Source: Introduction to Programming & Computer Science

What is programming?

Ans: Programming means :

- Attempting to get a computer to complete a specific task without making mistakes.

Computer only understands Machine Code

- A series of 1's and 0's fed and interpreted by the computer
- Only way a computer can read english instructions

All computer program consistes of two elements : code & data

A program can be conceptually organized around its data.

What is OOP ?

Primitive data types store singles, simple values.

byte, boolean, int, float, boolean, double, char

OOP : Object Oriented Programming.

All relevant data and tasks are grouped together in entities known as Objects.

Objects has three different properties: *State, identity & behaviour*.

- ***State*** *=> value of object / data type*
- ***identity*** *=> id or address of the object in memory. It distinguishes one object from another.*
- ***behaviour*** *=> effect of the internal state/datas by using functions/ methods/ arithmetical operations.*

Objects has a physical reality, that is, it occupies space in memory.

Example, students are the objects of a class 'X'.

Student function/ behaviour: classwork, homework, tasks, discipline, speaking and writing.

Student state: school student, gender, age.

Student identity: Roll number / ID card

- **Objects** are instances (or member) of Class.
- **Classes** are templates of objects. It defines properties & behaviour of objects. It doesn't have any size in the memory, rather the variables and methods inside the class takes memory space.
- **OOP** emphasizes on *protection of data.*
- OOP treats *data* as a *critical element* in the program development and *restricts* its flow freely around the system.
- **OOP** helps programmers to create *complex programs* by grouping together related data and functions.
- OOP organizes a program around its objects and a set of well defined interfaces to that data.
- OOP is task based (as it considers operations); as well as data-based (as these operations are grouped with the relevant data).
- OOP can be characteried as *data controlling access to code.*
- OOP is used for solving real world problems on computers because real world is made of objects.

4 main principles of OOP :

- **Encapsulation**
- **Abstraction**
- **Inheritance**
- **Polymorphism**

Encapsulation : The mechanism that binds code and data together and keeps them secure from outside world is known as Encapsulation.

It is the idea of *hiding data* within a class, preventing anything outside the class from directly interacting / accesing with it.

Since, the purpose of the class is to encapsulate complexity, there are mechanisms for hiding the complexity of the implementation inside the class.

Such mechanisms are by using access specifiers – public, private, protected. Also, we can use getters and setters.

Each method or varibale in a class can be public or private.

- The public members can be accessed outside the class and package too. So, this means the public interface must be carefully designed not to expose too much of inner working of the class.
- The private members can only be accessed by the methods or code in that class only. So, the other methods not the member of that class cannot access it.

Members of other classes can interact with the attributes of another object through its methods that are made public.

Why do we need encapsulation?

- It's generally best to not allow external classes to directly edit an object's attributes.
- This is very imp when working on large and complex programs
- Each piece should not have access to, or rely on the inner workings of other sections of code

Information Hiding, or keeping the data of one class hidden from external classes, helps you keep control of your program and prevents it from becoming too complicated.

Thus, we conclude that Encapsulation plays a *vital* role in OOP as it :

- Keeps the programmer in control of access to data
- Prevents the program from ending up in any strange or unwanted states.

Abstraction : It allows users to show only *essential details* and keep everything else hidden.

- The interface refers to the way sections of code can communicate with one another. This is typically done through methods that each class is able to access.

- The implementation of these methods, or how these methods are coded, should be hidden.

Abstraction – Interface and Implementation

If classes are entangled, then one can change; creates a ripple effect that cause many more change.

Creating an interface through which class can interact ensures that each piece can be individually developed.

Abstraction prevents program from becoming entangled and complex.

Example: Consider a car, people needs to know the basic things to drive a car, like accerlerator, break, steering, seat-belt; but they don't need to know how the car works internally like How the break system works ?, or internal components of engine, or the electronic systems that makes the car work. We don't need to know these stuffs to drive a car. So, it basically states that we need to know just the essential things to drive a car and not the internal working system of the car.

This is what abstraction is, we only need to know essential details and keep other things hidden.

Inheritance : It is a property by which we can inherit the properties of one class to another class.

- The class from which the properties are inherited are called as *Super class* or base-class.
- The class which inherits the property of super-class are called as *Sub-class* or derived class.
- A sub-class can be said as *specialized version* of a super-class. Because it inherits all of the members defined by the super-class and adds its own essence with additional features and unique elememts.
- To inherit a class we use *extends* keyword while defining or declaring a sub-class.

```
class Base{
    // methods and statements
}
class Derived extends Base{
```

```
        // methods and statements
    }
```

Types of inheritance:

- *Singleton or Single*
- *Multi-level*
- *Hierachical*
- *Multiple*
- *Hybrid*