**Procedure-Oriented Programming**

1. Procedure-oriented programming basically consists of writing a list of instructions for the computer to follow and organizing these instructions into groups known as functions.
2. The problem is viewed as a sequence of things to be done.
3. The primary focus is on functions.
4. Traditional procedural language, such as assembly language or a high-level like COBOL, FORTRAN, C, etc.
5. To revise an external data structure, we also need to revise all functions that access the data.
6. This approach does not model real world problems. This is because functions are action-oriented and do not really correspond to the elements of the problem.
7. In multi-function program, many important data items are placed as global so it may access by all function.
8. Drawback:- It does not model real world problems very well. Due to function are action-oriented and do not really corresponds to elements of the problem.

**Object-Oriented Programming**

1. Object-Oriented Programming (OOP) is the term used to describe a programming approach based on classes and objects.
2. The object-oriented pogramming allows us to organise software as collection of objects that consist of both data and behaviour.
3. OOP treat data as a critical element in the program development and does not allow it to flow freely around the system.
4. It ties data more closely to the functions that operate on it, and protects it from accidental modification from outside functions.
5. The data of an object can be accessed only by the functions associated with that object.
6. Functions of one object can access the functions of another objects.
7. Example: Class bank\_account, that have two object bank b1 and bank b2. In both have some functions are same and some not same.

Abstraction

1. Abstraction is the method of hiding the unwanted information.
2. Abstraction is the process of gaining the information.
3. In abstraction, problems are solved at the design or interface level.
4. We can implement abstraction using abstract class and interfaces.
5. In abstraction, implementation complexities are hidden using abstract classes and interfaces.
6. The objects that help to perform abstraction are encapsulated

ENCAPSULATION

1. encapsulation is the process to contain the information.
2. in encapsulation, problems are solved at the implementation level
3. encapsulation is a method to hide the data in a single entity or unit along with a method to protect information from outside.
4. encapsulation can be implemented using by access modifier i.e private,protected and public.
5. in encapsulation, the data is hidden using methods of getters and setters.
6. the objects that result in encapsulation need not be abstracted.

Inheritance

1. Inheritance is the process, by which class can acquire the properties and methods of another class.
2. The mechanism of deriving a new class from an old class is called inheritance.
3. The new class is called derived class and old class is called base class.
4. The derived class may have all the features of the base class and the programmer can add new features to the derived class

Types of Inheritance

• Single Inheritance

• Multilevel Inheritance

• Multiple Inheritance

• Hierarchical Inheritance

• Hybrid Inheritance

• Single Inheritance

If a class is derived from a single class then it is called single inheritance.

Multilevel Inheritance

If A class is derived from a class which is derived from another class then it is called multilevel inheritance

Multiple Inheritance

• If a class is derived from more than one class then it is called multiple inheritance.

Hierarchical Inheritance

• If one or more classes are derived from one class then it is called hierarchical inheritance.

Hybrid Inheritance

• It is a combination of any above inheritance types. That is either

multiple or multilevel or hierarchical or any other combination

**Polymorphism**

1. Polymorphism means the ability to take more than one form.
2. It allows a single name to be used for more than one related purpose.
3. It means ability of operators and functions to act differently in different situations.
4. Different types of polymorphism are Compile time polymorphism and run time polymorphism
5. Compile time polymorphism are also of two types which are Function overloading and operator overloading
6. Run time polymorphism can be virtual function

**Dynamic binding**

1. Dynamic binding, also known as late binding or runtime binding
2. it refers to the process of determining which method or function to call at runtime.
3. At run-time, the code matching the object under current reference will be called.
4. This decision is based on the actual type of an object rather than the declared type of a reference or variable
5. Dynamic binding allows for polymorphism, a fundamental principle of OOP, where objects of different classes can respond to the same method name in a way that is appropriate for their specific class.

**Message Passing**

1. An oop consists of a set of objects that communicate with each other
2. Objects communicate with one another by sending and receiving information
3. A message for an object is a request for execution of a procedure.
4. The receiving object will call a function and generates results
5. Message passing involves specifying:

o The name of the Object.

o The name of the Function.

o The information to be send