Object Oriented Programming (OOP)

Object-Oriented Programming (OOP) is a programming paradigm that is based on the concept of objects, which have both data (attributes) and behavior (methods). OOP focuses on creating reusable code by organizing code into objects and classes, which can be easily maintained and extended.
Video Link

OOP is helpful for a number of reasons:

- Reusability: OOP promotes code reusability, since objects and classes can be easily reused in different parts of a program or in different programs altogether. This can help to save time and effort in programming.
- Modularity: OOP promotes modularity, which means that code is organized into small, self-contained units (objects and classes). This makes it easier to understand, modify, and maintain the code.

Overall, OOP is helpful because it promotes code reusability, modularity, encapsulation, inheritance, and polymorphism. These features can help to make code more organized, understandable, and maintainable, which can save time and effort in programming.

Encapsulation: Encapsulation is one of the fundamental concepts of Object-Oriented Programming (OOP). It is the practice of bundling data (attributes) and behavior (methods) together in a single unit (object) and hiding the implementation details from the outside world. In other words, it is the process of encapsulating data and methods inside a class and protecting them from unauthorized access and modification.
Encapsulation helps to maintain data integrity by preventing unauthorized access to the data. This is achieved by making the data private, so that it can only be accessed through methods that are defined in the same class. The methods that are used to access and modify the data are known as getters and setters.
In summary, encapsulation is the process of bundling data and behavior together in a single unit (object) and hiding the implementation details from the outside world. Encapsulation helps to maintain data integrity and makes it easier to modify the internal workings of a class without affecting other parts of the program.

Example: Suppose you are designing a program for a bank that needs to store customer data, including their account number, balance, and other sensitive information. In this scenario, you want to ensure that the data is protected from unauthorized access and modification.
You can use encapsulation to achieve this goal. You can create a Customer class that encapsulates the sensitive data and provides methods to access and modify the data.

Abstraction: Hiding unnecessary details and showing only the necessary information Abstraction is one of the fundamental concepts of Object-Oriented Programming (OOP) that allows developers to focus on the essential features of an object while hiding its unnecessary details.
In other words, abstraction is a way of managing complexity by breaking down a complex system into smaller, more manageable parts. It allows developers to create abstract classes and interfaces that provide a blueprint for other classes to follow, without exposing the internal workings of the class.The main purpose of abstraction is to reduce complexity and increase efficiency by hiding unnecessary details and providing a simpler interface for users to work with. It also helps in achieving better modularity, reusability, and maintainability of code.
Here are some key features of abstraction:

- It is the process of hiding implementation details while showing only the necessary information to the user.

- It is achieved through abstract classes and interfaces that provide a blueprint for other classes to follow.
- It helps in managing complexity by breaking down a complex system into smaller, more manageable parts.
- It provides a simpler interface for users to work with, without exposing the internal workings of the class.
- It helps in achieving better modularity, reusability, and maintainability of code.

Inheritance: Inheritance enables a new class to inherit properties and methods from an existing class(Parent Class), and to extend or modify them to create new functionality.

Inheritance promotes the reuse of code and helps in organizing and structuring large software applications. By using inheritance, developers can define a base or parent class with common attributes and methods that can be shared among multiple child or derived classes. The child classes can then add their own specific attributes and methods to the base class, while still retaining the properties and behavior of the parent class.

There are two types of inheritance in OOP:

- Single Inheritance: In this type of inheritance, a child class inherits from only one parent class.
- Multiple Inheritance: In this type of inheritance, a child class inherits from multiple parent classes.

Polymorphism: 'Poly' means many 'Morphism' means ways to represent Polymorphism is a concept in Object-Oriented Programming (OOP) that allows objects of different classes to be treated as if they are of the same class. In other words, polymorphism enables objects to take on multiple forms or behaviors depending on the context in which they are used.

Polymorphism in OOP can be achieved in two ways:

- Overloading: Overloading occurs when two or more methods in a class have the same name but different parameters. This allows methods to be called with different arguments and perform different actions based on the type or number of parameters passed to them.
- Overriding: Overriding occurs when a method in a subclass has the same name, return type, and parameters as a method in the parent class. The method in the subclass overrides the method in the parent class and provides its own implementation.