**Advance Assignment - 02**

Q1. What is the relationship between classes and modules?

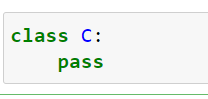
**Answer :**

* Classes are blueprints that allow you to create instances with attributes and bound functionality. Classes support inheritance, metaclasses, and descriptors.
* A module in python is simply a way to organize the code, and it contains either python classes or just functions. If you need those classes or functions in your project, you just import them.
* Classes may generate instances (objects), and have per-instance state (instance variables). Modules may be mixed in to classes and other modules.

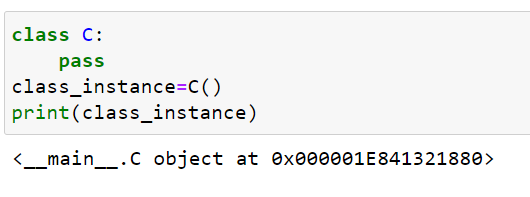
Q2. How do you make instances and classes?

**Answer :**

* A Class is like an object constructor, or a "blueprint" for creating objects. To create a class we use keyword “class”.



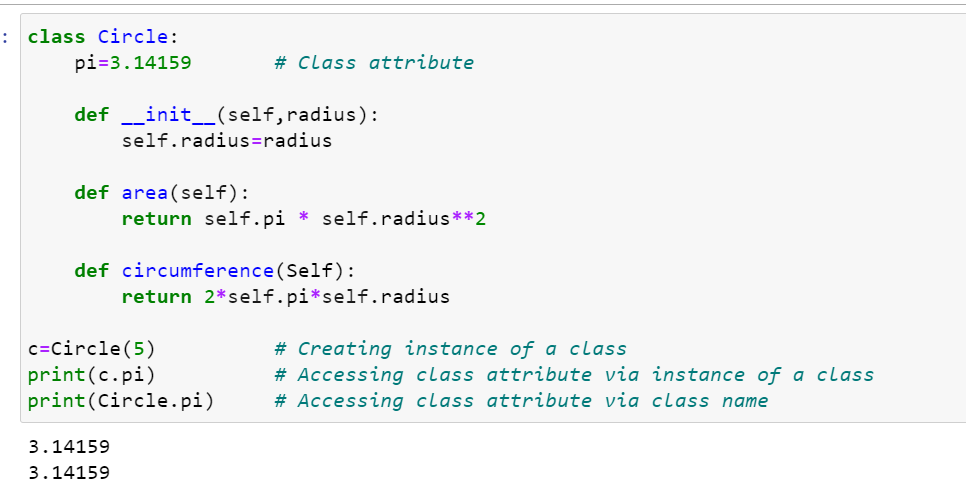
* Use the class name to create an instance of class. Call ClassName() to create a new instance of the class ClassName.



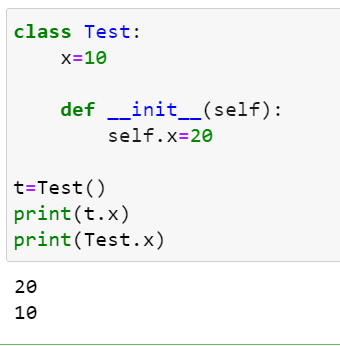
Q3. Where and how should be class attributes created?

**Answer :**

* The class attributes don’t associate with any specific instance of the class. But they’re shared by all instances of the class.
* To define a class attribute, you place it outside of the \_\_init\_\_() method.



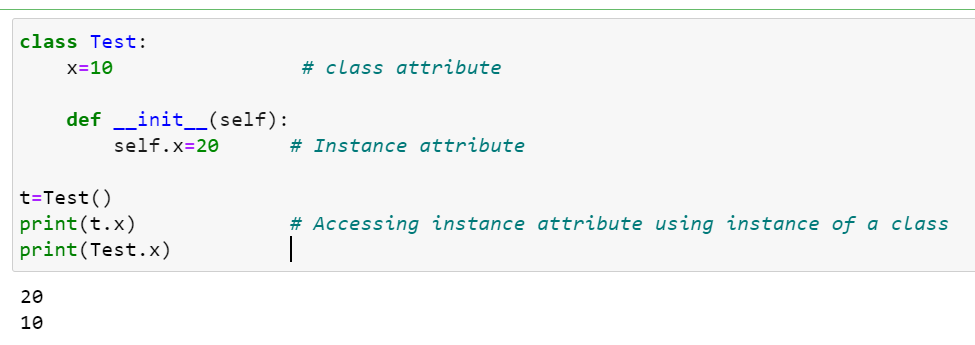
* When you access an attribute via an instance of the class, Python searches for the attribute in the instance attribute list. If the instance attribute list doesn’t have that attribute, Python continues looking up the attribute in the class attribute list.
* If you access an attribute directly using class name, Python directly searches for the attribute in the class attribute list.
* Simple example :



Q4. Where and how are instance attributes created?

**Answer :**

* An instance attribute is a Python variable belonging to one, and only one, object. This variable is only accessible in the scope of this object and it is defined inside the constructor function, \_\_init\_\_(self,..) of the class.



Q5. What does the term “self” in a Python class mean?

