**Course: C339 Data Fundamentals**

**Date: March 6, 2023**

**Title: Object-oriented programming (OOP)**

In Object-oriented programming, when we design a class, we use class and instance variables.

**Class Variables**

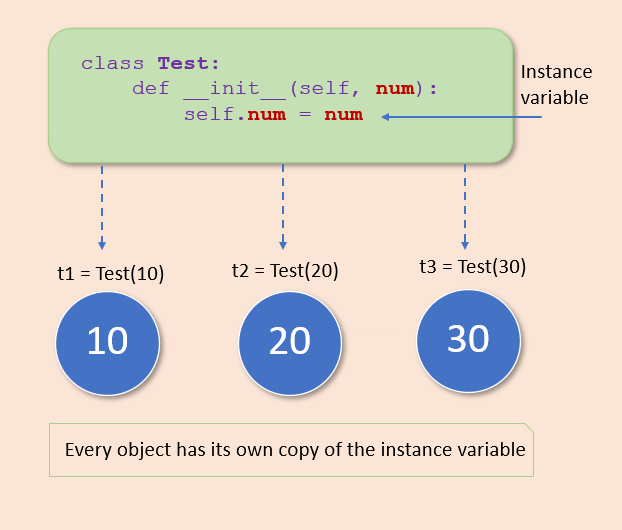
A class variable is a variable that is declared inside of class, but outside of any instance method or \_\_init\_\_() method.

**Instance variables**

If the value of a variable varies from object to object, then such variables are called instance variables. For every object, a separate copy of the instance variable will be created. instance variables are not shared by objects. Every object has its own copy of the instance attribute. This means that for each object of a class, the instance variable value is different.

When we create classes in Python, instance methods are used regularly. We need to create an object to execute the block of code or action defined in the instance method.

Instance variables are used within the instance method. We use the instance method to perform a set of actions on the data/value provided by the instance variable. We can access the instance variable using the object and dot (.) operator. In Python, to work with an instance variable and method, we use the ‘self’ keyword. We use the self keyword as the first parameter to a method. The self refers to the current object.

Figure 1 - Declaring an instance variable.

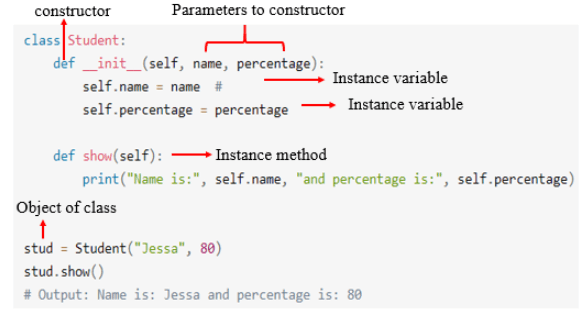


Figure 2: Instance variables and methods

**Creating instance variables**

Instance variables are declared inside a method using the self keyword. We use a constructor to define and initialize the instance variables (Figure 2).

Note:

* When we created an object, we passed the values to the instance variables using a constructor.
* Each object contains different values because we passed different values to a constructor to initialize the object.
* Variables declared outside \_\_init\_\_() belong to the class. They’re shared by all instances.

**Instance Variables Naming Conventions**

1. Instance variable names should be all lower case (e.g. id).
2. Words in an instance variable name should be separated by an underscore (e.g. store\_name).
3. Non-public instance variables should begin with a single underscore (e.g. \_secret\_variable).
4. If an instance name needs to be mangled, two underscores may begin its name.

**Getters and Setters: Manage Attributes in Python**

When you define a class OOP, you’ll likely end up with some instance and class attributes. These attributes are just variables that you can access through the instance, the class, or both.

Attributes hold the internal state of objects. In many cases, you’ll need to access and mutate this state, which involves accessing and mutating the attributes. Typically, you’ll have at least two ways to access and mutate attributes. You can either:

1. Access and mutate the attribute directly, or
2. Use methods to access and mutate the attribute

What Are Getter and Setter Methods?

Getter and setter methods are quite popular in many object-oriented programming languages. So, it’s pretty likely that you’ve heard about them already. As a rough definition, you can say that getters and setters are:

1. Getter: A method that allows you to access an attribute in a given class
2. Setter: A method that allows you to set or mutate the value of an attribute in a class

Instance variable

An instance variable is a variable that is specific to a certain object.