$$key=$value;

printf("%s : %s\n", $key,$value);

}

echo $name;

echo PHP\_EOL;

echo $class;

echo PHP\_EOL:

echo 'Without using variable to variable';

echo PHP\_EOL;

$car['name']='Hyundai';

$car['class']='1st Class';

$car['price']='20 Lakhs';

extract($car);

printf("The name of car : %s\n",$name);

printf("The class of car : %s\n",$class);

printf("The price of car : %s\n",$price);

?>

**Output :**

10

Ravi

50

The value of w=\_100

The value of King=0

The value of \_100=King

x

string10

name : Suraj

class : Btech

Suraj

Btech Without using variable to variable

The name of car : Hyundai

The class of car : 1st Class

The price of car : 20 Lakhs

**CONSTANTS IN PHP**

Constants are that identifier which are responsible to hold single value.

If we try to assign new value it will generate error.

**OR**

The identifier whose value remains constant through out the execution of the program is called as constant.

There are two types of constants :

**1). Pre-define**

In PHP there are 1800+ pre-defined constants.

get\_defined\_constants();

**Example :**

<?php

// WAP in PHP to show total pre-defined constants.

$count=count(get\_defined\_constants());

printf("The total constants=%d",$count);

?>

**Output :**

The total constants=1872

**IMPORTANT PRE-DEFINED CONSTANTS**

**1). PHP\_VERSION :** Returns the version of PHP.

**2). TRUE :** Return 1

**3). FALSE :** Return nothing

**4). NULL :** Return nothing

**5). PHP\_EOL :** \r\n newline

**6). PHP\_SAPI :** Return the SAPI name (cli|cli-server|apache2handler).

**7). PHP\_OS :** Return OS extension.

**8). PHP\_OS\_FAMILY :** Returns OS name.

**9). STDIN :** Returns the resource object of standard input.

**10). STDOUT :** Returns the resource object of standard output.

**11). STDERR :** Return the resource object of standard error.

**ERROR LEVEL CONSTANTS : 16**

[E\_ERROR]

[E\_RECOVERABLE\_ERROR]

[E\_WARNING]

[E\_PARSE]

[E\_NOTICE]

[E\_STRICT]

[E\_DEPRECATED]

[E\_CORE\_ERROR]

[E\_CORE\_WARNING]

[E\_COMPILE\_ERROR]

[E\_COMPILE\_WARNING]

[E\_USER\_ERROR]

[E\_USER\_WARNING]

[E\_USER\_NOTICE]

[E\_USER\_DEPRECATED]

[E\_ALL]

**ERROS TREE**

**ERROR**

**|**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**| |**

**STRICT NON-STRICT (Recoverable Error)**

(Encounter) Warning

1. Compile **1.** Notice
2. Parse **2.** Warning

**<-------------------------- Three Types ----------------------->**

**1).** Core

**2).** User

**3).** Deprecated

**2). User-defined**

User can create their own constants.

**1).** Procedural

**Syntax :** define(‘<identifier-name>’,<value>,<case-sensitive>);

This types of constant have global scope and can be accessed anywhere.

**Example :**

<?php

// WAP in PHP to show procedural pre-defined constant.

define('gravity',10);

define('project\_name','My\_App');

define('DB\_name','test\_db');

echo gravity;

echo PHP\_EOL;

echo project\_name;

echo PHP\_EOL;

echo DB\_name;

?>

**Output :**

10

My\_App

test\_db

**Example :**

<?php

// WAP in PHP to show case sensitive constant.

define('gravity',10); // lowercase

define('Gravity',20); // Capitalize

define('GRAVITY',30); // Uppercase

define('gRAVITY',40); // Toggle Case

echo gravity;

echo PHP\_EOL;

echo Gravity;

echo PHP\_EOL;

echo GRAVITY;

echo PHP\_EOL;

echo gRAVITY;

?>

**Output :**

10

20

30

40

**WAP in PHP to show re-declaration errors in constants.**

<?php

// WAP in PHP to show re-declaration errors in constants.

define('gravity',10);

echo gravity;

echo PHP\_EOL;

define('Gravity',10.5);

echo Gravity;

// re-declaration error.

define('gravity',20);

echo gravity;

?>

**Output :**

10

10.5

Notice: Constant gravity already defined in C:\laragon\www\app-2021\cli\constants\p5.php on line 9

10 // Can’t be changed and appears a notice error.

