**Access Modifiers:**

Access modifiers are used by object oriented programming languages like C++ java,python etc. to restrict the access of the class member variable and methods from outside the class.Access modifiers like

## Public Access Modifier

## Private Access Modifier

## Protected Access Modifier

## **Public Access Modifier:**

By default the member variables and methods are public which means they can be accessed from anywhere outside or inside the class. No public keyword is required to make the class or methods and properties public.

**Example:**

class Student:

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

self.name = name

self.age = age

def display(self):

print("Name:", self.name)

print("Age:", self.age)

#Create an instance of the Student class

s = Student("John", 20)

# Call the display method to print the student's information

s.display()

## **Private Access Modifier:**

## Class properties and methods with private access modifier can only be accessed within the class where they are defined and cannot be accessed outside the class.

private properties and methods are declared by adding a prefix with two underscores(‘\_\_’) before their declaration.

##With out Private Access##

class BankAccount:

def \_\_init\_\_(self, account\_number, balance):

self.\_account\_number = account\_number # Single underscore for "protected" attribute

self.\_balance = balance

def \_display\_balance(self): # Single underscore for "protected" method

print("Balance:", self.\_balance)

b = BankAccount(1234567890, 5000)

b.\_display\_balance() # Calling a method with a single underscore

**## With Private Method#**

class BankAccount:

def \_\_init\_\_(self, account\_number, balance):

self.\_\_account\_number = account\_number

self.\_\_balance = balance

def \_\_display\_balance(self):

print("Balance:", self.\_\_balance)

b = BankAccount(1234567890, 5000)

b.\_BankAccount\_\_display\_balance() # Correct way to call a mangled method

## **Protected Access Modifier**

Class properties and methods with protected access modifier can be accessed within the class and from the class that inherits the protected class. In python, protected members and methods are declared using single underscore(‘\_’) as prefix before their names.

**Example:**

class Person:

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

self.\_name = name

self.\_age = age

def \_display(self):

print("Name:", self.\_name)

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

class Student(Person):

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

super().\_\_init\_\_(name, age)

self.\_roll\_number = roll\_number

def display(self):

self.\_display()

print("Roll Number:", self.\_roll\_number)

# Create an instance of the Student class

s = Student("John", 20, 123)

# Call the display method of the Student class

s.display()

Explanation:

**Person class:**

The Person class has a constructor (\_\_init\_\_) that takes name and age as parameters and initializes the \_name and \_age attributes.

It also has a method \_display that prints the name and age.

* **Student class:**
* The Student class is a subclass of Person. It inherits from Person.
* The \_\_init\_\_ method of Student uses super().\_\_init\_\_(name, age) to call the constructor of the parent class (Person), initializing the name and age attributes.
* It has an additional attribute \_roll\_number specific to the Student class.
* The display method of Student calls the \_display method from the parent class and then prints the roll number.
* **Creating an instance and calling methods:**
* An instance s of the Student class is created with the name "John," age 20, and roll number 123.
* The display method of the Student class is called on the instance s. This, in turn, calls the \_display method from the parent class and prints the roll number.