Cascading Style Sheets (CSS)

COURSE OUTLINE

Course Description

This course will provide a comprehensive guide on learning Cascading Style Sheet or CSS, one of the foundations of web designing. This will tackle everything that is to know about CSS to enable to create and style web layouts or pages.

Learning Outcomes

- Learn the basics of CSS
- Able to create and style layouts
- Able to apply the tricks and techniques for easier coding

Teaching Methods

This course will use lecture- discussion, actual demonstration, and hands-on exercises and practices to test students learning. The course will also use examples for visual learning.

Course Outline

Overview of CSS

- Benefits CSS
- CSS Rules
- Basic terminologies and elements
- Units of Measurement in CSS

Understanding CSS Fonts

- Compatible fonts
- Formatting fonts using CSS

Understanding CSS Texts

- Text Properties
- Formatting text

Colors and Backgrounds

- About Color Values
- Images or colors as background
- Background properties and tricks

Industrial Advancement Academy of the Philippines
708 Victoria corner Escuela Streets, Intramuros, Manila 1002
Tel No.: +63 2 241-2598/ 527-9032
Website: www.iaap.edu.ph
Email: info@iaap.education

CSS Links

- CSS Button Links
- Modifying Links

Border, Margins, and Padding

- Using and modifying borders
- Using and adjusting margins
- Using and adjusting padding

CSS Tables

- Creating Tables using CSS
- Styling Tables

Positioning

- Element Flow
- Position divs and other elements

CSS Lists

- Menus as Lists
- Different kinds of menus

CSS Forms

- Styling and modifying forms

Compatibility Issues with Browsers

Industrial Advancement Academy of the Philippines
708 Victoria corner Escuela Streets, Intramuros, Manila 1002
Tel No.: +63 2 241-2598/ 527-9032
Website: www.iaap.edu.ph
Email: info@iaap.education

HTML5

COURSE OUTLINE

Course Description

HTML5 is the newest version of the markup language used for structuring web pages. This course will give a refresher on basic HTML then dive into the latest version. The course provides a comprehensive guide on how to create and manipulate websites or webpages.

Learning Outcomes

- Know what is new with HTML5
- Use the new features of HTML5 to improve web design
- Create websites that are compatible with browsers

Teaching Methods

This course will use lecture- discussion, actual demonstration, and hands-on exercises and practices to test students learning. The course will also use examples for visual learning.

Course Outline

Introduction to HTML5

- New features on HTML5
- New HTML5 Structural Tags
- HTML5 vs. HTML4

Section and Articles

- Sectioning using HTML
- Outlining
- Using the Article Tag

Using HTML5 Audio and Video

- Audio formats
- Audio tag attributes
- Video tag attributes
- Dealing with non-supporting browser

HTML5 Forms

- Input Types
- New Form Attributes
- Field Attributes

Industrial Advancement Academy of the Philippines
708 Victoria corner Escuela Streets, Intramuros, Manila 1002
Tel No.: +63 2 241-2598/ 527-9032
Website: www.iaap.edu.ph
Email: info@iaap.education

HTML5 Web Storage and other storages

Integrated APIs

- Offline Application
- Drag and Drop Application

Industrial Advancement Academy of the Philippines
708 Victoria corner Escuela Streets, Intramuros, Manila 1002
Tel No.: +63 2 241-2598/ 527-9032
Website: www.iaap.edu.ph
Email: info@iaap.education

Java Server Pages

Course Description

JavaServer Pages help create dynamic and flexible web content in lightning speed. Through this course, students will have a comprehensive and hands-on guide on using the many components of Java technologies as well as techniques and strategies incorporating these technologies in creating powerful web applications

Learning Outcomes

- Understand the different technologies and concepts involve in JavaServer Pages
- Use the different technologies and concepts in developing JSP application
- Understand how to use JSTL and Javabeans

Teaching Methods

This course will use lecture-discussion with visual presentation and demonstration, hands-on exercises and practices.

Course Outline

JavaServer Pages Overview

- Introduction to JSP
- Setting up the JSP Environment
- JSP Architecture
- Understanding the JSP Life Cycle

Working with JSP

- JSP Implicit Objects
- Sending Email with JSP
- Handling dates
- Uploading files
- Working with HTTP
- Handling cookies
- Using filters
- Session Tracking
- Creating database

JSP Tags and Syntax

- Directives
- Declarations
- Scriplets
- Expressions
- Operators
- Statements

JSP Actions

- Identifying and using actions
- Common attributes

Learn Today Succeed Tomorrow

Industrial Advancement Academy of the Philippines
708 Victoria corner Escuela Streets, Intramuros, Manila 1002
Tel No.: +63 2 241-2598/ 527-9032
Website: www.iaap.edu.ph
Email: info@iaap.education

JSP Form Processing

- GET Method
- POST Method

Introduction to JSTL

- Installing JSTL
- JSTL Tags
- Working with JSP and JSTL

Using Javabeans

- Javabeans properties and features
- Accessing Javabeans

Debugging and Securing

- Using Debugging Tools
- Authenticating in JSP

Industrial Advancement Academy of the Philippines
708 Victoria corner Escuela Streets, Intramuros, Manila 1002
Tel No.: +63 2 241-2598/ 527-9032
Website: www.iaap.edu.ph
Email: info@iaap.education

Microsoft Exchange Server 2010

OVERVIEW

This course is designed to guide students in using Microsoft Exchange Server in version 2010. The course will give practical lessons in managing, configuring, and troubleshooting Exchange .

PREREQUISITES

Experience with Windows Server operating systems

LEARNING OUTCOMES

- Learn the basic and functions of MS Exchange
- Use the MS Exchange with ease
- Implement messaging security in MS Exchange
- Creating backup and restore for the server roles.

TRAINING METHODS

This course will use lecture-discussion with visual presentation, and hands-on exercises and practices.

COURSE OUTLINE

Overview of Exchange Server 2010

- Basic Functions and tools
- Installing Exchange Server 2010
- Requirements for an Exchange Server Installation

Mailbox Servers

- Mailbox Server Roles
- Mailbox Databases
- Public Folders

Managing Recipient Objects

- Email Address Policies
- Address Lists and Address Book Policies
- Performing Bulk Recipient Management Tasks

Managing Client Access

- Client Access Server Role
- The Outlook Web App
- Configuring Mobile Messaging

Implementing Messaging Security

- Edge Transport Servers

- Antivirus Solution
- an Anti-Spam Solution
- Secure SMTP Messaging

Implementing High Availability
- Configuring Highly Available Mailbox Databases
- Deploying Highly Available Non-Mailbox Servers

Implementing Backup and Recovery

Messaging Policy and Compliance
- Configuring Transport Rules
- Configuring Journaling and Multi-Mailbox Search
- Configuring Personal Archives
- Configuring Messaging Records Management

Securing Microsoft Exchange Server 2010
- Configuring Role Based Access Control
- Configuring Audit Logging
- Configuring Secure Internet Access

Maintaining Microsoft Exchange Server 2010
- Monitoring Exchange Server 2010
- Maintaining Exchange Server 2010
- Troubleshooting Exchange Server 2010

Microsoft Exchange Server 2013

OVERVIEW

This course is designed to guide students in using Microsoft Exchange Server in version 2013. The course will give practical lessons in managing, configuring, and troubleshooting Exchange .

PREREQUISITES

Experience with Windows Server operating systems

LEARNING OUTCOMES

- Experience and transition with ease to latest version of Windows Exchange
- Learn the basic tools and functions of Exchange
- Apply lessons in practical using in business or personal use

TRAINING METHODS

This course will use lecture-discussion with visual presentation, interactive discussion, and hands-on exercises and practices to test students learning.

COURSE OUTLINE

- Exchange Server 2013
  - Overview
  - Prerequisites and Requirements
  - Installing Exchange Server 2013
  - Managing Exchange Server 2013

- Mailbox Servers
  - Planning the Mailbox Server Deployment
  - Configuring Mailbox Servers

- Recipient Objects
  - Managing Exchange Recipients
  - Managing Address Lists and Policies on Mailbox Server Role

- Client Access Servers
  - Planning Client Access Server Deployment
  - Configuring the Client Access Server Role
  - Managing Client Access Services

- Messaging Client Connectivity
  - Client Connectivity to Client Access Server
  - Configuring Outlook Web App

- Message Transport
  - Planning and Configuring Message Transport
  - Managing Transport Rules

- High Availability
  - Configuring Highly Available Mailbox Databases
  - Configuring Highly Available Client Access Servers

- Disaster Recovery
  - Disaster Mitigation
  - Exchange Server 2013 Backup
  - Exchange Server 2013 Recovery

- Message Security Options
  - Antivirus Solution for Exchange Server 2013

- Administrative Security and Auditing
  - Role-Based Access Control
  - Audit Logging

- Maintaining, Monitoring, and Troubleshooting Exchange Server 2013

Microsoft Windows Server 2012

OVERVIEW

This course will give you the fundamental knowledge in Windows Server 2012, a group of operating systems created by Microsoft to serve the needs to business markets as well as common individuals.

LEARNING OUTCOMES

- Learn the basic in running a Windows Server 2012 R2
- Manipulate the Active Directory with graphical and PowerShell tools
- Create folder security, file filtering and recovery backup plans
- Understand the Hyper-V and how it works

TRAINING METHODS

This course will use lecture-discussion with visual presentation, and hands-on exercises and practices.

COURSE OUTLINE

- Introduction to MS Windows Server 2012
    - Upgrading the existing operating system
    - Leveraging the multi-server Server Manager
    - Performing administrative tasks with PowerShell
    -
- Constructing the Active Directory
    - Active Directory infrastructure components
    - Migrating existing domains to Windows Server 2012
    - Creating objects with the GUI and PowerShell
    -
- Protecting Disk and File System Resources
    - Backing up, restoring and organizing disks
    - Defining security on files and folders
    -
- Managing and Troubleshooting Servers
    - Manipulating administrative tools and applets
    - Measuring reliability and analyzing performance
    -
- Setting Up Network Components
    - Establishing network settings
    - Insuring network resource availability

- Administering the Active Directory

    - Assigning administrative rights
    - Deploying Group Policies for central management
- Virtualizing Servers with Hyper-V
    - Managing virtual machines
    - Minimizing downtime with replication
    - Remote Desktop Services and Virtual Desktop Infrastructure

MS SQL Database

OVERVIEW

This course provides complete training Microsoft SQL Databases. Starting with the basic to a more complex approach, the course will guide you thoroughly in mastering MS SQL Database.

LEARNING OUTCOMES

- Be familiarize with the functions of MS SQL program
- Ability to use MS SQL functions with ease and proficiency

TRAINING METHODS

This course will use lecture-discussion with visual presentation, and hands-on exercises and practices.

COURSE OUTLINE

- Introduction to SQL
- Database Creation in SQL
  - Create Database using MS SQL Server Management Studio
  - Create Database using Command Tool
  - Create Table using Template GUI
- Database Design and RDBMS Concepts in Microsoft SQL
- Primary and Foreign Key
- Create, Delete, Insert and Update
- Transactions in MS SQL
- Function and Operators in SQL
  - String Functions in MS SQL
  - Aggregate Functions
  - Date Functions in MS SQL
- Basic Select Statements in SQL
- Complex Select Statements in SQL
- Insert and Deletes in MS SQL
- Joins, Unions and Subqueries in SQL
- Having, Order By, Distinct in SQL
- Schema, Views and Indexes in SQL
- Stored Procedures in MS SQL
- Functions and Triggers in SQL
- Data Types in MS SQ

MS SQL Programming

OVERVIEW

The skill to write the SQL language is pertinent for people involved in developing database applications. This course gives you a complete guide in SQL programming language which prepares you to build, query and manipulate databases.

LEARNING OUTCOMES

- Update database content with SQL and transaction handling
- Retrieve data with filter conditions and from multiple tables using various types of join
- Process data with row and aggregate functions

TRAINING METHODS

This course will use lecture-discussion with visual presentation, and hands-on exercises and practices.

COURSE OUTLINE

SQL Overview

- The fundamental building blocks

Building the Database Schema

Creating tables and columns

Protecting data integrity with constraints

- Primary key constraints
- Foreign key constraints

Improving performance with indexes

- Data retrieval
- Guidelines for index creation

Manipulating Data

Modifying table contents

Applying transactions

Working with the SELECT Statement

Writing Single Table queries

Restricting rows with the WHERE filter

Querying Multiple Tables

Applying the ANSI/ISO standard join syntax

Combining results with set operators

Employing Functions in Data Retrieval

Processing data with row functions

- CASE expression
- NULL values

Performing analysis with aggregate functions

- Summarizing data
- Summary level
- Filter conditions

Constructing Nested Queries

Applying subqueries in filter conditions

- Correlated vs. noncorrelated subqueries

Including subqueries in expressions

Developing In-Line and Stored Views

Breaking down complex problems

Creating views in a database

- Updateable vs. non-updateable views

PHP Coding/Programming

Course Description

The course aims to teach students the fundamental knowledge in PHP. Students will learn how to write and develop web applications using solely PHP language like numbers, strings, and other variables, expressions, and operators.

Learning Outcomes

- Understanding the programing fundamentals
- Identify the different functions, variables, and expressions and use them propeprly
- Create power and dynamic web applications through PHP

Teaching Methods

This course will use lecture-discussion with visual presentation and demonstration, hands-on exercises and practices.

Course Outline

Introduction to PHP

- Basic Components, Concepts, and Functions
- Web Application Architecture

Working with PHP

- Basic PHP Language
- PHP Functions
- PHP Variables
- PHP expressions
- Manipulating variables and expressions

Control Structures

- Operators
- Statement Blocks
- Loop Functions

Working with Arrays

- Function of Arrays
- Using Arrays with PHP
- Manipulating Arrays

PHP and HTML

- Integrating PHP with HTML
- HTML forms

Industrial Advancement Academy of the Philippines
708 Victoria corner Escuela Streets, Intramuros, Manila 1002
Tel No.: +63 2 241-2598/ 527-9032
Website: www.iaap.edu.ph
Email: info@iaap.education

PHP Laravel Framework

Course Description

Learn how to use and manipulate one of the latest, Laravel. Create powerful and elegant web applications as we teach and guide you on the fundamental knowledge and basic tools needed to work with this popular framework. Expound your skills as we equip you with basic and advanced techniques that will help you master this framework dubbed as the PHP Framework for Web Artisans.

Learning Outcomes

- Identify the difference between Laravel and other PHP Frameworks
- Completely understand the Laravel architecture and technologies, and learn how to manipulate them
- Create and deploy web applications using the Laravel Framework

Teaching Methods

This course will use lecture-discussion with visual presentation and demonstration, hands-on exercises and practices.

