PROGRAMMING LANGUAGES & PROGRAMMING PARADIGMS

Overview of Programming Languages

A programming language is a formal language used to give instructions to a computer. It acts as an interface between humans and machines and enables development of business, web, mobile, and enterprise systems

Programming languages are tools to solve business problems, not just technical exercises.

Example
POS systems, banking apps, payroll software

Purpose of Programming Languages in Business

- Automate business processes
- Store and process organizational data
- Enable digital transformation
- Improve efficiency and accuracy

Classification of Programming Languages

Low-level languages

Low-level languages are programming languages that provide little or no abstraction from computer hardware. They are closely tied to the machine and are mainly used for system-level programming such as operating systems and embedded systems.

High-level languages

High-level languages are user-friendly programming languages designed to be easy to read, write, and maintain. They are machine-independent and widely used in developing business, web, and enterprise applications.

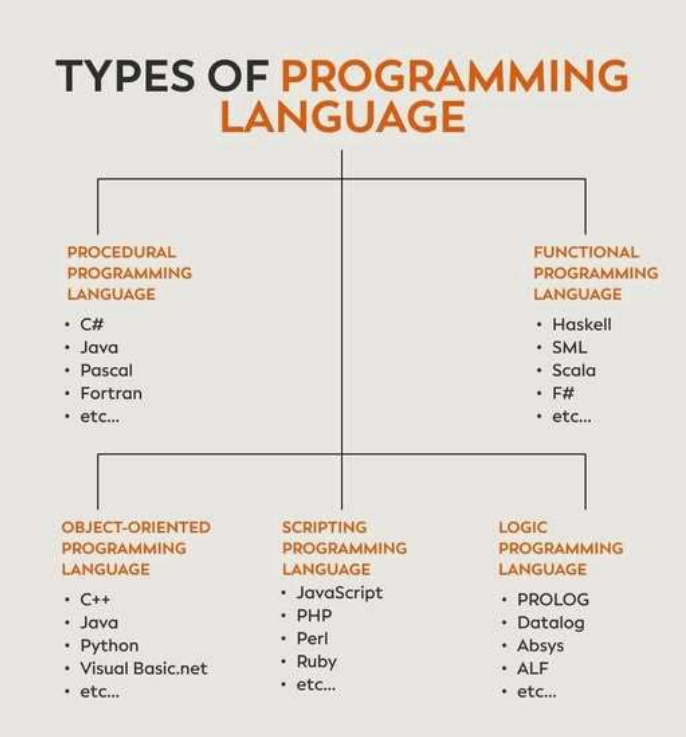
Scripting languages

Scripting languages are high-level languages used mainly for automation, rapid application development, and web programming. They are usually interpreted and allow developers to write and test code quickly.

Domain-specific languages

Domain-specific languages are specialized programming languages designed for a specific application area or problem domain. They simplify tasks within that domain but are not intended for general-purpose programming.

Types of Programming Languages



Programming Paradigms

- Paradigm can also be termed as method to solve some problem or do some task.
- Programming paradigm is an approach to solve problem using some

programming language or also we can say it is a method to solve a problem using tools and techniques that are available to us following some approach.
There are lots for programming language that are known but all of them need to follow some strategy when they are implemented and this methodology/strategy is paradigms. Apart from varieties of programming language there are lots of paradigms to fulfil each and every demand. The programming paradigm is divided into two broad categories.

- Imperative programming paradigm
- Declarative programming paradigm

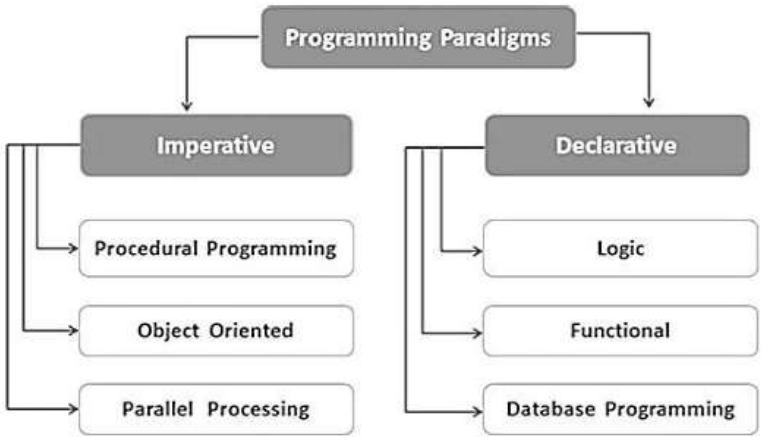
Imperative programming paradigm

It is one of the oldest programming paradigm. It features close relation to machine architecture. It is based on Von Neumann architecture. It works by changing the program state through assignment statements. It performs step by step task by changing state. The main focus is on how to achieve the goal. The paradigm consist of several statements and after execution of all the result is stored.

