



# Programming Languages

**In this chapter, you will learn about :**

- Know the concept of program
- Understand the categories of programming language
- Know the object oriented programming

## COMPUTER PROGRAM

A computer program is a set of instructions that directs a computer to perform the tasks necessary to process data into information. A computer programmer might write these instructions using a programming language, which are sets of words, symbols and codes used to create which a computer can process and execute. The steps, that a programmer uses to build a computer program collectively, are called the program development life-cycle.



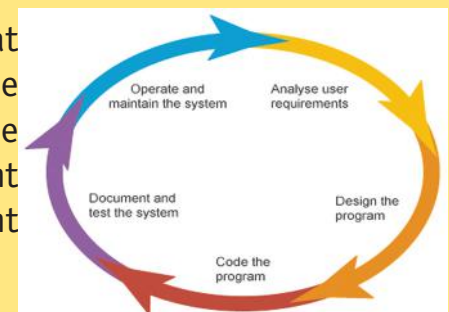
## PROGRAMMING LANGUAGES AND PROGRAM DEVELOPMENT TOOLS

A computer programmer can select from a variety of programming languages or program development tools to create solution to information system requirements. A programming language is a set of words, symbols and codes that enables a programmer to communicate a solution algorithm to the computer. Just as humans understand a variety of spoken languages (English, Hindi, French and so on), computers recognize a variety of programming languages.

A program development tool consists of user friendly software products designed to assist in the creation of information system solutions. These program development tools automatically create the programming language instructions necessary to communicate with the computer. With program development tools, a developer typically does not need to learn a programming language.

## THE PROGRAM DEVELOPMENT LIFE-CYCLE

The program development life-cycle (PDLC) is a set of steps that programmers use to build computer programs. Whereas much of the system development life-cycle guides system analysts through the development of an information system, the program development life-cycle guides computer programmers through the development of a program.



The steps are as follows :

**Step 1:** Developing the program logic to solve the particular problem.



**Step 2:** Writing the program logic in a specific programming language (coding the program).

**Step 3:** Assembling or compiling the program to turn it into machine language.

**Step 4:** Testing and debugging the program.

**Step 5:** Preparing the necessary documentation.

The logic is generally the most difficult part of programming. However, depending on the programming language, writing the statements may also be laborious. One thing is certain, documenting the program is considered the most annoying activity by most programmers.

## CATEGORIES OF PROGRAMMING LANGUAGES

Several hundred programming languages exist each with its own language rules or syntax. Some languages were developed for specific computers, others were developed for specific uses, such as scientific or business applications. As previously noted, the American National Standards Institute (ANSI) has standardized some of these languages. Programs written in an ANSI standard language, thus can run on many different types of computers, as well as on many different operating systems.



**Programming languages are classified into five major categories:** machine languages, assembly languages, third generation languages, fourth generation languages and natural languages. Machine and assembly languages are referred to as low level languages. Third generation, fourth generation and natural languages are called high level languages. A low level language is written to run on one particular computer. A high level language can run on many different types of computers.

## MACHINE LANGUAGE

The only language that the computer directly understands, is machine language, which also is called a first generation language. Machine language instruction uses a series of binary digits (1's and 0's ) that correspond to the ON and OFF electrical states of a computer.



Machine language programs run only on the computer, for which they were developed *i.e.* they are machine dependent. One disadvantage of machine language programs is that they are not portable to other computers. Second, as you might imagine, coding in the 1's and 0's of machine can be tedious and time consuming.

## ASSEMBLY LANGUAGE

Because machine language programs were so difficult to write, a second generation of programming language, called assembly language evolved. With an assembly language, instructions are written using abbreviations and codes. As with machine language assembly language often are difficult to learn and are machine dependent.



Assembly languages do have several advantages over machine languages. Instead of using a series of bits, the programmer uses meaningful abbreviations for program instructions, called symbolic instruction codes or mnemonics. With an assembly language, a programmer writes codes, such as A for addition, C for compare, L for load and M for multiply. Another advantage of assembly languages is that the programmer can refer to storage locations with symbolic addresses. For example – instead of using the actual numeric storage address of a unit price, a programmer can use the symbolic name PRICE.

One disadvantage of an assembly language program is that it must be translated into machine language before the computer can understand it. The program containing the assembly language code is called the source program. The computer cannot understand or execute this source program until it is translated.

