Q1. What is the concept of a metaclass?

- 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. 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?

- We can add a __metaclass__ attribute when you write a class (see next section for the Python 3 syntax) class Foo(object) __metaclass__ = something... [...] If you do so, Python will use the metaclass to create the class Foo.

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

- The class decorators sometimes overlap with metaclasses in terms of functionality. Although they are typically used for managing or augmenting instances, class decorators can also augment classes, independent of any created instances.

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

- The class decorators overlap with metaclasses in terms of functionality. Although they are typically used for managing or augmenting instances, class decorators can also augment classes, independent of any created instances.