**Example :**

**WAP in PHP to make non-case-sensitive user-defined error.**

<?php

// WAP in PHP to make non-case-sensitive user-defined error.

error\_reporting(E\_USER\_DEPRECATED);

define('gravity',10,1);

echo gravity;

echo PHP\_EOL;

echo GRAVITY;

echo PHP\_EOL;

echo Gravity;

echo PHP\_EOL;

echo GrAviTy;

?>

**Output :**

10

10

10

10

**2).** OOPS : In case of interface and abstract class.

**PROPERTIES OF CONSTANTS**

* The declaration of constants are atomic (declaration once) and case-sensitive.
* The scope of constants are global.
* The constants can’t be re-declared.
* If we want to make incase-sensitive constants we need to set third argument is define function as true.
* The incase-sensitive, constants were deprecated in version 7.x but this feature is already supported and in 8.x this feature is again supported.

**Note ::** During this concept we have two types of errors occurred 1 : E\_DEPRECATED and 2 : E\_USER\_DEPRECATED.

E\_DEPRECATED is reserved for pre-defined standards of PHP.

E\_USER\_DEPRECATED is reserved for user-defined deprecated error.

**WORKING WITH STRING LITERALS**

A variable in PHP can hold any type of strings value but in case of output string, then there is a vast difference between single quote ‘’ and double quote “” and its combinations.

These combinations are invalid :

1). ‘’ inside ‘’ : Invalid

2). “” inside “” : Invalid

These combinations are valid :

1). ‘’ inside “” : Valid

2). “” inside ‘’ : Valid

3). `` inside ‘’ : Valid

4). `` inside “” : Valid

**Note ::** `` can be used inside ‘’ or “” only.

**TYPES OF OUTPUT STRINGS**

**1). Raw String**

Any string enclosed/wrapped with ‘’ will be raw string. In this case escape character sequence like, \n,\t,\f,\v,\r,\a or not processed and variables are not processed.

**2). Processed String**

In the case of processed string, every character must be in “”.

Any string enclosed inside “” are known as processed string, and they will process each and every special character like escape sequences \n,\t,\f,\v,\a, and variables will be also processed.

**3). heredoc String**

It works under block level. Heredoc is capable of processing each and every possible combinations.

1). Escape character sequence

2). Variable process

3). Multiple lines

4). Extra space

5). Invalid combination ‘’ and ‘’ and “” inside “” and `` inside `` hence, heredoc is also called block level processed string or block level double quotes string.

6). Behavior : Processed string

7). The best use of heredoc is to generate dynamic forms.

**Syntax :**

echo <<< BLOCK

This is a heredoc string, $a \n

“This is again new line of string “$a” ”

BLOCK;

**OR**

echo <<< “BLOCKNAME”

This is a heredoc string, $a \n

“This is again new line of string “$a” ”

BLOCKNAME;

**Example :**

<?php

// WAP in PHP to show heredoc string.

$a=100;

echo <<< BLOCK

This is a heredoc string, $a \n

“This is again new line of string “$a” ”

'This string is inside $a single quotes \t tabbed' = $a

BLOCK;

echo <<< "BLOCK"

This is a heredoc string, $a \n

“This is again new line of string “$a” ”

'This string is inside $a single quotes \t tabbed' = $a

BLOCK;

?>

**Output :**

This is a heredoc string, 100

“This is again new line of string “ ”

'This string is inside 100 single quotes tabbed' = 100

Notice: Undefined variable: a” in C:\laragon\www\app-2021\cli\output-string\heredoc.php on line 13

This is a heredoc string, 100

“This is again new line of string “ ”

'This string is inside 100 single quotes tabbed' = 100

**4). newdoc String**

It works under block level. newdoc is not capable of processing each and every possible combinations hence generate the raw output.

