MODULE-3

PPT-2

Python - public, private and protected

- Classical object-oriented languages, such as C++ and Java, control the access to class resources by public, private and protected keywords. Private members of a class are denied access from the environment outside the class. They can be handled only from within the class.
- Public members (generally methods declared in a class) are accessible from outside the class. The object of the same class is required to invoke a public method. This arrangement of private instance variables and public methods ensures the principle of data encapsulation.
- Protected members of a class are accessible from within the class and are also available to its sub-classes. No other environment is permitted access to it. This enables specific resources of the parent class to be inherited by the child class.
- Python doesn't have any mechanism that effectively restricts access to any instance variable or method. Python prescribes a convention of prefixing the name of the variable/method with single or double underscore to emulate the behaviour of protected and private access specifiers.

Public Access Modifier:

- All members in a Python class are public by default. Any member can be accessed from outside the class environment.

```
class employee:
    def __init__(self, name, sal):
        self.name=name
        self.salary=sal
e1=employee("Kiran",10000)
print(e1.salary)
```

Protected Access Modifier:

- Python's convention to make an instance variable protected is to add a prefix _ (single underscore) to it. This effectively prevents it to be accessed, unless it is from within a sub-class.

```
class employee:
    def __init__(self, name, sal):
        self._name=name  # protected attribute
        self._salary=sal # protected attribute
e1=employee("Swati", 10000)
print(e1._salary)
```

- In fact, this doesn't prevent instance variables from accessing or modifyingthe instance.
- Hence, the responsible programmer would refrain from accessing and modifying instance variables prefixed with _ from outside its class.

Private Access Modifier:

- The members of a class that are declared private are accessible within the class only, private access modifier is the most secure access modifier. Data members of a class are declared private by adding a double underscore '__' symbol before the data member of that class.
- It gives a strong suggestion not to touch it from outside the class. Any attempt to do so will result in an AttributeError:

```
class employee:
    def __init__(self, name, sal):
        self.__name=name  # private attribute
        self.__salary=sal # private attribute
        self.sal=sal
e1=employee("Bill",10000)
print(e1.sal)
print(e1.__salary)
```

```
class person:
        __name = None# private members
        __roll = None
        __branch = None
        def __init__(self, name, roll, branch): # constructor
                self.__name = name
                self.__roll = roll
                self.__branch = branch
        def __displayDetails(self): # private member function
                # accessing private data members
                print("Name: ", self.__name)
                print("Roll: ", self.__roll)
                print("Branch: ", self.__branch)
        def accessPrivateFunction(self): # public member function
                self.__displayDetails() # accesing private member function
p1= person("Priyanka", 1706256, "Information Technology")
p1.accessPrivateFunction()
```