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.

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

**class** Student:

**pass**

# Print type of Student class

print("Type of Student class is:", type(Student))

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

A decorator is a function that takes a function as its only parameter and returns a function. This is helpful to **“wrap”** functionality with the same code over and over again. For example, the above code can be rewritten as follows.

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

Metaclasses operate at the lower level and allow you to change the structure or behavior of the class, like the class methods, attributes, and inheritance. Decorators, however, are used to modify the functions' behavior. They allowed you to add functionality to the existing functions without changing the code