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.

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

**Ans:** class ABC(metaclass=something):

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

**Ans:** For metaclass implementation, in type() we require 3 arguments first ClassName, second base classes name and third attributes of class(so all base classes will written as 2nd parameter).

For decorator, we decorate class to use another class above class name using @ symbol. This is how decorator overlaps metaclasses for handling classes by providing a simpler way to handle classes.

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

**Ans:** To instantiate a metaclass, we have to pass 3 parameters in type(), first is class name, second is base classes and third is attributes and methods used in class. However to instantiate decorator class we don’t need any specific syntax to instantiate class, we can use the same way as we do for normal class. This is how decorator overlaps with metaclasses for handling instances.