Access Specifiers

Like other programming languages (Which supports OOPs concept) for example C++, Java; Python also uses Access specifiers which are used to restrict access to variables and methods of the class. Python has three access specifiers, which are Public, Protected and Private. Python uses underscore (“_”) symbol to determine the access specifiers for a specific data member or a member function of a class. Access specifiers have an important role to play in securing data from unauthorised access.

Public Access Specifiers

All data members which have declared initially without any prevention are public access specified. This type of data members are open to use within the class method or outside in any non member function. This type of data members have no underscore in the beginning of their name.

Example

```
class Student:
    schoolName = 'XYZ School' # class attribute

    def __init__(self, name, age):
        self.name=name # instance attribute
        self.age=age # instance attribute
```

All data members and member functions are public by default.

Protected Access Specifiers

Those data members which are declared as Protected are only accessible into the present class and derived class. In non member functions, Protected data members are not accessible. We use a single underscore in the beginning of their name to declare them as private.

Example

```
class Student:
    _schoolName = 'XYZ School' # protected class attribute
```

```
    def __init__(self, name, age):
        self._name=name  # protected instance attribute
        self._age=age # protected instance attribute
```

Private Access Specifiers

The members of a class which are declared as Private, are only accessible within the class only. This is the most secure access specifier. We use double underscore in the beginning of their name to declare them as private.

Example

```
class Student:
    __schoolName = 'XYZ School' # private class attribute

    def __init__(self, name, age):
        self.__name=name  # private instance attribute
        self.__salary=age # private instance attribute
    def __display(self):  # private method
         print('This is private method.')
```