Advantage

1. Very simple to implement
2. It contains loops, variables etc.

Disadvantage
1. Complex problem cannot be solved
2. Less efficient and less productive
3. Parallel programming is not possible
Examples of Imperative programming paradigm: C, FORTAN, Basic



Imperative programming is divided into three broad categories: Procedural, OOP and parallel processing. These paradigms are as follows:

Procedural programming paradigm
This programming has a single program that is divided into small piece called procedure (also known as functions, routines, subroutines). These procedures are combined into one single location with the help of return statements.
From the main controlling procedure, a procedure call is used to invoke the required procedure. After the sequence is processed, the flow of control continues from where the call was made. The main program coordinates calls to procedures and hands over appropriate data as parameters. The data is processed by the procedures and once the program has finished, the resulting data is displayed.
Example: C, Pascal.

Object Oriented Programming
It is a type of programming in which programmers define not only the data type of a data structure, but also the types of operations (functions) that can be applied to the data structure.

In this way, the data structure becomes an object that includes both data and functions. In addition, programmers can create relationships between one object and another. For example, objects can inherit characteristics from other objects.

One of the principal advantages of object-oriented programming techniques over procedural programming techniques is that they enable programmers to create modules that do not need to be changed when a new type of object is added. A programmer can simply create a new object that inherits many of its features from existing objects. This makes object-oriented programs easier to modify. The basic concepts of OOP are as follows:

1. Objects
2. Classes
3. Data abstraction and encapsulation
4. Inheritance
5. Polymorphism
6. Dynamic binding
7. Message passing.

Objects

Objects are the basic run-time entities in an object oriented system. They may represent a person, place or a bank a/c or a table of data that the program has to handle.
When a program is executed the objects interact by sending messages to one another.
They can interact without knowing the details of each other's data or code. Thus an object is considered to be a partitioned area of computer memory that stores data and set of functions that can access the data.

Classes

The entire set of data and code of an object can be made a user-defined data type with the help of a class. Objects are variables of the type class. Each object is associated with the data of type class with which they are created. A class is thus a collection of objects of similar type.

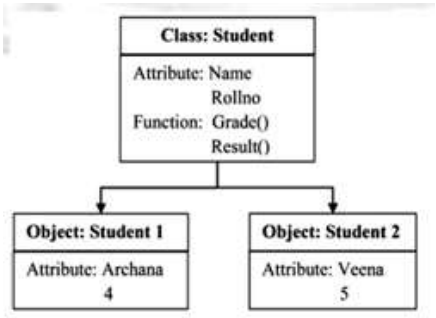


Fig. 1.2 Class and Objects

Data abstraction and encapsulation

Encapsulation:

The wrapping up of data and functions in to a single unit (that unit is called a class) is known as encapsulation. The data is not accessible to the outside world (other functions which are not the members of that class) and only those functions which are wrapped in the class can access it. This insulation of data from the direct access by the program is called data hiding or information hiding.

Abstraction:

Abstraction refers to the act of representing essential features without including the background details or explanations. Classes use the concept of abstraction and are defined as a list of abstract attributes such as size, weight and cost and functions to operate on these attributes. The attributes

are sometimes called data members because they hold information. The functions that operate on these data are called member functions.

Inheritance:
Inheritance is the process by which objects of one class acquire the properties of objects of another class. It supports the concept of hierarchical classification. In OOP, the concept of inheritance provides the idea of reusability. This means that we can include additional features to an existing class without modifying it. It is important because it supports the concept of classification. C++ supports different types of inheritance such as single inheritance, multiple inheritance, multilevel inheritance and hierarchical inheritance.

To understand the concept of inheritance, let us consider an example of vehicles as shown in Fig.1.3. Here the class car is a subclass of automatic vehicle, which is again a subclass of the class vehicle. This implies that the class car has all the characteristics of automatic vehicles which in turn has all the properties of vehicles. However, car has some unique features which differentiate it from other subclasses. For example, it has four wheels and five gears, while scooter has two wheels and four gears.