Course Outline

Introduction to Laravel

- Getting to know its Architecture
- Basic Features and Technologies
- Setting up Composer
- Configuring Laravel

Handling Routing

- Basic Routing
- Advanced Routing
- Routing Parameters
- Handling Invalid Routes

Introduction to Controllers

- Basic Controllers
- Creating Controllers
- Control Routing

Using Filters

- Basic Filters
- Types of Filters
- Filter Classes

Blade Templates

- Creating Templates
- PHP Output
- Template Inheritance

Generating URLs

- Generating Framework URLs

Learn Today Succeed Tomorrow

Industrial Advancement Academy of the Philippines
708 Victoria corner Escuela Streets, Intramuros, Manila 1002
Tel No.: +63 2 241-2598/ 527-9032
Website: www.iaap.edu.ph
Email: info@iaap.education

- Generation shortcuts

Requesting Data
- Retrieving Data
- Working with old inputs
- Uploading files
- Working with cookies

Responses
- Working with Views
- Custom Responses

Working with Database
- Introduction to Databases
- Configuring Databases
- Migration

Using Schema Builder
- Working with Tables
- Formatting Tables

Introduction to Forms
- Opening Forms
- Different Kinds of Forms

Introduction Validation
- Validation Rules and Custom Rules
- Handling Error Messages

Working with Models
- Defining Models
- Working with Models
- Working with Eloquent ORM

Industrial Advancement Academy of the Philippines
708 Victoria corner Escuela Streets, Intramuros, Manila 1002
Tel No.: +63 2 241-2598/ 527-9032
Website: www.iaap.edu.ph
Email: info@iaap.education

PHP Symfony Framework

Course Description

This course will give you a comprehensive walkthrough on the basic tools and technologies of Symfony to create powerful web applications using the stated framework. The course will equip with you all the necessary skills and knowledge to fully master from basic to advanced PHP Symfony programming.

Learning Outcomes

- Compare Symfony with other frameworks
- Understand the different concepts and technologies in Symfony
- Learn how to configure, route and use templates in Symfony
- Use Twig, PHPUnit, Doctrine ORM, and other technologies in creating web applications

Teaching Methods

This course will use lecture-discussion with visual presentation and demonstration, hands-on exercises and practices.

Course Outline

Introduction to Symfony

- Basic Concepts and Technologies
- Getting to know its Architecture
- Difference with other Frameworks
- Symfony and the Model View Controller (MVC)

Using the Web Debug Toolbar

- Using the Web Profiler application
- Understanding the recorded logs

Configuring Symfony

- Using built-in XML, YAML and PHP
- Using global settings
- Configuring routes and URLs

Templates

- Working with Twig Templates
- Using the Template Inheritance feature

Introduction to Controllers

- Creating a Controller
- Annotation Configuration

Requesting Data

- Accessing information
- Working with Session's Data
- Working with cookies

Responses

Learn Today Succeed Tomorrow

Industrial Advancement Academy of the Philippines
708 Victoria corner Escuela Streets, Intramuros, Manila 1002
Tel No.: +63 2 241-2598/ 527-9032
Website: www.iaap.edu.ph
Email: info@iaap.education

- Generating responses

Using Forms
- Creating forms
- Formatting and configuring forms
- Processing Data
- Applying rules

Internationalization and Localization
- Understanding the Translation component
- Changing the default user's locale

Dependency injection principle
- Understanding the dependency injection principle

Introduction to Symfony Service Container
- Using command line tools
- Registering new custom business services

Understanding Automated Tests
- Understanding the tests automation framework
- Creating and executing test commands
- Configuring tests

Authentication and Authorization
- Form Based Authentication Strategy
- Access control policy
- User Permission

HTTP Caching
- Identifying the different kinds of cache systems
- Using cache strategies

Database
- Basics of Doctrine ORM
- Components and Tools of the Doctrine ORM

PHP Training

Course Description

Learning Outcomes

Teaching Methods

Course Outline

Introduction to PHP

- PHP Overview
- Identifying PHP Features
- Installing Apache, MySQL, PHP
- Creating a simple PHP example

Using PHP Echo and PHP Print

- Printing string
- Printing multi line string
- Printing escaping characters
- Printing variable value

PHP Variables

- Important Reminders with PHP Variables
- Rules of PHP Variables
- String, integer and float

PHP Constants

- Using define() function
- Using const keyword

PHP Data Types

1. Scalar Types
2. Compound Types
3. Special Types

Industrial Advancement Academy of the Philippines
708 Victoria corner Escuela Streets, Intramuros, Manila 1002
Tel No.: +63 2 241-2598/ 527-9032
Website: www.iaap.edu.ph
Email: info@iaap.education

Programming

COURSE OUTLINE

Course Description

The course provides a strong foundation on computer programming. Students will get an overview on the different programming languages, and learn how to use them. The course will give a thorough guide on how to properly and easily write and execute codes and commands.

Learning Outcomes

- Understand the various programming languages
- Create conditions, loops, variables, and expressions
- Find and solve error in codes

Teaching Methods

This course will use lecture- discussion, actual demonstration, and hands-on exercises and practices to test students learning.

Course Outline

Programming Basics

- Basic Concepts and Principles
- The Source Code
- Programming languages
- Softwares

Overview on Javascript

- Basic Concepts and Terminologies
- Creating Javascripts

Variables and Data Types

- Understanding different kind of languages
- Working numbers, characters, strings, and operators

Writing Conditional Code

- The “if” statement
- The switch statement
- Working with complex conditions
- Setting comparison operators

Modular Code

- Breaking and splitting your code

Industrial Advancement Academy of the Philippines
708 Victoria corner Escuela Streets, Intramuros, Manila 002
Tel No.: +63 2 24-2598/ 527-9032
Website: www.iaap.edu.ph
Email: info@iaap.education

- Creating and calling functions
- Setting parameters and arguments
- Understanding variable scope

Creating Loops

- Overview on iteration

More about Strings

- Cleaning up with string concatenation
- Finding patterns in strings
- Introduction to regular expressions

Collections

- Understanding Arrays
- Iterating through collections
- Collections in other languages

Input and Output

- Input/output and persistence
- Overview on Document Object Model
- Event driven programming
- Introduction to file I/O

Debugging

- Using debuggers
- Working around error messages

Introduction to Object Orientation

- Object-oriented languages
- Using classes and objects
- Reviewing object-oriented languages

Industrial Advancement Academy of the Philippines
708 Victoria corner Escuela Streets, Intramuros, Manila 002
Tel No.: +63 2 24-2598/ 527-9032
Website: www.iaap.edu.ph
Email: info@iaap.education

Struts 2

Course Description

Learning Outcomes

Teaching Methods

This course will use lecture-discussion with visual presentation and demonstration, hands-on exercises and practices.

Course Outline

Struts 2 Overview

- Basic MVC Architecture
- Basic concepts and features of Struts 2
- The Struts 2 Framework
- Setting up the Struts 2 Environment

Working with Struts 2

- Packaging and implement actions
- Using interceptors
- Configuring xml files
- Uploading and configuring files

Industrial Advancement Academy of the Philippines
708 Victoria corner Escuela Streets, Intramuros, Manila 1002
Tel No.: +63 2 241-2598/ 527-9032
Website: www.iaap.edu.ph
Email: info@iaap.education

PHP TRAINING

What is PHP

PHP is a open source, interpreted and object-oriented scripting language i.e. executed at server side. It is used to develop web applications (an application i.e. executed at server side and generates dynamic page).

What is PHP

- PHP is a server side scripting language.
- PHP is an interpreted language, i.e. there is no need for compilation.
- PHP is an object-oriented language.
- PHP is an open-source scripting language.
- PHP is simple and easy to learn language.

PHP Features

There are given many features of PHP.

- **Performance**: Script written in PHP executes much faster then those scripts written in other languages such as JSP & ASP.
- **Open Source Software**: PHP source code is free available on the web, you can developed all the version of PHP according to your requirement without paying any cost.
- **Platform Independent**: PHP are available for WINDOWS, MAC, LINUX & UNIX operating system. A PHP application developed in one OS can be easily executed in other OS also.
- **Compatibility**: PHP is compatible with almost all local servers used today like Apache, IIS etc.
- **Embedded**: PHP code can be easily embedded within HTML tags and script.

Install PHP

To install PHP, we will suggest you to install AMP (Apache, MySQL, PHP) software stack. It is available for all operating systems. There are many AMP options available in the market that are given below:

- **WAMP** for Windows
- **LAMP** for Linux
- **MAMP** for Mac
- **SAMP** for Solaris
- **FAMP** for FreeBSD
- **XAMPP** (Cross, Apache, MySQL, PHP, Perl) for Cross Platform: It includes some other components too such as FileZilla, OpenSSL, Webalizer, OpenSSL, Mercury Mail etc.

If you are on Windows and don't want Perl and other features of XAMPP, you should go for WAMP. In a similar way, you may use LAMP for Linux and MAMP for Macintosh.

PHP Example

It is very easy to create a simple PHP example. To do so, create a file and write HTML tags + PHP code and save this file with .php extension.
All PHP code goes between php tag. A syntax of PHP tag is given below:

```
1. <?php
2. //your code here
3. ?>
```

Let's see a simple PHP example where we are writing some text using PHP echo command.

File: first.php

```
1. <!DOCTYPE>
2. <html>
3. <body>
4. <?php
5. echo "<h2>Hello First PHP</h2>";
6. ?>
7. </body>
8. </html>
```

Output:

Hello First PH

PHP Echo

PHP echo is a language construct not a function, so you don't need to use parenthesis with it. But if you want to use more than one parameters, it is required to use parenthesis.
The syntax of PHP echo is given below:

```
1. void echo ( string $arg1 [, string $... ] )
```

PHP echo statement can be used to print string, multi line strings, escaping characters, variable, array etc.

PHP echo: printing string

File: echo1.php

```
1. <?php
2. echo "Hello by PHP echo";
```

```
3. ?>
```

Output:

```
Hello by PHP echo
```

PHP echo: printing multi line string

File: echo2.php

```
1. <?php
2. echo "Hello by PHP echo
3. this is multi line
4. text printed by
5. PHP echo statement
6. ";
7. ?>
```

Output:

```
Hello by PHP echo this is multi line text printed by PHP echo statement
```

PHP echo: printing escaping characters

File: echo3.php

```
1. <?php
2. echo "Hello escape \"sequence\" characters";
3. ?>
```

Output:

```
Hello escape "sequence" characters
```

PHP echo: printing variable value

File: echo4.php

```
1. <?php
2. $msg="Hello JavaTpoint PHP";
3. echo "Message is: $msg";
4. ?>
```

Output:

```
Message is: Hello JavaTpoint PHP
```

PHP Print

Like PHP echo, PHP print is a language construct, so you don't need to use parenthesis with the argument list. Unlike echo, it always returns 1.

The syntax of PHP print is given below:

```
1. int print(string $arg)
```

PHP print statement can be used to print string, multi line strings, escaping characters, variable, array etc.

PHP print: printing string

File: print1.php

```
1. <?php
2. print "Hello by PHP print ";
3. print ("Hello by PHP print()");
4. ?>
```

Output:

```
Hello by PHP print Hello by PHP print()
```

PHP print: printing multi line string

File: print2.php

```
1. <?php
2. print "Hello by PHP print
3. this is multi line
4. text printed by
5. PHP print statement
6. ";
7. ?>
```

Output:

```
Hello by PHP print this is multi line text printed by PHP print statement
```

PHP print: printing escaping characters

File: print3.php

```
1. <?php
2. print "Hello escape \"sequence\" characters by PHP print";
3. ?>
```

Output:

```
Hello escape "sequence" characters by PHP print
```

PHP print: printing variable value

File: print4.php

```
1. <?php
2. $msg="Hello print() in PHP";
3. print "Message is: $msg";
4. ?>
```

Output:

```
Message is: Hello print() in PHP
```

PHP Variables

A variable in PHP is a name of memory location that holds data. A variable is a temporary storage that is used to store data temporarily.

In PHP, a variable is declared using $ sign followed by variable name.

Syntax of declaring a variable in PHP is given below:

```
$variablename=value;
```

PHP Variable: Declaring string, integer and float

Let's see the example to store string, integer and float values in PHP variables.

File: variable1.php

```
<?php
$str="hello string";
$x=200;
$y=44.6;
echo "string is: $str <br/>";
echo "integer is: $x <br/>";
echo "float is: $y <br/>";
?>
```

Output:

```
string is: hello string
integer is: 200
float is: 44.6
```

PHP Variable: Sum of two variables

File: variable2.php

```
<?php
$x=5;
$y=6;
$z=$x+$y;
echo $z;
?>
```

Output:

```
11
```

PHP Variable: case sensitive

In PHP, variable names are case sensitive. So variable name "color" is different from Color, COLOR, COLor etc.

File: variable3.php

```
<?php
$color="red";
echo "My car is " . $color . "<br>";
echo "My house is " . $COLOR . "<br>";
echo "My boat is " . $coLOR . "<br>";
?>
```

Output:

```
My car is red
```

```
Notice: Undefined variable: COLOR in C:\wamp\www\variable.php on
line 4
My house is
Notice: Undefined variable: coLOR in C:\wamp\www\variable.php on
line 5
My boat is
```


PHP Variable: Rules

PHP variables must start with letter or underscore only.
PHP variable can't be start with numbers and special symbols.

File: variablevalid.php

```
1. <?php
2. $a="hello";//letter (valid)
3. $_b="hello";//underscore (valid)
4.
5. echo "$a <br/> $_b";
6. ?>
```

Output:

```
hello
hello
```

File: variableinvalid.php

```
1. <?php
2. $4c="hello";//number (invalid)
3. $*d="hello";//special symbol (invalid)
4.
5. echo "$4c <br/> $*d";
6. ?>
```

Output:

```
Parse error: syntax error, unexpected '4' (T_LNUMBER), expecting
variable (T_VARIABLE)
 or '$' in C:\wamp\www\variableinvalid.php on line 2
```

PHP: Loosely typed language

PHP is a loosely typed language, it means PHP automatically converts the variable to its correct data type.

PHP Constants

PHP constants are name or identifier that can't be changed during the execution of the script. PHP constants can be defined by 2 ways:

1. Using define() function
2. Using const keyword

PHP constants follow the same PHP variable rules. For example, it can be started with letter or underscore only.
Conventionally, PHP constants should be defined in uppercase letters.

PHP constant: define()

Let's see the syntax of define() function in PHP.

1. define(name, value, case-insensitive)
    1. name: specifies the constant name
    2. value: specifies the constant value
    3. case-insensitive: Default value is false. It means it is case sensitive by default.

