**Python variable**

A class consists of variables and methods. Variables define the properties of an instance of a class and methods define the behaviors of an instance of a class.

Variables are of 2 types:

**1) Instance Variables:** Variables that are assigned values inside class methods are instance variables. Instance variables or non-static variables are owned by instances of the class. This means that for each object or instance of a class, the instance variables are different.

The variables that we have dealt with so far were the instance variables.

**2) Class Variables:** All variables which are assigned a value in the class declaration are class variables. Class or Static variables are the variables that belong to the class and not to objects.

Class or Static variables are shared amongst objects of the class

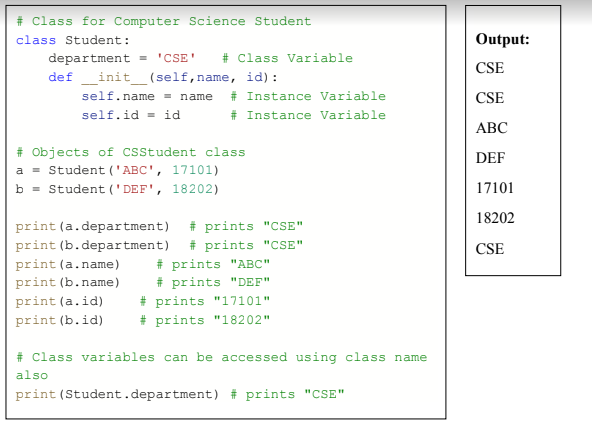
**1.1 Class Variable or Static Variable:**

Class variables are defined within the class construction. Because they are owned by the class itself, class variables are shared by all instances of the class. They therefore will generally have the same value for every instance unless you are using the class variable to initialize a variable.

Class variables can defined outside of all the methods and placed right below the class header and before the constructor method and other methods.

**In python, it doesn’t require a static keyword. All variables which are assigned a value in class declaration are class variables. And variables which are assigned values inside methods are instance variables.**

In the following example, ***department*** *is a class variable because it is defined outside of all the class methods* and *inside the class definition.* ***name*** *and* ***id*** *are instance variable as it is defined inside a method.*



**This is confirmed using the print statement where the department variable is referenced using the class name Student while the instance variable is referenced using the different object references.**

This example shows a scenario where there are different objects each belonging to the same category but are of different types, so each object of the class have the same category which we have made the class variable and the instance variable is different for all the objects hence it is an instance variable. Also class variables can be accessed using class name also.

**Python Method**

Python method is like a Python function. It must be called on an object and put inside a class. A

method has a name, and may take parameters and have a return statement.

There are three types of methods in Python-

**1) Instance Method:** The methods that we will see in this lecture are instance methods. Instance methods can access the attributes and the other methods through the “self” parameter. Thus, instance methods can be only invoked through an instance of the class.

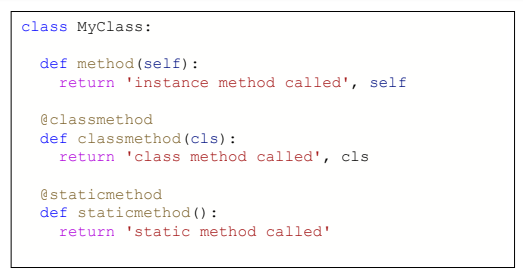
**2) Class Method:** The class method takes a different parameter “cls” instead of “self” that

points to the class instead of the object. Hence, it can’t modify the state of an instance but

can modify the state of a class.

**3) Static method:** The static method does not take any parameter like “cls” or “self”. In fact,

it has nothing to do with the state of the class nor the state of the object.



Previously, we have discussed about instance method. So now let’s discuss about class and static method.

**2.1 Class Method:**

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.

Method with a @classmethod decorator to flag it as a class method. Class methods don’t need a class instance. They can’t access the instance but they have access (self) to the class itself via “cls”.

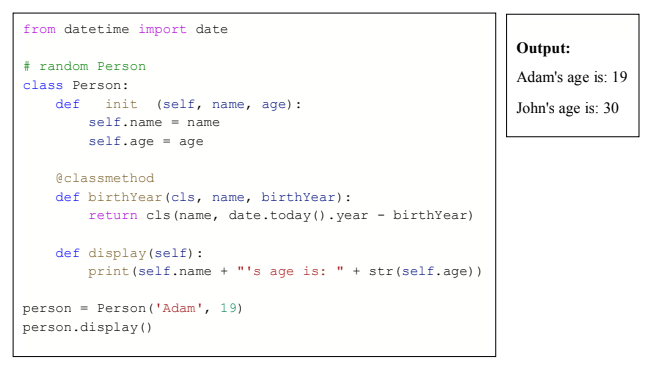
This method only has access to this “cls” argument, it can’t modify object instance state. That would require access to “self”. Class methods can still modify class state that applies across all instances of the class.

Here in the following example there two class instance creator, a constructor and a birthYear method.

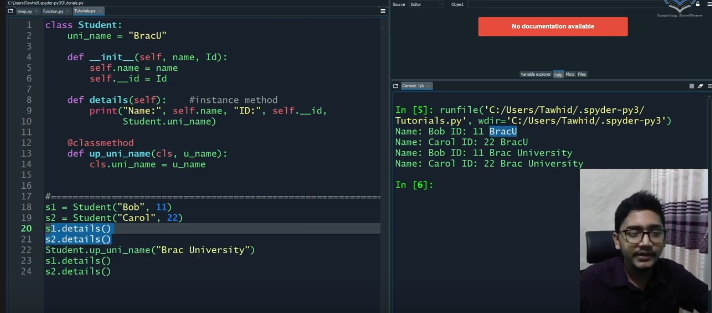
The constructor takes normal parameters name and age. While, birthYear takes class, name and birthYear, calculates the current age by subtracting it with the current year and returns the class instance.

The birthYear method takes Person class (not Person object) as the first parameter cls and returns the constructor by calling cls(name, date.today().year - birthYear), which is equivalent to Person(name, date.today().year - birthYear)

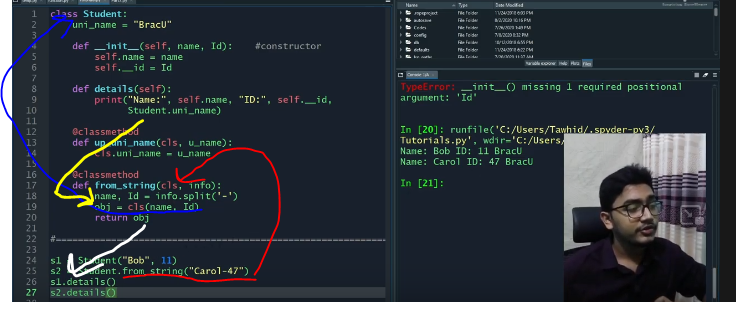
**Before the method, we see @classmethod. This is called a decorator for converting birthYear to a class method as classmethod().**



**Using class method to update class variable:**



**Constructor overloading using class method and procedure steps:**



1. Red
2. Blue
3. Yellow
4. White

**2.2 Static Method:**

Static methods in Python are extremely similar to python class level methods, the difference being that a static method is bound to a class rather than the objects for that class. This means that a static method can be called without an object for that class

MyClass.staticmethod was marked with a @staticmethod decorator to flag it as a static method. This type of method takes neither a self nor a cls parameter (but of course it’s free to accept an arbitrary number of other parameters).

This method can neither modify object state nor class state. Static methods are restricted in what data they can access - and they’re primarily a way to namespace your methods. It can be called without an object for that class, using the class name directly. If we want to do something extra with a class we can use static methods.