A program called an assembler, converts the assembly language source program into machine language that the computer can understand.

One assembly language instruction usually translates into one machine language instruction. In some cases however, the assembly language includes macros, which generate more than one machine language instruction for a single assembly language instruction. Macros save the programmer time during program development, because one assembly language instruction can trigger several action.

### **THIRD GENERATION LANGUAGE**

The disadvantages of low level machine and assembly language led to the development of high level language in the late 1950s and 1960s. Unlike low level languages, high level languages make it easy for programmers to develop and maintain programs. In addition to being easy for programmers to learn and use, high level languages are machine independent, meaning they run on many different types of computers. Three categories of high level languages exist ; third generation languages, fourth generation languages and natural languages.


A third generation language (3GL) instruction is written as a series of English-like words. For example, a programmer writes ADD for addition or PRINT to Print. Many third generation languages also use arithmetic operators such \* as for multiplication and + for addition. These English like words and arithmetic notations simplify the program development process for the programmer.



Third generation languages require that the program instructions tell the computer, what to do accomplish, and how to do it, thus 3 GLs often are called procedural languages. With most 3 GLs, these procedures are developed using both the top-down approach (modules) and structured constructs (sequence, selection and repetition) within each module.

As in an assembly language program the 3 GL code is called the source program, which must be translated to machine language before the computer can understand it. Thus translation process often is very complex, because one 3GL source program instruction translates into many machine





language instruction. For third generation languages, the translation is performed using one of two types of programs, a compiler or an interpreter.

## COMPILER

A compiler converts the entire source program into machine language at one time. The machine language version that results from compiling the 3GL, is called the object code or object program. The object code is stored on disk for execution at a later time.

While it is compiling the source program into object code, the compiler checks the source program's syntax and verifies that the program properly defines the data it will use in calculation or comparisons. The compiler then produces a program listing, which contains the source code and a list of any syntax errors. This listing helps the programmer make necessary changes to the source code and debug the program.

### Know More

With an interpreter, one line of the source program at a time is converted into machine language and then immediately executed by the computer. If the interpreter counters an error while converting a line of code, an error message immediately displays on the screen and the interpretation stops.

## INTERPRETER

While a compiler translates an entire program at once, an interpreter translates one program code statement at a time. An interpreter reads a code statement, converts into one or more machine language instructions, and then executes the machine language instructions, all before moving to the next code statement in the program. Each time you run the source program, it is interpreted into machine language, statement by statement, and then executed. No object program is produced.

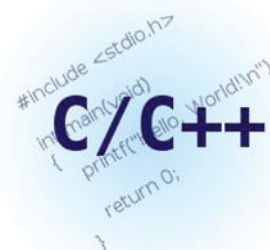
One advantage of an interpreter is that it immediately displays feedback when it finds a syntax error. The programmer can correct any error *i.e.* debug the code, before the interpreter evaluates the next line. The disadvantage is that interpreted programs do not run as fast as compiled programs because the program must be translated to machine language each time when it is executed.

Many programming languages include both an interpreter and a compiler.

The programmer can use the interpreter to debug the program, and then compile the program when it is ready to be delivered to the users.

## FOURTH GENERATION LANGUAGE

Like a 3GL, a fourth generation language (4GL) uses English like statements. A 4GL, however, is a non-procedural language, which means the programmer only specifies what the program should accomplish, without explaining how. Consequently, coding programs in a 4GL requires much less time and effort on the part of the programmer. In fact, 4GLs are so easy to use that users with very little programming background can develop programs using a fourth generation language.



```
#include <stdio.h>
int main(void)
{
    printf("Hello World!\n");
    return 0;
}
```

## Know More

SQL is a fourth generation language that can be used to query database tables. This query produces an alphabetical list of those customers who receive a discount i.e. their discount code is not equal to zero.

Many 4GLs work in combination with a database and its project dictionary. These powerful languages allow database administrators to define the database and its structure, help programmers maintain the data in the database and allow users to query the database. SQL is a query language enabling users and programmers to retrieve data from database tables. One query, for example, might request a list of all customers receiving a discount.

Last_Name	First_Name	Net_Amount_Due
Vipin	Kumar	1000
Aish	Arya	2000
Anju	Tomar	2500

## NATURAL LANGUAGES

Whereas a fourth generation language program must follow a specific set of rules and syntax, a natural language program does not. A natural language, sometimes called a fifth generation language is a type of query language that allows the user to enter requests that resemble human speech.

