Python Notes

Python Class or Static Attributes

1. What is Python class attributes?
    A. All variables which are assigned a value in class declaration are class variables.
    B. And variables which are assigned values inside method are instance variables.
    C. For class variables, all objects a single copy maintained at class level.
    D. There is no difference between python class and static variables, both are same.

2. How can we able to create and access class attributes?
    A. Inside Class
        a. Inside class directly
        b. Inside constructor by using class name
        c. Inside instance method by using class name
        d. Inside class method by using cls variable or class name
        e. Inside static method by using class name
    B. Outside Class
        a. From outside of class by using class name

```
class StaticDemo:
    # 1. Inside class directly
    a = 10

    # 2. Inside constructor by using class name
    def __init__(self):
        StaticDemo.b = 20

    # 3. Inside instance method by using class name
    def m1(self):
        StaticDemo.c = 30

    # 4. Inside class method by using cls variable or class name
    @classmethod
    def m2(cls):
        # 4a. using cls variable
        cls.d = 40
        # 4b. Using class name
        StaticDemo.e = 50
```

```
    # 5. Inside static method by using class name
    @staticmethod
    def m3():
        StaticDemo.f = 60


# 6. From outside of class by using class name
StaticDemo.g = 70

object_1 = StaticDemo()
object_1.m1()
# StaticDemo.m2()
object_1.m2()
# StaticDemo.m3()
object_1.m3()
print('Class Level  :', StaticDemo.__dict__)
print('*' * 50)
print('Object Level :', object_1.__dict__)
```

3. How can we able to modify class attributes?
    A. Inside Class
        a. Inside Constructor using Class name
        b. Inside Class method using Class name or cls variable
        c. Inside Static method using Class name
    B. Outside Class
        a. Outside of class by using class name

```
class StaticDemo:
    a = 10

    # 1. Inside Constructor using Class name
    def __init__(self):
        StaticDemo.a = 100
        # Important topic
        self.a = 101

    # 2. Inside Class method using Class name or cls variable
    @classmethod
    def m1(cls):
        # 2a. using class name
        StaticDemo.a = 200
        # 2b. using cls variable
        cls.a = 300
```

```
    # 3. Inside Static method using Class name
    @staticmethod
    def m2():
        StaticDemo.a = 400


object_1 = StaticDemo()
# StaticDemo.m1()
object_1.m1()
# StaticDemo.m2()
object_1.m2()

# 4. Outside of class by using class name
StaticDemo.a = 500
print('Class Level :', StaticDemo.__dict__)
print('Object Level:', object_1.__dict__)
```

4. How can we able to delete class attributes?
    A. Inside Class
        a. Inside Constructor using Class name
        b. Inside Class method using Class name or cls variable
        c. Inside Static method using Class name
    B. Outside Class
        a. Outside of class by using class name

```
class StaticDemo:
    a = 10
    b = 20
    c = 30
    d = 40
    e = 50
    f = 60

    # 1. Inside Constructor using Class name
    def __init__(self):
        del StaticDemo.a

    # 2. Inside Class method using Class name or cls variable
    @classmethod
    def m1(cls):
        # 2a. Using class name
        del StaticDemo.b
```

```
        # 2b. Using cls variable
        del cls.c

    # 3. Inside Static method using Class name
    @staticmethod
    def m2():
        del StaticDemo.d


print('Before object creation :', StaticDemo.__dict__)
object_1 = StaticDemo()
print('After object creation :', StaticDemo.__dict__)
StaticDemo.m1()
print('After calling m1() method :', StaticDemo.__dict__)
StaticDemo.m2()
print('After calling m2() method :', StaticDemo.__dict__)

# 4. Outside of class by using class name
del StaticDemo.e
print('Outside class del :', StaticDemo.__dict__)
```