Let's see the example to define PHP constant using define().
File: constant1.php

```
1. <?php
2. define("MESSAGE","Hello JavaTpoint PHP");
3. echo MESSAGE;
4. ?>
```

Output:

```
Hello JavaTpoint PHP
```

File: constant2.php

```
1. <?php
2. define("MESSAGE","Hello JavaTpoint PHP",true);//not case sensitive
3. echo MESSAGE;
4. echo message;
5. ?>
```

Output:

```
Hello JavaTpoint PHPHello JavaTpoint PHP
```

File: constant3.php

```
1. <?php
2. define("MESSAGE","Hello JavaTpoint PHP",false);//case sensitive
3. echo MESSAGE;
4. echo message;
5. ?>
```

Output:

```
Hello JavaTpoint PHP
Notice: Use of undefined constant message - assumed 'message'
in C:\wamp\www\vconstant3.php on line 4
message
```

PHP constant: const keyword

The const keyword defines constants at compile time. It is a language construct not a function.

It is bit faster than define().

It is always case sensitive.

File: constant4.php

```
1. <?php
2. const MESSAGE="Hello const by JavaTpoint PHP";
3. echo MESSAGE;
4. ?>
```

Output:

```
Hello const by JavaTpoint PHP
```

PHP Data Types

PHP data types are used to hold different types of data or values. PHP supports 8 primitive data types that can be categorized further in 3 types:

1. Scalar Types
2. Compound Types
3. Special Types

PHP Data Types: Scalar Types

There are 4 scalar data types in PHP.

1. boolean
2. integer
3. float
4. string

PHP Data Types: Compound Types

There are 2 compound data types in PHP.

1. array
2. object

PHP Data Types: Special Types

There are 2 special data types in PHP.

1. resource
2. NULL

PHP Operators

PHP Operator is a symbol i.e used to perform operations on operands. For example:

1. $num=10+20;//+ is the operator and 10,20 are operands

In the above example, + is the binary + operator, 10 and 20 are operands and $num is variable.

PHP Operators can be categorized in following forms:

- Arithmetic Operators
- Comparison Operators
- Bitwise Operators
- Logical Operators
- String Operators
- Incrementing/Decrementing Operators
- Array Operators
- Type Operators
- Execution Operators
- Error Control Operators
- Assignment Operators

We can also categorize operators on behalf of operands. They can be categorized in 3 forms:

- Unary Operators: works on single operands such as ++, -- etc.
- Binary Operators: works on two operands such as binary +, -, *, / etc.
- Ternary Operators: works on three operands such as "?:".

PHP Operators Precedence

Let's see the precedence of PHP operators with associativity.

Operators	Additional Information	Associativity
clone new	clone and new	non-associative
[	array()	left
**	arithmetic	right
++ -- ~ (int) (float) (string) (array) (object) (bool) @	increment/decrement and types	right
instanceof	types	non-associative
!	logical (negation)	right
* / %	arithmetic	left
+ - .	arithmetic and string concatenation	left
<< >>	bitwise (shift)	left
< <= > >=	comparison	non-associative
== != === !== <>	comparison	non-associative
&	bitwise AND	left
^	bitwise XOR	left
|	bitwise OR	left
&&	logical AND	left
||	logical OR	left
?:	ternary	left
= += -= *= **= /= .= %= &= |= ^= <<= >>= =>	assignment	right
and	logical	left

xor	logical	left
or	logical	left
,	many uses (comma)	left

PHP Comments

PHP comments can be used to describe any line of code so that other developer can understand the code easily. It can also be used to hide any code.

PHP supports single line and multi line comments. These comments are similar to C/C++ and Perl style (Unix shell style) comments.

PHP Single Line Comments

There are two ways to use single line comments in PHP.

- // (C++ style single line comment)
- # (Unix Shell style single line comment)

```
<?php
// this is C++ style single line comment
# this is Unix Shell style single line comment
echo "Welcome to PHP single line comments";
?>
```

Output:

```
Welcome to PHP single line comments
```

PHP Multi Line Comments

In PHP, we can comments multiple lines also. To do so, we need to enclose all lines within /* */. Let's see a simple example of PHP multiple line comment.

```
<?php
/*
Anything placed
within comment
will not be displayed
on the browser;
*/
echo "Welcome to PHP multi line comment";
?>
```

Output:

```
Welcome to PHP multi line comment
```

CONTROL STATEMENTS

PHP If Else

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

- if
- if-else
- if-else-if
- nested if

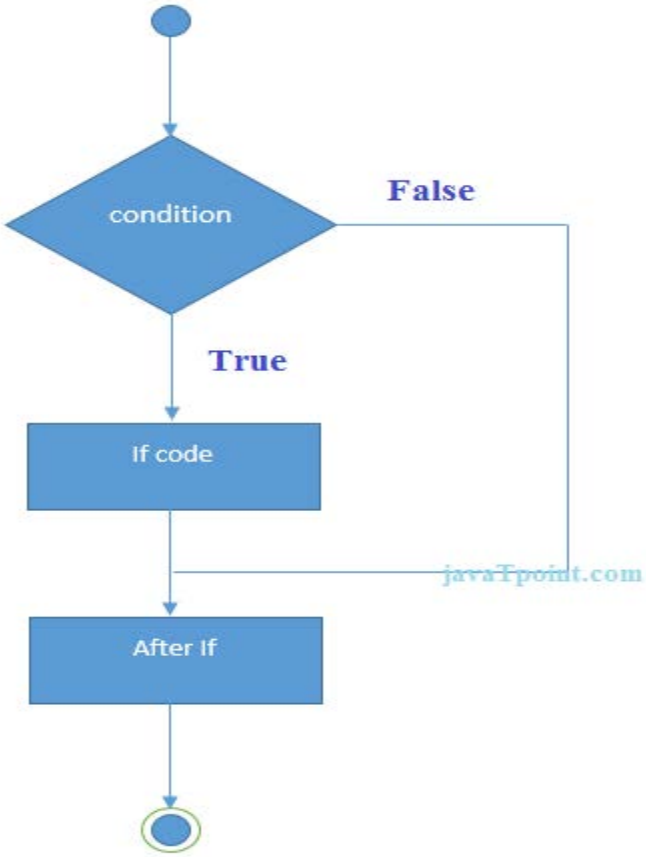
PHP If Statement

PHP if statement is executed if condition is true.

Syntax

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

Flowchart



Example

```
1. <?php
2. $num=12;
```

```
3. if($num<100){
4. echo "$num is less than 100";
5. }
6. ?>
```

Output:

```
12 is less than 100
```

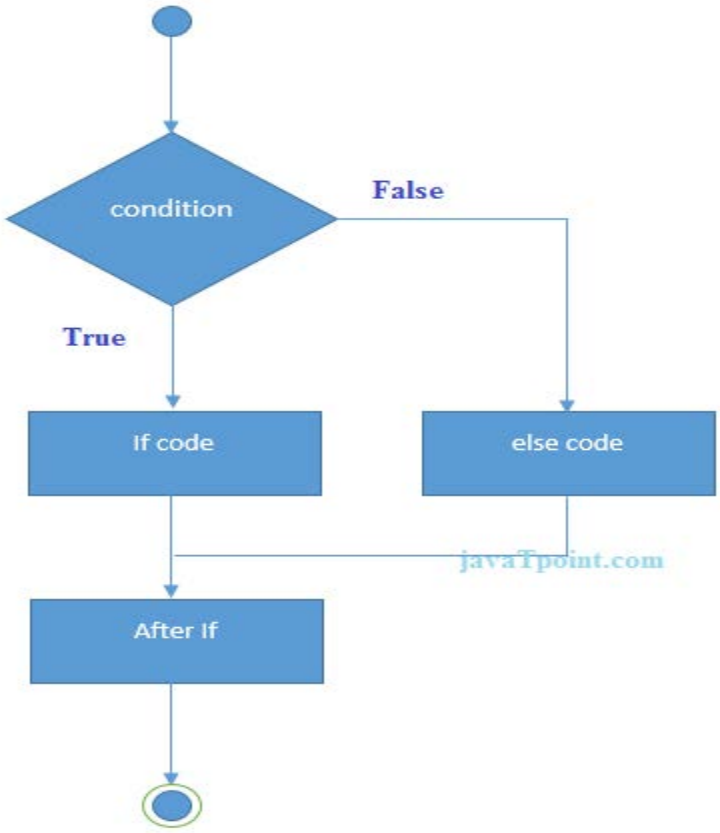

PHP If-else Statement

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

Syntax

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

Flowchart



Example

```
<?php
$num=12;
if($num%2==0){
echo "$num is even number";
}else{
echo "$num is odd number";
}
?>
```

Output:

```
12 is even number
```


PHP Switch

PHP switch statement is used to execute one statement from multiple conditions. It works like PHP if-else-if statement.

Syntax

```
switch(expression){
case value1:
 //code to be executed
 break;
case value2:
 //code to be executed
 break;
.....
default:
        code to be executed if all cases are not matched;
       }
```

Flowchart

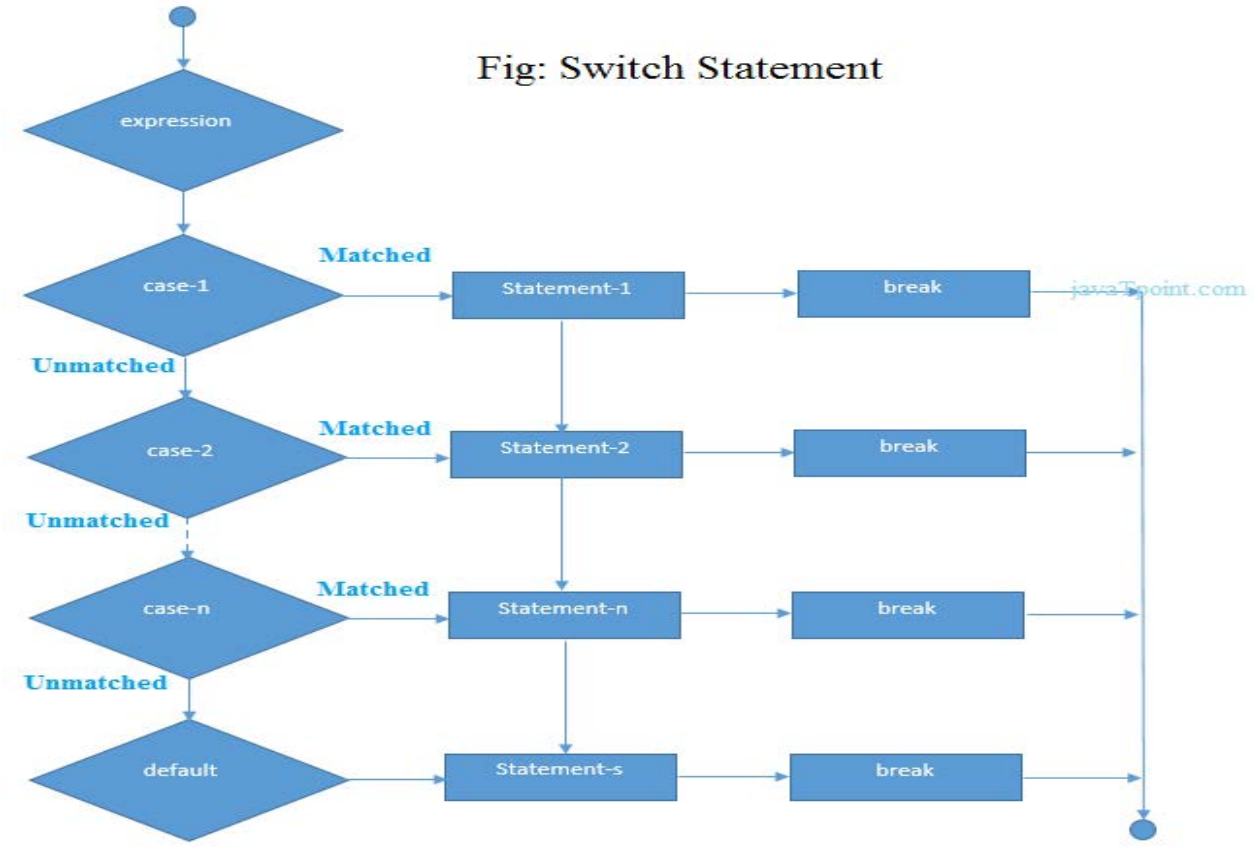


Fig: Switch Statement

Example

```
1. <?php
2. $num=20;
3. switch($num){
4. case 10:
5. echo("number is equals to 10");
6. break;
7. case 20:
8. echo("number is equal to 20");
9. break;
10.       case 30:
11.       echo("number is equal to 30");
12.       break;
13.       default:
14.       echo("number is not equal to 10, 20 or 30");
15.       }
16.       ?>
```

Output:

```
number is equal to 20
```

PHP For Loop

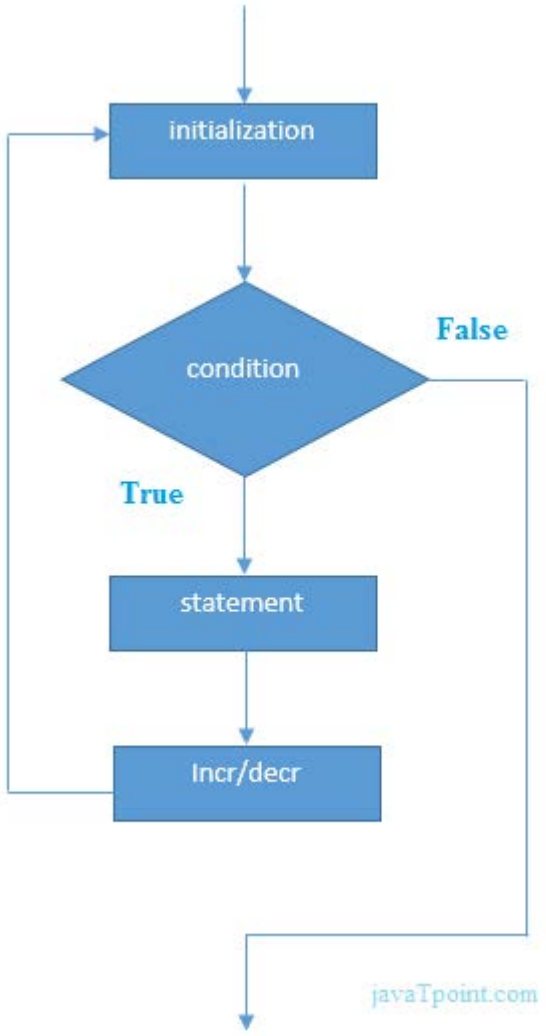
PHP for loop can be used to traverse set of code for the specified number of times.

