## **Encapsulation in python involves bundling data(attributes) and**

## **Methods that operate on the data within a single unit**,

# Typically a class.

# It aims to restrict direct access to some of an object’s

# Components, preventing unintended modification and promoting data integrity .

# While python does not enforce strict access modifiers like public,

# Private, or protected as in some other language, encapsulation is

|  |
| --- |
| Types of Access Modifier |

|  |
| --- |
| Public |

|  |
| --- |
| Protected |

|  |
| --- |
| Private |

# Achieved through conventions and name mangling.

# 1.Public Members

Example: This example shows how a public attribute (name) and a public method

(display\_name) can be accessed from outside the class using an object.

# 2. Protected members

Protected members are variables or methods that are intended to be accessed only within the class and its subclasses. They are not strictly private but should be

Treated as internal.

In python, protected members are defined with a single underscore prefix (e.g..,self.\_name).

Example: This example shows how a protected attribute can be accessed within a subclass, demonstrating that protected members are meant for use within the class and its subclass.

class Employee:

def \_\_init\_\_(self, name, age):

self.name = name

self.\_age = age

class SubEmployee(Employee):

def show\_age(self):

print("Age:",self.\_age)

emp = SubEmployee("lekhana"):

print(emp.name)

emp.show\_age()

# 3.Protected

Private members are variables or methods that cannot be directly from

Outside the class. They are used to restrict access and protect internal data.

In python,private members are defined with a double underscore prefix.

Python applies by internally renaming them to prevent direct access.

Example: This example shows how a private attribute(\_\_salary) is acceced within the class using a public menthod, demonstrating that private members cannot be accessed directly from outside the class.