→ "Access Modifiers in Python: Public, Private & Protected" :→

Access modifiers are used by object oriented programming languages like C++, Java, Python etc. to restrict the access of the class member variables & methods from outside the class. Encapsulation is an OOPs principle which protects the internal data of the class using Access modifiers like "Public, Private & Protected"

Python supports three types of access modifiers which are public, private & protected. These access modifiers provide restrictions on the access of member variables & methods of the class from any object outside the class.

1> Public Access Modifier :→

By default the member variables & 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 & properties public.

Eg :→

```
Class Student:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def display(self):
        print("Name:", self.name)
        print("Age:", self.age)

s = Student("Ravi", 20)
s.display
```

Student class has two member variables, name & age & a method display which prints the member variable values. Both these variables and the methods are public as no specific keyword is assigned to them.

2→ 'Private Access Modifier':→

Class properties & methods with private access modifier can only be accessed within the class where they are defined & cannot be accessed outside the class. In python private properties & methods are declared by adding a prefix with two underscores ('__') before their declaration.

Eg:→
```
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.__display_balance()
```

O/p:→
```
AttributeError: 'BankAccount' Object has no attribute '__display_balance'
```

3→ 'Protected Access Modifier':→

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

(P.T.O)

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)
s = Student("Ravi", 20, 123)
s.display()
```

O/p:→

```
Name: Ravi
Age: 20
Roll Number: 123
```