It should be used if number of iteration is known otherwise use while loop.

Syntax

```
1. for(initialization; condition; increment/decrement){
2. //code to be executed
3. }
```

Flowchart



Example

```
<?php
for($n=1;$n<=10;$n++){
echo "$n<br/>";
}
?>
```

Output:

```
1
2
3
4
5
6
7
8
9
10
```

PHP Nested For Loop

We can use for loop inside for loop in PHP, it is known as nested for loop.

In case of inner or nested for loop, nested for loop is executed fully for one outer for loop. If outer for loop is to be executed for 3 times and inner for loop for 3 times, inner for loop will be executed 9 times (3 times for 1st outer loop, 3 times for 2nd outer loop and 3 times for 3rd outer loop).

Example

```
<?php
for($i=1;$i<=3;$i++){
for($j=1;$j<=3;$j++){
echo "$i  $j<br/>";
}
}
?>
```

Output:

```
1 1
1 2
1 3
2 1
2 2
2 3
3 1
3 2
3 3
```

PHP For Each Loop

PHP for each loop is used to traverse array elements.

Syntax

```
foreach( $array as $var ){
 //code to be executed
}
?>
```

Example

```
<?php
$season=array("summer","winter","spring","autumn");
foreach( $season as $arr ){
  echo "Season is: $arr<br />";
}
?>
```

Output:

```
Season is: summer
Season is: winter
Season is: spring
Season is: autumn
```

PHP While Loop

PHP while loop can be used to traverse set of code like for loop.
It should be used if number of iteration is not known.

Syntax

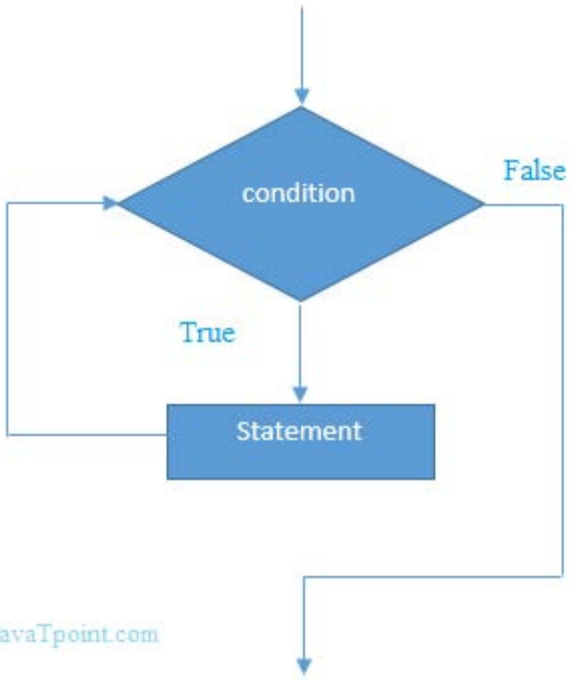
```
while(condition){
//code to be executed
}
```

Alternative Syntax

```
while(condition):
//code to be executed

endwhile;
```

Flowchart



Example

```
1. <?php
2. $n=1;
3. while($n<=10){
4. echo "$n<br/>";
5. $n++;
6. }
7. ?>
```

Output:

```
1
2
3
4
5
6
7
8
9
10
```

Alternative Example

```
<?php
$n=1;
while($n<=10):
echo "$n<br/>";
$n++;
endwhile;
?>
```

Output:

```
1
2
3
4
5
6
7
8
9
10
```

PHP Nested While Loop

We can use while loop inside another while loop in PHP, it is known as nested while loop.

In case of inner or nested while loop, nested while loop is executed fully for one outer while loop. If outer while loop is to be executed for 3 times and nested while loop for 3 times, nested while loop will be executed 9 times (3 times for 1st outer loop, 3 times for 2nd outer loop and 3 times for 3rd outer loop).

Example

```
<?php
$i=1;
while($i<=3){
$j=1;
while($j<=3){
echo "$i  $j<br/>";
$j++;
}
$i++;
    }
    ?>
```

Output:

```
1 1
```

```
1 2
1 3
2 1
2 2
2 3
3 1
3 2
3 3
```


PHP do while loop

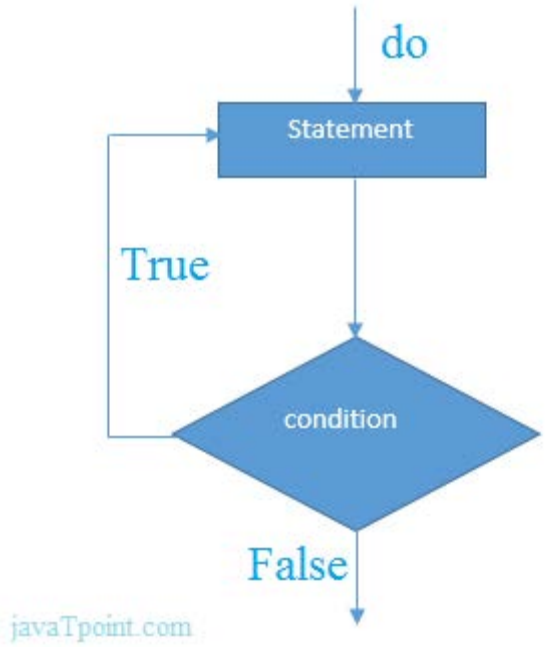
PHP do while loop can be used to traverse set of code like php while loop. The PHP do-while loop is guaranteed to run at least once.

It executes the code at least one time always because condition is checked after executing the code.

Syntax

```
1. do{
2. //code to be executed
3. }while(condition);
```

Flowchart



Example

```
1. <?php
2. $n=1;
3. do{
4. echo "$n<br/>";
5. $n++;
6. }while($n<=10);
7. ?>
```

Output:

```
1
2
3
4
5
6
7
8
9
10
```

PHP Break

PHP break statement breaks the execution of current for, while, do-while, switch and for-each loop. If you use break inside inner loop, it breaks the execution of inner loop only.

Syntax

```
1. jump statement;
2. break;
```

Flowchart

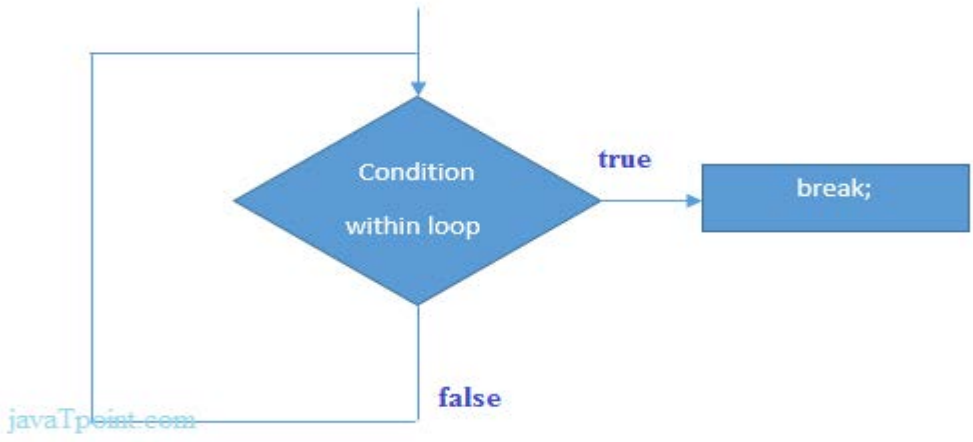


Figure: Flowchart of break statement

PHP Break: inside loop

Let's see a simple example to break the execution of for loop if value of i is equal to 5.

```
1. <?php
2. for($i=1;$i<=10;$i++){
3. echo "$i <br/>";
4. if($i==5){
5. break;
6. }
7. }
8. ?>
```

Output:

```
1
2
3
4
5
```

PHP Break: inside inner loop

The PHP break statement breaks the execution of inner loop only.

```
1. <?php
2. for($i=1;$i<=3;$i++){
3.  for($j=1;$j<=3;$j++){
4.   echo "$i   $j<br/>";
5.   if($i==2 && $j==2){
6.    break;
7.   }
8.  }
9. }
10.      ?>
```

Output:

```
1  1
1  2
1  3
2  1
2  2
3  1
3  2
3  3
```

PHP Break: inside switch statement

The PHP break statement breaks the flow of switch case also.

```
1. <?php
2. $num=200;
3. switch($num){
4. case 100:
5. echo("number is equals to 100");
6. break;
7. case 200:
8. echo("number is equal to 200");
9. break;
10.     case 50:
11.     echo("number is equal to 300");
12.     break;
13.     default:
14.     echo("number is not equal to 100, 200 or 500");
15.     }
16.     ?>
```

Output:

```
number is equal to 200
```

PHP FUNCTIONS

PHP function is a piece of code that can be reused many times. It can take input as argument list and return value. There are thousands of built-in functions in PHP.

In PHP, we can define **Conditional function**, **Function within Function** and**Recursive function** also.

Advantage of PHP Functions

Code Reusability: PHP functions are defined only once and can be invoked many times, like in other programming languages.

Less Code: It saves a lot of code because you don't need to write the logic many times. By the use of function, you can write the logic only once and reuse it.

Easy to understand: PHP functions separate the programming logic. So it is easier to understand the flow of the application because every logic is divided in the form of functions.

PHP User-defined Functions

We can declare and call user-defined functions easily. Let's see the syntax to declare user-defined functions.

Syntax

```
1. function functionname(){
2. //code to be executed
3. }
```

Note: Function name must be start with letter and underscore only like other labels in PHP. It can't be start with numbers or special symbols.

Example

File: function1.php

```
1. <?php
2. function sayHello(){
3. echo "Hello PHP Function";
4. }
5. sayHello();//calling function
6. ?>
```

Output:

```
Hello PHP Function
```

PHP Function Arguments

We can pass the information in PHP function through arguments which is separated by comma.

PHP supports **Call by Value** (default), **Call by Reference**, **Default argument values** and **Variable-length argument list**.

Let's see the example to pass single argument in PHP function.

File: functionarg.php

```
<?php
function sayHello($name){
echo "Hello $name<br/>";
}
sayHello("Sonoo");
sayHello("Vimal");
sayHello("John");
?>
```

Output:

```
Hello Sonoo
Hello Vimal
Hello John
```

Let's see the example to pass two argument in PHP function.

File: functionarg2.php

```
<?php
function sayHello($name,$age){
echo "Hello $name, you are $age years old<br/>";
}
sayHello("Sonoo",27);
sayHello("Vimal",29);
sayHello("John",23);
?>
```

Output:

```
Hello Sonoo, you are 27 years old
Hello Vimal, you are 29 years old
Hello John, you are 23 years old
```

PHP Call By Reference

Value passed to the function doesn't modify the actual value by default (call by value). But we can do so by passing value as a reference.

By default, value passed to the function is call by value. To pass value as a reference, you need to use ampersand (&) symbol before the argument name.

Let's see a simple example of call by reference in PHP.

File: functionref.php

```
1. <?php
2. function adder(&$str2)
3. {
4.    $str2 .= 'Call By Reference';
5. }
6. $str = 'Hello ';
7. adder($str);
8. echo $str;
9. ?>
```

Output:

```
Hello Call By Reference
```

PHP Function: Default Argument Value

We can specify a default argument value in function. While calling PHP function if you don't specify any argument, it will take the default argument. Let's see a simple example of using default argument value in PHP function.

File: functiondefaultarg.php

```
1. <?php
2. function sayHello($name="Sonoo"){
3. echo "Hello $name<br/>";
4. }
5. sayHello("Rajesh");
6. sayHello();//passing no value
7. sayHello("John");
8. ?>
```

Output:

```
Hello Rajesh
Hello Sonoo
Hello John
```

PHP Function: Returning Value

Let's see an example of PHP function that returns value.

File: functiondefaultarg.php

```
1. <?php
2. function cube($n){
3. return $n*$n*$n;
```

```
}
echo "Cube of 3 is: ".cube(3);
?>
```

Output:

```
Cube of 3 is: 27
```


PHP Call By Value

PHP allows you to call function by value and reference both. In case of PHP call by value, actual value is not modified if it is modified inside the function. Let's understand the concept of call by value by the help of examples.

Example 1

In this example, variable $str is passed to the adder function where it is concatenated with 'Call By Value' string. But, printing $str variable results 'Hello' only. It is because changes are done in the local variable $str2 only. It doesn't reflect to $str variable.

```
<?php
function adder($str2)
{
   $str2 .= 'Call By Value';
}
$str = 'Hello ';
adder($str);
echo $str;
?>
```

Output:

```
Hello
```

Example 2

Let's understand PHP call by value concept through another example.

```
<?php
function increment($i)
{
   $i++;
}
$i = 10;
increment($i);
echo $i;
?>
```

Output:

```
10
```

PHP Call By Reference

In case of PHP call by reference, actual value is modified if it is modified inside the function. In such case, you need to use & (ampersand) symbol with formal arguments. The & represents reference of the variable.

Let's understand the concept of call by reference by the help of examples.

Example 1

In this example, variable $str is passed to the adder function where it is concatenated with 'Call By Reference' string. Here, printing $str variable results 'This is Call By Reference'. It is because changes are done in the actual variable $str.

```
1. <?php
2. function adder(&$str2)
3. {
4.    $str2 .= 'Call By Reference';
5. }
6. $str = 'This is ';
7. adder($str);
8. echo $str;
9. ?>
```

Output:

```
This is Call By Reference
```

Example 2

Let's understand PHP call by reference concept through another example.

```
1. <?php
2. function increment(&$i)
3. {
4.    $i++;
5. }
6. $i = 10;
7. increment($i);
8. echo $i;
9. ?>
```

Output:

```
11
```

PHP Default Argument Values Function

PHP allows you to define C++ style default argument values. In such case, if you don't pass any value to the function, it will use default argument value. Let' see the simple example of using PHP default arguments in function.

Example 1

```
1. <?php
2. function sayHello($name="Ram"){
3. echo "Hello $name<br/>";
4. }
5. sayHello("Sonoo");
6. sayHello();//passing no value
7. sayHello("Vimal");
8. ?>
```

Output:

```
Hello Sonoo
Hello Ram
Hello Vimal
```

Since PHP 5, you can use the concept of default argument value with call by reference also.

