**Encapsulation**

Encapsulation is about bundling the data (variables) and the methods (functions) that operate on the data into a single unit, which is typically a class in OOP. It also involves restricting access to the inner workings of that class, which is known as information hiding. Here's how encapsulation works:

Information Hiding: This is achieved by making the class attributes (data) private or protected, meaning they cannot be accessed directly from outside the class. Instead, public methods are provided to interact with these private attributes, known as getters and setters. This control over data helps in preventing unauthorized access and modification.

Benefits: Encapsulation leads to increased security and prevents the accidental modification of data. It also makes the code more maintainable and flexible, as changes inside a class do not affect the code that uses the class.

**Abstraction**

Abstraction, on the other hand, is about hiding the complex reality of the implementation details and showing only the necessary parts to the user. It’s like creating a simplified model of a more complex entity. In OOP, this is achieved through the use of abstract classes and interfaces. Here’s how abstraction works:

Simplifying Complexity: Abstraction allows a programmer to hide all but the relevant data about an object in order to reduce complexity and increase efficiency. For instance, when you use a method of a class, you don't need to know how it works internally; you just need to know what it does (its interface).

Abstract Classes and Interfaces: These are used to define templates for other classes. They outline a set of methods that the deriving classes must implement, but they don't define how these methods are implemented.

Benefits: Abstraction reduces complexity and allows the programmer to focus on interactions at a higher level. It also helps in reducing code duplication and increases the reusability of code.

**Implementation of Encapsulation and Abstraction in Python**

Encapsulation: In Python, there are no explicit ways to define private variables or methods (like in Java or C++). However, a convention is followed using underscores. A single underscore (e.g., \_variable) indicates a variable should be treated as "protected", and a double underscore (\_\_variable) makes the variable name mangling to create a private-like behavior.

Abstraction: Python supports the creation of abstract classes through the ABC (Abstract Base Class) module. An abstract class can define an interface, and other classes derive from this abstract class, implementing the abstract methods.