1). Doesn’t support escape character sequence

2). Doesn’t support variable process

3). Multiple lines

4). Extra space

5). Invalid combination ‘’ and ‘’ and “” inside “” and `` inside `` hence, newdoc is also called block level processed string or block level double quotes string.

6). Behavior : Raw string hence newdoc is also called block level raw string or block level single quote string.

7). The best use of newdoc is to generate static forms.

**Example :**

<?php

// WAP in PHP to show newdoc string.

$a=100;

echo <<< 'BLOCK'

This is a newdoc string, $a \n

“This is again new line of string “$a” ”

'This string is inside $a single quotes \t tabbed' = $a

BLOCK;

$code=<<< 'BLOCK'

This is a newdoc string, $a \n

“This is again new line of string “$a” ”

'This string is inside $a single quotes \t tabbed' = $a

BLOCK;

echo $code;

?>

**Output :**

This is a newdoc string, $a \n

“This is again new line of string “$a” ”

'This string is inside $a single quotes \t tabbed' = $a This is a newdoc string, $a \n

“This is again new line of string “$a” ”

'This string is inside $a single quotes \t tabbed' = $a

**TYPES OF DATA TYPES**

**1). Scaler Data Types (Fundamental Data Types)**

(i) int

(ii) float, double

(iii) string

(iv) boolean

**2). Complex (Compound) Data Types**

(i) Array

(ii) Object

(iii) Callable \*\*(When OOPS)

(iv) Iterable \*\*(When OOPS)

**3). Special Data Types**

(i) Null

(ii) Resource

These are collectively called 10 primitive data types.

**TYPES OF PHP TAGS**

1. Universal Tag or Simple PHP Tag <?php?>
2. Short Open Tag : <? ?>
3. ASP Style Tag : <% %>
4. Script Style Tag : <script language=”PHP”></script>

**TYPES OF PHP SCRIPTS**

1. **Core PHP File**

Entire code must be written in PHP, it can’t contain any code related to another language.

Even if you don’t close the closing of PHP tag, it will work fine opening of PHP file mandatory but closing of PHP tag is not mandatory. These core PHP files consists libraries, functions, packages, or classes which are written in plain PHP code.

1. **Inclusive PHP File**

This is also same as hybrid PHP file, but when this file included in any core PHP file then it becomes inclusive PHP file.

It is recommended to make an extension of .inc.php to identify that these files are inclusive PHP files.

1. **Hybrid PHP File**

The hybrid PHP file contains mixed code of HTML, JS and other languages including PHP file in this case PHP closing is mandatory.

**Note ::** [ WAP in PHP to restrict any file from directly running as main file.]

**external.inc.php**

<?php

defined('APP\_KEY') or exit('No permission to access this file directly');

**main.php**

**function** wish(){

echo 'Merry Christmas...!!!';

}

wish();

<?php

define('APP\_KEY','Deep@12345');

include 'external.inc.php';

**What is the difference between define() and defined()?**

**defined() :** returns true or false if constants is already defined.

**define()** **:** it defines the undefined constants.

**WAP in PHP to make dynamic web form using heredoc?**

**WAP in PHP to make static web form using newdoc?**

**Note ::** The older universal name of newdoc is since php version 5.x and latest name of newdoc is nowdoc string.

**VARIABLE SCOPE**

In any program, there can be only two types of scope.

1). Local Scope

2). Global Scope

* When a variable defined outside of any function block is at global scope.
* When a variable defined inside any function block is at local scope.
* We can access local variable inside function.
* If we try to access global scope variable it will be undefined.

