Class or Static Variables in Python

All objects share class or static variables. An instance or non-static variables are different for different objects (every object has a copy). For example, let a Computer Science Student be represented by class CSStudent. The class may have a static variable whose value is “cse” for all objects. And class may also have non-static members like name and roll. In C++ and Java, we can use static keywords to make a variable a class variable. The variables which don’t have a preceding static keyword are instance variables. See this for the Java example and this for the C++ example. The Python approach is simple; it doesn’t require a static keyword.

All variables which are assigned a value in the class declaration are class variables. And variables that are assigned values inside methods are instance variables.

In [1]:
```
# Python program to show that the variables with a value
# assigned in class declaration, are class variables

# Class for Computer Science Student
class CSStudent:
        stream = 'cse'                              # Class Variable
        def __init__(self,name,roll):
                self.name = name                    # Instance Variable
                self.roll = roll                    # Instance Variable

# Objects of CSStudent class
a = CSStudent('Flex', 1)
b = CSStudent('Nerd', 2)

print(a.stream) # prints "cse"
print(b.stream) # prints "cse"
print(a.name) # prints "Geek"
print(b.name) # prints "Nerd"
print(a.roll) # prints "1"
print(b.roll) # prints "2"

# Class variables can be accessed using class
# name also
print(CSStudent.stream) # prints "cse"

# Now if we change the stream for just a it won't be changed for b
a.stream = 'ece'
print(a.stream) # prints 'ece'
print(b.stream) # prints 'cse'

# To change the stream for all instances of the class we can change it
# directly from the class
CSStudent.stream = 'mech'

print(a.stream) # prints 'ece'
print(b.stream) # prints 'mech'
```

```
cse
cse
Flex
Nerd
1
2
```

```
cse
ece
cse
ece
mech
```

class method vs static method in Python

Class Method

The @classmethod decorator, is a builtin function decorator that is an expression that gets evaluated after your function is defined. The result of that evaluation shadows your function definition. A class method receives the class as implicit first argument, just like an instance method receives the instance

In [2]:
```
#class C(object):
#   @classmethod
#    def fun(cls, arg1, arg2, ...):
#       ....
#fun: function that needs to be converted into a class method
#returns: a class method for function.
```

- A class method is a method which is bound to the class and not the object of the class.
- They have the access to the state of the class as it takes a class parameter that points to the class and not the object instance.
- It can modify a class state that would apply across all the instances of the class. For example it can modify a class variable that will be applicable to all the instances.

Static Method

A static method does not receive an implicit first argument. Syntax:

In [3]:
```
#class C(object):
#    @staticmethod
#    def fun(arg1, arg2, ...):
#        ...
#returns: a static method for function fun.
```

- A static method is also a method which is bound to the class and not the object of the class.
- A static method can’t access or modify class state.
- It is present in a class because it makes sense for the method to be present in class.

Class method vs Static Method

- A class method takes cls as first parameter while a static method needs no specific parameters.
- A class method can access or modify class state while a static method can't access or modify it.
- In general, static methods know nothing about class state. They are utility type methods that take some parameters and work upon those parameters. On the other hand class methods must have class as parameter.
- We use @classmethod decorator in python to create a class method and we use @staticmethod decorator to create a static method in python.

When to use what?

- We generally use class method to create factory methods. Factory methods return class object (similar to a constructor) for different use cases.
- We generally use static methods to create utility functions.

How to define a class method and a static method?

To define a class method in python, we use @classmethod decorator and to define a static method we use @staticmethod decorator. Let us look at an example to understand the difference between both of them. Let us say we want to create a class Person. Now, python doesn't support method overloading like C++ or Java so we use class methods to create factory methods. In the below example we use a class method to create a person object from birth year.

As explained above we use static methods to create utility functions. In the below example we use a static method to check if a person is adult or not.

Implementation

In [4]:

```
# Python program to demonstrate
# use of class method and static method.
from datetime import date

class Person:
        def __init__(self, name, age):
                self.name = name
                self.age = age

        # a class method to create a Person object by birth year.
        @classmethod
        def fromBirthYear(cls, name, year):
                return cls(name, date.today().year - year)

        # a static method to check if a Person is adult or not.
        @staticmethod
        def isAdult(age):
                return age > 18

person1 = Person('mayank', 21)
```

```
person2 = Person.fromBirthYear('mayank', 1996)

print (person1.age)
print (person2.age)

# print the result
print (Person.isAdult(22))
```

```
21
25
True
```