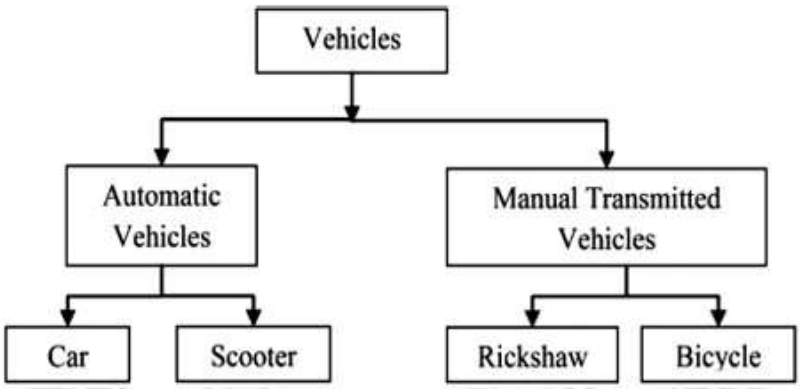


Fig. 1.3 Inheritance

Polymorphism:
Polymorphism, a Greek term means the ability to take more than one form. An operation may exhibit different behaviour in different instances. The behaviour depends up on the types of data used in the operation. The concepts of polymorphism are Operator overloading and Function overloading. For two numbers, the operator + will give the sum. If the operands are strings, then the operation would produce a third string by concatenation. Thus the process of making an operator to exhibit different behaviours in different instances is known as operator overloading. Similarly, we can use a single function to perform different tasks which is known as function overloading. A single function can be used to handle different number and types of arguments.

Binding:
Linking of procedure call to the corresponding code in response to the call.

Dynamic Binding:
Dynamic binding means that the code associated with a given procedure call is not known until the time of the call at run-time. It is associated with polymorphism concept.

Message Passing:
OOP consists of a set of objects that communicate with each other by sending and receiving information. A message for an object is a request for execution of a procedure (function) and therefore will invoke a function in the receiving object that generates the desired result.

Benefits of OOP
- Through inheritance we can eliminate redundant code and extend the use of existing classes.
- We can build secure programs by the principle of data hiding.
- It is easy to partition the work in a project based on objects.
- Object oriented systems can be easily upgraded from small to large systems.
- Communication with external systems are much simpler by means of message passing techniques.
- Software complexity can be easily managed.

Applications of OOP
OOP can be applied in the following areas:
- Real time systems
- Simulation and modeling
- Object oriented databases
- Hypertext, hypermedia and expertext
- Artificial Intelligence and expert systems.
- Neural networks and parallel programming
- Office automation systems
- CIM / CAM / CAD systems

Example: Simula, JAVA, Python, VB.NET, Ruby.

Parallel processing approach
Parallel processing is the processing of program instructions by dividing them among multiple processors. A parallel processing system possess many numbers of processor with the objective of running a program in less time by dividing them. This approach seems to be like divide and conquer.
Examples are NESL (one of the oldest one) and C/C++ also supports because of some library function.

Declarative programming paradigm
- It is divided as Logic, Functional, and Database. In computer science the declarative programming is a style of building programs that expresses logic of computation without talking about its control flow.

- It often considers programs as theories of some logic. It may simplify writing parallel programs. The focus is on what needs to be done rather how it should be done basically emphasize on what code is actually doing.
- It just declare the result we want rather how it has be produced. This is the only difference between imperative (how to do) and declarative (what to do) programming paradigms.

Logic programming paradigms

- It can be termed as abstract model of computation. It would solve logical problems like puzzles, series etc.
- In logic programming we have a knowledge base which we know before and along with the question and knowledge base which is given to machine, it produces result. In normal programming languages, such concept of knowledge base is not available but while using the concept of artificial intelligence, machine learning.
- In logical programming the main emphasize is on knowledge base and the problem. The execution of the program is very much like proof of mathematical statement, e.g., Prolog Functional programming paradigms
- The functional programming paradigms has its roots in mathematics and it is language independent. The key principal of this paradigms is the execution of series of mathematical functions.
- The central model for the abstraction is the function which are meant for some specific computation and not the data structure. Data are loosely coupled to functions.
- The function hide their implementation. Function can be replaced with their values without changing the meaning of the program. Some of the languages like Perl, java script mostly uses this paradigm.

Database/Data driven programming approach

- This programming methodology is based on data and its movement. Program statements are defined by data rather than hard-coding a series of steps.
- A database program is the heart of a business information system and provides file creation, data entry, update, query and reporting functions. There are several programming languages that are developed mostly for database application.
- For example SQL. It is applied to streams of structured data, for filtering, transforming, aggregating (such as computing statistics), or calling other programs. So it has its own wide application.

Characteristics of a Good Programming Language

- A programming language must be simple, easy to learn and use, have good readability and human recognizable.
- Abstraction is a must-have Characteristics for a programming language in which ability to define the complex structure and then its degree of usability comes.
- A portable programming language is always preferred.
- Programming language's efficiency must be high so that it can be easily converted into a machine code and executed consumes little space in memory.
- A programming language should be well structured and documented so that it is suitable for application development.

- Necessary tools for development, debugging, testing, and maintenance of a program must be provided by a programming language.
- A programming language should provide single environment known as Integrated Development Environment (IDE).
- A programming language must be consistent in terms of syntax and semantics.