Example 2

```
1. <?php
2. function greeting($first="Sonoo",$last="Jaiswal"){
3. echo "Greeting: $first $last<br/>";
4. }
5. greeting();
6. greeting("Rahul");
7. greeting("Michael","Clark");
8. ?>
```

Output:

```
Greeting: Sonoo Jaiswal
Greeting: Rahul Jaiswal
Greeting: Michael Clark
```

Example 3

```
1. <?php
2. function add($n1=10,$n2=10){
3. $n3=$n1+$n2;
4. echo "Addition is: $n3<br/>";
5. }
6. add();
7. add(20);
8. add(40,40);
9. ?>
```

Output:

```
Addition is: 20
Addition is: 30
Addition is: 80
```


PHP Variable Length Argument Function

PHP supports variable length argument function. It means you can pass 0, 1 or n number of arguments in function. To do so, you need to use 3 ellipses (dots) before the argument name.

The 3 dot concept is implemented for variable length argument since PHP 5.6.

Let's see a simple example of PHP variable length argument function.

```php
<?php
function add(...$numbers) {
   $sum = 0;
   foreach ($numbers as $n) {
      $sum += $n;
   }
   return $sum;
}

    echo add(1, 2, 3, 4);
    ?>
```

Output:

```
10
```

PHP Recursive Function

PHP also supports recursive function call like C/C++. In such case, we call current function within function. It is also known as recursion.

It is recommended to avoid recursive function call over 200 recursion level because it may smash the stack and may cause the termination of script.

Example 1: Printing number

```php
<?php
function display($number) {
   if($number<=5){
    echo "$number <br/>";
    display($number+1);
   }
}

```

```
9. display(1);
10.     ?>
```

Output:

```
1
2
3
4
5
```


Example 2 : Factorial Number

```
1. <?php
2. function factorial($n)
3. {
4.     if ($n < 0)
5.         return -1; /*Wrong value*/
6.     if ($n == 0)
7.         return 1; /*Terminating condition*/
8.     return ($n * factorial ($n -1));
9. }
10.
11.     echo factorial(5);
12.     ?>
```

Output:

```
120
```

PHP ARRAYS

PHP array is an ordered map (contains value on the basis of key). It is used to hold multiple values of similar type in a single variable.

Advantage of PHP Array

Less Code: We don't need to define multiple variables.
Easy to traverse: By the help of single loop, we can traverse all the elements of an array.
Sorting: We can sort the elements of array.

PHP Array Types

There are 3 types of array in PHP.

1. Indexed Array
2. Associative Array
3. Multidimensional Array

PHP Indexed Array

PHP index is represented by number which starts from 0. We can store number, string and object in the PHP array. All PHP array elements are assigned to an index number by default.
There are two ways to define indexed array:
1st way:

```
1. $season=array("summer","winter","spring","autumn");
```

2nd way:

```
1. $season[0]="summer";
2. $season[1]="winter";
3. $season[2]="spring";
4. $season[3]="autumn";
```

Example
File: array1.php

```
1. <?php
2. $season=array("summer","winter","spring","autumn");
3. echo "Season are: $season[0], $season[1], $season[2] and $season[3]";
4. ?>
```

Output:

```
Season are: summer, winter, spring and autumn
```

File: array2.php

```
1. <?php
2. $season[0]="summer";
```

```
3. $season[1]="winter";
4. $season[2]="spring";
5. $season[3]="autumn";
6. echo "Season are: $season[0], $season[1], $season[2] and $season[3]";
7. ?>
```

Output:

```
Season are: summer, winter, spring and autumn
```

PHP Associative Array

We can associate name with each array elements in PHP using => symbol. There are two ways to define associative array:

1st way:

```
1. $salary=array("Sonoo"=>"350000","John"=>"450000","Kartik"=>"200000"
   );
```

2nd way:

```
1. $salary["Sonoo"]="350000";
2. $salary["John"]="450000";
3. $salary["Kartik"]="200000";
```

Example

File: arrayassociative1.php

```
1. <?php
2. $salary=array("Sonoo"=>"350000","John"=>"450000","Kartik"=>"200000"
   );
3. echo "Sonoo salary: ".$salary["Sonoo"]."<br/>";
4. echo "John salary: ".$salary["John"]."<br/>";
5. echo "Kartik salary: ".$salary["Kartik"]."<br/>";
6. ?>
```

Output:

```
Sonoo salary: 350000
John salary: 450000
Kartik salary: 200000
```

File: arrayassociative2.php

```
1. <?php
2. $salary["Sonoo"]="350000";
3. $salary["John"]="450000";
4. $salary["Kartik"]="200000";
5. echo "Sonoo salary: ".$salary["Sonoo"]."<br/>";
6. echo "John salary: ".$salary["John"]."<br/>";
7. echo "Kartik salary: ".$salary["Kartik"]."<br/>";
8. ?>
```

Output:

```
Sonoo salary: 350000
John salary: 450000
```

```
Kartik salary: 200000
```

PHP Indexed Array

PHP indexed array is an array which is represented by an index number by default. All elements of array are represented by an index number which starts from 0.

PHP indexed array can store numbers, strings or any object. PHP indexed array is also known as numeric array.

Definition

There are two ways to define indexed array:

1st way:

```
1. $size=array("Big","Medium","Short");
```

2nd way:

```
1. $size[0]="Big";
2. $size[1]="Medium";
3. $size[2]="Short";
```

Example

File: array1.php

```
1. <?php
2. $size=array("Big","Medium","Short");
3. echo "Size: $size[0], $size[1] and $size[2]";
4. ?>
```

Output:

```
Size: Big, Medium and Short
```

File: array2.php

```
1. <?php
2. $size[0]="Big";
3. $size[1]="Medium";
4. $size[2]="Short";
5. echo "Size: $size[0], $size[1] and $size[2]";
6. ?>
```

Output:

```
Size: Big, Medium and Short
```

Traversing PHP Indexed Array

We can easily traverse array in PHP using foreach loop. Let's see a simple example to traverse all the elements of PHP array.

File: array3.php

```
<?php
$size=array("Big","Medium","Short");
foreach( $size as $s )
{
  echo "Size is: $s<br />";
}
?>
```

Output:

```
Size is: Big
Size is: Medium
Size is: Short
```

Count Length of PHP Indexed Array

PHP provides count() function which returns length of an array.

```
<?php
$size=array("Big","Medium","Short");
echo count($size);
?>
```

Output:

```
3
```

PHP Associative Array

PHP allows you to associate name/label with each array elements in PHP using => symbol. Such way, you can easily remember the element because each element is represented by label than an incremented number.

Definition

There are two ways to define associative array:

1st way:

```
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
```

2nd way:

```
$salary["Sonoo"]="550000";
$salary["Vimal"]="250000";
$salary["Ratan"]="200000";
```

Example

File: arrayassociative1.php

```
<?php
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
echo "Sonoo salary: ".$salary["Sonoo"]."<br/>";
echo "Vimal salary: ".$salary["Vimal"]."<br/>";
```

```
5. echo "Ratan salary: ".$salary["Ratan"]."<br/>";
6. ?>
```

Output:

```
Sonoo salary: 550000
Vimal salary: 250000
Ratan salary: 200000
```


File: arrayassociative2.php

```
1. <?php
2. $salary["Sonoo"]="550000";
3. $salary["Vimal"]="250000";
4. $salary["Ratan"]="200000";
5. echo "Sonoo salary: ".$salary["Sonoo"]."<br/>";
6. echo "Vimal salary: ".$salary["Vimal"]."<br/>";
7. echo "Ratan salary: ".$salary["Ratan"]."<br/>";
8. ?>
```

Output:

```
Sonoo salary: 550000
Vimal salary: 250000
Ratan salary: 200000
```

Traversing PHP Associative Array

By the help of PHP for each loop, we can easily traverse the elements of PHP associative array.

```
1. <?php
2. $salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000
   ");
3. foreach($salary as $k => $v) {
4. echo "Key: ".$k." Value: ".$v."<br/>";
5. }
6. ?>
```

Output:

```
Key: Sonoo Value: 550000
Key: Vimal Value: 250000
Key: Ratan Value: 200000
```

PHP Multidimensional Array

PHP multidimensional array is also known as array of arrays. It allows you to store tabular data in an array. PHP multidimensional array can be represented in the form of matrix which is represented by row * column.

Definition

```
$emp = array
  (
  array(1,"sonoo",400000),
  array(2,"john",500000),
  array(3,"rahul",300000)
  );
```

Example

Let's see a simple example of PHP multidimensional array to display following tabular data. In this example, we are displaying 3 rows and 3 columns.

Id	Name	Salary
1	sonoo	400000
2	john	500000
3	rahul	300000

File: multiarray.php

```
<?php
$emp = array
  (
  array(1,"sonoo",400000),
  array(2,"john",500000),
  array(3,"rahul",300000)
  );

for ($row = 0; $row < 3; $row++) {
        for ($col = 0; $col < 3; $col++) {
          echo $emp[$row][$col]." ";
        }
        echo "<br/>";
      }
     ?>
```

Output:

```
1 sonoo 400000
2 john 500000
```

```
3 rahul 300000
```

PHP Array Functions

PHP provides various array functions to access and manipulate the elements of array. The important PHP array functions are given below.

1) PHP array() function

PHP array() function creates and returns an array. It allows you to create indexed, associative and multidimensional arrays.

Syntax

```
array array ([ mixed $... ] )
```

Example

```
<?php
$season=array("summer","winter","spring","autumn");
echo "Season are: $season[0], $season[1], $season[2] and $season[3]";
?>
```

Output:

```
Season are: summer, winter, spring and autumn
```

2) PHP array_change_key_case() function

PHP array_change_key_case() function changes the case of all key of an array.

Note: It changes case of key only.

Syntax

```
array array_change_key_case ( array $array [, int $case = CASE_LOWER ] )
```

Example

```
<?php
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
print_r(array_change_key_case($salary,CASE_UPPER));
?>
```

Output:

```
Array ( [SONOO] => 550000 [VIMAL] => 250000 [RATAN] => 200000 )
```

Example

```
<?php
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
print_r(array_change_key_case($salary,CASE_LOWER));
?>
```

Output:

```
Array ( [sonoo] => 550000 [vimal] => 250000 [ratan] => 200000 )
```

3) PHP array_chunk() function

PHP array_chunk() function splits array into chunks. By using array_chunk() method, you can divide array into many parts.

Syntax

```
array array_chunk ( array $array , int $size [, bool $preserve_keys = false ] )
```

Example

```
<?php
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
print_r(array_chunk($salary,2));
?>
```

Output:

```
Array (
[0] => Array ( [0] => 550000 [1] => 250000 )
[1] => Array ( [0] => 200000 )
)
```

4) PHP count() function

PHP count() function counts all elements in an array.

Syntax

```
int count ( mixed $array_or_countable [, int $mode = COUNT_NORMAL ] )
```

Example

```
<?php
$season=array("summer","winter","spring","autumn");
echo count($season);
?>
```

Output:

```
4
```

5) PHP sort() function

PHP sort() function sorts all the elements in an array.

Syntax

```
bool sort ( array &$array [, int $sort_flags = SORT_REGULAR ] )
```

Example

```
<?php
$season=array("summer","winter","spring","autumn");
sort($season);
foreach( $season as $s )
{
  echo "$s<br />";
}
?>
```

Output:

```
autumn
```

```
spring
summer
winter
```


6) PHP array_reverse() function

PHP array_reverse() function returns an array containing elements in reversed order.

Syntax

```
1. array array_reverse ( array $array [, bool $preserve_keys = false ] )
```

Example

```
1. <?php
2. $season=array("summer","winter","spring","autumn");
3. $reverseseason=array_reverse($season);
4. foreach( $reverseseason as $s )
5. {
6.   echo "$s<br />";
7. }
8. ?>
```

Output:

```
autumn
spring
winter
summer
```

7) PHP array_search() function

PHP array_search() function searches the specified value in an array. It returns key if search is successful.

Syntax

```
1. mixed array_search ( mixed $needle , array $haystack [, bool $strict = fals
   e ] )
```

Example

```
1. <?php
2. $season=array("summer","winter","spring","autumn");
3. $key=array_search("spring",$season);
4. echo $key;
5. ?>
```

Output:

```
2
```

8) PHP array_intersect() function

PHP array_intersect() function returns the intersection of two array. In other words, it returns the matching elements of two array.

Syntax

```
array array_intersect ( array $array1 , array $array2 [, array $... ] )
```


Example

```
<?php
$name1=array("sonoo","john","vivek","smith");
$name2=array("umesh","sonoo","kartik","smith");
$name3=array_intersect($name1,$name2);
foreach( $name3 as $n )
{
  echo "$n<br />";
}
?>
```

Output: sonoo smith

PHP String

A PHP string is a sequence of characters i.e. used to store and manipulate text. There are 4 ways to specify string in PHP.

- single quoted
- double quoted
- heredoc syntax
- newdoc syntax (since PHP 5.3)

Single Quoted PHP String

We can create a string in PHP by enclosing text in a single quote. It is the easiest way to specify string in PHP.

```
<?php
$str='Hello text within single quote';
echo $str;
?>
```

Output:

Hello text within single quote

We can store multiple line text, special characters and escape sequences in a single quoted PHP string.

```
<?php
$str1='Hello text
multiple line
text within single quoted string';
$str2='Using double "quote" directly inside single quoted string';
$str3='Using escape sequences \n in single quoted string';
echo "$str1 <br/> $str2 <br/> $str3";
?>
```

Output:

```
Hello text multiple line text within single quoted string
Using double "quote" directly inside single quoted string
Using escape sequences \n in single quoted string
```

Note: In single quoted PHP strings, most escape sequences and variables will not be interpreted. But, we can use single quote through \' and backslash through \\ inside single quoted PHP strings.


```
1. <?php
2. $num1=10;
3. $str1='trying variable $num1';
4. $str2='trying backslash n and backslash t inside single quoted string \n \t';
5. $str3='Using single quote \'my quote\' and \\backslash';
6. echo "$str1 <br/> $str2 <br/> $str3";
7. ?>
```

Output:

```
trying variable $num1
trying backslash n and backslash t inside single quoted string \n \t
Using single quote 'my quote' and \backslash
```

Double Quoted PHP String

In PHP, we can specify string through enclosing text within double quote also. But escape sequences and variables will be interpreted using double quote PHP strings.

```
1. <?php
2. $str="Hello text within double quote";
3. echo $str;
4. ?>
```

Output:

```
Hello text within double quote
```

Now, you **can't use double quote directly** inside double quoted string.

```
1. <?php
2. $str1="Using double "quote" directly inside double quoted string";
3. echo $str1;
4. ?>
```

Output:

```
Parse error: syntax error, unexpected 'quote' (T_STRING) in C:\wamp\www\string1.php on line 2
```

We **can store multiple line text, special characters and escape sequences** in a double quoted PHP string.

```
1. <?php
2. $str1="Hello text
3. multiple line
4. text within double quoted string";
5. $str2="Using double \"quote\" with backslash inside double quoted string";
```

```
6. $str3="Using escape sequences \n in double quoted string";
7. echo "$str1 <br/> $str2 <br/> $str3";
8. ?>
```

Output:

```
Hello text multiple line text within double quoted string
Using double "quote" with backslash inside double quoted string
Using escape sequences in double quoted string
```


In double quoted strings, **variable will be interpreted**.

```
1. <?php
2. $num1=10;
3. echo "Number is: $num1";
4. ?>
```

Output:

```
Number is: 10
```

PHP STRING FUNCTIONS

PHP provides various string functions to access and manipulate strings. A list of important PHP string functions are given below.