**MEMORY ALLOACTION IN GLOBAL & LOCAL**

function display(){}

// local

// global

$a=20;

$b=200;

$a=20;

Stack

Heap

20

200, 20

**echo $a; // 20**

**echo $b; // 200**

global $a;

**Temporary Scope : local**

**Permanent Scope : global**

* The local variables gets the memory allocation at local scope ie at stack memory while global variable get the memory allocation at global scope ie at heap memory hence if you want the value of global variable it should be accessed from correct scope.
* In order to access global variable we declare global <variable-name1, variable-name2, variable-name3, …> in the first line of the function before using the variable global variable.

**Example :**

**function** display(){

**global** $a, $b, $c;

$z=200; // Global

echo $a; // Global

echo $b; // Global

echo $c; // Global

echo $z; // Local

}

**MEMORY ALLOACTION IN GLOBAL & LOCAL**

$b=200;

function display(){}

// local

// global

$a=20;

$a=900;

$a=300;

**global $a; // 20**

**$a=400**

Stack

Heap

400

200, 300

**echo $a; // 20**

**echo $b; // 200**

global $a;

**Temporary Scope : local**

**Permanent Scope : global**

* Since, global variable is at global scope if try to overwrite the value of global variable the overall changes occurred in the entire program.
* So, highly recommended to use the variable at local but don’t change the value at local.
* The core concept with local and global scope is that, outside the function the variable occupies global scope without global keyword.
* But inside function global scope requires global keyword.

**Example :**

**// For global scope**

$a=200;

echo $a;

**function** display(){

**global** $a;

echo $a;

}

**// For local scope inside the function**

**function** display(){

**global** $a;

$a=100;

echo $a;

}

echo $a;

**WORKING WITH OPERATORS**

The PHP supports two versions of operators.

**1). Legacy Operators** 5.x version

Trick : Are (Arithmetic) Re (Relational) Kaun (Conditional) Log (Logical) Special (Special) Bit (Bitwise) Assign (Assignment) Karenge

1. **Arithmetic Operator**

Used to perform arithmetic or mathematical calculations like +, -, /, \*, % etc. Arithmetic operators are classified into two categories,

1. Unary Operator
2. Binary Operator

**Expression = Operand + Operand (Where, + is operator)**

**Operand :** Variables, constants participating in mathematical expression is called operands.

Example : $x+$y ($x and $y are operands and + is operator)

20-10 (20 and 10 are constants and – is operator)

**Operator :** Symbol b/w two are more than two operands with unique property of mathematical solution is called as operators.

Example : 50+20 = 70

**Expression :** N number of operands separated by logical arrangement of operators.

Example : /2+2 = Expression (Invalid)

+2/2 = Expression (Valid)

**Solution :** The result after evaluating the expression is known as solution.

**Unary :** Operators that can bring change in the value by prefixing at the starting position.

2 : 2>0 2 is a +ve integer : Unsigned Integer

-2 : 2<0 -2 is a -ve integer : Signed Integer

**Note ::** Total size of an integer : **16-bit** [2 bytes] | **32-bit** [4 bytes] | **64-bit** [8 bytes]

In the case of division,

Division it is divided into 2 types :

**1). Float Division (Complete Division)**

**2). Int Division (Partial Division)**

**Complete Division :** Dynamically typed language partial division strict type language.

Example : Dividend = Divisor + Quotient + Remainder (%)

*Remainder(%) = Dividend – (Divisor\*Quotient)*

If the language is strict type otherwise it will divide completely.

**Tips :**

1). int / int = int

2). float / float = float

3). float / int = float

4). int / float = float

**What is intval()?**

It is a pre-defined function which is used to typecast or change one data type into integer data type.

1. **Relational Operator**

**A relation b**

**Equality In-equality**

**a==b value comparison Not equal Strict In-equal**

**a===b value with type comparison**

**>, <**

**!=**

**<>**

**Slack equality**

Relational operators are used for comparison.