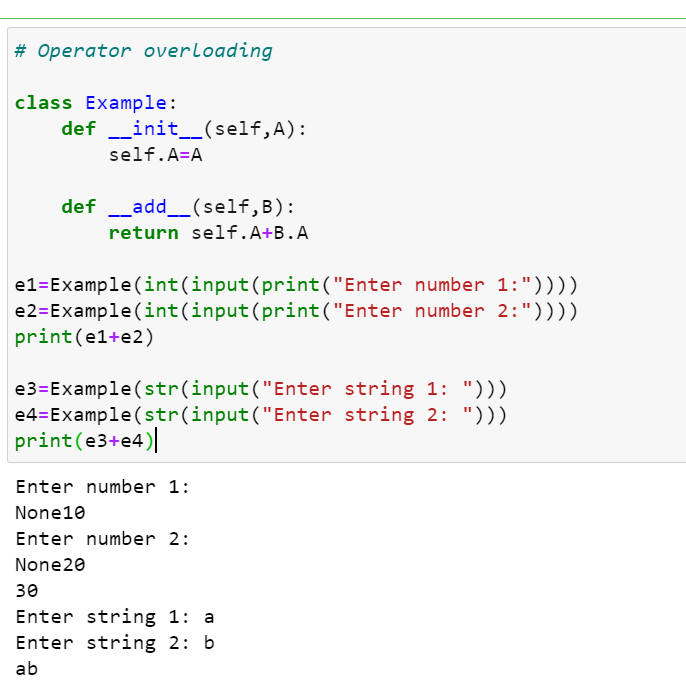
**Answer :**

* self represents the instance of the class. By using the “self” we can access the attributes and methods of the class in python. It binds the attributes with the given arguments.
* The reason you need to use self, is because Python does not use the @ syntax to refer to instance attributes.
* Self is always pointing to Current Object.

Q6. How does a Python class handle operator overloading?

**Answer :**

* Operator overloading work for built-in classes. But the same operator behaves differently with different types. For example, the ‘+’ operator will perform arithmetic addition on two numbers, merge two lists, or concatenate two strings.
* This feature in Python that allows the same operator to have different meaning according to the context is called operator overloading.



Q7. When do you consider allowing operator overloading of your classes?

**Answer :**

* The purpose of operator overloading is to provide a special meaning of an operator for a user-defined data type. With the help of operator overloading, you can redefine the majority of the operators. You can also use operator overloading to perform different operations using one operator.
* It allows you to provide an intuitive interface to users of your class, plus makes it possible for templates to work equally well with classes and built-in/intrinsic types (classes).

Q8. What is the most popular form of operator overloading?

**Answer :**

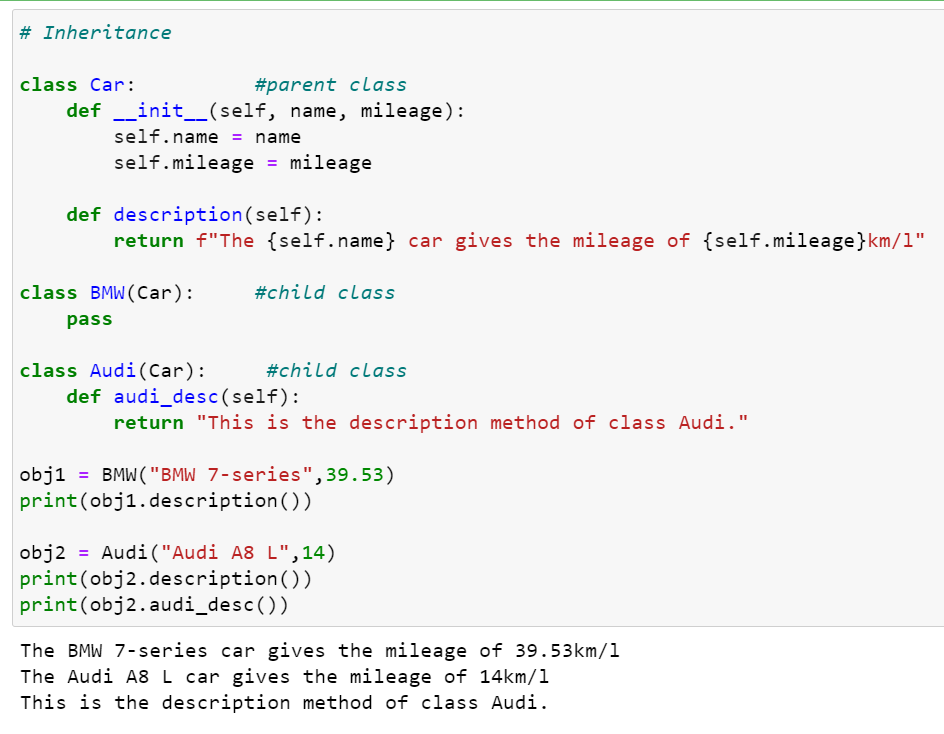
A very popular and convenient example is the Addition (+) operator. Just think how the '+' operator operates on two numbers and the same operator operates on two strings. It performs “Addition” on numbers whereas it performs “Concatenation” on strings.

Q9. What are the two most important concepts to grasp in order to comprehend Python OOP code?

**Answer :**

Inheritance, polymorphism, abstraction and encapsulation are the most important concepts of OOP. These are the pillars of oops.

(i) Inheritance - Inheritance is the procedure in which one class inherits the attributes and methods of another class. The class whose properties and methods are inherited is known as Parent class. And the class that inherits the properties from the parent class is the Child class.



(ii) Polymorphism - Polymorphism means having many forms. In OOP it refers to the functions having the same names but carrying different functionalities.

