```
class Test(object):

    def __init__(self):
        self.id = 101    # can access directly
        self._id = 101   # should be considered protected
        self.__id = 101  # considered private, name mangled

t = Test()
print(t.id)
print(t._id)
print(t.__id) # AttributeError: 'C' object has no attribute '__a'
```

```
# private v in constructors and we can access private v in instance method
class User:
    def __init__(self):
        self.__id = 101  # Private V
        self.__name = 'NameOne'  # Private V

    # Instance Method
    def display(self):
        print(self.__id)  # Accessing private V, 101
        print(self.__name)  # Accessing private V, NameOne

u = User()
u.display()

Output
101
NameOne
```

```
# private v in constructors and we cannot access them directly
class User:
   def __init__(self):
      self.id = 101  # Public V
      self.__name = 'NameOne'  # Private V

u = User()
print(u.id) # 101
print(u.__name) # AttributeError: 'User' object has no attribute '__name'
```

```
# Mangling is used for private class members to call directly from class
class Student:
   def __init__(self):
      self.__id = 123  # Private V
      self.__name = 'John'  # Private V

s = Student()

# Mangling is used for private class members
# Name mangling (_Student__id, Student__name)
# Syntax: _classname__dataMember
print(s._Student__id)
print(s._Student__name)
```

```
#we wont have direct access to private variables, so we can use setters and getters
class Student:
    # Mutator Methods / setters
    def setDetails(self, sid, sname):  # @ReservedAssignment
        self.__id = sid
        self.__name = sname

    # Accessor Methods / getters
    def getDetails(self):
        return self.__id, self.__name

s = Student()
s.setDetails(101, "SaiKiran")
print(s.getDetails()) # (101, 'SaiKiran')

# we don't have any direct access to private variables
# print(self.__id) # NameError: name 'self' is not defined
```