Object-oriented programming (OOP) :-
Object-oriented programming (OOP) is a computer programming model
that organizes software design around data, or objects, rather than functions and logic.
An object can be defined as a data field that has unique attributes and behavior.

The technical definition of object-oriented programming is as follows:

The object-oriented programming is basically a computer programming design philosophy or methodology
that organizes/ models software design around data, or objects rather than functions and logic.

An object is referred to as a data field that has unique attributes and behavior. Everything in OOP is
grouped as self-sustainable objects.

It is the most popular programming model among developers. It is well suited for programs that are large, complex,
and actively updated or maintained. It simplifies software development and maintenance by providing
major concepts such as abstraction, inheritance, polymorphism, and encapsulation.
These core concepts support OOP.

A real-world example of OOP is the automobile. It more completely illustrates the power of object-oriented design.

Points to Remember
Everything is an object
Developer manipulates objects that uses message passing.
Every object is an instance of a class.
The class contains the attribute and behavior associated with an object.
Pillars of OOPs
The major concepts that we have discussed above are known as pillars of OOPs. There are four pillars on which OOP rests.

Abstraction
Encapsulation
Inheritance
Polymorphism
Object
Class

Abstraction
The concept allows us to hide the implementation from the user but shows only essential information to the user.
Using the concept developer can easily make changes and added over time.
There are the following advantages of abstraction:
It reduces complexity.
It avoids delicacy.
Eases the burden of maintenance
Increase security and confidentially.

Encapsulation

Encapsulation is a mechanism that allows us to bind data and functions of a class into an entity. It protects data and functions from outside interference and misuse. Therefore, it also provides security. A class is the best example of encapsulation.

Inheritance
The concept allows us to inherit or acquire the properties of an existing class (parent class) into a newly created class (child class). It is known as inheritance. It provides code reusability.

Polymorphism
The word polymorphism is derived from the two words i.e. ploy and morphs. Poly means many and morphs means forms. It allows us to create methods with the same name but different method signatures. It allows the developer to create clean, sensible, readable, and resilient code.

Object
An object is a real-world entity that has attributes, behavior, and properties. It is referred to as an instance of the class. It contains member functions, variables that we have defined in the class. It occupies space in the memory. Different objects have different states or attributes, and behaviors.

Class
A class is a blueprint or template of an object. It is a user-defined data type. Inside a class, we define variables, constants, member functions, and other functionality. it binds data and functions together in a single unit. It does not consume memory at run time. Note that classes are not considered as a data structure. It is a logical entity. It is the best example of data binding. Note that a class can exist without an object but vice-versa is not possible.

The following figure best illustrates the class and object in OOP.

Why should we use OOP?
Object-oriented programming is an evolutionary development in software engineering. Using OOP in software development is a good habit because it accomplishes the three major software engineering goals, as we have shown in the following figure.

Where it is used?
OOP is often the best use when we are dealing with manufacturing and designing applications. It provides modularity in programming. It allows us to break down the software into chunks of small problems that we then can solve one object at a time.

It should be used where the reusability of code and maintenance is a major concern. Because it makes development easy and we can easily append code without affecting other code blocks. It should be used where complex programming is a challenge.

Benefits of OOP
Modular, scalable, extensible, reusable, and maintainable.
It models the complex problem in a simple structure.
Object can be used across the program.
Code can be reused.
We can easily modify, append code without affecting the other code blocs.
Provides security through encapsulation and data hiding features.
Beneficial to collaborative development in which a large project is divided into groups.
Debugging is easy.
Limitations of OOP

Requires
intensive testing processes.
Solving problems takes more time as compared to Procedure
Oriented Programming.
The size of the programs created using this approach may become larger
than the programs written using the procedure-oriented programming approach.
Software
developed using this approach requires a substantial amount of pre-work and planning.
OOP code
is difficult to understand if you do not have the corresponding class documentation.
In
certain scenarios, these programs can consume a large amount of memory.
Not suitable for small
problems.
Takes more time to solve problems.
List of Object-Oriented Programming
Languages
There are various object-oriented programming languages are present. But we have
enlisted some popular and widely used OOP languages.

According to the TIOBE index, the top
twenty OOP languages are Java, C++, C#, Python, R, PHP, Visual Basic.NET, JavaScript, Ruby, Perl, Object Pascal, Objective-C, Dart, Swift, Scala, Kotlin, Common Lisp, MATLAB, and Smalltalk.