**1). Equality Relation**

(i) Strict equality : ==b, a===b

1. Value comparison a==b

2. Value with type comparison a==b

(ii) Slack equality

1. Less than equals to (<=)

2. Greater than equals to (>=)

**2). In-equality Relation**

(i) Unequal (Not equal)

1. != (Not equal)

2. <> (Never happens)

(ii) Strict In-equality

1. > (Greater than)
2. < (Less than)

Result will be always in Boolean true or false.

**Note ::** Except, slack equality all are simple relation where as slack equality is a compound relations.

1. **Conditional Operator**

Conditional operators are the part of ternary operator. It’s also known as short hand notation of if-else.

**Syntax :** <expression or condition> ? return <true>: return <false>

There are total 3 operands are participating that’s why we call it ternary operator.

If expression is true then it will return true otherwise it will return false.

**Example :**

$a=10;

$x = ($a>2) ? ‘Hii’ : ‘Hello’; // R-L

echo $x; // Hii

($a>2) ? $x=’Hi’ : $x=‘Hello’; // L-R

echo “The value of x using ternary $x L to R \n”;

1. **Logical Operator**

These are used to provide conjunction b/w two or more than two conditions.

1). && : Match all the condition to be true.

2). || : Match one or more condition to be true.

3). ! : It converts any false expression into true and vice versa.

**&& Table**

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

**A | B | A&&B**

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

0 | 0 | 0

1 | 0 | 0

0 | 1 | 0

1 | 1 | 1

**|| Table**

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

**A | B | A||B**

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

0 | 0 | 0

1 | 0 | 1

0 | 1 | 1

1 | 1 | 1

**! Table**

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

**A | !A**

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

0 | 1

1 | 0

1. **Special Operator**

PHP supports few operators which are reserved for special purpose and are called as, special operator.

Special operators have following types :

**1). Modify Operator**

(i) Pre (++$i, --$i)

(ii) Post ($i++, $i--)

**2). Reference Operator (&)**

**3). String Operator (Concatenation Operator)**

(i) Concatenation

(ii) Append Operation

String operator is basically used for concatenate the values together.

We know this PHP have dynamic nature so it will type caste int values to string for concatenation.

Since this, concatenation will be performed on the string data types only hence called as string operators.

**Difference between concatenation and append :**

If .= (String assignment operator) used once then it’s concat operator and if we use this more than one time or multiple time then it’s append operator.

**4). Array Operator**

[] – Subscript / Index / Key operator

In order to declare the array we can use subscript operator.

We can use few mathematical operation to perform, array operation.

**Union of array :** + Operator

**Slack equality of array (value equality) :** ==

**Strict equality of array (value and type (order) equality) or identity operation :** ===

**Not equal array :** !=

**Not identical array operation :** !==

1. **Bitwise Operator**
2. **Assignment Operator**

It is used to assign value of evaluated expression from right to left.

$sum = $sum + $n; ---> $sum+=$n;

$sum = $sum - $n; ---> $sum-=$n;

$sum = $sum \* $n; ---> $sum\*=$n;

$sum = $sum / $n; ---> $sum/=$n;

$sum = $sum % $n; ---> $sum%=$n;

$sum = $sum . $n; ---> $sum.=$n;

**2). Modern Operators** 7.x version

**(i) Membership Operator**

&& ---> and, || ---> or, ^ ---> xor

**(ii) Null Coalescing Operator**

??

**(iii) Flagship (Space-ship) Operator**

<=>

**FLOW CONTROLS IN PHP**

**Flow Controls**

Flow controls basically means how a program will execute in primary memory at runtime.

**Normal Execution of Program**

There are two types of approach for normal execution of any program.

**1). Top to Bottom Approach**

The program will start from very first line and stop at last line of the program then it’s called normal execution of program in Top to Bottom Approach.