For example – An SQL query to obtain a list of GPAs that exceed 3.5 weight be written as select Last name, First-Name from student where GPA>3.5. A natural language version of that same query might be written or spoken as Tell me the name of students with GPA over 3.5.



Natural language often are associated with expert system and artificial intelligence. These systems are popular in the medical fields, but are not widely used in business applications.

## OBJECT ORIENTED PROGRAMMING

Abbreviated OOP programming that supports object technology. It is an evolutionary form of modular programming with more formal rules that allows pieces of software to be reused and interchanged between programs. Major concepts are—

- (i) Encapsulation
- (ii) Inheritance
- (iii) Polymorphism

Encapsulation is the creation of self sufficient modules that contain the data and the processing (data structure and functions that manipulates the data). These user defined or abstract data types are called classes. One instance of a class is called an object. For example – In a payroll system, a class could be defined as manager, and Pat and Jau, the actual objects, are instances of that class. Classes are created in hierarchies, and inheritance allows the knowledge in one class to be passed down the hierarchy. That means less programming is required when adding functions to complex systems. If a step is added at the bottom of a hierarchy, then only the



processing and data associated with that unique step needs to be added. Everything else about that step is inherited.

Object oriented programming allows procedures about objects to be created whose exact type is not known until run time. For example, a screen cursor may change its shape from an arrow to a line depending on the program mode. The routine to move the cursor on screen in response to mouse movement would be written for cursor and polymorphism would allow that cursor to be whatever shape is required at runtime. It would also allow a new shape to be easily integrated into the program.

The simulation language was the original object oriented language. It was used to model the behaviour of complex systems. Xerox's small talk was the first object oriented programming language and was used to create the graphical user interface, whose derivations are so popular today. C++ has become the major commercial OOP language, because it combines traditional C programming with object oriented capabilities.

### Points to Remember

- A programming language is a set of words, symbols and codes used to create instructions a computer can process and execute.
- The program development life-cycle is a set of steps that programmers use to build computer programs.
- Programming languages are classified into five major categories—machine language, assembly language, third generation language, fourth generation language and natural languages. The first two are referred to as low level languages. The last three are called high level languages.
- Object oriented programming supports objects technology more formal rules that allow pieces of software to be reused and interchanged between programs.



#### A. Tick (✓) the correct option :

- Machine language and assembly language are referred to as \_\_\_\_\_.
  - (a) Low level language
  - (b) Middle level language
  - (c) High level language
- Computer understand only \_\_\_\_\_ language.
  - (a) Machine  (b) Assembly  (c) Both (i) & (ii)
- \_\_\_\_\_ translate on program code statement at a time.
  - (a) Machine language  (b) Interpreter  (c) Compiler
- Abstract data types are called \_\_\_\_\_.
  - (i) Class  (ii) Objects  (iii) Inheritance



5. OOP stands for\_\_\_\_\_ .

(a) Object oriented programming



(b) Object orient programm



(c) Object orientation program



6. SQL stands for\_\_\_\_\_ .

(a) System query language

(b) Support query language

(c) Structured query language

### B. Fill in the blanks :

1. Machine language and assembly language are referred to as \_\_\_\_\_ languages.
2. The program containing the assembly language code is called the \_\_\_\_\_.
3. A \_\_\_\_\_ instruction is written as a series of words.
4. Many programming languages include both an \_\_\_\_\_ and a compiler.
5. OOP stands for \_\_\_\_\_.
6. PDLC stands for \_\_\_\_\_.

### C. Write (T) for true and (F) for false :

1. Computer program is a set of instructions that directs to perform tasks.
2. A programming language is a set of words only.
3. PDLC stands for programmer development life-cycle.
4. The logic is generally very easy part of programming.
5. ANSI stands for American National Standards Institute.

### D. Answer the following questions :

1. What is a computer program?
2. What is a program development tool? Explain.
3. What are the steps a programmer use in program development life-cycle (PDLC)?
4. What is the difference between a low level language and a high level language?
5. Why is an assembly language better than a machine language?
6. What were the main reasons behind developing third generation language (3GL)?



### ACTIVITY

- In the school computer laboratory :
  1. Find out the computer languages installed in the computer.
  2. Categories them into the five generations.
  3. Write two important features of each.