1) PHP strtolower() function

The strtolower() function returns string in lowercase letter.

Syntax

```
string strtolower ( string $string )
```

Example

```
<?php
$str="My name is KHAN";
$str=strtolower($str);
echo $str;
?>
```

Output:

```
my name is khan
```

2) PHP strtoupper() function

The strtoupper() function returns string in uppercase letter.

Syntax

```
string strtoupper ( string $string )
```

Example

```
<?php
$str="My name is KHAN";
$str=strtoupper($str);
echo $str;
?>
```

Output:

```
MY NAME IS KHAN
```

3) PHP ucfirst() function

The ucfirst() function returns string converting first character into uppercase. It doesn't change the case of other characters.

Syntax

```
string ucfirst ( string $str )
```

Example

```
<?php
$str="my name is KHAN";
$str=ucfirst($str);
echo $str;
?>
```

Output:

```
My name is KHAN
```

4) PHP lcfirst() function

The lcfirst() function returns string converting first character into lowercase. It doesn't change the case of other characters.

Syntax

```
1. string lcfirst ( string $str )
```

Example

```
1. <?php
2. $str="MY name IS KHAN";
3. $str=lcfirst($str);
4. echo $str;
5. ?>
```

Output:

```
mY name IS KHAN
```

5) PHP ucwords() function

The ucwords() function returns string converting first character of each word into uppercase.

Syntax

```
1. string ucwords ( string $str )
```

Example

```
1. <?php
2. $str="my name is Sonoo jaiswal";
3. $str=ucwords($str);
4. echo $str;
5. ?>
```

Output:

```
My Name Is Sonoo Jaiswal
```

6) PHP strrev() function

The strrev() function returns reversed string.

Syntax

```
1. string strrev ( string $string )
```

Example

```
1. <?php
2. $str="my name is Sonoo jaiswal";
3. $str=strrev($str);
4. echo $str;
5. ?>
```

Output:

```
lawsiaj oonoS si eman ym
```

7) PHP strlen() function

The strlen() function returns length of the string.

Syntax

```
1. int strlen ( string $string )
```

Example

```
1. <?php
2. $str="my name is Sonoo jaiswal";
3. $str=strlen($str);
4. echo $str;
5. ?>
```

Output:

```
24
```

PHP MATH

PHP provides many predefined math constants and functions that can be used to perform mathematical operations.

PHP Math: abs() function

The abs() function returns absolute value of given number. It returns an integer value but if you pass floating point value, it returns a float value.

Syntax

```
number abs ( mixed $number )
```

Example

```
<?php
echo (abs(-7)."<br/>"); // 7 (integer)
echo (abs(7)."<br/>"); //7 (integer)
echo (abs(-7.2)."<br/>"); //7.2 (float/double)
?>
```

Output:

```
7
7
7.2
```

PHP Math: ceil() function

The ceil() function rounds fractions up.

Syntax

```
float ceil ( float $value )
```

Example

```
<?php
echo (ceil(3.3)."<br/>");// 4
echo (ceil(7.333)."<br/>");// 8
echo (ceil(-4.8)."<br/>");// -4
?>
```

Output:

```
4
8
-4
```

PHP Math: floor() function

The floor() function rounds fractions down.

Syntax

```
float floor ( float $value )
```

Example

```
<?php
echo (floor(3.3)."<br/>");// 3
echo (floor(7.333)."<br/>");// 7
echo (floor(-4.8)."<br/>");// -5
?>
```

Output:

```
3
7
-5
```

PHP Math: sqrt() function

The sqrt() function returns square root of given argument.

Syntax

```
1. float sqrt ( float $arg )
```

Example

```
1. <?php
2. echo (sqrt(16)."<br/>");// 4
3. echo (sqrt(25)."<br/>");// 5
4. echo (sqrt(7)."<br/>");// 2.6457513110646
5. ?>
```

Output:

```
4
5
2.6457513110646
```

PHP Math: decbin() function

The decbin() function converts decimal number into binary. It returns binary number as a string.

Syntax

```
1. string decbin ( int $number )
```

Example

```
1. <?php
2. echo (decbin(2)."<br/>");// 10
3. echo (decbin(10)."<br/>");// 1010
4. echo (decbin(22)."<br/>");// 10110
5. ?>
```

Output:

```
10
1010
10110
```

PHP Math: dechex() function

The dechex() function converts decimal number into hexadecimal. It returns hexadecimal representation of given number as a string.

Syntax

```
1. string dechex ( int $number )
```

Example

```
1. <?php
2. echo (dechex(2)."<br/>");// 2
3. echo (dechex(10)."<br/>");// a
4. echo (dechex(22)."<br/>");// 16
5. ?>
```

Output:

```
2
a
16
```


PHP Math: decoct() function

The decoct() function converts decimal number into octal. It returns octal representation of given number as a string.

Syntax

1. string decoct (int $number)

Example

```
1. <?php
2. echo (decoct(2)."<br/>");// 2
3. echo (decoct(10)."<br/>");// 12
4. echo (decoct(22)."<br/>");// 26
5. ?>
```

Output:

```
2
12
26
```

PHP Math: base_convert() function

The base_convert() function allows you to convert any base number to any base number. For example, you can convert hexadecimal number to binary, hexadecimal to octal, binary to octal, octal to hexadecimal, binary to decimal etc.

Syntax

1. string base_convert (string $number , int $frombase , int $tobase)

Example

```
1. <?php
2. $n1=10;
3. echo (base_convert($n1,10,2)."<br/>");// 1010
4. ?>
```

Output:

```
1010
```

PHP Math: bindec() function

The bindec() function converts binary number into decimal.

Syntax

```
number bindec ( string $binary_string )
```

Example

```
<?php
echo (bindec(10)."<br/>");// 2
echo (bindec(1010)."<br/>");// 10
echo (bindec(1011)."<br/>");// 11
?>
```

Output:

```
2
10
11
```

PHP Math Functions

Let's see the list of important PHP math functions.

- abs()
- acos()
- acosh()
- asin()
- asinh()
- atan()
- atan2()
- atanh()
- base_convert()
- bindec()
- ceil()
- cos()
- cosh()
- decbin()
- dechex()
- decoct()
- deg2rad()
- exp()
- expm1()
- floor()
- fmod()
- getrandmax()
- hexdec()
- hypot()
- is_finite()
- is_infinite()
- is_nan()
- lcg_value()

- log()
- log10()
- log1p()
- max()
- min()
- mt_getrandmax()
- mt_rand()
- mt_srand()
- octdec()
- pi()
- pow()
- rad2deg()
- rand()
- round()
- sin()
- sinh()
- sqrt()
- srand()
- tan()
- tanh()

PHP FORM

We can create and use forms in PHP. To get form data, we need to use PHP superglobals $_GET and $_POST.

The form request may be get or post. To retrieve data from get request, we need to use $_GET, for post request $_POST.

PHP Get Form

Get request is the default form request. The data passed through get request is visible on the URL browser so it is not secured. You can send limited amount of data through get request.

Let's see a simple example to receive data from get request in PHP.

File: form1.html

```
1. <form action="welcome.php" method="get">
2. Name: <input type="text" name="name"/>
3. <input type="submit" value="visit"/>
4. </form>
```

File: welcome.php

```
1. <?php
2. $name=$_GET["name"];//receiving name field value in $name variable
3. echo "Welcome, $name";
4. ?>
```

PHP Post Form

Post request is widely used to submit form that have large amount of data such as file upload, image upload, login form, registration form etc.

The data passed through post request is not visible on the URL browser so it is secured. You can send large amount of data through post request.

Let's see a simple example to receive data from post request in PHP.

File: form1.html

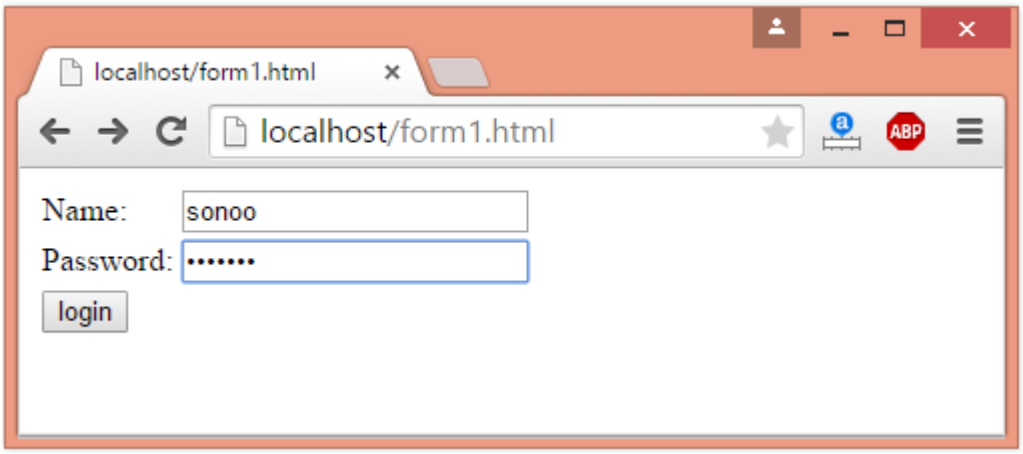
```
1. <form action="login.php" method="post">
2. <table>
3. <tr><td>Name:</td><td> <input type="text" name="name"/></td></tr
   >
4. <tr><td>Password:</td><td> <input type="password" name="password"/
   ></td></tr>
5. <tr><td colspan="2"><input type="submit" value="login"/>  </td></tr>
6. </table>
7. </form>
```

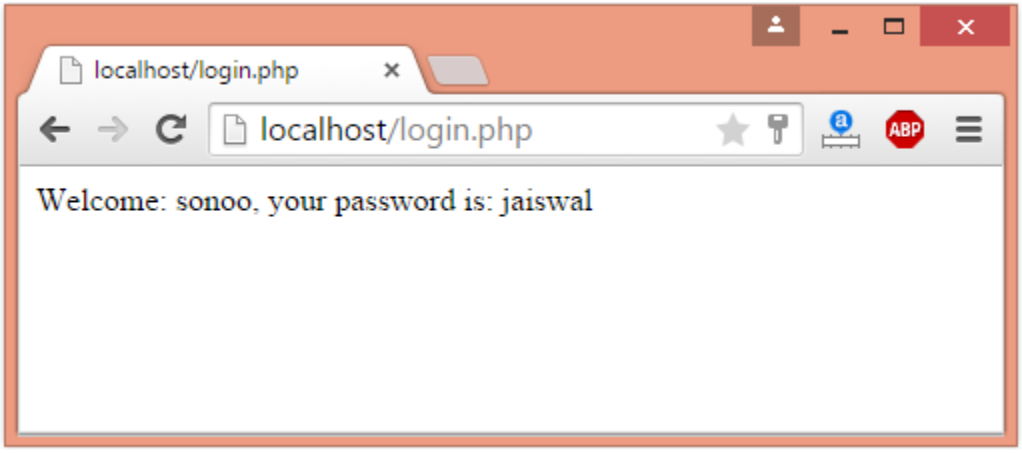
File: login.php

```
1. <?php
2. $name=$_POST["name"];//receiving name field value in $name variable
3. $password=$_POST["password"];//receiving password field value in $passw
   ord variable
4.
```

5. echo "Welcome: $name, your password is: $password";
6. ?>

Output:





PHP INCLUDE/REQUIRE

PHP allows you to include file so that a page content can be reused many times. There are two ways to include file in PHP.

1. include
2. require

Advantage

Code Reusability: By the help of include and require construct, we can reuse HTML code or PHP script in many PHP scripts.

PHP include example

PHP include is used to include file on the basis of given path. You may use relative or absolute path of the file. Let's see a simple PHP include example.

File: menu.html

```
1. <a href="http://www.javatpoint.com">Home</a> |
2. <a href="http://www.javatpoint.com/php-tutorial">PHP</a> |
3. <a href="http://www.javatpoint.com/java-tutorial">Java</a> |
4. <a href="http://www.javatpoint.com/html-tutorial">HTML</a>
```

File: include1.php

```
1. <?php include("menu.html"); ?>
2. <h1>This is Main Page</h1>
```

Output:

Home | PHP | Java | HTML

This is Main Page

PHP require example

PHP require is similar to include. Let's see a simple PHP require example.

File: menu.html

```
1. <a href="http://www.javatpoint.com">Home</a> |
2. <a href="http://www.javatpoint.com/php-tutorial">PHP</a> |
3. <a href="http://www.javatpoint.com/java-tutorial">Java</a> |
4. <a href="http://www.javatpoint.com/html-tutorial">HTML</a>
```

File: require1.php

```
1. <?php require("menu.html"); ?>
2. <h1>This is Main Page</h1>
```

Output:

Home | PHP | Java | HTML

This is Main Page

PHP INCLUDE VS REQUIRE

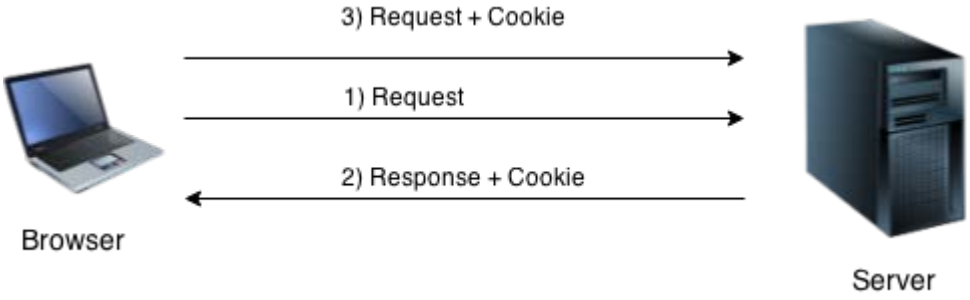
If file is missing or inclusion fails, **include** allows the *script to continue* but **require***halts the script* producing a fatal E_COMPILE_ERROR level error.

STATE MANAGEMENT

PHP Cookie

PHP cookie is a small piece of information which is stored at client browser. It is used to recognize the user.

Cookie is created at server side and saved to client browser. Each time when client sends request to the server, cookie is embedded with request. Such way, cookie can be received at the server side.



