ASSIGNMENT-7

If we have an object which has both instance as well as class attr, who will get preference?

In [1]:
```
#In Python, when an instance attribute and a class attribute have the same name,the instance attribute takes
#preference over the class attribute.We can see from the output of the below code
class MyClass:
    x = 1  # class attribute

    def __init__(self):
        self.x = 2  # instance attribute

obj = MyClass()
print(obj.x)
```

```
2
```

Therefore, if an object has both an instance attribute and a class attribute with the same name, the instance attribute takes preference over the class attribute. This is because Python looks for the attribute in the instance namespace first, and if it is found there, any attribute with the same name in the class namespace is ignored.
If the instance attribute does not exist, then the class attribute is used. This is because Python will look for the attribute in the class namespace if it is not found in the instance namespace

Create a class and make three object with different parameters and change the values when you retrive data from the class.

In [4]:
```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

person1 = Person("Jaislin", 21)
person2 = Person("Taniya", 30)
person3 = Person("Indus", 21)

print(person1.name, person1.age)
print(person2.name, person2.age)
print(person3.name, person3.age)

# Change the values of the object attributes
person1.name = "Jaislin"
person2.age = 35
person3.name = "IndusRishi"
person3.age = 40

print(person1.name, person1.age)
print(person2.name, person2.age)
print(person3.name, person3.age)
```

```
Jaislin 21
Taniya 30
Indus 21
Jaislin 21
Taniya 35
IndusRishi 40
```

Define Super method and Class method

super() method-
In Python, super() is a built-in function that returns a temporary object of the superclass, allowing you to call methods of the superclass on the subclass. It is often used in method overriding in inheritance.

Class methods-
In Python, a class method is a method that is bound to the class and not the instance of the class. It can be called on the class itself and does not require an instance of the class. Class methods are defined using the classmethod.

In [ ]: