Q1. What is the concept of a metaclass?

Ans : A metaclass in Python is a class of a class that defines how a class behaves. A class is itself an instance of a metaclass. A class in Python defines how the instance of the class will behave. In order to understand metaclasses well, one needs to have prior experience working with Python classes.

Q2. What is the best way to declare a class's metaclass?

Ans : There are several ways to do this, but one way is to set \_\_metaclass\_\_ at the module level. This way, all classes of this module will be created using this metaclass, and we just have to tell the metaclass to turn all attributes to uppercase.

Q3. How do class decorators overlap with metaclasses for handling classes?

Ans : Anything you can do with a class decorator, you can of course do with a custom metaclass (just apply the functionality of the "decorator function", i.e., the one that takes a class object and modifies it, in the course of the metaclass's \_\_new\_\_ or \_\_init\_\_ that make the class object!-)

Q4. How do class decorators overlap with metaclasses for handling instances?

Ans : I have something roughly like the following. Basically I need to access the class of an instance method from a decorator used upon the instance method in its definition.