In short, cookie can be created, sent and received at server end.

Note: PHP Cookie must be used before <html> tag.

PHP setcookie() function

PHP setcookie() function is used to set cookie with HTTP response. Once cookie is set, you can access it by $_COOKIE superglobal variable.

Syntax

1. bool setcookie (string $name [, string $value [, int $expire = 0 [, string $pa
2. th
3. [, string $domain [, bool $secure = false [, bool $httponly = false]]]]])

Example

```
1. setcookie("CookieName", "CookieValue");/* defining name and value only*/

2. setcookie("CookieName", "CookieValue", time()+1*60*60);//using expiry in
   1 hour(1*60*60 seconds or 3600 seconds)
3. setcookie("CookieName", "CookieValue", time()+1*60*60, "/mypath/", "my
   domain.com", 1);
```

PHP $_COOKIE

PHP $_COOKIE superglobal variable is used to get cookie.

Example

```
1. $value=$_COOKIE["CookieName"];//returns cookie value
```

PHP Cookie Example

File: cookie1.php

```
1. <?php
2. setcookie("user", "Sonoo");
3. ?>
4. <html>
```

```
5. <body>
6. <?php
7. if(!isset($_COOKIE["user"])) {
8.     echo "Sorry, cookie is not found!";
9. } else {
       echo "<br/>Cookie Value: " . $_COOKIE["user"];
10.                                   }
11.                                   ?>
12.                                   </body>
13.                                   </html>
```

Output:

```
Sorry, cookie is not found!
```

Firstly cookie is not set. But, if you *refresh* the page, you will see cookie is set now.

Output:

```
Cookie Value: Sonoo
```

PHP Delete Cookie

If you set the expiration date in past, cookie will be deleted.

File: cookie1.php

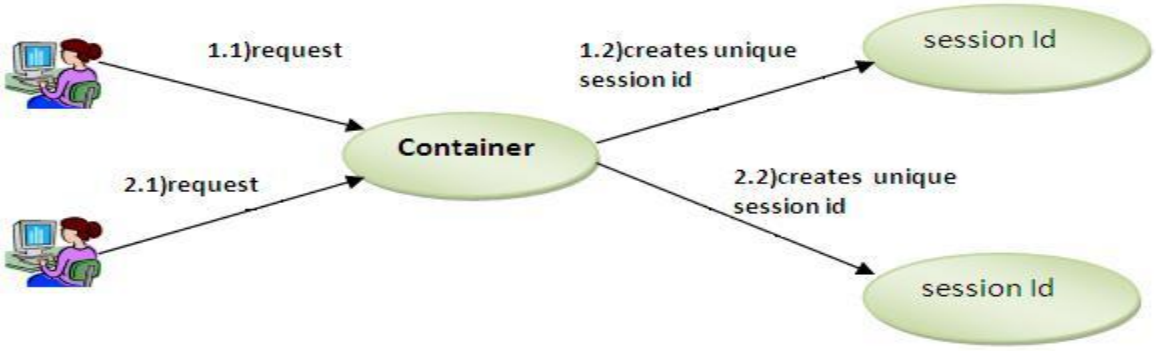
```
1. <?php
2. setcookie ("CookieName", "", time() -
    3600);// set the expiration date to one hour ago
3. ?>
```

PHP Session

PHP session is used to store and pass information from one page to another temporarily (until user close the website).

PHP session technique is widely used in shopping websites where we need to store and pass cart information e.g. username, product code, product name, product price etc from one page to another.

PHP session creates unique user id for each browser to recognize the user and avoid conflict between multiple browsers.



PHP session_start() function

PHP session_start() function is used to start the session. It starts a new or resumes existing session. It returns existing session if session is created already. If session is not available, it creates and returns new session.

Syntax

```
1. bool session_start ( void )
```

Example

```
1. session_start();
```

PHP $_SESSION

PHP $_SESSION is an associative array that contains all session variables. It is used to set and get session variable values.

Example: Store information

```
1. $_SESSION["user"] = "Sachin";
```

Example: Get information

```
1. echo $_SESSION["user"];
```

PHP Session Example

File: session1.php

```
1. <?php
2. session_start();
3. ?>
4. <html>
5. <body>
6. <?php
7. $_SESSION["user"] = "Sachin";
8. echo "Session information are set successfully.<br/>";
9. ?>
10.      <a href="session2.php">Visit next page</a>
11.      </body>
12.      </html>
```

File: session2.php

```
1. <?php
2. session_start();
3. ?>
4. <html>
5. <body>
6. <?php
7. echo "User is: ".$_SESSION["user"];
8. ?>
9. </body>
10.      </html>
```

PHP Session Counter Example

File: sessioncounter.php

```
1. <?php
2.   session_start();
3.
4.   if (!isset($_SESSION['counter'])) {
5.     $_SESSION['counter'] = 1;
6.   } else {
7.     $_SESSION['counter']++;
8.   }
9.   echo ("Page Views: ".$_SESSION['counter']);
10.      ?>
```


PHP Destroying Session

PHP session_destroy() function is used to destroy all session variables completely.

File: session3.php

```
1. <?php
2. session_start();
3. session_destroy();
4. ?>
```

PHP FILE HANDLING

PHP File System allows us to create file, read file line by line, read file character by character, write file, append file, delete file and close file.

PHP Open File - fopen()

The PHP fopen() function is used to open a file.

Syntax

```
1. resource fopen ( string $filename , string $mode [, bool $use_include_path
   = false [, resource $context ]] )
```

Example

```
1. <?php
2. $handle = fopen("c:\\folder\\file.txt", "r");
3. ?>
```

PHP Close File - fclose()

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

Syntax

```
1. ool fclose ( resource $handle )
```

Example

```
1. <?php
2. fclose($handle);
3. ?>
```

PHP Read File - fread()

The PHP fread() function is used to read the content of the file. It accepts two arguments: resource and file size.

Syntax

```
1. string fread ( resource $handle , int $length )
```

Example

```
1. <?php
2. $filename = "c:\\myfile.txt";
3. $handle = fopen($filename, "r");//open file in read mode
4.
5. $contents = fread($handle, filesize($filename));//read file
6.
7. echo $contents;//printing data of file
8. fclose($handle);//close file
9. ?>
```

Output

```
hello php file
```

PHP Write File - fwrite()

The PHP fwrite() function is used to write content of the string into file.

Syntax

```
1. int fwrite ( resource $handle , string $string [, int $length ] )
```

Example

```
1. <?php
2. $fp = fopen('data.txt', 'w');//open file in write mode
3. fwrite($fp, 'hello ');
4. fwrite($fp, 'php file');
5. fclose($fp);
6.
7. echo "File written successfully";
8. ?>
```

Output

```
File written successfully
```

PHP Delete File - unlink()

The PHP unlink() function is used to delete file.

Syntax

```
1. bool unlink ( string $filename [, resource $context ] )
```

Example

```
1. <?php
2. unlink('data.txt');
3.
4. echo "File deleted successfully";
5. ?>
```

PHP Open File

PHP fopen() function is used to open file or URL and returns resource. The fopen() function accepts two arguments: $filename and $mode. The $filename represents the file to be opended and $mode represents the file mode for example read-only, read-write, write-only etc.

Syntax

```
1. resource fopen ( string $filename , string $mode [, bool $use_include_path
   = false [, resource $context ]] )
```

PHP Open File Mode

Mode	Description
r	Opens file in **read-only** mode. It places the file pointer at the beginning of the file.
r+	Opens file in **read-write** mode. It places the file pointer at the beginning of the file.
w	Opens file in **write-only** mode. It places the file pointer to the beginning of the file and truncates the file to zero length. If file is not found, it creates a new file.
w+	Opens file in **read-write** mode. It places the file pointer to the beginning of the file and truncates the file to zero length. If file is not found, it creates a new file.
a	Opens file in **write-only** mode. It places the file pointer to the end of the file. If file is not found, it creates a new file.
a+	Opens file in **read-write** mode. It places the file pointer to the end of the file. If file is not found, it creates a new file.
x	Creates and opens file in **write-only** mode. It places the file pointer at the beginning of the file. If file is found, fopen() function returns FALSE.
x+	It is same as x but it creates and opens file in **read-write** mode.
c	Opens file in **write-only** mode. If the file does not exist, it is created. If it exists, it is neither truncated (as opposed to 'w'), nor the call to this function fails (as is the case with 'x'). The file pointer is positioned on the beginning of the file
c+	It is same as c but it opens file in **read-write** mode.

PHP Open File Example

```
1. <?php
2. $handle = fopen("c:\\folder\\file.txt", "r");
3. ?>
```

PHP Read File

PHP provides various functions to read data from file. There are different functions that allow you to read all file data, read data line by line and read data character by character.

The available PHP file read functions are given below.

- fread()
- fgets()
- fgetc()

PHP Read File - fread()

The PHP fread() function is used to read data of the file. It requires two arguments: file resource and file size.

Syntax

```
string fread (resource $handle , int $length )
```

$handle represents file pointer that is created by fopen() function.

$length represents length of byte to be read.

Example

```
<?php
$filename = "c:\\file1.txt";
$fp = fopen($filename, "r");//open file in read mode

$contents = fread($fp, filesize($filename));//read file

echo "<pre>$contents</pre>";//printing data of file
fclose($fp);//close file
?>
```

Output

```
this is first line
this is another line
this is third line
```

PHP Read File - fgets()

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

Syntax

```
string fgets ( resource $handle [, int $length ] )
```

Example

```
<?php
$fp = fopen("c:\\file1.txt", "r");//open file in read mode
echo fgets($fp);
fclose($fp);
?>
```

Output

```
this is first line
```

PHP Read File - fgetc()

The PHP fgetc() function is used to read single character from the file. To get all data using fgetc() function, use !feof() function inside the while loop.

Syntax

```
1. string fgetc ( resource $handle )
```

Example

```
1. <?php
2. $fp = fopen("c:\\file1.txt", "r");//open file in read mode
3. while(!feof($fp)) {
4.   echo fgetc($fp);
5. }
6. fclose($fp);
7. ?>
```

Output

```
this is first line this is another line this is third line
```

PHP Write File

PHP fwrite() and fputs() functions are used to write data into file. To write data into file, you need to use w, r+, w+, x, x+, c or c+ mode.

PHP Write File - fwrite()

The PHP fwrite() function is used to write content of the string into file.

Syntax

```
1. int fwrite ( resource $handle , string $string [, int $length ] )
```

Example

```
1. <?php
2. $fp = fopen('data.txt', 'w');//opens file in write-only mode
3. fwrite($fp, 'welcome ');
4. fwrite($fp, 'to php file write');
5. fclose($fp);
6.
7. echo "File written successfully";
8. ?>
```

Output: data.txt

```
welcome to php file write
```

PHP Overwriting File

If you run the above code again, it will erase the previous data of the file and writes the new data. Let's see the code that writes only new data into data.txt file.

```
1. <?php
2. $fp = fopen('data.txt', 'w');//opens file in write-only mode
3. fwrite($fp, 'hello');
4. fclose($fp);
5.
```

```
6. echo "File written successfully";
7. ?>
```

Output: data.txt

```
hello
```

PHP Append to File

If you use **a** mode, it will not erase the data of the file. It will write the data at the end of the file. Visit the next page to see the example of appending data into file.

PHP Append to File

You can append data into file by using a or a+ mode in fopen() function. Let's see a simple example that appends data into data.txt file.

Let's see the data of file first.

data.txt

```
welcome to php file write
```

PHP Append to File - fwrite()

The PHP fwrite() function is used to write and append data into file.

Example

```
1. <?php
2. $fp = fopen('data.txt', 'a');//opens file in append mode
3. fwrite($fp, ' this is additional text ');
4. fwrite($fp, 'appending data');
5. fclose($fp);
6.
7. echo "File appended successfully";
8. ?>
```

Output: data.txt

```
welcome to php file write this is additional text appending data
```

PHP Delete File

In PHP, we can delete any file using unlink() function. The unlink() function accepts one argument only: file name. It is similar to UNIX C unlink() function.

PHP unlink() generates E_WARNING level error if file is not deleted. It returns TRUE if file is deleted successfully otherwise FALSE.

Syntax

```
1. bool unlink ( string $filename [, resource $context ] )
```

$filename represents the name of the file to be deleted.

PHP Delete File Example

```
1. <?php
2. $status=unlink('data.txt');
3. if($status){
4. echo "File deleted successfully";
5. }else{
6. echo "Sorry!";
7. }
8. ?>
```

Output

```
File deleted successfully
```

PHP FILE UPLOAD

PHP allows you to upload single and multiple files through few lines of code only.

PHP file upload features allows you to upload binary and text files both. Moreover, you can have the full control over the file to be uploaded through PHP authentication and file operation functions.

PHP $_FILES

The PHP global $_FILES contains all the information of file. By the help of $_FILES global, we can get file name, file type, file size, temp file name and errors associated with file.

Here, we are assuming that file name is *filename*.

$_FILES['filename']['name']

returns file name.

$_FILES['filename']['type']

returns MIME type of the file.

$_FILES['filename']['size']

returns size of the file (in bytes).

$_FILES['filename']['tmp_name']

returns temporary file name of the file which was stored on the server.

$_FILES['filename']['error']

returns error code associated with this file.

move_uploaded_file() function

The move_uploaded_file() function moves the uploaded file to a new location. The move_uploaded_file() function checks internally if the file is uploaded thorough the POST request. It moves the file if it is uploaded through the POST request.

Syntax

1. bool move_uploaded_file (string $filename , string $destination)

PHP File Upload Example

File: uploadform.html

```
1. <form action="uploader.php" method="post" enctype="multipart/form-
   data">
2.     Select File:
3.     <input type="file" name="fileToUpload"/>
4.     <input type="submit" value="Upload Image" name="submit"/>
5. </form>
```

File: uploader.php

```
1. <?php
2. $target_path = "e:/";
3. $target_path = $target_path.basename( $_FILES['fileToUpload']['name']);
4.
5. if(move_uploaded_file($_FILES['fileToUpload']['tmp_name'], $target_path))
   {
6.     echo "File uploaded successfully!";
7. } else{
8.     echo "Sorry, file not uploaded, please try again!";
9. }
10.       ?>
```

PHP Download File

PHP enables you to download file easily using built-in readfile() function. The readfile() function reads a file and writes it to the output buffer.

PHP readfile() function

Syntax

```
1. int readfile ( string $filename [, bool $use_include_path = false [, resource $
   context ]] )
```

$filename: represents the file name
$use_include_path: it is the optional parameter. It is by default false. You can set it to true to the search the file in the included_path.
$context: represents the context stream resource.
int: it returns the number of bytes read from the file.

