Python Access Modifiers

- **Public Member:** Accessible anywhere from outside the class.
- **Private Member:** Accessible only within the class.
- **Protected Member:** Accessible within the class and it's sub-classes.

```
#defining class Student
class Student:
    #constructor is defined
    def __init__(self, name, age, salary):
        self.age = age              # public Attribute
        self._name = name           # protected Attribute
        self.__salary = salary      # private Attribute

    def _funName(self):             # protected method
        pass

    def __funName(self):            # private method
        pass

# object creation
obj = Student('Mohit',53434)
```

- **Public Access Modifier**
- By default, all the variables and member functions of a class are public in a python program.

```
class Employee:
    # constructor
    def __init__(self, name, sal):
        self.name = name;
        self.sal = sal;
obj = Employee('mohit',555)
print(obj.name)
print(obj.sal)

Output:-
Mohit
555
```

- **protected Access Modifier:-** Accessible within the class and it's sub-classes.
- adding a prefix _(single underscore) to a variable name makes it protected.

```
class Employee:
    # constructor
    def __init__(self, name, sal):
        self._name = name;    # protected attribute
        self._sal = sal;      # protected attribute
```

```
obj = Employee('mohit',15)
print(obj._name)

Output:-
Mohit

# Example2:-
class Employee:
    # constructor
    def __init__(self, name, sal):
        self._name = name;   # protected attribute
        self._sal = sal;     # protected attribute

class HR(Employee):
    def task(self):
        print ("We manage Employees")


hrEmp = HR("Captain", 10000);
print(hrEmp._sal)
print(hrEmp.task())

Output:-
10000
We manage Employees
```

- **private Access Modifier** While the addition of prefix __(double underscore) results in a member variable or function becoming private.

```
class Person:
    def __init__(self, name, age, height):
        self.name     = name   # public
        self._age     = age    # protected
        self.__height = height # private

p1 = Person("John", 20, 170)

print(p1.name)        # public: can be accessed
print(p1._age)        # protected: can be accessed but not advised
# print(p1.__height)  # private: will give AttributeError
```