**Example :**

<?php

// WAP in PHP to show normal execution of the program from top to bottom.

echo "Line ".\_\_LINE\_\_." is executing...\n";

// Where \_\_LINE\_\_ magic constant and magled constant.

echo "Line ".\_\_LINE\_\_." is executing...\n";

echo "Line ".\_\_LINE\_\_." is executing...\n";

echo "Line ".\_\_LINE\_\_." is executing...\n";

echo "Line ".\_\_LINE\_\_." is executing...\n";

echo "Line ".\_\_LINE\_\_." is executing...\n";

echo "Line ".\_\_LINE\_\_." is executing...\n";

echo "Line ".\_\_LINE\_\_." is executing...\n";

echo "Line ".\_\_LINE\_\_." is executing...\n";

echo "Line ".\_\_LINE\_\_." is executing...\n";

?>

**We can control flow of any** program **from various way,**

1). Decision Control Block : It allows the user, to control the execution of specific line of code.

(i) Selection Block

(ii) Conditional Block

(iii) Nested Conditional Block

2). Return Block

3). Loop Block

4). Jump Control Block

**Note ::** Statement basically means piece of code inside any block and wrapping those lines of statement is called block.

**2). Left to Right Approach**

When any expression will be solved it will be solved from left to right by default. But if we want to control this normal execution we can modify it to right to left also.

This can be done by using operators, means operator will utilize associativity and precedence order to achieve this.

**e.g.**

2+2+2-2 **L -> R** (+, - operator associativity L -> R)

$x=2+2 **R -> L** (= operator associativity R -> L)

**Execution L-R**

echo “The result = “.10+20; Associativity L -> R

**Error** : Non-numeric value encountered error, because .has highest priority then +/-.

In case if we use parenthesis () then,

Execution will be R->L

echo “The result = “.(10+20);