PHP Download File Example: Text File

File: download1.php

```
1. <?php
2. $file_url = 'http://www.javatpoint.com/f.txt';
3. header('Content-Type: application/octet-stream');
4. header("Content-Transfer-Encoding: utf-8");
5. header("Content-
   disposition: attachment; filename=\"" . basename($file_url) . "\"");
```

```
6. readfile($file_url);
7. ?>
```

PHP Download File Example: Binary File

File: download2.php

```
1. <?php
2. $file_url = 'http://www.myremoteserver.com/file.exe';
3. header('Content-Type: application/octet-stream');
4. header("Content-Transfer-Encoding: Binary");
5. header("Content-
   disposition: attachment; filename=\"" . basename($file_url) . "\"");
6. readfile($file_url);
7. ?>
```

PHP MAIL

PHP mail() function is used to send email in PHP. You can send text message, html message and attachment with message using PHP mail() function.

PHP mail() function

Syntax

1. bool mail (string $to , string $subject , string $message [, string $additional_headers [, string $additional_parameters]])

$to: specifies receiver or receivers of the mail. The receiver must be specified one of the following forms.

- user@example.com
- user@example.com, anotheruser@example.com
- User <user@example.com>
- User <user@example.com>, Another User <anotheruser@example.com>

$subject: represents subject of the mail.

$message: represents message of the mail to be sent.

Note: Each line of the message should be separated with a CRLF (\r\n) and lines should not be larger than 70 characters.

$additional_headers (optional): specifies the additional headers such as From, CC, BCC etc. Extra additional headers should also be separated with CRLF (\r\n).

PHP Mail Example

File: mailer.php

```
1. <?php
2.   ini_set("sendmail_from", "sonoojaiswal@javatpoint.com");
3.   $to = "sonoojaiswal1987@gmail.com";//change receiver address
4.   $subject = "This is subject";
5.   $message = "This is simple text message.";
6.   $header = "From:sonoojaiswal@javatpoint.com \r\n";
7.
8.   $result = mail ($to,$subject,$message,$header);
9.
10.        if( $result == true ){
11.          echo "Message sent successfully...";
12.        }else{
13.          echo "Sorry, unable to send mail...";
14.        }
15.     ?>
```

If you run this code on the live server, it will send an email to the specified receiver.

PHP Mail: Send HTML Message

To send HTML message, you need to mention Content-type **text/html** in the message header.

```
<?php
  $to = "abc@example.com";//change receiver address
  $subject = "This is subject";
  $message = "<h1>This is HTML heading</h1>";

  $header = "From:xyz@example.com \r\n";
  $header .= "MIME-Version: 1.0 \r\n";
  $header .= "Content-type: text/html;charset=UTF-8 \r\n";

        $result = mail ($to,$subject,$message,$header);

        if( $result == true ){
          echo "Message sent successfully...";
        }else{
          echo "Sorry, unable to send mail...";
        }
     ?>
```

PHP Mail: Send Mail with Attachment

To send message with attachment, you need to mention many header information which is used in the example given below.

```
<?php
  $to = "abc@example.com";
  $subject = "This is subject";
  $message = "This is a text message.";
  # Open a file
  $file = fopen("/tmp/test.txt", "r" );//change your file location
  if( $file == false )
  {
    echo "Error in opening file";
          exit();
        }
        # Read the file into a variable
        $size = filesize("/tmp/test.txt");
        $content = fread( $file, $size);

        # encode the data for safe transit
        # and insert \r\n after every 76 chars.
        $encoded_content = chunk_split( base64_encode($content));
```

```
19.
20.       # Get a random 32 bit number using time() as seed.
21.       $num = md5( time() );
22.
23.       # Define the main headers.
24.       $header = "From:xyz@example.com\r\n";
25.       $header .= "MIME-Version: 1.0\r\n";
26.       $header .= "Content-Type: multipart/mixed; ";
27.       $header .= "boundary=$num\r\n";
28.       $header .= "--$num\r\n";
29.
30.       # Define the message section
31.       $header .= "Content-Type: text/plain\r\n";
32.       $header .= "Content-Transfer-Encoding:8bit\r\n\n";
33.       $header .= "$message\r\n";
34.       $header .= "--$num\r\n";
35.
36.       # Define the attachment section
37.       $header .= "Content-Type:  multipart/mixed; ";
38.       $header .= "name=\"test.txt\"\r\n";
39.       $header .= "Content-Transfer-Encoding:base64\r\n";
40.       $header .= "Content-Disposition:attachment; ";
41.       $header .= "filename=\"test.txt\"\r\n\n";
42.       $header .= "$encoded_content\r\n";
43.       $header .= "--$num--";
44.
45.       # Send email now
46.       $result = mail ( $to, $subject, "", $header );
47.       if( $result == true ){
48.          echo "Message sent successfully...";
49.        }else{
50.          echo "Sorry, unable to send mail...";
51.        }
52.     ?>
```

PHP MySQL Connect

Since PHP 5.5, **mysql_connect()** extension is *deprecated*. Now it is recommended to use one of the 2 alternatives.

- **mysqli_connect()**
- **PDO::__construct()**

PHP mysqli_connect()

PHP **mysqli_connect() function** is used to connect with MySQL database. It returns*resource* if connection is established or *null*.

Syntax

```
1. resource mysqli_connect (server, username, password)
```

PHP mysqli_close()

PHP **mysqli_close() function** is used to disconnect with MySQL database. It returns*true* if connection is closed or *false*.

Syntax

```
1. bool mysqli_close(resource $resource_link)
```

PHP MySQL Connect Example

Example

```
1. <?php
2. $host = 'localhost:3306';
3. $user = '';
4. $pass = '';
5. $conn = mysqli_connect($host, $user, $pass);
6. if(! $conn )
7. {
8.   die('Could not connect: ' . mysqli_error());
9. }
10.       echo 'Connected successfully';
11.       mysqli_close($conn);
12.       ?>
```

Output:

```
Connected successfully
```

PHP MySQL Create Database

Since PHP 4.3, **mysql_create_db()** function is *deprecated*. Now it is recommended to use one of the 2 alternatives.

- **mysqli_query()**
- **PDO::__query()**

PHP MySQLi Create Database Example

Example

```
<?php
$host = 'localhost:3306';
$user = '';
$pass = '';
$conn = mysqli_connect($host, $user, $pass);
if(! $conn )
{
  die('Could not connect: ' . mysqli_connect_error());
}
      echo 'Connected successfully<br/>';

      $sql = 'CREATE Database mydb';
      if(mysqli_query( $conn,$sql)){
        echo "Database mydb created successfully.";
      }else{
      echo "Sorry, database creation failed ".mysqli_error($conn);
      }
      mysqli_close($conn);
      ?>
```

Output:

```
Connected successfully
Database mydb created successfully.
```

PHP MySQL Create Table

PHP mysql_query() function is used to create table. Since PHP 5.5, **mysql_query()**function is *deprecated*. Now it is recommended to use one of the 2 alternatives.

- **mysqli_query()**
- **PDO::__query()**

PHP MySQLi Create Table Example

Example

```
<?php
$host = 'localhost:3306';
$user = '';
$pass = '';
$dbname = 'test';

$conn = mysqli_connect($host, $user, $pass,$dbname);
if(!$conn){
  die('Could not connect: '.mysqli_connect_error());
      }
      echo 'Connected successfully<br/>';
```

```
12.
13.       $sql = "create table emp5(id INT AUTO_INCREMENT,name VARCHAR(
    20) NOT NULL,
14.       emp_salary INT NOT NULL,primary key (id))";
15.       if(mysqli_query($conn, $sql)){
16.        echo "Table emp5 created successfully";
17.       }else{
18.       echo "Could not create table: ". mysqli_error($conn);
19.       }
20.
21.       mysqli_close($conn);
22.       ?>
```

Output:

```
Connected successfully
Table emp5 created successfully
```

PHP MySQL Insert Record

PHP mysql_query() function is used to insert record in a table. Since PHP 5.5,**mysql_query()** function is *deprecated*. Now it is recommended to use one of the 2 alternatives.

- **mysqli_query()**
- **PDO::__query()**

PHP MySQLi Insert Record Example

Example

```
1. <?php
2. $host = 'localhost:3306';
3. $user = '';
4. $pass = '';
5. $dbname = 'test';
6.
7. $conn = mysqli_connect($host, $user, $pass,$dbname);
8. if(!$conn){
9.   die('Could not connect: '.mysqli_connect_error());
10.       }
11.       echo 'Connected successfully<br/>';
12.
13.       $sql = 'INSERT INTO emp4(name,salary) VALUES ("sonoo", 9000)';
14.       if(mysqli_query($conn, $sql)){
15.        echo "Record inserted successfully";
16.       }else{
17.       echo "Could not insert record: ". mysqli_error($conn);
18.       }
19.
20.       mysqli_close($conn);
```

```
21.     ?>
```

Output:

```
Connected successfully
Record inserted successfully
```

PHP MySQL Update Record

PHP mysql_query() function is used to update record in a table. Since PHP 5.5,**mysql_query()** function is *deprecated*. Now it is recommended to use one of the 2 alternatives.

- **mysqli_query()**
- **PDO::__query()**

PHP MySQLi Update Record Example

Example

```
1. <?php
2. $host = 'localhost:3306';
3. $user = '';
4. $pass = '';
5. $dbname = 'test';
6.
7. $conn = mysqli_connect($host, $user, $pass,$dbname);
8. if(!$conn){
9.   die('Could not connect: '.mysqli_connect_error());
10.      }
11.      echo 'Connected successfully<br/>';
12.
13.      $id=2;
14.      $name="Rahul";
15.      $salary=80000;
16.      $sql = "update emp4 set name=\"$name\", salary=$salary where id=
   $id";
17.      if(mysqli_query($conn, $sql)){
18.       echo "Record updated successfully";
19.      }else{
20.      echo "Could not update record: ". mysqli_error($conn);
21.      }
22.
23.      mysqli_close($conn);
24.      ?>
```

Output:

```
Connected successfully
Record updated successfully
```

PHP MySQL Delete Record

PHP mysql_query() function is used to delete record in a table. Since PHP 5.5,**mysql_query()** function is *deprecated*. Now it is recommended to use one of the 2 alternatives.

- **mysqli_query()**
- **PDO::__query()**

PHP MySQLi Delete Record Example

Example

```
1. <?php
2. $host = 'localhost:3306';
3. $user = '';
4. $pass = '';
5. $dbname = 'test';
6.
7. $conn = mysqli_connect($host, $user, $pass,$dbname);
8. if(!$conn){
9.   die('Could not connect: '.mysqli_connect_error());
10.      }
11.      echo 'Connected successfully<br/>';
12.
13.      $id=2;
14.      $sql = "delete from emp4 where id=$id";
15.      if(mysqli_query($conn, $sql)){
16.       echo "Record deleted successfully";
17.      }else{
18.      echo "Could not deleted record: ". mysqli_error($conn);
19.      }
20.
21.      mysqli_close($conn);
22.      ?>
```

Output:

```
Connected successfully
Record deleted successfully
```

PHP MySQL Select Query

PHP mysql_query() function is used to execute select query. Since PHP 5.5,**mysql_query()** function is *deprecated*. Now it is recommended to use one of the 2 alternatives.

- **mysqli_query()**
- **PDO::__query()**

There are two other MySQLi functions used in select query.

- **mysqli_num_rows(mysqli_result $result)**: returns number of rows.
- **mysqli_fetch_assoc(mysqli_result $result)**: returns row as an associative array. Each key of the array represents the column name of the table. It return NULL if there are no more rows.

PHP MySQLi Select Query Example

Example

```
1. <?php
2. $host = 'localhost:3306';
3. $user = '';
4. $pass = '';
5. $dbname = 'test';
6. $conn = mysqli_connect($host, $user, $pass,$dbname);
7. if(!$conn){
8.   die('Could not connect: '.mysqli_connect_error());
9. }
10.      echo 'Connected successfully<br/>';
11.
12.      $sql = 'SELECT * FROM emp4';
13.      $retval=mysqli_query($conn, $sql);
14.
15.      if(mysqli_num_rows($retval) > 0){
16.       while($row = mysqli_fetch_assoc($retval)){
17.         echo "EMP ID :{$row['id']}  <br> ".
18.              "EMP NAME : {$row['name']} <br> ".
19.              "EMP SALARY : {$row['salary']} <br> ".
20.              "------------------------------<br>";
21.       } //end of while
22.      }else{
23.      echo "0 results";
24.      }
25.      mysqli_close($conn);
26.      ?>
```

Output:

```
Connected successfully
EMP ID :1
```

```
EMP NAME : ratan
EMP SALARY : 9000
--------------------------------
EMP ID :2
EMP NAME : karan
EMP SALARY : 40000
--------------------------------
EMP ID :3
EMP NAME : jai
EMP SALARY : 90000
--------------------------------
```


PHP MySQL Order By

PHP mysql_query() function is used to execute select query with order by clause. Since PHP 5.5, **mysql_query()** function is *deprecated*. Now it is recommended to use one of the 2 alternatives.

- **mysqli_query()**
- **PDO::__query()**

The order by clause is used to fetch data in ascending order or descending order on the basis of column.

Let's see the query to select data from emp4 table in ascending order on the basis of name column.

```
SELECT * FROM emp4 order by name
```

Let's see the query to select data from emp4 table in descending order on the basis of name column.

```
SELECT * FROM emp4 order by name desc
```

PHP MySQLi Order by Example

Example

```
<?php
$host = 'localhost:3306';
$user = '';
$pass = '';
$dbname = 'test';
$conn = mysqli_connect($host, $user, $pass,$dbname);
if(!$conn){
  die('Could not connect: '.mysqli_connect_error());
}
      echo 'Connected successfully<br/>';

      $sql = 'SELECT * FROM emp4 order by name';
      $retval=mysqli_query($conn, $sql);

      if(mysqli_num_rows($retval) > 0){
       while($row = mysqli_fetch_assoc($retval)){
```

```
17.         echo "EMP ID :{$row['id']}  <br> ".
18.            "EMP NAME : {$row['name']} <br> ".
19.            "EMP SALARY : {$row['salary']} <br> ".
20.            "------------------------------<br>";
21.       } //end of while
22.      }else{
23.      echo "0 results";
24.      }
25.      mysqli_close($conn);
26.      ?>
```

Output:

```
Connected successfully
EMP ID :3
EMP NAME : jai
EMP SALARY : 90000
--------------------------------
EMP ID :2
EMP NAME : karan
EMP SALARY : 40000
--------------------------------
EMP ID :1
EMP NAME : ratan
EMP SALARY : 9000
--------------------------------
```