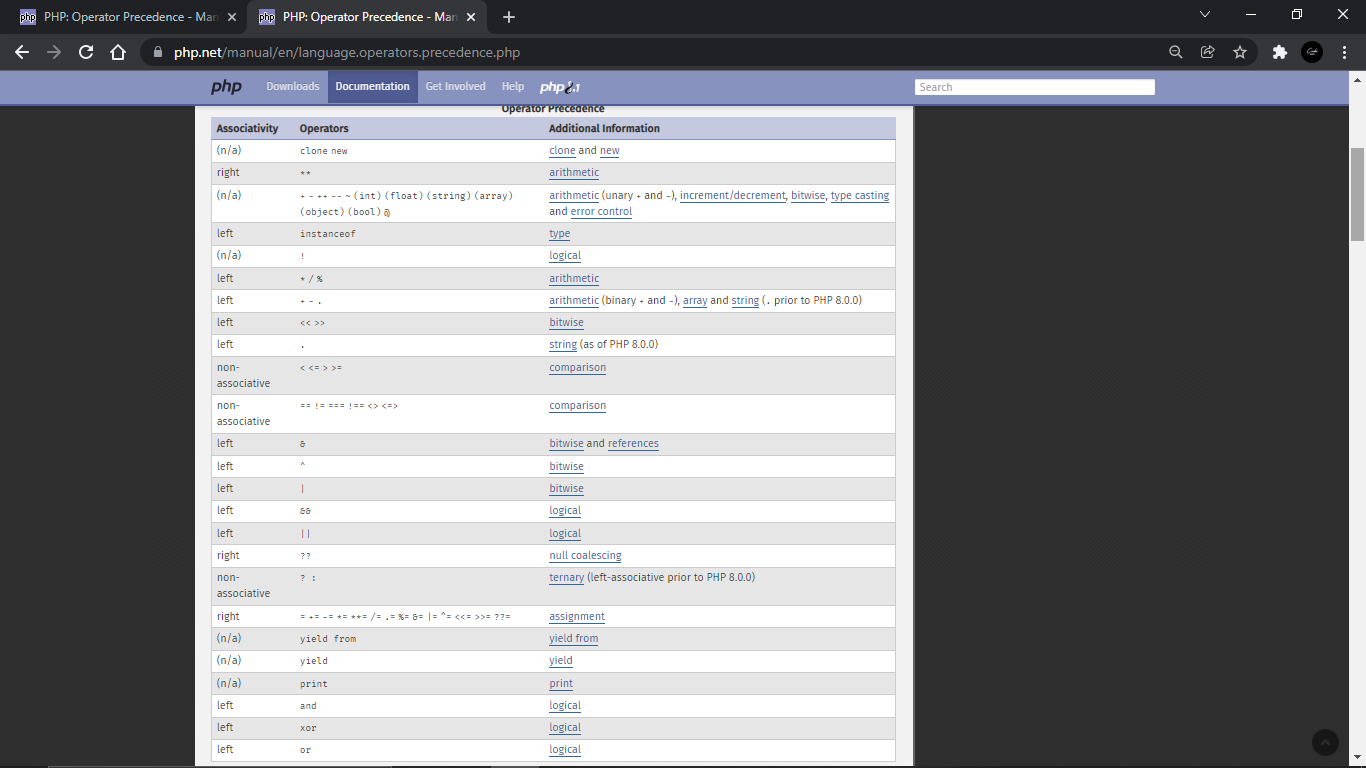
^

L <- R <--|

**Step** : echo “The result = “.30;

^

|--> L -> R

**OPERATOR PRECEDENCE**

**Selection Block**

**Switch Block** : When we have multiple cases and we want to execute, a given case then we go for switch block (statements).

Switch statements are considered to be best alternative of else-if ladder.

Switch can be two types :

1). Unchained Switch

2). Chained Switch

**General (Unchained) Syntax of Switch Block**

Switch(<evaluated expression>){

case <value 1> :

// statement 1

// statement …

// statement n

break;

case <value 2> :

// statement 1

// statement …

// statement n

break;

case <value n> :

// statement 1

// statement …

// statement n

break;

default :

// statement 1

// statement …

// statement n

break;

}

**Note ::** Expression --> Evaluate --> Result

2>2 --> true or false invalid

This expression should be equal to one of case.

**PROGRAM\_1**

<?php

// WAP in PHP to find out odd and even number using switch statement.

$num=readline("Enter a number : ");

switch($num%2){

case 0:

echo "$num is even";

break;

case 1:

echo "$num is odd";

break;

}

?>

**PROGRAM\_2**

<?php

// WAP in PHP to find out odd and even number using switch statement.

$num=readline("Enter a number : ");

switch(!($num%2==0)){

case 0:

echo "$num is even";

break;

case 1:

echo "$num is odd";

break;

}

?>

**PROGRAM\_3**

<?php

// WAP in PHP to find out odd and even number using switch statement.

$num=readline("Enter a number : ");

switch(!($num%2==0)){

case 0:

echo "$num is even";

break;

default:

echo "$num is odd";

break;

}

?>

**CTYPE FUNCTIONS**

**ctype\_alnum —** Check for alphanumeric character(s)

**ctype\_alpha —** Check for alphabetic character(s)

**ctype\_cntrl —** Check for control character(s)

**ctype\_digit —** Check for numeric character(s)

**ctype\_graph —** Check for any printable character(s) except space

**ctype\_lower —** Check for lowercase character(s)

**ctype\_print —** Check for printable character(s)

**ctype\_punct —** Check for any printable character which is not whitespace or an alphanumeric character

**ctype\_space —** Check for whitespace character(s)

**ctype\_upper —** Check for uppercase character(s)

**ctype\_xdigit —** Check for character(s) representing a hexadecimal digit

**General (Chained) Syntax of Switch Block**

A switch statement without any break statement is called chained switch and this effect is called as chaining effect.

In order to use chaining effect in switch we use at least one break statement for each block.

We can break chaining effect of switch by using,

**1). break** : Breaks the execution of switch statement

**2). exit** : Breaks the termination of entire program

**3). return** : It will returns the execution of the program where ever we want

**4). goto exit\_label;** : It will take the execution of the program where ever we want